Fundamentals of Cyber Systems Test and Evaluation

COURSE DESCRIPTION

SecureNinja's (5) Fundamentals of Cyber Systems Test and Evaluation course is heavily focused on teaching attendees the basics of testing and evaluation in the cyber world. Custom workshops help attendees understand cybersecurity vulnerabilities, design test plans, and log results.

WHO WOULD BENEFIT

Engineers, Technicians, Managers and anyone working with cybersecurity in the DoD.

COURSE LENGTH

- 5 Days
- 40 Hours

FOLLOW ON COURSES

- Exploit Development Boot Camp
- Advanced Systems & Applications Attack & Defense

COURSE DETAILS

1. Buffer Overflows and Code Injections

• Stack Overflows attacks

• Heap overflows attacks

• Array indexing attacks

• Format strings attacks

• Unsafe API's

• Safer API's

• Stack guards

• Compiler checks

• Better ways to manipulate strings and buffers.

2. Integer Overflows

• Int / Double overflows

- Integer conversion rules
- Signed and unsigned problems
- Safe integer usage
- Enforcing limits on integer values
- Preventing lost or misinterpreted data due to conversion
- Using secure integer libraries

3. Safe API

- Dangerous and banned APIs
- Real-World Risks
- Using safe API’s
- The ‘n’ Functions
- Detecting Dangerous APIs
- Alternatives
- StrSafe

4. Secure Memory Usage

- Secure memory handling
- Erasing Data
- Secure pointer usage
- Memory Dumps
- Use smart pointers for resource management
- Ensure pointer arithmetic
- Avoid null pointer dereferencing
- Ensure sensitive data is not paged to disk

Hands On Lab.

5. Input Validation

• What is Input?

• Common Errors - Unbounded string copies, Null-termination errors, Truncation, Write outside array bounds, Off-by-one errors,

Improper data sanitization

• Black List VS. White List Validation

• ATTACK SCENARIO: Canonicalization

• String Manipulation and Comparison

• Data Type Conversion

• Regular Expressions

• Validation practices - Validating format strings, Validating buffer input, Validating filenames & URLs, Validating emails

6. Secure File Handling

• Directory Traversal attacks

• File canonicalization attacks

• Creating files with correct ACLs

• Ensure files are closed when no longer needed

• Insecure usage of shared directories

7. Application Denial of Service vulnerabilities

• Application / OS crash

• CPU starvation Memory starvation

• File system starvation

• Resource starvation

• Triggering high network bandwidth

• User level DOS

• Exploiting a specific vulnerability to cause DoS

Hands on lab

8. Network Security

• Introduction to Networking

• Network attacks

• Insecure Services

• Application Layer Threats and attacks

• Traffic Sniffing

• Traffic Manipulation

• Man-in-the-Middle

• Avoiding Server Socket Hijacking

• Firewall Friendly Application

9. Encryption in C/C++

• Introduction to cryptography

• ATTACK SCENARIO: Weak Encryption

• Symmetric encryption

• Asymmetric encryption

• Transport Level Encryption

• Storage Level Encryption

• Cryptographic API's – CryptoApi, DPAPI, Crypro++

10. Authentication & Authorization

• Authentication scenarios

• Common mistakes

• Attack scenario: brute force

• Authentication protocols

• Attack scenario: weak passwords

• Authorization models

- Access Control List (ACL)
- Role Based Access Control (RBAC)
- Attack scenario: exposed functionality via anonymous authentication

Hands on lab

11. Thread safety

- Concurrency & Race conditions
- Mutual Exclusion
- Deadlock
- Time of Check/Time of Use (TOCTOU)
- Files as Locks
- Symbolic link attacks
- Temporary files
- Handling the race window
- Controlling race objects
- Using atomic operations

12. Logging & Error handling

- How to use exceptions properly
- Process uncaught and unexpected exceptions
- Prevent sensitive information disclosure via errors
- Declare new exception classes for security
- Events you should log
- Events you should not log
- Log Integration with exception management
- Secure Coding Tips
- Prefer Streams to C-Style Input and Output

- Do not replace secure functions with less secure functions
- Avoid defining macros
- Do not ignore values returned by functions or methods
- Secure defaults and initializations
- The least privilege principle
- The defense in depth principle
- The segmentation principle
- Avoiding hard coded secrets
- Use Static Code Tools
- Integrating security into the development lifecycle

13. Anti-reversing

- Eliminate “symbolic info”
- Obfuscate the program
- Code Encryption
- Use anti-debugger tricks
- Code Checksums
- Confusing a Disassembler
- Inlining and Outlining sensitive code
- Interleaving Code
- Existing Tools