Introduction

- Business Logic Vulnerabilities (BLV): “The flaw present in the faulty implementation of the business logic within the application code.”
- Error-prone choice in a code branch during execution.
- Unidirectional and criticality assessment of BLVs.
- White-box code analysis: Static analysis, Dynamic analysis, Fuzzy Logic heuristics.
- Source code classification for vulnerability evaluation.

Dynamic Analysis

- Daikon tool from MIT monitors the execution of the AUT with input data.
- Extracts rules representing business logic and expected application behavior — called Invariants.

Static Analysis

- NASA’s JPF tool gathers execution paths and states with variable valuations.
- Method invocation and variable data is compared with invariants for BLV detection.

Source Code classification — AST tree generation

Fuzzy Logic Severity ranking: Risk of source code points

Input Vector and Tainted object propagation

Daikon Dynamic Analysis — invariant generation

JPF static analysis — execution paths and states

Fuzzy Logic Severity ranking: Logical error detection

Criticality results: BLV detection and ranking

Linguistic value

Condition

Vulnerability level

Low

No invariant violations or improper sanitation check.

0

Medium

Multiple propagation of input data with general checks.

2

High

No check or improper checks in variables depended on input data for branch conditions.

5

Figure 5: Ranking values for the potential vulnerabilities detected according to classification

Linguistic Value

Condition

Severity Level

Low

Random variable Severity

1

Medium

Data sinks (i.e. variables with data from user input)

3

High

Variables used in a CB ONCE (“IF” and/or a "SWITCH")

4

Medium

Variables used in a CB TWICE OR MORE (“IF” and/or a "SWITCH")

5

High

Variables used as data sink AND in a CB (“IF” and/or a "SWITCH")

5

Figure 6: Severity Ranking values for source code points according to classification

NASA RJC controller tests

- Logical error detection: Detect logical errors injected in real-world applications.
- Methods yielded extensive information about the application:
  - More than 3,000 invariants generated.
  - Injection of malformed values in objects to test logical error detection.

- More application tests where instrumented to test APP_LogIC’s method:
  - JSCH SSH2 framework for detecting unchecked input data.
  - CleanSheets excel application for detecting GUI input data.
  - Test-beds emulating authentication systems with a logical error to bypass them.

Conclusions

- Extended method, capable of detecting LVs in real-world JAVA applications.
- Fuzzy Logic system with solid mathematical base.
- Human-like ranking system using Fuzzy Logic and heuristics.
- Can be used to enhance published research results.

References


The Fuzzy Logic ranking System:

- **Criticality** is the final, calculated risk value assigned for each variable.
- Combines Severity and Vulnerability ranks by aggregating the two sets.

**Criticality(x) = Severity(x) \cdot Vulnerability(x)**