**OWASP ASVS with STRIDE**

| **STRIDE** | **Potential Threat** | **Mitigation** | **CWE** |
| --- | --- | --- | --- |
| SPOOFING | No or weak authentication | Communications between application components, including APIs, middleware and data layers, are authenticated. Components should have the least necessary privileges needed. | 306 |
| Leverage unique or special low-privilege operating system accounts for all application components, services, and servers | 250 |
| The application should use a single vetted authentication mechanism that is known to be secure, can be extended to include strong authentication, and has sufficient logging and monitoring to detect account abuse or breaches. | 306 |
| All authentication pathways and identity management APIs should implement consistent authentication security control strength, such that there are no weaker alternatives per the risk of the application. | 306 |
| Elevation of Privilege | No or weak Access Control | Trusted enforcement points, such as access control gateways, servers, and serverless functions should enforce access controls. Never enforce access controls on the client | 602 |
| The application should use a single and well-vetted access control mechanism for accessing protected data and resources. All requests must pass through this single mechanism to avoid copy and paste or insecure alternative paths | 284 |
| Attribute or feature-based access control should be used whereby the code checks the user's authorization for a feature/data item rather than just their role. Permissions should still be allocated using roles. | 275 |
| Tampering | Input / Output Attacks | The input and output requirements should clearly define how to handle and process data based on type, content, and applicable laws, regulations, and other policy compliance | 1029 |
| The serialization should not be used when communicating with untrusted clients. If this is not possible, ensure that adequate integrity controls (and possibly encryption if sensitive data is sent) are enforced to prevent deserialization attacks including object injection. | 502 |
| The input validation is enforced on a trusted service layer. | 602 |
| Output encoding occurs close to or by the interpreter for which it is intended. | 116 |
| The user-uploaded files - if required to be displayed or downloaded from the application - should be served by either octet stream downloads, or from an unrelated domain, such as a cloud file storage bucket. Implement a suitable Content Security Policy (CSP) to reduce the risk from XSS vectors or other attacks from the uploaded file | 646 |
| Information Disclosure,  Tampering | No or Weak Cryptography | Verify that there is an explicit policy for management of cryptographic keys and that a cryptographic key lifecycle follows a key management standard such as NIST SP 800-57 | 320 |
| Verify that consumers of cryptographic services protect key material and other secrets by using key vaults or API based alternatives | 320 |
| Verify that all keys and passwords are replaceable and are part of a well-defined process to re-encrypt sensitive data. | 320 |
| Verify that the architecture treats client-side secrets--such as symmetric keys, passwords, or API tokens--as insecure and never uses them to protect or access sensitive data | 320 |
| Repudiation | Error Logging and Auditing (Accountability) | Verify that a common logging format and approach is used across the system | 1009 |
| Verify that logs are securely transmitted to a preferably remote system for analysis, detection, alerting, and escalation |  |
| Verify that logs does not contain any sensitive data in plain text |  |
| Information Disclosure | Data breach | Verify that all sensitive data is identified and classified into protection levels |  |
| Verify that all protection levels have an associated set of protection requirements, such as encryption requirements, integrity requirements, retention, privacy and other confidentiality requirements, and that these are applied in the architecture |  |
| Insecure Communication | Verify the application encrypts communications between components, particularly when these components are in different containers, systems, sites, or cloud providers | 319 |
| Verify that application components verify the authenticity of each side in a communication link to prevent person-in-the-middle attacks. For example, application components should validate TLS certificates and chains | 295 |
| Spoofing,  Tampering | Misconfiguration | Verify the segregation of components of differing trust levels through well-defined security controls, firewall rules, API gateways, reverse proxies, cloud-based security groups, or similar mechanisms. | 923 |
| Verify that binary signatures, trusted connections, and verified endpoints are used to deploy binaries to remote devices | 494 |
| Verify that the build pipeline warns of out-of-date or insecure components and takes appropriate actions. | 1104 |
| Verify that the build pipeline contains a build step to automatically build and verify the secure deployment of the application, particularly if the application infrastructure is software defined, such as cloud environment build scripts |  |
| Verify that application deployments adequately sandbox, containerize and/or isolate at the network level to delay and deter attackers from attacking other applications, especially when they are performing sensitive or dangerous actions such as deserialization | 265 |
| Verify the application does not use unsupported, insecure, or deprecated client-side technologies such as NSAPI plugins, Flash, Shockwave, ActiveX, Silverlight, NACL, or client-side Java applets | 477 |

**Reference:**