# IN5280 Home Exam

## 1. Asset identification, categorization, and criticality assessment

To identify the information assets in the application, we take a firm look at what features the system is supposed to support. Through such an analysis, we found 3 categories of information that the system contains:

1. Statistics
2. Personal
3. Integrations

These are the assets we have identified:

| Asset | Category | C | I | A | Total |
|---|---|---|---|---|---|
| COVID-19 statistics - general | Statistics | Insignificant | Medium | Medium | Medium |
| Vaccination statistics - general | Statistics | Insignificant | Medium | Medium | Medium |
| COVID-19 statistics - app users | Statistics | Low | Medium | Medium | Medium |
| Vaccination statistics - app users | Statistics | Low | Medium | Medium | Medium |
| Payment methods | Personal | High | High | High | High |
| Location data | Personal | High | High | High | High |
| Telephone numbers | Personal | High | High | High | High |
| Email addresses | Personal | High | High | High | High |
| Personal test data | Personal | High | High | High | High |
| Authentication details | Personal | High | High | High | High |
| Facebook account link | Integrations | Low | High | High | High |
| Google account link | Integrations | Low | High | High | High |

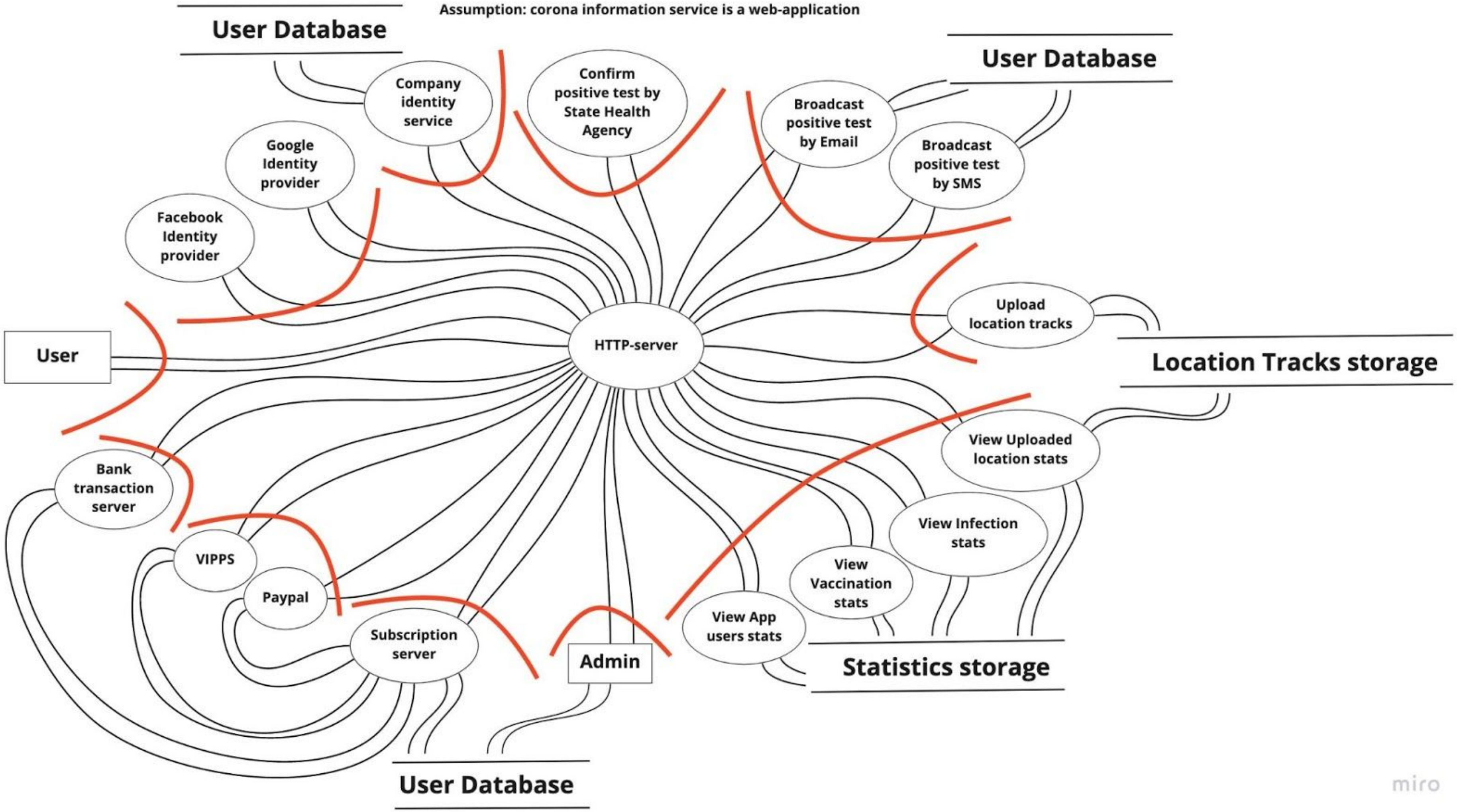
*Some of our justifications for the ratings provided above*

Statistics information is in general non-confidential information, although as the app user data is an important part of the product we sell it should be considered company internal data. As for the integrity, it is quite important that the data is correct, for the same reason. Still, we do not consider it at high criticality as the economic consequences of a slight mistake in the data is unlikely to be very large. This presupposes that the mistake is slight and gets fixed at a not-too-distant later point; if the data is always off, our service is far less useful and is unlikely to take off. With regards to availability, the time lag in testing and tracing means updated data on an hourly basis does not make that much sense; making sure availability is on a day to day basis seems appropriate for the type of system we are examining.

With regards to personal data, it is all considered of high criticality in all categories. When we store such data we must be cognizant of the sensitivity of this data across the board; thus we found no reason to classify any of these assets' criticality in a different way.

Finally, there is the category of integrations between our app and other services, like Facebook and Google. We don't see confidentiality to be very important in this area, as these tend to be semi-public anyway. However, integrity is needed as failing to link, or worse, linking the wrong accounts, has potentially serious effects for our users. Additionally, as these integrations are a big part of our login system, it is important for these to not go down. If they do, our app is useless to anyone who authenticates through those integrations.

### 2.a Dataflow diagram and attack surfaces https://miro.com/app/board/o9J_IRL0fao=/



**2.b and 2.c Threat agents and attack goals:**

| Threat agent | Goal |
|---|---|
| State-actors, who are interested in local politicians' location tracks and their personal data, may be other people's data, they will be interested in the future. Any exploitable vulnerability will be interesting. | Extract moving patterns/routes and location tracks of central politicians |
| | Create massive false-positive corona-reports to create panic |
| | Gain overview over all registered users and their personalia |
| Organized criminals pursuing profit, who are willing to damage any service, regardless of its purpose or importance for the public. | Encrypt all data in location tracks and require service owners to pay (ransomware) |
| | Hack user database to sell userdata, like name and email on blackmarket |
| | Fetch users credit/debit card details |
| | Acquire specific users location tracks, which are secret from their spouse, for blackmailing |
| Covid-19 apps competitors, other applications, who wish to have their market share and are making money on the latest correct information and paying users. | Rewire location tracks to random users, making the service worthless |
| | Scrape user location tracks data from the storage to provide similar service |
| | Change a subscription price to an extremely high value, so users would want to delete their subscriptions as soon as possible and use a similar service from a competitor |
| Anti-vaxxers, people who have limited resources to hack, but have one purpose to find alternative facts or change the data to “create” alternative facts. | Change vaccination statistics so it shows, for example, a correlation between high vaccination rates and high infection-rates statistics |
| | Broadcast fake-news to the users, like vaccines gives severe side-effects and can even cause death to a healthy person |

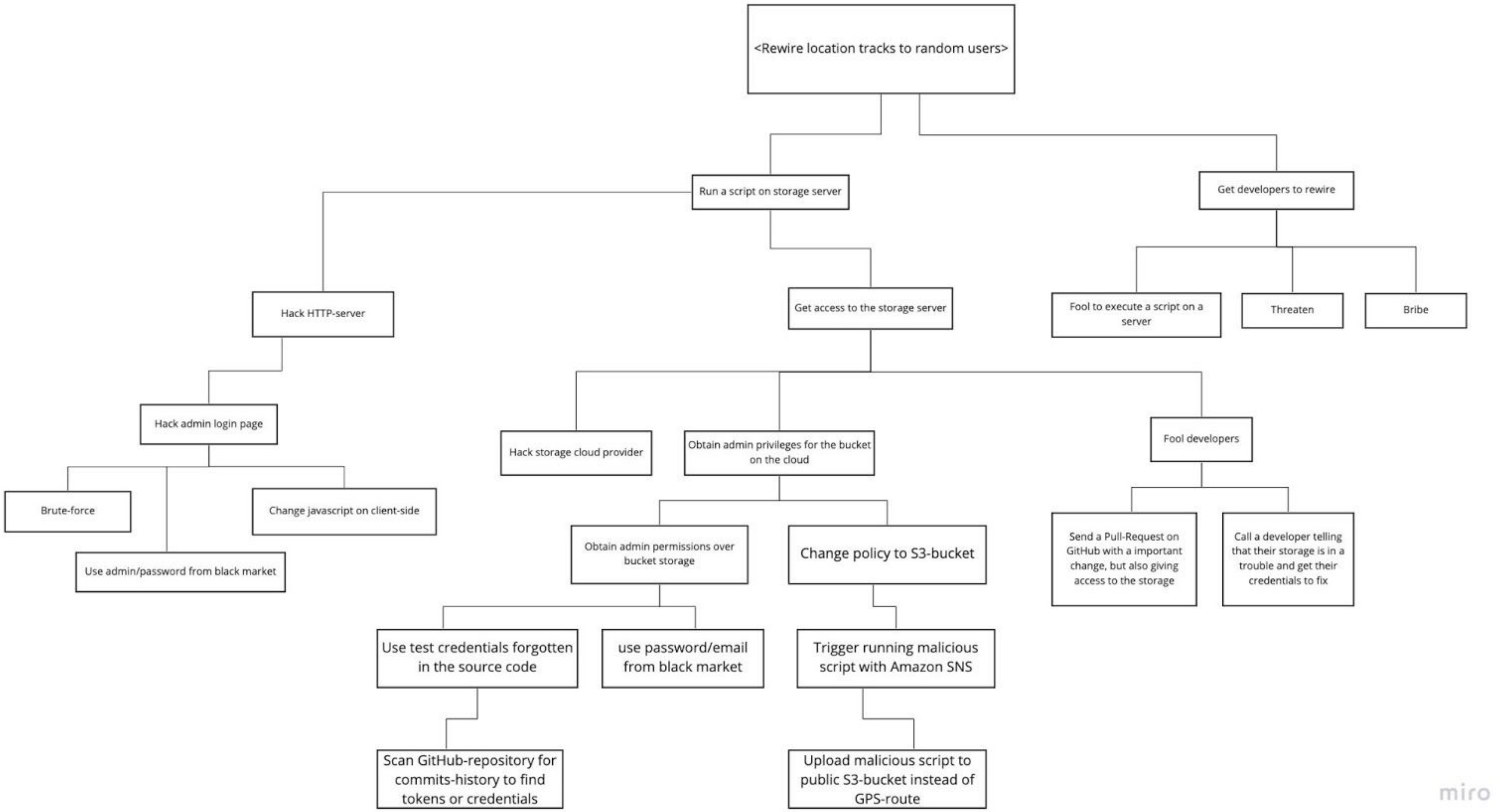
| | |
|---|---|
| (cont.) | Download users records to further use them in disinformation campaign |
| Script-kiddies, lone wolves. People who have limited resources, not so much knowledge, but a lot of time due to lockdowns. They do it for fun or because of boredom. | Broadcast spam via email or sms |
| | Make corona information service unavailable |
| | Confirm positive SARS-CoV-2 for every user in database |
| | “Facerape” users who have chosen to authenticate with Facebook |

**2.d Attack trees.**

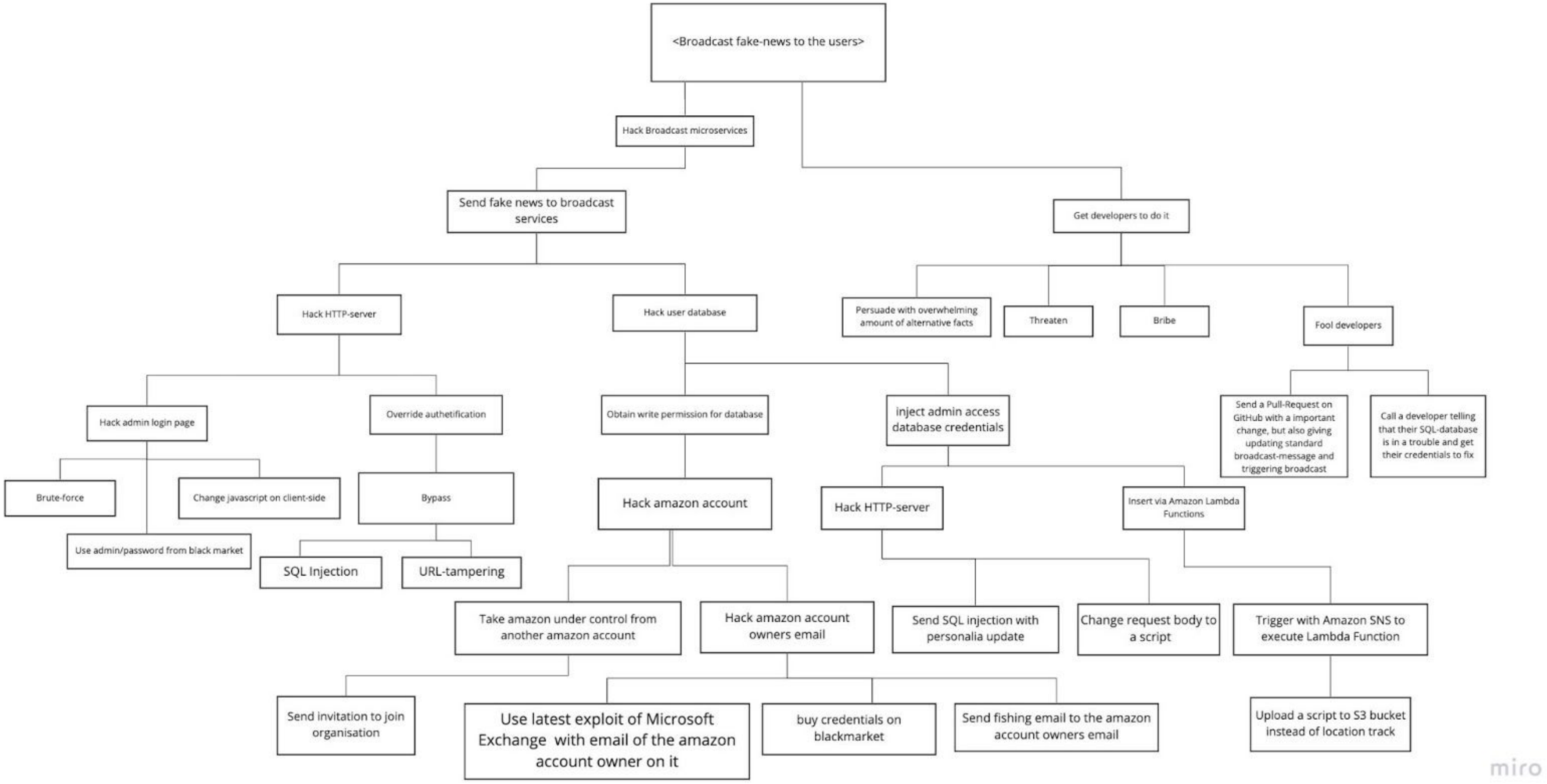
Assumptions:

1. Users GPS-location storage is in a public cloud provider like Amazon
2. The source code is in a public repository on GitHub
3. Business logic is running on the HTTP-server
4. Amazon SNS triggers on each uploaded file to Amazon S3 and sends its content to execute in Amazon Lambda Function
5. User database is a relational database running in the Amazon cloud
6. Amazon account owner has his email in Microsoft Exchange running on premises (having MS Exchange exploit announced last week)
7. Database changes triggers sending of broadcast SMS and email messages to other users about possible infection

**Covid-19 apps competitors - rewire location tracks to random users, making the service worthless**
https://miro.com/app/board/o9J_lQnfZjU=/


<Rewire location tracks to random users>
Run a script on storage server
Get developers to rewire
Hack HTTP-server
Get access to the storage server
Fool to execute a script on a server
Threaten
Bribe
Hack admin login page
Hack storage cloud provider
Obtain admin privileges for the bucket on the cloud
Fool developers
Brute-force
Change javascript on client-side
Use admin/password from black market
Obtain admin permissions over bucket storage
Change policy to S3-bucket
Send a Pull-Request on GitHub with a important change, but also giving access to the storage
Call a developer telling that their storage is in a trouble and get their credentials to fix
Use test credentials forgotten in the source code
use password/email from black market
Trigger running malicious script with Amazon SNS
Scan GitHub-repository for commits-history to find tokens or credentials
Upload malicious script to public S3-bucket instead of GPS-route
miro

**Anti-vaxxers - Broadcast fake-news to the users, like vaccines gives severe side-effects and can even cause death to a healthy person**
https://miro.com/app/board/o9J_lQLyrsQ=/



<Broadcast fake-news to the users>
Hack Broadcast microservices
Send fake news to broadcast services
Get developers to do it
Hack HTTP-server
Hack user database
Persuade with overwhelming amount of alternative facts
Threaten
Bribe
Fool developers
Hack admin login page
Override authetification
Obtain write permission for database
inject admin access database credentials
Send a Pull-Request on GitHub with a important change, but also giving updating standard broadcast-message and triggering broadcast
Call a developer telling that their SQL-database is in a trouble and get their credentials to fix
Brute-force
Change javascript on client-side
Bypass
Hack amazon account
Hack HTTP-server
Insert via Amazon Lambda Functions
Use admin/password from black market
SQL Injection
URL-tampering
Take amazon under control from another amazon account
Hack amazon account owners email
Send SQL injection with personalia update
Change request body to a script
Trigger with Amazon SNS to execute Lambda Function
Send invitation to join organisation
Use latest exploit of Microsoft Exchange with email of the amazon account owner on it
buy credentials on blackmarket
Send fishing email to the amazon account owners email
Upload a script to S3 bucket instead of location track
miro

## 3. Risk analysis, using the RMF

### a. Identify business goals for this system

| **Business goals** | |
|---|---|
| **BG1** | An easier way to view the infection statistics. |
| **BG2** | An easier way to view the vaccination statistics. |
| **BG3** | More information on infection tracking. |
| **BG4** | A viable business model through a monthly fee. |
| **BG5** | Providing authentic information. |

**b. From the business goals, identify business risks**

| **Business risks** | |
|---|---|
| **BR1** | Inaccurate statistics. |
| **BR2** | Wrong location input. |
| **BR3** | Users may receive fake additional fees to pay. |
| **BR4** | Information might not be trustworthy. |

### c. From the attack trees and Data flow diagram, identify technical risks for the system

| **Technical risks** | |
|---|---|
| **TR1** | Brute-force. |
| **TR2** | Database breach. |
| **TR3** | Cross-site scripting. |
| **TR4** | Social engineering. |
| **TR5** | Hacking with AWS. |
| **TR6** | GitHub token scanning. |
| **TR7** | SQL injection. |
| **TR8** | URL-tampering. |

**d. Link the technical risks to the business risks – what technical risks affect what business risks?**

| Technical risks | Probability | Consequences | Risks | Mitigation |
|---|---|---|---|---|
| ***BR1: Inaccurate statistics.*** | | | | |
| **TR1: social engineering.** | | | | |
| TR1.1: Users providing inaccurate infection information | H | M | H | Require users to document their inputs. |
| **TR2: cross-site scripting.** | | | | |
| TR2.1: change JavaScript. | M | H | H | Encode data on input. |
| **TR3: URL-tampering** | | | | |
| TR3.1: web parameter tampering software | M | H | H | Enable session state protection. |

| Technical risks | Probability | Consequences | Risks | Mitigation |
|---|---|---|---|---|
| ***BR2: wrong location input.*** | | | | |
| **TR1: brute-force.** | | | | |
| TR1.1: hack user authentication. | L | M | M | Limit failed login attempts. |
| **TR2: database breach.** | | | | |
| TR2.1: hack stolen user database. | M | H | H | Use encrypted data. |

| Technical risks | Probability | Consequences | Risks | Mitigation |
|---|---|---|---|---|
| ***BR3: users may receive fake additional fees to pay.*** | | | | |
| **TR1: social engineering.** | | | | |
| TR1.1: Facebook/ Google phishing. | L | L | L | Make users more aware of such incidents. |
| ***BR4: information might not be trustworthy.*** | | | | |
| **TR1: hacking with AWS.** | | | | |
| TR1.1: storage cloud hack. | L | H | M | Enable multi-factor authentication. |
| **TR2: SQL injection.** | | | | |
| TR2.1: manipulation of information. | L | H | M | Input validation. |

**4. Security requirements**

- The system should verify location tracks before using them.
- The system should validate user data input.
- The source code should not be public.
- Administering pages should have a limited availability to certain geographical areas.
- The system should perform regular data backups and have data recovery procedures.
- It should be possible to pause SMS and email broadcasting.
- All users with administrative access should utilize multi-factor authentication.
- The system should anonymize personal data about the users.