Bypassing Security Project 2 – Final Report

Course: SDA Protik

Dean

Kevin

Emanuel

Date:11.03.2025

Table of Contents

📌 **Findings:**

- **HTTP traffic is unencrypted**, making it easy to steal credentials via packet sniffing.
- **HTTPS encrypts data**, preventing password leaks.

---

4. SSH Traffic Analysis

- Monitoring an **SSH connection** between Kali Linux and another machine.
- Uploading **secret1.txt** and **secret2.txt** via **FTP**.
- Checking if file contents can be captured.

📌 **Findings:**

- **SSH traffic is encrypted**, preventing attackers from easily extracting credentials or file contents.

---

5. FTP Traffic Analysis

- Using Wireshark to monitor FTP file transfers.
- Uploading & downloading text files between two machines.
- Checking if file contents appear in captured network traffic.

📌 **Findings:**

- **FTP transmits data in plain text**, making it vulnerable to interception.
- **SFTP (Secure FTP) or SCP** should be used instead.

Project Overview

This project demonstrates how weak passwords and unencrypted communication protocols can be exploited using tools like **Hashcat, John the Ripper, and Wireshark**. It covers **password cracking** (brute-force and dictionary attacks) and **network traffic analysis** (HTTP, SSH, and FTP).

Task 1 - Cracking passwords (brute-forcemethod) 1/3

For the following hashes, specify the type of hashing algorithm used, and then crack the password using the brute-force method.

Each password consists of up to 5 characters (only numbers).

1. 81dc9bdb52d04dc20036dbd8313ed055

2.021d0862bc76b0995927902ec697d97b5080341a53cd90b780f50fd5886f4160bbb9d4a573b76c23004c9b3a44ac95cfde45399e3357d1f651b556dfbd0d58f

3. 7aaa0f57

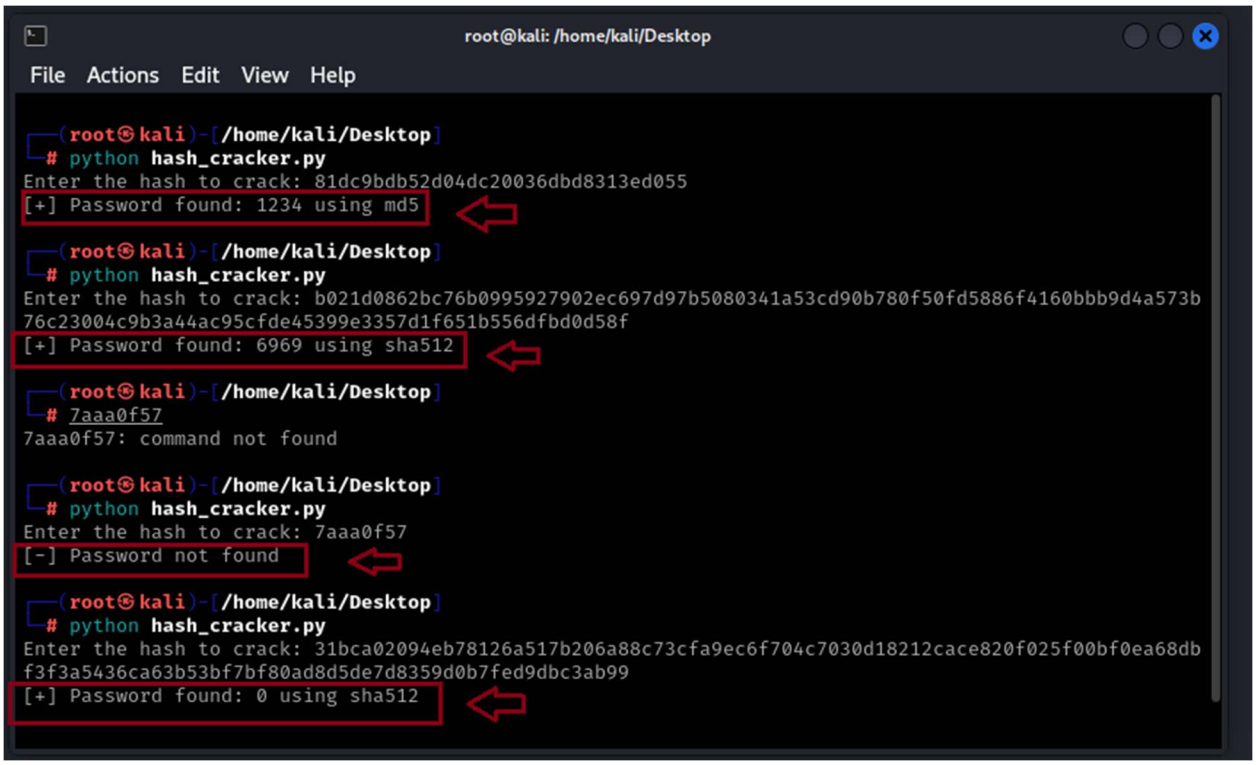
4.31bca02094eb78126a517b206a88c73cfa9ec6f704c7030d18212cace820f025f00bf0ea68dbf3f3a5436ca63b53bf7bf80ad8d5de7d8359d0b7fed9dbc3ab99

~/Desktop/hash_cracker.py - Mousepad

File Edit Search View Document Help

```python
import hashlib
from itertools import product

# List of possible hashing algorithms to try
hash_algorithms = ['md5', 'sha1', 'sha256', 'sha512']

# Function to hash a password using the specified algorithm
def hash_password(password, algorithm):
    hash_obj = hashlib.new(algorithm)
    hash_obj.update(password.encode('utf-8'))
    return hash_obj.hexdigest()

# Function to brute-force the password
def brute_force(hash_value):
    digits = '0123456789'
    for length in range(1, 6):  # Password lengths from 1 to 5
        for combination in product(digits, repeat=length):
            password = ''.join(combination)
            for algorithm in hash_algorithms:
                hashed = hash_password(password, algorithm)
                if hashed == hash_value:
                    print(f'[+] Password found: {password} using {algorithm}')
                    return

    print('[-] Password not found')

if __name__ == '__main__':
    hash_input = input('Enter the hash to crack: ').strip()
    brute_force(hash_input)

```

- The script asks you to enter a hash.
- It tries all possible combinations of digits (from 1 to 5 digits long) as potential passwords.
- For each potential password, it calculates the hash using different algorithms (MD5, SHA1, etc.).
- It compares the generated hash with the hash you entered.
- If a match is found, it prints the password and the algorithm used.
- If no match is found after trying all combinations, it reports that the password was not found.



Task 1 - Cracking passwords (brute-force method) 2/3

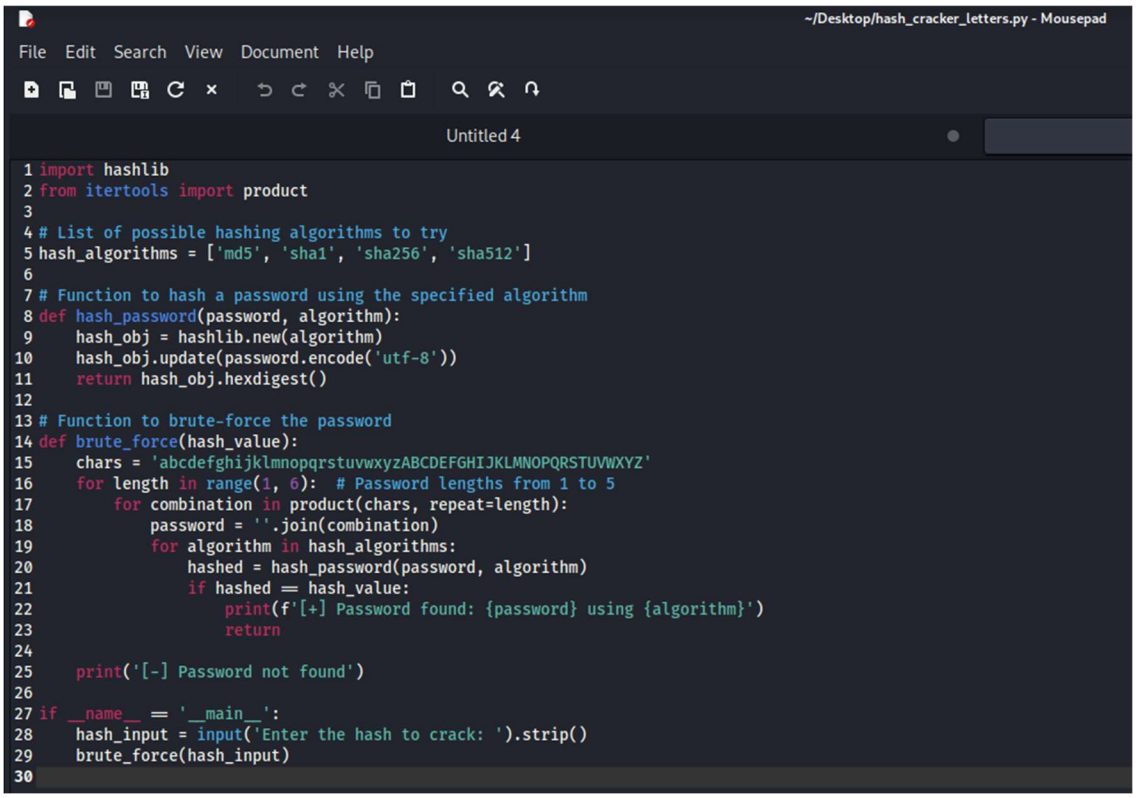
Each password consists of a maximum of 5 characters (upper and lower case letters).

1. 9e66d646cfb6c84d06a42ee1975ffaae90352bd016da18f51721e2042d9067dcb120accc5741

05b43139b6c9c887dda8202eff20cc4b98bad7b3be1e471b3aa5

2. 8a04bd2d079ee38f1af784317c4e2442625518780ccff3213feb2e207d2be42ca0760fd847618

4a004b71bcb5841db5cd0a546b9b8870f1cafee57991077c4a9



1. Setting Up

- The script starts by importing two tools:
  - `hashlib`: This tool knows how to create special codes called "hashes" from passwords. Hashes are like secret codes that are very hard to turn back into the original password.
  - `itertools.product`: This tool helps create all possible combinations of letters, kind of like a lock's combination wheel.

- It also has a list of different ways to make hashes (like MD5, SHA1, etc.). Think of these as different "secret code languages."

2. Making Hashes

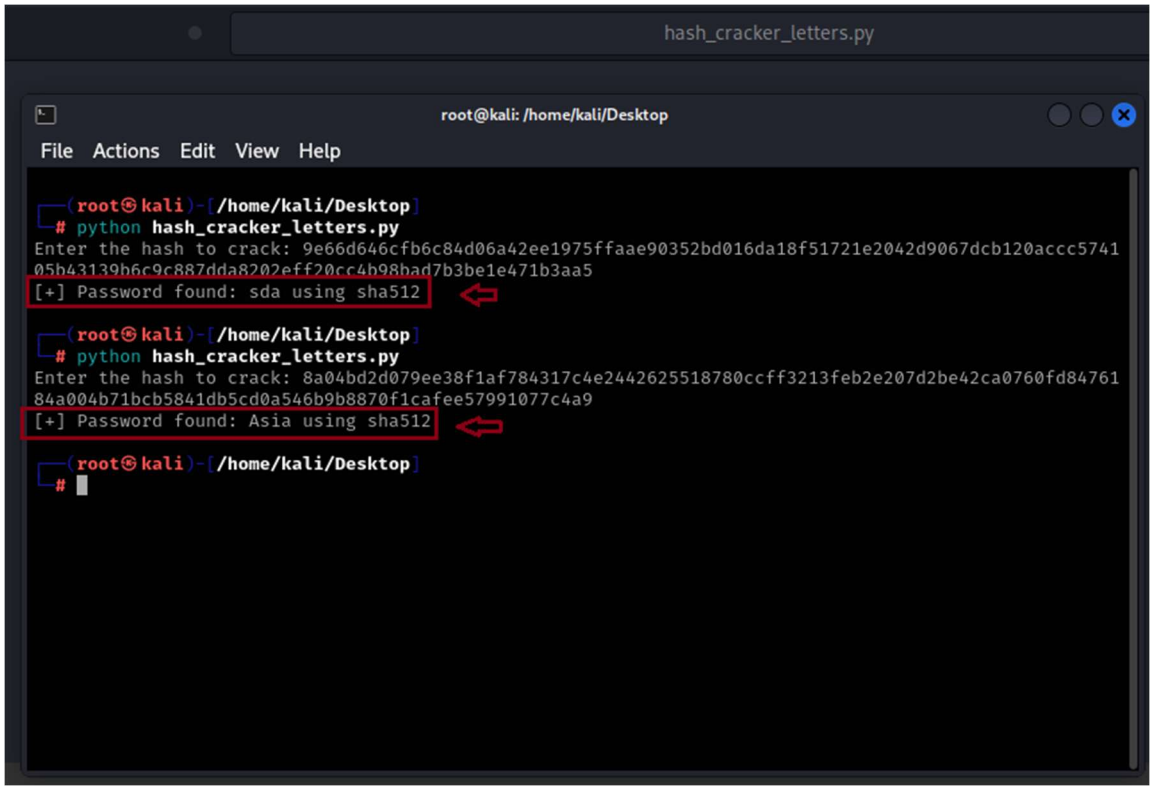
- There's a part called `hash_password`. It takes a password and a "secret code language" and creates a hash. It's like putting a password through a machine that spits out a secret code.

3. Guessing Passwords

- The main part is called `brute_force`. This is where the guessing happens.
- First, it decides to try passwords from 1 letter long to 5 letters long.
- Then, it uses `itertools.product` to create every possible combination of letters (both uppercase and lowercase) for each length. It's like trying every combination on a lock.
- For each combination, it creates a hash using all the different "secret code languages."
- It compares each hash it creates with the hash you gave it. If they match, it means it found the password!

4. Running the Script

- When you run the script, it asks you for the hash you want to crack.
- Then, it starts the `brute_force` part to do the guessing.
- If it finds a match, it tells you the password and how it made the hash.
- If it tries all the combinations and doesn't find a match, it tells you it couldn't crack the password.


hash_cracker_letters.py
root@kali: /home/kali/Desktop
File Actions Edit View Help
┌──(root㉿kali)-[/home/kali/Desktop]
└─# python hash_cracker_letters.py
Enter the hash to crack: 9e66d646cfb6c84d06a42ee1975ffaae90352bd016da18f51721e2042d9067dcb120accc5741
05b43139b6c9c887dda8202eff20cc4b98bad7b3be1e471b3aa5
[+] Password found: sda using sha512
┌──(root㉿kali)-[/home/kali/Desktop]
└─# python hash_cracker_letters.py
Enter the hash to crack: 8a04bd2d079ee38f1af784317c4e2442625518780ccff3213feb2e207d2be42ca0760fd84761
84a004b71bcb5841db5cd0a546b9b8870f1cafee57991077c4a9
[+] Password found: Asia using sha512
┌──(root㉿kali)-[/home/kali/Desktop]
└─#

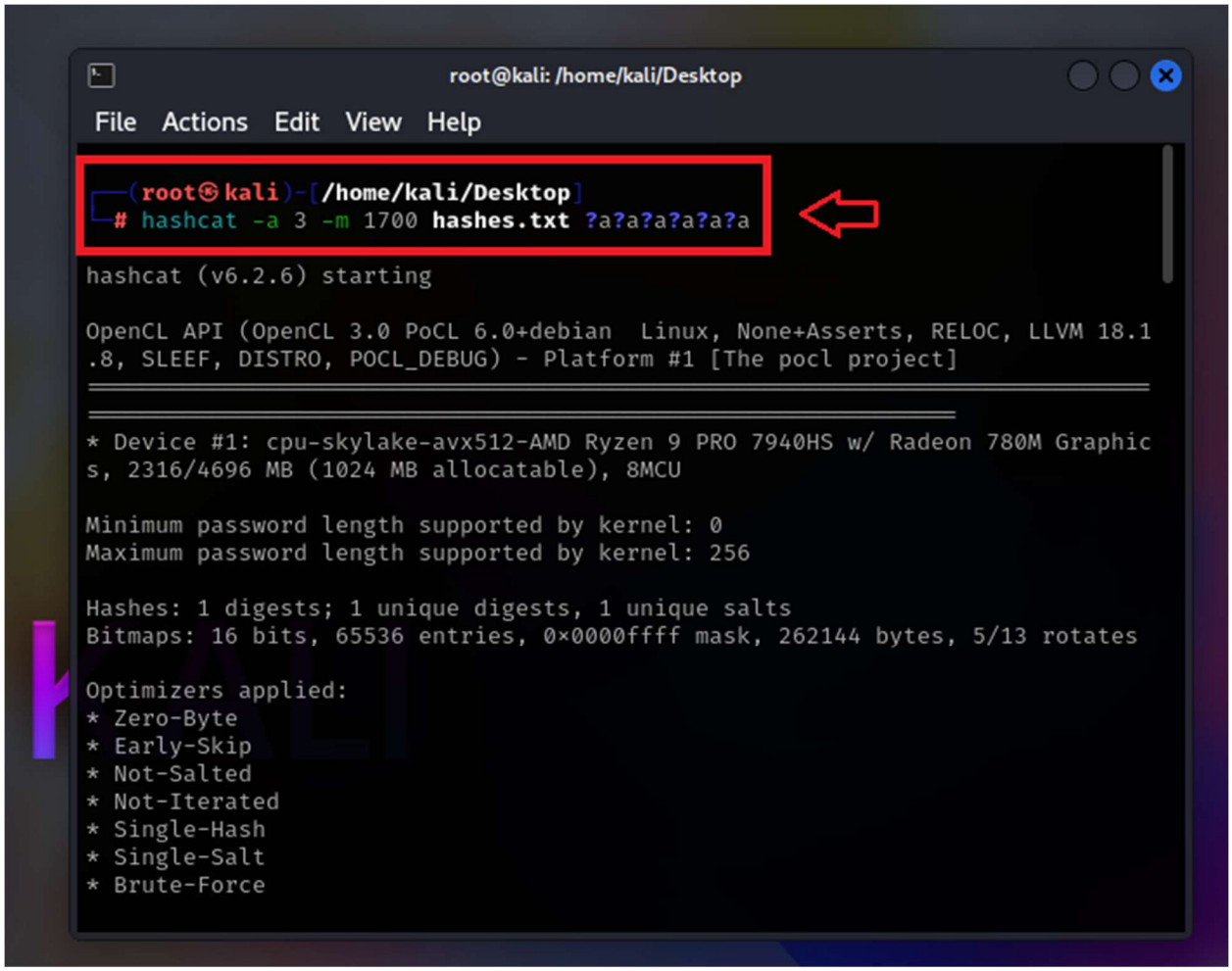
Task 1 - Cracking passwords (brute-force method) 3/3

The password consists of exactly 6 alphanumeric characters.

If possible, use the GPU or complete an after-class assignment.

1. 44d9886c0a57ddbfdb31aa936bd498bf2ab70f741ee47047851e768db953fc4e43f92be953

e205a3d1b3ab752ed90379444b651b582b0bc209a739a624e109da



The Command:

Bash

```
hashcat -a 3 -m 1700 hashes.txt ?a?a?a?a?a?a
```

- **`hashcat`:** This is the name of the program being run.
- **`-a 3`:** This specifies the attack mode. "3" corresponds to a "brute-force" attack, meaning `hashcat` will try every possible combination of characters within a defined set.

- **`-m 1700`:** This option indicates the hash type being targeted. "1700" is the identifier for "NTLM" hashes, which are commonly used in Windows systems.
- **`hashes.txt`:** This is the name of the file containing the NTLM hash(es) that `hashcat` will attempt to crack.
- **`?a?a?a?a?a?a`:** This is the mask that defines the character set and length of the passwords `hashcat` will generate.
    - `?a` represents all possible characters (lowercase and uppercase letters, numbers, and symbols).
    - The six `?a`s indicate that `hashcat` should try passwords of exactly six characters in length.

2. System Information:

The output below the command provides details about the system and `hashcat`'s configuration:

- **`hashcat (v6.2.6) starting`:** Shows the version of `hashcat` being used.
- **OpenCL API:** Indicates that `hashcat` is using OpenCL for hardware acceleration, allowing it to utilize the GPU for faster cracking.
- **Platform #1 [The pocl project]:** Specifies the OpenCL platform being used.
- **Device #1:** Provides details about the CPU and integrated GPU being used for cracking.
- **Minimum/Maximum password length:** Displays the password length limits supported by the cracking kernel.

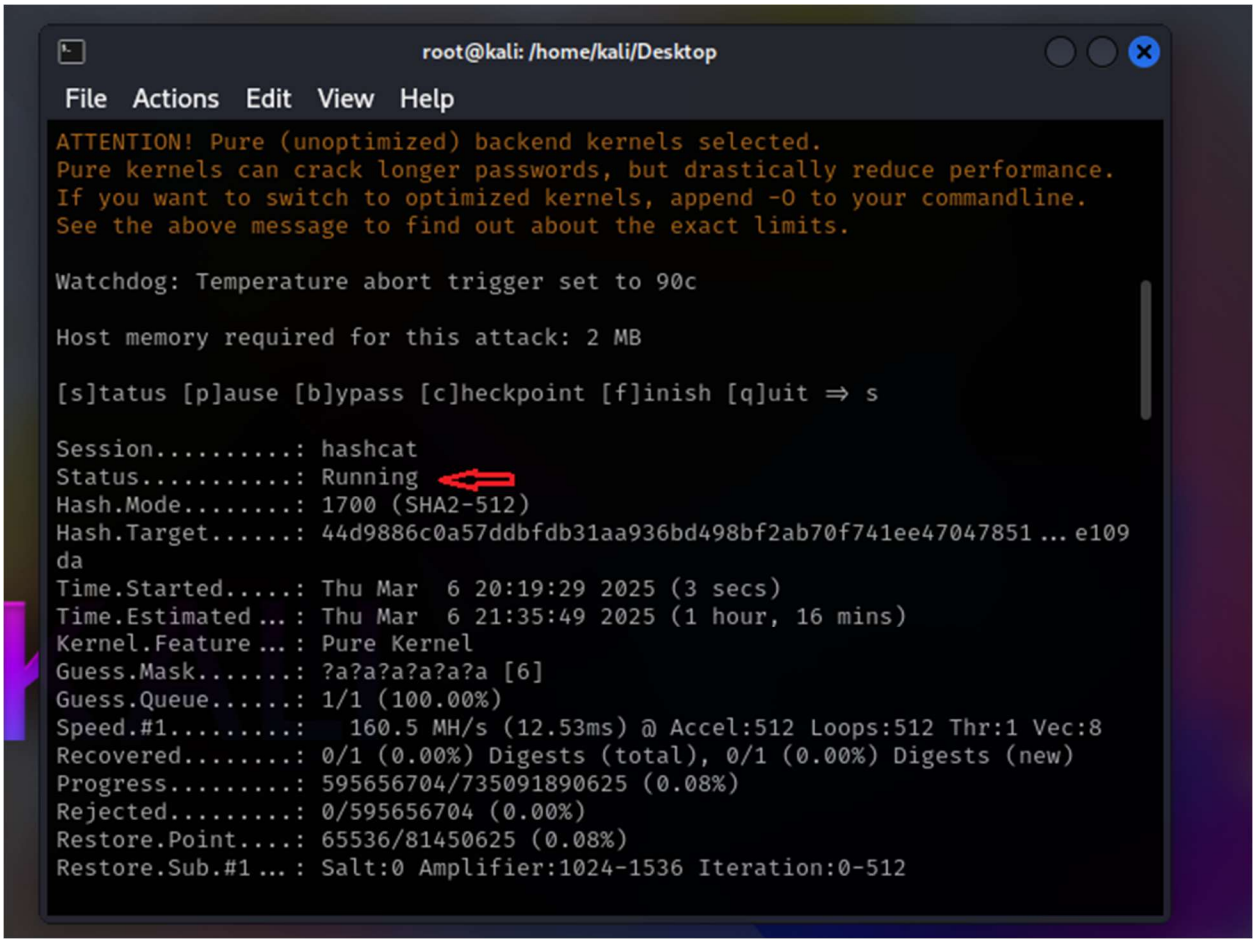
3. Hash Information:

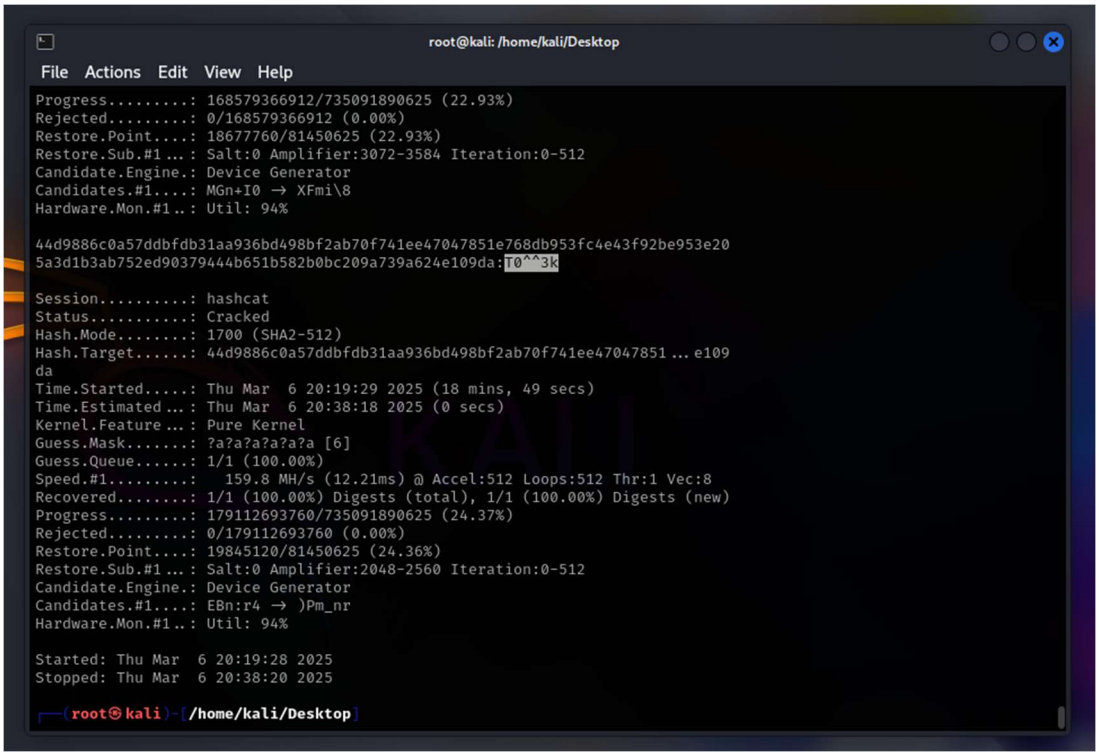
- **`Hashes: 1 digests; 1 unique digests, 1 unique salts`:** Indicates that there is one hash in the `hashes.txt` file, and it is a unique hash with a unique salt.

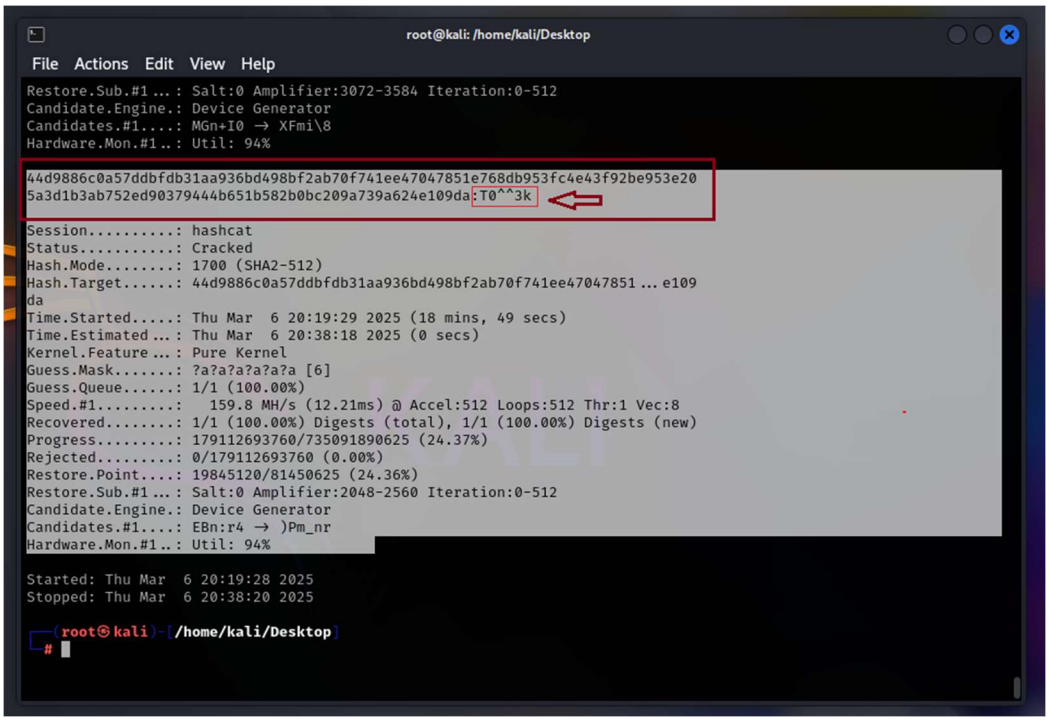
4. Bitmaps and Optimizers:

- **Bitmaps:** These are used for memory management during the cracking process.
- **Optimizers applied:** These are techniques `hashcat` uses to speed up the cracking process, such as skipping redundant calculations.

In summary, this screenshot shows the execution of `hashcat` to perform a brute-force attack on a single NTLM hash of a six-character password using all possible characters. It demonstrates the use of `hashcat`'s command-line options and the system information it provides during operation.


root@kali: /home/kali/Desktop
File Actions Edit View Help
ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.
Watchdog: Temperature abort trigger set to 90c
Host memory required for this attack: 2 MB
[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit ⇒ s
Session..........: hashcat
Status...........: Running
Hash.Mode........: 1700 (SHA2-512)
Hash.Target......: 44d9886c0a57ddbfdb31aa936bd498bf2ab70f741ee47047851 ... e109
da
Time.Started.....: Thu Mar 6 20:19:29 2025 (3 secs)
Time.Estimated ... : Thu Mar 6 21:35:49 2025 (1 hour, 16 mins)
Kernel.Feature ... : Pure Kernel
Guess.Mask.......: ?a?a?a?a?a?a [6]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 160.5 MH/s (12.53ms) @ Accel:512 Loops:512 Thr:1 Vec:8
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 595656704/735091890625 (0.08%)
Rejected.........: 0/595656704 (0.00%)
Restore.Point....: 65536/81450625 (0.08%)
Restore.Sub.#1 ... : Salt:0 Amplifier:1024-1536 Iteration:0-512


root@kali: /home/kali/Desktop
File Actions Edit View Help
Progress.....: 168579366912/735091890625 (22.93%)
Rejected.....: 0/168579366912 (0.00%)
Restore.Point....: 18677760/81450625 (22.93%)
Restore.Sub.#1...: Salt:0 Amplifier:3072-3584 Iteration:0-512
Candidate.Engine.: Device Generator
Candidates.#1....: MGn+I0 → XFmi\8
Hardware.Mon.#1..: Util: 94%
44d9886c0a57ddbfdb31aa936bd498bf2ab70f741ee47047851e768db953fc4e43f92be953e20
5a3d1b3ab752ed90379444b651b582b0bc209a739a624e109da:T0^^3k
Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 1700 (SHA2-512)
Hash.Target.....: 44d9886c0a57ddbfdb31aa936bd498bf2ab70f741ee47047851...e109
da
Time.Started.....: Thu Mar 6 20:19:29 2025 (18 mins, 49 secs)
Time.Estimated...: Thu Mar 6 20:38:18 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.....: ?a?a?a?a?a?a [6]
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 159.8 MH/s (12.21ms) @ Accel:512 Loops:512 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 179112693760/735091890625 (24.37%)
Rejected.....: 0/179112693760 (0.00%)
Restore.Point....: 19845120/81450625 (24.36%)
Restore.Sub.#1...: Salt:0 Amplifier:2048-2560 Iteration:0-512
Candidate.Engine.: Device Generator
Candidates.#1....: EBn:r4 →)Pm_nr
Hardware.Mon.#1..: Util: 94%
Started: Thu Mar 6 20:19:28 2025
Stopped: Thu Mar 6 20:38:20 2025
┌──(root㉿kali)-[/home/kali/Desktop]


root@kali: /home/kali/Desktop
File Actions Edit View Help
Restore.Sub.#1...: Salt:0 Amplifier:3072-3584 Iteration:0-512
Candidate.Engine.: Device Generator
Candidates.#1....: MGn+I0 → XFmi\8
Hardware.Mon.#1..: Util: 94%
44d9886c0a57ddbfdb31aa936bd498bf2ab70f741ee47047851e768db953fc4e43f92be953e20
5a3d1b3ab752ed90379444b651b582b0bc209a739a624e109da:T0^^3k
Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 1700 (SHA2-512)
Hash.Target.....: 44d9886c0a57ddbfdb31aa936bd498bf2ab70f741ee47047851...e109
da
Time.Started.....: Thu Mar 6 20:19:29 2025 (18 mins, 49 secs)
Time.Estimated...: Thu Mar 6 20:38:18 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.....: ?a?a?a?a?a?a [6]
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 159.8 MH/s (12.21ms) @ Accel:512 Loops:512 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 179112693760/735091890625 (24.37%)
Rejected.....: 0/179112693760 (0.00%)
Restore.Point....: 19845120/81450625 (24.36%)
Restore.Sub.#1...: Salt:0 Amplifier:2048-2560 Iteration:0-512
Candidate.Engine.: Device Generator
Candidates.#1....: EBn:r4 →)Pm_nr
Hardware.Mon.#1..: Util: 94%
Started: Thu Mar 6 20:19:28 2025
Stopped: Thu Mar 6 20:38:20 2025
┌──(root㉿kali)-[/home/kali/Desktop]
└─#

Task 2 - Cracking passwords (dictionary method) 1/2

The entries are from the rockyou-50 dictionary.

1. 9fd8301ac24fb88e65d9d7cd1dd1b1ec

2. 7f9a6871b86f40c330132c4fc42cda59

3. 6104df369888589d6dbea304b59a32d4

4. 276f8db0b86edaa7fc805516c852c889

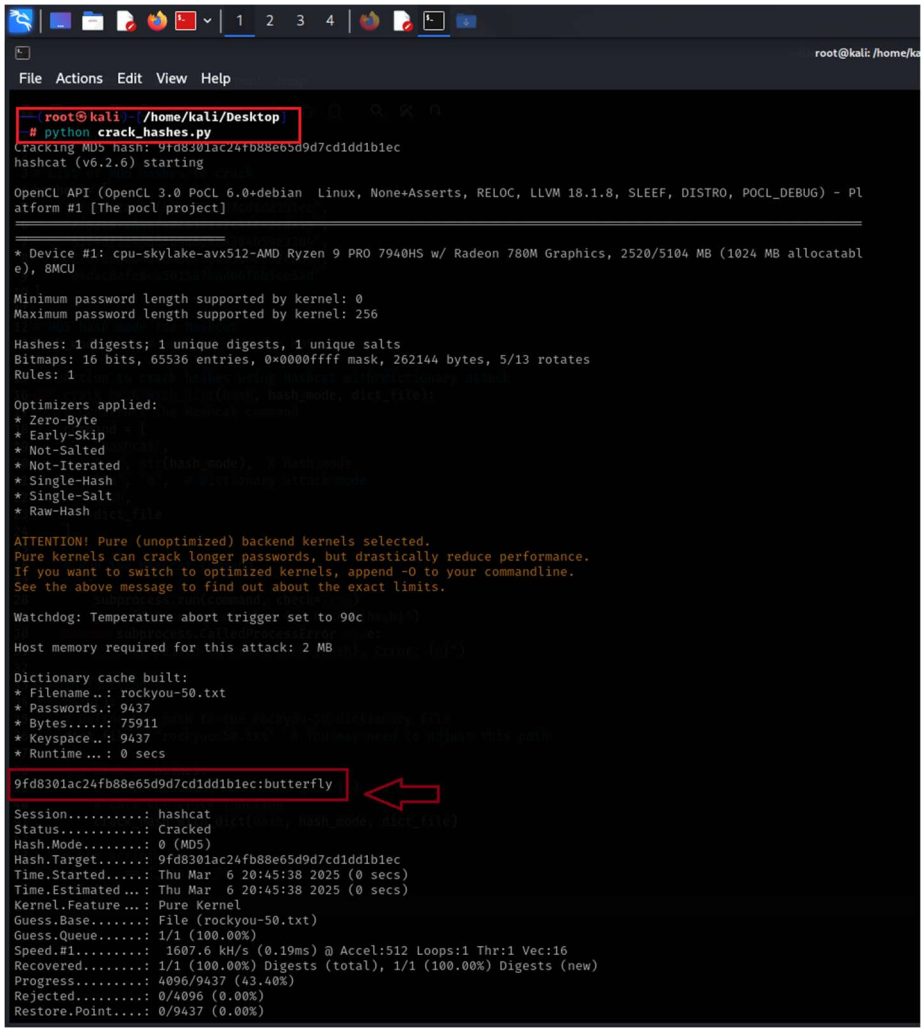
5. 04dac8afe0ca501587bad66f6b5ce5ad

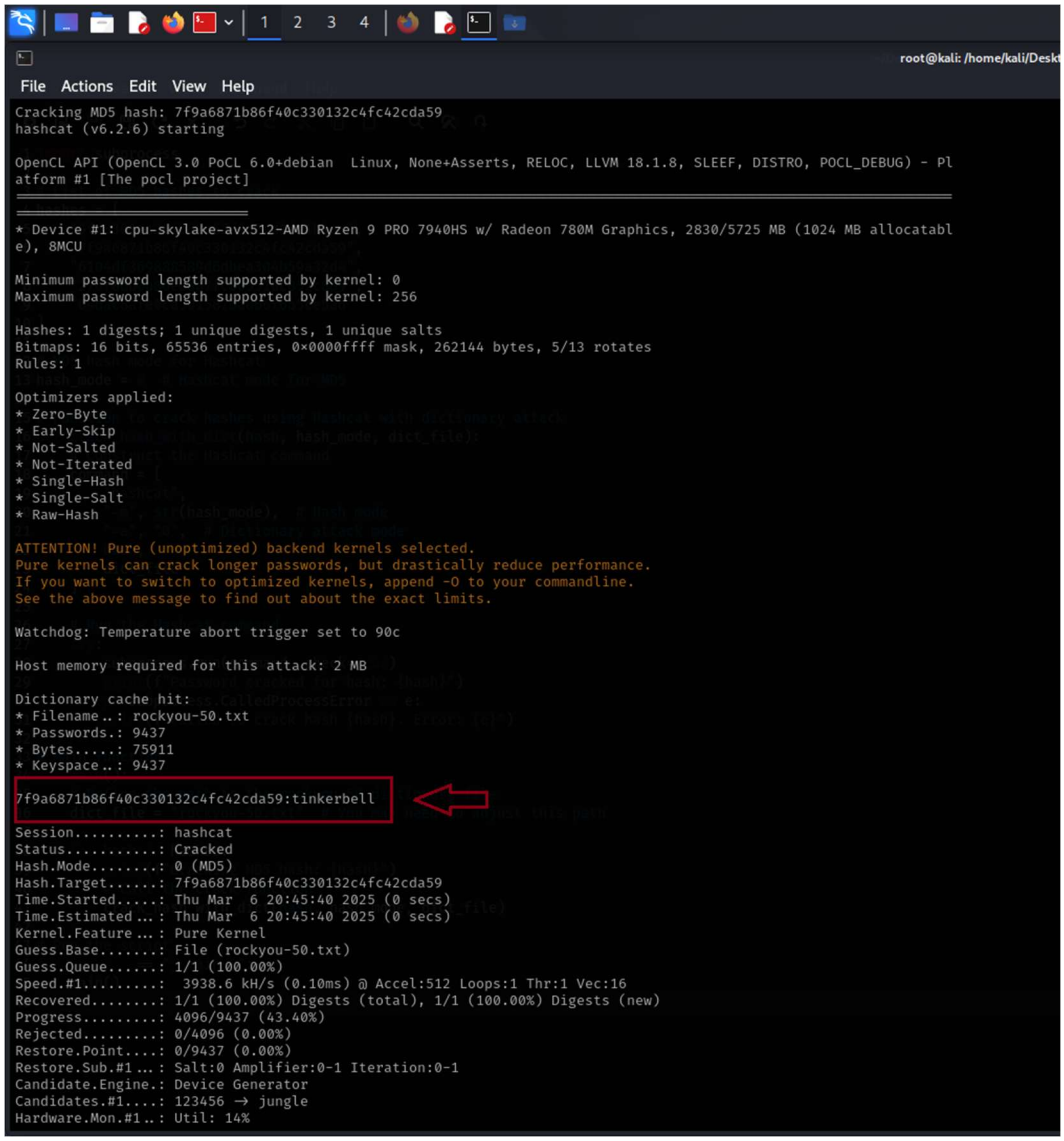
1. The script has a list of MD5 hashes it wants to crack.
2. It uses the `hashcat` tool to do the actual cracking.
3. It tells `hashcat` to use a dictionary attack, which means it will try passwords from a wordlist (the `rockyou-50.txt` file).
4. For each hash, it runs `hashcat` and checks if it finds the password.
5. If `hashcat` finds the password, the script prints a message.
6. If `hashcat` fails to crack a hash, the script prints an error message.

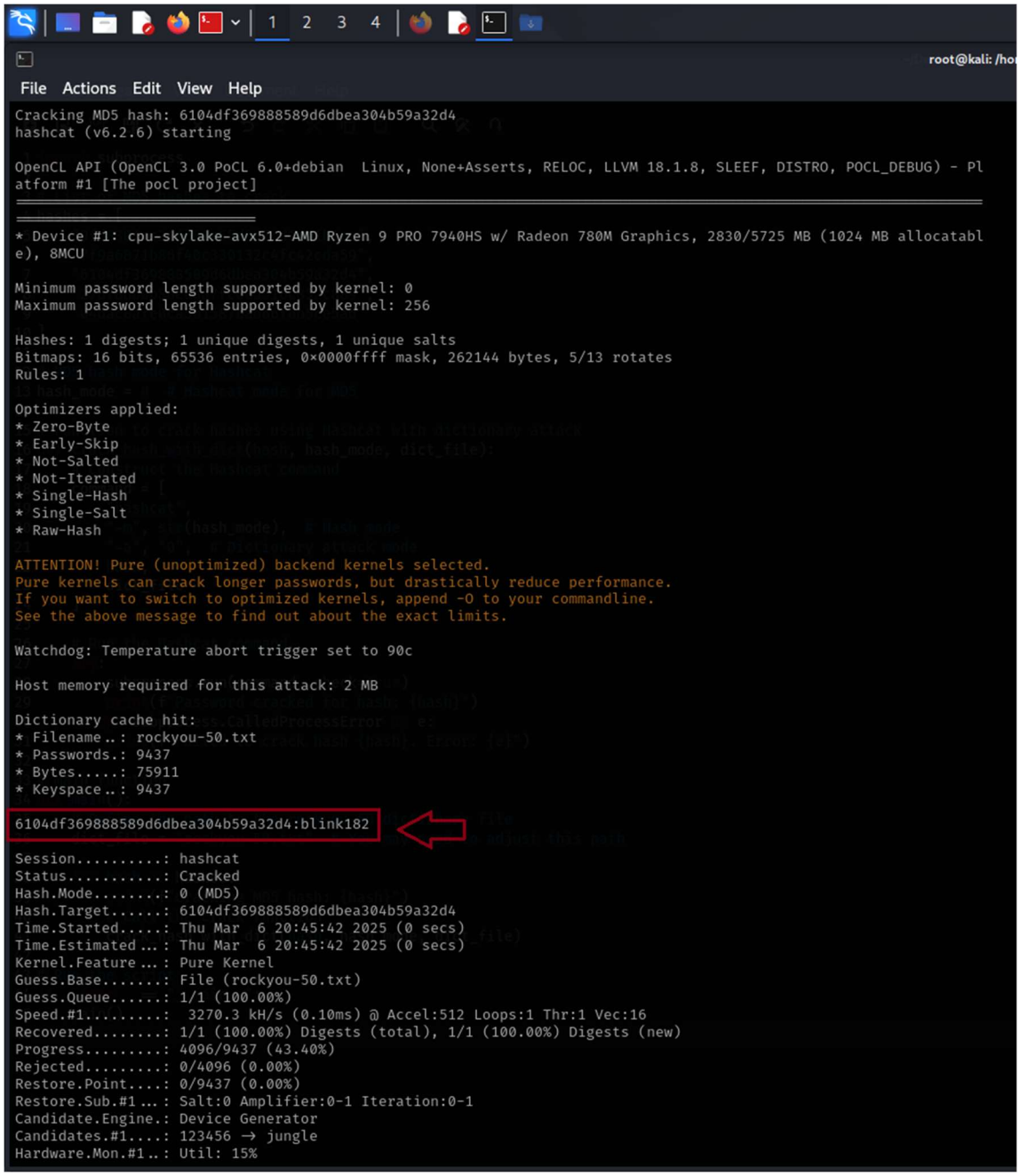
Important Considerations:

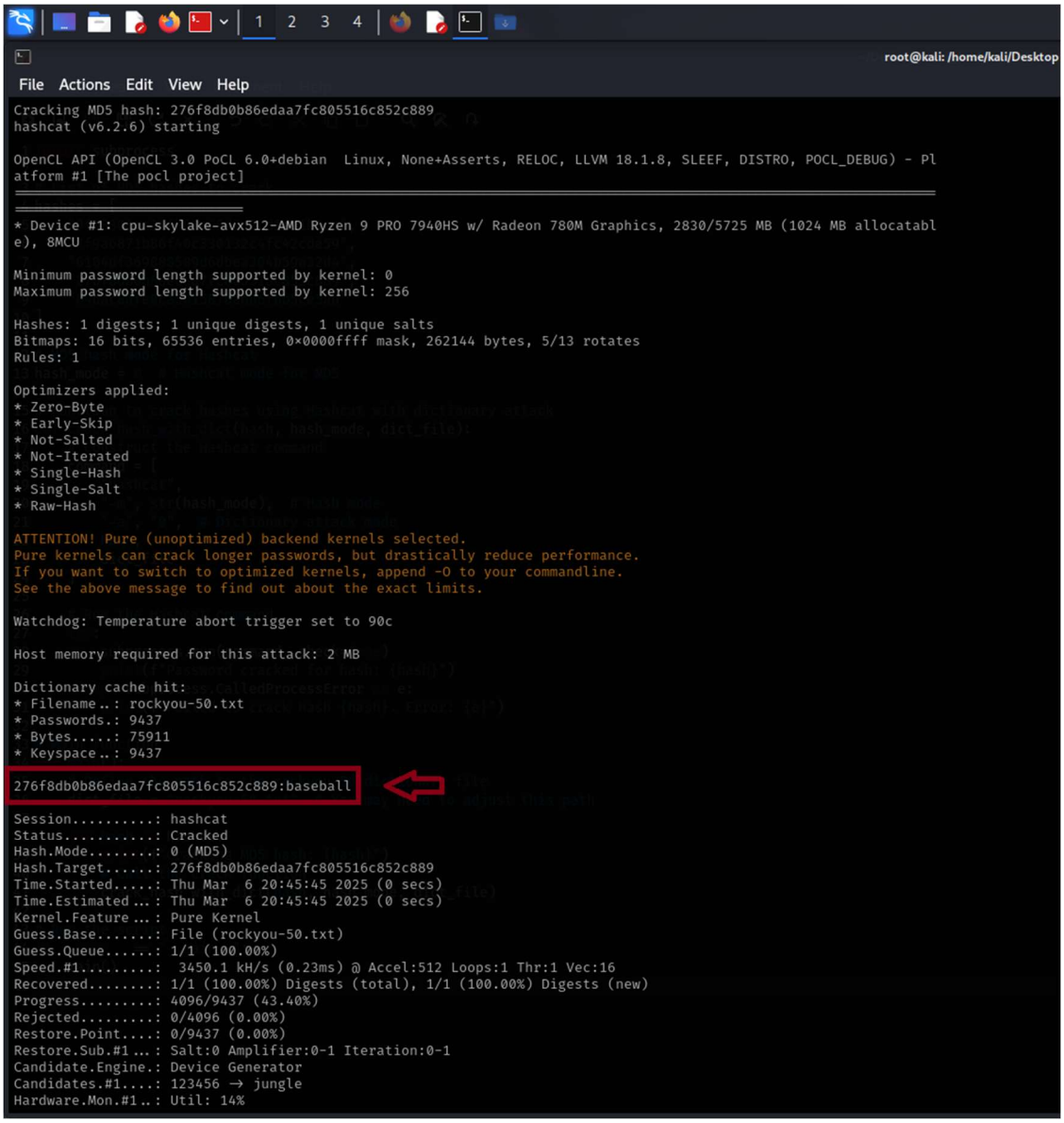
- **`hashcat` Installation:** You need to have `hashcat` installed on your system for this script to work.
- **Dictionary File:** The `rockyou-50.txt` file is a common password dictionary. You might need to download it separately or use a different dictionary file. **Make sure the path to the dictionary file in the script is correct.**
- **Performance:** Dictionary attacks can be effective for cracking weak passwords, but they might not work for strong or complex passwords.

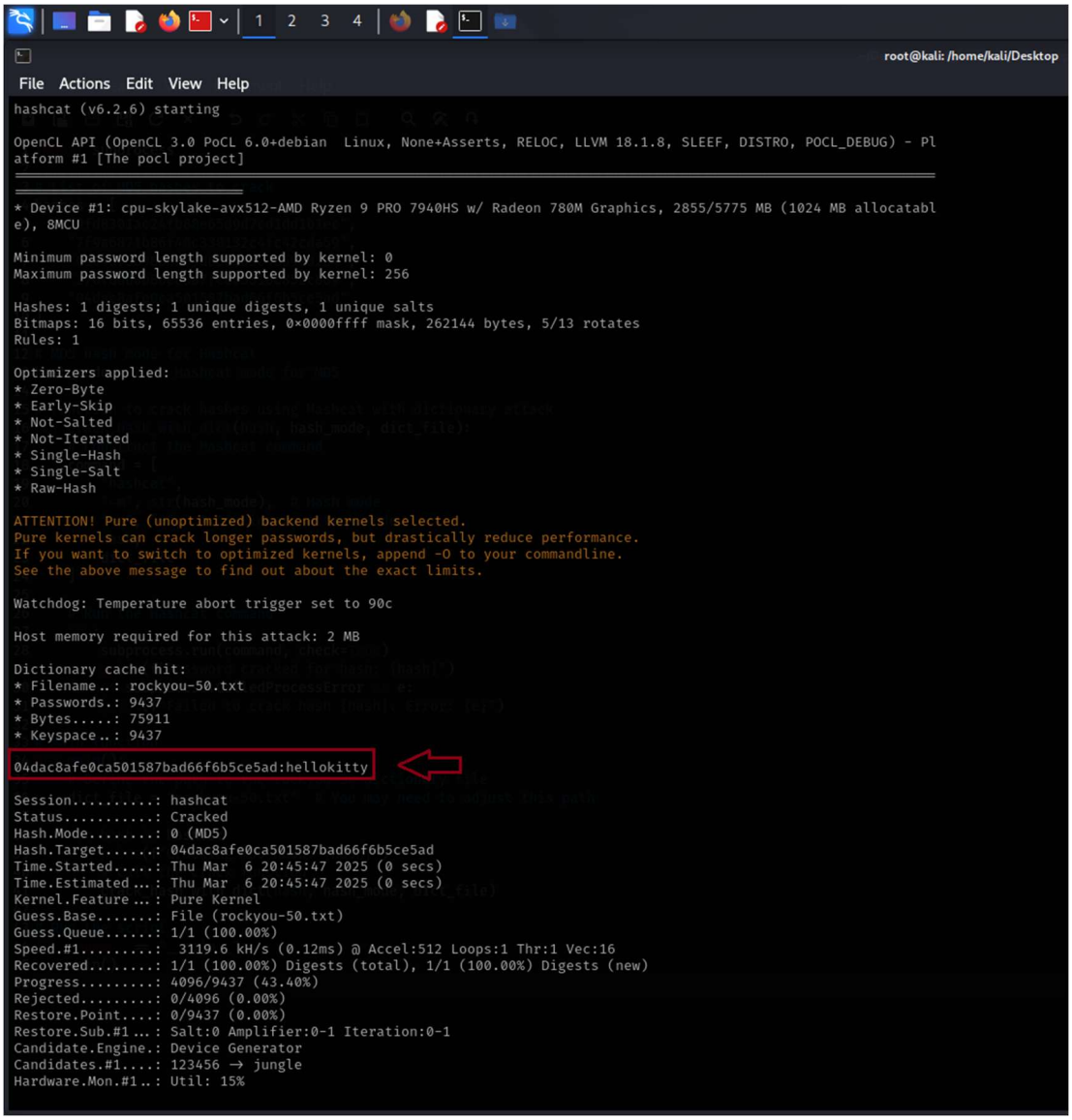
This script demonstrates how to automate `hashcat` using Python for dictionary attacks on MD5 hashes. Remember to use it ethically and responsibly.


File Actions Edit View Help
root@kali: /home/kali/Desktop
python crack_hashes.py
Cracking MD5 hash: 9fd8301ac24fb88e65d9d7cd1dd1b1ec
hashcat (v6.2.6) starting
OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, LLVM 18.1.8, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
* Device #1: cpu-skylake-avx512-AMD Ryzen 9 PRO 7940HS w/ Radeon 780M Graphics, 2520/5104 MB (1024 MB allocatable), 8MCU
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1
Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash
ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.
Watchdog: Temperature abort trigger set to 90c
Host memory required for this attack: 2 MB
Dictionary cache built:
* Filename..: rockyou-50.txt
* Passwords.: 9437
* Bytes.....: 75911
* Keyspace..: 9437
* Runtime...: 0 secs
9fd8301ac24fb88e65d9d7cd1dd1b1ec:butterfly
Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: 9fd8301ac24fb88e65d9d7cd1dd1b1ec
Time.Started.....: Thu Mar 6 20:45:38 2025 (0 secs)
Time.Estimated...: Thu Mar 6 20:45:38 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (rockyou-50.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 1607.6 kH/s (0.19ms) @ Accel:512 Loops:1 Thr:1 Vec:16
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 4096/9437 (43.40%)
Rejected.........: 0/4096 (0.00%)
Restore.Point....: 0/9437 (0.00%)


root@kali: /home/kali/Desk
File Actions Edit View Help
Cracking MD5 hash: 7f9a6871b86f40c330132c4fc42cda59
hashcat (v6.2.6) starting
OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, LLVM 18.1.8, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
* Device #1: cpu-skylake-avx512-AMD Ryzen 9 PRO 7940HS w/ Radeon 780M Graphics, 2830/5725 MB (1024 MB allocatable), 8MCU
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1
Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash
ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.
Watchdog: Temperature abort trigger set to 90c
Host memory required for this attack: 2 MB
Dictionary cache hit:
* Filename..: rockyou-50.txt
* Passwords.: 9437
* Bytes.....: 75911
* Keyspace..: 9437
7f9a6871b86f40c330132c4fc42cda59:tinkerbell
Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: 7f9a6871b86f40c330132c4fc42cda59
Time.Started.....: Thu Mar 6 20:45:40 2025 (0 secs)
Time.Estimated...: Thu Mar 6 20:45:40 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (rockyou-50.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 3938.6 kH/s (0.10ms) @ Accel:512 Loops:1 Thr:1 Vec:16
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 4096/9437 (43.40%)
Rejected.........: 0/4096 (0.00%)
Restore.Point....: 0/9437 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 123456 → jungle
Hardware.Mon.#1..: Util: 14%


1 2 3 4
root@kali: /ho
File Actions Edit View Help
Cracking MD5 hash: 6104df369888589d6dbea304b59a32d4
hashcat (v6.2.6) starting
OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, LLVM 18.1.8, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
* Device #1: cpu-skylake-avx512-AMD Ryzen 9 PRO 7940HS w/ Radeon 780M Graphics, 2830/5725 MB (1024 MB allocatable), 8MCU
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0×0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1
Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash
ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.
Watchdog: Temperature abort trigger set to 90c
Host memory required for this attack: 2 MB
Dictionary cache hit:
* Filename..: rockyou-50.txt
* Passwords.: 9437
* Bytes.....: 75911
* Keyspace..: 9437
6104df369888589d6dbea304b59a32d4:blink182
Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 0 (MD5)
Hash.Target.....: 6104df369888589d6dbea304b59a32d4
Time.Started.....: Thu Mar 6 20:45:42 2025 (0 secs)
Time.Estimated...: Thu Mar 6 20:45:42 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (rockyou-50.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 3270.3 kH/s (0.10ms) @ Accel:512 Loops:1 Thr:1 Vec:16
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 4096/9437 (43.40%)
Rejected.....: 0/4096 (0.00%)
Restore.Point....: 0/9437 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 123456 → jungle
Hardware.Mon.#1..: Util: 15%


root@kali: /home/kali/Desktop
File Actions Edit View Help
Cracking MD5 hash: 276f8db0b86edaa7fc805516c852c889
hashcat (v6.2.6) starting
OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, LLVM 18.1.8, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
* Device #1: cpu-skylake-avx512-AMD Ryzen 9 PRO 7940HS w/ Radeon 780M Graphics, 2830/5725 MB (1024 MB allocatable), 8MCU
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0×0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1
Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash
ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.
Watchdog: Temperature abort trigger set to 90c
Host memory required for this attack: 2 MB
Dictionary cache hit:
* Filename..: rockyou-50.txt
* Passwords.: 9437
* Bytes.....: 75911
* Keyspace..: 9437
276f8db0b86edaa7fc805516c852c889:baseball
Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: 276f8db0b86edaa7fc805516c852c889
Time.Started.....: Thu Mar 6 20:45:45 2025 (0 secs)
Time.Estimated...: Thu Mar 6 20:45:45 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (rockyou-50.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 3450.1 kH/s (0.23ms) @ Accel:512 Loops:1 Thr:1 Vec:16
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 4096/9437 (43.40%)
Rejected.........: 0/4096 (0.00%)
Restore.Point....: 0/9437 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 123456 → jungle
Hardware.Mon.#1..: Util: 14%


root@kali: /home/kali/Desktop
File Actions Edit View Help
hashcat (v6.2.6) starting
OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, LLVM 18.1.8, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
* Device #1: cpu-skylake-avx512-AMD Ryzen 9 PRO 7940HS w/ Radeon 780M Graphics, 2855/5775 MB (1024 MB allocatable), 8MCU
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0×0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1
Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash
ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.
Watchdog: Temperature abort trigger set to 90c
Host memory required for this attack: 2 MB
Dictionary cache hit:
* Filename..: rockyou-50.txt
* Passwords.: 9437
* Bytes.....: 75911
* Keyspace..: 9437
04dac8afe0ca501587bad66f6b5ce5ad:hellokitty
Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: 04dac8afe0ca501587bad66f6b5ce5ad
Time.Started.....: Thu Mar 6 20:45:47 2025 (0 secs)
Time.Estimated...: Thu Mar 6 20:45:47 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (rockyou-50.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 3119.6 kH/s (0.12ms) @ Accel:512 Loops:1 Thr:1 Vec:16
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 4096/9437 (43.40%)
Rejected.........: 0/4096 (0.00%)
Restore.Point....: 0/9437 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 123456 → jungle
Hardware.Mon.#1..: Util: 15%

Task 2 - Cracking passwords (dictionary method) 2/2

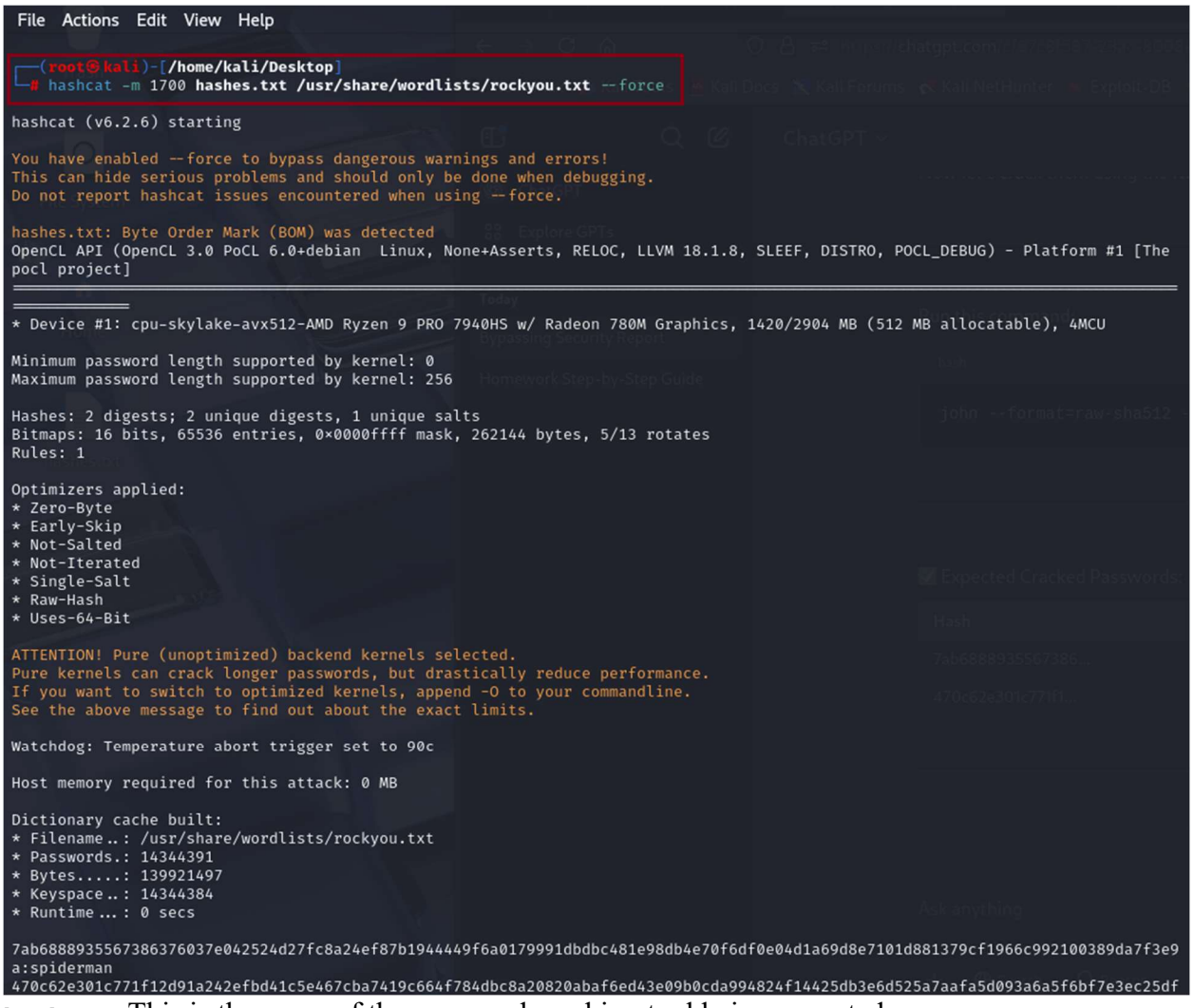
The entries are from the rockyou-50 dictionary.

1. 7ab6888935567386376037e042524d27fc8a24ef87b1944449f6a0179991dbdbc481e98db4

e70f6df0e04d1a69d8e7101d881379cf1966c992100389da7f3e9a

2. 470c62e301c771f12d91a242efbd41c5e467cba7419c664f784dbc8a20820abaf6ed43e09b0c

da994824f14425db3e6d525a7aafa5d093a6a5f6bf7e3ec25dfa



- **`hashcat`:** This is the name of the password cracking tool being executed.
- **`-m 1700`:** This option specifies the hash type. "1700" corresponds to NTLM hashes, commonly used in Windows systems.
- **`hashes.txt`:** This is the file containing the NTLM hashes that `hashcat` will attempt to crack.

- **`/usr/share/wordlists/rockyou.txt`:** This is the path to the dictionary file (`rockyou.txt`), which contains a list of potential passwords. `hashcat` will try each word in this file as a potential password.
- **`--force`:** This option forces `hashcat` to bypass some warnings and potential errors. This is generally discouraged except for debugging, as it can mask serious problems.

2. Output and Information:

- **`hashcat (v6.2.6) starting`:** Indicates the version of `hashcat` being used.
- **Warnings:** The screenshot shows warnings about using `--force` and the presence of a Byte Order Mark (BOM) in the `hashes.txt` file.
- **OpenCL API and Device Information:** Displays information about the OpenCL API being used for hardware acceleration and the CPU/GPU device being utilized for cracking.
- **Hash Information:** Shows that there are 2 hashes being targeted, they are unique, and one salt is being used.
- **Dictionary Information:** Provides details about the `rockyou.txt` dictionary file, including the number of passwords and bytes in the file.
- **Cracked Hashes:** The most important part is the output showing the cracked hashes:
    - `7ab6888935567386376037604 2524d27fc8a24ef87b1944449f6a0179991dbdbc481e98db4e70f6df0e04d1a69d8e7101d881379cf1966c992100389da7f3e9a:spiderman`
    - `470c62e301c771f12d91a242efbd41c5e467cba7419c664f784dbc8a20820abaf6ed43e09b0cda994824f14425db3e6d525a7aafa5d093a6a5f6bf7e3ec25df`
    - This indicates that the password for the first hash was "spiderman". The second hash was also cracked, but the password is not shown in this screenshot.

In essence, this screenshot shows the successful cracking of two NTLM hashes using `hashcat` and a dictionary attack with the `rockyou.txt` wordlist.

```
ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 0 MB

Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344391
* Bytes.....: 139921497
* Keyspace..: 14344384
* Runtime...: 0 secs

7ab688893556738637e042524d27fc8a24ef87b1944449f6a0179991dbdbc481e98db4e70f6df0e04d1a69d8e7101d881379cf1966c992100389da7f3e9
a:spiderman
470c62e301c771f12d91a242efbd41c5e467cba7419c664f784dbc8a20820abaf6ed43e09b0cda994824f14425db3e6d525a7aafa5d093a6a5f6bf7e3ec25df
a:rockstar

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1700 (SHA2-512)
Hash.Target......: hashes.txt
Time.Started.....: Wed Mar  5 21:16:32 2025, (0 secs)
Time.Estimated...: Wed Mar  5 21:16:32 2025, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    21332 H/s (0.17ms) @ Accel:256 Loops:1 Thr:1 Vec:8
Recovered........: 2/2 (100.00%) Digests (total), 2/2 (100.00%) Digests (new)
Progress.........: 1024/14344384 (0.01%)
Rejected.........: 0/1024 (0.00%)
Restore.Point....: 0/14344384 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 123456 → bethany
Hardware.Mon.#1..: Util: 33%

Started: Wed Mar  5 21:16:10 2025
Stopped: Wed Mar  5 21:16:33 2025
```

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# john --format=raw-sha512 --wordlist=/usr/share/wordlists/rockyou.txt hashes.txt

Using default input encoding: UTF-8
Loaded 2 password hashes with no different salts (Raw-SHA512 [SHA512 512/512 AVX512BW 8x])
Warning: poor OpenMP scalability for this hash type, consider --fork=4
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
rockstar         (?)
spiderman        (?)
2g 0:00:00:00 DONE (2025-03-05 21:15) 66.66g/s 273066p/s 273066c/s 546133C/s 123456..total90
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

┌──(root㉿kali)-[/home/kali/Desktop]
```

Task 3 - Analyzing HTTP traffic

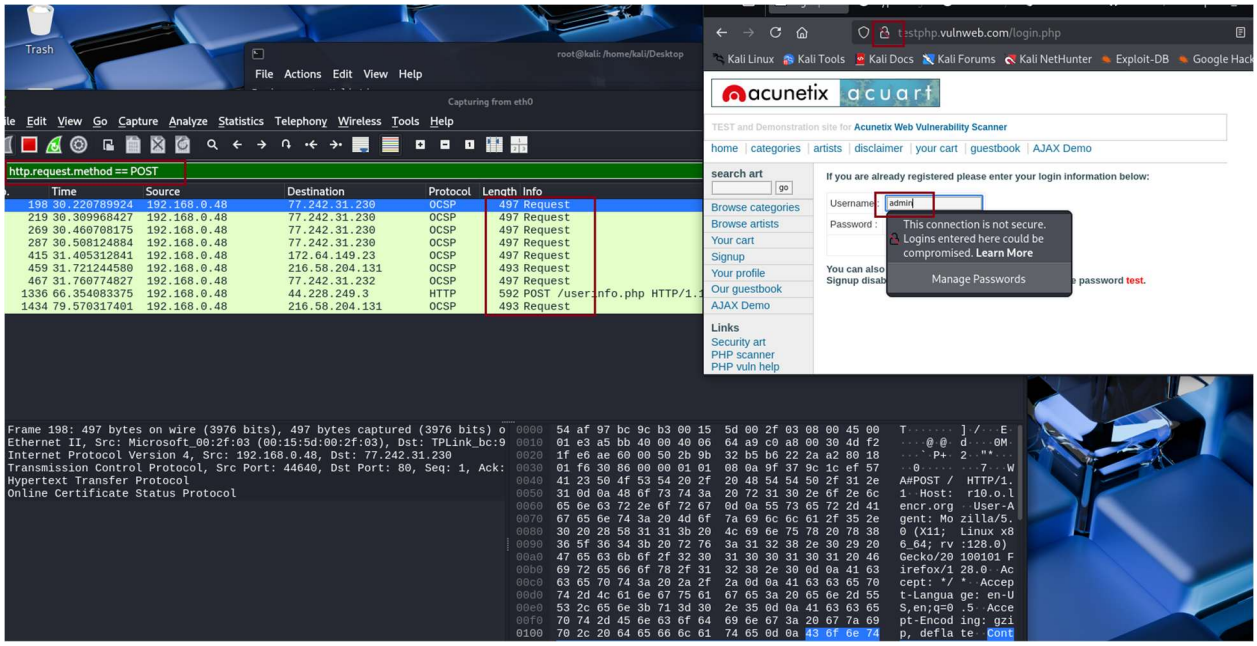
1. Start monitoring network traffic (Wireshark utility).

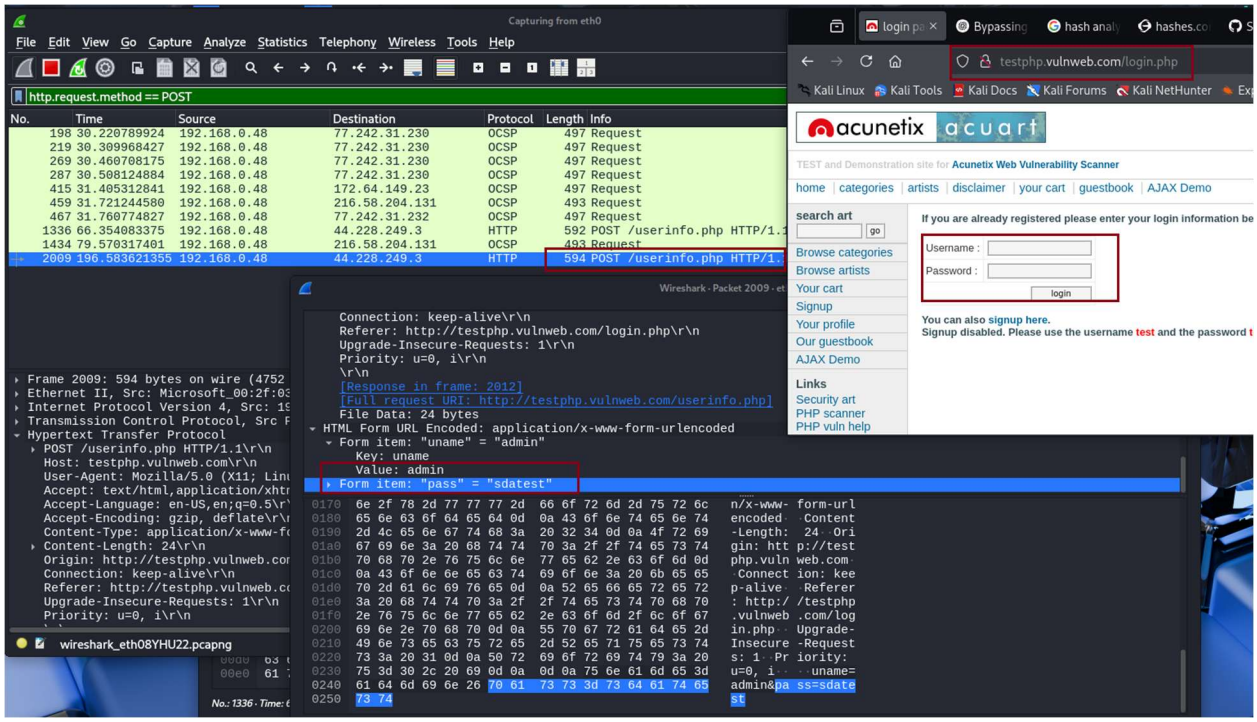
2. In the browser, connect to http://testphp.vulnweb.com/login.php

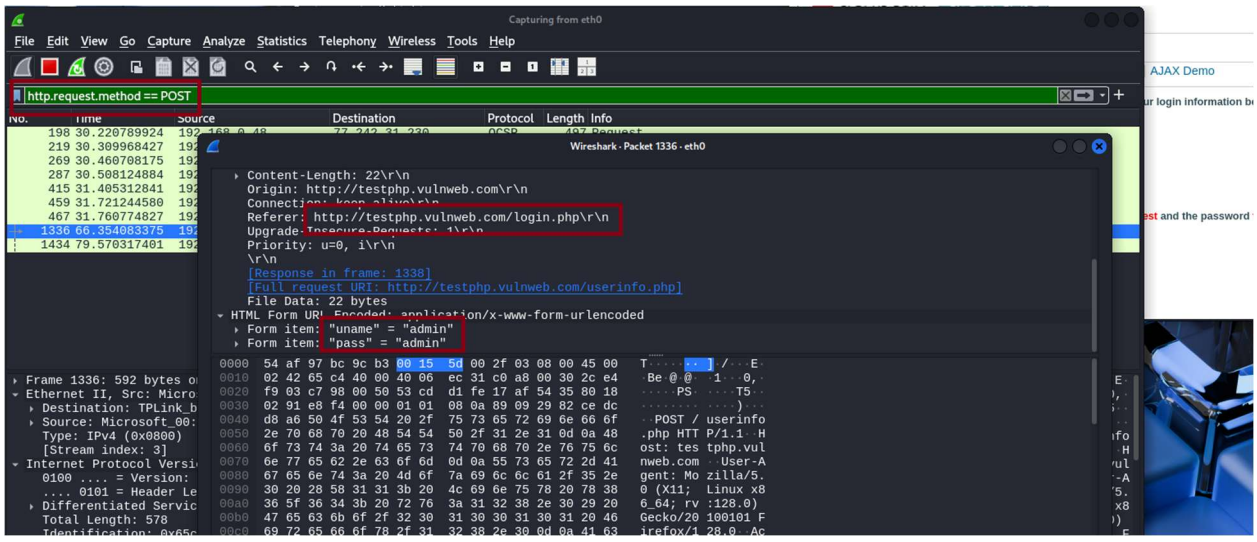
3. Make an attempt to log in (any data).

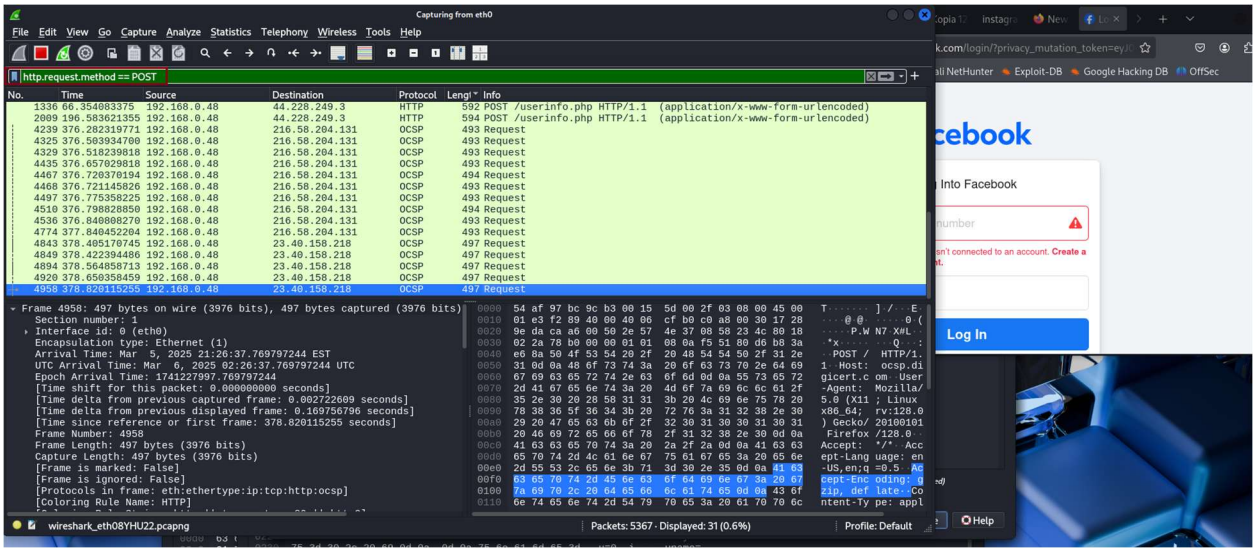
4. Find your login details in the saved traffic.

For comparison, repeat the exercise with logging in to e.g. Facebook (also any false login details)




Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http.request.method == POST
No. Time Source Destination Protocol Length Info
2009 196.583621355 192.168.0.48 44.228.249.3 HTTP 594 POST /userinfo.php HTTP/1.1
Wireshark · Packet 2009 · et
Connection: keep-alive\r\n
Referer: http://testphp.vulnweb.com/login.php\r\n
Upgrade-Insecure-Requests: 1\r\n
Priority: u=0, i\r\n
\r\n
[Response in frame: 2012]
[Full request URI: http://testphp.vulnweb.com/userinfo.php]
File Data: 24 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
Form item: "uname" = "admin"
Key: uname
Value: admin
Form item: "pass" = "sdatest"
wireshark_eth08YHU22.pcapng
testphp.vulnweb.com/login.php
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter
acunetix acuart
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Links
Security art
PHP scanner
PHP vuln help
Username :
Password :
login
You can also signup here.
Signup disabled. Please use the username test and the password t


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http.request.method == POST
Wireshark · Packet 1336 · eth0
Content-Length: 22\r\n
Origin: http://testphp.vulnweb.com\r\n
Referer: http://testphp.vulnweb.com/login.php\r\n
Priority: u=0, i\r\n
\r\n
[Response in frame: 1338]
[Full request URI: http://testphp.vulnweb.com/userinfo.php]
File Data: 22 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
Form item: "uname" = "admin"
Form item: "pass" = "admin"



Website	Protocol	Visible Password
vulnweb.com	HTTP	✅ Yes
Facebook	HTTPS	❌ No

Key Differences and Why They Matter

- **HTTP (Vulnerable Website):**
  - Data is sent in plain text.
  - Anyone monitoring the network can see your login credentials.
  - This is highly insecure.
- **HTTPS (Facebook):**
  - Data is encrypted using SSL/TLS.
  - Even if someone intercepts the traffic, they can't read the content without the decryption key.
  - This is the standard for secure communication on the web.

Task 4 - SSH traffic analysis

Environment: Kali Linux, SDA machine from project 1

1. Start monitoring network traffic (Wireshark utility).

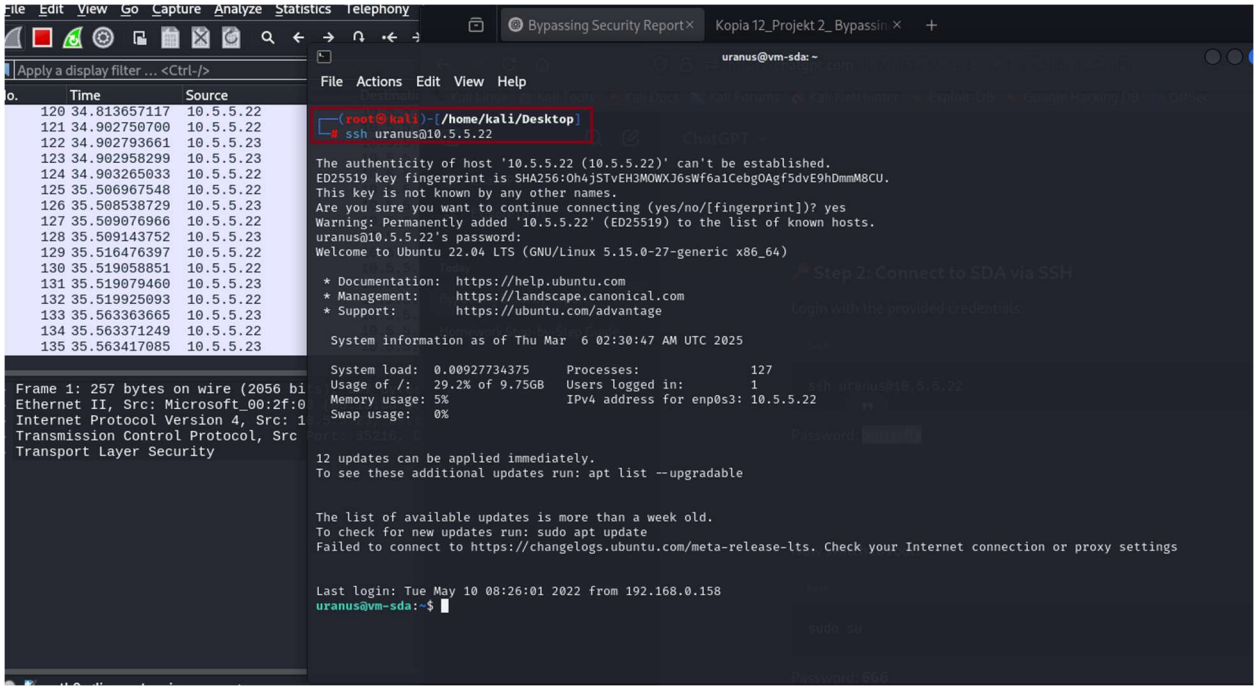
2. Establish a connection between Kalim and SDA via SSH.

3. Create secret1.txt and secret2.txt files with secret passwords.

4. Edit the vsFTPd configuration to allow file upload via FTP.

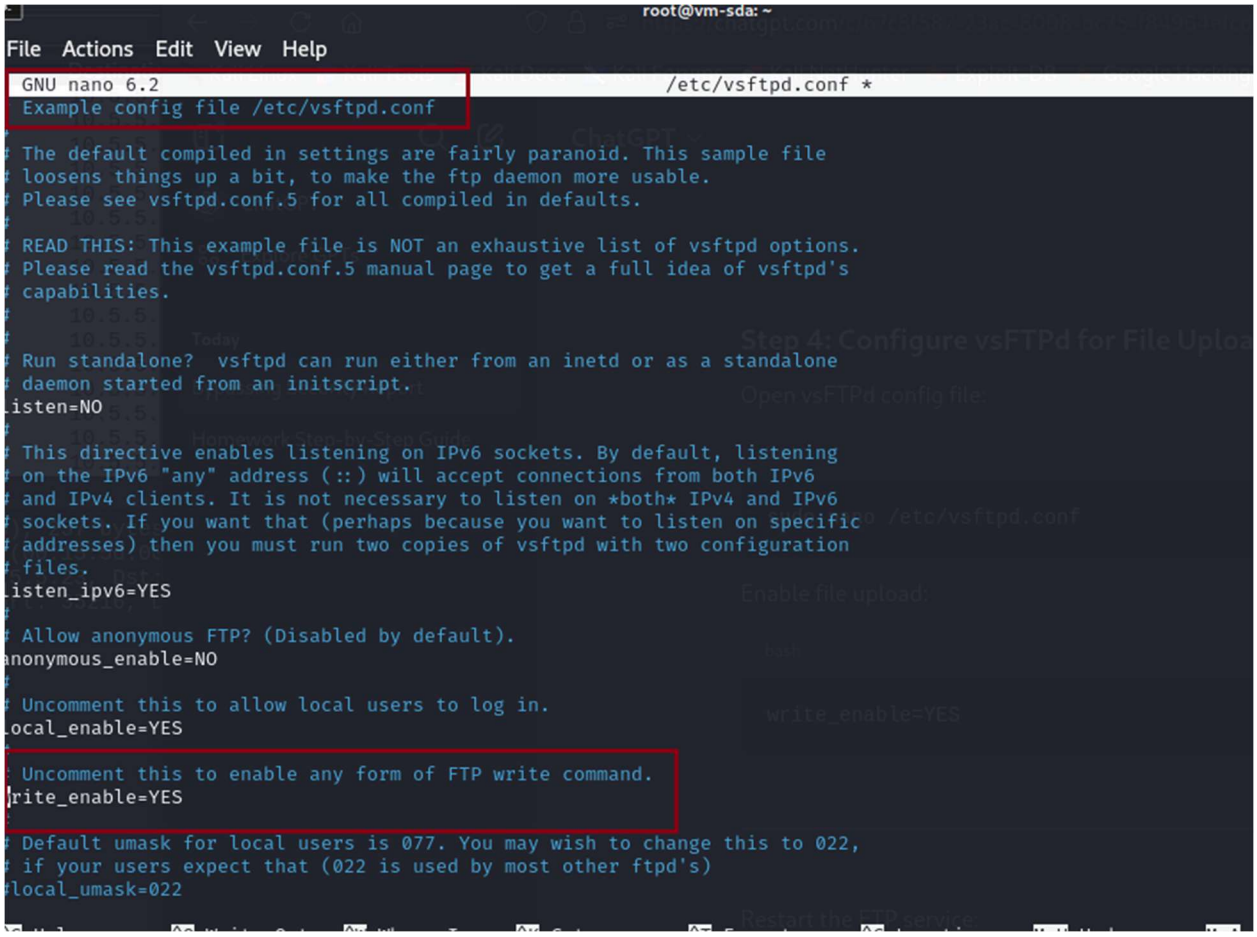
5. End the SSH connection.

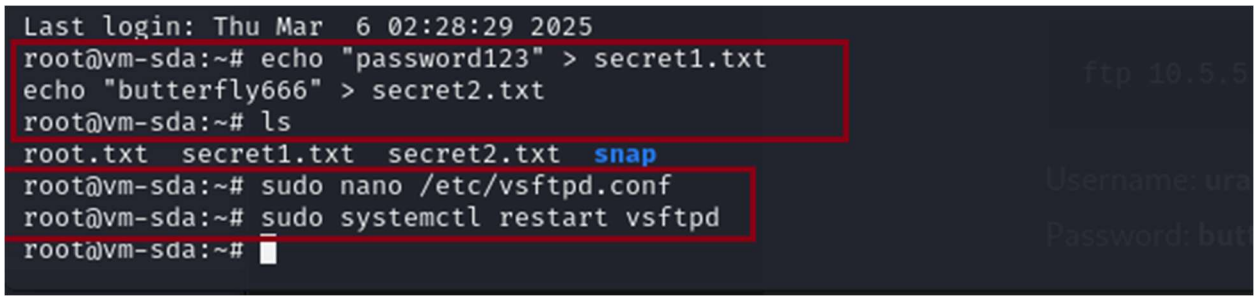
6. Try to look for the contents of the secret1.txt and secret2.txt files in the saved network Traffic

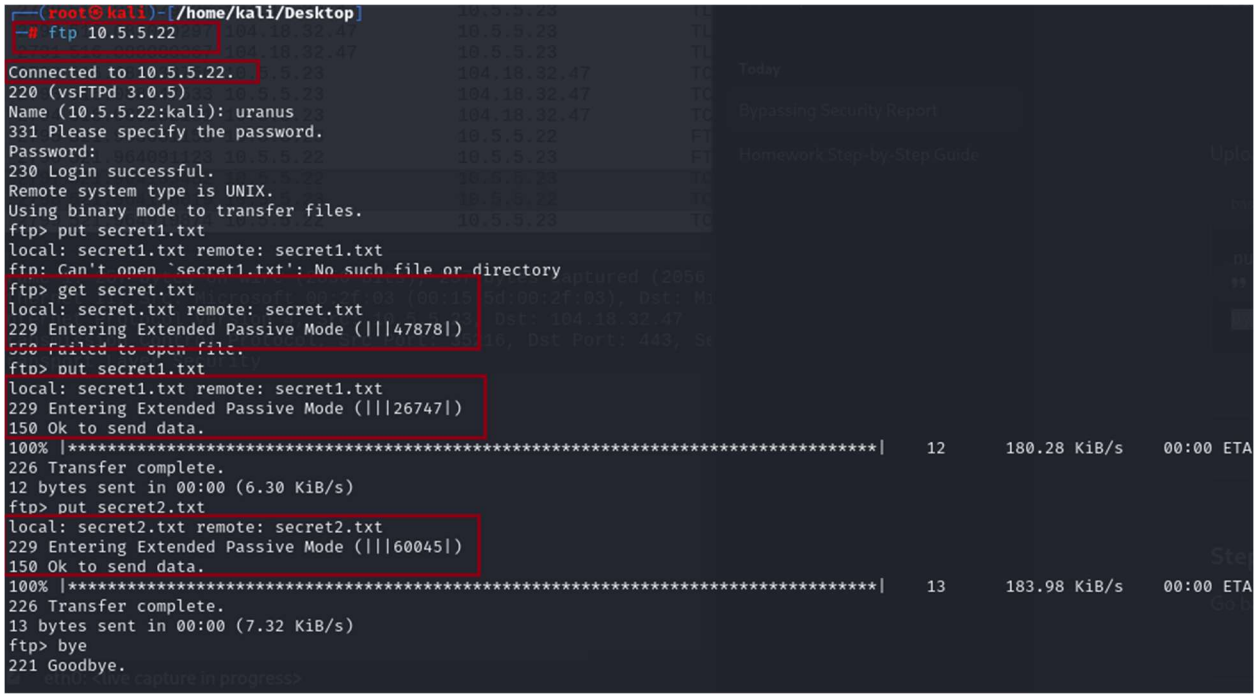


```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# ssh root@10.5.5.22

root@10.5.5.22's password:
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-27-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Mar  6 02:32:22 AM UTC 2025

  System load:  0.23388671875     Processes:               120
  Usage of /:   29.2% of 9.75GB   Users logged in:         1
  Memory usage: 6%                IPv4 address for enp0s3: 10.5.5.22
  Swap usage:   0%


12 updates can be applied immediately.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Thu Mar  6 02:28:29 2025
```

```
Last login: Thu Mar  6 02:28:29 2025
root@vm-sda:~# echo "password123" > secret1.txt
echo "butterfly666" > secret2.txt
root@vm-sda:~# ls
root.txt  secret1.txt  secret2.txt  snap
root@vm-sda:~#
```


root@vm-sda: ~
File Actions Edit View Help
GNU nano 6.2 /etc/vsftpd.conf *
Example config file /etc/vsftpd.conf
# The default compiled in settings are fairly paranoid. This sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.
# READ THIS: This example file is NOT an exhaustive list of vsftpd options.
# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
# capabilities.
# Run standalone? vsftpd can run either from an inetd or as a standalone
# daemon started from an initscript.
listen=NO
# This directive enables listening on IPv6 sockets. By default, listening
# on the IPv6 "any" address (::) will accept connections from both IPv6
# and IPv4 clients. It is not necessary to listen on *both* IPv4 and IPv6
# sockets. If you want that (perhaps because you want to listen on specific
# addresses) then you must run two copies of vsftpd with two configuration
# files.
listen_ipv6=YES
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=NO
# Uncomment this to allow local users to log in.
local_enable=YES
# Uncomment this to enable any form of FTP write command.
write_enable=YES
# Default umask for local users is 077. You may wish to change this to 022,
# if your users expect that (022 is used by most other ftpd's)
#local_umask=022


Last login: Thu Mar 6 02:28:29 2025
root@vm-sda:~# echo "password123" > secret1.txt
echo "butterfly666" > secret2.txt
root@vm-sda:~# ls
root.txt secret1.txt secret2.txt snap
root@vm-sda:~# sudo nano /etc/vsftpd.conf
root@vm-sda:~# sudo systemctl restart vsftpd
root@vm-sda:~#



Key Concepts

- **Encryption:** SSH uses strong encryption algorithms to protect data from unauthorized access.
- **Secure Shell (SSH):** SSH is a secure protocol used for remote login and other network services.
- **Data Confidentiality:** SSH ensures that data remains confidential and cannot be read by eavesdroppers.

Task 5 - FTP traffic analysis

Environment: Kali Linux, SDA machine from project 1

1. Start monitoring network traffic (Wireshark utility).

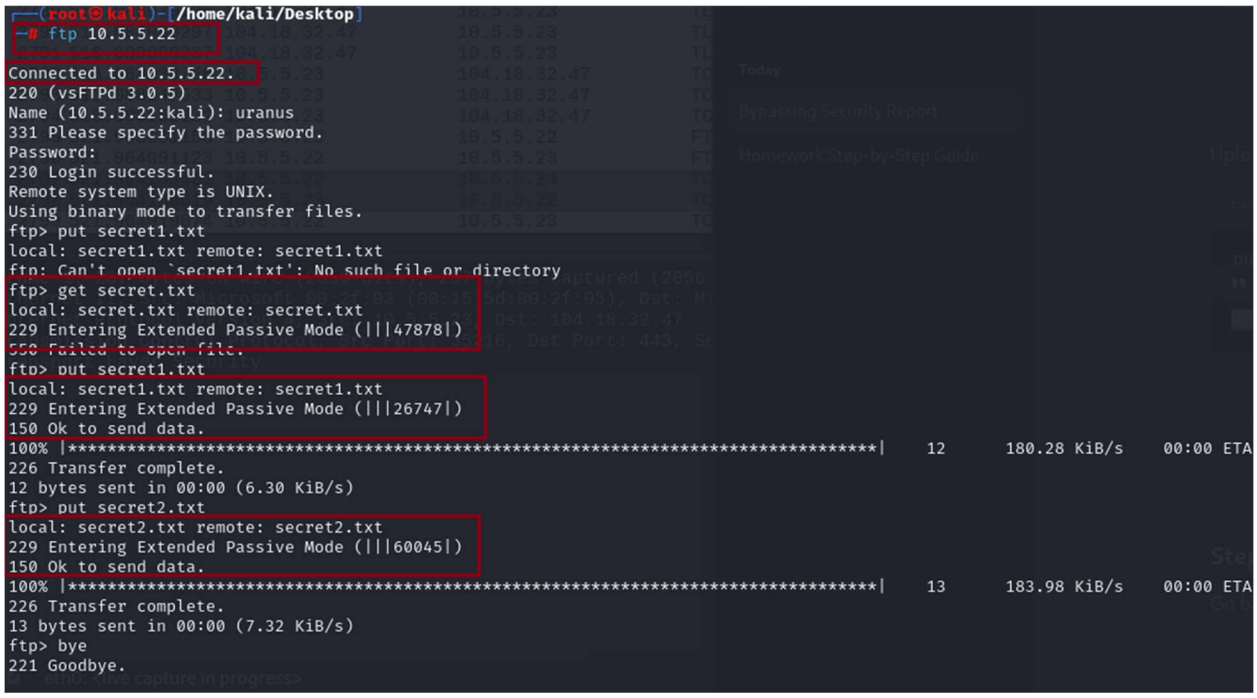
2. Establish a connection between Kalim and SDA via FTP.

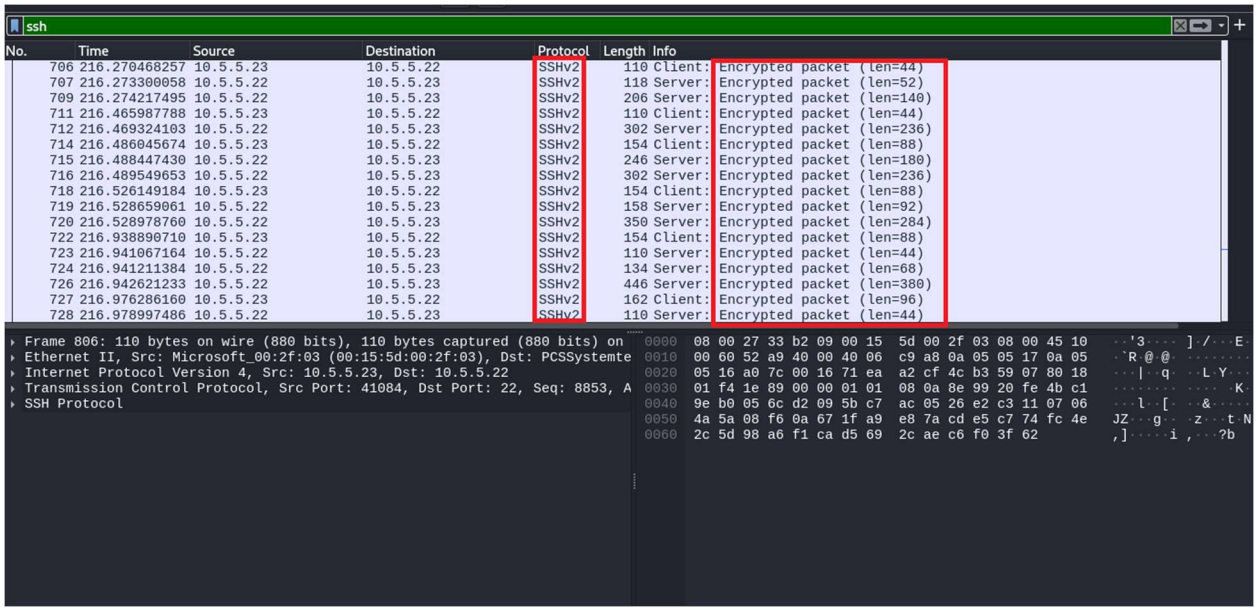
3. Upload a plain text file (with your own content) from Kali to the SDA.

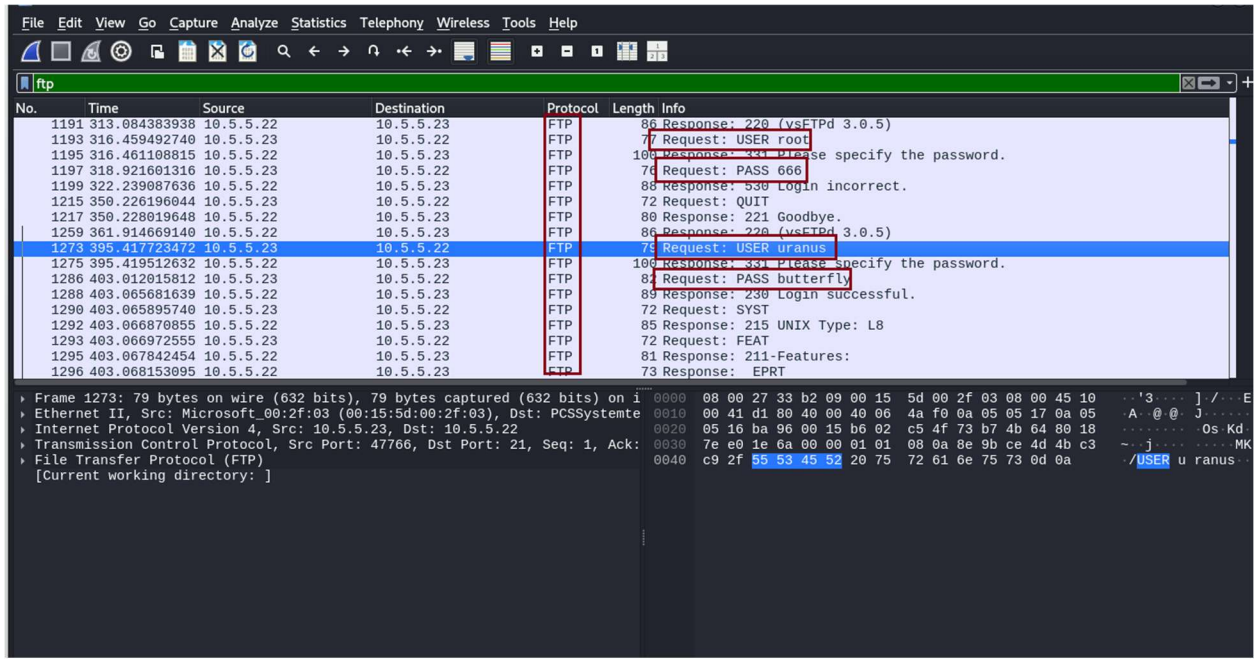
4. Download the files sekret1.txt and sekret2.txt from SDA to Kali

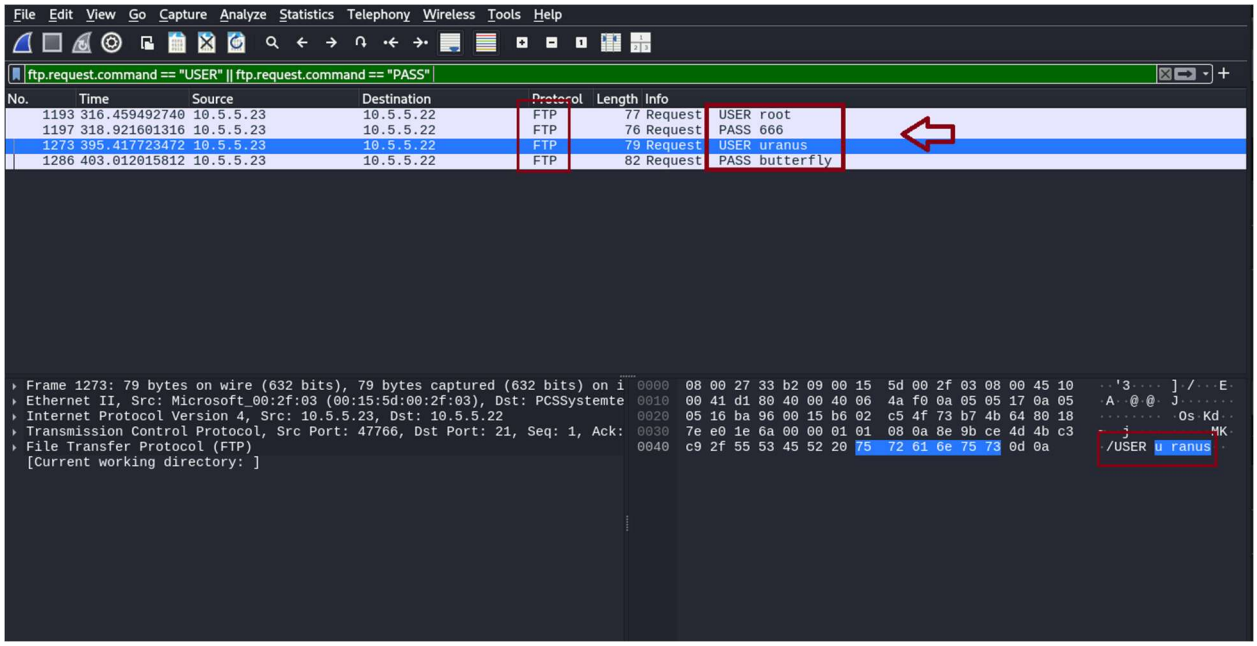
5. End the connection.

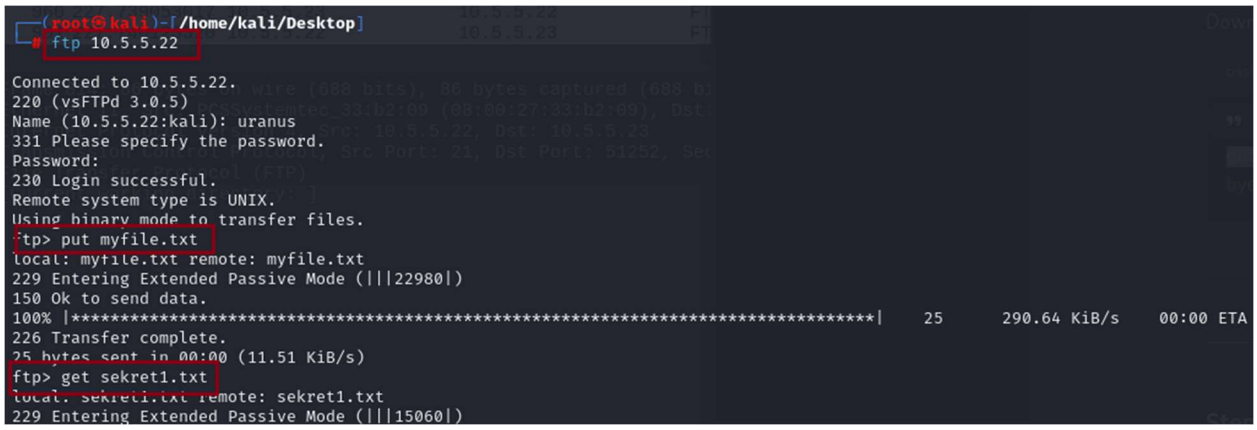
6. Find the content of uploaded and downloaded files in the saved network traffic.

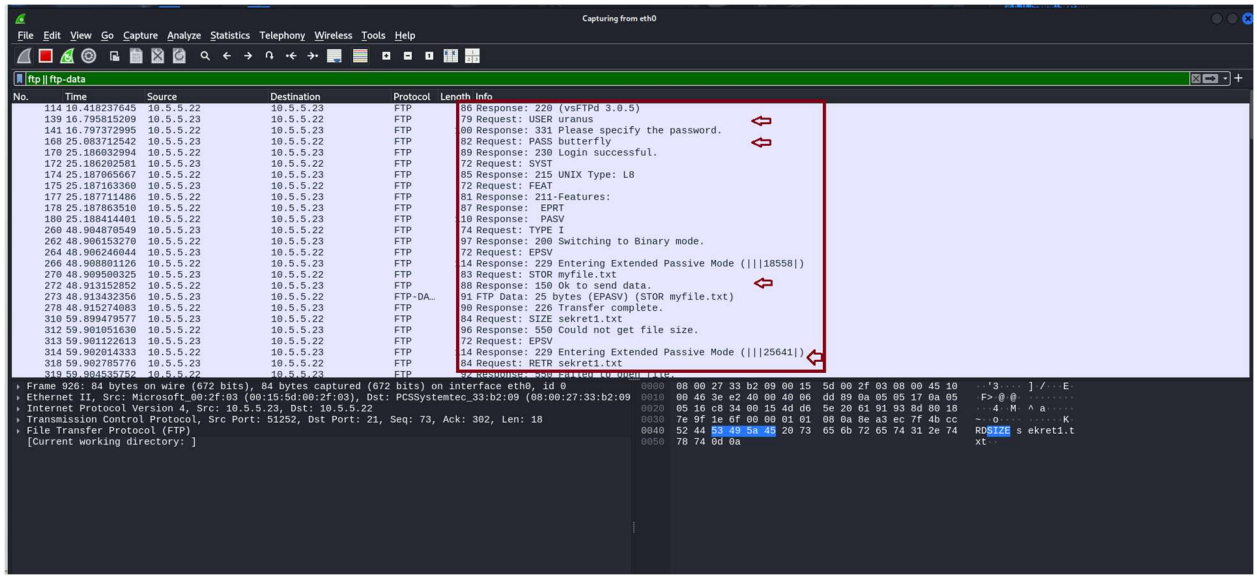



ssh
No. Time Source Destination Protocol Length Info
706 216.270468257 10.5.5.23 10.5.5.22 SSHv2 110 Client: Encrypted packet (len=44)
707 216.273300058 10.5.5.22 10.5.5.23 SSHv2 118 Server: Encrypted packet (len=52)
709 216.274217495 10.5.5.22 10.5.5.23 SSHv2 206 Server: Encrypted packet (len=140)
711 216.465987788 10.5.5.23 10.5.5.22 SSHv2 110 Client: Encrypted packet (len=44)
712 216.469324103 10.5.5.23 10.5.5.22 SSHv2 302 Server: Encrypted packet (len=236)
714 216.486045674 10.5.5.23 10.5.5.22 SSHv2 154 Client: Encrypted packet (len=88)
715 216.488447430 10.5.5.23 10.5.5.22 SSHv2 246 Server: Encrypted packet (len=180)
716 216.489549653 10.5.5.22 10.5.5.23 SSHv2 302 Server: Encrypted packet (len=236)
718 216.526149184 10.5.5.23 10.5.5.22 SSHv2 154 Client: Encrypted packet (len=88)
719 216.528659061 10.5.5.22 10.5.5.23 SSHv2 158 Server: Encrypted packet (len=92)
720 216.528978760 10.5.5.22 10.5.5.23 SSHv2 350 Server: Encrypted packet (len=284)
722 216.938890710 10.5.5.23 10.5.5.22 SSHv2 154 Client: Encrypted packet (len=88)
723 216.941067164 10.5.5.22 10.5.5.23 SSHv2 110 Server: Encrypted packet (len=44)
724 216.941211384 10.5.5.23 10.5.5.22 SSHv2 134 Server: Encrypted packet (len=68)
726 216.942621233 10.5.5.23 10.5.5.22 SSHv2 446 Server: Encrypted packet (len=380)
727 216.976286160 10.5.5.23 10.5.5.22 SSHv2 162 Client: Encrypted packet (len=96)
728 216.978997486 10.5.5.22 10.5.5.23 SSHv2 110 Server: Encrypted packet (len=44)
Frame 806: 110 bytes on wire (880 bits), 110 bytes captured (880 bits) on
Ethernet II, Src: Microsoft_00:2f:03 (00:15:5d:00:2f:03), Dst: PCSSystemte
Internet Protocol Version 4, Src: 10.5.5.23, Dst: 10.5.5.22
Transmission Control Protocol, Src Port: 41084, Dst Port: 22, Seq: 8853, A
SSH Protocol
0000 08 00 27 33 b2 09 00 15 5d 00 2f 03 08 00 45 10
0010 00 60 52 a9 40 00 40 06 c9 a8 0a 05 05 17 0a 05
0020 05 16 a0 7c 00 16 71 ea a2 cf 4c b3 59 07 80 18
0030 01 f4 1e 89 00 00 01 01 08 0a 8e 99 20 fe 4b c1
0040 9e b0 05 6c d2 09 5b c7 ac 05 26 e2 c3 11 07 06
0050 4a 5a 08 f6 0a 67 1f a9 e8 7a cd e5 c7 74 fc 4e
0060 2c 5d 98 a6 f1 ca d5 69 2c ae c6 f0 3f 62


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ftp
No. Time Source Destination Protocol Length Info
1191 313.084383938 10.5.5.22 10.5.5.23 FTP 86 Response: 220 (vsFTPd 3.0.5)
1193 316.459492740 10.5.5.23 10.5.5.22 FTP 77 Request: USER root
1195 316.461108815 10.5.5.22 10.5.5.23 FTP 100 Response: 331 Please specify the password.
1197 318.921601316 10.5.5.23 10.5.5.22 FTP 76 Request: PASS 666
1199 322.239087636 10.5.5.22 10.5.5.23 FTP 88 Response: 530 Login incorrect.
1215 350.226196044 10.5.5.23 10.5.5.22 FTP 72 Request: QUIT
1217 350.228019648 10.5.5.22 10.5.5.23 FTP 80 Response: 221 Goodbye.
1259 361.914669140 10.5.5.22 10.5.5.23 FTP 86 Response: 220 (vsFTPd 3.0.5)
1273 395.417723472 10.5.5.23 10.5.5.22 FTP 79 Request: USER uranus
1275 395.419512632 10.5.5.22 10.5.5.23 FTP 100 Response: 331 Please specify the password.
1286 403.012015812 10.5.5.23 10.5.5.22 FTP 81 Request: PASS butterfly
1288 403.065681639 10.5.5.22 10.5.5.23 FTP 89 Response: 230 Login successful.
1290 403.065895740 10.5.5.23 10.5.5.22 FTP 72 Request: SYST
1292 403.066870855 10.5.5.22 10.5.5.23 FTP 85 Response: 215 UNIX Type: L8
1293 403.066972555 10.5.5.23 10.5.5.22 FTP 72 Request: FEAT
1295 403.067842454 10.5.5.22 10.5.5.23 FTP 81 Response: 211-Features:
1296 403.068153095 10.5.5.22 10.5.5.23 FTP 73 Response: EPRT
Frame 1273: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on i
Ethernet II, Src: Microsoft_00:2f:03 (00:15:5d:00:2f:03), Dst: PCSSystemte
Internet Protocol Version 4, Src: 10.5.5.23, Dst: 10.5.5.22
Transmission Control Protocol, Src Port: 47766, Dst Port: 21, Seq: 1, Ack:
File Transfer Protocol (FTP)
[Current working directory:]
0000 08 00 27 33 b2 09 00 15 5d 00 2f 03 08 00 45 10
0010 00 41 d1 80 40 00 40 06 4a f0 0a 05 05 17 0a 05
0020 05 16 ba 96 00 15 b6 02 c5 4f 73 b7 4b 64 80 18
0030 7e e0 1e 6a 00 00 01 01 08 0a 8e 9b ce 4d 4b c3
0040 c9 2f 55 53 45 52 20 75 72 61 6e 75 73 0d 0a
/USER u ranus


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ftp.request.command == "USER" || ftp.request.command == "PASS"
No. Time Source Destination Protocol Length Info
1193 316.459492740 10.5.5.23 10.5.5.22 FTP 77 Request USER root
1197 318.921601316 10.5.5.23 10.5.5.22 FTP 76 Request PASS 666
1273 395.417723472 10.5.5.23 10.5.5.22 FTP 79 Request USER uranus
1286 403.012015812 10.5.5.23 10.5.5.22 FTP 82 Request PASS butterfly
Frame 1273: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on i
Ethernet II, Src: Microsoft_00:2f:03 (00:15:5d:00:2f:03), Dst: PCSSysteme
Internet Protocol Version 4, Src: 10.5.5.23, Dst: 10.5.5.22
Transmission Control Protocol, Src Port: 47766, Dst Port: 21, Seq: 1, Ack:
File Transfer Protocol (FTP)
[Current working directory:]
0000 08 00 27 33 b2 09 00 15 5d 00 2f 03 08 00 45 10
0010 00 41 d1 80 40 00 40 06 4a f0 0a 05 05 17 0a 05
0020 05 16 ba 96 00 15 b6 02 c5 4f 73 b7 4b 64 80 18
0030 7e e0 1e 6a 00 00 01 01 08 0a 8e 9b ce 4d 4b c3
0040 c9 2f 55 53 45 52 20 75 72 61 6e 75 73 0d 0a
/USER uranus


(root㉿kali)-[/home/kali/Desktop]
ftp 10.5.5.22
Connected to 10.5.5.22.
220 (vsFTPd 3.0.5)
Name (10.5.5.22:kali): uranus
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> put myfile.txt
local: myfile.txt remote: myfile.txt
229 Entering Extended Passive Mode (|||22980|)
150 Ok to send data.
100% |**********| 25 290.64 KiB/s 00:00 ETA
226 Transfer complete.
25 bytes sent in 00:00 (11.51 KiB/s)
ftp> get sekret1.txt
local: sekret1.txt remote: sekret1.txt
229 Entering Extended Passive Mode (|||15060|)


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ftp || ftp-data
No. Time Source Destination Protocol Length Info
114 10.418237645 10.5.5.22 10.5.5.23 FTP 86 Response: 220 (vsFTPd 3.0.5)
139 16.795815209 10.5.5.23 10.5.5.22 FTP 79 Request: USER uranus
141 16.797372995 10.5.5.22 10.5.5.23 FTP 00 Response: 331 Please specify the password.
168 25.083712542 10.5.5.23 10.5.5.22 FTP 82 Request: PASS butterfly
170 25.186032994 10.5.5.22 10.5.5.23 FTP 89 Response: 230 Login successful.
172 25.186202581 10.5.5.23 10.5.5.22 FTP 72 Request: SYST
174 25.187065667 10.5.5.22 10.5.5.23 FTP 85 Response: 215 UNIX Type: L8
175 25.187163360 10.5.5.23 10.5.5.22 FTP 72 Request: FEAT
177 25.187711486 10.5.5.22 10.5.5.23 FTP 81 Response: 211-Features:
178 25.187863510 10.5.5.22 10.5.5.23 FTP 87 Response: EPRT
180 25.188414401 10.5.5.22 10.5.5.23 FTP 10 Response: PASV
260 48.904870549 10.5.5.23 10.5.5.22 FTP 74 Request: TYPE I
262 48.906153270 10.5.5.22 10.5.5.23 FTP 97 Response: 200 Switching to Binary mode.
264 48.906246044 10.5.5.23 10.5.5.22 FTP 72 Request: EPSV
266 48.908801126 10.5.5.22 10.5.5.23 FTP 14 Response: 229 Entering Extended Passive Mode (|||18558|)
270 48.909500325 10.5.5.23 10.5.5.22 FTP 83 Request: STOR myfile.txt
272 48.913152852 10.5.5.22 10.5.5.23 FTP 88 Response: 150 Ok to send data.
273 48.913432356 10.5.5.23 10.5.5.22 FTP-DA… 91 FTP Data: 25 bytes (EPASV) (STOR myfile.txt)
278 48.915274083 10.5.5.22 10.5.5.23 FTP 90 Response: 226 Transfer complete.
310 59.899479577 10.5.5.23 10.5.5.22 FTP 84 Request: SIZE sekret1.txt
312 59.901051630 10.5.5.22 10.5.5.23 FTP 96 Response: 550 Could not get file size.
313 59.901122613 10.5.5.23 10.5.5.22 FTP 72 Request: EPSV
314 59.902014333 10.5.5.22 10.5.5.23 FTP 14 Response: 229 Entering Extended Passive Mode (|||25641|)
318 59.902785776 10.5.5.23 10.5.5.22 FTP 84 Request: RETR sekret1.txt
319 59.904535752 10.5.5.22 10.5.5.23 FTP 97 Response: 550 Failed to open file.
Frame 926: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface eth0, id 0
Ethernet II, Src: Microsoft_00:2f:03 (00:15:5d:00:2f:03), Dst: PCSSystemtec_33:b2:09 (08:00:27:33:b2:09
Internet Protocol Version 4, Src: 10.5.5.23, Dst: 10.5.5.22
Transmission Control Protocol, Src Port: 51252, Dst Port: 21, Seq: 73, Ack: 302, Len: 18
File Transfer Protocol (FTP)
[Current working directory:]
0000 08 00 27 33 b2 09 00 15 5d 00 2f 03 08 00 45 10
0010 00 46 3e e2 40 00 40 06 dd 89 0a 05 05 17 0a 05
0020 05 16 c8 34 00 15 4d d6 5e 20 61 91 93 8d 80 18
0030 7e 9f 1e 6f 00 00 01 01 08 0a a3 ec 7f 4b cc
0040 52 44 53 49 5a 45 20 73 65 6b 72 65 74 31 2e 74
0050 78 74 0d 0a
RDSIZE s ekret1.t
xt

Key Concepts

- **FTP (File Transfer Protocol):** A standard network protocol used to transfer files between a client and a server.
- **Plain Text:** Data sent without encryption, readable by anyone who intercepts it.
- **Security Risks:** FTP's lack of encryption makes it vulnerable to eavesdropping and data theft.

7. Conclusion

This project illustrates several critical aspects of network and password security:

- **Password Cracking:**
    - **Brute-force attacks** on SHA-256 and SHA-512 hashes demonstrate that if the password space is limited (e.g., maximum 5 characters or exactly 6 alphanumeric characters), even modern hash algorithms can be compromised using GPU-accelerated tools like Hashcat.
    - **Dictionary attacks** against MD5 and SHA-512 hashes (using the RockYou-50 dictionary) further show how common passwords are easily cracked if they are part of known wordlists.
- **Traffic Analysis:**
    - **HTTP traffic** analysis reveals that credentials sent over unencrypted HTTP connections are vulnerable to interception, emphasizing the importance of using HTTPS.
    - **SSH traffic** remains secure due to encryption, although connection details can be observed.
    - **FTP traffic** is particularly vulnerable because it transmits both control commands and data in clear text, allowing attackers to view sensitive file contents and login credentials.

Overall, the project reinforces the need for:

- Using strong, unpredictable passwords.
- Employing encrypted communication channels (e.g., HTTPS, SSH, SFTP) to protect data in transit.
- Regularly updating and securing network configurations to defend against eavesdropping and brute-force attacks.