Cheat Sheet: Web Application Development

Revision History

Date	Action
5/23/2015	Initial Draft – Mike Cook
8/06/2015	Reviewed. Content suggestions. – Hien Huynh
11/10/2015	Incorporated changes from campus constituents – Distributed to Campus.

Table of Contents

Vulnerability Matrix

Level	Risk Rating	Examples
Session Management	High	Wherever possible use built in session management. Where possible, do not store session information in headers or cookies. Use "HttpOnly" and "secure" flags for where cookies are needed. Validate authentication/tokens and user security roles on each page performing database actions or displaying confidential data. Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
Data Validation	High	The most common web application security weakness is the failure to properly validate input from the client or environment. This weakness leads to almost all of the major vulnerabilities in applications, such as Interpreter Injection, locale/Unicode attacks, file system attacks and buffer overflows. Where possible, web applications validate all data for expected values, passed to interpreters, including Web browsers, database systems, and command shells, use server-side, data from another source needs to be trustworthy, etc.
Input Sanitization & Buffer Overflow Prevention	High/Med	Fields used as input mechanisms for database queries or other tools capable of gaining access to confidential data must be validated (Regex or other mechanism) prior to being sent to the command interpreter. Set character sets. Set correct locale. Set content types. Set input constraints. Validate field lengths (input data size) and data type when processing data submitted through forms. Prior to execution, ensure option and radio buttons data is part of the expected data set.
Injection & Cross-site Scripting (XSS)	High	Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization. XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
Server Configuration	High	Ensure servers are running a currently supported version (manufacturer is releasing security updates regularly) of Operating Systems, Databases and web application software (IIS, Apache, Tomcat, etc.). Ensure servers are on routine patch management cycles. Ensure servers are running appropriate antivirus software. Servers touching public-facing networks which handle confidential Level 1 Data and/or credentials must have encryption enabled using a currently supported encryption mechanism

Level	Risk Rating	Examples
		(I.e. TLS 1.2) and all unsupported protocols (TLS 1.0, SSL 2.0, etc.) must be disabled. Security ciphers must be AES 128 or better. Hashes must be SHA 256 or better. Ensure non-web data (Web.config, logs, etc.) is stored outside the webroot. Do not use hard-coded credentials in source code. Do not store uncompiled source code on any public network facing or database server. Separate production from non-production where possible. Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
Access Control & Request Forgery	High	Ensure all pages within an authenticated web application require authentication. Validate logged in user role is sufficient to perform action or view content on all pages. Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization. A Cross-Site Request Forgery (CSRF) attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
Outdated Libraries	High	Use components, libraries and frameworks which are currently supported by the manufacturer and receive regular security updates. Outdated libraries (i.e. Java 6) shall not be supported. Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts

Testing for Readiness

Action	Requirement Checked	Where to get help

Qualysguard	Identify OWASP Top 10 Vulnerabilities (e.g. SQL Injection/XSS, CSRF, Unvalidated Redirects) TCP/UDP Port Scan & Service Discovery Hidden Malware Buffer Overflow Outdated Libraries Outdated OS Patches Installed Encryption/Certificates/Protocols	Qualysguard for Web Applications is available at no cost to all state and auxiliary IT departments. Contact the Information Security Office for more details.
Manual check for Source Code on web/database server	Source Code on web/database server	This should be performed by the team responsible for code deployment. Contact Computing Services or Database Services for assistance.
Manual check for Antivirus	Antivirus (Sophos)	This should be performed by the team responsible for code deployment. Contact Computing Services for assistance.
Manual check for Patch Management	Automated Patch Management (BigFix)	This should be performed by the team responsible for code deployment. Contact Computing Services for assistance.
Manual check for data inside the Webroot	Ensure Web.config, logs and other files are not accessible by users of the web application.	This should be performed by the team responsible for code deployment. Contact Computing Services for assistance.
Peer review of code	Coding Practices: • Do not store passwords in source code • Authentication validation must be performed on all pages and prior to all database read/writes. • Input fields must be sanitized (SQL Injection Prevention, etc.) • Do not redirect URL's to untrusted sites • Do not make use of hashes without a salt	Administrative Applications, Information Security Office.

	• Do not use outdated libraries. • Sanitize error messages displayed to users • Data Validation: E.g. validate size and type of data received from input fields, validate all data for expected values, etc.	