Cybersecurity

Architecture and Design

2.3.3 Secure Coding Techniques

What are different secure coding techniques?

Overview
The student will summarize secure application development, deployment, and automation concepts.

Grade Level(s)
10, 11, 12

Cyber Connections

- **Threats & Vulnerabilities**
- **Networks & Internet**
- **Hardware & Software**

This content is based upon work supported by the US Department of Homeland Security's Cybersecurity & Infrastructure Security Agency under the Cybersecurity Education Training and Assistance Program (CETAP).

Copyright © 2023 Cyber Innovation Center
All Rights Reserved. Not for Distribution.

Teacher Notes:

CompTIA SY0-601 Security+ Objectives

Objective 2.3

- Summarize secure application development, deployment, and automation concepts.
    - Secure coding techniques
        - Normalization
        - Stored procedures
        - Obfuscation/camouflage
        - Code reuse/dead code
        - Server-side vs. client-side execution and validation
        - Memory management
        - Use of third-party libraries and software development kits (SDKs)
        - Data exposure

Secure Coding Techniques

What's Normal

The first step in validation is ***normalization***, which is simply checking if answers are normal. For example, if asked for a first name and you put X Æ A-12, that is not normal (however, that is the name of Elon Musk's child).

Stored Procedures

If a user is able to modify SQL requests, they may gain access to database information that they should not have access to. To counter this, developers will implement ***stored procedures***. Stored procedures are defined functions that are stored in a database engine. These procedures can be used with input validation without altering the database itself. Instead of making a complex request to the database, you use a single call command with the name of the stored procedure and that is the only way of getting data from the database.

Boufcstaoin

Obfuscation is the art of making something obscure, unclear, or unintelligible. You take source codes or scripts that are easy to follow and turn them into something difficult to follow. This hides the logic behind the way the script and source code operate as well as hide potential security holes. This does not change what the code does.

Copyright © 2023 Cyber Innovation Center
All Rights Reserved. Not for Distribution.

Teacher Notes:

Dead Code

A lot of developers will reuse code from one application to another (it is not just Disney reusing the same scenes from one movie to the next). If there is a common process occurring in two applications, it makes sense to copy the code from one application to another. Obviously, whatever security issues the original code has will be transferred with the copy.

Another challenge with reusing code is what is known as ***dead code***. With dead code, your application runs an executable, makes some calculations, and determines results of that calculation, but you never use the results of that calculation. Essentially, you have spent the time performing a calculation to then not use it; therefore it is dead code. Since the introduction of any code has the potential to create security problems, you can make your application more secure by removing the dead code.

UFC 123456 Server vs. Client

When you need to validate data in an application, you can perform the validation on the server or on the client. With ***server-side validation***, all your checks of the data occur on the server. If malicious users are not using the front end that you would usually use for your application, you are still going to check and maintain the validation once the data arrives at the server. With ***client-side validation***, you can provide a way to filter out legitimate users from illegitimate users and speed up the validation process since all the checks are taking place on the user's local computer. You can increase security by validating on both the server side and the client side.

128 GB 4000 MHz DDR4 Trident Z

An application and its data execute in the memory of your computer. As a developer, you need to monitor how memory is used by your application. ***Memory manipulation*** is a good way to gain access to parts of an application or OS that you should not have access to. Users cannot be trusted, so you always want to validate inputs into your application to make sure no one tries to manipulate the memory.

A good example of a memory vulnerability is ***buffer overflow***. Attackers exploit buffer overflow issues by overwriting the memory of an application. This changes the execution path of the program, triggering a response that damages files or exposes private information.

Copyright © 2023 Cyber Innovation Center
All Rights Reserved. Not for Distribution.

Teacher Notes:

SDK's

If you have written in an application programming language, you understand the language itself does a large amount of what you need for the creation of the application. However, sometimes the language does not provide you with everything you need. In these cases, you can use a third-party library or ***software developers kit*** (SDK) to enhance or add capabilities to the programming language.

These are used all the time because they dramatically speed up the programming process, but there is the potential for security risks because someone else wrote the code. Test, test, test, so you can verify the security of the third-party library.

Data, Exposed

Data exposure is losing control of data during operations. You must always protect sensitive data whether you have ***data at rest*** (storage), ***data in transit*** (being communicated), or ***data in use*** (being used). If exposed, not only have you released confidential information, but you will lose trust and could seriously damage your organization's reputation.

Copyright © 2023 Cyber Innovation Center
All Rights Reserved. Not for Distribution.