CySA+ Lab Series

Lab 13: Password Cracking with Cain and Abel, Hashcat, and John the Ripper

Document Version: 2022-10-10

Material in this Lab Aligns to the Following	
CompTIA CySA+ (CS0-002) Exam Objectives	1.4 - Given a scenario, analyze the output from common vulnerability tools 2.1 - Explain software assurance best practices 4.4 - Given a scenario, utilize basic digital forensics techniques 5.3 - Explain the importance of frameworks, policies, procedures and controls
All-In-One CompTIA CySA+ Second Edition ISBN-13: 978-1260464306 Chapters	4: Vulnerability Assessment Tools 8: Security Solutions for Infrastructure Management 18: Utilize Basic Digital Forensics Techniques 21: The Importance of Frameworks, Policies, Procedures, and Controls

Contents

Introduction

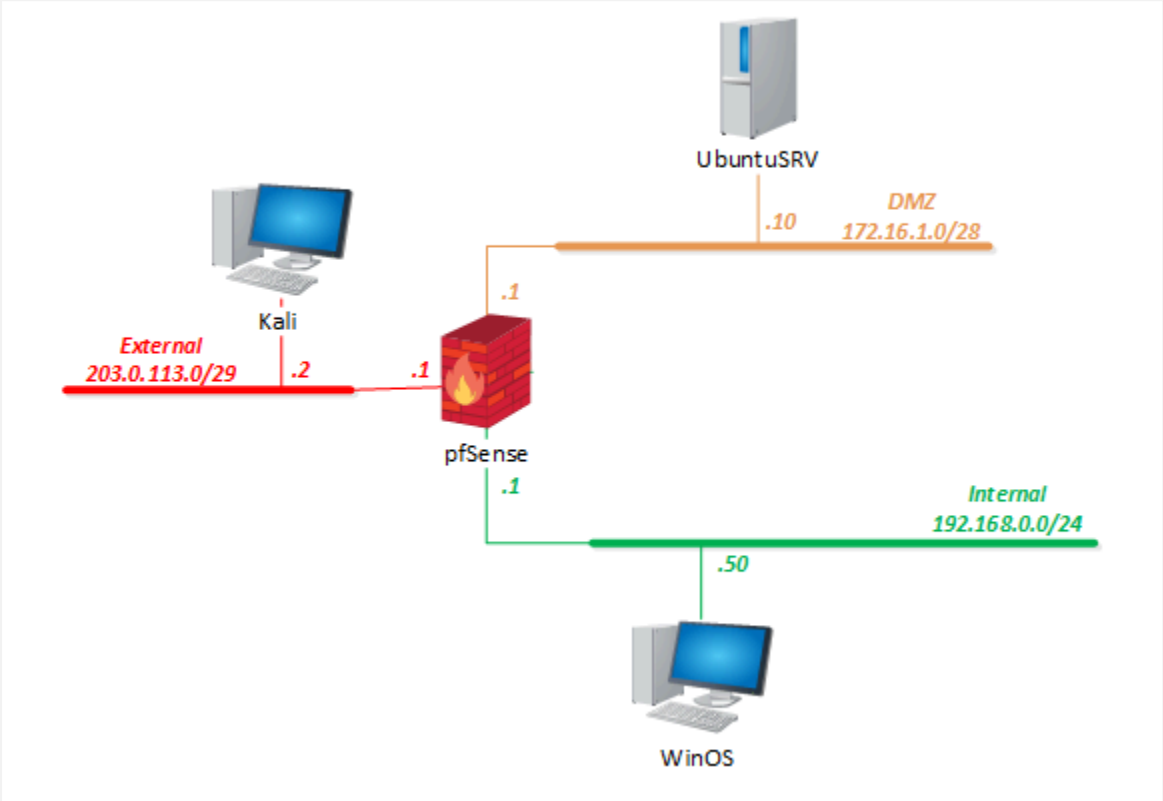
Hashing is a way to store a password in a secure non-plaintext format. The way you crack (or guess) a password that has been stored as a hash is to figure out the algorithm originally used to hash it and then use that algorithm to hash plain text passwords of your own until the two hashes match.

In this lab, you will utilize Cain and Able, John the Ripper, and Hashcat to crack some simple passwords.

Objectives

- Explore the options and capabilities of the Cain and Able, John the Ripper, and Hashcat tools
- Learn what masking is and how to utilize it
- Learn how to append/prepend masks to word lists
- Explore and utilize these various rules to crack Linux hashes

Lab Topology



Lab Settings

The information in the table below will be needed in order to complete the lab. The task sections below provide details on the use of this information.

Virtual Machine	IP Address	Account	Password
WinOS (Server 2019)	192.168.0.50	Administrator	NDGlabpass123!
MintOS (Linux Mint)	192.168.0.60	sysadmin	NDGlabpass123!
OSSIM (AlienVault)	172.16.1.2	root	NDGlabpass123!
UbuntuSRV (Ubuntu Server)	172.16.1.10	sysadmin	NDGlabpass123!
Kali	203.0.113.2	sysadmin	NDGlabpass123!
pfSense	203.0.113.1 172.16.1.1 192.168.0.1	admin	NDGlabpass123!

1 Generating Word Lists for Password Cracking

An effective method for cracking user passwords uses a *Dictionary Attack* which is a method where a hacker systematically enters dictionary words until the password is discovered. This type of attack is effective when people do not create strong passwords. A wordlist is a list of possible passwords that can be used for dictionary attacks.

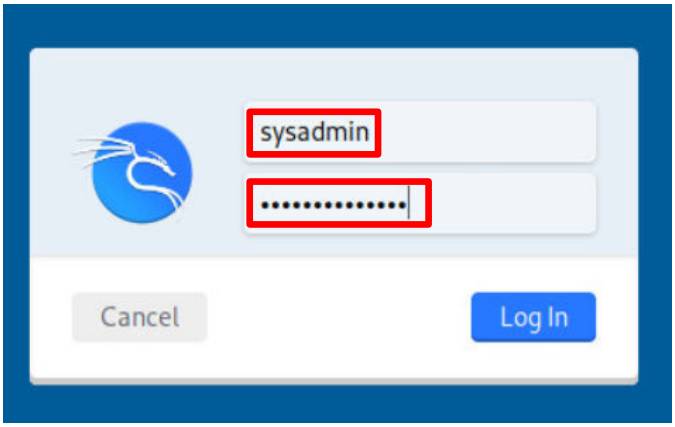
According to the UK's National Cyber Security Center, the most common guessable password was **123456** with 23.2 million accounts. **qwerty** accounted for 7.7 million users.

https://auth0.com/blog/dont-pass-on-the-new-nist-password-guidelines/

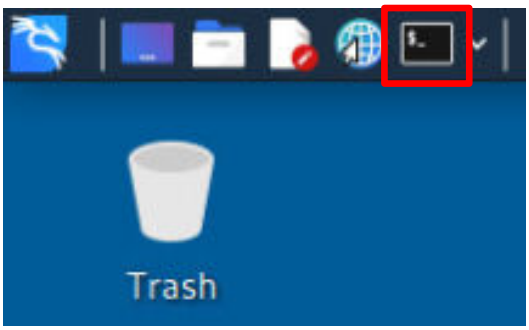
According to NIST Special Publication 800-63B-3, longer passwords, with complexity and a minimum size of 8 characters, but most security professionals are now recommending a minimum of 12 and some recommend 15 characters.

In the first part of this lab, you will be creating a wordlist that can be used to crack passwords.

1. Set the focus on the **Kali** computer.
2. Log in as `sysadmin` using the password: `NDGlabpass123!`



3. Click the **Terminal** icon to open a terminal window.



4. In the terminal window, observe the options available for the *cewl* tool. This tool is used for gathering text from a website. Type the command below, followed by pressing the **Enter** key.

```
cewl -help
```

```
sysadmin@kali: ~
File  Actions  Edit  View  Help
└─$ cewl -help
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
Usage: cewl [OPTIONS] ... <url>

    OPTIONS:
        -h, --help: Show help.
        -k, --keep: Keep the downloaded file.
        -d <x>,--depth <x>: Depth to spider to, default 2.
        -m, --min_word_length: Minimum word length, default 3.
        -o, --offsite: Let the spider visit other sites.
        --exclude: A file containing a list of paths to exclude
        --allowed: A regex pattern that path must match to be followed
        -w, --write: Write the output to the file.
        -u, --ua <agent>: User agent to send.
        -n, --no-words: Don't output the wordlist.
        -g <x>, --groups <x>: Return groups of words as well
        --lowercase: Lowercase all parsed words
        --with-numbers: Accept words with numbers in as well as just letters
        --convert-umlauts: Convert common ISO-8859-1 (Latin-1) umlauts (ä-ae, ö-oe, ü-ue, ß-ss)
        -a, --meta: include meta data.
        --meta_file file: Output file for meta data.
        -e, --email: Include email addresses.
        --email_file <file>: Output file for email addresses.
        --meta-temp-dir <dir>: The temporary directory used by exiftool when parsing files, default /tmp.
        -c, --count: Show the count for each word found.
        -v, --verbose: Verbose.
        --debug: Extra debug information.

        Authentication
        --auth_type: Digest or basic.
        --auth_user: Authentication username.
        --auth_pass: Authentication password.

        Proxy Support
        --proxy_host: Proxy host.
        --proxy_port: Proxy port, default 8080.
        --proxy_username: Username for proxy, if required.
        --proxy_password: Password for proxy, if required.

        Headers
        --header, -H: In format name:value - can pass multiple.
```

5. Target the **UbuntuSRV** machine using the *cewl* tool and copy the text to a file named *pwords.txt*. Type the command below:

```
cewl -w pwords.txt -d 2 -m 5 172.16.1.10
```

```
┌──(sysadmin㉿kali)-[~]
└─$ cewl -w pwords.txt -d 2 -m 5 172.16.1.10
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
```


6. View the contents of the wordlist by typing the following command:

```
cat pwords.txt
```

```
┌──(sysadmin㉿kali)-[~]
└─$ cat pwords.txt
Request
server
browser
request
could
understand
Reason
speaking
plain
enabled
Instead
HTTPS
scheme
access
please
Apache
Ubuntu
Server
```

7. Another tool for creating wordlists is *crunch*. Type the following command to show a description of the *crunch* utility:

```
crunch
```

```
┌──(sysadmin㉿kali)-[~]
└─$ crunch
crunch version 3.6

Crunch can create a wordlist based on criteria you specify.  The output from cru
nch can be sent to the screen, file, or to another program.

Usage: crunch <min> <max> [options]
where min and max are numbers

Please refer to the man page for instructions and examples on how to use crunch.
```


8. Using *crunch*, generate a set of passwords that are a minimum of **4 characters** and a maximum of **8 characters** in length, made up of all the letters of the alphabet in lowercase only, and save the passwords in a file named **list.txt**. Type the command below:

```
crunch 4 8 charset.lst lalpha –o list.txt
```

```
┌──(sysadmin㉿kali)-[~]
└─$ crunch 4 8 charset.lst lappha -o list.txt
Crunch will now generate the following amount of data: 429791427 bytes
409 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 48426741

crunch: 100% completed generating output
```

9. View the contents of the wordlist by typing the following command:

```
cat list.txt | more
```

```
File  Actions  Edit  View  Help
cccc
ccch
ccca
cccr
cccs
ccce
ccct
ccc.
cccl
cchc
cchh
ccha
cchr
cchs
cche
ccht
cch.
cchl
ccac
ccah
ccaa
ccar
ccas
ccae
ccat
cca.
ccal
ccrc
ccrh
ccra
ccrr
ccrs
ccre
ccrt
--More--
```

10. Press `Q` to exit the cat list. (There are more than 480,000 entries on the list.)

11. *Kali Linux* also has various wordlists that can be used with the dictionary attack tools that are included in the distro. To see the various wordlists, navigate to the **/usr/share/wordlists** directory by typing the following command:

```
cd /usr/share/wordlists
```

```
┌──(sysadmin㉿kali)-[~]
└─$ cd /usr/share/wordlists
```

12. Type the following command to show the directories that contain various wordlists:

```
ls -l
```

```
┌──(sysadmin㉿kali)-[/usr/share/wordlists]
└─$ ls -l
total 52108
lrwxrwxrwx 1 root root       25 Jul 29 16:45 dirb -> /usr/share/dirb/wordlists
lrwxrwxrwx 1 root root       30 Jul 29 16:45 dirbuster -> /usr/share/dirbuster/wordlists
lrwxrwxrwx 1 root root       41 Jul 29 16:45 fasttrack.txt -> /usr/share/set/src/fasttrack
lrwxrwxrwx 1 root root       45 Jul 29 16:45 fern-wifi -> /usr/share/fern-wifi-cracker/ext
lrwxrwxrwx 1 root root       46 Jul 29 16:45 metasploit -> /usr/share/metasploit-framework
lrwxrwxrwx 1 root root       41 Jul 29 16:45 nmap.lst -> /usr/share/nmap/nselib/data/passw
-rw-r--r-- 1 root root 53357329 Jul 17  2019 rockyou.txt.gz
lrwxrwxrwx 1 root root       25 Jul 29 16:45 wfuzz -> /usr/share/wfuzz/wordlist
```

"There's the **dirb** directory for the wordlists to be used while using the *dirb* tool to perform **Directory Bruteforce**. Then we have the **dirbuster** that is a similar tool that also performs **Directory Bruteforce** but with some additional options. Then we have a *fern-wifi* directory which helps to break the Wi-Fi Authentications. Then we have the **Metasploit** which uses wordlists for almost everything. Then there is a **nmap** wordlist that contains that can be used while scanning some specific services. Then we have the Rockstar of Wordlists: **rockyou**. This is compressed by default and you will have to extract it before using it. It is very large with 14,442,062 values that could be passwords for a lot of user accounts on the internet. At last, we have the **wfuzz** directory that has the wordlists that can be used clubbed with *wfuzz*."

https://www.hackingarticles.in/wordlists-for-pentester/

You can also download many wordlists from the internet.

13. Return to the *sysadmin* home directory by typing in the following command:

```
cd ~
```

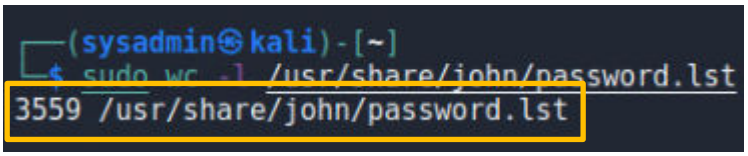
14. Since there is only one password per line, use the following command to see that 18 passwords are in the pwords.txt file that was culled from a website using *cewl*:

    If asked for the **sudo** password, type: `NDGlabpass123!`

```
sudo wc -l pwords.txt
```

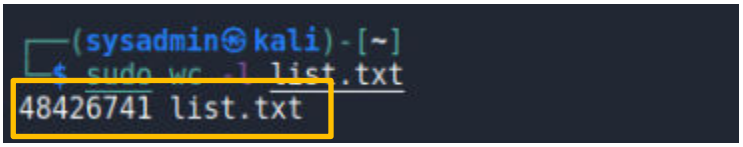
15. Get the count of passwords from the default list provided with the *John the Ripper* password cracking program:

```
sudo wc -l /usr/share/john/password.lst
```



16. Finally, get the password count the *crunch* program:

```
sudo wc -l list.txt
```



    The more passwords there are in the list, the more likely a password match will be found; however, it takes significantly more time to process.

17. Leave the terminal window in *Kali* open and proceed to the next task.

2 Create User Accounts to be Cracked

1. Create the user account, **jkirk,** with the password **123456** using a **SHA-512** hash by typing the following command:

```
sudo useradd –m –p $(mkpasswd –m sha-512 “123456”) –s /bin/bash jkirk
```

If asked for the **sudo** password, type: `NDGlabpass123!`

```
┌──(sysadmin㉿kali)-[~]
└─$ sudo useradd -m -p $(mkpasswd -m sha-512 "123456") -s /bin/bash jkirk
```

2. Create another user account, **mrspock** with the password **password** using a **SHA-512** hash by typing the following command:

```
sudo useradd –m –p $(mkpasswd –m sha-512 “password”) –s /bin/bash mrspock
```

If asked for the **sudo** password, type: `NDGlabpass123!`

```
┌──(sysadmin㉿kali)-[~]
└─$ sudo useradd -m -p $(mkpasswd -m sha-512 "password") -s /bin/bash mrspock
```

The reason that the **useradd** command is accompanied by the **mkpasswd** command and using **SHA-512** hash is that by default the latest versions of Linux use the **yescrypt** hash method which is not supported by *Hashcat*.

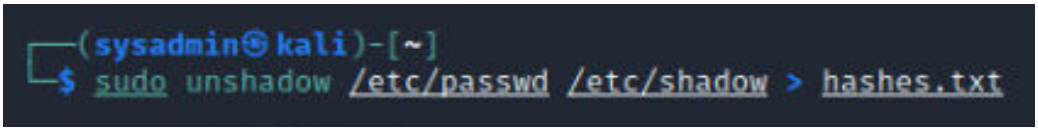
3. Linux systems store usernames and passwords in the */etc/shadow* file that can only be accessed by the root user. The passwords are stored as a salted-hash. Confirm that the new users have passwords assigned to them by checking the **/etc/shadow** file. Type the following command:

```
sudo cat /etc/shadow
```

```
jkirk:$6$XfuCBpIN49XCCce9$20KsmCr6vC3qBJT3IhTPXeHx8iKToZ
c.:18976:0:99999:7:::
mrspock:$6$YCbdc6hUulHNDMeD$7YZpMLWScPbgUMQha6yORxRJ0NLb
6Uu.:18976:0:99999:7:::
```

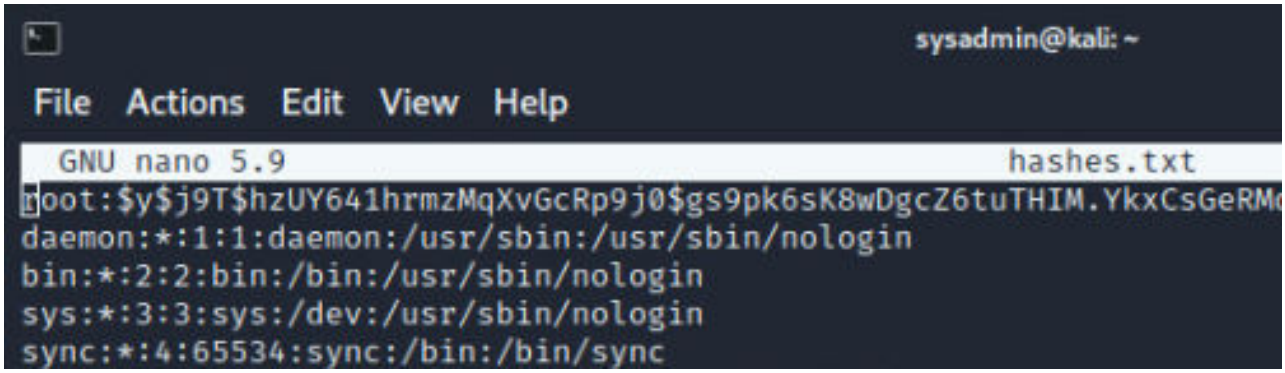

4. Combine the two files that make updates to the user's credentials; both **/etc/passwd** and **/etc/shadow,** by typing the following command:

```
sudo unshadow /etc/passwd /etc/shadow > hashes.txt
```



5. Use *nano* to edit the **hashes.txt** file so that it is narrowed down to only the two accounts that need to be cracked.

```
sudo nano hashes.txt
```



6. Hold down the **SHIFT** key and press the **down arrow** key to select (highlight) all of the accounts up to **jkirk,** as shown below:

```
File  Actions  Edit  View  Help
  GNU nano 5.9                                        hashes.txt
ntp:*:107:112::/nonexistent:/usr/sbin/nologin
messagebus:*:108:113::/nonexistent:/usr/sbin/nologin
redsocks:!:109:114::/var/run/redsocks:/usr/sbin/nologin
rwhod:*:110:65534::/var/spool/rwho:/usr/sbin/nologin
iodine:*:111:65534::/run/iodine:/usr/sbin/nologin
miredo:*:112:65534::/var/run/miredo:/usr/sbin/nologin
_rpc:*:113:65534::/run/rpcbind:/usr/sbin/nologin
usbmux:*:114:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
tcpdump:*:115:121::/nonexistent:/usr/sbin/nologin
rtkit:*:116:122:RealtimeKit,,,:/proc:/usr/sbin/nologin
sshd:*:117:65534::/run/sshd:/usr/sbin/nologin
statd:*:118:65534::/var/lib/nfs:/usr/sbin/nologin
postgres:*:119:124:PostgreSQL administrator,,,:/var/lib/postgresql:/b
avahi:*:120:126:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nolo
stunnel4:!:121:127::/var/run/stunnel4:/usr/sbin/nologin
Debian-snmp:!:122:128::/var/lib/snmp:/bin/false
speech-dispatcher:!:123:29:Speech Dispatcher,,,:/run/speech-dispatche
sslh:!:124:129::/nonexistent:/usr/sbin/nologin
nm-openvpn:*:125:130:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroo
nm-openconnect:*:126:131:NetworkManager OpenConnect plugin,,,:/var/li
pulse:*:127:132:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin
saned:*:128:135::/var/lib/saned:/usr/sbin/nologin
inetsim:*:129:137::/var/lib/inetsim:/usr/sbin/nologin
colord:*:130:138:colord colour management daemon,,,:/var/lib/colord:/
geoclue:*:131:139::/var/lib/geoclue:/usr/sbin/nologin
lightdm:*:132:140:Light Display Manager:/var/lib/lightdm:/bin/false
king-phisher:*:133:141::/var/lib/king-phisher:/usr/sbin/nologin
sysadmin:$y$j9T$BdMyYYv7tHml7RC/FFtgp0$QPvCx6RgCa37WDoE9uqfonWSG/tNWY
systemd-coredump:!*:999:999:systemd Core Dumper:/:/usr/sbin/nologin
scotty:$6$5hkeS2k6ScKfbVsc$/MR1HZxvq/eRe6HYR/POfBwtKFMO64jU9Yqi50EQ9R
jkirk:$6$XfuCBpIN49XCCce9$20KsmCr6vC3qBJT3IhTPXeHx8iKToZFb8XUOlGganU0
mrspock:$6$YCbdc6hUulHNDMeD$7YZpMLWScPbgUMQha6yORxRJ0NLbqs.x0WvRdjyKW
```

7. Press **CTRL+K** to cut the selected text. You should only be left with the two new accounts.

```
File  Actions  Edit  View  Help
  GNU nano 5.9                                        hashes.txt *
jkirk:$6$XfuCBpIN49XCCce9$20KsmCr6vC3qBJT3IhTPXeHx8iKToZFb8XUOlGganU0
mrspock:$6$YCbdc6hUulHNDMeD$7YZpMLWScPbgUMQha6yORxRJ0NLbqs.x0WvRdjyKW
```

8. Press **CTRL+O** and press **Enter** to save the file, and then press **CTRL+X** to exit the *nano* editor.

9. View the contents of the **hashes.txt** file.

```
cat hashes.txt
```

```
┌──(sysadmin㉿kali)-[~]
└─$ cat hashes.txt
jkirk:$6$XfuCBpIN49XCCce9$20KsmCr6vC3qBJT3IhTPXeHx8iKToZFb8XUOl
c.:1004:1004::/home/jkirk:/bin/bash
mrspock:$6$YCbdc6hUulHNDMeD$7YZpMLWScPbgUMQha6yORxRJ0NLbqs.x0Wv
6Uu.:1005:1005::/home/mrspock:/bin/bash
```

10. Remain on the *Kali* computer with the terminal window open and continue to the next task.

3 Crack Passwords with John the Ripper

One of the easiest ways to crack a password that has been stored as a hash is to compare the hash of the password against a list of well-known passwords that have been hashed using the same algorithm (such as **SHA-512**). The trick is to get a long list of possible words to create the hash lists from and, of course, a user who uses a password that would be on such a list.

One of the tasks that a security administrator should do is to double-check to make sure that users have passwords that are not found in typical wordlists.

In the next three sections, you will use different tools and utilities to discover passwords that are easy to compromise using hashes found in wordlists.

John the Ripper is an Open Source tool that was built for password security auditing and for password recovery. It was originally developed for Unix, but it has since been ported to other platforms, including Linux, Windows, and macOS. It is an extremely popular tool used by security analysts to determine what password strength should be utilized. You will use John the Ripper to find the user's passwords that were set in the previous task.

1. *John the Ripper* uses the CLI. Use the following command to see the syntax and the options that are available

```
john --help
```

```
┌──(sysadmin㉿kali)-[~]
└─$ john --help
John the Ripper 1.9.0-jumbo-1+bleeding-aec1328d6c 2021-11-02 10:45:52 +0100 OMP
Copyright (c) 1996-2021 by Solar Designer and others
Homepage: https://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]

--help                      Print usage summary
--single[=SECTION[,..]]     "Single crack" mode, using default or named rules
--single=:rule[,..]         Same, using "immediate" rule(s)
--single-seed=WORD[,WORD]   Add static seed word(s) for all salts in single mode
--single-wordlist=FILE      *Short* wordlist with static seed words/morphemes
--single-user-seed=FILE     Wordlist with seeds per username (user:password[s]
                            format)
--single-pair-max=N         Override max. number of word pairs generated (6)
--no-single-pair            Disable single word pair generation
--[no-]single-retest-guess  Override config for SingleRetestGuess
--wordlist[=FILE] --stdin   Wordlist mode, read words from FILE or stdin
                  --pipe    like --stdin, but bulk reads, and allows rules
--rules[=SECTION[,..]]      Enable word mangling rules (for wordlist or PRINCE
                            modes), using default or named rules
--rules=:rule[;..]]         Same, using "immediate" rule(s)
--rules-stack=SECTION[,..]  Stacked rules, applied after regular rules or to
                            modes that otherwise don't support rules
--rules-stack=:rule[;..]    Same, using "immediate" rule(s)
--rules-skip-nop            Skip any NOP ":" rules (you already ran w/o rules)
--loopback[=FILE]           Like --wordlist, but extract words from a .pot file
--mem-file-size=SIZE        Size threshold for wordlist preload (default 2048 MB)
--dupe-suppression          Suppress all dupes in wordlist (and force preload)
```


2. Use the **pwords.txt** list to try to crack the passwords in the **hashes.txt** file.

```
sudo john –format=crypt –wordlist=pwords.txt hashes.txt
```

If asked for the **sudo** password, type: `NDGlabpass123!`

```
┌──(sysadmin㉿kali)-[~]
└─$ sudo john --format=crypt --wordlist=pwords.txt  hashes.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with 4 different salts (crypt, generic crypt(3) [?/64]
)
Cost 1 (algorithm [1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha5
12crypt]) is 0 for all loaded hashes
Cost 2 (algorithm specific iterations) is 1 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 18 candidates left, minimum 96 needed for performance.
0g 0:00:00:00 DONE (2021-12-06 13:50) 0g/s 30.50p/s 122.0c/s 122.0C/s Request..
Server
Session completed
```

Since the password list that *cewl* generated from the website is very small, it does not have the passwords that were created for the two users, as common as they may be.

3. Use *John the Ripper's* default password list to try to crack the passwords in the **hashes.txt** file.

```
sudo john –wordlist=/usr/share/john/password.lst hashes.txt
```

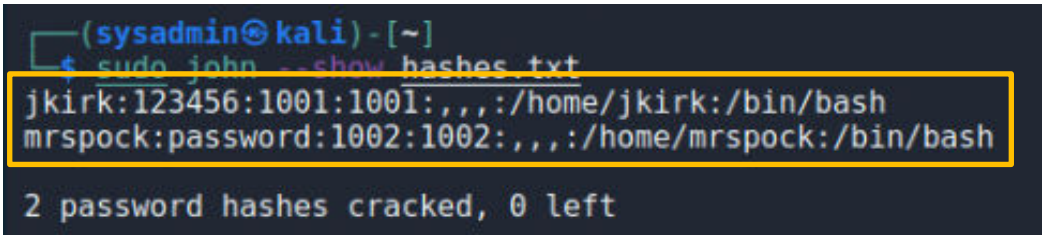
```
┌──(sysadmin㉿kali)-[~]
└─$ sudo john -wordlist=/usr/share/john/password.lst hashes.txt
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (sha512crypt, crypt(3) $6$ [SHA512
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Press Ctrl-C to abort, or send SIGUSR1 to john process for status
123456           (jkirk)
password         (mrspock)
2g 0:00:00:00 DONE (2021-12-15 02:19) 8.333g/s 1066p/s 2133c/s 2133C/s 123456..fran
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

As you can see, *John the Ripper* has discovered the passwords for *jkirk* and *mrspock*.

You can use any wordlist with *John the Ripper*. Be aware the larger the wordlist, the longer the process will take. The *crunch* wordlist, **list.txt** is over 48 million lines long. It will take a **<u>considerable</u>** amount of time (7-8 days) to process.

4. *John the Ripper* saves cracked passwords in a file called **/root/.john/john.pot**. To view the cracked passwords, type the following command:

```
sudo john –show hashes.txt
```



If you run *John the Ripper* again, the passwords that have already been cracked will not show up in the output. To force it to crack passwords every time, you must delete the **john.pot** file. Use this command to remove the file:

```
sudo rm /root/.john/john.pot
```

5. Leave the terminal window in *Kali* open and proceed to the next task.

4 Crack Passwords with Hashcat

Hashcat is a powerful password-cracking tool that uses an OpenCL library. This allows the tool to use not only the CPU, but GPUs (graphics cards), and other compatible OpenCL hardware. In this section, you will use only the CPU.

1. Type the command below to become familiar with *hashcat* and the options available.

```
sudo Desktop/LabFiles/hashcat-6.1.1/hashcat.bin -h | more
```

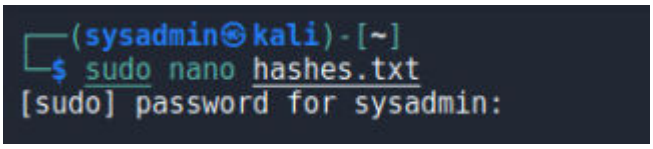
```
┌──(sysadmin㉿kali)-[~]
└─$ sudo Desktop/LabFiles/hashcat-6.1.1/hashcat.bin -h |more
hashcat (v6.1.1) starting ...

Usage: hashcat [options] ... hash|hashfile|hccapxfile [dictionary|mask|directory] ...

- [ Options ] -

 Options Short / Long          | Type | Description
=============================+======+=============================================
 -m, --hash-type              | Num  | Hash-type, see references below
 -a, --attack-mode            | Num  | Attack-mode, see references below
 -V, --version                |      | Print version
 -h, --help                   |      | Print help
     --quiet                  |      | Suppress output
     --hex-charset            |      | Assume charset is given in hex
     --hex-salt               |      | Assume salt is given in hex
     --hex-wordlist           |      | Assume words in wordlist are given in hex
     --force                  |      | Ignore warnings
     --status                 |      | Enable automatic update of the status screen
     --status-json            |      | Enable JSON format for status ouput
     --status-timer           | Num  | Sets seconds between status screen updates to X
     --stdin-timeout-abort    | Num  | Abort if there is no input from stdin for X seconds
     --machine-readable       |      | Display the status view in a machine-readable format
     --keep-guessing          |      | Keep guessing the hash after it has been cracked
     --self-test-disable      |      | Disable self-test functionality on startup
     --loopback               |      | Add new plains to induct directory
     --markov-hcstat2         | File | Specify hcstat2 file to use
     --markov-disable         |      | Disables markov-chains, emulates classic brute-force
     --markov-classic         |      | Enables classic markov-chains, no per-position
 -t, --markov-threshold       | Num  | Threshold X when to stop accepting new markov-chains
```

Press the **Spacebar** to skip to the next page or the **Enter** key to skip by each line. Press **Q** to quit the help screen at any given time.

2. Use the *nano* editor to make some changes to the **hashes.txt** file.

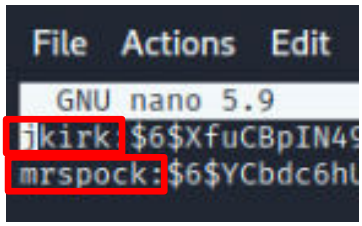
```
sudo nano hashes.txt
```

If asked for the **sudo** password, type: **NDGlabpass123!**

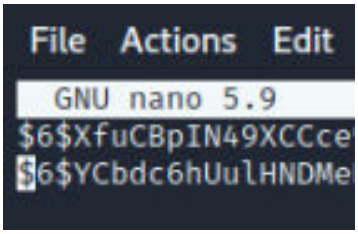


3. Remove the account names, **jkirk** and **mrspock**, along with the colon (**:**), as shown below.

Before:

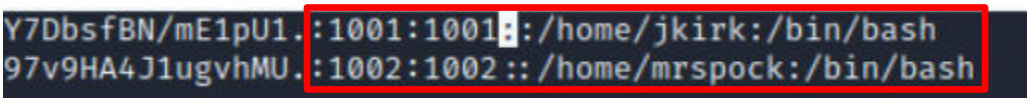


After:



4. At the end of each line, you should see additional user information (the **UserID**, **GroupID,** and **home bash directory**), which needs to be deleted as well. Press the **END** key and delete everything back to, and including, the colon (**:**) preceding the **UserID** number.

Before:



After:

11. Press **CTRL+O** and change the name to `hashes2.txt`**,** and press **Enter**. Press **Y** to save the file under a different name, then press **CTRL+X** to exit the *nano* editor.
12. Type the command below to use *hashcat* in an attempt to password crack the two usernames in the hashes2.txt file using the wordlist provided for this lab.

```
sudo Desktop/LabFiles/hashcat-6.1.1/hashcat.bin --force -m 1800 -a 0 hashes2.txt
Desktop/LabFiles/HashCat/password.lst
```

```
┌──(sysadmin㉿kali)-[~]
└─$ hashcat --force -m 1800 -a 0 hashes2.txt Desktop/LabFiles/HashCat/password.lst
hashcat (v6.1.1) starting ...

You have enabled --force to bypass dangerous warnings and errors!
This can hide serious problems and should only be done when debugging.
Do not report hashcat issues encountered when using --force.
OpenCL API (OpenCL 2.1 LINUX) - Platform #1 [Intel(R) Corporation]
=================================================================================
* Device #1: Intel(R) Xeon(R) D-2146NT CPU @ 2.30GHz, 1917/1981 MB (495 MB allocatab
le), 2MCU

OpenCL API (OpenCL 2.0 pocl 1.8  Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, DIS
TRO, POCL_DEBUG) - Platform #2 [The pocl project]
=================================================================================
=================================================================================
* Device #2: pthread-Intel(R) Xeon(R) D-2146NT CPU @ 2.30GHz, skipped

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 2 digests; 2 unique digests, 2 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Applicable optimizers applied:
* Zero-Byte
* Uses-64-Bit
```

Explanation of the *hashcat* command with these options:

- **-m 1800** - SHA-512 (type 6) password hashes
- **-a 0** - Use a dictionary attack
- **hashes2.txt** – the password shadow hash file
- **Desktop/LabFiles/HashCat/password.lst** – the wordlist to be used

The program will compile the list and then begin cracking through the iterations. When passwords are cracked, they will be displayed at the end of the hash, as shown below:

```
Dictionary cache hit:
* Filename..: Desktop/LabFiles/HashCat/password.lst
* Passwords.: 3559
* Bytes.....: 26325
* Keyspace..: 3559

$6$XfuCBpIN49XCCce9$20KsmCr6vC3qBJT3IhTPXeHx8iKToZFb8XUOlGganU0CY9c5artjAKz
NPbe0TlEAykWlzKmoy5Sc.:123456
$6$YCbdc6hUulHNDMeD$7YZpMLWSc2bgUMQha6yORxRJ0NLbqs.x0WvRdjyKWtcYy0fgINFDWIT
wgI.lgsTgdAdjdDexD6Uu.:password

Session..........: hashcat
Status...........: Cracked
Hash.Name........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: hashes2.txt
Time.Started.....: Wed Dec 15 02:45:45 2021, (1 sec)
Time.Estimated...: Wed Dec 15 02:45:46 2021, (0 secs)
Guess.Base.......: File (Desktop/LabFiles/HashCat/password.lst)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:      641 H/s (9.97ms) @ Accel:256 Loops:64 Thr:1 Vec:4
Recovered........: 2/2 (100.00%) Digests, 2/2 (100.00%) Salts
Progress.........: 1024/7118 (14.39%)
Rejected.........: 0/1024 (0.00%)
Restore.Point....: 0/3559 (0.00%)
Restore.Sub.#1...: Salt:1 Amplifier:0-1 Iteration:4992-5000
Candidates.#1....: #!comment: This list has been compiled by Solar Designer
all Project → cheese

Started: Wed Dec 15 02:45:43 2021
Stopped: Wed Dec 15 02:45:47 2021
```

13. You can close the terminal window in *Kali* and proceed to the next task.

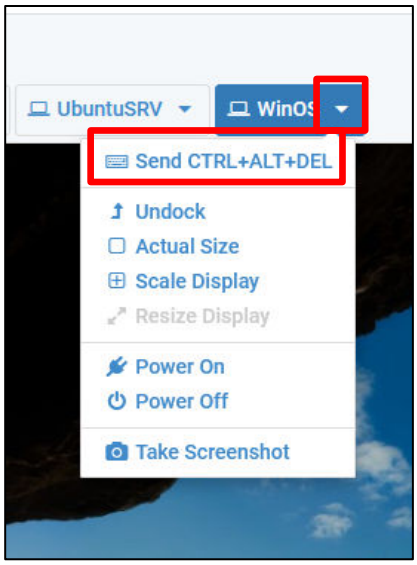
5 Crack Passwords with Cain and Abel

When security analysts think about password cracking, or should we say, password analysis, their first thought is to Linux-based utilities such as Hashcat and John the Ripper. However, there are a few utilities that have been developed for the Windows operating system. One of those utilities is *Cain and Abel*.

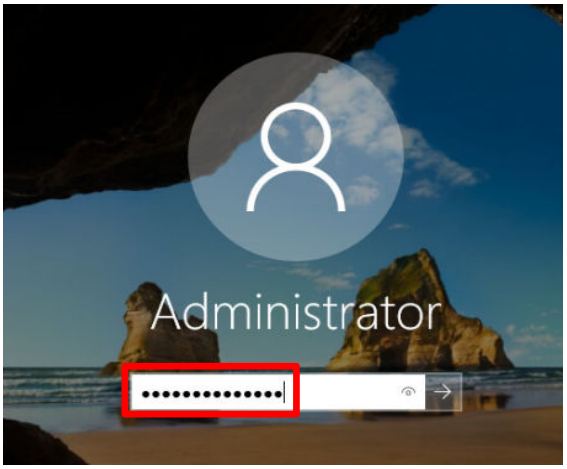
This password recovery and analysis tool handles a wide variety of password tasks, including sniffing the network, cracking encrypted passwords using dictionary, brute force, and cryptanalysis attacks, decoding scrambled passwords, as well as a great many other useful analysis tools.

In this section, you will use Cain and Abel to crack the hashes of passwords using both brute force and dictionary attacks.

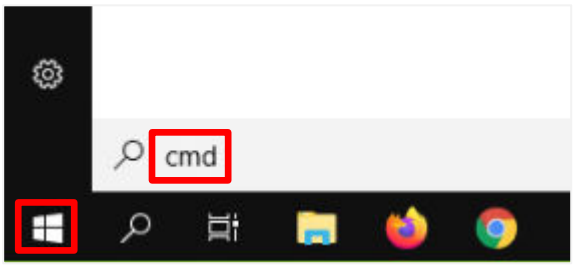
1. Click on the **WinOS** tab to access the graphical login screen.
2. Bring up the login window by sending a Ctrl + Alt + Delete. To do this, click the **WinOS** dropdown menu and click **Send CTRL+ALT+DEL**.



3. Log in as *Administrator* using the password: `NDGlabpass123!`

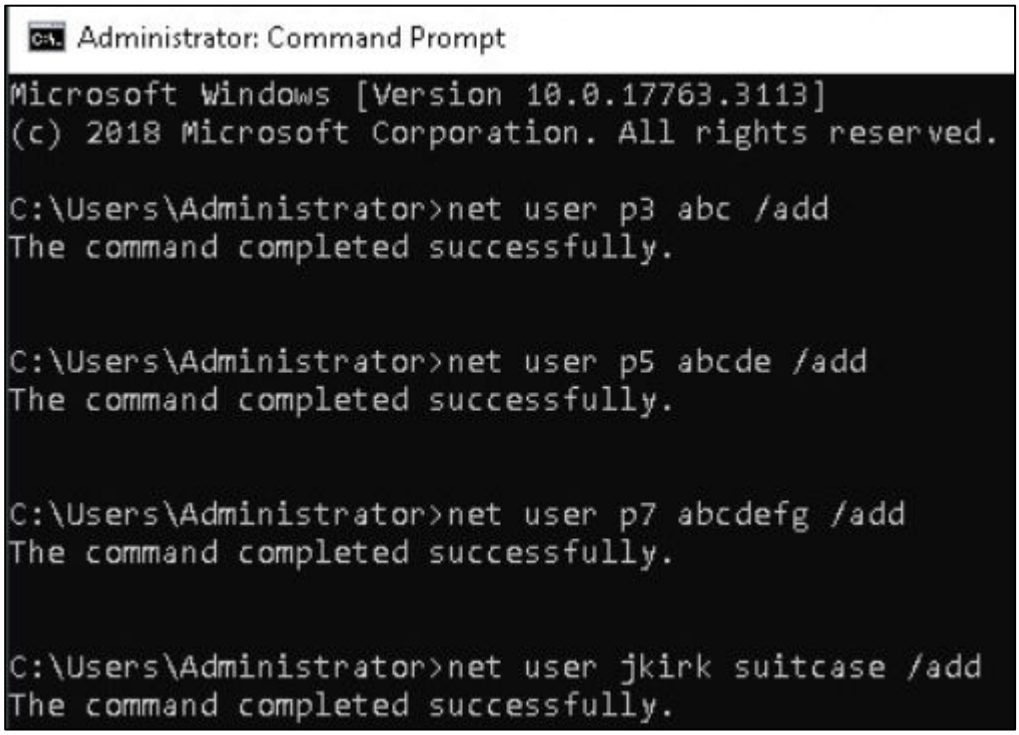


4. Click on the **Windows Start** button in the bottom-left corner and type `cmd`**,** and press the **Enter** key to bring up the *Command Prompt* window.



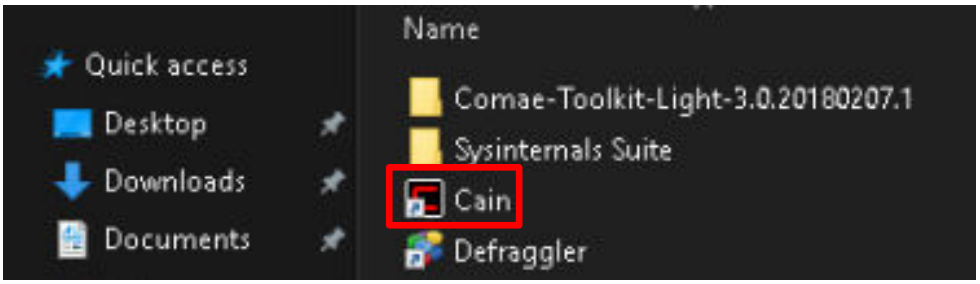
5. In the *Command Prompt* window, type the following commands to create several users and passwords.

```
net user p3 abc /add
net user p5 abcde /add
net user p7 abcdefg /add
net user jkirk suitcase /add
```



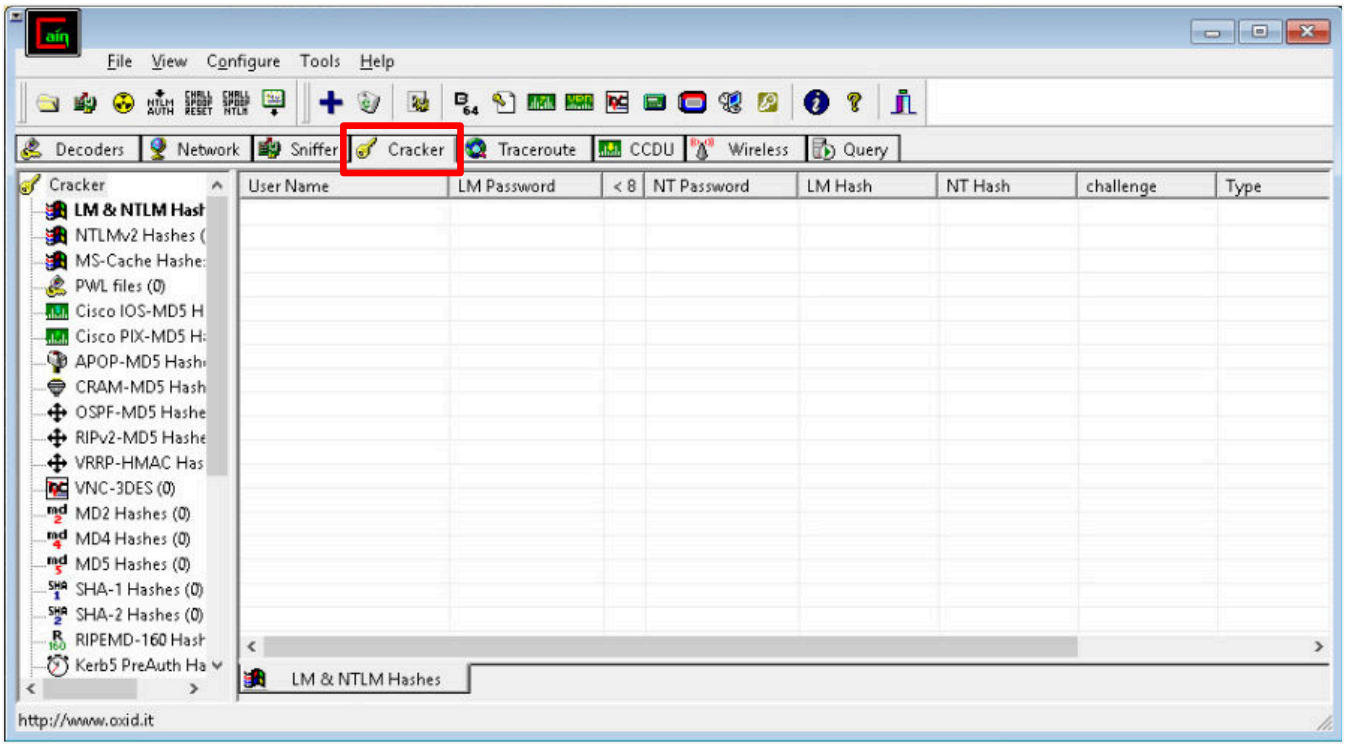
6. Close the **Command Prompt** window.
7. Double-click to open the **Toolbox** folder on the desktop.

8. Double-click on the **Cain** shortcut.

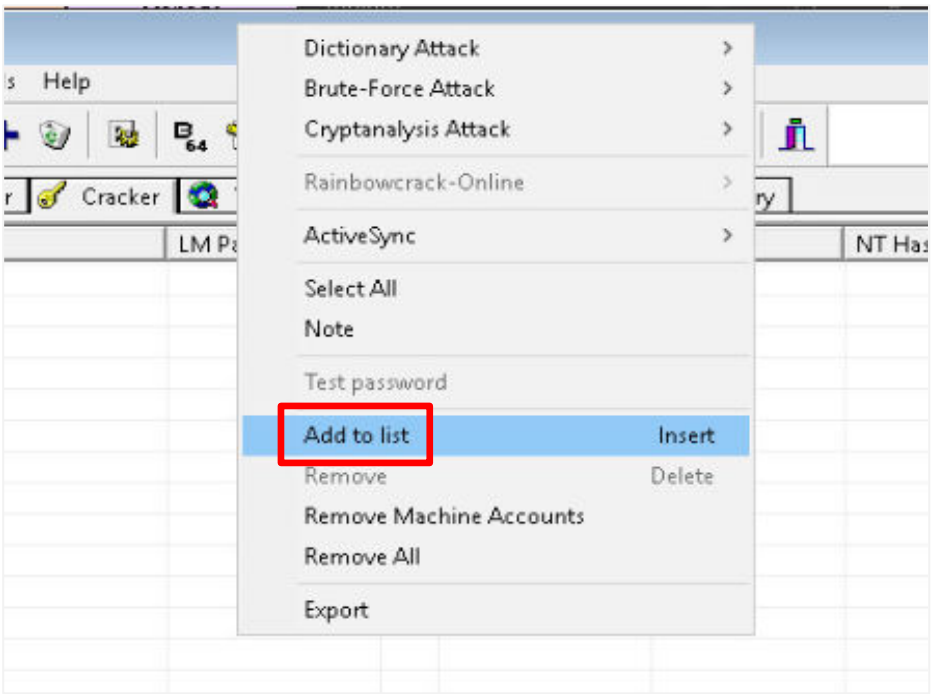


9. Click **OK** on the warning popup indicating that the *Windows firewall* is enabled.

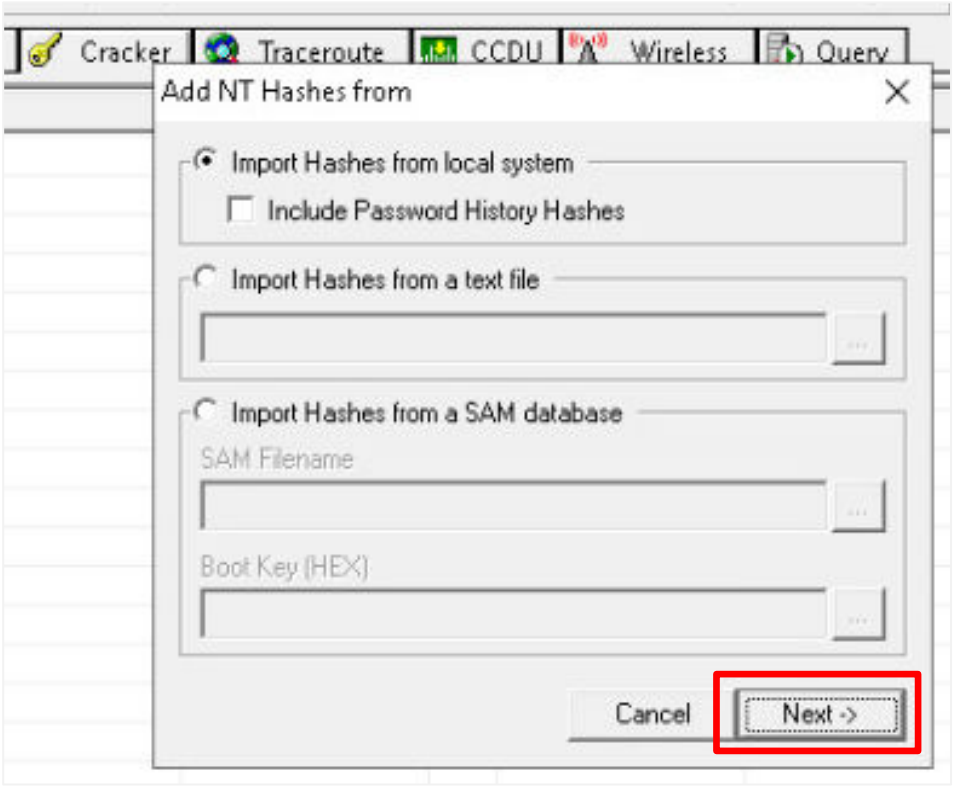
10. In the *Cain* window, at the top, click the **Cracker** tab.



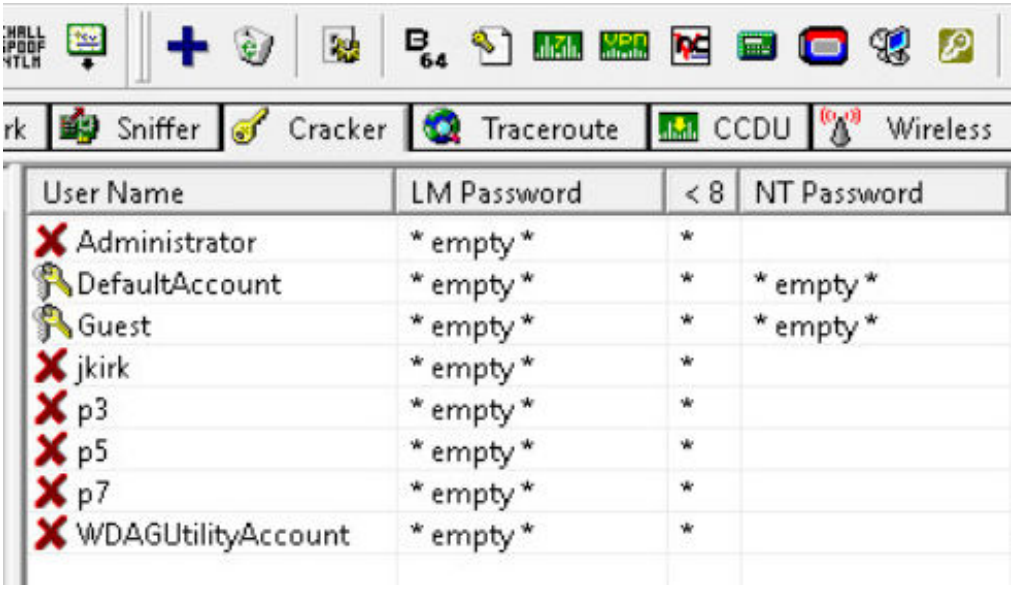
11. Move the mouse to the right panel where the blank pane appears with a grey grid, and right-click and select **Add to List** from the menu.



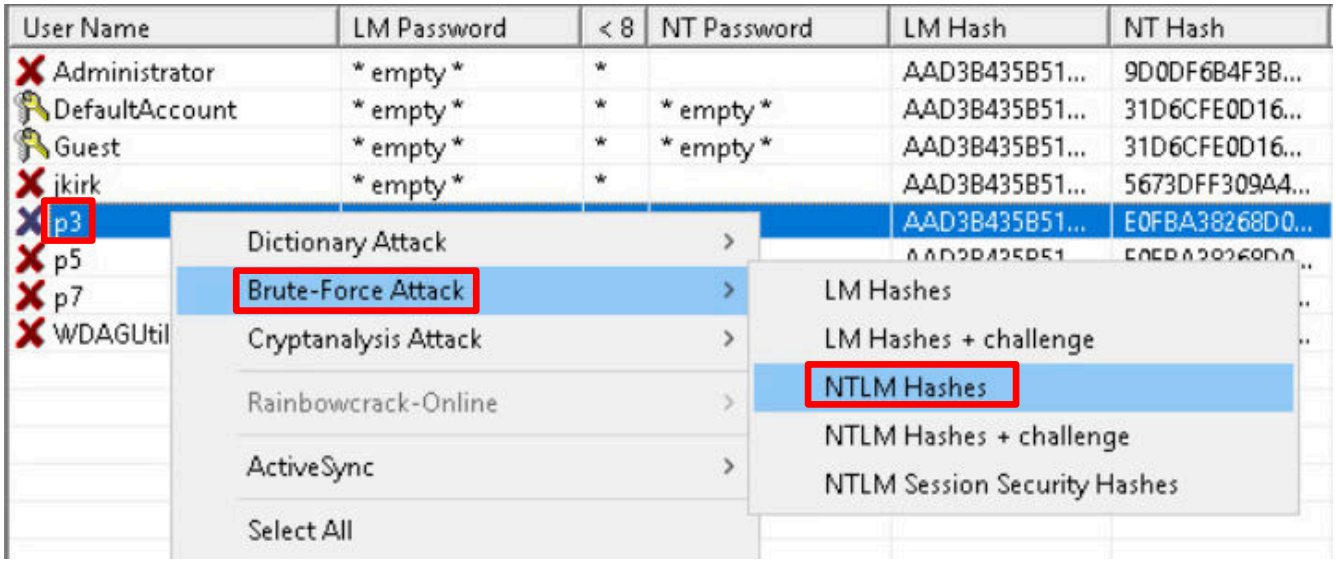
12. On the *Add NT Hashes from* window, click the **Next** button.



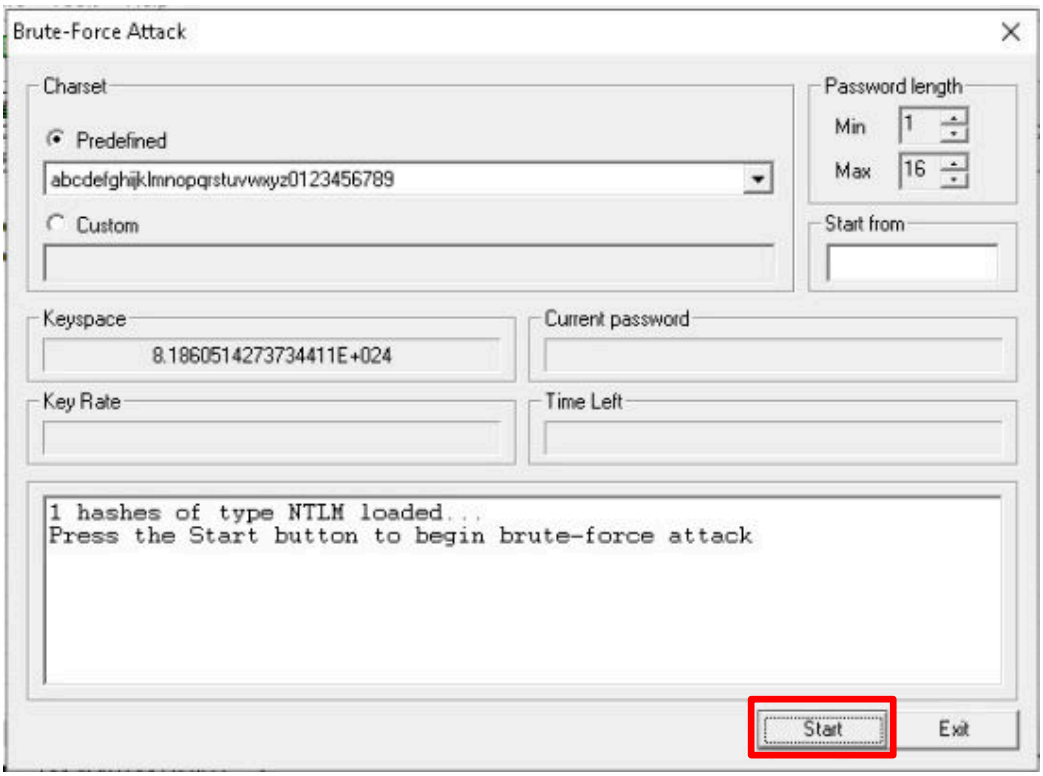
The password hashes appear as shown below:



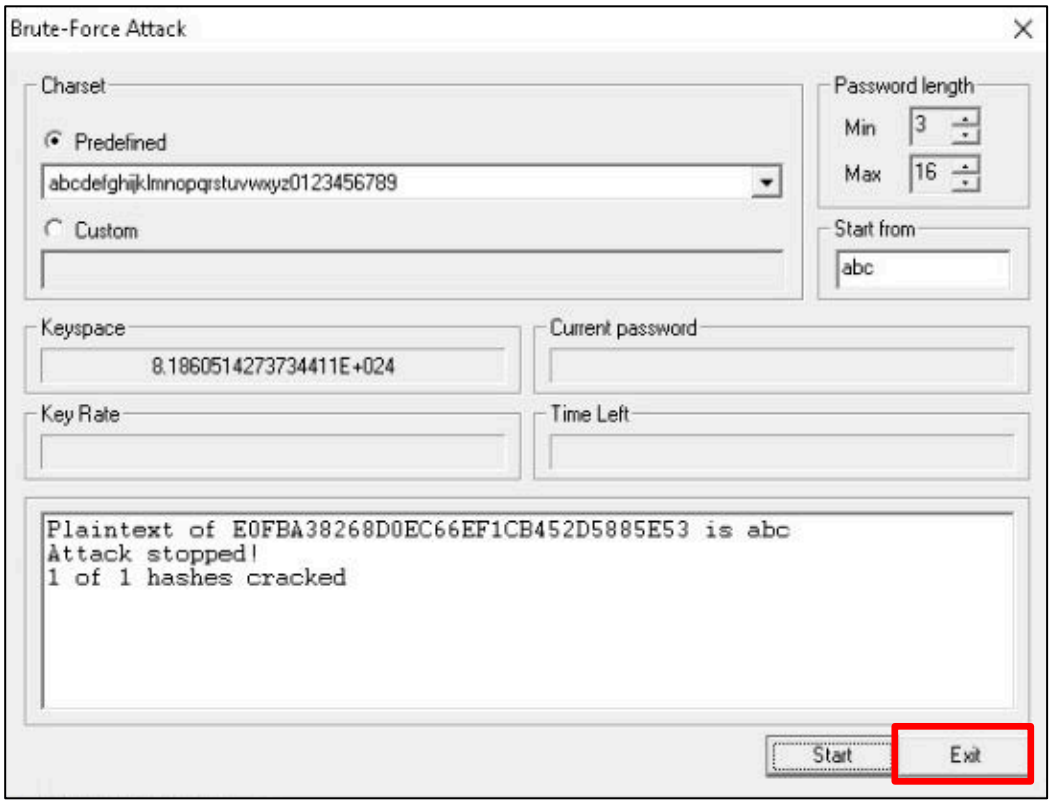
13. First, let's see how easy it is to use the **Brute Force Attack** to crack a 3-character password. Right-click on the **p3** user name and select **Brute-Force Attack>NTLM Hashes.**



14. In the *Brute-Force Attack* window, click the **Start** button.



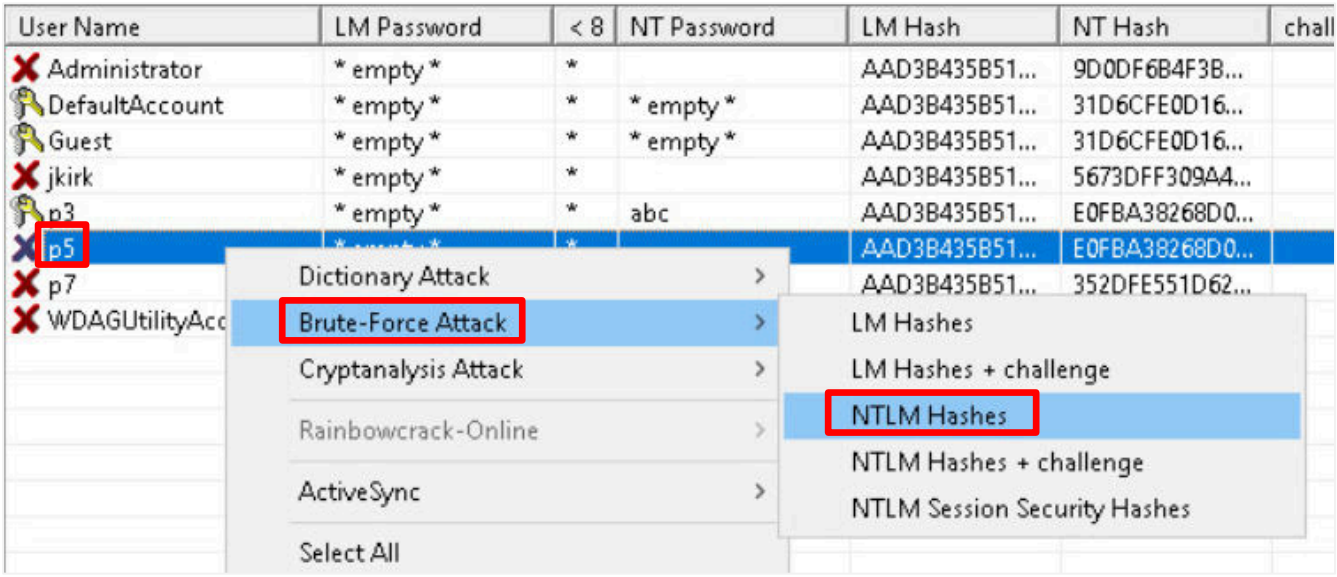
15. The *Brute-Force Attack* should find the three-letter password immediately. Close the *Brute-Force Attack* window by clicking on the **Exit** button.



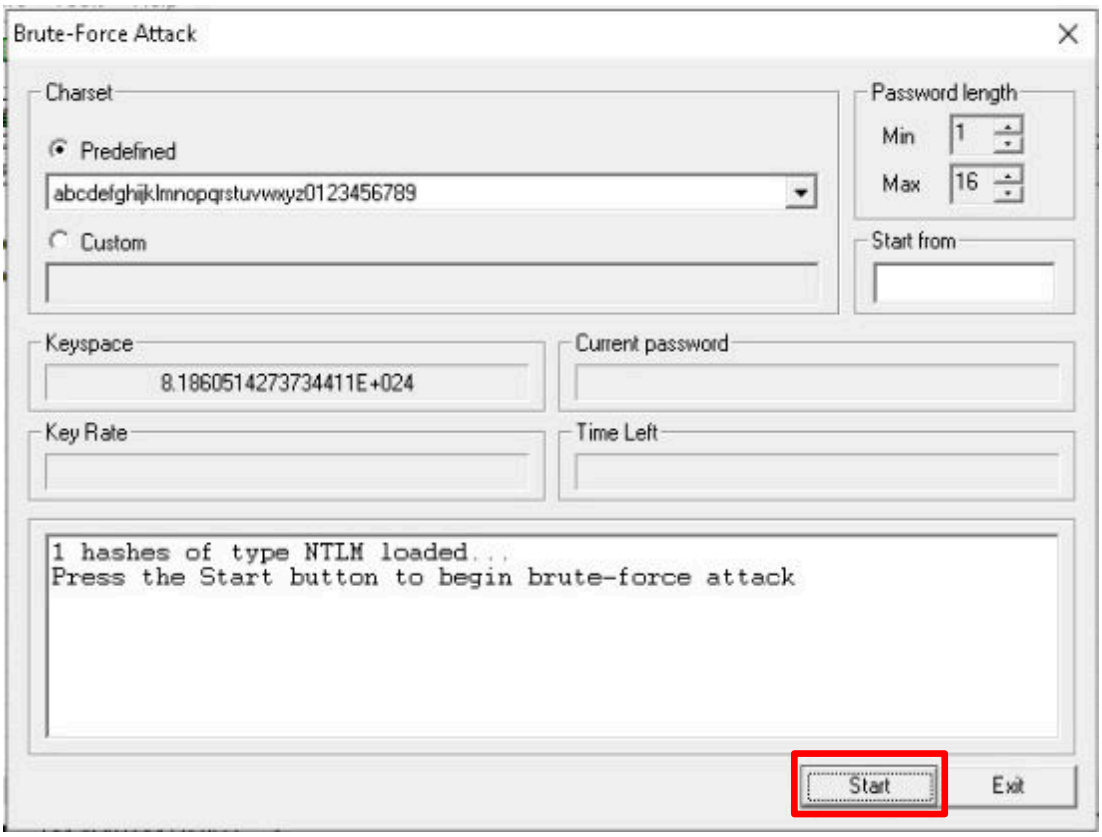
You should see the password of **abc** in the *NT Password* column.

User Name	LM Password	< 8	NT Password
Administrator	* empty *	*	
DefaultAccount	* empty *	*	* empty *
Guest	* empty *	*	* empty *
jkirk	* empty *	*	
p3	* empty *	*	abc
p5	* empty *	*	
p7	* empty *	*	
WDAGUtilityAccount	* empty *	*	

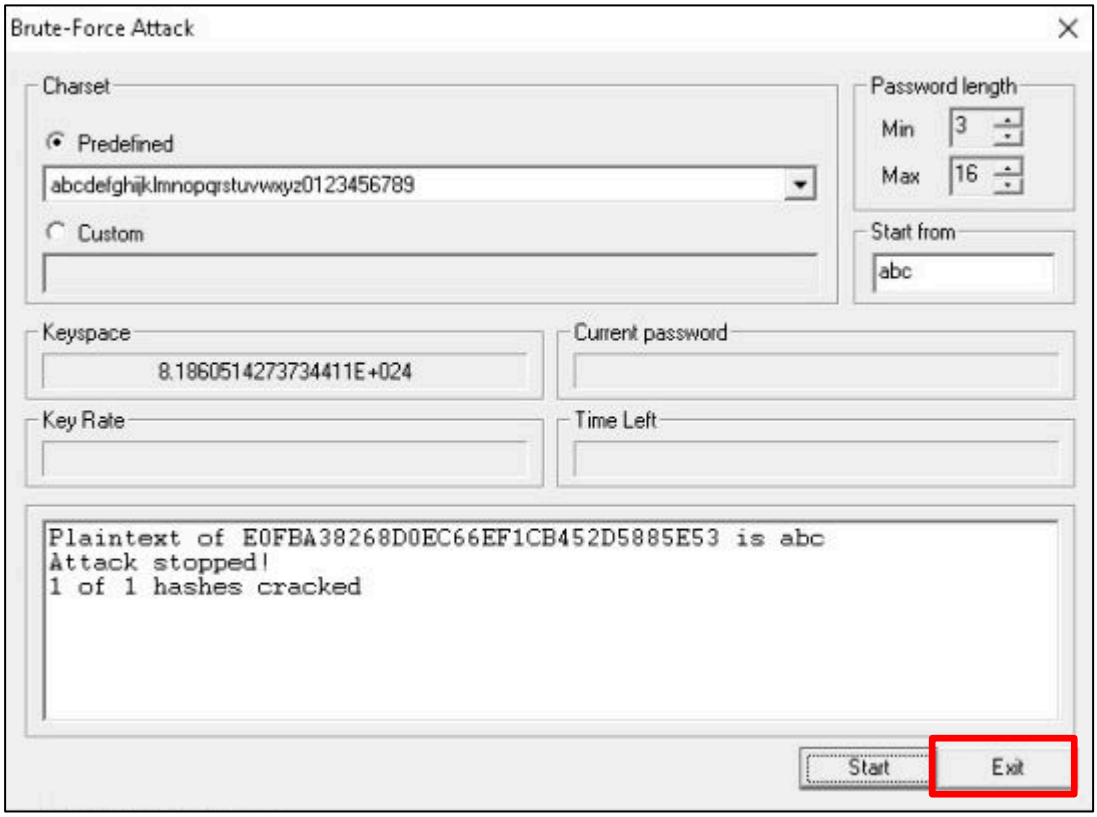
16. Now, let's try the *Brute Force Attack* to crack a 5-character password. Right-click on the **p5** user name and select **Brute-Force Attack>NTLM Hashes.**



17. In the *Brute-Force Attack* window, click the **Start** button.



18. The **Brute-Force Attack** should find the 5-letter password within a few seconds. Close the *Brute-Force Attack* window by clicking on the **Exit** button



This time you should see the password of **abcde** in the *NT Password* column for the **p5** user account.

User Name	LM Password	< 8	NT Password
Administrator	* empty *	*	
DefaultAccount	* empty *	*	* empty *
Guest	* empty *	*	* empty *
jkirk	* empty *	*	
p3	* empty *	*	abc
p5	* empty *	*	abcde
p7	* empty *	*	
WDAGUtilityAccount	* empty *	*	

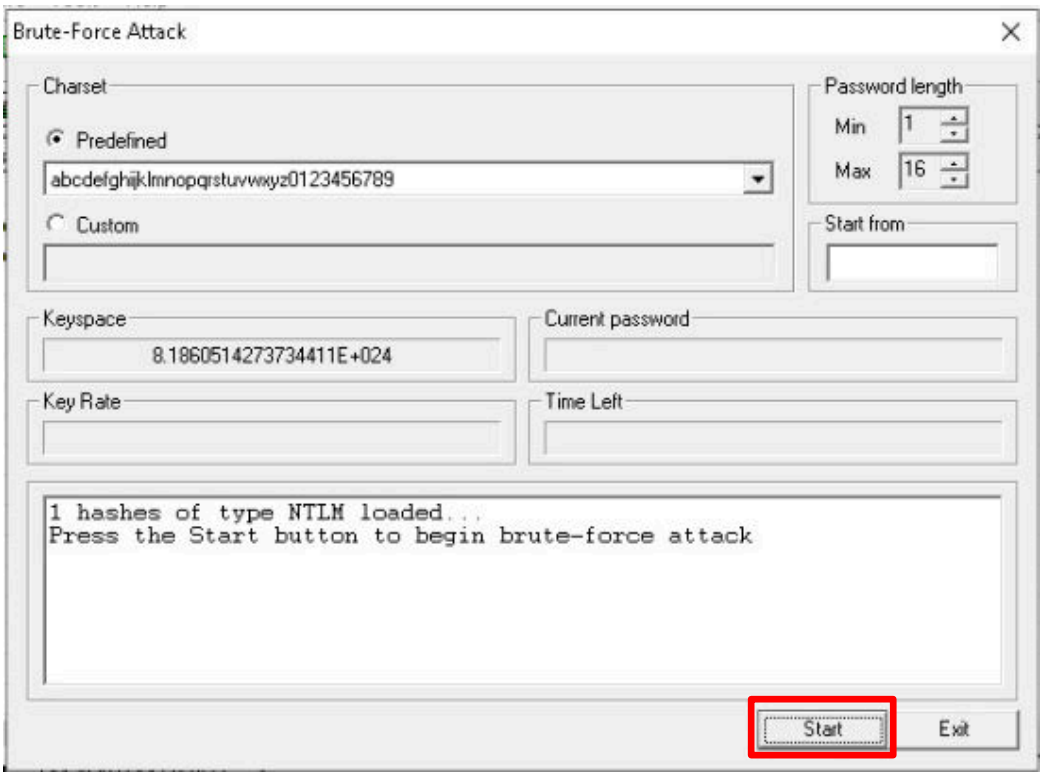
19. Next, let's try the *Brute Force Attack* to crack a 7-character password. Right-click on the **p7** user name and select **Brute-Force Attack>NTLM Hashes.**

User Name	LM Password	< 8	NT Password	LM Hash	NT Hash
Administrator	* empty *	*		AAD3B435B51...	9D0DF6B4F3B...
DefaultAccount	* empty *	*	* empty *	AAD3B435B51...	31D6CFE0D16...
Guest	* empty *	*	* empty *	AAD3B435B51...	31D6CFE0D16...
jkirk	* empty *	*		AAD3B435B51...	5673DFF309A4...
p3	* empty *	*	abc	AAD3B435B51...	E0FBA38268D0...
p5	* empty *	*	abcde	AAD3B435B51...	3F5156E39D9C...
p7	* empty *	*		AAD3B435B51...	352DFE551D62...
WDAGUtili				AAD3B435B51...	DB5070F29B75...

Dictionary Attack
Brute-Force Attack
Cryptanalysis Attack
Rainbowcrack-Online
ActiveSync
Select All

LM Hashes
LM Hashes + challenge
NTLM Hashes
NTLM Hashes + challenge
NTLM Session Security Hashes

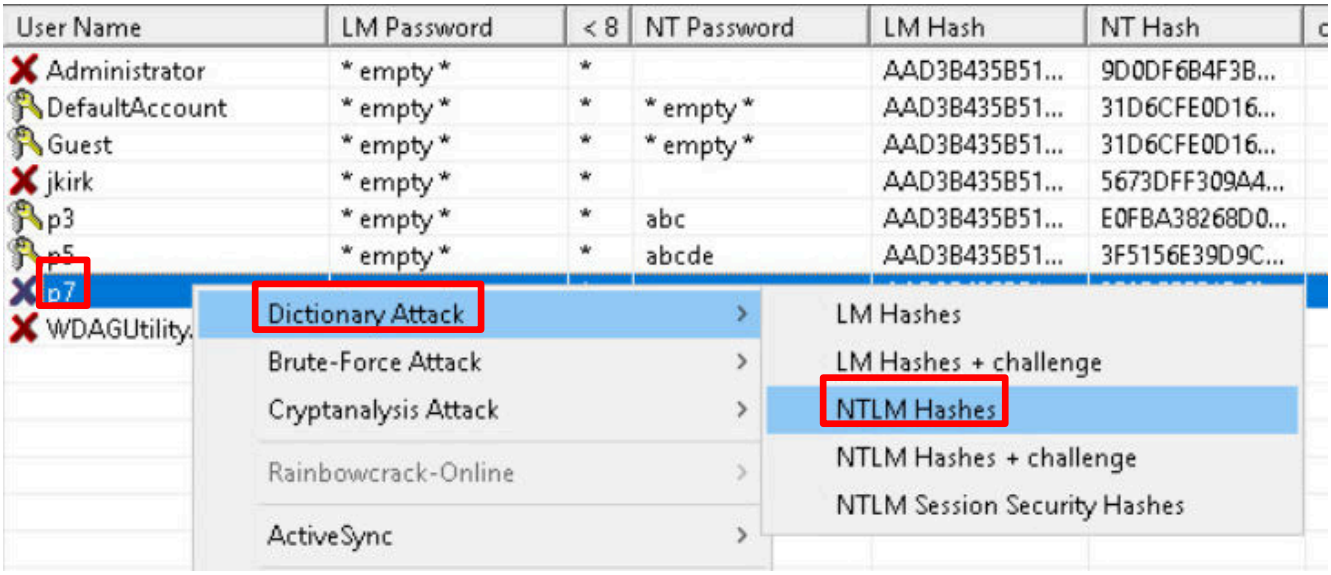
20. In *the Brute-Force Attack* window, click the **Start** button.



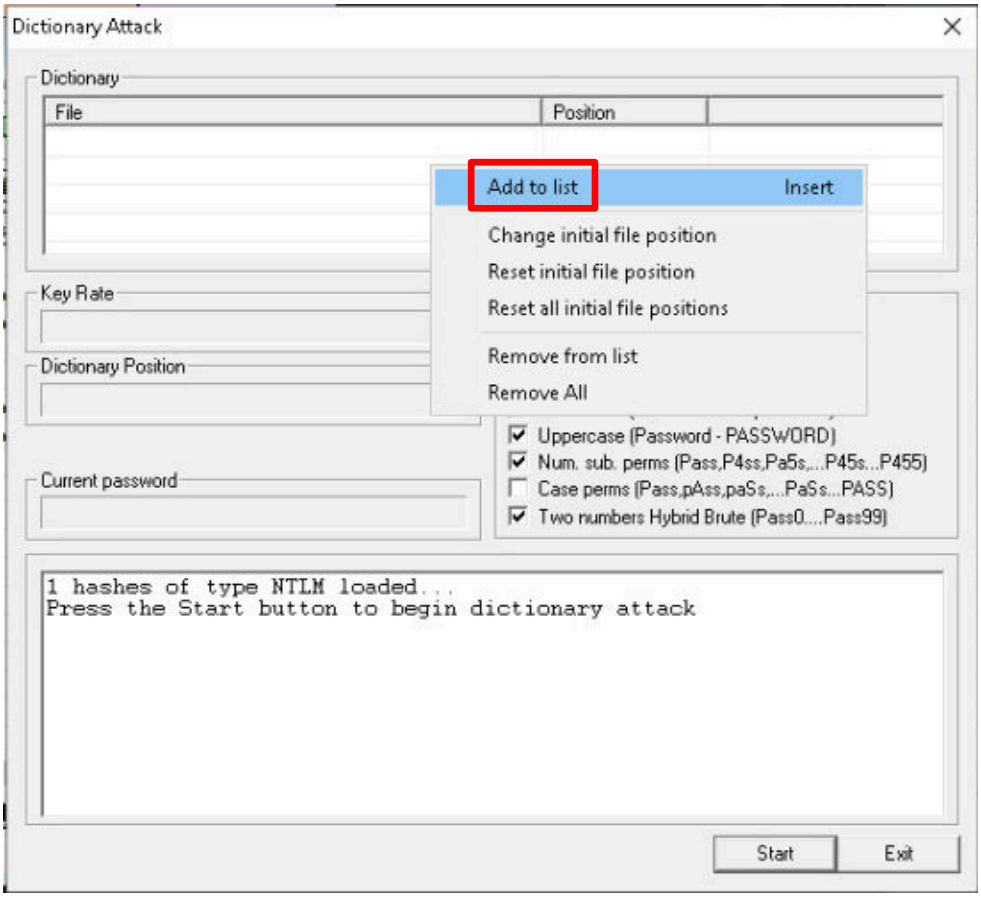
21. The 7-character password is much harder to crack, and it will take several hours (at least) to crack, so if you don't want to wait (which you don't have to), click the **Stop** button, then click the **Exit** button.

    As passwords get longer, it requires a lot more time (exponentially, at least) to crack them using brute force. But, what about using a dictionary attack like the ones you did using *John the Ripper* and *Hashcat*? Let's give it a try.

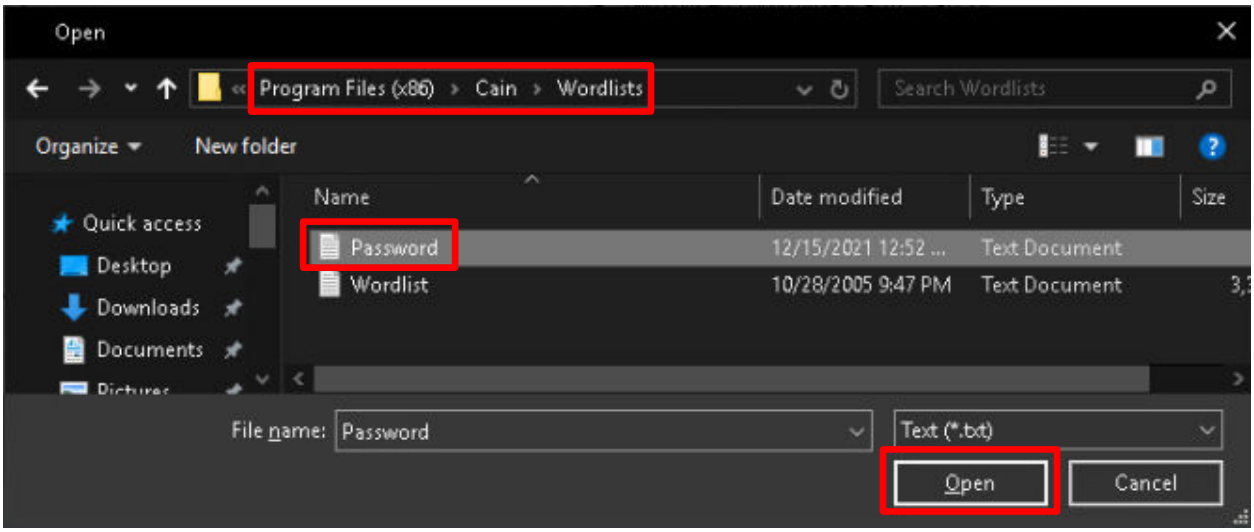
22. Right-click on the **p7** user name and select **Dictionary Attack>NTLM Hashes.**



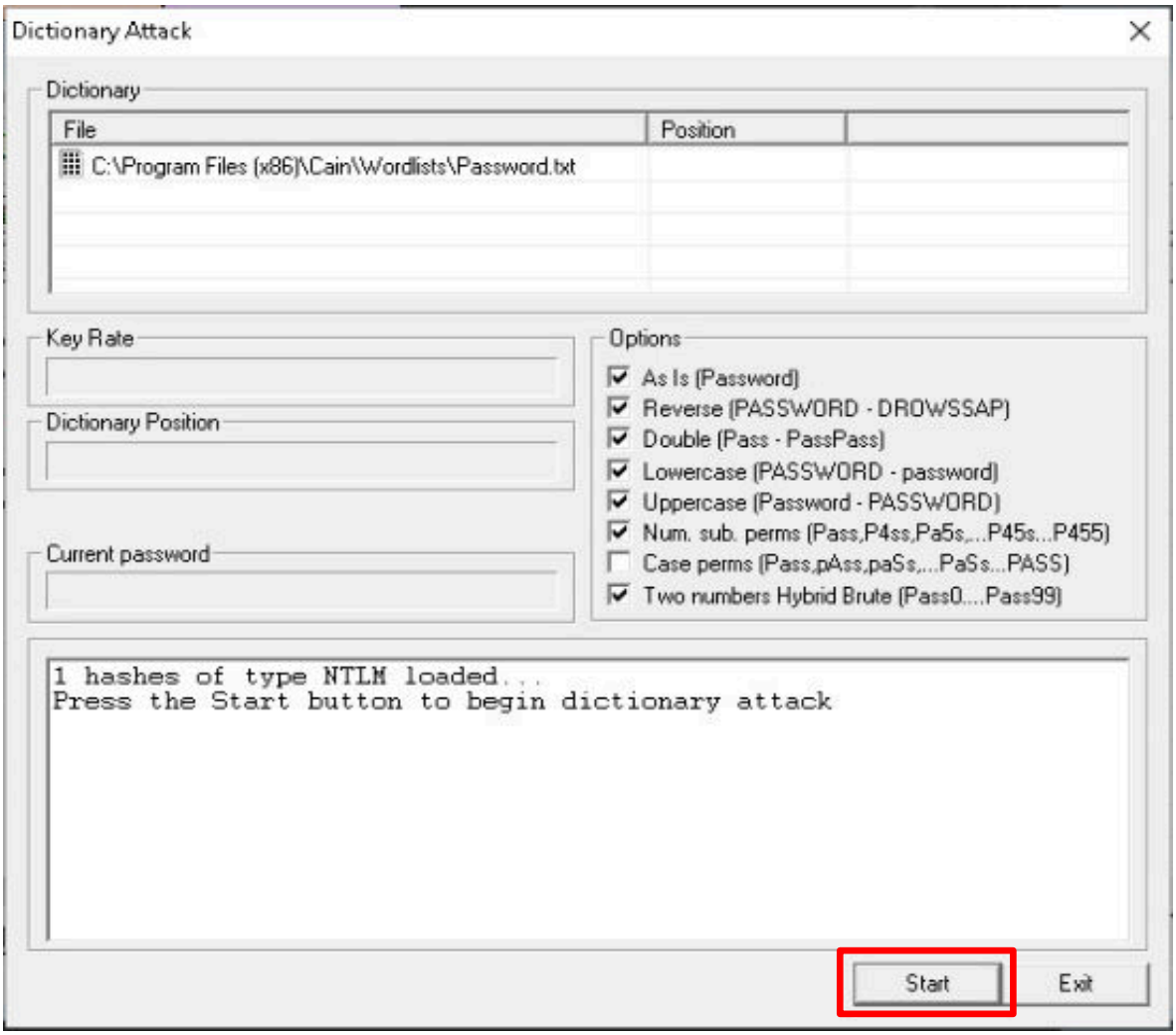
23. In the *Dictionary Attack* window, move the mouse to the **Dictionary/File** panel, where the blank pane appears with a grey grid. Right-click and select **Add to List** from the menu.



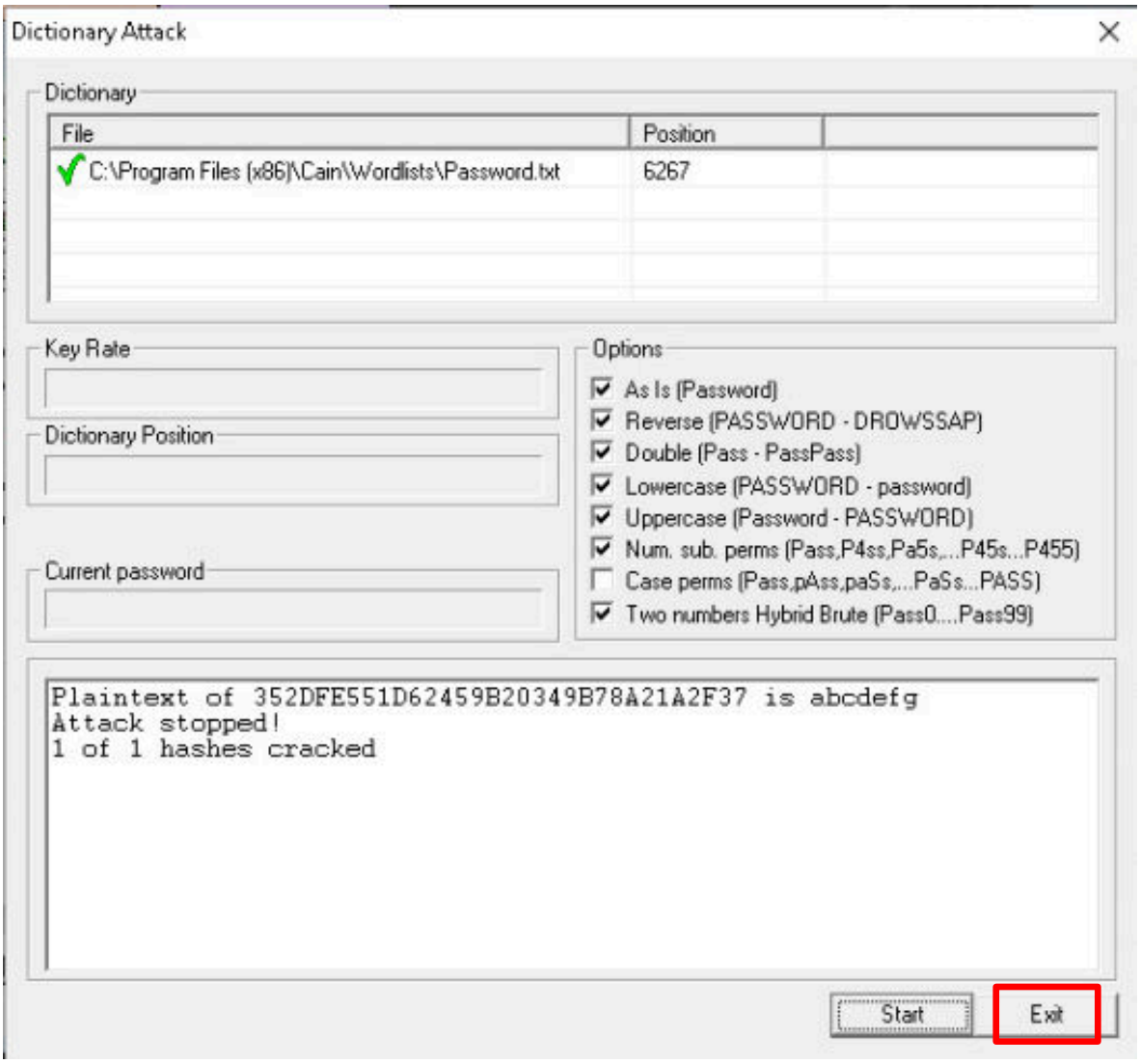
24. In the *Open* (file) screen, navigate to **Local Disk (C:)>Program Files(x86)>Cain>Wordlists** and select the **Password** text file from the list. Then, click the **Open** button.



25. Confirm that the **Password.txt** file is in the *File* list and click on the **Start** button.



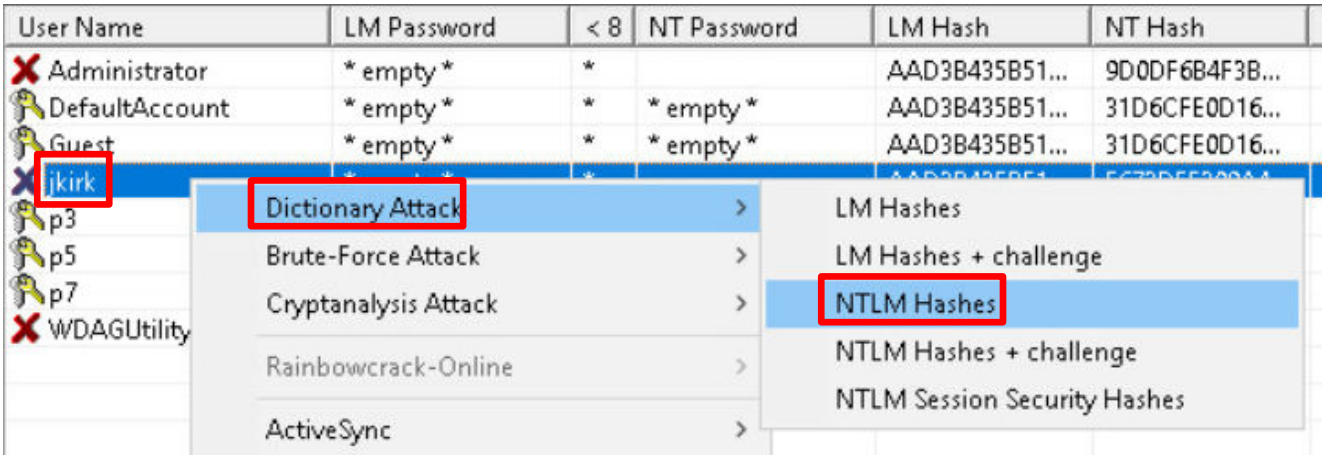
26. It might take 3-4 minutes to find the password. Once the password is found, click **Exit.**



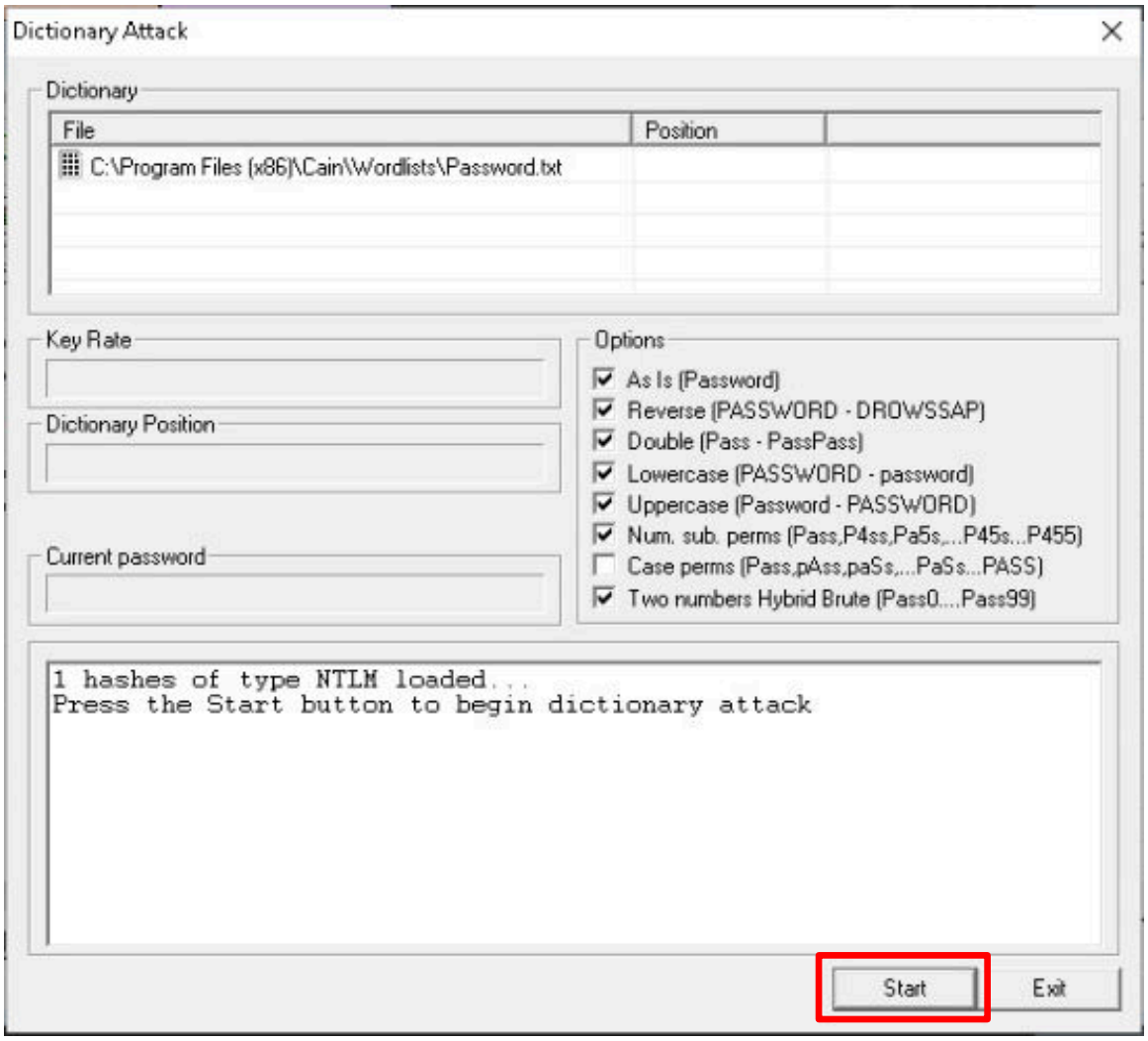
You should see the password of **abcdefg** in the *NT Password* column for the **p7** user account.

User Name	LM Password	< 8	NT Password
Administrator	* empty *	*	
DefaultAccount	* empty *	*	* empty *
Guest	* empty *	*	* empty *
jkirk	* empty *	*	
p3	* empty *	*	abc
p5	* empty *	*	abcde
p7	* empty *	*	abcdefg
WDAGUtilityAccount	* empty *	*	

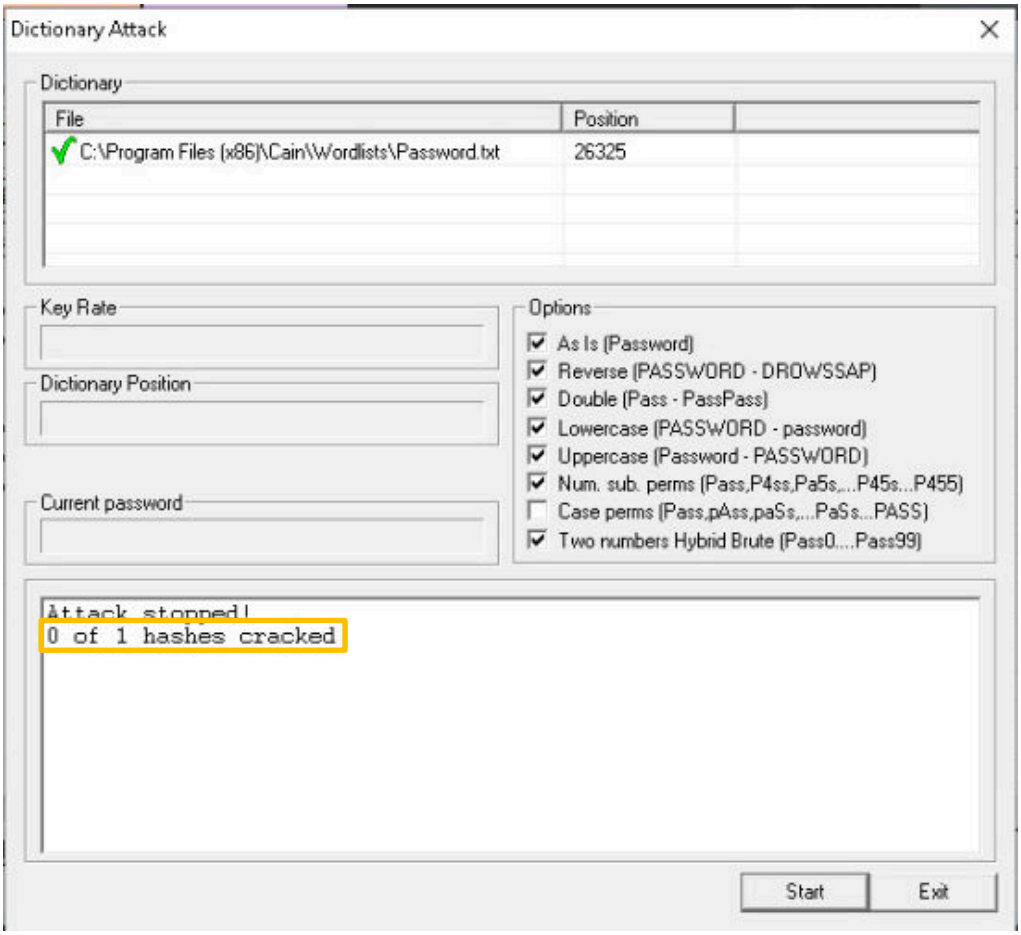
27. Now, use the *Dictionary Attack* on the **jkirk** user and see if the password can be cracked. Right-click on the **jkirk** user name and select **Dictionary Attack>NTLM Hashes.**



28. Confirm that the **Password.txt** file is in the *File* list and click on the **Start** button.

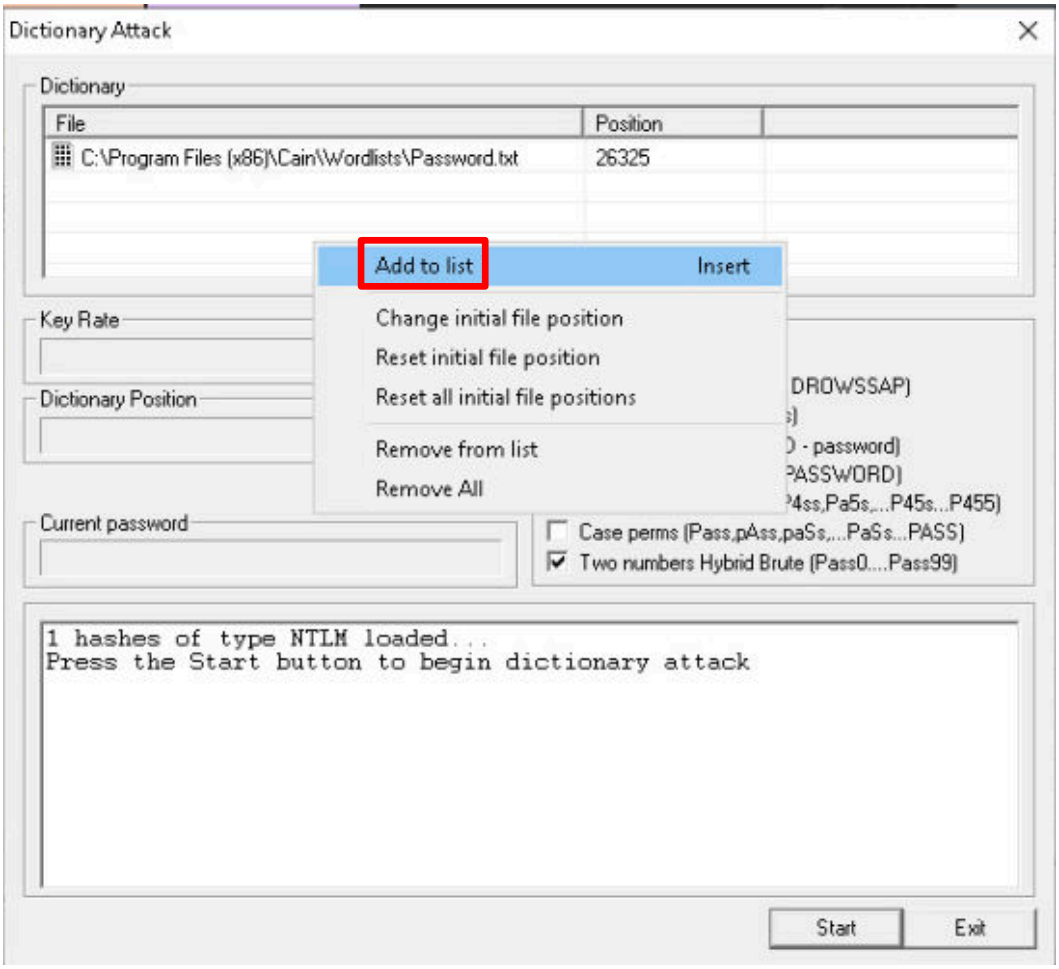


You should see that the *Dictionary Attack* did not result in any hashes being cracked. This means *jkirk's* password is not in the Password.txt wordlist.

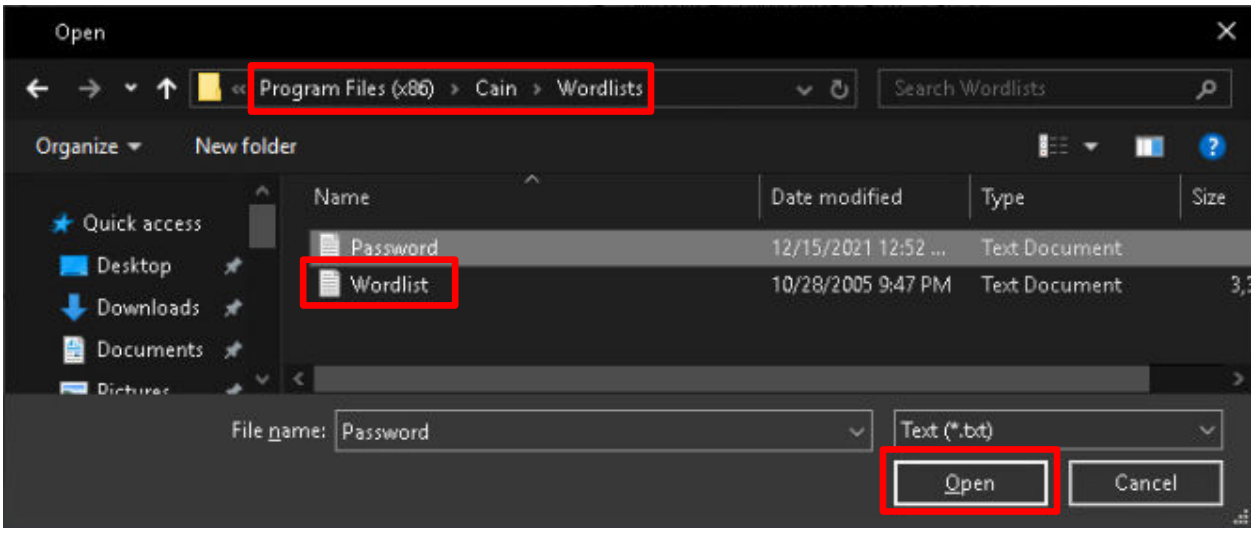


Let's expand the search by adding another wordlist to the dictionary.

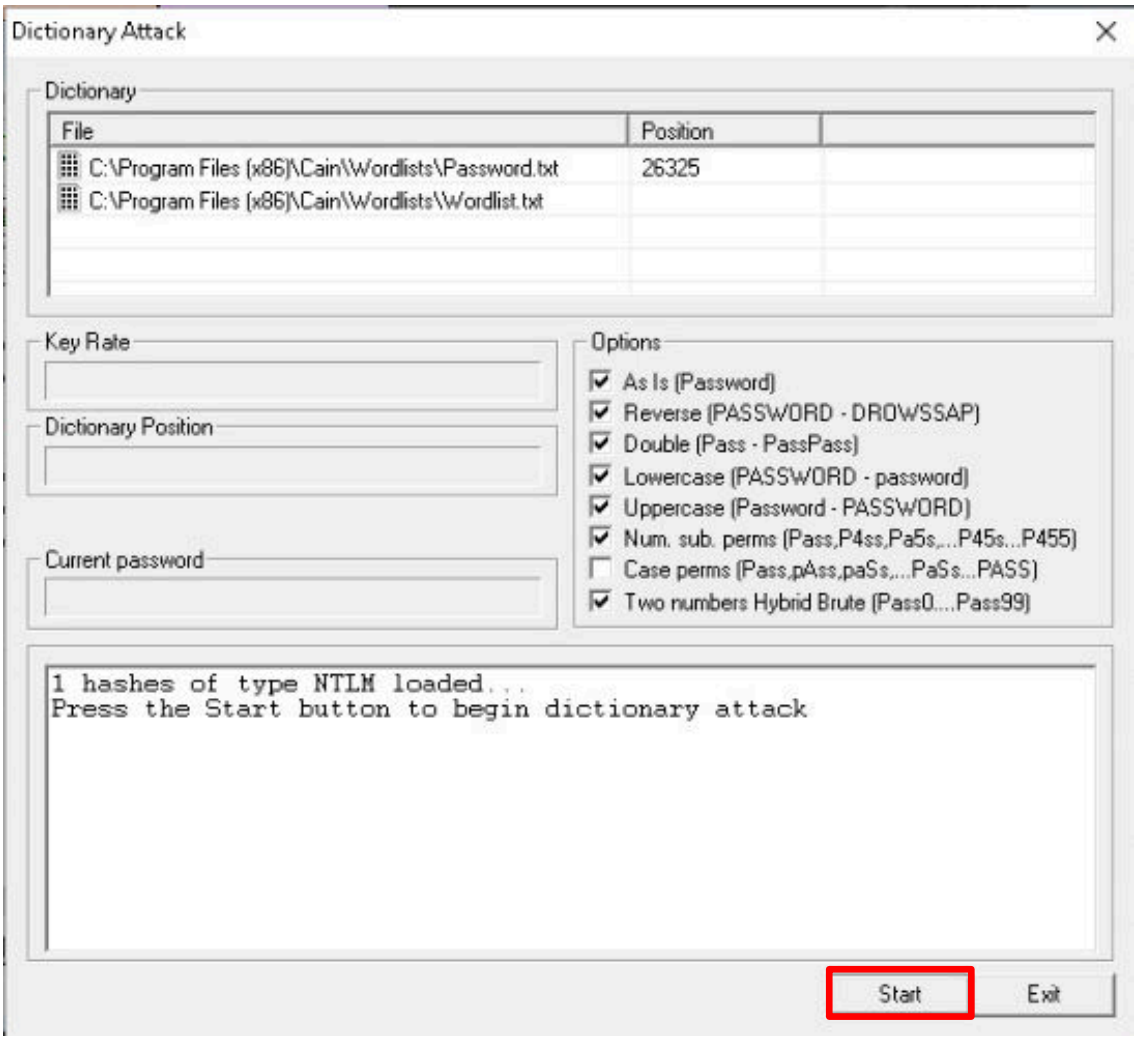
29. In the *Dictionary Attack* window, move the mouse to the **Dictionary/File** panel, right-click and select **Add to List** from the menu.



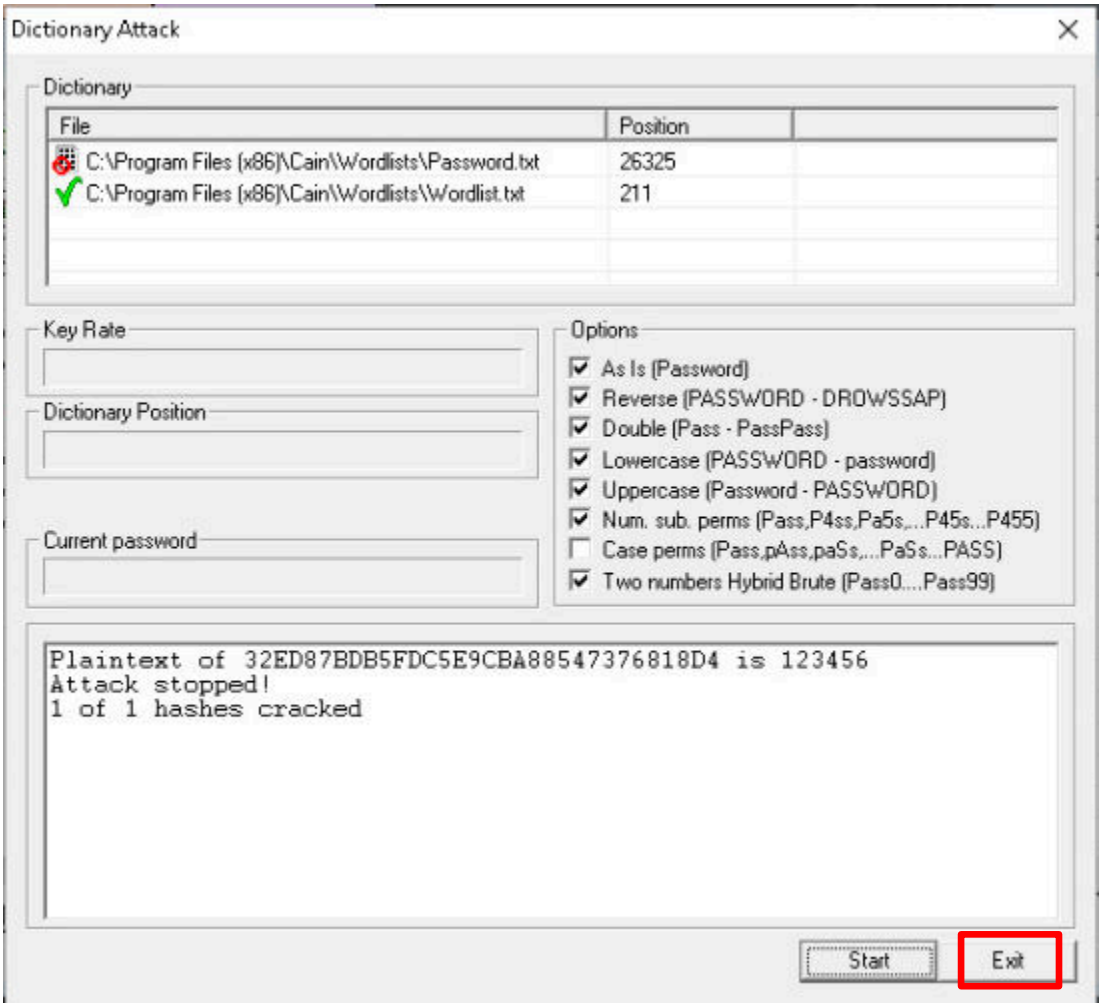
30. In the *Open* (file) screen, navigate to **Local Disk (C:)>Program Files(x86)>Cain>Wordlists** (if not already shown) and select the **Wordlist** text file from the list. Then, click the **Open** button.



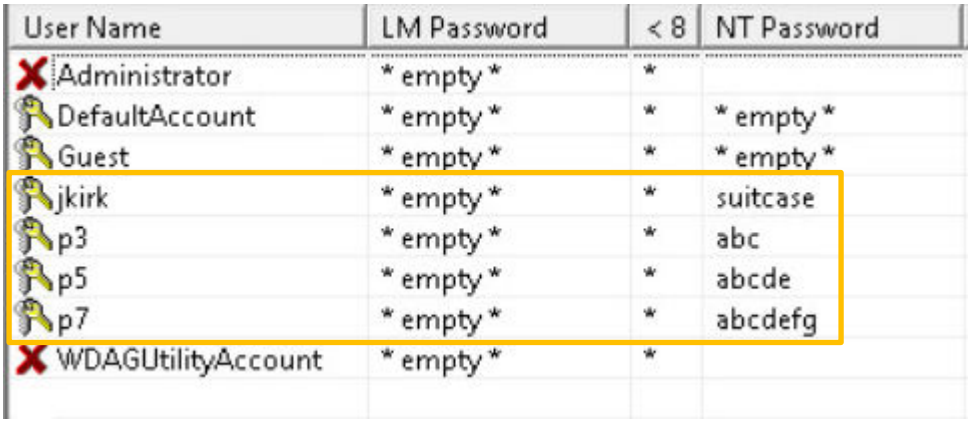
31. Confirm that both files are in the *Dictionary* list and click on the **Start** button.



32. It might take a couple of minutes to find the password. Once the password is found, click the **Exit** button.



33. You should see that all of the user's passwords have been cracked.



User Name	LM Password	< 8	NT Password
Administrator	* empty *	*	
DefaultAccount	* empty *	*	* empty *
Guest	* empty *	*	* empty *
jkirk	* empty *	*	suitcase
p3	* empty *	*	abc
p5	* empty *	*	abcde
p7	* empty *	*	abcdefg
WDAGUtilityAccount	* empty *	*	

34. This concludes the lab. You may now end the reservation.