Team –

a. **Suraj Bhatia**
b. **Soumya Mohanty**

1. **Include a copy of the four password files.**

Password Cracking :-

Accessing the Password Hashes:

The purpose of this step is to get the files containing all the password hashes from Linux and Windows VMs.

By using "pwdump" we gained access to the windows LM hashes and stored them into the file "windows_passwds.txt" . The command used was-

Pwdump localhost >> windows_passwds.txt

Then we move on to the Linux VM, to gather the hashes from both /etc/passwd and /etc/shadow and unshadowing them . This is done by using-

unshadow /etc/passwd /etc/shadow > linux_passwords.txt

Then we remove the fields with no passwords and the pruned files are- linux_passwords_pruned.txt and windows_passwords_pruned.txt

These files contain the username and hash of the respective password.

Two files were also downloaded from http://strawman.nslab/lab6/. The names of these files were lin_pwd.txt and win-pwd.txt

We obtained 4 password files in total, their content after pruning is as follows:

1. windows_passwd_pruned.txt

Administrator:500:19AEF73B3E9995CFD8768A463BD7D00D:848F4B13A7CB0568A8F4F496E9768066:::
:503:8B7B1E1005D9851A5B5D54ACD64E1AB6:74D0C3DF46E1D9D35229F74936490FA5:::
robert:1011:4F315D32FD63DDE5AAD3B435B51404EE:D6B3D2E9E6D4EE5443996E01BC448CBC:::
andrew:1012:0BDCCA0EABFF8A8BAAD3B435B51404EE:2A6BB0E4AAA8E933661977EC5EFD02F6:::
carlos:1013:95C475EEF9DAD191AAD3B435B51404EE:4A546A8FD69A1FE0121C00EB7404B773:::
andrea:1014:23D75446FB6E0A1CAAD3B435B51404EE:15E837FB855C92A70A1D75B91226E924:::
michael:1015:3986CFFB5BD9EA86AAD3B435B51404EE:4BE619B5D33FD71EB222AD1DF041B8BC:::
richard:1016:D5C0A71E867EF4DAAAD3B435B51404EE:A3D5EAE39100B5276794C11DA366B071:::
smith:1017:3A539909A32DEC39AAD3B435B51404EE:3CF77AE0DFA519BCDD7CFCD64B3FAE16:::
mohanty.s:1018:C11BDE9A3B94B760922A93340EB6804D:0E01EE5B464AEEE66A26CD5EBDF8E658:::

2. linux_passwords_pruned.txt

root:!:0:0:root:/root:/bin/bash
uuidd:!:100:101::/run/uuidd:/bin/false

netsec-admin:6R5lYLS74$YXLmjhZlFgU/3UBm0m.qQZ5noHZDdm34Ie51IeRdQHIohoaXdLKvcx8Esm1ef9whzI.XmCbu/1QDzakmklxrE0:1000:1000:Netsec TA,,,:/home/netsec-admin:/bin/bash
team:6F09eQ9ZM$.XSdFNgK39hMppnISPRyh0C.VYbl2JeTUEUPE0IU2kBP.qDZ9Cwf6aRi2pKV6.2H96dfGZebogWm8RfFVovl5/:1001:1001:Team Login:/home/team:/bin/bash
mohanty.s:6kgdsihr7$6oka.XP0G.asFr.qMsYn/XdNNJP0qEK1Ui/dL18SGUdGLbQ0XCbs7HycCmCphjoTeIcROjQLQmbBF7hVGwlcS/:1002:1002::/home/mohanty.s:/bin/bash

3. lin_pwd.txt

   magellan:1oMRtGdiY$CdaII9KyvKWqKsYBwJwVq0:1002:1002::/home/magellan:/bin/sh

   marcopolo:1OUnMA/nW$XdvYJVquDCgce0cCrBuiJ1:1003:1003::/home/marcopolo:/bin/sh

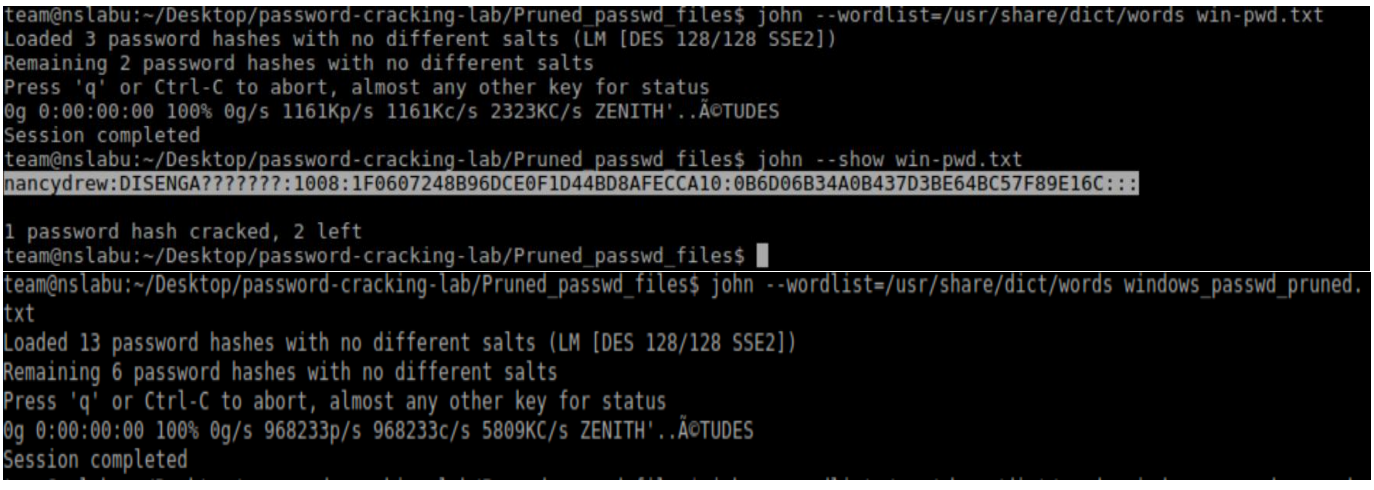
4. win-pwd.txt

   nancydrew:1008:1F0607248B96DCE0F1D44BD8AFECCA10:0B6D06B34A0B437D3BE64BC57F89E16C:::
   sherlock:1007:D568A3C648982EE3AAD3B435B51404EE:7F6010574E24B264D81F0225164AEE6C:::

2. Include a list of all passwords that you cracked, along with the technique used for cracking (dictionary, brute-force or precomputation). Each row should have the username, the password hash, and the cleartext password.

Dictionary Attack:

This type of attack compares the words present in a dictionary of words, also referred to as wordlists to the hash provided in the file. It does so by converting the words present in the file into hashes and then comparing them. For our test we used Linux system's built-in spell-check dictionary (/usr/share/dict/words). We used it against all the four password hash files obtained, and could only get one partial password.



```
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$ john --wordlist=/usr/share/dict/words lin_pwd.txt
Loaded 2 password hashes with 2 different salts (md5crypt [MD5 32/32])
Remaining 1 password hash
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:07 57% 0g/s 8252p/s 8252c/s 8252C/s lewder
0g 0:00:00:10 84% 0g/s 8357p/s 8357c/s 8357C/s smoke's
0g 0:00:00:11 100% 0g/s 8401p/s 8401c/s 8401C/s Ã©tudes
Session completed
```

By using this technique, we obtained a partial password-

nancydrew: DISENGA????? :0B6D06B34A0B437D3BE64BC57F89E16C

Then we used the command: john <filename>

This command runs John in single mode. In single mode it uses the “GECOS”/ ”Full name” fields present in the file as a wordlist. These fields contain the general information of the user and are not very big in size. More-over John the Ripper applies a large set of word mangling rules, which help it produce many possible passwords. These techniques put together increase the chances of cracking the passwords. We cracked a considerable number of passwords using this technique.

```
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$ john win-pwd.txt
Loaded 3 password hashes with no different salts (LM [DES 128/128 SSE2])
Remaining 1 password hash
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:04 3/3 0g/s 23635Kp/s 23635Kc/s 23635KC/s 056APOS..056APTO
0g 0:00:00:43 3/3 0g/s 31788Kp/s 31788Kc/s 31788KC/s IZZYREG..IZZYNDG
0g 0:00:03:05 3/3 0g/s 32448Kp/s 32448Kc/s 32448KC/s ROKGR3I..ROKGR29
0g 0:00:08:43 3/3 0g/s 32593Kp/s 32593Kc/s 32593KC/s 3EDTPNO..3EDTPYO
0g 0:00:13:08 3/3 0g/s 32381Kp/s 32381Kc/s 32381KC/s JM8P82V..JM8P8BI
0g 0:00:17:48 3/3 0g/s 32480Kp/s 32480Kc/s 32480KC/s 68YIEHN..68YI12T
0g 0:00:20:01 3/3 0g/s 32477Kp/s 32477Kc/s 32477KC/s DRSIBJ!..DRSIQKG
0g 0:00:23:04 3/3 0g/s 31986Kp/s 31986Kc/s 31986KC/s 44Z9L0P..44Z9LJT
0g 0:00:23:05 3/3 0g/s 31988Kp/s 31988Kc/s 31988KC/s 33URY9A..33URYH3
0g 0:00:23:06 3/3 0g/s 31991Kp/s 31991Kc/s 31991KC/s 56@GW22..56@GWNP
0g 0:00:23:08 3/3 0g/s 31993Kp/s 31993Kc/s 31993KC/s YFBSO2K..YFBSOE2
0g 0:00:23:09 3/3 0g/s 31991Kp/s 31991Kc/s 31991KC/s 7EYOOTQ..7EYOORB
0g 0:00:24:18 3/3 0g/s 31811Kp/s 31811Kc/s 31811KC/s TSC8HQP..TSC8]ST
0g 0:00:24:19 3/3 0g/s 31808Kp/s 31808Kc/s 31808KC/s DASSN6O..DASSJ6*
0g 0:00:24:22 3/3 0g/s 31814Kp/s 31814Kc/s 31814KC/s KS3LW-0..KS3L7_F
0g 0:00:30:19 3/3 0g/s 31948Kp/s 31948Kc/s 31948KC/s Q7/9Y99..Q7/9Y8P
0g 0:00:30:20 3/3 0g/s 31946Kp/s 31946Kc/s 31946KC/s QV4-B5V..QV4-HEE
0g 0:00:30:21 3/3 0g/s 31948Kp/s 31948Kc/s 31948KC/s Q89TX2Z..Q89TX02
5LYYPM           (sherlock)
1g 0:00:38:25 3/3 0.000433g/s 31924Kp/s 31924Kc/s 31924KC/s 5LY5+!..5LYYP6
Warning: passwords printed above might be partial
Use the "--show" option to display all of the cracked passwords reliably
Session completed
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$ john --show win-pwd.txt
nancydrew:DISENGAGES:1008:1F0607248B96DCE0F1D44BD8AFECCA10:0B6D06B34A0B437D3BE64BC57F89E16C:::
sherlock:5LYYPM:1007:D568A3C648982EE3AAD3B435B51404EE:7F6010574E24B264D81F0225164AEE6C:::

3 password hashes cracked, 0 left
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$
```

```
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$ john windows_passwd_pruned.txt
Loaded 13 password hashes with no different salts (LM [DES 128/128 SSE2])
Press 'q' or Ctrl-C to abort, almost any other key for status
PANACHE          (robert)
COFF33           (andrea)
DENTIST          (carlos)
ABACUS5          (michael)
S0CCR1           (andrew)
DC72B4           (smith)
Z1AYPM           (richard)
7g 0:02:52:01 3/3 0.000678g/s 29599Kp/s 29599Kc/s 179496KC/s UNRZ*N&..UNR AW(
7g 0:02:52:02 3/3 0.000678g/s 29598Kp/s 29598Kc/s 179493KC/s UG0EVA ..UG0EDX#
7g 0:02:52:09 3/3 0.000677g/s 29594Kp/s 29594Kc/s 179465KC/s Q23J71$..Q23J#G,
7g 0:02:52:11 3/3 0.000677g/s 29592Kp/s 29592Kc/s 179455KC/s QJQN7Z$..QJQNYK?
7g 0:02:52:12 3/3 0.000677g/s 29592Kp/s 29592Kc/s 179452KC/s Q53/GS&..Q53/\I,
7g 0:03:03:06 3/3 0.000637g/s 29581Kp/s 29581Kc/s 179273KC/s 1DOX5_$..1DOXJG,
7g 0:03:14:30 3/3 0.000599g/s 29365Kp/s 29365Kc/s 177871KC/s Z,GE48=..Z,GE3*X
7g 0:03:14:31 3/3 0.000599g/s 29364Kp/s 29364Kc/s 177868KC/s UNQMBT/..UNQMVR&
7g 0:03:57:47 3/3 0.000490g/s 29277Kp/s 29277Kc/s 177040KC/s QE6@3D4..QE6@3BP
7g 0:03:57:50 3/3 0.000490g/s 29277Kp/s 29277Kc/s 177041KC/s Q4Y.TIR..Q4Y.TO@
Warning: passwords printed above might be partial
Use the "--show" option to display all of the cracked passwords reliably
Session aborted
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$ john --show windows_passwd_pruned.txt
robert:PANACHE:1011:4F315D32FD63DDE5AAD3B435B51404EE:D6B3D2E9E6D4EE5443996E01BC448CBC:::
andrew:S0CCR1:1012:0BDCCA0EABFF8A8BAAD3B435B51404EE:2A6BB0E4AAA8E933661977EC5EFD02F6:::
carlos:DENTIST:1013:95C475EEF9DAD191AAD3B435B51404EE:4A546A8FD69A1FE0121C00EB7404B773:::
andrea:COFF33:1014:23D75446FB6E0A1CAAD3B435B51404EE:15E837FB855C92A70A1D75B91226E924:::
michael:ABACUS5:1015:3986CFFB5BD9EA86AAD3B435B51404EE:4BE619B5D33FD71EB222AD1DF041B8BC:::
richard:Z1AYPM:1016:D5C0A71E867EF4DAAAD3B435B51404EE:A3D5EAE39100B5276794C11DA366B071:::
smith:DC72B4:1017:3A539909A32DEC39AAD3B435B51404EE:3CF77AE0DFA519BCDD7CFCD64B3FAE16:::

7 password hashes cracked, 6 left
```

By using this technique, we cracked 10 passwords:

From file lin_pwd.txt -

marcopolo: 1OUnMA/nW$XdvYJVquDCgce0cCrBuiJ1 : grimaces

From file win-pwd.txt :

nancydrew: DISENGAGES: 0B6D06B34A0B437D3BE64BC57F89E16C

sherlock: 5LYYPM: 7F6010574E24B264D81F0225164AEE6C

From file windows_passwd_pruned.txt :

robert: PANACHE: D6B3D2E9E6D4EE5443996E01BC448CBC

andrew: S0CCR1: 2A6BB0E4AAA8E933661977EC5EFD02F6

carlos: DENTIST: 4A546A8FD69A1FE0121C00EB7404B773

andrea: COFF33: 15E837FB855C92A70A1D75B91226E924

michael: ABACUS5: 4BE619B5D33FD71EB222AD1DF041B8BC

richard: Z1AYPM: A3D5EAE39100B5276794C11DA366B071

smith: DC72B4: 3CF77AE0DFA519BCDD7CFCD64B3FAE16

Brute-Force Attack:

Dictionary attacks along with word mangling rules are a very strong type of attack, but they have one limitation. They will not be able to crack a password if it's simply a random string, even if it's of a relatively short length. To overcome this, we are using a Brute force attack. We will be using John in the incremental mode. This is the most powerful mode, as it will test all possible combinations but it will take a very long time, we terminate it after it has run for a certain amount of time. But john could not crack any passwords using this technique. The command used in this case was-

john –incremental <filename>

Ex. john –incremental lin_pwd.txt

john –incremental windows_passwd_pruned.txt

```
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$ john --incremental windows_passwd_pruned.txt
Loaded 13 password hashes with no different salts (LM [DES 128/128 SSE2])
Remaining 6 password hashes with no different salts
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:10:30 0g/s 31925Kp/s 31925Kc/s 191554KC/s 9NUMIDG..9NUMIZG
0g 0:00:10:37 0g/s 31935Kp/s 31935Kc/s 191614KC/s PPLZUM3..PPLZUS1
0g 0:00:36:23 0g/s 31212Kp/s 31212Kc/s 187276KC/s PX$IO1S..PX$IOSS
0g 0:01:17:04 0g/s 30758Kp/s 30758Kc/s 184548KC/s OMZ?B8/..OMZ?B5N
0g 0:01:32:17 0g/s 30675Kp/s 30675Kc/s 184054KC/s FHK'=(#..FHK'RQE
0g 0:01:38:03 0g/s 30750Kp/s 30750Kc/s 184501KC/s #B.YMBJ..#B.YMRH
0g 0:01:38:13 0g/s 30754Kp/s 30754Kc/s 184526KC/s CU7W@I6..CU7W@!8
Session aborted
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$
```

Precomputation Attack:

As a brute force attack is usually infeasible against most good hashing algorithms, so in order to overcome this we use the time-memory trade off attack also known as precomputation attacks. Here we precompute a rainbow table and use it to speed up the password cracking process. This attack is executed in three steps-

1. We generate the rainbow table by using rtgen . This creates a table according to the parameters we provide it. The command we used was-

   rtgen lm alpha-numeric 1 6 0 2100 2000000 0

   In the above command we specify that the hashing algorithm used to create the hashes should be the 'LM' algorithm. The character-set used must be 'alpha-numeric'. The password length must be a minimum of 1 and a maximum of 6.
   The rainbow-chain-length should be 2100 and the rainbow chain count must be 2000000. Upon execution this command generates a 2100x2000000 rainbow table for LM hashes.

```
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$ rtgen lm alpha-numeric 1 6 0 2100 2000000 0
rainbow table lm_alpha-numeric#1-6_0_2100x2000000_0.rt parameters
hash algorithm:         lm
hash length:            8
charset:                ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
charset in hex:         41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 30 31 32 33 34 35 36
 38 39
charset length:         36
plaintext length range: 1 - 6
reduce offset:          0x00000000
plaintext total:        2238976116

sequential starting point begin from 0 (0x0000000000000000)
generating...
32768 of 2000000 rainbow chains generated (0 m 34.7 s)
65536 of 2000000 rainbow chains generated (0 m 34.8 s)
98304 of 2000000 rainbow chains generated (0 m 35.0 s)
131072 of 2000000 rainbow chains generated (0 m 36.7 s)
163840 of 2000000 rainbow chains generated (0 m 35.1 s)
196608 of 2000000 rainbow chains generated (0 m 37.4 s)
229376 of 2000000 rainbow chains generated (0 m 34.8 s)
262144 of 2000000 rainbow chains generated (0 m 33.8 s)
294912 of 2000000 rainbow chains generated (0 m 34.2 s)
327680 of 2000000 rainbow chains generated (0 m 33.5 s)
360448 of 2000000 rainbow chains generated (0 m 34.6 s)
393216 of 2000000 rainbow chains generated (0 m 34.3 s)
425984 of 2000000 rainbow chains generated (0 m 33.7 s)
458752 of 2000000 rainbow chains generated (0 m 34.5 s)
491520 of 2000000 rainbow chains generated (0 m 34.1 s)
524288 of 2000000 rainbow chains generated (0 m 34.1 s)
557056 of 2000000 rainbow chains generated (0 m 33.8 s)
589824 of 2000000 rainbow chains generated (0 m 34.0 s)
622592 of 2000000 rainbow chains generated (0 m 33.9 s)
655360 of 2000000 rainbow chains generated (0 m 33.7 s)
688128 of 2000000 rainbow chains generated (0 m 34.0 s)
720896 of 2000000 rainbow chains generated (0 m 34.0 s)
```

```
688128 of 2000000 rainbow chains generated (0 m 34.0 s)
720896 of 2000000 rainbow chains generated (0 m 34.0 s)
753664 of 2000000 rainbow chains generated (0 m 34.0 s)
786432 of 2000000 rainbow chains generated (0 m 35.0 s)
819200 of 2000000 rainbow chains generated (0 m 34.0 s)
851968 of 2000000 rainbow chains generated (0 m 33.9 s)
884736 of 2000000 rainbow chains generated (0 m 33.6 s)
917504 of 2000000 rainbow chains generated (0 m 33.9 s)
950272 of 2000000 rainbow chains generated (0 m 34.2 s)
983040 of 2000000 rainbow chains generated (0 m 34.3 s)
1015808 of 2000000 rainbow chains generated (0 m 33.6 s)
1048576 of 2000000 rainbow chains generated (0 m 34.1 s)
1081344 of 2000000 rainbow chains generated (0 m 33.9 s)
1114112 of 2000000 rainbow chains generated (0 m 33.6 s)
1146880 of 2000000 rainbow chains generated (0 m 33.8 s)
1179648 of 2000000 rainbow chains generated (0 m 33.9 s)
1212416 of 2000000 rainbow chains generated (0 m 33.9 s)
1245184 of 2000000 rainbow chains generated (0 m 33.9 s)
1277952 of 2000000 rainbow chains generated (0 m 34.0 s)
1310720 of 2000000 rainbow chains generated (0 m 33.7 s)
1343488 of 2000000 rainbow chains generated (0 m 34.1 s)
1376256 of 2000000 rainbow chains generated (0 m 34.8 s)
1409024 of 2000000 rainbow chains generated (0 m 34.0 s)
1441792 of 2000000 rainbow chains generated (0 m 33.9 s)
1474560 of 2000000 rainbow chains generated (0 m 34.1 s)
1507328 of 2000000 rainbow chains generated (0 m 34.0 s)
1540096 of 2000000 rainbow chains generated (0 m 34.0 s)
1572864 of 2000000 rainbow chains generated (0 m 34.1 s)
1605632 of 2000000 rainbow chains generated (0 m 34.1 s)
1638400 of 2000000 rainbow chains generated (0 m 33.8 s)
1671168 of 2000000 rainbow chains generated (0 m 34.1 s)
1703936 of 2000000 rainbow chains generated (0 m 34.0 s)
1736704 of 2000000 rainbow chains generated (0 m 33.6 s)
1769472 of 2000000 rainbow chains generated (0 m 35.9 s)
1802240 of 2000000 rainbow chains generated (0 m 34.1 s)
1835008 of 2000000 rainbow chains generated (0 m 34.0 s)
1867776 of 2000000 rainbow chains generated (0 m 33.7 s)
1900544 of 2000000 rainbow chains generated (0 m 33.9 s)
1933312 of 2000000 rainbow chains generated (0 m 33.8 s)
1966080 of 2000000 rainbow chains generated (0 m 34.4 s)
1998848 of 2000000 rainbow chains generated (0 m 33.6 s)
2000000 of 2000000 rainbow chains generated (0 m 1.2 s)
```

2. Now we need to sort the table we created, this will make our cracking process faster. We can do this by using rtsort. The command used for sorting the table was-

   rtsort lm_alpha-numeric#1-6_0_2100x2000000_0.rt

```
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$ rtsort lm_alpha-numeric#1-6_0_2100x2000000_0.rt
lm_alpha-numeric#1-6_0_2100x2000000_0.rt:
722411520 bytes memory available
loading rainbow table...
sorting rainbow table by end point...
writing sorted rainbow table...

team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$
```

1. Before we start the cracking process we need to create a file which will contain only the hashes. Where each line is one hash. We need to give it as input to the rtcrack command. In order to create this file of hashes we use the awk command and the fact that the hash files were delimited by a ':' .
   In the windows hash file, retrieved using pwdump the fourth column of hashes is the LM hash and the first is the NTLM hash.

   We need to grab the LM hashes into a text file where each line is a hash.

   We will call this file windows-hashes-in-order.txt

   command: used to return the 4th column of the file
   awk -F ':' '{print $4}' windows_passwd_pruned.txt > windows-hashes-in-order.txt

   command to append to the list of hashes with hashes from the next file.
   awk -F ':' '{print $4}' win-pwd.txt >> windows-hashes-in-order.txt

```
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$ awk -F ':' '{print $4}' win-pwd.txt >> windows-hashes-in-or
er.txt
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$ cat win-pwd.txt
nancydrew:1008:1F0607248B96DCE0F1D44BD8AFECCA10:0B6D06B34A0B437D3BE64BC57F89E16C:::
sherlock:1007:D568A3C648982EE3AAD3B435B51404EE:7F6010574E24B264D81F0225164AEE6C:::
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$ cat windows-hashes-in-order.txt
848F4B13A7CB0568A8F4F496E9768066
74D0C3DF46E1D9D35229F74936490FA5
D6B3D2E9E6D4EE5443996E01BC448CBC
2A6BB0E4AAA8E933661977EC5EFD02F6
4A546A8FD69A1FE0121C00EB7404B773
15E837FB855C92A70A1D75B91226E924
4BE619B5D33FD71EB222AD1DF041B8BC
A3D5EAE39100B5276794C11DA366B071
3CF77AE0DFA519BCDD7CFCD64B3FAE16
0E01EE5B464AEEE66A26CD5EBDF8E658
0B6D06B34A0B437D3BE64BC57F89E16C
7F6010574E24B264D81F0225164AEE6C
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$
```

2. Now we use rcrack, to start the password cracking process. We provide the sorted rainbow table created in step 2 and the file containing the hashes created in step 3 as input to rcrack. Command used to start the cracking process-

   rcrack lm_alpha-numeric#1-6_0_2100x2000000_0.rt windows-hashes-in-order.txt

```
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$ rcrack lm_alpha-numeric#1-6_0_2100x2000000_0.rt -l windows-h
ashes-in-order.txt
818889523 bytes memory available
1 x 32000000 bytes memory allocated for table buffer
403200 bytes memory allocated for chain traverse
disk: lm_alpha-numeric#1-6_0_2100x2000000_0.rt: 32000000 bytes read
searching for 0 hash...
disk: finished reading all files

statistics
----------------------------------------------------------
plaintext found:                        0 of 12
total time:                             0.03 s
  time of chain traverse:               0.00 s
  time of alarm check:                  0.00 s
  time of wait:                         0.03 s
  time of other operation:              0.00 s
time of disk read:                      0.01 s
hash & reduce calculation of chain traverse:  0
hash & reduce calculation of alarm check:     0
number of alarm:                        0
speed of chain traverse:                0.00 million/s
speed of alarm check:                   0.00 million/s

result
----------------------------------------------------------
848f4b13a7cb0568a8f4f496e9768066  <not found>  hex:<not found>
74d0c3df46e1d9d35229f74936490fa5  <not found>  hex:<not found>
d6b3d2e9e6d4ee5443996e01bc448cbc  <not found>  hex:<not found>
2a6bb0e4aaa8e933661977ec5efd02f6  <not found>  hex:<not found>
4a546a8fd69a1fe0121c00eb7404b773  <not found>  hex:<not found>
15e837fb855c92a70a1d75b91226e924  <not found>  hex:<not found>
4be619b5d33fd71eb222ad1df041b8bc  <not found>  hex:<not found>
a3d5eae39100b5276794c11da366b071  <not found>  hex:<not found>
3cf77ae0dfa519bcdd7cfcd64b3fae16  <not found>  hex:<not found>
0e01ee5b464aeee66a26cd5ebdf8e658  <not found>  hex:<not found>
0b6d06b34a0b437d3be64bc57f89e16c  <not found>  hex:<not found>
7f6010574e24b264d81f0225164aee6c  <not found>  hex:<not found>
team@nslabu:~/Desktop/password-cracking-lab/Pruned_passwd_files$
```

3. Name at least three reasons why LM hashes are easier to crack than salted SHA-1 hashes.

a. Although LM hash is based on DES, it truly not a one-way function as the password can be determined from the hash. A brute force attack on each half can crack the password easily.

b. LM hash does not use crypto salt which prevents pre-computed dictionary attacks.

c. LM hashes change only when the password is changed and hence are susceptible to pass the hash attack.

d. If password is greater than 14 characters, the LM hash appears the same as for an empty password.

e. During encryption, all characters of password are converted to upper case.

4. LM hashes are disabled by default in Windows Server 2012. However, many administrators enable it on their servers. In previous versions, it was enabled by default. Why are LM hashes still required?

LanMan (LM) Hash is a legacy hash technique which was used by older systems like Windows 95, Windows 98 and Windows ME to hash a saved password.

Newer Windows Server and machines that need access to such systems and older architecture maintain LM hashes for backward compatibility. For all Windows versions post Windows NT 6, a new hash technique NT is used.

5. **Suppose a user selects a random 8 character password from the set of characters [A-Za-z0-9]. The password is stored as an unsalted SHA-1 hash. If an attacker wishes to precompute all possible 8 character password hashes for this character set and store the pairs in a simple list, how many megabytes of disk space would this require at a minimum?**

   8-character password with combination of [a-z] (26 characters) [A-Z] (26 characters) AND [0-9] (10 characters).

   62 ^ 8 = 218340105584896 (Combo of all three)

   36 ^ 8 = 2821109907456 (Numeric and lower) / (Numeric and upper)

   52 ^ 8 = 53459728531456 (Upper and lower)

   26 ^ 8 = 208827064576 (Only upper)= (Only lower)

   10 * 8 = 100000000 (Only numeric)

   Possible combinations = 62 ^ 8 - 36 ^ 8 - 36 ^ 8 - 52 ^ 8 + 10 * 8 + 26 ^ 8 + 26 ^ 8 = **1.59x10^14** key space

   According to rainbow table, size for mix alpha-numeric rainbow tables for 8-bit characters is of size between **127-160GB**.