Developing Secure Java EE Applications

FROM THREATS TO BEST PRACTICES PART 1

MASSIMO CALIMAN

1: Introduction

- Welcome and Objective
- Current Threat Landscape
- Java EE / Jakarta EE Context

2: Why Are We Here? - The Objective of This Talk

- **Developer Empowerment**
  - Equip Java EE developers with concrete knowledge and tools
- **Security is Everyone's Responsibility**
  - Not just a problem for cybersecurity experts or IT Ops
- **From Threats to Prevention**
  - Understand what to look for and how to defend against it
- **Improve Code & Processes**
  - Integrate security into the software development lifecycle

Tools

- **Burp Suite Community Edition**
  - https://portswigger.net/burp
  - HTTP(s) / WebSockets proxy and history
  - Essential tools - Repeater, Decoder, Sequencer, and Comparer.
- **SonarQube Server**
  - Automated code quality and security reviews
  - https://www.sonarsource.com/products/sonarqube/server/

New Entry!

- **Prompt Injection**
  - Direct (Model)
  - Indirect (Document)
  - https://genai.owasp.org/llmrisk/llm01-prompt-injection/

3: The Evolving Threat Landscape (The "Why" - Part 1)

- **Frequency & Sophistication**
  - Attacks are no longer just 'script kiddies'
- **Motivations**
  - Financial Gain (ransomware, data theft)
  - Industrial/State Espionage
  - Vandalism/Reputational Damage
- **Real-World Consequences**
  - Data Breaches
  - Significant Financial Losses
  - Reputational Harm, Loss of Customer Trust
  - Legal & Regulatory Penalties (e.g., GDPR)

4: Applications are the Target (The "Why" - Part 2)

- From Infrastructure to Code
  - Attackers focus on where they find the most vulnerabilities
- Common Attack Vectors
  - Code Vulnerabilities
    - SQL Injection, XSS, Broken Authentication, etc
  - Misconfigurations:
    - Servers, Frameworks, Libraries
  - Components with Known Vulnerabilities
    - Third-party dependencies
- The OWASP Top 10
  - Our reference guide for the most critical web application vulnerabilities

5: The Java EE/Jakarta EE Context: Our Ecosystem

- **Power & Complexity**
  - The chosen platform for mission-critical enterprise applications
- **Architecture**
  - Distributed Components (EJBs, Servlets, JPA, JMS, JAX-RS, etc.).
- **Unique Security Challenges**
  - **Inherent Complexity**
    - Many components must be securely configured and interact
  - **Legacy Applications**
    - Older applications with outdated security practices
  - **Delegating to the Container (But Not Too Much!)**
    - The Java EE container offers us exceptional 'out-of-the-box' security features: authentication, role-based authorization, and session management. But it's essential to understand that the container is only one part of the solution. If our application code is vulnerable (e.g., SQL Injection, XSS), the container cannot work miracles. Application code security is our primary responsibility.
- **Security:**
  - A Shared Responsibility Starts with Developers

6: The “Plan”

- Developing Secure Java EE Applications - (1)
  - Introduction (this Introduction)
- Developing Secure Java EE Applications - (2)
  - Understanding Common Threats
- Developing Secure Java EE Applications - (3)
  - Secure Development Principles in Java EE
- Developing Secure Java EE Applications - (4)
  - Specific Best Practices for Java EE Jakarta EE
- Developing Secure Java EE Applications - (5)
  - Security Testing and Monitoring