Task1

1. Assets:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Name | | Description | Trust level |
| A1 | | Back end database | Assets that relate the website database. | T3 |
| B1 | | User | Assets that relate to a website user |  |
| B1.1 | | User’s login data | User’s credentials such as username and password. This needs protection since if the data is stolen the attacker would be able to do anything the user can. | T2 T3 |
| B1.2 | | User’s personal data | User’s personal data including contact information. This needs protection because contact information such as email addresses or mobile numbers can be accessed. | T2  T3 |
| C1 | | Admin | Assets that relate to a database administrator. |  |
| C1.1 | | Admin’s login data | Admin’s credentials. This needs protection because if stolen, the attacker could do a privilege escalation which will result in the attacker having all admin rights. | T3 |
| C1.2 | | Admin’s personal data | Admin’s personal data including contact information. This needs protection because contact information such as the admin’s email addresses or mobile numbers can be accessed. | T3 |
| D1 | | Audio file | Assets that relate to an audio file uploaded on the website. |  |
| D1.1 | | File upload | Only authorized users should be able upload files. This needs protection so that file upload function is only available for registered users. | T2  T3 |
| D1.2 | | File size | File size should be restricted to an average audio track size. This need protection so that large files aren’t uploaded. (which could be malicious files) | T2  T3 |
| D1.3 | | File data type | File data type should be checked. This needs protection because a file can be uploaded as an audio extension but in reality, is a malicious script. | T2  T3 |
| D1.4 | | Track details | The page where all uploaded tracks are shown. | T2  T3 |

1. Trust levels:

|  |  |  |
| --- | --- | --- |
| Id | Name | Description |
| T1 | Guest | A user who has not yet authenticated to the website |
| T2 | Authenticated user | A registered user who has valid credentials |
| T3 | Database server administrator | A user who can do any operation on the database. |

1. Entry Points:

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Name | Description | Trust level |
| E1 | Web server listening port | Port on which the web server listens. All web pages are layered on this port. | T1  T2 |
| E2 | Login page | Page where guests are able to login. | T1 |
| E2.1 | Login method | Credentials entered by the guest are compared with those in the database. If credentials are available in database a session will be created. | T1 |
| E3 | Registration page | Page where guests are able to register | T1 |
| E3.1 | Registration method | Data such as username, password and email are requested and added to the database so that the guest could login as an authenticated user. | T1 |
| E4 | New Track page | Allows the user to add a new audio file. | T2  T3 |
| E4.1 | Add track method | Accepts inputs such as track title, genre and a button which allows the user to upload an audio file | T2  T3 |

d/e. Threats & Mitigation:

|  |  |
| --- | --- |
| Id | TR1 |
| Name | An attacker is in possession of a valid username and password |
| Description | The attacker now has the same privileges as the user logged on. |
| Stride | Elevation of privilege, Tampering, Information disclosure |
| Entry Points | (E2) Login page |
| Assets | (B1.1) User’s login data, (B1.2) User’s personal data |
| Mitigation Strategy | External access protection implemented on database,  hashing the password in the database,  Strong passwords should be enforced |

|  |  |
| --- | --- |
| Id | TR2 |
| Name | Attacker tries to supply malicious data when logging in |
| Description | The attacker is able to login without having applicable username and password, or trying to input special characters to be able to infiltrate in the system as another user. |
| Stride | Elevation of privilege, Tampering |
| Entry Points | (E2) Login page |
| Assets | (B1.1) User’s login data, (A1) Back end database. |
| Mitigation Strategy | Implementing security on database such as using parameterized queries and stored procedures,  Blocking user after 3 attempts |

|  |  |
| --- | --- |
| Id | TR3 |
| Name | The attacker alters the URL to Access Track details |
| Description | The attacker tempers the URL to have access to unauthorised track information which eventually can lead to user information. |
| Stride | Tempering, Information disclosure |
| Entry Points | (E2) Login page |
| Assets | (D1.4) Track details |
| Mitigation Strategy | Query string encryption |

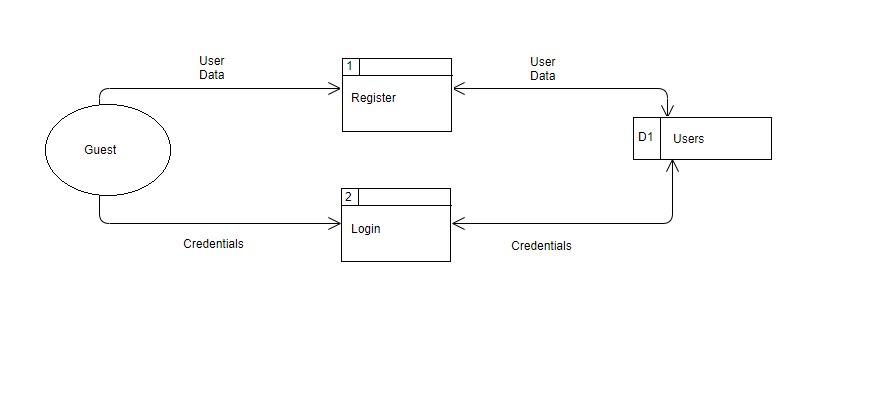
|  |  |
| --- | --- |
| Id | TR4 |
| Name | Attacker tries to upload a malicious script instead of an audio file |
| Description | Attacker uploads a malicious script with the file extension altered as the one requested. |
| Stride | Tempering |
| Entry Points | (E4) New Track page, (E4.1) Add track method |
| Assets | (D1)Audio file, (D1.1) File upload, (D1.3)File data type |
| Mitigation Strategy | Check that file signature of the uploaded file matches the one requested |

|  |  |
| --- | --- |
| Id | TR5 |
| Name | Attacker tries to upload a file which is larger than requested |
| Description | Attacker uploads a file which has a larger file size than requested. |
| Stride | Tempering |
| Entry Points | (E4) New Track page, (E4.1) Add track method |
| Assets | (D1)Audio file, (D1.1) File upload, (D1.2) File Size |
| Mitigation Strategy | Check that file size of the uploaded file isn’t larger than requested. |

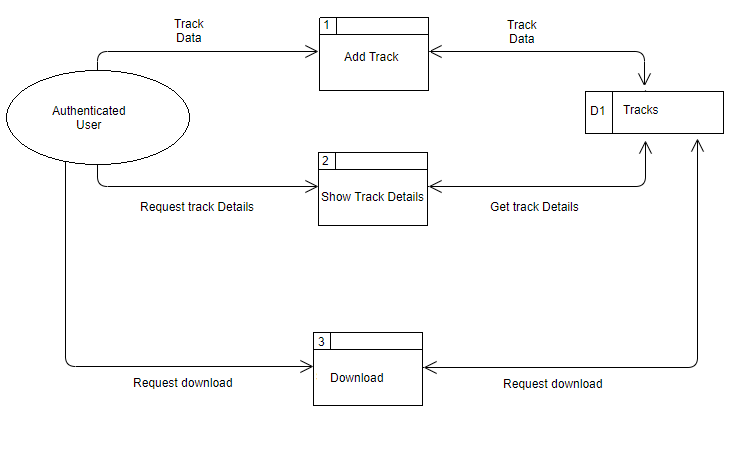
|  |  |
| --- | --- |
| Id | TR6 |
| Name | Attackers use scripts to create fake accounts on the website’s database |
| Description | The attacker tries to use a malicious script to create multiple accounts on the website’s database. |
| Stride | Spoofing |
| Entry Points | (E3.1) Registration method |
| Assets | (A1) Backend database |
| Mitigation Strategy | Implementing a CAPTCHA which will prevent scripts from creating multiple “Fake accounts” |

Task2 (DFD)

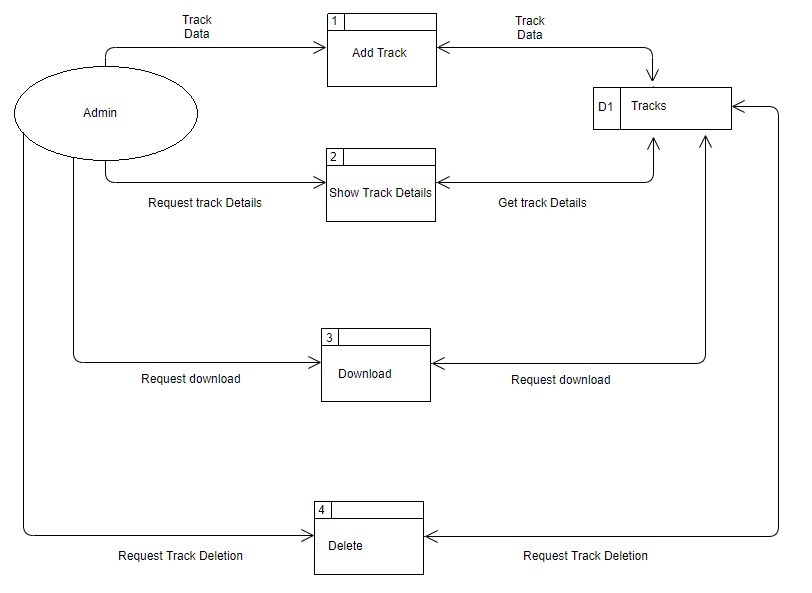
Guest



Authenticated User



Admin



Task 8 (Review)

Test unit #1

A test method with the name of TestingLoginMethod() was created to test the website's authentication system. The mentioned testing method was being tested for the Broken Authentication (A2) from the OWASP (Open Web Application Security Project) top 10 list.

The aim of this testing method is to make sure that the website’s authentication is working correctly without any flaws and to see how it will handle different logging scenarios. It is important to stress test your system to Its limits so that you know it’s breaking point and if so, improve it / fix it.

Despite all my efforts, the system seemed to be unbreakable or at least couldn’t manage to break it.

Assuming that the password chosen by the user wasn’t strong enough a security counter measure was implemented as a precaution for those types of attacks. If the guest which is trying to login happen to enter an incorrect password for the third time consecutively, the system will automatically block the account until it is once again unblocked by a database admin.

Test unit #2

The second unit testing method Test\_Register\_Method\_Validation() was created to test the website’s validation in the registration section. From the OWASP top 10 list I am trying to test for Injections (A1).

The objective of this testing method is that of testing the environment in which the user will be registering an account in. It is important to see how the validation will react to different inputs. Validation such as filling all the fields and performing checks on the data types of the input which are than validated with those found in the model. Last but not least, there is also the CAPTCHA method which was implemented.

Even though a range of different inputs where entered, the system seemed to be working fine and the system wasn’t breached.

One of the greatest security measures is the CAPTCHA method which prevents attackers from using scripts to register fake users. The CAPTCHA forces the user to tick on the box before registering or else ticking several boxes to a related question to validate that the user is not a robot. Other validation such as checking if the username and/or email already exists and if any field is left empty.