Secure Coding.
Practical steps to defend your web apps.

Application Security

Dan McGinn-Combs

Application Security

Applications on a network are exposed directly to the end user. The interactions between the user and the application can be a vehicle for malicious activity. When an application is made publicly available, it is too late to bolt on security measures designed to protect the confidentiality, integrity, and availability of the application data. It is important that the CISSP understand how a project is planned, changed, and executed so that security is woven into the fabric of the application as it is being built.

A project is defined as a temporary endeavor that has a beginning, an end, and a well-defined outcome. For this reason, projects are often viewed as having a life cycle comprised of several phases.

Initiation is the first step of any project. During this beginning phase, the existing application or process is carefully scrutinized so that it can be used as a basis for the final application. During the *definition* phase, the scope, or the breadth, of change from the initial process to the finished product is identified. The *design* phase is when the steps required to produce the desired application are set. When the design is complete, the active *development* phase begins. After a *testing* phase, the application is deployed. After deployment, it enters an operational or *maintenance* phase during which updates can be applied periodically to keep the system current. Finally, during the *disposal* phase, the application or process is retired.

Some consider the final stage of the life cycle to be the *Postmortem Review.* During this phase, the project team goes through an organized and structured review of the overall project. The purpose is to identify areas that can be improved, streamlined, or even eliminated. This last review can have an important impact on the methodology used on the next project.

Verification and Validation

Evaluate projects during the early stages of the life cycle using two crucial elements, the design specifications and the real-world requirements. Verification is the term used when the project specifications are lined up next to the project design and the definition ensures that they match. This is also known as “doing the job right.” Validation is the term used when the design and development team step back from the specifications and take a hard look at the real-world problem they need to solve. Often, a project design can be verified completely against the specifications but the real-world problems that they need to solve is something else entirely.

Defining Security in the System Development Life Cycle

The perfect time to inject security into the development process is before and while the systems are developed. Before the design phases and during the definition phase, a layered and integrated security reference monitor, a mechanism that authorizes or rejects access, can be put into place easily. After the application is complete, bolting on security from the outside can be expensive because it can require redeveloping or reengineering the entire application.

Software development techniques such as modularity and reusability, hallmarks of object-oriented practices, provide for consistency in the development of security modules and components. These core components need to be as simple as possible. Complexity in the implementation of the reference monitor can lead to confusion and functional errors. The security modules are one area where smaller is better. A large and unwieldy authentication component can be difficult to debug. Its logic can be difficult to follow. The result can be code prone to errors either in the implementation phase or during the maintenance phase when updates are required.

During the testing phases, separation of duties is important. Those who develop the code cannot be charged with looking for security flaws. This must be done by a different group or at least by peer developers. They need to begin by defining different ways in which the code might be abused. For example, using all numbers rather than alphabetic characters in a name field or modifying hidden fields in an HTML form. Test methods and results must be rigorously documented in both the formal and informal analysis to provide a roadmap for fixing any discovered errors.

During the development phase, it is common practice to use development hooks, or back doors, for testing and quick modification of the code base. These hooks must be carefully documented, and at the end of the development phase, they must be meticulously removed.

Change Control Process and CCB

Critical to the practice of systematic changes, especially during development, is the change control process. Change (sometimes referred to as configuration) management has been defined as identifying, controlling, accounting for, and auditing changes made to the baseline trusted computing base (TCB) that includes changes to the hardware, software, and firmware. Changes are initiated in a formal way by the owner of or stakeholder in the process or system being changed. These formal change requests are then reviewed and considered by a Change Control Board (CCB) that makes the decision to move forward with a particular change or not. The CCB is a committee that typically consists of technical and leadership members who oversee the changes made to the project. Without the clearly defined lines of authority for the CCB and separation of duties for the process or project owner, there are many opportunities for ill-considered changes to make unworkable modifications to the underlying application.

Understanding Security with Development Life Cycles

There are many recommendations on how to incorporate security into the software development life cycle. At each phase of the cycle, some activities can benefit the initial process and speed the deployment of the eventual application. During the *initiation* phase, one key decision is to broadly determine what the impact a security breach can have on any organization deploying the application. This decision drives the next step in the security process, describing the security the application needs. A system with high impact on the corporation and in an environment where the threat level is high requires much more rigorous security system than one with less impact or in a less hostile environment.

During the *development* phase, additional information on the risk can be used to determine the eventual cost associated with the security needs of the application over its lifetime. In situations where the cost of the impact is found to be less or equal to the cost of the security system itself, the executive owner of the project can choose to implement other controls. These controls might take the form of limiting not only authorization, but also access to the application.

At *implementation*, the security of the system is certified. Certified carries the meaning that the security controls have been verified using well-defined and established guidelines. This gives the stakeholders a high level of confidence that the appropriate level of safeguards have been put in place. The system is also accredited during implementation. Accreditation carries the meaning that a senior organizational official has authorized the system to collect and store the data.

The longest section of the life of an application is the *operations* phase. During this phase, the organization must carefully monitor security status of the application. In addition, any changes must follow a clearly defined process to avoid unexpected modifications to the security posture of the application.

Comparison of Software Development Life Cycles

There are two primary and classic models of development. The first, the Waterfall Model, was developed during the 1970s during a time when end users did not interact directly with the computer or its programs. In this model, each of the phases of the application development life cycle is broken into a mini-project of its own and followed sequentially from one to the next. At each step along the way, a formal review is done. The phases are fully documented. They are treated almost as if each phase were to be done by a different team. The object is for each team to fully hand off the project to the next. The Waterfall Model is limited in that there is no formal way to meet changing requirements after the development has begun. Because users interact so closely with the applications today, a development team that is not flexible enough to incorporate changes cannot be successful during the implementation phase.

The Spiral Model is designed to address the shortcomings of the Waterfall Model. The Spiral Model begins with a small subset of the final requirements and produces an initial prototype. During a series of successive passes through each development phase, the application gets closer and closer to its target. During the testing portion of each spiral, the prototypes and simulations can be used to more completely analyze security issues and environment. The Spiral Model allows the development team to grow in its understanding of the operational requirements. The prototypes and simulations allow the development team to better understand the user interaction. Both of these improve the potential for catching security issues before the implementation phase. This is a more appropriate development model in situations where all the requirements are not initially known or well understood.

Works Consulted

Harris, Shon. *All-in-One CISSP Exam Guide*, Third Edition. McGraw-Hill/Osborne, 2005.

PrepLogic. *CISSP Mega Guide*. PrepLogic, 2006.

Global Knowledge. *CISSP Prep Course*. En Pointe Technologies, 2006.

Change Control Information, http://www.processimpact.com/goodies.shtml

Project, http://en.wikipedia.org/wiki/Project

Grance, Tim Joan Hash, Marc Stevens. *Security Considerations in the Information System Development Life Cycle*. NIST, http://csrc.nist.gov/publications/nistpubs/800-64/NIST-SP800-64.pdf

Laplante, Phillip, and Colin Neill. Penn State University, *Game Development* Volume 1, No. 10, February 2004, http://acmqueue.com/modules.php?name=Content&pa=printer_friendly&pid=110&page=1

Purcell, James E. *Comparison of Software Development Lifecycle Methodologies*, http://www.giac.org/resources/whitepaper/application/217.pdf

Last Updated: January 22nd, 2018

Upcoming SANS App Sec Training

Cloud Security Summit & Training 2018	San Diego, CA	Feb 19, 2018 - Feb 26, 2018	Live Event
San Francisco Spring 2018 - DEV522: Defending Web Applications Security Essentials	San Francisco, CA	Mar 12, 2018 - Apr 17, 2018	vLive
SANS San Francisco Spring 2018	San Francisco, CA	Mar 12, 2018 - Mar 17, 2018	Live Event
SANS Northern VA Spring - Tysons 2018	McLean, VA	Mar 17, 2018 - Mar 24, 2018	Live Event
SANS 2018	Orlando, FL	Apr 03, 2018 - Apr 10, 2018	Live Event
Pre-RSA® Conference Training	San Francisco, CA	Apr 11, 2018 - Apr 16, 2018	Live Event
SANS Baltimore Spring 2018	Baltimore, MD	Apr 21, 2018 - Apr 28, 2018	Live Event
Community SANS New York DEV522	New York, NY	Apr 23, 2018 - Apr 28, 2018	Community SANS
SANS Riyadh April 2018	Riyadh, Saudi Arabia	Apr 28, 2018 - May 03, 2018	Live Event
SANS Security West 2018	San Diego, CA	May 11, 2018 - May 18, 2018	Live Event
SANS Northern VA Reston Spring 2018	Reston, VA	May 20, 2018 - May 25, 2018	Live Event
SANS Oslo June 2018	Oslo, Norway	Jun 18, 2018 - Jun 23, 2018	Live Event
SANSFIRE 2018	Washington, DC	Jul 14, 2018 - Jul 21, 2018	Live Event
SANS Northern Virginia- Alexandria 2018	Alexandria, VA	Aug 13, 2018 - Aug 18, 2018	Live Event
SANS Virginia Beach 2018	Virginia Beach, VA	Aug 20, 2018 - Aug 31, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced