ASU Secure Web Development Standard

Version 1.3
Last Revised February 19, 2010

Table of Contents

1 Document Revision 3
2 Purpose 4
3 Applicability 4
4 Principles 4
5 Recommendations 9
6 Definition of Terms 9
7 Enforcement 10
8 Standard Safeguarding 11

1 Document Revision

Date	Revision	Revised By	Basis for Revision & Description

Document Change/Correction Request

Your feedback is valuable. Anyone who uses this document is invited to use this form to submit requests for changes or corrections to this document. Please complete all of the sections and mail to DL.WG.INFOSEC@mainex1.asu.edu

Document Identification

Document title
Date of Document
Master File Location

Requester Identification

Name	
Department	
Phone Number	( ) -
Email Address	

Request

The following Change is requested:

Please describe the change and reason(s) why the change is requested.

Correction:

Section ___, page ____ contains the following erroneous information:

This information should be changed to:

Attach additional sheets if necessary to fully describe the change or correction.

2 Purpose

The ASU Secure Web Development Standard establishes guidelines and standards for the preservation of the confidentiality, integrity and availability of University information resources associated with web sites. Additionally, this minimum standard provides for the integrity of institutional processes and records, and supports the University's compliance with state and federal laws, rules and regulations. Adherence to the standard will increase the security of web applications and help safeguard university information.

3 Applicability

- All ASU faculty, staff, student or other employees, affiliates, vendors, service providers, contractors and sub-contractors.
- All web sites within the asu.edu DNS domain that process, transmit, store, or have access to University "Public," "Confidential," or "High Risk" data (defined below).
- All web sites within ASU's networks, regardless of DNS domain, that process, transmit, store, or have access to University "Public," "Confidential" or "High Risk" data.
- All web sites that share resources with those listed above. Sharing resources includes:
  - Hosted on the same physical or virtual machine.
  - Communicating through the same application server.
  - Accessing the same data.

All web sites that fall within the scope defined above, and implemented after the effective date of this standard, should comply with this standard. Existing web sites that fall within the scope defined above must be brought into compliance as soon as practical. In some circumstances, compliance with specific requirements may not be immediately possible. Under those circumstances, academic or business units must confer with the Information Security Office to develop a plan for coming into compliance with this standard within a reasonable amount of time.

4 Principles

The purpose of the following principles is to protect information from accidental or intentional unauthorized modification, destruction or disclosure in accordance with the information's sensitivity, criticality and value to the University.

See ASU's Data Classification Standard for a definition of "Public", "Confidential", and "High Risk" data. If a project uses a mixture of data from different classification levels, the standards for the highest level apply to the entire project.

Alignment with Other Standards, Policies and Regulations

See ASU's Data Classification Standard at http://getprotected.asu.edu/files/data.pdf for a definition of "Public", "Confidential", and "High Risk" data, and the steps that must be taken to protect data depending on its classification.

See ASU's Patch Management Standard at http://getprotected.asu.edu/files/patch.pdf for systems that must be regularly updated with security patches.

See ASU's Software Development LifeCycle (SDLC) at http://getprotected.asu.edu/files/stand_sec_dev_life.pdf for security requirements and checkpoints throughout the life of the system.

Authentication & Authorization

Use ASU standard mechanisms such as WebAuth for authentication instead of managing your own passwords, and EDNA to make authorization decisions based on roles rather than individual user names.

Change Management

Changes to executable code and systems should be made as part of a defined process with notification and review appropriate to the potential impact of the change. Changes should not be approved for production unless all review checkpoints in the SDLC have been successfully passed.

Data Classification

Systems that handle different classification levels of data must be separated physically or virtually. If this is not practical, the system that handles a lower classification level of data must be secured to the standards of the higher data classification. For example, if a "Public" web site shares resources with a "High Risk" site, the Public site must be secured as if it were a High Risk site, so that the Public site cannot be easily breached and used to attack the High Risk site.

Defense in Depth

When possible, do not rely on a single layer of protection. Assume any one layer will be breached, and provide additional layers between the attacker and the protected information.

See the section "Multi Tier Architecture" below for a specific example of this principle.

Fail Securely

Error messages that are visible to users or transmitted to browsers should not reveal information a site visitor does not need, but that might be useful to attackers, such as details about the location or type of back-end systems used. For example, the following error message reveals information an attacker could potentially abuse in combination with other footholds:

```
Cannot connect to database "webdb1" using ID "siteuser".
```

A better message would omit such details:

```
Temporary server error. Please try again later.
```

Unexpected conditions should result in access being denied by default. For example, the following pseudocode inadvertently allows the sensitive transaction if the "forbidden" or "block" functions exit prematurely due to an internal error:

```
if (forbidden(user)) then block(request)
/* sensitive admin-only transaction here */
```

If the "is_admin" function in this pseudocode fails for some unexpected reason, the default action is to deny the administrative function instead of permitting it:

```
if (is_admin(user)) then {
        /* sensitive admin-only transaction here */
}
```

Similarly, the is_admin function should also default to denying access, and only grant access when everything flows normally:

```
function is_admin {
        result = false /* default: you are not an admin */
        exit unless passed_admin_test_1
        exit unless passed_admin_test_2
        result = true /* you passed all the admin tests */
}
```

Footholds for Attack Escalation

If a hardware or software system is successfully compromised by an attacker, to the extent practical it must not provide a platform granting additional access or privileges to the attacker over what was previously available.

For example, if a web server or web host becomes fully "owned" or "rooted", the attacker should not be able to exploit that stance to gain additional access to a database used by the web server. This can be accomplished by ensuring that the web server itself does not have database access beyond the minimum necessary to accomplish its legitimate functions. An application server between the web server and the database can filter all requests against a white list of known valid requests, by default denying all others.

This also implies that a web server, for example, should not store passwords that, if discovered, could be used by an attacker to gain additional access to other systems beyond those already accessible from the web server.

Least Privilege

A system or process should not have any access or authority beyond the minimum necessary to accomplish its intended purpose. This reduces the damage that can be done if a compromise occurs.

Logging and Auditing

Changes to systems or data should be logged using a tamper resistant mechanism so that future audits can determine what happened, when, and under whose authority.

Maintain Current Software

Computers must be regularly updated with security patches as described in ASU's Patch Management Standard. Security flaws discovered in locally developed software must be corrected within a time appropriate to the corresponding risk. This implies that while a web site remains live, resources must be available to support it, and if flaws cannot be corrected within a reasonable time, the site must be shut down.

Minimize Attack Surface

A system should only expose those functions necessary for its purpose. A white list of required services and ports should be enabled, and by default everything else should be disabled or removed. As part of every system's life cycle, define an exit plan and resources to decommission obsolete web sites or portions of web sites. Also define automated or periodic manual processes to remove people from roles when they should no longer have access to the web site.

Multi Tier Architecture

Secure web architecture requires designing for the circumstance where a web server falls to the total control of a malicious hacker. If the web server were to have the ability to issue arbitrary SQL commands directly to a database, the successful hacker would also gain that ability. The web server, therefore, must be extremely limited in what it can do with any back-end systems.

A web site that uses a database should have a minimum of three tiers: (1) web server, (2) application server, and (3) database server. Each tier should reside on physically separate hardware. The web server should communicate with the application server, while the application server communicates with the database server. The web server should canonicalize incoming requests, translating them to a different protocol, to reduce the chance of malformed attacks being passed along to the application server. The application server has a white list of acceptable functions that the web server needs to perform, and by default forbids every other type of request.

Web servers should not be able to communicate directly with databases. The network should be configured (by segmentation and/or firewall rules) so that application servers can only receive requests from a white list of web servers, and database servers can only receive requests from a white list of application servers.

References:

- http://en.wikipedia.org/wiki/Multitier_architecture
- http://en.wikipedia.org/wiki/Application_server

Positive and Negative Test Cases

During the design phase of a web development project, both positive and negative test cases should be identified. For example:

- Positive test case: the web site shall display a complete and accurate list of classes taken by the student.

- Negative test case: the web site shall not display the student's list of classes to any unauthorized person.

During development, prior to launch, and as part of ongoing maintenance, both positive and negative test cases should be retested to ensure the site still functions as designed.

Physical Security

Equipment must be physically secured against unauthorized access with measures appropriate to the data classification and risk.

Positive Security Model (White List / Default Deny)

Any type of filter or restriction should by default block or deny everything, only allowing things that have been specifically identified as valid and acceptable. For example, it is not sufficient to remove "bad" characters from user data. Instead, use a white list of known "good" harmless characters and reject all others.

Reuse Previously Validated Code

Instead of developing a custom solution to a previously solved problem, take advantage of existing solutions that have already proven their value.

Threat Modeling and Security Testing

All new web applications, or significant change to an existing web application should use a set of threat modeling procedures during development and require security testing prior to a production migration. See Security Standard for ASU Web Applications document for detailed information.

Separation of Duties

It should not be possible for one person, acting alone, to compromise the security of University information resources. For example:

- The person who writes a module of software should not have the ability to place that software into production. Otherwise, that person could deploy malicious code without review. Instead, software authors should stage their production-ready work in an area where someone else can retrieve it and deploy it into production.
- The person who has the ability to put code into production should not, by the same reasoning, have the ability to stage software for deployment.

As mentioned earlier, logs should be maintained to show who placed the software into the staging area and when, and who deployed it to production and when.

Strong Encryption

At a minimum, 128-bit encryption should be mandated when establishing secure communication channels or storing data that must be encrypted. Stronger levels of encryption may be necessary depending on advances in technology. When establishing an encrypted connection, verify the digital certificate to ensure the other party is who they claim to be. Maintain current and correct encryption certificates on web servers. Encrypt sensitive data in all places it is stored, not just when it is in transit.

Trusted Authorship of Content and Code

Anonymous or unauthenticated parties must not be allowed to insert potentially malicious or inappropriate content into ASU web sites. This implies, for example:

- Web pages should not include by reference scripted code from other web sites, unless ASU has a contractual relationship with the other party, such contract to include these standards by reference.
- Contractually require all other third parties to adhere to this standard.

Validate Untrusted Data

All information, whatever its source, whenever it is passed to other systems, used in decision making, or displayed to the user, must first be validated against a white list of known good

patterns, values or characters. This includes user input, browser supplied headers such as cookies, and data acquired from other systems.

Validation must effectively prevent SQL Injection, Cross-Site Scripting, buffer overflows, and similar attacks. See the latest OWASP Development Guide at http://owasp.org/index.php/Category:OWASP_Guide_Project for details and examples.

5 Recommendations

Recommendations are not mandatory, however, web developers are urged to follow them as best practices when possible. Some of these recommendations can help a web site fit into future enterprise infrastructure developments with a minimum of redesign or rework.

- Servers that have access to "Confidential" or "High Risk" data should not mix http and https URLs on the same DNS hostname. Future web protection devices (application firewalls, intrusion detection or prevention devices, etc.) may need to exist between the point of https decryption and the web server. In this scenario, one way to prevent http connections to https URLs would be to associate the https URLs with a distinct DNS hostname.
- Colleges and departments may host the site with UTO Web Hosting to benefit from current and future techniques to improve site availability and security, such as application firewalls, redundancy, software patches, incident monitoring etc.
- Validate HTML using http://validator.w3.org/
- Validate CSS using http://jigsaw.w3.org/css-validator/
- When possible, design so that a client can effectively or partially use the web site even if their browser does not support (or shuts off) the following technologies:
  - JavaScript or any other client side scripting
  - Pop-up windows
  - Client side Java
  - Active X
  - Flash and similar technologies
  - PDF viewer
  - Other plug-ins / media players
- Write standards compliant code so that the site will work for all standards compliant browsers regardless of operating system. Do not code specifically for one type of user agent device.

6 Definition of Terms

An ***application server*** communicates with a web server and a database server, preventing the web server from directly accessing the database server, and ensuring that all queries and updates conform to a white list of authorized transactions, programs, and authentication credentials. This term may refer to a dedicated hardware appliance, or to a software server running on generic hardware.

A ***black list*** specifically identifies items that are known to be undesirable, and allows all others. See http://en.wikipedia.org/wiki/Blacklist_(computing) and "white list". Because it is difficult to specify every possible variation of an undesirable pattern, black lists are generally considered insufficient.

The ***client*** typically refers to a web browser and the associated machine and human user, however in some cases a client may be a search engine or other automated source of web requests.

A ***database*** includes any technique for storing and/or retrieving bytes that are incorporated into, collected from, or modified by a web page. However the term does not include the normal web logs that are produced by a web server, nor the files that comprise a static web page.

A ***database server*** is a software package that responds to data query and update requests.

A ***form*** is web page that includes one or more <form> tags or their functional equivalent.

A ***request*** is an HTTP message from a client to a web server.

A ***response*** consists of HTTP headers and optional content from a web server to a client in reply to the client's request.

A ***server*** is any computer that accepts incoming connections. A desktop computer may be a server if it provides services to other computers. Servers typically, but not necessarily, have static IP addresses so that other systems may readily locate them and connect to them.

A ***server program*** is any software, other than a web server, that runs on the server side and generates a real-time response to a request. This term applies regardless of the programming language (for example Java or Perl) or page suffix (for example .asp .cgi or .pl).

A ***static page*** is a web page that is not generated at the time of the request. It does not contain or invoke any web server-side code beyond the code of the web server itself. It only changes when an author manually modifies it. It is not a server program. It does not receive form input via POST requests nor by query parameters. A page that includes, for example, a client-side script to display the current date or time is still a static page because the script executes on the browser, not the web server, and the text of the script changes only when an author modifies it.

A ***web application*** is server side logic that determines what content is sent to the users internet browser based on data from a database or a web service.

A ***web host*** is a physical or virtual machine that hosts one or more web sites.

A ***web page*** is the combination of HTTP headers, HTML, CSS, images, scripts, frames etc. that are sent to a browser in response to an initial URL request and the subsequent URLs recursively embedded within the first response.

A ***web server*** is a software package that responds to web requests. Popular web servers include Apache and IIS.

A ***web site*** is a collection of web pages supporting a particular organization or function. The component web pages are usually, but not always, contained on a single web host.

A ***white list*** specifically identifies items that are known to be acceptable, and denies all others. See http://en.wikipedia.org/wiki/Whitelist and "black list". Generally speaking, white lists are preferred over black lists, because they reject by default anything the list maker did not anticipate.

7 Enforcement

Violation of this standard may lead to discipline as well as other applicable sanctions.

In some circumstances, compliance with specific requirements may not be immediately possible. Under those circumstances, academic or business units must confer with the Information Security Office to develop a plan for coming into compliance with this standard within a reasonable amount of time.

8 Standard Safeguarding

Modification and review of this standard will occur annually to ensure that information provided within the standard remains current. Changes are to be approved and put into place by the Information Security Officer. To offer suggestions and/or recommendations regarding this standard, please complete the appropriate Documentation Revision form (Section 2) and e-mail the form to InfoSecurity@asu.edu.