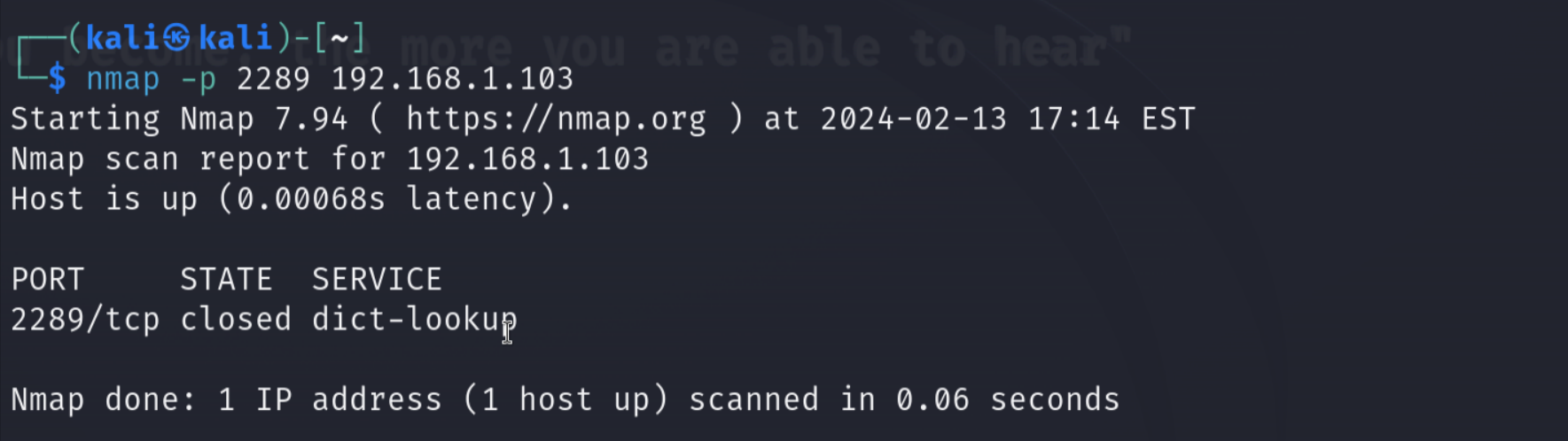
# Section 3. Exploring Password-Based Attacks

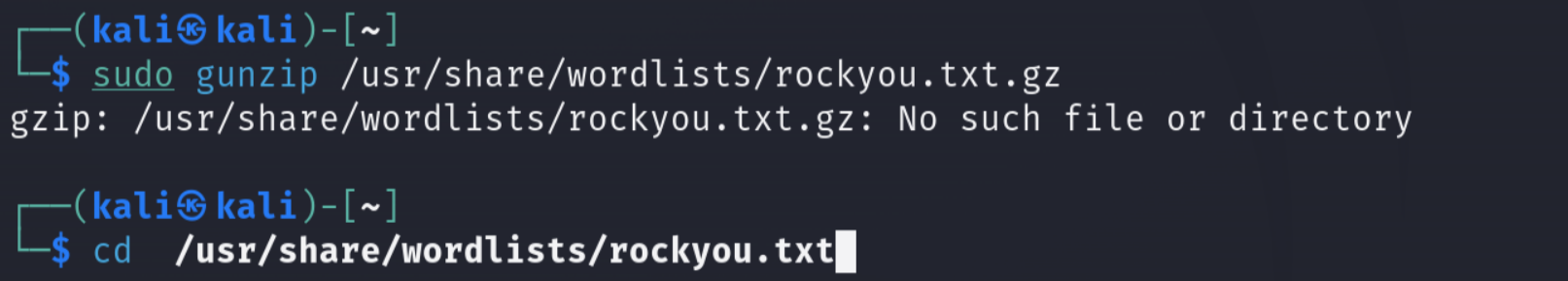
## A. Exploiting Windows Remote Desktop Protocol

**Step 1:** Metasploitable 3 and Kali Linux on and connected.

**Step 2:** Scan for RDP on Metasploitable 3.

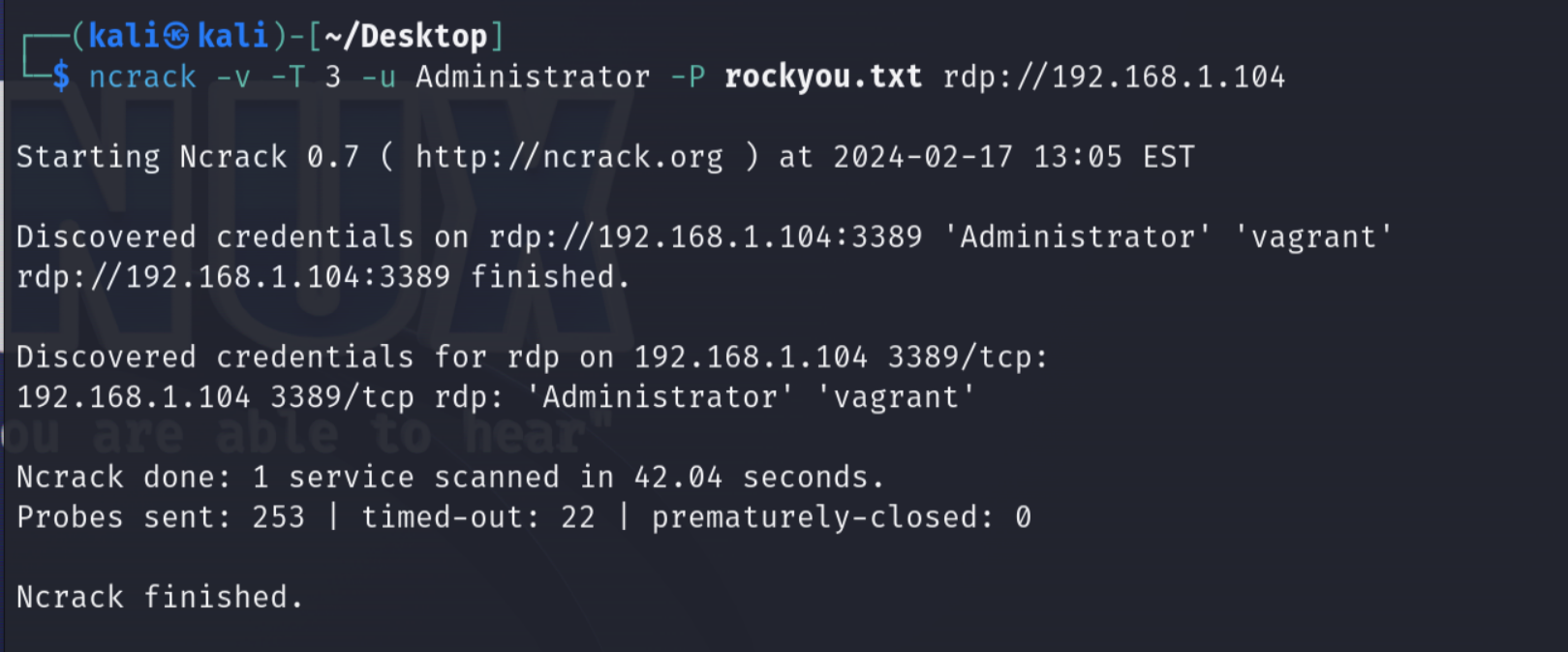


**Step 3:** Unzip the rockyou.txt.gz wordlist file.

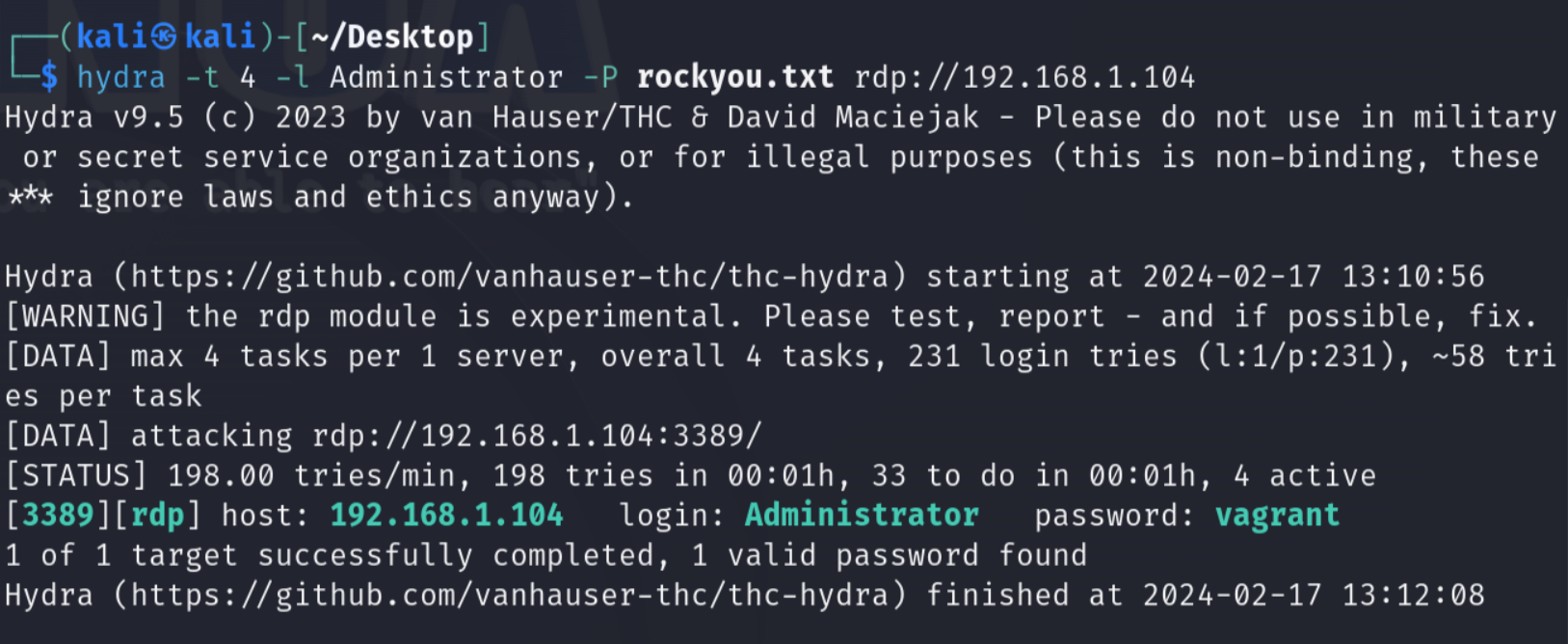


(Already unzipped)

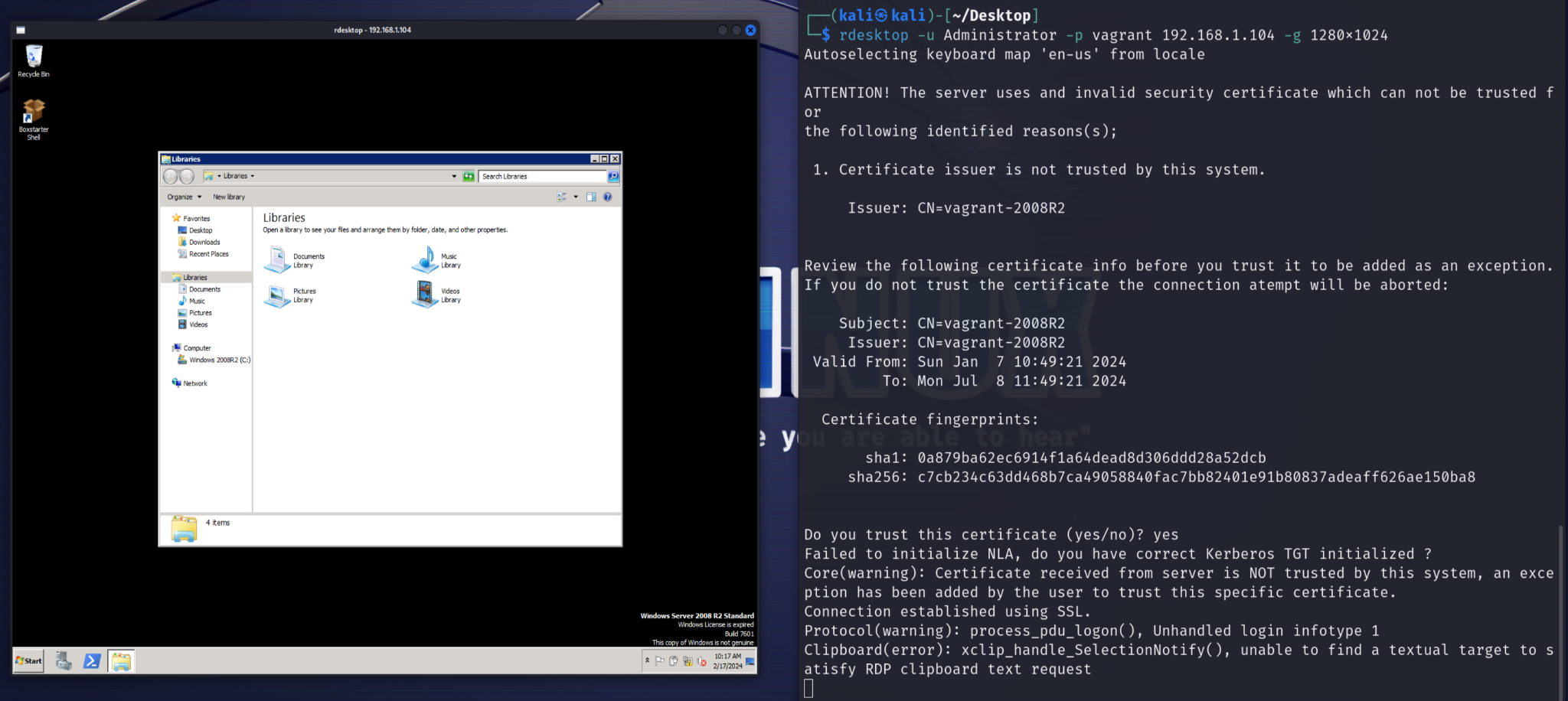
**Step 4:** Use ncrack to attack Metasploitable 3.



**Step 5:** Use Hydra to attack Metasploitable 3.

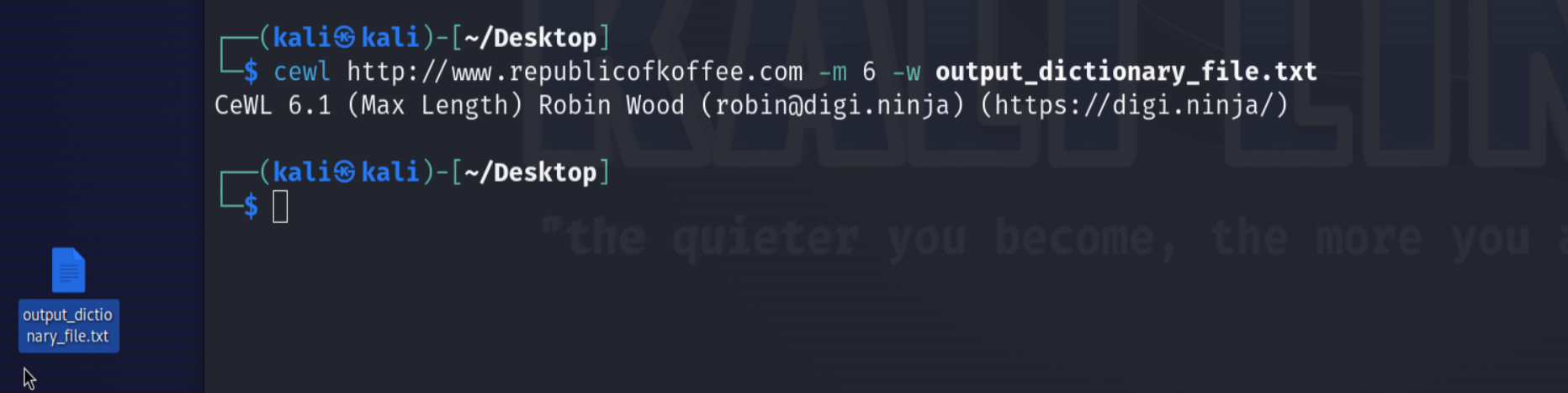


**Step 5:** Using rdesktop, log in using the username and password.



## B. Creating Wordlists using Keywords

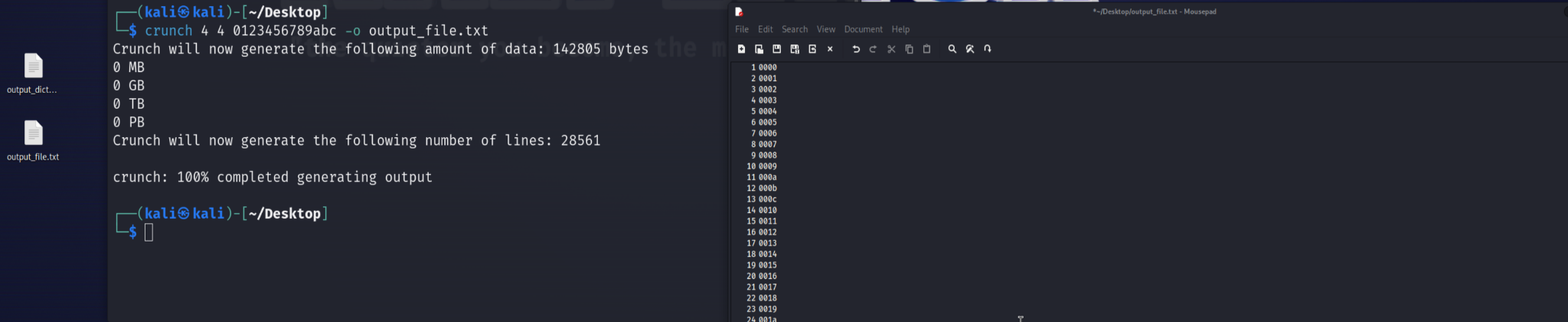
**Step 1:** Using CeWL, create a custom text file listing potential passwords.



## 

## C. Crunching the Wordlists

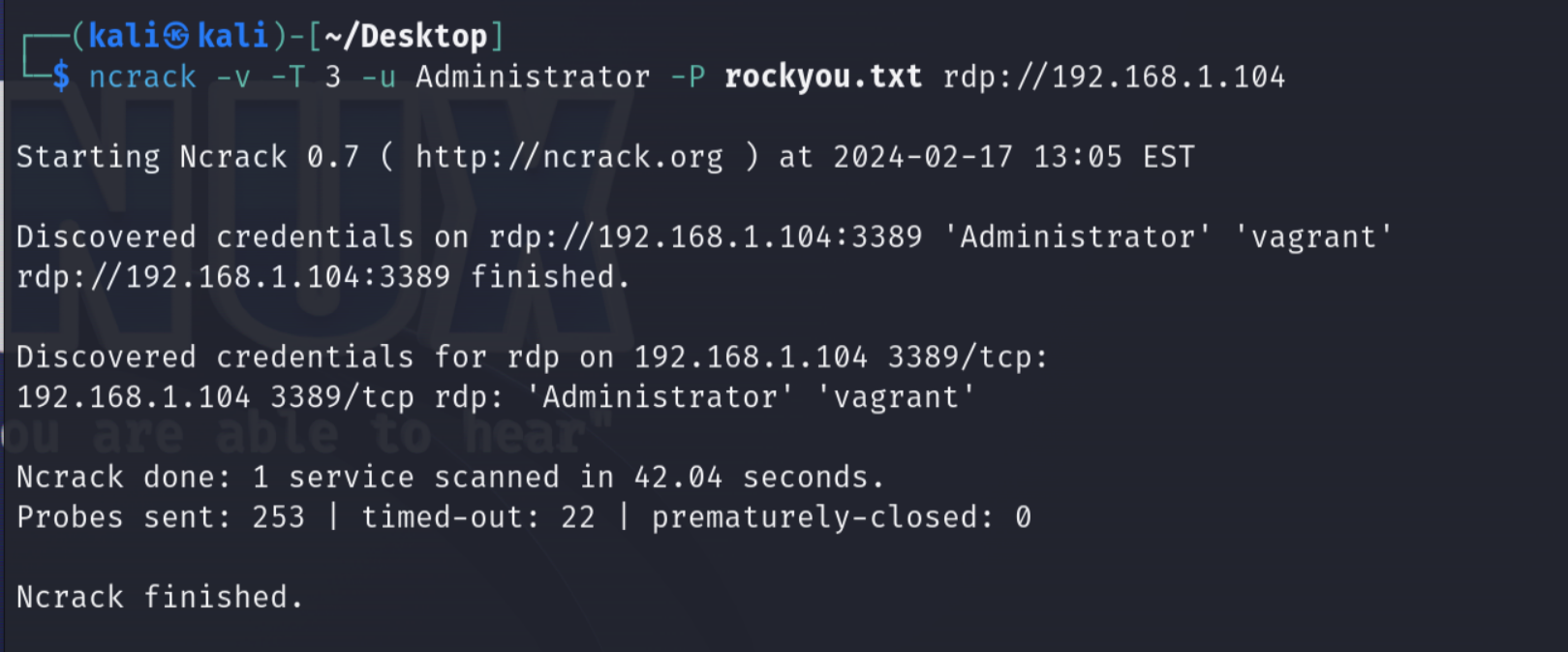
**Step 1:** Using Crunch, create a list of potential passwords with custom specifications.

Left - written file, middle - command used, right - sample of created passwords in file

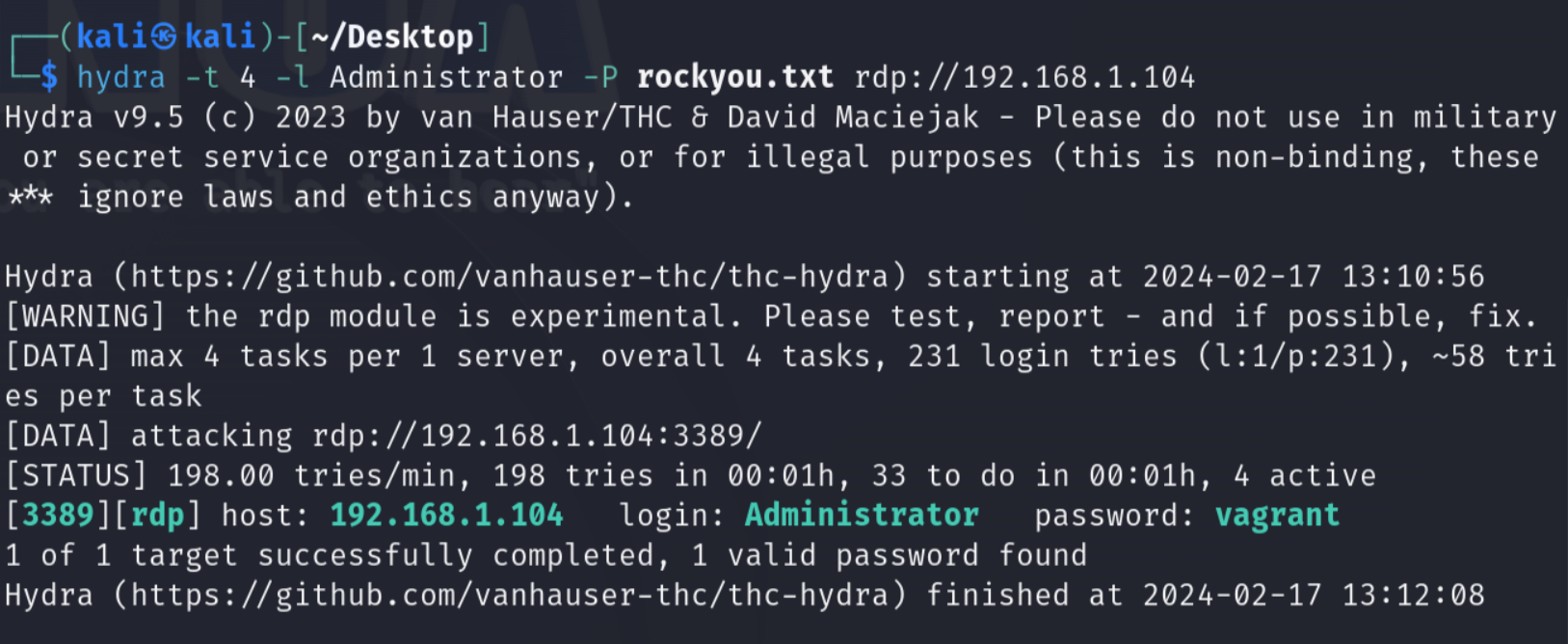
# **Deliverables**

## Exercise 1. Use Hydra or Ncrack to discover the Metasploitable 3 virtual machine password.

**Ncrack:**

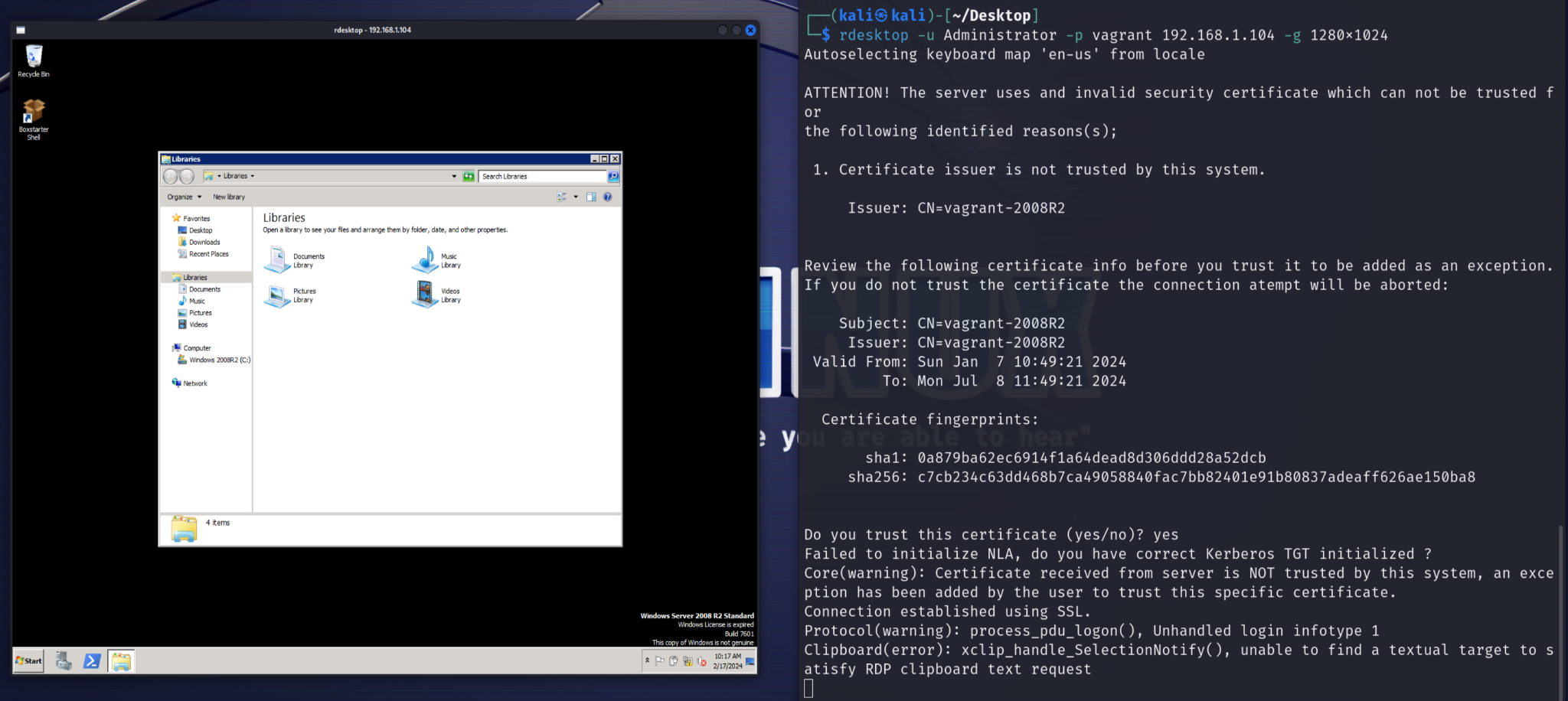


**Hydra:**



## 

## Exercise 2. Use the password uncovered in the previous exercise to establish an RDP session

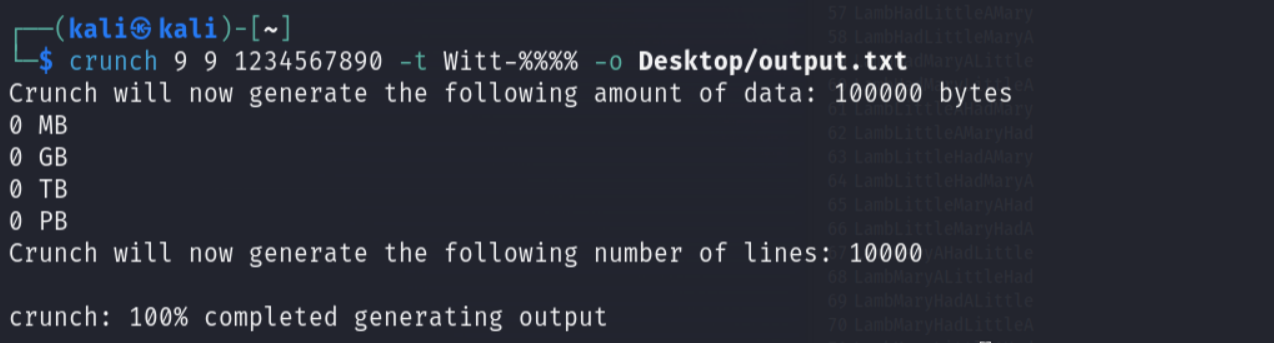


## Exercise 3. Try to crack the following hashes

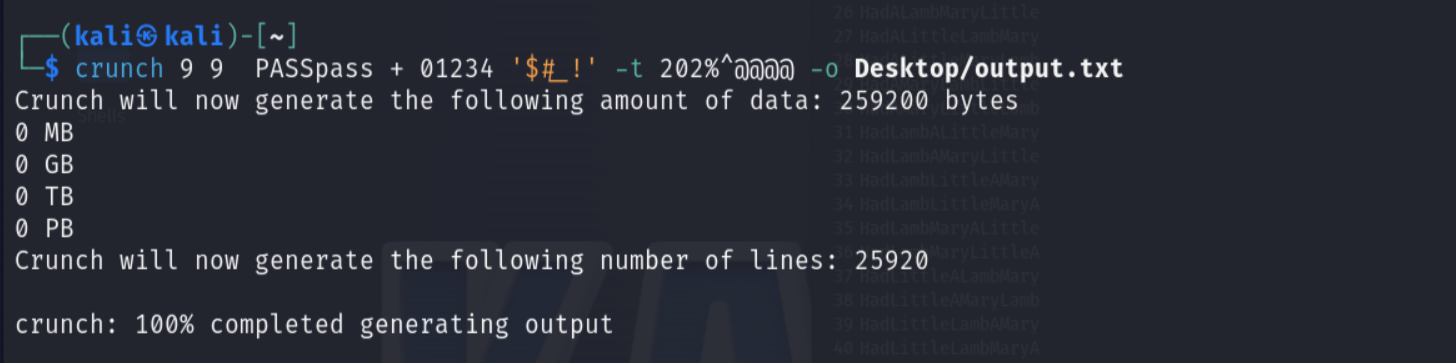
| User: | Hash: | WordList Command: | Result: |
| --- | --- | --- | --- |
| Joe | 05b47156dac156b841c412527eb08642 | crunch 9 9 1234567890 -t Witt-%%%% -o Desktop/output.txt | Witt-3251  (MD5) |
| Malik | D883E2D53B20240026AA3A0D202AD267 | crunch 9 9 PASSpass + 01234 ‘$\_!#’ -t 202%^@@@@ -o Desktop/output.txt | 2023$PaSS  (ntlm) |
| Zoe | eaf187e4eb6bfa7d913f0afc4d6f94f1f0ae67d452526beccf8534ebd09e6b953578ed21acd10e015a439ba0dbb4b91a2abeb0aece4492b5a1b93a0ad1a10c05 | Googlable answer | liverpool  (SHA) |
| Jane | 5ef22fe0b6b2868a9f8ae4bb7adc14cd | crunch 18 18 -o Desktop/output.txt -p Mary Had A Little Lamb | LittleALambHadMary  (md5) |

(Proof of running successfully shown below)

User 1.



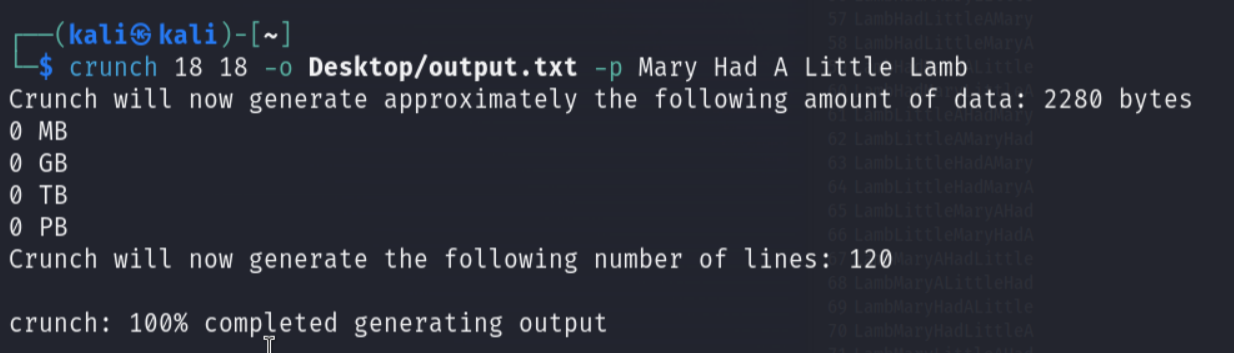
User 2.



User 3.

Because of its commonality as a password, this solution was found quickly online.

User 4.



Note: My VM did not have enough memory to successfully run Hashcat successfully so the following Python code was produced and a Ctrl-F was used to find the right hash:

**MD5:**

import hashlib

def generate\_md5\_hash(password):

return hashlib.md5(password.encode()).hexdigest()

def create\_md5\_hash\_file(input\_file, output\_file):

with open(input\_file, 'r') as f\_in, open(output\_file, 'w') as f\_out:

for line in f\_in:

password = line.strip()

md5\_hash = generate\_md5\_hash(password)

f\_out.write(f"{password}:{md5\_hash}\n")

if \_\_name\_\_ == "\_\_main\_\_":

input\_file = "hashes.txt"

output\_file = "md5\_hashes.txt"

create\_md5\_hash\_file(input\_file, output\_file)

print(f"MD5 aved to {output\_file}.")

**ntlm:**

import hashlib

def generate\_ntlm\_hash(password):

return hashlib.new('md4', password.encode('utf-16le')).hexdigest()

def create\_ntlm\_hash\_file(input\_file, output\_file):

with open(input\_file, 'r') as f\_in, open(output\_file, 'w') as f\_out:

for line in f\_in:

password = line.strip()

ntlm\_hash = generate\_ntlm\_hash(password)

f\_out.write(f"{password}:{ntlm\_hash}\n")

if \_\_name\_\_ == "\_\_main\_\_":

input\_file = "hashes.txt"

output\_file = "ntlm\_hashes.txt"

create\_ntlm\_hash\_file(input\_file, output\_file)

print(f"NTLM saved to {output\_file}.