Nuclear Regulatory Commission
Computer Security Office
Computer Security Standard

Office Instruction:	**CSO-STD-1108**
Office Instruction Title:	**Web Application Standard**
Revision Number:	**1.0**
Effective Date:	**December 1, 2012**
Primary Contacts:	**Kathy Lyons-Burke, SITSO**
Responsible Organization:	**CSO/PST**
Summary of Changes:	CSO-STD-1108, "Web Application Standard," provides descriptions and protection methods for major web application vulnerabilities that must not be present for web applications in the NRC production environment.
Training:	As requested
ADAMS Accession No.:	ML12185A187

Approvals				
Primary Office Owner	Policies, Standards, and Training		**Signature**	**Date**
Standards Working Group Chair	Bill Dabbs			
Responsible SITSO	Kathy Lyons-Burke		**/RA/**	**8/8/2012**
CSO Standards DAA	CISO	Thorne Graham (Acting)	**/RA/**	**8/8/2012**
	DeputyDirector, OIS	Mary Givvines (Acting)	**/RA/**	**8/13/2012**

TABLE OF CONTENTS

Computer Security Standard
CSO-STD-1108

Web Application Standard

---

1 PURPOSE

All NRC web applications in production within the NRC must be protected against the vulnerabilities identified in this web application standard and must meet all federally mandated and NRC-required security requirements. This standard includes but is not restricted to web applications that are Commercial off-the-shelf (COTS), Government off-the-shelf (GOTS), custom developed and/or any combination thereof.

This standard is intended to be used by Information System Security Officers (ISSOs) to secure their web applications, and developers to ensure that web applications are developed with a focus on application security and design. Furthermore, the standard is intended to help ISSOs and other NRC stakeholders select and procure secure web applications.

The vulnerabilities described in this standard include:

- Injection Flaws
- Cross-Site Scripting (XSS)
- Broken Authentication and Session Management
- Insecure Direct Object References
- Cross-Site Request Forgery (CSRF)
- Security Misconfiguration
- Insecure Cryptographic Storage
- Failure to Restrict Universal Resource Locator (URL) Access
- Insufficient Transport Layer Protection
- Unvalidated Redirects and Forwards

The standard describes each of the vulnerabilities, identifies risks posed by the vulnerabilities, and provides strategies for preventing and mitigating the vulnerabilities.

2 GENERAL REQUIREMENTS

At a minimum, web applications in production within the NRC that satisfy the criteria below must comply with this standard:

- COTS, GOTS, custom developed, and/or any combination thereof, and
- Owned, managed, and/or operated by NRC or other parties on behalf of NRC.

2.1 Compliance Assessments

The Computer Security Office (CSO) performs web application assessments on NRC web applications in accordance with this standard. Web applications will be assessed against the most recent effective version of the Open Web Application Security Project (OWASP) Top Ten list of web application security vulnerabilities. CSO may use any of the following tools and methods to perform the web application assessments:

- Cenzic Hailstorm – Dynamic application security testing tool
- Core Impact – Dynamic application security testing tool
- Veracode – Static code analysis and dynamic application security testing tool
- Manual Review

2.2 Agency Deviation Request Process

There may be circumstances where a specific mitigation strategy cannot be applied due to technical limitations, or because the strategy would adversely affect business processes. In other circumstances, a cost-risk analysis may indicate that other mitigating factors and compensating controls are preferable to application of the strategy provided in this standard. Implementations that do not satisfy the requirements stated in this standard must follow CSO-PROS-1324, “U.S. Nuclear Regulatory Commission Deviation Request Process” to obtain approval to operate before the server/cluster is placed into production use. The agency template, CSO-TEMP-2017, “Deviation Request Form Template” must be used to complete the Deviation Request Form, and CSO-TEMP-2017, “Deviation Request Transmittal Memo” must be used to prepare the memo used to submit the deviation request to the Designated Approving Authority (DAA).

3 SPECIFIC REQUIREMENTS

Common web vulnerabilities that must be mitigated per this standard are described in Table 3-1 on page 5. The vulnerabilities were published in the OWASP Top Ten web application security vulnerabilities list for 2010[1] and 2007[2]. The 2010 OWASP Top Ten list is the externally published standard that supplements the requirements in CSO-STD-1108; two additional vulnerabilities (published in the 2007 OWASP Top Ten list) must also be mitigated per this standard. All vulnerabilities that must be mitigated per this standard are listed and described in Table 3-1 on page 5.

Under certain circumstances, NRC web sites may be required to address additional critical vulnerabilities. Notification of additional critical vulnerabilities will be provided to system owners via a formal memorandum from the CSO Standards DAA.

[1] OWASP Top Ten 2010 - https://www.owasp.org/index.php/Top_10_2010
[2] OWASP Top Ten 2007 - https://www.owasp.org/index.php/Top_10_2007

This page intentionally left blank.

Table 3-1: Common Web Application Vulnerabilities & Mitigation Strategies

Vulnerability	Description	Risk	Mitigation Strategies	OWASP Version
Injection Flaws	• Injection flaws allow attackers to pass malicious code through a web application to another system. • Attacks include calls to the operating system via system calls, the use of external programs via shell commands, and calls to backend databases via Structured Query Language (SQL) (i.e., SQL injection). • SQL injection attacks are the most common. These attacks execute SQL queries entered in a text form. Other injection flaws include Operating System (OS) command and Lightweight Directory Access Protocol (LDAP) injections.	• Malicious users can exploit injection flaws if the web application is not configured to properly validate user input. • Attackers might attempt to trick the web application into providing unauthorized data, prevent specific site functions, or locate other vulnerabilities to exploit.	• Separate user entered or untrusted data from commands and queries. User input should be validated before it is incorporated into commands and queries. • Use bind variables for all prepared statements. Use stored procedures for SQL queries. Never build SQL statements directly from user input. • For custom developed applications, conduct a static code analysis of the web application before putting into operation. • If a parameterized Application Programming Interface (API) is not available, carefully escape special characters using the escape syntax for the appropriate program interpreter.	2010

Vulnerability	Description	Risk	Mitigation Strategies	OWASP Version
Cross-Site Scripting	• XSS is a web application vulnerability that allows attackers to inject client-side scripts into web pages viewed by other users. • Malicious script is usually JavaScript.	• XSS allows attackers to execute scripts in the victim's browser by exploiting the trust between the user and website. XSS can be used to hijack user sessions, deface websites, insert hostile content, conduct phishing attacks, and take over the user's browser using scripting malware.	• Before accepting the data/content, use a standard input validation mechanism to validate all input data for length, type, syntax, and business rules, then escape them before the data/content is displayed on the browser. Proper output encoding/escaping ensures that input is always treated as text in the browser (rather than active content that might be executed). • User entered or untrusted data should be separated from active browser content.	2010

Vulnerability	Description	Risk	Mitigation Strategies	OWASP Version
Broken Authentication and Session Management	• Authentication and session management is critical to web application security. These activities handle user interaction with a web application, such as logging in, saving preferences, or timing out due to inactivity. • Authentication and session management flaws usually involve the failure to protect credentials and session tokens throughout their lifecycle. Common vulnerabilities include showing session IDs in the URL and transmitting sensitive information without a Secure Sockets Layer (SSL)/ Transport Layer Security (TLS) connection.	• Malicious users can exploit vulnerabilities to take over sessions, impersonate another user, or hijack user or administrative accounts. This circumvents authorization and accountability controls and might result in privacy violations.	• Applications should not store any part of their access credentials in clear text. • Applications should not expose the credential in untrusted locations, such as cookies, headers or hidden fields. • Access credentials should be protected using encryption (e.g. hashing). • Session IDs, passwords, and user credentials should always be sent over SSL/TLS connections and should not be included in the URL. • Inactive sessions should be automatically logged off. • Users should be required to enter their old password when changing to a new password.	2010

Vulnerability	Description	Risk	Mitigation Strategies	OWASP Version
Insecure Direct Object References	• A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, database record, or key, as a URL or form parameter. • An example could be a malicious user changing a User ID parameter appearing on a URL to attempt to gain unauthorized access to another account.	• If an access control check is not in place, attackers can manipulate direct object references for the purpose of accessing other data or functions without authorization.	• Avoid exposing your private object references to users (e.g. paths and filename paths). • Replace the actual reference with a mapping that is specific to that specific user for that particular session. • When direct object references are used, verify access permissions for the user before displaying data or executing actions.	2010

Vulnerability	Description	Risk	Mitigation Strategies	OWASP Version
Cross-Site Request Forgery (CSRF)	• A CSRF attack forces a logged-on victim's browser to send a request to a vulnerable web application, which then performs the chosen action on behalf of the victim. • The attack takes advantage of a website's predictable access-restricted actions, such as updating the email address or the password for an account. • Any vulnerable web application without authorization checks for actions will process an action if the application thinks that the actions are legitimate requests from the victim. • Clickjacking is a form of CSRF, where a user is deceived into performing undesired actions by clicking on hidden links. For example, an attacker can show a fake button on a page controlled by the attacker, and then load another page over the first page in a transparent layer. Users may click on the visible button not realizing they are actually performing actions on the hidden page.	• If malicious users can predict the details for a particular action, they can trick logged-in users into clicking a forged link, typically through a phishing email designed for the purpose of executing actions in users' accounts. • A successful CSRF exploit can compromise end user data or execute other actions with the permissions of the victim.	• Use a random token in the URL or body of each Hypertext Transfer Protocol (HTTP) request. • Force a logoff immediately after using a web application.	2010

Vulnerability	Description	Risk	Mitigation Strategies	OWASP Version
Security Misconfiguration	• Security misconfiguration attacks exploit configuration weaknesses found in web applications. • Many applications come with unnecessary and unsafe features enabled by default and do not meet the least functionality requirement out of the box. • This also includes failure to keep web servers, applications, and OS software up-to-date.	• A poorly configured application could allow attackers to obtain unauthorized access, compromise files, or perform other unintended actions.	• For COTS/GOTS components, use a repeatable, documented, hardening process. • Keep web application components up-to-date. Test and implement security patches/fixes. • Disable unnecessary services/features for the application. • Change default user names and passwords. Use strong, unique passwords for every account in compliance with CSO-STD-0001, “NRC Strong Password Standard.” • Disable directory listings that are not required, and set access controls to deny all requests to non-public files. • Remove unnecessary files, such as configuration files, install files, and demonstration/sample websites that come with default web applications or web component installations. • Set access controls based on least privilege, allowing the minimum access required for users to accomplish their tasks within the web application.	2010

Vulnerability	Description	Risk	Mitigation Strategies	OWASP Version
Insecure Cryptographic Storage	• A collection of vulnerabilities that serve to ensure important data is encrypted when necessary. This includes, but is not limited to, encrypting the correct data, proper key storage and management, and using strong algorithms. For example, Safeguards Information (SGI) and Personally Identifiable Information (PII) data types always require encryption, whereas Sensitive Unclassified Non-Safeguards Information (SUNSI) may require encryption, depending on the nature of the data.	• Malicious users can more easily access insecurely stored data. • This may lead to unauthorized disclosure of sensitive data, identity theft, and compliance violations. • Regulatory requirements may force encryption of data at rest for various data types.	• Use Federal Information Processing Standard (FIPS) 140-2 validated encryption in compliance with CSO-STD-2009, “Cryptographic Control Standard.” • Only allow authorized users to access decrypted data. • Do not mark private keys as exportable when generating the certificate-signing request. • Avoid storage of keys within source code or binary code.	2010

Vulnerability	Description	Risk	Mitigation Strategies	OWASP Version
Failure to Restrict URL Access	• Typically, the only protection of a URL that links to restricted content is simply not linking that page to unauthorized users. This *security by obscurity* method is not sufficient to protect sensitive functions and data in an application. • Failure to ensure that access control checks are performed before providing access to web application functionality or content may lead to unauthorized disclosure of sensitive functions and content. • For example, an application developer might attempt to hide functionality from users by creating "hidden pages," or pages that do not have a link pointing to them, preventing web crawlers from indexing them. The developer mistakenly assumes that these pages would never be found by anyone who does not know the exact URL. However, attackers typically find these pages through forceful browsing and access controls on these pages are usually not restrictive.	• A motivated or skilled attacker may be able to find and access "hidden pages," invoke functions, and view data.	• Always protect web pages that provide administrative and high privilege functionality. URLs and business functions should be protected by access controls that verify the user's role and entitlements prior to any processing taking place. Do not assume that users will be unaware of special or hidden URLs. • Enforcement mechanisms should deny all access by default, requiring explicit grants to specific users and roles for access to every page excluding publicly available content. • Consider developing a matrix that maps the roles and functions of the application to protect against unrestricted URL access. • Implement role-based access control within the web application. This will ease the level of effort required to maintain access controls.	2010

Vulnerability	Description	Risk	Mitigation Strategies	OWASP Version
Insufficient Transport Layer Protection	• Applications frequently fail to encrypt network traffic when it is necessary to protect SUNSI and all SGI communications. Encryption (SSL/TLS) must be used for all authenticated connections, access-restricted pages, non-public content, and cookies or session information during transit from a user's browser to the web server.	• Attackers can abuse applications that utilize weak encryption algorithms, fall back, or can be forced out of encryption mode. • Applications that fail to encrypt network traffic may expose authentication or session tokens to a malicious user who might intercept and view the information.	• Require SSL 3.1 or TLS 1.0 as the minimum versions used for all authenticated traffic. • Certificates should be valid. Certificates should not be expired or revoked, and should match all domains used by the site. • Non-SSL/TLS requests should be redirected to the SSL/TLS page. • Set the secure flag for session cookies to ensure that the browser never transmits them in the clear. • Configure your SSL/TLS provider to only support strong (e.g., FIPS 140 compliant) algorithms when appropriate. • Only provide support for the TLS protocols. • Do not mix SSL/TLS and Non-SSL/Non-TLS content. • When using SSL/TLS, perform session encryption for the entire session. Only protecting the logon credentials is insufficient. Data and session information should be encrypted as well.	2010

Vulnerability	Description	Risk	Mitigation Strategies	OWASP Version
Unvalidated Redirects and Forwards	• Web applications frequently use redirects or forwards to send users to other destinations. If the web application does not verify the destination, redirects or forwards might be vulnerable to modification.	• An attacker can change the destination address to send visitors to a malicious site that appears to be part of the original location. • Phishing schemes often exploit un-validated redirects and forwards, because an attacker can hide a malicious URL behind the original address.	• Avoid using redirects and forwards unless absolutely required. If redirects or forwards are required, validate against a “white-list” for authorized destinations. • Consider disallowing the requests for off-site redirects or forwards.	2010
Malicious File Execution	• Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, resulting in devastating attacks, such as total server compromise. Malicious file execution attacks affect PHP, XML and any framework, which accepts filenames or files from users.	• An attacker may be able to remotely execute code through providing input (such as remote URLs) to the application. • Depending on the hardening state of the web server, this remote code execution may lead to the full compromise of the server.	• Restrict the web application from using user-provided input (e.g., URLs to remote servers) for filenames of any server-based resources to prevent attackers. • Implement firewall rules to restrict the web application from creating new outbound connections to the Internet to prevent attackers from executing attacks that rely on the inclusion of remote files.	2007

Vulnerability	Description	Risk	Mitigation Strategies	OWASP Version
Information Leakage and Improper Error Handling	• Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. • Examples of improper error handling include when a web application provides the user with excessive information (e.g., stack traces, failed SQL statements, or other debugging information). • Examples of other information leakage includes when web application functions produce different results for different user inputs, such as when web application login functionality indicates that a user does not exist or that the password is incorrect, which would allow an attacker to enumerate valid usernames, as opposed to indicating that the user and password combination is incorrect.	• Attackers use this weakness to steal sensitive data, or to provide necessary information to perform attacks that are more serious.	• Disable detailed error handling within the web application. If that is not possible, limit information disclosed in errors as much as possible so as not to disclose information that could be used by an attacker. • Make certain that the debug information, stack trace, SQL statements, internal IP addresses or hostnames, or path information is not displayed to users within error messages. • Several layers of the web application may return errors, such as the database layer, the web server, or middleware. Make sure that errors from all layers adequately checked and configured to restrict the information provided to the end user. • To assist the end user in understanding the impact of the errors, create a default error handler(s) that returns sanitized error messages.	2007

This page intentionally left blank.

4 DEFINITIONS

Application Developer	A computer professional whose primary role involves developing and/or implementing software applications.
Bind Variables	A variable within an application that associates validated user data to an SQL statement and data when communicating to the database. Bind variables are used in the VALUES clause of INSERT statements, WHERE clauses, or the SET clause of UPDATE statements. After user input is validated and bound to a bind variable, the variable can be used to prepare SQL statements that are executed by the application. Bind variables are also called INTO variables or substitution variables.
Commercial off-the-shelf (COTS)	Software products developed by commercial organizations for the general public.
Dynamic Application Security Testing	A security assessment method that takes place while executing the application being assessed. The assessment method includes examining the application in the running state, executing application functionality and providing it expected and unexpected data input, and observing the application's behavior to discover security vulnerabilities.
Government off-the-shelf (GOTS)	Software products (available for use by Government Agencies) that are developed by the NRC or another U.S. Federal, State, Local, or Tribal Government Agency.
Interpreter	A computer program that executes source code or translates it into intermediate code that can then be executed.
Intranet	A private network that is employed within the confines of a given enterprise (i.e., internal to a business or agency).
Open Web Application Security Project (OWASP)	An open-source application security project that is managed and supported by the OWASP Foundation, which is a non-profit organization that focuses on improving the security of application software.
OWASP Top Ten	The OWASP Top Ten identifies the most critical web application vulnerabilities that face organizations today.
Static Code Analysis	An analysis of computer software code that takes place without executing the code being analyzed. Static code analysis might involve a manual examination of the code or use of an automated tool to examine the code without executing the programs.

Stored Procedures	A subroutine or object that is available to applications that access a database. The procedures are stored in the database and typically contain business logic.
Structured Query Language (SQL)	A programming language designed for the purpose of managing data in databases. SQL is used to find information and update information stored in databases.
Web Application	An application that is accessed using a web browser over a network, such as the Internet or the NRC intranet. Static file listings on a web server are not considered web applications.
White-list	A list of allowable/authorized actions or input (e.g., characters). All actions or input that is not included on the white list is prohibited.

5 ACRONYMS

API	Application Programming Interface
COTS	Commercial off-the-shelf
CSO	Computer Security Office
CSRF	Cross-Site Request Forgery
DAA	Designated Approving Authority
FIPS	Federal Information Processing Standard
GOTS	Government off-the-shelf
HTTP	Hypertext Transfer Protocol
ISSO	Information System Security Officer
LDAP	Lightweight Directory Access Protocol
OS	Operating System
OWASP	Open Web Application Security Project
PII	Personally Identifiable Information
SGI	Safeguards Information
SQL	Structured Query Language
SSL	Secure Sockets Layer
SUNSI	Sensitive Unclassified Non-Safeguards Information
TLS	Transport Layer Security
URL	Universal Resource Locator
XSS	Cross-Site Scripting

CSO-STD-1108 Change History

Date	Version	Description of Changes	Method Used to Announce & Distribute	Training
08-Aug-12	1.0	Initial Release	CSO web page and notification of ISSO forum	Upon request