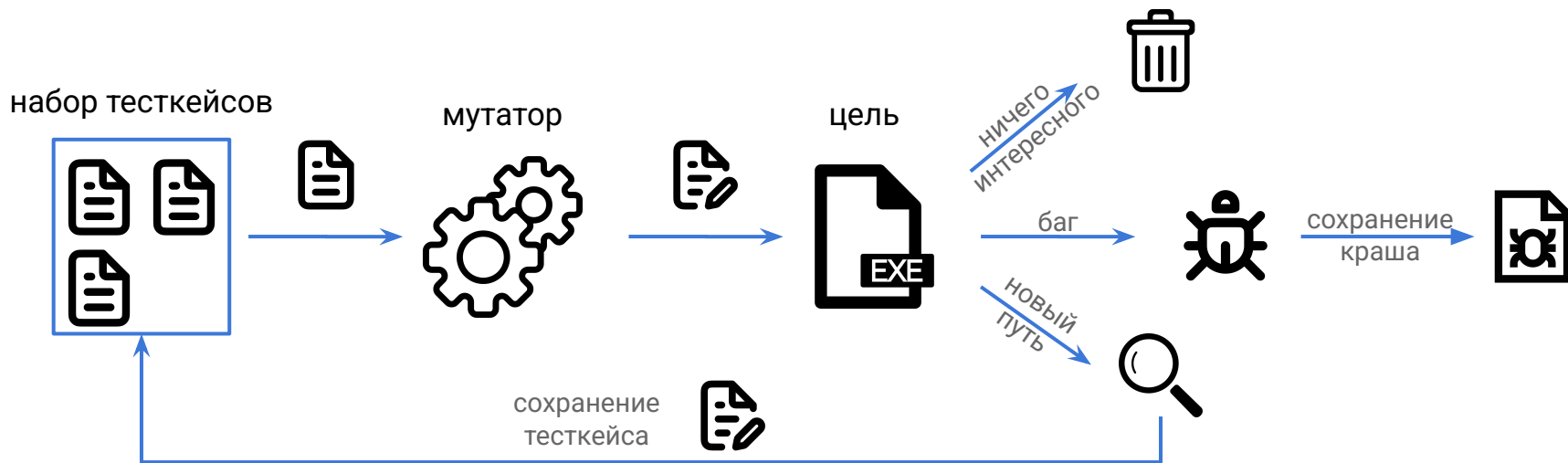


# workshop по фаззингу java



# Структурная схема серого фаззера




# Важность фаззинга managed языков

Фаззинг способен находить не только memory corruption в C/C++. Применяя подход фаззинга для управляемых языков можно обнаружить следующие проблемы:

- Зависания и таймауты
- Race Conditions
- Undefined Behavior
- Uncaught Exceptions
- Excessive Resource Consumption
- Логические ошибки
- Memory Leaks
- ...

BDU:2024-04627: Уязвимость компонента парсера LDAP URL программного обеспечения Apache Directory LDAP API, позволяющая злоумышленнику вызвать отказ в обслуживании


**Описание уязвимости** Уязвимость компонента парсера LDAP URL программного обеспечения Apache Directory LDAP API связана с отсутствием контроля вводимых пользователем данных. Эксплуатация уязвимости может позволить нарушителю, действующему удалённо, вызвать отказ в обслуживании

**Вендор**  Apache Software Foundation

**Наименование ПО**  Apache Directory LDAP API


# Цель для воркшопа









<https://github.com/apache/pdfbox>

 **pdfbox** Public

Watch 90 Fork 880 Star 2.7k

trunk 18 Branches 76 Tags  t Add file Code

 **THausherr** PDFBOX-5660: update owasp plugin ... ✓ 214629d · 2 days ago 12,199 Commits

 .github/workflows	PDFBOX-4892: fix problem with apache rat plugin (that hap...	3 years ago
 app	PDFBOX-5660: use log4j-transform-maven-shade-plugin-ex...	4 months ago
 benchmark	PDFBOX-5660: update log4j, introduce bom	3 months ago
 debugger-app	PDFBOX-5660: use log4j-transform-maven-shade-plugin-ex...	4 months ago
 debugger	PDFBOX-5952: remove short constructor, use the long one	last week
 examples	PDFBOX-5952: remove short constructor, use the long one	last week
 fontbox	PDFBOX-5944: enable blws test, remove println	2 weeks ago
 io	PDFBOX-5660: update log4j, introduce bom	3 months ago

## About

Mirror of Apache PDFBox

[pdfbox.apache.org/](https://pdfbox.apache.org/)

[java](#) [content](#) [library](#) [pdfbox](#)

- Readme
- Apache-2.0 license
- Code of conduct
- Security policy
- Activity
- Custom properties

2.7k stars

90 watching

880 forks

# Подготовка окружения

Собираем docker образ:

```
#bash
$ git clone https://github.com/lenix123/java_fuzz_workshop.git
$ cd java_fuzz_workshop
$ docker build --tag=pdfbox_workshop_img .
```

Запускаем контейнер:

```
#bash
$ docker run -it -v "$(pwd)/artifacts:/home/fuzz/artifacts:ro" \
--name=pdfbox_fuzz pdfbox_workshop_img
```

```
1 FROM ubuntu:22.04
2
3 RUN apt-get update && apt-get install -y wget git unzip
4
5 # install jdk
6 RUN wget '$'https://download.oracle.com/java/22/archive/
7 |jdk-22.0.2_linux-x64_bin.tar.gz'
8 RUN tar -xf jdk-22.0.2_linux-x64_bin.tar.gz
9 RUN mv jdk-22.0.2 /opt/
10 ENV JAVA_HOME='/opt/jdk-22.0.2'
11 ENV PATH="$JAVA_HOME/bin:$PATH"
12
13 # install mvn
14 RUN wget '$'https://dlcdn.apache.org/maven/maven-3/3.9.6/
15 |binaries/apache-maven-3.9.6-bin.tar.gz'
16 RUN tar -xf apache-maven-3.9.6-bin.tar.gz
17 RUN mv apache-maven-3.9.6 /opt/
18 ENV M2_HOME='/opt/apache-maven-3.9.6'
19 ENV PATH="$M2_HOME/bin:$PATH"
20
21 # clone pdfbox repository
22 RUN mkdir /home/fuzz
23 WORKDIR /home/fuzz
24 RUN git clone --depth 1 https://github.com/apache/pdfbox/
25
26 # install jazzer
27 RUN mkdir jazzer
28 WORKDIR /home/fuzz/jazzer
29 RUN wget '$'https://github.com/CodeIntelligenceTesting/jazzer/
30 |releases/download/v0.24.0/jazzer-linux.tar.gz'
31 RUN tar xf jazzer-linux.tar.gz && rm jazzer-linux.tar.gz
32 ENV PATH="$PATH:/home/fuzz/jazzer"
```

# Фаззинг обёртка как unit test

[Jazzer](#) имеет интеграцию с Junit 5 testing framework. Это позволяет писать фаззинг обёртку в привычном для разработчиков формате unit-тестов.

Патч pdfbox.patch содержит добавление зависимости jazzer-junit, а также простую фаззинг обёртку, добавленную в уже существующий файл с unit-тестами

Применяем патч:

```
#bash
$ cd /home/fuzz/pdfbox
$ git apply ../artifacts/pdfbox.patch
```

```
8 | | | <artifactId>junit-jupiter</artifactId>
9 | -   <version>${junit.version}</version>
10| +   <version>5.9.1</version>
11| | | <scope>test</scope>
12| | </dependency>
13| +   <dependency>
14| +       <groupId>com.code-intelligence</groupId>
15| +       <artifactId>jazzer-junit</artifactId>
16| +       <version>0.22.1</version>
17| +       <scope>test</scope>
18| +   </dependency>

37| class PDFStreamParserTest
38| {
39| +   @FuzzTest
40| +   void FuzzTestPDFParser(byte[] data) {
41| +       PDFStreamParser pdfStreamParser = new PDFStreamParser(data);
42| +       try {
43| +           pdfStreamParser.parse();
44| +       } catch (IOException e) {
45| +       }
46| +   }
47| }
```

# Фаззинг обёртка как unit test 2

Подготавливаем входной корпус:

```
#bash
$ mkdir pdfbox/src/test/resources/org/apache/pdfbox/pdfparser/PDFStreamParserTestInputs
$ cp /home/fuzz/artifacts/corpus/* pdfbox/src/test/resources/org/apache/pdfbox/pdfparser/PDFStreamParserTestInputs
```

Собираем тесты и запускаем фаззинг FuzzTestPDFParser:

```
#bash
$ mvn clean install -DskipTests
$ cd /home/fuzz/pdfbox/pdfbox
$ JAZZER_FUZZ=1 mvn test -Dtest=PDFStreamParserTest#FuzzTestPDFParser
```

Наработанный корпус хранится в `/home/fuzz/pdfbox/pdfbox/.cifuzz-corpus/`

# Написание отдельной обёртки

## примеры обёрток для jazzer

Для jazzer-а нужно реализовать класс с функцией `public static void fuzzerTestOneInput(input)`. В нашем случае обёртка просто вызывает целевую функцию `parse()` у класса `PDFStreamParser`.

Также в обёртке можно указать, какие исключения являются “ожидаемыми” (`IOException`). Необработанное обёрткой исключение будет интерпретироваться фаззером как критическое состояние.

```
1  import java.io.IOException;
2  import org.apache.pdfbox.pdfparser.PDFStreamParser;
3
4
5  public class PDFStreamParserFuzzer {
6      public static void fuzzerTestOneInput(byte[] input) {
7          PDFStreamParser pdfStreamParser = new PDFStreamParser(input);
8
9          try {
10             pdfStreamParser.parse();
11         } catch (IOException e) {
12             }
13     }
14 }
15
```



# Фаззинг java кода с помощью jazzer

<https://github.com/CodeIntelligenceTesting/jazzer>

Собираем обёртку для pdfbox:

```
#bash
$ cd /home/fuzz/pdfbox
$ mkdir fuzz && cd fuzz
$ cp -r /home/fuzz/artifacts/* .
$ javac -cp ../pdfbox/target/classes/ ./PDFStreamParserFuzzer.java
```

Запускаем фаззинг PDF парсера:

```
#bash
$ jazzer
--cp=../pdfbox/target/classes/../../io/target/classes:/home/fuzz/log4j/log4j-api-2.24.3.jar:/home/fuzz/log4j/commons-logging-1.2/commons-logging-1.2.jar --target_class=PDFStreamParserFuzzer -dict=pdf.dict
-close_fd_mask=3 -- corpus
```

# Сбор покрытия

<https://www.jacoco.org/jacoco/trunk/doc/cli.html>

<https://github.com/CodeIntelligenceTesting/jazzer/blob/main/docs/advanced.md#export-coverage-information>

Прогоняем наработанный корпус:

```
#bash
$ cd /home/fuzz/pdfbox/fuzz
$ jazzer
--cp=.../pdfbox/target/classes/..io/target/classes:/home/fuzz/log4j/log4j-api-2.24.3.jar:/home/fuzz/log4j/commo
ns-logging-1.2/commons-logging-1.2.jar --target_class=PDFStreamParserFuzzer -close_fd_mask=3
--coverage_dump=coverage.exec -runs=1 -- corpus
```

Генерируем html отчёт с помощью jacococli:

```
#bash
$ java -jar /home/fuzz/jacoco/lib/jacococli.jar report coverage.exec --classfiles ../pdfbox/target/pdfbox-3.0.4.jar
--html report --sourcefiles ../pdfbox/src/main/java/
```

# Сбор покрытия 2

Копируем папку с html отчётом на хост:

```
#bash на хосте
```

```
$ docker cp pdfbox_fuzz:/home/fuzz/pdfbox/fuzz/report .
```

Открываем отчёт с помощью браузера

[JaCoCo Coverage Report](#) > org.apache.pdfbox.pdfparser

## org.apache.pdfbox.pdfparser

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
<a href="#">COSParser</a>	<div><div></div></div>	0%	<div><div></div></div>	0%	156	156	407	407	37	37	1	1
<a href="#">BruteForceParser</a>	<div><div></div></div>	0%	<div><div></div></div>	0%	138	138	372	372	23	23	1	1
<a href="#">XrefParser</a>	<div><div></div></div>	0%	<div><div></div></div>	0%	81	81	272	272	15	15	1	1
<a href="#">PDFXRefStream</a>	<div><div></div></div>	0%	<div><div></div></div>	0%	34	34	98	98	10	10	1	1
<a href="#">XrefTrailerResolver</a>	<div><div></div></div>	0%	<div><div></div></div>	0%	35	35	86	86	16	16	1	1
<a href="#">PDFObjectStreamParser</a>	<div><div></div></div>	0%	<div><div></div></div>	0%	28	28	76	76	6	6	1	1
<a href="#">PDFXRefStreamParser</a>	<div><div></div></div>	0%	<div><div></div></div>	0%	23	23	57	57	6	6	1	1
<a href="#">PDFXRefStreamParser.ObjectNumbers</a>	<div><div></div></div>	0%	<div><div></div></div>	0%	13	13	36	36	3	3	1	1
<a href="#">EndstreamFilterStream</a>	<div><div></div></div>	0%	<div><div></div></div>	0%	23	23	40	40	3	3	1	1
<a href="#">PDFParser</a>	<div><div></div></div>	0%	<div><div></div></div>	0%	20	20	38	38	11	11	1	1
<a href="#">BaseParser</a>	<div><div></div></div>	94%	<div><div></div></div>	94%	12	232	20	448	1	43	0	1
<a href="#">PDFStreamParser</a>	<div><div></div></div>	91%	<div><div></div></div>	86%	16	106	12	164	1	11	0	1
<a href="#">FDParser</a>	<div><div></div></div>	0%	<div><div></div></div>	0%	7	7	17	17	3	3	1	1
<a href="#">XrefTrailerResolver.XrefTrailerObj</a>	<div><div></div></div>	0%	<div><div></div></div>	n/a	2	2	7	7	2	2	1	1
<a href="#">XrefTrailerResolver.XrefType</a>	<div><div></div></div>	0%	<div><div></div></div>	n/a	1	1	3	3	1	1	1	1
Total	6,748 of 9,210	26%	893 of 1,393	35%	589	899	1,541	2,121	138	190	13	15

# Structure aware fuzzing в java

<https://github.com/rohanpadhye/JQF>

Для реализации structure aware фаззинга java кода можно использовать JQF + Zest. Zest позволяет описать генератор семантически валидных данных. Таким образом, каждый рандомный тесткейс от JQF сначала трансформируются в валидную структуру и потом подаётся на вход целевой программе.

Задание на подумать: реализовать фаззинг pdfbox с помощью JQF.

```
public String generate(SourceOfRandomness random, GenerationStatus status) {
    String output = "";

    // choose one random command from array
    String command = random.choose(AllCommands);
    output += command + "\n";

    // content-length header is the only header which is checked in parsing
    if (random.nextBoolean()) {
        // set content-length header [-1000;1000]
        int length_value = random.nextInt(2000)-1000;
        output += "content-length:" + String.valueOf(length_value) + "\n";
    }

    // generate random set of headers
    int headers_len = random.nextInt(15);

    for (int i = 0; i < headers_len; i++) {
        String key = makeString(random, status);
        output += key + ":";
        String value = makeString(random, status);
        output += value + "\n";
    }

    // generate content 2 times more often than not
    if (random.nextInt(3) == 0) {
        return output;
    }
}
```