Institute of
Software Security

TUHH
Hamburg University of Technology

Reproduce vulnerabilities with Proof-of-Vulnerability (PoV) tests

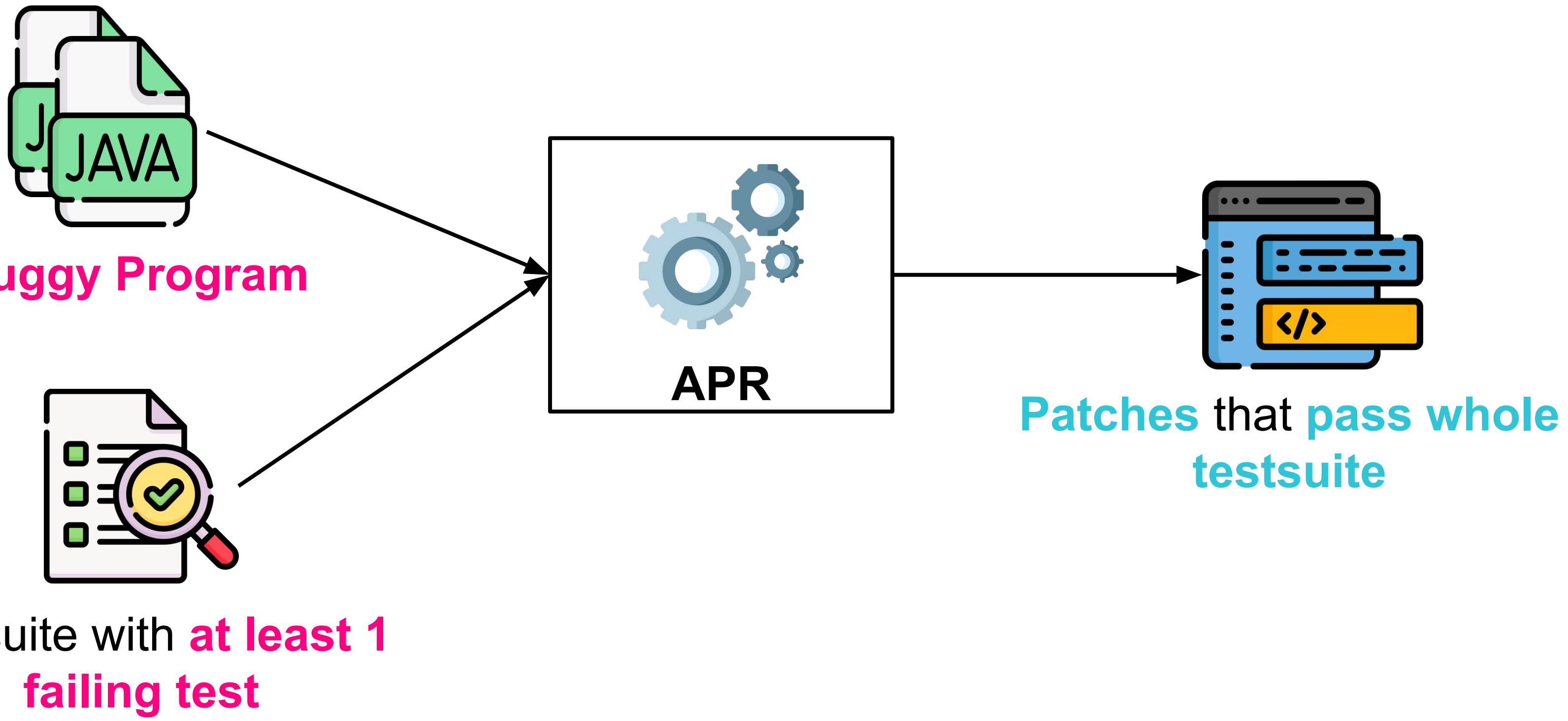
Requirements

- Your computer with Docker Desktop installed
  - See: https://docs.docker.com/get-docker/
- A familiar background of Java and the vulnerabilities in Java programs/systems
  - Understand Java code
  - Write Java JUnit test code

What are Proof-of-Vulnerability (PoV) tests?

- **PoV tests** are the **exploitation code** that are used to **detect** the **presences** of vulnerabilities in the system
  - i.e., failing if vulnerabilities exist, passing if vulnerabilities are eliminated
- PoV are useful for:
  - Detecting the safety of your systems against the known vulnerabilities
  - For Automated Program Repair (APR) tools!!!

Automated Program Repair - APR


JAVA
Buggy Program
APR
Patches that pass whole testsuite
Testsuite with at least 1 failing test

Automated Program Repair aims to repair software bugs automatically, help to reduce or even remove human intervention from bug fixing process

Availability of PoV tests

- Some developers patch the vulnerabilities and provide the PoVs as the bug test cases to verify if the patch works
- Some developers DO NOT do that:
  - Giving available PoVs (open-source project) can let attackers the tools to scan and exploit many systems
  - Even the vulnerability fixes are released silently

Your tasks in this Lab

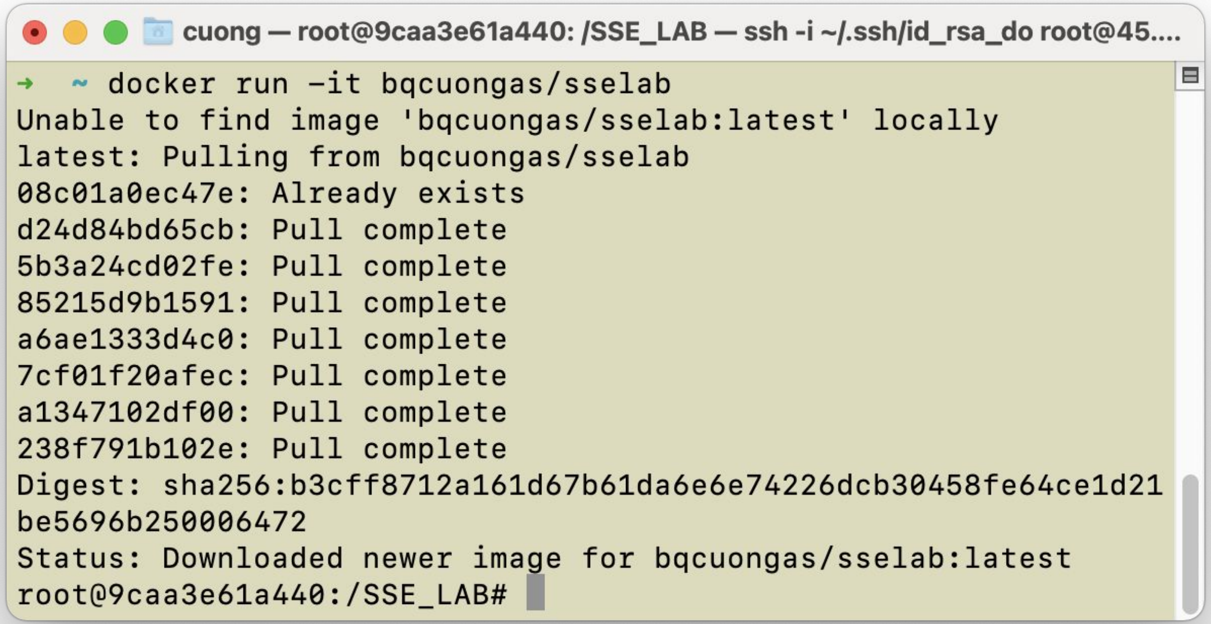
- Collect/Create the PoVs tests for **4 Java vulnerabilities**
  - Some of them are with the patches containing added JUnit tests, you should verify if the added tests are indeed the PoVs or not
  - Some of them are not, you need to read the patches, vulnerability reports and create the PoV yourself

Setup the Lab

Step 1: Start up your Docker Desktop

Step 2: Run the container for the lab

- *Open the terminal/cmd and run this command:*
  `docker run -it bqcuongas/sselab`



cuong — root@9caa3e61a440: /SSE_LAB — ssh -i ~/.ssh/id_rsa_do root@45....

```
➜  ~ docker run -it bqcuongas/sselab
Unable to find image 'bqcuongas/sselab:latest' locally
latest: Pulling from bqcuongas/sselab
08c01a0ec47e: Already exists
d24d84bd65cb: Pull complete
5b3a24cd02fe: Pull complete
85215d9b1591: Pull complete
a6ae1333d4c0: Pull complete
7cf01f20afec: Pull complete
a1347102df00: Pull complete
238f791b102e: Pull complete
Digest: sha256:b3cff8712a161d67b61da6e6e74226dcb30458fe64ce1d21
be5696b250006472
Status: Downloaded newer image for bqcuongas/sselab:latest
root@9caa3e61a440:/SSE_LAB# 
```

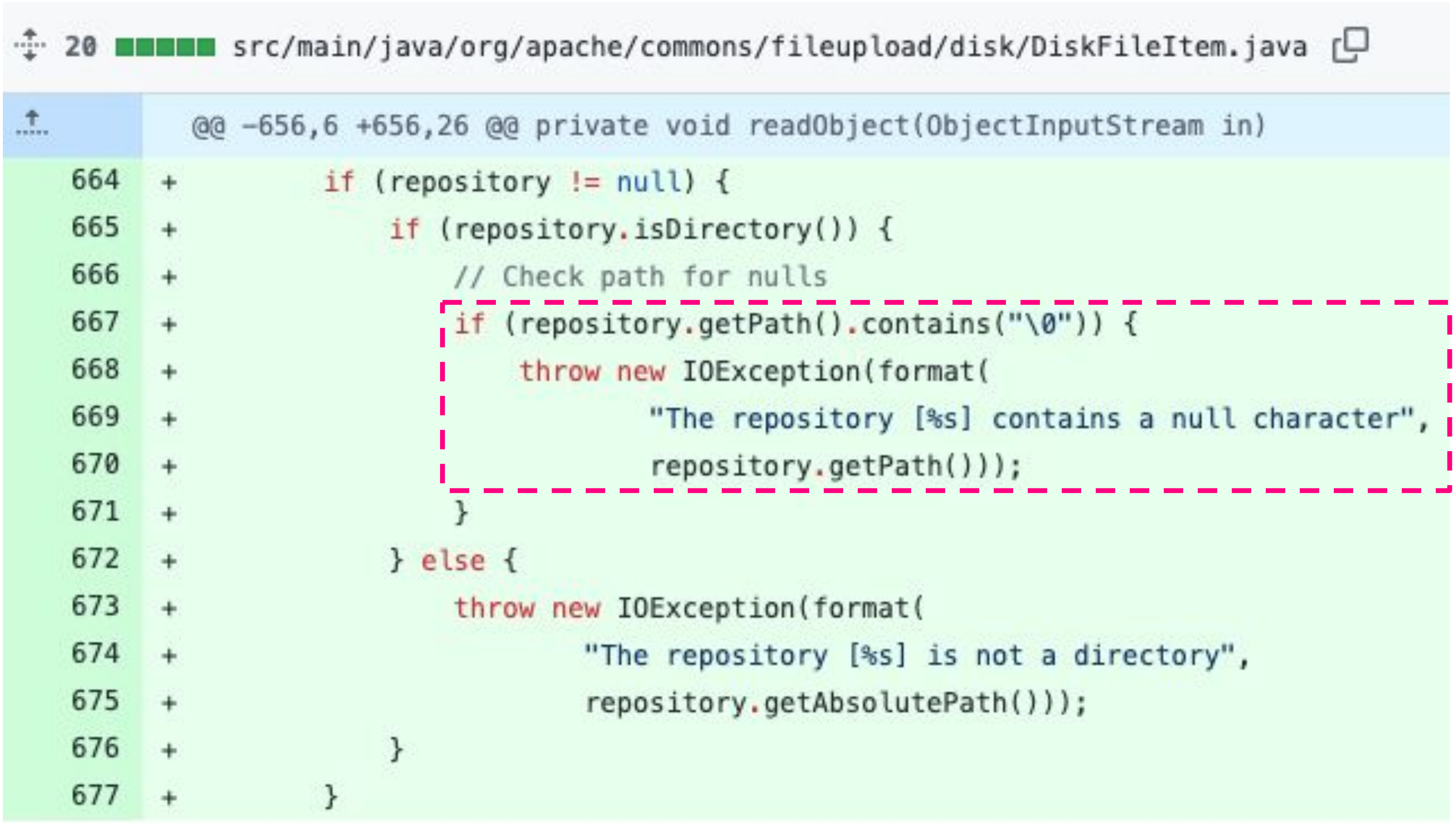
CVE-2013-2186

https://nvd.nist.gov/vuln/detail/CVE-2013-2186

NVD Report: The DiskFileItem class in Apache Commons FileUpload, as used in Red Hat JBoss BRMS 5.3.1; JBoss Portal 4.3 CP07, 5.2.2, and 6.0.0; and Red Hat JBoss Web Server 1.0.2 allows remote attackers to write to arbitrary files via a **NULL byte** in a **file name** in a serialized instance.

Developer’s patch

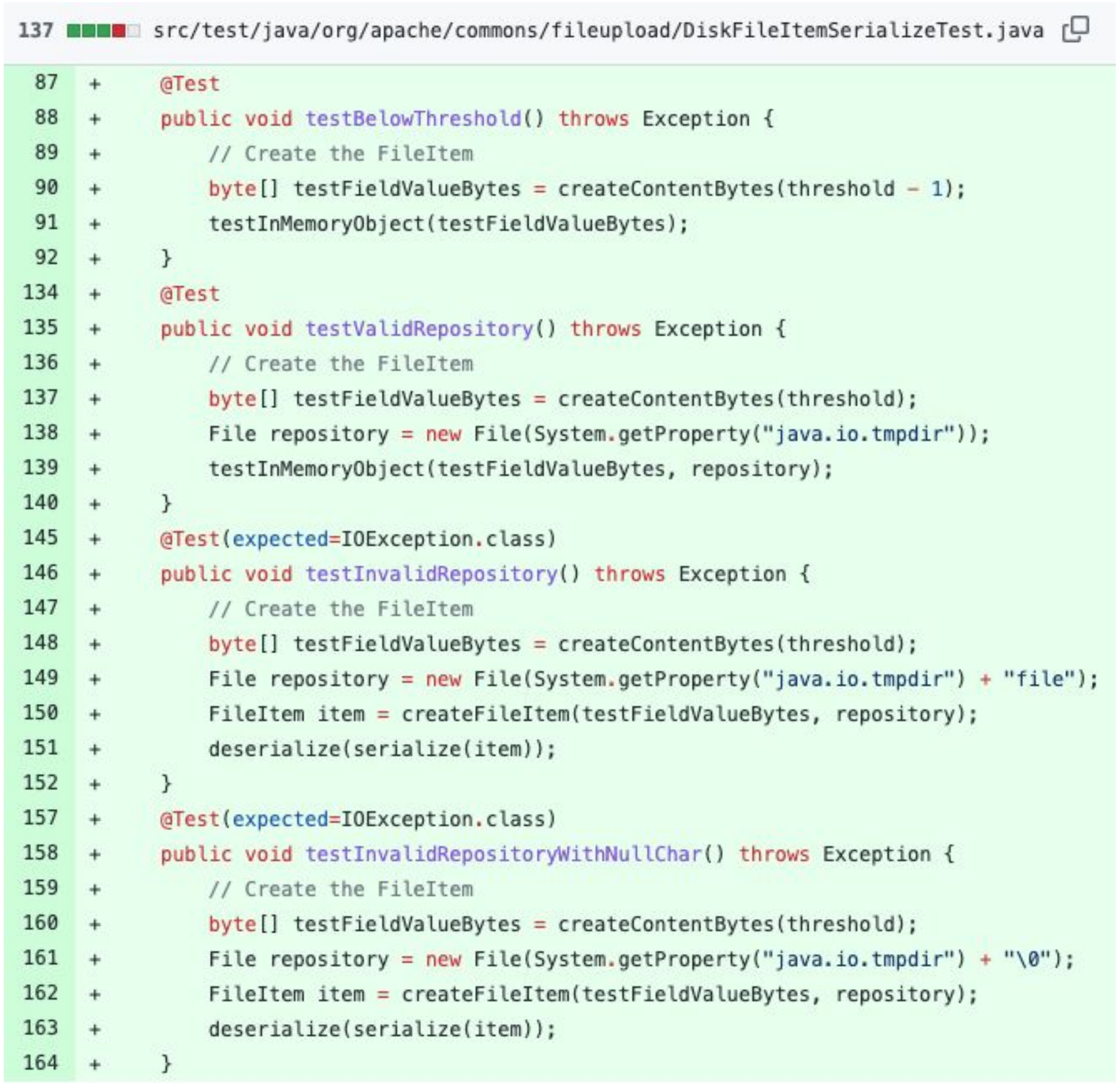
https://github.com/apache/commons-fileupload/commit/163a6061fbc077d4b6e4787d26857c2baba495d1



```
20 ■■■■■ src/main/java/org/apache/commons/fileupload/disk/DiskFileItem.java

@@ -656,6 +656,26 @@ private void readObject(ObjectInputStream in)
+            if (repository != null) {
+                if (repository.isDirectory()) {
+                    // Check path for nulls
+                    if (repository.getPath().contains("\0")) {
+                        throw new IOException(format(
+                                "The repository [%s] contains a null character",
+                                repository.getPath()));
+                    }
+                } else {
+                    throw new IOException(format(
+                            "The repository [%s] is not a directory",
+                            repository.getAbsolutePath()));
+                }
+            }
```

Added tests by developer



```
137 🟩🟩🟩🟥 src/test/java/org/apache/commons/fileupload/DiskFileItemSerializeTest.java
```

```
+    @Test
+    public void testBelowThreshold() throws Exception {
+        // Create the FileItem
+        byte[] testFieldValueBytes = createContentBytes(threshold - 1);
+        testInMemoryObject(testFieldValueBytes);
+    }
+    @Test
+    public void testValidRepository() throws Exception {
+        // Create the FileItem
+        byte[] testFieldValueBytes = createContentBytes(threshold);
+        File repository = new File(System.getProperty("java.io.tmpdir"));
+        testInMemoryObject(testFieldValueBytes, repository);
+    }
+    @Test(expected=IOException.class)
+    public void testInvalidRepository() throws Exception {
+        // Create the FileItem
+        byte[] testFieldValueBytes = createContentBytes(threshold);
+        File repository = new File(System.getProperty("java.io.tmpdir") + "file");
+        FileItem item = createFileItem(testFieldValueBytes, repository);
+        deserialize(serialize(item));
+    }
+    @Test(expected=IOException.class)
+    public void testInvalidRepositoryWithNullChar() throws Exception {
+        // Create the FileItem
+        byte[] testFieldValueBytes = createContentBytes(threshold);
+        File repository = new File(System.getProperty("java.io.tmpdir") + "\0");
+        FileItem item = createFileItem(testFieldValueBytes, repository);
+        deserialize(serialize(item));
+    }
```

Verify if the tests are PoVs

In vulnerable revision

```
$ cd CVE-2013-2186
$ mvn test
```

```
testInvalidRepositoryWithNullChar(org.apache.commons
.fileupload.DiskFileItemSerializeTest)  Time
elapsed: 0.098 sec  <<< FAILURE!
testInvalidRepository(org.apache.commons.fileupload.
DiskFileItemSerializeTest)  Time elapsed: 0.001 sec
<<< FAILURE!
Failed tests:
   Expected exception: java.io.IOException
   Expected exception: java.io.IOException

Tests run: 70, Failures: 2, Errors: 0, Skipped: 0
```

Verify if the tests are PoVs

In patched revision

```
$ git checkout -f
163a6061fbc077d4b6e4787d26857c2baba495d1
$ mvn test
```

```
Results :
Tests run: 70, Failures: 0, Errors: 0,
Skipped: 0
```

→ PoVs:

`testInvalidRepositoryWithNullChar``(org.apache.commons.fileupload.DiskFileItemSerializeTest)`

`testInvalidRepository``(org.apache.commons.fileupload.DiskFileItemSerializeTest)`

Questions ?