Chapter 1. Chapter 4

Table of Contents

Security Architecture and Design

Secure by Design - Policy Frameworks

Overview

Secure applications do not just happen – they are the result of an organization deciding that they will produce secure applications. OWASP's does not wish to force a particular approach or require an organization to pick up compliance with laws that do not affect them - every organization is different.

However, for a secure application, the following at a minimum are required:

- Organizational management which champions security
- Written information security policy properly derived from national standards
- A development methodology with adequate security checkpoints and activities
- Secure release and configuration management

Many of the controls within OWASP Guide 2.0 are influenced by requirements in national standards or control frameworks like COBIT; typically controls selected out of OWASP will satisfy relevant ISO 17799 and COBIT controls.

Organizational Commitment to Security

Organizations that have security buy-in from the highest levels will generally produce and procure applications that meet basic information security principles. This is the first of many steps along the path between ad hoc "possibly secure (but probably not)" to "pretty secure".

Organizations that do not have management buy-in, or simply do not care about security, are extraordinarily unlikely to produce secure applications. Each secure organization documents its "taste" for risk in their information security policy, thus making it easy to determine which risks will be accepted, mitigated, or assigned.

Insecure organizations simply don't know where this "taste" is set, and so when projects run by the insecure organization select controls, they will either end up implementing the wrong controls or not nearly enough controls. Rare examples have been found where every control, including a kitchen sink tealeaf strainer has been implemented, usually at huge cost.

Most organizations produce information security policies derived from ISO 17799, or if in the US, from COBIT, or occasionally both or other standards. There is no hard and fast rule for how to produce information security policies, but in general:

- If you're publicly traded in most countries, you must have an information security policy
- If you're publicly traded in the US, you must have an information security policy which is compliant with SOX requirements, which generally means COBIT controls
- If you're privately held, but have more than a few employees or coders, you probably need one
- Popular FOSS projects, which are not typical organizations, should also have an information security policy

It is perfectly fine to mix and match controls from COBIT and ISO 17799 and almost any other information security standard; rarely do they disagree on the details. The method of production is sometimes tricky – if you "need" certified policy, you will need to engage qualified firms to help you.

OWASP's Place at the Framework table

Organizations need to establish information security policy informed by relevant national legislation, industry regulation, merchant agreements, and subsidiary best practice guides, such as OWASP. As it is impossible to draw a small diagram containing all relevant laws and regulations, you should assume all of the relevant laws, standards, regulations, and guidelines are missing – you need to find out which affect your organization, customers (as applicable), and where the application is deployed.

The following diagram demonstrates where OWASP fits in (substitute your own country and its laws, regulations and standards if it does not appear):

- COBIT
- ISO 17799
- Sarbanes-Oxley

IANAL: OWASP is not a qualified source of legal advice; you should seek your own legal advice.

COBIT

COBIT is a popular risk management framework structured around four domains:

- Planning and organization
- Acquisition and implementation
- Delivery and support
- Monitoring

Each of the four domains has 13 high level objectives, such as DS5 Ensure Systems Security. Each high level objective has a number of detailed objectives, such as 5.2 Identification, Authentication, and Access. Objectives can be fulfilled in a variety of methods that are likely to be different for each organization. COBIT is typically used as a SOX control framework, or as a complement to ISO 17799 controls. OWASP does not dwell on the management and business risk aspects of COBIT. If you are implementing COBIT, OWASP is an excellent start for systems development risks and to ensure that custom and off the shelf applications comply with COBIT controls, but OWASP is not a COBIT compliance magic wand.

Where a COBIT objective is achieved with an OWASP control, you will see "COBIT XXy z.z" to help direct you to the relevant portion of COBIT control documentation. Such controls should be a part of all applications.

For more information about COBIT, please visit:

ISO 17799

ISO 17799 is a risk-based Information Security Management framework directly derived from the AS / NZS 4444 and BS 7799 standards. It is an international standard and used heavily in most organizations not in the US. Although somewhat rare, US organizations use ISO 17799 as well, particularly if they have subsidiaries outside the US. ISO 17799 dates back to the mid-1990's, and some of the control objectives reflect this age – for example calling administrative interfaces "diagnostic ports".

Organizations using ISO 17799 can use OWASP as detailed guidance when selecting and implementing a wide range of ISO 17999 controls, particularly those in the Systems Development chapter, amongst others. Where a 17799 objective is achieved with an OWASP control, you will see "ISO 17799 X.y.z" to help direct you to the relevant portion of ISO 17799. Such controls should be a part of all applications.

For more information about ISO 17799, please visit and the relevant standards bodies, such as Standards Australia ( ), Standards New Zealand ( ), or British Standards International ( ).

Sarbanes-Oxley

A primary motivator for many US organizations in adopting OWASP controls is to assist with ongoing Sarbanes-Oxley compliance. If an organization followed every control in this book, it would not grant the organization SOX compliance. The Guide can be used as a suitable control for application procurement and in-house development, as part of a wider compliance program.

However, SOX compliance is often used as necessary cover for previously resource starved IT managers to implement long neglected security controls, so it is important to understand what SOX actually requires. A summary of SOX, section 404 obtained from AICPA's web site at states:

Section 404: Management Assessment of Internal Controls

Requires each annual report of an issuer to contain an "internal control report", which shall:

1. state the responsibility of management for establishing and maintaining an adequate internal control structure and procedures for financial reporting; and

2. contain an assessment, as of the end of the issuer's fiscal year, of the effectiveness of the internal control structure and procedures of the issuer for financial reporting.

This essentially states that management must establish and maintain internal control structures and procedures, and an annual evaluation that the controls are effective. As finance is no longer conducted using double entry in ledger books, "SOX compliance" is often extended to mean IT.

The Guide can assist SOX compliance by providing effective controls for all applications, and not just for the purposes of financial reporting. It allows organizations to buy products which claim they use OWASP controls, or allow organizations to dictate to custom software houses that they must use OWASP controls to produce more secure software.

However, SOX should not be used as an excuse. SOX controls are necessary to prevent another Enron, not to buy widgets that may or may not help. All controls, whether off the shelf widgets, training, code controls, or process changes, should be selected based on measurable efficacy and ability to treat risk, and not "tick the boxes".

Development Methodology

High performing development shops have chosen a development methodology and coding standards. The choice of development methodology is not as important as simply having one.

Ad hoc development is not structured enough to produce secure applications. Organizations who wish to produce secure code all the time need to have a methodology that supports that goal. Choose the right methodology – small teams should never consider heavy weight methodologies that identify many different roles. Large teams should choose methodologies that scale to their needs.

Attributes to look for in a development methodology:

- Strong acceptance of design, testing and documentation
- Places where security controls (such as threat risk analysis, peer reviews, code reviews, etc) can be slotted in
- Works for the organization's size and maturity
- Has the potential to reduce the current error rate and improve developer productivity

Coding Standards

Methodologies are not coding standards; each shop will need to determine what to use based upon either common practice, or simply to lay down the law based upon known best practices.

Artifacts to consider:

- Architectural guidance (i.e., "web tier is not to call the database directly")
- Minimum levels of documentation required
- Mandatory testing requirements
- Minimum levels of in-code comments and preferred comment style
- Use of exception handling
- Use of flow of control blocks (e.g., "All uses of conditional flow are to use explicit statement blocks")
- Preferred variable, function, class and table method naming
- Prefer maintainable and readable code over clever or complex code

Indent style and tabbing are a holy war, and from a security perspective, they simply do not matter that much. However, it should be noted that we no longer use 80x24 terminals, so vertical space usage is not as important as it once was. Indent and tabbing can be "fixed" using automated tools or simply a style within a code editor, so do not get overly fussy on this issue.

Source Code Control

High performing software engineering requires the use of regular improvements to code, along with associated testing regimes. All code and tests must be able to be reverted and versioned.

This could be done by copying folders on a file server, but it is better performed by source code revision tools, such as Git, Subversion, CVS, SourceSafe, or ClearCase.

Why include tests in a revision? Tests for later builds do not match the tests required for earlier builds. It is vital that a test is applied to the build for which it was built.