ISC2

Domain 8 Key Takeaways

Overall, security professionals reviewing CISSP Domain 8 need to understand the principles of software development security and the different tools and techniques that can be used to incorporate security into the SDLC. This includes understanding secure coding practices, code reviews and testing, and incorporating security into the design, deployment, and maintenance of software.

1. Software development security involves incorporating security into the software development lifecycle (SDLC) to prevent security vulnerabilities and ensure that software is secure by design.

2. The SDLC consists of several phases, including planning, design, implementation, testing, and deployment.

3. Secure coding practices, such as input validation and error handling, can help prevent common vulnerabilities, such as buffer overflows and SQL injection.

4. Code reviews and testing, such as static code analysis and dynamic testing, can help identify security vulnerabilities in software.

5. Security should be incorporated into the design and architecture of software, including the use of secure protocols and encryption.

6. Security should also be incorporated into the deployment and maintenance of software, including the use of secure configuration management and patch management.

7. Security awareness and training for developers and other stakeholders is important to ensure that security is incorporated throughout the SDLC.

SDLC and Security Controls

- Software exists at every layer of the ring and Open Systems Interconnection (OSI) models; that software often depends upon configuration parameters, scripts and initialization (or default) data values that are stored in files associated with that software. A typical applications program may consist of tens of thousands of such files of executable software, parameter data, scripts and other data.

- Each step across the cycle of building, deploying, using and ultimately retiring systems should have its set of associated processes, tools and policies that provide sufficient governance as the organization acquires or develops and then uses its range of software systems, applications, applets, widgets, bots and other types of software.

- Loosely considered, the software environment consists of all elements, activities, information, people and systems that affect the way that a software system or application is created, used, modified and ultimately disposed of.

- Any systems development process—any management methodology— has the potential to allow simple mistakes of omission or commission made at each stage in that process to turn into exploitable vulnerabilities in the software system being produced or modified.

- Senior managers and organizational leadership must consider how much software security and safety to build in up front—and how to do that. As a security professional, you have a vital opportunity to inform and advise upon these choices, and a further role to play in assuring that their implementation achieves the required security posture.

- Organizations need to choose methodologies carefully, as the model chosen should be based on the requirements of the organization. The key point is not that a formalized SDLC needs to be used, but that the entire development process needs to involve security. The best security is always what is designed into the system, not what is added later.

- The choice of one methodology over another may ultimately be driven by management's perception of a window of competitive advantage, or a window of accentuated risk and liability, rather than on other principles. Security professionals and systems developers alike need to be sensitive to this nonstop shift in emphasis.

- Organizations in many industries use maturity models to understand their current processes, assess opportunities for improving them and set targets (plans and schedules) for those improvements. Effective management of this journey toward more mature processes is a measurement strategy.

- The objective of information security is to make sure that the system and its resources are available when needed, that the integrity of the data processing and the data itself is ensured, and that the information's confidentiality, integrity, and availability is protected throughout its life cycle. All of these requirements rely upon secure, consistent, reliable, and properly operating application software. In addition, the integrity of both application and OS software itself must be maintained in terms of both change control and attack from malicious software such as viruses.

Assess the Effectiveness of Security and Impact of Software

- Secure software cannot be kept secure if at any point in its development pipeline its specifications, designs or source code can be tampered with, or its builds and controls scripts, configuration data and other process information modified to allow for the substitution of foreign, untrustworthy code.

- Change management and control must first decide at what level of granularity the elements of the system will be defined, enumerated, managed, and controlled. This establishes the set of configuration items that are formally subjected to configuration-management decision making and configuration-control enforcement and audit.

- There are four commonly used types of software systems security risk assessment:

    - Certification and accreditation
    - Risk management frameworks
    - Software process capabilities maturity models (CMMs)
    - Software quality assurance or software assurance

- Ongoing security assessment and evaluation testing evaluates whether the same system that passed acceptance testing is still secure in the face of today's evolved threat landscape.

- Four basic phases of activity shape the ways in which organizations acquire software systems via third-party systems providers, integrators or consultants:

    - Planning
    - Contracting
    - Monitoring, acceptance and deployment
    - Ongoing use and support

Define and Apply Secure Coding Guidelines and Standards

- Software vulnerabilities are self-induced by designers, programmers, those who deploy and those who use the software. As a security professional, you do not have to become a software engineer in order to help your organization develop more secure software systems and keep them safe and secure. But before you can offer effective support to help improve the software security posture of your organization, you will need a working knowledge of some of the fundamentals of software and how it is built and deployed.

- Organizations can mandate through policy that software designers and developers apply these coding standards during software development to create systems and applications that address proper security requirements based on the organization's needs. There are several secure coding guidelines and standards that have been developed by groups and industries to address this requirement.

- Software-defined security (SDS) is a major paradigm shift in the ways in which we think about information systems architectures and keeping them secure. It is the difference between the client-server-based model using an on-premises server and a fully cloud-based model with no on-premises architecture beyond its internet point of presence and local access points.

- Several things are already clearly established as part of the superset of functions and ideas now being thought of as SDS:
    - Identity management is absolutely vital to the success of any security effort.
    - Entities, not just users, must be the level at which identity is defined, authenticated, managed, and tracked.
    - Defense has become more proactive, which means that defenders must learn as fast as attackers. Every member (human and nonhuman) of an organization's security team, and the organization itself, must achieve greater fluency and proficiency with cybersecurity, information assurance, and information asset protection.
    - Systems and solutions must be agile, flexible, and responsive.
    - Access management has to be pushed down to the lowest level asset, and server-level or system-level access cannot be available to routine users anymore.
    - Zero trust architecture (ZTA) concepts both enable SDS solutions and are required by them.