Shift-Left Remediation with DbSaicle: OSS and Veracode Findings Resolution for Developers

Abstract

This paper presents a remediation-first approach to shift-left security using DbSaicle. It focuses on resolving high and critical findings from open-source dependency scans and Veracode pipeline scans through a repeatable loop of triage, fix, rebuild, and rescan. The scans provide actionable inputs, while the workflow delivers measurable remediation outcomes without slowing delivery.

1 Executive Overview

Security shifts left only when remediation becomes routine, not when scanning is available. This paper focuses on how DbSaicle enables remediation-first workflows for two critical classes of findings:

- Open-source dependency vulnerabilities (OSS)
- Static analysis findings from build artifacts (Veracode pipeline scan)

The scans are inputs. The real value is a repeatable loop that helps developers remediate high and critical findings quickly, validate fixes, and keep builds healthy.

2 Why Remediation-First Matters

Late security findings increase cost, delay releases, and create context switching. A remediation-first approach changes the economics by:

- Fixing issues while the developer still has context
- Reducing rework caused by late-stage findings
- Converting security into a daily, measurable workflow

3 Remediation Scope and Inputs

DbSaicle provides two scanning tools that feed remediation workflows:

- `oss_vulnerability_scan` gathers dependency findings across Maven, Gradle, and npm.
- `veracode_pipeline_scan` returns static analysis findings from a build artifact.

Remediation is the focus: triage, fix, rebuild, and rescan until high and critical findings are cleared or explicitly accepted.

4 The Tools That Power Remediation

4.1 OSS Vulnerability Scan

Scans dependency graphs and returns a Markdown report with severity, vulnerable versions, and recommended fixes. It is designed for fast, targeted remediation rather than exhaustive reporting.

Key outcomes:

- Clear remediation targets
- Direct or transitive dependency visibility
- Fewer cycles to resolve high/critical items

4.2 Veracode Pipeline Scan

Performs static analysis on build artifacts (JAR/WAR/EAR/ZIP/APK) and returns findings with severity and location details. It supports local and CI workflows and is fast enough to run during active remediation.

Key outcomes:

- Actionable findings tied to code locations
- Repeatable remediation loop before merge
- Consistent gates for high-risk changes

5 Remediation Workflow Summary

Developers can run both tools from the DbSaicle plugin in VS Code or IntelliJ IDEA and follow a standardized loop:

1. Build the project (with tests first, then without tests during remediation).
2. Run `oss_vulnerability_scan` and triage high/critical findings.
3. Run `veracode_pipeline_scan` on the build artifact and triage high/critical findings.
4. Apply fixes, rebuild, and rescan until results are clean or accepted.

6 Workflow References

- OSS Remediation Workflow: `<<OSS_REMEDIATION_WORKFLOW_LINK>>`
- Veracode Remediation Workflow: `<<VERACODE_REMEDIATION_WORKFLOW_LINK>>`

7 Infographic Placeholder

Use this placeholder for a visual remediation loop (scan, triage, fix, rebuild, rescan):

Infographic placeholder. Replace with a final diagram or image that shows the remediation loop and the two tool inputs.

Figure 1: Remediation workflow infographic placeholder.

8 Configure DbSaicle for Remediation

Add the following to your `config.yaml` (adjust URLs and secrets as needed). Environment variables are supported for secrets.

```
ossVulnerability:
  jfrogPlatformUrl: "https://your-jfrog-host"
  jfrogAccessToken: "${JFROG_ACCESS_TOKEN}"
  mavenCommand: "/path/to/mvn" # optional
  mavenHome: "/path/to/maven"  # optional

veracode:
  apiKeyId: "${VERACODE_API_KEY_ID}"
  apiKeySecret: "${VERACODE_API_KEY_SECRET}"
  baseUrl: "https://api.veracode.com" # optional
  userAgent: "dbsaicle-veracode"       # optional
```

9 Example Usage (Tool and Workflow)

9.1 OSS Remediation Example

1. Ask for the project root.
2. Run the tool:

```
{
  "project_dir": "/path/to/project-root"
}
```

3. Fix high/critical dependency findings.
4. Re-run the tool to confirm remediation.

9.2 Veracode Remediation Example

1. Build the artifact (JAR/WAR/ZIP/APK).
2. If the artifact is larger than 200 MB, package smaller outputs (for Java, `target/classes`) into a ZIP and use that as the artifact.
3. Run the tool:

```
{
  "artifact_path": "/path/to/artifact-or-zip",
  "output_json_path": "veracode-findings.json"
}
```

4. Fix high/critical findings and rebuild without tests.

5. Re-run the tool until results are acceptable.

10 Productivity and Quality Impact

Remediation-first workflows drive measurable improvements:

- Lower remediation time: fixes happen while context is fresh.
- Reduced rework: findings are handled before they cascade downstream.
- Higher release confidence: repeated scans validate fixes quickly.

11 Recommended Adoption Path

1. Pilot with 1 to 2 teams on active services.
2. Track time-to-remediate and defect rates for 2 to 4 sprints.
3. Expand to more teams after a positive signal.
4. Establish policy: high/critical findings must be remediated before merge.

12 Conclusion

DbSaicle enables remediation-focused security that is practical for developers. The OSS and Veracode tools provide fast, actionable inputs, while the workflows turn scanning into measurable remediation outcomes. This is a sustainable shift-left strategy that improves security without slowing delivery.