Desktop application security in Java

CYDJvDsk3d | 3 days | On-site or online | Hands-on

Your application written in Java works as intended, so you are done, right? But did you consider feeding in incorrect values? 16Gbs of data? A null? An apostrophe? Negative numbers, or specifically -1 or -2^{31}? Because that's what the bad guys will do – and the list is far from complete.

Handling security needs a healthy level of paranoia, and this is what this course provides: a strong emotional engagement by lots of hands on labs and stories from real life, all to substantially improve code hygiene. Mistakes, consequences, and best practices are our blood, sweat and tears.

All this is put in the context of Java, and extended by core programming issues, discussing security pitfalls of the Java language and framework.

So that you are prepared for the forces of the dark side.

So that nothing unexpected happens.

Nothing.

Cyber security skills and drills

Audience

Java developers working on desktop applications

Group size

12 participants

Outline

- Cyber security basics
- Input validation
- Security features
- Time and state
- Errors
- Cryptography for developers
- Common software security weaknesses
- Using vulnerable components
- Wrap up

Preparedness

General Java development

19 labs

7 case studies

What you'll have learned

- Getting familiar with essential cyber security concepts
- Identify vulnerabilities and their consequences
- Learn the security best practices in Java
- Input validation approaches and principles
- Understanding how cryptography can support appplication security
- Learning how to use cryptographic APIs correctly in Java
- Managing vulnerabilities in third party components

Table of contents

Day 1

    - Using ProcessBuilder
    - *Case study – Shellshock*
    - *Lab - Shellshock*
    - *Case study – Command injection via ping*
  - Script injection
- General protection best practices

Integer handling problems

- Representing signed numbers
- Integer visualization
- Integer overflow
- *Lab – Integer overflow*
- Signed / unsigned confusion in Java
- *Case study – The Stockholm Stock Exchange*
- Integer truncation
- Best practices
  - Upcasting
  - Precondition testing
  - Postcondition testing
  - Using big integer libraries
  - Integer handling in Java
  - *Lab – Integer handling*

Files and streams

- Path traversal
- Path traversal-related examples
- *Lab – Path traversal*
- Additional challenges in Windows
- Path traversal best practices

Unsafe reflection

- Reflection without validation
- *Lab – Unsafe reflection*

Unsafe native code

- Native code dependence
- *Lab – Unsafe JNI*
- Best practices for dealing with native code

Day 2

› Security features

Authentication

- Authentication basics
- Multi-factor authentication
- Authentication weaknesses - spoofing

Case study – PayPal 2FA bypass

- User interface best practices

Lab – On-line password brute forcing

- Password management
  - Inbound password management
    - Storing account passwords
    - Password in transit

    Lab – Is just hashing passwords enough?

    - Dictionary attacks and brute forcing
    - Salting
    - Adaptive hash functions for password storage
    - Password policy
    - NIST authenticator requirements for memorized secrets
    - Password length
    - Password hardening
    - Using passphrases

    Lab – Applying a password policy

    Case study – The Ashley Madison data breach

    The dictionary attack

    The ultimate crack

    Exploitation and the lessons learned

    - Password database migration
  - Outbound password management
    - Hard coded passwords
    - Best practices

    Lab – Hardcoded password

    - Protecting sensitive information in memory
    - Challenges in protecting memory
    - Storing sensitive data in memory

Authorization

- Access control basics

Information exposure

- Exposure through extracted data and aggregation

Case study – Strava data exposure

- System information leakage
  - Leaking system information
- Information exposure best practices

Java platform security

- The Java programming language and runtime environment
- Type safety and security
- Security features of the JRE
  - The ClassLoader and the BytecodeVerifier
- Application-level access control in Java
  - Permissions and the Security Manager
  - Privilege best practices
- Role-based access control
  - Java Authentication and Authorization Services (JAAS)
- Protecting Java code and applications
  - Code signing

Lab – Code signing and permissions

UI security

- UI security principles
- Sensitive information in the user interface
- Misinterpretation of UI features or actions
- Insufficient UI feedback
- Relying on hidden or disabled UI element
- Insufficient anti-automation

› Time and state

Race conditions

- Race condition in object data members
  - Singleton member fields

Lab – Singleton member fields

- File race condition
  - Time of check to time of usage – TOCTTOU
  - Insecure temporary file
- Database race conditions

Lab – Database race conditions

- Avoiding race conditions in Java

› Errors

Error and exception handling principles

Error handling

- Returning a misleading status code
- Reachable assertion
- Information exposure through error reporting

Exception handling

- In the catch block. And now what?
- Catching NullPointerException
- Empty catch block

Day 3

› Cryptography for developers

Cryptography basics

Java Cryptographic Architecture (JCA) in brief

Elementary algorithms

- Random number generation
  - Pseudo random number generators (PRNGs)
  - Cryptographically strong PRNGs
  - Using virtual random streams
  - Weak and strong PRNGs in Java
  - Using random numbers in Java
  - *Case study – Equifax credit account freeze*
  - *Lab – Random numbers in Java*
- Hashing
  - Hashing basics
  - Common hashing mistakes
  - Hashing in Java
  - *Lab – Hashing in JCA*

Confidentiality protection

- Symmetric encryption
  - Block ciphers
  - Modes of operation
  - Modes of operation and IV – best practices

- Symmetric encryption in Java
- *Lab – Symmetric encryption in JCA*
- Asymmetric encryption
  - The RSA algorithm
  - Using RSA – best practices
  - RSA in Java
  - *Lab – Using RSA in JCA*
  - Elliptic Curve Cryptography
  - The ECC algorithm
  - Using ECC – best practices
  - ECC in Java
  - *Lab – Using ECC in JCA*
  - Combining symmetric and asymmetric algorithms

Integrity protection

- Message Authentication Code (MAC)
  - Calculating MAC in Java
  - *Lab – Using MAC in JCA*
- Digital signature
  - Digital signature with RSA
  - Digital signature with ECC
  - Digital signature in Java
  - *Lab – Digital signature in JCA*

Public Key Infrastructure (PKI)

- Some further key management challenges
- Certificates
  - Chain of trust
  - Certificate management – best practices

› Common software security weaknesses

Code quality

- Data handling
  - Initialization and cleanup
    - Constructors and destructors
    - Class initialization cycles
    - *Lab – Initialization cycles*
  - Unreleased resource
    - The finalize() method – best practices
- Object oriented programming pitfalls
  - Accessibility modifiers
    - Are accessibility modifiers a security feature?

- Accessibility modifiers – best practices
- Overriding and accessibility modifiers
- Inheritance and overriding
- Mutability
  - *Lab – Mutable object*
- Cloning

› Using vulnerable components

Assessing the environment

Hardening

Vulnerability management

- Patch management
- Vulnerability management
- Vulnerability databases

Lab – Finding vulnerabilities in third-party components

› Wrap up

Secure coding principles

- Principles of robust programming by Matt Bishop
- Secure design principles of Saltzer and Schröder

And now what?

- Software security sources and further reading
- Java resources