# USING TACACS+ AUTHENTICATION WITH CONFD

tail-f a Cisco company

# Table of Contents

# USING TACACS+ AUTHENTICATION WITH CONFD

## 1. Abstract

ConfD's built-in authentication and authorization mechanisms uses YANG data modeled AAA-related information which is stored in CDB. This is the most common AAA scenario. Additionally, ConfD can be integrated with external authentication and authorization mechanisms. For this, PAM can be used via pre-integrated confd.conf settings or another external solution such as RADIUS or TACACS+ server via an integration API.

This application note looks at ConfD external authentication integration with a TACACS+ server. For this integration, we use the external authentication method as described in the ConfD User Guide, section 15.4.4, "External Authentication".

## 2. Background

### 2.1 Configure TACACS+ Server

In order to use external authentication with ConfD, the following needs to be configured in confd.conf:

```
<confdConfig>
  <aaa>
    <authOrder>externalAuthentication localAuthentication</authOrder>
    <externalAuthentication>
      <enabled>true</enabled>
      <executable>./auth.py</executable>
     </externalAuthentication>
  </aaa>
</confdConfig>
```

**`/confdConfig/aaa/authOrder`** is used to define the order of authentication methods which ConfD will try when authenticating a user. The default order is "localAuthentication pam externalAuthentication".

`/confdConfig/aaa/externalAuthentication/executable` is used to define the name (full path) of the executable or script to be invoked to authenticate a user. The executable will receive the username and the clear text password on its standard input in the format:

```
[${USER};${PASS};]\n
```

For example, if the user is bob and the password is secret, the executable will receive the string `[bob;secret;]` followed by a newline on its standard input. The program must parse this line which it receives from ConfD.

The format of the output sent to standard output from an external authentication program when authentication is successful is:

```
accept $groups $uid $gid $supplementary_gids $HOME\n
```

Where:

- `$groups` is a space separated list of the group names the user is a member of.
- `$uid` is the UNIX integer user id ConfD should use as default when executing OS commands for this user.
- `$gid` is the UNIX integer group id ConfD should use as default when executing OS commands for this user.
- `$supplementary_gids` is a (possibly empty) space separated list of additional UNIX group ids which the user is also a member of.
- `$HOME` is the directory which should be used as $HOME for this user when ConfD executes OS commands on behalf of this user.

If authentication fails, the program should output reject or abort, possibly followed by a reason for the rejection.

See ConfD User Guide, section 15.4.4, "External Authentication" for more information.

## 3. TACACS+ Server

### 3.1 Install TACACS+ Server

TACACS+ server can usually be installed as a pre-built binmary package by the package manager in common Linux distributions. For this application note, we used Ubuntu 18.04 and installed the tacacs+ package version 4.0.4.27a-3:

```
apt-get update
apt-get -y install tacacs+
```

### 3.2 Configure TACACS+ Server

Configuration of the TACACS+ server is found in **`/etc/tacacs+/tac_plus.conf.`**

In this file, we can find a server key:

```
key = testing123
```

The key can be changed. For our testing purposes, we will keep it as testing123. NOTE: In case you change the key, don't forget to change it in the examples and scripts used in this note.

To be able to use external authentication, we need information the external executable returns, i.e. a space separated list of ConfD groups, system uid, gid and home directory. We can get such configuration from TACACS+ in form of so called AV pairs (attribute value pairs).

For this, we introduce a TACACS+ service called tailf with mandatory attribute groups (comma separated list of groups) and optional attributes uid, gid, and home. If an optional argument is not configured, a default value will be used. Default values are 9000 for uid, 100 for gid and /var/confd/homes/+ username for home. This corresponds to ConfD AAA initial default configuration.

Example of the TACACS+ tailf service configuration:

```
service = tailf {
 groups = "public"
 uid = "9000"
 gid = "100"
 home = "/tmp/public"
}
```

The service can be added directly under user configuration or under TACACS+ group configuration. If the service is under user configuration, it has higher priority than group configuration.

**NOTE:** It is important to note the difference between TACACS+ group and groups in the tailf service. TACACS+ has concept of groups in the configuration, but these groups are used only internally by the TACACS+ server.

Example of TACACS+ server configuration file **`/etc/tacacs+/tac_plus.conf`** used in this note:

```
key = testing123

...

user = confdtester {
   login =  cleartext "confd"
   service = tailf {
     groups = "user public"
     uid = "1000"
     gid = "100"
     home = "/tmp"
   }
}

user = confdadmin {
   login =  cleartext "adminconfd"
   member = admin
}

group = admin {
   service = tailf {
     groups = "admin"
   }
}

group = public {
   service = tailf {
     groups = "public"
     uid = "9000"
     gid = "100"
     home = "/tmp/public"
   }
}
```

**NOTE:** If you modify the TACACS+ server configuration, you need to restart it with the command "service tacacs_plus restart".

## 4. TACACS+ Client

Once we have a TACACS+ server configured, we need to access it and perform authentication against it. For this, we need to use a TACACS+ client.

In this application note, we will use the Python based TACACS+ client which is published at https://github.com/ansible/tacacs_plus. This client comes with a command line application which we can use to test our configuration. It also provides a Python API, which we will use to implement our executable, a Python script, for external authentication.

### 4.1 Install TACACS+ Client

Install the TACACS+ client:

```
pip install tacacs_plus
```

Use the command **"tacacs_plus -h"** to verify that the client is installed and working.

### 4.2 Performing Simple Tests with TACACS+ Client

We can test our TACACS+ serverconfiguration with following command line commands:

```
tacacs_client -v -H localhost -k testing123 -u confdtester \
  authenticate --password confd
status: PASS

tacacs_client -v -H localhost -k testing123 -u confdtester \
  authenticate --password confdwrong
status: FAIL

tacacs_client -v -H localhost -k testing123 -u confdtester \
  authorize -c service=tailf
status: PASS
av-pairs:
  groups=user public
  uid=1000
  gid=100
  home=/tmp

tacacs_client -v -H localhost -k testing123 -u confdadmin \
  authenticate --password adminconfd
status: PASS

tacacs_client -v -H localhost -k testing123 -u confdadmin \
  authorize -c service=tailf
status: PASS
av-pairs:
  groups=admin
```

### 4.3 Using the TACACS+ Client in a Python Script

The TACACS+ server configuration can be tested with the use of Python commands:

```
from tacacs_plus.client import TACACSClient
import socket

cli = TACACSClient('localhost', 49, 'testing123', timeout=10, family=socket.
AF_INET)
authen = cli.authenticate('confdtester', 'confd')
print(authen.valid)

auth = cli.authorize("confdtester",arguments=[])
print(auth.arguments)
```

## 5. External Authentication Script Using Python

Now we have all the knowledge needed in order to write the external authentication application.

The application receives username and password on the standard input and authenticates user against the TACACS+ server. If authentication passes, it will retrieve the TACACS+ tailf service for the given user and get groups, uid, gid, and home attributes. The groups attribute must be present, but the other attributes can be missing. Any missing attributes will be replaced with default values. The result will be output to standard output according to the ConfD rules for an external authentication executable.

External authentication application (Python script):

```
#!/usr/bin/env python
from tacacs_plus.client import TACACSClient
import socket
import sys

line=sys.stdin.readline()
line = line.replace("[", "")
line = line.replace("]", "")
token = line.split(";") # token[0] username, token[1] password

def get_av_pair(arguments, key, default = None):
    ret = default
    for av in arguments:
        avf = av.split("=")
        if avf[0] == key:
            ret = avf[1]
            break
    return ret
```

```
cli = TACACSClient('localhost', 49, 'testing123', timeout=10,
                   family=socket.AF_INET)
authen = cli.authenticate(token[0], token[1])
if authen.valid == True:
    auth = cli.authorize(token[0], arguments=["service=tailf"])
    groups = get_av_pair(auth.arguments, key="groups")
    if groups != None:
        uid = get_av_pair(auth.arguments, key="uid", default=9000)
        gid = get_av_pair(auth.arguments, key="gid", default=100)
        home = "/var/confd/homes/{}".format(token[0])
        print("accept {} {} {} {}".format(groups, uid, gid, home))
    else:
        print(
            "reject Cannot retrieve groups AV pair (tailf service) for user
{}"
                .format(token[0]))
else:
    print("reject")
```

The following commands can be used to test the Python script with our TACACS+ server configuration:

```
echo "[confdtester;confd;]" | ./auth.py
accept user public 1000 100 /var/confd/homes/confdtester

echo "[confdadmin;adminconfd;]" | ./auth.py
accept admin 9000 100 /var/confd/homes/confdadmin

echo "[confdtester;confdwrong;]" | ./auth.py
reject
```

## 6. Accessing ConfD with External Authentication using TACACS+

After we configure external authentication in confd.conf and specify the path to the Python script as the executable, we can test TACACS+ server access for ConfD external authentication. In the following subsections, there are examples for NETCONF and CLI (through the internal ConfD SSH server). Other ConfD northbound interfaces should work as well.

### 6.1 NETCONF

Use the netconf-console utility to test NETCONF authentication:

```
netconf-console -u confdtester -p confd --hello
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
  ...
  </capabilities>
  <session-id>12</session-id>
</hello>

netconf-console -u confdtester -p confdwrong --hello
Authentication failed.

netconf-console -u confdadmin -p adminconfd --hello
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
...
  </capabilities>
  <session-id>14</session-id>
</hello>
```

### 6.2 CLI via SSH

Use the corresponding password with following commands to test SSH authentication via the internal ConfD SSH server:

```
ssh confdtester@localhost -p 2024
Welcome to the ConfD CLI
confdtester connected from 127.0.0.1 using ssh on confdtac
confdtester@confdtac 09:56:18>

ssh confdadmin@localhost -p 2024
confdadmin@localhost's password:
Welcome to the ConfD CLI
confdadmin connected from 127.0.0.1 using ssh on confdtac
confdadmin@confdtac 10:01:45>
```

### 6.3 audit.log

External authentication can be traced in ConfD's audit.log if it is enabled in confd.conf.

For example:

```
cat /opt/tailf/confd/var/confd/log/audit.log
<INFO> 17-May-2019::10:58:00.110 confdtac confd[18]: audit \
       user: confdtester/0 external authentication succeeded via netconf \
       from 127.0.0.1:40214 with ssh, member of groups: user,public
<INFO> 17-May-2019::10:58:00.136 confdtac confd[18]: audit \
       user: confdtester/0 logged in via netconf from 127.0.0.1:40214 with
ssh \
       using external authentication
<INFO> 17-May-2019::10:58:00.143 confdtac confd[18]: audit \
       user: confdtester/12 assigned to groups: user,public
<INFO> 17-May-2019::10:58:00.144 confdtac confd[18]: audit \
       user: confdtester/12 created new session via netconf \
       from 127.0.0.1:40214 with ssh
<INFO> 17-May-2019::10:58:00.197 confdtac confd[18]: audit \
       user: confdtester/12 terminated session (reason: normal)
<INFO> 17-May-2019::10:58:00.198 confdtac confd[18]: audit \
       user: confdtester/0 logged out <extauth> user
```

## 7. Conclusion

In this application note, we have provided a description of how to integrate  external authentication using a TACACS+ server with ConfD.

External authorization integration of TACACS+ with ConfD through the Data Provider API is also possible using the TACACS+ authorize commands.  If you are interested in learning more about external authorization, consult the "Authorization Callbacks" section of the confd_lib_dp(3) man page.

## 8. Appendix - Docker Environment for Testing

The gradle script found in this section can be used to build a docker image with ConfD, TACACS+ server, and tacacs_plus client. The script should be run in a 64-bit Linux environment with docker. Gradle (version 5.0 or higher) can be installed with use of **`SDKMAN https://sdkman.io`**. The gradle script and the auth.py application need to be located in the same directory. The public SSH key has to be **`~/.ssh/id_rsa.pub`** to be present so that the Docker image can be accessed with SSH.

Docker image can be built with this command:

```
gradle -b confdtacacs.gradle clean buildConfdTacacs \
  -PCONFD_VERSION_TF=7.1 -PCONFD_EXAMPLES_TF=false \
  -PPACKAGE_DIR_TF=<dir with confd installation files>
```

**NOTE:** It is possible to create a Dockerfile only. For this, replace buildConfdTacacs with createConfdTacacs. The created Dockerfile and all files which are to be copied into image are located in the build/confdtacimg directory.

To start and run the container, use:

```
docker run -d -p 3322:22 --name confdtac \
  --hostname confdtac confdtacimg
```

The running container can be accessed with the command:

```
ssh root@localhost -p 3322
```

Inside the container, one can test TACACS+ server access with the tacacs_plus client, external authentication script, and ConfD access as described in the previous sections.

Everything should be pre-installed, configured, and running.

To stop, delete the container, and delete the image use:

```
docker stop confdtac
docker rm confdtac
docker rmi confdtacimg
```

The gradle script for building the Docker image:

```
/**
 *  Build ConfD (Basic) + TACACS+ docker Linux environment
 *  accessible with Ssh with use of TailfDockerPlugin.
 *  Possible usage :
 *
 *  Build:
    gradle -b confdtacacs.gradle clean buildConfdTacacs \
    -PCONFD_VERSION_TF=7.1 -PCONFD_EXAMPLES_TF=false \
    -PPACKAGE_DIR_TF=<dir with confd installation files>
 *
 *  Run:
    docker run -d -p 3322:22 --name confdtac \
    --hostname confdtac confdtacimg

 *  Ssh into running container:
    ssh root@localhost -p 3322
 *  OR
    ssh -o UserKnownHostsFile=/dev/null \
   -o StrictHostKeyChecking=no root@localhost -p 3322
 *
 * Check TACACS+ server is running:
   pgrep -lf tac_plus
 *
 *  Stop and clean:
    docker stop confdtac; docker rm confdtac; \
    docker rmi confdtacimg
 *
 * Note:
 * It is important to have 'clean' target in the gradle command
 * ~/.ssh/id_rsa.pub has to be present as is copied to the
 *  docker image
 * -PCONFD_VERSION_TF=7.1 can be skipped, then default confd
 *   version from TailfDockerfileTask is used
 * -PPACKAGE_DIR_TF can be skipped, if ConfD installation
 *    files are in ${HOME}/tailf
 * -PCONFD_EXAMPLES_TF=true (or remove this from command line)
 *   -install also examples and development environment
 *    (image is bigger)
 * -PIMAGE_TAG=... can be used to overwrite default
 *   image name "confdtacimg"
 * Dockerfile is located in file
 *    'build/confdimg/confdtac.dockerfile'
 */

buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.github.novakmi:tailfdocker:0.6.1'
    }
}

apply plugin: 'base'
apply plugin: 'com.bmuschko.docker-remote-api'
apply plugin: com.github.novakmi.tailfdocker.TailfDockerPlugin

import com.bmuschko.gradle.docker.tasks.image.DockerBuildImage
import com.github.novakmi.tailfdocker.TailfDockerfileTask

def tacsconf = "/etc/tacacs+/tac_plus.conf"

def tailfService = { myDelegate, args ->
    delegate = myDelegate
    if (args.tailfgroups || args.tailfuid || args.tailfgid || args.tailfhome)
{
        runCommand "echo '    service = tailf {'  >> $tacsconf"
        if (args.tailfgroups) {
            runCommand "echo '       groups = \"$args.tailfgroups\"'  >> $tac-
sconf"
        }
        if (args.tailfuid) {
            runCommand "echo '       uid = \"$args.tailfuid\"'  >> $tacsconf"
        }
        if (args.tailfgid) {
            runCommand "echo '       gid = \"$args.tailfgid\"'  >> $tacsconf"
        }
        if (args.tailfhome) {
            runCommand "echo '       home = \"$args.tailfhome\"'  >> $tacsconf"
        }
        runCommand "echo '    }'  >> $tacsconf"
    }
}

def addGroup = { myDelegate, groupname, args ->
    delegate = myDelegate
    runCommand "echo '' >> $tacsconf"
    runCommand "echo 'group = $groupname {' >> $tacsconf"
    tailfService(delegate, args)
    runCommand "echo '}'  >> $tacsconf"
}

def addUser = { myDelegate, username, password, args ->
    delegate = myDelegate
    runCommand "echo '' >> $tacsconf"
    runCommand "echo 'user = $username {' >> $tacsconf"
    if (args.name) {
        runCommand "echo '    name = \"$args.name\"'  >> $tacsconf"
    }
    runCommand "echo '   login =  cleartext \"$password\"'  >> $tacsconf"
    if (args.group) {
        runCommand "echo '   member = $args.group'  >> $tacsconf"
    }
    tailfService(delegate, args)

    runCommand "echo '}'  >> $tacsconf"
}

def setupTacacspServer = { myDelegate ->
    delegate = myDelegate
    addUser(delegate, "confdtester", "confd",
            [tailfgroups: "user public", tailfuid: "1000",
             tailfgid: "100", tailfhome: "/tmp"])
    addUser(delegate, "confdadmin", "adminconfd", [ group: "admin"])
    addGroup(delegate, "admin", [tailfgroups: "admin"])
    addGroup(delegate, "public",
            [tailfgroups: "public", tailfuid: "9000",
             tailfgid: "100", tailfhome: "/tmp/public"])
}

def installTacacspClient = { myDelegate ->
    delegate = myDelegate
    runCommand "pip install tacacs_plus"
}

task createConfdTacacs(type: TailfDockerfileTask) {
    tag = getProp("IMAGE_TAG", "confdtacimg")
    packages += utilsPkgs + buildPkgs +
            ["tacacs+", "git", "python-pip", "ipython"]
    configSsh rootAccess: "confd", keys: [true: ["root"] as Set]
    confd = [:]
    exposePort 22
    instructionsLast = {
        copyFilesToContext(["./auth.py"])
        copyFile("auth.py", "${confdDir}/bin")
        runCommand("chmod +x ${confdDir}/bin/auth.py")
        setupTacacspServer(delegate)
        installTacacspClient(delegate)
        editConfdConf("${confdDir}/etc/confd/confd.conf", [
                "/confdConfig/aaa/authOrder"
                : '"externalAuthentication localAuthentication"',
                "/confdConfig/aaa/externalAuthentication/enabled"
                : "true",
                "/confdConfig/aaa/externalAuthentication/executable"
                : "${confdDir}/bin/auth.py",
                "/confdConfig/logs/auditLog/enabled": "true",
                "/confdConfig/logs/auditLog/file/enabled": "true"
        ])
    }
    //start SSH, ConfD and TACACS+ daemon
    supervisor = [supervisorSsh, supervisorConfD,
              [program: "tacacsp", command: "service tacacs_plus start"]
    ]
}

task buildConfdTacacs(type: DockerBuildImage) {
    def dep = createConfdTacacs
    dependsOn dep
    dockerFile = dep.destFile
    tags = [dep.tag]
    inputDir = new File(dep.getDestDir())
}
```

## For More Information

For more information about ConfD, visit https://www.tail-f.com tail-f a Cisco company

www.tail-f.com
info@tail-f.com

**Corporate Headquarters**

Sveavagen 25
111 34 Stockholm
Sweden
+46 8 21 37 40