Recommended App Go Live Control List - FinCSIRT

Main components	Sub components	Sub components categories	Actions to be testing at each component
Initial stage			Identify the data criticality/impact to the process and to the business by the application
			Access Control to the application eco system is properly maintained and given minimum permission required to get the assigned job done
			Ensure necessary policies and procedures are defined for all/necessary activities mentioned in the checklist according to the business requirements
			Checklist results should be supported by solid evidence
Backup	**Code**		Ensure the organization defined recovery point objectives is applied to regular backing up
			Regular automatic and manual backup taking should tested
			Change management procedures should be defined and followed
			Backup, testing and rollback strategy should be defined
			Rollback strategy should be tested
			Code must be versioned
			If the source code is not given to the FI, if possible ensure escrow agreements are available based on the business criticality of the application
	Data		Ensure the organization defined recovery point objectives is applied to regular backing up
			Regular automatic and manual backup taking should tested
			Change management procedures should be defined and followed
			Backup, testing and rollback strategy should be defined
			Rollback strategy should be tested
	Application		Application data Create, Update, Read and Delete locations should be documented and reviewed (E.g. ip_address.log will contain the IP address of logged in users))
			Regular automatic and manual backup taking should tested
			Change management procedures should be defined and followed
			Backup, testing and rollback strategy should be defined
			Backup before any changes take in place
			Rollback strategy should be tested
	Infrastructure		Ensure the organization defined recovery point objectives is applied to regular backing up
			Regular automatic and manual backup taking should tested
			Change management procedures should be defined and followed
			Backup, testing and rollback strategy should be defined
			Backup before any changes take in place
			Rollback strategy should be tested
Testing		**Errors and exceptions handling**	Exception and error points are identified, handled correctly and securely
			All exceptions and errors show correct and understandable custom error messages to the user. All unhandled exceptions are to be captured and handled gracefully)
			Error and exceptions does not print any sensitive data to the logs
			Test if the system crashes does not show error/debug information that exposes sensitive information at any point (E.g. alerts, logs…etc)
		Quality	Source code formatting is performed according to the adopted standards (E.g. Linters)
			Source code follows the approved system architecture
			Appropriate comment are used to explain the code according to used coding standard
	Code reviewing	**Security testing**	No hard coded or sensitive constants are being used. If sensitive information is used, ensure proper security is applied (ag: access permission)
			Inventoried all the libraries/frameworks/protocols used to develop the application
			Error pages (404 etc…) are edited to custom error pages (keep the http code as it is) that does not expose sensitive information to client.
			Application code is regularly reviewed to ensure depricated and vulnerable code is identified and managed
			All inputs are properly sanitized minimum in server side
			All input data is validated. Not only html form fields but all sources of input such as API calls, query parameters, http headers, cookies
			Structured data are strongly typed and validated against a defined schema including allowed characters, length and pattern
			Unstructured data is sanitized to enforce generic safety measures such as allowed characters and length, and characters potentially harmful in given context should be escaped
			Log event includes basic necessary information that would allow for a detailed investigation of the timeline when an event happens.
			Security logs are protected from unauthorized access and modification
			All sql queries are protected by the prepared statements or query parameterization in order to secure against sql injection attacks
			Static security testing has completed, fixed and reviewed successfully before the launch
			Application requirements (performance/load etc..) are compatible with allocated hardware resources. Future and current hardware requirements are gathered and ensure it is facilitated

		Server	Application scalability requirements are supported according to the requirement
			All/critical supporting hardware and software systems have valid maintenance agreements
			Based on the allocated server criticality, has required high availability resources (multiple network interfaces etc..)
			Allocated server/appliances can handle power outages with following minimum requirements based on the criticality
			• backup power is available
			• graceful shutdown configured with the Uninterrupted Power Supply (UPS)
			• backup power duration is enough to support backing-up critical/all systems and gracefully shutdown/any other according to the organizational policies
			• data integrity and high availability supported via raid configuration
			Verify that authorized administrators have the capability to verify the integrity of all security-relevant configurations to ensure that they have not been tampered with. This could be verified by the line managers.
			Staging environment is available for testing platform and application updates
		Network devices	Time sources should be synchronized to ensure logs have the correct time (E.g. NTP)
			All required ACLs are provided accordingly to the application design
			Network devices are patched to the latest security release.
			All administrative access to the network devices are maintained with proper credentials and authorization management
			Device patch /update and configuration rollback procedures are maintained.
		Application software	TLS protocol version and cipher suite for your application is properly configured (recommended TLS 1.2 +)
			Backend TLS connection failures are logged
			If possible, TLS certificate public key pinning (hpkp) is implemented as appropriately
			If possible, Http strict transport security headers are included on all requests and for all subdomains,
			Only strong and non-deprecated algorithms, ciphers, and protocols are used (E.g. Use resources such as NIST)
			TLS settings are in line with current leading practices, particularly as common configurations, ciphers, and algorithms become insecure.
			Web server accepts only a defined set of required http request methods, such as get and post are accepted, and unused methods (e.g. Trace, put, and delete) are explicitly blocked
			Suitable x-frame-options header is in use for sites where content should not be viewed in a 3rd-party x-frame
			Http headers or any part of the http response do not expose detailed version information of system components
			Content security policy (CSP) is in place that helps mitigate common DOM, XSS, JSON, and javascript injection vulnerabilities.
			Verify that the application has defenses against http parameter pollution attacks(get, post, cookies, headers)
			All development and debug features should be disabled.
			Application software (Apache , Tomcat etc..) should be in production mode rather than in debug or development mode.
	Infrastructure	**Repositories (code/library)**	Production repositories are version controlled for tracking the change management
			Production repositories are monitored through a FIMS for integrity verifications
			Source code repositories should be properly versioned for tracking and change management
			Only the licensed repositories/libraries are used
		Databases	Database is secured using a firewall/network level access control mechanism which filters out unauthorized connections.
			All confidential/sensitive data are encrypted
			Ensure access control mechanism available for database
			Database service accounts are properly secured using strong passwords, proper authorizations and other mechanisms like views
			Password are stored in hashes/salted hashes
			Verify account lockout mechanism implemented
			Audit logs are enabled and procedures for continues monitoring on both successful logins and unsuccessful attempts are in place
			Appropriate policy procedure like change management procedure/user creation procedure /backup &recovery policy do exists
			Database backups are taken and tested regularly (backups should be encrypted when in storage)
			Database configuration files and source codes are only accessible through authorized accounts
			Database update and rollback procedures are documented and tested
			User level authentication for function (SELECT, DELETE, DROP, UPDATE ..etc) should be given to access tables
			Super user accounts should be kept aside securely without used for operations (Unless an emergency) and use only the pertaining user based, minimum permissioned accounts
		Operating System	Operating systems are maintained with available security updates (no discontinued versions)
			Systems are hardened with following minimum requirements and the rest is hardened based on the criticality of the server to the process
			• OS and supporting applications are up-to date with latest security patches
			• Unnecessary os services are disabled
			• OS audit trail enabled
			• verify anti-virus is enabled and up-to date
			Only the authorized personals are allowed to connect to the system locally/remotely
			Patch management and continuous security updates are maintained.
			Patch and update rollback procedures are maintained and tested.

		Cabling and network architecture	Cabling matched with the performance requirements of the service provided.
			Network architecture supports business requirement level segmentation and isolation levels required for immediate incident response
			Network architecture, physical and virtual network diagrams are available with the placement of the application in its designated location.
			Only the required services/ports/entities are allowed to the production application
	Application	**Functionality vs business logic testing**	All application features/components are identified and confirm as required.
			All library modules and external systems needed by the application are identified
			Threat model for the target application has been completed and covers: spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege risks
			All external communications/API calls are recommended to handle via a central management system where it can manage Authentication, Authorization, Audit (AAA)
			Application has up-to-date maintenance and service agreements with the relevant parties
		Session handling testing	Language specific session managers are being used or custom session manager is resistant against all common session management attacks
			Sessions are invalidated when the user logs out.
			Sessions timeouts after a specified period of inactivity
			Session id is never disclosed/sent in urls, error messages, or logs. This includes verifying that the application does not support url rewriting of session cookies
			All successful authentication and re-authentication generates a new session and a new session id.
			Sufficient measures are taken to ensure session ids generated by the application are the only ones recognized as active by the clients
			Session ids are sufficiently long, random and unique
			Session ids stored in cookies have their path set to an appropriately restrictive value. Additionally set the “http only” and “secure” attributes.
			Users are prompted the option to terminate all other active sessions after a successful change password process
			An active session list is displayed in the account profile
		Security testing	Server side input handling
			Server side error handling
			Dynamic security test/vulnerability assessment should be conducted , fixed and verified
			There are no sensitive business logic, secret keys or other proprietary information in client side code
			All data communication ingress/egress and data sinks should be validated for securing against backdoors and other malicious codes
			All the default password should be forcefully changed at the first login
			Any kind of field pre-filling with referenced to the credential fields are not recommended
			Authentication & authorization controls are enforced on both server and client side
			Verify all authentication failures are securely handled
			Password complexity or passphrase policies are enforced
			Authentication decisions are logged, without storing sensitive session identifiers or passwords
			Passwords are stored in one way hashed with a salt
			Credentials are only being transported as encrypted
			Password reset function does not reveal the current password.
			Multiple failed login attempts will result in account lockout
			Two factor authentication is highly encouraged
			Administrative interfaces are not accessible to untrusted parties/outside, and secured using multiple ACLs
			TLS is being used in all communications between a client and server. This includes connection to database via database drivers
			Code is signed with an authorized digital certificate where ever it is possible
		Usability testing	Application/content is easy to understand by the end users
			Provided instructions are clear and will satisfy its purpose
			Clear navigation options should be provided at every location.
		Interface testing (communication interfaces)	Proper error messages are passed when unexpected interrupts happen between interfaces.
			Client is provided with clear message, but sensitive information should not be provided in the error shown to them.
		Compatibility testing	All supported compatibility client and other platforms are well documented (browsers/mobile platforms/operating systems etc…)
			Informational / enforcement actions are performed when the application is accessed from an unsupported platform /client
		Performance testing	Application is load tested and all systems are ensured to work as expected on peak load times
			Application is stress tested to identify breaking loads and requests so that alerts get triggered when they are being reached.
			Work-around are well tested and documented to respond in a high load situation
		Integration testing (with other systems)	All integrations with other systems are well documented
			All integrations are tested with respective development teams to ensure smooth operations

			Exceptions are handled within each system so that acid (atomicity, consistency, isolation, durability) properties are maintained
		Application development environment	Unit tests conducted and successfully fixed
			Code reviews conducted and all reviews are successfully completed
			Integration tests conducted and successfully fixed
			Regression testing conducted on the last build and all issues are fixed [doesn't have to be last build, can be done in GA phase]
			Alpha and beta testing has conducted with positive results
	User testing		QA & UAT testing has successfully completed on the final build of the application
	Change management		Bug tracking and management procedures/systems are available
			Emergency change procedures are defined
			Change management procedures for the system are implemented and enforced