Automated Application Testing for Java (including Web Services)

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

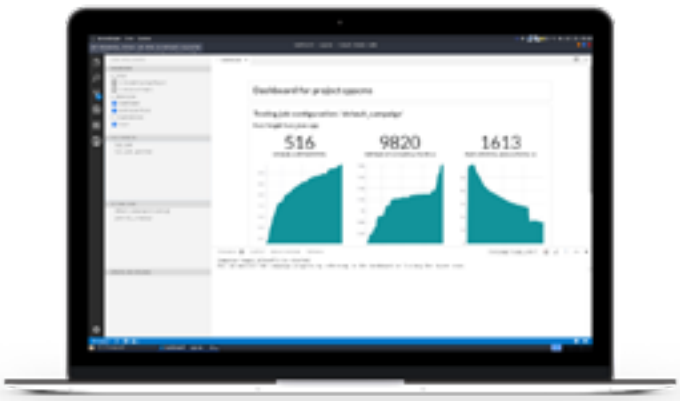
Fuzzing gained an enormous popularity both in the industry and scientific community in the last few years. For instance, 16.000 bugs have been automatically discovered in Google Chrome in 2019. While proven to be a very effective technology to find vulnerabilities and bugs, modern fuzzing is still too complex to integrate and maintain correctly for most development projects. Code Intelligences has the goal to make modern fuzzing more usable and thus to increase the security of their software.

Why Code Intelligence?

- **Reduce costs:** Save over 60% of developers time due to our fully automated solution and our easy-to-use IDE plugin
- **Fuzzing for everyone:** Create and execute modern software tests without any expert knowledge supported by our preloaded settings and intelligent execution engine
- **Increase productivity:** Maximize the productivity of your costly developers on programming code instead of the time-consuming task of hunting bugs and security issues
- **No prerequisites needed:** Our agnostic approach ensures seamless integration into your existing process landscape
- **Access to state-of-the-art technology:** Unparalleled combination of latest breakthrough technologies under one roof to deliver added business values
- **One-click setup:** Quickly build intelligent tests for microservices using frameworks (e.g., Spring Boot) through our smart code scanning, detection, and automated execution technologies

Methodology

Our core technology consists of feedback-based fuzzing using instrumentation, the most promising dynamic code analysis technique. Here, the software under test is executed with inputs, which are systematically mutated during the testing process. In this process, the mutation engine gets coverage and data structure feedback from executed code via instrumentation. This allows the fuzzer to explore the program states effectively and thus significantly increase the code coverage and the number of triggered bugs and vulnerabilities. An initial static analysis of the code allows the deduction of the input structure and is used to guide the fuzzer allowing to produce inputs reaching important states in the program.



Usability

CI Fuzz offers an easy-to-use interface to integrate modern testing techniques. No deep technical fuzzing knowledge is required to set-up the system. Instead, developers are able to define which program interfaces (e.g. REST MicroServices, Protobuf/GRPC services, or API functions) they want to have tested. CI Fuzz is then able to generate corresponding test cases automatically.

Our user interface, available as IDE plugin or soon as web application, provides several dashboards and visualizes the fuzzing process by showing the parts of the code which have been reached by the fuzzer. Found crashes, information leaks and other types of 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 (e.g., 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.

Technical features

- **Feedback-based fuzzing** based on instrumentation for Java and other JVM-based languages such as Kotlin or Scale. Frameworks such as Spring / Spring Boot / J2EE are supported
- **Low false-positive rate** due to actual execution of the code rather than static analysis based on patterns and data flows
- **Structure-aware fuzzing** approach for maximization of code coverage with support for custom data types and protocols defined in your own codebase with low integration effort
- **Automated API schema deduction** for common serialization formats
- **Easy to integrate into Devops:** Support for common build systems (e.g., Ant, Maven, Gradle) and Continuous Integration Pipelines (e.g. Jenkins, Gitlab Pipelines)
- **Automated fuzz test generation** for API testing, including web services such as REST, SOAP, Protobuf/GRPC, and URLEncoded interfaces
- **Multitude of targets** (Network, file and device-based interface fuzzing)

Identified Vulnerabilities

With CI Fuzz you are able to discover the following vulnerabilities out of Mitre's CWE Top 25:

Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	Improper Restriction of Operations within the Bounds of a Memory Buffer	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
Improper Input Validation	Use After Free	Information Exposure	Out-of-bounds Read
Deserialization of Untrusted Data	Uncontrolled Resource Consumption	Use of Hard-coded Credentials	Use of Hard-coded Credentials
Improper Restriction of XML External Entity Reference	Improper Control of Generation of Code ('Code Injection')	Missing Release of Resource after Effective Lifetime	Integer Overflow or Wraparound
NULL Pointer Dereference	Out-of-bounds Write		

With CI Fuzz you are able to discover the following vulnerabilities out of OWASP Top 10:

Injection	Broken Authentication	Sensitive Data Exposure	XML External Entities (XXE)
Broken Access Control	Security Misconfiguration	Insecure Deserialization	

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

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

Follow us
@CI_GmbH
company/codeintelligence/