Generic Security Test Plan

➢ Security Plan Identifier:

Product Name	Example
Product Release Version	1.x
Project Name	Example
Link to PDS Location	
Author	Simon Heath
Creation Date	2023 Feb
Acceptance Date	2023 Feb

How to use this document:

- The following document only provides a list of recommended test cases; additional testing may be required depending on features or any other specific functional areas for any of the applications.
- Please use the "Additional Identified Test Cases" section to add additional test cases identified for any specific feature or functional area.
- There may be cases where all the identified categories or test cases are not applicable; you can mark the applicability accordingly for the application under test.

1. Authentication

Is the category applicable?	Yes	No
	☐	☒

All extensions are contained in Windchill. No extension manages or controls any authentication procedure. [Wincom]

Authentication is the process of verification that an individual, entity or website is who it claims to be. Authentication in the context of web applications is commonly performed by submitting a user name or ID and one or more items of private information that only a given user should have. It is a process in which the credentials provided are compared to those on file in a database of authorized users' information on a local operating system or within an authentication server.

a. Testing for Credentials Transported over and Encrypted Channel

Is the sub-category applicable?	Yes	No
	☐	☒

See above

Guidelines	Verified?		
	Yes	No	N.A
Test and ensure credentials are sent using an encrypted channel	☐	☐	☐
Verify that the credentials are not readable by a malicious user using a sniffer or any other intercepting tool	☐	☐	☐
Verify GET method is not used for transmitting credentials or any other sensitive information. This will ensure that the data is not displayed in clear text in the URL	☐	☐	☐

b. Testing for Weak password policy

Is the sub-category applicable?	Yes	No
	☐	☒

Define a strong password policy with the following considerations and verify it-

Guidelines	Verified?		
	Yes	No	N.A
Length	☐	☐	☐
Complexity (Inclusion of lower/uppercase letters, special characters and numeric values)	☐	☐	☐
Reuse of same passwords again for the same user should not be allowed	☐	☐	☐
Aging requirements of passwords	☐	☐	☐
Verify what information is required to reset the password	☐	☐	☐
Verify how are reset passwords communicated to the user	☐	☐	☐

Verify if reset passwords are generated randomly	☐	☐	☐
Verify if the reset password functionality requesting confirmation before changing the password	☐	☐	☐
Verify if the old password requested to complete the change	☐	☐	☐

Authentication cheat sheet can be found here

c. Additional identified Test Cases

2. Authorization

Is the category applicable?	Yes	No
	☒	☐

Authorization ensures that the authenticated user has the appropriate privileges to access resources. The resources a user has access to depend on his/her role. The process of an administrator granting rights and the process of checking user account permissions for access to resources are both referred to as authorization. The privileges and preferences granted for the authorized account depend on the user's permissions, which are either stored locally or on the authentication server. The settings defined for all these environment variables are set by an administrator.

a. Testing for Bypassing Authorization Schema

Is the sub-category applicable?	Yes	No
	☒	☐

Guidelines	Verified?		
	Yes	No	N.A
Verify that access controlled resources are not accessible if the user in not authenticated	☒	☐	☐
Verify that access controlled resources are not accessible after a valid logout in the same session	☐	☐	☒

b. Testing for Privilege escalation

Is the sub-category applicable?	Yes	No
	☒	☐

Guidelines	Verified		
	Yes	No	N.A
In every portion of the application where an authenticated user can create information in the database (e.g. making a payment, adding a contact, or sending a message), can receive information (statement of account, order details, etc.), or delete information (drop users, messages, etc.), it is necessary to record that functionality. The tester should try to access such functions as another user in order to verify if it is possible to access a function that should not be permitted by the user's role/privilege (but might be permitted as another user). • Test scenarios for all applicable user roles	☒	☐	☐

• Test scenarios for different users in same role			
Verify that it is not possible to escalate privileges by modifying the parameter values.	☐	☐	☒
Verify administrator functions and access controlled resources are not accessible to other valid logged in non-administrative users.	☐	☐	☒

Access Control cheat sheet can be found here

c. Additional identified Test Cases

3. Data Validation

Is the category applicable?	Yes	No
	☐	☒

a. SQL Injection:

Is the sub-category applicable?	Yes	No
	☐	☒

Only Windchill APIs are used, the code is security scanned to prevent any direct access to the database. [Wincom]

A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to effect the execution of predefined SQL commands.

Guidelines	Verified?		
	Yes	No	N.A
The first step in this test is to understand when the application interacts with a DB Server in order to access some data. Typical examples of cases when an application needs to talk to a DB include: • **Authentication forms:** when authentication is performed using a web form, chances are that the user credentials are checked against a database that contains all usernames and passwords (or, better, password hashes). • **Search engines:** the string submitted by the user could be used in a SQL query that extracts all relevant records from a database. The tester has to make a list of all input fields whose values could be used in crafting a SQL query, including the hidden fields of POST requests and then test them separately, trying to interfere with the query and to generate an error. Consider also HTTP headers and Cookies. The very first test usually consists of adding a single quote (') or a semicolon (;) to the field or parameter under test. The first is used in SQL as a string terminator and, if not filtered by the application, would lead to an incorrect query. The second is used	☐	☐	☐

to end a SQL statement and, if it is not filtered, it is also likely to generate an error.			
• You can manually test for SQL injection using these entry points. Please refer : Testing for SQL Injection • You can use SQL Inject Me for running automated tests for those entry points.	☐	☐	☐

SQL injection prevention cheat sheet can be found here

b. Cross Site Request Forgery

Is the sub-category applicable?	Yes	No
	☒	☐

Note: No iframes in the code in addition the security of the pages and configuration of the server is managed by Windchill. [Wincom]

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged request. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

Guidelines	Verified?		
	Yes	No	N.A
Audit the application to ascertain if its session management is vulnerable. If session management relies only on client side values (information available to the browser), then the application is vulnerable. "Client side values" mean cookies and HTTP authentication credentials (Basic Authentication and other forms of HTTP authentication; not form-based authentication, which is an application-level authentication). **For an application to not be vulnerable, it must include session-related information in the URL, in a form of unidentifiable or unpredictable by the user** (uses the term secret to refer to this piece of information).	☐	☐	☒
Verify SAME ORIGIN policy is in place and X-Frame options shows SAME ORIGIN	☐	☐	☒
When a Web application formulates a request (by generating a link or form that causes a request when submitted or clicked by the user), the application should include a hidden input parameter with a common name such as "CSRFToken". The value of this token must be randomly generated such that it cannot be guessed by an attacker. Consider leveraging the java.security.SecureRandom class for Java applications to generate a sufficiently long random token.	☒	☐	☐
If you are using Synchronizer Token or NONCE for protection are user friendly messages displayed after modifying the nonce and action is not executed?	☒	☐	☐
If you are using Synchronizer Token or NONCE for protection are user friendly messages displayed after removing the nonce and action is not executed?	☒	☐	☐
If you are using Synchronizer Token or NONCE for protection verify the masked synchronizer token changes value in each server response	☒	☐	☐

If you are using Synchronizer Token or NONCE for protection verify expired synchronizer token gets rejected during active valid session	☒	☐	☐
If you are using Synchronizer Token or NONCE for protection verify valid synchronizer token gets accepted after session expires	☒	☐	☐

CSRF prevention cheat sheet can be found here

c. Additional identified Test Cases

4. Secure File Handling

Is the category applicable?	Yes	No
	☐	☒

Server file system access must be done securely to prevent users from manipulating the application into allowing them unauthorized access to the server file system which could lead to inserting malicious data and corrupting the system.

Guidelines	Verified?		
	Yes	No	N.A
Check that users not have the capability to specify server file path(s).	☐	☐	☐
Check that untrusted data that is used to construct the file path is validated to ensure that it is in the correct format and that it points to a valid location	☐	☐	☐
Filename(s) obtained from users or constructed from untrusted data are to be validated for correct format and/or expected values	☐	☐	☐
Check that the files (including temporary files) are uploaded to or created in locations outside the web root.	☐	☐	☐
Verify that the temporary files are created with unique and unpredictable names.	☐	☐	☐
Verify that the temporary files are deleted when processing is complete.	☐	☐	☐

a. Additional identified Test Cases

5. Session Management

Is the category applicable?	Yes	No
	☐	☒

Note: Contained completely in Windchill – all session management managed by Windchill [Wincom]

One of the core components of any web-based application is the mechanism by which it controls and maintains the state for a user interacting with it. This is referred to this as Session Management and is defined as the set of all controls governing state-full interaction between a user and the web-based application. This broadly covers anything from how user authentication is performed, to what happens upon them logging out.

HTTP is a stateless protocol, meaning that web servers respond to client requests without linking them to each other. Even simple application logic requires a user's multiple requests to be associated with each other across a "session". This necessitates third party solutions – through either Off-The-Shelf (OTS) middleware and web server solutions, or bespoke developer implementations. Most popular web application environments, such as ASP and PHP, provide developers with built-in session handling routines. Some kind of identification token will typically be issued, which will be referred to as a "Session ID" or Cookie.

a. Testing for Session Management Schema

Is the sub-category applicable?	Yes	No
	☐	☒

In analysing Session ID sequences, patterns or cycles, static elements and client dependencies should all be considered as possible contributing elements to the structure and function of the application.

Guidelines	Verified?		
	Yes	No	N.A
Verify that the Session IDs generated are random in nature	☐	☐	☐
Verify the same input conditions doesn't produce the same ID on a subsequent run	☐	☐	☐
Verify the token length of the Session ID	☐	☐	☐
Verify the cookie configuration for session ID, it should be secure (set only on HTTPS channel) or HTTPOnly (not readable by a script)	☐	☐	☐
For a authenticated user verify that application doesn't support multiple sessions (won't be applicable for all applications)	☐	☐	☐

The following areas should be addressed during the single and multiple Session ID structure testing:

Guidelines	Verified?		
	Yes	No	N.A
Check what parts of the Session ID are static	☐	☐	☐
Check what clear-text confidential information is stored in the Session ID	☐	☐	☐
Check what information can be deduced from the structure of the Session ID	☐	☐	☐
Check what portions of Session ID are static for the same log in condition	☐	☐	☐
Verify all Set-Cookie directives are tagged as Secure	☐	☐	☐
Verify Cookie operations take place over encrypted transport	☐	☐	☐
Verify what Expires= times are used while using persistent cookies, check if they are reasonable	☐	☐	☐

b. Testing for Session Timeout and Logout

Is the sub-category applicable?	Yes	No
	☐	☒

Guidelines	Verified?		
	Yes	No	N.A
Verify that the log out function effectively destroys all session token, or at least renders them unusable	☐	☐	☐
Verify the server performs proper checks on the session state, disallowing an attacker to replay previously destroyed session ID's	☐	☐	☐
Verify the timeout is properly enforced by the server	☐	☐	☐

Session Management cheat sheet can be found here

c. Additional identified Test Cases

6. Configuration Management

Is the category applicable?	Yes	No
	☐	☒

Note: N.A. because contained completely in Windchill [Wincom]

Understanding the deployed configuration of the server hosting the web application is almost as important as the application security testing itself. After all, an application chain is only as strong as its weakest link. Application platforms are wide and varied, but some key platform configuration errors can compromise the application in the same way an unsecured application can compromise the server

a. Test HTTP Methods

Guidelines	Verified?		
	Yes	No	N.A
Discover and validate the supported HTTP Methods for the application- To perform this test; the tester needs some way to figure out which HTTP methods are supported by the web server that is being examined. The OPTIONS HTTP method provides the tester with the most direct and effective way to do that. The OPTIONS method represents a request for information about the communication options available on the request/response chain identified by the Request-URI.	☐	☐	☐
The HTTP Strict Transport Security (HSTS) feature lets a web application to inform the browser, through the use of a special response header, that it should never establish a connection to the specified domain servers using HTTP. Instead it should automatically establish all connection requests to access the site through HTTPS. The HTTP strict transport security header uses two directives: • **max-age:** to indicate the number of seconds that the browser should automatically convert all HTTP requests to HTTPS. • **includeSubDomains:** to indicate that all web application's sub-domains must use HTTPS.	☐	☐	☐
Testing for the presence of HSTS header can be done by checking for the existence of the HSTS header in the server's response in an interception proxy.	☐	☐	☐
Verify all client and server communication is using HTTPS secure protocol for communication	☐	☐	☐

b. Additional identified Test Cases

7. Error Handling and Logging

Is the category applicable?	Yes	No
	☒	☐

Standard users have no access to log. Site administrators can access the logs for diagnostics purposes. Identical info is also available through standard Windchill interface. [Wincom]

An important aspect of secure application development is to prevent information leakage. Error messages give an attacker great insight into the inner workings of an application. The purpose of reviewing the Error Handling code is to assure the application fails safely under all possible error conditions, expected and unexpected. No sensitive information is presented to the user when an error occurs.

Guidelines	Verified?		
	Yes	No	N.A
Any error message should not provide any information about web server version, OS, modules and other products used.	☒	☐	☐
Excessive access attempts to non-existent files?	☐	☐	☒
Web service stopped/started/failed message?	☐	☐	☒
Access to “risky” pages that accept user input?	☐	☐	☒
Application exception handling minimizes the information disclosure in case of an exception?	☒	☐	☐
HTTP response codes such as 400 Bad request, 405 Method Not Allowed, 501 Method Not Implemented, 408 Request Time-out and 505 HTTP Version Not supported can be forced by an attacker. Verify when receiving specially crafted inputs, web servers may provide one of these error codes depending on their HTTP implementation.	☐	☐	☒
For application errors (error messages from framework code like ASP, JSP) make sure they are not providing information of server paths, installed libraries and application versions.	☒	☐	☐
For DB errors (MySQL, Oracle or MSSQL) make sure they are not providing sensible information such as DB server IP, tables, columns and login details. **Note: No direct access to DB, only through Windchill API [Wincom]**	☐	☐	☒
Verify application errors not contain stack traces which provide sensible information. There are a variety of techniques that will cause exception messages. Some tests to try include: • Invalid input (such as input that is not consistent with application logic). • Input that contains non alphanumeric characters or query syntax. • Empty inputs. • Inputs that are too long. • Access to internal pages without authentication. • Bypassing application flow.	☒	☐	☐

*** Verified where applicable [Wincom]**			
Verify these best practices for logs: • Use <cflog> for customized logging • Incorporate into custom error handling • Record application specific messages • Actively monitor and fix errors • Optimize logging settings • Rotate log files to keep them current • Keep files size manageable • Enable logging of slow running pages • Set the time interval lower than the configured Timeout Request value in the CFAM Settings screen • Long running page timings are recorded in the server.log • Use <cftrace> sparingly for audit trails • Use with inline="false" • Use it to track user input – Form and/or URL variables *** Verified where applicable [Wincom]**	☒	☐	☐

a. Additional identified Test Cases

8. Output Validation

Is the category applicable?	Yes	No
	☒	☐

Note: No output produced [Wincom]

a. Stored XSS

Is the sub-category applicable?	Yes	No
	☒	☐

Stored attacks are those where the injected script is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, etc. The victim then retrieves the malicious script from the server when it requests the stored information. Stored XSS is also sometimes referred to as Persistent or Type-I XSS.

Guidelines	Verified?		
	Yes	No	N.A
Detect input vectors: For each web page, the tester must determine all the web application's user-defined variables and how to input them. This includes hidden or non-obvious inputs such as HTTP parameters, POST data, hidden form field values, and predefined radio or selection values. Typically in-browser HTML editors or web proxies are used to view these hidden variables.	☒	☐	☐
Analyse each input vector to detect potential vulnerabilities. To detect XSS vulnerability, the tester will typically use specially crafted input data with each input vector. Such input data is typically harmless, but trigger responses from the web browser that manifests the vulnerability.	☒	☐	☐
For each test input attempted in the previous phase, the tester will analyse the result and determine if it represents a vulnerability that has a realistic impact on the web application's security. This requires examining the resulting web page HTML and searching for the test input. Once found, the tester identifies any special characters that were not properly encoded, replaced, or filtered out. The set of vulnerable unfiltered special characters will depend on the context of that section of HTML.	☒	☐	☐
• You can manually test for reflected XSS for these input fields. Please refer: Testing XSS • You can use XSS Me for running automated tests for the input fields.	☒	☐	☐

b. Reflected XSS

Is the sub-category applicable?	Yes	No
	☒	☐

Reflected attacks are those where the injected script is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request. Reflected attacks are delivered to victims via another route, such as in an e-mail message, or on some other web site. When a user is tricked into clicking on a malicious link, submitting a specially crafted form, or even just browsing to a malicious site, the injected code travels to the vulnerable web site, which reflects the attack back to the user's browser. The browser then executes the code because it came from a "trusted" server. Reflected XSS is also sometimes referred to as Non-Persistent or Type-II XSS.

Guidelines	Verified?		
	Yes	No	N.A
Input Forms: The first step is to identify all points where user input is stored into the back-end and then displayed by the application. Typical examples of stored user input can be found in: • **User/Profiles page:** the application allows the user to edit/change profile details such as first name, last name, nickname, avatar, picture, address, etc. • **Shopping cart like functionality :** the application allows the user to store items into the shopping cart which can then be reviewed later • **File Manager:** application that allows upload of files • **Application settings/preferences:** application that allows the user to set preferences • **Forum/Message board:** application that permits exchange of posts among users • **Blog:** if the blog application permits to users submitting comments • **Log:** if the application stores some users input into logs.	☒	☐	☐
Testing for stored XSS: This involves testing the input validation and filtering controls of the application. Basic injection examples in this case: • aaa@aa.com"><script>alert(document.cookie)</script> • aaa@aa.com%22%3E%3Cscript%3Ealert(document.cookie)%3C%2Fscript%3E	☒	☐	☐
Ensure the input is submitted through the application. This normally involves disabling JavaScript if client-side security controls are implemented or modifying the HTTP request with a web proxy.	☒	☐	☐

XSS prevention cheat sheet can be found here

c. Additional identified Test Cases

9. Encryption

Is the category applicable?	Yes	No
	☐	☒

Note: This extension has been scanned for any potential data that may need to be encryption.

Such as password, usernames etc. This extension does not require the encryption of any data.

[Wincom]

Encryption is the process of encoding messages or information in such a way that only authorized parties can read it. Encryption does not of itself prevent interception, but denies the message content to the interceptor. In an encryption scheme, the intended communication information or message, referred to as plaintext, is encrypted using an encryption algorithm, generating cipher text that can only be read if decrypted. For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm. It is in principle possible to decrypt the message without possessing the key, but, for a well-designed encryption scheme, large computational resources and skill are required. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients, but not to unauthorized interceptors

Guidelines	Applicable?		
	Yes	No	N.A
Check if data which should be encrypted is encrypted	☐	☐	☐
Check for wrong algorithms usage depending on context	☐	☐	☐
Check for weak algorithms usage	☐	☐	☐
Check for certificate revocation. If a certificate is used without first checking to ensure it was not revoked, the certificate may be compromised.	☐	☐	☐
Check if key exchange is happening without entity authentication or not. Performing a key exchange without verifying the identity of the entity being communicated with will preserve the integrity of the information sent between the two entities; this will not, however, guarantee the identity of end entity.	☐	☐	☐
Check the key length used for encryption is as per recommended guidelines	☐	☐	☐
Check for impacts if a customer key is leaked	☐	☐	☐

Cryptographic storage cheat sheet can be found here

a. Additional identified Test Cases

➢ Appendix

Inputs	Link
References	OWASP Top Ten Most Critical Web Application Security Risks Rdwiki: Application Security Application Security SharePoint