|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#TC** | **TC Scenario** | **TC Steps** | **Expectation** | **Observation** | **Risk** |
| TC01 | To check the given **access control** to the users for  application | (*Recruitment*) | Log in (as hiring manager) → Create a vacancy post → Apply for the vacancy as a candidate → Log out → Log in as a regular user → Attempt to accept/reject the application. | Only the hiring manager should be able to accept/reject the application | Hiring manager, and Admin can accept/reject the application; other user cant. | None |
| TC02 | To perform the **purge action** to all users and vacancy posts **|** (*Maintenance***)** | Start EtterCap → Scan hosts → Select gateway and target machine (as target 1, and 2) → Click **ARP Poisoning** → Get credentials → Login as admin → Purge all users’ info and posts | Password & uname should not be shown in the console if provided by user | No output observed in the console | None |
| TC03 | Start ZAP → Scan and **intercept requests** (Ajax spider) → Look for token → Extract credentials → Login as admin → Purge all users’ info and posts | Credential info should be encrypted | Plain uname & pswd found in the response token | Yes |
| TC04 | To check **unauthorized update** of users’ info | (*PIM*) | Start **proxy** in ZAP → Login (**as non-admin**) → PIM → Employee List →  Access users info → Capture this request in ZAP → Modify request → Resend request → Check if info is changed | Non-admin users shouldn’t be able to update any info of other users | Info can’t be updated by non-admin users | None |
| TC05 | To **view other users’ personal info  as non-admin** user | (*PIM*) | Get url of any user → Set **ZAP Requester** giving the url → Iterate for other users (as non-admin access) → View info | Others private info shouldn’t be visible to non-admin | Info is not visible to non-admin users | None |
| TC06 | To upload vulnerable **XSS payloads** in the file attachment |  (*My Info*) | My Info → Personal Details → Attachments → Add **executable payload** → Save → Try to execute  the file | Executable file should not be allowed to be uploaded | Cannot upload executable file | None |
| TC07 | To insert with improper file name to do **Path Traversal Attacks** |  (*My Info*) | Start ZAP intercept → My Info → Personal Details → Attachments → Upload a jpg picture (as it’s acceptable) → Intercept the request → **Modify file name to different path** → Send request → Check if the file is modified and accessible | Shouldn’t be allowed to modify the content with different traversal path | Content can be updated but still needed predefined extensions (i.e., jpg, png, etc) | None |
| TC08 | To check **SQL injection** vulnerability |  (*Login*) | Login page → Enter different **SQL injection payloads** in the uname/pswd field → Check if  login is successful | SQL injections shouldn’t be allowed to be executed | Fails to login by SQL injection | None |
| TC09 | To check for **bruteforce attempts** vulnerability |  (*Login*) | Open browser (from ZAP) as Intruder  → Get a **valid uname** → Tool → Option → Authentication → Position → Add **payloads (pswd)** → Start attack | Multiple attempts from a same uname should cause the account lockout  (or ask to try later) | Nothing happened (after 10 reqs) | Yes |
| TC10 | To check user **creation with malicious entries** |  (*Employee Information*) | Login → PIM → Add (employee) → Enter **malicious texts as name** → Check if it fails/succeeds | Should reject the user creation attempt if invalid text is provided | Allows malicious entries as First/Last name | Yes |