Jesse Tiitta

Design and Implementation of the Automated Hacker for Cybersecurity Courses

Bachelor's thesis

Information and Communication Technology

2023

South-Eastern Finland
University of Applied Sciences

South-Eastern Finland University of Applied Sciences

Tutkintonimike	Insinööri (AMK)
Tekijä/tekijät	Jesse Tiitta
Työn nimi	Automaattisen Hakkerin Suunnittelu ja Implementointi Kyberturvallisuuden Kursseille
Toimeksiantaja	Kaakkois-Suomen ammattikorkeakoulu (XAMK), CyberLab-oppimisympäristö
Vuosi	2023
Sivut	30 sivua, liitteitä 0 sivua
Työn ohjaaja(t)	Vesa Kankare

Tiivistelmä

Tämä opinnäytetyö käsittelee kyberturvallisuuden kannalta haavoittuneen verkkopalvelimen ja siihen liitetyn relaatiotietokannan rakentamista, sekä automatisoidun Python koodin luomista, joka hyväksi käyttää kyseisiä haavoittuvuuksia. Työ toteutettiin Kaakkois-Suomen ammattikorkeakoululle Kotkan kampuksen CyberLab oppimisympäristöön.

Verkkopalvelimien tietoturvauhkien tunnistaminen on tärkeää, mutta työn pääasiallisena tarkoituksena oli rakentaa automaattinen harjoitusvastustaja kyberturvallisuuden opintojaksoille, joka ei olisi liian ilmiselvä ja staattinen. Verkkopalvelimien ylläpitämät verkkosovellukset valittiin kohteeksi niiden yleisyyden ja haavoittuvuuksien yksinkertaisuuden vuoksi, mahdollistaen yksinkertaisen, mutta helposti jatkettavan koodin rakentamisen.

Opinnäytetyössä hyödynnetään etnografisen tutkimuksenstrategiaa ymmärtämään haitallisen käyttäjän toimintaa, ja tutkimuspäiväkirjan avulla voitiin tallentaa kriittisiä havaintoja ja verrata niitä aikaisempiin kurssikokemuksiin ja tutkimuksiin. Näiden tietojen avulla voitiin kehittää harjoitusvastustaja, joka tarjoaa opiskelijoille realistisen ja monipuolisen oppimiskokemuksen.

Opinnäytetyön tulokset voivat hyödyntää kyberturvallisuuden opetusta tarjoamalla opettajille ja opiskelijoille käytännönläheisen työkalun, joka edistää aktiivista oppimista ja valmistaa opiskelijoita paremmin tuleviin haasteisiin kyberturvallisuusalalla.

Avainsanat: kyberturvallisuus, verkkokehitys, penetraatiotestaus

South-Eastern Finland University of Applied Sciences

Degree title	Bachelor of Engineering
Author (authors)	Jesse Tiitta
Thesis title	Design and Implementation of Automated Hacker for the Cybersecurity Courses
Commissioned by	South-Eastern Finland University of Applied Sciences (XAMK), CyberLab Learning Environment
Time	2023
Pages	30 pages, 0 pages of appendices
Supervisor	Vesa Kankare

ABSTRACT

This thesis focuses on building a vulnerable web server and an associated relational database from the cybersecurity perspective, as well as creating automated Python code that exploits these vulnerabilities. The work was made for the South-Eastern Finland University of Applied Sciences Kotka campus CyberLab Learning Environment.

While identifying security threats to web servers is important, the primary purpose of this work was to construct an automated practice opponent for cybersecurity courses that would not be too obvious and static. Web applications maintained by web servers were chosen as the target due to their prevalence and simplicity of vulnerabilities, enabling the construction of straightforward but easily extendable code.

The thesis utilizes an ethnographic research strategy to understand the actions of a malicious user and uses the field diary method to record critical observations to compare against previous course experiences and studies. With this information, a practice opponent was developed to provide students with a realistic and diverse learning experience.

The results of the thesis can benefit cybersecurity education by offering teachers and students a practical tool that promotes active learning and prepares students better for future challenges in the field of cybersecurity.

Keywords: cybersecurity, web development, penetration testing

CONTENTS

ABBREVIATIONS

GUI	Graphical User Interface
DDoS	Distributed Denial of Service
IP	Internet Protocol

KVM	Kernel-based Virtual Machine
LXC	Linux Container
MITM	Man-In-The-Middle Attack
OS	Operating System
OWASP	Open Web Application Security Project
PHP	Hypertext Preprocessor
SQL	Structured Query Language
VE	Virtual Environment
XSS	Cross-site Script

1 INTRODUCTION

Inspiration for the thesis came from Senior Lecturer Vesa Kankare as there where a demand for more rounded Cybersecurity practice. The idea is that there would be an automated malicious hacker implemented on the South-Eastern Finland University of Applied Sciences (XAMK) CyberLab Learning Environment as a challenger for the Datacentre Application Delivery course.

In Xamk CyberLab Learning Environment is located as a mini data center, which was built as part of the project Cybersecurity Expertise's and Business Model. It is a simulation environment, which allows simulations of large real-life scenarios, where different systems and devices operate in collaboration. Possible scenarios include complex networks, data center virtualization, and Cybersecurity. (Nurmi 2018.)

The course is implemented around Proxmox and Linux Containers (LXC), so a vulnerable web application is going to be built as an LXC template for simple implementation of the course. LXC is powerful and small, so it suits this kind of usage perfectly (Linux Container 2022).

1.1 Background and significance

With the widespread use of web applications for various purposes, ensuring their security has become of paramount importance. However, web applications are often targeted by malicious users who exploit vulnerabilities to gain unauthorized access, compromise user data, and disrupt critical services. As such, understanding and mitigating web application vulnerabilities have become essential to maintaining a secure digital environment.

NTT Application Security reported that 50% of all web applications were vulnerable to at least one serious exploitable vulnerability in 2021. VentureBeat article mentioned that this report was the product of an exhaustive analysis of the data generated from more than 15 million application scans performed in 2021 (VentureBeat 2022.), Gary Sloper and Ken Hess mentioned in O'Reilly's report

“Protecting Your Web Applications” that the 2018 SANS Institute Incident Response Survey for Business applications, which includes web applications, is the most common system type to be involved in breaches. The report mentions that in 2001 web application security was so high-profile topic cybersecurity expert Mark Curphey founded the Open Web Application Security Project (OWASP) (Sloper & Hess 2023.)

1.2 Ethicality and law surrounding hacking

Hacking and penetration testing raises important ethical and legal considerations that cannot be overlooked. Ethicality in hacking involves questioning the boundaries of acceptable behavior and the potential harm caused by unauthorized access and manipulation of systems. Sherri Davidoff (2019) has written The Pentester’s Code of Conduct to allow penetration testers to have guidance and rules to keep everyone safe.

This paragraph reminds everyone that hacking or penetration testing systems without permission is punishable by law. Chapter 34, Section 9a of the Finnish law of Criminal Code (2015/368), states that providing tools, access details, or instructions to cause harm or damage to data processing or the functioning or security of an information system or a communications system, be sentenced for endangerment of data processing to a fine or imprisonment for at most two years.

All methods mentioned in the thesis are used in a closed environment to self-build systems and limit tools to only work for the previously mentioned systems.

1.3 Research objectives

The primary objective of this thesis is to develop a malicious hacker script using Python 3 for web application vulnerabilities, with the creation of Proxmox containers holding vulnerable web applications and databases the web application is relying on. By focusing on real-world scenarios, this study aims to identify common web application vulnerabilities, analyze their impact, and propose effective countermeasures. The malicious hacker script will serve as a

tool to exploit the identified vulnerabilities, enabling a deeper understanding of their exploitation techniques and giving simulated attackers for students to defend against. The web application will showcase the practical implications of these vulnerabilities and highlight the significance of implementing robust defense strategies.

The thesis focuses more on the script than the web application, so the web application is going to be more of a proof of concept.

1.4 Research questions and problems

The main problem that the thesis is trying to solve is that it is hard to implement defensive exercises in cybersecurity courses with a lack of an attacking side or the attacking side being too obvious. The project is being implemented with already existing courses and devices so the problem is not to dismantle anything that already exists, but to improve it. The project needs to be built so that it is easy to modify and upgrade in the future and simple to set up.

Research questions for the thesis are least the following:

- What are the most common problems that web applications could have?
- What problems are for course exercise?
- How can these be implemented?
- How can students resolve these problems?

2 DATA COLLECTION AND ANALYSIS METHODS

The thesis project aims for functionality that is research and development-oriented. The study applies both quantitative and qualitative research methods to have meaningful results.

The thesis material can be split into two, primary and secondary materials. Primary materials are detailed documentation on the project and testing in field diary form. Secondary materials are previously encountered in different courses, and case studies.

The thesis uses diary studies, an ethnographic method to analyze primary and secondary material. Allowing an in-depth exploration of the daily experiences, motivations, and decision-making processes of individuals involved in cybersecurity, this approach provides valuable insights into the malicious hacker mindset and the underlying reasons driving their actions (Salazar 2016.)

Testing is an important part of the project to confirm the functionality of the web application and the malicious hacker script. Testing is done focusing on the unit testing and integration test on malicious hacker script and end-to-end test on the web application case. (Gathoni 2022.)

2.1 Research methods

The project will use mixed methods research that integrates both qualitative and quantitative research. It provides a comprehensive approach to combining and analyzing the statistical data with deeper contextualized insights. (University of Newcastle, Australia 2023.)

Quantitative research focuses on various modes of classification, exploration of causality, comparison, and explanations of phenomena through numeric variables while Qualitative research enables you to increase the overall understanding of the quality, characteristics, and meanings of your research object or topic (Jokinen n.d.)

The project uses a constructive research approach in the form of making something new to solve a problem (Lukka 2003). Something new in this case is the malicious hacker script that is solving problems on the courses that do not have tangible opponents to improve the learning experience. The script brings intervention research (Melnyk & Morrison-Beedy 2012) with it because it focuses on chance and together, constructive and intervention research form a method of cyclical development (Roger 2020).

A big part of the research and data collection is going to be testing. Open Web Application Security Project (OWASP) has made a Testing Project that helps web application developers understand what, why, when, where, and how to test web applications. On this testing guide the focus is going to be on section 4.0 Web application security testing. (OWASP) 2023a).

2.2 Research goals

The thesis project is aiming for a functional research and development project, which may achieve many objectives. Primary objectives are least the following:

- Building vulnerable web applications in the form of proof of concept.
- Produce scanning technique for the script and what it is looking for.
- Build the script to have different kinds of attacks to manipulate web applications.
- Automate script fully so it does not need user input for scanning or attacking but gives a clear indication of whether something works or not.
- Implementing everything seamlessly to the course laboratory scenario.

3 THEORETICAL FRAMEWORK

The theoretical framework of this study provides a conceptual underpinning for understanding and analyzing web application vulnerabilities. It encompasses various components that contribute to the overall knowledge base and inform the development of the malicious hacker script. By integrating these theoretical perspectives, the study aims to enhance the understanding of web application security and guide investigation into vulnerabilities and their exploitation.

3.1 Malicious hacker script

A script is going to be built with Python 3 because it is going to bel implemented to already existing code. The existing code includes automated user activity in the course network, and it is built by the Course Lecturer Vesa Kankare. The idea is that the hacker is going to be hidden behind normal activity.

Python 3.0 was released December 3rd, 2008, and Python 2 reached End of Life status in January 2020. The newest Python version is 3.12.0. Python standard library and third-party modules allow for endless possibilities (Python Software Foundation 2023a.)

There are dozens of different kinds of vulnerabilities, but the focus is on the most common ones that can be seen in OWASP Top 10 (Open Web Application Security Project (OWASP) 2021).
To make a script look like a hacker we need to understand how hackers work and what kind of tools they use. As Andrew Hoffman mentioned in his book (2020) the first part of exploiting web applications is recon and the second part by the book is an offense, so a script needs to have ability to find the vulnerable parts and have ability to utilize them and mess with the web application.

Some of the legwork has already been done. In the school course Advanced Project, I and my teammates produced a proof of concept script for the hacker. The script can scan networks and do simple Cross-site script (XSS) and SQL injection. In this project, the aim is to improve the already existing script to have a more advanced XSS attack, implement automated SQL injection, and have the possibility for brute force attack.

3.2 Proxmox

Proxmox is open-source software that simplifies server management. Proxmox Virtual Environment (VE) lets you have compute, network, and storage in a single solution. It is integrated with Kernel-based Virtual Machine (KVM) hypervisor and Linux Containers. Management is done in the web-based user interface that allows Virtual Machine and container management.

Proxmox VE eases virtualization of IT infrastructure, optimizing existing resources and increasing efficiencies. With it, you can virtualize Linux and Windows application workloads, and dynamically scale computing and storage. Proxmox has three main products: Proxmox Virtual Environment, Proxmox Backup Server, and Proxmox Mail Gateway. (Proxmox Server Solution 2022.)

Linux Containers, often referred to as LXC, allow virtualization close to a virtual machine, but without the overhead that comes with running a separate kernel and simulating all the hardware. Linux Container is considered as something in the middle between a chroot and a full-fledged virtual machine allowing an environment as close as possible to a standard Linux installation without the need for a separated kernel. (Linux Container 2022.)

Utilizing Linux container images, referred to as templates, enables running containers in the most common Linux distributions, but gives the ability to make images of the existing container. This means that you can start the container in one of the found Linux distribution's images, build your needed dependencies and services, and then make a new image out of the container. This implements the web application and database seamlessly into the course environment.

3.3 Web application

A web application is a computer program that uses a web browser to function. Web applications differ from web pages by functionality. A web page is static and provides visual and text content that the user can see and read, but not affect in any way. A web application on the other hand enables the user to not only read the page content but manipulate the data on the page. The interaction takes the form of a dialog: the user clicks a button or submits a form and gets a response from the page. This response may take the form of a document download, online chat, electronic payment, and more. (Belsky 2017).

Throughout my studies, I have participated in courses and projects that have given me an understanding of web applications and their security. In one of the project courses my teammate and I developed a vulnerable web application using Hypertext Preprocessor (PHP) and one of my optional courses was Secure Web Services.

By leveraging the knowledge and skills gained from these experiences, I am well prepared to approach the development and assessment of the web application in the study project.

4 PROTOTYPE IMPLEMENTATION

A prototype implementation of the systems was developed on my workstation using Oracle VirtualBox, which is a versatile virtualization platform designed for both enterprise and home use (VirtualBox n.d.) This approach involved running a type 1 hypervisor on top of a type 2 hypervisor, which is not recommended due to the complexity of the system, but it allowed working with minimal resources.

VirtualBox was utilized as the virtualization platform to host the Proxmox environment, which would hold the containers for the web application and database. The container operating system (OS) was chosen to be Alpine Linux for its lightweight and resource efficiencies (Alpine Linux 2022.) Additionally, an Ubuntu workstation was set up within VirtualBox to be a test workstation and facilitate the development of the hacker script and enable testing the web application as a user.

Running Proxmox as nested virtualization was not completely problem-free. To get Proxmox installed and the graphical user interface (GUI) working, VirtualBox needed some additional configurations. Proxmox has a guide on what needs to be changed on VirtualBox, (Proxmox 2020) so I used this guide in combination with the Sidheeq instructions, which guided the installation in more detail (Sidheeq 2023.)

The most important changes before installation were:

- Hardware virtualization acceleration must be activated from EFI/BIOS.
- From VirtualBox the Proxmox Processor needs to have Nested VT-x/ADM-V enabled as shown in Figure 1.
- For Internet access network adapter needs to be attached to NAT and port forwarding done to port 8006 as shown in Figure 2.

- For connection between Proxmox and the test workstation virtual network needed to be done in VirtualBox Network Manager and a secondary adapter connected to this virtual network.

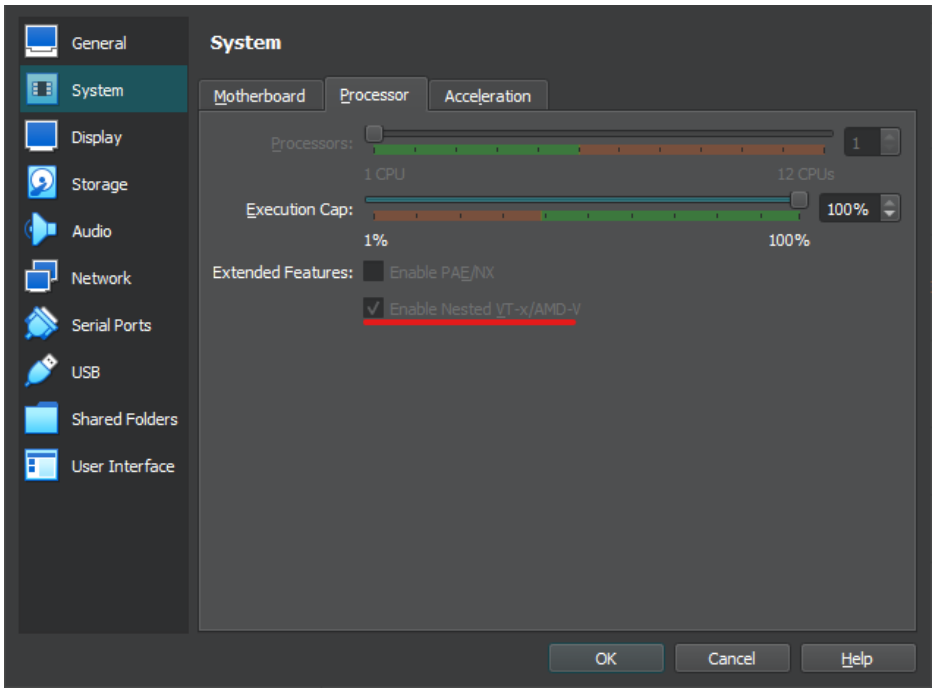

Figure 1. Processor configuration

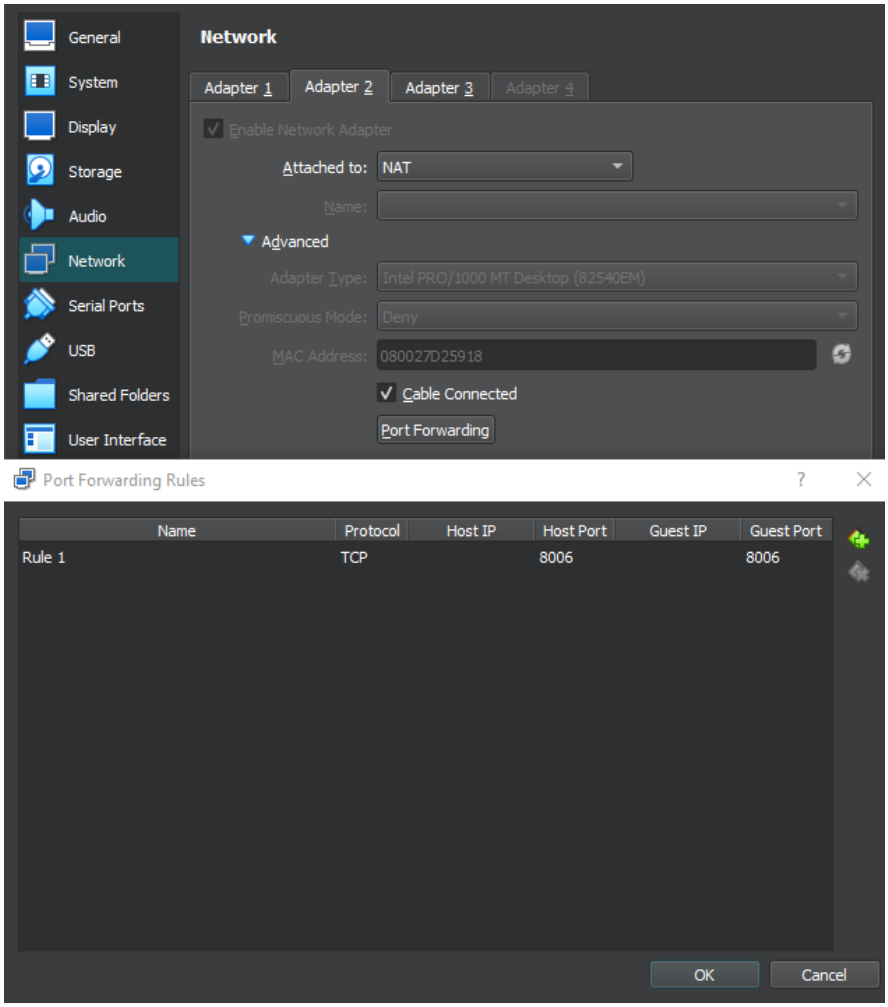

Figure 2. Network adapter configuration

With these configurations, Proxmox, and the test workstations have a connection to each other and the internet. For the containers, they needed a secondary network adapter that needed to be connected to the virtual network to be able to communicate with the test workstation. With these initial setups, the environment was ready to be built upon.

4.1 Web application and database

At the early stage of the study, the selections of functions that had to be on the web application, thereby limiting the scope of the web application and keeping it as simple as possible. These functions were:

- Users must be able to log in and out.
- There must be some kind of search functionality.

- There must be a comment or note field.

These features enable three kinds of attacks that I wanted the malicious hacker script to least have. The attacks that I wanted were: brute force, SQL injection, and cross-site script (XSS).

Deciding which tools to use on the web application was not easy because many frameworks, tools, and applications were available. Choosing a database does lean on a web application because all frameworks do not work with all databases.

4.1.1 Choosing the framework

The first decision was to develop the web application using PHP as a programming language. Although there was prior experience with PHP, Alpine Linux was a new operating system for this project. Challenges arose during attempts to install PHP packages within the container, despite various troubleshooting efforts.

Seeking a solution led to the discovery of several outdated pages and tutorials suggesting potential fixes. Unfortunately, none of these proved successful in addressing the package installation issue. Exploring alternative languages or frameworks became necessary for the development of the web application. Fortunately, this minor setback did not impact the research questions, as vulnerabilities in web applications are generally relevant across different languages and frameworks.

The second choice to make the web application was WordPress. WordPress is an application that is built on PHP and MySQL to make implementing web applications simple. (WordPress 2023.)

Installing WordPress and configuring an external database proved to be a straightforward setup. The graphical user interface for website construction was intuitive, with a surprisingly short learning curve. However, when attempting to

implement the required functionality in a web application, it became apparent that the approach was not viable.

Nested virtualization made WordPress slow in most parts and in WordPress, functionality is built using plugins, but most of these come at a cost. This limited the usage of the plugins and made the application more complex than needed.

After researching methods to develop the web application, a YouTube video from Tech With Tim caught my attention. The video demonstrated the creation of a Python website using Flask, incorporating features like authentication and database integration. Authentication posed a significant challenge at that point, making this tutorial particularly helpful. (Ruscica 2021.)

Flask is a Python-based microframework that is designed to keep the core of the application simple and scalable. Flask depends on the Werkzeug WSGI toolkit, the Jinja template engine, and the Click CLI toolkit that provides most of the functionality. (Pallets 2023a.)

Flask ended up being my tool of choice and it did fit perfectly in the way that malicious hacker script was going to be written in Python so the programming language would be identical between a web application and a hacker script. Flask gave me more confidence in the way that I did not need to start learning something completely new but could rely on my previous knowledge and experience.

4.1.2 Making the web application

Making the web application started making containers on the Proxmox using Alpine Linux as the operating system. After the container was made it needed a second network connection to the test workstation to allow it to connect to containers. Internet connection was established by making sure when a container was made that it was on the same subnet as Proxmox and using the same gateway. This allowed downloading needed dependencies and packets to start following the Tech With Tim tutorial and making the web application.

Most of the web applications were made following the Tech With Tim tutorial with some key changes on the functionality and adding a search page as the home page. Tutorial helped create the login, sign up, and note pages, understanding a web applications file structure and how different modules worked. The most provident visual changes I made is that on the header of a web application, it shows the connected workstation IP address and after login the username that is logged in on every page, also a CS file that provides a visual look for a web application is self-made so in the future container didn't need to have a working internet connection to look bit better.

The login page as shown in Figure 3 asks users to input their email address and password, these are then tested against the database to see if credentials exist giving the user information about the email address, and then if the password is incorrect. The main chance to the login page was that the email address field input type is not email, but text to allow login as a user without properly formatted email addresses, and in password checking hashing is not implemented. This I am going to explain better on web application vulnerabilities.

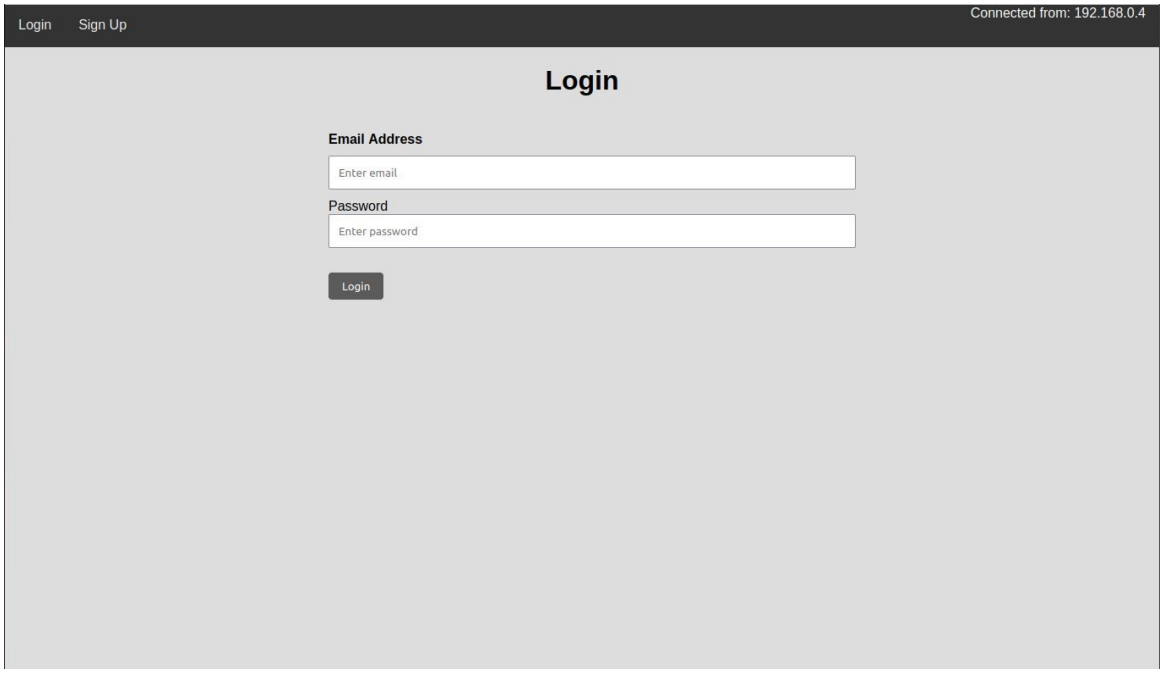

Figure 3. Login page.

The sign up page as shown in Figure 4 is mostly from the tutorial. The form on the signup page asks the user to input their email address, first name, password, and password confirmation. Email address is used as user login credentials and first name is just held on the database to differentiate users. The main modification on this was that the password is not hashed when stored in the database.

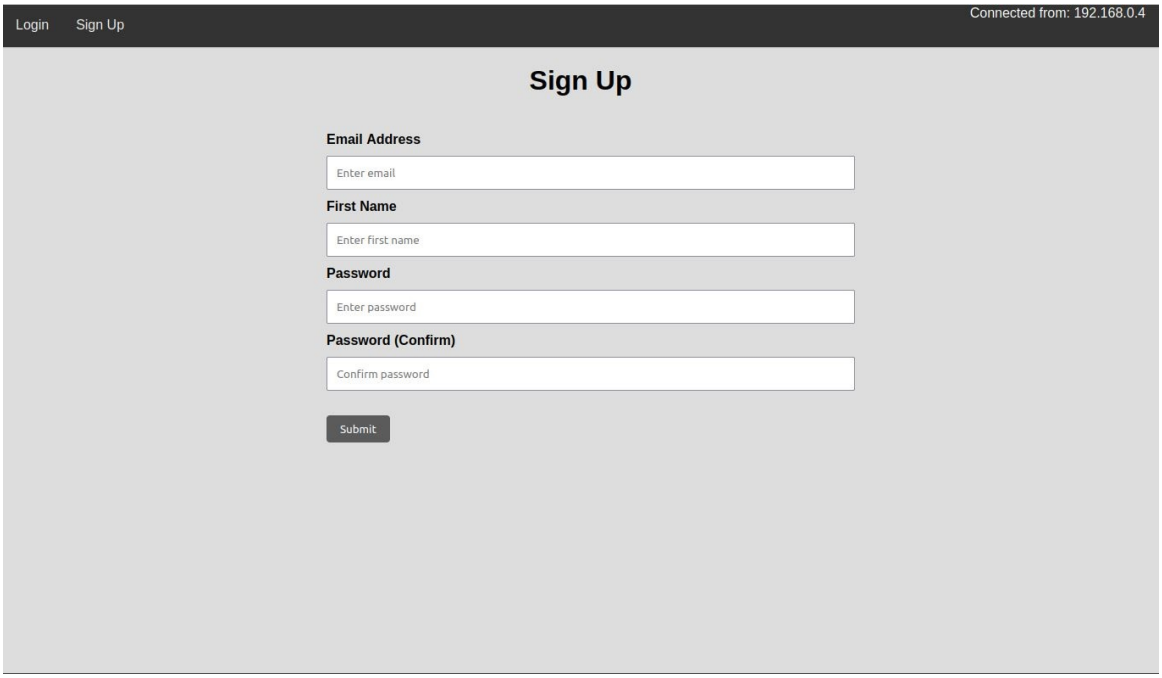

Figure 4. Sign-up page.

The home page is completely self-made, and it includes a search field and title Tool listing as shown in Figure 5. I held on to the idea that the web application could be some kind of accounting for some imaginary warehouse. So, the search field finds all the product that the warehouse holds and how many there is left. There is no way from a web application to add or remove items from the database or change their amounts, but this feature was unnecessary looking at the project focus. The search field finds items based on does the items in the database hold the characters that are sent using the LIKE operator, leaving the search field empty when sending the information does not bring any listings.

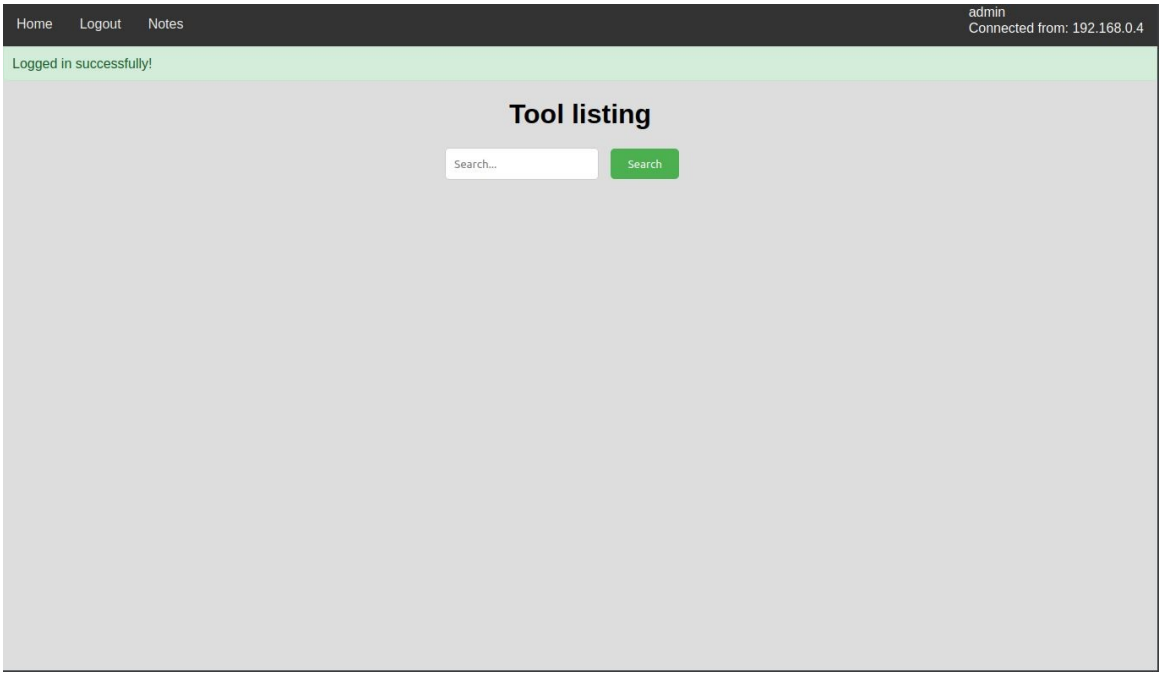

Figure 5. Home page

The last page for the web application is the Note page which holds a text area where the user can leave notes, and these are connected to the user on the database as shown in Figure 6. Making it so that other users don’t see other users' notes on their page. This is mostly made following the Tech With Tim tutorial and functionality is the same as on the tutorial, but the HTML page and how the notes are shown is modified to allow cross-site script vulnerability to happen.

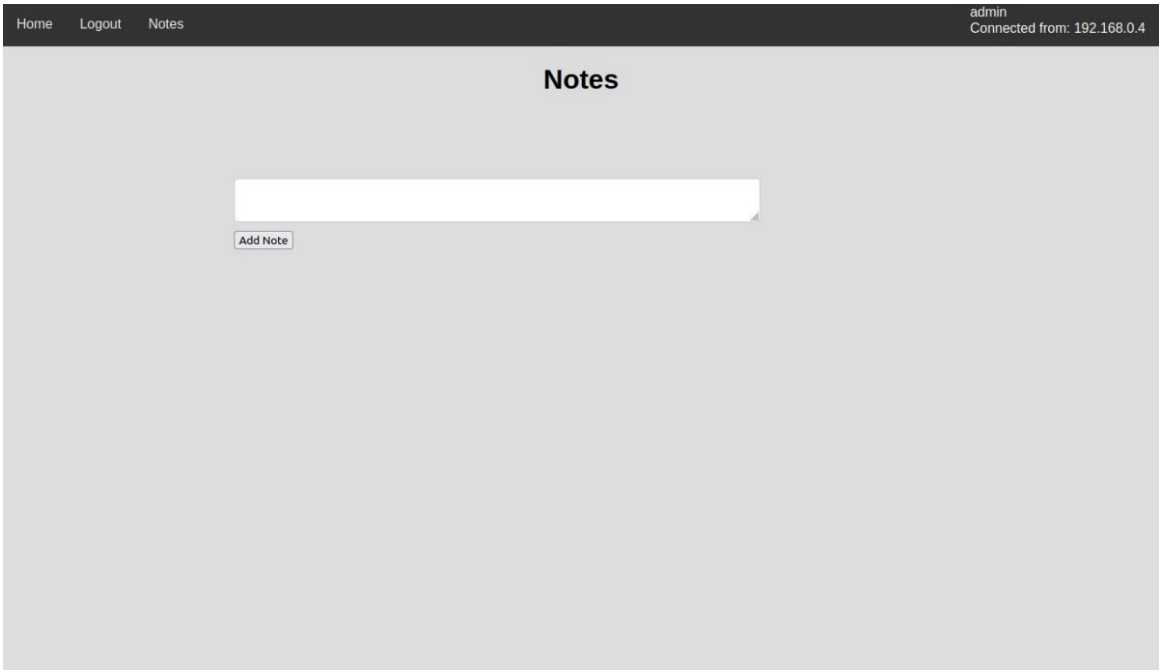

Figure 6. Note page

4.1.3 Making the database

For the database of the web application, I decided to use MySQL. Given my previous experience with MySQL, but to my surprise Alpine Linux repositories no longer include the actual MySQL. Installing MySQL packages will instead install MariaDB (Alpine Linux 2023). Installing MariaDB instead of MySQL did not cause major problems due to its compatibility and similarities with MySQL. (MariaDB Foundation 2023.)

One of the requirements for the project where that the database would be installed in separate containers, independent of the web application. This separation allowed for easier management and modification of each component individually, enabling more efficient scaling and maintenance of the overall system.

For the installation, I followed Alpine Linux's tutorial adding a password for the root user, removing anonymous users, removing the test database, and reloading privilege tables. I did not set up socket authentication that would ensure that only administrators could log into the engine and did not set that root user could not log in remotely. I did run commands that the installation tutorial recommended for

development servers allowing databases to be accessible anywhere from the internet. This small command enables easy setup for the web application to establish a connection, but malicious user if they get credentials to the database, could connect their application to the database or change configuration on the database.

I made a user for the database that could connect from any IP address and had full privilege rights for the database to make the vulnerability even more serious. To prevent this kind of vulnerability you need to think hard about what users you give what kind of permissions to and allow those permissions only from one or if needed few sources to narrow attack vectors as much as you can.

Making the database was a lot later in the timeline than making the web application, so I did want to install phpMyAdmin on the database for visual modification and make one more attack vector to the database. PhpMyAdmin is a free software written in PHP, intended to handle the configuration of MySQL over the web. It enables users to make frequently used operations managing databases, tables, columns, relations, indexes, users, permissions, and other functions, and all this is performed via the user interface while still giving the ability to directly execute any SQL statements. (phpMyAdmin 2023.)

At this point, I remembered my previous problems installing PHP packages to the Alpine Linux but thought that I would give it another try. Shortly after trying I faced the same problems with downloading and installing packages, so instead of going to find a solution in the previous style, I thought I would try to solve the problem using ChatGPT.

ChatGPT is an artificial intelligence language model developed by OpenAI. It has been trained with a vast amount of text data from the internet allowing it to respond to a wide range of questions and prompts (OpenAI 2023).

ChatGPT's first recommendation was to make sure I had all the Alpine Linux repositories. ChatGPT gave me the repositories, but not the location where the

file was located so I turned to Alpine Linux wiki to find the information. The repositories were located at /etc/apt/repositories (Alpine Linux 2023) and the repositories that I had were:

https://dl-cdn.alpinelinux.org/alpine/v3.16/main
https://dl-cdn.alpinelinux.org/alpine/v3.16/community
https://dl-cdn.alpinelinux.org/alpine/edge/testing

What I needed to have:

http://dl-cdn.alpinelinux.org/alpine/v3.16/main
http://dl-cdn.alpinelinux.org/alpine/v3.16/community
http://dl-cdn.alpinelinux.org/alpine/edge/main
http://dl-cdn.alpinelinux.org/alpine/edge/community
http://dl-cdn.alpinelinux.org/alpine/edge/testing

So, I was missing edge repositories and changed all of them to use HTTP instead of HTTPS. This was not on the wiki site, but I thought while adding them that I just try with this and chance if it didn't work. This chance allowed download and install packages without a problem, and I was a bit frustrated the fix was something so small.

To get the phpMyAdmin to work I needed a web server on my database, so I installed the lighttpd for its high performance and lightweight. A Lighttpd uses memory and CPU efficiently and is a very flexible web server compared to other popular choices (Lighttpd 2023).

At the installation stage, I did not have any problems, but I could not get the lighttpd to start, so I turned back to try troubleshooting the problem using ChatGPT. ChatGPT informed that Lighttpd was unable to start because it was unable to execute the '/usr/bin/php-cgi' binary. After a bit of back and forth with ChatGPT trying different things it concluded that installing the 'php7-cgi' package installation was not completed successfully so it recommended trying to install

the ‘php7-cgi’ package from the source. This involved installing necessary build tools, downloading the PHP source code, and building the package myself and installing that. Thanks to this complicated solution, I got ‘php7-cgi’ installed and solved the problem that made it impossible to start Lighttpd.

On phpMyAdmin I made the database, tables, and populated user and product tables to have some random users on the database and to have products that the search field could find and I could print onto the home page. For the random users, I used Fake Name Generator which allows users to generate between 0 to 100,000 users with different randomized attributes (Corban Works 2023).

4.1.4 Web application vulnerabilities

Just following the tutorial, the web application would have ended up relatively safe from vulnerabilities, but that would have been against my research goal, so I started figuratively drilling holes in a web application's defense.

Firstly, I changed the login page email address field, so it allows text instead of properly formatted email addresses, this allowed making local administrator with username admin that simulates applications default administrator account. The reason for this change is to simulate broken authentication or as OWASP calls it Identification and Authentication Failures. The login page is not protected against credential stuffing, or brute force, and allows weak well-known usernames and passwords. (OWASP 2023b.)

To prevent this kind of attack default users should be disabled or passwords should be as secure as possible. Login should allow only a limited amount of tries with wrong credentials and give a delay after the limit. The web application should not allow weak passwords and force users to use lower case, upper case, numbers, and symbols on their passwords. People tend to use simple passwords and reuse them everywhere so if it is possible to check a password against a password list that holds weak passwords or even better already leaked passwords would steer users to use safer passwords.

Secondly, I made the home page search function to build the query to the database raw, so it was easily manipulated. Query was sent to the database in the form of SELECT * FROM product WHERE name LIKE “%{query}%” where the query was user input. This allows manipulating the query and implementing SQL injection that allows attackers to view data that they should not be able to retrieve. (OWASP 2023c.) With simple modification that I have implemented to the malicious hacker script makes the query look like “SELECT * FROM product WHERE name LIKE “%zzzz” UNION (SELECT first_name, password, email FROM user);#’. Zzzz makes it so that none of the products is found, and the rest of the query brings all users from the user table printing their first name. password and email giving attacker access to all users on the database. All the extracted passwords were in plain text because the hashing was removed from the passwords.

To prevent this kind of attack zero trust needs to be implemented for all user input. All user input needs to be validated, filtered, or/and sanitized by the application. This is unfortunately sometimes problematic because some applications require special characters like text areas or an application programming interface (API) for mobile applications. To filter user input whitelisting is always a better way than blacklisting. It is easier to allow something and forbid everything else than try to forbid every possible combination that someone could use with malicious intent. All data that is stored should be encrypted in this case the passwords should be encrypted to slow malicious usage of the passwords and have a chance to stop automated attacks.

The third and last vulnerability that I made for the web application was on the note page. Flask uses templates from Jinja to store the HTML information and Jinja auto escapes all HTML tags and special characters from users' stored information like comments (Pallets 2023b). Telling the Flask application that notes that users leave are “safe” prevents this escaping and allows users to implement the cross-site script on the text area. The note page does demonstrate broken access control. The link to a note page shows only on the admin account,

but this does not prevent other users from accessing the page. By modifying the URL every user can access the note page. (OWASP) 2023d.)

To prevent the cross-site script is like the SQL injection in a form that is included in injections. User input needs to be validated, filtered, or/and sanitized to prevent users from including malicious code in their inputs. For broken access control users that should not be able to access, create, read, delete, or update should be denied in the form of whitelisting. Making robust defenses and opening them only enough that normal users can access what they need is better than opening too many doors for the sake of usability.

These were the vulnerabilities that I focused on. The web application may hold more vulnerabilities, but I did not have enough resources or know-how to do full-scale testing of it. The only fields that raised concern for more possible vulnerabilities are login and sign-up pages and their forms.

5 IMPLEMENTATION

For the final implementation of the project, I need to import everything I've done into the course's virtual environment. I made some minor changes to both servers and script to make their use much easier and informative and to hopefully help develop them further in the future.

5.1 Web server and database

Chances that I made for web server before implementing where to help future users as much as possible. The most important chance where to comment code as much as possible to make understanding it much easier. I included the logging function on the web server on the debug level so if there were actions on the server it was shown on the screen and included in the "error.log" file for future reference. The logging function can be found on the main.py file in the root directory and chance that I would suggest lowering debug to the info level so the log file is not overfilled by unnecessary information.

I included fixes for the vulnerabilities in the code, but those fixes are commented out and are hard to find if there is no previous knowledge reading the code. To help with this I included a “help.txt” file on the server webserver folder. This should point towards what to change and where.

For final implementation, I made the templates of the web server and database to have snapshot copies of the containers and provided these copies to Senior Lectures Vesa Kankare. This allows students or teachers to start as many containers as needed and change the configuration setup for the containers without breaking the functionality of the web server or database.

5.2 Malicious hacker script

On the script side changes were the same as on the servers. I added comments on every major line to explain what the code was doing, other major changes were to include the sleep function between the biggest parts of the code. This chance idea was to make the script seem a bit more human and take time between different actions instead of making everything at inhuman speed.

I added logging on the script side to help see was the script's last step was or what it has done before. Logging was done the same way as on the webserver to show on the screen and writing on the file. The file name in this case is “hackerbot.log” and it is saved on the root directory as other parts of the code.

6 RESULTS

With the malicious hacker script, students will have an active attacker that they can use to defend against. While the web server and the script are more on the basic side does not mean that they would be obsolete for understanding and learning cybersecurity.

With these additions to the course, students would be able to have web servers and databases and see what kind of network traffic they are having between them. They would be able to see network traffic between web servers and

malicious hacker script in real-time and see what kind of difference it makes related to the normal traffic.

Including these tools in the course, should help understand the importance of network traffic and logs and how to read them. It shows how small differences in web development can prevent huge vulnerabilities that can take the whole website down or allow someone to steal or even manipulate saved information.

This opens the possibility to further develop both sides in the future. Maybe more complex web servers with different frameworks to help understand different tools. The hacker script could be more complex in ways that are testing systems on the defending side and maybe in the future it will not just focus on the browser side, but also the server itself.

7 CONCLUSION

While web servers can be built with different tools and frameworks, they still have the same vulnerabilities in one way or another. The amount of web services that we use in our daily lives is staggering, ranging from online shopping platforms and social media networks to banking portals and healthcare systems.

Most common vulnerabilities are simply over-trusting users or data that they input and malicious users will benefit from it. The Internet is full of different kinds of tools and services to find these kinds of weaknesses, but these tools don’t always make understanding the weaknesses any easier.

The web server and script are simple in terms of what we can find on the internet in the form of what web servers can do and how hackers try to break them. I hope this project makes learning web application vulnerabilities easier and more interesting.

For future development, the malicious hacker script could extend its capabilities beyond its current limitations. Currently, the script is designed to target only this web application, and its functionalities are constrained by the tools at my

disposal. One significant enhancement I envisioned for the script is the inclusion of a Distributed Denial of Service (DDoS) attack. However, my ability to implement this was restricted by the limitations of my personal computer and its hardware. The lack of resources prevented me from creating additional bots that could potentially overwhelm the service and bring it down.

Another idea that I could not do relating to the web server was that malicious hacker script now implements a dummy payload that looks like information is forwarded to a separate server, but there is no server to connect. This could be a good development idea to build a server for man-in-the-middle (MITM) attacks or just for hackers to save information.

Also, as I mentioned in the results the malicious hacker script does not affect the server in any way. This could be something that would be interesting to develop or maybe some future student’s thesis project.

REFERENCES

Alpine Linux Development Team, 2023. MariaDB. Web page. Updated 26 September 2023. Available at: https://wiki.alpinelinux.org/wiki/MariaDB [Accessed 2 December 2023].

Alpine Linux Development Team. 2022. Small. Simple. Secure. Web page. Updated 23 November 2023. Available at: https://alpinelinux.org/about/ [Accessed 2 December 2023].

Belsky, V. 2017. ScienceSoft: Web application vs. website: finally answered. Blog. 9 November 2017. Available at: https://www.scnsoft.com/blog/web-application-vs-website-finally-answered [Accessed 2 December 2023].

Criminal Code 2015/368.

Corban Works, LCC. 2006-2023. Fake Name Generator - Order In Bulk. Web page. Updated 5 December 2023. Available at: https://www.fakenamegenerator.com/faq.php [Accessed 5 December 2023].

Davidoff, S. 2019. The Penterster's Code of Conduct – Rules that Keep Everyone Safe. Blog. 19 November 2019. Available at: https://www.lmgsecurity.com/the-pentesters-code-of-conduct-rules-that-keep-everyone-safe/ [Accessed 2 December 2023].

Fortinet 2023. What Is A Brute Force Attack? Web page. Updated 7 December 2023. Available at: https://www.fortinet.com/resources/cyberglossary/brute-force-attack [Accessed 10 December 2023].

Gathoni, M. 2022. 4 Testing Methods Every Developer Should Know. *Make Use Of*, 13 September 2022. Electronic newspaper. Available at: https://www.makeuseof.com/testing-methods-developers-should-know/ [Accessed 2 December 2023].

Grigas, L. 2022. Learning Password Security Jargon: Dictionary Attack. Blog. 1 April 2022. Available at: https://nordpass.com/blog/what-is-a-dictionary-attack/ [Accessed 18 May 2023].

Hoffman, A. 2020. O'Reilly: Web Application Security – Exploitation and Countermeasures for Modern Web Applications. Pdf-document. Available at: https://www.oreilly.com/library/view/web-application-security/9781492053101/ [Accessed 24 February 2022].

Jokinen, A. n.d. Laadullisen tutkimuksen näkökulmat. Teoksessa Vuori, J. (toim.) Laadullisen tutkimuksen verkkokäsikirja. Tampere: Yhteiskuntatieteellinen tietoarkisto. Web page. Available at: https://www.fsd.tuni.fi/fi/palvelut/menetelmaopetus/kvali/mita-on-laadullinen-tutkimus/laadullisen-tutkimuksen-nakokulmat/ [Accessed 29 November 2023].

Linux Containers. 2022. What's LXC? Web page. Updated 5 December 2023. Available at: https://linuxcontainers.org/lxc/introduction/ [Accessed 5 December 2023].

Lukka, K. 2003. The Constructive Research Approach. In Ojala, L & Ilmola O-P. (eds.) Case study research in logistics. Turku: Publications of the Turku School of Economics and Business Administration, Series B 2003: 1, 83-101. Turku: Turku School of Economics and Business Administration. PDF-document. Available at: https://www.researchgate.net/publication/247817908_The_Constructive_Research_Approach [Accessed 2 December 2023].

Lighttpd. 2023. lighttpd wiki and documentation. Web page. Updated 31 October 2023. Available at: https://www.lighttpd.net/ [Accessed 2 December 2023].

MariaDB Foundation. 2023. MariaDB Server: The open-source relational database. Web page. Updated 5 December 2023. Available at: https://mariadb.org/ [Accessed 5 December 2023].

MechanicalSoup. 2022. MechanicalSoup – Welcome to MechanicalSoup's documentation! Web page. Updated 4 July 2023 Available at: https://mechanicalsoup.readthedocs.io/en/stable/ [Accessed 2 December 2023].

Mechanize. 2017. Mechanize. Web page. Updated 16 May 2023. Available at: https://mechanize.readthedocs.io/en/latest/ [Accessed 2 December 2023].

Melnyk, B. & Morrison-Beeny, D. 2012. Intervention Research: Designing, Conducting, Analysing, and Funding. New York City: Springer Publishing. E-Book. Available at: https://connect.springerpub.com/content/book/978-0-8261-0958-3 [Accessed 2 December 2023].

Nurmi, J. 2018. Statistics of First Year of XAMK Virtual Laboratory as a Cloud Service. Web page. Updated 5 December 2023. Available at: https://next.xamk.fi/uutta-luomassa/statistics-of-first-year-of-xamk-virtual-laboratory-as-a-cloud-service/ [Accessed 5 December 2023].

OpenAI. 2023. Introducing ChatGPT. Web Page. Updated 5 December 2023. Available at: https://openai.com/blog/chatgpt [Accessed 5 December 2023].

Open Source Foundation for Application Security (OWASP). 2021. OWASP Top 10. Web page. Updated 25 May 2023. Available at: https://owasp.org/Top10/ [Accessed 3 December 2023].

Open Source Foundation for Application Security (OWASP). 2023a. OWASP Web Security Testing Guide. Web page. Updated 3 December 2023. Available at: https://owasp.org/www-project-web-security-testing-guide/ [Accessed 3 December 2023].

Open Source Foundation for Application Security (OWASP). 2023b. A07:2021 – Identification and Authentication Failures. Web page. Updated 25 May 2023.

Available at: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/ [Accessed 3 December 2023].

Open Source Foundation for Application Security (OWASP). 2023c. A03:2021 – Injection. Web page. Updated 25 May 2023. Available at: https://owasp.org/Top10/A03_2021-Injection/ [Accessed 3 December 2023].

Open Source Foundation for Application Security (OWASP). 2023d. A01:2021 – Broken Access Control. Web Page. Updated 25 May 2023. Available at: https://owasp.org/Top10/A01_2021-Broken_Access_Control/ [Accessed 3 December 2023].

Pallets. 2023a. Flask. Web page. Updated 21 June 2023. Available at: https://flask.palletsprojects.com/en/2.3.x/ [Accessed 2 December 2023].

Pallets. 2023b. Jinja - API. Web page. Updated 27 June 2023. Available at: https://jinja.palletsprojects.com/en/3.1.x/api/ [Accessed 2 December 2023]

Proxmox. 2020. Proxmox VE inside VirtualBox. Web page. Updated 15 September 2020. Available at: https://pve.proxmox.com/wiki/Proxmox_VE_inside_VirtualBox [Accessed 5 December 2023].

Proxmox Server Solution. 2022. Proxmox Virtual Environment. Web page. Updated 5 December 2023. Available at: https://www.proxmox.com/en/proxmox-ve [Accessed 5 December 2023].

Python Software Foundation. 2023a. Python About. Web page. Updated 5 December. Available at: https://www.python.org/about/ [Accessed 5 December 2023].

Python Software Foundation. 2023b. python-nmap 0.7.1. Web page. Updated 5 December. Available at: https://pypi.org/project/python-nmap/ [Accessed 5 December 2023].

Richardson, L. 2004-2023. Beautiful Soup Documentation. Web page. Updated 5 April 2023. Available at: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ [Accessed 2 December 2023].

Roger, S. 2020. Different Methodologies in Software Development Life Cycle. Blog. 20 August 2020. Available at: https://www.h2kinfosys.com/blog/different-methodologies-in-software-development-life-cycle/ [Accessed 7 March 2022].

Ruscica, T. 2021. Python Website Full Tutorial – Flask, Authentication, Databases & More. Tech With Tim. YouTube. Video clip. 1 February 2021. Available at: https://www.youtube.com/watch?v=dam0GPOAvVI [Accessed 15 May 2023].

Salazar, K. 2016. Diary Studies: Understanding Long-Term User Behavior and Experiences. *Nielsen Norman Group*, 5 June 2019. Electronic newspaper. Available at: https://www.nngroup.com/articles/diary-studies/ [Accessed 2 December 2023].

Sidheeq, S. 2023. How to Install Proxmox VE on VirtualBox? | Step by Step. Web page. Updated 5 December 2023. Available at: https://getlabsdone.com/how-to-install-proxmox-ve-on-virtualbox-step-by-step/ [Accessed 5 December 2023].

Sloper, G & Hess, K. 2023. O'Reilly: Protecting Your Web Applications by Gary Sloper, Ken Hess. Newton: O'Reilly Media Inc. E-book. Available at: https://www.oreilly.com/library/view/protecting-your-web/9781492052791/ [Access 14 May 2023].

University of Newcastle, Australia, 2023. Research Methods: What are research methods? Web page. Updated 14 March 2023. Available at: https://libguides.newcastle.edu.au/researchmethods [Accessed 3 December 2023].

University of Jyväskylä, 2010. Quantitative Research – KOPPA Jyväskylän Yliopisto. Web page. Updated 9 March 2010. Available at: https://koppa.jyu.fi/avoimet/hum/menetelmapolkuja/en/methodmap/strategies/quantitative-research?set_language=en&cl=en [Accessed 21 February 2022].

Utosu, K. 2020. The K.I.S.S Principle in Programming. Blog. 15 July 2020. Available at: https://dev.to/kwereutosu/the-k-i-s-s-principle-in-programming-1jfg [Accessed 17 May 2023].

VentureBeat, 2022. Report: 50% of all web applications were vulnerable to attacks in 2021. *VentureBeat*, 21 February 2022. Electronic newspaper. Available at: https://venturebeat.com/security/report-50-of-all-web-applications-were-vulnerable-to-attacks-in-2021/ [Accessed 14 May 2023].

VirtualBox, n.d. About VirtualBox. Web page. Updated 5 December 2023. Available at: https://www.virtualbox.org/wiki/VirtualBox [Accessed 5 December 2023].

WordPress. 2023. Features. Web page. Updated 5 December 2023. Available at: https://wordpress.org/about/features/ [Accessed 5 December 2023].

W3Schools. 2023. MySQL Wildcards. Web page. Updated 5 December 2023. Available at: https://www.w3schools.com/mysql/mysql_wildcards.asp [Accessed 5 December 2023].