**5.1 DREAD Analysis**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Vulnerability | D | R | E | A | D | Score | Rating | Mitigation & Recommendation |
| Security Misconfiguration | 9 | 6 | 8 | 9 | 6 | 38 | High | * Do not install unused features and frameworks. * Review and update patches as part of the patch management process |
| Software and Data Integrity Failures | 9 | 8 | 8 | 6 | 5 | 36 | High | * Implement a review process for code and configuration changes. |
| Improper error handling | 8 | 8 | 6 | 4 | 4 | 30 | Medium | * Ensure sensitive data or credentials are not stored insecurely including; plaintext credentials in code, public credentials in repositories, and credentials stored in public cloud storage. * Change of default password * SSH keys should be updated periodically and properly secured. |
| SQL Injections | 6 | 6 | 6 | 3 | 3 | 24 | medium | * Input validation * Enforce prepared statements and parameterisation * Actively manage patches and updates * Harden your OS and applications |
| Cryptographic Failures | 8 | 2 | 5 | 6 | 2 | 23 | Medium | * Classify data processed, stored, or transmitted by an application. Identify which data is sensitive according to regulation and business needs. * Ensure up-to-date and strong standard algorithms, protocols, and keys are in place; use proper key management. * Encrypt all data in transit with secure protocols such as TLS with forward secrecy (FS) ciphers, cipher prioritisation by the server, and secure parameters. * Enforce encryption using directives like HTTP Strict Transport Security. |
| Cross Site Scripting (XSS) | 6 | 5 | 4 | 3 | 2 | 20 | Medium | * Prevent HTML code from being entered into inputs whenever possible by preventing them posting that code. * Validating the data to ensure that it meets specific criteria. * Data sanitisation before execution to detect XSS |
| Cross-Site Request Forgery (CSRF) | 6 | 6 | 4 | 2 | 2 | 20 | Medium | * Enforce “deny by default” firewall policies * Sanitise and validate all client-supplied input data |
| Insufficient Logging & Monitoring | 5 | 3 | 3 | 1 | 2 | 14 | Low | * Ensure all login, access control, and server-side input validation failures can be logged to identify malicious accounts and held for enough time to allow delayed forensic analysis. * Ensure log data is encoded correctly to prevent injections or attacks on the logging or monitoring systems. |
| Vulnerable and Outdated Components | 2 | 2 | 2 | 2 | 2 | 10 | Low | * Remove unused dependencies, unnecessary features, components, files, and documentation. * Continuously monitor sources like Common Vulnerability and Exposures (CVE) and National Vulnerability Database (NVD) for vulnerabilities in the components. * Use software composition analysis tools to automate the process. |
| Open ports | 2 | 2 | 2 | 2 | 1 | 9 | Low | * Only use ports that encrypt traffic * Only necessary ports should be used and open * Open ports to the internet should sit behind a firewall. |
| Broken Access Control | 2 | 1 | 2 | 2 | 2 | 9 | Low | * Implement access control mechanisms once and re-use them throughout the application, including minimising Cross-Origin Resource Sharing (CORS) usage |
| Weak login credentials | 2 | 2 | 2 | 1 | 1 | 5 | Low | * Implement a strong password   policy |