C++ ile Güvenli Kodlama Eğitimi

Eğitim Hakkında

The C++ Secure Coding training course is designed to introduce programmers to the vulnerabilities that creep into these applications and how to defend against them.

The course will start by exploring how security relates to applications and then jumps right into imagining what can go wrong at any point during the program execution. These issues are addressed by exploring common coding vulnerabilities that occur during software development, that the programmer may or may not be aware of. Next, the course explores the results of vulnerabilities, and protecting against them is reinforced by the hands-on labs. Specific issues surrounding cryptography, client authentication, and overflow conditions will be addressed. The course concludes with a lesson on how the application of object-oriented design principles, the CERT, and security design principles are addressed, as well as how the computer architecture and operating system architecture help and sometimes fail to protect applications.

Neler Öğreneceksiniz

- C/C++ programming bugs
- Protection principles
- Input validation
- Improper error and exception handling
- Buffer overflow
- Stack overflow
- Heap overflow
- Protection against stack overflow
- Address Space Layout Randomization (ASLR)
- Secure coding sources

Eğitim İçeriği

Security

- Types of attacks: denial of service and data mining
- Vectors of attack: network, libraries, malware
- Defense in depth
- Classification of security flaws

- What Could Possibly Go Wrong?
    - Always ask: what happens if this fails?
    - What happens if the application crashes?
    - What happens if an exception is thrown?
    - Network problems?
    - Operating system crashes?
    - Protections failure (firewall, physical security, etc)
    - What about programs launched from the application?
    - Where does the application fail to?
    - Fail securely

Coding Vulnerabilities

- Input validation: XML injection, SQL injection, path traversal, log forging
- Race Conditions: time-of-check to time-of-use. memory corruption
- Time and state
- Variable parameters
- Error and exception handling
- Automatic and controlled data conversions
- Memory locking, threads, and semaphores
- File Handling
- Cryptography
    - Symmetric-key
    - Asymmetric-key
    - Hashing
    - The dependency of randomization
    - Password and key management

- Passwords and keys in memory

Client Authentication

- Web – basic
- Web – digest
- Biometrics
- Cryptographic
- Two-factor authentication
- Data Overflow
  - Buffer overflow
  - Array indexing
  - Stack overflow & Stack smashing
  - Overflow and index on the heap and the stack

Security Design Principles

- Fail-safes
- Mediation: did the data change since last checked?
- Separation of privileges
- Least privilege
- Psychological Acceptability
- CERT and Design Principles
  - CERT C++ coding standards
  - Addressing CERT requirements
  - Object-oriented design principles and design patterns
  - Testing, unit testing, and test-driven-development

Intel Architecture

- Processors, registers, memory
- Function calling conventions
- Stack frame & non-executable (NX) memory areas
- Recursion
- Address space layout randomization

Third-Party Code

- Any code that is not your own, including other internal groups
- Package management
- Vetting third-party code: source, reverse compilers
- Monitoring network connections