

Sander Løken Berntsen
Erlend Einmo
Sondre Granerud
Tobias Moe

Pentesting Exercise Management Application

PEMA

Bachelor's project in IT-Operations and Information Security
Supervisor: Erik Hjelmås
May 2019

Sander Løken Berntsen
Erlend Einmo
Sondre Granerud
Tobias Moe

Pentesting Exercise Management Application

PEMA

Bachelor's project in IT-Operations and Information Security
Supervisor: Erik Hjelmås
May 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Information Security and Communication Technology



Norwegian University of
Science and Technology

Pentesting Exercise Management Application (PEMA)

Authors

Sander L. Berntsen

Erlend Einmo

Sondre Granerud

Tobias Moe

Bachelor in IT-Operations and Information Security

20 ECTS

Department of Information Security and Communication Technology

Norwegian University of Science and Technology,

20.05.2019

Supervisor

Erik Hjelmås

Sammendrag av Bacheloroppgaven

Tittel:	Pentesting Exercise Management Application (PEMA)
Dato:	20.05.2019
Deltakere:	Sander L. Berntsen Erlend Einmo Sondre Granerud Tobias Moe
Veiledere:	Erik Hjelmås
Oppdragsgiver:	Norwegian Cyber Range, NTNU
Kontaktperson:	Basel Katt, basel.katt@ntnu.no, 61135176
Nøkkelord:	Norway, English
Antall sider:	83
Antall vedlegg:	9
Tilgjengelighet:	Åpen

Sammendrag: Pentesting Exercise Management Application (PEMA) er en plattform som tillater brukere å planlegge, lage og utføre labber og oppgaver som omhandler temaer som hacking, malware-analyse og penetrasjons-testing. For denne bacheloroppgaven ble vi bedt om å utvikle plattformens grunnfunksjonalitet, samt tilrettelegge for videre utvikling. PEMA er bygget ved hjelp av mikrotjenester i docker containere, disse inkluderer selve nettsiden som er bygget på WordPress som kjører på en Apache webserver og databasen som håndteres av MariaDB.

Summary of Graduate Project

Title:	Pentesting Exercise Management Application (PEMA)
Date:	20.05.2019
Authors:	Sander L. Berntsen Erlend Einmo Sondre Granerud Tobias Moe
Supervisor:	Erik Hjelmås
Employer:	Norwegian Cyber Range, NTNU
Contact Person:	Basel Katt, basel.katt@ntnu.no, 61135176
Keywords:	Thesis, Latex, Template, IMT
Pages:	83
Attachments:	9
Availability:	Open

Abstract: The Pentesting Exercise Management Application (PEMA) is a platform in which one may coordinate the planning, creation and execution of labs and tasks on the themes of hacking, malware-analysis and penetration testing. In this bachelor thesis, we were tasked with developing the foundation upon which our employer may continue to expand the functionality of the platform. PEMA is built using docker containers, these include the site itself using WordPress running on an Apache web server and the database running MariaDB.

Preface

This bachelor thesis is written at NTNU under the Department of Information Security and Communication Technology.

We would like to thank:

- Erik Hjelmås for his guidance through the project.
- Danny Lopez and Basel Katt for providing good communication with, and facilitating the project.
- Eigil Obrestad for helping us understand what Openstack API parameters we needed in order to make the Openstack integration work
- Øivind Kolloen for helping us choose a fitting framework.

Contents

Preface	iii
Contents	iv
List of Figures	ix
List of Tables	x
Listings	xi
Abbreviations and Other Terminology	xii
1 Introduction	1
1.1 Norwegian Cyber Range	1
1.2 Background	2
1.3 Project Description	2
1.4 Boundaries	3
1.5 Project Goals	3
1.5.1 Learning Objectives	3
1.5.2 Impact Objectives	3
1.6 Target Audience	4
1.6.1 Application Audience	4
1.6.2 Report Audience	4
1.7 Academic Background	5
1.8 Project Organization	6
1.8.1 Administrative Roles	6
1.8.2 Functional Roles	6
1.8.3 Project Rules	6
1.8.4 Tools	7
1.9 Software Development Methodology	8
1.9.1 Schedule	8
1.10 The Report	9
1.10.1 Organization	9
1.10.2 Practical information	9
2 Requirements	10
2.1 Functional Requirements	10
2.1.1 Student functionality	10
2.1.2 Instructor functionality	10
2.1.3 Security requirements	11
2.2 Non-Functional Requirements	12
2.2.1 Accessibility	12

2.2.2	Availability	12
2.2.3	Maintainability	12
2.2.4	Reliability	12
2.2.5	Scalability	12
2.2.6	Security	12
2.2.7	Usability	12
2.3	Use Case	13
2.3.1	Use Case-Diagram	13
2.3.2	Misuse Cases	16
2.3.3	High-level Use Cases	16
2.4	Sequence Diagrams	18
3	Technical Design	19
3.1	Frameworks	19
3.1.1	Deciding on server side programming language	19
3.1.2	Deciding on frontend programming language	20
3.2	CMS and LMS	21
3.2.1	Moodle	21
3.2.2	WordPress	21
3.2.3	Conclusion	23
3.3	Web development	23
3.3.1	Theme	23
3.3.2	Page Templates	24
3.3.3	WPDB Class	24
3.3.4	Code Practices	25
3.3.5	Enqueuing Scripts	25
3.3.6	Plugins	25
3.3.7	AJAX	26
3.3.8	Hooks	27
3.3.9	jQuery	27
3.3.10	jQuery Plugins	27
3.4	Database	28
3.4.1	MySQL	28
3.4.2	MariaDB	28
3.4.3	Conclusion	28
3.5	Webserver	29
3.5.1	Nginx	29
3.5.2	Apache	29
3.5.3	Conclusion	29
3.6	Choosing an Operating System	30
3.6.1	Windows	30

3.6.2	Linux	30
3.6.3	Distributions	30
3.6.4	Conclusion	30
3.7	Containers or Virtual Machines	31
3.7.1	Virtual Machines	31
3.7.2	Containers	31
3.7.3	Scaling	32
3.7.4	Conclusion	32
3.8	PEMA Cooperation	33
3.8.1	PLED	33
3.8.2	DSL	33
4	Development Process	34
4.1	Development Tools	34
4.1.1	Coding Environment	34
4.1.2	Project Management	34
4.2	Planning for Future Work	37
5	Implementation	38
5.1	Setup of test environment	38
5.1.1	Webserver	38
5.1.2	Docker	38
5.2	Installation and configuration	38
5.2.1	Configuration	38
5.2.2	The source code	39
5.2.3	Docker swarm	39
5.3	Docker implementation	40
5.4	Web-application	41
5.4.1	Directory and File Structure	41
5.4.2	PEMA Roles	41
5.4.3	PEMA-Lab Hierarchy	42
5.4.4	The Loop	44
5.4.5	Page Templates	44
5.4.6	Custom Admin Pages	45
5.4.7	Enqueueing scripts and styles	47
5.4.8	Site Navigation	48
5.4.9	Usage of WPDB class	48
5.4.10	Custom hooks	50
5.4.11	jQuery	50
5.4.12	jQuery Plugin	51
5.5	Plugins	52
5.5.1	Openstack Plugin	52

5.5.2	Integration with PLED	53
6	Security	55
6.1	Wordpress	55
6.1.1	Prepare function	55
6.1.2	Output Encoding	55
6.1.3	Nonces	56
6.1.4	Security Plugins	57
6.1.5	Permissions	59
6.2	Backend	62
6.2.1	HTTPS	62
6.2.2	Docker swarm	62
7	Deployment	64
8	Testing and User Feedback	66
8.1	Purpose	66
8.2	Testing Scope	66
8.3	Static Analysis	66
8.4	Dynamic Testing	66
8.5	Fuzz Testing	66
8.6	Code Review	67
9	Discussion	68
9.1	Tools	68
9.1.1	Writing tools	68
9.1.2	Task Management	68
9.2	Results	68
9.2.1	Project Outcome	68
9.2.2	Unfulfilled Requirements	68
9.2.3	Alternative Solutions	69
9.2.4	For Future Implementation	69
9.2.5	Schedule	70
9.3	Complications	70
9.3.1	Static Analysis	70
9.3.2	Development methodology	71
9.3.3	HTTPS	71
9.3.4	Linking to pages in WordPress	72
9.3.5	jQuery data()	72
9.3.6	jQuery Plugins	72
9.3.7	WordPress plugins	73
9.4	Evaluation	73
9.4.1	Carrying Out The Project	73
9.4.2	Group Evaluation	74

10 Conclusion	76
10.1 Future Work	76
Bibliography	78
A Project Agreement	84
B Group Contract	88
C Permissions Tables	91
C.1 Custom Capabilities	91
C.2 WordPress Standard Capabilities	91
D Custom WordPress Hooks Table	92
D.1 Backend actions	92
D.2 Frontend actions	93
E OWASP ZAP Report	94
F Diagrams	102
F.1 ER-Diagram	102
F.2 PEMA Database Schema	105
F.3 Lab Deployment Sequence Diagram	107
F.4 Task Delivery Sequence Diagram	111
F.5 Misuse Case: Student - Query Injection	113
F.6 Misuse Case: Instructor - Query Injection	114
F.7 Misuse Case: Instructor - Unintentional Misuse	115
F.8 Misuse Case: Admin - Unintentional Misuse	116
F.9 Misuse Case: Login	117
F.10 PEMA Directory and File-Structure	118
F.11 Gantt scheme - Start	119
F.12 Gantt scheme - End	120
G Pre-project	121
H Meeting Logs	133
H.1 May meeting logs	133
H.2 April meeting logs	138
H.3 March meeting logs	141
H.4 February meeting logs	150
H.5 January meeting logs	160
I Time Tracking Output	171

List of Figures

1	NCR Stakeholders	1
2	PEMA Stakeholders	4
3	Admin use-case diagram.	13
4	Instructor use-case diagram.	14
5	Student use-case diagram.	15
6	Latest WordPress Vulnerabilities.	22
7	Latest WordPress Plugin Vulnerabilites.	22
8	Latest WordPress Plugin Vulnerabilites [1].	23
9	WordPress Admin-dashboard menus per role	42
10	PEMA-Object hierarchy.	43
11	WordPress admin-dashboard default sidebar.	46
12	Preview of PEMA-navigation.	48
13	Flatpickr date and time example	51
14	jQuery-confirm example	53
15	Example of WordFence sending an email when a user logs in on our test environment	57
16	Example of typing being locked out	58
17	Visualization of the software relationship.	64
18	View of the master thread in the fuzzing process after 48 days	67
19	Student assistant use-case diagram.	69
20	Appendix: Student query injection misuse case.	113
21	Appendix: Instructor query injection misuse case.	114
22	Appendix: Instructor unintentional misuse case.	115
23	Appendix: Administrator unintentional misuse case.	116
24	Appendix: Login misuse case.	117
25	Appendix: PEMA Directory and File-Structure.	118
26	Appendix: Initial Gantt Schema.	119
27	Appendix: Gantt schema showing actual project timeline.	120

List of Tables

1	Table of Tools	7
2	Table of custom WordPress permissions/capabilities	91
3	Table of standard WordPress permissions/capabilities used in PEMA	91
4	Appendix: Table of custom WordPress hooks.	92
5	Appendix: Table of custom WordPress hooks.	93

Listings

3.1	Vue.js binding objects to HTML elements	20
3.2	Gathering information about a lab using the \$wpdb class.	24
3.3	Enqueueing scripts in WordPress	25
3.4	Creating objects for jQuery	26
3.5	Docker scaling command	32
5.1	Deploying the service into the docker swarm.	38
5.2	Initializing the docker swarm.	39
5.3	"The loop" Searching for and displaying posts.	44
5.4	WordPress page creation example.	44
5.5	Adding menu and sub-menu entry for the admin-dashboard.	45
5.6	Rendering a page after menu-call.	47
5.7	Example of registering all .css files	47
5.8	Example of enqueueing scripts and stylesheets on a specific page	47
5.9	Example of inserting a lab	49
5.10	Example of using the update function	49
5.11	Example of creating a custom hook.	50
5.12	Example of using the created custom hook.	50
5.13	JavaScript code example. Using animation to reveal object.	50
5.14	Flatpickr implementation	52
5.15	Scheduling using WP-Cron	53
5.16	Example of using the update function	54
6.1	Example of WordPress prepared statement.	55
6.2	Utilizing the wp_localize_script function.	56
6.3	Getting the nonce in jQuery	56
6.4	Adding a role to WordPress.	59
6.5	Adding a role to WordPress.	59
6.6	Function removing unused default WordPress roles.	60
6.7	Repopulates the default WordPress roles.	60
6.8	Function removing instructors ability to promote to administrator or instructor.	61
9.1	Example of getting data attribute in an select element	72

Abbreviations and Other Terminology

Abbreviations

AJAX Asynchronous JavaScript And XML
API Application Programming Interface
CLI Command-Line Interface
CMS Content Management System
CRUD Create Read Update Delete
CSRF Cross-site Request Forgery
CSS Cascading Style Sheets (Language)
CURL Client URL
GNU GPL GNU's not Unix General Purpose License
GUI Graphical User Interface
HTML Hypertext Markup Language
JSON JavaScript Object Notation
LAMP Linux, Apache, MySQL/MariaDB, PHP
LMS Learning Management System
MVC Model-View-Controller
MTBF Mean Time Between Failure
NCR Norwegian Cyber Range
OS Operating System
PHP PHP: Hypertext Preprocessor
PoC Proof of Concept
RHEL Red Hat Enterprise Linux
URL Uniform Resource Locator
VM Virtual Machine
WP WordPress
XAMPP XAMPP Apache MariaDB PHP Perl
XML eXtensible Markup Language

Terminology

Blue team

Blue team is what one would call a team of security researchers working to defend systems and improve security.

Cyber Range

Cyber ranges are interactive, simulated representations of an organization's local network, system, tools, and applications that are connected to a simulated Internet level environment [2].

Pentest(ing)

A penetration test, colloquially known as a pen test, is an authorized simulated cyberattack on a computer system, performed to evaluate the security of the system.

Phabricator

Phabricator is a project-sharing platform, in this case hosted by NCR [3].

Proof of Concept

Evidence which demonstrates that a design concept, etc. is feasible [4].

Unix time

Unix time, also known as Unix epoch, is the number of seconds elapsed since January 1, 1970 UTC/GMT [5].

1 Introduction

1.1 Norwegian Cyber Range

The Norwegian Cyber Range (NCR) is a training, testing and cyber security arena under development by a team of researchers at the Institute for Information Security of the Norwegian University of Science and Technology. Its goal is to address all sectors of society related to Cyber Security. The NCR strives for better and more realistic environments for education, training, testing and research in the cyber domain. This is to make the blue teams more qualified in regards to addressing current and future threats. These threats can be unintended consequences of our technological advances, our connectivity needs and the criminal actions taken by malicious agents in the cyber field. The NCR as an open arena will help proactively, as well as reactively to prevent, mitigate and correct many of the issues that our future society will bring. The stakeholder structure of the NCR can be found in figure 1.

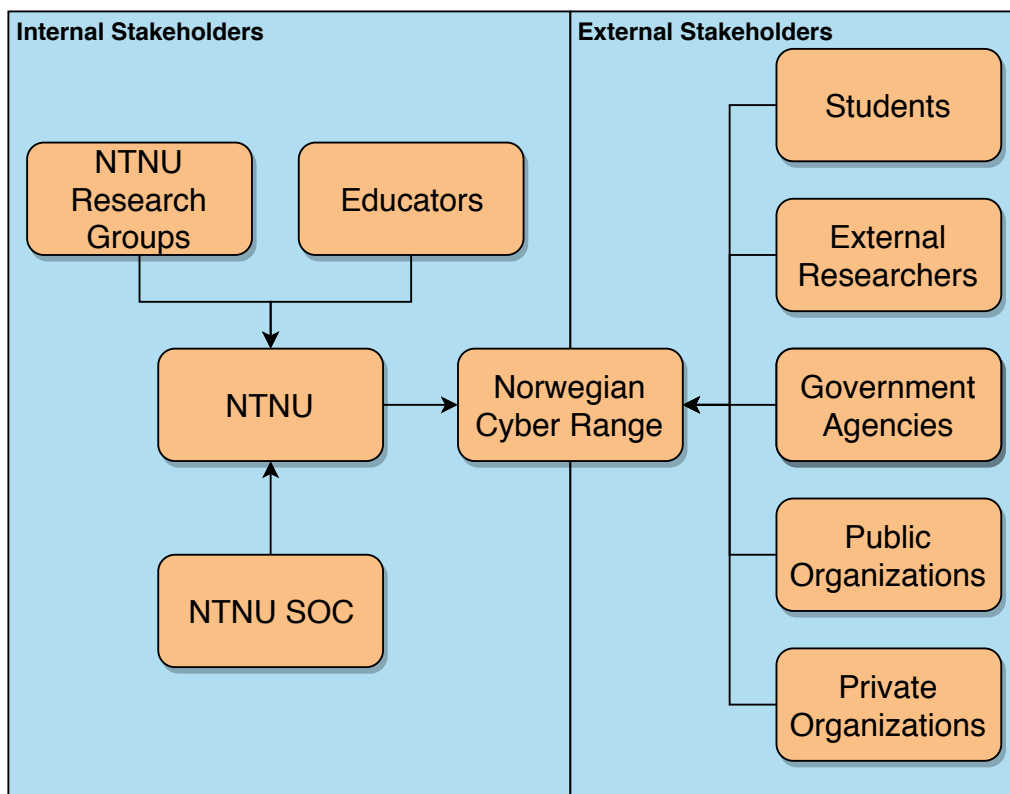


Figure 1: NCR Stakeholders

1.2 Background

The NCR sees a clear need for a platform that can create, deploy and manage virtual penetration testing scenarios. This need comes from experiences obtained when trying to conduct the course Incident Response, Ethical Hacking and Forensics(IMT3004) in 2018. During the course it became clear that there is a need for a central platform that can be used to manage the IT infrastructure of the course, provide quick feedback for the students and give the lecturer an overview over the students' progress. The NCR wants to have a bachelor group develop a proof of concept that can address these needs. The result of this bachelor thesis will lay the foundation for the further development of a fully integrated penetration exercise management application.

1.3 Project Description

The Pentesting Exercise Management Application Platform, henceforth called PEMAP, is an initiative intended to be a modular, scalable and virtualization agnostic platform that facilitates the deployment of virtual scenarios for cyber security education for research purposes. This platform consists of a database with existing vulnerabilities and related software called PLED, a Domain Specific Language to deploy and manage virtual instances called DSL, and a web application called PEMA.

The Penetration Exercise Management Application(PEMA) will help educators set up and manage hacking labs and CTFs and provides students with tasks and goals that they must complete in order to finish the assigned lab. The application is to be developed in the context of the course "Ethical Hacking and Penetration Testing" at NTNU and aims to complement the current infrastructure for managing ethical hacking projects.

An educator can use the application to define Labs and the topology of the lab. In the lab the educator will also define the goals and tasks the student must complete as well as steps and corresponding hints for those steps. There will also be set individual or group difficulty levels in order to match different levels of competence between the students. Students can use the application as an assistant tool that explains the lab goal and topology, the tasks that need to be completed and the steps that need to be performed. Finally, students can submit solutions and flags for different task. The Educator should be able to monitor the individual student's lab progress so that they can better identify achievement goals students need to accomplish. PEMA should also provide a set of modules, that can be activated or deactivated by an Educator. These modules contain various functionalities, enhancing the experience of the labs. The students will also be able to monitor their own progress in order to give them a clear overview of the goals they have achieved and the importance of the remaining uncompleted tasks.

The PEMA application should be easily deployable, handle multiple simultaneous users and it should follow best practice when it comes to security standards.

1.4 Boundaries

This bachelor thesis is limited to only the web interface that the students and educators will use and its corresponding IT infrastructure. The thesis does not include compiling a database of existing vulnerabilities and corresponding example software. That is the PLED part of the project.

The thesis also does not include the Domain Specific Language that will be used to deploy the lab environments in the final revision of this project. This is its own master thesis by Mihkal Dunfjeld.

PEMA will however include a proof of concept for deploying the infrastructure through a web portal, and a working concept for querying PLED for the information it will provide. These features will be implemented as plugins to showcase how the DSL and PLED will be easily integrated into the PEMA platform later down the line, when the DSL and PLED are at a stage when they can be properly implemented.

This means that PEMA will include the web portal for the teaching environment, e.g., the frontend and backend. The project description specifies that the web portal should contain a diverse variety of modules which enhance the functionality of the web portal. However due to the limited time frame it was decided in accordance with the project leaders that that the group was not going to devote any attention to the development of the modules that are not mentioned in the last paragraph.

1.5 Project Goals

1.5.1 Learning Objectives

- Understand the differences in frameworks.
- Coordination between two other theses in a complex application architecture.
- The use of Kanban in practice.
- Create the starting point for an ongoing project that extends beyond our timeframe.
- Learn to develop software in a proper environment with actual employers that have expectations.

1.5.2 Impact Objectives

- Create a webdesign that the employer is satisfied with.
- Create a scaling backend that can be easily deployed.
- Make it easy for other developers to further work on our code base.

1.6 Target Audience

1.6.1 Application Audience

PEMA is envisioned as a platform that could easily be integrated into existing cybersecurity curricula of penetration testing and ethical hacking courses in order to increase the quality of the practical sessions of such courses. Students, educators and other event-organizers may take interest in this project, as the goal of the platform is to simplify the execution of said events, an overview of the project stakeholders can be found in figure 2. By simplifying this process, we hope that the learning-experience and -payout is improved, by allowing instructors and students to focus more on what is being taught, and solving the task at hand.

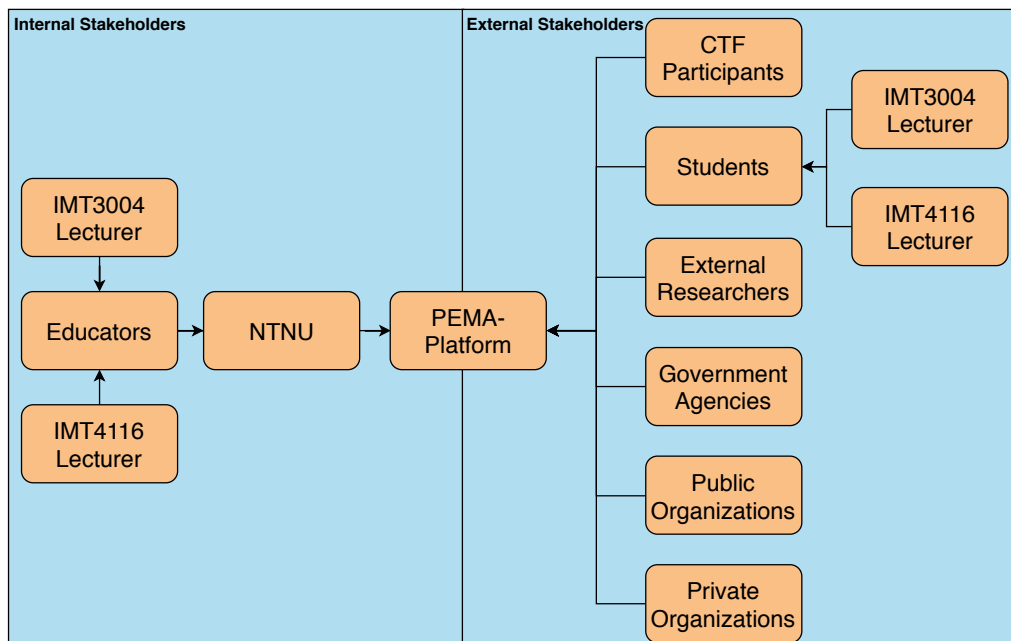


Figure 2: PEMA Stakeholders

1.6.2 Report Audience

This bachelor thesis is intended for the NCR in addition to the instructor of the ethical hacking part of the course IMT3004, and is meant to highlight the feasibility of the platform as it has been outlined, in addition to whoever will continue the work which we have begun. The report aims to highlight the goal of the platform as a whole, including our cooperation with, and the platforms integration with the domain specific language which will handle the physical deployment of labs, and the bachelor group PLED who is developing the technical information-gathering needed to deploy labs.

1.7 Academic Background

Sondre, Tobias, Sander and Erlend are all third year students of the study program "Bachelor of IT operations and information security". Throughout our almost 3 years, we have had subjects such as programming, networking, databases, datamodelling and infrastructure as code. Erlend, Sander and Tobias have in addition had Software Security, while Sondre had Cloud Technologies at that same time. In addition, Sander has some background from participating in bug-bounty programs.

This bachelor thesis will revolve heavily around subjects which we have had training on, such as programming, infrastructure design and operations, and information security, however it will also be challenging, as web development is an area in which we have lacking experience. In addition, the challenge of performing the technical design of an entire platform such as this, following the specifications given to us, is a challenge which we have not yet been presented with.

1.8 Project Organization

1.8.1 Administrative Roles

Group leader

Tobias Moe

Tobias Moe will act as the project leader, whose responsibility it is to follow up on meeting hours, appointments, ensuring tasks are split evenly between the group members and be the contact person to the project owner.

Secretary (Logging)

Erlend Einmo

Responsible for writing a log of the meetings held with third parties. In the case that he is unable to fulfill his task, he is responsible for handing the task over to another group member on a per-meeting basis.

Supervisor

Erik Hjelmås

Erik is an Associate Professor at NTNU, he is our supervisor through this project, giving us guidance and helping with some decisions along the way.

Employers

Danny Lopez

Danny is a research assistant and he will be our main contact person if we have any questions regarding the project.

Basel Katt

Basel is an associate professor at NTNU in the Department of Information Security and Communication Technology.

1.8.2 Functional Roles

Programmers

Erlend Einmo & Tobias Moe The Programmers are responsible for developing the WordPress front- and back-end and all its functionality, as well as plugins used with WordPress.

Infrastructure Designer

Sondre Granerud The Infrastructure Designer is responsible for designing the layout and relationship between the virtual machines and docker solutions, in addition to implementing HTTPS.

Tester

Sander L. Berntsen The Tester is responsible for testing the PEMA application and its functionalities. He is also responsible for conducting extensive code reviews to ensure the quality of the PEMA code.

1.8.3 Project Rules

As a part of the project, we have created a group contract with rules which every group member has to follow. These can be found in appendix [B](#).

1.8.4 Tools

The group agrees on what tools we are going to use during the project and thesis. These decisions are based on previous experience, as we have worked together as a group several times previously. The tools are chosen to keep our development consistent, and to allow for effortless collaboration, with the ability to see what the other members of the group are working on at any time. A list of all the tools used can be seen in table 1.

Table 1: Table of Tools

Name	Type	Usage
Overleaf	Collaborative \LaTeX writing	Report writing
Toggl	Online tool used to track time	Time tracking
Trello	Used to visualize Kanban boards	Tasklist
Google Drive	Host documents	Storage
Google Docs	Used to write notes	Noting
Draw.io	Used to create diagrams	Diagrams
TeamGantt	Used to create gantt-schemas	Diagrams/Schemas
The Box	Used for sharing documents	Sharing
TeamSpeak 3	Voice communication platform	Communication
Facebook/Messenger	Text communication platform	Communication
Discord	Voice/text communication platform	Communication
Wordpress	CMS framework for basic functionality	Frontend
Docker	Software containerization platform	Backend
Openstack/SkyHiGh	Cloud platform used to host the backend	Backend

The choices of tools are further discussed in chapter 4, subsection 4.1.2.

1.9 Software Development Methodology

Kanban was chosen as the software development strategy for this project. Kanban is an open and flexible model that fits how the group members prefer to work on projects as well as it gives a good overview of task that are upcoming, in progress and completed. With Kanban we can easily change our focus from one thing to another. Kanban has fewer rules and is more lightweight than other methodologies, this also makes it more challenging as we need to be able to handle the lack of rules. However the group has well established and close communications in place to handle problems as they come up. Kanban also gives a good metric on development progress which makes it easier to make decisions based on priority and how the project is progressing.

The formula for Little's Law[6] is commonly expressed as $L = \lambda W$, where "L" is the average number of customers, " λ " is the average arrival rate and "W" is the average time in the system. By redefining the terms as follows we can use Little's Law to our benefit:

- L becomes Work In Progress (WIP)
- λ becomes Delivery Rate (DR)
- W becomes Lead Time (LT)

This gives us the formula:

$$WIP = DR * LT$$

or

$$DR = WIP / LT$$

or

$$LT = WIP / DR.$$

While using the Trello boards to visualize our workflow, one can use the redefined formula to estimate how long each board will take to complete by using the following example.

We have a board with 10 WIP cards on it. We know we can complete 2 cards in a day, which gives us a DR of 2. By using the formula we can estimate that it will take 5 days to complete the board.

If one were to increase the amount of WIP cards to 20 then the LT would increase to 10, meaning it would double the time to complete the board. In order to cope with the increase in WIP cards then we would need to find a way to change the DR to 4, and this could prove difficult as it can be hard to "push" more work onto people.

By using this formula to our benefit we can efficiently plan ahead.

1.9.1 Schedule

The original work plan in the form of a gantt-scheme can be found in appendix F11. This schema was created using the online-tool TeamGantt, facilitating collaborative work on the schema in the planning stages. Through the duration of the project PEMAs requirements and description has been altered by our employer. These changes are discussed in chapter 9.2.5.

1.10 The Report

In this section the organization of the report is discussed, in addition to some practical information for the reader.

1.10.1 Organization

1. Introduction - General introduction. Descriptions of employer, project background and description, boundaries, and audience.
2. Requirements - Functional and non-functional requirements, and use case exploration.
3. Technical Design - Frameworks, and technology-discussion
4. Development process - How the group has worked.
5. Implementation - How the pieces fit together.
6. Security - Application and backend-security.
7. Deployment - How the platform is deployed.
8. Testing and User Feedback - Testing methodology.
9. Discussion - General discussion of the project-progress.
10. Conclusion - Report summary.

1.10.2 Practical information

This report is written using the online tool Overleaf, which is a platform that allows people to collaborate on writing documents using \LaTeX . The report contains many references to other chapters and figures, these are marked in blue, and once clicked will navigate the reader to said chapter or figure. In certain areas there are code examples, these are written with code-highlighting enabled for whatever language said code is written in.

2 Requirements

2.1 Functional Requirements

In this chapter the functional requirements of the PEMA application are described. The functional requirements have been divided into the following three categories: Student Functionality, Instructor Functionality and Security Requirements.

2.1.1 Student functionality

- View only the topics a student is enrolled in
- See amount of points the student currently has
 - See changes in points (e.g., the reason why points are increasing/decreasing)
- Hints
 - See cost of hints
 - Pay for hint (i.e., use hint)
- Lab
 - See lab metadata (e.g., title, description, start date, etc..)
 - See the tasks that are in the lab
 - Reboot machines associated to the lab
- Tasks/Challenges
 - See time remaining
 - See which tasks have been successfully submitted
 - See metadata about task (e.g., name, description, topic, etc..)
 - Submit an answer to tasks (Based on the specified method in the task)
 - View which tasks the student has to perform and the related topic
- User Profile
 - Every student should have a user profile
 - Should be able to edit user settings
 - Should be able to hide most information from other students (e.g., full name)
- Should not be able to view sensitive information about instructor or admin

2.1.2 Instructor functionality

- Should be able to add both NTNU and external participants
- Labs
 - Should be able to View/Edit/Duplicate/Create/Remove labs

- Enable or Disable a created lab
- Each lab can consist of multiple tasks
- Specify groups that will take part in the lab
- Specify start/end date on lab
- Specify title/description of lab
- Specify total amount of points to earn on lab completion
- Manage groups for lab after lab creation
- Topics
 - Should be able to View/Edit/Create/Remove topics
- Answer methods
 - Should be able to Create/Remove answer methods
- Groups
 - Create groups
 - Add users to already existing groups
- Tasks
 - Should be able to View/Edit/Duplicate/Create/Remove tasks
 - Specify text to display for students after task completion
 - Specify how many points to earn on task
 - Specify a number of hints for the task
 - Specify cost of a hint

2.1.3 Security requirements

- All sensitive data being communicated between browser and webserver shall be protected against disclosure (HTTPS)
- PEMA shall prevent unauthorized access to remote users
- Authentication credentials should be protected against unauthorized access
- Instructors should not be able to modify passwords for students
- PEMA should verify the identity of the user before allowing him/her to update his/her user info
- PEMA should protect itself against malicious input in any of its input fields which are accessible to students and instructors
- PEMA should protect itself against erroneous input in any of its input fields which are accessible to instructors by validating entered data
- All authentication attempts should be logged
- Unauthorized access to the answers by the students shall be prevented
- Only instructors shall be able to edit, create, remove
 - Labs
 - Tasks
 - Groups
 - Users

2.2 Non-Functional Requirements

2.2.1 Accessibility

The PEMA application shall not intentionally hinder any person with vision or color impairment from using its features.

2.2.2 Availability

The system shall be available to the users for approximately 98% of a 24-hour cycle when it is deployed. This means that the applications should have a 98% availability when there is a lab running or when an educator is preparing a lab. When the lab is not deployed or when the instructor does not need to prepare the labs, or when the instructor does not run a CTF, PEMA can be down. Maintenance and updating of the application shall happen swiftly and with no more than 30 minutes of service interruption per 24-hour cycle. The application must lay the grounds for Continuous Delivery of its services.

2.2.3 Maintainability

The application shall follow established coding practices for the chosen tools and frameworks in order to make takeover and subsequent maintenance possible with minimal effort. Further expansion of the application must be possible, and the framework chosen has to allow this without in depth knowledge of the code developed during this project timeframe.

2.2.4 Reliability

The PEMA application should have a Mean Time Between Failure(MTBF) of approx 168 hours. This means that there should be at least a week between system failures.

2.2.5 Scalability

The infrastructure of the application should scale to fit the expected number of users of the system. The first iteration should handle 55 users.

2.2.6 Security

PEMA handles information about its users, and in the case of students or participants, their progress on a given task. The security on the platform is handled on many different levels, each of which serving its own purpose. In the code, the database is secured through handling all user input, decreasing the likelihood of for example successful SQL-injection. We will further be discussing our security-implementations in its own chapter.

2.2.7 Usability

PEMA should be usable by the project owners without much need for consulting the documentation. Since this is a proof of concept there is not much need for high usability.

2.3 Use Case

2.3.1 Use Case-Diagram

In this section, we will go through the use cases for PEMA. This showcases how we expect the different user roles will interact with the PEMA system.

Administrator use case diagram

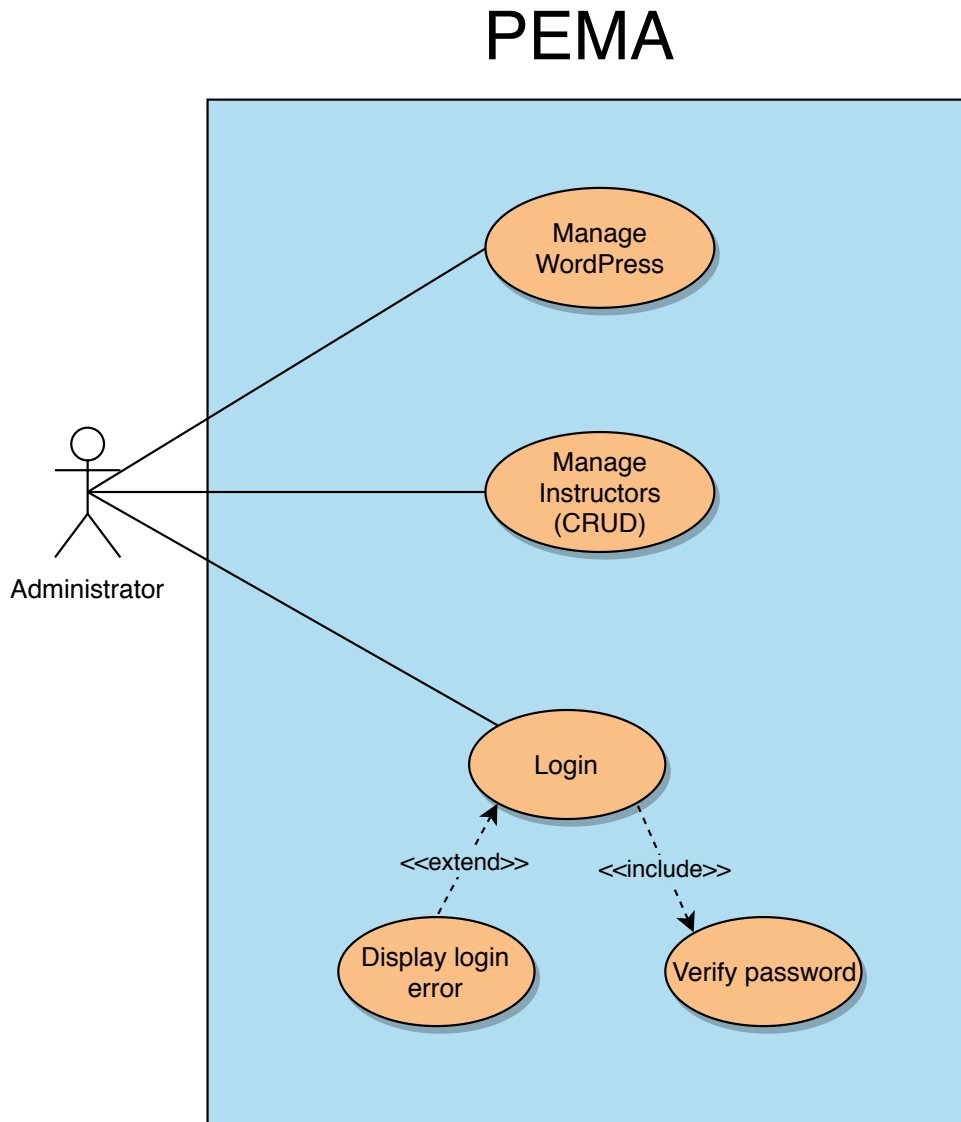


Figure 3: Admin use-case diagram.

Figure 3 shows a use case diagram for the administrator role. For the administrators, typical administrator actions are expected, like being able to correct mistakes, general system maintenance and adjusting, and add instructors. The administrator also has the possibility to do everything the instructors can do, like creating and managing labs and tasks etc.

Instructor use case diagram

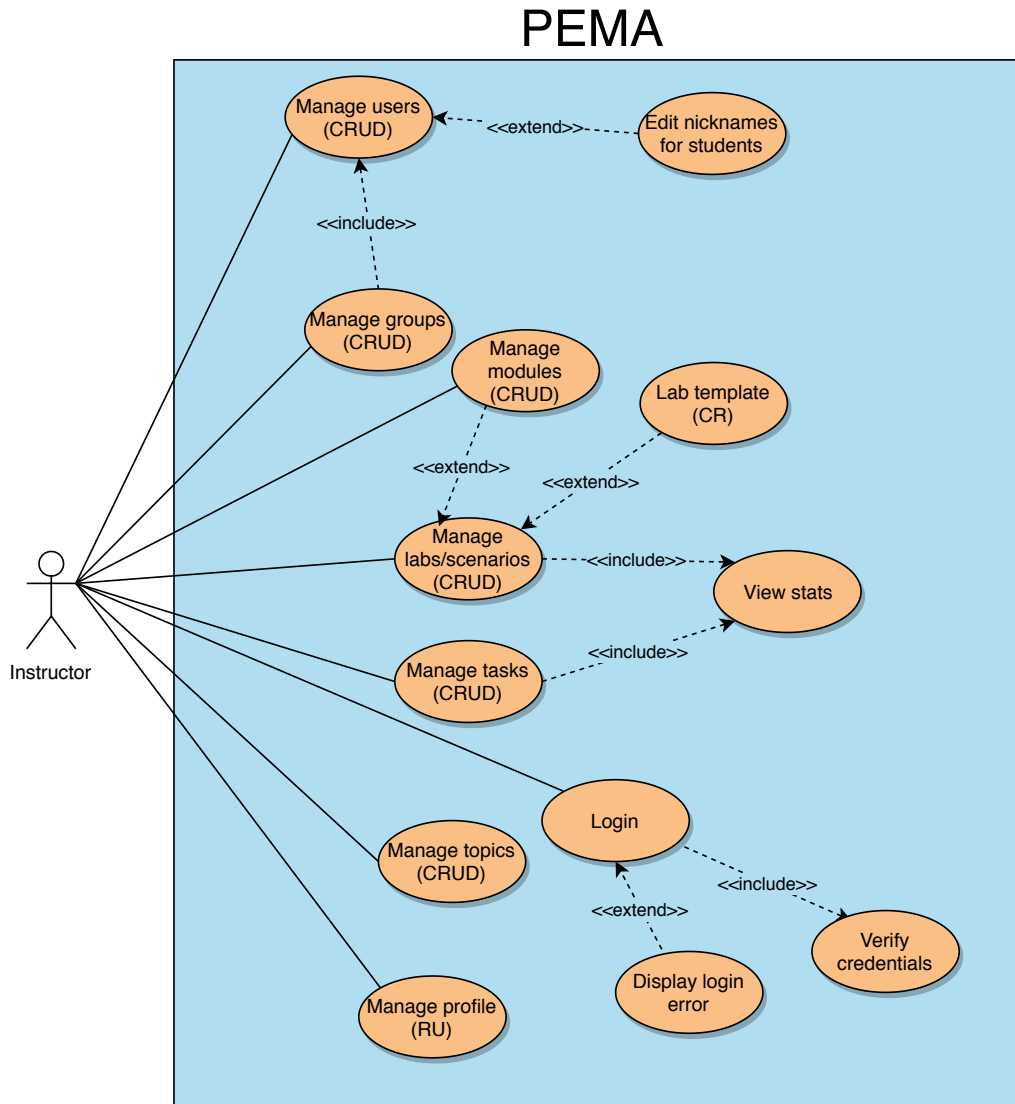


Figure 4: Instructor use-case diagram.

Figure 4 shows a use case diagram for the instructor role. The instructors should be able to manage everything required to make labs available to students with all the required functionality. This, as well as some limited user-management to add and manage student users are the actions expected from instructors. The user can also do what students can do.

Student use case diagram

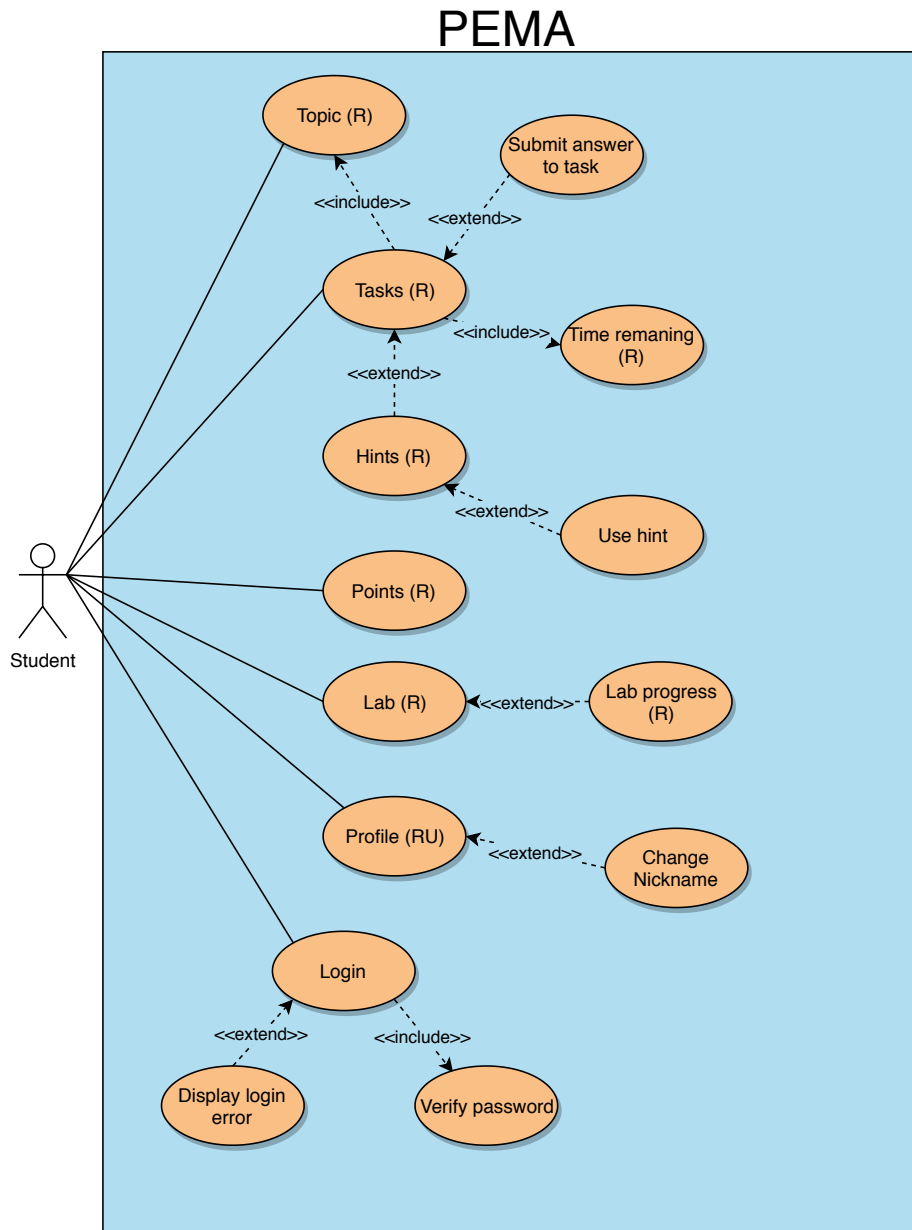


Figure 5: Student use-case diagram.

Figure 5 shows the use case diagram for the student role. Student actions include accessing the student-area of the platform, viewing labs and all their contents, as well as being able to submit answers to the tasks of the labs.

2.3.2 Misuse Cases

There are misuse cases created for the PEMA platform as well. The misuse cases show potential misuse of the platform, and some mitigations put in place to help combat these. These can be found in appendix F, figures 20 to 24.

2.3.3 High-level Use Cases

Use case: Users should be able to login

Actor: Instructor, Admin, and Student

Goal: Authenticate users before accessing the web page

Description: Every user should be authenticated before accessing any web page. This means that they should not be able to view the web page while they're not authenticated.

Use case: CRUD Lab

Actors: Instructor, Admin

Goal: Should be able to CRUD a Lab

Description: Every Lab should be created, updated and deleted. They should also be able to have an overview of all the labs currently deployed.

Use case: CRUD Task

Actors: Instructor, Admin

Goal: Should be able to CRUD a Task

Description: Every Task should be created, updated and deleted. They should also be able to have an overview of all the tasks currently deployed.

Use case: CRUD Topic

Actors: Instructor, Admin

Goal: Should be able to CRUD a Topic

Description: Every Topic should be created, updated, and deleted. They should also be able to have an overview of all topics.

Use case: CRUD Groups

Actors: Instructor, Admin

Goal: Should be able to CRUD a group

Description: Every group should be created, updated, and deleted.

Use case: Create Hints

Actors: Instructor, Admin

Goal: Should be able to CRUD hints for a specific task

Description: Every task could have multiple hints as specified by the actor. Every hint should have a cost value, and a student should be deducted in points when using a hint.

Use case: CRUD Profile

Actors: Instructor, Admin, and Student

Goal: Should be able to CRUD their own profile

Description: Every user should have a user profile where they can update specific information about themselves.

Use case: CRUD Users

Actors: Instructor, Admin

Goal: Should be able to CRUD all the users on the website

Description: A student can be removed from the site at any point by the actors. The actors should also be able to update specific information from a student. Some limitations is that an Instructor cannot remove another instructor or an admin from the website.

Use case: CRUD Answer Methods

Actors: Instructor, Admin

Goal: Should be able to CRUD an answer method

Description: An answer method is used on task template creation, meaning the actor should be able to create, updated and delete an answer method, while at the same time view all the different answer methods.

Use case: CRUD Task Template

Actors: Instructor, Admin

Goal: Should be able to CRUD a task template

Description: The actors should be able to create, update, and delete every task template. There should also be an overview of every task template.

Use case: HTTPS

Actors: System

Goal: Secure the connection between the server and the browser

Description: PEMA should use HTTPS for all communication so that the web-server and browser is protected against disclosure.

Use case: Authentication credentials

Actors: System

Goal: Authentication credentials should be protected against disclosure

Description: Passwords and Usernames should be protected against disclosure so that no unauthorized actor can gain access to the site.

Use case: Submit hash

Actors: Student

Goal: Should be able to submit a hash

Description: Every group should be able to submit a hash for a task that requires a hash as the answer type.

2.4 Sequence Diagrams

The sequence diagram for Lab Deployment is depicted in appendix [F.3](#). This diagram shows the steps an instructor would have take in order to fully create and deploy a lab.

A sequence diagram for task delivery can be found in appendix [F.4](#). This diagram shows the process of a user answering or trying to answer a task. It shows how a hash is checked for validity, then checked towards the students group. Lastly, the hash can also be checked against other groups.

3 Technical Design

3.1 Frameworks

There are a lot of options to choose from as the project description does not specify any framework to develop from. This "list" of options needs to be narrowed down in order to find the framework that fits us. Model-View-Controller is a widely known architectural pattern, which is simple to use and easy to understand. This is a good option for PEMA as all group members are somewhat familiar with it.

One of, if not the most, important part is choosing a programming language to write in. There exist a lot of different programming languages for backend, and there are always new languages that appear with new features, libraries and frameworks. For the backend there are only two options that are interesting to us, PHP and Python. This is because Python is one of the easiest languages to learn [7], and PHP is a language all members are experienced with. For the frontend we decided to look into HTML and JavaScript due to their wide use and our familiarity with the programming languages

The next sections will discuss more in detail about the different options regarding backend and frontend.

3.1.1 Deciding on server side programming language

PHP

The biggest benefit to PHP is that it is widely used and all group members have some experience coding in it, this is also beneficial for any future development. It has a large community and there are a lot of existing solutions out there, although, one must be careful because a lot of the solutions could be outdated. It was a good idea to keep in mind that this project will most likely be passed down to others at a later stage, with this in consideration PHP is a good choice as it is a widely known language.

Python

Python has gained popularity in recent years, making it a increasingly viable option. One of python's biggest strengths is its ease to read and ease to pick up and learn. In 2014, python became the number one introductory teaching language at universities in the U.S [8]. There are also a lot of different libraries which one can easily install, and python comes with a good community to offer help if you should stumble upon any trouble. The biggest framework for web development with Python is Django. Django uses the core concepts of the MVC pattern, and it is fast and simple to use [9]. It also has one of the best out-of-the-box security features, including:

- Clickjacking

- Cross-site scripting
- SQL injection

Conclusion

The conclusion is that PHP was a better option for the reason that our experience with PHP is greater than with Python. PHP also has a good community, which makes it easy to find solutions to various problems.

3.1.2 Deciding on frontend programming language

Vue.js is a JavaScript framework with various tools for building easy and good looking user interfaces. It is very small, the size of it is about 18-21KB [10]. It is relatively easy to understand and it can easily be added to a variety of different backends such as Django, Laravel, Wordpress, etc.. [11]. One simple "hello world" example of how vue.js works:

```
1 <div id="app">
2   <h1>{{ message }}</h1>
3 </div>
4
5 <script>
6 var myObject = new Vue({
7   el: '#app',
8   data: {message: 'Hello Vue!'}
9 })
10 </script>
```

Listing 3.1: Vue.js binding objects to HTML elements

The example shown in listing 3.1 shows how vue.js binds objects to HTML elements.

Conclusion

The other option is using regular HTML and JavaScript, which is a better option for us since learning a new framework could take a lot of time. There is also a larger community around regular HTML and JavaScript, meaning it would be easier to find solutions to already solved problems.

3.2 CMS and LMS

Content Management Systems and Learning Management Systems are good alternate approaches to developing a web-application as one does not have to start from scratch. The difference between LMS and CMS is that a CMS is a more passive application, which can be used as a standard website, while an LMS is more active approach, where users can be more interactive with the system. The biggest drawback of using a CMS or LMS is that one must learn how to use it which, depending on the CMS or LMS, could take a lot of time. The next subsections discuss the benefits and drawbacks of using Wordpress (CMS) or Moodle (LMS).

3.2.1 Moodle

Moodle is a free open-source Learning Management System, which is written in PHP. It is designed to provide administrators, educators and learners with a single secure, integrated and robust system to create personalized learning environments [12]. Moodle has, at the moment of writing this, over 100 000 registered sites, and over 150 million users [13]. It is also very well documented, and it has an active community where one can get help almost immediately. One of the drawbacks of Moodle, is that it seems like it is too big and too complicated for the functional requirements. PEMA does not need an entire learning platform where one could have multiple courses. PEMA needs something simple for between 50-100 students, where most of the interactive part will be submitting tasks and restarting virtual machines.

3.2.2 WordPress

WordPress is a widely known and free open-source Content Management System written in PHP. It is designed as an easy "click-and-drag" platform, where non-programmers could just click and drag the content they want to make the website look the way they want. WordPress is usually divided into three parts where all of them serve different purposes, Core, Themes and Plugins. The Core is WordPress itself; it comes with a ton of predefined functions and classes which are easy to use. Themes are the general layout of a page, if the Core is the cogs and wheels of the page, then the Theme is a layout of your page where you can interact with the "cogs and gears". There are plenty of both free and paid Themes on WordPress [14]. Plugins improve your website by enhancing the usability of your Theme. The best thing about WordPress, in regard to the functional requirements, is that it's really simple to remove and add plugins, i.e., easy to create a modular website where an instructor can remove/edit/add modules. There are also a lot of existing plugins which could prove useful to the website, especially in regards to security [15].

Latest WordPress Vulnerabilities

2019-03-13	WordPress 3.9-5.1 - Comment Cross-Site Scripting (XSS)
2019-02-19	WordPress 3.7-5.0 (except 4.9.9) - Authenticated Code Execution
2018-12-13	WordPress <= 5.0 - Authenticated File Delete
2018-12-13	WordPress <= 5.0 - Authenticated Post Type Bypass
2018-12-13	WordPress <= 5.0 - PHP Object Injection via Meta Data
2018-12-13	WordPress <= 5.0 - Authenticated Cross-Site Scripting (XSS)
2018-12-13	WordPress <= 5.0 - Cross-Site Scripting (XSS) that could affect plugins

Figure 6: Latest WordPress Vulnerabilities.

Latest Plugin Vulnerabilities

2019-04-23	Contact Form Builder <= 1.0.68 - CSRF to LFI
2019-04-17	WordPress Download Manager <= 2.9.93 - Authenticated Cross-Site Scripting (XSS)
2019-04-11	Download Advanced Contact form 7 DB <= 1.6.0 - Authenticated SQL Injection
2019-04-11	YellowPencil Visual CSS Style Editor - Unauthenticated Arbitrary Options Updates
2019-04-10	Yuzo Related Posts - Unauthenticated Call Any Action or Update Any Option
2019-04-09	WP Statistics <= 12.6.3 - Cross-Site Scripting (XSS)
2019-04-05	Duplicate Page <= 3.3 - Authenticated SQL Injection

Figure 7: Latest WordPress Plugin Vulnerabilities.

There are about 13900+ WordPress vulnerabilities in the WPScan Vulnerability Database [16]. The WPScan Vulnerability Database is a database that contains vulnerabilities for WordPress, both patched and unpatched vulnerabilities.

The latest WordPress vulnerabilities at the time of writing affecting the core can be seen in figure 6. There are only two incidents so far in 2019 that affects the core. Meanwhile, figure 7 shows there were seven plugin vulnerability incidents in the last month.

Almost all the recent WordPress vulnerabilities affects specific plugins or themes, it is very rare that vulnerabilities are for the core itself.

In 2016 a blog post on WordFence, a highly regarded security plugin for WordPress, talked about how WordPress sites were compromised [1].

Figure 8 shows that over 50% of sites were compromised because of a plugin they used.

One of the drawbacks with WordPress is that the Core is regularly updated, which means in order to prevent any vulnerabilities from appearing the plugins and themes would have to be regularly maintained.

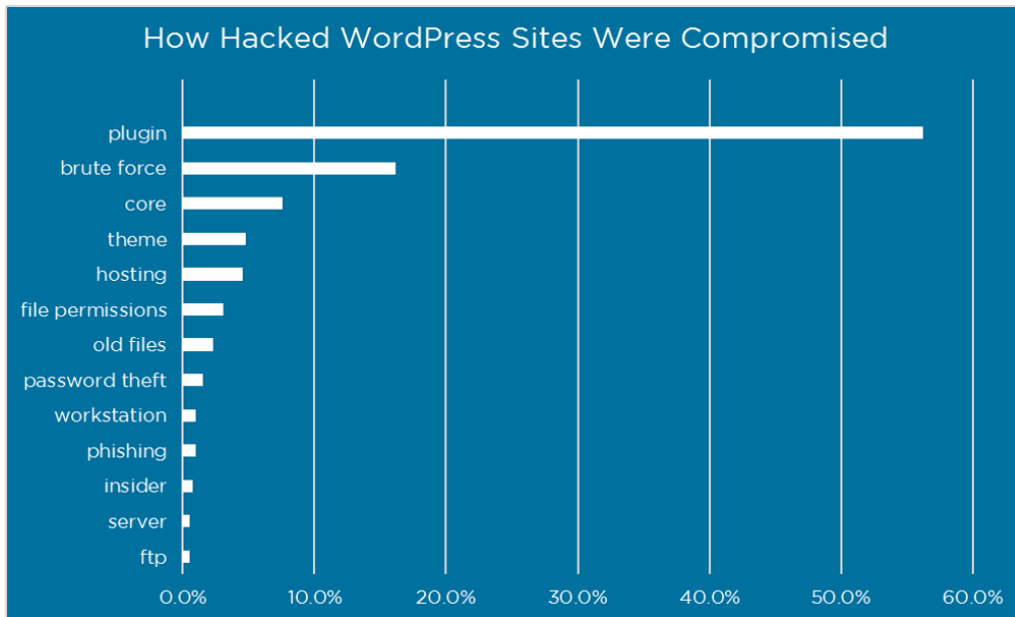


Figure 8: Latest WordPress Plugin Vulnerabilities [1].

3.2.3 Conclusion

Both WordPress and Moodle are similar, they both have a large community and good documentation. They both also have several plugins which could help satisfy the functional requirements. However, WordPress was a better option as Moodle would become too large and complex for the functional requirements. PEMA does not need a full blown site with tons of different courses, it only needs a simple site where an instructor can create labs for students. WordPress is just as good as Moodle, and most security vulnerabilities can be avoided by carefully choosing what plugins to use.

3.3 Web development

This section does a deep dive into most of the functionalities that WordPress offers.

3.3.1 Theme

WordPress Themes are what makes your WordPress site yours. The theme is the content and look of the site. These are either found pre-made as complete sites with loads of functionality, or templates that can be further built upon.

A pre-made WordPress theme would be a quick way to have a potential baseline. There are some difficulties with this approach. First, finding a theme that suits the functional requirements. There are some LMS themes that could work, as the PEMA front-end will not be much different from a standard LMS. For example the Education LMS theme [17], which is a clean LMS targeted at learning institutions, could be an alternative.

Using a pre-made theme is not just plug-and-play for this implementation. PEMA needs several features that are missing from most LMS themes, mean-

ing that there is a need to learn how the theme is built to implement features to meet the functional requirements. Furthermore, there are some support and security concerns with a theme developed by someone else. The developer of the theme can stop supporting it, potentially leaving it unsupported or vulnerable.

On the other hand, there is only the need to code plugins for the theme, which potentially can save time. Only developing plugins gives a very natural segmented development process.

Conclusion

For PEMA it fitted best to develop a theme from scratch. Not having control over the theme, and the potential for easier expansion and maintenance on the platform were prioritized, even if it takes longer.

3.3.2 Page Templates

A page template is a specific type of file that can be used on specific or multiple pages. They are primarily used to change the look and feel of a page or multiple pages. They could be created to target a specific page or multiple pages based on the name of the page template. If there is a need to create a page template for a page that will contain information about the developers of PEMA, the page template can be called `page-developers.php`. If there is a need to have a page template that would be used multiple times, we would just call it `page.php`. The reason `page-developers.php` would only be used in specific instances is because of how WordPress uses what is called a Template Hierarchy.

Simply put, WordPress searches down through the template hierarchy until it finds a matching template file. WordPress uses multiple different template hierarchies, there are 6 different page types, Archive Page, Singular Page, Site Front Page, Blog Posts Index Page, Error 404 Page, and Search Result Page [18].

3.3.3 WPDB Class

In the core of WordPress a class called `wpdb` is defined, which contains a set of functions that are used to interact with the database [19]. It's primary usage is to provide and interface with the WordPress database, however, it can also be used to communicate with any other appropriate database. The source code of the class is based on the `ezSQL` [20] class which is written and maintained by Justin Vincent. The limitations of the `wpdb` class is that it can only interact with one database at a time. When interacting with the `wpdb` class the global object variable called `$wpdb` is used. For it to be used, there is a need to declare the object as a global variable, an example of this is depicted below, in listing 3.2.

```
1 global $wpdb;
2 $results = $wpdb->get_row("SELECT * FROM
    pema_lab WHERE labID = 1");
```

Listing 3.2: Gathering information about a lab using the `$wpdb` class.

The example shown in listing 3.2 gathers all information about a lab with the labID of "1".

3.3.4 Code Practices

In order to create good readable code, PEMA follows the WordPress Coding Standards [21]. WordPress coding standards are divided into four different parts, PHP Coding Standards, HTML Coding Standards, CSS Coding Standards, and JavaScript Coding Standards. The coding standards are a set of guidelines that recommends practices, methods and programming style for developers of WordPress.

3.3.5 Enqueuing Scripts

Enqueuing scripts is a way to insert a meta link to your script. WordPress specifies that one should never hardcode such links in the headers [22].

Enqueuing scripts are enqueued with the `wp_enqueue_script()` function. The function takes at least one parameter, the name of the script, which is a unique name that a developer set [23]. However, before using this function one usually use the `wp_register_script()`, which takes at least two paramters, a unique name of the script and the path to the script [24].

`wp_register_script()` is a safe way of registering scripts before enqueueing them with `wp_enqueue_script()`. Both functions have a third optional parameter, which is used to set any dependencies for the function.

```
1 wp_enqueue_script( 'ajax-script',
2   plugins_url( '/js/myjquery.js', __FILE__ ),
3   array( 'jquery' )
4 );
```

Listing 3.3: Enqueueing scripts in WordPress

The listing above, 3.3, shows an example of how a script is enqueued in WordPress, firstly a unique name for the script is specified. Then the URL of where to find the script is specified and lastly the array of dependencies for the script are specified. The `plugins_url()` function just gets the directory of the plugin you are in, for getting the theme directory you would either use `template_directory()` or `stylesheet_directory()`.

3.3.6 Plugins

WordPress Plugins are pieces of software that allow you to easily modify, customize and add new functionality to a WP site. They are a way to extend the features and functions available in a WP theme, by simply adding the plugins that are needed. Using plugins instead of implementing everything into a theme or even the WP core having more features is to avoid bloat, to keep WP lightweight, and without many features you do not need for your theme. WP has an official plugin repository with over 55,000 plugins! [25] There is a saying in the WordPress community, "There's a plugin for that" [26], implying that if there is a feature you want or need, there is a good chance there is a plugin already made for it. Having all these pre-made plugins allow users

with limited knowledge of web-development to implement the functionality they need for their site without knowing a single line of code.

Adding plugins is the best practice when it comes to implementing future modules into PEMA. This way the required modularity is easily managed, as the different modules can be enabled and disabled as needed.

This is mostly based on the same decisions as with not using a pre-made theme. The lack of control over the development of the plugin and how secure the plugin is overall.

There is a possibility to use third-party plugins for PEMA later down the line, but the plugin will need to be reviewed first. For example, there are a plethora of forum plugins, several of which are used by thousands and even millions of users [27].

An example of a plugin that is created for PEMA is the PLED API integration. This is made as a separate plugin, where when enabled the option to add a vulnerable application to a task template appears on the "Create Task Template" page, with the query from the PLED database ready to be filtered.

Plugins are integrated into WP sites through the WordPress Plugin API, using hooks. There are two different types of hooks, Actions and Filters.

The PLED-API plugin is integrated using a custom hook of the action type, which calls the plugin code.

3.3.7 AJAX

AJAX is used to request specific information from the server and display it to the end-user without refreshing the page, greatly improving the user experience. The traditional data exchange format used is XML, however many people prefer JSON because it has an easier structure to interface with. WordPress uses AJAX a little differently than normal, the sequence of events usually goes as follows, firstly a page event initiates a JavaScript or jQuery function. The function gathers some data from the page and then sends it via http to the server. Afterwards, the server receives the request and does something with the data. Lastly, the jQuery or JavaScript function that sent the initial Ajax request receives the server response.

All Ajax requests sent through jQuery functions needs to be sent to the "wp-admin/admin-ajax.php" file. Since the URL to the file needs to come from PHP, as jQuery cannot determine the value on its own and it is not a good practice to hardcode the URL in jQuery, PEMA would need a way for jQuery to always grab the URL from PHP. This can be done with the `wp_localize_script()` function which, with the right parameters, creates global objects that the jQuery scripts can use.

```
1 wp_localize_script( 'ajax-script', 'my_ajax_obj'
2     , array(
3     'ajax_url' => admin_url( 'admin-ajax.php' ),
4 ) );
```

Listing 3.4: Creating objects for jQuery

In the example above, listing 3.4, an object called "ajax_url" is created which gets the URL to the "wp-admin/admin-ajax.php" file. The object can then be used in jQuery by calling it like this: `my_ajax_obj.ajax_url`. One can also create nonces, a number used once, with the `localize_script` function. This is further discussed in chapter 6 subsection 6.1.3.

A data tag that Ajax needs is the action argument, which is used with an action hook to tell WordPress what function to use on the server side. There are two different action hooks that Ajax uses with WordPress, `add_action('wp_ajax_my_function', 'my_ajax_handler')` which allows logged in users to utilize the Ajax exchange and `add_action('wp_ajax_nopriv_my_function', 'my_ajax_handler')` which allows users who are not logged in to utilize the Ajax exchange.

3.3.8 Hooks

Hooks are functions that allows your plugins or themes to 'hook into' the WordPress core at specific times. There are two different hooks, actions and filters. Filter hooks are used to modify output when it is sent to the front- or backend. They can be added with the `add_filter()` function [28]. Action hooks are used at more specific times, such as when a post or page is created or when a theme is activated. The action hooks are called with the `add_action()` function. It is also important to note that one can also create custom hooks by calling either `do_action()` or `do_filter()` depending on what hook you want. Creating custom hooks are important for the PEMA web-application as it allows plugins to 'hook into' our theme.

3.3.9 jQuery

jQuery is a lightweight open-source JavaScript library. It contains many features with cross-browser capability, making things like document traversal, event handling and animations simple and easy to implement. jQuery is a natural choice for any WordPress developer, as it is included with WP, as well as being one of the most commonly used libraries [29]. Using a library with many other users means there is an abundance of support from a large online community.

The main jQuery use in PEMA is as part of database communication. For every page where an entry is created, updated or removed from the database, jQuery gathers input from the page, checks that all requirements are met, and sends it to Ajax.

3.3.10 jQuery Plugins

According to jQuery's own website, a plugin is a collection of elements. Much like the jQuery core methods can be considered a plugin. [30] In other words, it is much like WordPress plugins. They add functions that extend the functionality of the site.

PEMA uses jQuery in a few different ways. Most of the jQuery plugins used in PEMA are for convenience and well-made solutions. jQuery plugins offer a quick way to solve a problem in an elegant manner, that would take days to

create manually.

Just like with any other third-party plugins, there are some concerns. Before using plugins in PEMA, they needed to be reviewed, much like the WordPress plugins.

3.4 Database

The choice of database is restricted mainly by the choice of content management system. WordPress natively only supports MySQL and MariaDB, however it is possible to use plugins to alter this behavior, and in turn support for example PostgreSQL or even other kinds of databases such as MongoDB. One of the requirements going into this project was to make the system as simple as possible by actively using standard systems and procedures. This will in turn make the web portal easy to pick up for those who will further develop and maintain it. With this in mind, our two real choices are then MySQL and MariaDB. This section discusses the reasoning when making this choice.

3.4.1 MySQL

MySQL is an implementation of a relational database and was until it was purchased by oracle a project licensed under the GNU GPL. After its integration into the oracle ecosystem, MySQL was split into two branches, the continued open source MySQL, and an enterprise version. The enterprise version includes extra utilities and support; however this is not relevant to the web portal, and as such does not support such a purchase. MySQL in general is a database with great performance, and which scales well. The support system around the open source version consists solely of forum posts, documentation and general user to user support.

3.4.2 MariaDB

MariaDB is an open source fork of the open source MySQL and is founded and maintained by much of MySQL's original team from before it was bought by oracle. Seeing as MariaDB is a fork of MySQL, it is also fully compatible with the database- and configuration-files made by a MySQL database and vice versa. As an example, this makes migration between the two in our infrastructure as simple as changing the package name from "mysql" to "mariadb". In addition to being fully compatible with MySQL, MariaDB provides better performance, and has built-in support for Galera, a clustering technology which would further increase performance.

3.4.3 Conclusion

When making the choice between MariaDB and MySQL, the decision was mainly based on our considerations about licensing and performance, seeing as these two technologies stem from the same root. The open nature, the better performance in addition to the potential for clustering was what made us choose MariaDB.

3.5 Webservers

When one chooses to develop a website using WordPress as the content management system, one would also need a webserver which can serve the site. In their documentation [31], WordPress recommends using either nginx or Apache, as in their view, these are the most feature rich and robust, however they also note any web server which supports PHP and MySQL will work.

3.5.1 Nginx

Nginx is a webserver, reverse proxy, load balancer, mail proxy and cache all-in-one. It is a lightweight webserver which works by spawning worker processes to handle incoming requests and has master processes which orchestrates this functionality. This master/worker functionality also implements a simple way of handling the updating of nginx itself, where nginx spawns a new master-process with the new functionality enabled, and then kills the old master when its workers are done. This functionality provides a site which can theoretically have an uptime of 100%. Nginx, like MySQL, provides both an open source web server, and a licensing agreement called nginx plus. In our project, nginx plus is not considered, as the extra functionality is deemed unneeded, and usage of open source software has been a wish set forth by our employers.

3.5.2 Apache

Apache is the most widely used web server, however recently it has been passed in one category, namely the market share of all sites [32]. It is an open source project with no paid options, making Apache totally open source, as opposed to nginx. The nature of Apache being open source also means there is no income related to the server, and the developers are dependent on donations from its users. As such, the official support is limited to documentation [33], and informally through forums, chat, and other means of user-to-user support. Restarting apache in the event of an update is a slow process, as the whole server needs to be restarted and rebuild its in-memory configuration, taking some time. Apache's recommended method of doing this is to do a graceful restart, which tells all masters to stop receiving new requests, and terminate when done. In the meantime, the engine recompiles itself, starting new master processes which will resume operation. If the configuration fails for any reason, the new process will not start, and will lead to the website going down, as there are no workers to handle requests.

3.5.3 Conclusion

When taking these points into account, nginx seems like a good choice, especially considering its high performance in high-throughput scenarios, which is one of the wishes for this platform to become. However, using a properly open-source project like apache would be a boon, seeing as open source is a wish from our employer. All these considerations are however thrown out the window when we look at what docker images are available for the technologies we have discussed in this chapter. WordPress provides their own im-

ages with ready-to-go WordPress installations, and these are solely built on Apache. It would not be impossible to install WordPress into an nginx-image, in fact, there are already community-made images which does this for us. The decision to use the official WordPress-Apache image was finally made because we deem the decreased complexity and increased reliability through officially supported images more important than using nginx.

3.6 Choosing an Operating System

As we will discuss in the next section, our choice landed on using containers for this service. This decision somewhat affects our decision on what operating system we would like to use for our deployment of the system, however it does not entirely dictate our choices here.

3.6.1 Windows

Windows is by far the most prominent OS on consumer desktops and laptops, consuming somewhere in the neighborhood of 70-87% of the market [34] [35]. On the web-server side, according to W3Techs, windows accounts for about 30% [36]. Windows is a competent operating system, and is used by everything from home-laptops to even some super-computers.

3.6.2 Linux

Linux is by far the most widespread operating system being run on websites, holding a market share of 69.5% [37]. Because of this, the Linux implementation of differing software is what has received the most support in documentation and support-forums. In addition, the traditional LAMP-stack (Linux, Apache, MySQL/MariaDB and PHP) is the preferred way of developing a website in many cases.

3.6.3 Distributions

If and when one chooses to use Linux, the second decision one would need to make is what distribution, or variant, one would use. Red Hat Enterprise Linux (RHEL), CentOS and Ubuntu/Debian are 3 major distributions used as web-platforms. RHEL is a licensed distribution, and as such, we will not be using it. CentOS is built upon RHELs binaries; however, CentOS is a free distribution. Debian is a distribution which focuses heavily on the usage of free software and stable releases. Ubuntu is built upon Debian, and its aim is being as beginner friendly as possible. To achieve this, Ubuntu is built upon some proprietary software.

3.6.4 Conclusion

The infrastructure for PEMA is partially dependent on the operating system on which it runs, as it is ran as microservices deployed as docker-containers. These microservices can be deployed on any operating system which supports docker. However, as a web server is traditionally built as a LAMP-stack, the platform is designed and tested on Linux. One example of a dependency upon Linux is the implementation of volumes in the containers. The final deploy-

ment of the platform will be on NTNU's private cloud SkyHIGH, an Openstack-based cloud platform, using Ubuntu virtual machines.

3.7 Containers or Virtual Machines

When making a choice on the overarching infrastructure around our platform, the choice was between docker-containers and virtual machines. Both platforms offer distinct advantages, and these are what will be discussed here.

3.7.1 Virtual Machines

Deploying the infrastructure on "raw" virtual machines would offer the advantages that the infrastructure should be easily understood by just about anyone who would want to pick up this project. Deployment on virtual machines is an easy process that consists of simply installing the required services and starting them. This could also be somewhat automated through technologies like Heat-templates, wherein one would make a yaml-file which describes the machines and what software should be installed. When choosing to use a virtual machine, one important drawback is that the choice of operating system becomes much more important. A service deployed on an Ubuntu 16.04-image might be configured entirely differently than if it were deployed on a windows 10 server or even Ubuntu 18.04, and it is most certainly installed differently, this is not the case in a container, as the container abstracts this away.

3.7.2 Containers

When using containers, a distinct point to make is that virtual machines or "raw steel"-machines (actual hardware configured to act as the server) are still required. Containers are simply a way to install services in a partially separate environment, wherein the container has access to the core of the OS, but the software that is separate, but needed by the service is installed locally in the containers. This has the advantage that whenever a service needs to change, or be removed in its entirety, that container can be stopped and deleted, and all traces of that service ever existing will go with it, and in the event of a change, simply re-started. The disadvantage is that this form of deployment is not private to those who have access to the underlying virtual machine. To overcome this, containers should not be used as a replacement for virtual machines, but rather as a complimenting feature. Containers also have the advantage that whenever a reboot is required by the service, or whenever the service needs to scale up, a container is much quicker to start, as compared to a virtual machine. When developing the containers, and the services within them, the underlying operating system can be mostly ignored, however some complications do exist. Linux- and Windows-containers function somewhat differently, and, in the case of docker and more specifically docker-compose, will need differing composer-files. This behavior mostly comes from the fact that windows and Linux treat storage-drives entirely differently, as Windows views the drives as entirely separate entities with different drive letters, and Linux simply mounts the drives on given paths relative to the root directory.

3.7.3 Scaling

Another important aspect to any infrastructure is its scalability. Considering the slow boot-time of virtual machines, scaling a service based on the live demand is challenging, and would need careful planning ahead of time, and some careful crafting of OS-images which have the needed services ready to go. In contrast, scaling a service deployed as micro-services in a docker-swarm is as simple as altering the amount of replicas in that swarm by issuing a command such as:

```
1 docker service scale Pema_site=10
```

Listing 3.5: Docker scaling command

3.7.4 Conclusion

Designing the infrastructure for a relatively simple service is still a complicated issue. Choosing technologies, whether that is what web server should be used, or what operating system should be used is a decision which has the potential to severely affect the end-result. With this and the points made in their respective sections, our choice landed on using docker containers. Docker containers add complexity in that the technology needs to be learned before it can be properly utilized, and this cost comes with what might look like limited benefit if not properly understood. The benefit in choosing docker comes from the increased flexibility for whoever might want to use the platform, and the ease of updating the platform as a result of the relationship between services and files. Our employer gave us this task with the wish that the platform should be easily applied by whatever third-party would want to deploy a platform such as PEMA, and docker affords this ease of deployment. To deploy the web-platform of PEMA, one would only need a docker swarm to be configured, and to run our script called `stack_deploy.sh`, and this ease of deployment is the main point to us choosing docker. In addition, the simple scaling of the service through the `docker service scale`-command, makes the service relatively simple to scale to fit everything from a small class-environment, up to, in theory, a large-scale CTF challenge hosted on an international level. Even though the choice fell on a container-solution, there is a need to make a choice on what solution fits best. This choice was relatively simple, as the technology taught most heavily during this bachelor degree is docker, and we see no definitive benefit to choosing a different solution which would warrant learning another technology such as kubernetes.

3.8 PEMA Cooperation

The PEMA project is not just the front-end. It is a broader project being worked on by two bachelor groups, as well as one master student.

The groups are PEMA, which is ours. Then there is PLED, developing a database. And Mikhal Dunfeld, the master student developing a Domain-specific Language.

3.8.1 PLED

PLED is the Pentesting Lab Environment Database, developed by another bachelor group. The database will gather and store data about vulnerable applications, and where to download these. PLED is also developing an API which we will query. The development of querying PLED is further discussed in chapter 5 subsection [5.5.2](#)

3.8.2 DSL

The DSL is a domain specific language being developed by the master student Mikhal Dunfeld. It is a language designed to take a YAML-file as an input and will define vulnerable architectures based on said input. Its relationship with PEMA is defined by PEMA being able to utilize the DSL to deploy its labs, and keep records of what the DSL makes, and hand that over to the instructors and students. The benefit of this cooperation is that the instructor will have their task simplified by simply defining the architecture of a given lab once, and having PEMA handling the technical deployment of said lab.

4 Development Process

4.1 Development Tools

In this section, we will go through the tools we have used during our project.

4.1.1 Coding Environment

WordPress

The coding environment for the frontend has been kept quite similar between us. We have been using XAMPP to run an Apache and a MySQL server locally. Using the Apache server to host WordPress, and MySQL to host a PhpMyAdmin database used for the native WordPress database as well as for our theme. When developing the WordPress theme, this environment has been useful, as we can see our code in action right away.

The actual coding has been happening in different editors though. We have been using Notepad++, Sublime Text 3 and Atom. Notepad++ got swapped out early in the project, as it has minimal tab-completion, and requires some tedious setup for ease of use, and consistent formatting with the other text editors. Atom and Sublime on the other hand, natively support PHP, HTML, CSS and JavaScript natively, with tab-completion, correct colour-coding and many other useful assisting tools.

This code is then pushed to a central Git repository, hosted on NCRs Phabricator platform. All groups working on the broader PEMA-platform have their own repository on Phabricator platform.

4.1.2 Project Management

File sharing

For file sharing within the group, we decided on using Google Drive. This platform is something all members of PEMA are familiar with, as we have been using it for most projects through our study at NTNU.

Google Drive also offers several other useful programs and features. Google Docs has been especially useful for us. All our notes and meeting-logs are written in Docs, as it has excellent collaboration capability, and a quick and easy-to-use interface that we are particularly familiar with. Draw.IO is another one, we have used this tool to make all our diagrams, with Drive integration. This allows us to easily collaborate while creating all sorts of diagrams.

To share documents with our employer and other groups we are collaborating with, we have mainly been using Box, an online file-sharing tool, which was set up by our employer as a way to share files outside of code being shared through Phabricator.

Report

For our report, we are writing in \LaTeX , using the collaboration-tool Overleaf. This tool is familiar to the group, as our group has used it multiple times to write reports for other projects over the last few years. It provides a nice and user-friendly interface, with live collaboration, making the sharing of ideas and ideas for changes simple.

Group-communication

During our project we use several different types of communication for different uses. The communication within the group is good and flows easily, as we have become good friends through our study at NTNU, as well as having worked together on multiple projects earlier. Having such good relations between us make us good collaborators, as we are able to talk freely and honestly to each other. This can be particularly beneficial in a professional setting, because we are not afraid to tell other members that we for example are not satisfied with their work.

For communication outside of meetings we use several different platforms, each with their own uses. To communicate casually, ask for quick thoughts on ideas as well as off-topic conversations, we use Facebook's Messenger. Messenger is a messaging application owned by Facebook, where we mostly use the text functionality, but also has voice and video capabilities. We have a group chat on the platform we have used for over two and a half year, mostly for off-topic conversations, but also as the quickest way to either get another group members input on a topic, or to plan for when we will schedule the next meeting etc.

While programming, we like having the possibility of talking together, and to have the ability to quickly ask each other for input on a problem for example. We could naturally go to campus and sit together programming there with our laptops, but we find that to be highly ineffective compared to working from home at our workstations. To facilitate working from home and communicating efficiently, we have used two different platforms. Most of the time, we talk together using the VoIP client TeamSpeak 3, using our own server. This has been our go-to platform for VoIP communication for years, making it the natural choice for this use.

The only problem with using TeamSpeak is that it is missing one quite useful feature, screen-sharing. When you are stuck with a problem, rather than trying to explain your code to another group member, it is much easier to show them! To allow us to use screen-sharing, we used Discord. Discord is also a VoIP client, but with many other features as well. Discord is disliked by several members of the group, which is the reason we have not used it for all our VoIP communication, even though it would be possible.

Other communication

Other group members are not the only people we need to communicate with. We naturally need to communicate with our employer, our advisor, and the other groups we are collaborating with.

Most of our communication with our employer and advisor has been happening through e-mail and during meetings.

But for quicker communication with our employer and the other groups, our employer set up a Discord text-server for all groups working on the broader PEMA-project. This is used for quick chatting, questions, and meeting-planning.

Communication security

Regarding communication security, the employer did not have any requests or demands. But information regarding the project has not been thrown about without regard. There are safeguards for all platforms where information about PEMA has been shared.

Voice communication has been through a private server, with voice data being encrypted. The VoIP service primarily used, TeamSpeak 3, uses "AES-based encryption" for its voice encryption [38]. What variation of the Advanced Encryption Standard [39] is used is unclear, as TeamSpeak are quite secretive about this.

All code has been stored in a Git repository through NCRs Phabricator platform. Using Phabricator hosted by NCR, means that NCR has control over the contents stored there [3].

And all notes have been stored in a private Google Drive, where only group members have had access. Google claims all data stored in Google Docs form is encrypted both in transit and at rest [40].

Finally, the report has been written using Overleaf, endorsed by the university, as the accounts used on the platform are paid for by NTNU. Overleaf also has encryption for communication [41], and use Amazon S3 for storage, where some encryption is default [42]. What encryption is actually used for the Amazon storage is unclear.

Task distribution

Our task distribution has been through Kanban boards using the web tool Trello. Here we have distributed the tasks into multiple boards with different sections of the project, and further made tasks with checklists. Trello is yet another tool we are familiar with, through earlier projects. It provides a simple interface with all the information you need about the task, what is being worked on, and by who.

This has been a useful way to keep track of what needs to be done, what someone is currently working on, and what needs to be tested.

Time tracking

To track the time we have spent on the project, we decided to use Toggl. Toggl is an application designed for enterprise time-tracking, with way more features than we will ever need. We have created a project in Toggl, that all members of the group are part of. This project allows us to see when other members are working, what they are working on in the form of categories, and how much they have worked. Having this information visible to the group is a good motivator, as well as allowing other members to for example remind someone that they are falling behind, or congratulate someone on their good efforts. The output from the time tracking can be found in appendix I.

4.2 Planning for Future Work

From early on in the project, the employers have stated that this is a project that will be developed further after our work is done. This bachelors project is meant as a proof of concept, to figure out whether creating the platform is feasible [4].

With this in mind, the project has been created in a manner such that the person or persons taking over the project will not struggle learning how the platform is built. The platform strives to have good documentation and to have code that is commented well. A person with a similar or greater knowledge about relevant technologies than the bachelor group developing it should not struggle to get to know the platform.

The project has had a focus on core-functionality, rather than implementing every feature the employer could want, as that would not be possible within the given time-frame. Rather, facilitating simple implementation of new features on top of that core has been a focus. For example, the fact that PEMA is built on WordPress, and its capabilities to allow future developers to create modules in the form of WordPress plugins.

5 Implementation

5.1 Setup of test environment

5.1.1 Webservice

Development of the web application was done using XAMPP, which is a free open source web server solution developed by Apache Friends [43]. It comes with both PHP and MariaDB, with MySQL, which suits the groups needs when it comes to choice of programming language. XAMPP was chosen because all group members are familiar with it. For setting up the testing environment, a guide was followed which sets up both XAMPP and WordPress locally [44].

5.1.2 Docker

The docker deployment was mainly tested on a desktop computer running Manjaro-linux. This choice was made because the docker environment would be designed and developed both on and for a Linux machine to run. We concluded that it would be no different to develop and test on local machines than if we would actively test on a virtual machine hosted on the NTNU cloud service skyhigh. Testing the docker deployment was mainly done on the local machines where we verified that the local deployment is functional before deploying and fine-tuning in the production environment hosted on our virtual machine on skyhigh.

5.2 Installation and configuration

To install the PEMA web-platfrom, one needs two things: a configured docker swarm, and access to the git-repo which hosts all the php, javascript, bash scripts and other files. When these requirements are fulfilled, the platform can be launched by changing directory to the root of the repository, and running the command `./scripts/stack_deploy.sh`. This script does two thing: sources the file `.env` which should be duplicated from the file `default.env`, and populated with relevant information for the deployment, and it runs the command as seen in listing 5.1

```
1 docker stack deploy \  
2   --compose-file docker-compose.yml Pema
```

Listing 5.1: Deploying the service into the docker swarm.

5.2.1 Configuration

To configure a certain instance of the web-platform, one only needs to first copy the file `default.env` into another file called `.env`, this file is not included in the git repository, and as such, is safe to keep for example the password for the database. In this file, change the values to whatever fits your

needs, and the configuration is done on the backend. After deploying the stack with the script specified in the last section, open a web-browser and navigate to the website. Here, follow the instructions on screen, and create the administrator-user. Next, navigate to the admin-panel if you are not already there, by entering `https://YOURDOMAIN/wp-admin`, then click "Appearance -> Themes", and activate the theme "Pema-Webapplication". At this point, the website is ready to be used as intended.

5.2.2 The source code

The source code of the Pema-platform can be found on NCRs own repositories during the development of the platform. To gain access to this repository, one would need an invitation from the norwegian cyber range. However, our employer has expressed a wish that this platform should be open source, and available to anyone who wishes to host such a platform themselves, and as such needs to be moved off this private repository and onto a public one, like GitHub.

5.2.3 Docker swarm

To deploy the website, one does not necessarily need to use docker swarm, however our solution contains ready-made scripts and files which together makes it simple to deploy on such a swarm. The simplest possible way of deploying would be on a docker swarm with only one node, the manager-node. To do this, install docker on your preferred linux-machine (for example Ubuntu) [45]. Next, run the command as seen in listing 5.2

```
1  docker swarm init \  
2  --advertise-addr <MANAGER IP>
```

Listing 5.2: Initializing the docker swarm.

When doing this, take note of the output, as this will tell you what command needs to be ran to enter that swarm using other virtual machines. When this is done, simply run the script `stack_deploy.sh` which can be found in the `scripts` folder. Alternatively, or on demand, one can add more nodes to the swarm using the command which was output by the initialization. For instructions on how to properly install and deploy docker swarm, we highly recommend seeing the documentation provided directly by Docker [46], as there are some variations and considerations to take into account depending upon the environment.

5.3 Docker implementation

Our docker environment is a docker swarm. Docker swarm is a feature of docker which allows for replication and load-balancing of microservices. To achieve an infrastructure which would handle varying amounts of traffic, we made the choice to design the infrastructure such that the website is easily scaled up or down. In addition to the simple and proper scaling of the website, we chose to also place our database inside the swarm. The database is at this point not able to scale, however our choice of MariaDB should enable those who would like to use this platform to integrate a Galera cluster, which is a clustering-technology for databases. The benefit of placing our database comes from it then not being available from outside the swarm. This decision was made, as the database only ever needs to be accessed by the website which also runs in the swarm, and it somewhat increases the security of the platform, as all potential attacks would now have to go through either the website, or directly through access to the virtual machines included as a swarm-node, and can not come as a result of poor password-protection on the database itself.

5.4 Web-application

5.4.1 Directory and File Structure

A WordPress theme only needs four files, `index.php`, `style.css`, `header.php`, and `footer.php`, however a theme is usually made up of multiple files [47]. There is no structure syntax that WordPress offers, other than that files mentioned above should reside in the root directory of the theme. PEMA uses 6 directories, where each serves an individual purpose. Alongside the directories, it also uses a number of different files including `404.php`, `sidebar-top.php`, `sidebar-home.php`. A full list of directories and files can be found in Appendix F.10. The folders `admin-view`, `css-admin`, and `js-admin` exist to divide the menu and sub-menu pages to their own respective directories for easier editing and referencing. Their names represent what the menu or submenu does and they are easily distinguishable from one another. These files are also only loaded on their respective menu or submenu page. The folders `js` and `css` are used to define the default values or functions that all `css` and `js` files can access. Both `pema_admin_style.css` and `pema_admin_script.js` are files loaded on the admin dashboard, while the `script.js` file is loaded on the student view. The `style.css` file is required to be in the root of the directory, which is why it is not in the `css` folder. The `inc` folder is an extension to the `function.php` file, where the files have a specific purpose. `pema-on-startup.php` is used to maintain code that is supposed to be run when the PEMA theme is activated, while the `pema-on-switch.php` file is used to maintain code when PEMA is deactivated. The file `pema-ajax.php` is used to maintain all code that is sent through AJAX, this is the main bulk and code of PEMA.

5.4.2 PEMA Roles

The PEMA theme has two main roles, instructor and student. To separate these roles, the instructors will use a modified admin-dashboard, where the users abilities are restricted compared to an administrator. The students only use the theme view, the website part of the site, where they can see their labs, tasks, etc.

The separation is easily implemented through WP's permission system. It allows for deciding whether a user-class can see the admin-dashboard at all, and further what parts and features on the dashboard are available. This feature is used to keep students off the admin-dashboard entirely, just allowing them to see the relevant labs and courses they are enrolled in. The instructors are able to see only what they need, task, lab and topic management, limited user management and announcements. The administrators can see everything an instructor can see, as well as the admin tools. These include features like full user management and WP management, including plugins, settings and themes. How the admin-dashboard menu looks for each of the user-roles can be seen in figure 9.

WordPress roles and permissions are further discussed in chapter 6, regarding security.

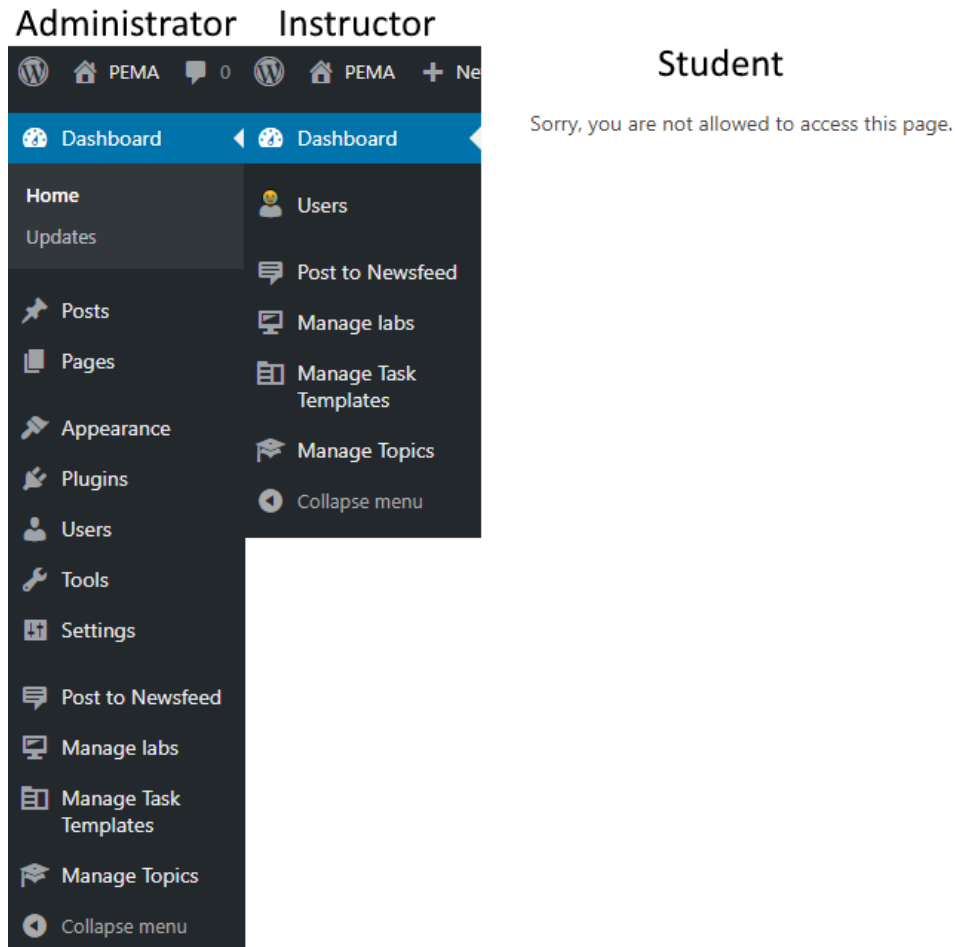


Figure 9: WordPress Admin-dashboard menus per role

5.4.3 PEMA-Lab Hierarchy

The PEMA lab and task system consists of PEMA-Objects. The main objects are labs, tasks and topics. The hierarchy showing how the different parts interact is shown in figure 10.

Starting from the "smallest" object, there are topics. Topics are simply the different categories a task can reflect. Some examples of these are XSS (Cross Site Scripting), SQL Injection or scanning.

Next there are tasks. These are the objectives of the labs. As previously mentioned, these have only one topic, and has an objective related to this. Each task has a title, and a description describing the objective with that task. Tasks can also be time-limited, as well as requiring an answer.

The "biggest" object is the lab. The lab has its own information in the form of a title and description. A lab is essentially a collection of tasks, with a broader objective, or mission. An example can be "You are given access to a network, your objective is to acquire a file containing information about the companies employees, located on one of the machines..." A lab also has information about when it will start and end, as well as what user groups are

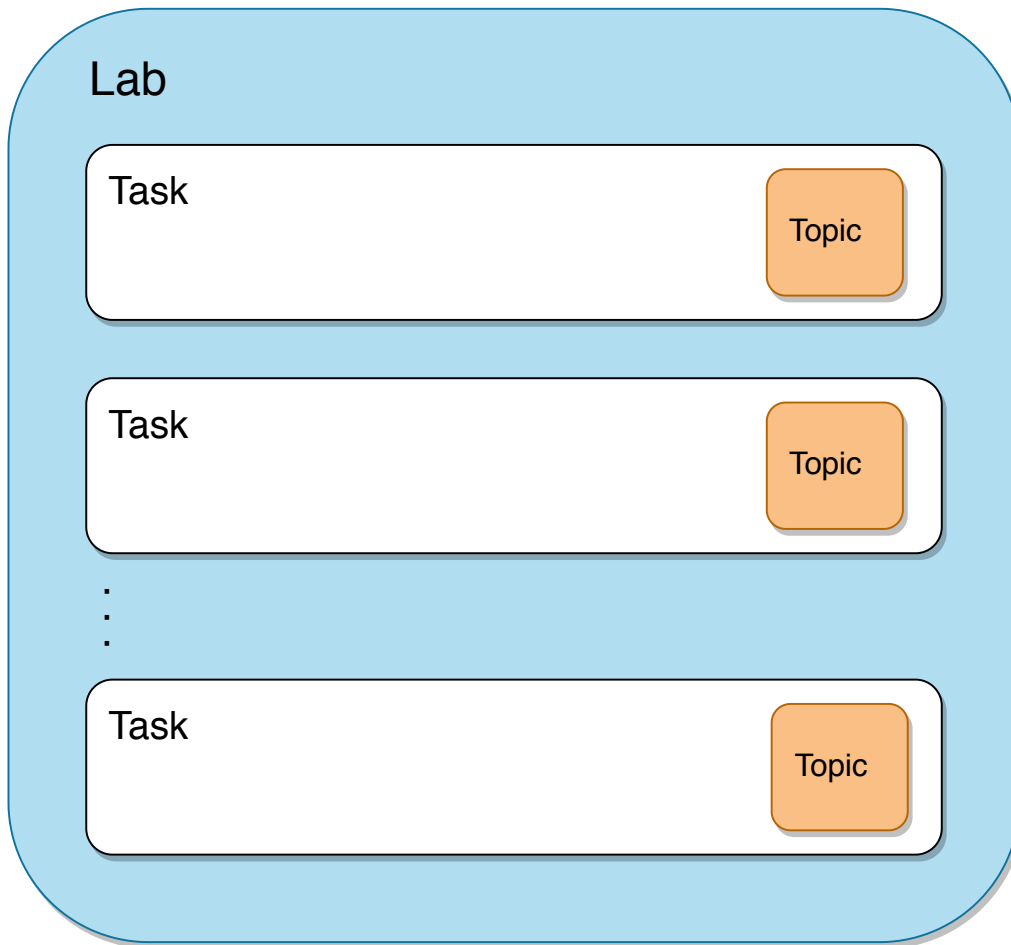


Figure 10: PEMA-Object hierarchy.

taking part in it.

For a more complete, but still simplified example:

- Hacking Lab: Acquire Information.
 - You are given access to a corporate network. Your objective is to gain access to as many machines as possible, and to acquire information about the employees. This is stored as a file on one of the machines.
- Tasks:
 - *Scanning*: Scan the network to gather information about the machines.
 - *Exploitation*: Gain access to as many machines as possible.
 - *Search*: Find the file with information about the employees.
 - **Answer type, String**: How much does Adrian from accounting make per year? [Answer input]

The example is a single lab with three tasks. Each task has a single topic, written in *italics*, and the final task has an answer in the form of a string.

5.4.4 The Loop

One of the powers of using WordPress is utilizing what is called "The Loop". The Loop is a set of different PHP functions used by WordPress to display posts, and process each post to be displayed on the current page. According to the WordPress Codex, The Loop should be placed in the Theme's "index.php" file to display post information [48]. As said, The Loop is meant to be used to display posts, however, the functional requirements didn't specify any uses of posts. This is why an idea of creating a "news" section was added, where an instructor can post updates and news regarding the PEMA platform.

```

1 <h1>Newsfeed</h1>
2 <?php
3 if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
4 <!-- Checks if there are any posts, then goes through the posts and
   displays it-->
5 <br>
6 <div class="pema-news-title">
7 <h2><?php the_title(); ?></h2>
8 <!-- Gets the title of the post -->
9 </div>
10 <div class="pema-news-content">
11 <?php the_content();
12 ?></div><?php
13 endwhile;
14 endif; ?>

```

Listing 5.3: "The loop" Searching for and displaying posts.

Listing 5.3 above shows how "The Loop" is used in PEMA, it first checks if there are any posts, and if there are it loops through all of them and displays their content. The code snippet uses only WordPress functions from the core.

5.4.5 Page Templates

PEMA uses a number of different page templates. PEMA utilizes a customized Error 404 page when a user tries to access a page it cannot find, e.g., user tries to enter something that isn't valid. When an instructor creates a lab, a unique page is also created for that lab. However, in order to create the link between the page and the lab it was required to create metadata about the lab which would connect the page and lab. This metadata is created when a page is created.

```

1 $newPage = array(
2   'post_title' => $title ,           // Title of page
3   'post_content' => $description ,   // Content of page
4   'post_status' => 'publish' ,      // Status is set to publish
5   'post_type' => 'page' ,           // Type is page
6   'post_author' => $id ,            // Author of lab page
7   'meta_input' => array(            // Insert metadata
8     'labID' => $lab_get_inserted_id , // Set labID as metadata for page
9   )
10 );
11
12 wp_insert_post($newPage);          // Insert the page

```

Listing 5.4: WordPress page creation example.

The block of code above, in listing 5.4, shows how a page is inserted. First an array of options is created. This array retrieves info which is sent from jQuery with AJAX. Then the page is inserted using the function `wp_insert_post()`.

The function `get_page_link()` is used to get the link to the page, the only required parameter is the page id. PEMA utilizes the page template "page.php" for displaying information about a specific lab. When a user tries to access a page called "Lab-1", WordPress will look for a page called "Lab-1". It will not find a page called "page-Lab-1" and instead load the page.php template, WordPress will also remember the post queried and that can be utilized with the `$post` class [49]. This means that every lab will be displayed with the "page.php" template.

The process for displaying tasks are very similar. When a task is created for a lab, then a page for the task is simultaneously created. The parent page for the task page is set to be the lab page, this way every task will fall under a lab. The page also gets a custom page template set which is called "template-task.php", this means that every time a user navigates to a task, WordPress will use this as a template page for displaying information.

5.4.6 Custom Admin Pages

For PEMA, having the ability to add custom pages to the admin-dashboard facilitated a natural separation of instructors and students. Having the instructors use custom pages on the WP admin-dashboard, rather than creating a whole separate area on the main page from scratch fits the project well. Controlling access is also made simple by this, which is touched on already, and will be further discussed in chapter 6.

Admin-dashboard pages are, quite intuitively, all the different pages found on the WordPress admin-dashboard. These pages are accessed through menu and sub-menu entries on a sidebar.

In figure 11, the default WordPress admin-dashboard sidebar can be seen. The figure shows the Theme page, which is a sub-menu entry under the Appearance menu entry.

Everything that is needed to create a custom page that is accessible through the WordPress dashboard is a file with source-code, and to create a menu-entry that links to that code.

```

1 add_action( 'admin_menu', 'pema_labs_register' );
2 function pema_labs_register() {
3     add_menu_page(          // Adding menu entry for Labs
4         'Lab View',         // page title
5         'Manage labs',     // menu title
6         'pema_view_lab',   // capability
7         'pema_manage_labs', // menu slug
8         'pema_view_lab',   // callback function
9         'dashicons-desktop' // Icon
10    );
11
12    add_submenu_page(       // Adding submenu for lab-creation.
13        'pema_manage_labs', // Parent menu slug.

```

```

14     'Create lab',           // Page title
15     'Create lab',         // Menu title
16     'pema_create_lab',    // Capability
17     'pema_create_labs',   // Menu slug
18     'pema_create_lab'     // Callback function
19 );
20 [ ... ]

```

Listing 5.5: Adding menu and sub-menu entry for the admin-dashboard.

The function above shows the creation of a menu and a sub-menu entry. This is done using the standard WordPress functions `add_menu_page()`; and `add_submenu_page()`, respectively. These functions take parameters for page title, the menu title, what capabilities/permissions are required to access the page, the "menu slug", a callback function and the name of an icon. The menu slug is the name WP uses for that specific menu, this is used to create sub-menu entries, seen as the first parameter in the `add_submenu_page()` function in the above example. The callback function is a function that is run whenever the menu item is accessed. This is used to call a function that displays the contents of the page.

Below, in listing 5.6, the function called when pressing the menu-entry added in the previous code example is shown.

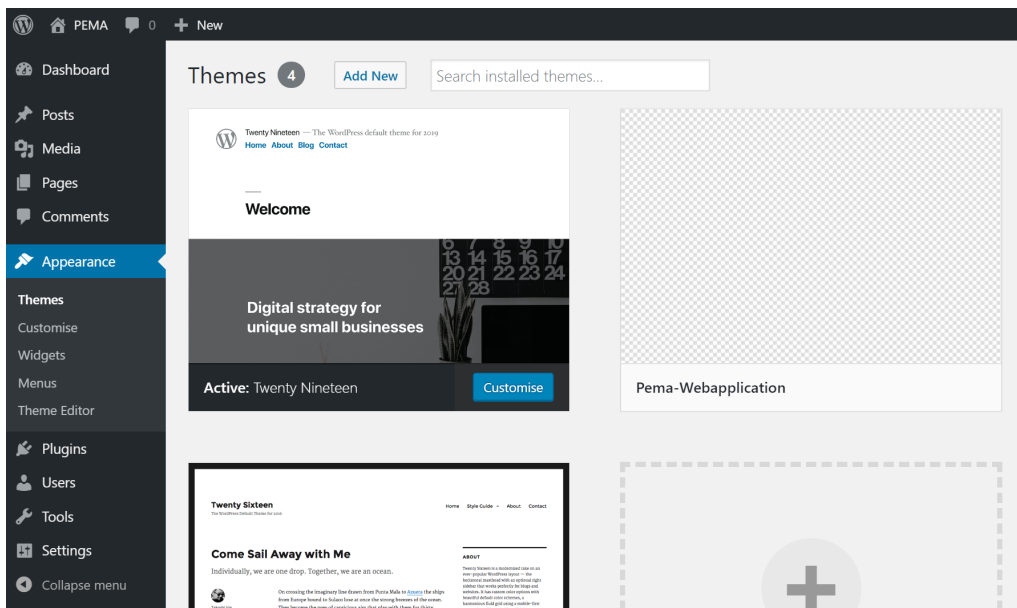


Figure 11: WordPress admin-dashboard default sidebar.

```

1 function pema_view_lab() {
2   // Get full path
3   $file = plugin_dir_path( __FILE__ ) . "admin-view/manageLab.php";
4
5   if ( file_exists( $file ) ) // Check if file was found/exists.
6     require $file;           // Require/load the file.
7 }

```

Listing 5.6: Rendering a page after menu-call.

The function gathers the full path to the file with the page code. There is then a check to see if it was able to get a path, and the file exists. If it exists, the file is required, and loaded on the page.

5.4.7 Enqueing scripts and styles

As mentioned in previous chapters, enqueueing scripts and styles is a way to insert meta data or stylesheets in the web page. PEMA utilizes quite a few different set of stylesheets, like the "flatpickr" jQuery plugin. In order to register all stylesheets PEMA uses a foreach loop as depicted in the example below.

```

1 // Goes through all CSS files in the css-admin directory and registers
   them for use
2 // The array is dependencies, i.e., it will only load after
   pema_admin_style
3 foreach ( glob( get_template_directory() . '/css-admin/*.css' ) as $file )
4   {
5   // Rename $file to be the file name only
6   $file = str_replace( get_template_directory() . '/css-admin/', '', $file )
7   ;
8   wp_register_style( $file , get_template_directory_uri() . '/css-admin/' .
9     $file , array( 'pema_admin_style' ) );
10 }

```

Listing 5.7: Example of registering all .css files

In listing 5.7, a loop goes through all files that ends with .css in the css-admin folder and registers them. The glob function simply returns an array of filenames or directories matching a specified pattern [50]. There is a similar foreach loop for registering all JavaScript files. PEMA only loads specific stylesheets and scripts on specific pages as depicted in the example below.

```

1 // Create Labs page
2 if ( 'manage-labs_page_pema_create_labs' === $hook ) {
3   wp_enqueue_script( 'create_lab.js' );
4   wp_enqueue_style( 'create_lab.css' );
5 }

```

Listing 5.8: Example of enqueueing scripts and stylesheets on a specific page

The if statement in listing 5.8 first checks if \$hook is equal to manage-labs_page_pema_create_labs. \$hook is sent on every page load and is a combined string which tells WordPress what page it is loading. By doing this we can ensure that scripts and stylesheets are only loaded on pages that actually utilize them.

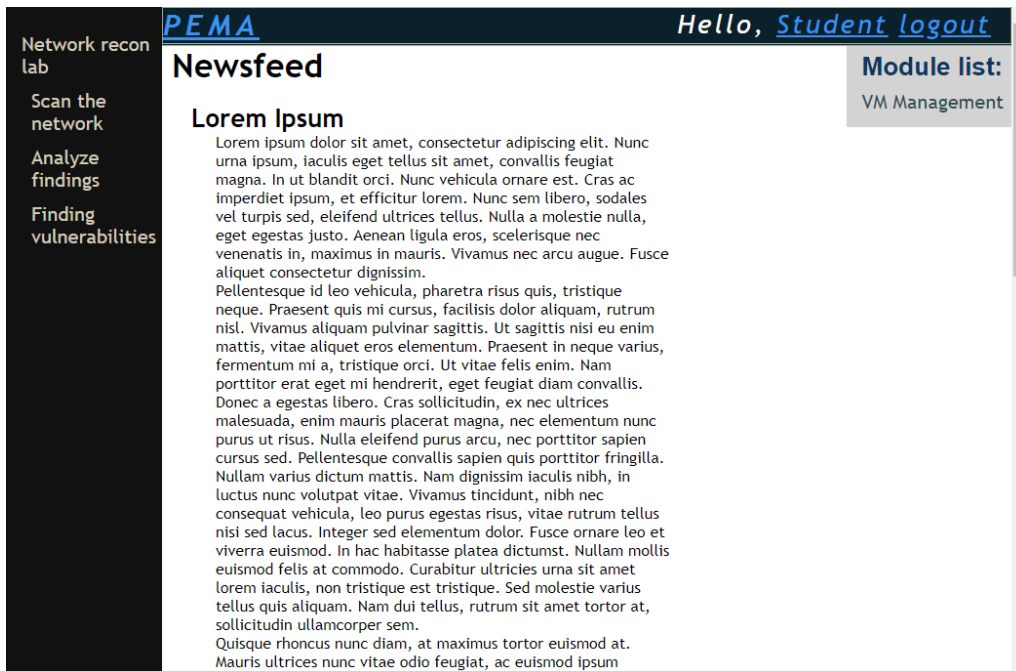


Figure 12: Preview of PEMA-navigation.

5.4.8 Site Navigation

The navigation around PEMA is done mainly through two navigation bars. A sidebar to the left, called 'home', and a module-navigation sidebar to the right. These can be seen in figure 12.

The sidebar lists labs and tasks, as well as having a link to the admin-dashboard for instructors and administrators. The module-navigation bar, located to the right on the page, lets the user navigate between the different module-pages. This bar allows plugins to hook into it, to create a link to the page for that module. In WordPress this bar is called "sidebar 'top'", because of how WordPress wants objects to be named, as well as its intended location was originally on the top, just below the header.

5.4.9 Usage of WPDB class

This subsection will highlight PEMA's usage of the WPDB class, and the functions most commonly used in the implementation. The `get_results` function gets multiple rows, it returns an array of all the results found. The default usage is `$wpdb->get_results('query', output_type)`; where the query is the SQL query you would run.

The return value will be an empty array if the query cannot find any matching rows or if there is an SQL error, otherwise the query returns all the rows that match the query. Another function which is very similar is the `$wpdb->get_row('query', column_offset)`; query which, instead of getting every row that match, gets just one row.

For inserting to the database, the `$wpdb->insert($table, $data, $format);` function is used. You first specify what table to insert into, then you specify what data you would like to update in an array format (in column => value pairs).

```

1 $wpdb->insert(
2   'pema_lab',           //Table name
3   array(
4     'labName' => $title,           //Sets labName
5     'labDescription' => $description, //Sets labDescription
6     'labStart' => $startdate,     //Sets labStart
7     'labEnd' => $enddate         //Sets labEnd
8   ),
9   array(
10    '%s',           //Int
11    '%s',           //Int
12    '%s',           //String
13    '%s'           //String
14  )
15 );

```

Listing 5.9: Example of inserting a lab

In listing 5.9, a new lab with parameters is sent by Ajax and inserted. One important thing to remember with this function, is that you should not SQL escape the `$data` as the function does this for you. It will return false if the function fails.

The update function `$wpdb->update($table, $data, $where, $format = null, $where_format = null)` is used to update either one or several rows in the database. You are required to specify the `$table` name, what `$data` to affect and the `$where`. The `$format` and `$where_format` are optional and are used with the `$prepare` function which is discussed in chapter 6.

```

1 $updated = $wpdb->update(
2   'pema_task_template', // Database table name.
3   array(
4     'templateName' => $title, // Title.
5     'templateDescription' => $description, // Description.
6     'topic' => $topic, // Topic.
7     'answerType' => $answerType, // AnswerType.
8   ),
9   array( 'templateID' => $id ), // Update row with correct templateID.
10  array( // Define value-type.
11    '%s', // title, string.
12    '%s', // description, string.
13    '%d' // topic, int.
14  )
15 );

```

Listing 5.10: Example of using the update function

Listing 5.10 above shows code that updates a task template with new data. If the function is successful it will return the number of rows affected, or else it will return error, however if the `$data` that is sent already matches it will instead not update any rows and return 0.

The last function we've mostly used is the `$wpdb->delete($table, $where, $where_format = null)` function, which is used to delete either one or several rows. It follows the same structure as `$insert` and `$update` where you are required to specify what table to delete from and where in the table to delete from, e.g., `$wpdb->delete('pema_lab', 'labID = 1')`. It will return the number of rows deleted if successful or false if there is an error.

5.4.10 Custom hooks

As mentioned in previous subsections [3.3.8](#) creating custom hooks are a way to allow plugins to hook into the PEMA code. There are a number of different custom hooks spread around the code. A list of these can be found in [appendix D](#). An example of usage can be found in [listing 5.11](#).

```
1 do_action('pema_after_insert_task', $taskID,
           $pageID);
```

Listing 5.11: Example of creating a custom hook.

The example creates a custom hook with two parameters, the ID of the task and the ID of the page. If a plugin developer wants to add more code to the web application they need to call the custom hook as [listing 5.12](#) shows.

```
1 function plugin_expand($taskID, $pageID) {
2     // Do something...
3 }
4 // Hooks into the custom hook, and run the function 'plugin expand'
5 // 10 means the priority it gets, and 2 means number of parameters sent
6 add_action('pema_after_insert_task', 'plugin_expand', 10, 2);
```

Listing 5.12: Example of using the created custom hook.

The example shows how the custom hooks can be used to inject code into PEMA by a plugin developer. The plugin developer is responsible for securing its code.

5.4.11 jQuery

jQuery is used in PEMA for JavaScript. It's use ranges from displaying HTML objects to sending information to AJAX for database insertion. As a simple example, the following block of code ([listing 5.13](#)) gets called when a button in a table is clicked. When the button is clicked, the code gathers information about what table row the button is in and toggles the visibility of some objects on the same row.

```
1 // When option is chosen.
2 $(document).on('click', '.pema-manage-topic-
3     button-edit', function(){
4     // Get current row.
5     var currentRow=$(this).closest("tr");
6     // Get first column in current row (task
7     number)
8     var col1 = currentRow.find("td:eq(0)").text();
```



```

7
8 // Toggles the input field visibility.
9 $('#pema-manage-topic-name-input-' + col1).
  slideToggle();
10 // Toggles submit button visibility.
11 $('#pema-manage-topic-button-submit-' + col1).
  slideToggle();
12 });

```

Listing 5.13: JavaScript code example. Using animation to reveal object.

For some added flair, the jQuery function `slideToggle()`; is used to reveal the objects. The function plays a sliding animation to reveal them, rather than just having them appear suddenly.

5.4.12 jQuery Plugin

PEMA utilizes a few jQuery plugins, they are mostly used as a convenient and nice-looking way to gather or display data. For example the flatpickr plugin [51]. Flatpickr is a plugin that gives a clean and intuitive calendar-like interface for picking dates and times, seen in figure 13.

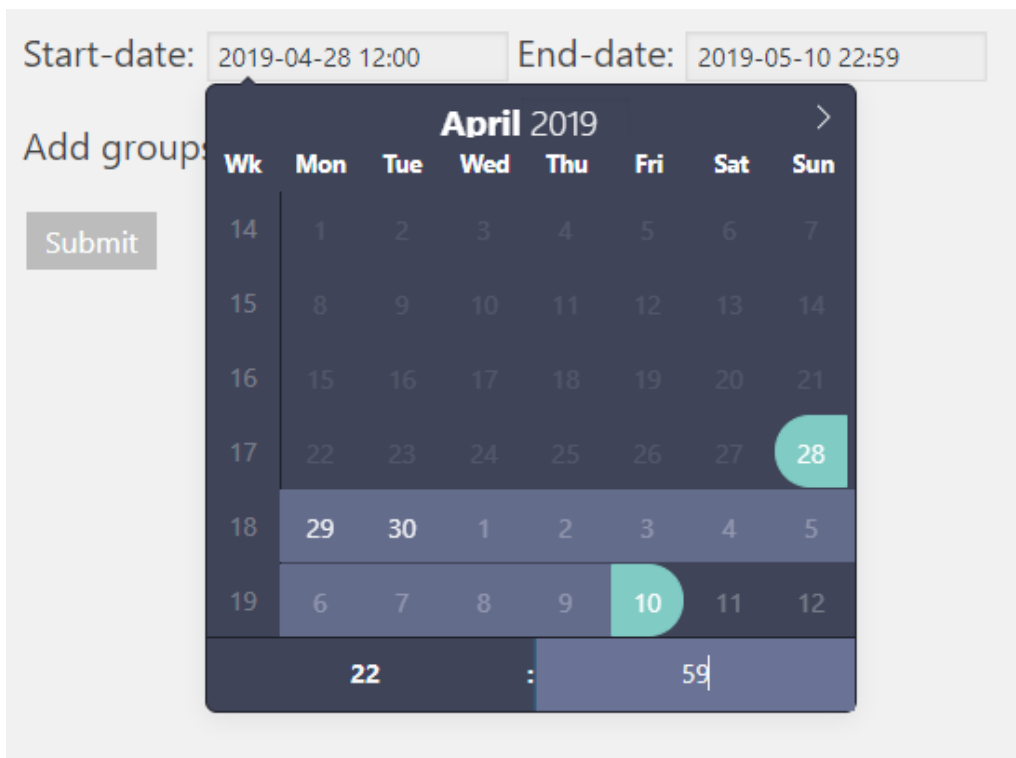


Figure 13: Flatpickr date and time example

In PEMA, this is particularly useful, as there are several places where dates are selected, like when a lab should start and end, as well as its tasks. The plugin also takes multiple parameters, allowing for customization of what information is gathered, and on what format. It even allows the selection of

a range, like in figure 13, where both Start and End-date is selected in one window. The plugin has a parameter that can limit the "legal" dates, in other words, the dates that can actually be selected. Which is useful for making sure for example a task does not start before or end after the lab it is part of.

The actual jQuery-implementation of the flatpickr example in figure 13 can be seen in figure 5.14

```

1 $('#pema-create-lab-startdate-input').flatpickr({
2   enableTime: true,           // Enable time (e.g., 12:00)
3   dateFormat: "Y-m-d H:i",   // Date format
4   time_24hr: true,          // Sets 24-hour times
5   weekNumbers: true,
6   minDate: "today",         // Minimum date is today
7   locale: {                 // For the Locale..
8     firstDayOfWeek: 1       // Set first day of week to monday, like
      it should be.
9   },
10  // Use plugin to select a range, putting second date in the enddate-
      input field.
11  "plugins": [new rangePlugin({ input: "#pema-create-lab-enddate-input" })
      ]
12 });

```

Listing 5.14: Flatpickr implementation

The function is called for an input-field, and customized using parameters. The flatpickr plugin even allows for the selection of a range, giving the user a seamless experience selecting the start- and end-date for the lab.

Another example of a plugin that is not used for input is the jQuery-confirm plugin [52]. This plugin is used for aesthetic purposes and convenience. The plugin gives a selection of alerts, with different looks and features.

PEMA uses colour-coded variations of these to notify users of events. jQuery-confirm also has a feature allowing for timed execution of an option, which has been useful for notifying users of a successful action, followed by a refresh a few seconds later to reveal the changes. An example of this can be seen in figure 14.

5.5 Plugins

5.5.1 Openstack Plugin

The OpenStack plugin is a plugin that utilizes the OpenStack API to create a Kali machine for every group that is added to a lab on lab creation. The reason we created this plugin was to have a proof of concept that we could create virtual machines without the DSL. In short, the plugin hooks into a custom hook that we created and runs through the plugin code. Mainly what the plugin does is create a keypair, floating ip and a server for each group that is added to the lab. At the same time it stores metadata about the server, floating IP, server id and private key, in the database so that a user can access the machine. In order to run all the API OpenStack calls, one needs a token, which is a randomly generated hash of about 250 characters. This token expires every 4 hours which means the plugin needs to create a new token at least

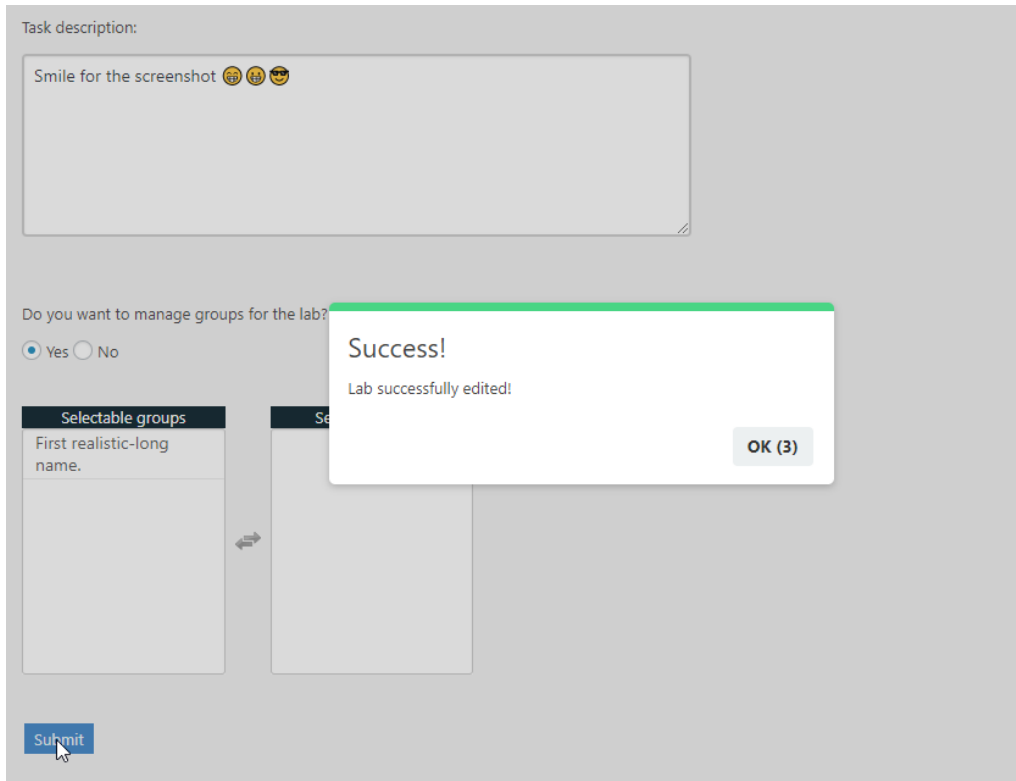


Figure 14: jQuery-confirm example

every 4 hours. This cannot be done manually, so the plugin utilizes WP-Cron [53]. WP-Cron works almost the same as cron on UNIX systems, the biggest difference is that WP-Cron only checks for scheduled tasks to run on page load. This does consume some resources, however, since the webapplication won't be having too much load it is acceptable to utilize WP-Cron.

```

1 // If the schedule event doesn't exist
2 if ( ! wp_next_scheduled ( 'pema_openstack_every_three_hours' ) ) {
3 // Schedule a function to run every three hours
4 wp_schedule_event( time () , 'three' , 'pema_openstack_every_three_hours' );
5 }

```

Listing 5.15: Scheduling using WP-Cron

The function above first checks if `pema_openstack_every_three_hours` is already scheduled, if not it will create the scheduling event.

The plugin also adds a new page to PEMA, where students have an overview of each of their Kali machines. From there they are able to hard reboot the machines.

5.5.2 Integration with PLED

PEMA uses a plugin which queries PLED's database for information about different types of vulnerable applications. The PLED database has an API which PEMA queries everytime it needs some type of data. This is done with the WordPress HTTP API [54]. In order to query their database PEMA needs an

API key, this has been provided by PLED. WordPress has some helper functions for using CURL in PHP and PEMA utilize these.

```
1 $args = array(  
2   'headers' => array(  
3     'accept' => 'application/json',  
4     'X-DreamFactory-API-Key' => 'key-string'  
5   ),  
6   'timeout' => 60,  
7   'sslverify' => false  
8 );  
9 $type = wp_remote_get('http://10.212.137.92/api/v2/customsearch/type',  
10  $args);  
11 $types = json_decode(wp_remote_retrieve_body($type));
```

Listing 5.16: Example of using the update function

The code in listing 5.16 above shows how PEMA queries a GET request towards the PLED database. First the arguments to pass in the CURL command are created, the only required parts are the headers parts, where the API-Key and the accept format is set. All API calls, that PEMA utilize, towards the PLED database return in JSON format. Afterwards, the function `wp_remote_get()` is used. This function runs a GET request and returns the raw response, the results include the headers and content [55]. Then the returned response is decoded to JSON and the response is minimized to only include the content with the function `wp_remote_retrieve_body()` [56]. PEMA does not run any POST requests to the PLED database, as it is only intended to retrieve information from it.

The plugin, more specifically, first retrieves the different types of types and platforms. This allows the instructor to filter by all the different types and platforms. The instructor can then query the database for vulnerable applications based on the filtering they entered. As of this moment, the plugin doesn't do anything else because the implementation towards the DSL is not finished. The whole point of the plugin was to pass on the information retrieved from PLED to the DSL, and because the DSL was not integrated with PEMA it doesn't really do anything other than being a proof of concept.

6 Security

6.1 Wordpress

6.1.1 Prepare function

In order to prevent SQL injection WordPress uses a function in the WPDB class called `prepare`. This function always takes two parameters, the query you want to run with `sprintf()`-like placeholders, and an array of arguments with the variables to substitute into the query's placeholder [57].

```
1 $topic = 'SQL Injection';
2 // Get Topic
3 $topics = $wpdb->get_row($wpdb->prepare("SELECT
    * FROM pema_topic WHERE topicName = %s",
    $topic));
4 // If database returned null, means it did not
    find any matching rows
5 if ( $topics === NULL ) {
6     wp_die('Error');
7 }
```

Listing 6.1: Exampe of WordPress prepared statement.

The code in listing 6.1 is used to check if a 'topicName' exist in the database, if it doesn't it will exit with an error message 'Error'. It also shows how the 'prepare' function uses `sprintf()`-like placeholders, in the example a '%s' is used because the 'topicName' is expected to be a string meaning it will not work if a integer is sent. This works exactly like a prepared statement would. The valid `sprintf()`-like placeholders the prepare statement takes are '%s' (string), '%d' (integer) and '%f' (float).

One important note here is that with some functions in the WPDB class, like 'insert()', 'delete()', 'update()', and 'replace()', there is no need to either escape the variables or use the 'prepare()' function. This is because WordPress escapes and utilizes the 'prepare()' function by default in the core on these specific functions.

6.1.2 Output Encoding

WordPress comes with a few helper functions when it comes to securing output from the database. The most noteworthy one is the `esc_html()` function, which is used to encode and strip some text for invalid or special characters like `<` `>` `&` `"` `'` (less than, greater than, ampersand, double quote, single quote). There are helper functions for escaping HTML attributes, textarea and URLs. PEMA utilizes these helper functions in order to create secure output. A list of all the different helper functions can be found in the WordPress Codex [58].

6.1.3 Nonces

WordPress uses what's called a nonce, number used once, to help protect URLs and forms from certain types of misuse. However, WordPress nonces are hashes, made up of numbers and letters, and they can be used more than once, although they have a limited lifetime after which they expire [59]. The WordPress nonces serve the same purpose even though they aren't a number used only once. They do not protect against relay attacks as the nonces aren't checked for one-time use, however, they do protect against CSRF. In PEMA nonces are only used for verifying all Ajax requests. Meaning a nonce needs to be created on the frontend when a user loads the page, then it needs to pass the nonce to Ajax and then confirm that the nonce is correct on the server side.

In chapter 3 the use of a function called `wp_localize_script()` was discussed. This function is a way to pass values into JavaScript objects properties, as PHP cannot directly 'echo' values into a JavaScript file. PEMA uses the function to also create a nonce as depicted in listing 6.2.

```

1 wp_localize_script( 'script' , 'ajax_params', array(
2   // The URL for admin-ajax.php
3   'ajaxurl' => admin_url( 'admin-ajax.php' ),
4   // Gets the currently logged in user ID
5   'loggedinuserid' => $current_user->ID,
6   // Creates a nonce
7   'ajax_nonce' => wp_create_nonce( 'pema_ajax_nonce' ),
8  ));

```

Listing 6.2: Utilizing the `wp_localize_script` function.

The example shows that a nonce is created and stored in the object `'ajax_nonce'` and jQuery can send this to the server side by using it as an object with Ajax.

```

1 $.ajax({
2   url : ajax_params.ajaxurl,           //Gets the ajax url.
3   type : 'post',                     //Sets post type.
4   data : {                            //Data values.
5     title1 : title,                  //Title.
6     content1 : content,              //Content.
7     id : ajax_params.loggedinuserid, //Gets the logged in users ID.
8     security : ajax_params.ajax_nonce, //Gets the nonce
9     action : 'pema_create_post'      //Calls function pema_createPost.
10  }

```

Listing 6.3: Getting the nonce in jQuery

In listing 6.3, the code retrieves the created nonce by calling `ajax_params.ajax_nonce` and sends it to the server side as the variable `'security'`. This means that PEMA can retrieve the variable and use it with a function called `check_ajax_referer('pema_ajax_nonce', 'security')` which checks if the nonces match [60]. This function will die and give a 401 error if the nonces do not match.

This way PEMA protects itself from CSRF attacks by validating nonces on all Ajax requests.

6.1.4 Security Plugins

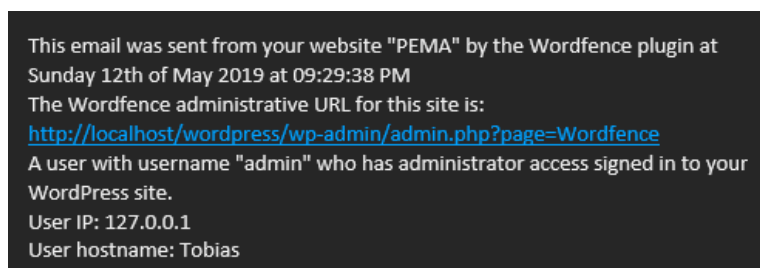
The plugins in this subsection is not already installed with PEMA, but they are recommended as they improve the security of WordPress.

Password-hashing

When a user is created in WordPress, the core salts and hashes the password with 8 passes of MD5 [61]. Since MD5 is fast it means an attacker can try billions of candidate passwords per second on a single GPU [62]. The reason WordPress uses MD5 still is because it is supported by all platforms, luckily, WordPress has made it simple to overwrite their password hashing functions with a plugin. For this, we chose to use the plugin 'PHP Native password hash' [63] because it swaps out the native WordPress hashing with the PHP function `password_hash()`. This is good because the PHP function uses `bcrypt` which is recommended because it uses deliberately slow hash constructions meaning it will take much longer to brute force. The plugin also salts passwords using a Cryptographically Secure Pseudo-Random Number Generator, the hashing is also iterated multiple times to provide good resistance. When the plugin is activated, it will also go through all the current user passwords and replace them with `bcrypt` password hashing instead of the native MD5 hashing in WordPress. This means that users don't have to reset their passwords when the plugin is activated, however, if the plugin is deactivated it will not revert back to the native password hashing meaning users will have to reset their passwords.

WordFence

WordFence is a security plugin for WordPress which comes with a list of improvements when it comes to security. The only drawback with WordFence is that some of the protection mechanisms are behind a pay-wall. The price for a 1 year license of WordFence is \$99. Even though there are some features behind a pay wall, it is still a good plugin. It includes a firewall and a WordPress security scanner, a full list of functionalities can be found on their page [64].

The image shows a screenshot of an email notification from the WordFence plugin. The text is white on a dark background. It reads: "This email was sent from your website 'PEMA' by the Wordfence plugin at Sunday 12th of May 2019 at 09:29:38 PM. The Wordfence administrative URL for this site is: [http://localhost/wordpress/wp-admin/admin.php?page=Wordfence](\"http://localhost/wordpress/wp-admin/admin.php?page=Wordfence\"). A user with username 'admin' who has administrator access signed in to your WordPress site. User IP: 127.0.0.1. User hostname: Tobias".

```
This email was sent from your website "PEMA" by the Wordfence plugin at
Sunday 12th of May 2019 at 09:29:38 PM
The Wordfence administrative URL for this site is:
http://localhost/wordpress/wp-admin/admin.php?page=Wordfence
A user with username "admin" who has administrator access signed in to your
WordPress site.
User IP: 127.0.0.1
User hostname: Tobias
```

Figure 15: Example of WordFence sending an email when a user logs in on our test environment

The example in figure 15 above is taken from an email that WordFence sends when a user logs in on the website. It includes who logged in, where they logged in from, and what access rights they have. WordFence can be accessed by the "WordFence" admin page on the admin dashboard, from there

an administrator have full access to all the functionalities for WordFence.

WP Limit Login Attempts

WordPress core does not have anything to limit login attempts, which means that PEMA would either need to look for existing plugins that does this, or create one on its own. The decision to use a plugin was made early on, because it simply would take too much time to develop a plugin specifically for PEMA. The decision to use the plugin called "Limit Login Attempts Reloaded" was made because of its GDPR compliance and because it blocks specific set of IPs after a specific limit on retries have been reached [65]. There are a lot of customizable options, where you can whitelist a specific set of IPs, change the number of retries before you are locked out or change how long you are locked out for. The plugin can be customized from the settings tab on the WordPress dashboard.

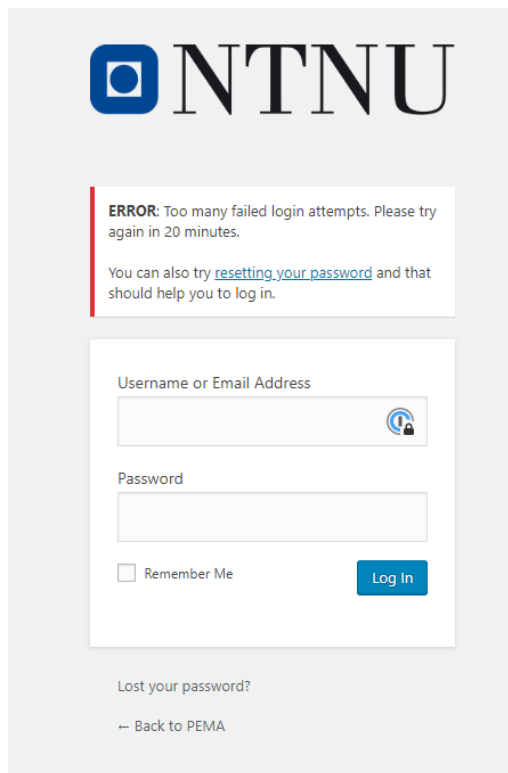


Figure 16: Example of typing being locked out

Figure 16 shows how the plugin locks you out after a set amount of tries to enter the password. Again, the amounts of attempts you have or the timer you are locked out for are customizable. The user will see how many attempts they have left before they are locked out. The plugin is GDPR compatible because it obfuscates the logged IPs.

WP Force Login

PEMA uses a plugin called "WP Force Login", which forces login for all visitors [66]. The plugin comes with a lot of different action hooks that allows a developer to change certain behaviors of the plugin. By default usage, the plugin redirects to the web page they tried to visit. This can be overwritten by setting a specific URL to always redirect to after logging in with an action hook. An example of this can be seen in listing 6.4

```

1  /**
2   * Set the URL to redirect to on login.
3   *
4   * @return string URL to redirect to on login. Must be absolute.
5   */
6  function my_forcelogin_redirect() {
7   return home_url( '/mypage/' );
8  }
9  add_filter( 'v_forcelogin_redirect', 'my_forcelogin_redirect' );

```

Listing 6.4: Adding a role to WordPress.

This action filter will always redirect logged in user to a specific web page.

6.1.5 Permissions

Roles

WordPress uses a concept of Roles, which puts the users into different user-groups. These roles have different capabilities, controlling what a user can and can not do on the site. What capabilities each role has is decided by the site owner.

The declaration of roles suits PEMA nicely, as each user naturally falls under one of three roles: administrator, instructor or student. Of these, only the administrator role is included natively with WordPress. The instructor and student roles, as well as any potential future roles, need to be created manually. WP provides functions for role-creation, making the process simple.

The function `add_role()`; is used for creating new roles [67]. It adds a role name, display name for the role, as well as an array of capabilities. An example of this can be seen in listing 6.5.

```

1  add_role(          // Adding student role.
2   'student',      // Role name.
3   'Student',     // Display name.
4   array(         // Array of capabilities.
5   'pema_see_lab' => true // Allows students to see labs.
6  ));

```

Listing 6.5: Adding a role to WordPress.

By default, WP includes six user roles. Super Admin, administrator, editor, author, contributor and subscriber [68]. These roles are clearly angled towards the most common use of WordPress, a blog. For PEMA, some of these roles are not needed, and are therefore removed. The removal of roles is done through a function which is called when the theme is activated.

```

1 // Gets array of WordPress roles.
2 global $wp_roles;
3
4 // Create an array of roles to remove.
5 $roles_to_remove = array(
6     'subscriber',
7     'contributor',
8     'author',
9     'editor'
10 );
11
12 // For every element in array.
13 foreach ( $roles_to_remove as $role ) {
14     // If role exists.
15     if ( isset($wp_roles->roles[$role]) ) {
16         // Remove role.
17         $wp_roles->remove_role($role);
18     }
19 }

```

Listing 6.6: Function removing unused default WordPress roles.

The function in listing 6.6 gathers the array of WP roles from the database. Next, an array of the roles that are to be removed is created. And lastly, a foreach loop goes through the array and removes the roles one by one. Much like when adding roles, this is easily done through a WP function, `remove_role()`; [69].

After removing default roles and adding custom ones, PEMA is left with the following hierarchy:

- *Administrator*: Can utilize all features in PEMA, and has full WordPress management capabilities.
- *Instructor*: Has limited access to the admin-dashboard. Can manage labs and all its content, as well as some user management.
- *Student*: Can view labs, submit answers to tasks and has some limited VM management.

Naturally, like other functions that make changes to the default WordPress behaviour, the function removing has a counterpart that is ran when the theme is disabled.

```

1 /* --- RESETS/REPOPULATES DEFAULT ROLES W/ DEFAULT CAPABILITIES --- */
2
3 // Checks if function exists already.
4 if ( !function_exists( 'populate_roles' ) ) {
5     // If it does not, gets it from a default wordpress file.
6     require_once( ABSPATH . 'wp-admin/includes/schema.php' );
7 }
8 // Runs function, repopulates default roles.
9 populate_roles();

```

Listing 6.7: Repopulates the default WordPress roles.

This function checks if the default WP function `populate_roles()`; is present, if it is not, it gets it from a default WP file. Then runs the function, resetting the WP roles back to default.

Capabilities

As previously stated, WordPress uses capabilities to decide what a role can and can not do. Although WP has an abundance of capabilities included in the core [70], it still allows for the creation of custom capabilities. Because PEMA adds to the core, for example by adding pages to the admin-dashboard, custom capabilities to access and use the features of these pages was a natural choice.

Adding capabilities to WordPress is simple, it is done by simply giving it to any user role or roles. An example of this is the capability `pema_see_lab` added to the Student role in listing 6.5. The capability can then be required to access a page, or checked before an action is executed.

PEMA uses these custom capabilities in combination with native WP capabilities. For example an instructor needs access to the admin-dashboard, so the instructor role is given the `read` capability. This capability only allows the instructor to access the "Dashboard" tab in the admin-dashboard, and will need further capabilities to access other functions and sub-pages.

Another capability given to instructors is the ability to promote other users to student, which is required to create student accounts, as well as viewing all users except administrator accounts, and deleting student users. The WP capability given to the instructor to allow for this functionality gives the user too much power. The capability is `promote_users`, which gives full ability to promote and demote users. Without any other restrictions, the instructor can set any account, including their own as administrator.

To restrict this, a function that removes the option to promote users to either administrator or instructor is used. This is done by calling the function through a hook when the promotion-list is called. The function then strips away the administrator and instructor options from the appropriate array before returning the remaining options.

```

1  /* --- Removes an Instructors ability to add/promote (to) admin and
   instructor accounts --- */
2  function pema_editable_users_filter( $roles ) {
3      // Gets current user information from database.
4      $current_user = wp_get_current_user();
5
6      // Checks if current user is an instructor.
7      if ( in_array( 'instructor', $current_user->roles ) ) {
8          // Unsets administrator and instructor from editable_roles array.
9          unset( $roles['administrator'] );
10         unset( $roles['instructor'] );
11     }
12     // Returns array $roles.
13     return $roles;
14 }
15 add_filter( 'editable_roles', 'pema_editable_users_filter' ); // Function

```

```
call via hook.
```

Listing 6.8: Function removing instructors ability to promote to administrator or instructor.

These types of functions are also used to remove administrator accounts from the user-list entirely for anyone except other administrators, and to block instructors from deleting administrator accounts, in the case that they are clever enough to find a way to do it outside of the GUI. Having the ability to customize everything, with even higher granularity than just the permissions themselves is particularly useful.

For a complete list of the permissions used in PEMA, refer to appendix C.

6.2 Backend

Some security measures must be placed on all levels of operation, one of which is the backend. In the backend, on a general basis, one has the option of blocking vulnerabilities, or rather opening whatever is needed for the service to function properly. In addition, some services live in the backend to serve as another layer of protection, one such service our https reverse-proxy.

6.2.1 HTTPS

One of the functional requirements for PEMA is that whatever information is being transmitted at any given moment shall be encrypted, to ensure no bad actor may "sniff" information while in transit. To do this, one must implement some sort of encryption-layer upon transmitting data, and the most natural method of doing this is https. HTTPS as a protocol enhances the traditional Hyper Text Transfer Protocol (HTTP), by using TLS (or its predecessor SSL) to encrypt the data before sending. To enable HTTPS, one can go one of many routes, and in our case, an nginx reverse-proxy is used (also known as https termination), where the communication between the nginx-server and the end user is transmitted using https, while the transmission between the proxy and the webserver itself is http. This decision was made based on the fact that the website should be able to scale up or down based on the number of end-users using the site at any one time. By using a proxy, it's possible to point out to the proxy where to send the unencrypted data when sending to the webserver using one simple configuration-file, instead of setting up HTTPS on all replicas of the website, simplifying initial design and configuration. By decrypting data before it enters the webserver, in theory, the security-measures taken are made extraneous, however our choice of using docker swarm enables us to confidently say that the data is still secure.

6.2.2 Docker swarm

Docker swarm is a technology which enables multiple host-machines to share their processing-capacity to run the micro-services defined in a given docker stack, in this case the multiple webserver, database and nginx reverse-proxy. Docker itself sets up a network between all nodes in a swarm, and all traffic headed to any service on said network needs to originate from that same network [71] unless specified. To enable outside traffic to enter a swarm, a port

must be opened, this is done by specifying a source and destination port in the `docker_compose`-file. All traffic inside the swarm-network is allowed, and as such, there is no need of specifying as an example what IP-adresses may access the database-service, as it is impossible to reach by any outside traffic. We also have no need to specify some other port than the default 3306 on the database on which queries may be sent as, again, the database can only be reached by services inside the swarm-network. The service which makes it possible for an end-user to access the website is the reverse-proxy, as this service has been configured such that its ports 80 and 443 are exposed to "the outside world". When an HTTP-request is made on port 80, it is forwarded such that it becomes an HTTPS-request, forcing the end-user-machine to make a new request, this time on port 443, which forwards the traffic to port 80 on whatever node inside the swarm is ready to handle the request. This ensures that only requests on port 80 and 443 are allowed, and these will be directed first to the nginx-proxy, then to the apache webserver running wordpress.

7 Deployment

PEMA is developed with simple replication and further development in mind. As a result of this, PEMA can be deployed with docker using docker swarm. A visualization of this can be seen in figure 17.

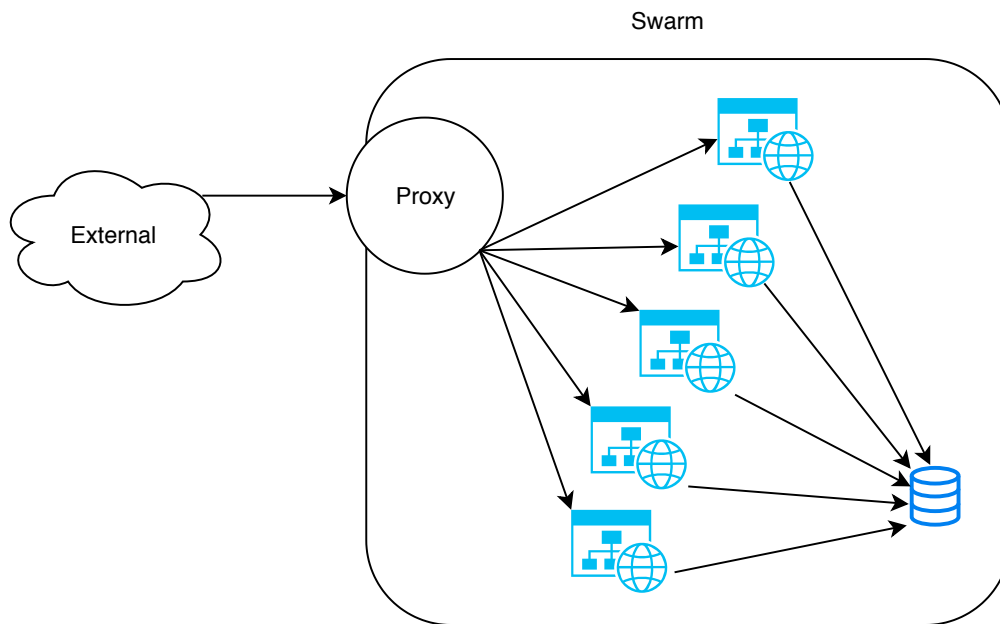


Figure 17: Visualization of the software relationship.

In order to make such a deployment, these steps must be taken:

- On the server(s) where the platform will be deployed, install docker and docker-compose.
- Initialize a docker swarm, and join whatever extra nodes will take part.
- Clone the repository currently hosted on the NCR internal git-repositories and move into that directory.
- Build the proxy-image using the build-script found in the "scripts" directory.
- Generate the environment-variables using the "genenv"-script, and then populate missing values in the ".env"-file that was made.
- Deploy the stack using the "stack_deploy"-script.

Once these steps are taken, the website will be active and accessible on the domain and/or IP-address specified in the "WP_DOMAIN"-variable.

Files that are relevant to the functionality of the website itself can be found in the sub-directories "Pema-Theme" and "plugins" under the "site"-directory.

These directories are what the website-replications you can see in the illustration above access and serve, as these directories are passed as docker-volumes to the instances. To further develop the website, split the git-repository into a master and dev-branch, and use the dev-branch for testing, and merge into master when it is ready. Once the master is updated, simply pull the changes on the docker swarm-master, the machine where the deployment was initially handled. The changes will immediately take effect, as a result of being a docker volume mounted on all instances of the website.

8 Testing and User Feedback

8.1 Purpose

The purpose of testing is to find weaknesses in the application, determine its source and then make the application more resilient. The test will ensure the quality of the application and document what steps were taken in order to meet and verify the requirements of the employer.

8.2 Testing Scope

The testing scope for the PEMA application is rather extensive. When the code is written, each individual functionality must be reviewed by at least one other member of the group before it is considered complete. All the code will be subject to both static and dynamic analysis near the end of the project period. A select few core PHP functions will be subject to fuzz testing during the project period. Finally, some often used, and security critical WordPress and PEMA functions will be subject to manual code review.

8.3 Static Analysis

The code in the PEMA project has not been subject to static analysis due to incompatibility between WordPress and mainstream PHP static analysis tools. The reasons why are discussed in chapter [9.3.1](#).

8.4 Dynamic Testing

The final deployment PEMA web application is dynamically tested with OWASP ZAP [[72](#)]. The final version has 9 alerts in a total of 67 places. All the medium and high-risk alerts are none exploitable and most likely false positives.

This is most evident in the SQL injection alerts in the robots.txt and sitemap.xml files. Two files that do not have any database query in them and can therefore not be exposed to any SQL vulnerabilities. This conclusion is also tested and gave the predicted result.

All remaining true positives are risks that have low impact and are deemed acceptable so they will not be addressed.

8.5 Fuzz Testing

Fundamentally important PHP functions are tested using American Fuzzy Lop in persistent mode with fuzzing dictionaries. The tested functions are `decode_json()` and `htmlentities()`. Neither of these functions show any kind of weakness after extensive nonstop testing over almost two months.

The function `decode_json` is the most thoroughly tested due to its complexity and extended usage with data from external sources (Mainly PLED). `htmlentities()` is mainly used when there is user input and output based on user input. It is used continuously throughout the application and any fault to it could be problematic to the function of PEMA. The results of the fuzzing can be seen in figure 18.

```

american fuzzy lop 2.52b (master)
-----
process timing | overall results
  run time : 48 days, 18 hrs, 27 min, 41 sec | cycles done : 4040
  last new path : 1 days, 18 hrs, 28 min, 18 sec | total paths : 1813
  last uniq crash : none seen yet | uniq crashes : 0
  last uniq hang : none seen yet | uniq hangs : 0
-----
cycle progress | map coverage
  now processing : 357 (19.69%) | map density : 1.78% / 4.58%
  paths timed out : 0 (0.00%) | count coverage : 4.82 bits/tuple
-----
stage progress | findings in depth
  now trying : splice 14 | favored paths : 204 (11.25%)
  stage execs : 39/48 (81.25%) | new edges on : 325 (17.93%)
  total execs : 2.21G | total crashes : 0 (0 unique)
  exec speed : 2035/sec | total tmouts : 4.62M (211 unique)
-----
fuzzing strategy yields | path geometry
  bit flips : 15/36.1M, 6/36.1M, 7/36.1M | levels : 4
  byte flips : 0/4.51M, 0/3.45M, 0/3.45M | pending : 0
  arithmetics : 6/192M, 0/638k, 0/7318 | pend fav : 0
  known ints : 3/18.7M, 1/96.6M, 0/151M | own finds : 214
  dictionary : 14/126M, 14/166M, 2/170M | imported : 1596
  havoc : 146/413M, 0/755M | stability : 91.54%
  trim : 3.44%/508k, 23.44%
-----
[cpu005:119%]

```

Figure 18: View of the master thread in the fuzzing process after 48 days

It is important to note that the function `decode_json()` is directly present in the WordPress core code. This means that PEMA only references this function indirectly and is not written anywhere in our source code, but it is in use if you review the WordPress core.

8.6 Code Review

All parts of our written code have been subject to code review either by one or multiple peers. Central parts of the code mainly functions regarding database connectivity and user interactions. The main effect of doing peer code review is that the quality of the code increased with little effort. The extra code review had the effect of ensuring the code quality in critical areas.

9 Discussion

9.1 Tools

9.1.1 Writing tools

Some group members started writing code using the tool Notepad++. This tool had been used for earlier projects, but not for a project of this scale.

The two other text-editors used by the group were Sublime Text and Atom. After trying these options, these were clearly more suited for a large-scale project like this one. With features like native language support, tab-completion and Git-integration, as well as a prettier interface, changing over was easy.

The group ended up with two Sublime users and two Atom users.

9.1.2 Task Management

Early in the project there was an interest in using Todoist for task management. Todoist is an application used to manage tasks in the form of to-do lists [73]. It has support for projects, but only one to-do list per project. Although it does have some other nice features, such as priority-sorting and calendar integration, we still decided to go with Trello, which is a more familiar platform with the ability to have several boards per project and different stages for each task, which is vital in a project of this scale.

9.2 Results

9.2.1 Project Outcome

The outcome of the project is that we have created a fully functional web platform, where an instructor can create descriptive labs and tasks for students. We've also created a proof of concept with the implementation of the PLED database, as well as with the OpenStack plugin in regards to that one actually can launch instances using the OpenStack API.

9.2.2 Unfulfilled Requirements

There are a lot of modules that we didn't have time to implement, such as a forum or scoreboard. However, these were not really feasible from the beginning as we already had a lot to learn and develop. We did not have time to implement a way to delete, create or update different 'Answer Types' that could be used when creating a task. There was not enough time for us, and we had other more pressing matters that needed to be done. A scoring system was also not implemented, the reason for this is that it will take a lot of time to develop a good system for this, however, it can be implemented at a future date using plugins with custom action hooks. Since a scoring system was not implemented, there is also no cost of using hints for students.

9.2.3 Alternative Solutions

We should have used Scrum with sprints instead of Kanban, this is discussed in subsection 9.3.2.

9.2.4 For Future Implementation

Student Assistant Role

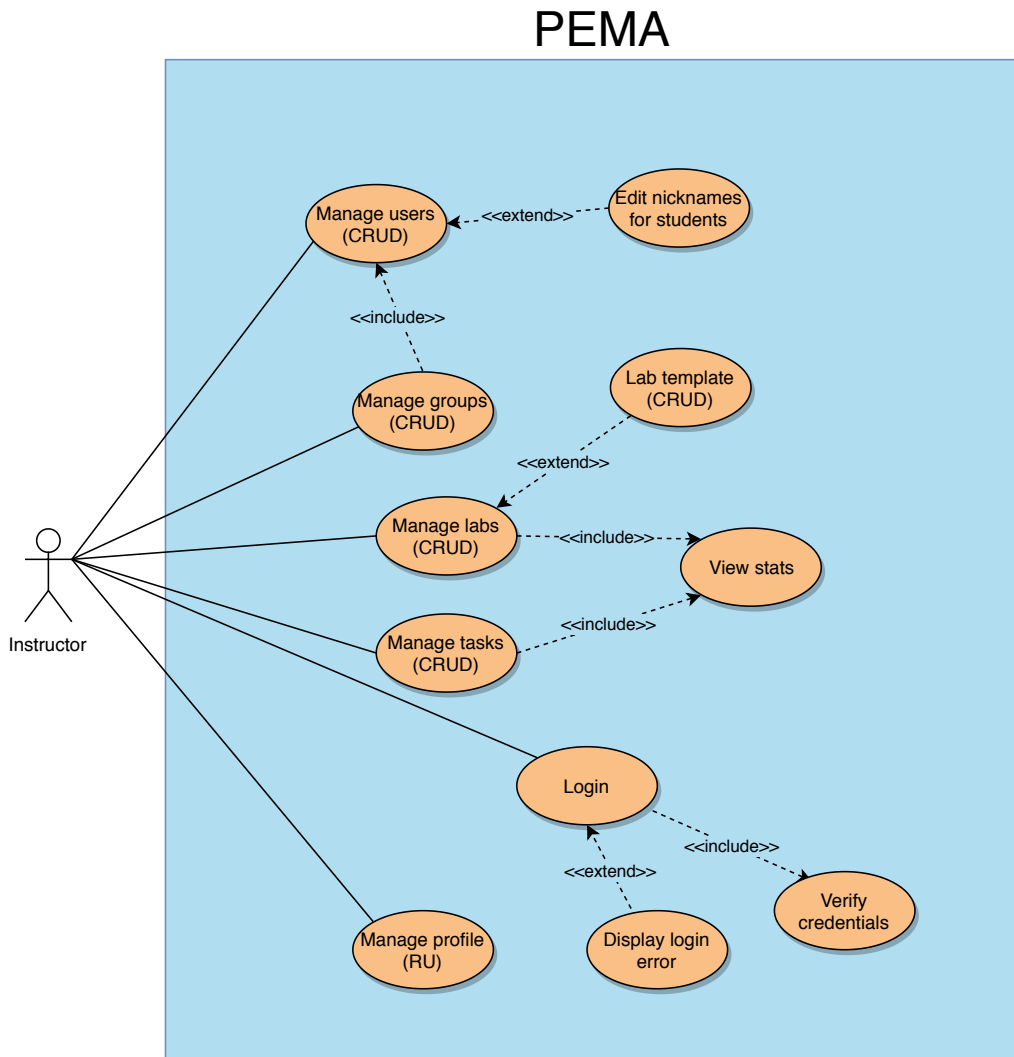


Figure 19: Student assistant use-case diagram.

Figure 19 is a use-case diagram made for the potential role of student-assistant. This role was suggested by our group as a potential new role for PEMA. The main intention for this role was to allow student assistants to help manage the forum, as well as similar tasks to assist students. The role was never added, as the forum module did not get implemented.

We still think this is a good idea to have in mind for the future.

9.2.5 Schedule

In the pre-project phase, we created a gantt-schema, this can be found in appendix [F.11](#). This schema outlined how we believed the timeline of the project would look. However, the schema turned out to be somewhat incorrect.

During front-end development, we decided to focus on a solid "core", that can be further built on in the future. This meant that the development of the modules was pushed back, as the prioritization leaned more towards getting core functionality in place.

We moved away from automatic testing due to incompatibility issues discussed in [9.3.1](#).

During the docker-development, among the larger time-sinks was getting the replications of the site to cooperate properly. Upon setting up TLS termination using a reverse proxy, we learned that the WordPress docker-image requires some environment variables to be set in order to properly configure cookies behind HTTPS. This is discussed further in section [9.3.3](#).

A minor change of the entire project, was that we initially thought the deadline was the 15th of May, not the 20th. The only change this inflicted on the project was giving us a little more time, allocated for the final part of the project.

How the timeline of the project actually turned out can be seen in a new gantt schema, found in appendix [F.12](#).

9.3 Complications

9.3.1 Static Analysis

Wordpress does not lend itself well to static analysis with traditional tools like PHPStan. There exist some unofficial WordPress extensions for PHPStan and a few standalone analyzers, but they do not meet the reliability standards that we would like to see when we are developing this application. Ideally we would like to see the developers of PHPStan add support for this, but this quote[[74](#)] from Ondřej Mirtes in 2017 does not give much hope for a quick resolution:

PHPStan is not yet ready for analysing code that mixes declarations and side effects. It works well on mature, object-oriented, codebases, and WordPress is not one of those. But I'm keeping this in mind and have ideas how to improve PHPStan to work on anything.

The alternative to this would then be to look at other static analysis tools. Psalm [[75](#)] looks like the best alternative to PHPStan when it comes to analyzing WordPress. However, as the article mentions Psalm is not without flaws. It has, amongst other things, a 5% false positive rate. Some of these false positives come from the way WordPress DB queries work, which is fundamental to the way we have constructed the application.

The 5% false positive rate also creates problems for your Continuous Integration workflow. With a code base this massive, a 5% false positive rate means that few of the commits will pass the automated testing.

This is not a scenario that we predicted in our risk analysis, which can be found in Chapter 5.2 in appendix G. Continuous Integration is not something that is necessary for our development so the solution to this problem became that we decided to up the effort when it came to code review and dynamic analysis. This has the cost of some of the problems evident in the commit log could have been avoided and solving these problems took some more time.

9.3.2 Development methodology

Upon starting this project, the chosen methodology of kanban was chosen as the freedom of working freely on differing parts as fit whenever a dependency upon differing parts of the website are discovered. During the development however, our employer highlighted new functionality and definitions of how PEMA should work. This made following a schedule problematic, as changes in the amount of work results in a need for more time or needing to rush existing requirements. Had we chosen to follow the scrum-model, where the development is done in predefined sprints, allocating time might have proven beneficial.

9.3.3 HTTPS

When implementing https, there were some complications which took some time to overcome. Firstly, in order to be valid, one needs a certificate, which is also valid. To get a certificate, one needs to either get one signed by a ticket authority, or create one which is self-signed. When setting up TLS in this infrastructure, we used a self-signed certificate, as the website lives on the internal NTNU-network, and as such does not have a domain associated with it, and is also hosted on the 10.0.0.0-subnet, which is only reachable to internal clients. This means any new connection made to the web-interface is reported by web-browsers as insecure, as they by default will warn its users of such certificates.

In addition there was an issue with logging in to the webplatform once HTTPS was implemented. Upon first activating TLS, users were unable to log in to the platform, and upon further inspection, it seemed like session-cookies were improperly set when logging in, setting them to expire immediately. Initial troubleshooting pointed towards this being an issue with how the reverse proxy interacted with the replicas of the website, or with the website configuration itself. After some more troubleshooting, this was debunked, as our problems were not consistent with what others on various forums and in github-issues were reporting. Later, we found a github-issue where someone had a much more similar issue, including a possible solution, which after testing turned out to be consistent with us. The issue being that when deploying the wordpress docker-images, there are optional values which can be set as salts and keys for various authentication purposes. When not supplied, all replicas of the image will generate their own values, and these are what ended up being the issue. After first testing with a single replica, this resulting in a functioning login on the website, and then generating our own values for these variables, we concluded this was the issue, and promptly implemented

generation of these values as part of the initial set-up process.

9.3.4 Linking to pages in WordPress

In the very early stages of the project, we tried to divide content across pages. We were used to just include pages with the `require('page-name.php')` function, however this was not a good practice with WordPress.

The solution to this was to utilize WordPress Pages [76]. You can either manually create new pages through the admin dashboard, or you can create them with the function `wp_insert_post` [77]. WordPress uses the same function for creating pages and posts, this was at first confusing for us as this function seemed to only create posts at first glance. In order to create a page with the function above, we would have to define the variable `'post_type'` to be `'page'` instead of the default `'post'`. This was a page would be created instead of a post. However, we would then have to manually create a file called `'page-nameOfPage.php'` and this is not really doable as we can't predict the names of the labs or tasks. This is why we chose to use template files for pages.

9.3.5 jQuery data()

The `jQuery.data()` function was intended to grab information from a range of different options in a select element. However, the problem is that `jQuery.data()` function doesn't manipulate the DOM, it instead stores the data in memory and doesn't update when another option is selected [78].

The solution to this was to first find the option that was selected and then get the appropriate data attribute. An example of this is depicted below.

```
1 var data = $(this).find("option:selected").data("data");
```

Listing 9.1: Example of getting data attribute in an select element

9.3.6 jQuery Plugins

Countdown Plugin for Time Left

Something we wanted to use a jQuery plugin for was displaying how much time was left on tasks and labs. We figured that doing this with a jQuery plugin would make for a pretty looking solution that actively counts down, rather than just displaying a static number when the page loads.

After a bit of searching, we found two promising plugins. `jQuery.countdown` [79] and `FlipClock.js` [80]. Both plugins offered different styles of countdown-timers, as well as varying degrees of customization.

We decided to try implementing `FlipClock.js` first. Trying to implement the feature to our `TaskView` page, we ran into some strange bugs, where the plugin did not want to display on our page. After troubleshooting for a bit, and discovering very little community support around the plugin, making it difficult to find a solution, we decided to move on to `jQuery.countdown`.

`jQuery.countdown` had less documentation to start with, and much like `FlipClock` was not updated for a few years, but we managed to get it to display it on our page. However, when trying to input a timestamp from our database, to display time left until that time, the plugin refused to display any timer.

After trying to translate the default MySQL timestamp format into various formats that were supposedly supported, even going as far as trying unix-time, the plugin still would not work.

At this point, we decided to abandon the implementation of a counter, as it was mostly aesthetic, and we did not want to waste any more time troubleshooting or looking for alternative plugins.

9.3.7 WordPress plugins

Third-party plugins

When deploying the website on the ntnu-cloud, we decided to attempt downloading third-party plugins, which failed. At first, the site asked for credentials for an FTP-server, which we have not set up. Changing the behaviour of WordPress allowed us to access local files, which also failed, due to improper permissions. When changing permissions for both the containing directory, as well as the files, the transactions still failed, however scaling the service such that there is only one replica of the website, and then changing permissions within that container allows us to download, alter and delete plugins at will. This solution is, however, not a permanent and good solution, as it takes many steps in addition to needing to scale down the service to do so.

A possible solution to this would be to investigate what it would take to set up said FTP-server, and allow this to host all plugins. Another solution is figuring out what ownership and permissions are needed to make a volumized solution viable. A third option is to not allow for multiple copies of the site as replicas, however this defeats the purpose of the docker swarm, and would necessitate a total redesign of the infrastructure. Finally, it would also be possible to install such that they are included in the repository, as self-made and otherwise already included plugins seem to be fully functional, however they are not possible to update. Adding plugins to the repository is a bad solution, as it is not modular, meaning it is difficult to add and remove plugins, in addition, it contradicts our points on plugin vulnerability in chapter 3.2.2, seeing as it is difficult to update said plugins.

9.4 Evaluation

9.4.1 Carrying Out The Project

This project was complex from the start. The coordination between three different theses, one master and two bachelors, proved to create some difficulties. Originally, we were supposed to implement the Domain Specific Language in our part of the PEMA platform, but it became evident that it would not be near ready for implementation in our project timeframe. This made our part of the project complex and we had to follow our mitigation plan from the project risk analysis, which can be found in Chapter 5.2 in appendix G. We would feel a lot better about finishing this project if the DSL was complete as it serves as an extremely important connection with PLED. It was intended that PLED was going to provide the DSL with information that PEMA would pass through and deliver. If we had the opportunity to facilitate this exchange

it would make the whole application, make a lot more sense.

The final product that we are delivering feels slightly incomplete, but it is also the intention. This project was meant to be the first iteration of an ongoing project of the NCR. This is the first time that we are aware of where this type of coordination between theses is attempted and the noncompletion of the other theses is something that we expected. We did what we could with the situation, and we believe that there is nothing more that can be done in order to facilitate the takeover of the project after we hand in the thesis, from a coding standpoint.

Even if the coordination between the different theses was sub optimal, we are grateful for the cooperation that the NCR has showed us. We would especially thank Danny for his dedication to this project and the guidance he gave us. There is no doubt to us that he had every intention to make this first iteration of the platform as good as possible.

9.4.2 Group Evaluation

When dividing work between group members, we chose to apply tasks according to the different group members' strengths and/or wishes. This was done so that we could each have an area of focus, allowing team members to more effectively gain a deep understanding of the differing parts of the project. Allowing each member to do this has proven effective in allowing us to gain a deep understanding and to the greatest degree possible perfect each individual part of the project. This system has worked quite well, and has had the intended effect. However, as we are aware, there is one instance where this splitting of the group led to a problem taking longer to solve than what would be the case had we had tighter cooperation across subjects on this project. When setting up HTTPS, cookies were suddenly no longer set correctly, making logging in to the platform impossible, as cookies seemingly were expiring immediately. This turned out to be an issue where replications of the webserver led to certain hash-values called nonces used for validation being set differently between webserver, nonces are explained in more detail in section 6.1.3. The problem would have been found earlier had Sondre, who set up HTTPS, known this fact about WordPress.

As stated, the group members have had different roles based on our initial division of tasks, however this does not mean we have all been totally focused on said tasks. Whenever needed and possible, each member have been active on any given problem, and given all relevant help whenever needed. As an example, when programming the logic for filtering results from the PLED API, Tobias had problems implementing the logic properly. Sondre, who worked mostly on backend and Docker, in addition to Erlend who, like Tobias, was mostly focused on programming and web-development, helped perfect said logic.

Distribution of Labour

The distribution of work throughout the different parts of the project have been handled pretty automatically. We have had different types of works, or

bigger tasks that needed to be done. The process for deciding who does what starts by asking if anyone has any preferences, if that does not result in a good distribution, the remaining tasks are distributed by the group leader.

For the largest part of the project duration, the roles were split in two main groups. Two WordPress / frontend developers, and two working with the backend and testing. The two sub-groups have then managed labour distribution among eachother.

We have had continuous development, using Trello to pick new tasks whenever the previous one is completed. Decisions about what tasks to prioritize have been discussed at the weekly meetings. In these meetings new tasks have continuously been added and removed, as well as the supervisor and employer giving their opinions on where our focus should be.

An example of the group following a suggestion from the supervisor, is the entire group having more focus on writing the report, as well as having one person shift their focus from backend to primarily report.

10 Conclusion

The NCR with Danny Lopez and Basel Katt tasked us to create a modular, scalable and virtualization agnostic platform that could facilitate the deployment of virtual scenarios for cyber security education and research purposes. The PEMA application that we have created fulfills these requirements to the best of our abilities with the amount of time we had for this project. The framework that we have chosen makes the planned expansion on this program possible and any requirements that we could not fulfill can be added later and expanded upon. This modularity makes the PEMA platform able to be picked up by other developers, as intended by the NCR.

The most important addition to the project that we contributed was designing how the application was to be deployed. By designing the platform as a stack of instances in a docker swarm, we simplify the deployment by reducing the amount of installed software needed to deploy the website. In addition, this design-choice lends itself well to use a reverse proxy for the implementation of HTTPS, as traffic between the proxy and web servers are protected by being behind a network only reachable by nodes inside the swarm. This choice also allows further development to easily be deployed once ready for the production environment, as the web servers simply read directly from the git-repository which hosts the necessary files. In short, the choice of docker massively simplifies our development of an easily usable self-hosted platform for any third party.

Our task in starting this project has been to develop a platform which will allow instructors and others to organize the execution of pen-testing scenarios. From the onset, our task has been to begin development knowing the platform will be continually updated after this bachelor-project is over. In addition to developing the task, one requirement has been to create the platform such that it is easily utilized by those who might have an interest in using and/or working on the platform. To do this, we have developed scripts and procedures to allow interested parties to easily download, work on and use the platform using Docker.

10.1 Future Work

Going forward, there are some features of the platform specified in the total platform specification which are not implemented, which should be. PEMA should implement the DSL that is being made to easily deploy vulnerable infrastructures. Applying the DSL and the PLED API once ready are tasks which would change PEMA from a platform in which students may get their Kali-machine IP and submit answers to tasks, to being a fully fledged platform which automatically deploy all relevant machines. In addition, the specification outlines PEMA as a platform for both penetration testing-labs and CTF

events, in addition malware analysis-labs have been mentioned by our employer. The current iteration of PEMA implements the penetration testing-labs, and support for CTF events and malware analysis have been outlined, yet would need some work to be fully functioning.

As for the graphical design, the platform has not been designed with this as a point of interest. There have been considerations for future implementations of new design, where someone who is skilled within design, and has CSS knowledge should not have a hard time making the site look good. This can be done with creating a Child-Theme of the PEMA theme [81]. What this essentially will do is change some aspects of the page, like the CSS, while still maintaining the functionality of the theme. Our focus has been on designing and implementing the functionality of the site, and from a design point simply making it usable.

Bibliography

- [1] Wordfence. How attackers gain access to wordpress sites. <https://www.wordfence.com/blog/2016/03/attackers-gain-access-wordpress-sites/>. (Visited February 2019).
- [2] National Institute of Standards and Technology. Cyber ranges. https://www.nist.gov/sites/default/files/documents/2018/02/13/cyber_ranges.pdf. (Visited May 2019).
- [3] Phabricator. Introduction. <https://secure.phabricator.com/book/phabricator/article/introduction/>. (Visited May 2019).
- [4] Oxford University. Oxford english dictionary, proof of concept. https://en.oxforddictionaries.com/definition/proof_of_concept. (Visited May 2019).
- [5] Misja.com. Epoch & unix timestamp conversion tools, what is epoch time? <https://www.epochconverter.com/>. (Visited May 2019).
- [6] Wikipedia. Little's law. https://en.wikipedia.org/wiki/Little's_law. (Visited January 2019).
- [7] Moe Long. 6 easiest programming languages to learn for beginners. <https://www.makeuseof.com/tag/easiest-programming-languages-beginners/>. (Visited May 2019).
- [8] Techspot. Python tops java as most popular introductory teaching language among us universities. <https://www.techspot.com/news/57345-python-tops-java-as-most-popular-introductory-teaching-language.html>. (Visited February 2019).
- [9] Djangostars. Why we use django framework & what is django used for. <https://djangostars.com/blog/why-we-use-django-framework/>. (Visited February 2019).
- [10] Hackernoon Proximity Costa Rica. What is vue.js and what are its advantages. <https://hackernoon.com/what-is-vue-js-and-what-are-its-advantages-4071b7c7993d>. (Visited February 2019).
- [11] Vuejsdevelopers Anthony Gore. 7 vue.js backends compared. <https://vuejsdevelopers.com/2018/05/07/vue-js-backends-express-laravel-firebase-wordpress-django-rails/>. (Visited February 2019).

-
- [12] Moodle. About moodle. https://docs.moodle.org/36/en/About_Moodle. (Visited February 2019).
- [13] Moodle. Moodle statistics. <https://moodle.net/stats/?lang=nn>. (Visited February 2019).
- [14] Wordpress. Theme directory. <https://wordpress.org/themes/browse/new/>. (Visited February 2019).
- [15] Wordpress. Plugins. <https://wordpress.org/plugins/>. (Visited February 2019).
- [16] WPScan. Wpscan vulnerability database. <https://wpscan.com/>. (Visited February 2019).
- [17] FilaThemes. Education lms. <https://wordpress.org/themes/education-lms/>. (Visited April 2019).
- [18] WordPress. Template hierarchy. <https://wphierarchy.com/>. (Visited March 2019).
- [19] Wordpress. Class reference/wpdb. https://codex.wordpress.org/Class_Reference/wpdb. (Visited February 2019).
- [20] Justin Vincent. ezsql. <http://justinvincent.com/ezsql>. (Visited April 2019).
- [21] WordPress. Wordpress coding standards. https://codex.wordpress.org/WordPress_Coding_Standards. (Visited March 2019).
- [22] WordPress. Server side php and enqueueing. <https://developer.wordpress.org/plugins/javascript/enqueueing/>. (Visited February 2019).
- [23] WordPress. Code reference wp_enqueue_script. https://developer.wordpress.org/reference/functions/wp_enqueue_script/. (Visited April 2019).
- [24] WordPress. Code reference wp_register_script. https://developer.wordpress.org/reference/functions/wp_register_script/. (Visited February 2019).
- [25] WordPress. Plugin repository. <https://wordpress.org/plugins/>. (Visited April 2019).
- [26] WPBeginner. What is: Plugins. <https://www.wpbeginner.com/glossary/plugin/>. (Visited April 2019).
- [27] WordPress. Wp forum plugin results. <https://wordpress.org/plugins/search/forum/>. (Visited April 2019).
- [28] WordPress. Plugin api. https://codex.wordpress.org/Plugin_API. (Visited March 2019).

- [29] WordPress. Javascript. <https://developer.wordpress.org/plugins/javascript/>. (Visited April 2019).
- [30] jQuery. Plugins. <https://learn.jquery.com/plugins/>. (Visited April 2019).
- [31] WordPress. Wordpress requirements. <https://wordpress.org/about/requirements/>. (Visited March 2019).
- [32] netcraft. November 2018 web server survey. <https://news.netcraft.com/archives/2018/11/26/november-2018-web-server-survey.html>. (Visited March 2019).
- [33] Apache. Apache http server version 2.4 documentation. <http://httpd.apache.org/docs/current/>. (Visited March 2019).
- [34] NET MARKETSHARE. Operating system market share. <https://netmarketshare.com/operating-system-market-share.aspx?options=%7B%22filter%22%3A%7B%22%24and%22%3A%5B%7B%22deviceType%22%3A%7B%22%24in%22%3A%5B%22Desktop%22%5D%7D%7D%5D%7D%2C%22dateLabel%22%3A%22Custom%22%2C%22attributes%22%3A%22share%22%2C%22group%22%3A%22platform%22%2C%22sort%22%3A%7B%22share%22%3A-1%7D%2C%22id%22%3A%22platformsDesktop%22%2C%22dateInterval%22%3A%22Monthly%22%2C%22dateStart%22%3A%222018-02%22%2C%22dateEnd%22%3A%222019-02%22%2C%22hiddenSeries%22%3A%7B%7D%2C%22segments%22%3A%22-1000%22%7D>. (Visited March 2019).
- [35] Statista. Global market share held by operating systems for desktop pcs, from january 2013 to january 2019. <https://www.statista.com/statistics/218089/global-market-share-of-windows-7/>. (Visited March 2019).
- [36] W3Techs. Usage statistics and market share of windows for websites. <https://w3techs.com/technologies/details/os-windows/all/all>. (Visited March 2019).
- [37] W3Techs. Usage of operating systems for websites. https://w3techs.com/technologies/overview/operating_system/all. (Visited March 2019).
- [38] TeamSpeak. Features. <https://www.teamspeak.com/en/features/overview/>. (Visited May 2019).
- [39] Margaret Rouse. Advanced encryption standard (aes). <https://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard>. (Visited May 2019).
- [40] Google. Security. <https://support.google.com/googlecloud/answer/6056693?hl=en>. (Visited January, May 2019).

-
- [41] Overleaf. Is my information secure on overleaf? https://www.overleaf.com/learn/how-to/Is_my_information_secure_on_Overleaf%3F. (Visited January, May 2019).
- [42] Amazon. Amazon s3 default encryption for s3 buckets. <https://docs.aws.amazon.com/AmazonS3/latest/dev/bucket-encryption.html>. (Visited May 2019).
- [43] Apache Friends. Xampp apache + mariadb + php + perl. <https://www.apachefriends.org/index.html>. (Visited January 2019).
- [44] Colin Newcomer. How to install xampp and wordpress locally on windows pc. <https://themeisle.com/blog/install-xampp-and-wordpress-locally/>. (Visited January 2019).
- [45] Docker. Get docker ce for ubuntu. <https://docs.docker.com/install/linux/docker-ce/ubuntu/>. (Visited April 2019).
- [46] Docker. Add nodes to the swarm. <https://docs.docker.com/engine/swarm/swarm-tutorial/add-nodes/>. (Visited April 2019).
- [47] WordPress. Organizing theme files. <https://developer.wordpress.org/themes/basics/organizing-theme-files/>. (Visited March 2019).
- [48] WordPress. The loop. https://codex.wordpress.org/The_Loop. (Visited March 2019).
- [49] WordPress. Class reference/wp post. https://codex.wordpress.org/Class_Reference/WP_Post. (Visited April 2019).
- [50] w3schools. Php glob() function. <https://www.php.net/manual/en/function.glob.php>. (Visited May 2019).
- [51] @flatpickr. flatpickr v4. <https://flatpickr.js.org/>. (Visited March/April 2019).
- [52] GitHub user: craftpip. jquery-confirm. <https://craftpip.github.io/jquery-confirm/>.
- [53] WordPress. Cron. <https://developer.wordpress.org/plugins/cron/>. (Visited April 2019).
- [54] WordPress. Http api. https://codex.wordpress.org/HTTP_API. (Visited March 2019).
- [55] WordPress. Function reference/wp remote get. https://codex.wordpress.org/Function_Reference/wp_remote_get. (Visited March 2019).

- [56] WordPress. Function reference/wp remote retrieve body. https://codex.wordpress.org/Function_Reference/wp_remote_retrieve_body. (Visited March 2019).
- [57] WordPress. Code reference, wpdb::prepare. <https://developer.wordpress.org/reference/classes/wpdb/prepare/>. (Visited March 2019).
- [58] WordPress. Output sanitization. https://codex.wordpress.org/Data_Validation#Output_Sanitization. (Visited March 2019).
- [59] WordPress. Wordpress nonces. https://codex.wordpress.org/WordPress_Nonces. (Visited March 2019).
- [60] WordPress. Function reference/check ajax referer. https://codex.wordpress.org/Function_Reference/check_ajax_referer. (Visited March 2019).
- [61] WordPress. Function reference/wp hash password. https://codex.wordpress.org/Function_Reference/wp_hash_password. (Visited May 2019).
- [62] CodesInChaos. Is md5 considered insecure? <https://security.stackexchange.com/questions/19906/is-md5-considered-insecure>. (Visited May 2019).
- [63] Ayesh Karunaratne. Php native password hash. <https://wordpress.org/plugins/password-hash/>. (Visited May 2019).
- [64] Wordfence. Wordfence security – firewall & malware scan. <https://nb.wordpress.org/plugins/wordfence/>. (Visited March 2019).
- [65] 2by2host. Limit login attempts reloaded. <https://wordpress.org/plugins/limit-login-attempts-reloaded/>. (Visited March 2019).
- [66] Kevin Vess. Force login. <https://wordpress.org/plugins/wp-force-login/>. (Visited February 2019).
- [67] WordPress. Code reference, add role. https://developer.wordpress.org/reference/functions/add_role/. (Visited April 2019).
- [68] WordPress. Roles and capabilities. https://codex.wordpress.org/Roles_and_Capabilities. (Visited March 2019).
- [69] WordPress. Function reference/remove role. https://codex.wordpress.org/Function_Reference/remove_role. (Visited March 2019).
- [70] WordPress. Roles and capabilities, capabilities. https://codex.wordpress.org/Roles_and_Capabilities#Capabilities. (Visited February 2019).

-
- [71] Docker. Manage swarm service networks, create an overlay network. <https://docs.docker.com/v17.09/engine/swarm/networking/#create-an-overlay-network>.
- [72] OWASP. Zed attack proxy. https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project. (Visited May 2019).
- [73] Todoist. Todoist. <https://todoist.com/>. (Visited January, May 2019).
- [74] Ondřej Mirtes. Wordpress support #35. <https://github.com/phpstan/phpstan/issues/35>. (Visited April 2019).
- [75] Matt Brown. Improving wordpress with static analysis. <https://medium.com/@muglug/improving-wordpress-with-static-analysis-505cc5ba495d>. (Visited April 2019).
- [76] WordPress. Pages. <https://codex.wordpress.org/Pages>. (Visited February 2019).
- [77] WordPress. Code reference/wp insert post. https://developer.wordpress.org/reference/functions/wp_insert_post/. (Visited February 2019).
- [78] jQuery. jquer.data(). <https://api.jquery.com/jQuery.data/>. (Visited March 2019).
- [79] @hiliOS. jquery.countdown, the final countdown. <http://hiliOS.github.io/jquery.countdown/>. (Visited March, May 2019) CSS Broken for https at time of writing.
- [80] Objective HTML Justin Kimbrell. Flipclock.js. <http://flipclockjs.com/>. (Visited March, May 2019).
- [81] WordPress. Child themes. <https://developer.wordpress.org/themes/advanced-topics/child-themes/>. (Visited March 2019).

A Project Agreement

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

Danny Lopez

Basel Katt

_____ (oppdragsgiver), og

Sander Løken Berntsen, Erlend Einmo, Sondre Granerud, Tobias Moe

_____ (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 7/01-19 til 20/05-19.

Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:

- Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra NTNU på Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
- Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv hvis de har skriftlig karakter A, B eller C.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.
6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.
7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.
8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.
9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.
10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.
11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): Erik Hjelmås

Oppdragsgivers kontaktperson (navn): Danny Lopez

Student(er) (signatur): Sander L Berntsen dato 29/01-19

Tolias Moe dato 29/01-19

Paul Gunn dato 29/01-19

Sondre Granerud dato 29/01-19

Oppdragsgiver (signatur): Danny Lopez M dato 29/01-19

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.

Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.

Plass for evt sign:

Instituttleder/faggruppeleder (signatur): _____ dato _____

B Group Contract

Group contract for PEMA bachelor

Timeframe:

This contract will last through the bachelor thesis for the spring semester of 2019.

Purpose:

The purpose of the contract is to ensure that all members of the group do their part in the work towards the bachelor's thesis.

Project leader:

Tobias Moe will act as the project leader, whose responsibility it is to follow up on meeting hours, appointments, tasks are split evenly between the group members and be the contact person to the project owner.

Meeting hours:

The group will have weekly meetings every monday. In these meetings we will follow up on what everyone did the last week, and we'll set weekly goals for the coming week. We will also be discussing talking points for the weekly meeting with the supervisor the next day.

Requirements:

Every group member is required to work approximately 30 hours a week starting from February 4th.

All meetings are obligatory, with the possibility of exceptions, when agreed upon by the group beforehand.

All members are required to write code following the code practices of the framework, and to comment their code.

All meetings with other parties need to have a meeting log (written in google Drive).

Decisions need to be documented, with reasoning for and against.

Decisions are voted on within the group, and need a majority (75%) to pass, in the case of a stalemate (50% / 2v2), we will ask a third party with relevant knowledge for their opinion. If the stalemate still stands after the conversation with the third party, the group leader has a veto/additional vote.

Consequences:

If a member of the group repeatedly breaks group-rules, or in other ways neglect the group work, the following consequences will be enacted:

1. First instance:
 - a. A conversation/meeting between all members of the group, where the problem is attempted mediated.
2. Second instance:
 - a. If the conversation/meeting did not yield results, a written warning is given to the disobedient member where:
 - i. The broken rule is explained.
 - ii. Specific measures are set in action to remedy the problem.
 - iii. Information about the consequence(s) if the member still neglects the work and rules set in place. First consequence: conversation/meeting with whole group, including supervisor present.
3. If no improvement in members efforts/behavior:
 - a. Written exclusion from the group via supervisor.
Nobody can be excluded from the group from the 1st of May until final submission deadline.

Signatures:

Erlend Einmo



Sander L. Berntsen



Sondre Granerud



Tobias Moe



C Permissions Tables

C.1 Custom Capabilities

Table 2: Table of custom WordPress permissions/capabilities

Permissions	Admin	Instructor	Student	Comment
pema_create_post	x	x		Can submit posts.*
pema_manage_tasks_lab	x	x		Can edit tasks for lab.*
pema_create_lab	x	x		Can create labs.*
pema_edit_lab	x	x		Can edit labs.*
pema_view_lab	x	x		Can view labs.*
pema_manage_task	x	x		Can manage tasks.*
pema_create_task	x	x		Can create tasks.*
pema_create_group	x	x		Can create user-groups.*
pema_manage_group	x	x		Can manage user-groups.
pema_edit_group	x	x		Can edit user-groups.*
pema_manage_topic	x	x		Can manage and create topics.*
pema_submitted_task	x	x		Can see who has submitted tasks.*
pema_see_lab			x	Can see labs user is enrolled in.

* These permissions also allow for the viewing of the relevant page in the admin-dashboard.

C.2 WordPress Standard Capabilities

Table 3: Table of standard WordPress permissions/capabilities used in PEMA

Permissions	Admin	Instructor	Student	Comment
read	x	x		Can access admin-dashboard.*
list_users	x	x		Can list/view users.*
create_users	x	x		Can create users.*
promote_users	x	x		Can promote users.*
delete_users	x	x		Can delete users.*
remove_users	x	x		Can remove users.*

* These permissions are limited for instructors.

D Custom WordPress Hooks Table

D.1 Backend actions

Table 4 lists all the custom hooks that have been created for the backend of the web application, e.g., when a lab is inserted, a custom hook is created in order to allow plugins to inject code into the specific place.

Table 4: Appendix: Table of custom WordPress hooks.

Name	Parameters	Description
pema_after_check_hash	user input, ID of task and the logged in user ID	After hash is checked
pema_after_deploy_lab	ID of the inserted lab, groups in lab, ID of lab author	After a lab is deployed
pema_after_create_group	ID of the inserted group, users in the group	After a group is created
pema_after_group_delete	ID of the group to be deleted	Before deleting a group
pema_before_lab_delete	ID of the lab to be deleted	Before deleting a lab
pema_after_insert_post	ID of the inserted post	After a post is inserted
pema_after_insert_task_template	ID of the inserted template	After a task template is inserted
pema_after_insert_task	ID of the inserted task and page	After a task is created
pema_before_task_delete	ID of the task to delete	Before a task is deleted
pema_after_update_task_template	ID of the template to update	After a task template is updated
pema_before_template_delete	ID of the template to delete	Before a task template is deleted
pema_before_topic_delete	ID of the topic to delete	Before a topic is deleted
pema_after_topic_update	ID of the topic to update	After a topic is updated
pema_after_insert_topic	ID of the newly inserted topic	After a topic is inserted
pema_after_lab_update	ID of the lab that is updated	After a lab is updated
pema_after_group_update	ID of the group that is updated	After a group is updated

D.2 Frontend actions

Table 5 lists all the custom hooks that are created for display on the frontend of the web application. They include all pages, menus or submenu pages, which allows a plugin to inject code into the specific place. These custom hooks allows a plugin developer to add HTML code to a page.

Table 5: Appendix: Table of custom WordPress hooks.

Name	Parameters	Description
pema_after_show_hint	taskID, userID (Does append)	After a hint is shown
pema_adding_vms_to_groups	groupID, labID	Used by Openstack plugin
pema_before_submit_create_group		Before a group is submitted.
pema_before_submit_create_lab		Before a lab is submitted.
pema_before_submit_create_post		Before a post is submitted.
pema_before_submit_create_task		Before a task template is submitted
pema_before_submit_edit_group	groupID	Before a group is edited.
pema_before_submit_edit_lab	labID	Before a lab is edited
pema_before_submit_edit_task	templateID	Before a task template is edited.
pema_display_lab_view	postID	In page.php
pema_display_task_view	postID	In template-task.php
pema_display_author_view	userID of current user, authorID	In author.php
pema_display_in_header	userID of current user,	In header.php
pema_display_home_page		In index.php

E OWASP ZAP Report

ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	1
Medium	3
Low	5
Informational	0

Alert Detail

SQL Injection

High (Medium)

Description

SQL injection may be possible.

- URL: <https://10.212.137.195/robots.txt/?query=query%27+AND+%271%27%3D%271%27+--+>
 - Method: GET
 - Parameter: query
 - Attack: query' OR '1'='1' --
- URL: <https://10.212.137.195/sitemap.xml/?query=query+AND+1%3D1+--+>
 - Method: GET
 - Parameter: query
 - Attack: query AND 1=1 --

Instances: 2

Solution

Do not trust client side input, even if there is client side validation in place.

In general, type check all data on the server side.

If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'

If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries.

If database Stored Procedures can be used, use them.

Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality!

Do not create dynamic SQL queries using simple string concatenation.

Escape all data received from the client.

Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input.

Apply the principle of least privilege by using the least privileged database user possible.

In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact.

Grant the minimum database access that is necessary for the application.

Other information

The page results were successfully manipulated using the boolean conditions [query' AND '1'='1' --] and [query' OR '1'='1' --]

The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison

Data was NOT returned for the original parameter.

The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter

Reference

- https://www.owasp.org/index.php/Top_10_2010-A1
- https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

CWE Id : 89

WASC Id : 19

Source ID : 1

Application Error Disclosure

Medium (Medium)

Description

This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page.

- URL: <https://10.212.137.195/wp-content/themes/Pema-Theme/>
 - Method: GET
 - Evidence: HTTP/1.1 500 Internal Server Error

Instances: 1

Solution

Review the source code of this page. Implement custom error pages. Consider implementing a mechanism to provide a unique error reference/identifier to the client (browser) while logging the details on the server side and

Incomplete or No Cache-control and Pragma HTTP Header Set

Low (Medium)

Description

The cache-control and pragma HTTP header have not been set properly or are missing allowing the browser and proxies to cache content.

- URL: <https://10.212.137.195/wp-admin/css/forms.min.css?ver=5.1.1>
 - Method: GET
 - Parameter: Cache-Control
- URL: <https://10.212.137.195/wp-login.php>
 - Method: POST
 - Parameter: Cache-Control
 - Evidence: no-cache, must-revalidate, max-age=0
- URL: <https://10.212.137.195/wp-includes/css/dashicons.min.css?ver=5.1.1>
 - Method: GET
 - Parameter: Cache-Control
- URL: https://10.212.137.195/wp-login.php?redirect_to=https%3A%2F%2F10.212.137.195%2F%3Fp%3D1
 - Method: GET
 - Parameter: Cache-Control
 - Evidence: no-cache, must-revalidate, max-age=0
- URL: https://10.212.137.195/index.php?rest_route=/
 - Method: GET
 - Parameter: Cache-Control
- URL: <https://10.212.137.195/wp-login.php?action=lostpassword>
 - Method: POST
 - Parameter: Cache-Control
 - Evidence: no-cache, must-revalidate, max-age=0
- URL: <https://10.212.137.195/wp-content/plugins/openstack-NTNU/openstack-api-style.css?ver=5.1.1>
 - Method: GET
 - Parameter: Cache-Control
- URL: <https://10.212.137.195/wp-login.php?action=lostpassword>
 - Method: GET
 - Parameter: Cache-Control
 - Evidence: no-cache, must-revalidate, max-age=0
- URL: <https://10.212.137.195/sitemap.xml/>
 - Method: GET
 - Parameter: Cache-Control
- URL: <https://10.212.137.195/wp-includes/wlwmanifest.xml>
 - Method: GET
 - Parameter: Cache-Control
- URL: <https://10.212.137.195/wp-includes/css/dist/block-library/style.min.css?ver=5.1.1>
 - Method: GET
 - Parameter: Cache-Control
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: Cache-Control
 - Evidence: no-cache, must-revalidate, max-age=0
- URL: <https://10.212.137.195/wp-content/themes/Pema-Theme/style.css?ver=5.1.1>
 - Method: GET
 - Parameter: Cache-Control
- URL: <https://10.212.137.195/wp-login.php>
 - Method: GET
 - Parameter: Cache-Control
 - Evidence: no-cache, must-revalidate, max-age=0
- URL: <https://10.212.137.195/wp-includes/css/buttons.min.css?ver=5.1.1>
 - Method: GET
 - Parameter: Cache-Control
- URL: <https://10.212.137.195/wp-admin/css/login.min.css?ver=5.1.1>
 - Method: GET

- Parameter: Cache-Control
- URL: <https://10.212.137.195/xmlrpc.php?rsd>
 - Method: GET
 - Parameter: Cache-Control
- URL: <https://10.212.137.195/>
 - Method: GET
 - Parameter: Cache-Control
- URL: <https://10.212.137.195/robots.txt/>
 - Method: GET
 - Parameter: Cache-Control
- URL: <https://10.212.137.195/wp-admin/css/l10n.min.css?ver=5.1.1>
 - Method: GET
 - Parameter: Cache-Control

Instances: 20

Solution

Whenever possible ensure the cache-control HTTP header is set with no-cache, no-store, must-revalidate; and that the pragma HTTP header is set with no-cache.

Reference

- https://www.owasp.org/index.php/Session_Management_Cheat_Sheet#Web_Content_Caching

CWE Id : 525

WASC Id : 13

Source ID : 3

Cookie No HttpOnly Flag

Low (Medium)

Description

A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.

- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wordpress_logged_in_62aede73102693e550411e8af31f773c
 - Evidence: Set-Cookie: wordpress_logged_in_62aede73102693e550411e8af31f773c
- URL: <https://10.212.137.195/wp-login.php?action=lostpassword>
 - Method: GET
 - Parameter: wordpress_test_cookie
 - Evidence: Set-Cookie: wordpress_test_cookie
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wordpress_62aede73102693e550411e8af31f773c
 - Evidence: Set-Cookie: wordpress_62aede73102693e550411e8af31f773c
- URL: <https://10.212.137.195/wp-login.php?action=lostpassword>
 - Method: POST
 - Parameter: wordpress_test_cookie
 - Evidence: Set-Cookie: wordpress_test_cookie
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wordpress_sec_62aede73102693e550411e8af31f773c
 - Evidence: Set-Cookie: wordpress_sec_62aede73102693e550411e8af31f773c
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wp-settings-time-0
 - Evidence: Set-Cookie: wp-settings-time-0
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wordpressuser_62aede73102693e550411e8af31f773c
 - Evidence: Set-Cookie: wordpressuser_62aede73102693e550411e8af31f773c
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wordpress_test_cookie

- Evidence: Set-Cookie: wordpress_test_cookie
- URL: <https://10.212.137.195/wp-login.php>
 - Method: POST
 - Parameter: wordpress_test_cookie
 - Evidence: Set-Cookie: wordpress_test_cookie
- URL: https://10.212.137.195/wp-login.php?redirect_to=https%3A%2F%2F10.212.137.195%2F%3Fp%3D1
 - Method: GET
 - Parameter: wordpress_test_cookie
 - Evidence: Set-Cookie: wordpress_test_cookie
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wordpresspass_62aede73102693e550411e8af31f773c
 - Evidence: Set-Cookie: wordpresspass_62aede73102693e550411e8af31f773c
- URL: <https://10.212.137.195/wp-login.php>
 - Method: GET
 - Parameter: wordpress_test_cookie
 - Evidence: Set-Cookie: wordpress_test_cookie
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wp-postpass_62aede73102693e550411e8af31f773c
 - Evidence: Set-Cookie: wp-postpass_62aede73102693e550411e8af31f773c
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wp-settings-0
 - Evidence: Set-Cookie: wp-settings-0

Instances: 14

Solution

Ensure that the HttpOnly flag is set for all cookies.

Reference

- <http://www.owasp.org/index.php/HttpOnly>

CWE Id : 16

WASC Id : 13

Source ID : 3

Cookie Without Secure Flag

Low (Medium)

Description

A cookie has been set without the secure flag, which means that the cookie can be accessed via unencrypted connections.

- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wp-settings-time-0
 - Evidence: Set-Cookie: wp-settings-time-0
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wordpress_62aede73102693e550411e8af31f773c
 - Evidence: Set-Cookie: wordpress_62aede73102693e550411e8af31f773c
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wordpressuser_62aede73102693e550411e8af31f773c
 - Evidence: Set-Cookie: wordpressuser_62aede73102693e550411e8af31f773c
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wordpresspass_62aede73102693e550411e8af31f773c
 - Evidence: Set-Cookie: wordpresspass_62aede73102693e550411e8af31f773c
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wordpress_sec_62aede73102693e550411e8af31f773c

- Evidence: Set-Cookie: wordpress_sec_62aede73102693e550411e8af31f773c
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wordpress_logged_in_62aede73102693e550411e8af31f773c
 - Evidence: Set-Cookie: wordpress_logged_in_62aede73102693e550411e8af31f773c
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wp-settings-0
 - Evidence: Set-Cookie: wp-settings-0
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: wp-postpass_62aede73102693e550411e8af31f773c
 - Evidence: Set-Cookie: wp-postpass_62aede73102693e550411e8af31f773c

Instances: 8

Solution

Whenever a cookie contains sensitive information or is a session token, then it should always be passed using an encrypted channel. Ensure that the secure flag is set for cookies containing such sensitive information.

Reference

- [http://www.owasp.org/index.php/Testing_for_cookies_attributes_\(OWASP-SM-002\)](http://www.owasp.org/index.php/Testing_for_cookies_attributes_(OWASP-SM-002))

CWE Id : 614

WASC Id : 13

Source ID : 3

Web Browser XSS Protection Not Enabled

Low (Medium)

Description

Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server

- URL: <https://10.212.137.195/wp-content/plugins/openstack-NTNU/>
 - Method: GET
 - Parameter: X-XSS-Protection
- URL: <https://10.212.137.195/wp-includes/js/>
 - Method: GET
 - Parameter: X-XSS-Protection
- URL: <https://10.212.137.195/wp-includes/css/>
 - Method: GET
 - Parameter: X-XSS-Protection
- URL: <https://10.212.137.195/wp-includes/>
 - Method: GET
 - Parameter: X-XSS-Protection
- URL: <https://10.212.137.195/wp-content/themes/Pema-Theme/js/>
 - Method: GET
 - Parameter: X-XSS-Protection
- URL: <https://10.212.137.195/wp-includes/css/dist/block-library/>
 - Method: GET
 - Parameter: X-XSS-Protection
- URL: <https://10.212.137.195/wp-includes/css/dist/>
 - Method: GET
 - Parameter: X-XSS-Protection
- URL: <https://10.212.137.195/wp-includes/js/jquery/>
 - Method: GET
 - Parameter: X-XSS-Protection
- URL: <https://10.212.137.195/robots.txt>
 - Method: GET
 - Parameter: X-XSS-Protection
- URL: <https://10.212.137.195/wp-admin/css/>
 - Method: GET
 - Parameter: X-XSS-Protection
- URL: <https://10.212.137.195/sitemap.xml/>

- Method: GET
- Parameter: X-XSS-Protection

Instances: 11

Solution

Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.

Other information

The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it:

X-XSS-Protection: 1; mode=block

X-XSS-Protection: 1; report=http://www.example.com/xss

The following values would disable it:

X-XSS-Protection: 0

The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit).

Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).

Reference

- [https://www.owasp.org/index.php/XSS_\(Cross_Site_Scripting\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)
- <https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/>

CWE Id : 933

WASC Id : 14

Source ID : 3

Password Autocomplete in Browser

Low (Medium)

Description

The AUTOCOMPLETE attribute is not disabled on an HTML FORM/INPUT element containing password type input. Passwords may be stored in browsers and retrieved.

- URL: https://10.212.137.195/wp-login.php?redirect_to=https%3A%2F%2F10.212.137.195%2F%3Fp%3D1
 - Method: GET
 - Parameter: user_pass
 - Evidence: `<input type="password" name="pwd" id="user_pass" class="input" value="" size="20" />`
- URL: <https://10.212.137.195/wp-login.php>
 - Method: GET
 - Parameter: user_pass
 - Evidence: `<input type="password" name="pwd" id="user_pass" class="input" value="" size="20" />`
- URL: <https://10.212.137.195/wp-login.php>
 - Method: POST
 - Parameter: user_pass
 - Evidence: `<input type="password" name="pwd" id="user_pass" aria-describedby="login_error" class="input" value="" size="20" />`
- URL: https://10.212.137.195/wp-login.php?reauth=1&redirect_to=https%3A%2F%2F10.212.137.195%2Fwp-admin%2F
 - Method: GET
 - Parameter: user_pass
 - Evidence: `<input type="password" name="pwd" id="user_pass" class="input" value="" size="20" />`

Instances: 4

Solution

Turn off the AUTOCOMPLETE attribute in forms or individual input elements containing password inputs by using AUTOCOMPLETE='OFF'.

Reference

- http://www.w3schools.com/tags/att_input_autocomplete.asp
- <https://msdn.microsoft.com/en-us/library/ms533486%28v=vs.85%29.aspx>

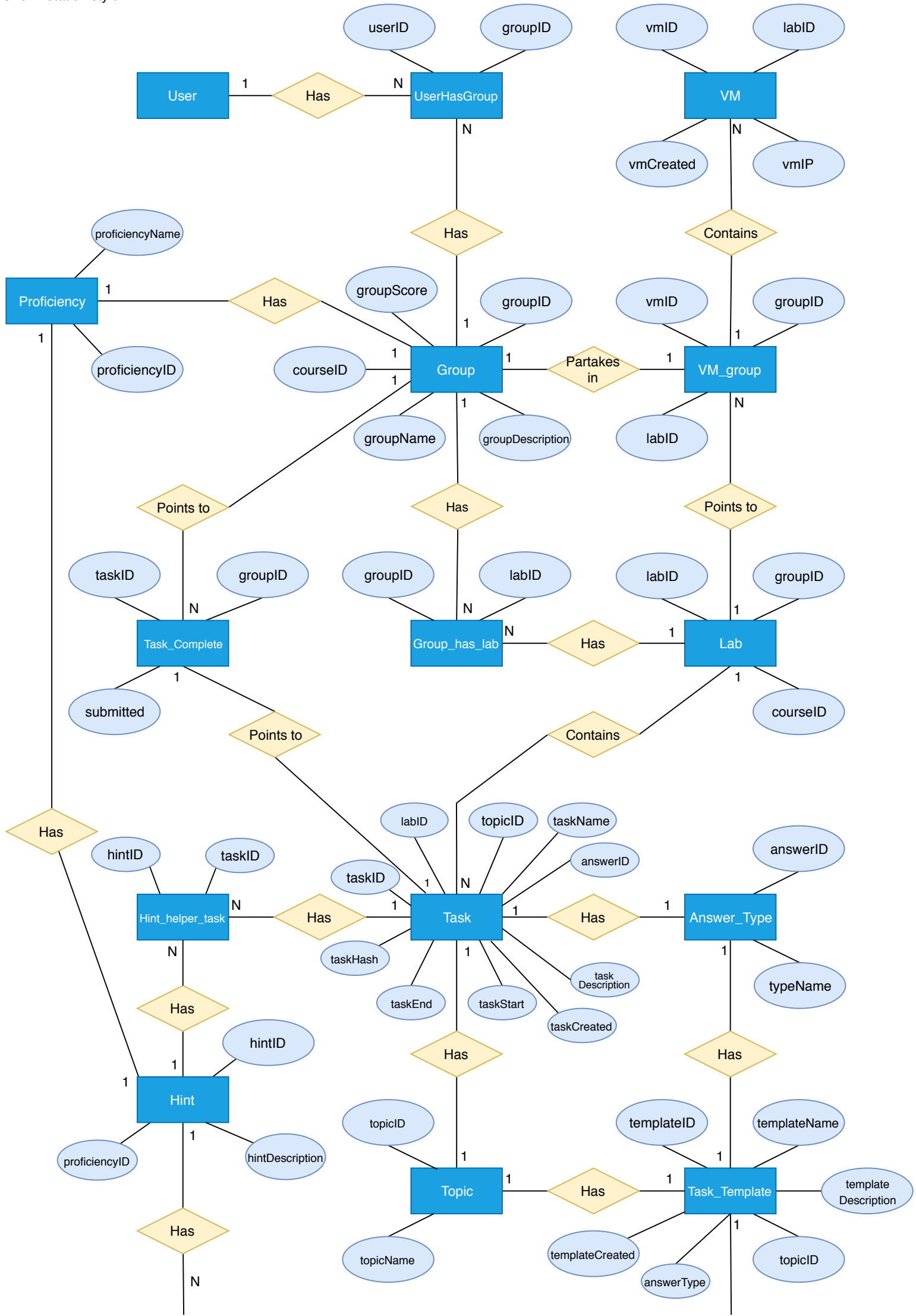
CWE Id : 525

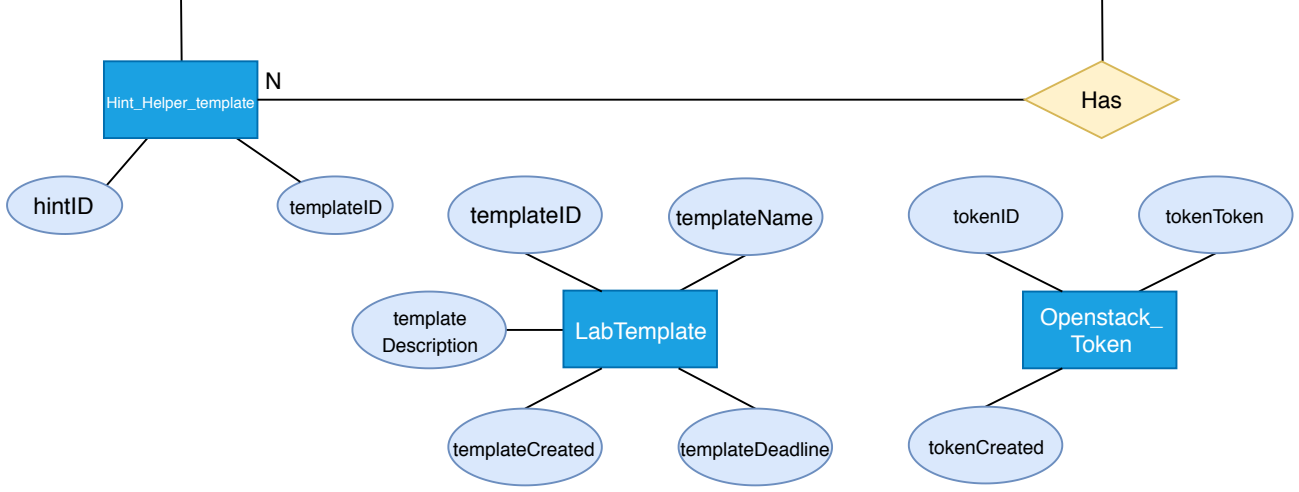
WASC Id : 15

Source ID : 3

F Diagrams

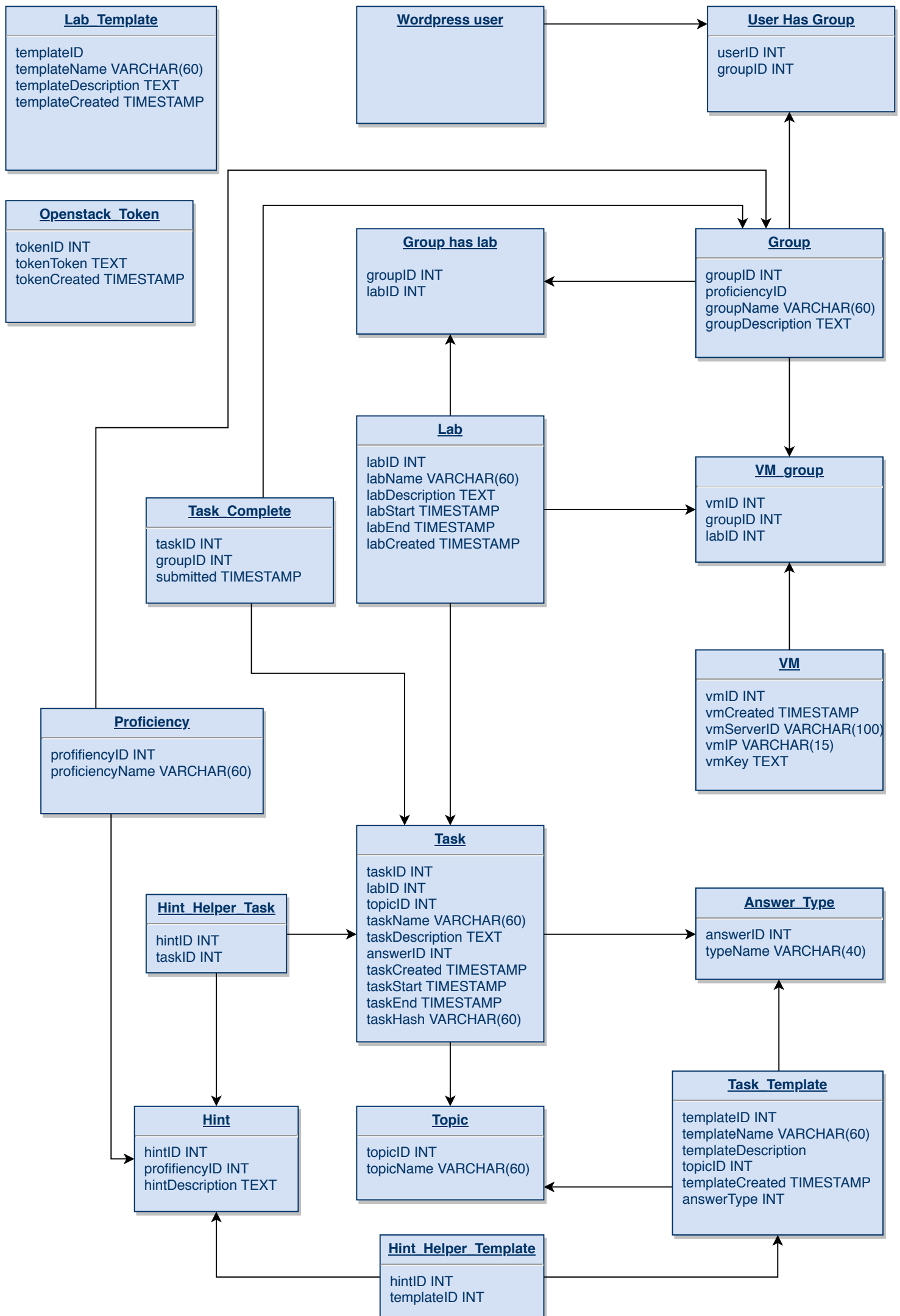
F.1 ER-Diagram





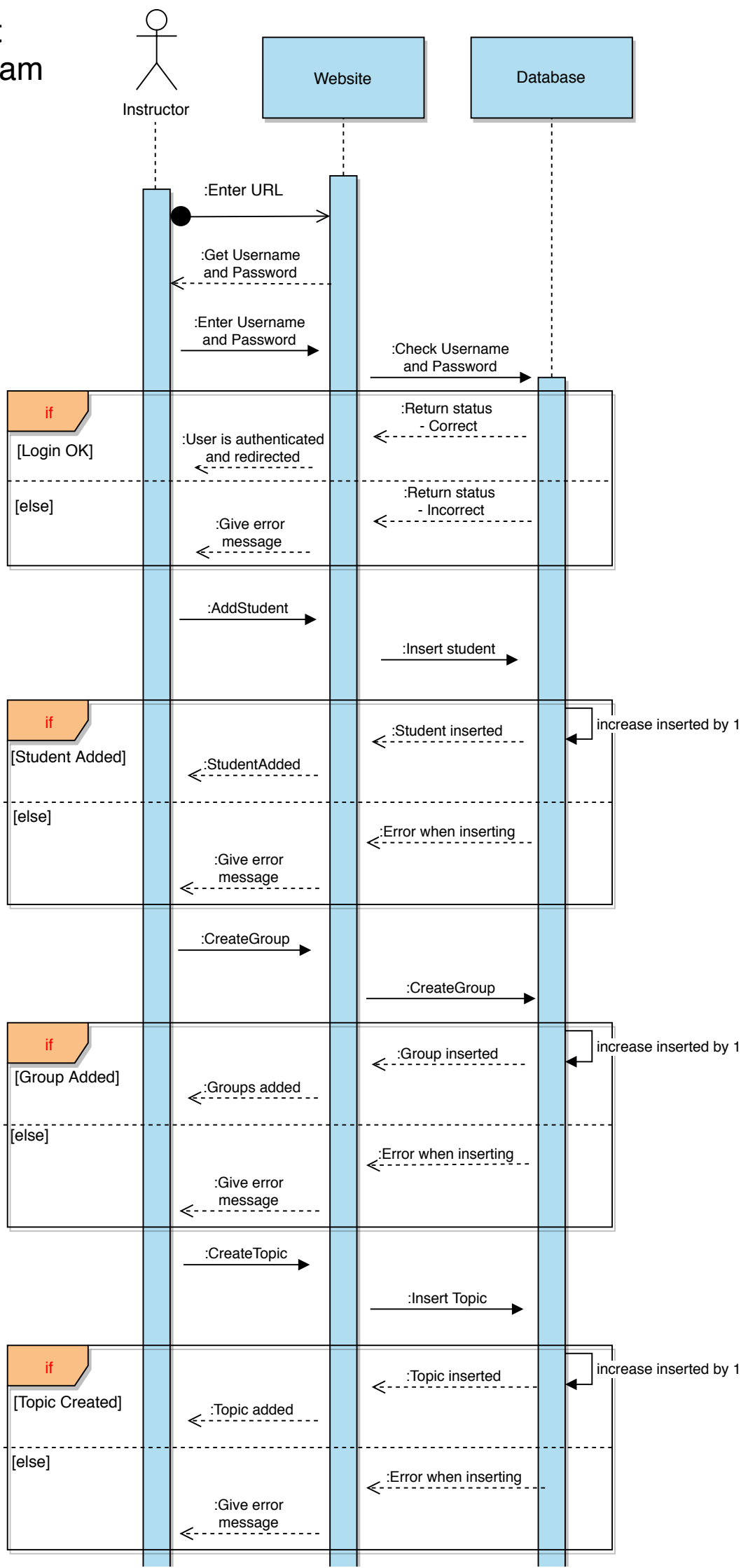
F.2 PEMA Database Schema

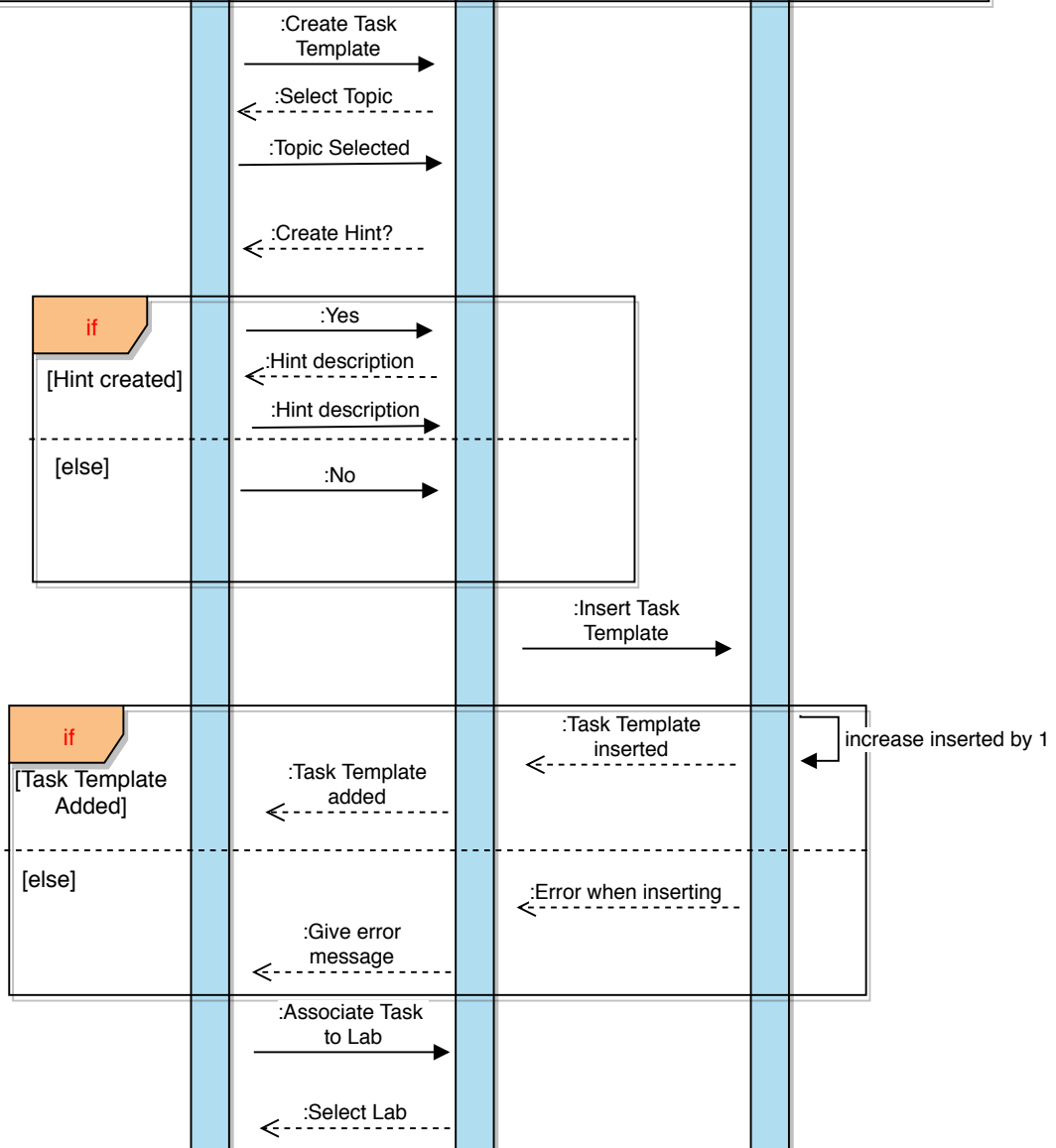
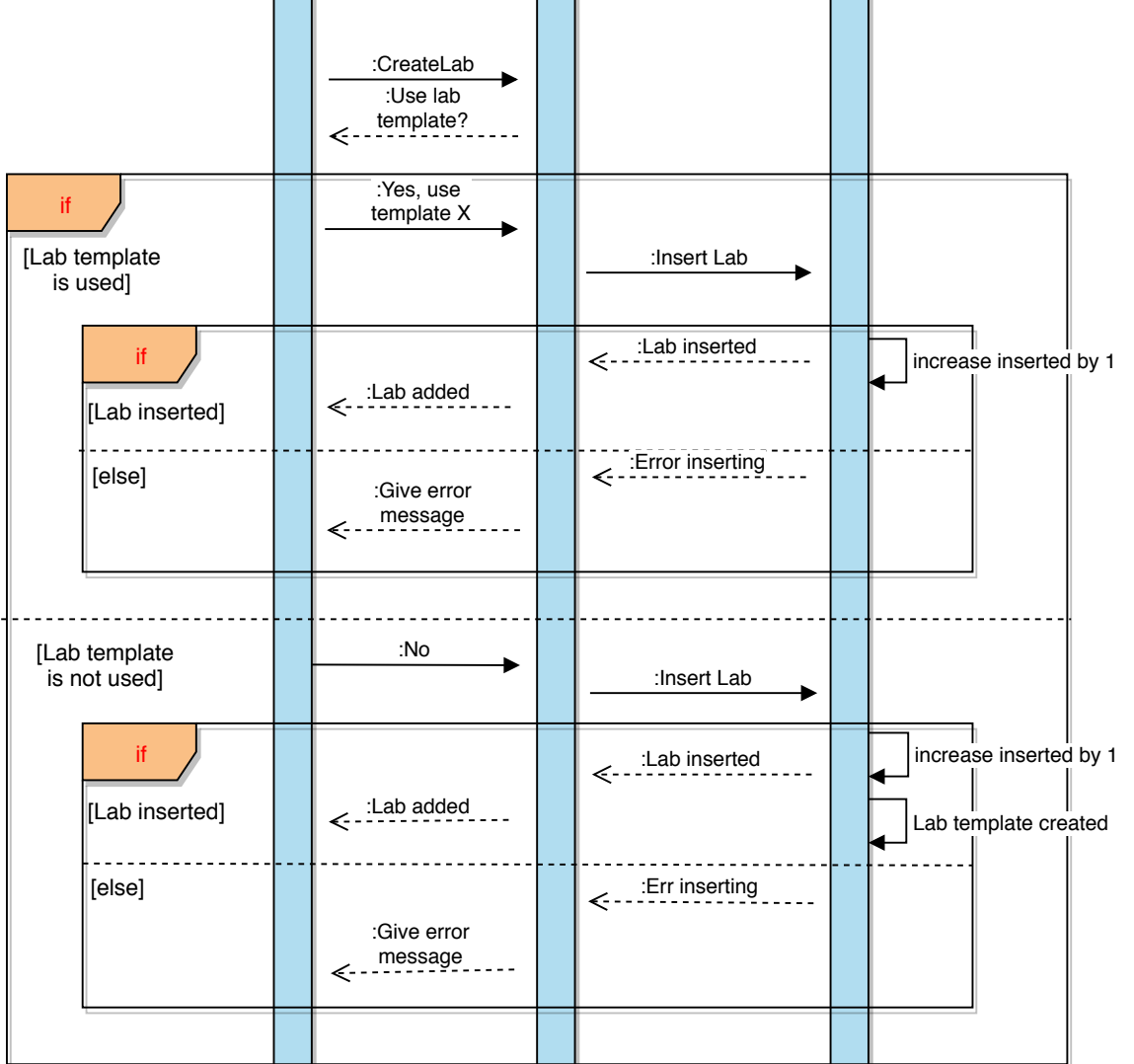
PEMA Database Schema

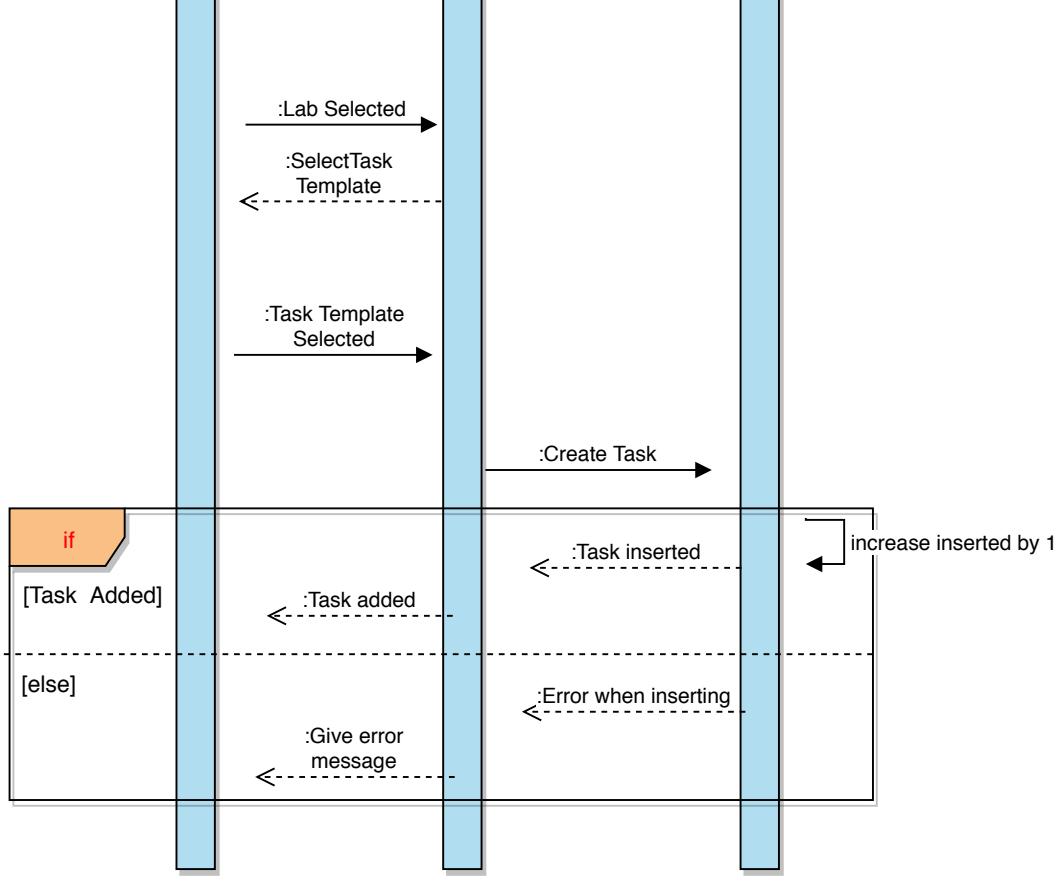


F3 Lab Deployment Sequence Diagram

Lab Deployment Sequence Diagram

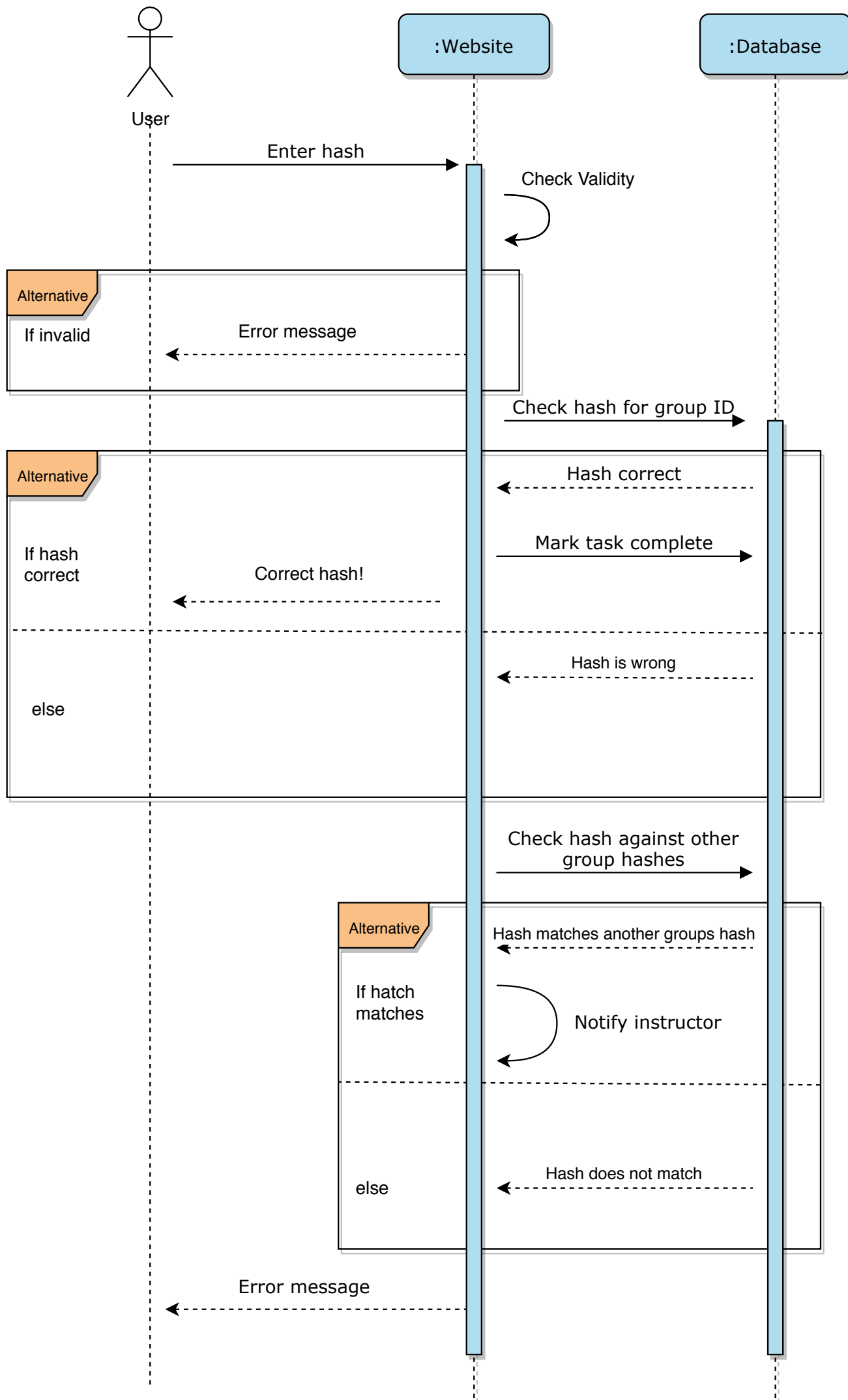






F.4 Task Delivery Sequence Diagram

Task Delivery Sequence Diagram



F.5 Misuse Case: Student - Query Injection

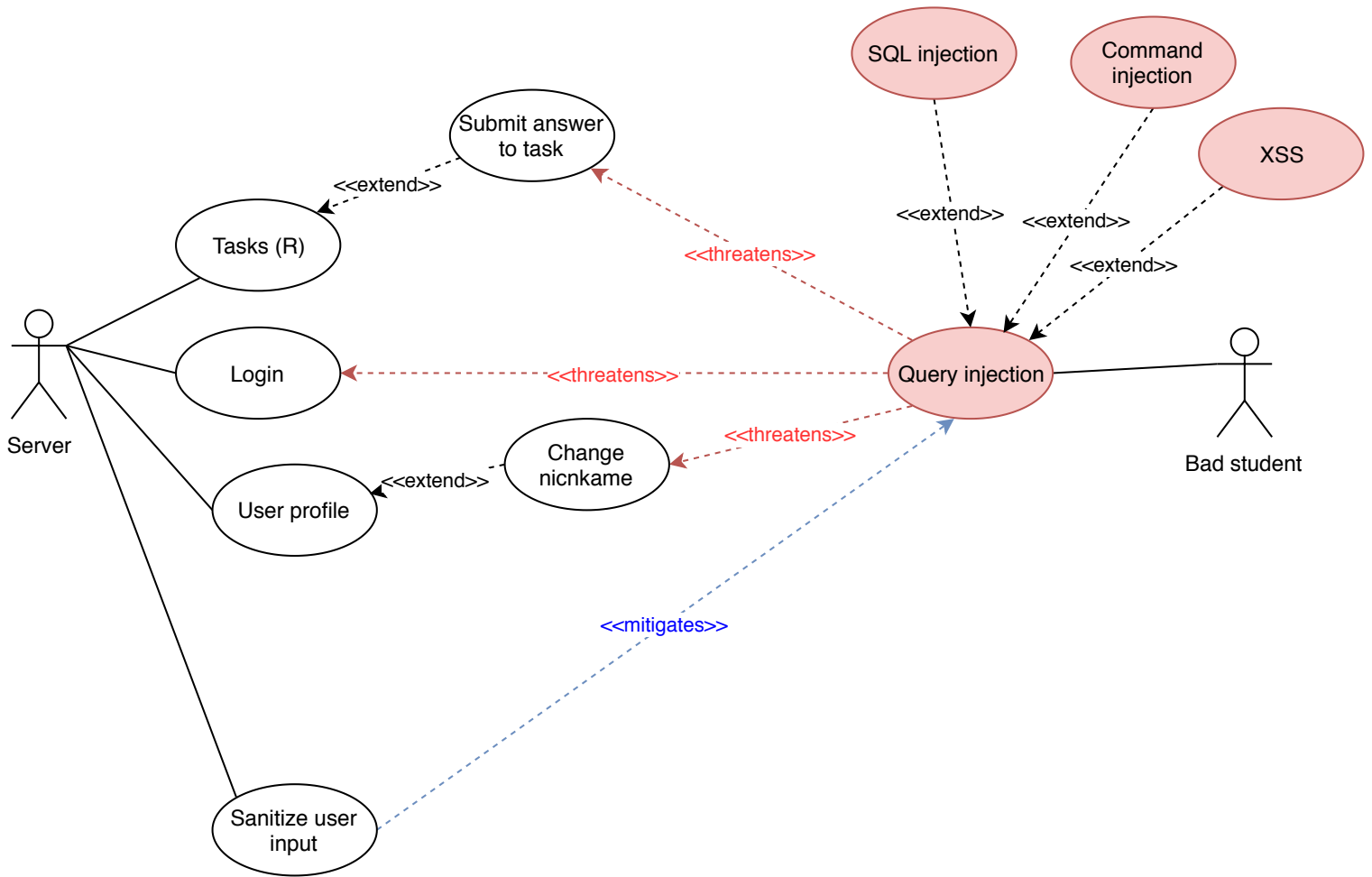


Figure 20: Appendix: Student query injection misuse case.

F.6 Misuse Case: Instructor - Query Injection

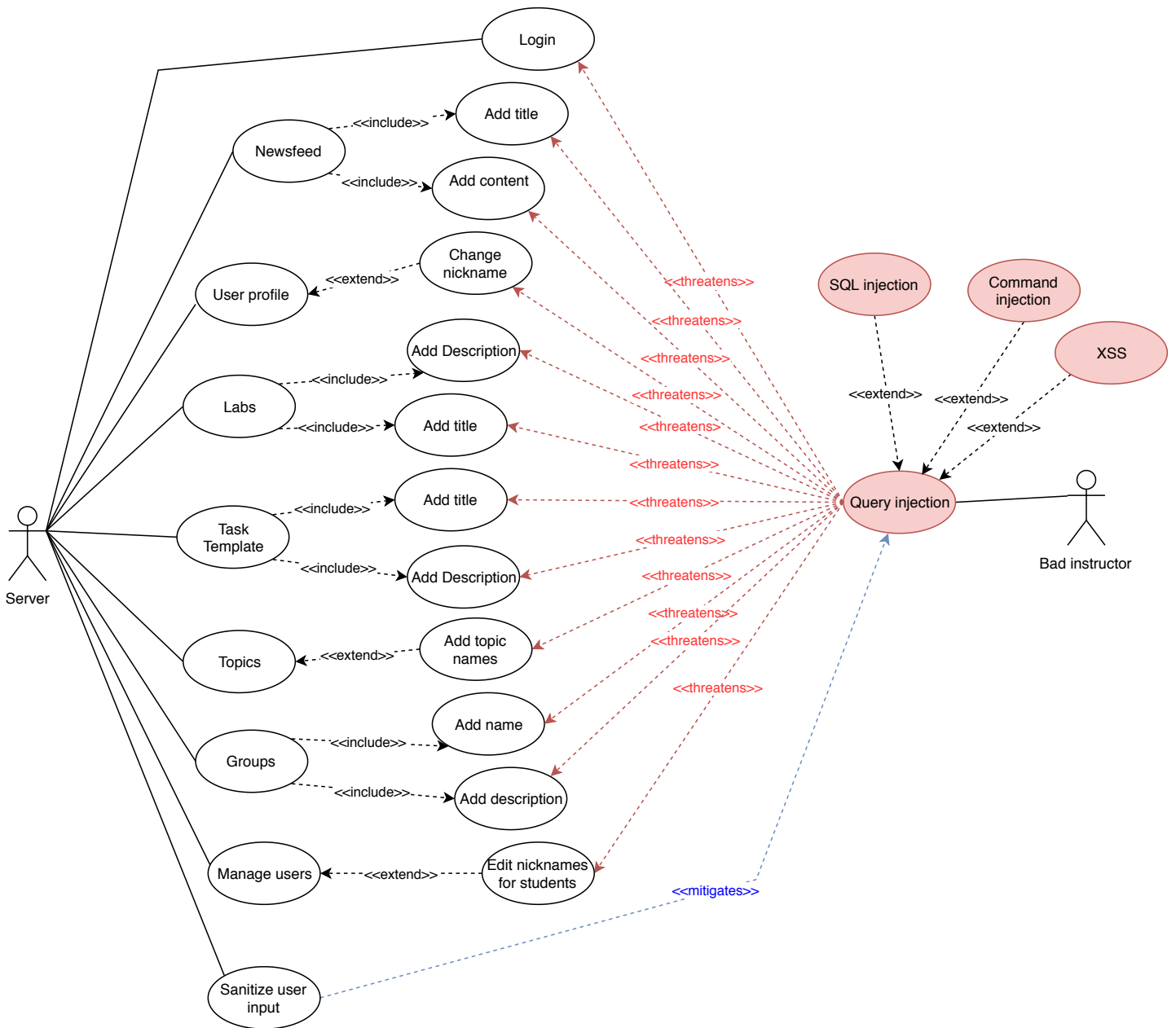


Figure 21: Appendix: Instructor query injection misuse case.

F.7 Misuse Case: Instructor - Unintentional Misuse

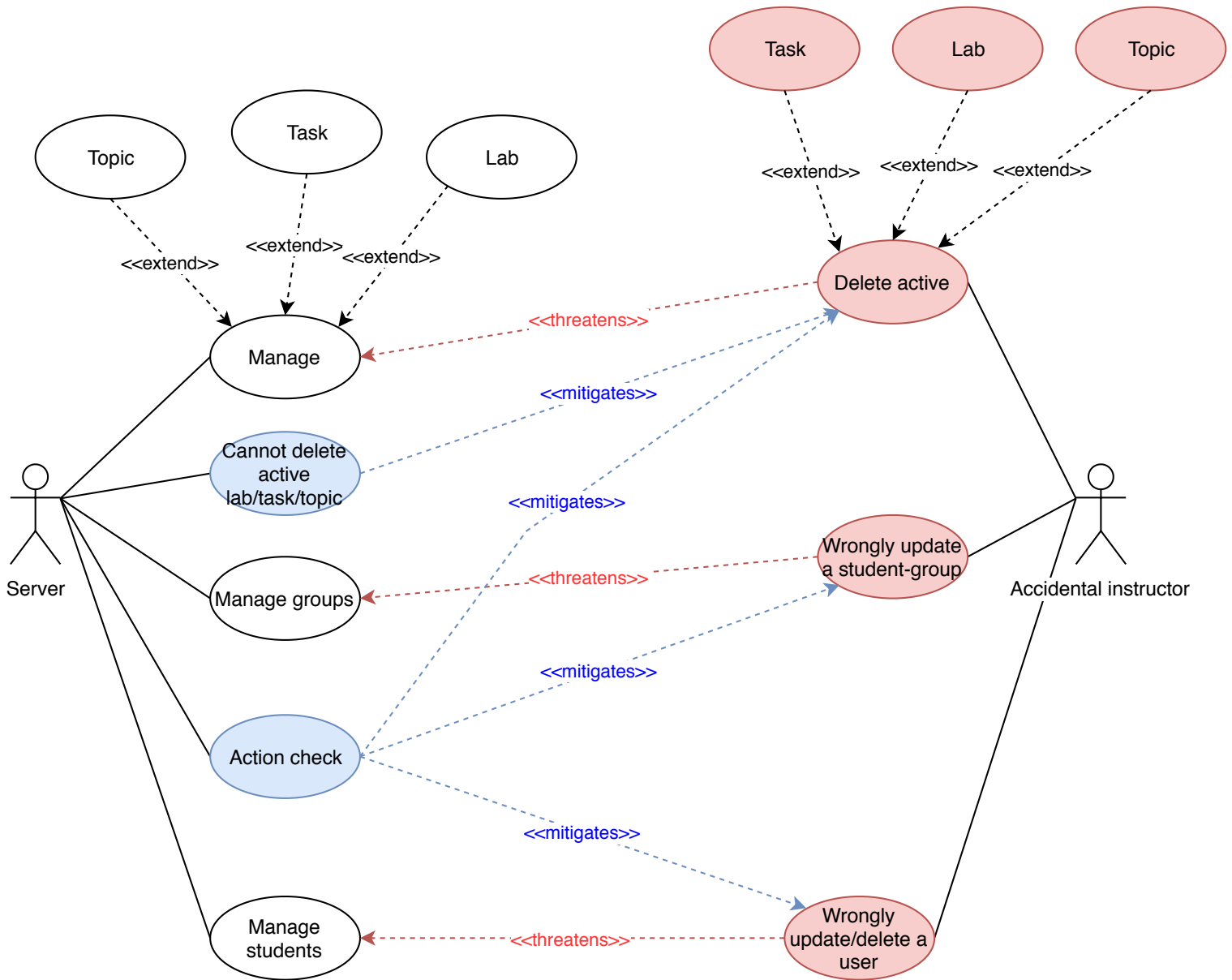


Figure 22: Appendix: Instructor unintentional misuse case.

F.8 Misuse Case: Admin - Unintentional Misuse

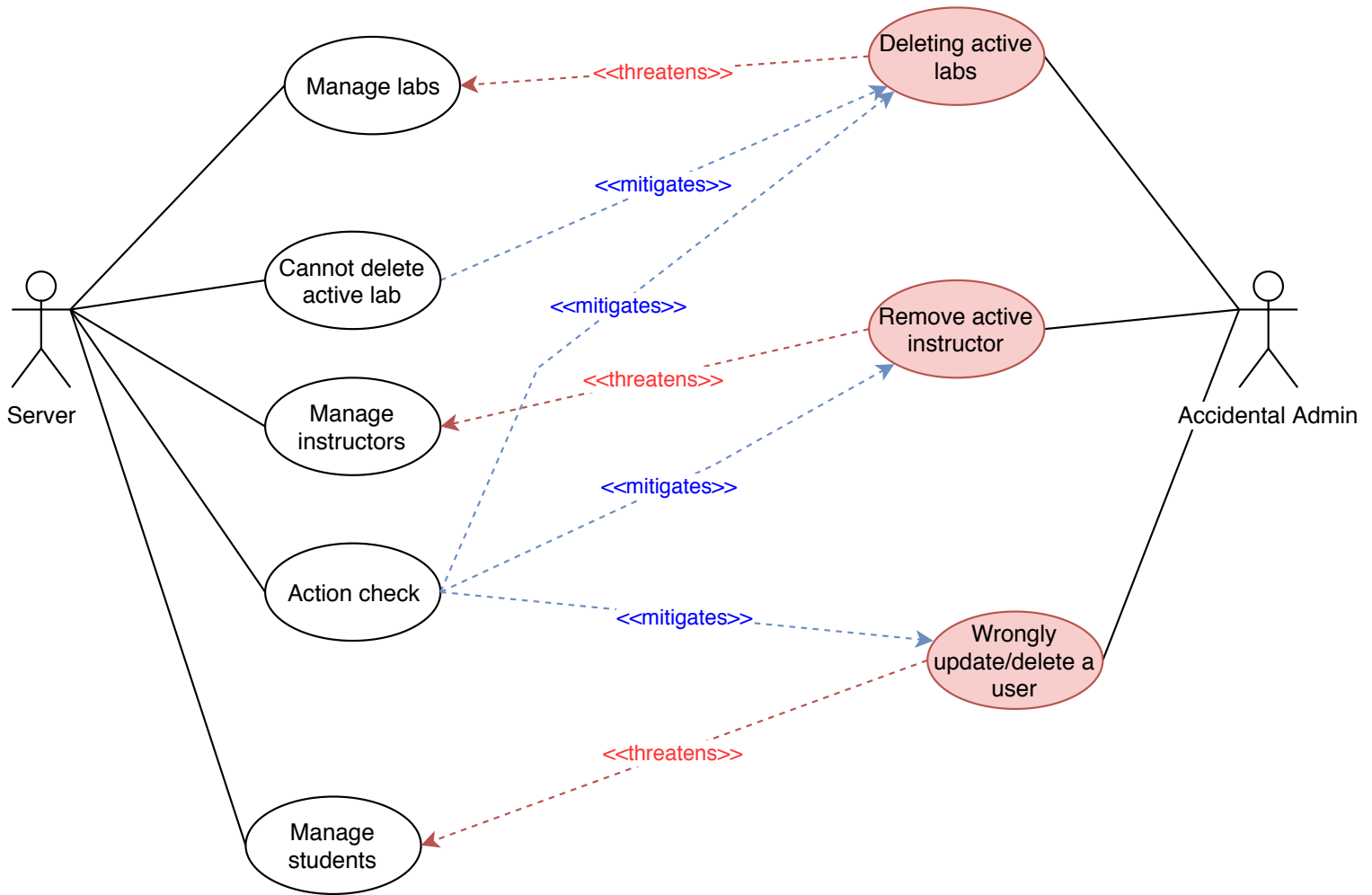


Figure 23: Appendix: Administrator unintentional misuse case.

F.9 Misuse Case: Login

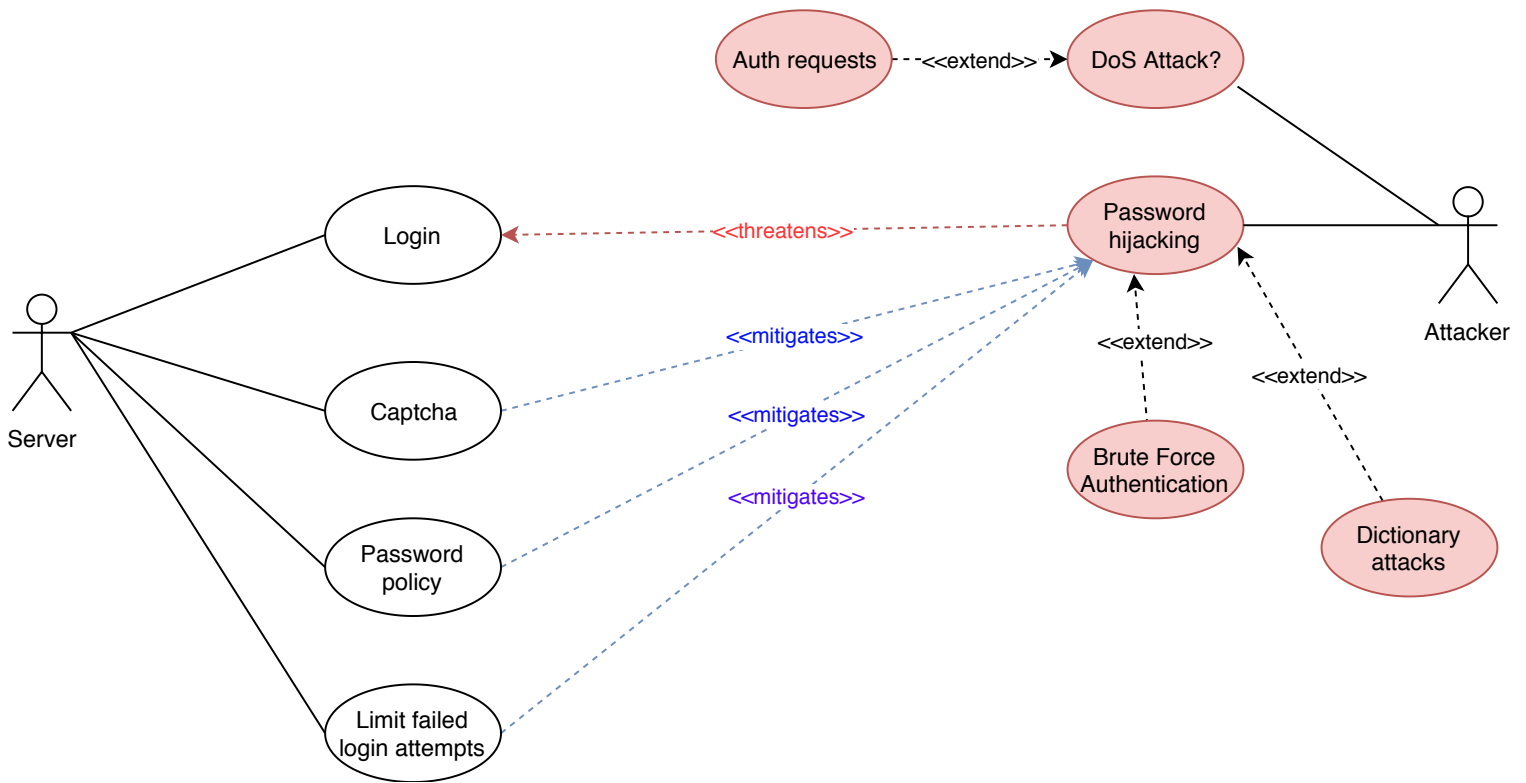


Figure 24: Appendix: Login misuse case.

F.10 PEMA Directory and File-Structure

```
1  admin-view (dir)
2      - createGroup.php
3      - createLab.php
4      - createPost.php
5      - createTask.php
6      - editGroup.php
7      - editLab.php
8      - editTask.php
9      - manageLab.php
10     - manageTask.php
11     - manageTasksForLab.php
12     - manageTopic.php
13     - submittedTasks.php
14  css (dir)
15     - pema_admin_style.css
16  css-admin (dir)
17     - create_group.css
18     - create_lab.css
19     - create_task.css
20     - edit_group.css
21     - edit_lab.css
22     - edit_task.css
23     - manage_lab.css
24     - manage_task.css
25     - manage_tasks_for_lab.css
26     - manage_topics.css
27     - post_news.css
28     - submitted_tasks.css
29  inc (dir)
30     - pema-ajax.php
31     - pema-on-startup.php
32     - pema-on-switch.php
33  js (dir)
34     - pema_admin_script.js
35     - script.js
36  js-admin (dir)
37     - create_group.js
38     - create_lab.js
39     - create_task.js
40     - edit_group.js
41     - edit_lab.js
42     - edit_task.js
43     - manage_lab.js
44     - manage_task.js
45     - manage_tasks_for_lab.js
46     - manage_topics.js
47     - post_news.js
48     - submitted_tasks.js
49  404.php
50  author.php
51  footer.php
52  functions.php
53  header.php
54  index.php
55  NTNU-logo.png
56  page.php
57  README.md
58  sidebar-home.php
59  sidebar-top.php
60  style.css
61  template-task.php
```

Figure 25: Appendix: PEMA Directory and File-Structure.

F.11 Gantt scheme - Start

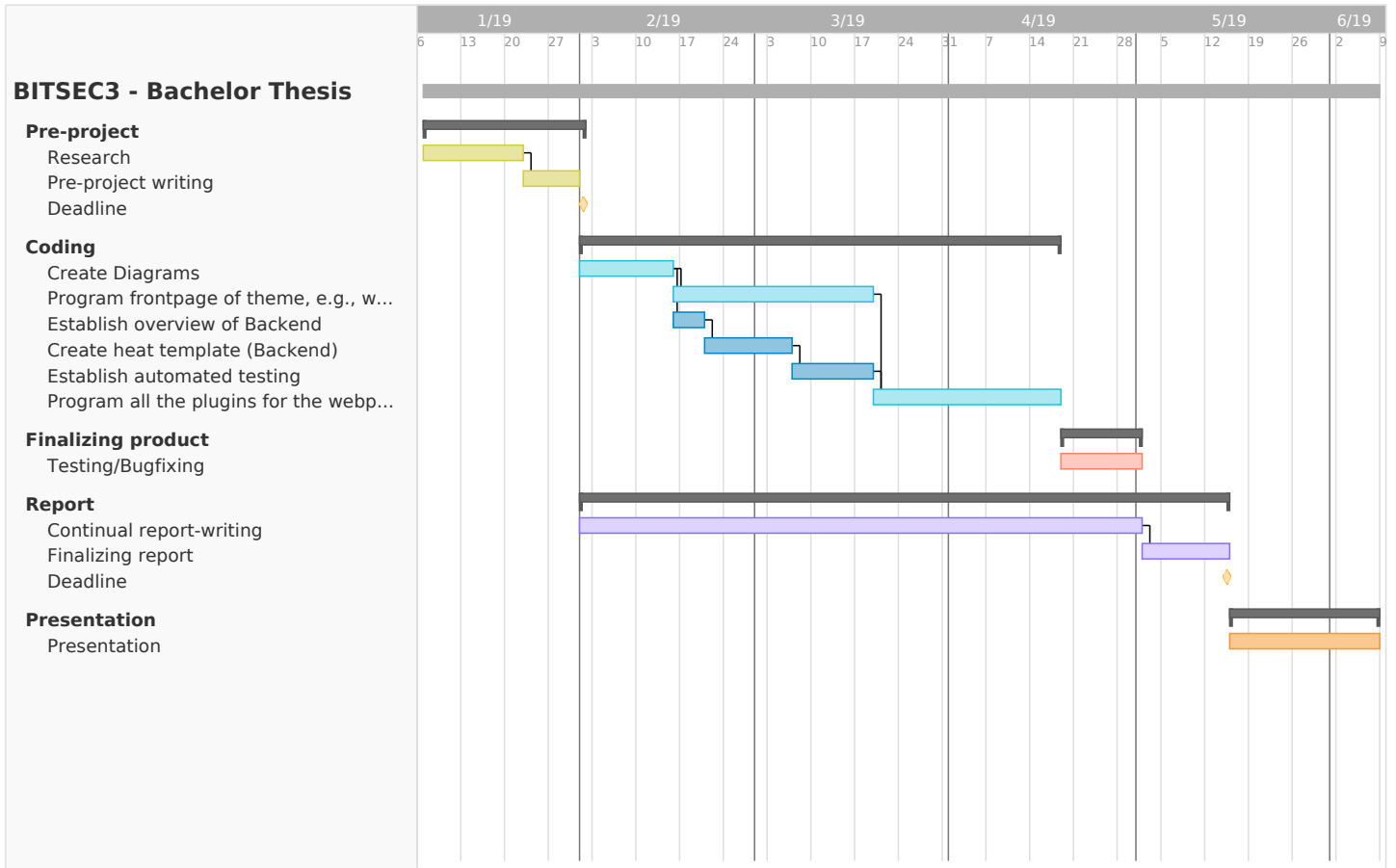


Figure 26: Appendix: Initial Gantt Schema.

F.12 Gantt scheme - End

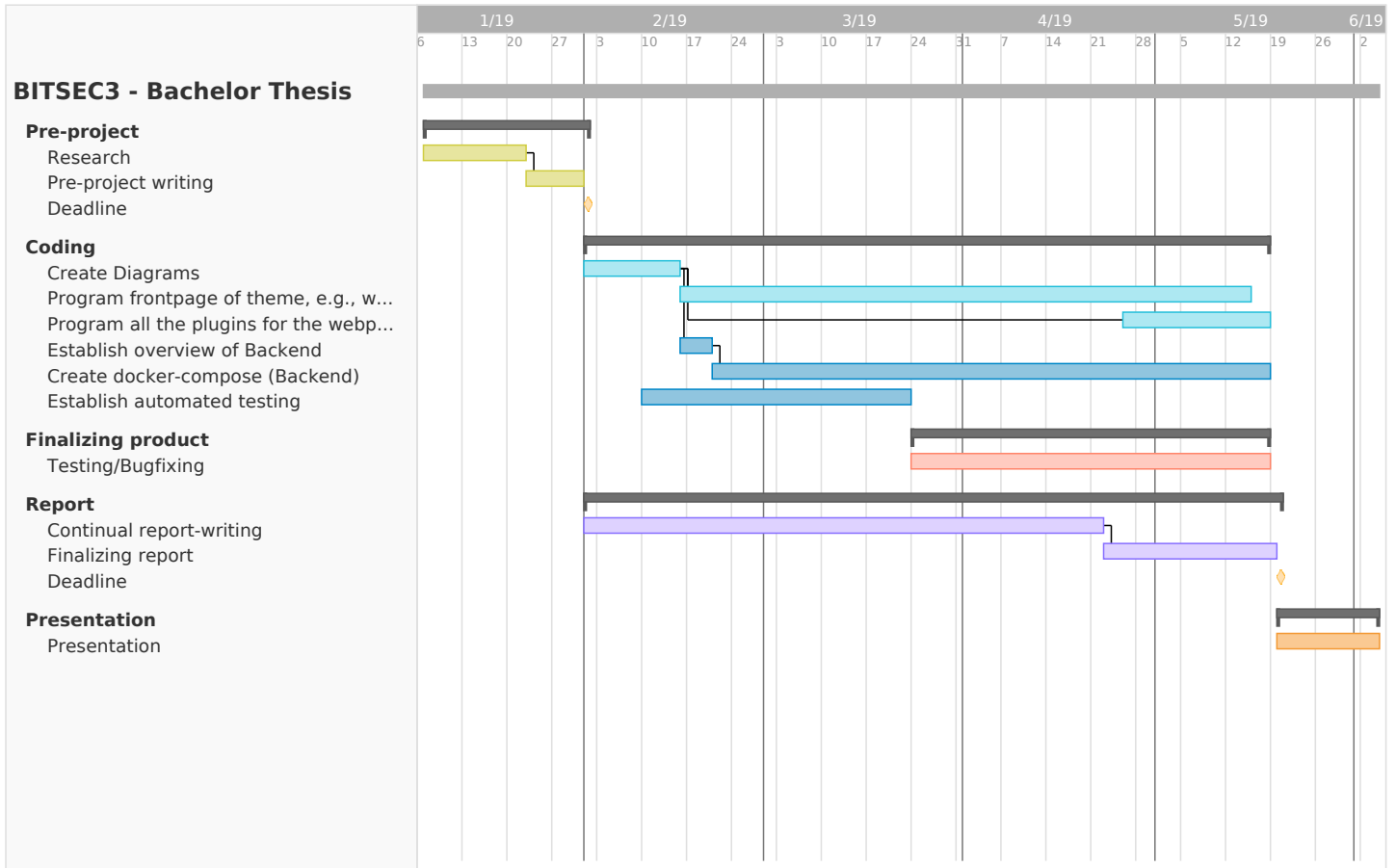


Figure 27: Appendix: Gantt schema showing actual project timeline.

G Pre-project

NTNU GJØVIK

BITSEC3

BACHELOR THESIS FOR IT-OPERATIONS & INFORMATION SECURITY

Pre-project report

Authors:

Sander L. BERNTSEN

Erlend EINMO

Tobias MOE

Sondre GRANERUD

May 19, 2019



NTNU

Contents

1	Introduction	1
1.1	Background	1
1.2	Objectives	1
1.2.1	Learning Objectives	1
1.2.2	Impact Objective	1
1.3	Target Audience	1
2	Scope	1
2.1	Project Description	1
2.2	Field of Study	2
3	Project Organization	2
3.1	Roles and responsibility	2
3.1.1	Group leader	2
3.1.2	Secretary (Logging)	2
3.1.3	Supervisor	2
3.1.4	Employers	2
3.2	Project rules	3
4	Planning, reporting and Monitoring	3
4.1	Software development methodology	3
4.2	Status meetings	3
5	Organization of Quality Assurance	3
5.1	Documentation, source code and standardization	3
5.2	Risk analysis	4
5.2.1	Countermeasures	5
5.3	Tools used	5
6	Project plan	6
6.1	Gantt Scheme	6
	Bibliography	7
	Appendices	8
A	Group contract	8

1 Introduction

1.1 Background

The Pentesting Exercise Management Application, hereby called PEMA, platform is an initiative intended to be a modular, scalable and virtualization agnostic platform that facilitates the deployment of virtual scenarios for cyber security education for research purposes. The realization of the PEMA platform and all of its components falls within the research scope of the Norwegian Cyber Security Range initiative. Together with the Pentesting Lab Environment Deployment, hereby called PLED, PEMA form part of a platform (a.k.a PEMA platform) that allows educators to deploy realistic environments suitable for training cyber security students of Penetration Testing, Ethical Hacking and Attack and Defense courses, as well as for cyber security competitions (such as Capture The Flag) and for research purposes.

1.2 Objectives

1.2.1 Learning Objectives

- Understand the differences in frameworks
- Coordination with another bachelor group (and a master thesis)
- Using Kanban in practice

1.2.2 Impact Objective

- Create a webdesign that the employer is satisfied with
- Create a scaling backend that can be easily deployed
- Make it easy for other people to work on our code

1.3 Target Audience

The target audience is learning institutions that focus on cybersecurity.

2 Scope

2.1 Project Description

1. Allow educators to create and deploy realistic environments that can supplement cyber security curricula (such as Ethical Hacking and Attack and Defense courses), and cyber security competitions such as CTF's. The intention of PEMA is to provide an abstraction layer in the form of a Domain Specific Language on top of lower level components such as configuration management tools (e.g. Puppet and Ansible) and virtualization/Cloud platforms, allowing instructors to programmatically model any given scenario (compose of both, vulnerable and not vulnerable systems) without having to worry about the underlying technologies.
2. Provides a teaching environment that can be implemented into existing curricula and that can be used by educators to impart knowledge related to cyber security in a practical way. This teaching environment will also be referred within this document as the "web portal". This web portal should support the creation of two different types of challenges, **task based challenges** and **project based challenges**, which can be balanced or unbalanced and which can be carried out individually or as a group.

3. Participants should be able to submit answers to the web portal in the form of hashes which will be evaluated automatically by the application for correctness. The web portal also contains a diverse variety of modules that enhances the functionality of the framework that can be enable/disable by the instructor upon demand, such as a scoreboard a black market, a forum and a module that allows students to reboot their own virtual machines.

2.2 Field of Study

- Web design
- Web security
- IT-infrastructure
- Relational database design
- Third party authentication
- Testing
- Automated testing
- REST-API

3 Project Organization

3.1 Roles and responsibility

3.1.1 Group leader

Tobias Moe

Tobias Moe will act as the project leader, whose responsibility it is to follow up on meeting hours, appointments, tasks are split evenly between the group members and be the contact person to the project owner.

3.1.2 Secretary (Logging)

Erlend Einmo

Responsible for writing a log of the meetings held with third parties. In the case that he is unable to fulfil his task, he is responsible for handing the task over to another group member on a per-meeting basis.

3.1.3 Supervisor

Erik Hjelmås

Erik is an Associate Professor at NTNU Gjøvik, he is our supervisor through this project, giving us guidance and helping with some decisions along the way.

3.1.4 Employers

Danny Lopez

Danny is a master student and he will be our main contact person if we have any questions regarding the project.

Basel Katt

Basel is an associate professor at NTNU Gjøvik in the Department of Information Security and Communication Technology.

3.2 Project rules

You can find the project rules in appendix A.

4 Planning, reporting and Monitoring

4.1 Software development methodology

Our first thought was to use Scrum as our software development model, but we quickly realized that scrum would become to big and complex for us with the sprints as we wanted a more open and flexible model. With Kanban we can easily change our focus from one thing to another, this is much harder if we were to use scrum with sprints. Kanban has fewer rules and is more lightweight than scrum, this also makes it more challenging as we need to be able to handle the lack of rules. However as we'll mostly be working together using a VoIP service (TeamSpeak) for our communication platform we feel confident that Kanban will work better than Scrum.

The formula for Little's Law[1] is commonly expressed as $L = \lambda W$, where "L" is the average number of costumers, " λ " is the average arrival rate and "W" is the average time in the system. By redefining the terms as follows we can use Little's Law to our benefit:

- L becomes Work In Progress (WIP)
- λ becomes Delivery Rate (DR)
- W becomes Lead Time (LT)

This gives us the formula $WIP = DR * LT$ or $DR = WIP/LT$ or $LT = WIP/DR$.

We will be using multiple Kanban boards on Trello to visualize our workflow. In our previous experiences with Trello boards, we realized we quickly got a big board with heaps of entries, this can feel daunting and disorganized. To cope with this we decided to split our tasks over multiple boards that will each represent a part of the project, for example we will have one board for the frontend, one for the backend, one for the report writing etc.

By using the Trello boards to visualize our workflow, we can use the redefined formula to estimate how long each board will take to complete by using the following example.

We have a board with 10 WIP cards on it.

We know we can complete 2 cards in a day, which gives us a DR of 2.

By using the formula we can estimate that it will take 5 days to complete the board.

If we were to increase the amount of WIP cards to 20 then the LT would increase to 10, meaning it would double the time to complete the board. In order to cope with the increase in WIP cards then we would need to find a way to change the DR to 4, and this could prove difficult as it can be hard to "push" more work onto people.

By using this formula to our benefit we can efficiently plan ahead.

4.2 Status meetings

The group will have weekly meetings with supervisor and employers every Tuesday. Sometimes we may have a joint meeting with the other bachelor group that we will be working with.

5 Organization of Quality Assurance

5.1 Documentation, source code and standardization

Our employers and ourselves have set high expectations for documentation and commenting when coding. As other people most likely will be working on our code at a later date, we want to ensure that we have

good documentation ready for them. We will be following the PHP coding standard [2] and the standard specified by the frameworks used. In case of a conflict between the two, the framework will take precedence. The framework has highest priority, as this is the system which will do all the translation if necessary. We will be storing our source code in a private git repository, e.g., Github or Bitbucket.

5.2 Risk analysis

Below are our identified risks. We have only included the most relevant risks.

#	Risk	Type	Likelihood	Consequence	Value
1	Some functionalities are not doable in chosen framework	Product	Likely	Significant	9
2	Frontend scope becomes too ambitious for our experience	Product	Likely	Significant	9
3	Project is not complete in time	Project	Unlikely	Critical	8
4	Significant change in requirements specification	Project	Unlikely	Significant	6
5	Slight change in requirements specification	Project	Likely	Slight	6
6	Illness over longer time period	Project	Unlikely	Significant	6
7	Integration with PLED group not being feasible	Product	Likely	Slight	6
8	Backend scope becomes too ambitious for our experience	Project	Unlikely	Significant	6
9	Application is vulnerable after project completion	Product	Low	Significant	3
10	Loss of documentation or source code	Project/Product	Low	Significant	3
11	The Domain Specific Language is not completed	Project	Likely	Significant	6

Table for identified risks

Acceptable risk (1-4) Considerable risk (5-10) Countermeasure necessary(11-16)

Table explaining colour and its values

We deem acceptable risks safe enough to leave them unaltered. Considerable risks are discussed internally on a case-by-case basis to figure out whether there is a need for countermeasures or not. Risks that fall under the category "countermeasure necessary" are critical risks that can significantly impact the project, and needs to be remedied.

Likelihood/Consequence	Small	Slight	Significant	Critical
Low			9, 10	
Unlikely			4, 6, 8	3
Likely		5, 7, 11	1, 2	
Very likely				

Table for risk matrix before countermeasures

5.2.1 Countermeasures

- Countermeasure for risk number 1:
We will especially look for modular frameworks where we can meet the product requirements. This will decrease the likelihood of the risk to unlikely.
- Countermeasure for risk number 2:
We will reach out to more experienced people for tips regarding frontend. By doing this we will reduce the likelihood of the risk to unlikely.
- Countermeasure for risk number 3:
By using a software development strategy like Kanban we will have a good overview of all tasks and their deadline, which reduces the likelihood of the risk to low.
- Countermeasure for risk number 4:
By having frequent meetings with employers we will reduce the likelihood of getting significant changes in requirements specification low.
- Countermeasure for risk number 11:
Continue project as planned, propose solution in final report.

Likelihood/Consequence	Small	Slight	Significant	Critical
Low			4, 9, 10	3
Unlikely			1, 2, 6, 8	
Likely	11	5, 7		
Very likely				

Table for risk matrix after countermeasures

5.3 Tools used

The group agrees on what tools we are going to use during the project and thesis. These decisions are based on previous experience, as we have worked together as a group several times previously. The tools are chosen to keep our development consistent, and to allow for effortless collaboration, with the ability to see what the other members of the group are working on at any time.

Name	Type	Usage
Overleaf	Collaborative \LaTeX writing	Report writing
Toggl	Online tool used to track time	Time tracking
Trello	Used to visualize Kanban boards	Tasklist
Google Drive	Host documents	Storage
Google Docs	Used to write notes	Noting
Draw.io	Used to create diagrams	Diagrams
TeamSpeak 3	Voice communication platform	Communication
Facebook/Messenger	Text communication platform	Communication
Wordpress	CMS framework for basic functionality	Frontend
Docker	Software containerization platform	Backend
Openstack/SkyHiGh	Cloud platform used to host the backend	Backend

6 Project plan

6.1 Gantt Scheme

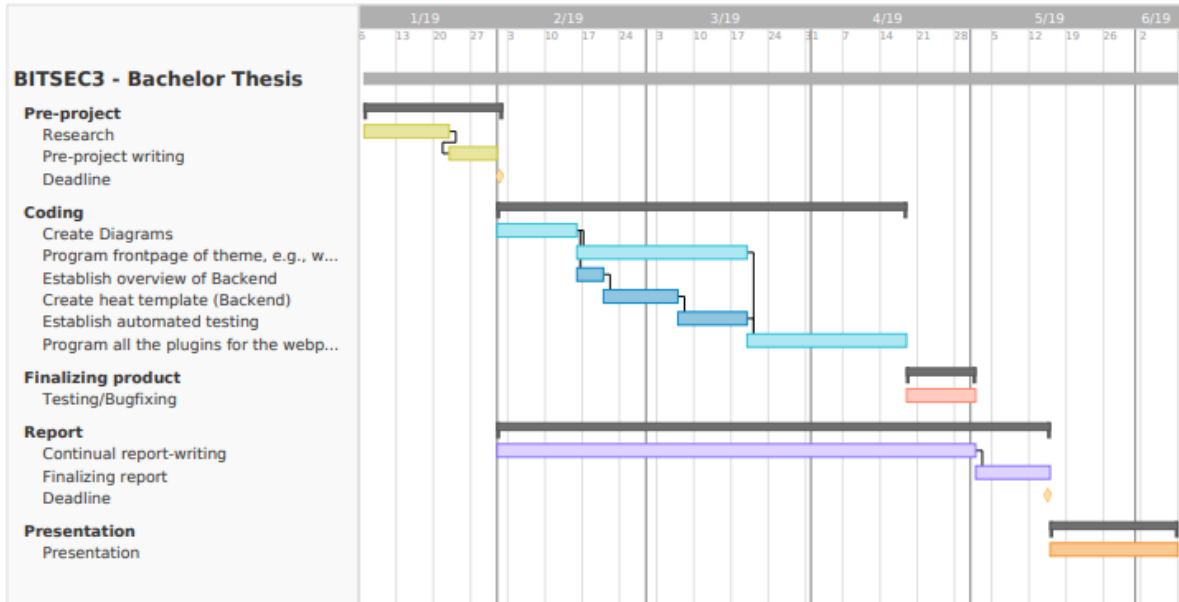


Figure 1: Gantt Schema.

In the image above one can see the proposed Gantt scheme for this project. The first month represents the phase where we research all relevant parts of the project. After this we move over to the "coding" phase, where we will first start drafting all relevant diagrams for the project (SQL diagram, Use cases, Flowcharts, etc.). We then move over to a stage where the group will be divided in two, where the focus for one part is to create the theme while the other half starts working on creating the backend. When the theme is functional, the entire group will begin making all the different plugins for the website. Then we will be doing relevant testing and bugfixes on the website. The report writing is a continual process.

Bibliography

- [1] Wikipedia. *Little's law*. (Accessed: 26.01.2019). URL: https://en.wikipedia.org/wiki/Little's_law.
- [2] PHP Framework Interop Group. *Coding Style Guide*. (Accessed: 26.01.2019). URL: <https://www.php-fig.org/psr/psr-2/>.

Appendices

A Group contract

Group contract for PEMA bachelor

Timeframe:

This contract will last through the bachelor thesis for the spring semester of 2019.

Purpose:

The purpose of the contract is to ensure that all members of the group do their part in the work towards the bachelor's thesis.

Project leader:

Tobias Moe will act as the project leader, whose responsibility it is to follow up on meeting hours, appointments, tasks are split evenly between the group members and be the contact person to the project owner.

Meeting hours:

The group will have weekly meetings every monday. In these meetings we will follow up on what everyone did the last week, and we'll set weekly goals for the coming week. We will also be discussing talking points for the weekly meeting with the supervisor the next day.

Requirements:

Every group member is required to work approximately 30 hours a week starting from February 4th. All meetings are obligatory, with the possibility of exceptions, when agreed upon by the group beforehand. All members are required to write code following the code practices of the framework, and to comment their code. All meetings with other parties need to have a meeting log (written in google Drive). Decisions need to be documented, with reasoning for and against. Decisions are voted on within the group, and need a majority (75%) to pass, in the case of a stalemate (50% / 2v2), we will ask a third party with relevant knowledge for their opinion. If the stalemate still stands after the conversation with the third party, the group leader has a veto/additional vote.

Consequences:

If a member of the group repeatedly breaks group-rules, or in other ways neglect the group work, the following consequences will be enacted:

First instance:

A conversation/meeting between all members of the group, where the problem is attempted mediated.

Second instance:

If the conversation/meeting did not yield results, a written warning is given to the disobedient member where: The broken rule is explained. Specific measures are set in action to remedy the problem. Information about the consequence(s) if the member still neglects the work and rules set in place. First consequence: conversation/meeting with whole group, including supervisor present.

If no improvement in members efforts/behavior:

Written exclusion from the group via supervisor. Nobody can be excluded from the group from the 1st of May until final submission deadline.

Signatures:

Tobias Moe

Erlend Einmo

Sander Løken Berntsen

Sondre Granerud

H Meeting Logs

Meeting log appendices, in reverse chronological order.

H.1 May meeting logs

07.05.19 - Week 19

Weekly meeting

Talking points:

- Status + demo

Present:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- Erik Hjelmås (Supervisor)
- Danny Lopez (Principal)

Duration:

40 minutes

Discussion:

- Status
- We have a proof of concept, spinning up Kali VMs for each group when a lab is started.
 - With some information returned.
- Horizon as an interface.
 - Making pema into a Horizon-like platform?
- A full interface for instructors
 - A lab has these objectives.
 - These machines are attached to it...
 - Etc
- An instructor view over the physical lab.
- Give instructors access to the Kali machines.
- Some discussion about users not providing their own keys, but getting ones generated by PEMA.
 - Decided that this is still the better solution.
- We show a little demo of recent changes.
 - Wait for condition, or loop request until success, instead of sleep.

03.05.19 - Week 18

Meeting with Danny

Talking points:

- Status, report and product

Present:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- Danny Lopez (Principal)

Duration:

100 minutes

Discussion:

- Report:
 - Content found in box.com
 - Documentation
 - First chapter not very clear
 - Not much differentiation on what the NCR is, and the course?
 - We need to talk about what NCR is.
 - Stakeholders local:
 - NTNU
 - Instructors in Ethical Hacking Course
 - Malware labs maybe?
 - External stakeholders:
 - CTF participants etc..
 - Ask hjelmås about Danny's notes.
 - We need more in-depth related to project.
 - Business requirements
 - High level.
 - How is the business supposed to benefit from this.
 - Quality aspects
 - Availability
 - Security
 - Etc.
 - Prioritize these.
 - Functional requirements
 - Can be related to the business requirements.
 - Need to be precise, measurable and follow the correct format.
 - Problem description
 - Must be clear in the first chapter.

- How it relates to PEMA and the DSL
- Change more on how PEMA relates to the DSL, not describing the actual DSL.
 - The introduction makes it seem like the DSL.
- We want:
 - Proof of Concept
 - Could describe the Ethical Hacking course, our experience.
- Assumptions and constraints
 - Little focus on graphical design.
 - We should not forget the order. (box, Danny is making a list)
 - Technical in the actual implementation.
- Related works:
 - There are similar projects, Danny has some notes in box.
 - Webgoat
 - Ec council link - important apparently.
 - It has a lot of functionality that we wanted initially.
 - Lacks proficiency levels.
- Architecture
 - OWASP Juice shop, good example for illustrations of how the system works.
 - Application level.
 - Appealing and easily understandable for the reviewer.
-
- General:
 - Need to not forget we work on the infrastructure as well.
 - Note:
 - We should illustrate the Lab - Task hierarchy.
 - PEMA is the PoC of the dashboard of NCR/PEMA.
 - Mikhals work is the core.
 - Danny will try to provide some functional requirements for the basic VM management module.
 - Test cases.
 - E.g "Allows a dozen users at the same time"
 - Roles and capabilities
 - Make a nice table out of it.
 - Grid of x - roles, y - permissions.
 - Draw.io (drive) folders with diagrams that we can use and contribute.
 - Especially Requirement Elicitation Process.
 - Also uses MindMap, basically a more user-friendly draw.io
 - Diagrams of what we have, and what we intended to have
 - Sequence diagram.
 - Activity diagram? ER-diagram

- Put more attention to the introduction.
 - How we envision the implementation with the DSL.
- We need the sequence diagram as well.
 - Danny will contact us.
- Compile mail with questions.

H.2 April meeting logs

30.04.19 - Week 18

Weekly meeting

Talking points:

- Report status ✓

Present:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- Erik Hjelmås (Supervisor)

Duration:

35 minutes

Discussion:

- Talking about status
 - A bit about our workflow writing the report..
 - We are still developing PEMA, but focus is the report.
 - Status about what is missing on the dev. Part.
 - SSL: Ask [Øivind Kolloen](#) maybe?
 - Feide: Talk to Lars Erik
- Discussion about meetings with the design student, and the student continuing the DSL dev.
- Think a little about what the edge of the project is.
 - What is the most important individual contribution from us.
 - For a good grade, the individuality and such is important.
 - Look at our punchline, our contribution to the project (as a group, without employer etc.)
 - How we have contributed to the project, other than the project description.

23.04.19 - Week 17

Weekly meeting

Talking points:

- Status on report

Present:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- Erik Hjelmås (Supervisor)

Duration:

35 minutes

Discussion:

- Talking about the report
 - We expected the response, and reasoning behind...
- How should we write about the web-development process in the report
 - Specify for each 3rd party function used etc...?
 - Look at other reports on how they have done this.
 - Using now-time, "this is how it is.." Not storytelling. Not tutorial.
 - Storytell if concisely telling about why something turned out the way it did.
- Something about logging, installation and moving of PEMA etc..
- Confirming no DSL
 - We should check with Basel and Danny what we should write about the integration that was supposed to happen with the DSL.
 - Gather some info from Basel and Danny on what to write here.
 - We have plans for this, as well as the integration with PLED.
 - What should have been in place to allow the DSL integration.
 - Write it in a better way than "We did not have time for this"
 - Emphasize the cooperation.
- HEAT php template to deploy Kali
- Make video demo.
- From chapter 3 and out
 - No figures.
 - More diagrams!
- Draft 2 - Weekend ->13th may.

H.3 March meeting logs

29.03.19 - Week 12

Meeting with all groups

Talking points:

-

Attendees:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- Erik Hjelmsås (Supervisor)
- Danny Lopez (Employer)

Duration:

2 hours 10 minutes

Discussion:

- We need to make sure our documentation is good enough
 - Requirements and such
 - Scenarios (labs) etc..
 - Test cases
 - E.g. how a user logs into the website.
- Today, we are going to work on the architecture between all the groups.
 - Sequence/activity diagram
 - Draw the links between all the systems.
 - General
- LAB styles
 - Pen
 - Malware
- CTF styles
 - Jeopardy
 - Attack/Defence
- Geir Olav wants to use PEMA for Malware labs from next year (semester?)
 - Do we need fileserver for malware uploads?
 - Answer types, likely text.
 - Isolated labs for malware, simulating traffic etc. (DSL)
- NCR Meetings:
 - 5th -
 - 12th - Geir Olav will present his requirements for PEMA.
- Start with general components of the system.
 - Try to see connection between components.
 - Architecture diagram

- Sequence diagram
- UML Diagrams (If we want to go one step further)
- For processes:
 - BPML?
 - Workflow management diagram
- We show what we want from PLED while Danny prints something.
 - The problem with “AND” statements on query.
 - Seems like it displays parent as well.
 - PLED will look at some good solutions for sorting between malware, challenges, ansible etc..
- We discuss how we will pass information to Mikhail (DSL)
 - YAML format.
 - Little documentation, will likely not be any either, as Mikhail will not prioritize it at all.
 - Look at Mikhails GitHub to look at yaml examples etc.
 - Mikhail needs at least a port that is defined in the image, we wonder if this is something that can be sent through PEMA when the yaml file is sent.
- Making a sequence diagram.
 - We are not too sure about the returning of IP addresses from the DSL, as the spawning takes some time, and the IPs are not ready until the servers are up.
- Answers to tasks are stored in PLED
 - “Everything that has to do with challenges (tasks), is stored in PLED.
 - Flags are gotten from PLED when we get the vuln. App.
 - We need a local temporary storage for “in-use flags”, that we can use for answer validation.
- **Passing of info to the DSL (yaml) is being put on the shelf? It will be implemented after our work is done**

26.03.19 - Week 12

Weekly meeting

Talking points:

-

Attendees:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- Erik Hjelmås (Supervisor)

Duration:

30 minutes

Discussion:

- Status
 - Basically good, but we need to have a talk with Danny regarding some clarifications.
 - And we need a meeting with PLED and Mikhal regarding the links between our projects.
- We show a little demo.
 - Hjelmås thinks we are doing well, and have good progress.
 - He also wants us to try our hand at designing the CSS ourselves.
 - This would be nice for the thesis, as it shows more diverse skills.
- Hjelmås wants a draft of the report the evening of April 10th.
 - + documentation.

22.03.19 - Week 11

NCR Showcase/demo

Talking points:

- Showcasing our project for Norwegian Cyber Range

Attendees:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- NCR (Including Danny and Basel)
- PEMA-Group
- PLED-Group
- Mikhal

Duration:

1 hour 5 minutes

Discussion:

- The instructor needs to be able to create topics when creating the lab.
- Events
 - Something Basel and Mikhal is working on.
- Store Lab templates for the DSL in yaml for imports and exports.
- Looking at wireframes
 - Make sure we follow the steps in there to create a lab, even if it is full of garbage.
- There is also a wish for CTF mode.
 - CTF Showcase?
 - Interface as is current, perfect for CTFs.
- Tasks and subtasks.
 - Do we need possibility for answers in subtasks?
 - If not, this can be made through the description.

18.03.19 - Week 11

Brainstorming meeting

Talking points:

- Where should we get a range of IP's from? Mikhal?

Attendees:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, ~~Sondre Granerud~~. (Group members)
- PLED - Group
- Mikhal - Masters student
- Erik Hjelmås (Supervisor)
- Danny Lopez (Employer)
- Basel Katt

Duration:

1 hour 50 minutes

Discussion:

- Danny gives intro.
 - Manual labor to create server-side of labs.
 - Puppet or ansible.
 - Mikhal likes ansible, yaml based works well with python (his work)
 - Danny thinks its not as agnostic as initially thought.
 - Decision: Basically, go for ansible. We are using HEAT anyways.
 - Or everything through DSL
- PLED explains their Docker setup.
- PLED explains their vulnerability setup
 - Automation is difficult, because of the nature of the vulnerability-files.
 - As of now atleast requires some manual work.
 - Offensive security has a db of basically all vulnerabilities, except what PLED wants.
 - They are using a scraper to gather info from exploitDB.
- AnsibleDirectory, basically PuppetForge
- Deployment
 - Services described in DSL?
- Something called Galaxy is used, did not get the context.
- [Mikhal] Need to deploy a service with some parameters.
- What info should be included, and what should be set as default
 - With regards to vm deployment.
- Basel gives PLED some ideas of where/how to get vulnerabilities.

- Their way of scraping seems alright, but if there is a website change, it needs a rework.
 - Their scraper can build a complete database in 40k seconds (11hrs).
 - Then update on change.
- Basel will try to contact exploitDB, try to get vulnerable applications.
- ~~— Danny has two design students that have shown interest in helping with GUI.~~
- ~~— We need to come up with some requirements for design students.~~
- ~~— Maybe use Dannys templates.~~
- We will have a meeting with the design students.
- Populating the VMs with “realistic” filesets etc.
 - Basel: Build database gradually manually.
 - Danny: Datasets for Cyber Forensics.
 - This will be integrated with PLED.
 - First in a group of Libraries(databases) of information useful to PEMA.
 - They will basically have a PoC, where Mikhal deploys a simple version with emails or something
- PLED: REST-API is up and running
- Vulnerability metadata
 - Used for deployment and scoring etc.
- **BPMN Diagram?**
 - Could be useful for thesis, show what part of the “big picture” we are talking about.
- UML Activity diagram?
- Having an interface towards PLED, basically a view of what is in the PLED databases.
 - What vulnerabilities are available etc...
 - Is available for testing.
 - They will send us the info needed. (Can be put in the phabricator-wiki)
- Authentication
 - Login.ncr.ntnu.no
 - Choose where you want to go, what platform you want to go to.
 - Hella confused on the NCR integration.
- Have CTF availability.

- We have a short talk with PLED regarding the API, after the meeting has ended.

12.03.19 - Week 10

Weekly meeting

Talking points:

- Clarification/specifying of what a task needs to contain.
 - Task notes.txt
 - Should a task have one or several possible topics?
- Having the possibility for an answer on a lab.
 - The answer being on the lab itself, having a single answer for entire lab.
- Who should handle group creation?
 - Students or instructors?
 - Perhaps instructor chooses if done by students or instructor?
- Should there always be assigned 3 VMs for a group (even if there is only 1 person on the group)?
- Where should we get a range of IP's from? Mikhal?

Attendees:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- Erik Hjelmås (Supervisor)

Duration:

40 minutes

Discussion:

- Status
- Pled wants to talk, related to login.
- We show this weeks demo/new features.
- Errors in popups vs displayed on the page.
 - Errors on page more modern.
- Labs displaying topics based on tasks, not set in the lab-creation.
- Create specific demo-username etc.
- Write a good, precise report.

05.03.19 - Week 11

Weekly meeting

Talking points:

-

Attendees:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- Erik Hjelmås (Supervisor)

Duration:

30 minutes

Discussion:

- Status
 - Progress is slower than anticipated
 - Should we have more focus on ease of use for other devs later down the line.
 - Code quality
 - Use Best-practice.
 - Clean, simple documentation.
 - Concise.
 - Code design that is easy to use for the average dev.
 - Ask someone who might actually take over the code. (To look at our documentation/code)
 - Other bachelor group?
 - Offer user-testing in exchange etc..
 - Ask some Prog. students maybe?
 - User testing
 - [Espen Torseth \(NCR\)](#)
 - Technically strong. Could be asked, but pref. Bachelor groups.
 - Sander is fighting with some testing software, but getting along fine.
 - Sondre has gotten a local swarm working.
- <https://snyk.io/> - Have a look at this. Seems like a good thing for security.

H.4 February meeting logs

27.02.19 - Week 9

Weekly meeting + demo.

Talking points:

- Showing a demo of our progress. ✓

Attendees:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- Erik Hjelmås (Supervisor)
- Danny Lopez (Employer)

Duration:

1 hour 15 minutes

Discussion:




- Starting with question about how we are doing.
 - We have had a slow week, because of Campus-net being priority. (Part-exams)
- Sondre shows the demo using docker ...
- Do you want the entire deployment automated?
 - Yes. If it is locked down you'll have to dig into the code to change things. (?)
- Tobias showing the wordpress-side of the demo.
- Database++ deletion on shutdown.
 - Could be negated to keep data.
- Multiple themes for usability etc...?
 - Child-themes sound like a better fit for what Danny wants.
- Danny will try to get Basel, he will likely want to see our progress.
 - He was busy.
- Using admin-dashboard for posts and such?
 - Make posts through the dashboard
- "Edit mode" ?
 - Editing things through the admin-dashboard.
 - Ask pros. About opinions on how we should integrate creating/auditing of labs, tasks etc...
 - Ask Lars-Erik?
- We are going to prioritize creation of labs.
- Check for add-user plugins?
- Add to user:
 - Proficiency level
- Focus on the core, ease-of-use for adding users is not focus.
- GDPR is a question. Might need to import users to dodge some GDPR problems.

- QuestBack
- Connect uses from CTF?
- Would we be interested in presenting our demo next week or week after, for NCR meeting?
 - Will get back to us when. We pref. 2 weeks.
 - Possibility for feedback from people who know what they are doing.
- Clarification on different themes
 - Basically just the layout/interface. Functionality needs to stay.
 - Core func. Is stored in the theme.
 - We could probably move interface to child-themes, to allow for easier interface-changes later on.
- Reference everything, even if you have a conversation with someone who might be an expert on a forum etc.
- A wild Basel appears.
- We show him the wordpress part of the demo.
- Divide interface into different parts
 - Topology... etc..
- Editor, drop down menus, drag & drop to create interfaces
 - We will stick to the editor I think, with the potential for drop downs.
- Hierarchy:
 - Lab
 - Vulnerabilities
 - Topic
 - Tasks
- Labs not necessarily linear.
 - Optional.
 - Grey out tasks until prev. Is completed etc.
- Ontology of topics
- A scenario should list what topics it contains.
- Start with set of topics
 - Maybe query them from somewhere else later. (Plugin?)
- From now on we will show a little demo in the weekly meetings, whenever we have made any significant change/progress.

19.02.19 - Week 8

Weekly meeting.

Talking points:

- Should we use other plugins that are already created (e.g., a plugin that forces login etc.)? This will make us dependant on others. 
- A “Newsfeed” showing latest updates and such. 
- File sharing. 
 - We can create a new Google Drive where we share diagrams etc.
 - As we are already using Drive for file-sharing internally in the group, this would be the easiest and most familiar option for us.

Attendees:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- Erik Hjelmås (Supervisor)
- Danny Lopez (Employer)

Duration:

45 minutes

Discussion:

- General status.
 - What we have done this week, and our discoveries
 - Plan for this week etc..
- Q1:
 - Dev [of plugins] might stop, so might be sketchy.
 - Wordfence seems good.
 - Look at it (with Danny) on a per-plugin basis.
 - Check how the dev. Looks, and if the community is active.
 - Make sure to document properly.
 - Can look at plugins as PoC.
 - Talk to Lars Erik Pedersen about NTNU login.
 - PLED was told to talk to him.
 - He knows his stuff.
- Q2:
 - Interesting, would be nice to look into.
 - Use post WP post functions for it?
- Q3:
 - Google Docs is proly fine.
 - Danny will ask Basel.
 - Could use a Phabricator repo.

- Might get access to The Box.
- Sharing a SkyHigh instance with Danny, with the repo running.
 - Hjelmås wants demo next week?
- Git-repo testing?
 - Pipelines.yaml
 - Running code through a “pipeline”.
 - Check if phabricator has this functionality.
 - We can ask Lars Erik about phabricator as well.
 - HarborMaster.
 - Push to BitBucket as well, and do testing there.
 - See obligs for OS-subject.

12.02.19 - Week 7

Weekly meeting

Talking points:

- Should we make misuse cases for rogue instructors/admins.
- Admin role, is it really needed.
 - Assuming Admin(s) have access to backend (database)
- Clarification about what "Topics" actually is.
 - Is it purely a "Topic" with a short description, that labs/tasks are placed under?

Attendees:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- Erik Hjelmås (Supervisor)
- Danny Lopez (Employer)

Duration:

40 minutes

Discussion:

- Status
- Q1:
 - We trust admin.
 - Accidental misuse. We already have.
 - Action confirmation.
- Q2:
 - GUI better than CLI
 - Harder for user to mess up in GUI.
 - Multiple roles for same person?
 - Instructor can be admin?
 - Only one 'active' role at a time.
 - *Hjelmås: Start with two roles.*
 - Admin: Manage plugins.
 - Admin: Stats.
 - Instructor: Also some stats.
 - Average time to solve a case.
 - Time stats at least for sure.
- Q3:
 - Forum and task topics are not directly related.
 - Way to categorize the topics.
 - Tagging with topic, allowing for multiple.

- See tasks listed under topic through clicks etc..
- Name and short desc.
- Discussion board topics:
 - Instructor makes them at least.
 - Moderation, aka instructors will always be able to see them.
- Functional requirements for modules:
 - Will be posted on phabricator.
 - + email for notification.
- Upload diagrams into phabricator.
 - Readme and folders.
- Ordering by difficulty.
- Task attribute:
 - Locked

08.02.19 - Week 6

Meeting with PLED group about API.

Talking points:

- API

Attendees:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- PLED-Group

Duration:

40 minutes

Discussion:

- CVE tall på vulnerabilities + evt. Kjente navn.
 - CVSS score.
 - Type
 - Platform
 - Type/kategori
- Authentication
 - API keys/tokens.
 - Lages i PEMA, sendes med i GET requesten.
- Faste tags ofc.
- Json/XML
- ? som delimiter.
- PLED viser eksempel
- Litt snakk om hvordan autentisering kan gjøres.
 - Inntil videre en key som legges til manuelt.
 - Etterhvert databaser eller noe med exchange av gyldige keys.
- Subqueries for å hent ut liste med ting osv.

05.02.19 - Week 6

Meeting with Basel about use case diagrams

Talking points:

- Ask about opinions on the current state of our use case diagrams ✓

Attendees:

- Tobias Moe, Erlend Einmo, Sondre Granerud. (Group members)
- Basel Katt (Employer)

Duration:

30 minutes

Discussion:

- Mainly discussing how our current use case diagrams look (feat. no internet)
 - Basel thinks these look good.
- Basel is skeptical to our use of the classification “labs”
 - Would rather like to have them called “scenarios”

01.02.19 - Week 5

Meeting with Danny for clarification about functional requirements.

Talking points:

- We had several questions about the functional requirements, these were written down in the "Functional Requirements" doc, and marked with **(Ask Danny)**

Attendees:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- Danny Lopez (Employer)

Duration:

1 hour 19 minutes

Discussion:

- Got clarification about our questions regarding the func. Reqs.
 - These are found in the functional requirements doc.
- Proposed a few ideas of our own.
- Discussed how the point/currency system will work.
 - Black market
 - A "store" maybe?
- Discussed a few potential new features, most of which fall under the "if we have time" category.
 - Embedded terminal. (Like codecademy) (Would be really cool)
- There is a wish for a 'CTF' mode.
 - We have a [link](#) to a proof of concept, but CTFs are pretty similar in format.
- We decided that a new role would be a good idea 'Student Assistant'.
 - Basically instructor but with like, no privs.
 - Mainly answer forum posts.
 - StudAss. role permissions will proly be discussed more in detail later.
- Danny will write functional requirements for the modules Soon™.

H.5 January meeting logs

Meeting logs for January

29.01.18 - Week 5

Subtitle

Talking points:

- Confirming everything to be written in english.
- Confirm thesis deadline. (15th or 20th?)
- Common git repo? (Put full wordpress in git, or just our themes/plugins etc.)
- CI/CD with auto-testing and deployment, or simply just automated testing.
- What diagrams do we need to create?



Present:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- Erik Hjelmås (Supervisor)
- Danny Lopez (Employer)

Duration:

34 minutes

Discussion:

- [Pre-project] Report looks good, nothing really missing.
 - Good enough for now
 - More on the individual modules
 - Specify the extent of “real equipment” with regards to realism.
- Would like local and FEIDE authentication.
 - Two different auths, not mixed.
- Ask IT dept. about SAML or LDAP?
- We will use Wordpress, this okay.
- Not too flashy graphics etc.
 - Not graphical designer bachelor..
- The scoring module modular
 - Different scoring algorithms etc..
- Cross-platform?
 - Wordpress should be pretty simple to make agnostic.
- Security concerns important for our thesis.
 - “Security champion”?
 - All of us should have this role. Because BITSEC.
 - Misuse cases.
- Private repository
- Deadline should be the 20th, but still says 15th.
 - 20th confirmed.

- Gantt scheme is fine.
- Write about the framework in the pre-project?
 - Hjelmås agrees its not the right place "already here".
- Diagrams starting next week:
 - Copy Danny's use-cases.
 - Sensors like diagrams :)
 - Database diagram.
 - Some use cases.
 - High level use-cases.
 - Misuse cases.
- Will hear about the repository

22.01.19 - Week 4

Meeting about Wordpress

Talking points:

- Wordpress
- Maybe other frameworks?



Present:

- Tobias Moe
- Danny Lopez (Employer)
- Carlos

Duration:

20 minutes

Discussion:

- Carlos didn't have much experience with Wordpress (or php in general)
- He said it would be difficult to learn plugins/themes/widgets and to fully understand Wordpress core
- Said we should look into Autolab (<https://github.com/autolab/Autolab>)
- There are benefits and drawbacks to using Wordpress, we need to weigh them ourselves

Group meeting with PLED - 22.01.19 - Week 4

Meeting with PLED to establish the division and exact assignments

Talking points:

- Choosing a framework - Discussed in the group-meeting.
 - Wordpress (+ VueJS) vs. Laravel from scratch.
- Prosjektavtale

Present:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- PLED-Group
- Erik Hjelmås (Supervisor)
- Basel Katt, Danny Lopez (Employer)

Duration:

55 minutes

Discussion:

- Setting up schedule for the semester.
 - Development plan. (scrum version)
 - Chart (gantt)
- Interfaces biggest concern. (Uncertain relevance to PEMA)
 - Deployment manually, or grepping from PLED.
 - Thoughts about API, but probably not needed.
 - Basel says API is required.
 - Separate from the DB?
- PLED-entries need metadata.
 - Telling if something is exploitable or not/compatible etc..
- Figuring out who does what in the link between PEMA and PLED.
 - Templates need to be a thing.
 - Danny is unsure about why the API is required.
 - PLED: Dynamic database.
- Will PEMA and PLED use the same interface?
 - Would be the best solution. - A PLED tab or something.
 - Instructor supposed to be able to use GUI for lab creation.
 - PEMA query PLED metadata.
 - Basel - Outside PLED, everything should be API.
 - Everything that has to do with vulns. should be in PLED.
 - Wants PLED to be separate interface.
 - Maybe admin interface?
 - PEMA still main interface for the instructor.
 - Define interfaces that work separately.
 - Then implement API stuff, and "merge" systems later, using the "subsystems" PLED and PEMA.

- Integration testing later? Would be advantage.
 - Would be nice for the bachelor thesis.
- Using git, or maybe a separate platform developed here on NTNU?
- Integration with Mihkal?
 - Would be cool, but do not rely on it.
- **Integrating concept of CTF into vulnerabilities**
- Should PEMA have GUI-editor?
 - Editor or wizard, (User friendly) for lab-creation. GUI or import from file.
 - Associate with student group
 - Describe the lab.
 - Mihkals work.
 - Specify where to deploy.
 - Possible to save the lab (template?)
 - With all dependencies etc...
- Project should be balanced or unbalanced.
 - Balancing like how you describe the task.
 - A lab can have several difficulties or not.
 - Can have two levels
 - Same topology and such.
 - Differentiated with hints and how the task is described.
 - **Instructor selects** if students choose level themselves or get assigned to a level.
 - Automatic filtering?
 - Based on a "test task" or something. Something to test the proficiency of the students.
- Grading still uncertain.
 - Difficulty and all this is technically out of scope, but would be nice.
 - Interface for evaluation perhaps?
 - Create it as a module. (Big plus - Danny)
- Wordpress?
 - Danny uncertain, ask Carlos?
 - We think it is a good option.
 - Uncertain about 3rd party auth.
 - FEIDE integration?
 - Enough time wasted looking at frameworks, will probably go with Wordpress.
 - (Or maybe VueJS)
 - There is a bachelor's thesis from a few years back using Wordpress. Would be interesting to look at.
 - 2016/17 Not many reports published....
 - Hjemås has a draft from May 15.
 - Development platform for ammunition and rocket company.
 - Scalable-secure wordpress.
- Got an explanation on the Prosjektavtale.

18.01.18 - Week 3 - Extra meeting

Meeting about Functional requirements and Frameworks

Present:

- All group members.
- Danny

Duration:

30 minutes

Functional requirements:

- Interface requirements
 - with regards to available labs/exploits etc..
- Feide login?
 - Login redirection based on role and student "balance" (diff.)
- Every module has its own requirements
 - Scoreboard...
 - Should be able to be enabled or disabled on demand by instructor.
 - (Scoreboard modular algorithms)
 - Students
 - Basic VM control (restart etc) and request larger VMs etc..
 - Should not know through interface how many targets.
- Task based
 - Scaling/balancing
 - Grading per difficulty.
 - Feedback after tasks/projects.
- Knowledge base
 - Forum basically.
 - Charts?
- Black market
 - Currency to buy hints etc.
 - Currency per task.
 - If hints used, points deducted from payout
 - Idea - Difficulty: Good students get less points?
- Important:
 - Score
 - Black market
 - VM management
- We will get a document with details.

18.01.18 - Week 3

Frameworks/patterns meeting

Present:

- All group members.
- Danny Lopez
- Øistein Kolloen

Duration:

23 minutes

Meeting contents:

- Danny explains the functional requirements.
 - Has to be modular.
 - Will be built on and merged in the future.
 - Will be a production system.
- Partially pre-made content
 - [Wordpress plugins](#)? - Has HUGE plugin database.
 - Likely the best option.
 - Carlos - Corner office, has experience with Wordpress.
- From scratch
 - Look into [Vue.JS](#) - Frontend
 - RestAPI - Feeds the frontend.
- Laravel, Django (Python)
 - Sort of same things.
 - Mye back and forth.
- Dropping Moodle.

15.01.19 - Week 3

Talking points:

- Make bachelor more BITSEC oriented ✓
- Moodle, look at pros/cons ✓
- What is this first delivery (1. feb) ✓
- Access to OpenStack and SkyHigh VMs ✓ (Solved, but not at meeting)

Present:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen, Sondre Granerud. (Group members)
- Erik Hjelmås (Supervisor)
- Danny Lopez (Employer)
- Mihkal (Master student)
- PLED-group (for first 10 minutes of meeting)

Duration:

45 minutes

Discussed:

- Explaining our views on moodle.
 - Everyone mostly agreed, and moodle is out of the picture, at least until meeting with Øyvind about frameworks.
- Will look into frameworks for PHP this week.
- [Øivind Kolloen](#) as co-supervisor.
 - Meeting with him regarding frameworks ++.
- Python for backend programming potentially?
- Containers? Scalability.
- Web-app as a docker container.
- PEMA and PLED separate, other than querying PLED database.
 - Queries PoC is basically the only thing required in collab with PLED (For now)
- Main funct.
 - Instructor view and Student view.
 - Create tasks.
 - Deploy labs.
 - GUI (Vs CLI ?)
 - Domain Specific Language translation.
 - Actual deployment, but very quick and dirty, as a proof of concept.
 - GUI Straight into OpenStack SDK or something. But not ignoring Mihkals work, making it easy to implement after completed thesis.

- Modules. Code modularity. - Someone will for sure continue work in the future.
 - Instructor to choose which modules to be used in a given task.
 - Scoring would be interesting
 - Deployment to either all or individual students.
 - Better explained tasks, easier etc...
 - Monitoring.
 - Student progress etc.
 - Separate flag (+ IP) database (with scoring system, Danny's work)
 - We will prolly make PoC

Good webapp bachelor:

[Webapplikasjon for statistisk analyse](#)

INGDAT

Kaja Hannestad, Marius S. Olsen, Jonas T. Nørvåg

Notes on Pre-project report (February submission)

- Look at previous groups handins (Webapp bachelor ^)
 - Usually appendix to thesis.+ basically the first 2 chapters of thesis.
- Not hugely important.
- Content is very important, as it will be what is mostly in chapter 1 and 2 of the thesis.
- The report is what is graded, remember this.
- Requirement descriptions, use cases,...

09.01.19 - Week 2

Present:

- Tobias Moe, Erlend Einmo, Sander L. Berntsen (Group members)
- Erik Hjelmås (Supervisor)
- Basel Katt, Danny Lopez (Employer)

Duration:

- 20 minutes

Discussed:

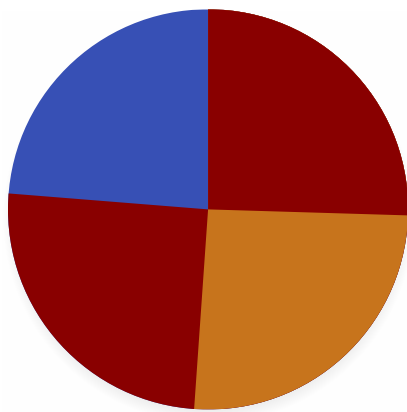
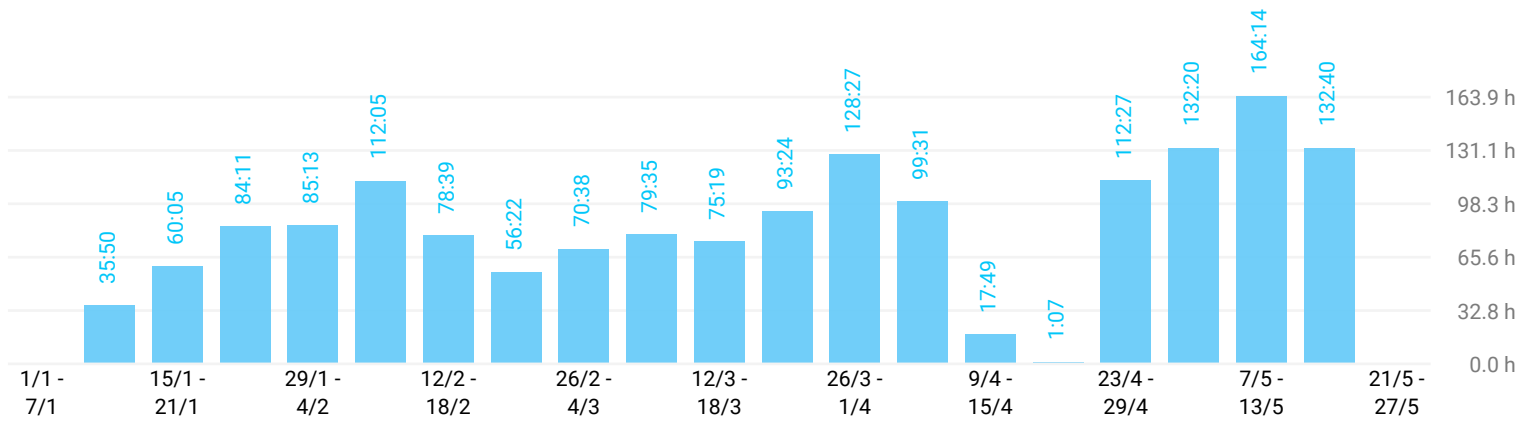
- Scheduled weekly meetings with Erik Hjelmås, Basel Katt and Danny Lopez
 - Tuesdays at 13:30
- We will be working with Mihkal
 - He's creating the deployment of labs for PEMA
- We presented our ideas of how we can make the bachelor more directed towards BITSEC
 - Will be further discussed (and decided) in the next meeting
- Our supervisor advised us to look into Moodle for our web design
- As well as API-based development.

I Time Tracking Output

Summary Report

January 01, 2019 – May 20, 2019

TOTAL HOURS: 1620:04:46

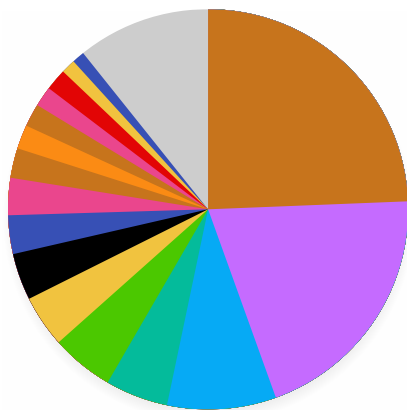


USER

- SL Sander Loken
- MR Mrtobbzi1
- 😊 Einmoerlend
- PH Phenipa

DURATION

- 413:54:36
- 413:09:49
- 409:47:50
- 383:12:31



TIME ENTRY

- Report
- Coding - Theme
- Meeting
- Research
- Database
- Testing
- Docker
- Code review
- Project Planning
- Coding - Wordpress
- Openstack integration
- SSL
- Diagrams
- Discussion
- Administrative
- Use cases

DURATION


- 393:33:28
- 327:44:25
- 142:01:42
- 82:23:51
- 81:44:33
- 67:52:36
- 61:17:59
- 49:36:18
- 48:42:06
- 39:37:24
- 30:45:15
- 30:13:59
- 27:59:59
- 27:21:32
- 18:43:31
- 16:56:00

● Other time entries

173:30:08


USER - TIME ENTRY

DURATION

 Einmoerlend	409:47:50
Administrative	9:43:31
Administrative/Research	1:04:15
Coding	4:55:00
Coding - Theme	117:45:35
Coding - Wordpress	39:37:24
Database	4:52:03
Diagrams	8:08:51
Discussion	0:43:56
General work	0:00:03
Meeting	36:40:04
Misuse Cases	5:30:11
Presentation	1:05:00
Project Planning	10:52:14
Report	123:31:24
Research	18:21:29

USER - TIME ENTRY

DURATION

Research - Wordpress	12:33:58
Research: Diagrams	7:02:26
Seminar	1:15:59
Use Cases	6:04:27
 Mrtobbzi1	413:09:49
Administrative	2:00:00
Brainstorm meeting	1:48:00
Coding - Plugin - OpenStack	14:32:04
Coding - Plugin/Theme	4:29:34
Coding - Theme	209:58:50
Database	2:52:01
Diagrams	4:08:20
Discussion	1:03:36
ERD diagram	3:20:44
Meeting	22:01:00
Meeting Danny	0:40:00
Meeting PLED	0:40:00


USER - TIME ENTRY

DURATION

USER - TIME ENTRY	DURATION
Meeting with Basel	0:30:00
Meeting with Carlos	0:20:00
Meeting with Danny	1:45:00
Meeting with Danny/Øivind	1:05:00
Meeting with supervisor	8:03:14
Misuse case	4:29:50
Preparing Demo	1:18:45
Presentation for NCR	1:00:00
Project Planning	16:32:07
Report	71:30:01
Research	14:35:34
Research diagrams	3:58:59
Research Laravel	2:54:12
Research Vuejs	2:04:28
Research Wordpress	4:58:17
Seminar	2:16:07
Sequence Diagram	2:02:16

USER - TIME ENTRY


DURATION

USER - TIME ENTRY	DURATION
Use case	6:11:50
 Phenipa	383:12:31
Administrative	7:00:00
Backups	9:16:58
Database	13:52:00
Deploying	2:54:11
Diagrams	8:37:47
Discussion	25:34:00
Docker	61:17:59
Finishing touches	7:33:00
General	14:39:29
Heat template research	4:49:00
Meeting	47:59:00
Moodle	0:21:05
Presentation prep	1:30:00
Project Planning	21:17:45
Report	98:35:28

USER - TIME ENTRY

DURATION

Research	6:16:00
SSL	30:13:59
SSL/Report	4:28:50
Use cases	16:56:00

 Sander Loken	413:54:36
Backend	7:57:00
Code review	49:36:18
Database	60:08:29
Diagrams	7:05:01
Fuzzing	7:57:31
Meeting	35:21:38
Openstack integration	30:45:15
Report	99:56:35
Research	43:10:48
Seminar	1:16:17
Static Analysis	2:47:08
Testing	67:52:36

