10.4.7 Application Development Security Facts

In our world today, information is exchanged constantly. This means that attackers are working relentlessly to access our data. It is essential that we begin security efforts at the coding level.

This lesson covers the following topics:

- Secure coding concepts

Secure Coding Concepts

Secure coding concepts include the following:

Concept	Description
Normalization	Normalization is data reorganized in a relational database with the intent to eliminate redundancy by having all related data stored in one place. Normalization: ▪ Increases performance by reducing disk space ▪ Provides quick and efficient access to manipulate the data ▪ Lowers the risk of exploitation
Stored procedures	Stored procedures are one or more database statements stored as a group in a database's data dictionary. When called, these procedures execute all the statements in the collection. Stored procedures: ▪ Centralize the code and eliminate the need to reproduce it ▪ Keep calling program rules consistent across programs ▪ Protect the code from users by allowing the user to call a stored procedure without seeing the actual code ▪ Limit injection attacks
Code obfuscation/code camouflage	Obfuscation is the deliberate act of creating source or machine code that is difficult for humans to understand. In other words, the code is camouflaged. ▪ Programmers use roundabout expressions to compose statements that deliberately obfuscate code to conceal its purpose or its logic. ▪ They use implicit values embedded in it to prevent tampering, deter reverse engineering, or as a puzzle or recreational challenge for someone reading the source code. ▪ This is usually done with an automated tool but can also be done manually. ▪ There are several methods, but most can be reverse engineered.

Code reuse	Code reuse is simply using the same code multiple times. Reusing code is a good idea if the programmer writes the same code at least three times. Code reuse:
	▪ Can create a shared library for others that use the same code
	▪ Can be a security problem if:
	  ▪ The code is not secure before it is shared and used multiple times
	  ▪ It has changes made to it to fit a new use, but the changes aren't secure
	▪ Be sure to comprehensively test code before allowing reuse

Dead code	

	▪ Sometimes dead code refers to code that is non-executable at runtime ▪ Sometimes it means source code in a program that is executed but is not used in any other computation, making it obsolete. ▪ Remove any dead code from your application for security ▪ If it doesn't exist, it can't be exploited

Memory management	Memory management is a resource-management process applied to computer memory. It allows your computer system to assign portions of memory called blocks to various running programs that optimize overall system performance.
	
	Memory management resides in the hardware, the operating system, programs, and applications. In the hardware, memory management involves components that physically store data, such as RAM chips, memory caches, and SSDs. In the OS, memory management involves the allocation of specific memory blocks to individual programs as user demands change. At the Application level, memory management ensures the availability of adequate memory for the objects and data structures of each running program at all times.
	
	When the program requests a block of memory, the allocator in the memory manager assigns that block to the program. When a program no longer needs the data in the previously allocated memory blocks, those blocks become available for reassignment. This task can be done automatically by the memory manager or manually by the programmer.
	
	▪ Most common memory vulnerabilities:
	  ▪ Size of input in buffer copy not checked
	  ▪ Buffer size calculated incorrectly
	  ▪ Format string not controlled
	▪ To prevent vulnerabilities:
	  ▪ Limit the amount of characters read into the buffer
	  ▪ Define constants for the size argument
	  ▪ Do not allow user input in format strings

Third-party libraries and software development kits (SDKs)	A third-party library is a library where the code is not maintained in house. A software development kit (SDK) is a set of software development tools that can be installed as one unit. Both can provide code frameworks or code snippets to help development go faster. Though they can be very helpful, there are risks involved. For example: ▪ Anytime code comes from an outside source there is risk that it may contain flaws and vulnerabilities ▪ Sometimes code comes in bundles, giving developers more code than they need ▪ Extra code can create extra opportunity for exploitation ▪ SDKs are often open-source and, as such, there may be no urgency to fix bugs Be sure to test code from third-party libraries and SDKs for functionality and security issues.
Sensitive data exposure	Sensitive data exposure involves unintended exposure of personal and confidential data. This can come from: ▪ Weak or missing encryption ▪ Coding flaws

	▪ Misapplied data uploads in a database To mitigate sensitive data exposure: ▪ Encrypt data in transit and at rest using cryptographic algorithms and keys ▪ Disable caching on forms that collect data ▪ Implement hashed and salted passwords

Fuzz testing	Fuzz testing (also known as fuzzing) is a software testing technique that exposes security problems by providing invalid, unexpected, or random data to the inputs of an application. Fuzzing program types are:
	
	▪ Mutation-based
	  ▪ Mutate existing data samples to create data
	▪ Generation-based
	  ▪ Define new test data based on models of the input

Code signing	Code signing is the process of digitally signing (encrypting) executables and scripts to confirm the software author and guarantee that the code has not been altered or corrupted since it was signed. The process employs the use of a cryptographic hash to validate authenticity and integrity. Code signing: ▪ Provides security when deployed ▪ Helps prevent namespace conflicts in some programming languages ▪ Provides a digital signature mechanism to verify the identity of the author or build system ▪ Provides a checksum to verify that the object has not been modified ▪ Provides versioning information about an object, or is used to store other metadata about an object