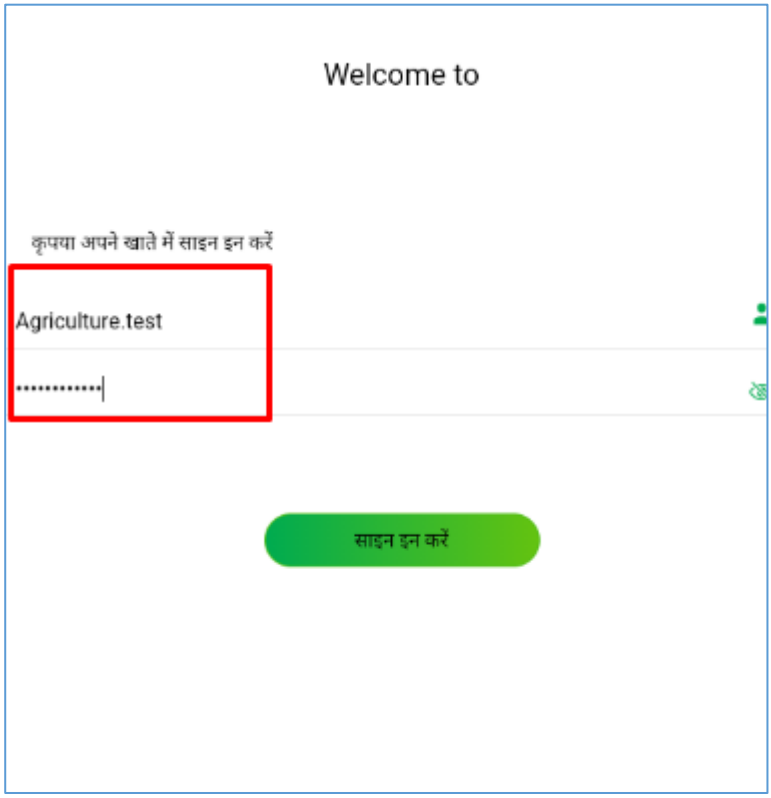
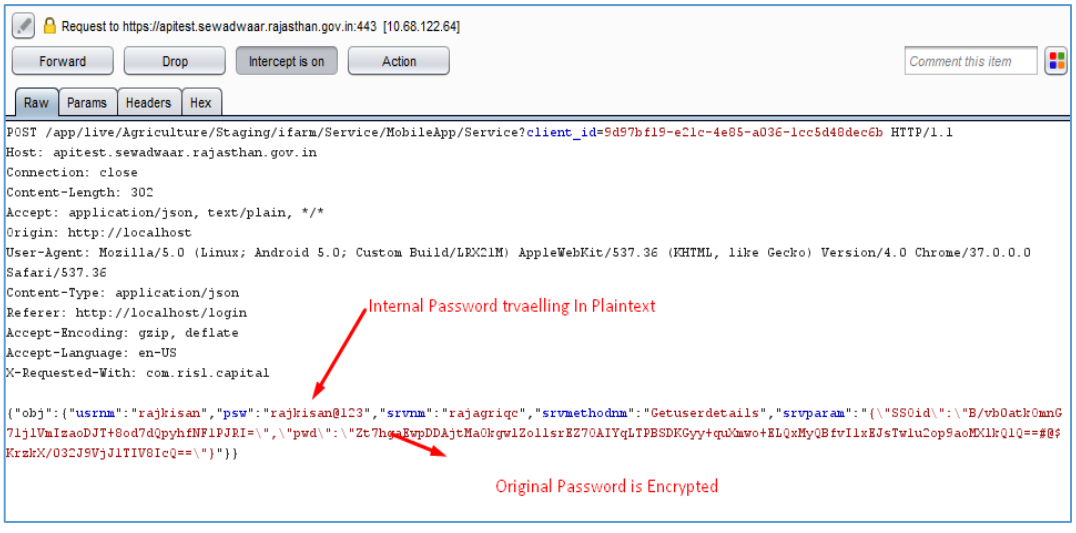
Case 1: Insecure Communication

Sensitive Data Exposure

1. Open the application and login as shown in the snapshot below:



2. Intercept the request using proxy and we get the original password is in plaintext as shown in the snapshot below:



Note:

1. **Password should be in Salted Hashing for Login Form.**
2. **Password should be in Basic Hashing in Registration or Change Password Forms.**
3. **Packets can be captured through Wireshark and alike tools.**

Solution:

I. Sensitive data should be kept encrypted within the database and it should never reflect within the web application interface.

II. Don't store sensitive data unnecessarily. Discard it as soon as possible.

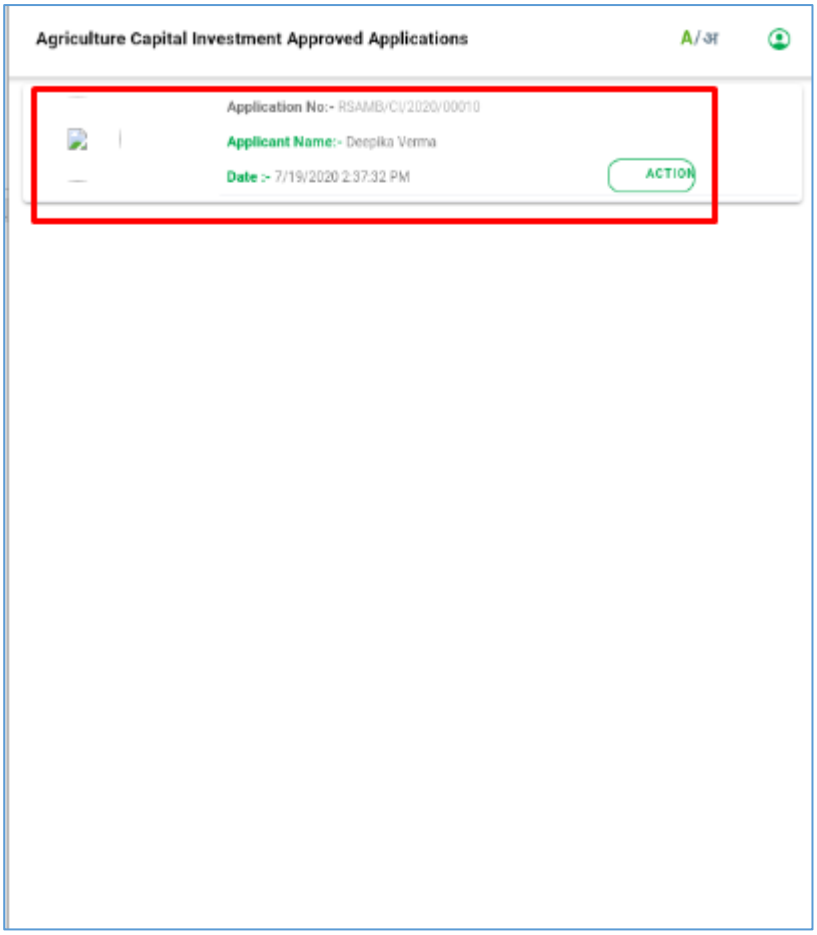
III. Ensure strong standard algorithms and strong keys are used, and proper key management is in place.

IV. At Login page, the password should be strongly encrypted using salted hashing before traversing into the LAN. Use pure hashing where application generates a new password for the users.

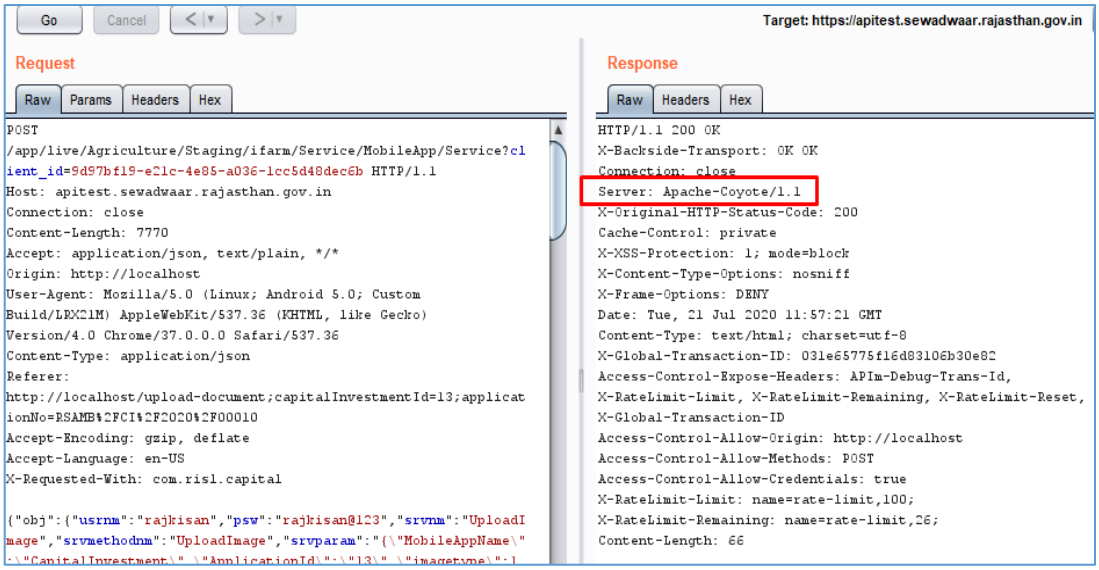
Case 2: Security Misconfiguration

1. Open the application as shown in the snapshot below:

Case: Technology version Disclosure



2. Intercept the response using proxy and we can observe that Server Banner version gets disclosed as shown in the snapshot below:



Solution:

Technology version Disclosure

Server information and version and other technologies used in application should not be disclosed in the response, it should be hidden.

I. Application should make secure to prevent revealing of any kind of error and Hardening process should be carried out periodically.

Case 3: Unsafe Permission Requested

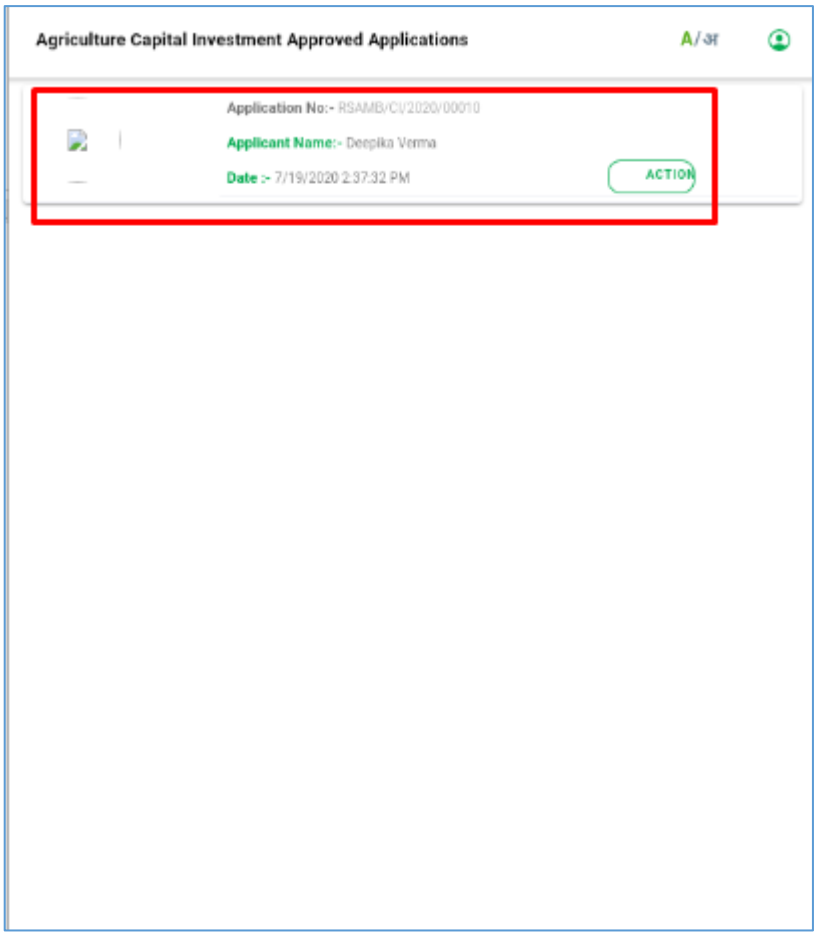
1. Using APK tool, find out the package name of the application; secondly export the components that can be accessed by other applications. Decompile .apk file using apktool then open up the AndroidManifest.xml file.

- Look for android: WRITE_EXTERNAL_STORAGE permission is declared in Android Manifest file. Please refer the screenshot

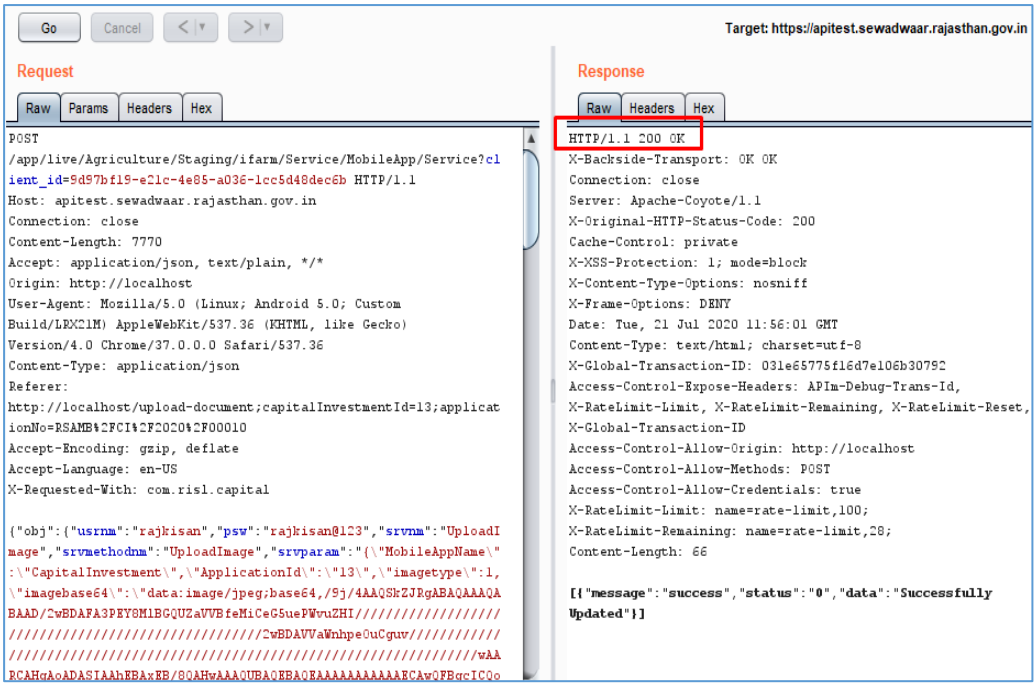
Solution: Perform input validation when handling data from external storage as you would with data from any untrusted source. Do not store executables or class files on external storage prior to dynamic loading. If the app does retrieve executable files from external storage, the files should be signed and cryptographically verified prior to dynamic loading.

Case 4: Insufficient Authorization

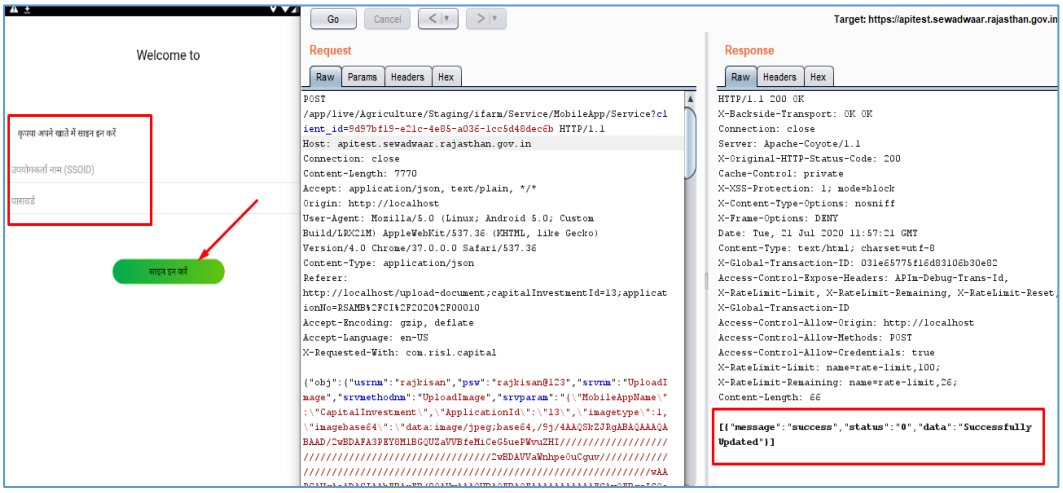
1. Open the application and login with valid user and go to Action as shown in the snapshot below:



2. Intercept the server request and as you can see the request is properly submitted and image is uploaded as shown in the snapshot below:



3. Now logout from the application and repeat the request and the same page will be displayed as shown in the snapshot below:



Access token is still active after logout

Solution:

I. Due to offline usage requirements, mobile apps may be required to perform local authentication or authorization checks within the mobile app's code. If this is the

case, developers should instrument local integrity checks within their code to detect any unauthorized code changes.

II. Password and PIN should be salted and validated against each login attempt.

III. Mobile applications must use strong credentials when accessing sensitive data. The authentication should reflect the user not the device.

Case 5: Security Decision via Untrusted inputs

Case I: Debuggable

1. Using APK tool, find out the package name of the application; secondly export the components that can be accessed by other applications. Decompile .apk file using apktool then open up the AndroidManifest.xml file.

- Look for **android: debuggable** value in the AndroidManifest.xml file, which turned out to be true. Please refer the screenshot

Note: Kindly define android:debuggable=false

Solution:

Case I: Debuggable

- This is considered a security issue because people could backup your app via ADB and then get private data of your app into their PC.

It is recommended that the value of **android: debuggable** must be false.

Case 6: Restrict the user attempt

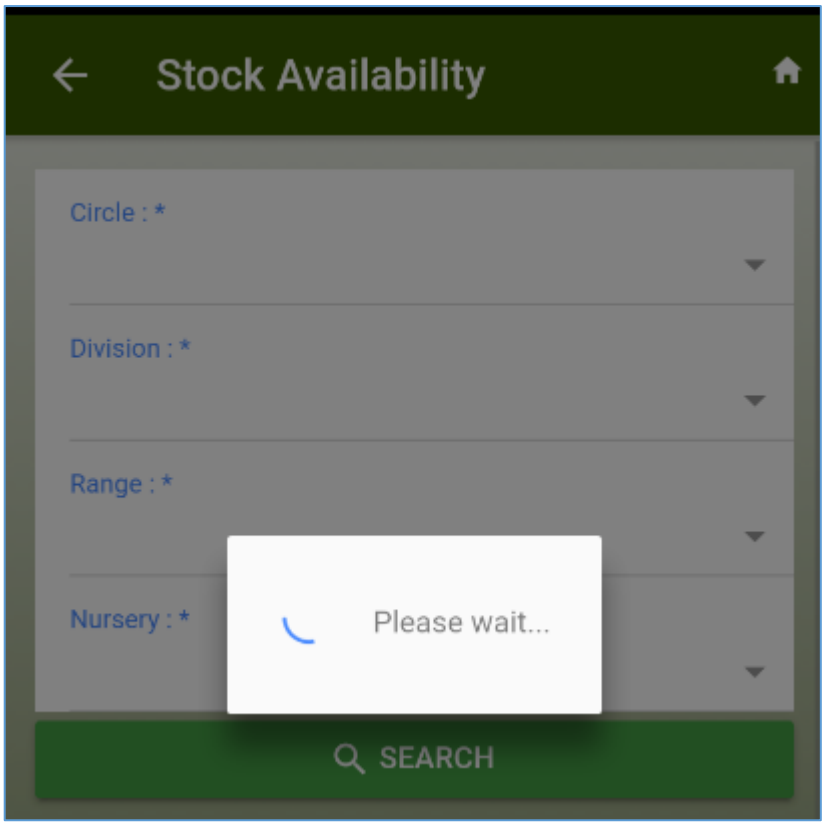
1. In login user can do brute forcing as shown in the screenshot below:

Solution:

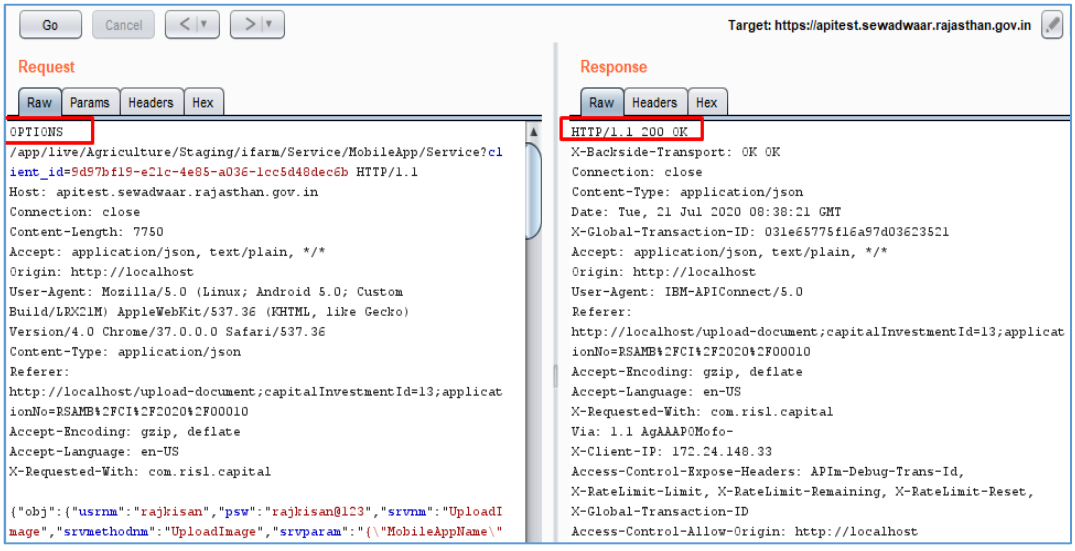
1. User should be block after five wrong attempts.
2. Captcha should be implement on login page.

Case 7: OPTIONS Method Enabled

1. Open the application as shown in the snapshot below:



2. Now intercept the request and send to repeater after that replace the http method GET to OPTIONS as shown in the snapshot below:



Solution: Disable unnecessary allowed (OPTIONS) methods in the application.