Web Application Server Control Checklist

	TECHNICAL CONFIGURATION	
	☐ Web Application Server Security Baseline	● Verify that critical information about system components (e.g. server name, version, installed program versions, etc.) of web, application and database servers is obscured and not revealed via HTTP responses or error messages. ● Verify that error messages of web applications and application-server default error messages are not displayed in details to clients ● Ensure that directory listing is disabled on application and web servers. ● Verify that sensitive links are not indexed by search engines by not listing them within robots.txt files. Verify that critical webpages (e.g. administration panel) are not explicitly linked within the web application, and are included within robot.txt files. ● Install latest patches for application frameworks, application servers, database servers and web servers. ● Configure patching deployment solution for application frameworks, application servers, database servers and web servers. ● Disable all HTTP methods except GET and POST if they are not in use ● Remove access to non-public resources (e.g. backup files, development test files) and unnecessary applications (e.g. default web server sites, demo applications) except where necessary. ● Enable security features for application frameworks (e.g. ASP.NET, PHP, STRUTS) ● Remove default user accounts from applications, systems and databases. ● Enable HTTP/HTML attributes (e.g. autocomplete, cache-control, pragma) and configure to prevent storage of sensitive information like passwords within caches ● Disable weak SSL algorithms and only allow secure algorithms for communication over SSL ● Configure cross-domain policies (for Flash corssdomain.xml and for SilverLight clienaccesspolicy.xml) in a secure manner. Allowing access to all domains should be prevented if not required ● Disable SSL renegotiation features to mitigate DoS and MITM attacks ● Configure strong and complex passwords for administrators and client use

		● Verify that passwords and secret question-answers of password retrieval functions are not stored in plain text ● Ensure that any critical data (e.g. password, credit card, and personal data) exchanged between clients and servers only travels over secure HTTPS protocol. ● Verify authentication and authorization for any access to non-public resources is performed server-side ● Configure password hash generation to use a salt value ● Configure user accounts so that they are forced to change their initial password, received via envelope or via email, when first accessing the system ● Ensure that all critical activities within applications are logged at application and server levels ● Configure authentication failure messages with a common message. Example “Username and/or Password is wrong” ● Verify that all successful and unsuccessful authentication attempts and access attempts to resources are logged ● Configure the generation of unique vales for session management (e.g. session identifiers, token, etc.) to use secure random number generators ● Ensure that an inactivity timeout is setup for sessions ● Verify that after each authentication and reauthentication, a new session id is created and the old session id is invalidated. Session id should be invalidated for logouts as well. ● Integrate solutions like tokens and CAPTCHA s for critical operation in order to prevent Cross-Site-Request-Forgery (CSRF) attacks ● Restrict values for the domain and path for cookies containing authenticated session identifiers to an appropriate value ● Configure httponly attribute on cookies. In addition, configure the secure attribute for cookies for HTTPS communications ● Verify that after successful authentication operations, users are redirected via HTTP 302 to internal pages ● Configure all pages of accessed applications with logout links ● Consider parameter manipulations within GET/POST request and prevent unauthorized access to legal user resources by attackers ● Ensure that system users that own an application process only have access rights to the directory of the application ● Ensure that database users only have access to the database resources that are used by the application

		● Verify that database users are only able to access database servers through the relevant application server IP address ● Implement synchronization mechanisms over critical resources, objects and methods to prevent race-conditions ● Configure web-based statistics applications installed on servers to only be accessible by authorized users ● Ensure that only authorized users and roles can access restricted URLs, functions, object references, services, application data, user information and security configuration files ● Establish process for removing access rights for users who no longer need access (e.g. leaving company, changing role within the project, etc.) as soon as possible ● Configure password changes to always require current password before changing password ● Configure password retrieval functions to support secret questions and similar arguments ● Verify that password retrieval functions do not send user names and passwords within emails. Instead, a link with certain lifetime should be sent that prompts a dialog for password change ● Reconfigure names of critical directories like administration panels with names that are not easily guessable ● Remove unnecessary resources (e.g. test codes, demo applications, backup files) from applications that are being transferred from development/integration environment to production environment. Remove source files if they are not required. Remove comments from source files. ● Check integrity of files transferred from development to production and guarantee their integrity. ● Verify that sensitive files and resources belonging to application domain are not indexed via Google/Bing search engines and are not accessible to third parties ● Verify that all user inputs are validated on server-side. White-lists should be preferred for validation instead of black-lists. Each input should be encoded to a common character set before validation (canonicalization) ● Ensure that user inputs are escaped and validated before using as part of command execution ● Implement prepared statement, parameterized query, bind variables and whitelist data validation to prevent SQL injection attacks ● Apply output escaping/encoding on all user inputs before they are displayed on their screens. For additional security, user inputs can be checked for type, length and content

		• Verify that user inputs regarding arithmetic operations are checked and validated for minimum and maximum values • Ensure that user inputs regarding file access operation are normalized and validated • Configure file upload operations to check the name, length, type and content of files • Configure HTTP redirects using user input to validate using a whitelist method to prevent phishing attacks (open redirect problem) • Verify that user inputs used for LDAP queries are sanitized before connection is established • Verify proper user input sanitization for CR/LF characters sent within user inputs. User inputs with CR/LF should not be directly appended within HTTP responses on the application side (CRLF injection attack) • Ensure that appropriate solutions against frame busting and clickjacking attacks are implemented within web applications. • Perform penetration testing of web application before application goes online • Implement CAPTCHA or similar anti-automation security controls within HTML forms to prevent DoS, brute-forcing and dictionary attacks • Enable timeout for search functionalities which force databases to perform CPU-intensive queries by using several search wildcards like "%" or "*" to prevent SQL Wildcard attacks • Configure authentication for accessing web services implemented with SOAP, Rest, XML-RPC or similar technologies

Tech Signature Indicating Above Procedures Performed: ______________________ ______________

Supervisor Signature Indicating Sever Hardening Procedures Performed: ____________________