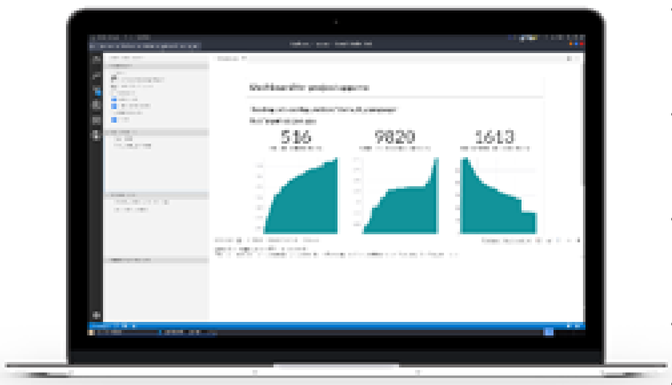
code intelligence

Automated Software Testing for Java Spring Boot

CI Fuzz is a state-of-the-art security testing software. It offers easy IDE integration that saves developers' time and effort while drastically improving the stability and reliability of the codebase.

Why Code Intelligence?

- **Get access to state-of-the-art technology:** Feedback-based fuzzing and concolic code execution
- **Achieve reliable testing results:** No false positives due to the combination of dynamic and static analysis
- **Extremely fast setup for Microservices:** More and more Business logic is migrated into microservices using frameworks, such as Springboot or J2EE. Quickly set up Feedback-based Fuzzing for these kind of services.
- **Improve the discovery of vulnerabilities:** Higher code coverage and sophisticated dynamic detection mechanisms
- **No additional effort:** IDE integration helps to define software interface while writing the code
- **Maximize your productivity:** Browse, debug and quickly fix the found bugs and vulnerabilities
- **Test when and where you need it:** Fast and reliable source code testing integrates seamlessly into your CI/CD process



Technical features

- Based on feedback-based fuzzing and concolic execution
- Feedback-Based Fuzzing for Java and Java-Frameworks (Spring / Spring Boot / J2EE)
- Automated Fuzz Test generation for REST, SOAP, and URL Encoded controllers
- URL recognition and Parameter Fuzzing
- Utilizes structure-awar fuzzing for JSON/XML/Protobuf but also APIs expecting custom data types
- API / Function Testing and easy Socket-Based Network Fuzzing

code intelligence

Methods used

- **Initial static analysis:** Our software scans the source code to identify which parts are worth be tested dynamically and guide the mutation engines by providing hints for data structures. If our static analysis identifies external interfaces such as RESTControllers, network communication or external API functions, we recommend fuzzing those interfaces and will help to set up corresponding fuzzers.
- **Dynamic testing:** The application is tested during its execution. The output derived in the fuzzing process is analyzed and continuously used as input for subsequent test runs. Due to execution testing, virtually all vulnerabilities can be discovered and the number of false positives practically reduced to zero.
- **Feedback-based fuzzing:** The software under test is fed with inputs, which are purposefully mutated in the testing process. The fuzzer gets feedback about the code covered during the execution of a given input. This allows the fuzzer to explore the program state efficiently and thus significantly increase the chances of triggering vulnerabilities.

Usability

CI Fuzz offers an easy-to-use interface to apply these advanced technologies. No deep technical fuzzing knowledge is required. Instead, developers are able to just define which functions or interfaces (e.g. RESTControllers) they want to have tested and our software is able to generate test cases automatically.

Our IDE plugin displays which parts of the code have been reached by the fuzzer and visualizes the fuzzing process. Found crashes, memory leaks and vulnerabilities can also be replayed by starting the IDE's debugger with the input causing the crash. Alternatively, you are able to interact with our core software using the command line.

Continuous integration

CI Fuzz easily integrates into a standard CI/CD workflow such as Jenkins. The fuzz tests are run automatically with each new code change and incidents are reported promptly. Fuzzing tests can be scaled on-demand on a Kubernetes cluster.

Code Intelligence GmbH
Rheinwerkallee 6
53227 Bonn
www.code-intelligence.com

Contact
Jonathan Reimer
+49 228 2869 5830
sales@code-intelligence.com

Follow us
@CI_GmbH
company/codeintelligence/