

# **SUMMER INTERNSHIP PROJECT**

**Jayant Rakeshkumar Parakh**

**B.TECH (ELECTRICAL ENGINEERING) YEAR 2024-25**

**Roll No. 21EE10035**



**DEPARTMENT OF ELECTRICAL ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR  
MAY - JULY 2024**

## A. CERTIFICATE OF TRAINING FROM ORACLE

<b>ORACLE®</b>	Oracle India Pvt. Ltd. India Development Center	Oracle Technology Park 3, Bannerghatta Park Bangaluru - 560 029	CIN: U74899DL1993PTC051764 Phone +91 80 4107 0000 Fax +91 80 2552 6124
----------------	--	---	--

**Private & Confidential**

Ref: Oracle-Interns- 1926454

17<sup>th</sup> July 2024

**TO WHOMSOEVER IT MAY CONCERN**


This is to certify that Mr. Jayant Rakeshkumar Parakh, student of Indian Institute of Technology, Kharagpur (Pursuing BE/B.Tech) has completed his project with Oracle India Private Limited.

The project was undertaken from 15-May-2024 to 09-Jul-2024. He worked on "Spring : Build From Source, Vulnerability Detection and Automation"

We wish him all the best in his future endeavours.

Yours sincerely

For Oracle India Private Limited.



**Rambabu Jagatha**  
**Director – HR Operations**

Regd. Office : F-01/02, Salcon Rasvilas, Plot No. D-1, District Centre, Saket, New Delhi - 110 017, India. Tel : +91 11 47469000, Fax : +91 11 40574722

Figure 1. Training Seminar Certificate

## B. INTRODUCTION

I had the privilege of undertaking my summer internship at **Oracle**, a major American multinational company, known for its database and cloud products worldwide. This training period, from **May 15th to July 9th, 2024**, was an invaluable experience that took place at Oracle's Corporate Architecture Business Unit in **Bangalore, India**.

The Corporate Architecture team plays a crucial role within Oracle, being responsible for third-party software strategy and policy. The team oversees the review and approval of requests for third-party open-source and commercial software, as well as cross-product packaging across the entire company.

During my internship, I contributed to building deployment-ready artifacts for **Spring Framework version 5.3.36** and **Spring Boot version 2.7.18** from source, ensuring their reliability and security. In addition to the core task of building these frameworks, I also automated the entire process, from build to artifact scanning, using Jenkins. I was also responsible for integrating upstream changes to resolve Critical and High severity Common Vulnerabilities and Exposures (CVEs), applying security patches and updates from the Spring Framework and Spring Boot communities. Furthermore, I developed a sample application using the built JARs and deployed it on WebLogic Server (WLS) to validate the accuracy of our build artifacts, ensuring that they met the high standards expected within Oracle.

This experience deepened my understanding of enterprise-level software development and the significance of automation in quality assurance. My contributions are now in production within Oracle's Fusion Middleware unit and various Global Business Units. My performance during the internship was recognized, and I was honoured to receive a **Pre-Placement Offer (PPO)** from the company.

I'm grateful to my institute for supporting my professional growth, and this internship has been a crucial milestone in my career.

## C. TRAINING SCHEDULE

### Week 1: Onboarding

My first week involved orientation, IT setup, meeting the WebLogic Server (WLS) team, and an introduction to my project.

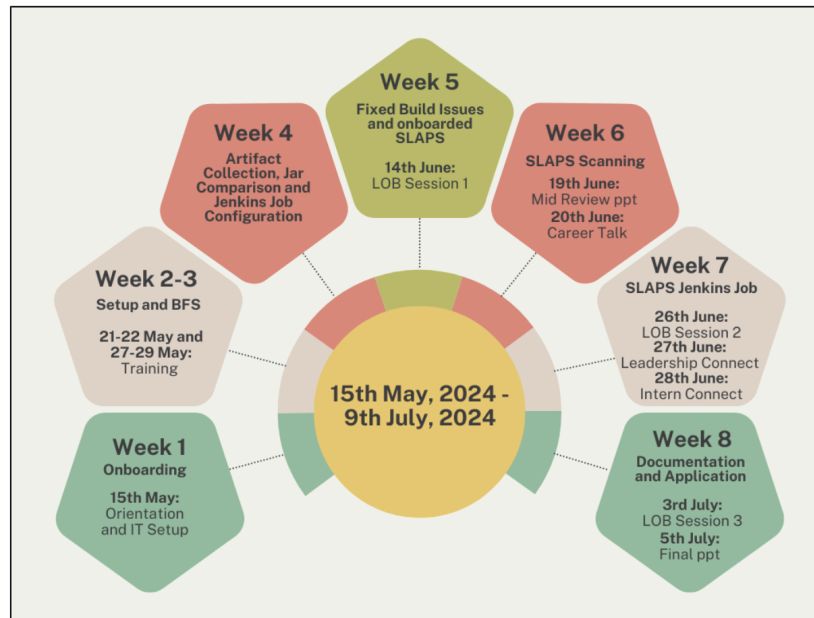


Figure 2. Training Seminar Schedule

### Weeks 2-3: Training and Setup

I completed training on Linux, Kubernetes, Java, SQL, and Oracle Cloud Infrastructure (OCI), set up my development environment, and successfully performed the initial build.

### Week 4: Artifact Collection and Automation

I identified and collected the necessary artifacts to build a standalone Spring application, compared JAR files, and automated the build process using Jenkins.

### Week 5: Build Issue Resolution

This week was dedicated to resolving build issues, ensuring the stability and reliability of the generated artifacts.

### **Week 6: Mid-Term Review and Security Scanning**

I presented my mid-term review, followed by scanning artifacts for security vulnerabilities using the SLAPS tool.

### **Week 7: CVE Integration**

I integrated upstream changes to address Critical and High severity CVEs, applying necessary security patches and updates.

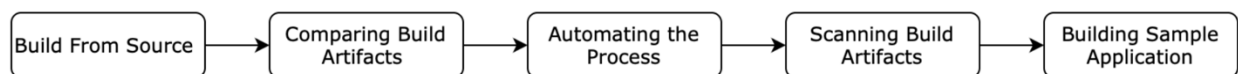
### **Week 8: Deployment and Off boarding**

In the final week, I built and deployed two sample Spring applications on WebLogic Server, presented my final work, and completed off boarding.

## **D. OBSERVATIONS INVOLVED IN INTERNSHIP**

In the ever-evolving landscape of enterprise Java development, Spring has emerged as a cornerstone for building robust, scalable, and maintainable applications. This feature-rich framework is widely acclaimed for creating enterprise-level applications, simplifying complex tasks, and promoting best practices in software development. There are a lot of Oracle internal applications running as well as external applications like Netflix and Udemy which are built on Spring.

My main task was to build Spring Framework version 5.3.36 and Spring Boot version 2.7.18 from source and automate this process. Through automation, including build, test, comparing and artifact scanning, my goal was to promptly detect and resolve any issues stemming from changes in the Spring codebase to ensure the continued stability and optimal performance of applications deployed on WebLogic Server.



## D.1 Initial Steps

### Researching Core Technologies

I began by understanding the working and features of the core technologies to be used in the project.

1. **Spring Framework:** Provides key features like dependency injection, aspect-oriented programming, and transaction management.
2. **Spring Boot:** Enhances Spring's capabilities for rapid application development, allowing the creation of standalone, production-ready applications.
3. **Gradle:** A build automation tool that manages dependencies and builds Java projects using Groovy or Kotlin. I used Gradle to build Spring from source.
4. **Jenkins:** An open-source automation server that supports CI/CD pipelines, which I utilized for integrating various tools.
5. **Perforce:** Used to manage commits and updates to our scripts.

### Initial Setup

In this section, I outline the initial setup process that was essential for starting the project.

1. **JDK:** Installed JDK 8u92 on macOS and a VM instance, setting JAVA\_HOME for compatibility with Spring Framework v5.3.36 and Spring Boot v2.7.18.
2. **Gradle:** Installed and configured Gradle on both macOS and the VM for building the project and running tests.
3. **VS Code:** Set up Visual Studio Code with essential extensions for Java support, Gradle integration, and Git.
4. **P4V (Perforce Visual Client):** Installed and configured to connect to the Perforce server for source code management.
5. **Git:** Installed Git and set up global configurations, cloning the Spring project repository for building.
6. **Virtual Machine (VM):** Configured a VM to provide a consistent development environment.

## D.2 Build from Source

My next step was build Spring Framework and Spring Boot from source. I used Gradle Wrapper as build automation tool to compile and build the projects.

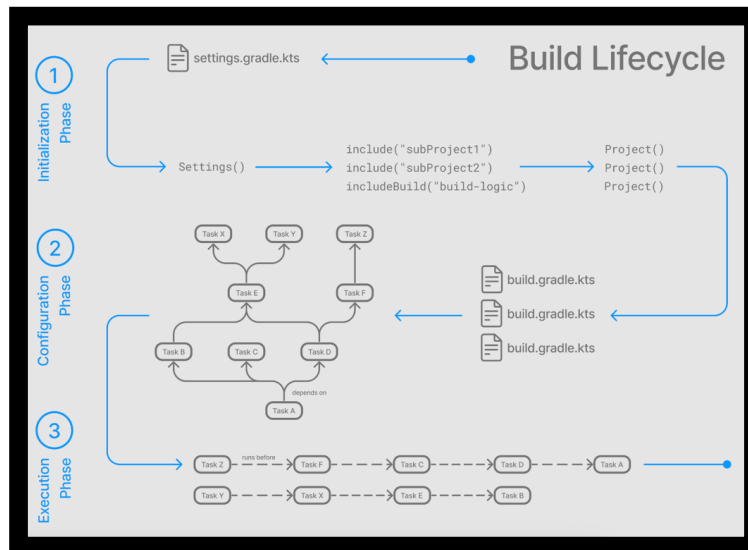


Figure 3. Gradle Build Lifecycle

Gradle builds go through three main phases:

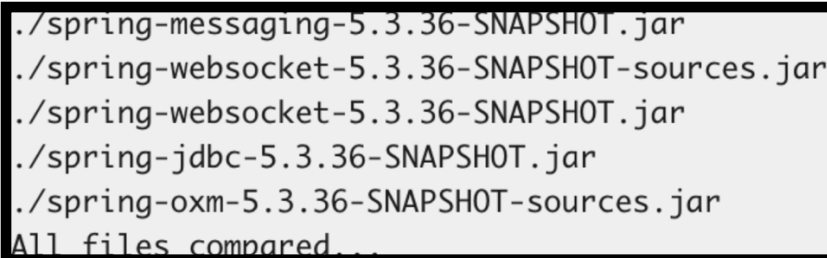
1. **Initialization:** Detects the `settings.gradle` file and establishes the project structure.
2. **Configuration:** Evaluates the `build.gradle` scripts and creates a task graph of dependencies.
3. **Execution:** Schedules and executes the tasks defined in the task graph

Next, we located and gathered all the build artifacts, which were distributed across the project submodules. These artifacts were found in the **submodule /build/libs** directory of each submodule.

### D.3 Jar Comparison

After successfully building and collecting all the artifacts, the next step was to compare the build artifacts with the original published artifacts to ensure there were no discrepancies. The steps we followed were:

1. Retrieved the original published JAR files from [spring.repo.io](http://spring.repo.io).
2. Integrated a JAR comparison step into our build script. This step iteratively compares all the published JARs with the newly built JARs using the standard JAR comparison script available in the Oracle Database.

A terminal window with a black border showing the output of a JAR comparison script. The text is as follows:

```
./spring-messaging-5.3.36-SNAPSHOT.jar  
./spring-websocket-5.3.36-SNAPSHOT-sources.jar  
./spring-websocket-5.3.36-SNAPSHOT.jar  
./spring-jdbc-5.3.36-SNAPSHOT.jar  
./spring-oxm-5.3.36-SNAPSHOT-sources.jar  
All files compared...
```

*Figure 4. JAR Comparison*

### D.4 Automating the BFS

Once I had successfully built the Spring Framework and Spring Boot from source and ensured the integrity of the build artifacts, I focused on automating the build process. Automation helps us ensure that builds are consistent, efficient, and easily repeatable. Steps to Automate the Build Process were as follows:

#### 1. Created a Build Script:

- Installed JDK 8.
- Cloned the Spring repository and checked out specific tags.
- Built the project to produce JAR files.
- Extracted generated JARs and compared them with original JARs.



## 2. Testing the Automation Script

After preparing the build script, I ran several iterations on my local VM instance to test its reliability and effectiveness. I encountered some build failures due to caching issues but resolved them after cleaning the builds and utilizing appropriate flags during the build. After creating the build script, I pushed these changes into Perforce.

## 3. Jenkins based CI/CD Pipeline

To further enhance our automation process, I integrated the build script into Jenkins, enabling continuous monitoring and deployment:

1. **Continuous Monitoring:** Detects source code changes and triggers predefined jobs.
2. **Build Phase:** Executes the build script and runs tests.
3. **Deployment Phase:** Deploys artifacts to specified environments.

I created separate Jenkins BFS jobs for Spring Framework and Spring Boot and archived the generated artifacts in the post build steps. I archived a total of **22** .jar files in Spring Framework, along with sources.jar, javadoc.jar, and test-fixtures.jar for each. In comparison, Spring Boot required archiving **70** .jar files for its extensive range of sub-modules, also including corresponding sources.jar, javadoc.jar, and test-fixtures.jar.

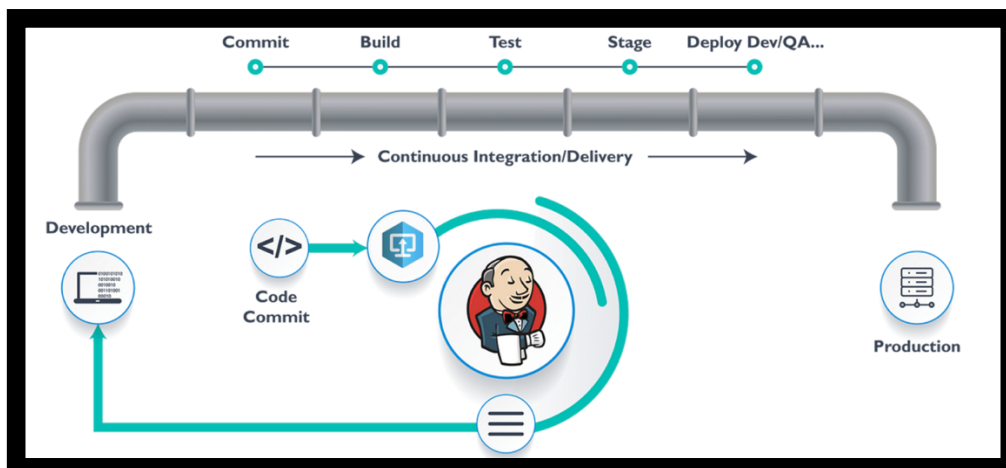


Figure 5. Jenkins Pipeline

## D.5 SLAPS Scan

I next moved on to scanning the artifacts built by pipeline for malware and vulnerabilities. For this I used the ORACLE Proprietary tool called **SLAPS**, Security Ledger and Provenance Service, that provides security inspection of code and artifacts. Below are the detailed steps involved:

**1. Downloading Artifacts:** First, I downloaded the artifacts to a designated **/net** path within oracle VCN.

**2. Conducting Thorough Scans:** Once the artifacts were downloaded, I conducted comprehensive scans to identify any potential vulnerabilities or issues.

**3. Generating Detailed Logs:** During the scanning process, I generated detailed logs. These logs tracked the scanning process meticulously, providing insights into each step and highlighting any detected issues.

## D.6 Scan Results

For spring-framework a vulnerability was reported in spring-web-5.3.36.jar. Based on the CVE number (**CVE-2016-1000027**) the STaaS tool identified that this was already reported in one of previous scan done for version 5.3.27. This CVE is a known issue in this spring-web jar. I also tried using Grype and Trivy to scan the artifacts and cross-verify the results from SLAPS scan. As a result, they pointed to the same CVE.

Spring Framework through version 5.3.16 suffers from a potential **remote code execution (RCE)** issue when used for Java deserialization of untrusted data. This mechanism is known to be vulnerable. The Spring Framework community has addressed this issue by deprecating the affected class in earlier versions and completely removing it from version 6. To mitigate this issue in my build process, I decided to stop using **HttpInvokerServiceExporter** and **SimpleHttpInvokerServiceExporter** classes which were vulnerable to deserialization, and integrated the upstream changes in my source code so that the applications don't break.

## D.7 Building Sample Application

My last step was to create a sample Spring application and deploy it on the WebLogic Server. This process was crucial for ensuring that the BFS jobs generate artifacts that are reliable and can be used by others within the WLS organization. Here's how I did it:

### Create a Sample Spring Application

I created two sample Spring applications to ensure that the build artifacts could be effectively used. I cloned the following applications from GitHub:

**1. Spring Petclinic:** It allows users to find veterinarians by location and update the database with pets and their owners. The application supports various pet types, such as dogs, cats, and birds.

**2. Spring Blogging:** Full-stack spring boot application blogging website where users can register, sign in, and sign up to create and manage their own blogs and articles. This project uses Gradle for building the application, managing dependencies, and running tasks.

### Build the Application

I changed the configuration of the project to ensure that the application uses my locally generated artifacts instead of those from Maven Central. I updated the `build.gradle` and `pom.xml` within the projects to point to the local artifacts.

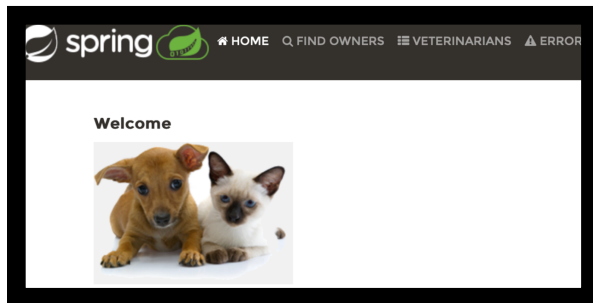


Figure 6. Spring Pet clinic Landing Page

 The image shows the 'Owner Information' section of the Spring Pet clinic interface. It contains a table with the following data:
 

Owner Information	
Name	Ojaswi Chopra
Address	Udaipur
City	Udaipur
Telephone	1234

 Below the table are two buttons: 'Edit Owner' and 'Add New Pet'. Underneath, there is a section titled 'Pets and Visits' with a table:
 

Name	Birth Date	Type	Visit Date	Description
Sky	2004-10-02	dog		

 At the bottom of the table, there are links for 'Edit Pet' and 'Add Visit'.

Figure 7. Spring Pet clinic Interface

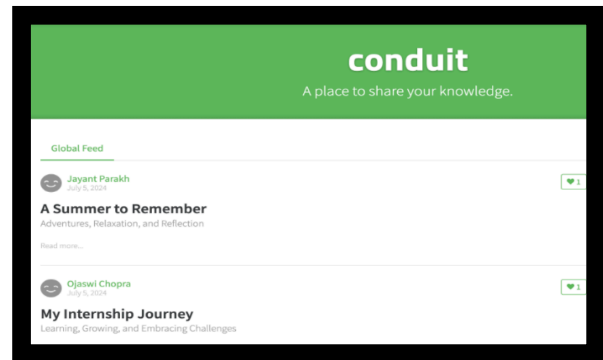


Figure 8. Spring Blogging Landing Page

 The image shows the 'Write Article' interface of the Spring Blogging application. It includes a text input for 'Article Title', a text area for 'What's this article about?', a larger text area for 'Write your article (in markdown)', and a text input for 'Enter tags'. A green 'Publish Article' button is located at the bottom right.

Figure 9. Spring Blogging Interface

## Preparation and Deployment

To ensure sample applications were ready for deployment on WebLogic Server, I configured the properties and packaged the applications into war files to make them deployment ready. Accessed the Web Logic Console and deployed the WAR files.

## **E. HOW THE TRAINING HAS ADDED TO MY PROFESSIONAL PRACTICAL EXPOSURE**

The training I received during my internship at Oracle has greatly enriched my professional and practical exposure, particularly in the field of enterprise software development. Through hands-on experience with advanced technologies such as Spring Boot, Spring Framework, Jenkins, and Linux Shell Scripting, I deepened my understanding of building and automating robust applications. These technical skills were further honed as I tackled and debugged challenges, such as resolving build issues and integrating security patches, which underscored the critical importance of maintaining both functionality and security in large-scale software systems.

In addition to technical growth, this internship provided me with invaluable opportunities to develop and refine my soft skills. Working closely with the WebLogic Server team, I enhanced my teamwork and communication abilities, learning the importance of effective collaboration in a professional setting. Regular team meetings, feedback sessions, and presentations, such as the mid-term and final reviews, allowed me to improve my ability to articulate complex technical concepts clearly and confidently. I also made some great friends with my fellow interns. These experiences have equipped me with the interpersonal skills necessary to thrive in a collaborative and dynamic work environment.

The practical exposure I gained by contributing to projects now in production has not only deepened my technical expertise but also provided me with a sense of accomplishment. This experience has clarified the software development lifecycle and prepared me to tackle complex technical challenges in my future career. I am grateful for this internship and for their unwavering support throughout this journey.