Designing a Tool to Teach Password Security to Future Developers

Constance Crowe

MInf Project (Part 2) Report

Master of Informatics
School of Informatics
University of Edinburgh

2018

Abstract

In this paper I evaluate a password security teaching environment which I built previously. The system aims to teach first and second year students about password security in an accessible and autonomous manner. By allowing the students to attack an insecure website, they learn how to patch these vulnerabilities from the point of view of a developer.

I evaluate the teaching abilities of this system by conducting two studies: one large-scale study focussing on breadth of response, and a smaller study which provided richer data. I found that the general attitude towards my system was very positive and that students enjoyed interacting with it. Users of the system displayed extremely statistically significant learning gains.

Acknowledgements

I would like to thank all the students who participated in my studies, and the Computer Security teaching assistant, lecturers and tutors.

Thank you to Kami Vaniea, my supervisor, for all her support and guidance.

Finally, I would also like to thank Simon Rovder, Paul Sinclair, Mattias Appelgren and my dad for proofreading this project. Thanks to Will Mathewson for his expert graph advice.

Table of Contents

Chapter 1

Introduction

Computer security is an increasingly important issue. In 2017 alone, over 46% of UK businesses reported that they were the target of at least one attack [11]. There are, of course, a number of defensive measures being taken in order to protect against these attacks, for example the creation of the National Cyber Security Centre in 2016. End-users are also being educated; they are presented with a multitude of guidelines and seminars on "good passwords". However, ultimately, many users do not see the need to secure their passwords, given how unlikely it seems for an attack to target them directly [15].

While a lack of user awareness is behind a large number of attacks such as phishing attacks, the cause for most large-scale data breaches seems to be a lack of precautions taken by developers [11]. For example, TalkTalk was the target of an attack in 2015 which exploited a very basic web vulnerability [6]. This resulted in hundreds of thousands of customers' personal details, credit card numbers and passwords being leaked. More recently, in March 2018, MyFitnessPal suffered a data breach resulting in a leak of all their clients' passwords [19]. Most passwords were hashed using `bcrypt`, a reputable hashing function, however some were only hashed under SHA-1, a hash function for which collisions were found last year thus rendering it insecure [25].

In many instances, developers provide the first line of defence against cyber attacks. Despite this, there is an overwhelming lack of cyber security professionals in industry, and security education in schools and universities. In fact, only 3 of the top 121 computer science universities in the US have a mandatory security module [29]. This is a trend which can be seen worldwide. As a result, in order to learn more about the area, developers must resort to other sources.

1.1 Project Aim

The aim of this project is to design an entry-level system which effectively teaches future developers about password security. While the system itself has already been built (see Chapter 3), I have yet to evaluate the key feature of this system: whether it effectively teaches password security.

With this in mind, I carry out a number of experiments in order to evaluate the hypothesis that this system is an effective method for teaching basic password security concepts. Not only do I hope to show that the system will be effective, but also that it can be used efficiently for teaching without the need for external guidance or aid, thus enabling it to be used autonomously by students.

1.2 Report Structure

The remaining chapters of this report are as follows:

Chapter 2: presents a brief overview of some key password security concepts covered by my system.

Chapter 3: presents the work I completed previously, namely designing and implementing the password security teaching tool which I am evaluating.

Chapter 4: presents a review of common evaluative methods. I establish my own plan for my studies based on these methods and those taken by related works.

Chapter 5: presents the evaluation of my first study. It covers its design, methodology and results.

Chapter 6: presents the evaluation of my second study. It is similar in structure to Chapter 5.

Chapter 7: concludes with a discussion of my results and an overall summary.

Chapter 2

Password Security

Before presenting the system which I built previously, a certain level of familiarity with the concepts I cover is important. The following is a summary of the core principles used in my system.

2.1 Password Storage

Incorrect password storage is the root of many security-related vulnerabilities. Demonstrating correct techniques is one of the key concepts covered by my system. Passwords should always be stored as salted hash values, and never in plaintext.

2.1.1 Cryptographic Hash Functions

A cryptographic hash function takes a variable length input message and produces a fixed length output. Hash functions have two important properties:

- they are "one-way functions", or "trapdoor functions". This means that they are easy to compute but hard to invert. As such, given the hash value of a message, it should be hard to retrieve the original message.
- they are collision-resistant. It should be very difficult to find two different messages which hash to the same value. This would constitute a collision.

The fact that hash values cannot be inverted means that storing passwords as hashes is more secure than storing them as plaintext. If passwords are stored in plaintext, as soon as the database is breached, all the passwords are known. This would not be the case with hashed passwords. However, it is important to note that if two users had the same password, they would have the same hash value. Hash functions are deterministic, so a given message will always produce the same hashed output.

The most common errors when hashing is using a weak hash function which can either be calculated rapidly given today's computational resources, or for which collisions have already been found. This is the case for MD5 or SHA-1.

2.1.2 Salting

A salt is a randomly generated string which is appended to a message before hashing it. By salting passwords, two identical passwords would end up with different hash values, as salts should never be reused.

Salting helps prevent rainbow table attacks (see Section 2.2.3), as it prevents rainbow table pre-computing. A common mistake is reusing salts. This negates most of the benefits of using a salt in the first place as two users with the same password would still have the same hash values. Another common error is using salts which are too short. In this case, an attacker could feasibly just brute-force the salt and password, and thus pre-compute lookup tables for all possible salts.

2.1.3 Shadow Files

On a Linux machine, the system stores usernames and hashed passwords in the shadow file. This file can only be read by the root account. Other systems have similar files for storing hashed passwords.

Each field in a shadow file entry is separated by colons. An example of this could be:

```
 bob:$1$1234$5c9ea3bc158636193369d95cda2acf6c:17073:0:99999:7:::
| 1 |                    2                    |  3  |4|  5  |6|7
```

Each numbered field corresponds to:

1. Username.
2. Hashed Password.
3. Number of days since last password change, in UNIX time (from 1st January 1970).
4. Minimum number of days between password changes. 0 indicates that the password can be changed at any time.
5. Number of days for which the password is valid. Once expired, the password must be changed.
6. Number of days after which the account will be disabled if the password is expired.
7. Extra fields for possible future use.

If we take a closer look at the password field, we can distinguish further sections separated by dollar signs:

```
$1$1234$5c9ea3bc158636193369d95cda2acf6c
|1| 2  |               3               |
```

where each field is:

1. Hash function used (1 - MD5, 2 - Blowfish, 2a - eksBlowfish, 5 - SHA-256, 6 - SHA-512).
2. Salt value.
3. Obtained Hash Value.

2.2 Common Attacks

Text-based passwords are the most common form of authentication on websites [16]. It is therefore no surprise that there are many attacks which target them. Here I cover some of the more common ones.

2.2.1 Brute-Force Attack

Brute-force attacks involve systematically submitting password attempts until the correct password is found. At the most basic level, one might just systematically submit all possible password combinations within the key space.

The main issue here is allowing an attack to continue with this trial-and-error approach until the correct password is found. In order to limit this, a developer could implement rate limiting (throttling the number of requests that can be made) or account lock-outs after too many incorrect attempts.

While ultimately effective in theory, this approach can take an infeasibly long time. The time complexity increases exponentially with the size of the search space.

2.2.2 Dictionary Attack

A dictionary attack is a specific type of brute-force attack which narrows down the search space by submitting each password in a list of common passwords. There exist a variety of different wordlists, but one of the most common is the Rockyou password list. Rockyou was a company which stored their customers' passwords in plaintext (see Section 2.1.1). As a result, when their password database was leaked, the longest ever list of real-world passwords was released [2].

The defences against this attack are the same as against any brute-force attack. The developer could also consider checking if the password a user has chosen on registration is very common, and forcing them to change it if so.

2.2.3 Rainbow Table Attack

A rainbow table associates a list of common passwords to their respective hash values. The key to this attack is finding the hash of the correct password in order to look it up in the table. A surprisingly common error made by developers is leaving debug information in the error messages of a website [1]. It is not unheard of that these messages contain the hash of the correct password. Alternatively, it might also be visible in the HTTP request sent to the server, though I do not cover this possibility in my system.

Once the hash is found, the attack is very rapid, as a key property of rainbow tables is that they are pre-computed and optimised for a fast look-up speed. The attacker does not need to rely on consistently sending requests to the vulnerable website, which makes this an offline attack as opposed to an online attack like a brute-force attack.

To protect against this, the first logical step is to be careful not to divulge information like the hash of the correct password to the attacker. To defeat the rainbow table itself, all passwords should be salted before being stored. This means that the table cannot be pre-computed as the attacker should have no way of knowing the salt in advance.

2.3 Summary

In this chapter we saw that password security is an important topic as passwords remain the main form of authentication for most websites. The most secure way of storing passwords is as a salted hash.

We also covered some of the most common attacks and how a developer might prevent these:

- brute-force attack
- dictionary attack, and
- rainbow table attack.

Chapter 3

My Previous Work

The system I built last year aims to teach users about password security. The goal was to teach users about vulnerabilities by allowing them to exploit some themselves [10].

3.1 System Overview

The system is comprised of six progressively more difficult levels, which teach the user about a variety of password security concepts. To progress through the levels, the user must "hack" each level in a specific way.

The system is primarily a teaching tool aimed at first and second year undergraduates with at least some programming experience. As such, it provides guidance throughout the levels in the form of hints and an introductory text for each level.

Through the latter, I introduce Alice, the developer of the vulnerable website. As the user learns about potential vulnerabilities in the website, Alice learns how to prevent these. By allowing Alice to learn alongside the user, I am able to summarise key learning outcomes from previous levels, as well as give the user a starting point for the current level.

A sample solution and an explanation of the specific exploit used is given at the end of each level. All of these elements can be seen in Figure 3.1, which shows a screenshot of Level 5. Further images of the system can be found in Appendix A.

On successfully finishing the game, participants should be able to:

- know and understand the properties of cryptographic hash functions and salts, and their use and importance in relation to password storage;
- know and understand common mistakes made when hashing and salting passwords;
- understand and perform a brute-force attack, a dictionary attack and a rainbow table attack; and
- know and understand how these attacks could be prevented.

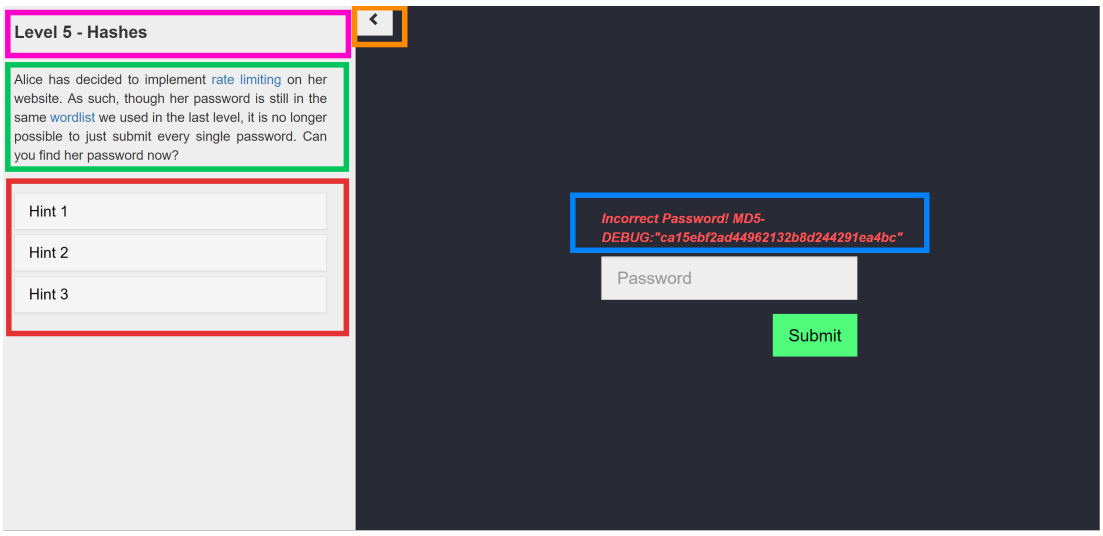

Figure 3.1: Screenshot of system showing the start of Level 5. In pink, the level title. In green, the level's introductory text. In red, the hints for the level. In blue, this level's error message. In orange, the button to collapse the sidebar.

3.2 Level Design

To achieve the learning outcomes, the system has six levels of increasing difficulty. Each level aims to cover a specific attack or concept, and has its own set of learning goals.

3.2.1 Level 1 - Manual Brute-Force Attack

The password for the first level is a single digit. To make this clear to the user, the input form will only accept a single digit. The aim of this level is to highlight the fact that it is possible for an attacker to simply guess a user's password by trying every single possibility until they find the correct solution (a brute-force attack). While this trial-and-error approach can be time consuming in practice, here there is a total of 10 possibilities to try.

3.2.2 Level 2 - Casing

This level does not have a password of its own, but rather displays the password form for Level 3. Before attempting to breach a website, an attacker first needs to identify potential vulnerabilities; this is called casing the website. In this level, I ask the user to answer four questions about the characteristics of the website and the composition of the password in order to progress to the next level. Thus the user enters Level 3 with an idea of how to attack it.

This level highlights the importance of securing all possible flaws, as an attacker *will* search for them. A common error which the user can observe in this level is leaking

information in error messages. The user will also learn about HTTP requests, specifically the differences between the GET and POST methods, along with some basic HTTP status codes.

3.2.3 Level 3 - Automated Brute-Force Attack

The password for Level 1 only had a single digit, and so there was a total of 10 possibilities. In this level, the password is three digits, thus there are 1000 possibilities. The goal is to highlight the fact that even with a restricted search space, the possibilities quickly become too numerous to try by hand and the time taken to perform the attack increases exponentially. The user is therefore invited to perform a brute-force attack by writing a Python script to automate it. While this approach will ultimately be successful, it is not the most time efficient attack.

This level allows the user to gain hands-on experience with scripting an attack, and serves to cement the basic concepts learned in Level 1.

3.2.4 Level 4 - Dictionary Attack

In this level, the password is far more realistic. It is taken from the Rockyou list of common passwords (see Section 2.1.1). This list constitutes one of the largest lists of leaked passwords and is often used as part of a first attempt to crack a user's password. It also highlights, in passing, that storing passwords in plaintext is not a good idea, though I return to this idea in later levels.

This level also reinforces the fact that the burden of choosing a secure password should not solely lie with the user, as we have seen that in many cases they do not feel it is worth the effort [15]. Instead, developers should put in place sensible requirements which deter users from using simple passwords but do not restrict the search space. Even when the password is not absurdly short and simple, it is still possible for an attacker to find it by trial-and-error techniques.

3.2.5 Level 5 - Hashes

To prevent the user from exploiting a brute-force attack again, I implemented rate limiting on this level. The user will only be able to submit five password attempts per minute, rendering a brute-force attack prohibitively slow. The user will therefore be forced to perform an offline attack.

In this level, the password is again taken from the Rockyou wordlist. The user must use what they have learned from Level 2 to notice that the hash of the correct password is leaked in the error message of the password form. From this, they can perform a rainbow table attack.

This level aims to teach the user about hash functions and their characteristics. It also highlights how hashing passwords before storing them would help the issue that Rockyou faced, when their password databases were leaked and were instantly exploitable.

3.2.6 Level 6 - Salted Hashed

As the user has learned from Level 5, hashing passwords helps but does not entirely prevent the possibility of an attacker using a rainbow table attack to crack the password. In this level, I want to highlight the fact that simply hashing a password is not enough, and that passwords should be salted in order to increase the overall entropy. When covering this topic, I also explain both why salting is important and some common errors which are made when using this technique.

The password for this level is provided in the form of an extract from a shadow file. This also allows the user to learn about how passwords are stored on Linux machines.

3.3 Evaluation

The system has gone through a small initial evaluation with regards to usability. In this context, I defined usability to be the the appropriateness of the system to the task, in terms of effectiveness (the users can complete the tasks), efficiency, and user satisfaction.

To evaluate the system, I used the System Usability Scale [8] on 23 participants, interviewed 2 expert users and performed Think Alouds with 5 participants. Though the evaluation was limited in scope, the system was deemed to be usable for the task at hand. There were, of course, a number of issues which the evaluation highlighted, as seen in Table 3.1.

3.4 Summary

In this chapter, I gave a brief overview of the structure and goals of my system. It aims to teach some basic password security concepts to the user through a series of progressively more difficult levels. The levels are as follows:

1. Manual Brute-Force Attack
2. Casing
3. Automated Brute-Force Attack
4. Dictionary Attack
5. Hashes
6. Salted Hashes

While the system has undergone an initial evaluation, the scope was limited to evaluating usability. It was shown to provide a good basis, though there are still a number of elements which could be improved.

Positive Elements	Elements to be improved on
• Hints gave useful and desired information • Explanations are clear and succinct • Attractive design that seems both usable and suited to the task • Sense of achievement and excitement when script yielded the correct password • Bold sections were well-chosen and useful	• Using hints can inspire a sense of defeat • "Salting Hashes" explanation is confusing • Restriction to numeric values on in Level 1 is confusing • Opening files in Python in Level 4 • Using the `hashlib` library in Level 5 • Forgetting "000" to "099" in Level 3 • Links could be more obvious or indicate that they open a modal • Forgetting to change the URL in Level 4

Table 3.1: Summary of positive elements and elements to be improved on that arose as a result of my Think Alouds. Modified from [10].

Chapter 4

Planning the Evaluation

The evaluation I conducted last year (see Section 3.3) showed that, overall, my system was relatively usable, albeit with some features which could be improved. This evaluation was executed on a very restricted scale, appropriate for my question at the time of whether the system was generally usable. However, it provided only qualitative data with regards to the usability of the system. While helpful, it did not allow me to draw any conclusions about the effectiveness of my system as a teaching tool at scale.

To fulfill its goal, my system must teach users about security. We recall that the system was initially designed with a specific set of learning goals in mind (see Section 3.1). As such, I define the "effectiveness" of the system to be whether the users of the system achieve the desired learning outcomes. These are:

- know and understand the properties of cryptographic hash functions and salts, and their use and importance in relation to password storage;
- know and understand common mistakes made when hashing and salting passwords;
- understand and perform a brute-force attack, a dictionary attack and a rainbow table attack; and
- know and understand how these attacks could be prevented.

Thus, in order to say that my system is complete, I must perform a summative evaluation to determine whether my system is an effective tool which teaches users in an accessible and engaging way. To do so, I first establish the best suited survey instrument and measures to the task.

4.1 Possible Measures

It is important to pick appropriate measures to evaluate the system in the correct way. These measures are referred to as dependent variables, as they change based on user behaviour and depend on the independent variables. The latter are varied by the researcher to control the experiment. Measures can be split into two categories: indirect and direct [3]. In both instances, I will examine each type of measure in the context of evaluating learning.

4.1.1 Indirect Measures

Indirect measures do not directly measure what the researcher intends to evaluate. Instead they measure something linked to it. In most cases, indirect measures provide behavioural or attitudinal information, and as such the collected data is generally qualitative. The most common methods to collect this data are through surveys, questionnaires and interviews. A common example of this would be perceived learning.

The term "perceived learning" refers to a student's self-reported knowledge gain, and relies on introspection and self-reflection on the student's part [4]. It is a helpful measure in order to observe a user's affective reaction to a system, as well as their general attitude. Other examples could include how enjoyable a user found interacting with a system. However, there is no correlation between these measures and "actual" learning [23]. As such, in order to evaluate fully the learning gains of my system, I would not be able to use solely indirect measures.

4.1.2 Direct Measures

In contrast, direct measures provide information about the value directly, and thus are often quantitative in nature. In this case, I would need to get a measure of how much (if at all) the users learned by using my system. This is most commonly done through pre- and post-testing [3]. The user is presented with a questionnaire both before and after interacting with the system. This is generally some form of knowledge quiz, though there can also be other components. The post-test portion of this can also be delayed for some amount of time in order to observe the retention of new skills. The "improvement" of each student can be given a value by taking the difference between the post-test and the pre-test for each participant. The researcher may also choose to divide this value by the pre-test score in order to account for high performers [3].

To determine if the obtained results are significant, the researcher performs a paired t-test [22]. A t-test compares the means of two groups, and a paired t-test is used when each observation in one of the groups can be paired with an observation in the other group. This allows the researcher to determine if the difference between the two means is statistically significant or not.

4.2 Approaches in Related Works

Having seen which measures are most commonly used in this type of evaluation, I now review the approaches taken by similar systems.

4.2.1 Cyber Security Teaching Systems

While the intended audience and specific topics covered may differ, there exist a variety of systems which aim to teach cyber security through gamification. However, we realise that the focus of many of these games are chiefly to engage first year undergraduate students and to raise awareness of cyber security. Teaching specific concepts often seems to be a secondary goal. Indeed, Hendrix et al. [14] observed that very few systems of this kind actually carry out any rigorous studies with regards to the learning outcomes of interacting with their systems.

My system is loosely based on the *CPE123* system of Flushman et al. [13]. Their system is composed of a series of levels which aim to engage and teach first year students through game-based learning. However, while my system specifically focusses on password security from the point of view of a developer, Flushman et al.'s system's primary goal is to increase awareness of cyber security and to boost confidence of first year students. As such, their evaluation focusses on self-reflection on the students' part. The students are invited to note down their thoughts in a journal, which was used as one of the main survey instruments in their study. While the results were positive and indicated that the students felt they had learned something, there was no quantitative data with regards to "actual" learning to reinforce this conclusion.

PicoCTF [9] takes a similar approach. The system is a Capture-the-Flag style game. Each participant has to complete various challenges in order to acquire the flag which they can then submit. Again, the goal of this system, as outlined by Chapman et al. [9], is mainly to increase awareness and interest in the domain of cyber security. Their main study was globally design-focussed, however they did ask each student to evaluate how much they believed they learned. Their reported results focus on the students' attitude to the system and the educational experience as a whole, as opposed to specific learning outcomes.

We notice that in both of these examples, there is no direct evaluation of the learning outcomes, but rather they measure the perceived learning of the students. This aligns with what Hendrix et al. [14] observed. However, while this approach suits the goal of their evaluations (to determine the system's effectiveness at engaging students), perceived learning cannot be used alone in order to fully evaluate the effectiveness of a system at teaching new concepts. As a result, I also look at the approaches taken by other, non-cyber security related, teaching systems.

4.2.2 Other Teaching Systems

Crystal Island is a narrative learning environment which teaches microbiology to 8th grade students [18]. It takes the form of a video game, in which the user is invited to solve the mystery of a disease outbreak on an island. To evaluate the learning gains of their system, they performed pre- and post-testing with a questionnaire consisting of eight questions which cover the microbiology curriculum. The difference in score between these two tests serve as a measure of how effectively the participants learned by playing the game. In this instance, McQuiggan et al. found that *Crystal Island* successfully taught students about microbiology, albeit not as well as a traditional classroom environment would. Additionally, the authors used the "Achievement Goals Questionnaire" [12] and the "Self-Efficacy for Self-Regulated Learning Scale" [5] as part of the pre- and post-tests. This allowed them also to gather data on the feelings and perceived learning of the participants.

Similarly, *Betty's Brain* is another learning environment which has been evaluated by pre- and post-testing [17]. *Betty's Brain* covers material typically taught in a high school science curriculum, specifically on the topic of river ecosystems. It does this through the method of learning-by-teaching, whereby the user must construct a lesson for Betty, a virtual student. Leelawong and Biswas designed a questionnaire of six problems which they asked participants to complete and looked at their activity patterns to analyse the students' learning behaviour. They found that on average, the students exhibited significant learning gains.

4.3 My Approach

From this, it seems that in order to be as rigorous as possible, I must use a mix of both direct and indirect measures. Taking example from *Crystal Island* and *Betty's Brain*, I need to perform pre- and post-tests on my system, in addition to evaluating the attitudes and behaviours of the users, as the *CPE123* system and *PicoCTF* did.

4.3.1 Planned Studies

I therefore decided to conduct two evaluations:

- a large scale evaluation focussing on breadth of response. The main focus will be the pre- and post-testing.
- a slightly smaller scale evaluation focussing on depth of response. I will aim to collect richer data here by using pre- and post-testing, alongside surveys to collect attitudinal data to better contextualise the direct measures. This will cover perceived learning, as well as general attitude towards each level, help-seeking behaviours, and time taken on each level.

4.3.2 Changes Made to my System

Finally, before I was able to conduct these evaluations, I addressed the existing known issues within the current system.

One issue is that the suggested solution was always visible in the source code of the webpage. To fix this, I removed the information from the main page, and now dynamically request it on completion of each level so it is not visible prior to this point. I also changed the colour of the links to a slightly brighter blue in the hope that this would make them more visible, as this had been an issue previously. This will need to be observed, to see if this change was effective.

As we saw in Section 3.3, there were some misunderstandings that stemmed from confusing text in the system. This was the case for the Python snippet in Level 5. The `hashlib` library which I recommend to use has unexpected behaviour in certain situations, which led to my users being confused. As knowledge of Python is not the main point which I wish to teach, I added an explanation and simplified the snippet I provided in order to eliminate this issue. The explanation of how to salt hashes in Level 6 was also flagged as confusing, and so I changed the text accordingly. These changes, alongside the originals, can be found in Appendix B.

4.4 Summary

In this chapter, I covered the two different types of measures which could be used to evaluate learning and which methods were typically used. Then, I reviewed the approaches taken by similar systems. I noticed that very few, if any, cyber security teaching systems performed a rigorous testing of the learning outcomes of their system, as they use only indirect measures. However, while this makes sense for the goals they defined, it would not for mine. As such, I turned towards other systems. From here I plan to conduct two studies:

- a large scale evaluation focussing on breadth of response. The main focus will be the pre- and post-testing.
- a slightly smaller scale evaluation focussing on depth of response. I will aim to collect richer data here by using pre- and post-testing, alongside surveys to collect attitudinal data to better contextualise the direct measures. This will cover perceived learning, as well as general attitude towards each level, help-seeking behaviours and time taken on each level.

Finally I outlined the changes which I made to my system prior to beginning the evaluations.

Chapter 5

Evaluation of Study 1

As I stated in Section 4.3.1, I begin my evaluation process with a large scale study which focusses on the pre- and post-test.

5.1 Study Design

The study was organised to take place during the tutorial sessions of the Computer Security course at the University of Edinburgh. As this limited the amount of time each participant would have to interact with the system, the design of the study reflects this.

5.1.1 Questionnaire

As seen in Chapter 4, I want to evaluate my system against a defined set of learning outcomes. However, due to the time contraints, I cannot expect all participants to complete all 6 levels of the system. Thus including too many questions which require concepts learned in the later levels would not generate representative data; the users cannot correctly answer the questions on topics they have not seen. This somewhat compromises the content validity of our questionnaire, as it will not fully cover all the concepts defined as our effectiveness criteria. With this in mind, I created a set of questions to be used on both the pre- and post-test which cover as many of the learning outcomes as possible, given these constraints. I include the numbering of each question in the form "Question <Pre-test Number>/<Post-test Number>" (e.g. Question 1/10) to facilitate cross-referencing. The pre-test questionnaire is shown in Figure 5.1 for convenience. The full questionnaire can be found in Appendix C. The questions cover the following areas.

Consent: This question is only present on the pre-test (Question 1). It confirms that the user agrees to their data being used in this study (see Section 5.2.2).

Levels Completed: This question is only on the post-test, and replaces the consent section of the pre-test. It asks the user how many levels they completed so that I can contextualise their responses.

Web Knowledge: This kind of information is used throughout the system from Level 2 onwards. I use the term "Web Knowledge" for information pertaining to HTTP requests and related areas. While it is not strictly an expected outcome of the system, it is actively used throughout and is essential to know for any kind of developer who needs to interact with a website at any point in their career. I ask about the characteristics of a POST request, though the method they use in the system is a GET request (Question 2/10). The participants should ideally be able to distinguish the two, especially as the differences are explained and they use the GET request in Levels 3 and 4. The options for this question encompass characteristics of both POST and GET requests, so only two answers are correct. The status code of a successful HTTP request is also used heavily in Levels 3 and 4 (Question 3/11).

Attacks and Defences: The learning outcomes include the goal of knowing about common attacks and how a developer might defend from them. I ask one question about each main attack which is covered.

- Question 4/12 covers the definition of a dictionary attack.
- Question 5/13 covers the defence of a brute force attack. The available choices for this question include two which are correct: rate limiting and account lockout. Of the remaining two options, "sanitising user input" is something developers should be doing, but not to defend from this specific attack, while "caching the hashes of users' passwords" makes little sense but does contain the term "hashes" which the user will have just learned about. If they understood it correctly, they should be able to eliminate this option.
- Question 7/15 covers the defence of a rainbow table attack. All the available options are terms related to some degree to cyber security, though only "salting" is correct.

Hash Functions: Question 6/14 asks for one of the two main characteristics of a hash function (one-way function), as covered in Level 5. As I mentioned, Question 5/13 also tangentially tests for understanding of hashing.

Salting: Question 7/15 covers one of the main uses of salting, as presented by my system. Question 8/16 covers shadow files. All possible file names are real files present on a Linux system. The "passwd" file has similar contents to the shadow file, but crucially, does not contain the hashed passwords.

Comments: Question 17 (post-test only) was reserved for optional additional comments.

Answer before you start

Please answer the questions below as best you can before starting the tutorial. If you don't know the answer then please select your best guess, or write "I don't know".

1. **We would like to use your answers in research publications and to improve this tutorial.**
   - ○ You may use my answers below in research publications
   - ○ Do not use my answers below in research publications

2. **Which of the following statements describe a POST request? Tick all that apply.**
   - □ retrieves information from the web server
   - □ sends information to the web server, most likely to be stored
   - □ data is enclosed in the body of the HTTP request
   - □ data is visible in the URL

3. **What is the status code of a successful HTTP request?**
   _____

4. **Which of the following statements best complete this description of a dictionary attack? An attacker performs a dictionary attack by systematically submitting:**
   - ○ all possible password combinations
   - ○ all the words in the English dictionary
   - ○ all the passwords in a pre-established list of passwords
   - ○ what is a dictionary attack?

5. **How would a developer secure their website against a brute force attack? Tick all that apply.**
   - □ caching the hashes of the users' passwords
   - □ account lock out if too many incorrect attempts
   - □ sanitising user input
   - □ rate limiting

6. **Which of the following statements best describes a cryptographic hash function?**
   - ○ hard to compute, hard to invert
   - ○ hard to compute, easy to invert
   - ○ easy to compute, hard to invert
   - ○ easy to compute, easy to invert

7. **How would a developer stop a rainbow table attack?**
   - ○ logging
   - ○ salting
   - ○ fuzzing
   - ○ blacklisting

8. **On a Linux system, where are the users' passwords stored?**
   - ○ the ".bashrc"file
   - ○ they aren't
   - ○ the "passwd"file
   - ○ the "shadow"file

3

Figure 5.1: Pre-test used during Study 1

After drafting the initial questionnaire, I conducted a short, informal pilot study to highlight any ambiguity or lack of clarity in the questions. Five students who were not members of the Computer Security course and the course Teaching Assistant were asked to complete the questionnaire. I rephrased ambiguous questions accordingly.

5.1.2 Environment

In this instance, the environment is an independent variable, as I can control it to a certain degree. The environment can be very important in a study as it can impact the data in unexpected ways.

Given the time constraints, ideally the environment should be familiar to the users, not take too long to understand, and minimise the variations between participants. Thus, each user ran their own instance of my system within a Virtual Machine (VM). Since the students had had previous experience using a VM during the Computer Security course, this ensured that the users would be familiar with the format. This had the benefit of reducing the delay between the beginning of the tutorial and starting the study, and of maximising the amount of time spent interacting with the system.

Using a VM also has the benefit of giving the students the guarantee that we were not recording information about their interaction with the system. This was partially done since the study took place during a tutorial setting, and I did not want the students to feel overly pressured.

I created two VMs in which to run my system with the help of the Computer Security Teaching Assistant to ensure that the systems would be similar to what student are used to. One of these acts as a server to run the system, and is therefore inaccessible to students. The users access the system through the second VM.

5.1.3 Tutorial Instruction Sheet

Finally, I wrote the instruction sheet for the tutorial sessions. It started with the consent form which covered the ethical point in Section 5.2.2, before giving step-by-step instructions on setting up the VM and starting the system. The full form can be found in Appendix C.

5.2 Method

My study took place during the tutorials of the Computer Security course at the University of Edinburgh. There were a total of 11 sessions, each overseen by a Computer Security tutor. I personally attended 7 sessions (as some sessions were run in parallel, it was not possible to attend them all).

5.2.1 Participants

I had a total of 89 participants, of which 87 allowed me to use their results in research. All participants were students in the Computer Security class. The course is typically taken by third year undergraduates, and since the study took place early in the year, the students had not yet learned about concepts relating to password security (such as hash functions). As a result, the majority of the participants were part of my target audience, as they will have had little exposure to password security concepts.

5.2.2 Ethical Considerations

All students who attended the tutorial interacted with the system, however participation in the study was entirely voluntary. To ensure anonymity, the survey was conducted on paper. The students were then free not only to opt out in Question 1 (see Section 5.1.1), but also simply to not hand in their survey. These details were presented to all participants in a consent form (see Appendix C) at the start of the tutorial. This study was approved by the Ethics Board of the School of Informatics at the University of Edinburgh.

5.2.3 Procedure

The participants were given the tutorial instruction sheet and invited to follow it. After reading the consent form, they were asked to complete the pre-test which they were given on a separate sheet of paper. Following this, they started the VMs and interacted with the system. Each session ran for 50 minutes, with the first 10 minutes being used for explanation and set-up time. As such, the average time for each study session was 40 minutes.

During this time, the tutors and I circulated through the classrooms and observed the students' progress. Once there were only 5 minutes left in the tutorial, the participants were asked to complete the post-test found on the reverse of the pre-test. They either left their completed questionnaire sheet by the door, or took it with them.

5.3 Results

5.3.1 Improved Test Scores

Figure 5.2 shows the overall distribution of scores on the pre- and post-test. We notice that both sets of scores follow a normal distribution, and that the post-test scores have shifted compared to the pre-test scores. Indeed, the average score in the pre-test was 3.13, compared to 4.17 in the post-test. This gives an early indication that the use of the system may have had a positive impact on the students' knowledge.

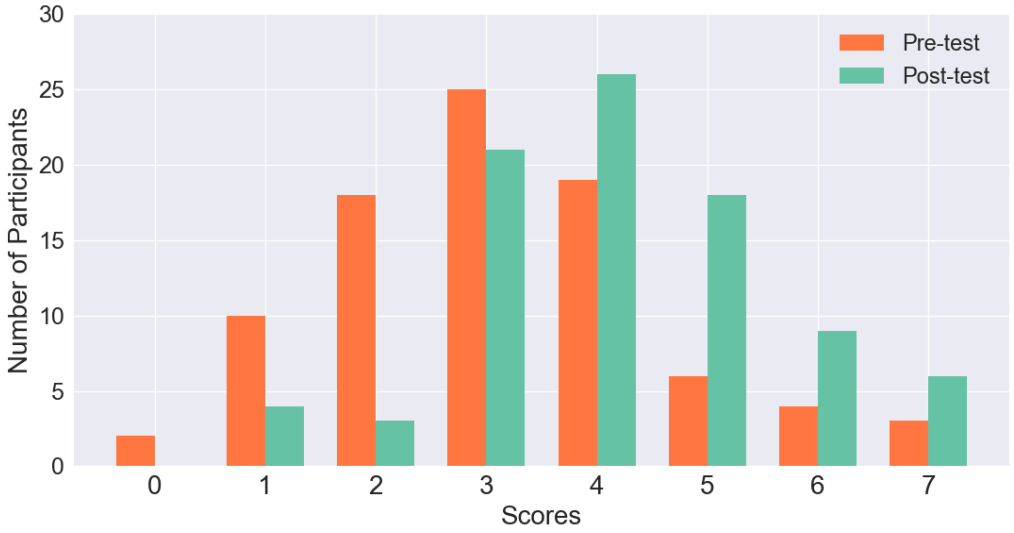

Figure 5.2: Plot of the scores of the study participants in both the pre- and post-test

In order to determine whether this hypothesis is accurate, I performed a paired t-test on the scores. I set the null hypothesis to be that any variations observed in the data are negligible, and are due only to noise. Table 5.1 shows the required statistics for both samples of this test.

There was indeed a statistically significant difference between the pre-test scores (M=3.13, SD=1.53) and the post-test scores (M=4.17, SD=1.42); $t(86) = 6.4656, p < 0.0001$. This allows us to reject the null hypothesis. As such, this result indicates that our system did indeed have a positive impact on the overall results of the students.

Scores	Mean	N	Standard Deviation	Mean Standard Error
Pre-test	3.13	87	1.53	0.16
Post-test	4.17	87	1.42	0.15

Table 5.1: Statistics for paired pre- and post-test samples

It is interesting to note that the average improvement in score (obtained by taking the difference between the post-test and the pre-test score) is higher for the participants who initially scored lower. As we can see in Figure 5.3, the participants with less initial knowledge of the subject benefited much more from interacting with the system, than those who already knew the material.

Figure 5.3: Plot of the average improvement on the post-test in relation to the score on the pre-test

5.3.2 Active Learning

As we can see in Figure 5.4, one question which is almost consistently answered correctly in the post-test is Question 3. During the pre-test the number of correct answers was 36, which increased to 86 (out of 87) on the post-test. We notice that another question which had a large increase in correct answers on the post-test is Question 8.

Both these questions cover material which is directly used by the participants while interacting with the system. In the case of Question 3, the HTTP status code which denotes success is used as a stopping condition in Levels 3 and 4 and is therefore directly required in order to successfully write a script to progress. Similarly, in order to complete Level 6, one must interact with a shadow file entry. This requires the user to understand the different elements of a shadow file, which is the subject of Question 8.

Conversely, Questions 2 and 6 for example, only show a slight increase in correct answers or a slight decrease respectively. However, the knowledge required to answer these questions is not directly exploited in the system, but rather presented during the in-system explanations. This suggests that the students are less likely to take note of material which they do not have to use directly while interacting with the system. It is interesting that even the end-of-level solutions do not receive much attention. It certainly highlights the fact that key information should not only be given as an explanation but also leveraged by the student in order to complete a level, if they are to retain it more consistently.

5.3.3 Variations due to Guessing

As we can see in Figure 5.4, there are no questions which are consistently answered incorrectly after interacting with the system. This suggests that there is no major source of confusion steaming from the system which would mislead the users.

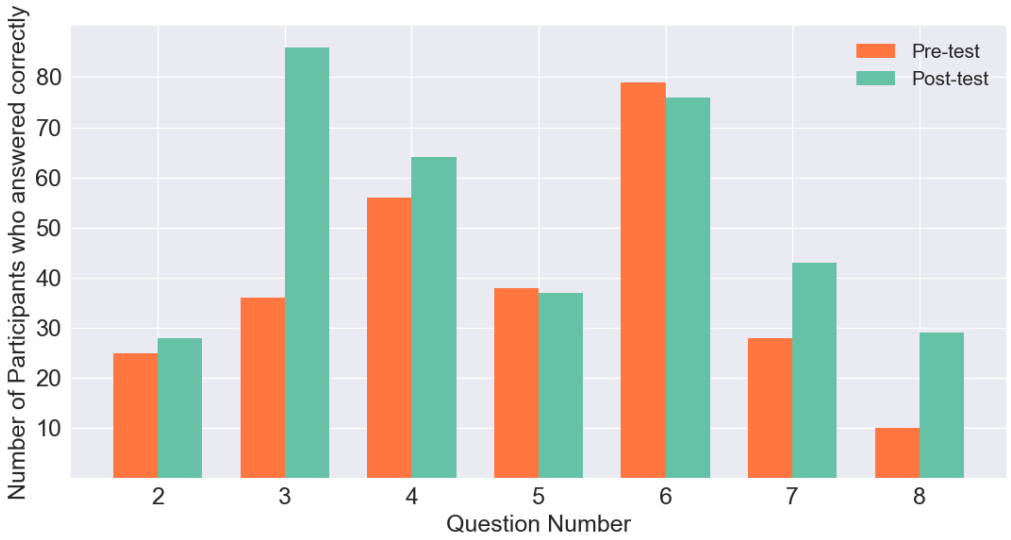

Figure 5.4: Plot of the number of participants who answered each question correctly in the pre- and post-test

However, we do notice that Questions 5 and 6 have fewer correct answers in the post-test than in the pre-test. Upon inspection of the answers given, there does not seem to be a specific question which is answered incorrectly, rather, the post-test has more "I don't know" answers. From this I surmise that the participants who were initially correct, and later marked "I don't know", were in fact guessing the answers in the pre-test. While this suggests that this is not an issue with the teaching abilities of the system, it does show that potentially users had a higher level of self-efficacy before interacting with it.

This also could explain the variance in improvement seen across the participants. While it is true that the average improvement is +1.04, we can see from Figure 5.5 that there is quite a lot of variation. The extreme values can be explained by the differing level of benefit each user has from interacting with the system, as we saw in Section 5.3.1. However, there are a large number of participants whose score did not change from the pre- to post-test. Upon inspection of the answers given, we notice that the answers given on the post-test do not perfectly match those given on the pre-test. Indeed, while some questions which were initially answered correctly are now incorrect, the reverse is also true. There does not seem to be a pattern in the questions which are answered incorrectly, nor does a specific incorrect answer become more popular after interacting with the system. Give this lack of consistency, I assume that this difference is due, once again, to students guessing the answer in the pre-test.

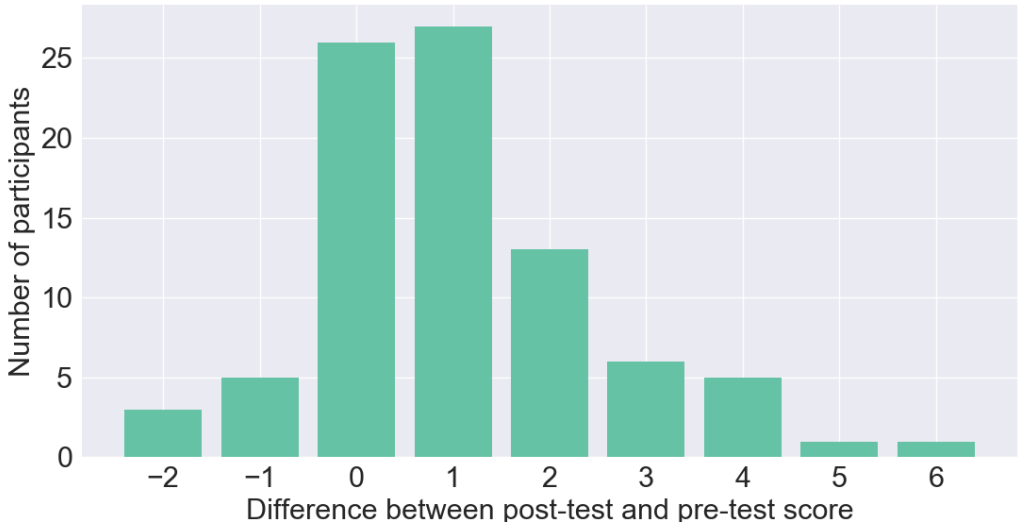

Figure 5.5: Plot of the difference between the scores of the post-test and the scores of the pre-test for each participant

5.3.4 Variations due to Time Constraints

One of the issues with this study is the time constraint. We see for example, in Figure 5.4, that Question 8 did not have very many correct answers overall. However, Question 8 relates to material taught only during Level 6. Therefore, if the students did not get that far, they would not have been able to answer this question (other than by guessing or already knowing).

In fact, if we look at Question 9 on the post-test, we can see that 14 students completed all the levels. If we look at the answers given for Question 8 by these participants, we see that one person was correct in the pre-test, compared to 12 correct on the post-test. However, were we to look at the 18 participants who only reached Level 3, there is one correct answer in the pre-test, compared to 2 in the post-test. Clearly this will skew my results, as for those who reached Level 6, the increase is quite significant, which suggests that Level 6 is in fact effective in teaching about shadow files.

5.3.5 Observations and Overall Impressions

From the comments at the end of the post-test (Question 17), the overall impressions of the participants were positive. There were many comments which expressed how engaging they felt the system was, for example “very fun, a little frustrating at first but when you get it it’s like ‘oooh’ [*sic*]” and “very accessible way to learn. Felt more like a game”. The only negative feedback was one student expressing their dislike of Python.

At the end of each tutorial, I asked the Computer Security tutors for their feedback and observations. The main points that we observed are as follows:

- The participants displayed varying levels of Python knowledge. This added unintended difficulty for some users, though as a side effect, the system also taught them a lot about Python. It would be interesting to include a question to gauge Python knowledge in the second study.
- Many participants seemed reluctant to use the hints. This resulted in many users searching for Python help online, when it was in fact in the provided hints. The challenge should not stem from a lack of Python knowledge. As such, I suggest renaming the hints according to their contents to give the user a better idea of whether they want to use them.
- A very frequent mistake was forgetting to change the URL when moving between Levels 3 and 4 which required similar python code. Most left it as the URL for Level 3, and as a result got stuck. I will add a hint covering this issue, given how common it was.
- Many seemed to not fully understand the concept of an offline attack in Level 5. While they correctly figured out how to obtain the correct password, many still tried to submit it via GET request in their script. It could be interesting to collect their solution scripts to observe how common this is in the next study.

One tutor, who allowed me to quote them, stated that it seemed to be "a solid introduction to modern password security concepts". They were generally positive, particularly regarding the autonomous nature of the system. They also remarked that the main source of difficulty seemed to be the Python concepts, rather than the security concepts. This aligns with what I noted above.

5.4 Summary

I conducted a first study on the effectiveness of my system using pre- and post-testing. The evaluation was carried out on 87 participants, over 11 tutorials of the Computer Security course.

An initial study of the test results, alongside a paired t-test, led to the conclusion that my system does indeed teach users about password security. However, we also noted that the effectiveness of the system varies based on the initial knowledge of the participant.

We also noted that there is some noise in the data, most likely due to guesswork on the part of the students and the time constraints. Most users did not have time to complete all the levels and were therefore unable to answer all the questions.

Overall, the participants seemed generally positive about the system, though I have some observations to take into account when planning the second study.

Chapter 6

Evaluation of Study 2

In this chapter I explain how I conducted my second study. Using the first study as a basis, I refine my approach and redesign the survey instruments accordingly.

6.1 Changes made to the System

6.1.1 Updates based on Observations

As we saw in Section 5.3.5, the users tended to forget to change the URL when moving from Level 3 to Level 4. I have added a hint which reminds the users to do this.

Many users seemed reluctant to use hints in the first place. In fact, they often ended up searching online for Python code which was provided in the hints. As such, I renamed the hints to reflect their content. For example, if the hint contained a Python snippet about file manipulation, it was renamed to "Python Snippet – Opening Files". The hints which did not contain code were simply named "Hints" followed by a number, unless the content was particularly useful and often missed.

6.1.2 Hosting the System Online

In order to increase accessibility to the system, I hosted it online on a Microsoft Azure server, a cloud computing server. An important consideration when hosting my system was the fact that I explicitly asked users to perform brute-force attacks. This could result in an ill-configured server becoming flooded with requests and thus becoming unresponsive, effectively causing a Denial-of-Service attack. The advantage of using an Azure server is that it can be scaled automatically, and is therefore much harder to overload.

6.1.3 Adding Analytical Features

Since the system is now online, I am able to collect additional data on the users' interactions with the system. Each user is now allocated an universally unique identifier which allows me to anonymously record their survey responses.

To get a sense of the duration of the levels, I save how long each level took to complete for each participant. The time taken to complete the surveys was not counted. At this stage, I also implement the framework for the inter-level surveys (see Section 6.4).

All this data is save in a SQLite database, chosen for its reliablility, ease of use and self-maintenance. There will not be a high write concurrency, nor do I require a relational database (which are weaknesses of SQLite databases), and as such, a SQLite database is sufficient for my requirements [24].

6.2 Study Design

6.3 Questionnaire

After the first study, I realised that it is much harder to locate the cause of errors in understanding when the participant can select multiple answers in a multiple choice question. There is no way to pinpoint what exactly is happening. As such, I chose to allow user to select only one answer per question.

Since I am no longer constrained by a time limit, I can ensure that my questions fully cover all the material I want to evaluate. This questionnaire has a better content validity than my previous one.

Each question has an option of "I don't know". The questions cover the following topics.

Demographics: Question 1 of the pre-test simply asks if the participant has prior knowledge in the field. In the post-test, this is replaced by a question which asks if the user has seen this system before. This became necessary as there is a chance that the user may have taken part in the previous study. Question 2 presents the participant with a short snippet of Python code and asks them to give the output. This question was created in reaction to the observations made in the first study (see Section 5.3.5), in which there were varying levels of Python knowledge. This should allow us to estimate their familiarity with the language, and should be more accurate than simply asking the user to evaluate their own knowledge of Python. While this question is related to the demographics of the participants, it is still counted as part of the pre- and post-test scores as there is a correct answer.

Attacks and Defences: I changed the format for evaluating the participants' knowledge of attacks and defences. Since I expect the participants to learn the relationship between the two, a matching problem is a more elegant solution. Questions 3-4 and 7-8 are descriptions of attacks; the user is asked to link these descriptions to either the name of the attack or to a defensive measure as indicated.

Hash Functions: Questions 9 and 10 cover both the one-way function and collision resistant properties of a cryptographic hash function. The choices given for Question 10 include two options which involve salting. While salting would prevent a collision, it doesn't affect the actual resistance to collisions of the hash function, and is certainly not the definition of collision resistant.

Salting: In addition to the mention of salting in Question 10, Question 5 from the matching problem covers shadow files. I added "password" as an option in the matching problem as a misleading answer for this question. Errors made when salting are covered in Question 11. Two of the available options are not errors at all, and are in fact true (the options that state salts are random and stored in the shadow file). "Salts are based on MD5", while an odd choice, is not an error. However, it includes "MD5" which is a hash function the participants must use when solving Level 5, which may lead to some confusion if the participants do not think carefully about it.

Web Knowledge: Questions 12 and 13 are similar to the web knowledge questions in the questionnaire of the first study (see Section 5.1.1). Question 13 has been reduced to a simple true or false answer, rather than the multiple choice options for the reasons I stated earlier.

Comments: The post-test ends with an optional comments section.

6.4 Inter-Level Survey

The aim of this survey is to gather behavioural and attitudinal data from the user. There is one survey for each level, and it is displayed within the system in between each level. An example of this survey can be found in Appendix D.2. The surveys for each level are identical.

Perceived Learning: The user is asked to evaluate their perceived learning on a five-point Likert scale. This gives an indication of their feelings towards the level, as perceived learning is linked to engagement [27].

Pre-questionnaire

Please answer the following questions to the best of your abilites.

1. **Have you taken a computer security class in the past, for example Computer Security (INFR10067)?**

   - Yes
   - No

2. **What is the output of the following Python code?**

```
for i in range(21):
    if i % 3 == 0 and i % 5 == 0:
        print "FizzBuzz"
    elif i % 3 == 0:
        print "Fizz"
    elif i % 5 == 0:
        print "Buzz"
    else:
        print i
```

Output of program:

Match each description to one of the terms.
Hint: Not all terms will be used. Each term should be used at most once.

3. The attacker has a list of common passwords. They submit each password until they find the correct one. This is a _____ attack.

4. The attacker has found the SHA-512 hash of the correct password. They also have access to a list of the SHA-512 hashes of common passwords. The attacker compares the correct hash to all the other hashes until they find a match. One way of preventing this attack would be to use _____.

5. On a Linux machine, the salted hashes of the passwords on the system are stored in the _____ file.

6. Before mounting an attack on a website, the attacker must first identify potential vulnerabilities. The attacker will likely want to assertain how the website works. This is called _____ a website.

7. The attacker performs a brute-force attack against the website by repeatedly submitting passwords, until they find the correct one. One possible way of prevent this would be to implement _____.

I don't know ▾

I don't know
shadow
salting
dictionary
rate limiting
password
rainbow table
sanitising user input
casing

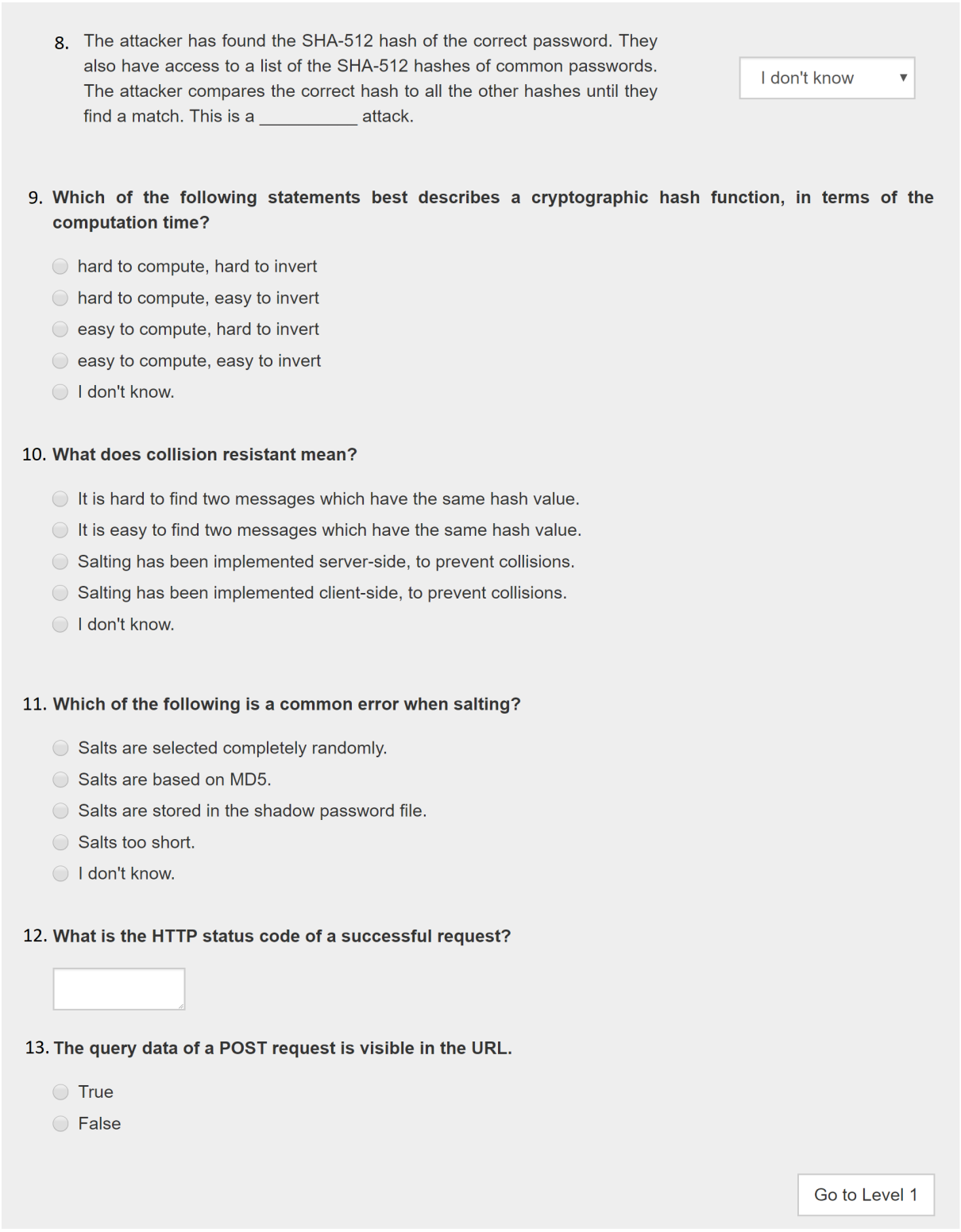

Figure 6.1: Screenshot of the pre-test of Study 2. Question numbers have been manually added to facilitate cross-referencing. They are not present in the system.

Feelings towards the level: The next three questions address the user's feelings towards the level. It asks the user to rate how engaging/frustrating, easy/hard and interesting/boring the level is. There is one five-point Likert scale for each question, inspired by common Likert type response anchors [28].

Help-seeking: I observed in the previous study that some users were reluctant to use the hints. This aligns with my observation from my previous work [10]. As such, I ask the users whether they used the hints and why. While I could trivially implement a way of tracking whether the user clicked on the hints, I would not be able to gather any information about their motivations for doing so.

Solution: To get a better sense of the ways in which the students solved the levels, or how they potentially got stuck, I ask them to enter their code. This question is optional as I do not want the participants to feel too pressured by this code review.

Comments: The survey ends with an optional comments box.

6.5 Method

The study was estimated to take just over an hour, based on the Think Aloud times from our study last year, and the progress made in the evaluation in the first semester. It took place in the Appleton Tower labs.

6.5.1 Participants

I recruited 23 participants, of which 22 were 2nd year undergraduate students, and one was a fifth year student. Of these, two had taken a Computer Security course in the past. None had ever interacted with my system before.

6.5.2 Ethical Considerations

This study obtained ethical clearance from the Ethics Board of the School of Informatics at the University of Edinburgh. Before interacting with the system, the user is required to read through an online consent form. This outlines the key factors of the survey, including the anonymity of the user and the storage of the collected data (including time taken). By ticking the checkbox, the user agreed to have read all the above guidelines and information, consented to taking part the study and having their data be used in research and publication.

The participants were compensated with £10 on completion of the study. The full form can be found in Appendix D.1.

6.5.3 Procedure

The participants were invited to sit down at a computer of their choice, or to use their personal computer. I then brought up the system in a web browser. I informed the participants that while I would not be able to answer any questions during the study, I would be happy to do so after. This is necessary as the system should be self-contained and should be usable without the guidance of an instructor. I asked them to read the consent form carefully and to begin when they felt ready.

Once the students agreed to participate in the study, the system displayed the pre-test in the browser. On completion, Level 1 was displayed. The inter-level surveys are displayed in the browser between each level. Following Level 6, there is a final level survey, followed by the post-test. They then reach a completion screen, clearly indicating the end of the study.

6.6 Results

6.6.1 Performance in Post-test

As we can see in Figure 6.2, it would seem that regardless of the pre-test score, all participants scored at least 7 marks in the post-test (with one exception). In fact, it appears that many participants doubled their pre-test score. The average improvement in score was +4.04.

Again, I perform a paired t-test to determine the statistical significance of the difference between the means of the pre- and post-test scores (see Table 6.1). Let the null hypothesis be that any variations in data are due to noise (e.g. guesswork).

Scores	Mean	N	Standard Deviation	Mean Standard Error
Pre-test	6.09	23	2.73	0.57
Post-test	10.13	23	2.34	0.49

Table 6.1: Statistics for paired pre- and post-test samples

I found that there was indeed a statistically significant difference between the pre-test scores (M=6.09, SD=2.73) and the post-test scores (M=10.13, SD=2.34); $t(22) = 6.6521, p < 0.0001$. This allows us to reject the null hypothesis. As such, I can again conclude that our system had a positive impact on the learning of the students.

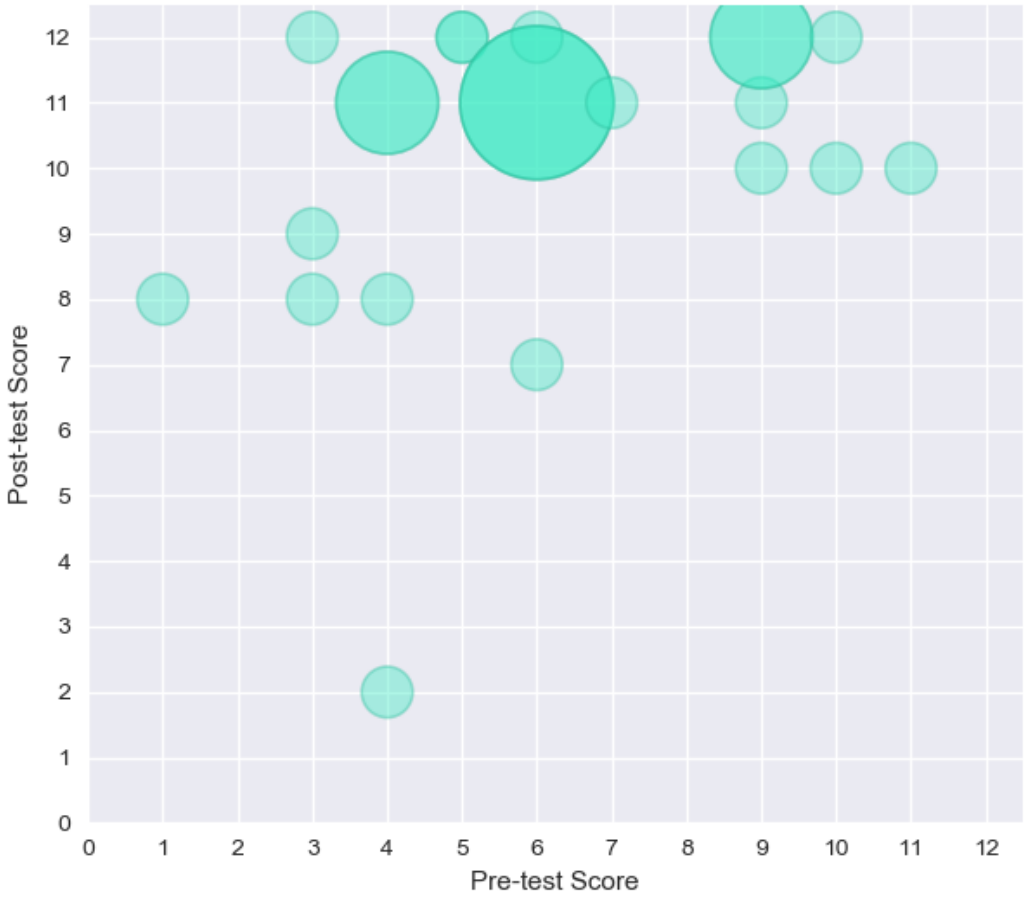

Figure 6.2: Graph of the pre-test and post-test scores for Study 2

6.6.2 Time taken on each Level

The intention was to have levels which are progressively more difficult. If we observe the plot in Figure 6.3, we can see that overall the average time taken increases until Level 4, at which point it decreases slightly for Levels 5 and 6. This trend is consistent with our findings from our study last year [10]. Since Levels 5 and 6 are offline attacks, it makes sense that they would take less time than Levels 3 and 4, despite the difficulty progression.

We can, however, observe a few outliers. In some cases, these can be explained by examining the answers they have given in the inter-level survey for that level. For example, the participant who took 20 minutes to complete Level 2 explains that they actually solved Level 3 (as the password field in Level 2 is that of Level 3). As a result they took only a minute to complete Level 3. This does suggest however, that this participant did not read the introductory text very carefully.

We also notice that quite a few participants completed Level 5 very quickly. On examining their submitted solutions, we can see that they chose to look the MD5 hash up online and thus found the plaintext very quickly. This is a perfectly legitimate approach to the problem, and is mentioned as a possible solution at the end of the level.

The average time taken to complete the entire system was exactly 1 hour.

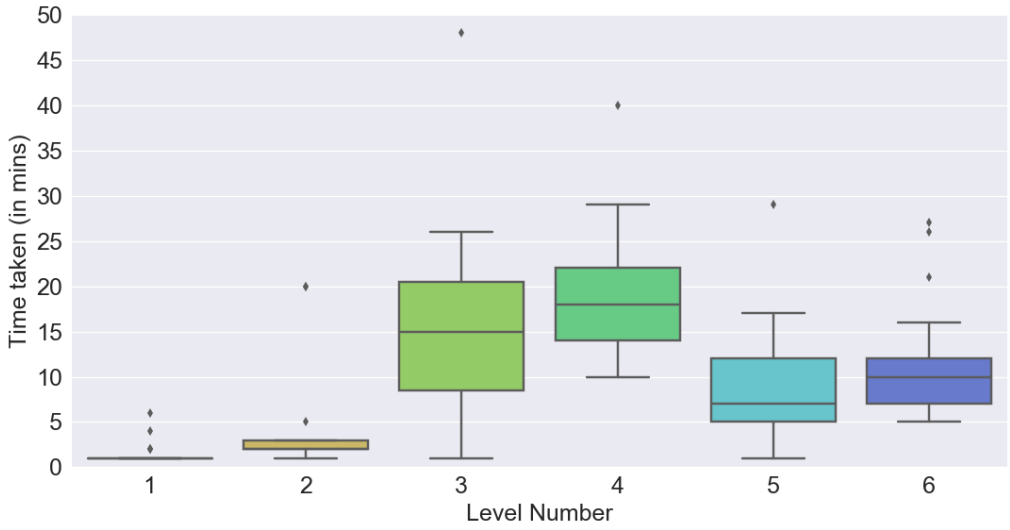

Figure 6.3: Box plot of the time taken in minutes to complete each level

6.6.3 Performance in Individual Levels

As we can see in Figure 6.4, there were always more correct answers given in the post-test than in the pre-test. If we look at the questions which had the largest increase in correct answers, we find that Question 5 and 11 have 18 and 14 additional correct answers respectively. Question 5 is about the shadow file, while Question 11 is about salting passwords. It is interesting that the answer to both these questions are in Level 6, the last level. In the previous study, most participants did not have time to reach this level. This is helpful as it confirms that Level 6 indeed does cover shadow files and salting well.

Once again, we notice that both of these concepts are things which the users have to directly manipulate at some point. This reinforces our hypothesis about active learning (see Section 5.3.2).

Finally we also note that the number of correct answers to Question 2, the Python knowledge test has also increased. This suggests that some user may have become more familiar with Python through using the system. This claim is supported by some of the user comments, for example: “learnt a lot about Python and security”.

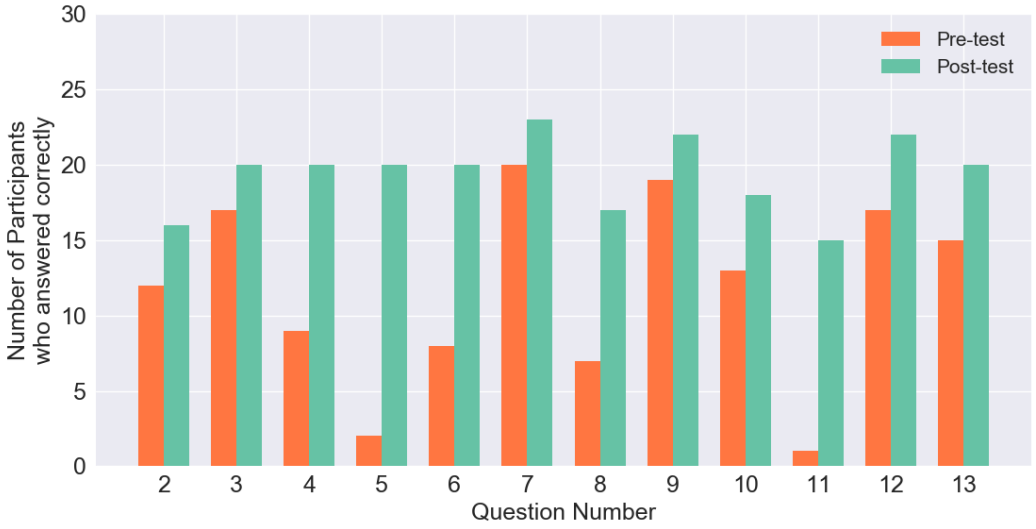

Figure 6.4: Graph of the number of participants who answered each question correctly in the pre- and post-test

6.6.4 User Perception of Levels

6.6.4.1 Perceived Learning

Figure 6.5 shows the answers given to the statement "I learned something from this level", which evaluates the perceived learning of the users. The results in Level 1 seem to be somewhat mixed, with "Definitely Agree" being the least common option. This is to be expected, as it is an introductory level, and the system does not go into the details of a brute-force attack at this point. However, we observe that the "Definitely Agree" option becomes progressively more prevalent as the users move through the levels. This seems to indicate that the levels became more "useful" in the eyes of the users. This makes sense as the levels cover progressively more elaborate and complex topics. The variable level of the participants' prior knowledge may also contribute to this, as those who know more initially are less likely to feel as though the level was useful.

6.6.4.2 Perceived Difficulty

Figure 6.6 shows the answers given by the participants on the perceived difficulty of each level. As anticipated, none of the participants found Level 1 to be particularly challenging. We can see that a few users found most levels to be relatively easy. Upon inspection, these correspond to the users who scored higher on the pre-test. This would indicate they had more prior knowledge, and so naturally would find the levels far simpler.

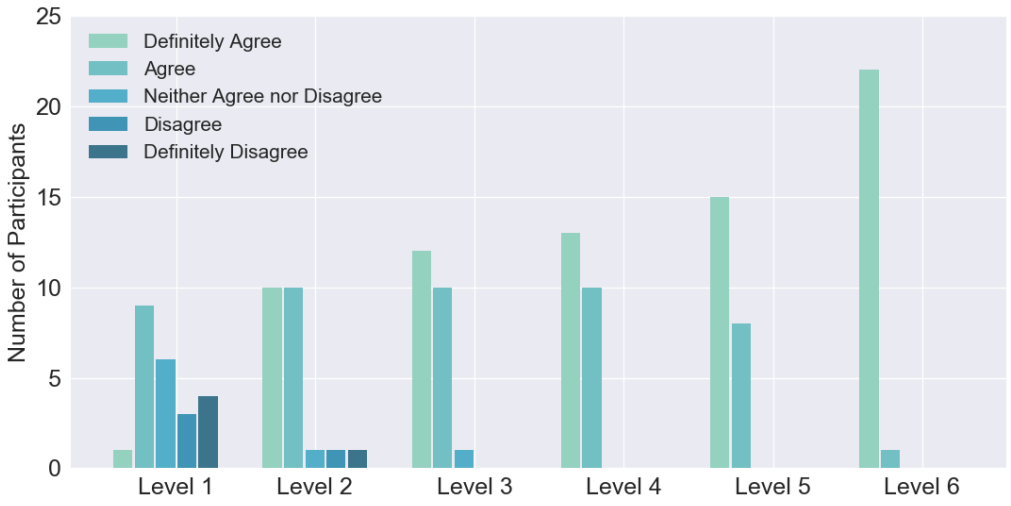

Figure 6.5: Graph showing the perceived learning of the participants in each level

Overall, the majority of respondents considered the levels to be of "neutral" difficulty: neither particularly hard nor easy. Only one user felt that Level 6 was too hard. This is a positive result, as it means that on the whole, beginners did not have a problem solving the levels and did not feel that they were far too hard. The converse is also true; most levels were not considered too easy with the exception of Level 1. This seems to indicate that the overall difficulty of the system is appropriate for the target audience.

We also notice that the later levels tend to be viewed as slightly harder. This aligns with what we observed with regards to the time taken to complete each challenge in Section 6.6.2. It also shows that I was likely successful in my aim of creating progressively more difficult levels.

6.6.5 General Observations

We can see in Figure 6.7 that the only level which was marked as "Frustrating" is Level 4. It was also the most "boring" (see Figure 6.8). On inspection of the comments left in the survey, we observe that this was likely due to the amount of time it took to find the correct password within the wordlist. We notice comments such as: "the solution could have occurred sooner" and "it would be better if you picked a password further up the list as the wait was tedious". Interestingly though, neither of the respondents who left these comments actually selected "Frustrating" or "Boring", but rather they were quite positive. While they may have become bored during the wait, they did not seem to find the level boring overall.

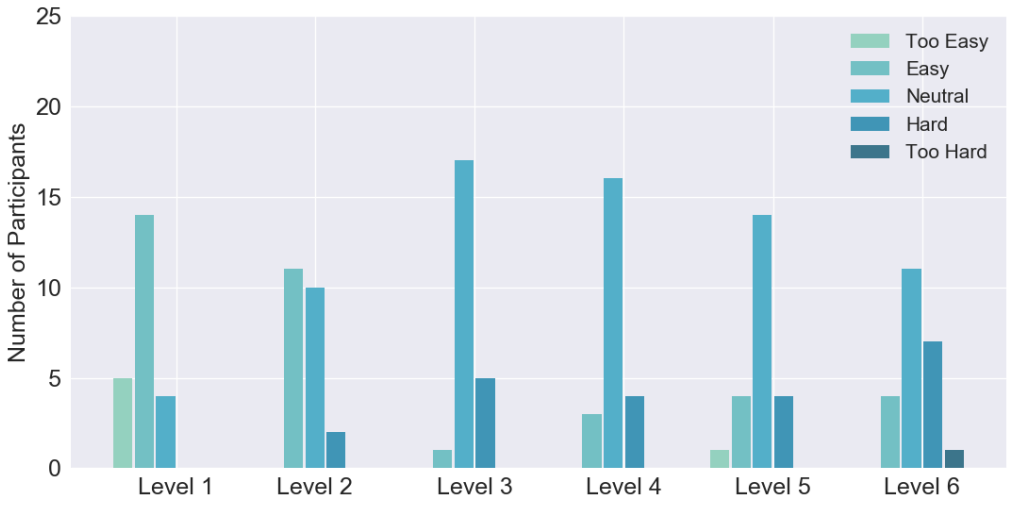

Figure 6.6: Graph displaying the responses to the perceived difficulty of each level according to the users

This conclusion is supported by the comment: "[I] liked how level 4 built upon level 3. Further insight into how wordlists can basically destroy a password's complexity and a repeated realisation of how slow python is". This suggests that the level was indeed interesting, but that the wait while the Python script found the correct answer was tedious.

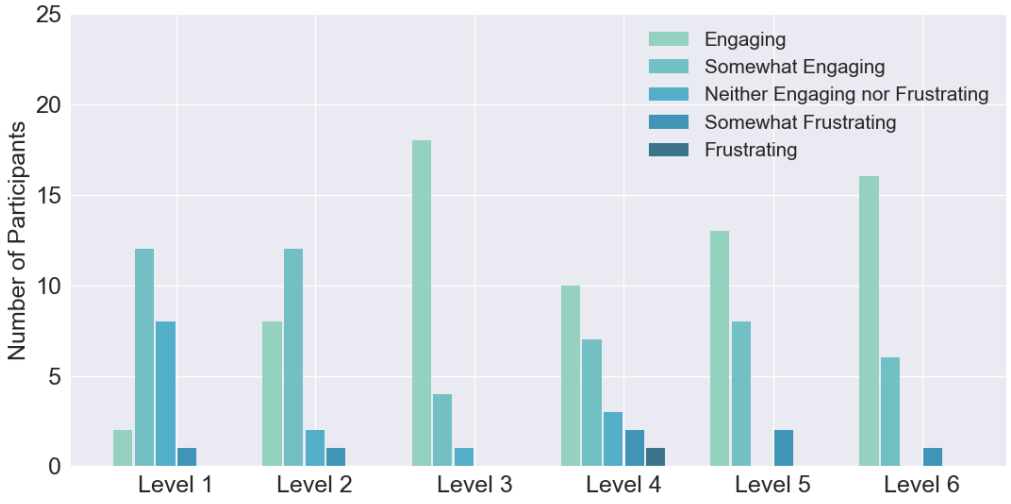

Figure 6.7: Graph display how the respondents rated each level on a scale of "Engaging" to "Frustrating"

From Figure 6.7, we also notice that Level 3 was the most engaging level, and also the only level which was never deemed frustrating. Once again, we find a parallel in Figure 6.8 as Level 3 is the only level to be classed purely positively (only "Interesting"

or “Somewhat Interesting” answers). I surmise that this is due to Level 3 being the first “real” level. Until this point, the user is not asked to hack anything in the traditional sense. While there is a sudden rise in perceived difficulty (see Figure 6.6), the level is also more interesting.

Overall, we notice that as the user progresses through the levels, they are deemed increasingly interesting and engaging. This mirrors the increase in perceived difficulty which we observed in Section 6.6.4.2.

As a general note, users seemed to be largely split into two group with regards to hint usage. One subsection used the hints immediately to provide a starting point. Others never used the hints at all unless they were stuck.

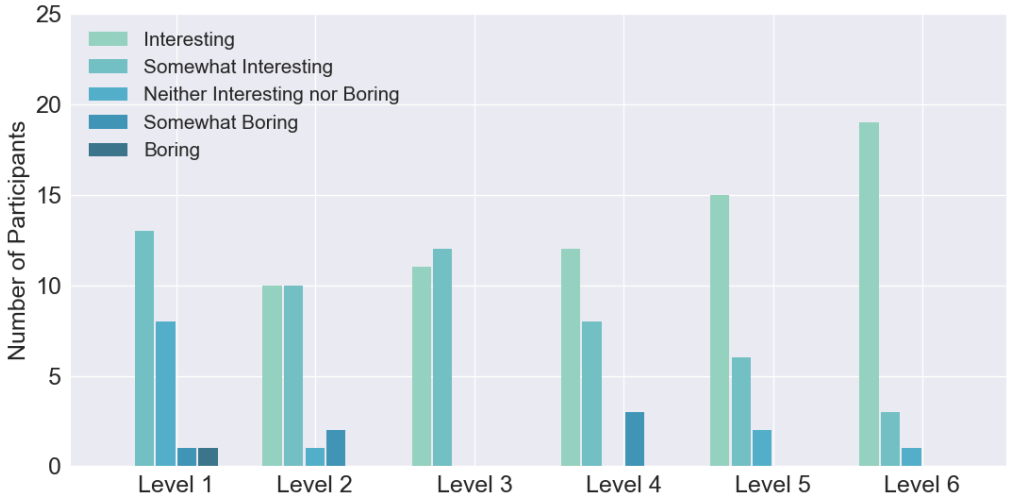

Figure 6.8: Graph display how the respondents rated each level on a scale of “Interesting” to “Boring”

6.7 Summary

In this chapter I explained the design and methodology of my second study. I drew many of the same conclusions as in the previous study as I found a statistically significant improvement in performance, and a higher knowledge retention rate for questions which cover topics with which the users had hands-on experience.

It was also evident that the time taken to complete each level did not vary too much from user to user, though there were of course some outliers. The overall impressions were very positive.

Chapter 7

Discussion and Conclusion

7.1 Discussion

Computer security is an increasingly important issue, and yet despite this, as we have seen, there is a lack of support for future developers. I aimed to fill this gap by creating a system which teaches password security through gamification and puzzle-based learning, to answer the question:

> "How might we design an entry-level tool to teach developers about password security?"

While my previous work addressed the design aspect of this question, I had not covered the crucial aspect of this problem: does the system actually teach password security to the target audience?

The outcome of both my studies indicate that there is an extremely statistically significant improvement in the test scores after interacting with the system. As such, I can say that my system is indeed effective. We also notice that the overall improvement was greater in the second study. One salient difference between the two studies was the time limit. This reinforces the hypothesis that, had the participants had more time in the first study, they might have exhibited a higher learning gain.

Regarding the type of learning, we saw that the students may be retaining concepts which they manipulate directly better. This aligns with the idea of active learning. Though the effects of this have been found to be uneven [20], we can hope that the gamified style of the system will increase retention over time. This could quite easily be the topic of further work, as could be comparing my current system's learning gains to those of traditional classroom methods for example.

Both my own observations and discussion with the Computer Security tutors indicated that the system can be used by students without the need of a supporting instructor. From the second study, I estimate the average time needed to complete the entire system is an hour. In light of this, the system could quite easily be used during a sufficiently long lab session during a computer security course. Equally it could be used

autonomously without the risk that the user would get too bored and give up [7], given the relatively short completion time.

The overall trend of high interest and engagement should also work in the system's favour. I observed that the "Interesting" and "Engaging" ratings in the second study seemed to follow similar patterns, as the same trends could be identified in both results. This is in line with the literature, as there have been correlations found between interest and student engagement [26]. Both of these can be a factor of motivation, which can also have a strong impact on learning [21].

7.2 Conclusion

As we have seen, I conducted two studies on my system in order to evaluate its effectiveness in teaching a specific set of learning outcomes. These studies, designed based on evaluations carried out on other teaching environments, were as follows:

- a large scale evaluation which focussed on breadth of response. The main focus was be the pre- and post-testing.
- a slightly smaller scale evaluation which focussed on depth of response. I collected richer data here by using pre- and post-testing, alongside surveys to collect attitudinal data to better contextualise the direct measures. This covered perceived learning, as well as general attitude towards each level, help-seeking behaviours and time taken on each level.

The findings of both studies aligned both with each other, and with the literature in the field of learning environments. Given the results, I can say that my system is indeed an effective teaching tool. The general reception of the system was very positive, as users found it interesting and engaging (particularly in the later levels), and enjoyed using it.

Bibliography

[1] Improper Error Handling - OWASP. [Online]. https://www.owasp.org/index.php/Improper_Error_Handling/. [Accessed: 08-April-2018].

[2] Passwords - SkullSecurity. [Online]. https://wiki.skullsecurity.org/index.php?title=Passwords. [Accessed: 08-April-2018].

[3] Sharon Ainsworth. Evaluative Methods for Learning Environments. [Online], 2003. http://www.inf.ed.ac.uk/teaching/courses/ale1/Ainsworth_tutorial.pdf. [Accessed: 10-April-2018].

[4] Donald R Bacon. Reporting actual and perceived student learning in education research, 2016.

[5] Albert Bandura. Guide for constructing self-efficacy scales. *Self-efficacy beliefs of adolescents*, 5(1):307–337, 2006.

[6] BBC. Boy, 17, admits TalkTalk hacking offences. [Online]. http://www.bbc.co.uk/news/uk-37990246. [Accessed: 01-April-2018].

[7] Neil A Bradbury. Attention span during lectures: 8 seconds, 10 minutes, or more?, 2016.

[8] John Brooke. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[9] Peter Chapman, Jonathan Burket, and David Brumley. Picoctf: A game-based computer security competition for high school students. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, San Diego, CA, 2014. USENIX Association.

[10] Constance Crowe. Designing a tool to teach password security to future developers. 2017.

[11] Jayesh Navin Shah Dr Rebecca Klahr, Paul Sheriffs, Tom Rossington, Gemma Pestell, Mark Button, and Victoria Wang. Cyber Security Breaches Survey 2017. [Online]. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/fil [Accessed: 05-April-2018].

[12] Andrew J Elliot and Holly A McGregor. A 2× 2 achievement goal framework. *Journal of personality and social psychology*, 80(3):501, 2001.

[13] Tanya Flushman, Mark Gondree, and Zachary NJ Peterson. This is not a game: early observations on using alternate reality games for teaching security concepts to first-year undergraduates. In *8th Workshop on Cyber Security Experimentation and Test (CSET 15), Washington, DC*, 2015.

[14] Maurice Hendrix, Ali Al-Sherbaz, and Victoria Bloom. Game based cyber security training: are serious games suitable for cyber security training? *Int. J. Serious Games*, 3(1), 2016.

[15] Cormac Herley. So long, and no thanks for the externalities: The rational rejection of security advice by users. In *Proceedings of the 2009 Workshop on New Security Paradigms Workshop*, NSPW '09, pages 133–144, New York, NY, USA, 2009. ACM.

[16] Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Julio Lopez. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 523–537, 2012.

[17] Krittaya Leelawong and Gautam Biswas. Designing learning by teaching agents: The betty's brain system. *International Journal of Artificial Intelligence in Education*, 18(3):181–208, 2008.

[18] Scott W Mcquiggan, Jonathan P Rowe, Sunyoung Lee, and James C Lester. Story-based learning: The impact of narrative on learning experiences and outcomes. In *International Conference on Intelligent Tutoring Systems*, pages 530–539. Springer, 2008.

[19] MyFitnessPal. MyFitnessPal Account Security Issue: Frequently Asked Questions. [Online]. https://content.myfitnesspal.com/security-information/FAQ.html. [Accessed: 01-April-2018].

[20] Michael Prince. Does active learning work? a review of the research. *Journal of engineering education*, 93(3):223–231, 2004.

[21] Jonathan P Rowe, Lucy R Shores, Bradford W Mott, and James C Lester. Integrating learning, problem solving, and engagement in narrative-centered learning environments. *International Journal of Artificial Intelligence in Education*, 21(1-2):115–133, 2011.

[22] Rosie Shier. Paired t-tests. [Online], 2004. http://www.statstutor.ac.uk/resources/uploaded/paired-t-test.pdf. [Accessed: 08-April-2018].

[23] Traci Sitzmann, Katherine Ely, Kenneth G Brown, and Kristina N Bauer. Self-assessment of knowledge: A cognitive learning or affective measure? *Academy of Management Learning & Education*, 9(2):169–191, 2010.

[24] SQLite. Appropriate Uses For SQLite. [Online]. https://www.sqlite.org/whentouse.html. [Accessed: 08-April-2018].

[25] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, Yarik Markov, Alex Petit Bianco, and Clement Baisse. Google Online Security Blog: Announcing the first SHA1 collision. [Online]. https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html. [Accessed: 01-April-2018].

[26] Jerry Chih-Yuan Sun and Robert Rueda. Situational interest, computer self-efficacy and self-regulation: Their impact on student engagement in distance education. *British Journal of Educational Technology*, 43(2):191–204, 2012.

[27] Yasemin Tas. The contribution of perceived classroom learning environment and motivation to student engagement in science. *European Journal of Psychology of Education*, 31(4):557–577, Oct 2016.

[28] Wade M Vagias. Likert-type scale response anchors. clemson international institute for tourism. *& Research Development, Department of Parks, Recreation and Tourism Management, Clemson University*, 2006.

[29] Ash Wilson. Universities still struggle to provide cybersecurity education. [Online].

Appendix A

System Screenshots

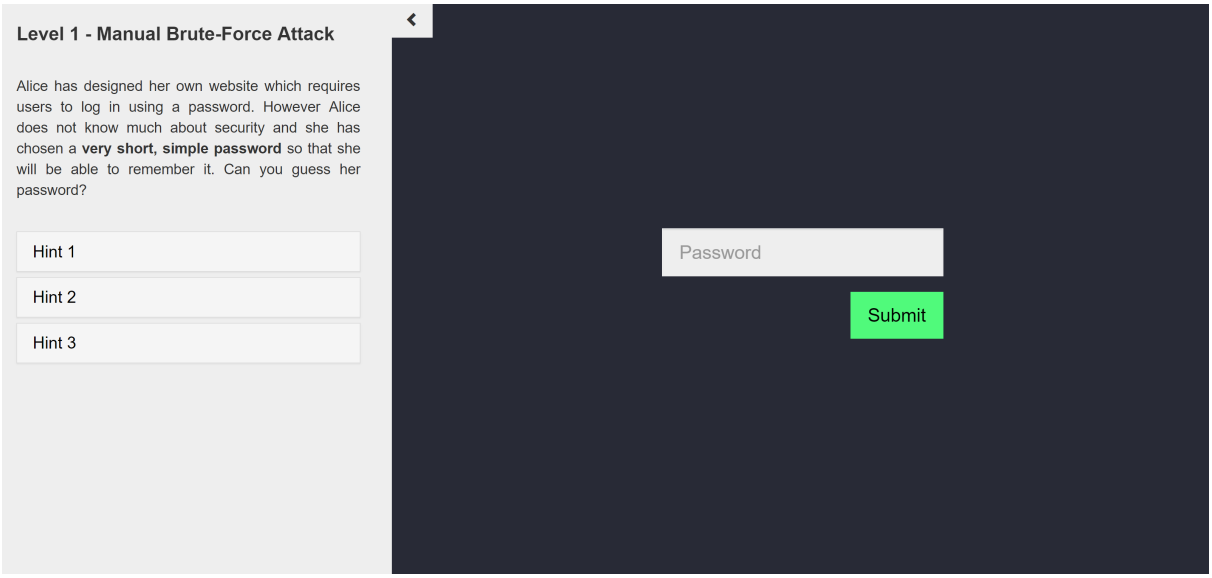

Figure A.1: Screenshot of Level 1

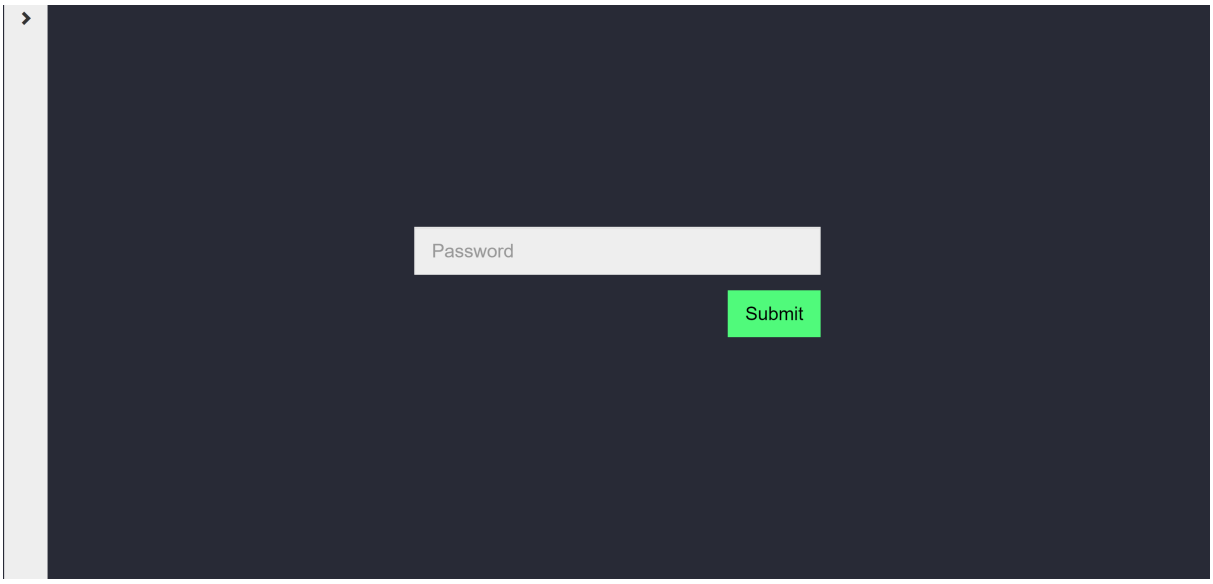

Figure A.2: Screenshot of Level 1 with collapsed sidebar

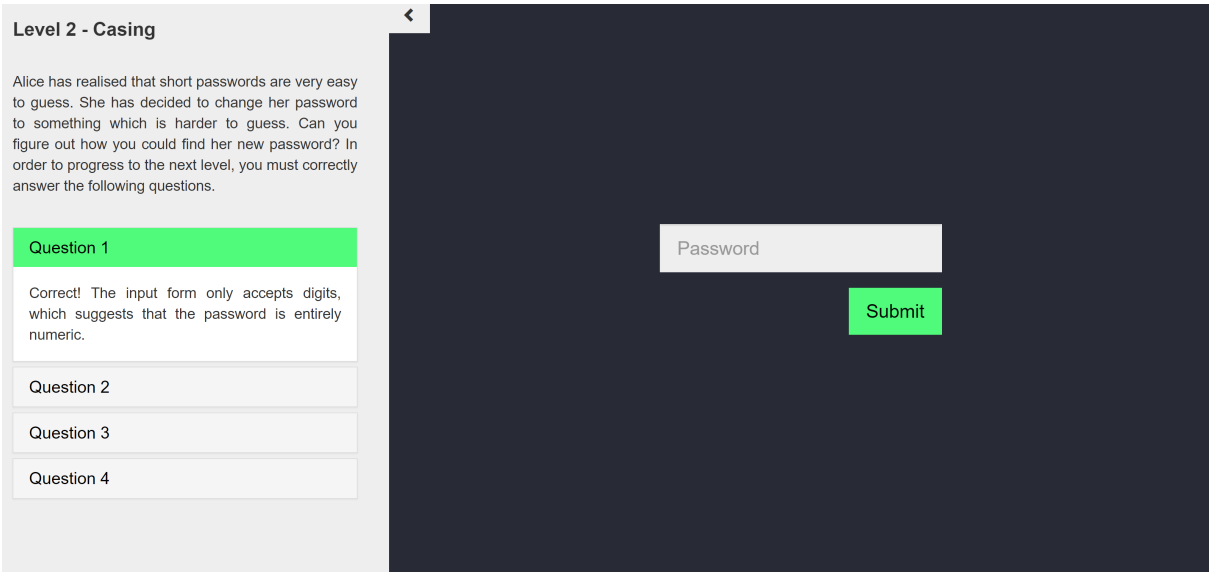

Figure A.3: Screenshot of Level 2 with first question answered correctly

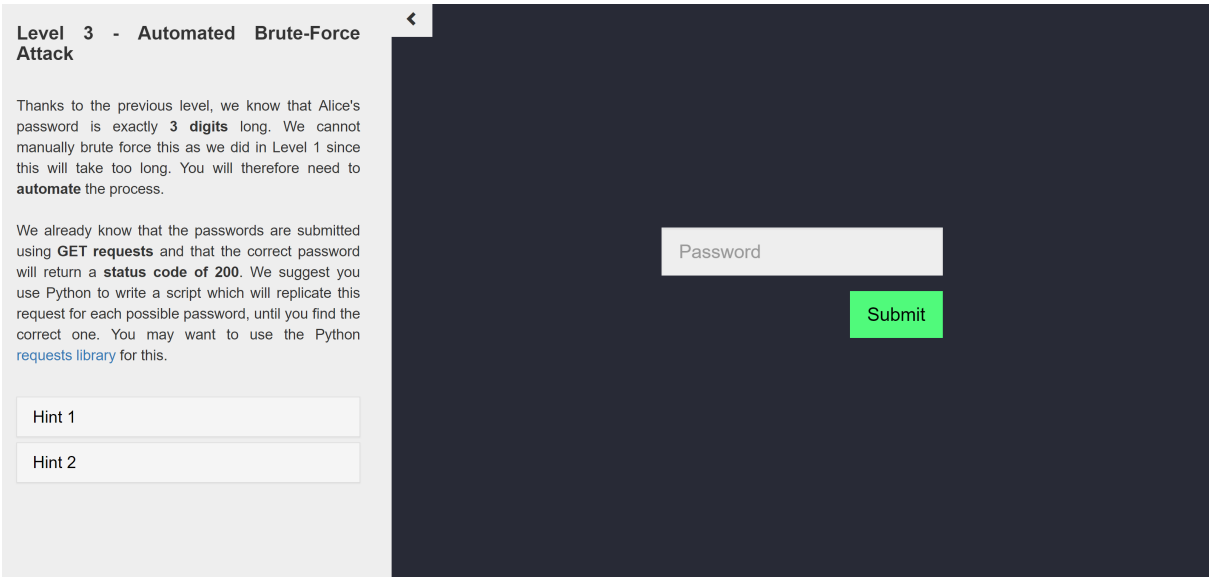

Figure A.4: Screenshot of Level 3

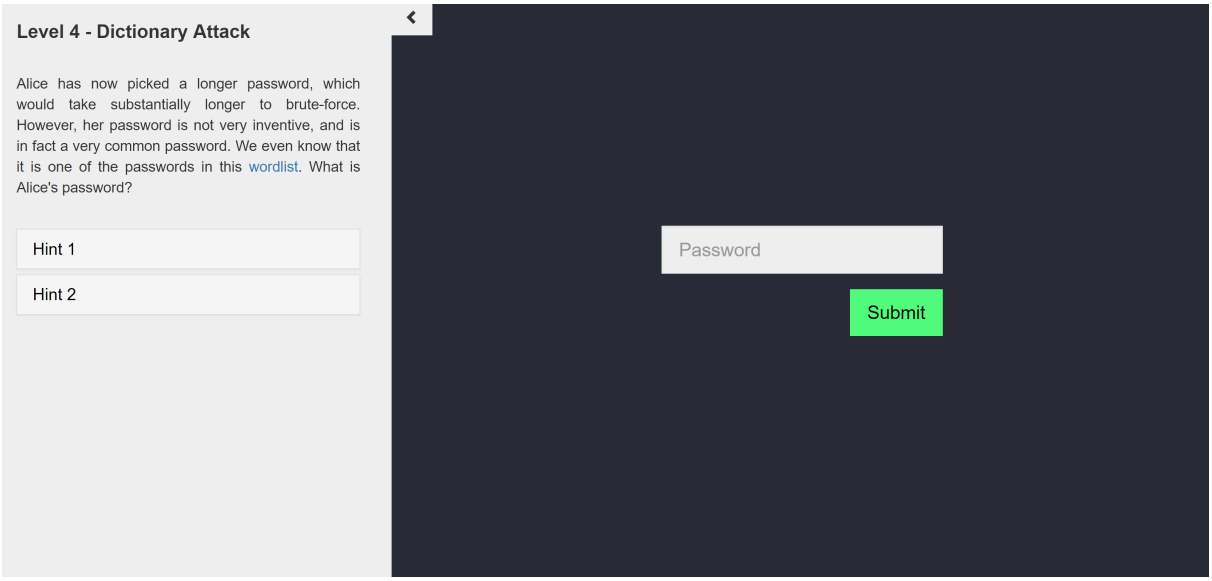

Figure A.5: Screenshot of Level 4

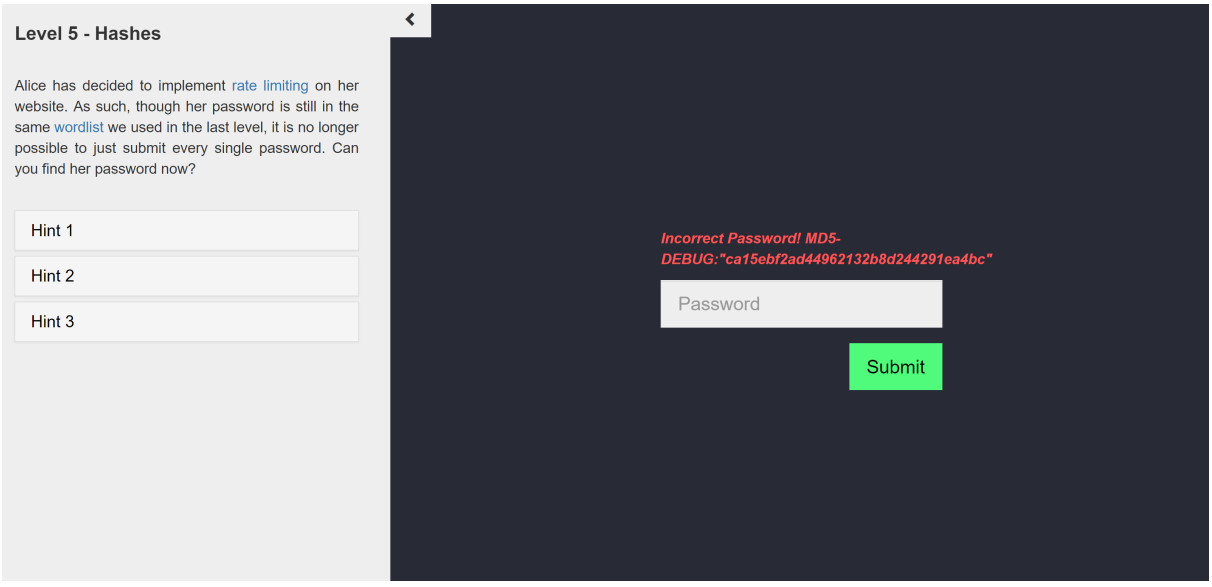

Figure A.6: Screenshot of Level 5 featuring hash of correct password leaked in error message

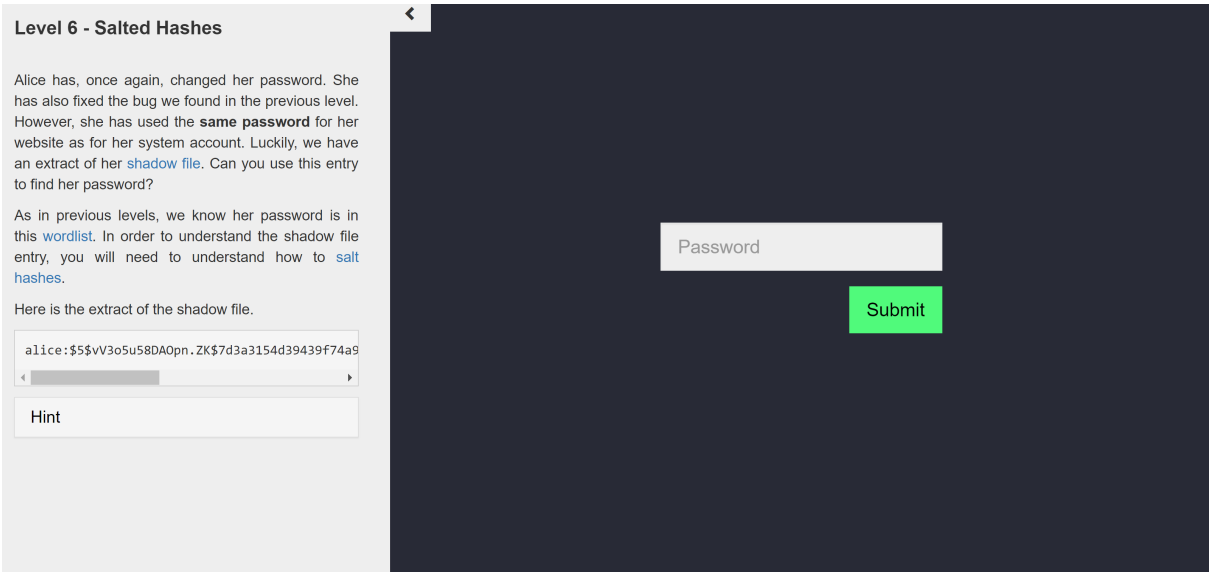

Figure A.7: Screenshot of Level 6

Appendix B

Changes made to my System

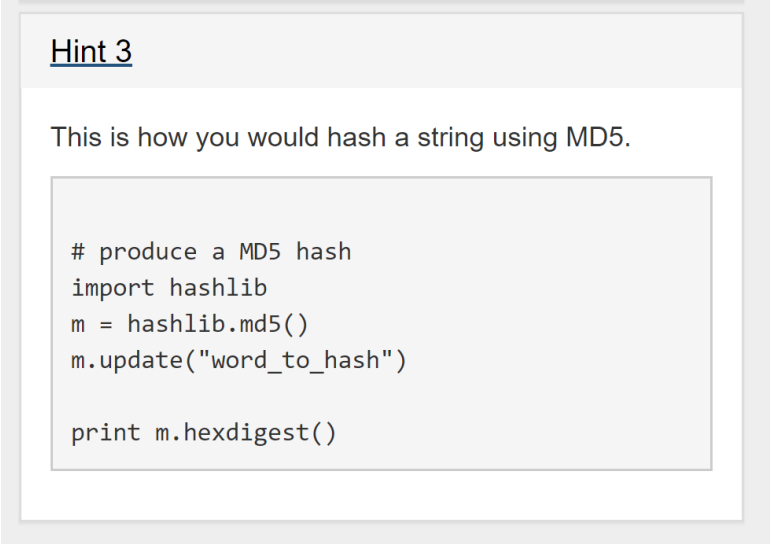

Figure B.1: Pre-update version of the Python Snippet Hint in Level 5

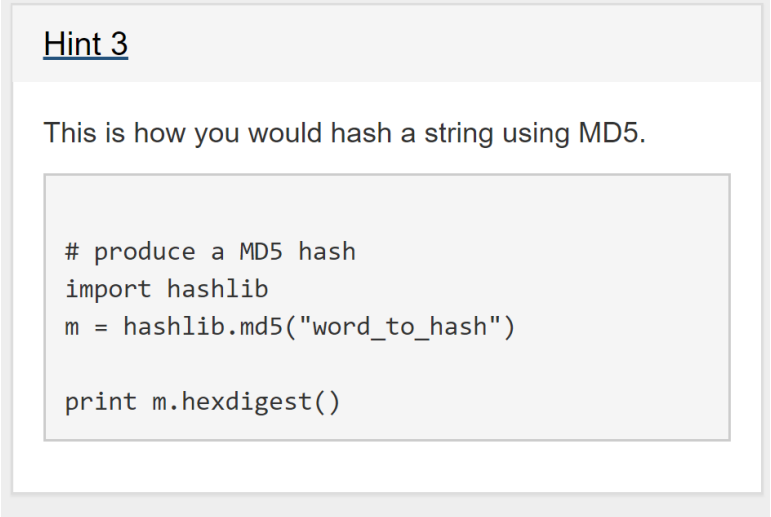

Figure B.2: Update made to Python Snippet Hint in Level 5.

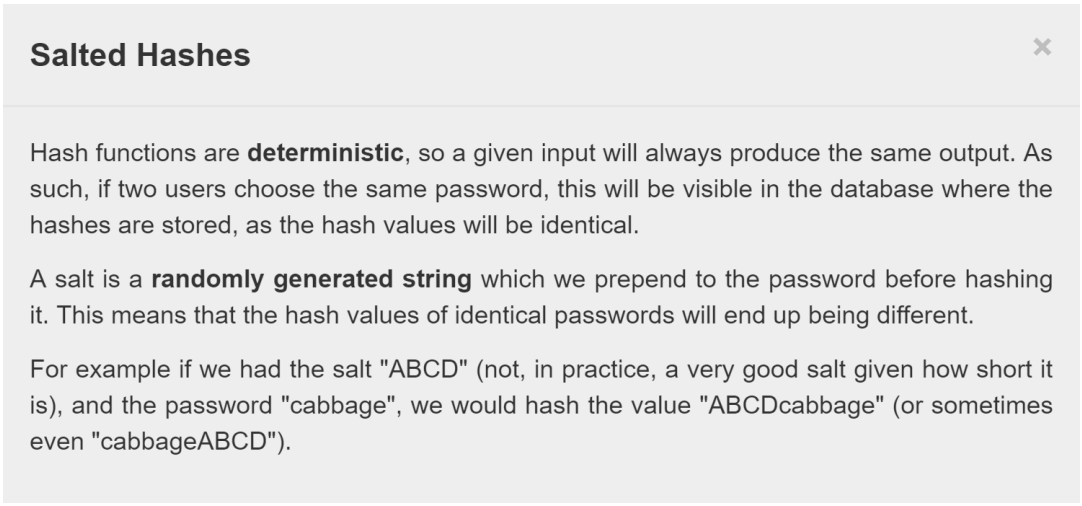

Figure B.3: Pre-update version of the “Salting Hashes” explanation in Level 6

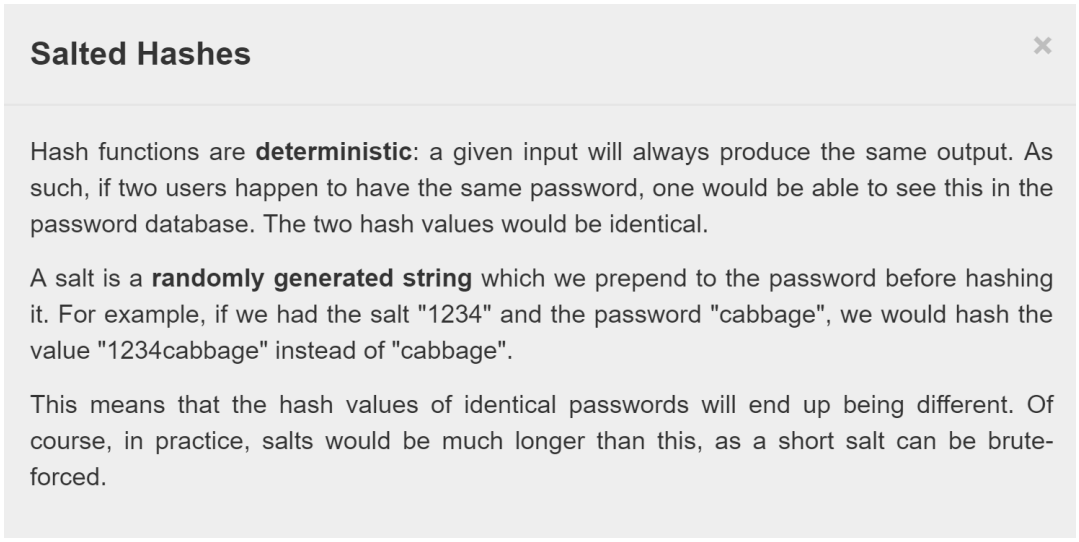

Figure B.4: Update made to “Salting Hashes” explanation in Level 6.

Appendix C

Study 1 Materials

Tutorial 4
Password Security

Informed Consent

Today's tutorial uses an interactive teaching system developed as part of a UG4 honours project last year. The system is intended to teach you about password security from a system administration perspective by having you go through several levels, each of which shows you a different aspect of password security.

This year we would like to test how effective the tutorial is by having you answer some questions before and after taking the tutorial. The questions will be about computer security concepts around passwords. Information you provide on the tutorial sheets will be kept completely anonymous. They will be used to better understand how effective the tutorial is at teaching password security concepts. The results will be used as part of a Masters of Informatics project to inform and improve the design of the system. The information may also be used in research publications on educational security systems.

Participation in the research component of the tutorial is completely voluntary. You will receive no benefit from participating other than the knowledge that you have helped a fellow student with their project. At the end of tutorial you may choose to either turn in this tutorial sheet to your tutor to be used in research or you may choose to opt-out by taking it home or recycling it.

1 Pre-questionaire

Before continuing, please complete the initial questionaire on page 3. This is for research purposes, in order to evaluate the effectiveness of the system. It is **not marked** and completely **anonymous**.

2 Setting up the VMs

The system which you will be testing is hosted in two VMs. You will need to start by installing these VMs by running the `setup.sh` script, located in the `/afs/inf.ed.ac.uk/group/teaching/cs/passwords` directory.

To do this, open a terminal and run:

```
$ cd /afs/inf.ed.ac.uk/group/teaching/cs/passwords
$ ./setup.sh
```

Follow the on-screen instructions. Once setup is complete, VirtualBox will open with two new VMs in the CS2017 folder. You will need to expand this folder and start (by double clicking) both the Passwords VM and the Kali_CS VM.

After started the Passwords VM, you will not need to interact with it again. **You do not need to log in to the Passwords VM.**

Log in to the Kali_CS VM:

username: root

password: toor

3 Using the "Password Security" System

Within your Kali_CS VM, open up a browser (top icon in the toolbar on the left of the screen). Internet access has been disabled, other than for the system itself.

Navigate to: **192.168.1.60/password_cracking/level1**

This is level 1. In order to progress to the next level, you must "hack"the website in a specific way (explained in each level) to figure out the password for the current level. There are no penalties for using the in-system hints.

The goal is to complete as many of the 6 levels as possible.

4 Post-questionaire

When instructed by the tutor or when you have completed all the levels, please complete the post-questionaire on page 4.

Answer before you start

Please answer the questions below as best you can before starting the tutorial. If you don't know the answer then please select your best guess, or write "I don't know".

1. **We would like to use your answers in research publications and to improve this tutorial.**
   - ○ You may use my answers below in research publications
   - ○ Do not use my answers below in research publications

2. **Which of the following statements describe a POST request? Tick all that apply.**
   - □ retrieves information from the web server
   - □ sends information to the web server, most likely to be stored
   - □ data is enclosed in the body of the HTTP request
   - □ data is visible in the URL

3. **What is the status code of a successful HTTP request?**
   _____

4. **Which of the following statements best complete this description of a dictionary attack? An attacker performs a dictionary attack by systematically submitting:**
   - ○ all possible password combinations
   - ○ all the words in the English dictionary
   - ○ all the passwords in a pre-established list of passwords
   - ○ what is a dictionary attack?

5. **How would a developer secure their website against a brute force attack? Tick all that apply.**
   - □ caching the hashes of the users' passwords
   - □ account lock out if too many incorrect attempts
   - □ sanitising user input
   - □ rate limiting

6. **Which of the following statements best describes a cryptographic hash function?**
   - ○ hard to compute, hard to invert
   - ○ hard to compute, easy to invert
   - ○ easy to compute, hard to invert
   - ○ easy to compute, easy to invert

7. **How would a developer stop a rainbow table attack?**
   - ○ logging
   - ○ salting
   - ○ fuzzing
   - ○ blacklisting

8. **On a Linux system, where are the users' passwords stored?**
   - ○ the ".bashrc" file
   - ○ they aren't
   - ○ the "passwd" file
   - ○ the "shadow" file

Answer after you are done

After you have completed the tutorial, please answer the questions below as best you can. If you don't know the answer then please select your best guess, or write "I don't know".

9. How many levels did you complete?

_____

10. Which of the following statements describe a POST request? Tick all that apply.

- ☐ retrieves information from the web server
- ☐ sends information to the web server, most likely to be stored
- ☐ data is enclosed in the body of the HTTP request
- ☐ data is visible in the URL

11. What is the status code of a successful HTTP request?

_____

12. Which of the following statements best complete this description of a dictionary attack? An attacker performs a dictionary attack by systematically submitting:

- ○ all possible password combinations
- ○ all the words in the English dictionary
- ○ all the passwords in a pre-established list of passwords

13. How would a developer secure their website against a brute force attack? Tick all that apply.

- ☐ caching the hashes of the users' passwords
- ☐ account lock out if too many incorrect attempts
- ☐ sanitising user input
- ☐ rate limiting

14. Which of the following statements best describes a cryptographic hash function?

- ○ hard to compute, hard to invert
- ○ hard to compute, easy to invert
- ○ easy to compute, hard to invert
- ○ easy to compute, easy to invert

15. How would a developer stop a rainbow table attack?

- ○ logging
- ○ salting
- ○ fuzzing
- ○ blacklisting

16. On a Linux system, where are the users' passwords stored?

- ○ the ".bashrc" file
- ○ they aren't
- ○ the "passwd" file
- ○ the "shadow" file

17. Comments?

_____

Appendix D

Study 2 Materials

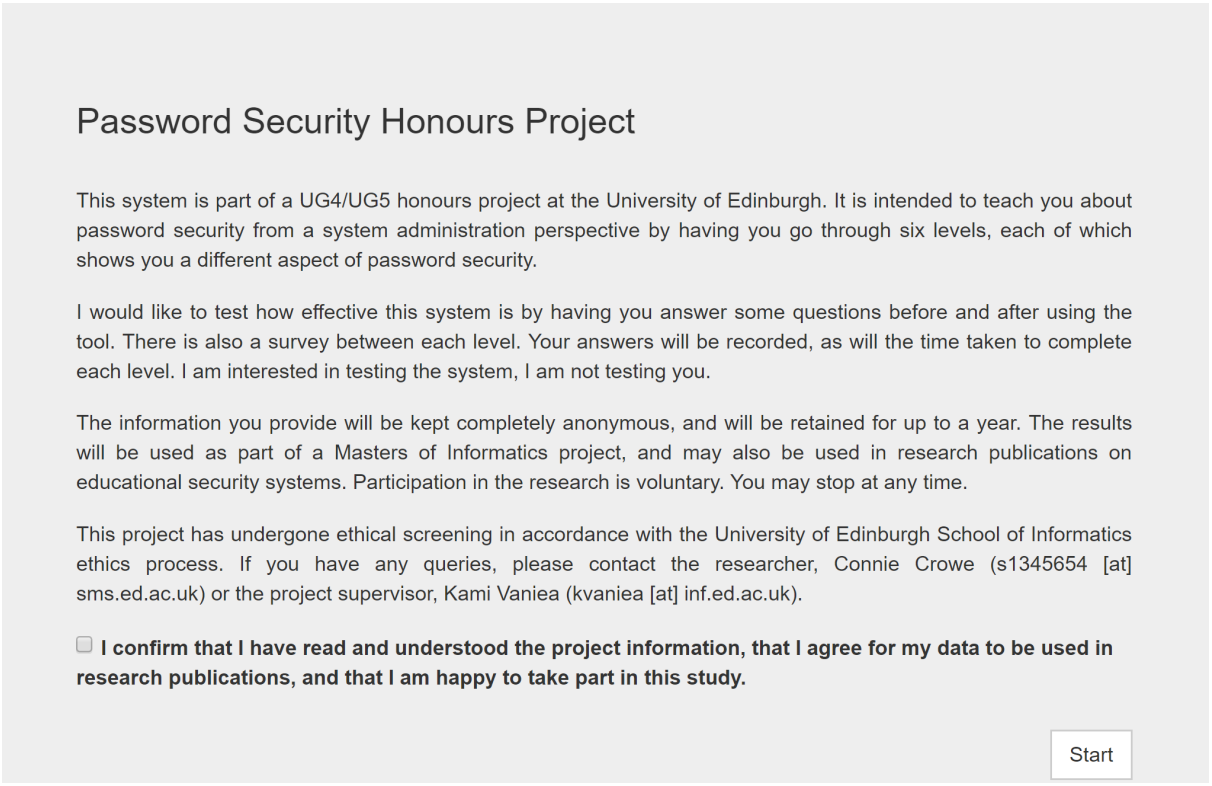

Figure D.1: Screenshot of the consent form. This is the home page of the system.

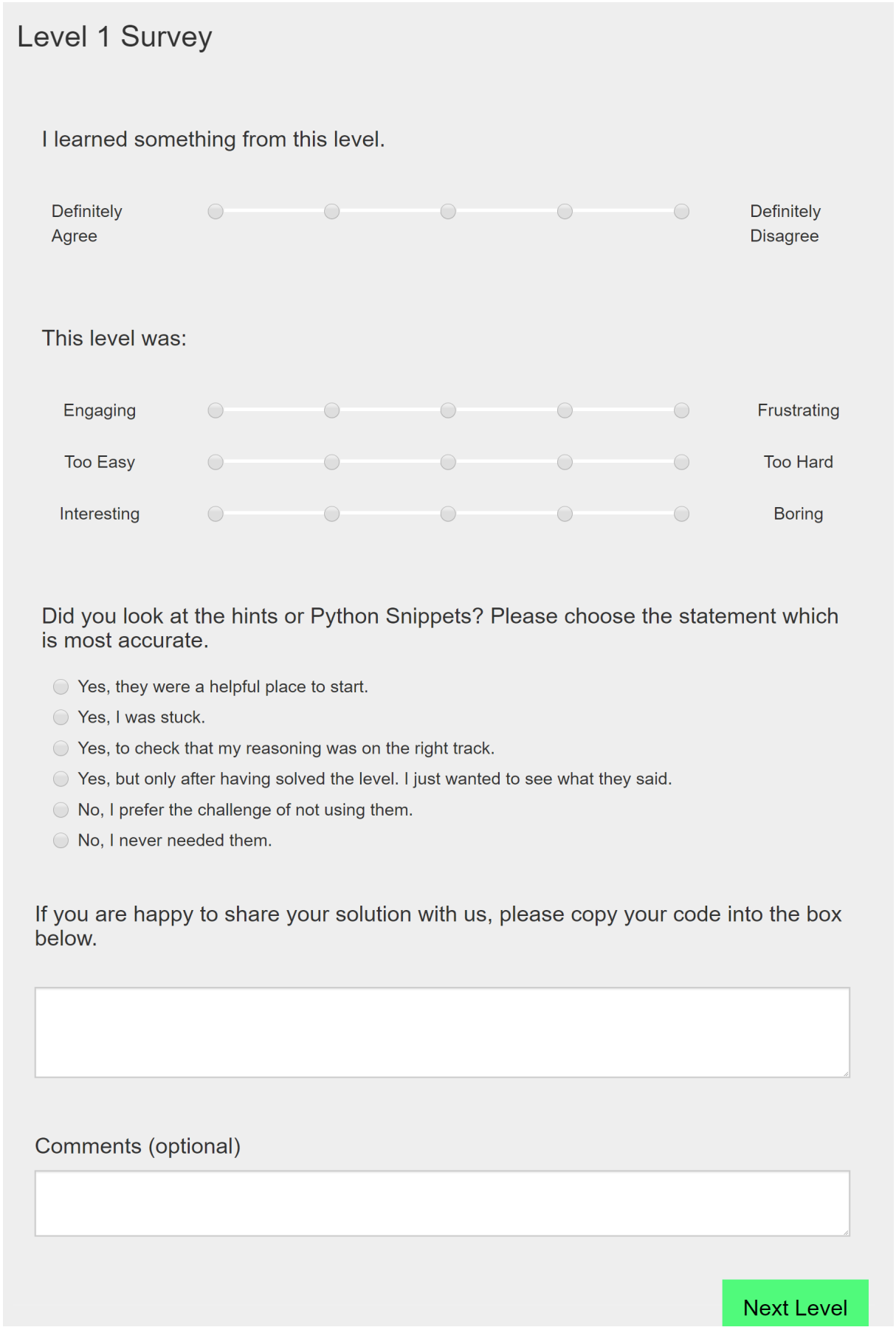

Figure D.2: Screenshot of the inter-level survey for Level 1