Introduction

This document represents the check list that has to be used to evaluate the security aspects of a CLIPS service. For each check, specify whether it is present by using T or F values and, if necessary, add some notes.

General Information

Service name	SAS03 Obtain census certificates
Service description	Procurement of official census certificates, which is the most demanded procedu Santander citizens. This document can be retrieved at http://clips.tst-sistemas.es by each person in possession of a valid digital certificate and a home in the c Santander.
Responsible organization	TST
Reference contact	Arturo Medela (amedela@tst-sistemas.es)

Input Validation

Conduct all data validation on a trusted system (e.g., The serv	T	
Identify all data sources and classify them into trusted a untrusted. Validate all data from untrusted sources (e. Databases, file streams, etc.)	F	
There should be a centralized input validation routine for th application	T	
Specify proper character sets, such as UTF-8, for all sources of input	T	
Encode data to a common character set before validat (Canonicalize)	T	
All validation failures should result in input rejection	T	
Determine if the system supports UTF-8 extended character set and if so, validate after UTF-8 decoding is completed	T	
Validate all client provided data before processing, including parameters, URLs and HTTP header content (e.g. Cookie n and values). Be sure to include automated post backs fr JavaScript, Flash or other embedded code	T	
Verify that header values in both requests and responses con only ASCII characters	T	
Validate data from redirects (An attacker may submit malici content directly to the target of the redirect, thus circumventin application logic and any validation performed before the redirect)	F	

Validate for expected data types	T	
Validate data range	T	
Validate data length	T	
Validate all input against a "white" list of allowed characte whenever possible	F	
If any potentially hazardous characters must be allowed as input be sure that you implement additional controls like outp encoding, secure task specific APIs and accounting for utilization of that data throughout the application . Examples o common hazardous characters include: < > " ' % ( ) & + \'	F	
If your standard validation routine cannot address the follow inputs, then they should be checked discretely Check for null bytes (%00) Check for new line characters (%0d, %0a, \r, \n) Check for "dot-dot-slash" (../ or \..) path alterations characters in cases where UTF-8 extended character set encoding is supporte address alternate representation like: %c0%ae%c0%ae/	T	

Output Encoding

Conduct all encoding on a trusted system (e.g., The server)	T	
Utilize a standard, tested routine for each type of outbou encoding	T	
Contextually output encode all data returned to the client that originated outside the application's trust boundary. HTML entity encoding is one example, but does not work in all cases	F	
Encode all characters unless they are known to be safe for intended interpreter	T	
Contextually sanitize all output of un-trusted data to queries for SQL, XML, and LDAP	F	
Sanitize all output of un-trusted data to operating systen commands	F	

Authentication and Password Management

Require authentication for all pages and resources, except tho specifically intended to be public	T	
All authentication controls must be enforced on a trusted syst (e.g., The server)	T	
Establish and utilize standard, tested, authentication servi whenever possible	T	
Use a centralized implementation for all authentication control	T	

including libraries that call external authentication services		
Segregate authentication logic from the resource being reques and use redirection to and from the centralized authenticati control	T	
All authentication controls should fail securely	T	
All administrative and account management functions must be least as secure as the primary authentication mechanism	T	
If your application manages a credential store, it should ens that only cryptographically strong one-way salted hashes o passwords are stored and that the table/file that stores t passwords and keys is writable only by the application. (Do no use the MD5 algorithm if it can be avoided)	F	
Password hashing must be implemented on a trusted syster (e.g., The server).	T	
Validate the authentication data only on completion of all da input, especially for sequential authentication implementations	T	
Authentication failure responses should not indicate which par the authentication data was incorrect. For example, instead of "Invalid username" or "Invalid password", just use "Inva username and/or password" for both. Error responses must truly identical in both display and source code	F	
Utilize authentication for connections to external systems th involve sensitive information or functions	T	
Authentication credentials for accessing services external to application should be encrypted and stored in a protect location on a trusted system (e.g., The server). The source co NOT a secure location	T	
Use only HTTP POST requests to transmit authentica credentials	T	
Only send non-temporary passwords over an encrypte connection or as encrypted data, such as in an encrypted er Temporary passwords associated with email resets may be exception	F	
Enforce password complexity requirements established by po or regulation. Authentication credentials should be sufficient withstand attacks that are typical of the threats in the deploy environment. (e.g., requiring the use of alphabetic as well as numeric and/or special characters)	F	
Enforce password length requirements established by policy regulation. Eight characters is commonly used, but 16 is bette consider the use of multi-word pass phrases	F	
Password entry should be obscured on the user's screen. (e.g., o	F	

web forms use the input type "password")		
Enforce account disabling after an established number of inv login attempts (e.g., five attempts is common). The account m be disabled for a period of time sufficient to discourage brute force guessing of credentials, but not so long as to allow fo denial-of-service attack to be performed	F	
Password reset and changing operations require the same lev controls as account creation and authentication.	F	
Password reset questions should support sufficiently rando answers. (e.g., "favorite book" is a bad question because “The Bible” is a very common answer)	F	
If using email based resets, only send email to a registered address with a temporary link/password	F	
Temporary passwords and links should have a short expira time	F	
Enforce the changing of temporary passwords on the next use	F	
Notify users when a password reset occurs	F	
Prevent password reuse	F	
Passwords should be at least one day old before they can be changed, to prevent attacks on password reuse	F	
Enforce password changes based on requirements establishe policy or regulation. Critical systems may require more frequ changes. The time between resets must be administrativ controlled	F	
Disable "remember me" functionality for password fields	F	
The last use (successful or unsuccessful) of a user account s be reported to the user at their next successful login	F	
Implement monitoring to identify attacks against multiple user accounts, utilizing the same password. This attack pattern is u to bypass standard lockouts, when user IDs can be harveste guessed	F	
Change all vendor-supplied default passwords and user IDs disable the associated accounts	F	
Re-authenticate users prior to performing critical operations	T	
Use Multi-Factor Authentication for highly sensitive or high valu transactional accounts	T	
If using third party code for authentication, inspect the cod carefully to ensure it is not affected by any malicious code	T	

Session Management

Use the server or framework's session management controls. The application should only recognize these session identifiers as valid	T	
Session identifier creation must always be done on a trus system (e.g., The server)	T	
Session management controls should use well vetted algorit that ensure sufficiently random session identifiers	F	
Set the domain and path for cookies containing authenticate session identifiers to an appropriately restricted value for the s	T	
Logout functionality should fully terminate the associated sess or connection	T	
Logout functionality should be available from all pages protected by authorization	F	
Establish a session inactivity timeout that is as short as poss based on balancing risk and business functional requirement most cases it should be no more than several hours	F	
Disallow persistent logins and enforce periodic session terminations, even when the session is active. Especially applications supporting rich network connections or connect to critical systems. Termination times should support busin requirements and the user should receive sufficient notification to mitigate negative impacts	T	
If a session was established before login, close that session establish a new session after a successful login	T	
Generate a new session identifier on any authentication	T	
Do not allow concurrent logins with the same user ID	T	
Do not expose session identifiers in URLs, error messages or Session identifiers should only be located in the HTTP co header. For example, do not pass session identifiers as parameters	T	
Protect server side session data from unauthorized access, other users of the server, by implementing appropriate acc controls on the server	T	
Generate a new session identifier and deactivate the old periodically. (This can mitigate certain session hijacking attacks where the original identifier was compromised)	F	
Generate a new session identifier if the connection secu changes from HTTP to HTTPS, as can occur during authentic Within an application, it is recommended to consistently util HTTPS rather than switching between HTTP to HTTPS.	T	
Supplement standard session management for sensitive ser	T	

side operations, like account management, by utilizing per-session strong random tokens or parameters. This method can be used to prevent Cross Site Request Forgery attacks		
Supplement standard session management for highly sensitive or critical operations by utilizing per-request, as opposed to per-session, strong random tokens or parameters	F	
Set the "secure" attribute for cookies transmitted over an TLS connection	T	
Set cookies with the HttpOnly attribute, unless you specifically require client-side scripts within your application to read or set a cookie's value	T	

Access Control

Use only trusted system objects, e.g. server side session objects, for making access authorization decisions	T	
Use a single site-wide component to check access authorization. This includes libraries that call external authorization services	T	
Access controls should fail securely	F	
Deny all access if the application cannot access its security configuration information	T	
Enforce authorization controls on every request, including those made by server side scripts, "includes" and requests from rich client-side technologies like AJAX and Flash	F	
Segregate privileged logic from other application code	F	
Restrict access to files or other resources, including those outside the application's direct control, to only authorized users	T	
Restrict access to protected URLs to only authorized users	T	
Restrict access to protected functions to only authorized users	T	
Restrict direct object references to only authorized users	F	
Restrict access to services to only authorized users	T	
Restrict access to application data to only authorized users	T	
Restrict access to user and data attributes and policy information used by access controls	T	
Restrict access security-relevant configuration information to only authorized users	T	
Server side implementation and presentation layer representations of access control rules must match	T	

If state data must be stored on the client, use encryption a integrity checking on the server side to catch state tampering.	F	
Enforce application logic flows to comply with business rules	F	
Limit the number of transactions a single user or device perform in a given period of time. The transactions/time sho be above the actual business requirement, but low enough deter automated attacks	F	
Use the "referer" header as a supplemental check only, it sho never be the sole authorization check, as it is can be spoofed	F	
If long authenticated sessions are allowed, periodically va a user's authorization to ensure that their privileges have not changed and if they have, log the user out and force them to authenticate	T	
Implement account auditing and enforce the disabling of unu accounts (e.g., After no more than 30 days from the expiratio an account's password.)	F	
The application must support disabling of accounts a terminating sessions when authorization ceases (e.g., Chang role, employment status, business process, etc.)	F	
Service accounts or accounts supporting connections to or f external systems should have the least privilege possible	F	
Create an Access Control Policy to document an applicati business rules, data types and access authorization criteria a processes so that access can be properly provisioned controlled. This includes identifying access requirements for both the data and system resources	F	

Cryptographic Practices

All cryptographic functions used to protect secrets from t application user must be implemented on a trusted system (e The server)	T	
Protect master secrets from unauthorized access	T	
Cryptographic modules should fail securely	F	
All random numbers, random file names, random GUIDs, random strings should be generated using the cryptograp module's approved random number generator when these random values are intended to be unguessable	F	
Cryptographic modules used by the application should compliant to FIPS 140-2 or an equivalent standard. (Se http://csrc.nist.gov/groups/STM/cmvp/validation.html)	F	
Establish and utilize a policy and process for how cryptographic	F	

keys will be managed		

Error Handling and Logging

Do not disclose sensitive information in error responses, includ system details, session identifiers or account information	T	
Use error handlers that do not display debugging or stack trace information	T	
Implement generic error messages and use custom error pages	F	
The application should handle application errors and not rely on the server configuration	T	
Properly free allocated memory when error conditions occur	F	
Error handling logic associated with security controls should deny access by default	T	
All logging controls should be implemented on a trusted system (e.g., The server)	T	
Logging controls should support both success and failure of specified security events	T	
Ensure logs contain important log event data	T	
Ensure log entries that include untrusted data will not execute as code in the intended log viewing interface or software	T	
Restrict access to logs to only authorized individuals	T	
Utilize a master routine for all logging operations	T	
Do not store sensitive information in logs, including unnecessary system details, session identifiers or passwords	T	
Ensure that a mechanism exists to conduct log analysis	T	
Log all input validation failures	T	
Log all authentication attempts, especially failures	T	
Log all access control failures	T	
Log all apparent tampering events, including unexpected changes to state data	F	
Log attempts to connect with invalid or expired session tokens	T	
Log all system exceptions	T	
Log all administrative functions, including changes to the security configuration settings	T	

Log all backend TLS connection failures	T	
Log cryptographic module failures	F	
Use a cryptographic hash function to validate log entry integrity	F	

Data Protection

Implement least privilege, restrict users to only the functionali data and system information that is required to perform the tasks	T	
Protect all cached or temporary copies of sensitive data stored the server from unauthorized access and purge those tempo working files a soon as they are no longer required.	T	
Encrypt highly sensitive stored information, like authenticati verification data, even on the server side. Always use well vet algorithms, see "Cryptographic Practices" for additional guidan	F	
Protect server-side source-code from being downloaded by a use	T	
Do not store passwords, connection strings or other sensi information in clear text or in any non-cryptographically secure manner on the client side. This includes embedding in insec formats like: MS viewstate, Adobe flash or compiled code	T	
Remove comments in user accessible production code that r reveal backend system or other sensitive information	T	
Remove unnecessary application and system documentation this can reveal useful information to attackers	T	
Do not include sensitive information in HTTP GET reque parameters	T	
Disable auto complete features on forms expected to cont sensitive information, including authentication	F	
Disable client side caching on pages containing sens information. Cache-Control: no-store, may be used in conjunction with the HTTP header control "Pragma: no-cache", which is less effective, but is HTTP/1.0 backward compatible	F	
The application should support the removal of sensitive d when that data is no longer required. (e.g. personal informat or certain financial data)	T	
Implement appropriate access controls for sensitive data sto on the server. This includes cached data, temporary files and that should be accessible only by specific system users	F	

Communication Security

Implement encryption for the transmission of all sensitiv	T	

information. This should include TLS for protecting the connec and may be supplemented by discrete encryption of sensitive or non-HTTP based connections		
TLS certificates should be valid and have the correct domain name, not be expired, and be installed with intermedia certificates when required	T	
Failed TLS connections should not fall back to an inse connection	T	
Utilize TLS connections for all content requiring authentica access and for all other sensitive information	T	
Utilize TLS for connections to external systems that invo sensitive information or functions	T	
Utilize a single standard TLS implementation that is configu appropriately	T	
Specify character encodings for all connections	F	
Filter parameters containing sensitive information from the HT referer, when linking to external sites	F	

System Configuration

Ensure servers, frameworks and system components are run the latest approved version	T	
Ensure servers, frameworks and system components have patches issued for the version in use	T	
Turn off directory listings	T	
Restrict the web server, process and service accounts to the privileges possible	T	
When exceptions occur, fail securely	T	
Remove all unnecessary functionality and files	T	
Remove test code or any functionality not intended f production, prior to deployment	T	
Prevent disclosure of your directory structure in the robots.txt by placing directories not intended for public indexing into an isolated parent directory. Then "Disallow" that entire pare directory in the robots.txt file rather than Disallowing ea individual directory	F	
Define which HTTP methods, Get or Post, the application support and whether it will be handled differently in different pages in the application	T	
Disable unnecessary HTTP methods, such as WebDAV exten	F	

If an extended HTTP method that supports file handling required, utilize a well-vetted authentication mechanism		
If the web server handles both HTTP 1.0 and 1.1, ensure that are configured in a similar manor or insure that you understa any difference that may exist (e.g. handling of extended H methods)	F	
Remove unnecessary information from HTTP respons headers related to the OS, web-server version and application framework	F	
The security configuration store for the application should be a to be output in human readable form to support auditing	T	
Implement an asset management system and regist system components and software in it	T	
Isolate development environments from the production netwo and provide access only to authorized development and t groups. Development environments are often configured l securely than production environments and attackers may us this difference to discover shared weaknesses or as an avenu exploitation	T	
Implement a software change control system to manage a record changes to the code both in development and productio	T	

Database Security

Use strongly typed parameterized queries	T	
Utilize input validation and output encoding and be sure address meta characters. If these fail, do not run the datab command	T	
Ensure that variables are strongly typed	T	
The application should use the lowest possible level of privil when accessing the database	T	
Use secure credentials for database access	T	
Connection strings should not be hard coded within t application. Connection strings should be stored in a sepa configuration file on a trusted system and they should encrypted.	T	
Use stored procedures to abstract data access and allow for removal of permissions to the base tables in the database	F	
Close the connection as soon as possible	T	
Remove or change all default database administrative passwo Utilize strong passwords/phrases or implement multi-factor authentication	F	

Turn off all unnecessary database functionality (e.g., unneces stored procedures or services, utility packages, install only minimum set of features and options required (surface area reduction))	F	
Remove unnecessary default vendor content (e.g., sam schemas)	F	
Disable any default accounts that are not required to supp business requirements	F	
The application should connect to the database with differe credentials for every trust distinction (e.g., user, read-only user, guest, administrators)	T	

File Management

Do not pass user supplied data directly to any dynamic incl function	T	
Require authentication before allowing a file to be uploaded	T	
Limit the type offiles that can be uploaded to only those typ that are needed for business purposes	T	
Validate uploaded files are the expected type by checking headers. Checking for file type by extension alone is not suffici	T	
Do not save files in the same web context as the application. File should either go to the content server or in the database.	T	
Prevent or restrict the uploading of any file that may interpreted by the web server.	T	
Turn off execution privileges on file upload directories	T	
Implement safe uploading in UNIX by mounting the targeted directory as a logical drive using the associated path or chrooted environment	F	
When referencing existing files, use a white list of allowed names and types. Validate the value of the parameter being passed and if it does not match one of the expected values, ei reject it or use a hard coded default file value for the conte instead	F	
Do not pass user supplied data into a dynamic redirect. If must be allowed, then the redirect should accept only validated relative path URLs	F	
Do not pass directory or file paths, use index values mappe pre-defined list of paths	T	
Never send the absolute file path to the client	T	
Ensure application files and resources are read-only	T	

Scan user uploaded files for viruses and malware	T	

Memory Management

Utilize input and output control for untrusted data	T	
Double check that the buffer is as large as specified	T	
When using functions that accept a number of bytes to copy, s as strncpy(), be aware that if the destination buffer size is eq to the source buffer size, it may not NULL-terminate the string	T	
Check buffer boundaries if calling the function in a loop and make sure there is no danger of writing past the allocated space	T	
Truncate all input strings to a reasonable length before pass them to the copy and concatenation functions	T	
Specifically close resources, don't rely on garbage collection. (e.g., connection objects, file handles, etc.)	T	
Use non-executable stacks when available	T	
Avoid the use of known vulnerable functions (e.g., printf, strc strcpy etc.)	T	
Properly free allocated memory upon the completion of functio and at all exit points	T	

General Coding Practices

Use tested and approved managed code rather than creating unmanaged code for common tasks	T	
Utilize task specific built-in APIs to conduct operating syste tasks. Do not allow the application to issue commands directly to the Operating System, especially through the use of applica initiated command shells	T	
Use checksums or hashes to verify the integrity of interpre code, libraries, executables, and configuration files	T	
Utilize locking to prevent multiple simultaneous requests or use synchronization mechanism to prevent race conditions	T	
Protect shared variables and resources from inappropri concurrent access	T	
Explicitly initialize all your variables and other data stores, eit during declaration or just before the first usage	T	
In cases where the application must run with elevated privileg raise privileges as late as possible, and drop them as soo possible	T	

Avoid calculation errors by understanding your programm language's underlying representation and how it interacts w numeric calculation. Pay close attention to byte s discrepancies, precision, signed/unsigned distinctions, truncat conversion and casting between types, "not a number" calculations, and how your language handles numbers that a too large or too small for its underlying representation	T	
Do not pass user supplied data to any dynamic execution func	T	
Restrict users from generating new code or altering existing co	T	
Review all secondary applications, third party code and librari to determine business necessity and validate safe functionality these can introduce new vulnerabilities	T	
Implement safe updating. If the application will utilize automa updates, then use cryptographic signatures for your code an ensure your download clients verify those signatures. U encrypted channels to transfer the code from the host server	T	