TechTarget | SearchSecurity

E-guide

PRO+ Content

CISSP Prep: Secure Software Development

In this e-guide

In this e-guide:

The Certified Information Systems Security Professional (CISSP) is an information security certification that was developed by the International Information Systems Security Certification Consortium, also known as (ISC)².

The CISSP exam covers 8 individual subject areas, which are referred to as domains. The 8 domains make up (ISC)² 's Common Body of Knowledge (CBK), which is a framework and collection of information security best practices, methodologies, technologies and concepts.

SearchSecurity partnered with Logical Security and expert Shon Harris to create the CISSP Essentials Security School.

This school offers free training that covers critical topics in each of these 8 domains to help practitioners prepare for the 6 hour exam which asks 250 questions.

In this CISSP training guide we take a deeper dive into the Software Development Security domain. Inside, Shon Harris

In this e-guide

explains the core concepts in the Software Development Security domain to help you prepare for this important area of the CISSP exam.

Topics covered include:

- **Secure software development processes**
- **Programming languages and distributed computing**
- **Database system security issues**
- **Software security threats and countermeasures**

Plus, expert Michael Cobb sheds light on how you can incorporate application security into short development cycles.

Additionally, test your knowledge on this topic area at the end of this guide with a quick quiz.

In this e-guide

CISSP online training: Software Development Security domain

Shon Harris , Contributor - Logical Security

Most companies rely upon controls such as firewalls, intrusion detection systems, content filtering, antimalware software, vulnerability scanners and other network technologies to solve security problems. This reliance on a long laundry list of controls occurs mainly because software contains many vulnerabilities that put its users at risk. Enterprise environments are sometimes referred to as "hard and crunchy on the outside and soft and chewy on the inside;" meaning the network perimeter security may be fortified, but internal software programs are easy to exploit once access has been obtained.

The best approach to dealing with software issues is to set up software development security processes in the first place. Unfortunately, software programs are usually developed for functionality first, not security. However, it would be far more effective to build security into every piece of software from the outset rather than "bolt it on" afterward.

In this spotlight article for the Software Development Security domain of the Certified Information Systems Security Professional (CISSP) exam, I will discuss how software programs are structured; what security mechanisms and strategies are commonly used to secure data during access, processing

In this e-guide

and storage; and the common threats and countermeasures of software development security. Topics covered will include:

- Software development security: The models, methods, lifecycle phases and management of the development process.
- Programming languages and distributed computing: Software architecture, programming languages and concepts, change control methods, improvement models, data modeling and structures, data interface and exchange methods.
- Database systems: Models, management systems, query languages, components, data warehousing and mining, schema and security measures.
- Security threats and countermeasures: Common threats to applications and systems, and how expert systems and artificial neural networks can be applied to mitigate threats.

Software development security organizations

Since software is the closest to the data that a company is responsible for protecting, there are many initiatives and efforts going on to increase the use of secure software development processes. There are also many groups and organizations that provide best practices in secure software development to help organizations achieve this protection.

The Web Application Security Consortium (WASC) is an organization that provides best practice security standards for the World Wide Web and the Web-based software that makes it up.

In this e-guide

The Open Web Application Security Project (OWASP) is another organization that deals specifically with Web security issues. This group provides software development security guidelines, testing procedures and code review steps, and it maintains The OWASP Top Ten, a list of the greatest Web application security risks facing enterprises today.

The ISO/IEC 27034 standard provides best practices for secure software development that aligns with ISO/IEC's Information Security Management System model and ISO/IEC 27000 series. This standard provides an application security overview and concepts, organization normative framework, application security management process, application security validation and security guidance for specific applications.

The Department of Homeland Security has a Software Assurance Program that maintains an initiative called *Build Security In*, or BSI. This program provides best practices, tools, guidelines, rules, principles and other resources that software developers, architects and security practitioners can use to build security into every phase of software development.

MITRE has the Common Weakness Enumeration (CWE) standard initiative that maintains the top most-dangerous software errors. CWE provides a common language and taxonomy for software development security issues and details vulnerabilities found in programming code, product design and system architecture. NIST has mapped these CWEs with its National Vulnerability Database (NVD), which is the U.S. government repository of standards-based vulnerability management data.

In this e-guide

Secure software development

Determining the appropriate level of security for a particular system is a difficult judgment call, and it depends on many factors, including the trust level of the operating environment, the security levels of the systems it will connect to, who will be using the system, the sensitivity of the data, how critical the functions are to the business and how costly it will be to apply optimal security measures. Understanding the processes and economics of system development is essential to comprehending why few systems used in production today can be considered sufficiently secure.

This section of the Secure Software Development domain covers how different environments demand different types of security, the importance of addressing failure states and the difficulty of balancing both security and functionality demands to meet business needs.

An overview of the history of system building and software development helps demonstrate why yesterday's approaches are no longer adequate in today's super-connected world, proving that the increasing complexity of modern environments and technology rules out a "one-size-fits-all" security approach.

In this e-guide

The system development lifecycle

Every system has its own developmental lifecycle, which comprises the following phases: initiation, acquisition/development, implementation, operation/maintenance and disposal. Collectively these are referred to as the system development lifecycle (SDLC).

Each SDLC phase has specific goals and requirements; this domain focuses on these specific security goals and requirements and how they should be integrated into an SDLC model. Some of the SDLC models covered in this domain include:

- Waterfall -- A sequential approach that requires each phase to complete before the next one can begin. Difficult to integrate changes, inflexible model.
- V-Model -- This model emphasizes verification and validation at each phase and requires testing to take place throughout the project, not just at the end.
- Incremental -- Multiple development cycles are carried out on a piece of software throughout its development stages. Each phase provides a usable version of software.
- Spiral --This is an iterative approach that emphasizes risk analysis per iteration. It allows for customer feedback to be integrated through a flexible evolutionary approach.

In this e-guide

- Rapid application development -- This model combines prototyping and iterative development procedures with the goal of accelerating the software development process.
- Agile -- In this model, iterative and incremental development processes that encourage team-based collaboration are used. Flexibility and adaptability are used instead of a strict process structure.

Balancing security with functionality

Open and distributed environments may be using both legacy and newer technology, as well as intranets and business partner extranets, all while maintaining marketing presence on the Internet for e-commerce purposes -- which in an entirety presents many security challenges. Yet strategies are being developed to better protect such systems by layering security controls at different technology levels. Being the last bastion of defense, security controls applied at the system and application level, however, should be as rigorous as possible

This section of the domain goes into the challenges and pressures software vendors face that commonly result in a lack of security built into the software products. Individuals preparing for the CISSP exam will gain insight into the decision-making process and the perils of relying too heavily on environment-based security devices and appliances rather than building the right level of security into a product from the start.

In this e-guide

Most commercial applications have security controls built in, though only recently have vendors begun to set security on by default. This has forced users to make deliberate risk decisions to lower their security protection from the level the vendor recommends. While these approaches may prove annoying to the user at first, the increasing worldwide threat level necessitates not only an increased level of accountability from commercial vendors but also an increased level of awareness and responsibility on the part of the user.

Unfortunately, the economics of building secure software can be a tradeoff between the security and functionality of systems. While many in the industry voice concerns over the insecurity of today's software products, the customer requirement of functionality constantly overrides security needs. However, as attackers become more sophisticated in their methods, the industry as a whole is obligated to seek out new ways to reveal system vulnerabilities that result from uncommon conditions and to trap these threats so they won't be available for malicious use.

Securely built software programs depend on our ability to elevate the visibility and priority of security throughout each phase of the development process. As early as the project initiation phase, we can begin to formulate security goals based on business needs, liability risks and investment constraints. Throughout the requirements and design phase, we can systematically uncover hidden functional and architectural flaws that could compromise security. We can then apply inspection methods and automation during the construction and testing phase to root out coding flaws or failure conditions that are known to be vectors for security attacks. At every decision point, risk analysis should guide decision makers about the

In this e-guide

risks they will need to accept as a tradeoff for lower prices, faster time to market, increased functionality or improved usability. By using operational checklists for installation and administration, and by applying rigorous change control methods, software vendors can be sure that their products will meet both user needs and enterprise security standards now and in the future.

Vulnerability identification processes should be built into the application development process, including:

- Attack surface analysis: a process to identify and reduce the amount of code accessible to untrusted users.
- Threat modeling: a systematic approach used to understand how different threats could be realized and how a successful compromise could take place.
- Static analysis: a debugging technique that is carried out by examining the code without executing the program and therefore is carried out before the program is compiled.
- Fuzzing: a technique used to discover flaws and vulnerabilities in software.

The Capability Maturity Model Integration (CMMI) is a process-improvement model that provides a pathway for incremental software development improvement. The model uses five maturity levels designated by the numbers 1 through 5, each representing the maturity level of the process quality and optimization: 1 = Initial, 2 = Managed, 3 = Defined, 4 = Quantitatively Managed, 5 = Optimizing.

In this e-guide

The CMMI addresses the different phases of a software development lifecycle, including concept definition, requirements analysis, design, development, integration, installation, operations and maintenance, as well as what should happen in each phase. It can be used to evaluate security engineering practices and identify the ways to improve them. It can also be used by customers evaluating a software vendor. In the best of both worlds, software vendors would use the model to help improve their processes and customers would use the model to assess the vendors' practices.

Programming languages and distributed computing

After a brief overview of programming development, this domain of the CISSP centers on object-oriented programming, its encapsulation of code chunks as class objects and how those objects can be altered and reused. In creating application designs, the use of data by the proposed application is modeled and the data paths it will take through the application are analyzed. This domain is concerned about the atomicity of objects -- their cohesion and coupling properties -- as it will drive the ease with which they can be safely updated.

Software programming languages have evolved over time and are broken down into the following generations:

- Generation one: Machine language
- Generation two: Assembly language
- Generation three: High-level language
- Generation four: Very high-level language

In this e-guide

- Generation five: Natural language

Each generation increases the capabilities of the programming languages that make it up, and each programming language has its own security issues that security professionals must be aware of.
This domain covers how data is imported and exported from the application. The usefulness of standards and technologies that ensure component communication (COM, DCOM), the seamless exchange of data between disparate systems (ORB, CORBA, ODBC, DDE), and the presentation or access to data outside the native application (OLE) are covered in this domain, as are the security issues surrounding the use of each.

This domain also covers service-oriented architectures (SOAs), which provide standardized access to the most needed services to many different applications at one time. Services within an SOA are usually provided through Web services, which allow Web-based communication to happen seamlessly using Web-based standards such as Simple Object Access Protocol (SOAP), HTTP, Web Services Description Language (WSDL), Universal Description, Discovery and Integration (UDDI), and Extensible Markup Language (XML).

Web security issues -- such as server-side includes, client-side validation, cross-site scripting and parameter validation -- are covered, along with countermeasures for each vulnerability type.

In this e-guide

Database technology

Databases hold the data needed to conduct business, guide business strategies and prove business performance history. In this domain, database management software is covered, along with an overview of different types of database models. The database model defines the relationships between different data elements, dictates how data can be accessed, and defines acceptable operations, the type of integrity offered and how the data is organized. A model also provides a formal method of representing data in a conceptual form and delivers the necessary means of manipulating the data held within the database. Databases come in several types of models, including:

- Relational
- Hierarchical
- Network
- Object-oriented
- Object-relational

Relational databases are covered in depth, including how a schema is represented and used in the data dictionary, how it applies to security, how primary and foreign keys are related, how checkpoints and save points work, and how maintaining the integrity of a data set is essential to ensuring that no data falls outside the schema or the security controls built into the schema.

In this e-guide

Data is useless if it can't be accessed and used; applications need to be able to obtain and interact with the information stored in databases. They also need some type of interface and communication mechanism. This domain addresses some of these interface languages:

- Open Database Connectivity (ODBC)
- Object Linking and Embedding Database (OLE DB)
- ActiveX Data Objects (ADO)
- Java Database Connectivity (JDBC)

Database security issues are covered in this domain of the CISSP, including concurrency protection, rollback capabilities, two-phase commits, checkpoints, and aggregation and inference protections. Secure and stable databases provide the ACID characteristics:

- Atomicity -- All changes to the database take effect or none do.
- Consistency -- Transaction must meet defined integrity constraints.
- Isolation -- No users or processes will be able to view a transaction until it completes properly.
- Durability -- When a transaction takes place, it is permanent.

Data warehouses (aggregators of disparate data sets) and data marts (copies of subsets of data warehouses) pose similar challenges, but the effort and cost that goes into these systems makes the metadata they yield very valuable to businesses, which warrants a correspondingly high level of protection.

In this e-guide

Strategies for administering data systems for optimal security are also discussed. Topics include how to use security views to enforce security policies, content- and context-driven access control strategies, the challenges presented by aggregation and inference attacks, and the use of diversionary tactics (e.g., cell suppression, noise and perturbation).

Security threats and countermeasures

In this section of the CISSP, exam preparation includes an overview of the most common threat attacks affecting or engaging applications and systems as well as how they are executed. These include denial of service, timing attacks, viruses, worms, Trojan horses, rootkits and crimeware, among others.

Advanced systems employing artificial intelligence such as expert systems and artificial neural networks can aid in revealing connections between disparate pieces of information and recognizing anomalous patterns in network traffic or application behaviors that might signal an attack in progress.

CISSP® is a registered certification mark of the International Information Systems Security Certification Consortium, Inc., also known as (ISC)2.

Next article

In this e-guide

Application development security requires forethought

Michael Cobb, CISSP-ISSAP and IT security author

To stay ahead of the competition, enterprises need to be able to turn an Internet-based business idea into a fully functioning service in double-quick time. The pressure to get software applications ready for market in the shortest time possible has spawned numerous application development philosophies, ranging from sequential design, such as the Waterfall model, to SWAT Team and Agile software development process models. Proponents of each different method hail them as revolutionizing the software application development process, and many development teams rush to embrace the latest time-saving approach in the hope of being able to finally meet delivery deadlines.

Sadly, none of these methodologies really have software security testing at their core, though there are various initiatives and technologies that do aim to reduce the number of security flaws that make it into the final release. SAST, or static application security testing, has been around since 1999, and dynamic application security testing (DAST) and interactive application security testing (IAST) are also widely used by teams that are conscious about application development security. Microsoft possibly initiated the biggest change to application development security with its presentation in 2004 of its Trustworthy Computing Security Development Lifecycle, which puts secure coding at the heart of software creation while aiming to reduce

In this e-guide

development costs. However, given the daily announcement of successful attacks against applications developed and managed by the world's biggest businesses, it's clear that real-life methods for developing robust and secure applications are still failing.

RASP closes app security testing gap

This may be why a very different tactic to tackling application development security is attracting a growing number of followers. RASP, or runtime application self-protection, was first championed by Joseph Feiman in 2012. It aims to close the gap left by application security testing and network perimeter controls, neither of which have enough insight into real-time data and event flows to either prevent vulnerabilities slipping through the review process or block new threats that were unforeseen during development.

RASP technology is built or linked into an application or its runtime environment so that it has the capability to control application execution. For example, HP's Application Defender monitors an application's API calls to common core libraries. This enables it to analyze application flow and spot potentially malicious events, such as cross-site scripting and code injection. Arxan Technologies offers RASP features for Java applications. Web application firewalls are also dedicated application protection technologies and go some way towards providing context when filtering application requests, such as blocking access to certain functions at certain times of day, but unlike RASP technologies they don't have the benefit of context from within the application.

In this e-guide

This extra insight means RASP-protected applications rely less on external devices like firewalls to provide runtime security protection; so if perimeter defenses fail to spot malicious traffic and the development team failed to implement proper validation checks on data input by a user, the application can still protect itself. For example, only by seeing a complete database query, constructed within the application, can it be accurately determined if it's legitimate or malicious. Enabling an application to continuously run security checks on itself and respond to live attacks provides a further level of protection, as it is can terminate a malicious user's session if an attack is detected and alert monitoring systems to what has happened.

If RASP technologies can really deliver "self-protection," then applications sitting on the Internet will be far more robust and able to repel attacks. My concern, though, is that when new technologies or methodologies come along, development teams tend to let slip application development security best practices, relying instead on their new solution to do security for them. Commercial pressures mean developers are made to focus on frequent releases in short development cycles in order to introduce new features more quickly. This means security is often overlooked even though integrating security into the software development lifecycle has been shown to result in more secure software.

Define security requirements at the beginning

Whichever method of software design and testing is used, developers should not depend on an elaborate network of defenses and other security controls to cover for poor practices. Defining minimum security

In this e-guide

requirements for a project during the design and architecture stages is essential, as it allows developers to plan and integrate security controls far more effectively than trying to bolt them on as an afterthought. Development teams must understand the concepts of secure design and topics such as threat modeling, secure coding and security testing, and have clear policies and standards to follow. Security must be seen by developers as a feature that will be tested just like any other feature or requirement. SANS has long maintained that one of the primary causes of computer security vulnerability is "assigning untrained people to maintain security and providing neither the training nor the time to make it possible to learn and do the job." Code will never be 100% bug-free, but the same vulnerabilities should not keep appearing on the OWASP Top 10 and SANS Top 25 dangerous programing errors.

Implementing a structured software development lifecycle for software developers to work with will provide assurance that code has passed inspection and testing, and ensure controls are validated before being used in a production environment. Using methods such as static and dynamic code analysis throughout the development process can substantially reduce the time spent looking for bugs, improve developer efficiency and cut costs spent on dealing with flaws later on. Scanners and code reviews won't find every bug but can dramatically improve the overall security of application. Early and regular testing not only reduces the overall cost of software development and maintenance but improves product reliability, profits and reputation.

Adding RASP as another layer of security to protect live applications certainly makes sense, but there are no quick fixes when it comes to

In this e-guide

security. One approach alone will never be sufficient, so don't forget the importance of maintaining application development security best practices when management rushes to embrace the latest acronym in the pursuit of a faster approach.

Next article

In this e-guide

Software Development Security CISSP Quiz: Test Your Knowledge

If you're planning on getting your CISSP certification, make sure to test your knowledge on the Software Development Security domain.

Start Quiz

This quiz is part of SearchSecurity.com's CISSP Essentials Security School lesson, Domain 6, Application and System Development. This exclusive quiz offers free prep questions similar to those on the real CISSP exam. For additional resources, visit our CISSP Essentials Security School.

Next article

In this e-guide

About SearchSecurity

IT security pros turn to SearchSecurity.com for the information they require to keep their corporate data, systems and assets secure.

We're the only information resource that provides immediate access to breaking industry news, virus alerts, new hacker threats and attacks, security certification training resources, security standard compliance, webcasts, white papers, podcasts, Security Schools, a selection of highly focused security newsletters and more -- all at no cost.

For further reading, visit us at http://SearchSecurity.com/

Images; Fotalia