Individual Project

Mehmet Ustek - 2363741

# Part 1

## Task 1

1)

The main problem with EnA’s system architecture is that they use a single database to operate each workload. They distribute this database file to third party organisations without filtering or anonymizing. EnA does not tokenize or uniquely identify their data to determine their data flow. The reason access request to fulfilment ratio is low is because EnA does not know how their data is processed through all the shared third-party organisations. They give all their data including the highly sensitive and sensitive ones to other organisations that they have no idea how these data are being used. That is why they cannot completely give information about their data usage and distribution policy to their users, hence not fulfilling the request.

2)

EnA should refactor their system architecture by separating their data to categories and marking all by classification. For example, retrofit contractors do not need to know users’ healthcare data. All they need to know is essential business and house data, e.g. total floor area.

Applying classification to EnA’s data, we will mark each data and store them at their classification level. Highly sensitive, sensitive, quasi sensitive, financial, healthcare, house or demographic data will be stored at different tables or databases. We will sign all these databases to a certain authorization level, which will be only reachable by people who have the ideal access level. EnA must also be highly selective when distributing their data to third party. A very strictly applied and well-defined policy should indicate which data is distributed to third party suppliers.

3)

A)

The obvious problem with HAM is that energy assessment providers (EAPs) are allowed to directly reach customers, and they can use the “homeowners” data provided by EnA in any way they want. Since EnA does not mark their data as their own, there is no way that EnA can know what their data is being used for. They simply publish every data they have to those that ask for it and they furthermore do not track the data flow after it is sent to third party. This system is not trustworthy for the users since it does not offer any sort of user data privacy and protection. All user data whether it is sensitive or not distributed to all parties, and hence adversaries who can understand the value of this data can easily use the data to trick people using their demographic data and behaving like a trusted supplier.

B)

## Task 2

1)

|  |  |  |
| --- | --- | --- |
| Field | Tag (business|personal:lsi|qsi|si|hsi - financial | healthcare | house | demographic) | Comments |
| employer | personal:qsi-financial-employer |  |
| payslip | personal:hsi-financial-payslip |  |
| employmentstatus | personal:lsi-financial-employment-status |  |
| nationalinsurancenumber | personal:hsi-healthcare-nationalinsurance |  |
| currentaddress | personal:hsi-house-address-currentaddress |  |
| previousaddress | personal:hsi-house-address-previousaddress | An adversary may distinguish a person by looking at the previous locations of an individual. |
| ageofproperty | business:lsi-house-ageofproperty |  |
| energyrating | business:lsi-house-energyrating |  |
| counciltaxband | personal:qsi-financial-counciltaxband |  |
| student | personal:qsi--demographic-student |  |
| healthstatus | personal:hsi-healthcare-healthstatus |  |
| existingloan | personal:hsi-financial-existingloan |  |
| tenant | personal:qsi-demographic-tenant |  |
| drivinglicence | personal:hsi-drivinglicense |  |
| livingwithpartner | personal:qsi-demographic-livingwithpartner |  |
| fostercarer | personal:si-demographic-fostercarer |  |
| spousalmaintenanceincome | personal:hsi-financial-spousalmaintenanceincome |  |
| city | business:qsi-house-address-city | Business requires access to the city information of the household to operate. |
| ethnicity | personal:qsi-demographic-ethnicity |  |
| propertytype | business:qsi-house-propertytype | This field can give away the financial status of someone when combined with other qsi fields. |
| norooms | business:qsi-house-norooms | This field can give away the financial status of someone when combined with other qsi fields. |
| totalfloorarea | business:qsi-house-totalfloorarea |  |
| carer | personal:si-healthcare-carer | Values are just yes or no, but it would still give information about one's demographic |
| socialtenant | personal:si-demographic-socialtenant |  |
| pensioner | personal:si-demographic-pensioner |  |
| nochildren | personal:qsi-demographic-nochildren |  |
| totalamountinbank | personal:hsi-financial-totalamountinbank |  |
| fullname | personal:si-demographic-fullname |  |
| dateofbirth | personal:qsi-demographic-dateofbirth |  |
| bankaccountno | personal:hsi-financial-bankaccountno |  |
| creditcardno | personal:hsi-financial-creditcardno |  |
| accountbalance | personal:hsi-financial-accountbalance |  |
| disabilityallowance | personal:hsi-financial-disabilityallowance |  |
| citizenship | personal:qsi-demographic-citizenship |  |
| email | personal:si-email |  |
| gender | personal:qsi-demographic-gender |  |
| yearlyenergycost | business:qsi-financial-yearlyenergycost |  |
| passportno | personal:hsi-passportno |  |
| phoneno | personal:hsi-phoneno |  |
| married | personal:qsi-demographic-married |  |

## 3)

{

"quasi\_percentage": "0.43",

"highly\_sensitive\_percentage": "0.40",

"low\_sensitive\_percentage": "0.07",

"sensitive\_percentage": "0.09",

"healthcare\_percentage": "0.06",

"house\_percentage": "0.21",

"demographic\_percentage": "0.30",

"financial\_percentage": "0.33",

"hsi\_financial\_percentage": "0.22",

"si\_financial\_percentage": "0.00",

"qsi\_financial\_percentage": "0.08",

"lsi\_financial\_percentage": "0.04",

"hsi\_financial\_to\_financial": "0.66",

"si\_financial\_to\_financial": "0.00",

"qsi\_financial\_to\_financial": "0.23",

"lsi\_financial\_to\_financial": "0.11"

}

### a)

Highly sensitive data proportion is 40%.

### b)

Sensitive data proportion is 9%.

### c)

The following data indicates the proportion of sensitivity of financial data across all data.

"hsi\_financial\_percentage": "0.22",

"si\_financial\_percentage": "0.00",

"qsi\_financial\_percentage": "0.08",

"lsi\_financial\_percentage": "0.04",

The following data is normalized version to indicate proportion of sensitivity financial data with respect to the financial data only.

"hsi\_financial\_to\_financial": "0.66",

"si\_financial\_to\_financial": "0.00",

"qsi\_financial\_to\_financial": "0.23",

"lsi\_financial\_to\_financial": "0.11"

### d)

Data Retention Policies:

Delete highly sensitive financial data after 1 week or after use, whichever is earlier. A large proportion of financial data is highly sensitive, reduce access to this database to only a few people and try to do automations for processing this data.

Delete rest of the highly sensitive and financially sensitive data after 2 weeks or after use, whichever is earlier.

Delete sensitive data after 1 month or after use, whichever is earlier. This database can be accessed by more people than highly sensitive, however still authorize people that really need operational access.

Limit the access of highly sensitive and sensitive data to be only used for automated processes and operational users. Do not give read/write access to those who are not from operational departments to reach these sensitive data.

Delete quasi-sensitive data after 6 months, or after use, whichever is earlier. Since this data can also be used with another data to determine a specific person, it is best practice to limit access to these databases too. Still, quasi sensitive financial data should be kept at another place and should be deleted before 2 months. On the other hand, read access for other quasi sensitive data can be given to analysts.

Delete low sensitive financial data after 6 months or after use, whichever is earlier. Delete other low sensitive data after 1 year or after use, whichever is earlier. It is legitimate to give read access to analysts for the low sensitive databases except for financial.

## Task 3

Data protection risk:

Now that we have multiple datastores, we need to assign access levels to these files. In addition to that, we need to establish an authorization policy for everyone trying to grant access to any specific file and datastore. If we do not distinguish between an authorized user and a regular low-access user, the same vulnerabilities that is discussed at Task 1 and Task 2 would still be valid. After assigning each datastore and user an access level, we need to make sure that correct read/write privileges are given to the users. Not everyone should have access to write up even though their access level permits writing up.

## Task 4

1)

|  |  |  |
| --- | --- | --- |
|  | Readable | Writable |
| Analyst | Homeowner-h | homeowner-f-f |

2)

Based on Bell-Lapadula model, analyst can only read house data. The house data is classified low sensitivity, meaning that the data at stake is at low sensitive profile. Because low sensitivity data do not constitute a high risk, we can assign low risk to this analyst.

(Sensitive + highly sensitive data that the analyst can read) / Total data the analyst can read = 0

Very low risk.

3)

Since the analyst will only analyse data and will not alter it, only the read access is sufficient. Homeowner-f-f database contains highly sensitive financial data along with full names of users, which makes a highly sensitive financial information when combined. Granting write access to such critical datastore may constitute a bigger problem if the analyst alters the data either deliberately or not. Therefore, combining these problems with principle of least privilege, we should not grant write access for homeowner-f-f file to the analyst.

# Part 2

## 1)

Out of 107 dependencies, 51 are sourced from unknown countries, constituting nearly half of all dependencies. Among the remaining 56, 32 originate directly from the United States. Given that Facebook's headquarters is in the United States, the localization of data in their dependencies to the U.S. is crucial for their operations. However, Facebook is a global platform, operating in numerous countries with distinct regulatory requirements. For instance, India mandates a unique approach for handling its citizens' data, making it imperative to have dependencies that facilitate integration with Indian regulations. As a result, the dependence on software components accommodating diverse regulations may become obligatory. All European countries and the UK are responsible for 9 other dependencies, where Facebook delivers services at significant scales.

The problem contributing to the high risk is the presence of unknown sources of dependencies. If individuals responsible for maintaining these dependencies introduce malware or backdoors into the distributed versions on which Facebook relies, the threat would significantly escalate, raising concerns about the accountability and reliability of Facebook. Given that Facebook manages extensive datasets related to the privacy and security of its users, such risks could have serious implications.

## 2)

Even though dependencies such as isaacs, browserify, Jshint, mochajs, felixge, mozilla, webdriverio are all significantly old, they are maintained regularly. Whereas jugglingmike is 6 years old and last update to its source code was again 6 years ago. Number of contributors to this project is also critically low, with only one person responsible for the maintenance. All the repositories that have significantly lower number of contributors constitute a high risk as the contributors simply un-publish the repository. Repositories such as cfware, andrewrk, nmccready, ljharb, johanbrook, ljharb, codemirror, markis, kemitchell with highly low number of contributors indicates a threat and must be avoided. All the repositories that are updated more than half of a year ago with critical number of issues must also be avoided. Positively, facebook-nodejs-business-sdk does not have a dependency with last update more than half of a year and critical number of issues.

Number of issues alone also is a significant indication of high risk. Repositories electron, babel, vuejs, rollup, jshint, browserify, PrismJS, sindresorhus, and webpack have more than 300 issues each. On a positive note, all these repositories are well-maintained with high numbers of contributors.

Size of a dependency is also an indicator of vulnerabilities since it is easier to detect a vulnerability on a lower size source code. Jquery, babel, eslint, webpack, webdriverio, mozilla, and electron are all well-known and highly distributed repositories with a size constituting a high risk.

Repositories with a significantly low number of commits and interests may also exhibit a risk, such as jugglinmike, cfware,markis, ljharb, nmccready, johanbrook and humanwhocodes, since these values indicate a direct risk to the project alone.

Overall, all these dependencies described above constitute significantly high risks towards the project. However, if I were to sort them and give them a high-risk analysis based on all the attributes in table 1, the following would be my approach to dependencies with the most higher integration risk:

1. Non-maintained, low interest repositories.: Repositories such as Jugglinmike, cfware, markis, ljharb, gulp-sourcemaps, visionmedia, and nmccready that are critically high age, low interest, and low number of commits.
2. Electron: Large size, high age, highest number of issues, high number of total dependencies.
3. Babel: Large size, high age, high number of issues.
4. jshint: Large size, high age, high number of issues, highest number of total dependencies.
5. Other repositories with high number of total dependencies that are high in size or age: Repositories such as juliangruber, facebook, lydell.
6. Vuejs: Middle-size, high age, high number of issues.
7. Webpack: Large size, high age, high number of issues.

## 3)