Abstract

In today’s digital society, the importance of cybersecurity knowledge is ever-increasing. Many platforms exist which teach different topics in cybersecurity. However, most of these platforms are either targeting professional employees of cybersecurity companies, or they are used for hosting CTF competitions. The platforms tailored towards a younger and less experienced audience still require the installation of additional software, creating an unnecessary entry barrier. This thesis presents a novel learning platform which is entirely browser-based and teaches cybersecurity topics to high-school students. It consists of a lesson center which houses the lesson content and provides helper tools for solving the lessons. Additionally, a target application supplies a playground where students are able to implement what they learned and try their own ideas. In our prototype, the target application is implemented as a social media network with intentional vulnerabilities which are demonstrated to the students in the lessons. To ensure the practical usability of the constructed prototype, it was tested with 10 experts. The feedback collected from this process is categorized and implemented based on concept coding.
Acknowledgements

I would like to thank my supervisor Daniele Lain for the regular meetings and the valuable feedback throughout the whole process of developing the tools presented in this thesis. My thanks also go to Dennis Komm, who provided me with precious insights and a workspace in his group’s office. Additional thanks go to Srdan Čapkun, who played a key part in making this thesis happen in the first place. Next, I would also like to thank my friend Adrian Kress with whom I initially discussed the idea for the thesis. Furthermore, I thank my friend Clemens Bachmann for letting me stay at the apartment in the mountains for a nice and refreshing breather. Moreover, I would like to thank the experts for voluntarily testing the platform and providing valuable feedback. Last but not least, I thank my family and friends for their continued support throughout my studies.
# Contents

<table>
<thead>
<tr>
<th>Contents</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2 Background</td>
<td>3</td>
</tr>
<tr>
<td>2.1 Education</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Education Platforms</td>
<td>4</td>
</tr>
<tr>
<td>2.2.1 E.Tutorial</td>
<td>4</td>
</tr>
<tr>
<td>2.2.2 Hack the Box</td>
<td>4</td>
</tr>
<tr>
<td>2.2.3 OverTheWire Wargames</td>
<td>4</td>
</tr>
<tr>
<td>2.2.4 PicoCTF</td>
<td>5</td>
</tr>
<tr>
<td>2.2.5 pwn.college</td>
<td>5</td>
</tr>
<tr>
<td>2.3 Technology</td>
<td>5</td>
</tr>
<tr>
<td>2.3.1 OpenAPI Specifications</td>
<td>5</td>
</tr>
<tr>
<td>2.3.2 Django</td>
<td>6</td>
</tr>
<tr>
<td>2.3.3 Django REST API</td>
<td>6</td>
</tr>
<tr>
<td>2.3.4 Django Channels</td>
<td>7</td>
</tr>
<tr>
<td>2.3.5 Python Requests</td>
<td>7</td>
</tr>
<tr>
<td>2.3.6 Postman</td>
<td>7</td>
</tr>
<tr>
<td>2.3.7 React</td>
<td>7</td>
</tr>
<tr>
<td>2.3.8 TypeScript</td>
<td>8</td>
</tr>
<tr>
<td>2.3.9 React Router</td>
<td>8</td>
</tr>
<tr>
<td>2.3.10 Axios</td>
<td>8</td>
</tr>
<tr>
<td>2.3.11 Bootstrap</td>
<td>8</td>
</tr>
<tr>
<td>2.3.12 Eruda</td>
<td>9</td>
</tr>
<tr>
<td>2.3.13 Docker</td>
<td>9</td>
</tr>
<tr>
<td>3 Problem Statement</td>
<td>11</td>
</tr>
<tr>
<td>4 System Overview</td>
<td>13</td>
</tr>
</tbody>
</table>
4.1 Introduction ............................................. 13
4.2 Lesson Center ......................................... 14
4.3 Managing Lesson Content ............................... 16
4.4 Social Media Network .................................. 16
4.5 Proxy Service .......................................... 17
4.6 Moderation Tools ....................................... 17
4.7 Research Aids ......................................... 18

5 Implementation ........................................ 21
  5.1 Lesson Center Backend ................................. 21
    5.1.1 User Model ...................................... 21
    5.1.2 User Signup ................................... 22
    5.1.3 Lesson Model ................................... 22
    5.1.4 Proxy Server ................................... 23
    5.1.5 WebSocket Connections ......................... 24
  5.2 Lesson Center Frontend ............................... 25
  5.3 Social Media Backend ................................. 28
    5.3.1 User Signup ................................... 28
    5.3.2 Intentional Vulnerabilities ....................... 29
    5.3.3 Content Moderation .............................. 29
  5.4 Social Media Frontend ............................... 30
    5.4.1 Custom Console ................................ 30
  5.5 Deployment .......................................... 31

6 Designing the Lessons ................................. 33
  6.1 Introductory Lesson .................................. 34
  6.2 General Knowledge Lesson about HTTP ............. 34
  6.3 Specific Lesson about HTTP Cookies ................. 35

7 Evaluation .............................................. 37
  7.1 Expert Testing ...................................... 37
    7.1.1 Method ......................................... 37
    7.1.2 Results ......................................... 37
  7.2 In-Class Testing ..................................... 39
    7.2.1 Planned Method ................................ 39

8 Conclusion ............................................ 41
  8.1 Future Work .......................................... 42

Bibliography .............................................. 45

A Semi-Structured Interview Questions for Experts .... 49
B Semi-Structured Interview Questions for Teachers .... 51
Chapter 1

Introduction

In today’s online world, the importance of cybersecurity is ever-increasing. It feels like a company is hacked every other week. To combat this issue, the overall education level of computer science and cybersecurity in particular has drastically increased over the past years – on a technical front, as well as an awareness front. In Switzerland, education in media and computer science already starts in primary schools.\(^1\) With education starting this early, proper tools are vital for the implementation of a sustainable learning experience.

Various platforms exist which teach computer science and specifically topics in cybersecurity to their users. However, most of these platforms are either focusing on furthering the education of existing industry professionals, or they are used in a leisure setting to host various cybersecurity-related challenges. The little number of platforms which are actually targeting a younger and/or less experienced audience mostly still have a rather high entry barrier, as they require the setup of additional software and potentially depend on existing programming knowledge.

This thesis introduces a novel learning platform which focuses on teaching cybersecurity topics to high-school students. The platform is fully browser-based and does not require any prior programming experience, thus removing the entry barrier of existing platforms. The platform consists of two components: the lesson center and a target application. The former holds various lessons which can be worked through by the students, while the latter serves as a playground for the students to apply their newly learned knowledge and experiment with their own ideas. Helper tools in the lesson center provide useful information about the student’s interaction with the target application. In our prototype implementation, a social media network serves as the target application, while a HTTP request inspector serves as a helper tool in the lesson center.

\(^1\)See Lehrplan 21, https://lehrplan.ch/ (German only).
Chapter 2

Background

This chapter explores some related work in the field of education and online learning platforms. Additionally, it introduces the technologies which are used in the implementation of our platform.

2.1 Education

The learning-by-doing principle has been explored by many previous works, including Dewey [1], Anzai and Simon [2], and Reese [3]. The general consensus is that the learning process is improved when applying the studied material in a practical setting.

The area of constructionism also plays a key part in today's computer science education. As a notable example, Papert [4] uses the “Turtle” as a simple programmable object that students can interact with and learn some basics of mathematics and programming in a playful manner.

To further investigate the learning-by-doing principle, Schank and Kass proposed a learning environment called the Goal-Based Scenario [5]. Such environments should follow the following principles, in the words of Schank and Kass [5]:

- “Generate goals that will motivate students to access information
- Provide an authentic context in which to situate the knowledge students access
- Confront students with specific challenges that require them to analyze the information they access, and put it to use”

Vykopal, Švábenský, and Chang explored the effects of using Capture the Flag (CTF) competitions in university courses [6]. They recommend using dynamic hints when a student is stuck at a particular challenge, as static hints have the potential of being useless since they do not account for the
2. Background

current progress of the student. They also found that using CTF competitions may lead to flag-sharing between students.

Petersen, Santos, Smith, Wetzel, and Witte propose the NICE framework [7] as part of the National Initiative for Cybersecurity Education (NICE). In their own words, the NICE framework "provides a reference taxonomy – that is, a common language – of the cybersecurity work and of the individuals who carry out that work." [7] They provide clear definitions for the terms Task, Knowledge, and Skill which can then be used to create learning material to teach new competencies to participants.

2.2 Education Platforms

There exist several platforms with the goal of educating users in the area of cybersecurity and computer science in general. This section explores some of these platforms.

2.2.1 E.Tutorial

The E.Tutorial platform [8] has been developed by a team at ETH Zurich and it provides several e-learning courses. These courses are mostly about basic computer science topics and general security awareness. The target group of the platform depends on the course. Some courses are explicitly designed for high-school students while others are created for university members. At the time of this thesis, there is one course about cybersecurity which focuses on several awareness principles, such as regularly updating a device or locking the screen when temporarily leaving the workspace.

2.2.2 Hack the Box

Hack the box [9] is an education platform that provides a gamified hands-on learning experience. It is mainly tailored towards existing hackers or professionals, but also provides a learning environment for universities. Their online academy also provides a way of self-learning the essentials of cybersecurity. It also functions as a platform for hosting hacking competitions.

2.2.3 OverTheWire Wargames

The OverTheWire Wargames [10] are a collection of gamified CTF challenges where the user needs to connect to a host by using SSH. Each game is made up of various levels where each level has a specific SSH user. To progress to the next level, the user needs to find the password for the SSH user of the next level. This can be done by following the instructions listed on the website.
2.2.4 PicoCTF

PicoCTF [11] is the largest high-school hacking competition according to their website. They host various CTF competitions as well as a practice section (appropriately called picoGym) where challenges of various difficulties can be solved on their own, including a hint system for users to receive further assistance when stuck at a problem.

2.2.5 pwn.college

The pwn.college [12] is an education platform developed by the Arizona State University. It provides a fully browser-based system using virtual desktops which are accessible directly in the web browser. This reduces the setup cost, as all needed programs are already installed on the virtual machines which users connect to. The challenges are complemented by free lectures which are available on the websites YouTube channel with additional live streams through their Twitch channel.

2.3 Technology

While the technology listed in this section is an essential part of the implementation of the platform, it is not necessary to know about the different frameworks and libraries which are used if only interested in the education portion of this project.

2.3.1 OpenAPI Specifications

In order to simplify the design and implementation process of the whole platform, the complete HTTP API has been documented using the OpenAPI specifications [13]. These specifications form a specification language to document HTTP APIs. They have been created by the OpenAPI initiative which itself is a part of the Linux foundation. Writing these specifications helps with documenting the API in its implementation phase for the backend as well as the frontend. It functions as the single source of truth. If there are ever any uncertainties while implementing either a part of the backend or a part of the frontend, a quick consultation of the documented specifications mostly resolves any issues.

Unfortunately, the OpenAPI specifications do not allow the documentation of WebSockets, their functionality, or their application-level communication protocol. This is because WebSockets are only initiated over HTTP but are otherwise a completely different protocol. They thus are beyond the scope of the HTTP API specification.
2. Background

2.3.2 Django

Django [14] is an open-source web framework written in Python. It facilitates rapid development of web services as it takes care of a lot of redundant code and lets developers build systems without the need to write a lot of boilerplate code. Django is used in the backends of the platform. It uses its own database abstraction layer in the form of models. This lets the developers work on their implementation without worrying about the specific database-level details. This is further strengthened by the fact that the database backend can be easily switched, even using different database management systems.

For our platform, the default database backend is used, which is an SQLite database file. While this is not the suggested setup for a real-life deployed system, the suspected traffic of the platform is low enough that the sequential accesses to the SQLite database file should not constitute a bottleneck. If it turns out that the network traffic is reaching a limit due to the database, it is rather simple to upgrade to a more sophisticated database setup, such as PostgreSQL or MySQL, thanks to the aforementioned database abstraction of Django.

2.3.3 Django REST API

The vanilla version of Django can be used to build a wide variety of web services. For this platform however, Django is used to only build REST APIs.\(^1\) A Python package which helps with the implementation of this goal is the Django REST framework [15]. It extends the functionality of the standard Django implementation with many features, most notably for this platform are API views, serializers, and authentication schemes. All of these features let the developers drastically reduce the amount of boilerplate code in their implementation.

Additionally, the Django REST framework has a plethora of customizable settings, such as the supported content type of the response. This way, the whole REST API can be changed from, e.g, JSON responses to XML responses by changing a single setting. The framework takes care of everything else, such as response formatting and setting the correct headers.

In the case of this platform, all backend REST API endpoints use a JSON response format. Wherever required, the request input data is also supported in JSON format. For the endpoints which require the client to send binary data, such as an image or video, the request content type “multipart/form-data” is used, as JSON does not support the sending of binary data.

\(^1\)REST is an abbreviation of “representational state transfer” which is a way for different software applications to communicate and exchange data over the internet in a structured and standardized manner.
2.3. Technology

2.3.4 Django Channels

The standard version of Django only handles HTTP requests. As the platform is also required to handle certain WebSocket connections, it uses the official Django extension called *Django Channels* [16]. *Django Channels* essentially adds support for other application-level protocols next to HTTP, such as WebSockets or certain IoT protocols. This is achieved by using an underlying asynchronous event loop while still preserving the default synchronous request management of Django. Depending on the application-level protocol of the incoming request, the request is routed to the correct protocol handler.

*Django Channels* also adds support for a cross-protocol distributed messaging system which they call the “channel layer”. This allows consumers to connect to message groups. Whenever a message is sent from anywhere to the message group, each consumer which is subscribed to that specific message group will be called.

Our platform uses *Django Channels* to add support for WebSockets. All WebSockets are implemented in a way that prohibits anonymous users from connecting. The channel layer is used to push messages from the HTTP endpoints to connected WebSockets.

2.3.5 Python Requests

The platform implements a proxy service which forwards HTTP requests from one component to another (further described in Section 4.5). To make this possible, the Python package *requests* [17] is used. It provides a simple-to-use interface for making HTTP requests.

2.3.6 Postman

In order to test the backend REST APIs thoroughly, a program called *Postman* [18] was used. It provides a way of creating a collection of HTTP API requests and also lets developers implement tests. There is a *Postman* collection provided in each backend repository in the `docs` folder.

2.3.7 React

*React* [19] is a JavaScript library which is used by the platform to build the frontend applications. It allows for a fast development process and lets developers create reusable components. These components are written in a function-based manner where a component is defined by a function which returns HTML elements. With its special JSX syntax, React lets developers directly write HTML elements in the JavaScript code. Special function hooks are provided to create advanced functionality, such as managing state, running side-effect functions, or accessing global context.
2. Background

2.3.8 TypeScript

TypeScript [20] is an extension of JavaScript which introduces a type system and static type checking. This helps with reducing a wide variety of bugs, such as accessing nonexistent variable attributes. A configuration file lets the developer further refine their preferences and add further code design principles, such as disallowing unused local variables or forcing a strict definition of function types. When the written code passes all configured type checks, it is transpiled to JavaScript code which can then be run in browsers.

To improve the development experience and reduce the amount of potential bugs, TypeScript is used for the development of the platform. React is fully compatible with TypeScript, including the special JSX syntax.

2.3.9 React Router

To support multiple pages within a single-page application, a React app needs to implement some sort of routing system. The package React Router [21] solves this problem by introducing routing components which allows for the display of different components based on the current location. All frontends use this package for routing.

2.3.10 Axios

To make requests from within a React component, the use of a request library is necessary. The HTTP client library Axios [22] has several advantages compared to the standard JavaScript library XMLHttpRequest, such as better backwards compatibility or automatic JSON data parsing. It is promise-based, taking further advantage of more modern features compared to its stone-age ancestor.

To incorporate these advantages, the frontends of the platform use the Axios library for all communication with the backends.

2.3.11 Bootstrap

In order to have an easier time with designing the style of the frontend, the use of a styling framework is imperative. For our platform, the Bootstrap [23] library is used. It was initially developed by a designer at Twitter and provides a responsive layout design to have a clean and modern-looking frontend on computers, tablets, and phones.

To neatly integrate into the React ecosystem, the package React Bootstrap [24] is used which provides React components for every Bootstrap component.
2.3.12 Eruda

For parts of the lessons it is necessary to interact with the website through the console (see Section 6.3). Unfortunately, there is no console integrated in a lot of mobile web browsers. Thus, the package Eruda [25] is used which implements an interactive tool that is mimicking the developer tools common in most computer browsers.

An additional feature of Eruda is its configurability. A danger of using the real developer tools is that students without any prior knowledge of the developer tools might be overwhelmed by the large amount of information. Eruda lets the developers configure which tools should be visible. Thus, it can be restricted to only show the console, but hide the other widgets, such as the HTML inspector, the network analysis, or the runtime analysis.

2.3.13 Docker

In order to bring all services together into a complete learning platform, Docker [26] is used, together with its compositing tool. Docker is a containerization platform where each individual component of a system is running in its own isolated container. A tool called Docker Compose is then used to bring the individual containers together to create a larger multi-container setup. This ecosystem is simple to use and easy to maintain, as it allows for host-independent deployment of the whole platform, saving developers many sleepless nights worrying about deploying.
Chapter 3

Problem Statement

Platforms for teaching cybersecurity mostly fall into two main categories, as explored in Section 2.2. Either they target cybersecurity professionals who are already at an advanced training level, or they are used for hosting CTF competitions.¹

Moreover, such platforms often have a rather high entry cost, as they are not solely browser-based and require additional tools to be installed on the user's host machine.

Therefore, the aim of this thesis is to develop a learning platform that satisfies the following conditions:

- It teaches important topics in cybersecurity
- It is tailored towards high-school students without a lot of prior knowledge in computer science
- There is no additional setup required, the platform should be entirely browser-based
- It provides a playground for applying the learned topics and lets students test out their own ideas

¹CTF stands for Capture the Flag. It is a type of competition where a task needs to be solved with the goal of finding a specific identifier – the flag.
Chapter 4

System Overview

This chapter focuses on a high-level overview of the learning platform. For a more detailed description of the implementation, please refer to Chapter 5.

4.1 Introduction

Given the problem statement provided in Chapter 3, a system was designed which hosts interactive cybersecurity lessons for high-school students to work through on their own, accompanied by appropriate helper tools. A target application forms the playground for the students to apply the learned knowledge and also to experiment with their own ideas. Thanks to the fully browser-based approach, there is no additional technological entry cost for the students other than the ability to use a web browser.

The system design consists of two main parts: The lesson center and the target application. A visual overview is provided in Figure 4.1. The lesson center can be used to create guided lessons to teach students about a certain topic. The lessons are designed to be interactive by providing the target application as playground where students can work through the lessons and complete the required tasks. The system is built such that all interactions with the target application are sent through the lesson center. This way, the lesson center can automatically track the student’s lesson progress.

As a prototype, a small social media network was chosen as the target application. Social media networks play an ever-increasing importance in the lives of students, thus this target application is highly relevant to the youth.

Note that also other forms of target applications are possible. As an example, the target application might be a gdb instance when creating a lesson about the computer’s memory.
4. System Overview

Figure 4.1: A complete system overview. The parts shaded in blue are part of the lesson center. The parts shaded in green are part of the target application.

4.2 Lesson Center

The lesson center is the entry point of the whole platform. Students are able to login to their account and access the lessons. In an overview, their personal lesson progress of all available lessons is visible (see Figure 4.2). If they choose to open a lesson, they are presented with the lesson content, as well as the target application’s helper tool (see Figure 4.3). In the case of the social media network, this helper tool is a list of all requests the user made to the social media network. Refer to Section 5.2 for further details about the request inspection helper tool.

Each lesson consists of a list of lesson steps which each can be of four different
4.2. Lesson Center

Figure 4.3: View of a single lesson. The request helper tool is on the right.

types: information, question, open question, or task.

Information  This type of lesson step is the simplest. It only consists of pure content, there is no interaction with the student. Due to this non-interactivity, steps of this type do not count towards the lesson progress.

Question  A lesson step of type “question” has some content and asks a certain question which has a single predefined answer. The answer may be a static string, or it may be dynamically evaluated based on the user. As an example, the students might be asked for the user ID of their social media account which is different for every user. Nonetheless, the answer is predefined and only one single answer per user is correct.

Open Question  This type of lesson step is similar to the previously mentioned “question” type as it also consists of some content and a question. However, the question asked is open and does not have a predefined answer. Students can submit arbitrary-length answers. The provided answers are not directly evaluated, but rather stored to be evaluated by a human afterwards. This may be useful for teachers to test whether the lesson content was actually understood. Open questions can be re-submitted.

Task  Task steps require the user to perform a specific action in the target application. In the case of the social media network, they might be asked to create a new post while logged in as a specific social media user. This type of lesson step does not require any input in the lesson center itself. Instead, it automatically detects when the required action has been performed. In that case, the lesson step is marked as complete.
4. System Overview

4.3 Managing Lesson Content

The principle of not needing to download any additional software other than a browser in order to use the lesson center also reflects the mentality of managing the lesson content itself. Administrators gain access to a special page (see Figure 4.4) where they can directly edit the individual lessons and their steps. Thanks to this simplistic setup, there is no need to edit the source code or connect to a database.

4.4 Social Media Network

As previously mentioned, the chosen target application for the prototype is a social media network. It supports many features known from the various famous social media networks, such as creating posts with images and videos, writing comments, and liking content. It is also possible to follow other users. All content posted by those users is directly displayed in the user’s home feed. Figure 4.5 shows what the social media network looks like.

To teach students about cybersecurity, some vulnerabilities have been implemented in the social media network on purpose. Currently, the following vulnerabilities are present:

**Broken Authentication** The authentication mechanism of the social media network has a particular flaw. Users are able to impersonate other users by simply telling the social media network that they are in fact that user. There is no security check in place. This is a very technical vulnerability. A more detailed explanation can be found in Section 5.3.2.
4.5 Proxy Service

As mentioned before, all interactions with the target application are sent through the lesson center. The interactions are then reported to the helper tool of the lesson center where the user can inspect the interaction thoroughly and use the gained information to progress in the lesson steps or generally increase their knowledge about the lesson topic through analysis. Additionally, all interactions are recorded in a database for further offline analysis.

In the example of the social media network, this means that all requests to the backend are sent through an HTTP proxy server. Refer to Section 5.1.4 for further information regarding the actual implementation details.

4.6 Moderation Tools

The learning platform should provide a safe learning environment. However, letting users create arbitrary content in a social media network might lead to unwanted text or media files being uploaded. To ensure the safety of the learning platform, a moderation tool is implemented in the social media network.
Figure 4.6: View of the content moderation page.

network. A mechanism is in place such that teachers will gain the required privileges to access the moderation tool through a supervisor page (see Figure 4.6) where they are informed about new content that is created by any user in real time. Reported content includes:

- New posts (text, images, and videos)
- New comments (text only)
- New stories (images and videos)
- Updated profile pictures (images only)

On the supervisor page, the reported content can be swiftly deleted at the click of a button. In contrast to the “keeping deleted content” vulnerability mentioned above, deleting content through the supervisor page actually removes all media files from the database, leaving no trace of the unwanted material.

For a technical explanation with more fine-grained details, refer to Section 5.3.3.

4.7 Research Aids

As previously mentioned, all interactions with the target application are stored in a database for further analysis. Additionally, each time a student submits an answer to a question in the lesson center, it is also saved – even if the submitted answer is wrong. Furthermore, the lesson progression is tracked. All of this data is saved with a corresponding timestamp, leading to a complete reconstruction of the timeline of the student’s actions. This
data may be useful in education to investigate where students struggle the most and which topics should be further strengthened by other teaching methods. It may also bring forth unclear instructions in the lesson content, thus helping to improve the learning platform overall.
This chapter focuses on an implementation-level description of the individual components. Refer to Chapter 4 for a high-level description of the system.

The implementation of the teaching platform consists of two main services: the lesson center and the social media network. Each service has a frontend and a backend, making it a total of four different servers. The subsequent chapters explore each component individually.

### 5.1 Lesson Center Backend

The lesson center backend holds all the educational content and task descriptions. It also provides a proxy service for the social media network to route the traffic through in order to enable automatic task completion tracking and to make reporting the traffic to the appropriate helper tool in the lesson center frontend possible. A way for administrators to update the existing lesson content and create new lesson content is also implemented in the system.

The backend is implemented as a Django application, additionally using the Django REST framework for improved support of REST API endpoints and the Django Channels extension to support WebSocket connections for real-time features. All REST endpoints support JSON as content type and are properly documented using OpenAPI specifications. A secure authentication cookie is used as the user authentication mechanism.

### 5.1.1 User Model

For improved customizability, Django’s default user class AbstractUser is subclassed. This is needed since for each user, the corresponding credentials

---

1See [https://app.swaggerhub.com/apis-docs/MT-GRUEBESV/helper](https://app.swaggerhub.com/apis-docs/MT-GRUEBESV/helper).
of the user’s social media accounts (own and target) are saved in the database. Additionally, the currently active lesson is saved. This allows improved progress tracking since the tool is designed such that progress can only be made in the currently active lesson. This is to prevent unexpected progress when casually browsing the social media network without actually solving a lesson.

The is_staff field of the AbstractUser model is used to distinguish normal users from administrators. Administrators have access to more API endpoints which allows them to make changes to the lessons and lesson steps without the need of making changes either directly in the database or in the code base.

5.1.2 User Signup

New users can be created through an email verification system. This ensures that all users have access to the email address with which they register and reduces the risk of being the victim of a spam signup campaign. When a new user is registered, two accounts are automatically created in the social media network – one for the user’s own enjoyment and one target account which the user should use as the target of any exploits and vulnerabilities that are introduced in the lessons.

5.1.3 Lesson Model

The lesson center contains a list of lessons. Each lesson has a title, as well as a list of lesson goals and a list of steps. Each step also has a title, as well as the step content. The content is represented as BBCode. This way, the available HTML tags can be easily customized and it can be ensured that certain HTML tags are excluded, such as JavaScript code which would lead to a potential XSS vulnerability.

As mentioned in Section 4.2, each lesson step has one of four types: information, question, open question, or task. The following paragraphs explore the implementation details of these types further.

Information There is no additional implementation detail for this type.

Question The steps of this type have a single correct answer, which might be a simple string or a regular expression. The answer can additionally be customized for every user by using string formatting. As an example, the configured answer might be “{target_id}”. When a user submits a guess, the solution is first translated to the ID of the user’s target social media

---

account before checking whether the submitted guess matches the string or regular expression. This way, each user might have a unique solution, making it hard for users to cheat by blindly using the solution of another user.

**Open Question** From the perspective of the database model, this type is equivalent to the information step as there is no configured solution in the open questions. Therefore, no additional database field is required. Regarding the user interaction, open questions allow submissions of arbitrary-length text. Since there is no configured solution, there is no checking involved during the submission process. The submission is simply recorded and the step is immediately marked as completed. The submitted solution can be updated by resubmitting a different solution. Submissions can be checked and graded offline if needed.

**Task** Tasks are steps which require a certain action in the social media network. The completion of the configured action is checked in the proxy on every request. Various aspects of the request and response can be configured to be conditions for the task completion. As for the question type, regular expressions and string formatting can be used to individualize the conditions to each user. As an example, consider the task that a user should login to their own social media account. The configuration might then contain the following information:

- Request URL is `/users/login`
- Request method is `POST`
- Response code is `200 OK`
- Response header contains `Set-Cookie: username={own_username}`

Once a request is encountered that fulfills all of these conditions, the task is marked as completed.

It is important to note that any steps must be completed in order. This guarantees that the flow of information taught to the user is respected and no unexpected lesson progress is encountered. If a user is for example asked to fill out a question and the next task is to login to their own account in the social media application, then the task is not marked as completed if the user logs in to their account before submitting a correct answer to the question.

### 5.1.4 Proxy Server

The lesson center provides a proxy service to forward any incoming requests to the backend of the social media network. This proxy service is useful in multiple ways:
5. Implementation

- Completion of tasks can be automatically checked
- Authentication: The proxy can make sure that the user only accesses their own account and their target account
- Requests can be reported to the lesson center frontend for further inspection by the user in the helper tool
- Requests can be logged for further offline analysis after the session

Before forwarding an incoming request, all hop-by-hop header fields are removed. Additionally, the authentication cookie of the lesson center is removed from the cookie header. This is to protect the user when they are inspecting the proxied request in the frontend while potentially other users are able to see the screen and thus intercept the authentication cookie value. Afterwards, a check is performed to make sure that the user actually has the right to access the requested account in the social media network. Recall that users only have access to their own account and a target account. All other accounts are inaccessible. If this check fails, the request is refused and the status code 403 (Forbidden) is returned with an appropriate error message. Otherwise, the request is forwarded to the backend of the social media network for further processing.

As soon as the response of the backend of the social media network is received, all data of the request as well as of the response are bundled into a single dictionary, along with some metadata such as timestamps of the request and response and the total duration of this interaction. The dictionary is then broadcast to the user’s “proxy” group on the channel layer.

Furthermore, it is checked if the user currently has a task as the next step in their active lesson. If this is the case, the data of the request and response is used to check the conditions of the task. If all task conditions are fulfilled, the task is marked as completed and a corresponding message is broadcast to the user’s “progress” group on the channel layer.

After this check, the hop-by-hop header fields are removed from the response before finally forwarding it back to the user. This concludes the proxying process of a single request.

If the backend of the social media network is not reachable for whatever reason, the status code 504 (Gateway Timeout) is returned with an appropriate error message.

5.1.5 WebSocket Connections

The channel layer of the Django Channels extension is used to push live data from the backend to any connected frontend. Each user has two groups in

---

5.2 Lesson Center Frontend

The channel layer: The “proxy” group for reporting proxied requests, and the “progress” group for reporting any lesson progress. An authentication mechanism is used to ensure that only the user themself is able to access their personal channel groups. Trying to connect to one of the channel groups via WebSocket without a valid authentication cookie will lead to the immediate closure of the connection with an appropriate closure reason (close code 1008 for policy violation⁴) prior to connecting to any channel group.

Opening a WebSocket connection to /ws/proxy-reporting adds the connection to the user’s “proxy” channel group. Whenever a proxy request is handled, the corresponding request and response data is broadcast to the user’s “proxy” channel group, sending the data directly to all connections that are members of this group. The open WebSocket connection is unidirectional, as only the server sends data to the listening party. Any incoming messages are ignored and discarded.

Opening a WebSocket connection to /ws/progress-reporting adds the connection to the user’s “progress” channel group. Whenever any progress is made in any lesson, a corresponding update message is broadcast to the user’s “progress” group which allows any group members to update their frontend views accordingly. In contrast to the “proxy” group, the WebSocket connection of the “progress” group is bidirectional. The client must report the currently active lesson to the server (or null if the user is not currently in any lesson) such that the server can appropriately track lesson progress.

5.2 Lesson Center Frontend

The frontend of the lesson center has been built as a React application in TypeScript. Client-side routing is done with the React Router. A custom API client is implemented to simplify the communication with the backend. The client uses Axios behind the scenes. Styling is done by using the Bootstrap framework, along with the React Bootstrap package.

All REST API endpoints of the backend are supported and used in the frontend. Users are able to create new accounts, login to existing accounts, and work through the available lessons. A functionality to request a password reset is also supported in case a user has forgotten the password to their account. Once logged in, the users are presented with an overview of the available lessons, along with the corresponding lesson goals and their individual lesson progress. Users are able to open any lesson of their choosing and look at the lesson description and tasks in more detail.

While in a lesson, users have access to the available helper tools. In the current implementation with the social media network being the target appli-

I
mplementation

5. Implementation

Figure 5.1: The header view of the inspection helper tool.

cation, the available helper tool is the request inspection tool. A WebSocket connection is used to receive up-to-date information about the requests performed by the user to the backend of the social media network. All requests are listed in a table. The requests can be further inspected by choosing one of three views: the header view, the content view, or the cookie view.

**Header View**  The header view shows all header fields of the corresponding request and response (see Figure 5.1). Additional pop-up options allow the user to gain further insight into the purpose of each individual header.

**Content View**  The content view shows the raw content which is contained in the request and the response (see Figure 5.2). It additionally supports simple formatting options. As an example, if the response content is of type `application/json`, then a toggle button is displayed which enables JSON pretty-printing. Images and videos are directly displayed.

**Cookie View**  The cookie view shows a nicely formatted overview of all cookies that were sent to the server, and an overview of the cookies which were received in the response, along with the complete cookie configuration and helpful pop-up options for further information (see Figure 5.3).

A second WebSocket connection is used for tracking the lesson progress in real time. Contrary to the other WebSocket, this one is used bidirectionally. The frontend sends the ID of the currently open lesson to the backend through this WebSocket channel – and in the other direction, the backend informs the frontend about any lesson progress that is achieved by the user. The currently open lesson ID is sent such that the progress tracking in the

![Figure 5.1: The header view of the inspection helper tool.](image-url)
Figure 5.2: The content view of the inspection helper tool.

Figure 5.3: The cookie view of the inspection helper tool.
backend can check for progress in the correct lesson. Without this check, users might accidentally progress in another lesson, which is of course to be avoided.

For staff users, an additional page unlocks: the lesson administration. On this page, all existing lessons can be edited, or new lessons can be created altogether.

5.3 Social Media Backend

The social media backend forms the main component of the learning platform’s target application as it houses the exploitable vulnerabilities. It allows the users to create their own posts and stories and let’s them interact with other users by providing a search functionality, a follow-system, a comment box, and a like functionality. These features were specifically chosen, as they are present in some of the well-known real social media networks. Additionally, a supervisor mode is implemented as a safeguard for efficient content moderation.

As for the backend of the lesson center, the backend of the social media network is written in Django, additionally using the Django REST API extension to add improved support for REST API endpoints. All endpoints work with JSON and are properly documented using OpenAPI specifications.\(^5\) Again, an authentication mechanism based on cookies is used. However, the mechanism is vulnerable to an attack, as will be explored further in Section 5.3.2. For real-time content updates in the supervisor mode (as introduced in Section 4.6), the Django Channels extension is used in order to add support for WebSocket connections.

5.3.1 User Signup

In contrast to the lesson center, the social media network does not allow the registration of arbitrary users. The only possible way of creating new users happens when signing up for the lesson center. A special “secret” signup endpoint exists which expects the necessary input data along with a shared token which is only known to the lesson center backend and the social media backend. Without this token, user registration is not possible.

The secret signup endpoint creates two accounts for every successful request and it returns the login credentials for these two accounts to the requesting client. This way, the lesson center can save the login credentials for each user and display it to them in the lessons. The endpoints to change the username or password have been disabled in order to not have out-of-date credentials in the lesson center.

\(^5\)See [https://app.swaggerhub.com/apis-docs/MT-GRUEBESV/social-media](https://app.swaggerhub.com/apis-docs/MT-GRUEBESV/social-media)
Of these two accounts, one account will be used as the user’s personal social media account, while the other is used as the target account for the attack which exploits the backend’s vulnerabilities.

### 5.3.2 Intentional Vulnerabilities

The social media network has an intentional authentication vulnerability which users should be able to exploit when working through the provided lessons. The vulnerability is located in the authentication scheme of the social media backend.

The authentication scheme of the social media backend looks for a username cookie in the cookie header field of the request and checks whether this is a valid username. This of course is trivially insecure, as a malicious user can provide an arbitrary username as the cookie value. This vulnerability was chosen due to its simplicity which helps to ease in new users to the world of cybersecurity with a straightforward example. Additionally, even if the vulnerability is not common anymore nowadays, it helps with understanding the historic progression that the security world went through over the decades.

Important to note is that there is a safeguard in place which prevents users from accessing any social media accounts which are not connected to their lesson center account. As all requests to the social media backend pass through the proxy service of the lesson center (refer to Section 5.1.4), the proxy will check that the social media account actually belongs to the lesson center user which makes the request.

The second intentional vulnerability is not of technical nature, but rather should strengthen the awareness of the users. Media files connected to posts and stories are actually not deleted when the post or story itself is deleted. By using the direct link to the media files, the files can still be accessed even after deletion. This should teach users that nothing they post on the internet is guaranteed to be deleted, even if they seemingly deleted the unwanted resource.

### 5.3.3 Content Moderation

To prevent inappropriate text or media files from being shared in the social media network, there is a content moderation system in place. Teachers will receive administrative privileges on their social media account which lets them connect to a WebSocket which reports all new user content (posts, comments, stories, and profile images). Using the administrative privileges, the teachers are then also able to access the deletion endpoints to remove the unwanted content immediately.
5. Implementation

Note that deleting posts and stories by using these administrative privileges actually deletes all connected media files from the file system of the backend, which ensures a safe environment for the users.

5.4 Social Media Frontend

As for the frontend of the lesson center, the frontend of the social media network has been built as a React application in TypeScript. Client-side routing is done with the React Router. A custom API client is implemented to simplify the communication with the backend. The client uses Axios behind the scenes. Styling is done by using the Bootstrap framework, along with the React Bootstrap package.

In the social media frontend, users are able to interact with the social media backend and perform the well-known actions from common social media networks. They may for example create new posts or stories, write comments for posts of other users, or search for other users and follow them.

Administrative users gain access to a separate page which is used for content moderation. When opening this page, a WebSocket connection is opened automatically to the social media backend (see Section 5.3.3) which reports any new content created by the users. Specifically, all new posts, stories, comments, and profile pictures are reported, including any potential media files. Administrative users are then able to immediately remove the content if they deem it to be inappropriate or otherwise unwanted.

5.4.1 Custom Console

To exploit the vulnerability of the social media application, the users will need to interact with the console at some point. Unfortunately, consoles are not very common in mobile or tablet browsers. To have a fully compatible setup, the social media frontend thus uses Eruda to show a mobile-friendly console directly on the website (see Figure 5.4). This has several advantages compared to using the standard console provided in many modern web browsers through their developer tools.

Firstly, it enables users with mobile devices and tablets to interact with the website through the console. As a lot of schools operate with a “bring your own device” policy, users bringing a tablet may otherwise not be able to use the platform to its full potential.

Secondly, while Eruda tries to mimic the real developer tools, it can be configured to only show certain widgets. Thus, to not overwhelm new users with too much information at once, it is configured to only show the console widget.
A downside of using Eruda is that it has some usability issues which has also been mentioned during testing sessions (see Chapter 7). These issues are unfortunately not solvable by the social media frontend, as the Eruda source code needs to be adapted separately.

5.5 Deployment

The platform has been deployed to a virtual machine within the infrastructure of ETH. All four servers are running in separate Docker containers; nginx is used as a reverse proxy to forward the incoming requests to the correct application. Each application is reachable on its own path, according to the following list. All paths are relative to the root of the deployed domain.\(^6\)

**Lesson Center Frontend** This is the entry point of the platform. It is reachable via /; nginx is configured to forward all incoming requests to port 3001, where the React application is served by a Node.js server in a Docker container. For security reasons, port 3001 is not reachable from an external source.

**Lesson Center Backend** It is reachable via /api/; nginx is configured to forward all incoming requests to port 8001, where the Django application is running in a Docker container. For security reasons, port 8001 is not reachable from an external source.

\(^6\)Deployed to [https://cyber.edtools.ethz.ch/](https://cyber.edtools.ethz.ch/).
5. Implementation

Social Media Frontend  It is reachable via /sm/; nginx is configured to forward all incoming requests to port 3000, where the React application is served by a Node.js server in a Docker container. For security reasons, port 3000 is not reachable from an external source.

Social Media Backend  The social media backend is not reachable directly from an external source. By going through the path /sm/api/, nginx is configured to forward all incoming requests to /proxy/ on port 8001, which is where the lesson center backend is running in a Docker container. The proxy server will then internally forward the request to the social media backend which is running in its own Docker container and listening on port 8000. For security reasons, port 8000 is not reachable from an external source.

For automated deployment of updated source code, a pipeline is set up in all repositories which will build the Docker image in the updated repository and push it to a private Docker registry. Afterwards, a webhook is used on the virtual machine to trigger it to pull the latest Docker image from the registry and update it.
A learning platform would be incomplete without properly designed lessons. As a result, the lesson content has been systematically assembled by considering the end goal of all lessons and methodically working backwards step by step, always keeping the required knowledge and learning goals of each step in mind. This way, the complete curriculum of the platform could be broken down into smaller learning chunks, starting with only little prior knowledge and slowly building up the knowledge until the end goal could be reached.

In the case of the social media network, the end goal is to exploit the authentication vulnerability introduced in Section 4.4. The systematic breakdown of the required knowledge looks as follows in this case:

- To exploit the vulnerability, the student needs to know how the authentication mechanism works
- To exploit the vulnerability, the student needs to know how to change the cookies of a website
- To change the cookies of a website, the student needs to execute some basic JavaScript commands\(^1\)
- To execute some basic JavaScript commands, the student needs to know how to interact with the console
- To execute some basic JavaScript commands, the student needs to know which commands to execute
- To change the cookies of a website, the student needs to know how cookies work

\(^1\)Other ways of changing the cookie value are of course also possible, such as directly changing the value through the web storage tab present in most modern browser developer tools. This approach was rejected, as introducing the developer tools could potentially scare away students due to their overwhelming amount of displayed information (see Section 5.4.1).
To know how cookies work, the student needs to know what cookies are.
To know what cookies are, the student needs to know what HTTP headers are.
To know what HTTP headers are, the student needs to know what an HTTP request is.
To know what an HTTP request is, the student needs to know what HTTP is.

These learning steps were categorized into two larger topics: General knowledge about HTTP and specific knowledge about HTTP cookies. For each topic, a separate lesson was created. A complementary introductory lesson was added with the goal of teaching how the lesson center and social media network interact, especially with the request inspection helper tool. The following sections go into further detail of each lesson and show the design choices which have been taken during the lesson writing process.

6.1 Introductory Lesson

For the introductory lesson, there is no required knowledge other than the ability to navigate around tabs and windows in a browser. The lesson is all about letting the student explore the learning platform. They receive their login credentials for their own social media account and they are encouraged to explore the social media network. The student learns that whenever they navigate around in the social media network, the resulting requests are reported to the lesson center and displayed in a list.

The student is instructed to inspect the reported requests. They learn to use the request inspection helper tool by simply clicking buttons and observing the displayed content. As a first quiz, the student is asked to find the ID of their own social media account amongst all the reported data. It is clear that the usual student will not have any relevant understanding of HTTP requests at this point. However, for the first quiz, this should not be an issue, as the ID is of a predefined recognizable form.

At the end of this introductory lesson, the student should have a clear understanding of how the learning platform is structured and how they can inspect a request through the helper tool in the lesson center.

6.2 General Knowledge Lesson about HTTP

Going into the second lesson, the student is expected to know how to navigate around in the learning platform. They should know how to use the request
6.3 Specific Lesson about HTTP Cookies

At the start of the lesson, the student should know about HTTP. They should be able to inspect arbitrary requests in the helper tool and understand what the displayed content means.

At the start of the lesson, the student is introduced to HTTP and the general concept of client/server communication. A “librarian” analogy is used to familiarize the student with the concept of sending requests in the internet with the goal of strengthening their understanding and building a foundation onto which other concepts can build.

In the next step, the student encounters the concept of HTTP request methods. The aforementioned analogy is used to convey the necessity and usage of request methods. In order to directly apply the knowledge, the user is asked to perform several actions in the social media network and observe the resulting request methods of those requests. They learn that different actions can be performed on the server by sending different request methods to the same API endpoint from the client.

Another introduced concept are the response status codes. Again, the initial analogy is used to show the usefulness of these status codes in an already familiar setting. The student learns about the most well-known status codes and their meaning, as well as the general status code families. Additional pop-ups in the request inspection helper tool provide further information about these codes if the student ever wished to know more about a specific code they encounter.

The final concept which is taught is the distinction between the header and the body of a request. Another analogy is used to strengthen the student’s understanding. As the concluding quiz of this lesson, the student is asked to find out which server program is used in the backend of the social media network. This involves further inspections of requests through the helper tool, as the students are encouraged to find the correct answer in a specific header field.

At the end of this lesson, the student should know about the general concept of client/server communication. They should also know about different HTTP request methods, as well as various response codes. Finally, they know about the distinction between the header and the body of a request or response. The student should also be able to use the request inspection helper tool to thoroughly investigate the header fields and the body content of an arbitrary request.

6.3 Specific Lesson about HTTP Cookies

At the start of this lesson, the student should know about HTTP. They should be able to inspect arbitrary requests in the helper tool and understand what
they are looking at – with the exception of the cookie inspection view, as cookies have not yet been introduced.

The lesson starts with an analogy that compares cookies to virtual ID cards which are received the first time when entering a building. Whenever the same building is entered again at a later point in time, the same virtual ID card is shown at the entrance. Next, the student learns how cookies are used in the case of user authentication, directly following the analogy. This is an essential step, as the social media network uses a vulnerable cookie authentication mechanism.

Furthermore, the student learns how the server sets cookies on the client. They also learn how the client then sends the cookie to the server in subsequent requests. In a series of interactive questions, the student must use the helper tool to inspect the headers of specific requests and responses. Additionally, the student is challenged by asking them an open question, namely whether they can spot the problem of the vulnerable authentication mechanism of the social media network.

As a next step, the JavaScript console is introduced. In another series of questions, the student is tasked with first reading and subsequently also writing cookie values through the console. At this point, the student has received all necessary information to successfully exploit the authentication vulnerability. Thus, the student receives the username to their target account in the social media network and they are tasked with breaking into this account. The task is fulfilled as soon as they create a new post through that target account.

Finally, the student is left with a concluding open question which asks for a potential fix to the authentication vulnerability. This question is not intended to receive a perfect answer from every student, as they are lacking critical information about security principles, but rather to get the student to reflect on what they have learned during the preceding lessons.

At the end of this lesson, the student should know about the concept of HTTP cookies, as well as why they are used. They should be able to read the current cookies from the console, as well as create new or update existing cookies. Finally, the student should know about the broken authentication mechanism of the social media network and be able to exploit it, making it possible for them to interact with the social media network through their own personal target account.
Chapter 7

Evaluation

For the evaluation process, an application was filed with the ethics commission of ETH Zurich for the implementation of a user study. The study was approved as application EK-2023-N-58. The user study would work in two phases: expert testing and in-class testing.

7.1 Expert Testing

As a first evaluation stage, the learning platform was tested by “experts”. These experts were chosen among the peers of the author with the condition of having prior knowledge of cybersecurity or computer science in general. In total, 10 such experts were available during the time period.

7.1.1 Method

Each expert was given a short introduction of the general idea behind the learning platform without giving away any of the features or lesson content. Each expert had the ability to work through the lessons until completion. After solving the lessons, the experts were invited to take part in a voluntary semi-structured interview. The guiding questions of the interview can be found in Appendix A. Each interview was recorded and transcribed for further analysis.

7.1.2 Results

Figure 7.1 shows the completion times for each expert per lesson, as well as the average, maximum, and minimum time required for each lesson. The average time required for all three lessons is just above 45 minutes. Anecdotal experience reports that there is a time factor of around 3–4 when comparing a student to an expert. This would mean that there is an expected runtime of the three lesson of around 3–4 classroom hours of 45 minutes each.
7. Evaluation

<table>
<thead>
<tr>
<th>Expert</th>
<th>Lesson 1</th>
<th>Lesson 2</th>
<th>Lesson 3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>7</td>
<td>12</td>
<td>26</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>14</td>
<td>47</td>
<td>75</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>10</td>
<td>17</td>
<td>33</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>11</td>
<td>19</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>16</td>
<td>24</td>
<td>48</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>10</td>
<td>22</td>
<td>42</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>19</td>
<td>30</td>
</tr>
<tr>
<td>8</td>
<td>11</td>
<td>12</td>
<td>26</td>
<td>49</td>
</tr>
<tr>
<td>9</td>
<td>13</td>
<td>17</td>
<td>33</td>
<td>63</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>20</td>
<td>26</td>
<td>57</td>
</tr>
<tr>
<td>Avg</td>
<td>9.2</td>
<td>12.2</td>
<td>24.5</td>
<td>45.9</td>
</tr>
<tr>
<td>Min</td>
<td>6</td>
<td>5</td>
<td>12</td>
<td>26</td>
</tr>
<tr>
<td>Max</td>
<td>14</td>
<td>20</td>
<td>47</td>
<td>75</td>
</tr>
</tbody>
</table>

Figure 7.1: The time required for each expert per lesson in minutes, as well as the average, minimum, and maximum time required per lesson in minutes.

The manual notes taken while the experts were working through the lessons, combined with the transcribed semi-structured interviews, resulted in a total of 89 individual points of feedback. For each point of feedback, the experts which mentioned it were noted. This way, it was possible to sort the feedback by the number of experts which gave the same pieces of feedback.

While most feedback is valuable, time to implement the feedback is unfortunately finite. Thus, prioritizing which feedback is implemented first is a tough but necessary choice. A process of concept coding [27] was applied, where feedback receives a higher priority the more it is mentioned. Several important improvements were implemented thanks to this method, including (but not limited to):

- Ability to use the helper tools while in a lesson. Previously, a separate page needed to be opened to view the helper tools, making it impossible to simultaneously read the instructions and use the helper tools, leading to heavy page swapping.

- Live update of the lesson progress. Previously, the refresh button at the top of the page needed to be clicked to reload the current lesson status. Now, the page is automatically updated whenever new lesson progress is reported.

- Improvement of lesson instructions. Some lesson instructions were not clear enough, leading to delays and frustration while solving the lessons.

- Allowing text-only and media-only posts in the social media network. Multiple testers were surprised that both text and media files were
needed to create a new post. This restriction has been eliminated, allowing for text-only and media-only posts, closer mimicking real social media networks.

Regarding the lesson content, 9 testers think that the introductory lesson achieves its goal of teaching the user about how to use the platform. A different set of 9 testers mentioned that the lesson about HTTP is correct from a technical perspective with one tester refusing to answer this question since they did not feel confident that their knowledge is good enough to judge this.

Regarding the content of the lesson about HTTP cookies, 7 testers mentioned that the vulnerability might be interesting for a student without prior knowledge in cybersecurity. Only 6 testers think that the concept of cookies is explained well enough, so there is definitely some room for improvement there. Additionally, just 5 testers are confident that the students will be able to make the logical leap to find potential fixes to the broken authentication mechanism, so there may need to be further hints or guidance required.

Generally, 8 testers always knew what they needed to do and what actions to perform. Similarly, 8 testers classified the complexity of the helper tool as being of adequate complexity.

7.2 In-Class Testing

The second stage of the approved user study is using the learning platform in a high-school classroom and testing how the students respond to the lessons and finding out what their learning experience is during the process. Unfortunately, due to timing constraints, running the platform in-class was not possible. Nonetheless, an outline for the testing process has been developed.

7.2.1 Planned Method

In the beginning, a pretest questionnaire is distributed to the students to evaluate the existing knowledge. Afterwards, the students are given access to the learning platform where they should work through the lessons on their own. As previously mentioned, the expected runtime of the current lesson content is 3–4 classroom hours with each taking 45 minutes. After this time, a posttest questionnaire is distributed to the students which tries to evaluate the learning experience. The session is concluded with a short debrief.

Afterwards, the teacher will be able to take part in a voluntary semi-structured interview which is recorded and transcribed for further analysis. The planned questions for the interview can be found in Appendix B.
Chapter 8

Conclusion

In this thesis, a novel learning platform for cybersecurity lessons was introduced. Using the platform does not require the installation of any additional software on the student’s machine, as it is fully browser-based. It is mainly targeted at the education of high-school students.

The platform has two main parts: the lesson center and the target application. The lesson center hosts all available lessons, as well as the helper tools which are used for working through the lessons. It also provides an administrative page which allows managing the lesson content directly in the browser. Conversely, the target application provides a playground for the student to directly implement what they have learned or to test their own ideas in a safe environment.

As a prototype implementation of the target application, a social media platform is provided which has most features of the well-known real social media networks. The student is able to follow other students, create their own content, and like and comment other students’ content. As a safeguard, a supervisor mode is implemented for teachers to keep an eye on the created content in real time with the ability to immediately and irrecoverably delete inappropriate or otherwise unwanted content. The social media network has two on-purpose vulnerabilities: broken authentication (technical vulnerability) and keeping deleted content (awareness vulnerability).

All interactions with the social media network are proxied through the lesson center. This allows for a request inspection tool to be implemented as helper tool. The tool allows the inspection of the header fields, the body content, and the sent cookies of each individual request.

Three lessons have been implemented so far: Introduction, HTTP, and Cookies. The introductory lesson shows the whole system and its interactions to the student. The second lesson teaches the student some important information about HTTP. The final lesson instructs the student about HTTP
8. Conclusion

cookies and how to manipulate them through the web console. There are also tasks with exploiting the authentication vulnerability to gain access to a target account. A safeguard is in place to prevent students from accessing the accounts of other students and their respective target accounts.

The learning platform has been evaluated in a first step by letting 10 experts work through the lesson content. The experts were able to solve all three lessons in an average time of around 45 minutes, resulting in an estimated runtime of 3–4 classroom hours when using these lessons in a real high-school class. Voluntary interviews were conducted with the experts, resulting in 89 individual points of feedback. Concept coding was applied to prioritize the feedback to be implemented in the available time.

8.1 Future Work

As the learning platform has been developed from scratch during the duration of this thesis, there is a lot of future work possible on the platform itself, as well as on the current target application and the evaluation process.

The platform itself can be extended by adding additional supported languages. Another extension could be the implementation of new target applications. As an example, the additional target application might be a gdb instance running on the server attached to a vulnerable program. The lesson center can be adapted with a new helper tool which shows the current state of the gdb instance and lets the student interact with it directly in the browser. New lessons can be created with the goal of teaching the student about computer memory and binary exploitation.

Regarding the lesson center, several improvements are possible. A hint system can be implemented which provides dynamic hints based on the individual progress of a student, as suggested by Vykopal, Švábenský, and Chang [6]. For an improved classroom experience, user accounts can be grouped into classes with the teacher being an administrator for this user group. The teacher could then be able to see the progress of each student in real time, learn which lesson steps the students struggle with the most, and potentially adapt their supplementary traditional teaching material accordingly. Furthermore, implementing a point system could lead to the addition of a real-time class scoreboard, creating a more competitive and gamified environment.

For the current target application, additional vulnerabilities can be implemented. The existing lessons can then be extended accordingly to support the teaching and exploitation of these new vulnerabilities. Interesting vulner-

\[ \text{gdb is a powerful debugging tool which allows the inspection of running programs, including their assembly instructions, their memory, and – if available – their source code.} \]
abilities might include SQL injection, cross-site scripting, or cross-site request forgery. Note however that these vulnerabilities present a drastic increase in the complexity of the exploits, which should be considered when creating the corresponding lessons and necessary helper tools.

In regards to the evaluation process, it would be incredibly beneficial to test the learning platform in a real high-school classroom. The user study approved by the ETH ethics commission already includes such a setup. The evaluation process has already been created, including an outline for the voluntary semi-structured interview with the teachers (refer to Appendix B).
Bibliography


Appendix A

Semi-Structured Interview Questions for Experts

Part 1  Design of the platform / User experience

1. Did you find the platform easy to use? Was there something confusing?
2. Was it clear how to interact with the lessons and perform the different steps?
3. Do you think the helper tools are of adequate complexity?

Part 2  Lesson content

Imagine you are a teenager aged 16–18 and you are using this platform semi-supervised (a teacher is available to answer, but you are supposed to go through the lessons by yourself).

1. How did you find the introductory lesson on how to use the platform?
2. How did you find the general knowledge lesson?
   a) Is there something incorrect or missing?
   b) Is some information overwhelming or irrelevant?
3. How did you find the authentication vulnerability lesson?
   a) Do you believe that the vulnerability is interesting?
   b) Do you think the concept of cookies is explained well enough?
   c) Do you believe the vulnerability is confusing?
   d) Do you think enough hints and help in logical thought are provided for the student to solve the problem in the last step?
Appendix B

Semi-Structured Interview Questions for Teachers

Part 1  Classroom Engagement

1. How do you think the experience was for the students?
2. Where did you see your students struggle the most?
3. What do you think your students learned the most?
4. Do you believe the learning platform with its social media network is an appropriate tool for connecting with the students?

Part 2  Platform in the real world

1. Which features did you prefer and why?
2. What is something that our platform is missing to be used in a classroom?
3. Why would you use / not use our platform for cybersecurity lessons?
4. Would you prefer more technical content, or more content related to awareness?
5. Do you think the platform is suitable for use by a single teacher?