ENTERPRISE AND IT COMPUTING (ENITS)
OFFENBURG UNIVERSITY OF APPLIED SCIENCES

SOFTWARE SECURITY

WINTER SEMESTER 2022/2023

SECURITY REQUIREMENTS REPORT

LECTURER : SCHAAD, ANDREAS

PROF. DR. PIL. M.SC

AUTHOR :

ARINN DANISH BIN ABDULLAH (191050-01)

TABLE OF CONTENT

INTRODUCTION

The purpose of creating this report is to present a completed analysis of the Security Requirements Lab and highlight key findings on the Password Management System. This study covers the following topics:

- An initial set of possible (security) requirements for the PMS
- Corresponding related Use / Misuse / Abuse cases (please also mention the difference)
- Corresponding related Attack Trees
- Any other means of communicating requirements you consider appropriate

For greater understanding, there will also be diagrams and images that are supplemented by textual descriptions.

DOCUMENT CONVENTIONS

The following conventions are applied in this document:

- PMS: Password Management System
- LA: Legacy App
- SA: System Admin
- SR: Security Requirements

SECURITY REQUIREMENTS

The security requirements for PMS are shown in the table below:

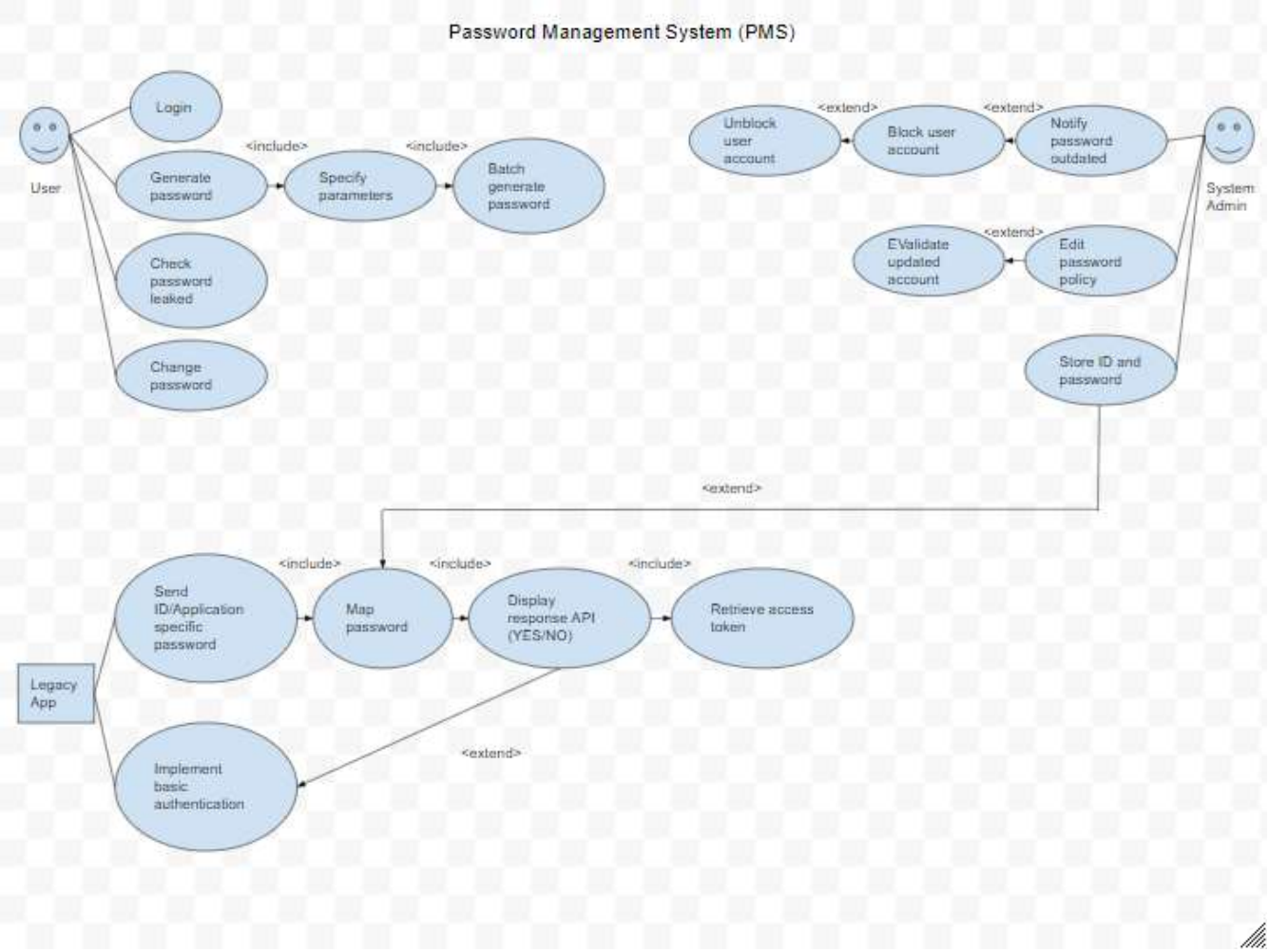
Use case ID	Type	Use Case Title	Description	Related_ Reqs	Preconditions	Postconditions / Test Outcome
SR1	Functional	Login	The system must allow the user to login using a master password and an ID		The user is not logged in a session	User logged in
SR2	Functional	Generate Password	The system must able to create a password for the user.		The saved parameters are available for use.	The generated password is stored and transmitted to user
SR2.1	Functional	Specify Parameters	The system must let the user specify certain criteria, such as the length and quantity of special characters, when generating a password (within the policy limitations)		The user set preferred restriction and requirement of the PMS-generated random password	Parameters are retained as input for the subsequent password generation.
SR2.2	Functional	Batch Generate Passwords	The system has to allow the user to batch generate password		The user specifies the amount wanted for of batch password generation	PMS generates exact amount of batch password according to users defined pre condition

SR3	Functional	Check Password Leaked	The system has to be able to determine whether a password created by a user has already been leaked.		User has a password he wants to check	System will return a Boolean saying if the password has been leaked
SR4	Non-Functional	Notify Password Outdated	The system must able to send a warning notification to the user when on of his password is outdated	SR5	The system will set a pre-defined lifecycle for password usage	The system will determine a pre-defined lifespan of password usage and notify the user; if it is exceeded, it will recommend a password change
SR5	Non-Functional	Block User Account	The SA must able to block temporarily a user account if the password is too old or if it breaches the policy	SR4, SR6	The SA monitors and records the overall amount of password usage time as well as any rules violations	For security purposes, the system prevents a user from logging in
SR6	Non-Functional	Unblock User Account	The system has to allow the SA to unblock a user account	SR5	The SA hears feedback or response from users and verifies their justifications for unblocking accounts.	The user can login as usual
SR7	Functional	Change Password	The system has to allow a user to change one of its passwords		The user presses the change password tab option	The system asks for current password for security reason in order to

						proceed with new password change
SR9	Functional	Send ID/Application Specific Password	The LA must be able to send ID/Password to the PMS		The LA will pass user information to PMS	The data will be stored in the PMS
SR9.1	Functional	Map password	The system must map password generated with a user account	SR11	The LA provides user information for PMS data storage	The user account linked with the requested random password generated
SR9.2	Functional	Display Response API (Yes/No)	The system must answer the LA's connection request with a yes or no answer.	SR12	The LA will pass user information to PMS	PMS displays answer box of Yes/No depending on the precision, accuracy and other conditions set on the data send in order for LA to access the API
SR9.3	Functional	Retrieve Access Token	The LA must retrieve an access token from the PMS		The LA sent user ID/Password to PMS	The PMS allows LA to access its API
SR10	Functional	Edit Password Policy	The system has to allow the SA to edit the password policy	SR13	The SA receive feedback and complaints from users on password policy	PMS updated the password policy and notify, display changes information to users
SR11	Functional	Store ID/Passwords	The system has to store application ID together with	SR9.1	The LA provides user information for	The information stored in the separate PMS data store to

			password and user ID in a separate data store		PMS data storage	avoid data loss and manipulation
SR12	Functional	Implement basic authentication	The system must support some basic authentication by the LA	SR9.2	The LA will pass user information to PMS	The PMS allows LA to access its API
SR13	Non-Functional	Validate updated account	The system must be able to identify and validate accounts that are updated within 24 hours	SR10	The SA edited and updated the password policy	The system notifies and display the information of updated password policy to the user

USE & ABUSE CASES



Use Case Diagram

The User, System Admin (SA), and Legacy App are the three actors identified in the use case diagram for the PMS security requirements (LA). We initially intended to include the PMS as an actor as well, but we soon recognized that the main system we are creating cannot be an actor since it can't operate independently or on its own. As a result, we developed these three, which involve two human and one external system as our actors. For a better understanding of what and how the system shall interact with one another, either internally or externally, *include* and *extend* functions are also mapped to the use cases.

Firstly, the User is assigned to five use cases, which are:

- Login
- Generate password
- Specify parameters
- Batch generate password
- Check password leaked
- Change password

Since the user can only specify the password parameters in line with their preferences after intending to generate the password through PMS, the "Specify parameters" use case is *included* in the "Generate Password" use case. After the user specifies the parameters, they will have to pick or choose how many passwords they wish to create - this is the use case for “Batch generate password” that follows.

Secondly. The System Admin (SA) is assigned to six use cases, which are:

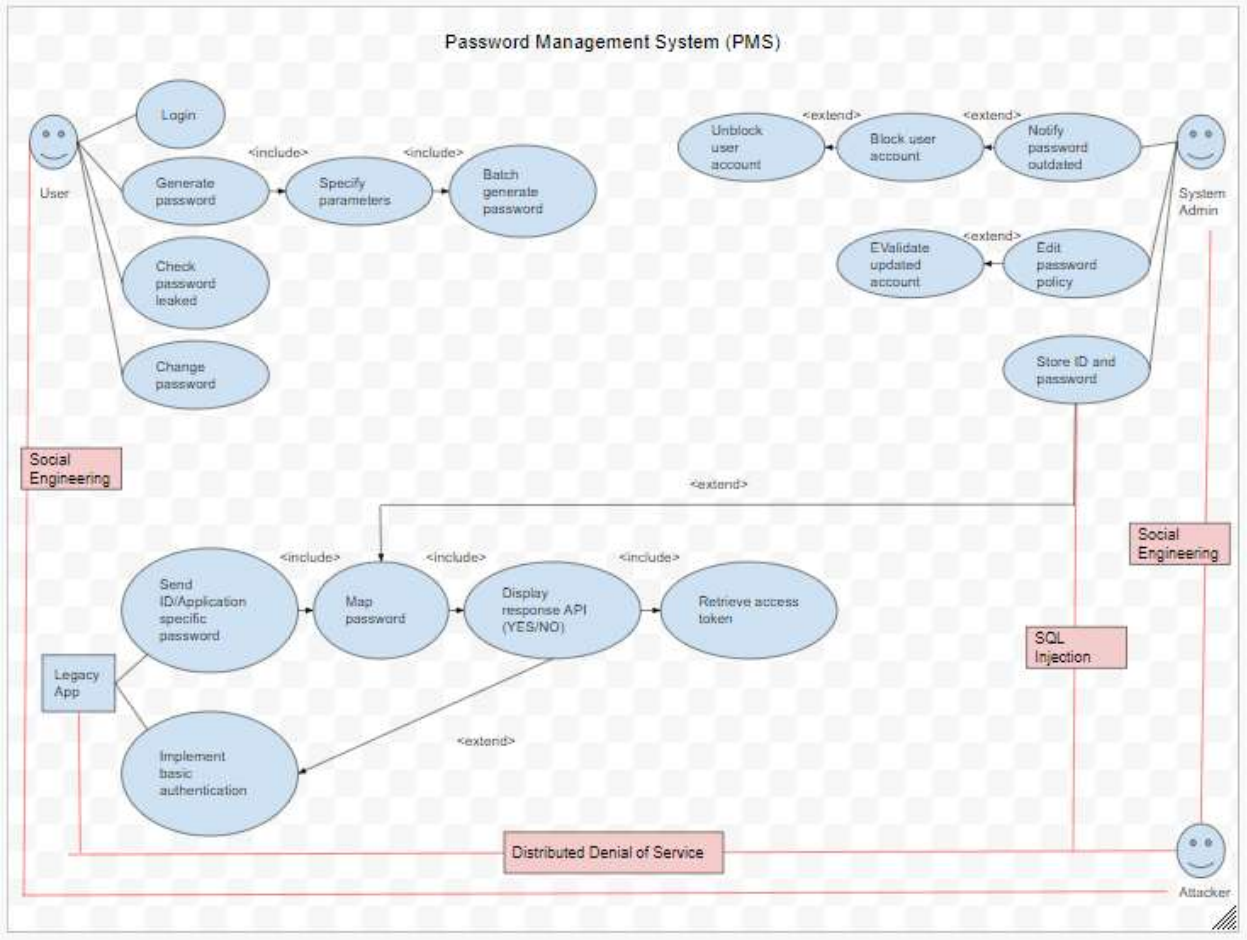
- Notify password outdated
- Edit password policy
- Store ID and password
- Block user account
- Unblock user account
- Validate updated account

Here, there are four use case non-functional requirement types. The first one is "Notify password obsolete," which requires the SA to notify the user if the password has been inactive for an excessive amount of time. Therefore, *extend* is utilized in this instance, which could result in second use case - "Block user account" if the user does not act after receiving the notice. The SA will then have to perform the third use case “Unblock user account” so that the user can log in as usual if they receive a positive answer from the user regarding their behavior. The *extend* to the “Edit password policy” that requires the SA to make sure that the password policy must be updated in only 24 hours is "Validate updated account." After that, they must notify all users of the updated policy. In addition, as SA may only keep all data in a separate store once PMS mapped all the credentials and received information from LA, the "Store ID and password" use case is then an *extend* to the "Map user account" use case.

Thirdly. The Legacy App (LA) is assigned to five use cases, which are:

- Send ID/Application specific password
- Map password
- Display response API (Yes/No)
- Retrieve access token
- Implement basic authentication

When the *include* is used, the LA must send the ID/Application specific password to PMS first, then map the password to the particular or unique user account. After receiving all credentials from LA, PMS will then have to display Response API with an answer of YES or NO; at this point, the *include* is also used. Only after receiving a YES response from the PMS, the LA will receive its access token. “Implement basic authentication” is related to the use case "Display response API (Yes/No)" because it requires PMS to support authentication from LA before it can receive data from LA and decide whether or not to grant access to the token.



Abuse Case Diagram

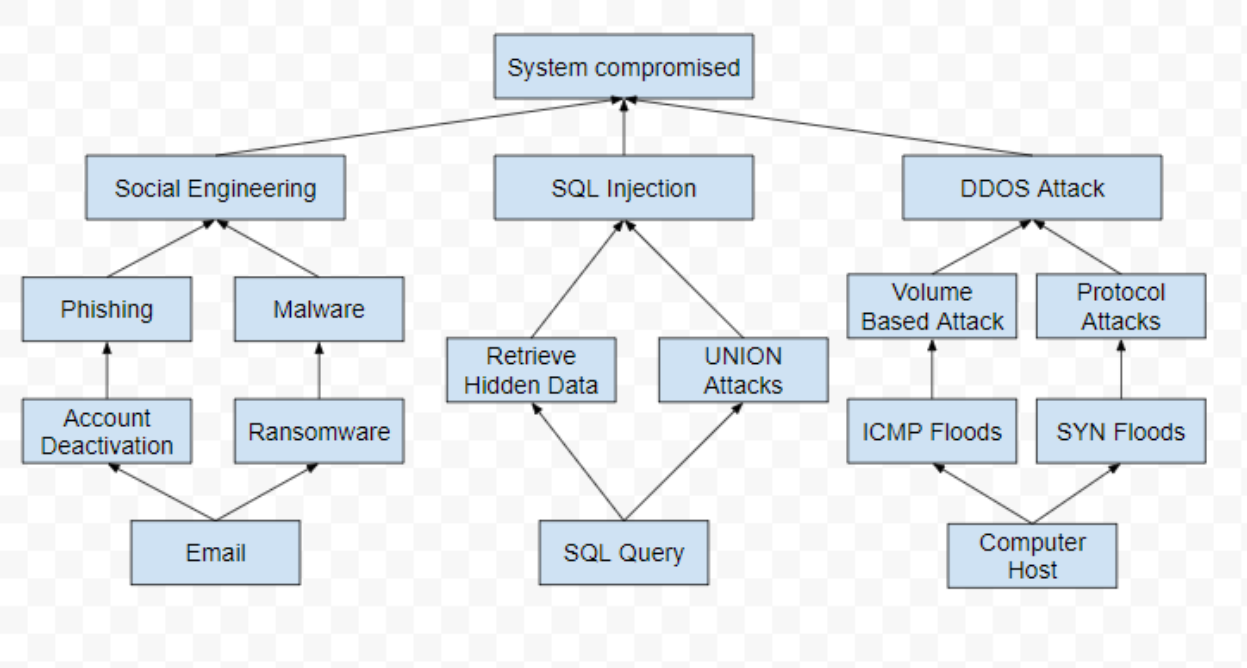
The abuse case depicted in the above diagram demonstrates three attack types that have a high likelihood of occurring because the actors may not be expecting it. These three actions - **Social engineering, SQL Injection, and Distributed-Denial-of-Service (DDOS) attack**, allow an attacker to influence the feature or the results of the use of the feature.

First off, **Social Engineering** is the practice of using artificial intelligence (AI) and people to manipulate and/or deceive others in order to get confidential information (Wikipedia). In this scenario, we recognize that the attack will only be effective against humans, which includes both the user and the SA as actors. For instance, they could come into contact with phishing, spear phishing, or fraud attacks that trick employees into granting access to their computers, routers, or Wi-Fi, which could result in PMS or even LA data theft and the installation of malware, posing a serious threat to the entire system.

Next, **SQL Injection** is a code injection technique used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution e.g. to dump the database contents to the attacker (Wikipedia). In this example, we can claim that the attack is concentrated on stealing and modifying only the database data that relates to "Store ID/Password," one of the SA use cases. For instance, the attacker can alter a SQL query to return more results using the retrieve hidden data technique. Alternatively, they can alter a query to interfere with the logic of the program by subverting it. Additionally, UNION attacks are used to acquire data from many database tables.

Last but not least, **Distributed-Denial-of Service (DDOS) Attack** is a malicious attempt to momentarily disrupt or suspend the hosting server's functions in order to render an online service unavailable to users. Because the legacy application is an external system that is referred to as non-human and machine in the scenario, it can be utilized as an example of this attack. The attacker can launch a distributed-denial-of-service (DoS) assault on the application layer by flooding the target resource with ICMP (Ping) packets, generally sending packets as quickly as possible without waiting for responses. Due to the volume of requests, the LA may momentarily cease operations, which would be difficult for both users and SA to perform their tasks.

ATTACK TREES



Attack Trees Diagram

The threat of information leak is depicted in the attack trees and is described by columns and levels by levels. There is a total of three columns that show three different forms of assaults, including Social Engineering (which targets people), SQL Injection (which targets databases), and DDOS (which targets machines or external systems). While the first stages for each demonstrate the media and platform on which an attacker could begin their hacking operations. The third and second floors display more focused techniques that the attacker can use to focus their attack and get closer to data alteration or theft, which would then signal that the system had been infiltrated.

For the first column, Social Engineering, the attacker will need to use email to launch the attack. They will have to deceive the users and system admins into thinking their accounts are being deactivated using phishing emails, which, when clicked, expose the users to vulnerabilities where sensitive personal or organizational data in the LA and PMS can be stolen. The next method involves the attacker deceiving users and system admins into installing ransomware on their device by sending them an urgent email message. If they don't pay the attacker a price to have the virus removed, they won't be able to access the LA or PMS.

For the second column, SQL Injection, the attacker will need to create SQL query to launch the attack. They will conduct two different types of assaults, the first of which is called "Retrieve Hidden Data," in which the attacker can alter a SQL query to produce extra results. The password information, for instance, can be acquired together with every batch-generated password. The second is known as "UNION Attacks," in which the attacker can access data from many database tables. For instance, by attempting to hack into the PMS database table, they can also access data at the LA.

The third column, DDOS Attack, requires the attacker to use the computer host as a launchpad for the attack. They will carry out two different kinds of attacks: ICMP Floods, which is a Volume Based Attack, and SYN Floods, which is a Protocol Attack. The ICMP flood sends packets as quickly as feasible without stopping to wait for responses, overwhelming the target resource with ICMP Echo Request (ping) packets and when the user and system admin's servers frequently attempt to respond with ICMP Echo Reply packets, it causes a large slowness in the entire system of LA and PMS, which matched the goals of Volume Based Attack. An established flaw in the TCP connection process (the "three-way handshake") is exploited by a SYN flood DDoS assault. It leads to denial of service since the host system keeps waiting for acknowledgement for each of the requests, binding resources until no new connections are possible.

USER STORIES

The user stories will be broke up into 5 sections, which are as follows:

- Story name
- User role
- Achievable actions
- Desired business value
- Acceptance criteria

1	**Story name:** User login **User role:** User **Achievable actions**: In order to log in, the user must enter their username or ID and password **Desired business value:** As a user, I want to login successfully into the LA so that I may access my information **Acceptance criteria:** User is able to complete his/her work
2	**Story name:** Generate password **User role:** User **Achievable actions**: The user must specify the parameter and the number of batch passwords to be generated. **Desired business value:** As a user, I want to generate random passwords in accordance with my own set of rules and batch sizes so that I can have stronger and more secure passwords **Acceptance criteria:** For LA login, the user will have to select from a variety of random, strong passwords
3	**Story name:** Check password leaked **User role:** User **Achievable actions**: To check if the input password has been compromised or not, the user must have come up with or his/her own password or a series of random ones. **Desired business value:** As a user, I want to determine whether or not my password has been compromised so that I can change it before my account in LA is put in risk

	Acceptance criteria: Prior to any assaults occurring, the user must be aware of the password vulnerability and change it
4	**Story name:** Change password **User role:** User **Achievable actions**: Whether the password has been compromised or was chosen voluntarily, the user may update it. **Desired business value:** As a user, I want to change my password so that I can keep using it for login into the LA without fear of it being leaked **Acceptance criteria:** The user has the option to change their password at any time, so they can use a different one at different times
5	**Story name:** Notify password outdated **User role:** SA **Achievable actions**: By tracking the timeframe of password usage, the SA must first determine whether the password is obsolete or not **Desired business value:** As a SA, I want to notify the users when their password is no longer valid so that they can reset it for security reasons. **Acceptance criteria:** Before the account was blocked, the user was informed and given the chance to change their password
6	**Story name:** Block and unblock user account **User role:** SA **Achievable actions**: The SA must first check if the user is either in violation of the policy or that their password is old and hasn't been updated in a while. **Desired business value:** As a SA, I want to block and unblock a user account o ensure the security of the LA **Acceptance criteria:** If a user is blocked, they are not permitted to log in; however, if they can give the SA a good reason, they will have to be unblocked
7	**Story name:** Edit password policy **User role:** SA **Achievable actions**: In order to amend or improve policy, the SA must be informed of customer complaints and feedback **Desired business value:** As a SA, I want to edit the password policy so that the user can use the PMS through LA and has a better experience

	Acceptance criteria: Within 24 hours of revisions, users will receive notification of the amended policy and their user accounts will also be validated.
8	**Story name:** Store ID and password **User role:** SA **Achievable actions**: After the LA mapped a user account to the PMS, the SA was required to keep the ID and password in a separate structure **Desired business value:** As a SA, I want to store information on ID and password in a separate (secure) data store to keep them safer **Acceptance criteria:** With official authority, the information data can be retrieved, changed, and destroyed, making it considerably safer.

REFERENCE

- https://www.imperva.com/learn/ddos/ddos-attacks/
- https://www.pmworld360.com/5-key-elements-of-an-agile-user-story/
- https://portswigger.net/web-security/sql-injection#retrieving-hidden-data
- https://terranovasecurity.com/top-examples-of-phishing-emails/
- https://en.wikipedia.org/wiki/Social_engineering
- https://en.wikipedia.org/wiki/SQL_injection