# Secure File Repository – Testing Results

Table of Contents

[Secure File Repository – Testing Results 1](#_Toc97485267)

[Authn-01: 2](#_Toc97485268)

[Test the login functionality of secure file repository with a valid password & username 2](#_Toc97485269)

[Authn-02: 3](#_Toc97485270)

[Test the login functionality of secure file repository with an invalid password & valid username 3](#_Toc97485271)

[Authn-03: 4](#_Toc97485272)

[Test the login functionality of secure file repository with an invalid username & any password 4](#_Toc97485273)

[Authn-04: 5](#_Toc97485274)

[Test the session management functionality when an authorized user logs out of a session and attempts to make use of the application after the expiration of the session 5](#_Toc97485275)

[Authn-05: 5](#_Toc97485276)

[Test that a file cannot be deleted or have its sharing permissions altered unless the user owns the file 5](#_Toc97485277)

[Authz-01: 6](#_Toc97485278)

[Test the group file sharing functionality of the secure file repository 6](#_Toc97485279)

[Authz-02: 7](#_Toc97485280)

[Verify file uploading functionality of the secure file repository 7](#_Toc97485281)

[Authz-03: 8](#_Toc97485282)

[Verify file downloading functionality of the secure file repository 8](#_Toc97485283)

[Oper-01: 9](#_Toc97485284)

[The SSH command line utility must create a valid application user 9](#_Toc97485285)

[Oper-02: 10](#_Toc97485286)

[Near real-time intrusion monitoring processes logged events for all active sessions on an MVC node and generates alerts on specified conditions 10](#_Toc97485287)

[Oper-03: 12](#_Toc97485288)

[Multiple MVC nodes must be able to use the same database or work from a replica database 12](#_Toc97485289)

[Oper-05 14](#_Toc97485290)

[Restored instance for the backup primary database on a secondary environment. 14](#_Toc97485291)

[Sec-01: 14](#_Toc97485292)

[Customized application logging must record all defined events to local file and forward specified security events when conditions are met 14](#_Toc97485293)

[Sec-02: 16](#_Toc97485294)

[Failed login events can generate a log message indicating the failure reason was the use of an invalid username 16](#_Toc97485295)

[Sec-03: 16](#_Toc97485296)

[Authentication bypass attempts do not allow access to the application 16](#_Toc97485297)

[Sec-04: 17](#_Toc97485298)

[Authentication bypass attempts will be logged locally and forward a real-time alert 17](#_Toc97485299)

[Sec-05: 18](#_Toc97485300)

[File ownership automatically assigned to the authenticated user uploading the file upon creation 18](#_Toc97485301)

[Sec-06: 18](#_Toc97485302)

[The initial login page will enforce input validation of the username field 18](#_Toc97485303)

[Sec-07: 20](#_Toc97485304)

[The file extension and file size are validated prior to acceptance for database storage 20](#_Toc97485305)

[Sec-08: 24](#_Toc97485306)

[Data sanitization measures applied to the file name and keyword input fields prior to acceptance for database storage 24](#_Toc97485307)

[Sec-09: 26](#_Toc97485308)

[Database transaction logging must record changes to the host database performed by an administrator 26](#_Toc97485309)

[Sec-10: 27](#_Toc97485310)

[Database transaction logging must record changes to the application database performed by an administrator 27](#_Toc97485311)

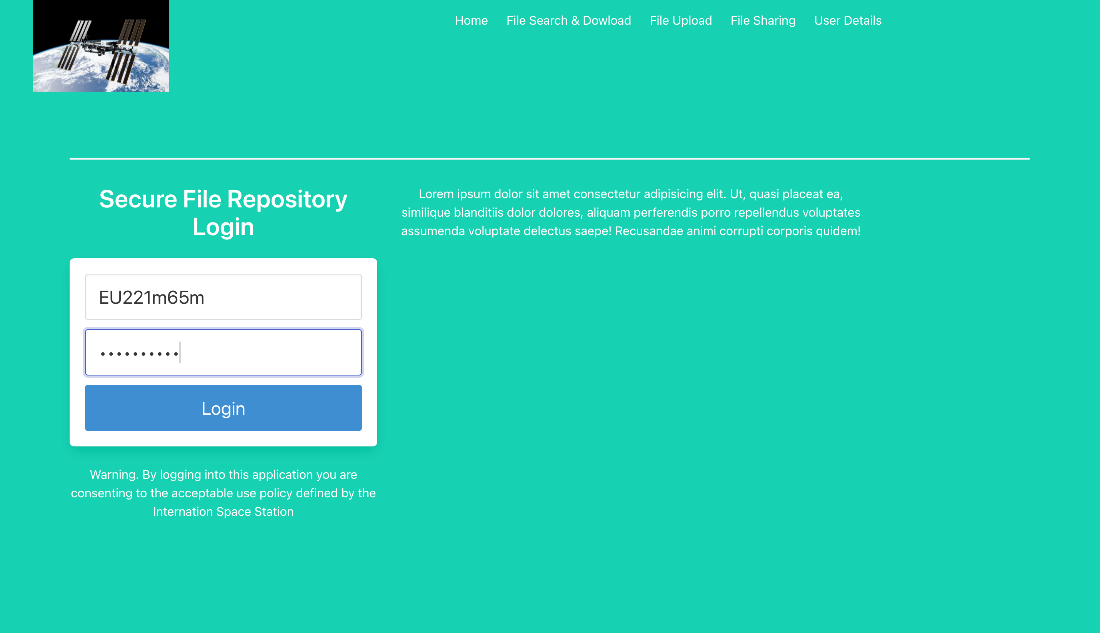
[Sec-11: 29](#_Toc97485312)

[Credentials stored in application or DBMS databases must not be in clear text and should be salted and hashed where technically feasible 29](#_Toc97485313)

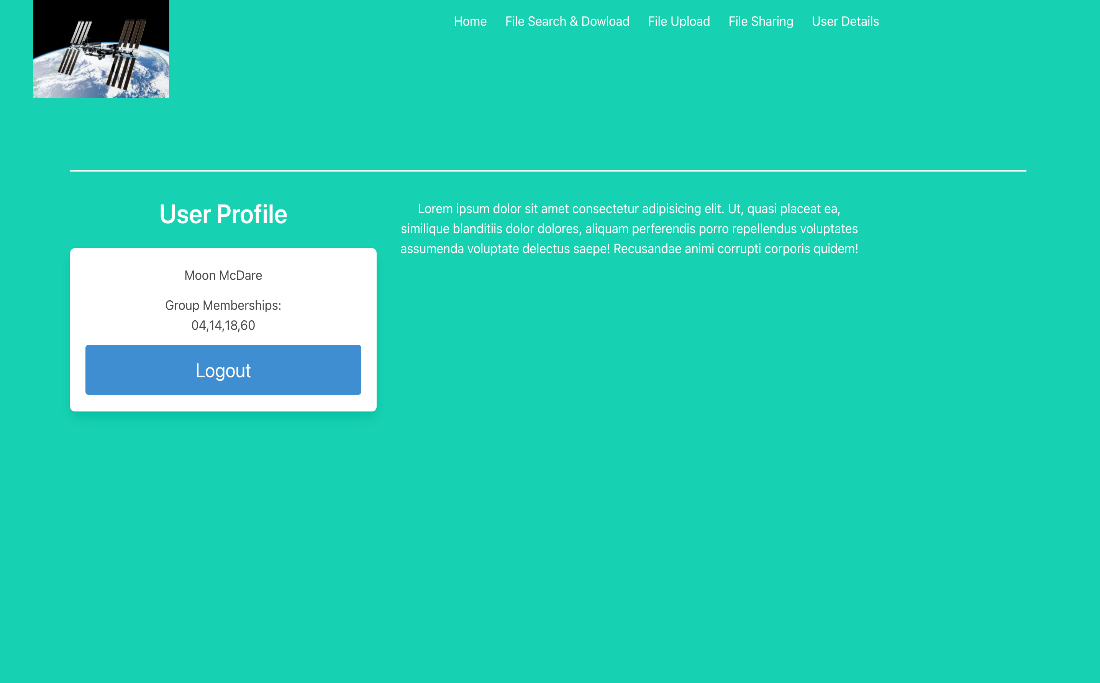
## Authn-01:

### Test the login functionality of secure file repository with a valid password & username

The screenshot below shows the input of a valid password & username serving as credentials for the authorized user ‘Moon McDare’ on the login page of the secure file repository,



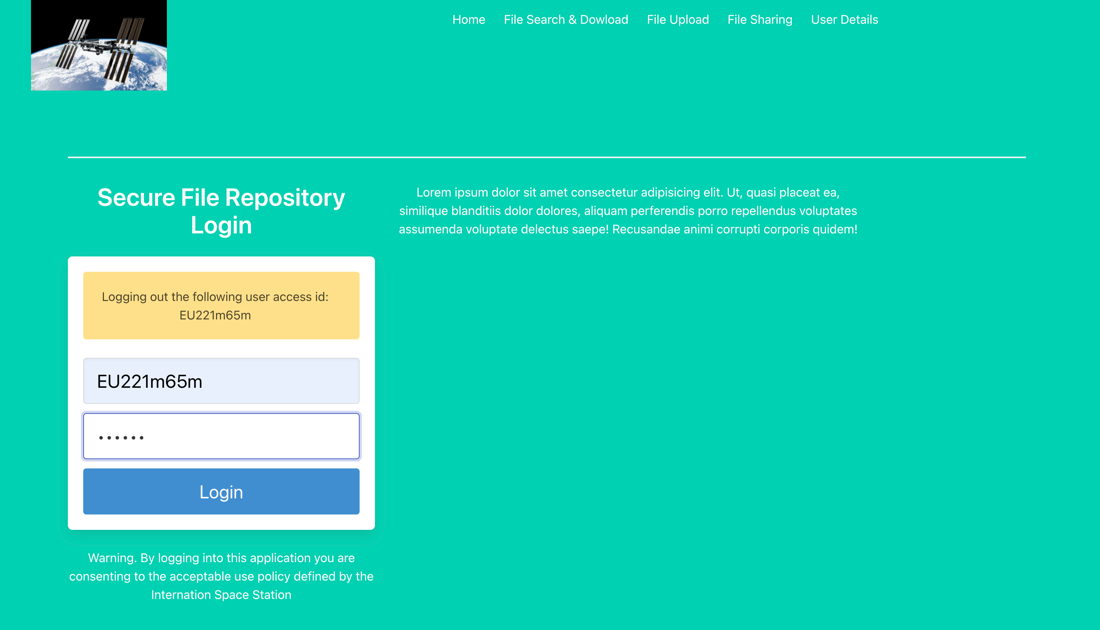
The screenshot below shows the successful login of the authorized user ‘Moon McDare’ following the input of valid credentials.



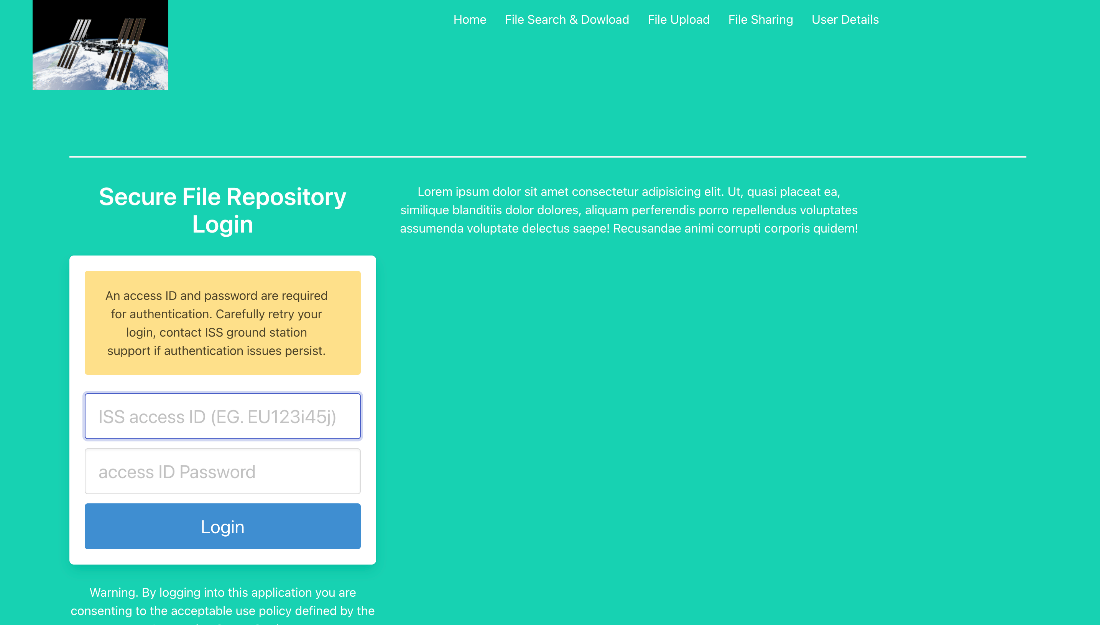
## Authn-02:

### Test the login functionality of secure file repository with an invalid password & valid username

The screenshot below shows the user with access ID EU221m65m attempting to sign in with an invalid password on the login page of the secure file repository.



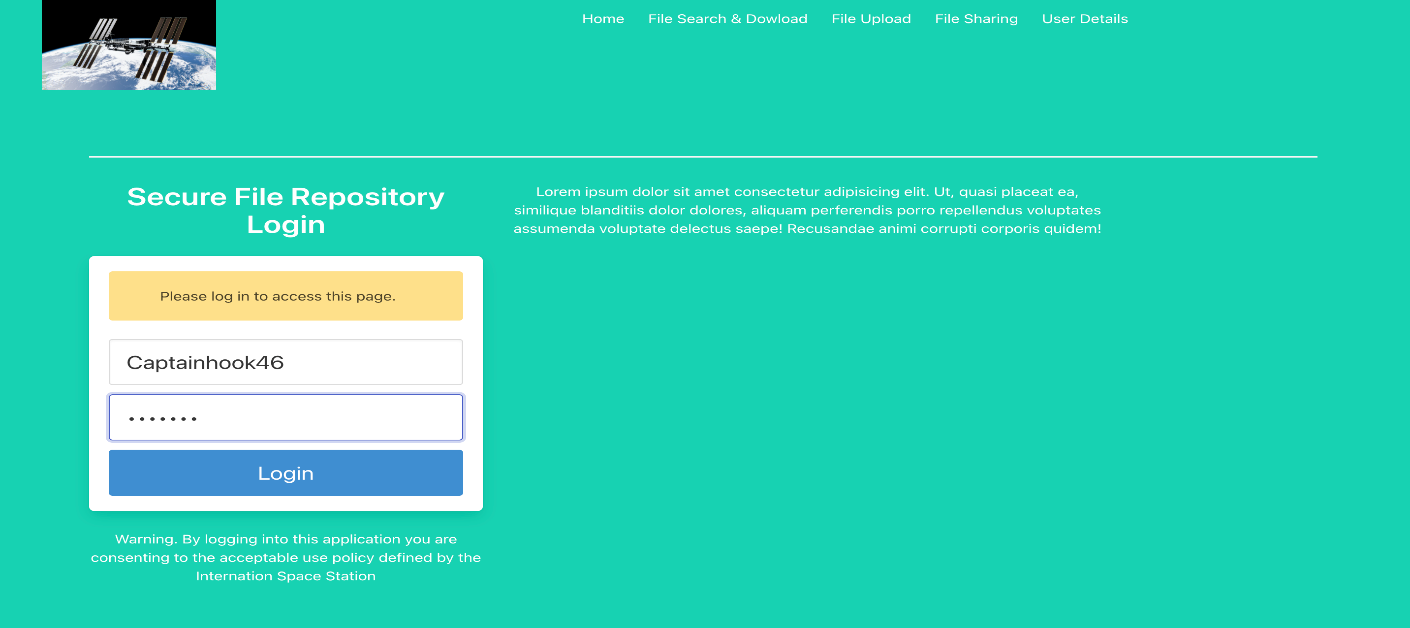
The screenshot below shows that after the user attempts to login with a valid username and invalid password, the login is attempt is unsuccessful and prompts the following message: An access ID and password are required for authentication. Carefully retry your login, contact ISS ground station for support if authentication issues persist.



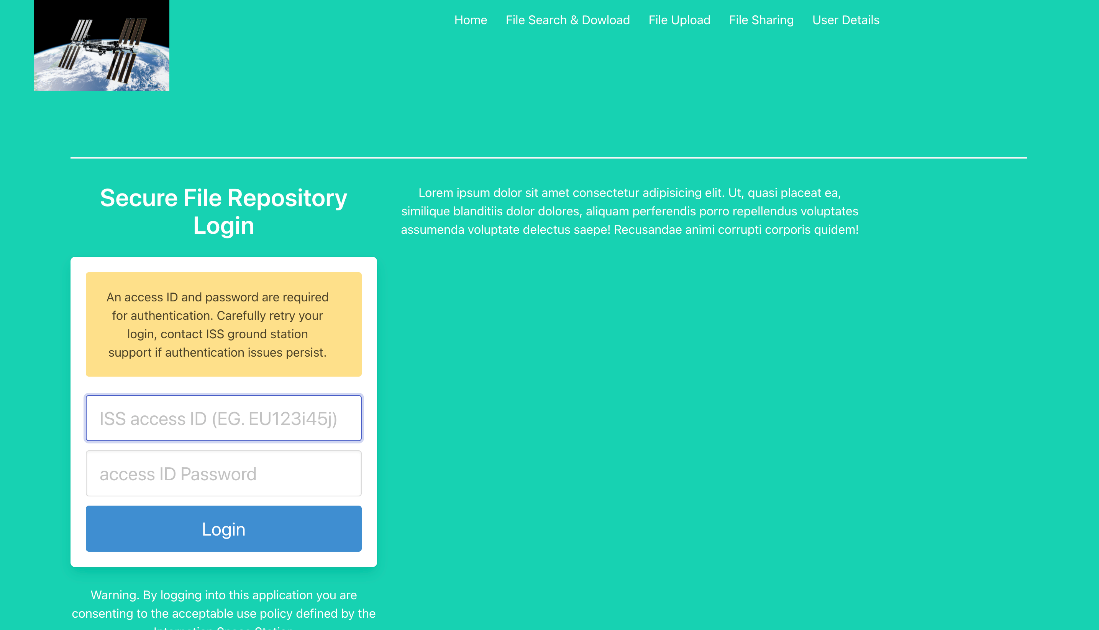
## Authn-03:

### Test the login functionality of secure file repository with an invalid username & any password

The screenshot below shows that a user attempts to login with an invalid username ‘Captainhook46’ and a random password.



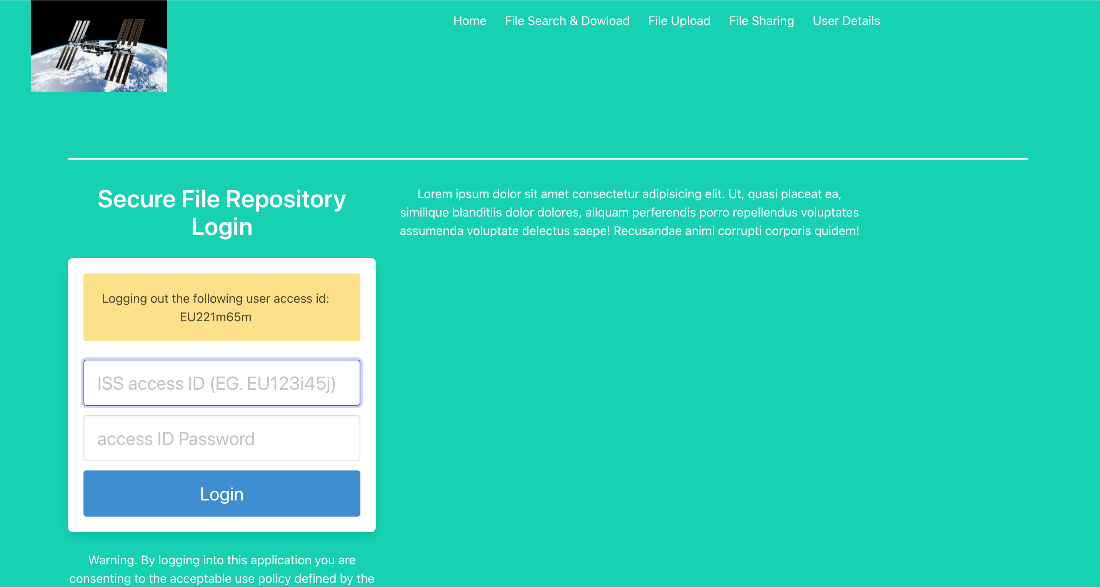
The screenshot below shows the unsuccessful login attempt of the user which provided an invalid username ‘Captainhook46’ and a random password with the prompted message: An access ID and password are required for authentication. Carefully retry your login, contact ISS ground station support if authentication issues persist.



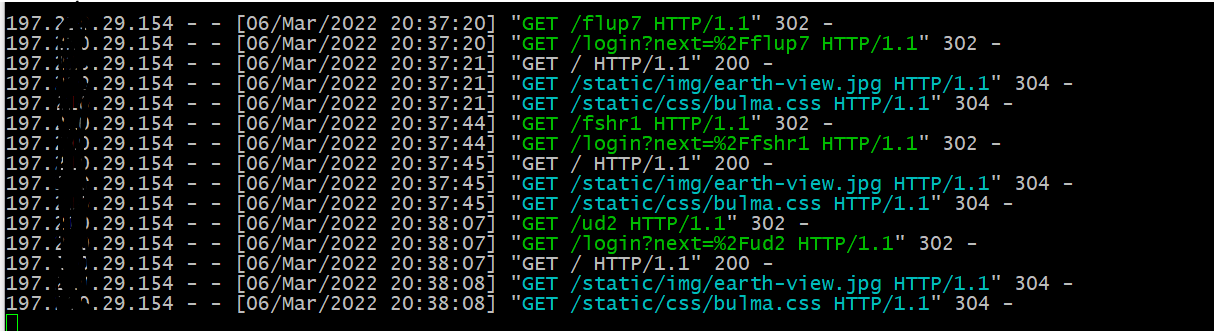
## Authn-04:

### Test the session management functionality when an authorized user logs out of a session and attempts to make use of the application after the expiration of the session

The screenshot below displays the authorized user with access ID EU221m65m logging out of the application which terminates the session as a result



The screenshot below confirms that after signing out, the authorized user with access ID EU221m65m attempted to access the file upload, file search & download, and user details functionalities of the application and was redirected to the login page each time.



## Authn-05:

### Test that a file cannot be deleted or have its sharing permissions altered unless the user owns the file

The screenshot below shows that the authorized user with access ID EU221m65m is unauthorized to delete the file ‘Cyber-City.jpg’ as they are not the authorized owner of the file.



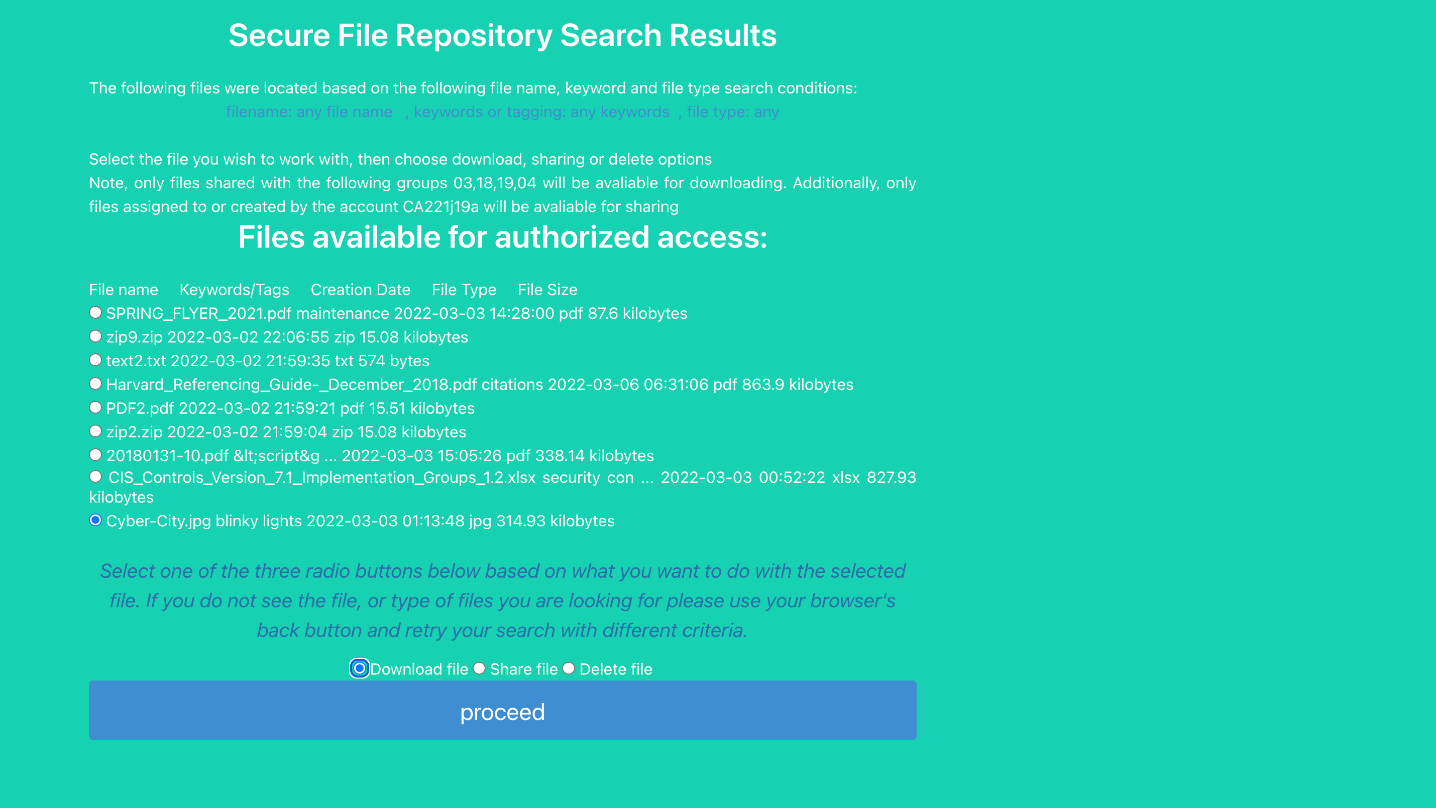
## Authz-01:

### Test the group file sharing functionality of the secure file repository

The screenshot below shows the authorized user Buzz Lightyear with access ID US221a27l sharing the file ‘Cyber-City.jpg’ with the permitted groups 03, 06, 11, and 18.



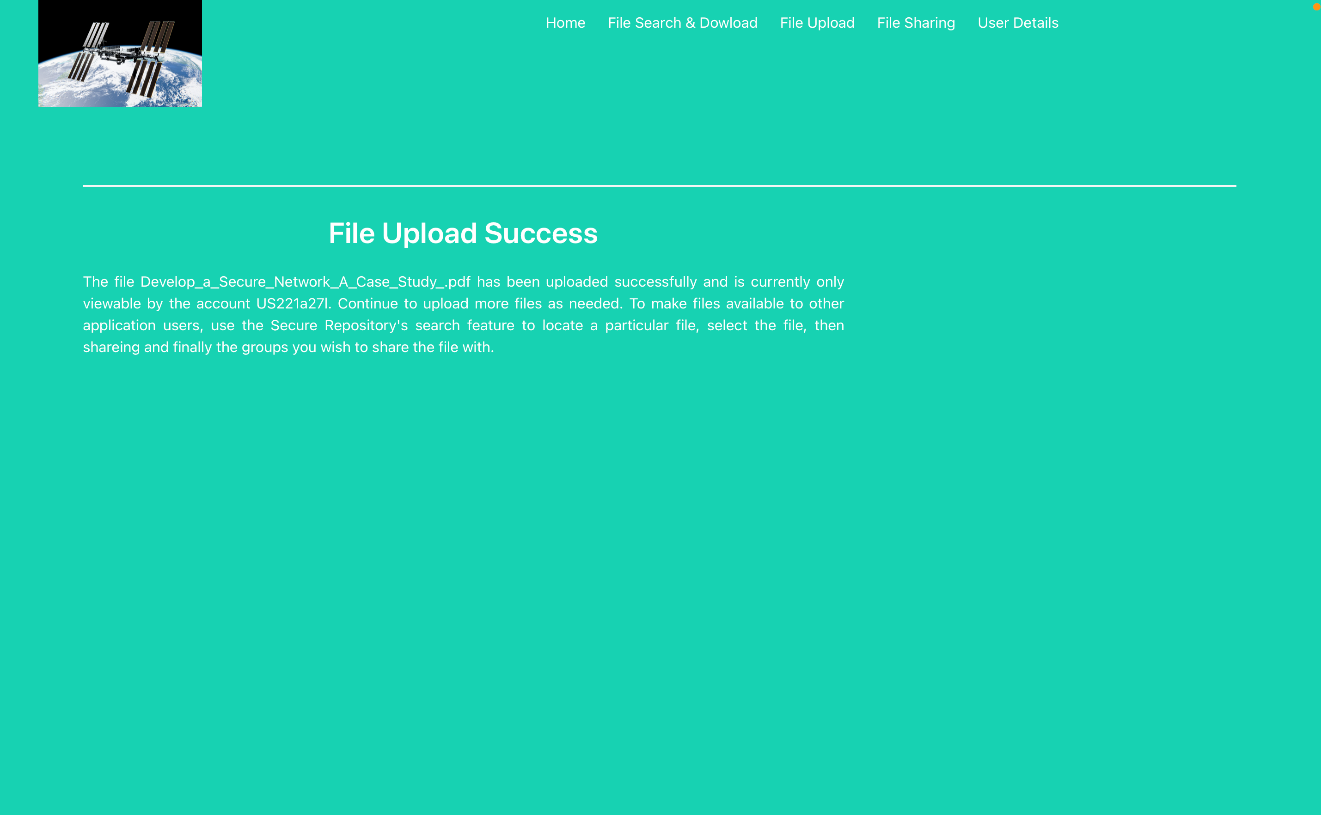
The screenshot below confirms that the intended file Cyber-City.jpg’ has been shared with user Jane Apollo, a member of group 03 with access ID CA221j19a. Consequently, this affirms that the file was shared succesfully from Buzz Lightyear to Jane Apollo.



## Authz-02:

### Verify file uploading functionality of the secure file repository

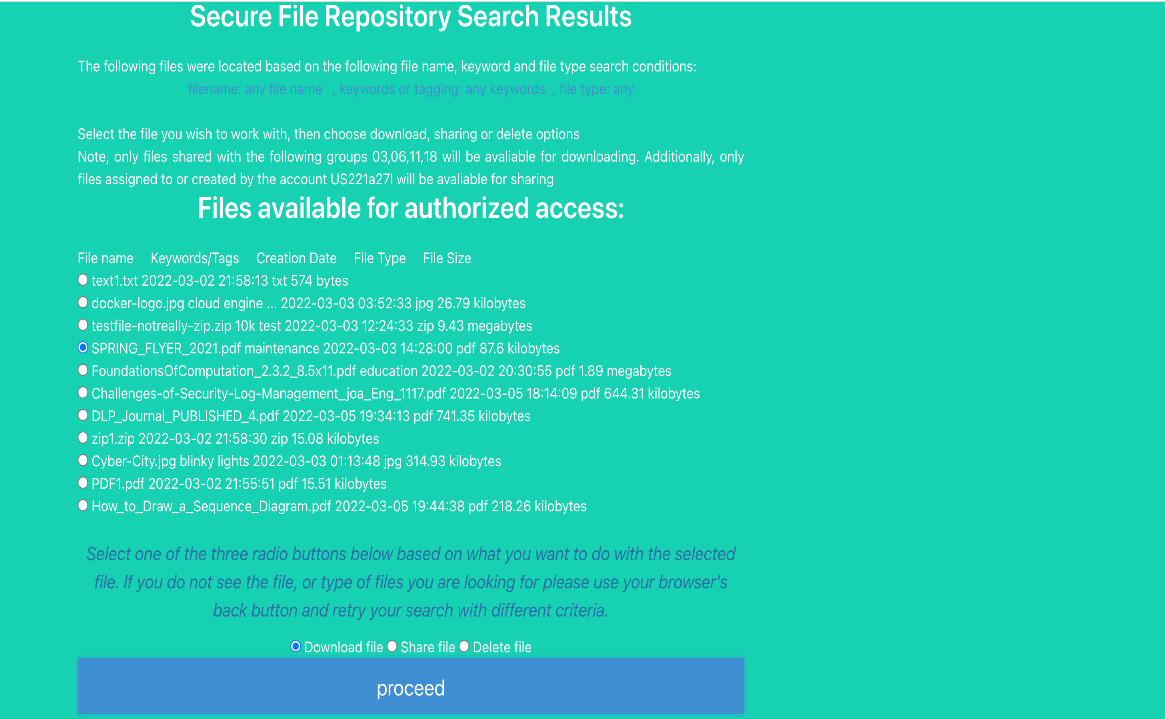
The screenshot below confirms the file uploading functionality of the application with the successful upload of pdf file ‘Develop a Secure Network A Case Study’ from the account of authorized user with access ID: US221a27l



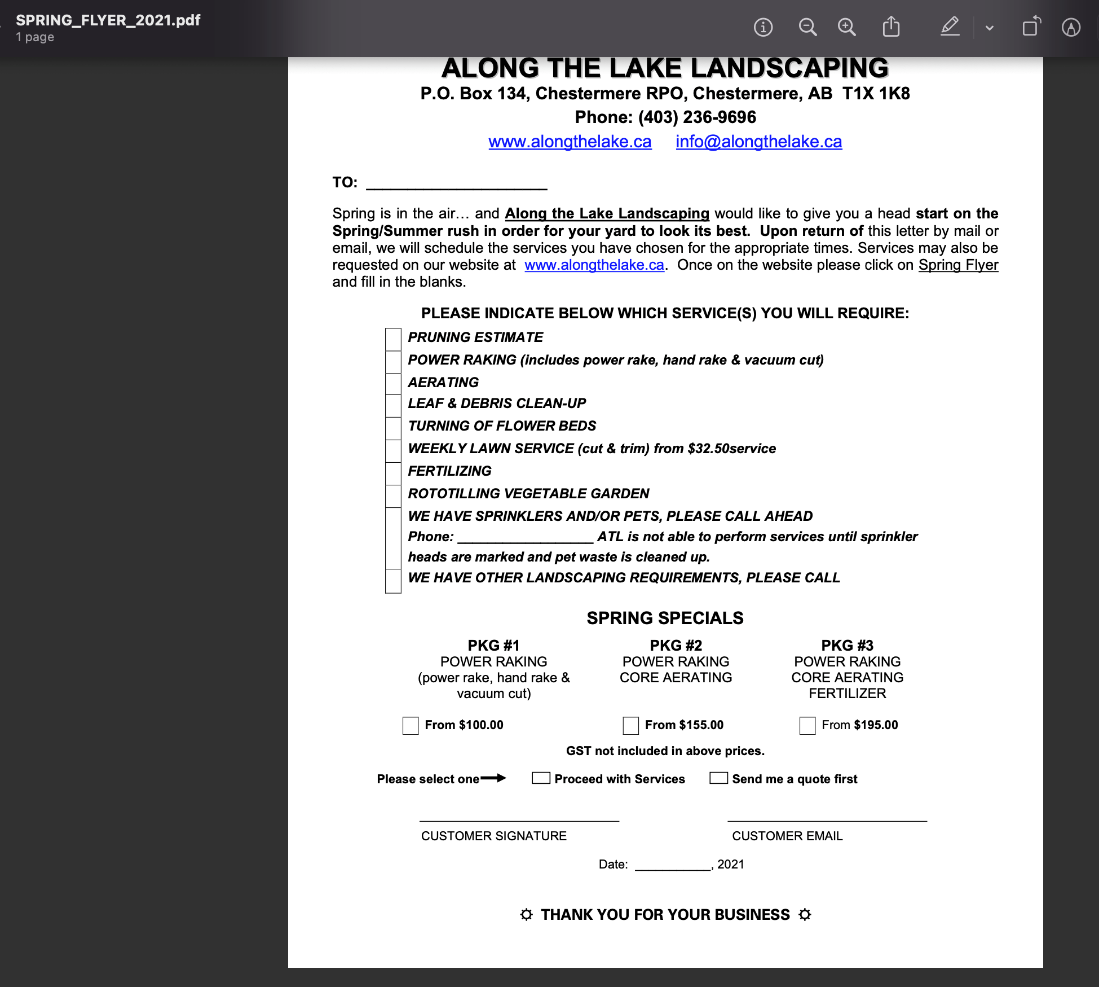
## Authz-03:

### Verify file downloading functionality of the secure file repository

The screenshot below shows the file downloading functionality of the application with the account of authorized user with access ID: US221a27l. The downloaded file is ‘SPRING FLYER 2021.pdf’



The screenshot below confirms the successful download of the file ‘SPRING FLYER 2021.pdf’ from the secure file repository



## Oper-01:

### The SSH command line utility must create a valid application user

Screen shot below is the python CLI application used to create users. The new user’s password is presented on the screen to allow visual confirmation and the ability for the system administrator to share the password with the owner of the newly created account.

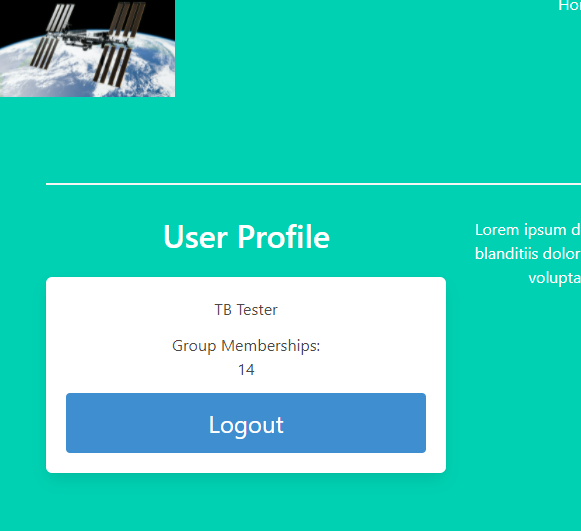
Text

Description automatically generated

The screen shot below confirms the new user’s information is written to the application database, including assignment to the affiliated agency’s private group.



This screen shot below confirms login to the HTTP based app was successful.



## Oper-02:

### Near real-time intrusion monitoring processes logged events for all active sessions on an MVC node and generates alerts on specified conditions

Graphical user interface

Description automatically generated

The log output below shows both events occurred within the same second

{"eventlogtime": "2022-03-06 23:00:45.473", "level": "info", "type": "ActivityTracking", "category": "AuthenticationSuccess", "userid": "0", "URL": "/login", "HTTPMethod": "POST", "SuccessReason": "ValidCredentialUse", "AccessID": "CA221j19a"}

{"eventlogtime": "2022-03-06 23:00:46.382", "level": "warning", "type": "EventOfInterest", "category": "AuthenticationFailure", "userid": "0", "URL": "/login", "HTTPMethod": "POST", "FailureReason": "InvalidPassword", "AccessID": "US221a27l"}

The log output below shows the invalid login also generated an real-time alert which was received at the remote syslog server within the same second time period

Mar 6 23:00:46 10.100.200.6 sfralerting[4242]: WARNING:sfrapp:{'eventlogtime': '2022-03-06 23:00:46.382', 'level': 'warning', 'type': 'EventOfInterest', 'category': 'AuthenticationFailure', 'userid': '0', 'URL': '/login', 'HTTPMethod': 'POST', 'FailureReason': 'InvalidPassword', 'AccessID': 'US221a27l'}

Mar 6 23:00:47 68185a3d7aed syslog-ng[1]: Syslog connection closed; fd='20', client='AF\_INET(10.100.200.6:53440)', local='AF\_INET(0.0.0.0:601)'

The screen shot below is staging the file upload activity for two different users connecting with two different browsers.

Graphical user interface

Description automatically generated

The log output below shows two new log entries created, one for each user. The upload time took approximately 35 seconds, therefore the slight delay with the Chrome client is not relevant to this test, both files were being uploaded at the same time.

{"eventlogtime": "2022-03-06 23:14:26.372", "level": "warning", "type": "EventOfInterest", "category": "FileCreation", "userid": "4", "AccessID": "US221a27l", "FileName": "jginn-5meg-archive.zip", "FileUUId": "bf5a2fa2dc114216a9161635be52cc86", "FileCreate": "2022-03-06 23:14:25"}

{"eventlogtime": "2022-03-06 23:14:31.275", "level": "warning", "type": "EventOfInterest", "category": "FileCreation", "userid": "5", "AccessID": "CA221j19a", "FileName": "jginn-5meg-archive.zip", "FileUUId": "13195b40c42846f281d3105ee511d4f1", "FileCreate": "2022-03-06 23:14:30"}

The syslog output below confirms that the log message creation and transmission is occurring at effectively the same time and both events are associated with different user contexts.

Mar 6 23:14:27 68185a3d7aed syslog-ng[1]: Syslog connection accepted; fd='20', client='AF\_INET(10.100.200.6:53478)', local='AF\_INET(0.0.0.0:601)'

Mar 6 23:14:26 10.100.200.6 sfralerting[4242]: WARNING:sfrapp:{'eventlogtime': '2022-03-06 23:14:26.372', 'level': 'warning', 'type': 'EventOfInterest', 'category': 'FileCreation', 'userid': '4', 'AccessID': 'US221a27l', 'FileName': 'jginn-5meg-archive.zip', 'FileUUId': 'bf5a2fa2dc114216a9161635be52cc86', 'FileCreate': '2022-03-06 23:14:25'}

Mar 6 23:14:27 68185a3d7aed syslog-ng[1]: Syslog connection closed; fd='20', client='AF\_INET(10.100.200.6:53478)', local='AF\_INET(0.0.0.0:601)'

Mar 6 23:14:32 68185a3d7aed syslog-ng[1]: Syslog connection accepted; fd='20', client='AF\_INET(10.100.200.6:53492)', local='AF\_INET(0.0.0.0:601)'

Mar 6 23:14:31 10.100.200.6 sfralerting[4242]: WARNING:sfrapp:{'eventlogtime': '2022-03-06 23:14:31.275', 'level': 'warning', 'type': 'EventOfInterest', 'category': 'FileCreation', 'userid': '5', 'AccessID': 'CA221j19a', 'FileName': 'jginn-5meg-archive.zip', 'FileUUId': '13195b40c42846f281d3105ee511d4f1', 'FileCreate': '2022-03-06 23:14:30'}

Mar 6 23:14:32 68185a3d7aed syslog-ng[1]: Syslog connection closed; fd='20', client='AF\_INET(10.100.200.6:53492)', local='AF\_INET(0.0.0.0:601)'

## Oper-03:

### Multiple MVC nodes must be able to use the same database or work from a replica database

The screen shots below are the configuration files for the new test machine and current production system used to make database connections.

Text

Description automatically generated

Text

Description automatically generated

Once the configuration files are changed over, the application can be restarted, reading the new database connection and users will not perceive the difference. The screenshot below confirms a new MVC node login using the original active database completed successfully.

Graphical user interface, website

Description automatically generated

## Oper-05

### Restored instance for the backup primary database on a secondary environment.

Graphical user interface, website

Description automatically generated

## Sec-01:

### Customized application logging must record all defined events to local file and forward specified security events when conditions are met

The screen shot below shows an HTTP application event that generates a logfile entry so user activity can be monitored but not warranting any immediate investigation.

Graphical user interface, text, application, chat or text message

Description automatically generated

The code output below shows two different users logging out of the application at different times is information that is saved in the application monitoring logs.

{"eventlogtime": "2022-03-06 20:35:32.591", "level": "info", "type": "ActivityTracking", "category": "AuthenticationSuccess", "userid": "0", "URL": "/logout", "HTTPMethod": "POST", "AccessID": "EU221m65m"}

{"eventlogtime": "2022-03-06 21:35:47.703", "level": "info", "type": "ActivityTracking", "category": "AuthenticationSuccess", "userid": "0", "URL": "/logout", "HTTPMethod": "POST", "AccessID": "EU221t22u"}

Both a successful and failed login attempt occurred on the MVC node within a minute, while both events were captured in the logs, the screen shot below confirms only the failed login generated an alert.

Text

Description automatically generated

## Sec-02:

### Failed login events can generate a log message indicating the failure reason was the use of an invalid username

The screen shot below show a login attempt using an access ID that conforms to the SFR convention but is not valid.

Graphical user interface

Description automatically generated

Multiple attempts were made, similar to password spraying attack being perpetrated. To differentiate a user occasionally mistyping their username from an attack, security monitoring teams typically check for a high volume of such events within a short time span. The screen shot below confirms the warnings are received in near real-time and the failure reason is identified.

Graphical user interface, text, application

Description automatically generated

## Sec-03:

### Authentication bypass attempts do not allow access to the application

The correct syntax for SQL injection is dependent on the type of database used by the application, consequently several different SQL injection strings were used. The screen shot below confirms SQL injection does not work as a bypass technique. The yellow warning is presented on all types of failed logins so this vector cannot be used for enumeration attacks either.

Graphical user interface, text, application

Description automatically generated

## Sec-04:

### Authentication bypass attempts will be logged locally and forward a real-time alert

Unknown to the attacker, login attempts that include more than alphanumeric characters in the username input field will generate a real-time alert that is immediately forwarded to the remote monitoring syslog server. To assist incident responders, the contents of the input payload are included in the log message. (Note, additional characters are appended to the malicious input to block any potential attacks to log parsing programs).

The screen below highlights both the input strings used and the security event classification that can be used for security monitoring prioritization and reporting.

Text

Description automatically generated

## Sec-05:

### File ownership automatically assigned to the authenticated user uploading the file upon creation

Database testing and event monitoring performed in conjunction with the file size validation testing illustrated in the screen shot below.

Graphical user interface, website

Description automatically generated

The screen shot of the database query confirms the file owner is ID 13 and the creation times are also recorded accurately

A screenshot of a computer screen

Description automatically generated with medium confidence

The SFR application is also monitoring the MVC node application logs in real time, all events are logged to file but certain events such as uploading a new file to the system are treated as an event of interest and immediately forwarded to the remote monitoring syslog server.

The screen shot below confirms the file owner, and filename are included in the JSON log output. Additionally, there is an event category called FileCreate that could be used by a security monitoring service to prioritize alerts of used as a reporting feature.



## Sec-06:

### The initial login page will enforce input validation of the username field

The screen shot below shows invalid characters being used in the username field

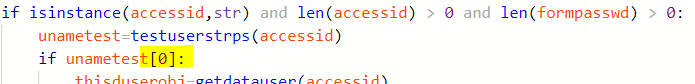
Graphical user interface, text, application

Description automatically generated

Source code review confirms any character from the username input string that is not an ascii letter or digit will result in the function returning a False value, which stops processing the login page immediately and generates a remote alert.

Text

Description automatically generated



Text, letter

Description automatically generated

## Sec-07:

### The file extension and file size are validated prior to acceptance for database storage

The screen shot below shows a text file with a name specially configured to appear like a JPEG file in a typical Windows explorer view, often referred to as a masquerading attack (MITRE, 2021).

Timeline

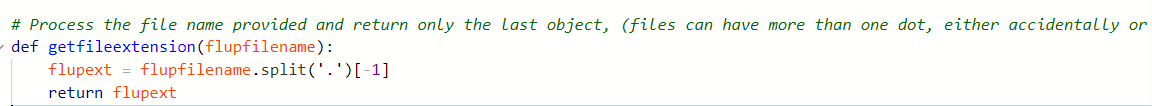
Description automatically generated

The application requires the user to specify the type of file that is being uploaded to the repository from a drop-down menu located beside the file chooser. The screenshot below shows the user selecting a JPEG image file type.

Timeline

Description automatically generated

Prior to upload a custom function called *getfileextension* is called to ensure only the characters after the very last dot in the file name are treated as the extension. As mentioned in the comments, this input validation catches both accidental and malicious use of extra extension format characters in the file name.



The screen shot below confirms the mismatched extension types were detected and no malicious file upload took place.

Graphical user interface

Description automatically generated

The screen shots below show a 4.1 megabyte file selected for upload by the tester completing successfully.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, website

Description automatically generated

Text

Description automatically generated with medium confidence

The screen shots below show a 9.2 megabyte file selected for upload by the tester was not accepted for uploading.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, website

Description automatically generated

A picture containing graphical user interface

Description automatically generated

## Sec-08:

## Data sanitization measures applied to the file name and keyword input fields prior to acceptance for database storage

The screen shot below shows a common cross site scripting test included in the file keyword field which can accommodate up to 255 characters, representing a potential threat vector.

Graphical user interface, text, website

Description automatically generated

The screen shot below confirms the potential HTML characters have been encoded prior to storage, mitigating the potential attack.

Text

Description automatically generated

The screen shots below confirm the use of two input validation functions that are included and maintained as part of the Flask framework.

Graphical user interface, text, application

Description automatically generated

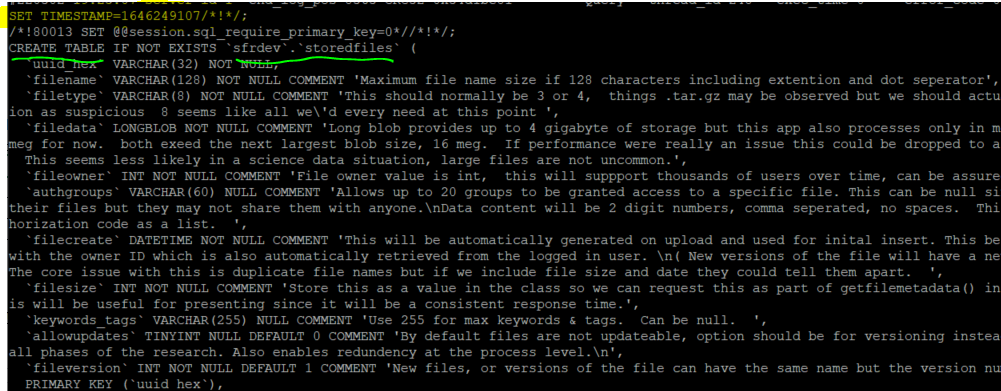
Graphical user interface, text, application, email

Description automatically generated

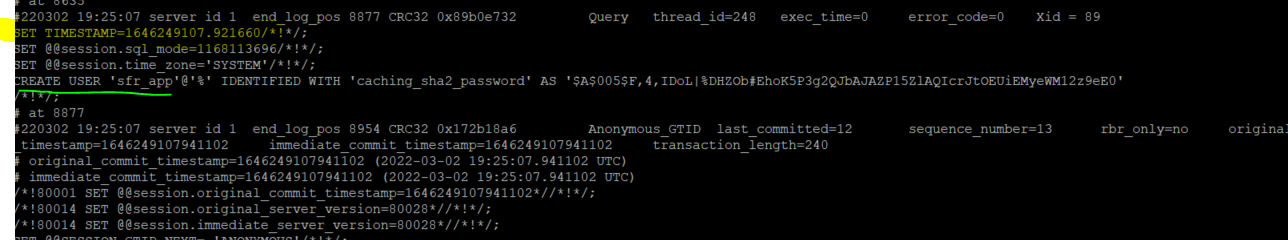
## Sec-09:

### Database transaction logging must record changes to the host database performed by an administrator

The screen shot below shows the creation of a new database table, similar actions such as modifying a table would also be recorded. This is an action performed within the database management system’s database, often called the DBMS or host database



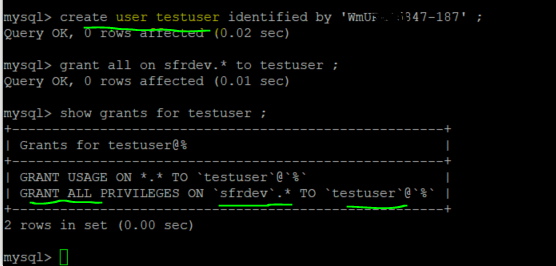
The screen shot below shows a new user being created who could then potentially access the host database remotely, therefore creation and changes to these accounts should be monitored closely.



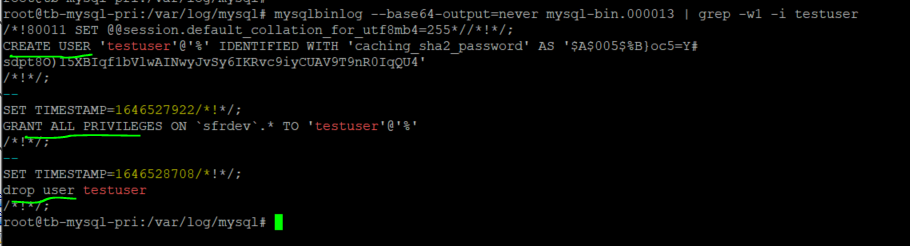
## Sec-10:

### Database transaction logging must record changes to the application database performed by an administrator

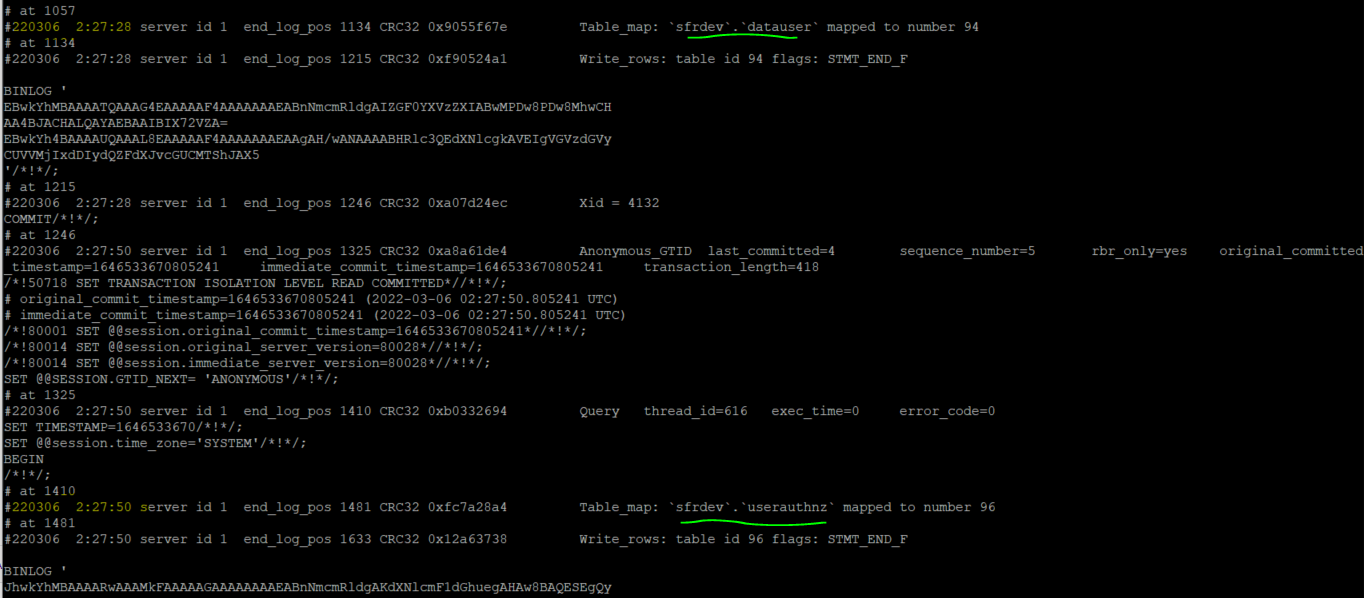
The screen shot below captures a simulated insider threat scenario, a rogue database administrator creates a temporary user with full access to a sensitive database. Presumably this account is then used to extract the desired data after which the account is then deleted. Any application audit logs tracking access to the sensitive data would show the temporary user responsible, yet since the user does not exist attribution for the suspicious activity becomes much more difficult.



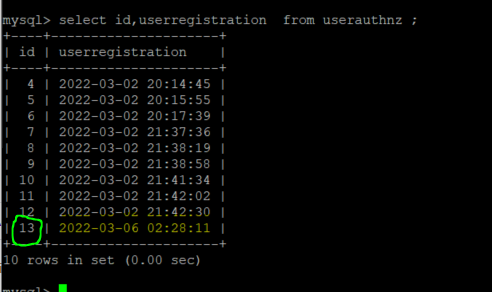
Fortunately, the mysqlbinlog utility has feature switches that support quickly parsing a large binary log for forensic indicators.



MySQL Database transaction may require some additional analysis to determine the sequence of events, in this case the creation of the user account for test case oper-01. The following screenshot highlights actions on two database tables, sfrdev.datauser which stores the personal information of the application user, and sfrdev.auserauthnz which stores the numeric ID and hashed credential. The name of the user created is not humanly readable because MySql transaction logs contain a good deal of binary content.



To ensure this connection can be made, in the case of an insider threat forensic investigation for example, a user creation date is generated and stored in the application table. The numeric ID is automatically generated and unique for the life of the database, therefore a deleted user would leave a gap in the sequence and user registration dates of the records adjacent to missing user can help when developing a timeline.



## Sec-11:

### Credentials stored in application or DBMS databases must not be in clear text and should be salted and hashed where technically feasible

The source code screen shots below show the custom function getauthnz is used to get the user’s password from the Flask User class instantiated for the authenticated user.

Graphical user interface, text, application

Description automatically generated

A picture containing logo

Description automatically generated

Reviewing the application’s models.py module confirms the data structure of the userauthnz table and the user’s password is stored as a 102-byte string.

Text

Description automatically generated

The screen shot below confirms the SHA 256 hash, further processed by 260,000 iterations of a password-based key derivation function called PKBDF2, are stored in the sfdev.authnz table.

