



code intelligence

Fuzzing Java with Jazzer



## Fabian Meumertzheim

### Senior Software Engineer



#### Background

- Mathematician by education
- OSS contributor (Bazel, Chromium, Android Password Store)

#### Responsibilities at Code Intelligence

- Fuzzing Technologies
- OSS Initiatives & Cooperations



@fhenneke



meumertzheim@code-intelligence.com



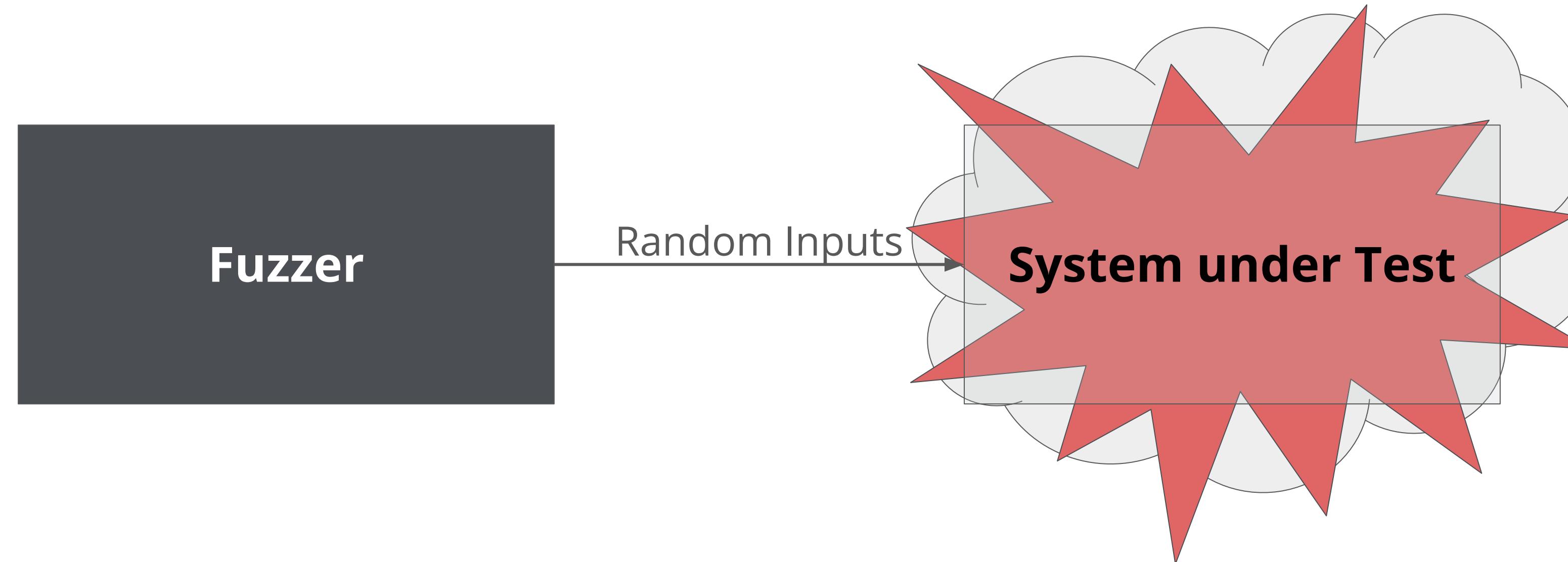
fmeum

# What is Fuzzing?

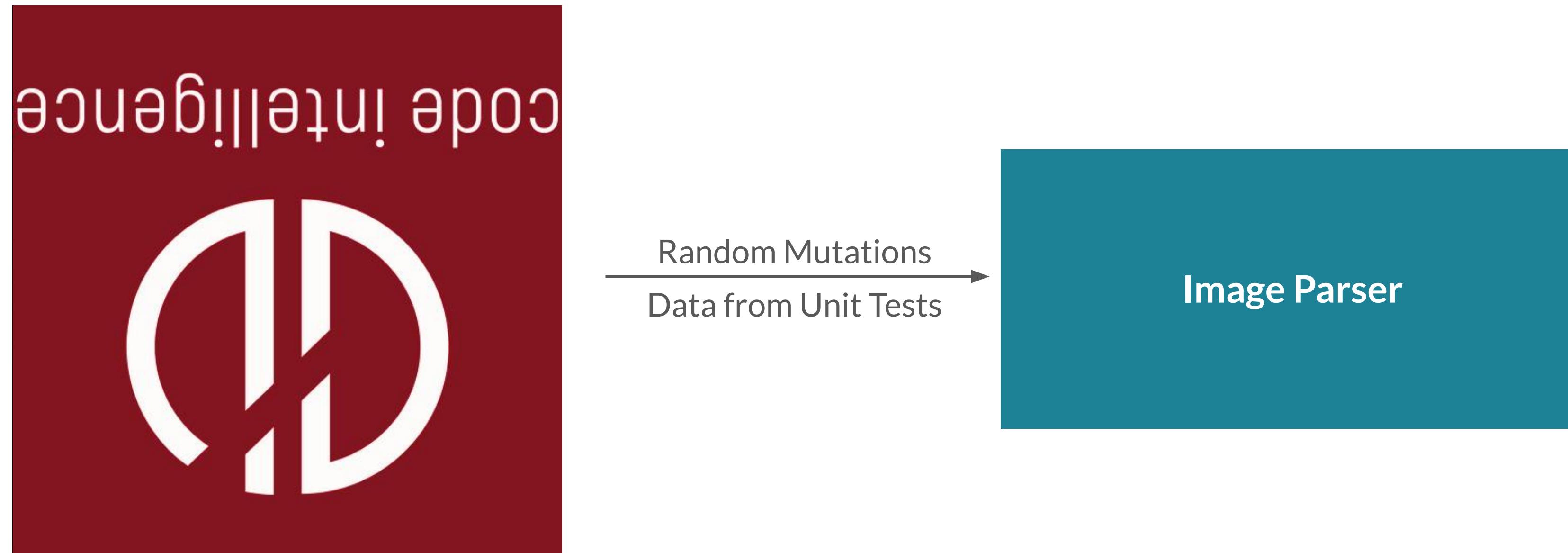
**fuzz** | verb.

/'fəz/

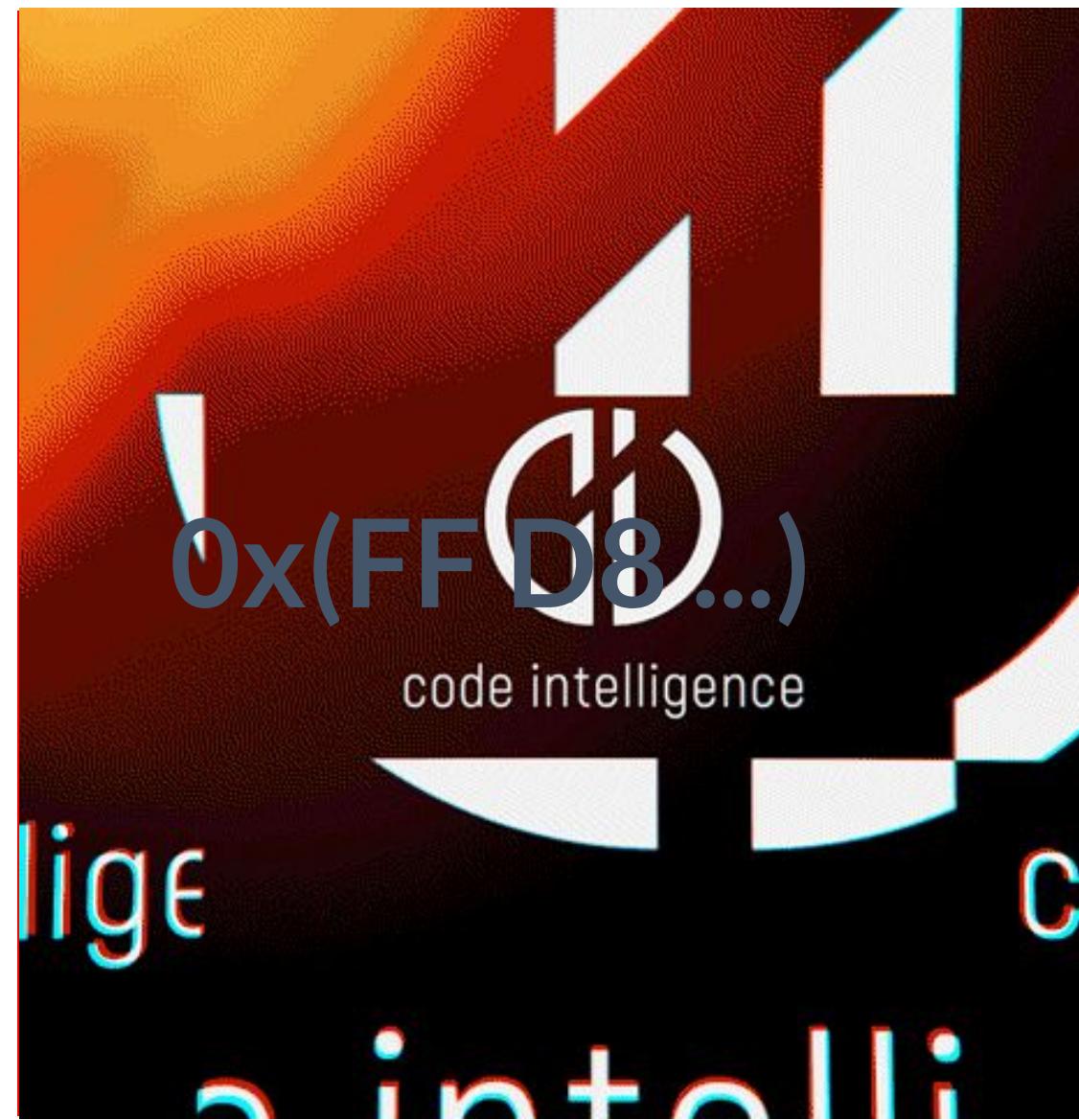
*I. to make or become blurred*



# Blackbox Fuzzing



# Whitebox Fuzzing



*covered branches, magic bytes,  
compared values*

```
private static final int[] MAGIC_NUMBERS_GIF = { 0x47, 0x49, };
private static final int[] MAGIC_NUMBERS_JPEG = { 0xff, 0xd8, };

// ...

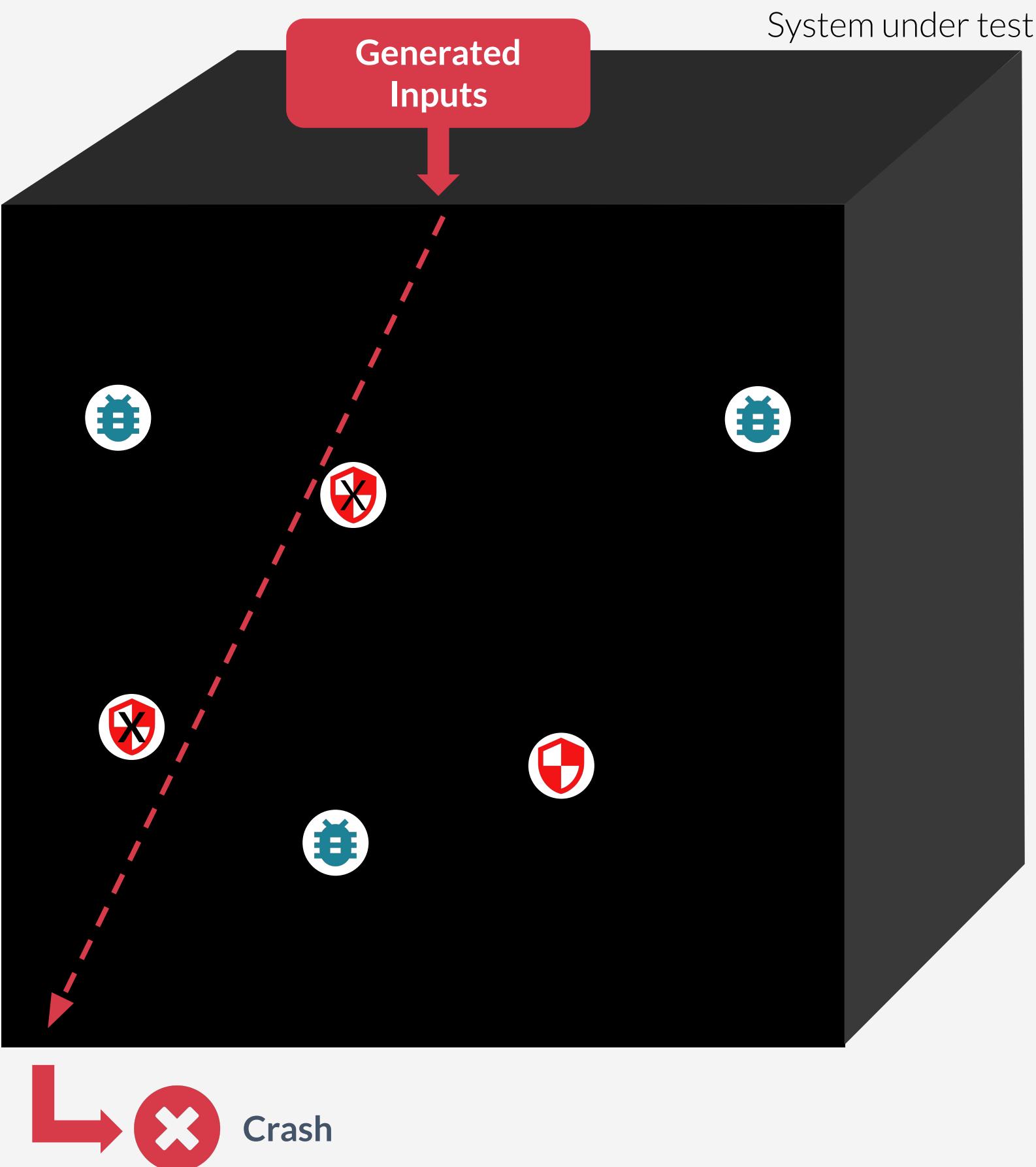
if (compareBytePair(MAGIC_NUMBERS_GIF, input)) {
    return ImageFormats.GIF;
} else if (compareBytePair(MAGIC_NUMBERS_JPEG, bytePair)) {
    return ImageFormats.JPGE;
}
return ImageFormats.UNKNOWN;
// ...
```

# Black-box vs Coverage-guided Fuzzing

 Vulnerabilities  
 Bugs

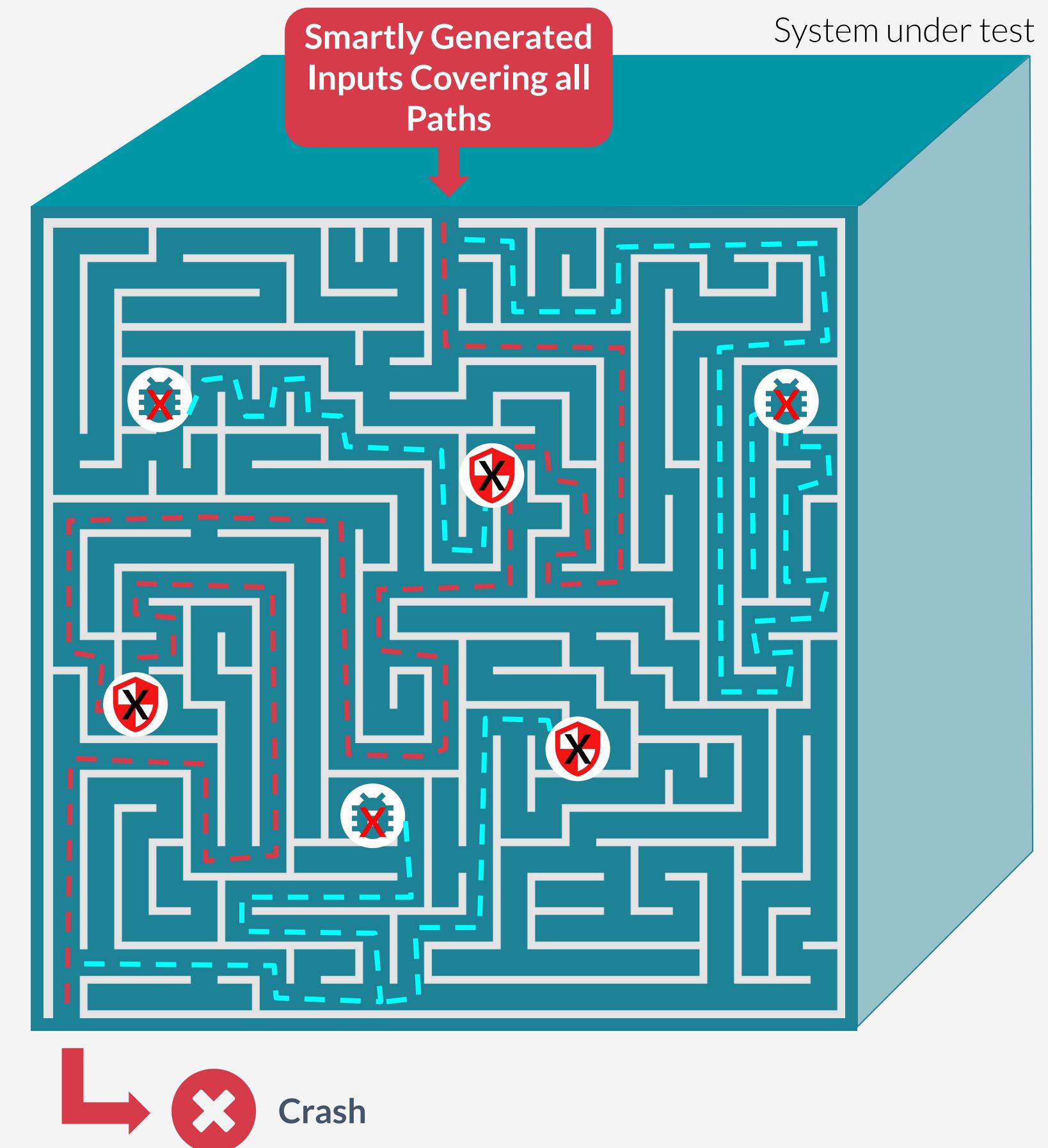
## Black-box Fuzzing

- No knowledge of which code is reached
- Misses critical bugs

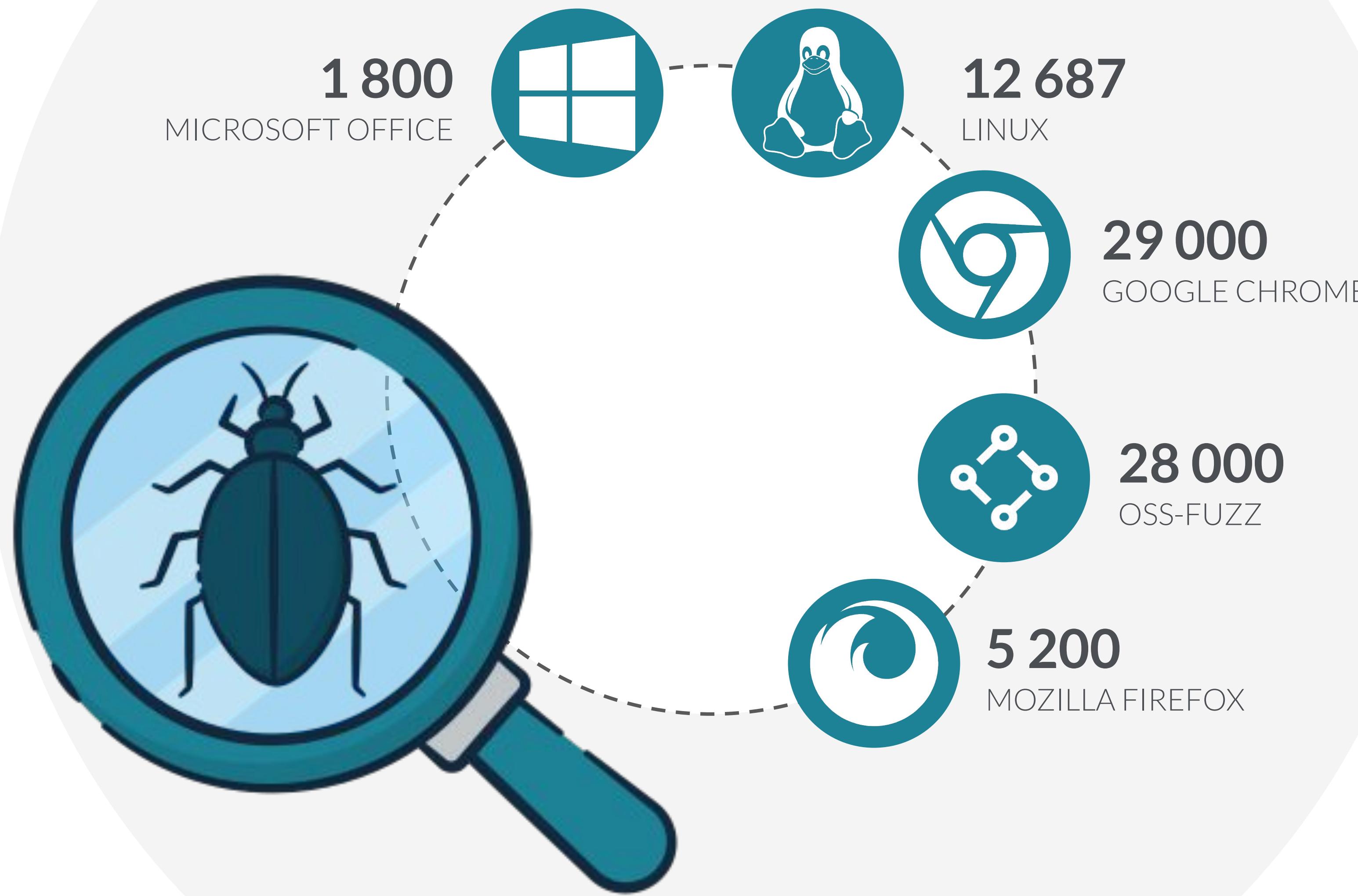


## Coverage-guided Fuzzing

- Intelligent & feedback-driven mutations
- Maximizes code coverage



# What's all the Fuzz About?



## Finding Heartbleed

This tutorial will show you how to find [Heartbleed](#) using libFuzzer and ClusterFuzz.

Abhishek Arya (@infernosec) posted a tweet on December 18, 2020, at 12:59 AM. The tweet discusses a critical DoS fix in Geth, which could affect the Ethereum network. It links to a blog post: [blog.ethereum.org/2020/11/12/geth-security-release-critical-patch-for-cve-2020-28362/](https://blog.ethereum.org/2020/11/12/geth-security-release-critical-patch-for-cve-2020-28362/). The tweet has 20 likes and a copy link option.

## 50 CVEs in 50 Days: Fuzzing Adobe Reader

December 12, 2018

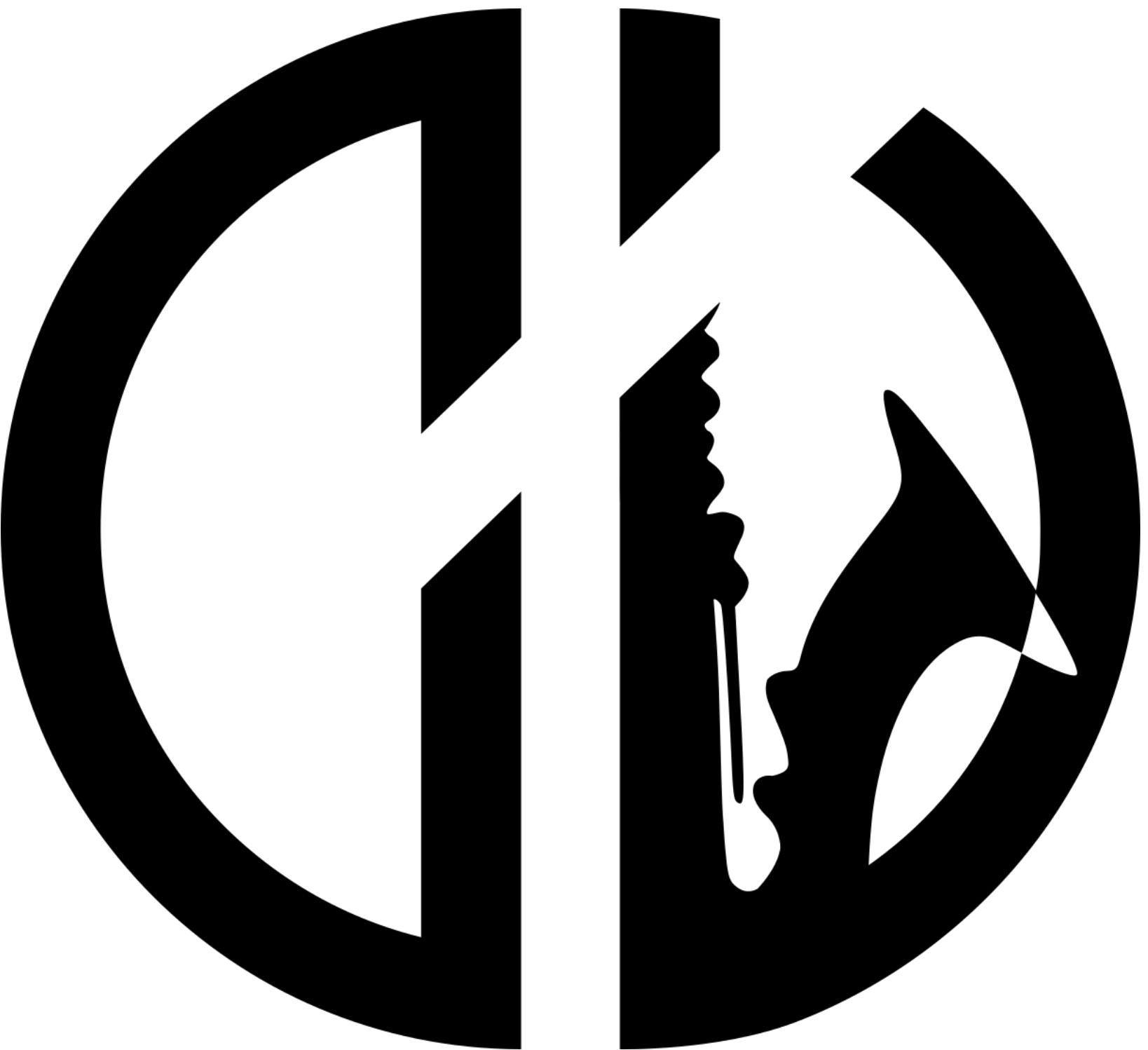
Research By: Yoav Alon, Netanel Ben-Simon

# Jazzer – Modern Fuzzing for the JVM

- Coverage-guided: based on libFuzzer & JaCoCo
- No sources required: agent-based instrumentation
- Collects dynamic data from comparisons & common functions
- Open-source since Feb 2021



[github.com/CodeIntelligenceTesting/jazzer](https://github.com/CodeIntelligenceTesting/jazzer)



# Jazzer – Modern Fuzzing for the JVM

Jazzer is available on/with:

- Linux (x64)
- macOS (x64 + aarch64)
- Windows (x64)
- Docker Hub ([cifuzz/jazzer](#) & [cifuzz/jazzer-autofuzz](#))
- Bazel ([rules\\_fuzzing](#))

# Why Fuzz Memory-Safe Languages?

Usually functional bugs:

- Uncaught exceptions
- Assertions
- Inconsistent implementations (*differential fuzzing*)

Usually security issues:

- Infinite loops
- OutOfMemoryError
- Remote Code Execution
- Injections into Domain Specific Languages (SQL, EL, Scripts, ...)
- ...

# Getting Started with Autofuzz

```
docker run \
    -v $(pwd):/fuzzing \
    -it cifuzz/jazzer-autofuzz \
    org.jsoup:jsoup:1.14.1 \
    "org.jsoup.Jsoup::parse(java.lang.String)"
```

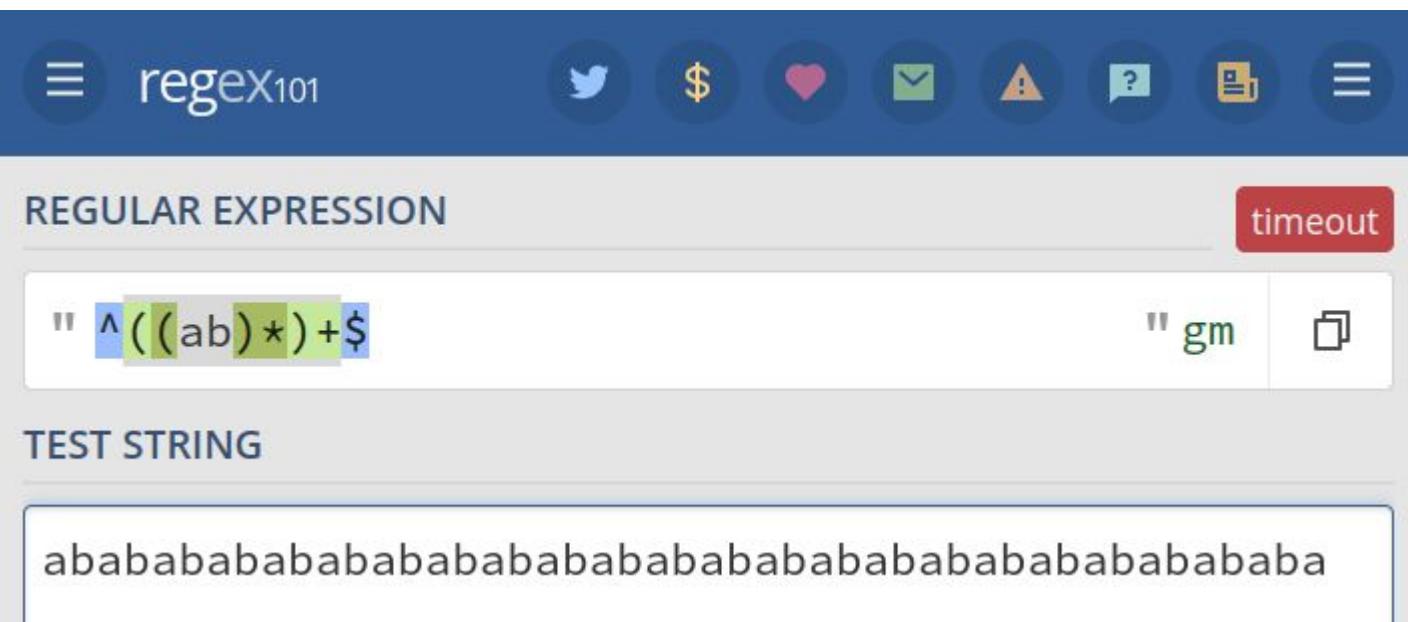
 [jsoup.org/apidocs/org/jsoup/Jsoup.html#parse\(java.lang.String\)](https://jsoup.org/apidocs/org/jsoup/Jsoup.html#parse(java.lang.String))

Optional arguments:

```
--keep_going=N          # Stop after N findings
--autofuzz_ignore=some.Exception,some.other.Exception
```

# Evil Regular Expressions

Even short regular expressions can consume exponential resources during *matching*:



Q: Are there regular expressions that consume exponential resources during *compilation*?

# Fuzzing the Java Standard Library

```
docker run \
    -v $(pwd):/fuzzing \
    -it cifuzz/jazzer \
    --autofuzz=java.util.regex.Pattern::compile \
    --autofuzz_ignore=java.lang.Exception \
    --instrumentation_includes=java.util.regex.**
```

# Writing Fuzz Targets with Jazzer

1. Add a dependency on [com.code-intelligence:jazzer-api](#).
2. Create a class with a `public static void fuzzertestOneInput(...)` method
  - a. receiving a `byte[]` with raw fuzzer input or
  - b. receiving a `FuzzedDataProvider` instance
3. Compile and package as a (deploy) jar.

<code>java.lang.String</code>	<code>consumeRemainingAsString()</code>	Consumes the remaining fuzzer input as a <code>String</code> .
<code>short</code>	<code>consumeShort()</code>	Consumes a <code>short</code> from the fuzzer input.
<code>short</code>	<code>consumeShort(short min, short max)</code>	Consumes a <code>short</code> between <code>min</code> and <code>max</code> from the fuzzer input.
<code>short[]</code>	<code>consumeShorts(int maxLength)</code>	Consumes a <code>short</code> array from the fuzzer input.
<code>java.lang.String</code>	<code>consumeString(int maxLength)</code>	Consumes a <code>String</code> from the fuzzer input.
<code>default boolean</code>	<code>pickValue(boolean[] array)</code>	Picks an element from <code>array</code> based on the fuzzer input.
<code>default byte</code>	<code>pickValue(byte[] array)</code>	Picks an element from <code>array</code> based on the fuzzer input.

# Running Fuzz Targets with Jazzer

```
docker run \
    -v $(pwd):/fuzzing \ # Should contain the jars.
    -it cifuzz/jazzer \
    --cp=project.jar:fuzzers.jar \
    --target_class=com.example.ExampleFuzzTarget
```

(or download/build release binaries and run them directly)

# Property-Based Fuzzing

## json-sanitizer build passing

Given JSON-like content, The JSON Sanitizer converts it to valid JSON.

### Output

The output is well-formed JSON as defined by [RFC 4627](#). The output satisfies these additional properties:

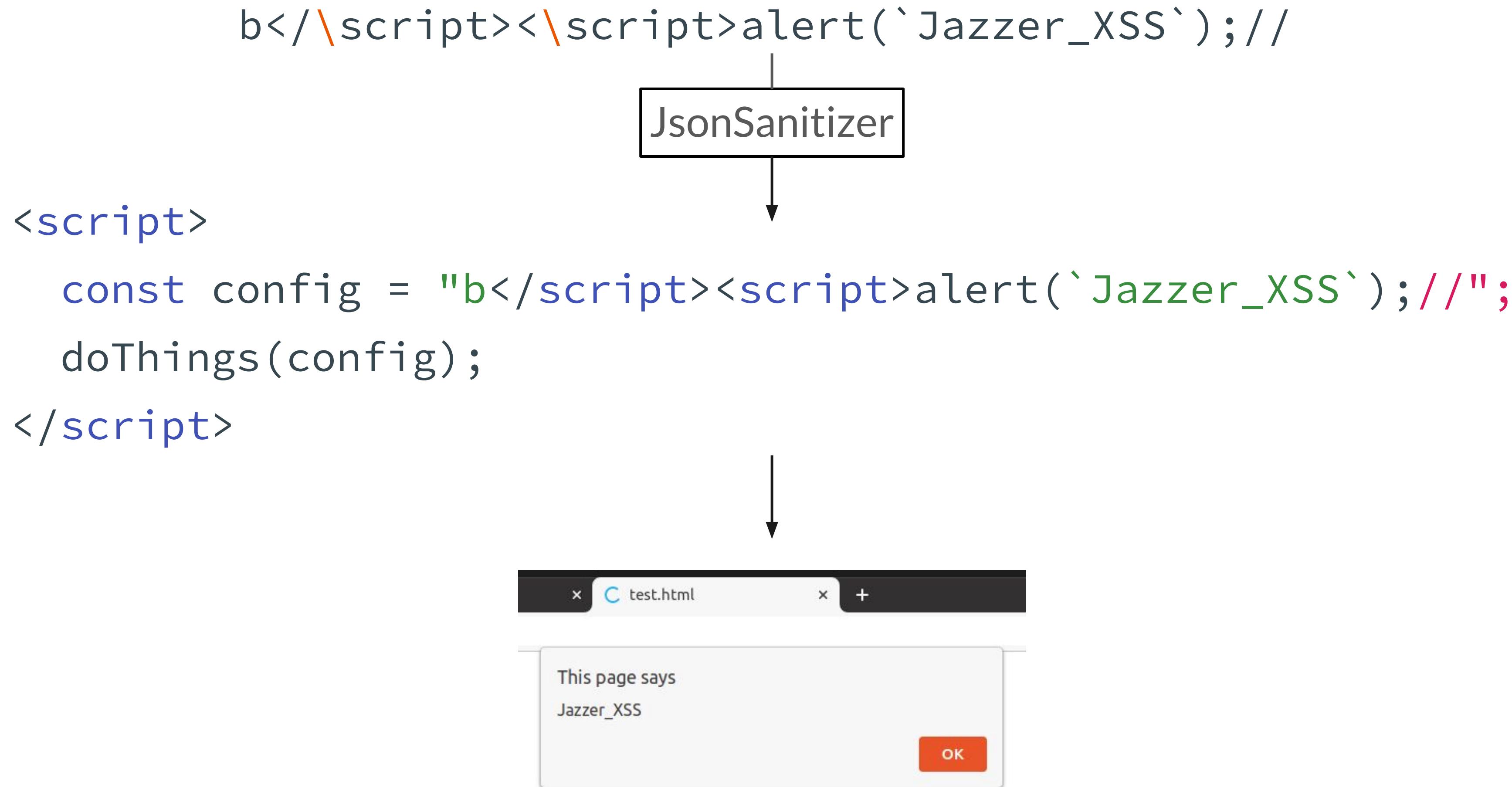
- The output will not contain the substrings (case-insensitively) "`<script`" , "`</script`" or "`<!--`" and can thus be embedded inside an HTML script element without further encoding.

```
<script>
  const config = {"foo": true, "bar": [1, 2]};
  doThings(config);
</script>
```

# Property-Based Fuzzing

```
public static void fuzzerTestOneInput(FuzzedDataProvider data) {  
  
    String input = data.consumeRemainingAsString();  
    String output = JsonSanitizer.sanitize(input);  
  
    assert !output.contains("</script>");  
}
```

# Property-Based Fuzzing



CVE-2021-23899: XSS in OWASP JsonSanitizer (+ multiple crashes)

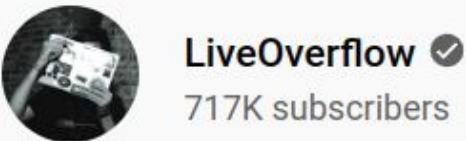
# Property-Based Fuzzing

```
#441393 REDUCE cov: 21 ft: 21 corp: 6/31b lim: 64 exec/s: 441393 rss: 271Mb L: 14/14 MS: 1 EraseBytes-  
#441940 REDUCE cov: 21 ft: 21 corp: 6/30b lim: 64 exec/s: 441940 rss: 271Mb L: 12/12 MS: 2 CopyPart-CMP- Bytes  
#442221 REDUCE cov: 21 ft: 21 corp: 6/25b lim: 64 exec/s: 44221  
#442552 REDUCE cov: 21 ft: 21 corp: 6/24b lim: 64 exec/s: 44252  
#442828 REDUCE cov: 21 ft: 21 corp: 6/22b lim: 64 exec/s: 442828  
#444104 REDUCE cov: 21 ft: 21 corp: 6/21b lim: 64 exec/s: 444104 rss: 271Mb L: 0/0 MS: 1 EraseBytes-  
#448270 REDUCE cov: 21 ft: 21 corp: 6/20b lim: 64 exec/s: 448270 rss: 271Mb L: 7/7 MS: 1 EraseBytes-  
#694457 REDUCE cov: 24 ft: 24 corp: 7/28b lim: 64 exec/s: 694457 rss: 274Mb L: 8/8 MS: 2 EraseBytes-CMP- DE: "1  
#1048576 pulse cov: 24 ft: 24 corp: 7/28b lim: 64 exec/s: 524288 rss: 274Mb  
#2097152 pulse cov: 24 ft: 24 corp: 7/28b lim: 64 exec/s: 419430 rss: 274Mb  
#4194304 pulse cov: 24 ft: 24 corp: 7/28b lim: 64 exec/s: 466033 rss: 274Mb  
#4887889 NEW cov: 27 ft: 27 corp: 8/45b lim: 64 exec/s: 444353 rss: 274Mb L: 17/17 MS: 2 CopyPart-CMP-  
#5111270 REDUCE cov: 27 ft: 27 corp: 8/45b lim: 64 exec/s: 464660 rss: 274Mb L: 16/16 MS: 1 EraseBytes-  
  
== Java Exception: java.lang.Exception: host mismatch  
name: 'ldap://localhost#.exploit.local'  
LDAP: 'ldap://localhost#.exploit.local'  
Log4j: 'ldap://localhost'  
  
at fuzz...  
DEDUP_TOKEN: 3b12a2edf1137283  
== libFuzzer crashing input ==  
MS: 2 CMP-InsertByte- DE: ".exploit.local"-; base unit: 113a48a694dc7660d3432729233ed2cb605d1619  
0x6c,0x64,0x61,0x70,0x3a,0x2f,0x2f,0x6c,0x6f,0x63,0x61,0x6c,0x68,0x6f,0x73,0x74,0x23,0x2e,0x65,0x78,0x70,0x6c,  
ldap://localhost#.exploit.local  
artifact_prefix='./'; Test unit written to ./crash-6cff77e4d6c822f2aab6a274c99ce4a08217ba78  
Base64: bGRhcDovL2xvY2FsaG9zdCMuZXhwbG9pdC5sb2Nhba==  
reproducer_path='.'; Java reproducer written to ./Crash_6cff77e4d6c822f2aab6a274c99ce4a08217ba78.java  
amlweems@errol:/vuln/log4j/fuzz$ |
```

Chapter #8: Increase Impact  
18:28

Anthony Weems (@amlweems)

Fuzzing Java to Find Log4j Vulnerability - CVE-2021-45046  
46,449 views • Feb 1, 2022



[youtube.com/watch?v=kvREvOvSWt4](https://youtube.com/watch?v=kvREvOvSWt4)

# OSS-Fuzz – Large-Scale (Java) Fuzzing

OSS-Fuzz is Google's fuzzing initiative for open-source software (est. 2016).

Jazzer has been available on OSS-Fuzz since March 2021.

Current stats:

- **23 Java projects**
  - including: jsoup, Jackson, zxing, protobuf-java, ...
- **>500 bugs found**
- **>50 issues with security impact**
- **13 CVEs for released security-critical bugs**



# "Long-tail" findings

## A potential Denial of Service issue in protobuf-java

High perezd published GHSA-wrvw-hg22-4m67 on Jan 6 (~6 months after OSS-Fuzz started fuzzing the project)

Package	Affected versions	Patched versions
com.google.protobuf:protobuf-java ( Maven )	< 3.19.2	3.16.1, 3.18.2, 3.19.2
com.google.protobuf:protobuf-kotlin ( Maven )	< 3.19.2	3.18.2, 3.19.2
google-protobuf ( RubyGems )	< 3.19.2	3.19.2

### Description

### Summary

A potential Denial of Service issue in protobuf-java was discovered in the parsing procedure for binary data.

Reporter: OSS-Fuzz

Affected versions: All versions of Java Protobufs (including Kotlin and JRuby) prior to the versions listed below. Protobuf "javalite" users (typically Android) are not affected.

### Severity

CVE-2021-22569 High - CVSS Score: 7.5, An implementation weakness in how unknown fields are parsed in Java. A small (~800 KB) malicious payload can occupy the parser for several minutes by creating large numbers of short-lived objects that cause frequent, repeated GC pauses.

# Bug Detectors – Fuzzing Beyond DoS

## Concept:

- Fuzzing is already good at dynamically exploring code and program state.
- Thus, all it takes to find vulnerabilities is to *detect them when they happen*.

## Precedent:

- AddressSanitizer (ASan)
  - Detects memory correctness issues in native code (e.g., out-of-bounds array reads/writes).
- ThreadSanitizer (TSan)
  - Detects data races in multi-threaded code (used by Go, experimental JVM support).

**Q:** Can we build similar bug detectors for common classes of Java vulnerabilities?

# Bug Detectors – Fuzzing Beyond DoS

## Implemented:

- reflective calls
- insecure deserialization
- Expression Language (EL) injection
- LDAP injection
- OS command injection (arguments aren't considered yet)
- regex injection
- JNDI source injection

Extending this list is an ongoing project with Google's OSS-Fuzz team.

# Reproducing "Log4Shell" with Jazzer

## Disclaimer:

- Hindsight is 20/20.
- Required improvements to `Map.get()` instrumentation and a bug detector for JNDI lookups.
- Only finds the vulnerability when fuzzing log4j itself.

```
public class Log4jFuzzer {  
    private static final Logger log = LogManager.getLogger(Log4jFuzzer.class.getName());  
    public static void fuzzTestOneInput(FuzzedDataProvider data) {  
        log.error(data.consumeRemainingAsString());  
    }  
    // + boilerplate to speed up log.error calls  
}
```

# Reproducing "Log4Shell" with Jazzer

```
#9801    REDUCE cov: 3372 ft: 4958 corp: 362/16Kb lim: 65 exec/s: 2450 rss: 1090Mb L: 55/65 MS: 2 ChangeByte-EraseBytes-
== Java Exception: com.code_intelligence.jazzer.api.FuzzerSecurityIssueCritical: Remote JNDI Lookup
JNDI lookups with attacker-controlled remote URLs can, depending on the JDK
version, lead to remote code execution or the exfiltration of information.
at com.code_intelligence.jazzer.sanitizers.NamingContextLookup.lookupHook(NamingContextLookup.kt:55)
at org.apache.logging.log4j.core.net.JndiManager.lookup(JndiManager.java:172)
at org.apache.logging.log4j.core.lookup.JndiLookup.lookup(JndiLookup.java:56)
at org.apache.logging.log4j.core.lookup.Interpolator.lookup(Interpolator.java:221)
at org.apache.logging.log4j.core.lookup.StrSubstitutor.resolveVariable(StrSubstitutor.java:1110)
at org.apache.logging.log4j.core.lookup.StrSubstitutor.substitute(StrSubstitutor.java:1033)
at org.apache.logging.log4j.core.lookup.StrSubstitutor.substitute(StrSubstitutor.java:912)
at org.apache.logging.log4j.core.lookup.StrSubstitutor.substitute(StrSubstitutor.java:978)
at org.apache.logging.log4j.core.lookup.StrSubstitutor.substitute(StrSubstitutor.java:912)
at org.apache.logging.log4j.core.lookup.StrSubstitutor.replace(StrSubstitutor.java:467)
at org.apache.logging.log4j.core.pattern.MessagePatternConverter.format(MessagePatternConverter.java:132)
...
at org.apache.logging.log4j.spi.AbstractLogger.error(AbstractLogger.java:740)
at com.example.Log4jFuzzer.fuzzerTestOneInput(Log4jFuzzer.java:40)
DEDUP_TOKEN: 39967e271d3922c3
== libFuzzer crashing input ==
MS: 4 ChangeByte-InsertByte-EraseBytes-PersAutoDict- DE: "ldap://g.co/-; base unit: c80f693736a2e246623e558e1f483edd3b6308c4
0x2a,0x0,0x24,0x7b,0x24,0x6a,0x27,0x29,0x76,0xff,0x61,0x2a,0x64,0x24,0x29,0x76,0x61,0x2a,0x64,0x24,0x7b,0x6a,0x6e,0x64,0x69,0x3a,
0x6c,0x64,0x61,0x70,0x3a,0x2f,0x2f,0x67,0x2e,0x63,0x6f,0x2f,0xf,0xe9,0x2a,0x29,0xd6,0x61,0x64,0xa5,0x24,0xa,0x0,0x69,0x3a,0xf,0xe9
0x2a,0x29,0x76,0x61,0x0,0x5f,0x5c,0x7d,0x7d,0x0,0x24,0x7b,
*\x00${$j')v\xffa*d${jndi:ldap://g.co/\x0f\xe9*)\xd6ad\xa5$\x0a\x00i:\x0f\xe9*)va\x00_\\"}\}\x00${
INFO: exiting: 77 time: 284s
```

# Advanced Functionality

- Track partial progress on difficult compares
  - Enable via `-use_value_profile=1`
  - Slows down fuzzing, but can uncover otherwise unreachable code paths
- Seamlessly fuzz native libraries
  - Compile and link with `-fsanitize=fuzzer-no-link, address`
- Use the Jazzer API to provide additional feedback or write custom bug detectors

```
static void exploreState(byte state, int id)  
  
static void guideTowardsContainment  
    (java.lang.String haystack,  
     java.lang.String needle, int id)  
  
static void guideTowardsEquality(byte[] current,  
                                byte[] target, int id)  
  
static void guideTowardsEquality(java.lang.String current,  
                                java.lang.String target, int id)
```

# Next Steps for Jazzer

- Make Jazzer available as a Maven dependency
  - UX goal: Fuzzing should be as accessible as Unit Testing.

```
7
8  class ParserTest {
9
10    @Test
11    void parse() {
12      assertEquals( expected: "Hello", new Parser().parse(new byte[]{5, 0x48, 0x65, 0x6c, 0x6f}));
13    }
14
15    @FuzzTest
16    void parseFuzzer(byte[] data) {
17      new Parser().parse(data);
18    }
19
```

- Extend and add bug detectors
  - Ideas (even rough ones) are very welcome!
- Improve Autofuzz' reliability and performance



[github.com/CodeIntelligenceTesting/jazze](https://github.com/CodeIntelligenceTesting/jazze)

# CI Fuzz – Web Apps, CI/CD & more

- Fuzzes multi-service deployments (REST/gRPC)
- Integrates with your CI/CD pipeline

PROJECT  
WebGoat

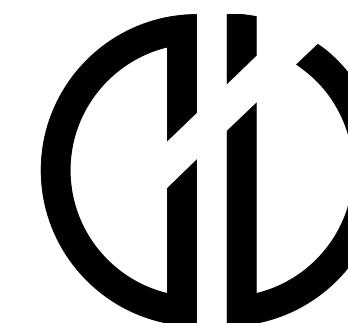
EXPORT PDF ▾ master with 3 findings ▾

Type	Severity ↑	Location	DEBUG	
SQL Injection	↑ Critical	webgoat-less.../SqlInjectionLesson4.java:52	DEBUG	
SQL Injection	↑ Critical	webgoat-less.../Assignment5.java:67	DEBUG	
<b>DETAILS</b> <b>STACKTRACE</b> <b>INPUT</b> <b>DESCRIPTION</b> <b>USEFUL LINKS</b>				
Method: POST Uri: /challenge/5?username_login=Larry&password_login=%27z Content type: text_html Body: =				
Cross Site Scripting	High	POST /WebGoat/CrossSiteScripting/phone-home-xss		

Merge branch 'master' into fuzzing\_challenge5  
Update .gitignore  
cifuzz bot commented on Jan 21  
Found "SQL Injection" while fuzzing: [View Finding](#)  
For more information, check [Code Intelligence Fuzzing Academy](#)

cifuzz bot commented on Jan 21

File	Coverage
webgoat-server/src/main/java/org/owasp/webgoat	
HSQldbDatabaseConfig.java	100%
StartWebGoat.java	100%
webgoat-less.../secure-passwords/src/main/java/org/owasp/webgoat/secure_password	
SecurePasswordsAssignment.java	100%
SecurePasswords.java	100%



code-intelligence.com