ID:	**AP-SEC-001**
Title:	**Web application security**
Domain:	**Application**
Discipline:	**Development Management**
Date updated: Revision no.: Original date:	**9/27/2019** **3** **4/29/2010**

I. Authority, Applicability and Purpose

A. **Authority:** Title 29 Chapter 90C Delaware Code, §9004C – General Powers, duties and functions of DTI "2) Create, implement and enforce statewide and agency technology solutions, policies, standards and guidelines, including as recommended by the Technology Investment Council on an ongoing basis and the CIO"

B. **Applicability:** Applies to all State of Delaware communications and computing resources. DTI is an Executive Branch Agency and has no authority over the customers in Legislative and Judicial Branches, as well as School Districts, and other Federal and Local Government entities that use these resources. However, all users, including these entities, must agree to abide by all policies, standards promulgated by DTI as a condition of funding and continued use of these resources.

C. **Purpose:** Currently, the State uses a variety of techniques for securing their web applications and this standard will document the required activities from the perspective of a developer.

II. Scope

A. **Audience**: This document is intended for Application Developers, their managers and application development contractors for the State, Systems Administrators, Network Administrators, and Computer Auditors. This document is not intended for use by non-IT personnel.

B. **Applicability:** This standard addresses all web application developed for use by the State of Delaware, including applications owned by the State but developed by third-party contractors.

C. **Environments:** This standard applies to all environments that must be protected due to their data classifications.

III. Process

A. **Adoption:** These standards have been adopted by the Department of Technology and Information (DTI) through the Technology and Architecture Standards Committee (TASC) and are applicable to all Information Technology use throughout the State of Delaware.

B. **Revision:** Technology is constantly evolving; therefore, the standards will need to be regularly reviewed. It is the intent of TASC to review each standard annually. TASC is open to suggestions and comments from knowledgeable individuals within the State, although we ask that they be channeled through your Information Resource Manager (IRM).

C. **Contractors:** Contractors or other third parties are required to comply with these standards when proposing technology solutions to DTI or other State entities. Failure to do so could result in rejection by the Delaware Technology Investment Council. For further guidance, or to seek review of a component that is not rated below, contact the TASC at dti_tasc@state.de.us.

D. **Implementation responsibility:** DTI and/or the organization's technical staff will implement this standard during the course of normal business activities, including business case review, architectural review, project execution and the design, development, or support of systems.

E. **Enforcement:** DTI will enforce this standard during the course of normal business activities, including business case and architectural review of proposed projects and during the design, development, or support of systems. This standard may also be enforced by others during the course of their normal business activities, including audits and design reviews.

F. **Contact us:** Any questions or comments should be directed to dti_tasc@state.de.us.

IV. Definitions/Declarations

A. Declarations

Web application security must:

1. Applications should permit any role to be assigned to more than one user account. For example, the role of 'Administrator' must be grantable to any user rather than tied to a generic 'admin' account.

B. Definitions

1. Content Category - A "type" of information, for example a physical or mailing address, phone number or email address. This is separate from the classification, in that two or more categories could have the same data classification, but the addition of the second content category would still trigger a review by the agency's ISO.

2. Hash - An algorithm that generates a fixed-length "hash value" (or simply "hash") from a variable-length input, such that a small change in the input results in a large and unpredictable change in the output. Hash functions are one-way, meaning that one cannot determine the input simply by looking at the output; the only method to reproduce a hash is to run the hash function using every possible input, until the same hash is generated. Hash functions are used to store passwords such a way that even if an attacker could see the hash, they would not be able to use that hash to get the original password. Hash functions can also be used to verify file integrity: e.g. a file can be hashed prior to storage, and again upon retrieval, and the hashes compared. If they match, then the file has not been changed (or corrupted) in storage.

3. Key Space – The set of all possible keys that can be generated. For a 128-bit key, the key space is ideally 2^{128}. However, if the key is generated using only alphanumeric characters, the key space may be reduced dramatically. For alphanumeric characters represented by 8 bits per character, the key space would only be approximately 96 bits (16 characters, each chosen from a pool of 62 possibilities = 62^{16}, which is a 96-bit number). Adding symbols increases the pool to 95^{16}, which is a 106-bit number. A completely random 128-bit key would be 7,731,464 times larger than the 106-bit key, and 7,137,932,110 times larger than the 96-bit key.

4. Salt - A random value that is added to a password before hashing, which further obscures the password, making common attacks more time consuming, and therefore more difficult.

V. Definitions of Ratings

Individual components within a Standard will be rated in one of the following categories.

COMPONENT RATING	USAGE NOTES
STANDARD – DTI offers internal support and/or has arranged for external vendor support as well (where applicable). DTI believes the component is robust and can be expected to enjoy a useful life of 3+ years from the Effective Date.	These components can be used without explicit DTI approval for both **new projects** and **enhancement** of existing systems.
DECLINING – Deprecated - DTI considers the component to be a likely candidate to have support discontinued in the near future. A deprecated element is one becoming invalid or obsolete.	Via the State's waiver process, these components must be explicitly approved by DTI for **all projects**. They must not be used for **minor enhancement** and **system maintenance** without explicit DTI approval via the State's waiver process.
DISALLOWED – DTI declares the component to be unacceptable for use and will actively intervene to disallow its use when discovered.	No waiver requests for new solutions with this component rating will be considered.

A. **Missing Components** – No conclusions should be inferred if a specific component is not listed. Instead, contact TASC to obtain further information.

VI. Component Assessments

Web Application Security **Pre-Development / Project Initiation Phase**	
Security controls must be designed into the application during initiation phase. This must be addressed prior to commencing development activities in the following ways: data classification, privacy impact, regulatory compliance, content approval and developer training	
Data Classification	Development project sponsor shall classify the types of data to be processed by the application prior to commencing development. Project sponsor shall utilize the State's Data Classification Policy Furthermore, this classification shall include a privacy impact review and determination if the application's data is subject to regulatory requirements.

	This data classification process will drive, at a minimum, encryption requirements.
Content Approval	For publicly accessible web sites, the content category presented to the public by the application must be reviewed and approved by the agency ISO. Content approvals could be based on categories of information.
Developer Training	It is recommended that developers of internet facing web applications or internal applications that process State of Delaware Confidential Information attend web application security training from organizations like SysAdmin, Audit, Network Security Institute (SANS).
Web Application Security Standards & Principles	
Software coding standards include considerations for development of secure software, features within the application to enhance security, and encryption controls.	
Input Validation Controls	Lack of input validation is often an enabler for numerous attack vectors, such as a SQL injection. To provide appropriate input validation controls, developers will ensure: • Use stored procedures or parameterized/prepared statements when performing database access. Under no circumstances should dynamically-generated/ad-hoc SQL statements be executed. • Any input controlled by the user shall not affect the meaning and structure of SQL queries, which utilize their values. • In order to prevent cross-site request forgery (CSRF), all requests that modify data must include the user's session id along with a challenge token. The token should be unique per request. • It is strongly recommended that developers follow recommendations by the Open Web Application Security Project (OWASP).[1]
Output Validation Controls	All output to users will be sanitized, including the content of server generated error messages. Improperly sanitized output can lead to Cross-Site Scripting (XSS) vulnerabilities. Developers will ensure that any output derived from user data is HTML encoded prior to sending to clients. Developers are strongly encouraged to HTML-encode any information that is displayed from a database, whether user-entered or not.
Cookie Management	Cookies must meet the following criteria: • HTTPOnly flag set to protect cookie from cross-site scripting attacks. • Secure flag set whenever HTTPS is used to ensure cookies are only transmitted over the secure channel. • Use the Fully Qualified Domain Name (FQDN) for domain scope (e.g., xyz.delaware.gov instead of delaware.gov). • Do not contain sensitive data.
Session Management Controls	Web applications operate on a stateless protocol; therefore the application or web server maintains session management controls. Developers should attempt to defer session management controls to the underlying framework/container. Cookies are most often used to maintain session state. In addition to the cookie settings required in the Cookie Management section, the following additional criteria must be met for cookies used for

[1] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

	session management/authentication/authorization: • Non-persistent, to ensure cookies not stored on the client*. • Randomly generated, not easily guessable (e.g., pseudo-random generation). • Generated each time a user logs in. • Destroyed and reset to have no permissions when a user logs off. • Become invalid after a reasonable period of inactivity. • To prevent session hijacking, destroy a user's session immediately or create a new session ID after authentication, and create a new session for the newly-authenticated user. Developers will ensure that the logoff button used to destroy a session is clearly visible to the user. * One exception to this is when a cookie is used to “remember” the user’s PC and require different levels of authentication if the PC is “known”. For example a bank may ask for username and password only if the PC is “known” but the bank will ask two secret questions in addition to the username and password if the PC is not known.
Authorization and Authentication	Any web application with information classified as State of Delaware Confidential or higher must validate authorization for every page requested by a user. The application should check access control permissions for every page or file classified as State of Delaware confidential or higher prior to loading. Users should be restricted to the information and functionality based on their role. Additionally, any user interface elements that the user is not authorized to use should not be sent to the user’s browser. For example, if an application has a reporting function that the current user is not authorized to use, the navigation link for reports should not appear for that user, even in such a way that it’s hidden from view.
Auditing and Logging Capabilities	Web applications must, at a minimum, log the following events: • Successful and failed authentication attempts • Authorization and access failures • Application errors • Account lockouts Log entries should contain the requestor’s source IP address, timestamp, requestor’s username, URL requested and parameters sent to the application.
Data Security / Encryption	It is *recommended* that developers use encryption for application configuration information data at rest for data as defined in the System Architecture standard. Other regulatory compliance requirements may require encryption of data at rest. Framework or component encryption libraries such as Advanced Encryption Standard (AES) should be utilized – homegrown encryption functions must not be used.
Password Standards Enforcement	Applications must enforce Strong Password Standard requirements.
Password Management	Applications supporting user authentication are required to permit the user to change their own passwords. Users must be required to enter their old password prior to changing. The application will require the user to change

	their password upon first logon. Applications using LDAP or Active Directory are exempt from this requirement, provided the user is able to change their passwords via their desktop computer or a separate account management application.
Account Lockouts	Applications supporting user authentication will enforce account locking in accordance with Delaware Information Security Policy Attempt to mitigate automated account lockouts via mechanisms such as CAPTCHAs.
Administrator Interface, Credential Management	Administrative interfaces will not display user passwords on administrator browser session when managing a user's account.
Forgotten Passwords and Password Resets	Functionality to permit users to reset their passwords in the event of a forgotten password will meet the following criteria: • Use of a secret question and answer. When specifying secret question, user should have a choice and should use questions with a large variety of possible answers such as "Who is your favorite actor, musician, or artist?" rather than "What is your favorite color?" • Old passwords will not be retrievable. • After 5 incorrect answers to the secret question, the user account will be locked and require an administrator to unlock their account. • Log password resets After unlocking an account, user should be forced to change their password. An email notification stating that their account has been reset should be sent to their pre-established email account. Applications using LDAP or Active Directory are exempt from this requirement, provided the user is able to change their passwords via their desktop computer or a separate account management application.
User Self Registration	Functionality to permit users to self-provision access to a web application should meet the following criteria: • Allow the collection of pertinent user details to identify the user as a unique user within the application. • Utilize word or image verification boxes on request submission form to prevent automated attacks (CAPTCHA). • Provide a method for a user to validate that they registered for access to the web application (i.e. email validation) prior to providing access to the web application. • Provide a method for provisioning access to the user once a successful self registration request is received. • Self registration without validation should only be used for web application resources classified as public.
Login Auto-Complete	User authentication forms must have the auto-complete turned off within the applications "input" tag.
Password Hashing	All passwords stored by the application will use one of the following hashing algorithms in order of precedence. Higher-ranked algorithms must be used if available in your language or framework: • SHA-512 • SHA-384 • SHA-256 • SHA224 • SHA-1

	Passwords will not be stored in clear text or using reversible encryption. Password hashes must be salted with a minimum of 128 bits of randomly-generated data, with the salt generated and stored on a per-user basis (a full 128-bit key space). Passwords must be hashed and salted for a minimum of 1000 iterations (i.e. hash the password, add the salt to the hash, and hash the result; repeat 1000 times, adding the original salt to the result before the next iteration).
Information Disclosure - Failed Logon Attempts	For applications that support user logons, the application will not provide information regarding the validity of a username or password during a failed logon. For example, the error message should state “Invalid Username or Password”, instead of “Invalid Password” or “Invalid User”.
Information Disclosure - Application Errors	Applications often give too much information in error pages. The application will use custom error messages. Details pertaining to internal application information such as database errors will not be displayed to the user.
Production data in all Environments	The State’s System Environment standard provides guidance.
Access to OS Commands	Web applications should limit interaction with underlying operating system calls. Applications will use standard system or application interfaces/components. User input should not be passed as parameters to these functions.
File Upload / Download Features	Application logic should restrict use of file upload and download capabilities. If this functionality is required, uploaded files should reside within domains whose boundaries and privileges are clearly defined and restricted. It is strongly recommended that any file being uploaded or downloaded be stored in a data base.
Buffer Management	Developers should avoid using languages that require manual memory management for web applications; instead, developers are encouraged to use languages with built-in memory management (e.g., Java, .Net). Otherwise, input should be validated and truncated to appropriate lengths to avoid buffer overflow vulnerabilities.
Patching	Developers will ensure their Integrated Development Environment (IDE) components and interpreters are fully patched.
Hard-Coded Passwords	Developers should avoid using hard coded passwords, especially when used in clear-text within a file.
Unused Code	Unused code and test scripts will be removed from deployment packages. Unused code and test scripts will not reside on the production platform.
Administrator Interfaces	Administrative interfaces are only accessible from within the State network or VPN.
GET Requests	Parameters passed in GET requests are both cached by the browser, and saved in web server log files, possibly enabling an unauthorized user to discover sensitive information. Therefore, sensitive information (e.g., session management variables, password, etc.) will not be used in GET requests, regardless of whether encryption is used. Data modification should never be performed as a result of a GET request. Use a POST or other verb for data modification.
Caching	All application pages containing sensitive information will utilize HTTP headers or HTML meta tags as a directive to the browser or network appliances to prevent them from caching data. Please consider that not all

	caching devices respect HTML meta tags.
Hidden fields	Parameters in hidden fields are cached by the browser, which makes it easier for unauthorized users to discover sensitive information. Therefore, developers will avoid including sensitive information in any hidden fields.
Web Application Security Pre-deployment Activities	
Pre-deployment Testing	The web applications must undergo web application security testing prior to being placed into production and post production for major code releases. Additionally, source code review is also strongly encouraged.

References:

- SANS Top 25 Most Dangerous Programming Errors. http://cwe.mitre.org/top25/#CWE-20
- OWASP, with specific emphasis on the Top 10 and Web Application Security Standards (WASS) projects: http://www.owasp.org