Application Code Development Standards

Overview

This document is intended to provide guidance to campus system owners and software developers regarding secure software engineering practices. These standards apply to both web-based and non web-based code, and to both centralized and decentralized campus resources. All new and existing code written for any campus resource must comply prior to deploying any Internet-facing application.

Procedures to ensure application source code is protected

Developers need to use source code version control management software which uses identity based access controls. The software should be capable of performing functions such as tracking and retrieving different versions of source code, comparing changes in current and previous versions of source code and checking the change log.

Procedures for testing code

The application testing process is vital in identifying security flaws before the application is released. Developers need to test the application's security controls to verify they are working properly, prior to deploying the system into a production environment. Test plan(s) and test results should be documented.

Previously deployed systems need to be tested as part of any significant upgrade or as determined necessary by an assessment of common vulnerabilities (e.g., SANS Top 20 Security Risks).

In addition to the standard functional testing performed (user-driven and/or automated), a separate developer/tester needs to be assigned specifically to test the application for security flaws, and ensure that the application does not modify data files outside the scope of the application. The tester needs to create, update, and execute test plans and procedures; both for changes in the application and prior to each production application release. All the test procedures should ensure system initialization, shutdown, and aborts are configured such that the system remains in a secure state.

Web software applications need to be developed per secure coding guidelines such as the Open Web Application Security Project (OWASP) guidelines. Before being placed into a campus production environment, such applications need to be reviewed and tested for the following vulnerabilities:

- Un-validated input
- Inadequate access control
- Inadequate authentication and session management
- Cross-site scripting (XSS) attacks
- Buffer overflows
- Injection flaws
- Improper error handling
- Insecure storage
- Denial of service
- Insecure configuration management

Protected data should not be used for testing or development purposes, except where unavoidable. Also, if production data must be used in a non-production environment, then security controls in the nonproduction environment need to be as strong as the security controls in the production environment.

Suggested testing methods and testing tools are further detailed in **Appendix A**.

Procedures for migration between development and production environments

In order to maintain a stable, protected production system, developers need to develop a multi-tier environment, such as Development, Quality Assurance, Staging, and Production. In tandem with this, there needs to be processes defining how to move data and applications out of development and into production, as well as a set of tools that we use for maintaining these processes.

All test data and test accounts need to be removed before deploying an information system into a production environment. Once in the production environment, regular risk assessments must be conducted to ensure the system has security controls that appropriately protect the confidentiality, integrity and availability of the system.

Procedures for user acceptance and deployment

Developers need to supply appropriate documentation when deploying an application. Such documentation may include the assigned duties of personnel using the application; including any prerequisite training, security issues or other special requirements. The documentation should also detail specific configuration requirements for the application, including necessary ports, protocols and services, and operational procedures. If the application handles protected information, then details regarding secure access and storage should be specified.

Finally, the system owner should ensure that, as long as the application is in use, a resource is available to address any security flaws discovered in the application.

Review/Approval History

Date	Audience	Action	Version
2/2/2009	System Security Meeting	Presented	v1.0
2/27/2009	Chief Information Officer	Reviewed	v1.0
3/2/2009	Cabinet	Reviewed	v1.0
4/24/2009	Chief Information Officer	Approved	V1.0

Appendix A: Testing Methods and Tools

Fuzz Testing

Fuzz testing is a testing method that can help uncover reliability and security vulnerabilities in a software product. Fuzz testing relies on building or manufacturing deliberately malformed data and then having the application under test consume the data. Processing the malformed data often leads to an application crash; of which a percentage of the crashes are in fact security vulnerabilities. It is important that all critical applications, most notably those facing the Internet or those that consume and parse files be fuzzed. The tester needs to ensure fuzz testing is included in the test plans and procedures and performed for each application release based on application exposure.

Code Coverage

Code coverage is the percentage of the application code exercised during the testing process. Security flaws often occur in areas of the code not regularly executed, so it is important to keep track of how often a code branch is executed and tested to ensure thorough testing is performed. If code coverage is low, then the tests must be evaluated to determine why code coverage is low. The tests should then be changed to increase the percentage of code covered by the test cases. The tester needs to ensure code coverage statistics are maintained for each release of the application.

Code Reviews

A code review is the process of reviewing application code to locate potential functionality problems. Security flaws may also be identified during the code review. Any security flaws found should be entered into the defect tracking system, clearly identified as a security defect, and fixed before the application is released.

Ideally, application reviewers should be objective parties holding no responsibilities for developing the application being certified. The application reviewer should have a strong background in the languages used by the application, as well as training in identifying security flaws.

Code reviews may be automated or manual, and there are many commercial companies offering code review services. The most comprehensive reviews will implement two or more of the review types.

Manual Code Review

A manual code review is performed by one or more application code reviewers. Manual reviews are more time consuming than automated reviews and are reliant upon the skill and experience of the reviewer to find security flaws, but they have the advantage of being able to detect forms of vulnerability that might not be found through automated analysis.

Automated Code Review

Automated code reviews can quickly identify weak areas of an application. Depending on the sophistication of the analysis, static analysis tools may be prone to false positives and may miss some classes of security defect. Automated code review procedures should be put in place to disqualify false positives and manually check for security vulnerabilities the tool fails to identify. Static analysis tools are valuable if the code is very large, but it is not a replacement for manual

code review. There should be manual code review for exposed code, such as Internet-facing code and mobile code.

Third Party Code Review

Several commercial companies offer code review services. These companies use a mixture of automated reviews and manual reviews by experienced auditors. Third-party review is valuable if the application is highly exposed, is large, or cannot be manually reviewed in depth.

Static Analysis Tools

Static code analysis is the analysis of computer software that is performed without actually executing programs built from that software. Static Analysis Tools should:

- Support the programming languages required
- Scan and report vulnerabilities with a minimum of false positives and false negatives
- Support a centralized security policy management so all scans use established policies
- Scan for malicious code detection
- Support the use of an underlying DBMS to collect, report, export and analyze scan results
- Provide remediation for vulnerabilities found
- Provide measurement metrics for long term trending of applications
- Enable collaboration between security teams and development and QA
- Provide customization capabilities to accommodate unique coding styles
- Correlate dynamic testing to assist in the prioritization of static results

There are many commercially available static analysis tools. A list of such tools includes:

Ounce Labs - Ounce Labs' code review tool (formerly Prexis) analyzes the source code of applications and identifies confirmed vulnerabilities and other potential security issues.

Parasoft's Jtest & C++test - Parasoft's Jtest is a Java testing product for development teams building Java EE, SOA, Web, and other Java applications. Parasoft's C++test provides coding policy enforcement, static analysis, code review, unit, and component testing for C and C++ code.

Klocwork Insight - Klocwork Insight finds software bugs and security vulnerabilities in C, C++ and Java code.

Fortify Source Code Analyzer (SCA) - Fortify SCA finds programming errors and vulnerabilities in 12 programming languages: Ajax (JavaScript), C/C++, Classic ASP, COBOL, ColdFusion, Java, .NET, PHP, PL/SQL, T/SQL and VB6. Fortify also finds errors in code that combines any of these 12 programming languages.

GrammaTech CodeSonar - GrammaTech CodeSonar is a source code analysis tool that performs interprocedural analysis on C/C++ code and identifies vulnerabilities in programming logic.

Coverity Prevent - Coverity Prevent identifies and resolves the critical security defects in C, C++ and Java source code.

Web Application Vulnerability Scanners

Web application scanners allow testers and application developers the ability to scan web applications in a fully operational environment and check for many known security vulnerabilities. Web application scanners parse URLs from the target website to find vulnerabilities. These scanners check web applications for common security problems such as SQL injection, cross site scripting, command injection, buffer overflow, session management, and other vulnerabilities. These tools can be used to satisfy code review requirements based on the security checks provided by the tool.

Web application scanners should be used on each web application release prior to deployment to a production environment.

Free tools:

WebScarab Project (OWASP.org) - WebScarab is a framework for analyzing applications that communicate using the HTTP and HTTPS protocols. It is written in Java, and is thus portable to many platforms. WebScarab has several modes of operation, implemented by a number of plugins. In its most common usage, WebScarab operates as an intercepting proxy, allowing the operator to review and modify requests created by the browser before they are sent to the server, and to review and modify responses returned from the server before they are received by the browser. WebScarab is able to intercept both HTTP and HTTPS communication.

Paros Proxy (parosproxy.org) - Paros Proxy is completely written in Java. Through Paros's proxy nature, all HTTP and HTTPS data between server and client, including cookies and form fields, can be intercepted and modified.

Commercial Tools:

WebInspect - HP's WebInspect is a web vulnerability scanning tool that scans web applications for potential security flaws such as buffer overruns, weak cryptography, race conditions, SQL injection, cross site scripting, and others. WebInspect is automatically updated with known hacking techniques with each assessment performed.

AppScan - IBM's AppScan is a web application security testing tool that scans and tests for all common web application vulnerabilities. Vulnerabilities scanned for include: SQL injection, cross site scripting, buffer overflow, and others.

Fortify Program Trace Analyzer (PTA) - Fortify PTA enables QA organizations to find security vulnerabilities and leaks while conducting functional testing enabling testers to uncover security vulnerabilities in the application with no additional effort. Fortify PTA also pinpoints vulnerabilities to specific lines of code, facilitating remediation.