# Information Technologies Security

## Assignment

## Looking for vulnerabilities in large networks

Author: Adam Žilla

ID: 92177

# Contents

## Assignment clarification

The specific task in my assignment is to look for and choose some recent CVE. Next part is to implement a simple Python script with use of NSE scripting language or some more advanced scanner like Shodan or so, in edge case I can use Metasploit framework, for detecting vulnerable devices in specified IP range.

## CVE-2020-11969

I have been looking for vulnerability I can detect via network. I chose CVE-2020-11969. Short description of this vulnerability is the following. If Apache TomEE is configured to use the embedded ActiveMQ broker, and the broker URI includes the useJMX=true parameter, a JMX port is opened on TCP port 1099, which does not include authentication. This affects Apache TomEE 8.0.0-M1 - 8.0.1, Apache TomEE 7.1.0 - 7.1.2, Apache TomEE 7.0.0-M1 - 7.0.7, Apache TomEE 1.0.0 - 1.7.5. (1)

## CWE-287

As stated in mitre Common Weakness Enumeration database, above mentioned CVE is type of Improper Authentication. IT means that an actor claims to have a given identity, the software does not prove or insufficiently proves that the claim is correct. (2)

## Analysis of problem area

I divide this analysis into two parts. First is to setup the apache server with ActiveMQ broker as a working example with useJmx set to true and JMX running on port 1099. The second part is implementation of detection as NSE script.

## Analysis of existing solutions

There are plenty of solutions to different CVE's on github, stackoverflow or elsewhere on the internet. For my CVE there is currently no solution as it is recent vulnerability or I was not able to find one.

### Example Usage of broker

As example usage of such tool I chose the tutorial from website tomitribe (3). Imagine running Apache on some raspberry pi with spider tool listening. Spider or so called Web crawler is an Internet bot that crawls/browse the world wide web or the specific website for images or links like in this example.

My spider firstly browse the website for images in website. Sample website is https://www.fiit.stuba.sk.

In apache terminal we can see orange links, these refers to images found on fiit website. After spider is finished with images spider then begin to browse for all hyperlinks.

This is just example for how the apache can be used.


## Environment

I am working in Vmware workstation, virtual machine Kali linux, where my apache server with ActiveMQ broker and MDB example are running. My public git repo can be found on https://github.com/itchybunion6235/BIT_project

## Apache

Following chapters contains some introduction to problematic. What are individual modules in our apache and so on.

### ActiveMQ broker

Simple answer to what is ActiveMQ is. ActiveMQ is a popular open source Java message-oriented middleware (MoM). For me that was not enough.

Another notion is JMS which stands for Java Messaging Service. So, JMS is like protocol but its specified by java, it allows applications to communicate in an easy way, send messages back and forth. ActiveMQ brokers are the implementation of JMS. It is very useful to have some standards in messaging services because as written in openlogic website (4):

Before messaging systems like ActiveMQ, it was difficult — if not impossible — to “federate” applications that:

- Were written in disparate languages.
- Resided on heterogeneous platforms.
- Used different data structures and protocols.

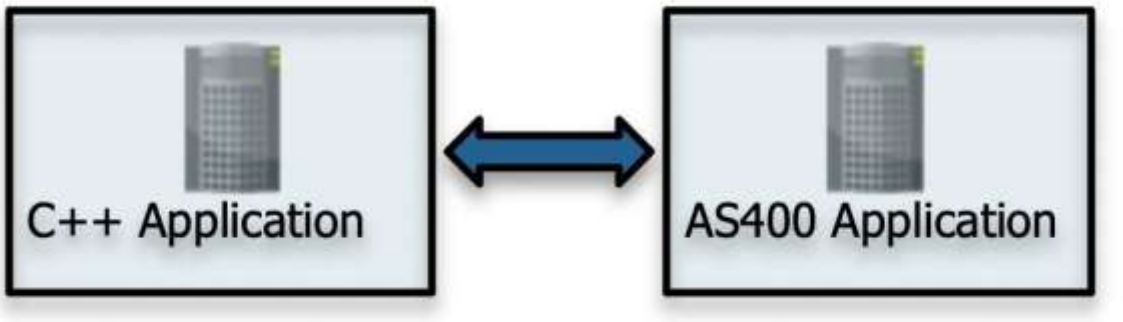


*Figure 1 Example of different applications before ActiveMQ (1)*

## TomEE

Brief introduction for those who are interested in web servers especially apache. In apache website the description of tome is following:

Apache TomEE, pronounced "Tommy", is an all-Apache Java EE 6 Web Profile certified stack where Apache Tomcat is top dog. Apache TomEE is assembled from a vanilla Apache Tomcat zip file. We start with Apache Tomcat, add our jars and zip up the rest. The result is Tomcat with added EE features - TomEE. (5) From my experience after I have read some articles and documentation, I can say the TomEE is very useful server and very simple to use. It consists of several open-source components that are shown below:

| Component | Description |
|---|---|
| Apache Tomcat | HTTP server and Servlet container supporting Java Servlet and JavaServer Pages (JSP). |
| Apache OpenEJB | Open-source Enterprise JavaBeans (EJB) container system. |
| Apache OpenWebBeans | Open-source Java Contexts and Dependency Injection (CDI) implementation. |
| Apache OpenJPA | Open-source Java Persistence API (JPA) 2.1 implementation. |

| | |
|---|---|
| Apache Geronimo Transaction | Open-source Java Transaction API (JTA) 1.2 implementation. |
| Apache MyFaces | Open-source Java Server Faces (JSF) implementation. |
| Apache ActiveMQ | Open-source Java Message Service (JMS) implementation. |
| Apache CXF | Web Services frameworks with a variety of protocols - such as SOAP, XML/HTTP, RESTful HTTP. |
| Apache Derby | Full-fledged relational database management system (RDBMS) with native Java Database Connectivity (JDBC) support. |

*Table 1 TomEE's components (2)*

In my setup I am using apache TomEE plume version 7.1.1 which is vulnerable version as stated in CVE-2020-11969 chapter.

I also want to mention that in version 7.1.1 which I am using, there are many releases with different components for example the table below is for version TomEE plus. There are versions like plume, webprofile, microprofile, OpenEJB Standalone and few more. You can see comparision of different releases in this link.

Deploying webapps is very simple as moving .war extension files to webapps folder. Apache then recognizes new application and apache then configure and run the application in server.

### JMX

Java management extension in apache allows you to monitor and control the behavior of ActiveMQ broker. You can control or monitor things like heap memory, threads, CPU usage and configure various settings such as MBeans and so on. Example JMX tool is JConsole. It lets you connect to a remote apache server and monitor it with a simple GUI and use some features to manage apache TomEE. (3)

## CVE setup

Following chapter is step by step guide for setup the Apache with all dependencies to support the CVE requirements. To follow my environment and my steps there are prerequisites to have already Kali Linux as VM installed and connected to internet. Then follow these steps:

1. Download Apache TomEE plume v 7.1.1 from this website as TAR.GZ.
2. Extract it with some archive tool or with command – **tar -xvf apache-tomee-7.1.1-plume.tar.gz**
3. There is a new folder in your working directory called **apache-tomee-7.1.1-plume**
4. Open directory from previous step and navigate to conf directory

5. There is **tomee.xml** file. Open it in your favorite editor and make sure it looks like in the picture below.

```
<?xml version="1.0" encoding="UTF-8"?>
<tomee>
  <Resource id="MyJmsResourceAdapter" type="ActiveMQResourceAdapter">
        BrokerXmlConfig =  broker:(tcp://localhost:1099)?useJmx=true
        ServerUrl       =  tcp://localhost:1099
  </Resource>
</tomee>
```

*Figure 2 Tomee.xml file content after configuration*

```
CATALINA_OPTS=" -Dcom.sun.management.jmxremote
                -Dcom.sun.management.jmxremote.port=1099
                -Dcom.sun.management.jmxremote.ssl=false
                -Dcom.sun.management.jmxremote.authenticate=false"
export JAVA_OPTS="$JAVA_OPTS -Djava.net.preferIPv4Stack=true"
```

*Figure 3 Content of setenv.sh file (5)*

6. Next you have to go to bin directory and create the **setenv.sh** file and make it look like in image below.
7. Make the file executable by running command **chmod +x setenv.sh**
8. Last step is to run TomEE by executing command **./catalina.sh run**
9. Now your Apache TomEE is running with embedded ActiveMQ broker and JMX configured on port 1099.

# Nmap

We all know Nmap so just a brief introduction for those who do not know Nmap. It is a free, open source tool for security auditing, network scanning and more. The important part in Nmap for this assignment is the NSE, which we will describe in the NSE chapter below.

## NSE

NSE stands for Nmap Scripting Engine, it allows network administrators or users to write their own scripts for Nmap, it uses Lua programming language. Nmap has its prewritten scripts, these are divided into following categories: auth, broadcast, default, discovery, dos, exploit, external, fuzzer, intrusive, malware, safe, version, and vuln. One script can be under many categories, for example my script will be under vuln and safe. Vuln scripts checks for specific known vulnerabilities and safe means it is not intrusive. (4)

## Python

I am using Python for evaluating the result of two NSE scripts. As I was not able to detect both information about CVE in one script, I wrote two scripts. One for detecting whether Apache TomEE server is running on host and the second one for detecting if JMX is configured on port 1099.

## Detection design

Chapters below describe two NSE scripts. The output from these scripts is then evaluated in Python script and the output whether certain host is vulnerable or not is sent to the stdout.

### Apache TomEE detection

Script issues a HTTP get call to host/server and captures the response which contains information about the server itself. The response is then parsed and few regex expressions are looking for information whether server is running Apache TomEE or not. If server is running TomEE another regex expression searches for version of TomEE. If the version found, is vulnerable, the script outputs the string "Vulnerable" else it outputs "Not Vulnerable".

### JMX detection

For JMX detection I am using Java Remote Method Invocation or Java RMI. It is the Java API that performs remote method invocation. The script queries the RMI registry of the server, it is the place where server stores information about services it is running. The response is as Java Object. The script is only looking for port 1099 so everything it finds it is on port 1099. If the response contains "jmxrmi" what means it is running JMX the message "Vulnerable" is print to the stdout.

## Code snippets

Following code snippets are core, the hearth, of the CVE detection.

### Python

```
args = arguments()
http = subprocess.check_output(["nmap","--script","http_check.nse",args.ip_address])
veci = re.findall('Nmap scan[\s\S]*?MAC',http.decode())
hosti = isVuln(veci)
rmi = subprocess.check_output(["nmap","--script","rmi_check.nse",args.ip_address])
veci = re.findall('Nmap scan[\s\S]*?MAC',rmi.decode())
hosti2 = isVuln(veci)
for k, v in hosti.items():
    if(hosti[k] == hosti2[k]):
        print(k,v)
    else:
        print(k,"Not Vulnerable")
```

*Figure 4 Nmap subprocesses are invoked and the output from them is evaluated.*

## Apache TomEE detection

```
tome = string.match (title, " %(%a+%)")
if (tome ~= nil) then
  print(tome)
  tome = string.match(tome, "%a+")
end
title = string.match(title, " %([%d.]*%)$")
if (tome ~= nil) then
  title = string.match(title, "[%d%.]+")
end
```

*Figure 5 Few regex expressions from the Apache TomEE detection script.*

## JMX detection

```
local registry = rmi.Registry:new( host, port )
local status, j_array = registry:list()
local data = j_array:getValues()
for i, name in pairs(data) do
  if name == "jmxrmi" then
    data = "Vulnerable"
    return data
  else
    data = "Not Vulnerable"
  end
```

*Figure 6 The core of JMX detection with RMI registry query.*

# Bibliography

1. NIST. [Online] https://nvd.nist.gov/vuln/detail/CVE-2020-11969#vulnCurrentDescriptionTitle.

2. Mitre. cve. [Online] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-11969.

3. Monson-Haefel, Richard. 5 Minutes or Less: Message-Driven Beans with ActiveMQ and TomEE. tomitribe. [Online] 5 23, 2019. https://www.tomitribe.com/blog/5-minutes-or-less-message-driven-beans-with-activemq-and-tomee/.

4. Reock, Justin. what-apache-activemq. openlogic. [Online] 6 4, 2020. [Cited: 11 3, 2020.] https://www.openlogic.com/blog/what-apache-activemq.

5. apache-tomee. apache. [Online] [Cited: 11 3, 2020.] http://openejb.apache.org/apache-tomee.html.

6. [Online] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-11969.

7. what apache activemq. openlogic.com. [Online] 6 4, 2020. [Cited: 11 8, 2020.] https://www.openlogic.com/blog/what-apache-activemq.

8. Apache TomEE. apache. [Online] apache.org. [Cited: 11 13, 2020.] http://tomee.apache.org/comparison.html.

9. remote jmx connection example using jconsole. cleantutorials.com. [Online] [Cited: 11 11, 2020.] https://www.cleantutorials.com/jconsole/remote-jmx-connection-example-using-jconsole.

10. man-nse. nmap.org. [Online] nmap. [Cited: 11 16, 2020.] https://nmap.org/book/man-nse.html.

11. enable jmx tomcat to monitor administer. geekflare.com. [Online] 1 20, 2020. [Cited: 11 15, 2020.] https://geekflare.com/enable-jmx-tomcat-to-monitor-administer/.