Vorlesung Fortgeschrittene Softwaretechnik

Wintersemester 2024/25

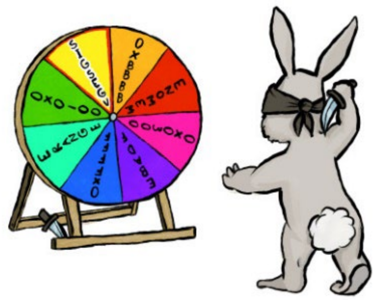
Prof. Dr. Stephan Diehl

Lucas Kreber, M.Sc.

Informatik

Universität Trier

Fuzzing



- **Eingabeschnittstelle** wird mit **zufällig** erzeugten Daten ausgeführt, z.B. Funktion mit Zufallsparametern aufgerufen, Zufallsnachrichten geschickt, Tastendrücke simuliert, etc.

- Konkrete Fuzzing-Methoden unterscheiden sich u.a. darin, welches Vorwissen über die zu testende Software benötigt wird (Blackbox bis Whitebox-Testen):
  - **Mutation-based Fuzzing**: Zufällige Änderungen werden in bekannte, gültige Eingaben eingebaut.
  - **Generation-based Fuzzing**: Eingaben werden aufgrund einer Spezifikation des Eingabeformats erzeugt. Zufällige, nicht der Spezifikation entsprechende Änderungen werden eingebaut.

Änderungen sind auf verschiedenen Ebenen möglich:

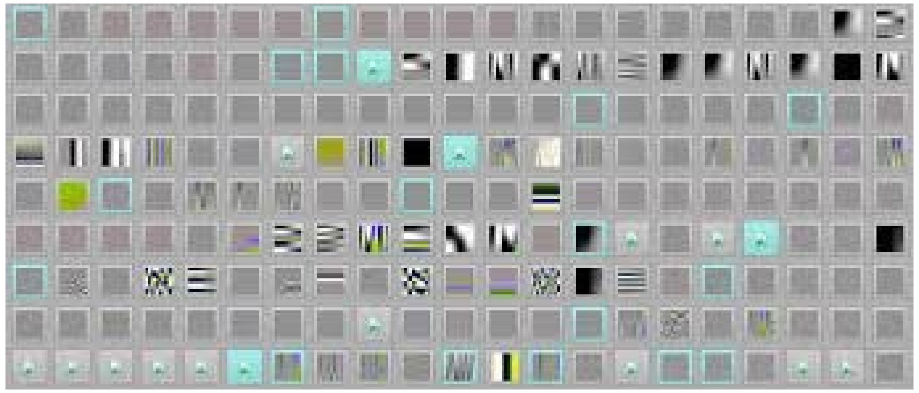
- Bits (z.B. Bit-Flipping)
- Operatoren (z.B. * statt +)
- Blöcke (z.B. Vertauschen von Blöcken)

Eingabeformate „entdecken“

- **Coverage-guided Mutation-based Fuzzing**: Sammele Coverage-Informationen während der Ausführung des Programms. Wähle solche Eingaben für die nächste Mutation, die neue Programmzweige ausgeführt haben.
- Beispiel: **afl-fuzz**
  - angewandt auf djpeg mit „hello“ als seed erzeugt JPEGs als Eingabe:

```
$ echo 'hello' >in_dir/hello
$ ./afl-fuzz -i in_dir -o out_dir ./jpeg-9a/djpeg
```

lcamtuf's old blog: Pulling JPEGs out of thin air



“The first image, hit after about six hours on an 8-core system”

Variante: **PerfFuzz** [Lemieux et al. 2018] entdeckt Eingaben für **Worst-Case Performanz**, indem es Eingaben wiederverwendet, die die Anzahl der Ausführungen eines Programmzweiges maximieren.

Links für den praktischen Teil

- Jazzer
  - https://github.com/CodeIntelligenceTesting/jazzer
  - https://www.code-intelligence.com/blog/java-fuzzing-with-jazzer
- libFuzzer Tutorial
  - https://aviii.hashnode.dev/the-art-of-fuzzing-a-step-by-step-guide-to-coverage-guided-fuzzing-with-libfuzzer
- AFL++
  - https://github.com/AFLplusplus/AFLplusplus

JAZZER

JAZZER

- **GitHub releases:** https://github.com/CodeIntelligenceTesting/jazzer/releases

You can also use GitHub release archives to run a standalone Jazzer binary that starts its own JVM configured for fuzzing:

- Download and extract the latest release from the GitHub releases page.
- Add a new class to your project with a **public static void fuzzerTestOneInput(FuzzedDataProvider data)** method.
- Compile your fuzz test with **jazzer_standalone.jar** on the classpath.
- Run the jazzer binary (jazzer.exe on Windows), specifying the classpath and fuzz test class:
- ./jazzer --cp=<classpath> --target_class=<fuzz test class>
- If you see an error saying that libjvm.so has not been found, make sure that JAVA_HOME points to a JDK.

- The examples directory includes both toy and real-world examples of fuzz tests.

JAZZER

Since coverage is not the only type of information that is used by libFuzzer to guide its exploration of the fuzz target, Jazzer also instruments other JVM constructs (see TraceDataFlowInstrumentor.kt):

- bytecode-level compares, such as the **lcmp**, **if_*,** and **if*** opcodes
- higher-level method-based compares, such as **String#equal** or **Arrays#compare**
- switch statements (corresponding to the **lookupswitch** and **tableswitch** opcodes)
- integer divisions (corresponding to the **idiv** and **ldiv** opcodes and the **divideUnsigned** methods)
- constant array indices (corresponding to the ***aload** opcodes and **get*** methods on array-like objects)
- In all these cases, interesting observed values are forwarded to libFuzzer callbacks such as __sanitizer_cov_trace_cmp4 (for comparisons of 32-bit integral types) and __sanitizer_weak_hook_strcmp (for string comparisons). **The values populate libFuzzer's table of recent compares and will be incorporated into the fuzzer input bytes via random mutations.**

Beispiel: ExampleFuzzer.java

```
/* Copyright 2024 Code Intelligence GmbH
 * By downloading, you agree to the Code Intelligence Jazzer Terms and Conditions.
 * The Code Intelligence Jazzer Terms and Conditions are provided in LICENSE-JAZZER.txt
 * located in the root directory of the project. */
```

```java
import com.code_intelligence.jazzer.api.FuzzedDataProvider;
import com.code_intelligence.jazzer.api.FuzzerSecurityIssueMedium;
import java.security.SecureRandom;

public class ExampleFuzzer {
  public static void fuzzerInitialize() {
    // Optional initialization to be run before the first call to fuzzerTestOneInput.
  }

  public static void fuzzerTestOneInput(FuzzedDataProvider data) {
    String input = data.consumeRemainingAsString();
    // Without the hook in ExampleFuzzerHooks.java, the value of random would change on every
    // invocation, making it almost impossible to guess for the fuzzer.
    long random = new SecureRandom().nextLong();
    if (input.startsWith("magicstring" + random)
        && input.length() > 30
        && input.charAt(25) == 'C') {
      mustNeverBeCalled();
    }
  }

  private static void mustNeverBeCalled() {
    throw new FuzzerSecurityIssueMedium("mustNeverBeCalled has been called");
  }
}
```

Methoden von FuzzedDataProvider:

- consumeString(int maxLength)
- **consumeRemainingAsString()** turns raw fuzzer input bytes into valid JVM strings by carefully applying the least number of bit changes required to turn the input bytes into valid UTF-8.
- consumeByte()
- consumeBytes(int maxLength) → returns a byte array
- consumeRemainingAsBytes()
- pickValue(T[] array) → returns an element from array based on fuzzer input
- …

Beispiel: ExampleFuzzer.java

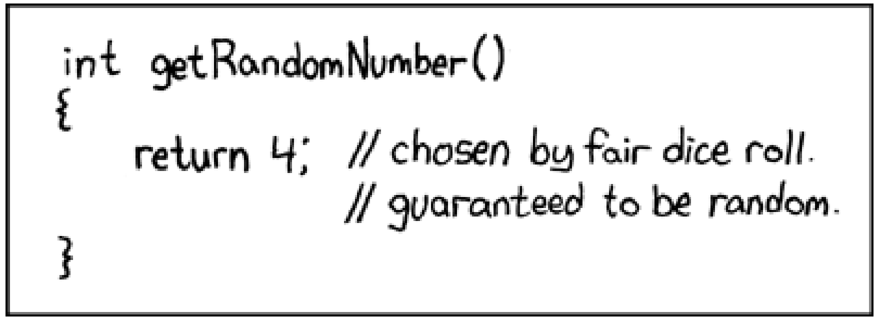
```
• C:\Users\diehl\Desktop\Fuzzing\JAZZER>java -cp .\;"C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot"\lib;.\jazzer_standalone.jar
  com.code_intelligence.jazzer.Jazzer --target_class=ExampleFuzzer
• WARNING: A Java agent has been loaded dynamically (C:\Users\diehl\AppData\Local\Temp\byteBuddyAgent6097447167869535372.jar)
• WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning
• WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more information
• WARNING: Dynamic loading of agents will be disallowed by default in a future release
• OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
• INFO: Loaded 278 hooks from com.code_intelligence.jazzer.runtime.TraceCmpHooks
• ...
• INFO: Instrumented ExampleFuzzer (took 335 ms, size +45%)
• INFO: found LLVMFuzzerCustomMutator (00007FFD5E6633D0). Disabling -len_control by default.
• INFO: libFuzzer ignores flags that start with '--'
• INFO: Running with entropic power schedule (0xFF, 100).
• INFO: Seed: 4287827936
• INFO: Loaded 1 modules   (512 inline 8-bit counters): 512 [000001FCAF36A040, 000001FCAF36A240),
• INFO: Loaded 1 PC tables (512 PCs): 512 [000001FCAA9257E0,000001FCAA9277E0),
• INFO: -max_len is not provided; libFuzzer will not generate inputs larger than 4096 bytes
• INFO: A corpus is not provided, starting from an empty corpus
• #2      INITED cov: 4 ft: 4 corp: 1/1b exec/s: 0 rss: 1449Mb
• #1048576       pulse  cov: 4 ft: 4 corp: 1/1b lim: 4096 exec/s: 349525 rss: 2400Mb
• #2097152       pulse  cov: 4 ft: 4 corp: 1/1b lim: 4096 exec/s: 349525 rss: 2400Mb
• #4194304       pulse  cov: 4 ft: 4 corp: 1/1b lim: 4096 exec/s: 381300 rss: 2400Mb
• #8388608       pulse  cov: 4 ft: 4 corp: 1/1b lim: 4096 exec/s: 381300 rss: 2400Mb
• #16777216      pulse  cov: 4 ft: 4 corp: 1/1b lim: 4096 exec/s: 372827 rss: 2400Mb
• #33554432      pulse  cov: 4 ft: 4 corp: 1/1b lim: 4096 exec/s: 372827 rss: 2400Mb
• #67108864      pulse  cov: 4 ft: 4 corp: 1/1b lim: 4096 exec/s: 370767 rss: 2400Mb
• #134217728     pulse  cov: 4 ft: 4 corp: 1/1b lim: 4096 exec/s: 371794 rss: 2400Mb
• #268435456     pulse  cov: 4 ft: 4 corp: 1/1b lim: 4096 exec/s: 371794 rss: 2400Mb
• #536870912     pulse  cov: 4 ft: 4 corp: 1/1b lim: 4096 exec/s: 371794 rss: 2400Mb
• #1073741824     pulse  cov: 4 ft: 4 corp: 1/1b lim: 4096 exec/s: 368097 rss: 2400Mb
```

Abgebrochen, da die Suche zu lange dauert.

Beispiel: ExampleFuzzerHooks.java

/* Copyright 2024 Code Intelligence GmbH
 * By downloading, you agree to the Code Intelligence Jazzer Terms and Conditions.
 * The Code Intelligence Jazzer Terms and Conditions are provided in LICENSE-JAZZER.txt
 * located in the root directory of the project. */

```java
import com.code_intelligence.jazzer.api.HookType;
import com.code_intelligence.jazzer.api.MethodHook;
import java.lang.invoke.MethodHandle;

public class ExampleFuzzerHooks {
    @MethodHook(
            type = HookType.REPLACE,
            targetClassName = "java.security.SecureRandom",
            targetMethod = "nextLong",
            targetMethodDescriptor = "()J")
    public static long getRandomNumber(
            MethodHandle handle, Object thisObject, Object[] args, int hookId) {
        return 4; // chosen by fair dice roll.
        // guaranteed to be random.
        // https://xkcd.com/221/
    }
}
```



```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

```
java -cp .\;"C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot"\lib;.\jazzer_standalone.jar com.code_intelligence.jazzer.Jazzer --target_class=ExampleFuzzer --custom_hooks=ExampleFuzzerHooks
WARNING: A Java agent has been loaded dynamically (C:\Users\diehl\AppData\Local\Temp\byteBuddyAgent14814129946866390780.jar)
WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a future release
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
INFO: Loaded 278 hooks from com.code_intelligence.jazzer.runtime.TraceCmpHooks
...
INFO: Loaded 1 hooks from ExampleFuzzerHooks
INFO: Instrumented ExampleFuzzer (took 102 ms, size +56%)
INFO: found LLVMFuzzerCustomMutator (00007FFD8B7E33D0). Disabling -len_control by default.
INFO: libFuzzer ignores flags that start with '--'
INFO: Running with entropic power schedule (0xFF, 100).
INFO: Seed: 4185080285
INFO: Loaded 1 modules   (512 inline 8-bit counters): 512 [000002B9D1DD5040, 000002B9D1DD5240),
INFO: Loaded 1 PC tables (512 PCs): 512 [000002B9FE4E9E50,000002B9FE4EBE50),
INFO: -max_len is not provided; libFuzzer will not generate inputs larger than 4096 bytes
INFO: A corpus is not provided, starting from an empty corpus
#2      INITED cov: 4 ft: 4 corp: 1/1b exec/s: 0 rss: 1397Mb
#1308   NEW    cov: 6 ft: 6 corp: 2/14b lim: 14 exec/s: 0 rss: 1401Mb L: 13/13 MS: 2 CMP-Custom- DE: "magicstring4"-
#1402   REDUCE cov: 6 ft: 6 corp: 2/13b lim: 14 exec/s: 0 rss: 1402Mb L: 12/12 MS: 8 ChangeBit-Custom-CMP-Custom-EraseBytes-Custom-CMP-Custom- DE: "magicstring4"-"magicstring4"-
#3421   REDUCE cov: 8 ft: 8 corp: 3/44b lim: 33 exec/s: 0 rss: 1404Mb L: 31/31 MS: 8 ShuffleBytes-Custom-ChangeBinInt-Custom-CMP-Custom-CopyPart-Custom- DE: "magicstring4"-

== Java Exception: com.code_intelligence.jazzer.api.FuzzerSecurityIssueMedium: mustNeverBeCalled has been called
        at ExampleFuzzer.mustNeverBeCalled(ExampleFuzzer.java:34)
        at ExampleFuzzer.fuzzerTestOneInput(ExampleFuzzer.java:29)
DEDUP_TOKEN: e3a40a92890b8e76
== libFuzzer crashing input ==
MS: 2 ChangeBit-Custom-; base unit: a6f2976a611f0bf6e697cca215146cf8477b7c67
0x6d,0x61,0x67,0x69,0x63,0x73,0x74,0x72,0x69,0x6e,0x67,0x34,0x6d,0x67,0x69,0x61,0x63,0x73,0x74,0x72,0x68,0x6e,0x67,0x34,0x61,0x43,0x73,0x74,0x72,0x68,0x6e,
magicstring4mgiacstrhng4aCstrhn
artifact_prefix='./'; Test unit written to ./crash-0c1791fda689001e383e01e24b7dffc0a4b372d4
Base64: bWFnaWNzdHJpbmc0bWdpYWNzdHJobmc0YUNzdHJobg==
reproducer_path='.'; Java reproducer written to .\Crash_0c1791fda689001e383e01e24b7dffc0a4b372d4.java
```

Beispiel: Crash_0c1791fda689001e383e01e24b7dffc0a4b372d4.java

```
mport java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

public class Crash_0c1791fda689001e383e01e24b7dffc0a4b372d4 {
   static final String base64Bytes = String.join("",    "rO0ABXNyABNqYXZhLnV0aWwuQXJyYXlMaXN0eIHSHZnHYZ0DAAFJAARzaXple "+
                                                         "HAAAAABdwQAAAABdAAfbWFnaWNzdHJpbmc0bWdpYWNzdHJobmc0YUNzdHJobng=");

   public static void main(String[] args) throws Throwable {
      Crash_0c1791fda689001e383e01e24b7dffc0a4b372d4.class.getClassLoader().setDefaultAssertionStatus(true);
      try {
         Method fuzzerInitialize = ExampleFuzzer.class.getMethod("fuzzerInitialize");
         fuzzerInitialize.invoke(null);
      } catch (NoSuchMethodException ignored) {
         try {
            Method fuzzerInitialize = ExampleFuzzer.class.getMethod("fuzzerInitialize", String[].class);
            fuzzerInitialize.invoke(null, (Object) args);
         } catch (NoSuchMethodException ignored1) {
         } catch (IllegalAccessException | InvocationTargetException e) {
            e.printStackTrace();
            System.exit(1);
         }
      } catch (IllegalAccessException | InvocationTargetException e) {
         e.printStackTrace();
         System.exit(1);
      }
      com.code_intelligence.jazzer.api.CannedFuzzedDataProvider input = new com.code_intelligence.jazzer.api.CannedFuzzedDataProvider(base64Bytes);
      ExampleFuzzer.fuzzerTestOneInput(input);
   }
}
```

JUNIT

https://junit.org/junit4/

https://github.com/junit-team/junit4/

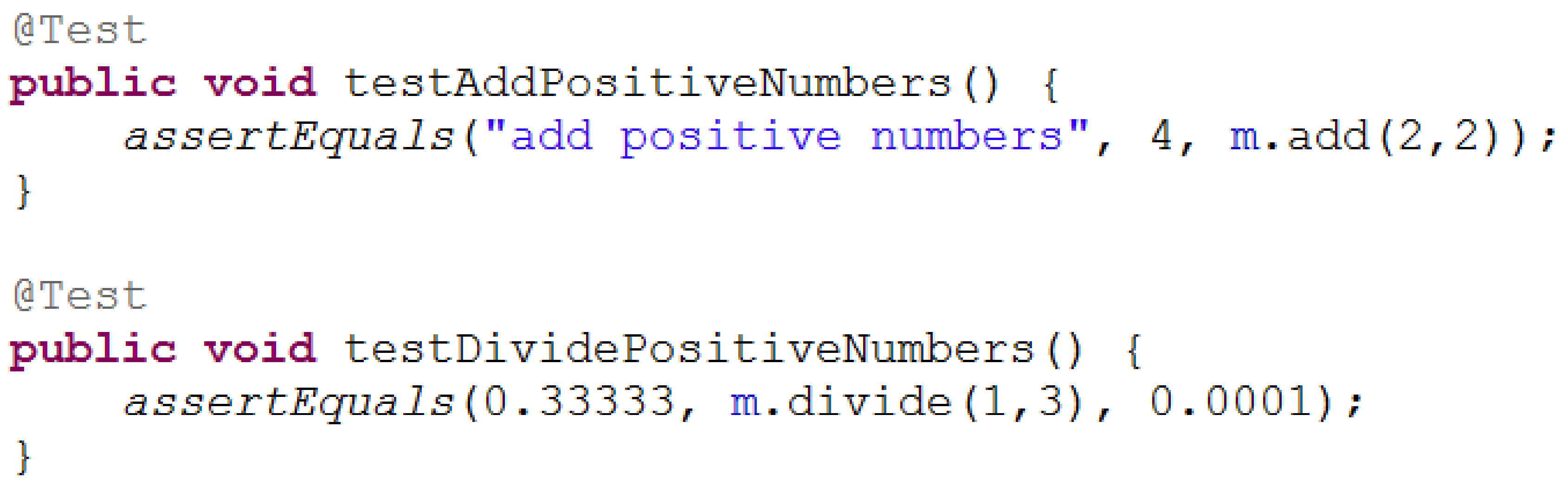
Annotations in Junit 4

Konventionen:
Testklassen: …Test
Testmethoden: test…()
Before/After: setUp(), tearDown()

- `@Test`
  Test-Methode
- `@Before/@After`
  Wird vor/nach jedem Test ausgeführt
- `@BeforeClass/@AfterClass`
  Wird einmalig vor/nach allen Tests der Klasse ausgeführt
- `@Ignore`
  „Auskommentierter" Test

Assertions (Zusicherungen)

- Methoden der Klasse `org.junit.Assert`
  - `assertEquals(expected, actual)`
  - `assertEquals(message, expected, actual)`
- Fließkommazahlen:
  - `assertEquals(expected, actual, epsilon)`
- Bedingungen:
  - `assertTrue(condition)`
- Beispiele:



```
@Test
public void testAddPositiveNumbers() {
    assertEquals("add positive numbers", 4, m.add(2,2));
}

@Test
public void testDividePositiveNumbers() {
    assertEquals(0.33333, m.divide(1,3), 0.0001);
}
```

Fehlerfälle testen

- Variante 1 (**`fail(…)`-Methode**):

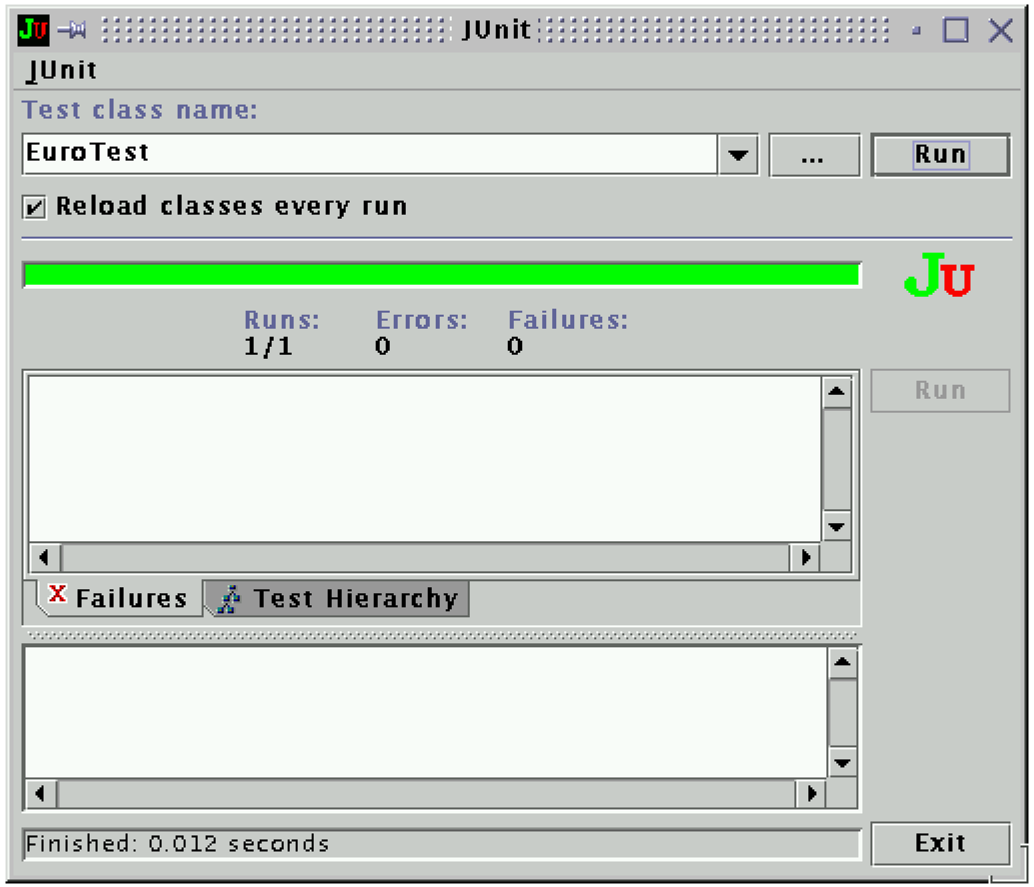
```java
@Test
public void testDivideByZero1() {
    try {
        m.divide(1,0);
        fail("Should not be possible to divide by zero");
    } catch (ArithmeticException e) {
    }
}
```

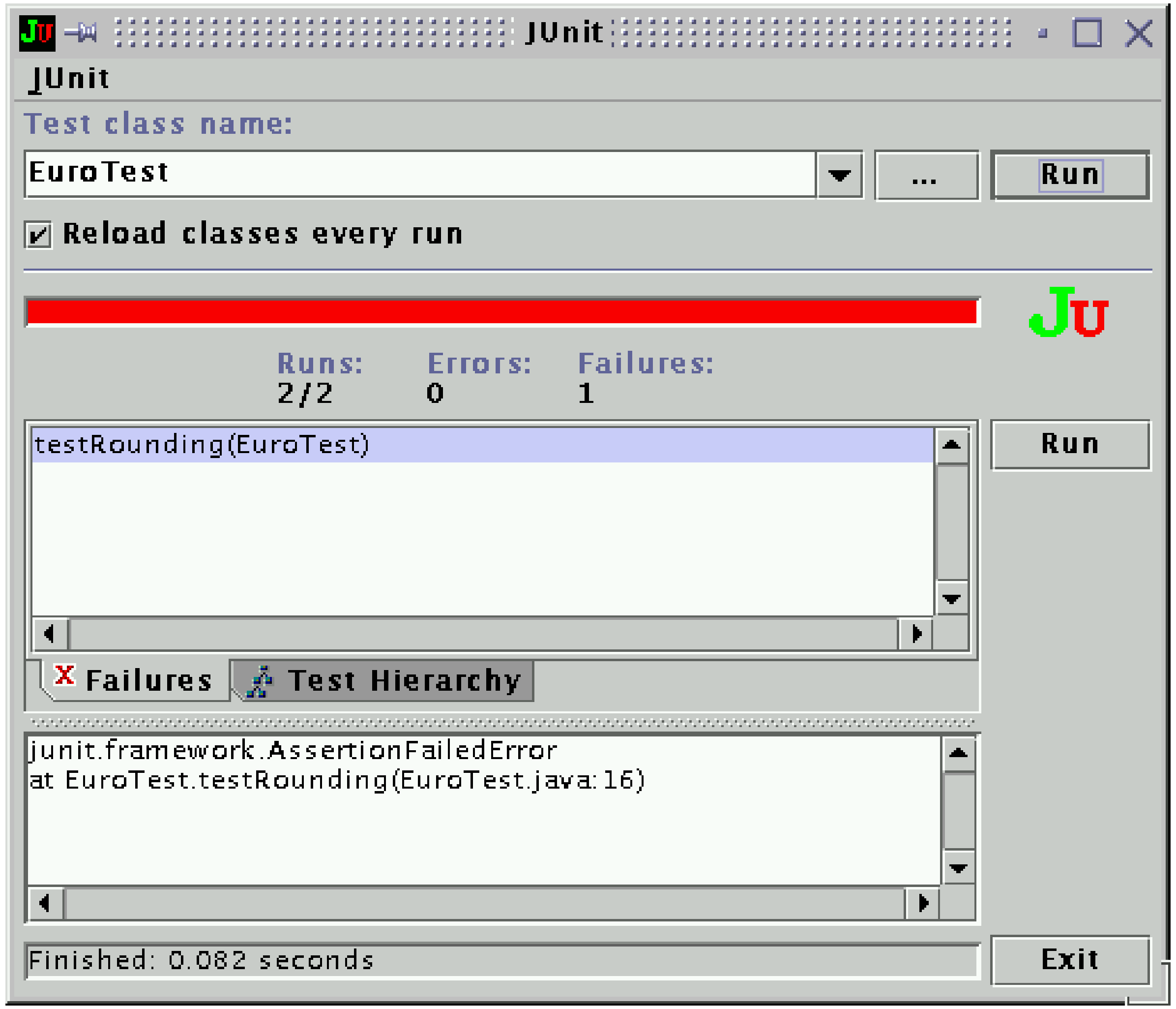
Wenn das Programm hier ankommt, wurde keine Exception ausgelöst!

- Variante 2 (**Assertion**):

```java
@Test (expected = ArithmeticException.class)
public void testDivideByZero2() {
    m.divide(1,0);
}
```

„Keep the bar green, to keep the code clean.“





```
testRounding(EuroTest)
```

```
junit.framework.AssertionFailedError
at EuroTest.testRounding(EuroTest.java:16)
```

JUNIT: Beispiel

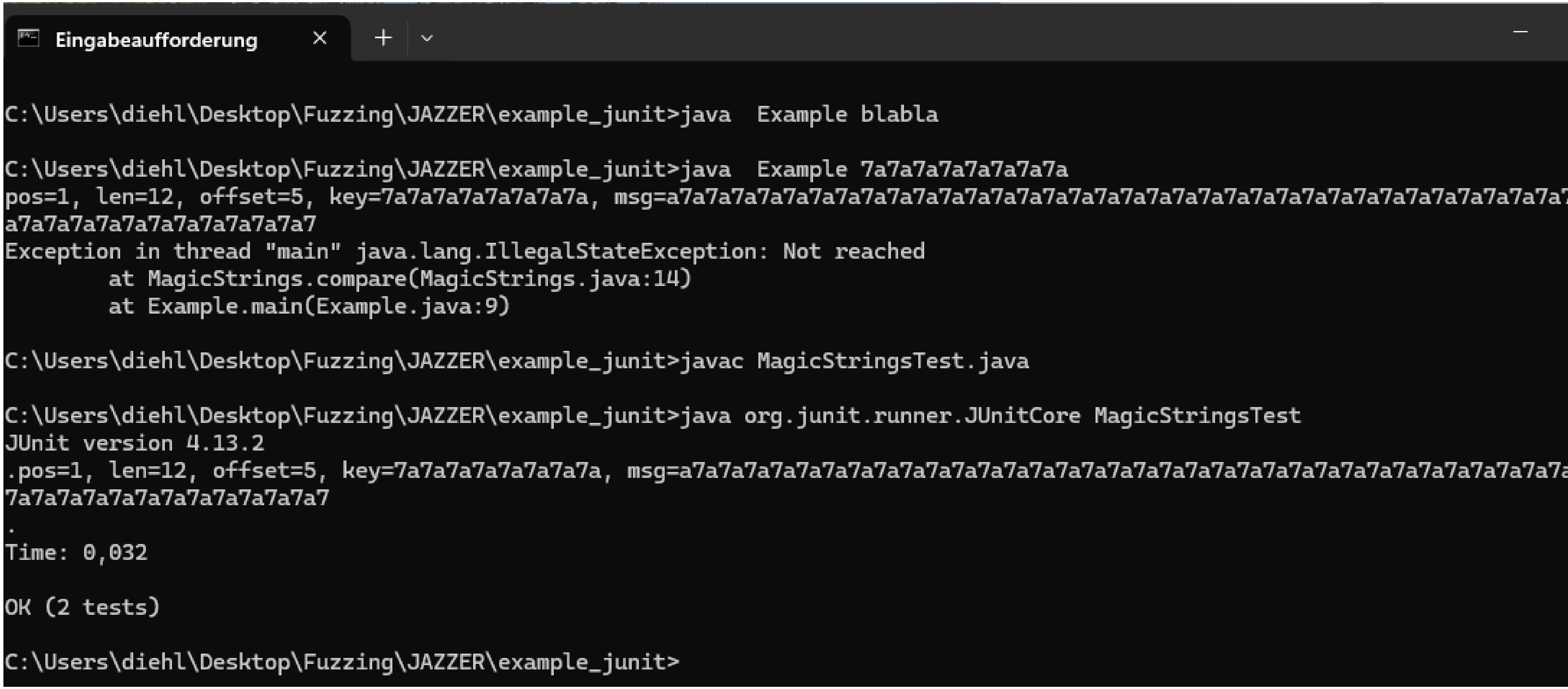
```
import java.nio.charset.StandardCharsets;

public class MagicStrings {

   public static void compare(int pos, int len, int offset, String key, String msg) {
      if (len > 10)
         if (key.length() > 5)
            if (key.charAt(4) == '7')
               if (onlyValidCharacters(key))
                  if (onlyValidCharacters(msg))
                     if (key.substring(pos, pos + len).equals(msg.substring(pos + offset, pos + offset + len))) {
                        System.out.println("pos=" + pos + ", len=" + len + ", offset=" + offset + ", key=" + key + ", msg=" + msg);
                        throw new IllegalStateException("Not reached");
                     }
   }

   static boolean onlyValidCharacters(String s) {
      byte[] bs = s.getBytes(StandardCharsets.UTF_8);
      for (int i = 0; i < bs.length; i++) {
         byte a = bs[i];
         if ((a < '0') || (a > 'z')) return false;
      }
      return true;
   }
}
```

JUNIT: Beispiel

```
public class Example {

    public static void main(String args[]) {
            if (args.length<1) {
                    System.out.println("No data provided as command line argument!");
            } else {

              MagicStrings.compare(1,12,5,args[0],"a7".repeat(50));

            }
    }
}
```

JUNIT: Beispiel

```
import org.junit.*;

public class MagicStringsTest {

  @Test (expected=NullPointerException.class)
  public void testNullPointer() {
    MagicStrings.compare(1, 12, 5, "7a7a7a7a7a7a7a7a", null);
   }

  @Test (expected=IllegalStateException.class)
  public void testMagicCombination() {
    MagicStrings.compare(1, 12, 5, "7a7a7a7a7a7a7a7a", "a7".repeat(50));
   }


}
```

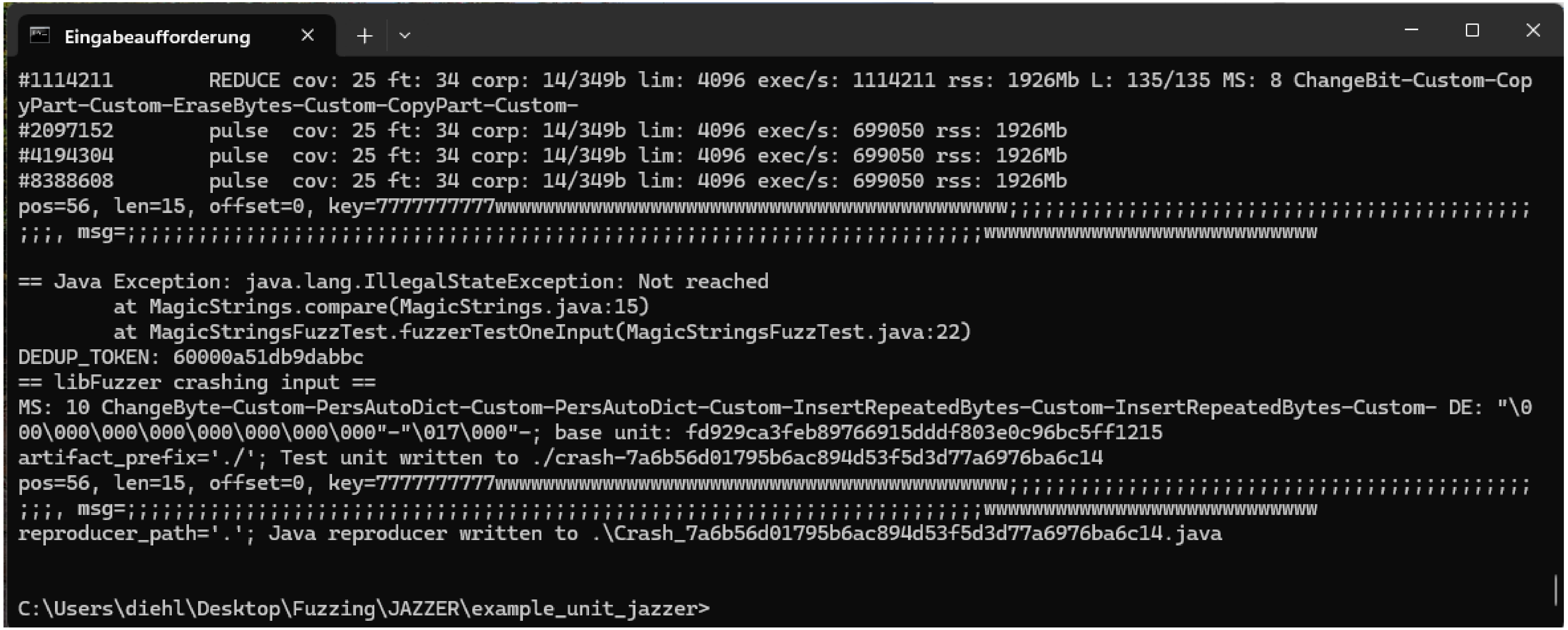


```
C:\Users\diehl\Desktop\Fuzzing\JAZZER\example_junit>java  Example blabla

C:\Users\diehl\Desktop\Fuzzing\JAZZER\example_junit>java  Example 7a7a7a7a7a7a7a7a
pos=1, len=12, offset=5, key=7a7a7a7a7a7a7a7a, msg=a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a
a7a7a7a7a7a7a7a7a7a7a7a7
Exception in thread "main" java.lang.IllegalStateException: Not reached
        at MagicStrings.compare(MagicStrings.java:14)
        at Example.main(Example.java:9)

C:\Users\diehl\Desktop\Fuzzing\JAZZER\example_junit>javac MagicStringsTest.java

C:\Users\diehl\Desktop\Fuzzing\JAZZER\example_junit>java org.junit.runner.JUnitCore MagicStringsTest
JUnit version 4.13.2
.pos=1, len=12, offset=5, key=7a7a7a7a7a7a7a7a, msg=a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a7a
7a7a7a7a7a7a7a7a7a7a7a7a7
.
Time: 0,032

OK (2 tests)

C:\Users\diehl\Desktop\Fuzzing\JAZZER\example_junit>
```

JAZZER and JUNIT: Beispiel

```java
import com.code_intelligence.jazzer.api.FuzzedDataProvider;
import com.code_intelligence.jazzer.junit.FuzzTest;

public class MagicStringsFuzzTest {

  @FuzzTest
  public static void fuzzerTestOneInput(FuzzedDataProvider data) {

      String key = data.consumeString(100);
      String msg = data.consumeString(100);

      int pos = data.consumeInt();
      int len = data.consumeInt();
      int offset = data.consumeInt();

      try {

          MagicStrings.compare(pos,len,offset,key,msg);

      } catch (StringIndexOutOfBoundsException e) { } // Dieser Fehler soll beim Testen ignoriert werden
  }

}
```

JAZZER and JUNIT: Beispiel



```
#1114211        REDUCE cov: 25 ft: 34 corp: 14/349b lim: 4096 exec/s: 1114211 rss: 1926Mb L: 135/135 MS: 8 ChangeBit-Custom-Cop
yPart-Custom-EraseBytes-Custom-CopyPart-Custom-
#2097152        pulse  cov: 25 ft: 34 corp: 14/349b lim: 4096 exec/s: 699050 rss: 1926Mb
#4194304        pulse  cov: 25 ft: 34 corp: 14/349b lim: 4096 exec/s: 699050 rss: 1926Mb
#8388608        pulse  cov: 25 ft: 34 corp: 14/349b lim: 4096 exec/s: 699050 rss: 1926Mb
pos=56, len=15, offset=0, key=7777777777wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;, msg=;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;wwwwwwwwwwwwwwwwwwwwwwwwwwww

== Java Exception: java.lang.IllegalStateException: Not reached
        at MagicStrings.compare(MagicStrings.java:15)
        at MagicStringsFuzzTest.fuzzerTestOneInput(MagicStringsFuzzTest.java:22)
DEDUP_TOKEN: 60000a51db9dabbc
== libFuzzer crashing input ==
MS: 10 ChangeByte-Custom-PersAutoDict-Custom-PersAutoDict-Custom-InsertRepeatedBytes-Custom-InsertRepeatedBytes-Custom- DE: "\0
00\000\000\000\000\000\000\000"-"\017\000"-; base unit: fd929ca3feb89766915dddf803e0c96bc5ff1215
artifact_prefix='./'; Test unit written to ./crash-7a6b56d01795b6ac894d53f5d3d77a6976ba6c14
pos=56, len=15, offset=0, key=7777777777wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;, msg=;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;wwwwwwwwwwwwwwwwwwwwwwwwwwww
reproducer_path='.'; Java reproducer written to .\Crash_7a6b56d01795b6ac894d53f5d3d77a6976ba6c14.java


C:\Users\diehl\Desktop\Fuzzing\JAZZER\example_unit_jazzer>
```

Aufgabe1: EmailMatcher

1. Laden Sie die Dateien aus dem Praxis-Ordner auf ihren Rechner und probieren Sie die Beispiele (für Windows oder Ubuntu-Linux) aus.
2. Erstellen Sie eine Datei EmailMatcherFuzzTest.java
3. Implementieren Sie darin einen Test, mit dem gültige Emailadressen gesucht werden sollen.
4. Passen Sie die Batch-Datei bzw. das Shell-Skript an.
5. Führen Sie ihren FuzzTest mehrfach aus und sammeln Sie die gefundenen gültigen Emailadressen.
6. Weichen die Emailadressen von ihren Erwartungen ab?
7. Beschreiben Sie Ihre Beobachtungen und ihre Vermutungen zur Ursache der Abweichungen.

Als Ausgangspunkt verwenden Sie am besten das Verzeichnis exampe_unit_jazzer

Aufgabe 2: UserRegistration

1. Erstellen Sie eine Datei UserRegistrationFuzzTest.java
2. Implementieren Sie darin einen Test, mit dem gültige Kombinationen von Benutzername, Passwort und Emailadresse gesucht werden.
3. Passen Sie die Batch-Datei bzw. das Shell-Skript an.
4. Führen Sie ihren FuzzTest aus.
5. Brechen Sie ggf. nach einigen Minuten den Test ab, falls keine Kombination gefunden wird.
6. Um die Suche zu beschleunigen, können Sie für die Email eine feste Adresse, z.B. aus Aufgabe 1, verwenden.
7. Führen Sie ihre FuzzTests mehrfach aus und sammeln Sie die gefundenen gültigen Kombinationen.
8. Weichen die Emailadressen von ihren Erwartungen ab?
9. Beschreiben Sie Ihre Beobachtungen und ihre Vermutungen zur Ursache der Abweichungen.

```
import org.apache.commons.lang3.StringEscapeUtils;

String euser=StringEscapeUtils.escapeJava(user);
```

Java in Ubuntu

- https://adoptium.net/de/installation/
- https://adoptium.net/de/installation/linux/