Challenge: Hammered Lab

Platform: CyberDefenders

Category: Endpoint Forensics

Difficulty: Medium

Tools Used: Linux Command Line Tools

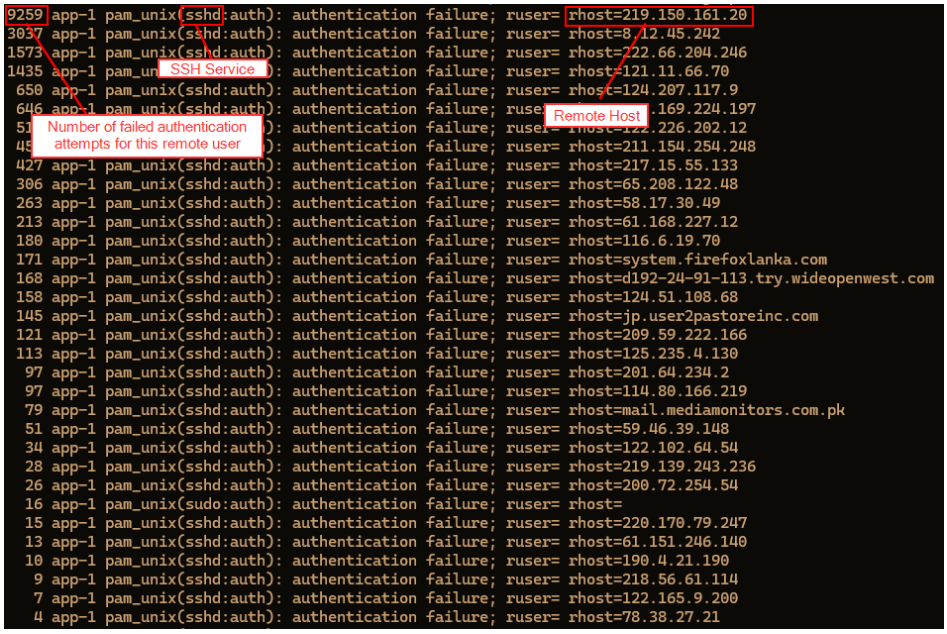
Summary: This challenge involved investigating a series of logs generated from an Ubuntu webserver honeypot. The only tools required were bash commands, such as grep, cut, etc. I found this enjoyable, and relatively easy. If you are familiar with bash and Unix logs, then this lab provides great practice.

Scenario: This challenge takes you into virtual systems and confusing log data. In this challenge, as a SOC Analyst figure out what happened to this webserver honeypot using the logs from a possibly compromised server.

Which service did the attackers use to gain access to the system?

Within Linux distributions, the auth.log file stores records of authorisation and authentication attempts on the system. To effectively filter this log, we can use the following command:

- `grep -i "authentication failure" auth.log | cut -d ' ' -f4,6,7,8,13,14 | sort | uniq -c | sort -nr`
    - This command performs a case insensitive search for logs that contain the text "authentication failure". It then extracts a series of fields from within those logs, counts the number of unique logs, and sorts it in descending order.




```
9259 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=219.150.161.20
3037 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=8.12.45.242
1573 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=222.66.204.246
1435 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=121.11.66.70
 650 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=124.207.117.9
 646 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=169.224.197
  51 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=122.226.202.12
  45 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=211.154.254.248
 427 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=217.15.55.133
 306 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=65.208.122.48
 263 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=58.17.30.49
 213 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=61.168.227.12
 180 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=116.6.19.70
 171 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=system.firefoxlanka.com
 168 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=d192-24-91-113.try.wideopenwest.com
 158 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=124.51.108.68
 145 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=jp.user2pastoreinc.com
 121 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=209.59.222.166
 113 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=125.235.4.130
  97 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=201.64.234.2
  97 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=114.80.166.219
  79 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=mail.mediamonitors.com.pk
  51 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=59.46.39.148
  34 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=122.102.64.54
  28 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=219.139.243.236
  26 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=200.72.254.54
  16 app-1 pam_unix(sudo:auth): authentication failure; ruser= rhost=
  15 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=220.170.79.247
  13 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=61.151.246.140
  10 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=190.4.21.190
   9 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=218.56.61.114
   7 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=122.165.9.200
   4 app-1 pam_unix(sshd:auth): authentication failure; ruser= rhost=78.38.27.21
```

As you can see, there is many failed authentication attempts targeting SSH from a variety of remote hosts, which is not abnormal to see on a honeypot.

Answer: ssh

What is the operating system version of the targeted system?

You can find the OS version in multiple log files, including dmesg:

```
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Linux version 2.6.24-26-server (buildd@crested) (gcc version 4.2.4 (Ubuntu 4.2.4-1ubuntu3)) #1 SMP Tue Dec 1 18:26:43 UTC 2009 (Ubuntu 2.6.24-26.64-server)
[    0.000000] Command line: root=UUID=a691743a-a4b7-482d-95ff-406e5acd83a3 ro quiet splash
[    0.000000] BIOS-provided physical RAM map:
[    0.000000]  BIOS-e820: 0000000000000000 - 000000000009f800 (usable)
[    0.000000]  BIOS-e820: 000000000009f800 - 00000000000a0000 (reserved)
[    0.000000]  BIOS-e820: 00000000000ca000 - 00000000000cc000 (reserved)
[    0.000000]  BIOS-e820: 00000000000dc000 - 00000000000e4000 (reserved)
[    0.000000]  BIOS-e820: 00000000000e8000 - 0000000000100000 (reserved)
```

Answer: 4.2.4-1ubuntu3

What is the name of the compromised account?

Recall how in the first question we observed a bunch of failed authentication attempts. If you extract another field, you can see that bulk of these failed authentication attempts are targeting the root user:

- `grep -i "authentication failure" auth.log | cut -d ' ' -f10 | sort | uniq -c | sort -nr`

```
20344 uid=0
    3 failures;
```

Failed authentication count for the root user

```
    2 'tty1'
    1 uid=1000
    1 on
    1 logname=LOGIN
```

For context, the UID of 0 on Unix-like operating systems is reserved for the root user.

Answer: root

How many attackers, represented by unique IP addresses, were able to successfully access the system after initial failed attempts?

To identify IPs associated with both failed and successful authentication attempts, we can use the following command:

- `comm -12 <(grep "Failed password for root" auth.log | awk '{for(i=1;i<=NF;i++) if($i=="from") print $(i+1)}' | sort | uniq) <(grep "Accepted password for root" auth.log | awk '{for(i=1;i<=NF;i++) if($i=="from") print $(i+1)}' | sort | uniq)`

```
10.0.1.2
121.11.66.70
122.226.202.12
188.131.23.37
219.150.161.20
222.169.224.197
222.66.204.246
61.168.227.12
94.52.185.9
```

```
   1560 219.150.161.20
   1429 121.11.66.70
    508 222.66.204.246
    313 122.226.202.12
    246 58.17.30.49
    193 61.168.227.12
    179 222.169.224.197
    122 124.207.117.9
    121 209.59.222.166
    113 116.6.19.70
     97 8.12.45.242
     78 203.81.226.86
     73 114.80.166.219
     71 211.154.254.248
     48 201.64.234.2
     48 173.9.147.165
     44 217.15.55.133
     42 59.46.39.148
     34 122.102.64.54
     28 219.139.243.236
     26 200.72.254.54
     24 125.235.4.130
     15 24.192.113.91
     13 61.151.246.140
     13 220.170.79.247
     10 190.4.21.190
      7 89.46.213.128
      5 210.68.70.170
      4 78.38.27.21
      3 222.240.223.88
      3 218.56.61.114
      2 188.131.23.37
      2 10.0.1.2
      1 94.52.185.9
      1 65.208.122.48
      1 124.51.108.68
      1 122.165.9.200
      1 12.172.224.140
```

As you can see, six unique public IP addresses were observed successfully authenticating to the root user after conducting a brute force attack.

Answer: 6

Which attacker's IP address successfully logged into the system the most number of times?

Using the following command, we can see that 219.150.161.20 logged into the system 4 times:

- `grep -i "accepted password for root" auth.log | cut -d ' ' -f11 | sort | uniq -c | sort -nr`

```
4 219.150.161.20
4 188.131.23.37
3 190.166.87.164
2 122.226.202.12
2 121.11.66.70
1 94.52.185.9
1 61.168.227.12
1 222.66.204.246
1 222.169.224.197
1 201.229.176.217
1 193.1.186.197
1 190.167.74.184
1 190.167.70.87
1 188.131.22.69
1 151.82.3.201
1 151.81.205.100
1 151.81.204.141
1 10.0.1.2
```

Answer: 219.150.161.20

How many requests were sent to the Apache Server?

The Apache www-access.log records every request processed by the server. We can read this log using cat and pipe the output to wc -l to count the number of lines (i.e., the number of requests):

- `cat www-access.log | wc -l`

```
/42-Hammered/temp_extract_dir/Hammered/apache2$ cat www-access.log | wc -l
365
```

Answer: 365

How many rules have been added to the firewall?

Aside from storing authentication logs, the auth.log records commands executed using elevated privileges. To modify the firewall used by the Ubuntu system (which is likely iptables), you would need to execute commands as an elevated user. We can use the following grep command to filter for these elevated commands:

- `grep -i COMMAND auth.log`

Near the bottom of the output, we can see a series of iptables commands:

```
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/sbin/iptables -L
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/usr/sbin/ufw allow 53
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/usr/sbin/ufw allow 113
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/usr/sbin/ufw disable
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/usr/sbin/ufw enable
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/usr/sbin/ufw disable
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/usr/bin/apt-get update
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/usr/bin/apt-get upgrade
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/sbin/iptables -A INPUT -p ssh -dport 2424 -j ACCEPT
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/sbin/iptables -A INPUT -p tcp -dport 53 -j ACCEPT
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/sbin/iptables -A INPUT -p udp -dport 53 -j ACCEPT
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/sbin/iptables -A INPUT -p tcp --dport ssh -j ACCEPT
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/sbin/iptables -A INPUT -p tcp --dport 53 -j ACCEPT
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/sbin/iptables -A INPUT -p tcp --dport 113 -j ACCEPT
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/usr/sbin/ufw disable
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/usr/sbin/ufw enable
```

To cut down the results further, we can grep for iptables:

- `grep -i iptables auth.log`

```
user1 : TTY=pts/0 ; PWD=/opt/software/web/app ; USER=root ; COMMAND=/usr/bin/tee ../templates/proxy/iptables.conf
user1 : TTY=pts/1 ; PWD=/opt/software/web/app ; USER=root ; COMMAND=/usr/bin/tee ../templates/proxy/iptables.conf
user1 : TTY=pts/1 ; PWD=/opt/software/web/app ; USER=root ; COMMAND=/usr/bin/tee ../templates/proxy/iptables.conf
user1 : TTY=pts/1 ; PWD=/opt/software/web/app ; USER=root ; COMMAND=/usr/bin/tee ../templates/proxy/iptables.conf
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/sbin/iptables -L
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/sbin/iptables -A INPUT -p ssh -dport 2424 -j ACCEPT
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/sbin/iptables -A INPUT -p tcp -dport 53 -j ACCEPT
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/sbin/iptables -A INPUT -p udp -dport 53 -j ACCEPT
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/sbin/iptables -A INPUT -p tcp --dport ssh -j ACCEPT
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/sbin/iptables -A INPUT -p tcp --dport 53 -j ACCEPT
  root : TTY=pts/2 ; PWD=/etc ; USER=root ; COMMAND=/sbin/iptables -A INPUT -p tcp --dport 113 -j ACCEPT
```

As you can see in the above image, 6 rules have been added to iptables:

- Allow SSH traffic on port 2424
- Allow TCP traffic on port 53
- Allow UDP traffic on port 53
- Allow SSH traffic on the default SSH port
- Allow TCP traffic on port 113

Answer: 6

One of the downloaded files on the target system is a scanning tool. What is the name of the tool?

The dpkg.log tracks the installation, update, and removal of software packages on the system. We can search through this log to identify the installation of the scanning tool. The most known network scanning tool is nmap, if we grep for nmap, we can see that it was installed on the system:

- `grep nmap dpkg.log`

```
2010-04-24 19:38:15 install nmap <none> 4.53-3
2010-04-24 19:38:15 status half-installed nmap 4.53-3
2010-04-24 19:38:15 status unpacked nmap 4.53-3
2010-04-24 19:38:15 status unpacked nmap 4.53-3
2010-04-24 19:38:16 configure nmap 4.53-3 4.53-3
2010-04-24 19:38:16 status unpacked nmap 4.53-3
2010-04-24 19:38:16 status half-configured nmap 4.53-3
2010-04-24 19:38:16 status installed nmap 4.53-3
```

Answer: nmap

When was the last login from the attacker with IP 219.150.161.20? Format: MM/DD/YYYY HH:MM:SS AM

If we grep for successful authentications and the threat actors IP, we can find the last time the threat actor logged in:

- `grep -i "accepted password" auth.log | grep 219.150.161.20`

```
Apr 19 05:41:44 app-1 sshd[8810]: Accepted password for root from 219.150.161.20 port 51249 ssh2
Apr 19 05:42:27 app-1 sshd[9031]: Accepted password for root from 219.150.161.20 port 40877 ssh2
Apr 19 05:55:20 app-1 sshd[12996]: Accepted password for root from 219.150.161.20 port 55545 ssh2
Apr 19 05:56:05 app-1 sshd[13218]: Accepted password for root from 219.150.161.20 port 36585 ssh2
```

Unfortunately, auth.log doesn't show the year a log was generated. We can use exiftool to find the last modification date, and assume that this log was generated on the same year:

- `exiftool auth.log`

```
ExifTool Version Number         : 12.76
File Name                       : auth.log
Directory                       : .
File Size                       : 10 MB
File Modification Date/Time     : 2010:07:04 03:53:20+10:00
File Access Date/Time           : 2010:07:04 19:38:57+10:00
File Inode Change Date/Time     : 2025:08:11 22:12:38+10:00
File Permissions                : -rwxrwxrwx
File Type                       : TXT
File Type Extension             : txt
MIME Type                       : text/plain
MIME Encoding                   : us-ascii
Newlines                        : Unix LF
Line Count                      : 102164
Word Count                      : 1289927
```

Answer: 2010-04-19 05:56

The database showed two warning messages. Please provide the most critical and potentially dangerous one.

The daemon.log file stores messages from system and application daemons and is where we can find warning messages from a database application. We can use the following command to search for logs that contain sql and warning:

- `grep -i "sql" daemon.log | grep -i "warning"`

This outputs 15 warnings for mysql:

```
Mar 18 10:18:42 app-1 /etc/mysql/debian-start[7566]: WARNING: mysql.user contains 2 root accounts without password!
Mar 18 17:01:44 app-1 /etc/mysql/debian-start[14717]: WARNING: mysql.user contains 2 root accounts without password!
Mar 22 13:49:49 app-1 /etc/mysql/debian-start[5599]: WARNING: mysql.user contains 2 root accounts without password!
Mar 22 18:43:41 app-1 /etc/mysql/debian-start[4755]: WARNING: mysql.user contains 2 root accounts without password!
Mar 22 18:45:25 app-1 /etc/mysql/debian-start[4749]: WARNING: mysql.user contains 2 root accounts without password!
Mar 25 11:56:53 app-1 /etc/mysql/debian-start[4848]: WARNING: mysql.user contains 2 root accounts without password!
Apr 14 14:44:34 app-1 /etc/mysql/debian-start[5369]: WARNING: mysql.user contains 2 root accounts without password!
Apr 14 14:44:36 app-1 /etc/mysql/debian-start[5624]: WARNING: mysqlcheck has found corrupt tables
Apr 18 18:04:00 app-1 /etc/mysql/debian-start[4647]: WARNING: mysql.user contains 2 root accounts without password!
Apr 24 20:21:24 app-1 /etc/mysql/debian-start[5427]: WARNING: mysql.user contains 2 root accounts without password!
Apr 28 07:34:26 app-1 /etc/mysql/debian-start[4782]: WARNING: mysql.user contains 2 root accounts without password!
Apr 28 07:34:27 app-1 /etc/mysql/debian-start[5032]: WARNING: mysqlcheck has found corrupt tables
Apr 28 07:34:27 app-1 /etc/mysql/debian-start[5032]: warning  : 1 client is using or hasn't closed the table properly
Apr 28 07:34:27 app-1 /etc/mysql/debian-start[5032]: warning  : 1 client is using or hasn't closed the table properly
May  2 23:05:54 app-1 /etc/mysql/debian-start[4774]: WARNING: mysql.user contains 2 root accounts without password!
```

Whilst there are 15 separate warnings, it's clear that the most concerning ones are mysql.user contains 2 root accounts without password!

Answer: mysql.user contains 2 root accounts without password!

Multiple accounts were created on the target system. Which account was created on April 26 at 04:43:15?

On Ubuntu, the useradd command is used to create new user accounts. This action is logged in the auth.log file, enabling us to query the useradd events by using grep:

- `grep -i useradd auth.log`

```
Mar 16 08:12:13 app-1 useradd[4692]: new user: name=user4, UID=1001, GID=1001, home=/home/user4, shell=/bin/bash
Mar 16 08:12:38 app-1 useradd[4703]: new user: name=user1, UID=1001, GID=1001, home=/home/user1, shell=/bin/bash
Mar 16 08:12:55 app-1 useradd[4711]: new user: name=user2, UID=1002, GID=1002, home=/home/user2, shell=/bin/bash
Mar 16 08:25:22 app-1 useradd[4845]: new user: name=sshd, UID=104, GID=65534, home=/var/run/sshd, shell=/usr/sbin/nologin
Mar 18 10:15:42 app-1 useradd[5393]: new user: name=Debian-exim, UID=105, GID=114, home=/var/spool/exim4, shell=/bin/false
Mar 18 10:18:26 app-1 useradd[6966]: new user: name=mysql, UID=106, GID=115, home=/var/lib/mysql, shell=/bin/false
Apr 19 22:38:00 app-1 useradd[2019]: new user: name=packet, UID=0, GID=0, home=/home/packet, shell=/bin/sh
Apr 19 22:45:13 app-1 useradd[2053]: new user: name=dhg, UID=1003, GID=1003, home=/home/dhg, shell=/bin/bash
Apr 24 19:27:35 app-1 useradd[1386]: new user: name=messagebus, UID=108, GID=117, home=/var/run/dbus, shell=/bin/false
Apr 25 10:41:44 app-1 useradd[9596]: new group: name=fido, GID=1004
Apr 25 10:41:44 app-1 useradd[9596]: new user: name=fido, UID=0, GID=1004, home=/home/fido, shell=/bin/sh
Apr 26 04:43:15 app-1 useradd[20115]: new user: name=wind3str0y, UID=1004, GID=1005, home=/home/wind3str0y, shell=/bin/bash
```

We can see that only one user was created on April 26 at 04:43:15 called wind3str0y.

Answer: wind3str0y

Few attackers were using a proxy to run their scans. What is the corresponding user-agent used by this proxy?

Recall earlier how we discussed that the www-access.log stores all requests proceeded by the web server. Using the following command, we can extract all the user-agent strings associated with these requests:

- `cat www-access.log | cut -d ' ' -f12 | sort | uniq -c | sort -nr`

```
272 "Apple-PubSub/65.12.1"
 32 "Mozilla/5.0
 32 "Mozilla/4.0
 18 "WordPress/2.9.2;
  8 "-"
  3 "pxyscand/2.1"
```

The interesting user-agent string is pxyscand/2.1.

Answer: pxyscand/2.1