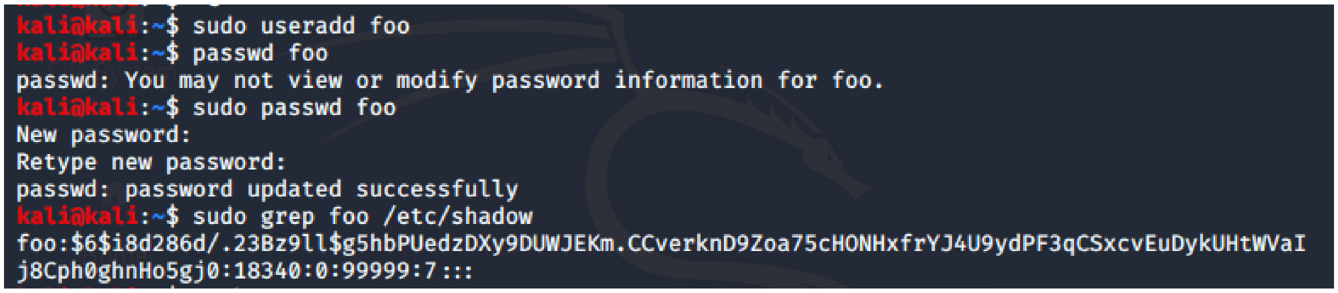
Lab 12
Varun Gunda
A20453991

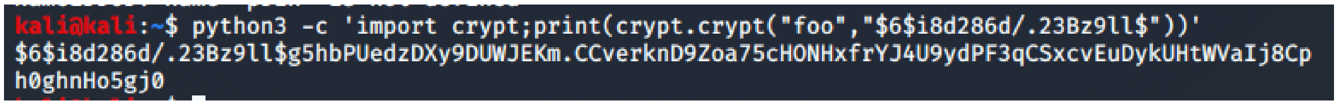
Task 1: Playing the Fool



```
kali@kali:~$ sudo useradd foo
kali@kali:~$ passwd foo
passwd: You may not view or modify password information for foo.
kali@kali:~$ sudo passwd foo
New password:
Retype new password:
passwd: password updated successfully
kali@kali:~$ sudo grep foo /etc/shadow
foo:$6$i8d286d/.23Bz9ll$g5hbPUedzDXy9DUWJEKm.CCverknD9Zoa75cHONHxfrYJ4U9ydPF3qCSxcvEuDykUHtWVaI
j8Cph0ghnHo5gj0:18340:0:99999:7:::
```

Created a user foo with password foo. The password is stored as displayed above

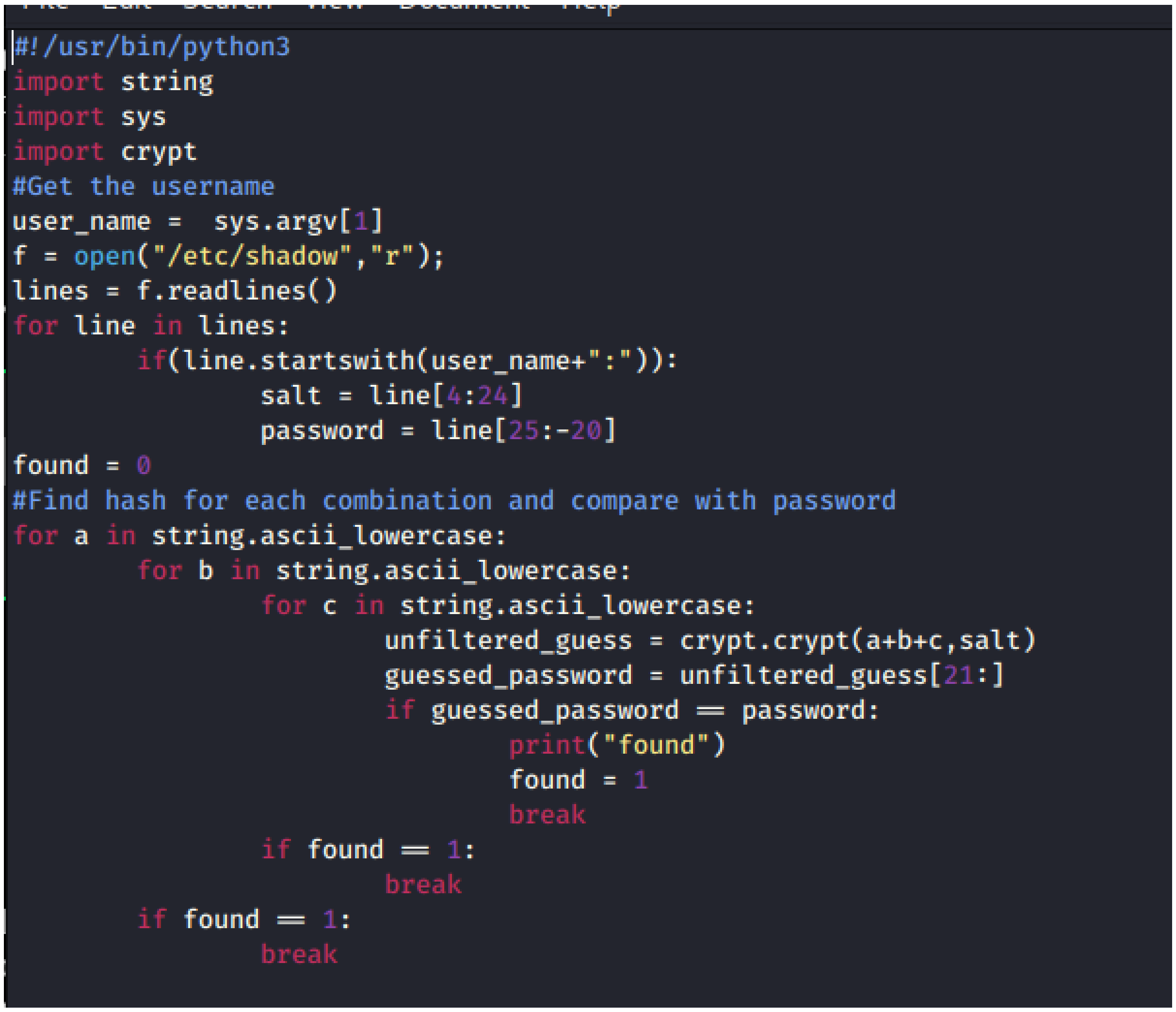
On using Python’s API to crypt with same salt, we get the same password:



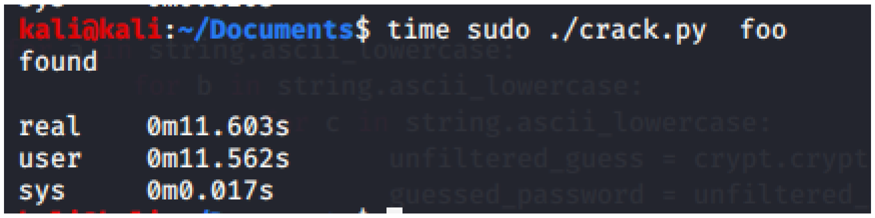
```
kali@kali:~$ python3 -c 'import crypt;print(crypt.crypt("foo","$6$i8d286d/.23Bz9ll$"))'
$6$i8d286d/.23Bz9ll$g5hbPUedzDXy9DUWJEKm.CCverknD9Zoa75cHONHxfrYJ4U9ydPF3qCSxcvEuDykUHtWVaIj8Cp
h0ghnHo5gj0
```

Task 2: Cracking

Following is the script written to test every three-letter possibility to crack foo’s password



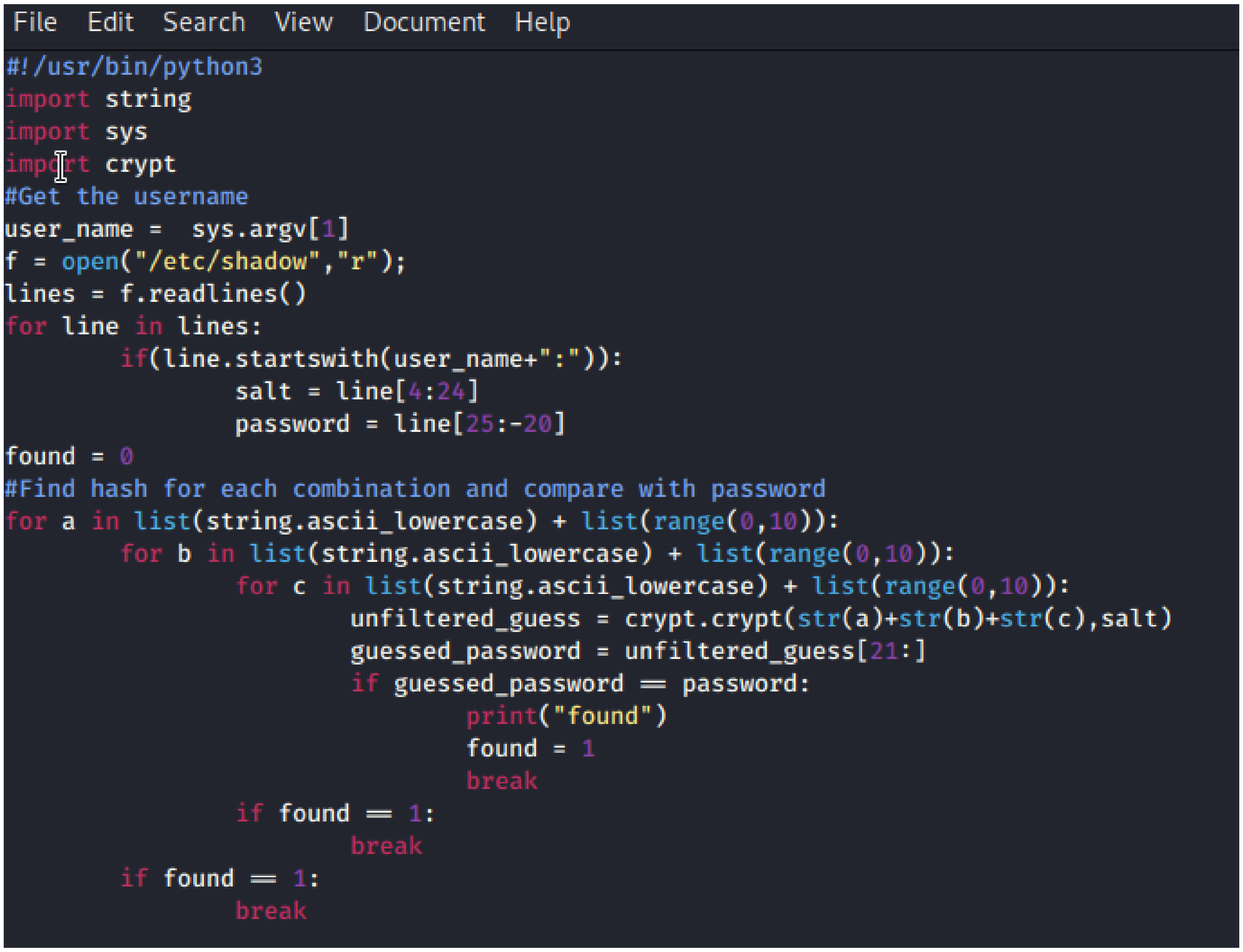
```
#!/usr/bin/python3
import string
import sys
import crypt
#Get the username
user_name =  sys.argv[1]
f = open("/etc/shadow","r");
lines = f.readlines()
for line in lines:
        if(line.startswith(user_name+":")):
                salt = line[4:24]
                password = line[25:-20]
found = 0
#Find hash for each combination and compare with password
for a in string.ascii_lowercase:
        for b in string.ascii_lowercase:
                for c in string.ascii_lowercase:
                        unfiltered_guess = crypt.crypt(a+b+c,salt)
                        guessed_password = unfiltered_guess[21:]
                        if guessed_password == password:
                                print("found")
                                found = 1
                                break
                if found == 1:
                        break
        if found == 1:
                break
```



```
kali@kali:~/Documents$ time sudo ./crack.py  foo
found

real    0m11.603s
user    0m11.562s
sys     0m0.017s
```

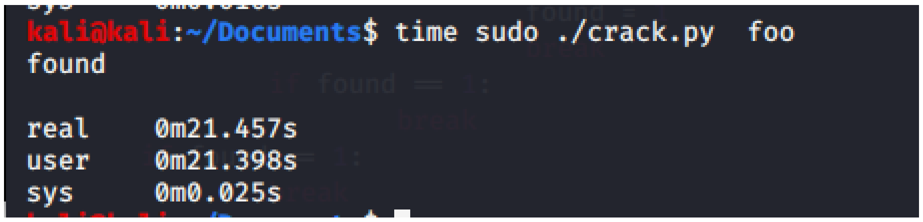
It roughly takes 11 seconds to do this on my machine.

On including numerals as seen below, the time further increases.



```
#!/usr/bin/python3
import string
import sys
import crypt
#Get the username
user_name =  sys.argv[1]
f = open("/etc/shadow","r");
lines = f.readlines()
for line in lines:
        if(line.startswith(user_name+":")):
                salt = line[4:24]
                password = line[25:-20]
found = 0
#Find hash for each combination and compare with password
for a in list(string.ascii_lowercase) + list(range(0,10)):
        for b in list(string.ascii_lowercase) + list(range(0,10)):
                for c in list(string.ascii_lowercase) + list(range(0,10)):
                        unfiltered_guess = crypt.crypt(str(a)+str(b)+str(c),salt)
                        guessed_password = unfiltered_guess[21:]
                        if guessed_password == password:
                                print("found")
                                found = 1
                                break
                if found == 1:
                        break
        if found == 1:
                break
```

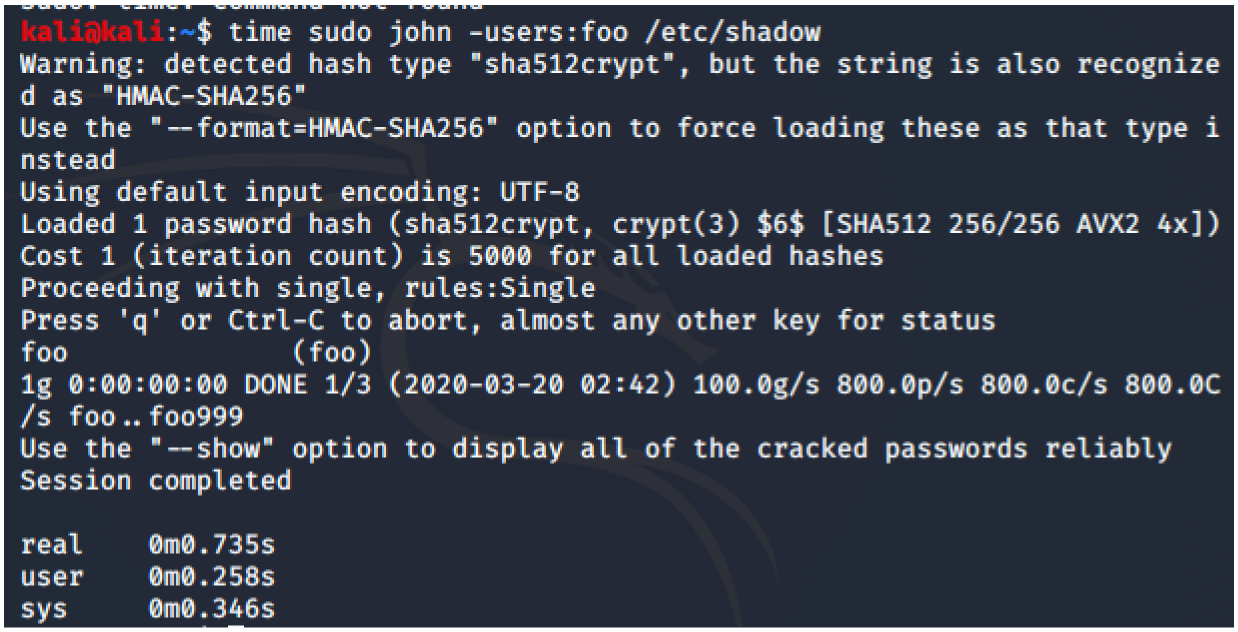
And this time it is 21.4 seconds



```
kali@kali:~/Documents$ time sudo ./crack.py  foo
found

real    0m21.457s
user    0m21.398s
sys     0m0.025s
```

Task 3: Cracking with John

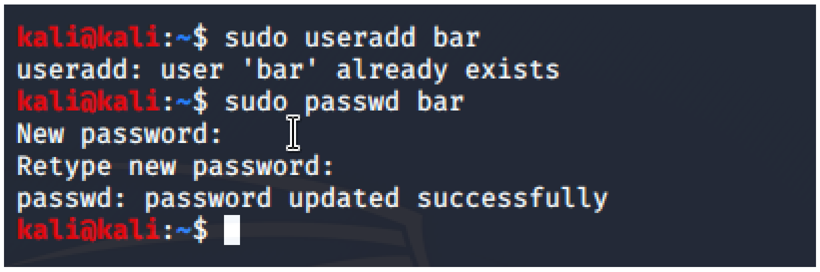
John the Ripper tool is used as shown below to crack the password of the user foo. This tool is much faster compared to the tool crack.py which took around 21 seconds but this took only 0.735 seconds to crack the password. Also, crack.py’s time would keep on increasing if we add other characters instead of only alphabet and numerals.



```
kali@kali:~$ time sudo john -users:foo /etc/shadow
Warning: detected hash type "sha512crypt", but the string is also recognize
d as "HMAC-SHA256"
Use the "--format=HMAC-SHA256" option to force loading these as that type i
nstead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
foo              (foo)
1g 0:00:00:00 DONE 1/3 (2020-03-20 02:42) 100.0g/s 800.0p/s 800.0c/s 800.0C
/s foo..foo999
Use the "--show" option to display all of the cracked passwords reliably
Session completed

real    0m0.735s
user    0m0.258s
sys     0m0.346s
```

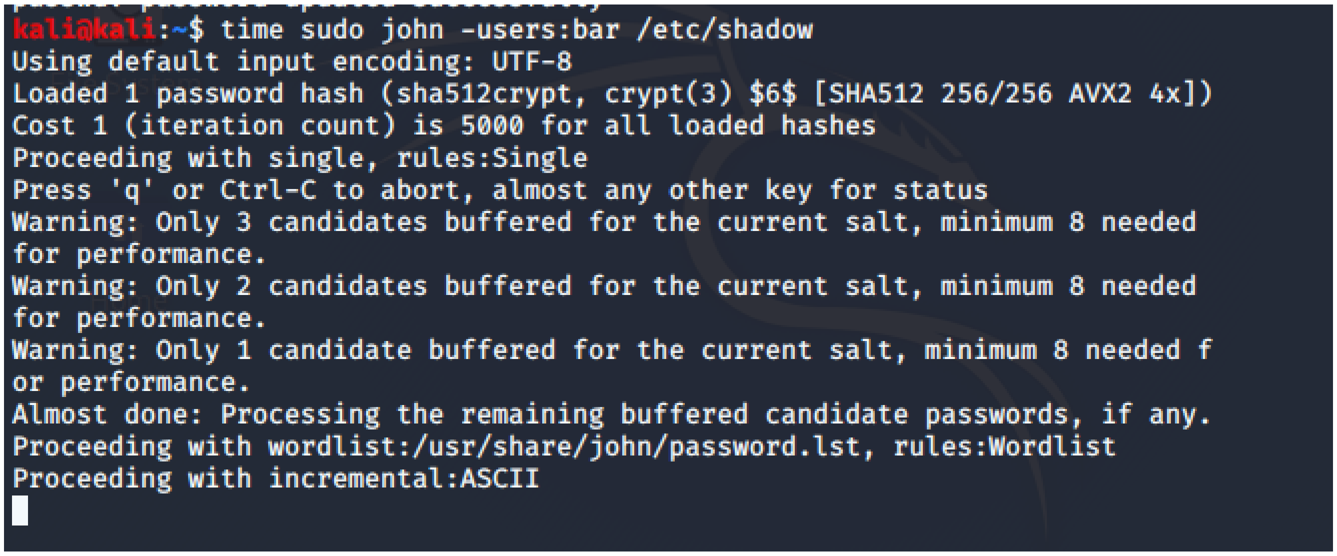
Task 4: Dictionary Attacks with Transformation Rules

The new user is created “bar” and its password is set as whit3s0cks1994.



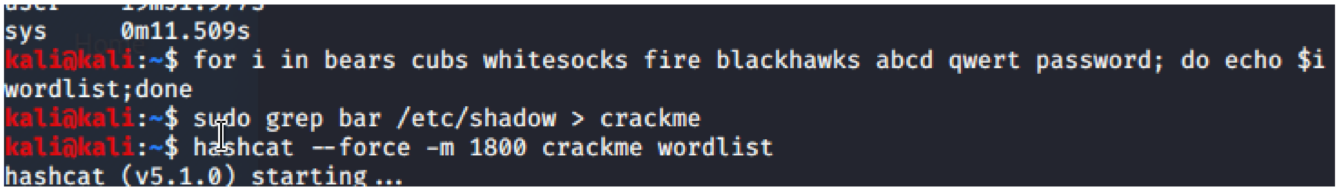
```
kali@kali:~$ sudo useradd bar
useradd: user 'bar' already exists
kali@kali:~$ sudo passwd bar
New password:
Retype new password:
passwd: password updated successfully
kali@kali:~$
```

On trying with john it just kept running for very long time as seen below. This is because of lengthy password and mix of symbols and alphabet.

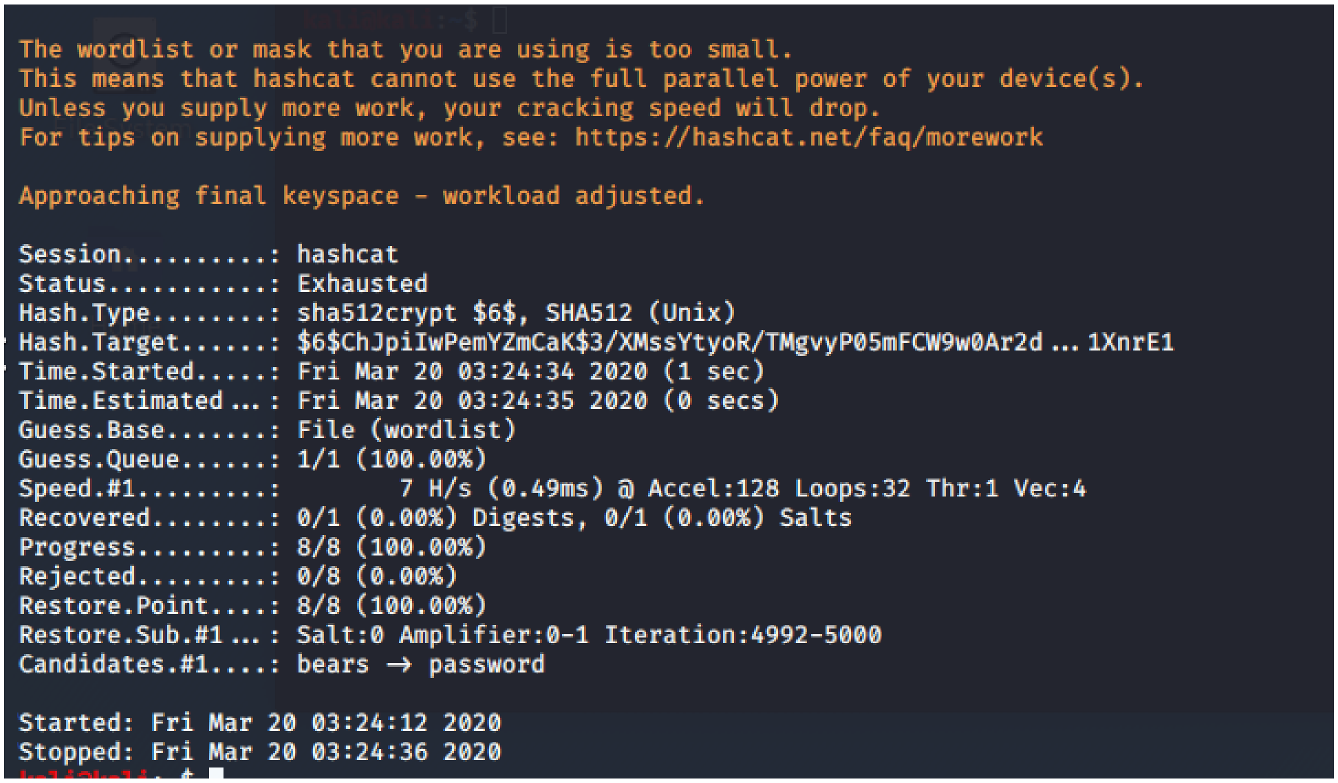


```
kali@kali:~$ time sudo john -users:bar /etc/shadow
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 3 candidates buffered for the current salt, minimum 8 needed
for performance.
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed
for performance.
Warning: Only 1 candidate buffered for the current salt, minimum 8 needed f
or performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
```

Adding words to the word list as shown below and giving it as an input to hashcat:

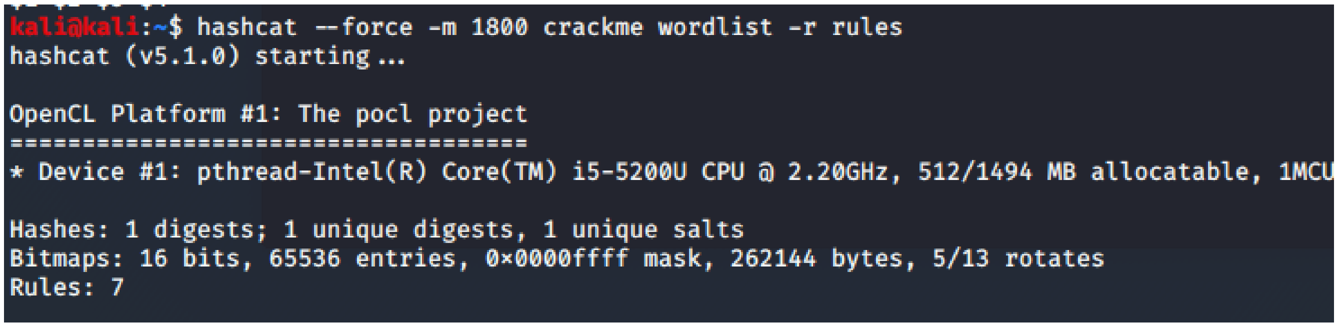


```
sys     0m11.509s
kali@kali:~$ for i in bears cubs whitesocks fire blackhawks abcd qwert password; do echo $i
wordlist;done
kali@kali:~$ sudo grep bar /etc/shadow > crackme
kali@kali:~$ hashcat --force -m 1800 crackme wordlist
hashcat (v5.1.0) starting...
```

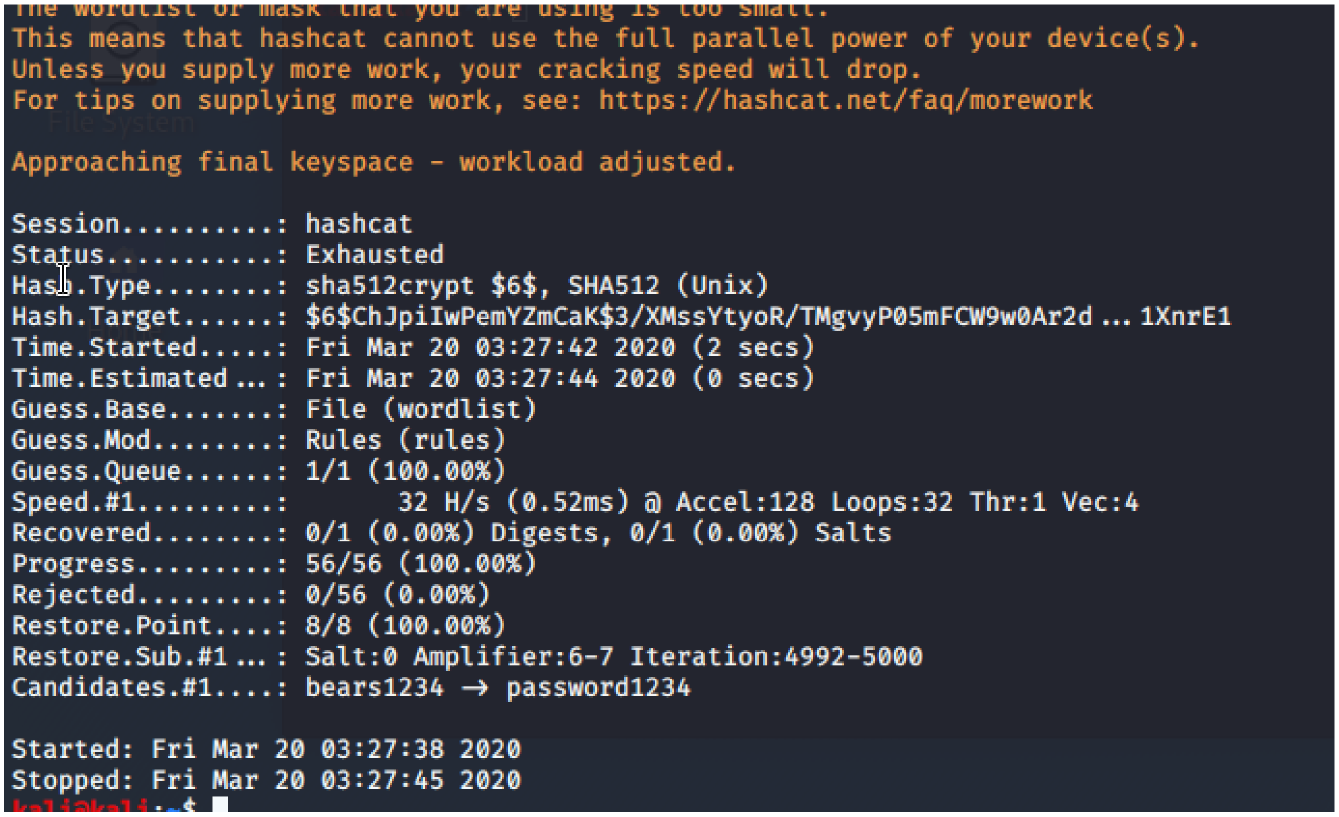

```
The wordlist or mask that you are using is too small.
This means that hashcat cannot use the full parallel power of your device(s).
Unless you supply more work, your cracking speed will drop.
For tips on supplying more work, see: https://hashcat.net/faq/morework

Approaching final keyspace - workload adjusted.

Session..........: hashcat
Status...........: Exhausted
Hash.Type........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: $6$ChJpiIwPemYZmCaK$3/XMssYtyoR/TMgvyP05mFCW9w0Ar2d ... 1XnrE1
Time.Started.....: Fri Mar 20 03:24:34 2020 (1 sec)
Time.Estimated ...: Fri Mar 20 03:24:35 2020 (0 secs)
Guess.Base.......: File (wordlist)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:        7 H/s (0.49ms) @ Accel:128 Loops:32 Thr:1 Vec:4
Recovered........: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 8/8 (100.00%)
Rejected.........: 0/8 (0.00%)
Restore.Point....: 8/8 (100.00%)
Restore.Sub.#1 ... : Salt:0 Amplifier:0-1 Iteration:4992-5000
Candidates.#1....: bears → password

Started: Fri Mar 20 03:24:12 2020
Stopped: Fri Mar 20 03:24:36 2020
```

The above command hashcat finishers quickly but fails.

Now adding rules that were mentioned in the task and giving it as input to crackme:


```
kali@kali:~$ hashcat --force -m 1800 crackme wordlist -r rules
hashcat (v5.1.0) starting...

OpenCL Platform #1: The pocl project
====================================
* Device #1: pthread-Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz, 512/1494 MB allocatable, 1MCU

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 7
```

But still, we don't get crack password as seen below:

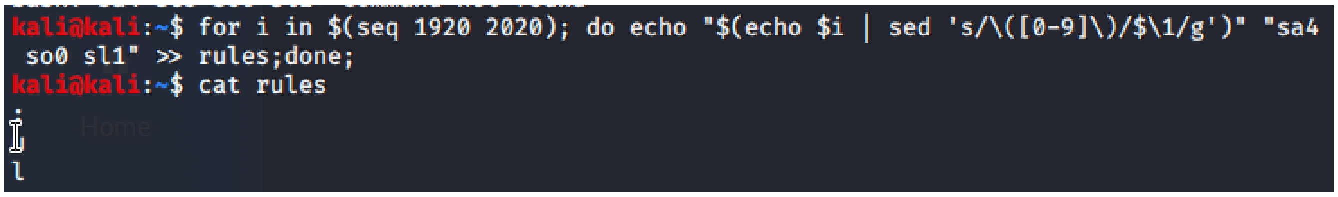


```
The wordlist or mask that you are using is too small.
This means that hashcat cannot use the full parallel power of your device(s).
Unless you supply more work, your cracking speed will drop.
For tips on supplying more work, see: https://hashcat.net/faq/morework

Approaching final keyspace - workload adjusted.

Session..........: hashcat
Status...........: Exhausted
Hash.Type........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: $6$ChJpiIwPemYZmCaK$3/XMssYtyoR/TMgvyP05mFCW9w0Ar2d ... 1XnrE1
Time.Started.....: Fri Mar 20 03:27:42 2020 (2 secs)
Time.Estimated...: Fri Mar 20 03:27:44 2020 (0 secs)
Guess.Base.......: File (wordlist)
Guess.Mod........: Rules (rules)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:       32 H/s (0.52ms) @ Accel:128 Loops:32 Thr:1 Vec:4
Recovered........: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 56/56 (100.00%)
Rejected.........: 0/56 (0.00%)
Restore.Point....: 8/8 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:6-7 Iteration:4992-5000
Candidates.#1....: bears1234 → password1234

Started: Fri Mar 20 03:27:38 2020
Stopped: Fri Mar 20 03:27:45 2020
kali@kali:~$
```

Adding more rules

```
$ echo "sa4 se3 sl1 so0" >> rules
```

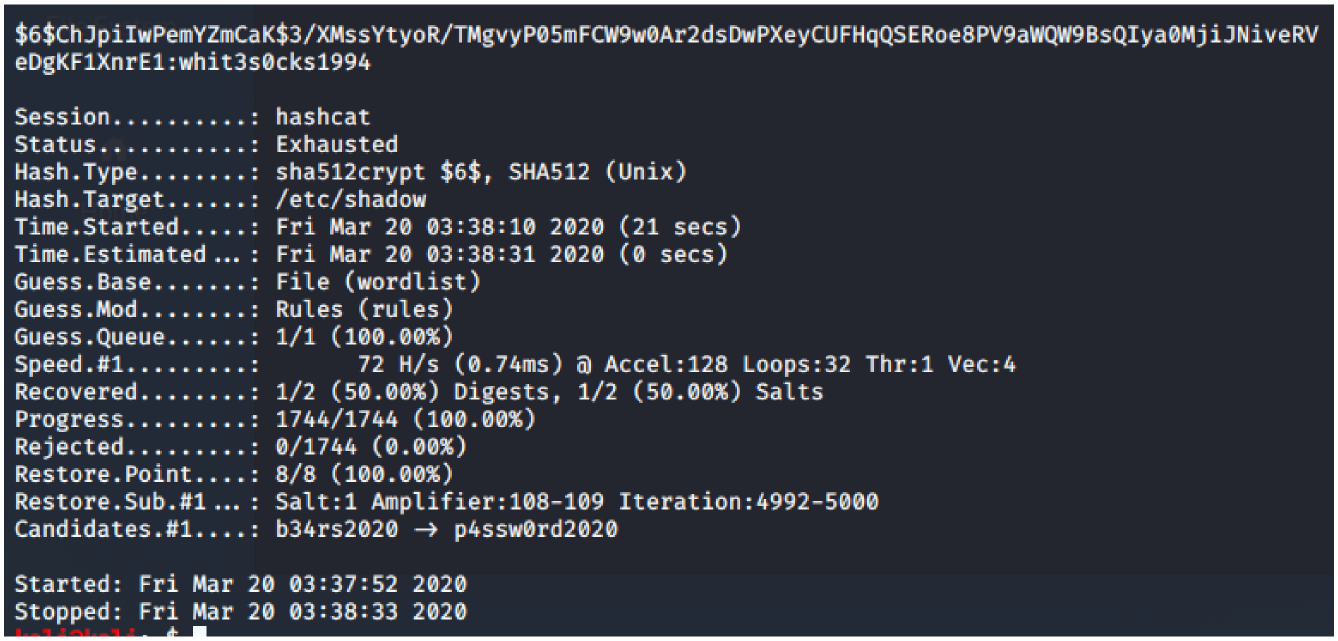
and



```
kali@kali:~$ for i in $(seq 1920 2020); do echo "$(echo $i | sed 's/\([0-9]\)/$\1/g')" "sa4
 so0 sl1" >> rules;done;
kali@kali:~$ cat rules
:
l
l
```

This time, on executing the following command, we get the password as seen in the next image.

```
hashcat --force -m 1800 /etc/shadow wordlist -r rules
```



```
$6$ChJpiIwPemYZmCaK$3/XMssYtyoR/TMgvyP05mFCW9w0Ar2dsDwPXeyCUFHqQSERoe8PV9aWQW9BsQIya0MjiJNiveRV
eDgKF1XnrE1:whit3s0cks1994

Session..........: hashcat
Status...........: Exhausted
Hash.Type........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: /etc/shadow
Time.Started.....: Fri Mar 20 03:38:10 2020 (21 secs)
Time.Estimated...: Fri Mar 20 03:38:31 2020 (0 secs)
Guess.Base.......: File (wordlist)
Guess.Mod........: Rules (rules)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:       72 H/s (0.74ms) @ Accel:128 Loops:32 Thr:1 Vec:4
Recovered........: 1/2 (50.00%) Digests, 1/2 (50.00%) Salts
Progress.........: 1744/1744 (100.00%)
Rejected.........: 0/1744 (0.00%)
Restore.Point....: 8/8 (100.00%)
Restore.Sub.#1...: Salt:1 Amplifier:108-109 Iteration:4992-5000
Candidates.#1....: b34rs2020 → p4ssw0rd2020

Started: Fri Mar 20 03:37:52 2020
Stopped: Fri Mar 20 03:38:33 2020
```

Task 5: Putting it Together

Initially taking all the words from halek.co website's personal section and feeding it into the words list similar to the above task and applying the same rules, hashcat is not successful and is exhausted. But on using combinators.bin tool provided by hashcat-utils to generate wordlist that contains 2 words appended(each word of the file is appended to every other word in the file) proved to be successful in this attack. The password that is cracked by hashcat is "s0nnys4x1987".