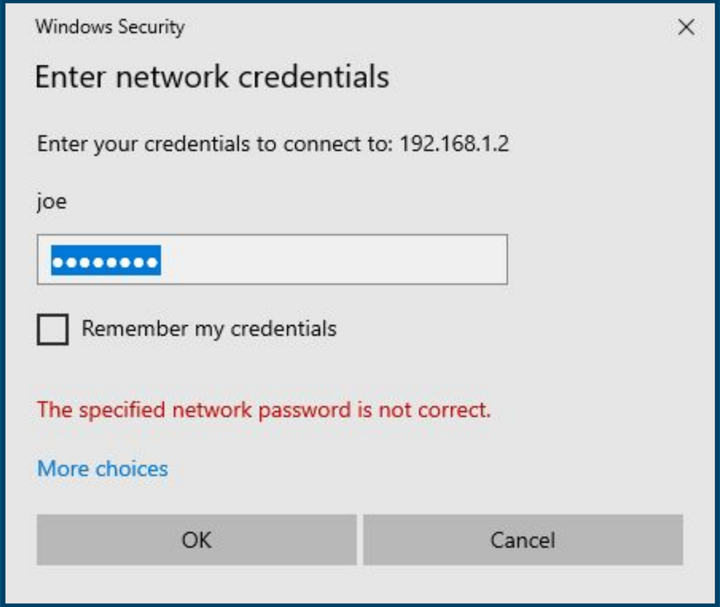
Understanding Password Security: How Hackers Crack Passwords

A Guide for Non-Technical Users

Ethical Hacking & Countermeasures | Brian F. Smith

Overview of Password Security

- Passwords are essential for protecting our online accounts
- Strong passwords prevent unauthorized access
- This presentation will explain how passwords work and how hackers can crack them



Windows Security

Enter network credentials

Enter your credentials to connect to: 192.168.1.2

joe

••••••••

☐ Remember my credentials

The specified network password is not correct.

Windows dialog box for credential log-in

How Windows Password Authentication Works

- Passwords are converted into unique digital fingerprints known as "hashes"
- Windows uses a system called NTLM to verify these hashes
  - "NTLM" stands for New Technology LAN Manager, which are Microsoft security protocols that protect your authentication processes
- The system checks if the password entered matches the stored hash

Fun fact!

The term "hash" originates from the idea of "mixing" or "chopping up" input data, much like how you would chop ingredients in cooking to create a new dish. The output, the hash, is unique to the input, and even a small change in the input data should result in a completely different hash value.

The Password Hashing Process

How does hashing *work*?

- Hashing turns a password into a fixed-length **string of characters**
- NTLMv2 **improves** security by adding **extra layers** to the process
- Hashes are **hard to reverse**, but they can still be cracked with the **right tools**

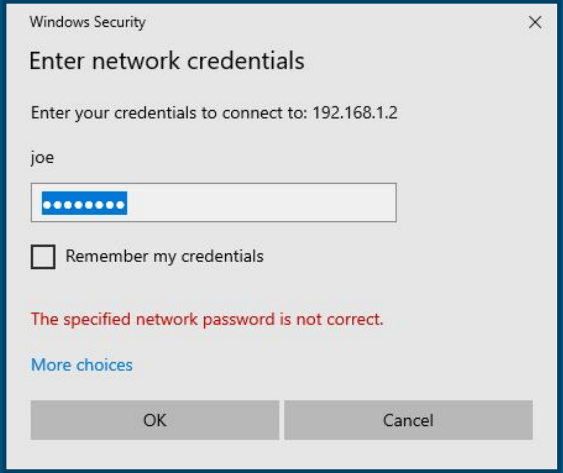
What are the "right tools" ?

- **Dictionary Attacks** use lists of common passwords to guess the right one
- **Brute Force Attacks** try every possible combination of characters
- **Hybrid Attacks** combine dictionary words with additional characters like numbers or symbols
- **Rainbow Tables** store precomputed hash values to speed up the cracking process

How do we actually use this tools though?

Capturing Passwords During Authentication

Remember that login screen from the first slide?



Windows Security

Enter network credentials

Enter your credentials to connect to: 192.168.1.2

joe

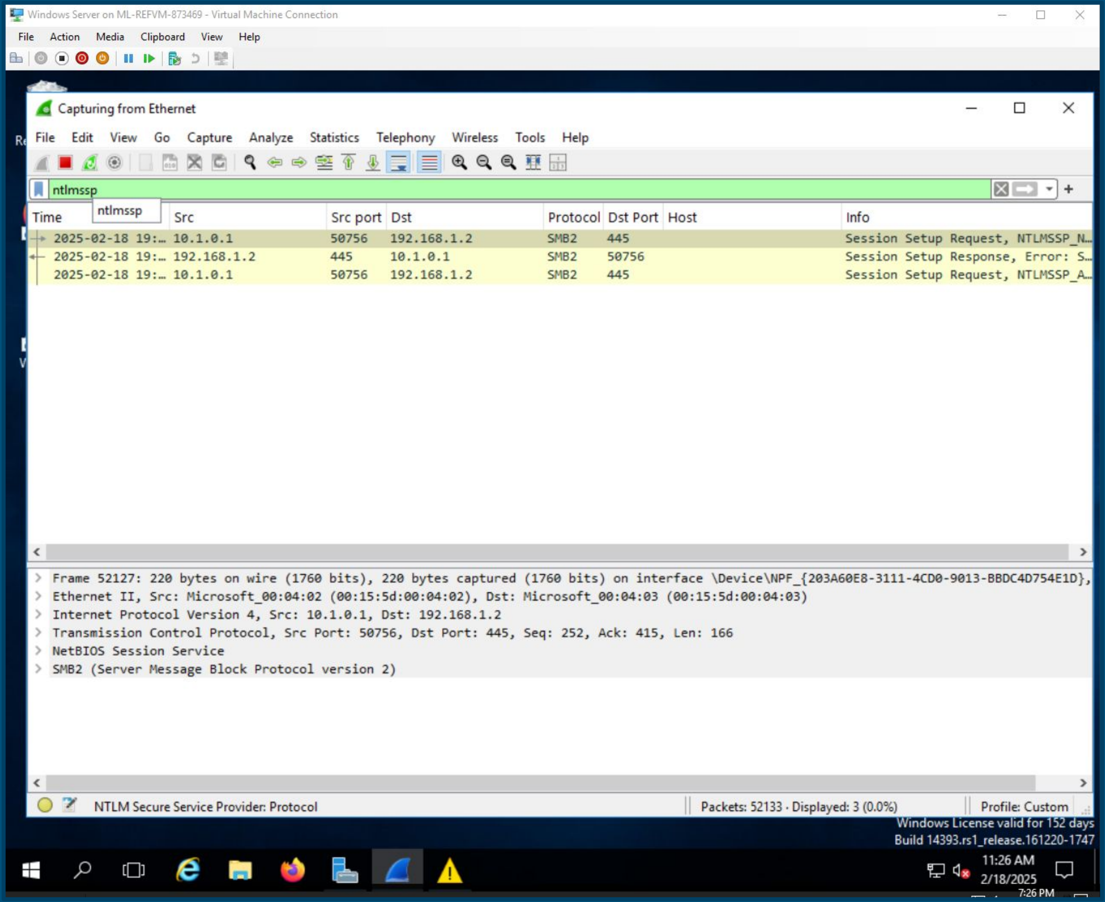
●●●●●●●●

□ Remember my credentials

The specified network password is not correct.

Example of Eavesdropping Using Wireshark

- Hackers can capture passwords using packet sniffers like **Wireshark**
  - Wireshark **monitors network traffic** for authentication attempts
  - It can capture the NTLMv2 **challenge and response messages**
- Password **hashes are transmitted** during the login process
- Once captured, hackers can attempt to **crack the hashes offline**
- Hackers use this data to crack passwords without **needing to know them directly**, or the users themselves
  - User error is very common however, and **social engineering attacks** can be used to **assist** these types of password cracking attacks


Windows Server on ML-REFVM-873469 - Virtual Machine Connection
File Action Media Clipboard View Help
Capturing from Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ntlmssp

Time	Src	Src port	Dst	Protocol	Dst Port	Host	Info
2025-02-18 19:…	10.1.0.1	50756	192.168.1.2	SMB2	445		Session Setup Request, NTLMSSP_N…
2025-02-18 19:…	192.168.1.2	445	10.1.0.1	SMB2	50756		Session Setup Response, Error: S…
2025-02-18 19:…	10.1.0.1	50756	192.168.1.2	SMB2	445		Session Setup Request, NTLMSSP_A…

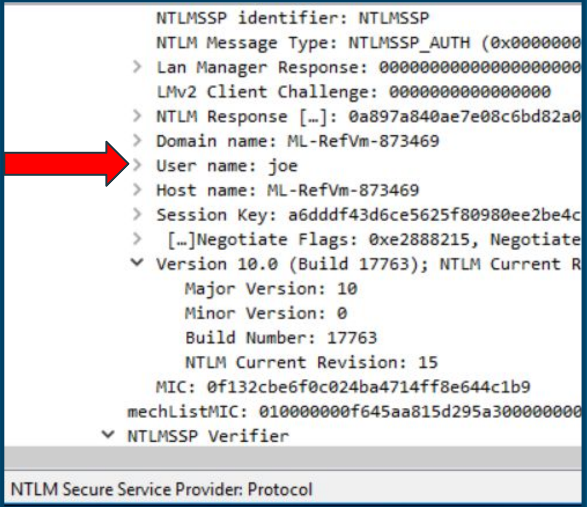
```
> Frame 52127: 220 bytes on wire (1760 bits), 220 bytes captured (1760 bits) on interface \Device\NPF_{203A60E8-3111-4CD0-9013-BBDC4D754E1D},
> Ethernet II, Src: Microsoft_00:04:02 (00:15:5d:00:04:02), Dst: Microsoft_00:04:03 (00:15:5d:00:04:03)
> Internet Protocol Version 4, Src: 10.1.0.1, Dst: 192.168.1.2
> Transmission Control Protocol, Src Port: 50756, Dst Port: 445, Seq: 252, Ack: 415, Len: 166
> NetBIOS Session Service
> SMB2 (Server Message Block Protocol version 2)
```

NTLM Secure Service Provider: Protocol | Packets: 52133 · Displayed: 3 (0.0%) | Profile: Custom

Windows License valid for 152 days
Build 14393.rs1_release.161220-1747
11:26 AM
2/18/2025
7:26 PM

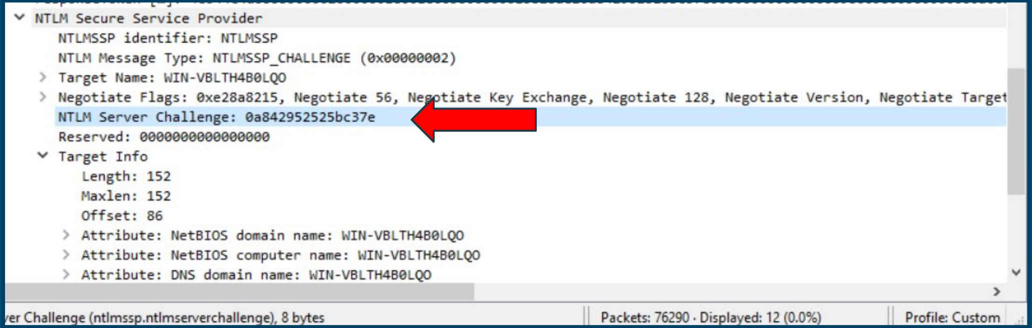
*Visualized Wireshark **eavesdropping** on a user's login attempt, using an NTLM filter*

Sifting through NTLM Protocols in Wireshark



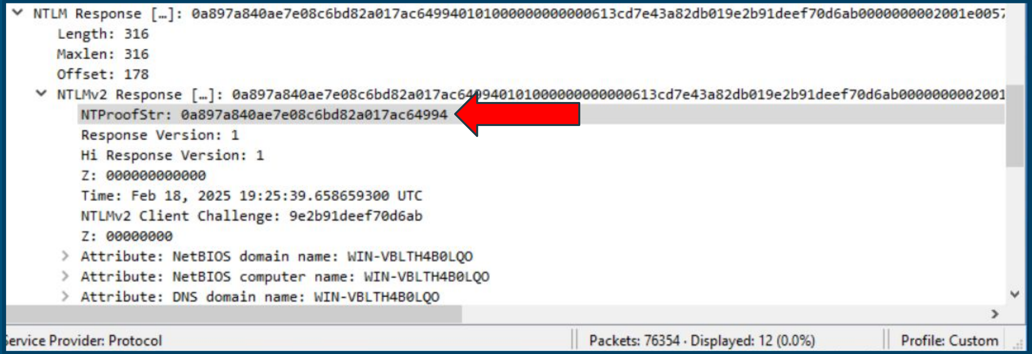
```
    NTLMSSP identifier: NTLMSSP
    NTLM Message Type: NTLMSSP_AUTH (0x0000000
  > Lan Manager Response: 000000000000000000000
    LMv2 Client Challenge: 0000000000000000
  > NTLM Response […]: 0a897a840ae7e08c6bd82a0
  > Domain name: ML-RefVm-873469
  > User name: joe
  > Host name: ML-RefVm-873469
  > Session Key: a6dddf43d6ce5625f80980ee2be4c
  >  […]Negotiate Flags: 0xe2888215, Negotiate
  ⌄ Version 10.0 (Build 17763); NTLM Current R
       Major Version: 10
       Minor Version: 0
       Build Number: 17763
       NTLM Current Revision: 15
    MIC: 0f132cbe6f0c024ba4714ff8e644c1b9
mechListMIC: 010000000f645aa815d295a300000000
⌄ NTLMSSP Verifier
```

NTLM Secure Service Provider: Protocol



```
⌄ NTLM Secure Service Provider
    NTLMSSP identifier: NTLMSSP
    NTLM Message Type: NTLMSSP_CHALLENGE (0x00000002)
  > Target Name: WIN-VBLTH4B0LQO
  > Negotiate Flags: 0xe28a8215, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version, Negotiate Target
    NTLM Server Challenge: 0a842952525bc37e
    Reserved: 0000000000000000
  ⌄ Target Info
       Length: 152
       Maxlen: 152
       Offset: 86
     > Attribute: NetBIOS domain name: WIN-VBLTH4B0LQO
     > Attribute: NetBIOS computer name: WIN-VBLTH4B0LQO
     > Attribute: DNS domain name: WIN-VBLTH4B0LQO
```

ver Challenge (ntlmssp.ntlmserverchallenge), 8 bytes | Packets: 76290 · Displayed: 12 (0.0%) | Profile: Custom



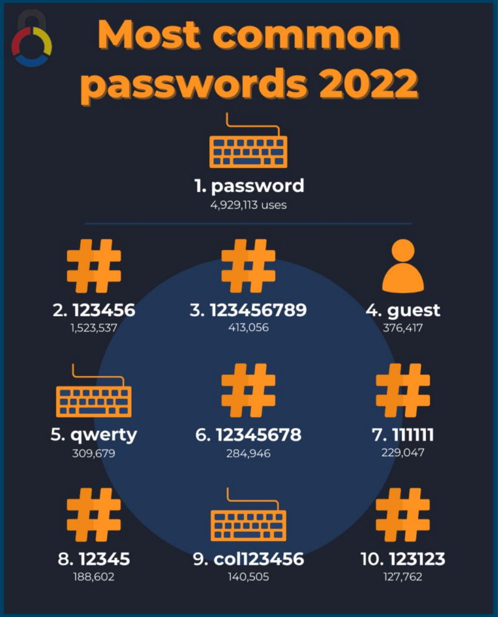
```
⌄ NTLM Response […]: 0a897a840ae7e08c6bd82a017ac6499401010000000000000613cd7e43a82db019e2b91deef70d6ab000000002001e0057
    Length: 316
    Maxlen: 316
    Offset: 178
  ⌄ NTLMv2 Response […]: 0a897a840ae7e08c6bd82a017ac6499401010000000000000613cd7e43a82db019e2b91deef70d6ab000000002001
       NTProofStr: 0a897a840ae7e08c6bd82a017ac64994
       Response Version: 1
       Hi Response Version: 1
       Z: 000000000000
       Time: Feb 18, 2025 19:25:39.658659300 UTC
       NTLMv2 Client Challenge: 9e2b91deef70d6ab
       Z: 00000000
     > Attribute: NetBIOS domain name: WIN-VBLTH4B0LQO
     > Attribute: NetBIOS computer name: WIN-VBLTH4B0LQO
     > Attribute: DNS domain name: WIN-VBLTH4B0LQO
```

ervice Provider: Protocol | Packets: 76354 · Displayed: 12 (0.0%) | Profile: Custom

Just by capturing network traffic in Wireshark, I was able to obtain the User name, his host name, and NTLM protocol data!

Cracking Passwords

- To test the **strength** of different passwords, I attempted to crack them using **Hashcat**
- Hashcat is a **powerful tool** for cracking password hashes
- It uses your computer's **GPU** (Graphical Processing Unit) to try **millions** of password combinations **quickly**
- Hashcat can use a **wordlist**, such as RockYou, to try **common** passwords
- It tests passwords **one by one** from the list until it finds a **match**
- This method is **fast**, but it only works on **weak** or common passwords



Wordlist Attacks using Hashcat

Let’s test four fake user’s passwords:

Examples Passwords and Why They Are Secure or Not

Steve: sunshine

- **Doesn't meet complexity rules**
- Too simple: no numbers, no uppercase, no special characters
- Insecure: Found on password lists like RockYou (easy to crack)

Johnny: Johnny2311*

- **Meets complexity rules**
- Contains uppercase, lowercase, number, and special character
- Insecure: Found on password lists (easy to guess)

Mark: bGlRoctCTht

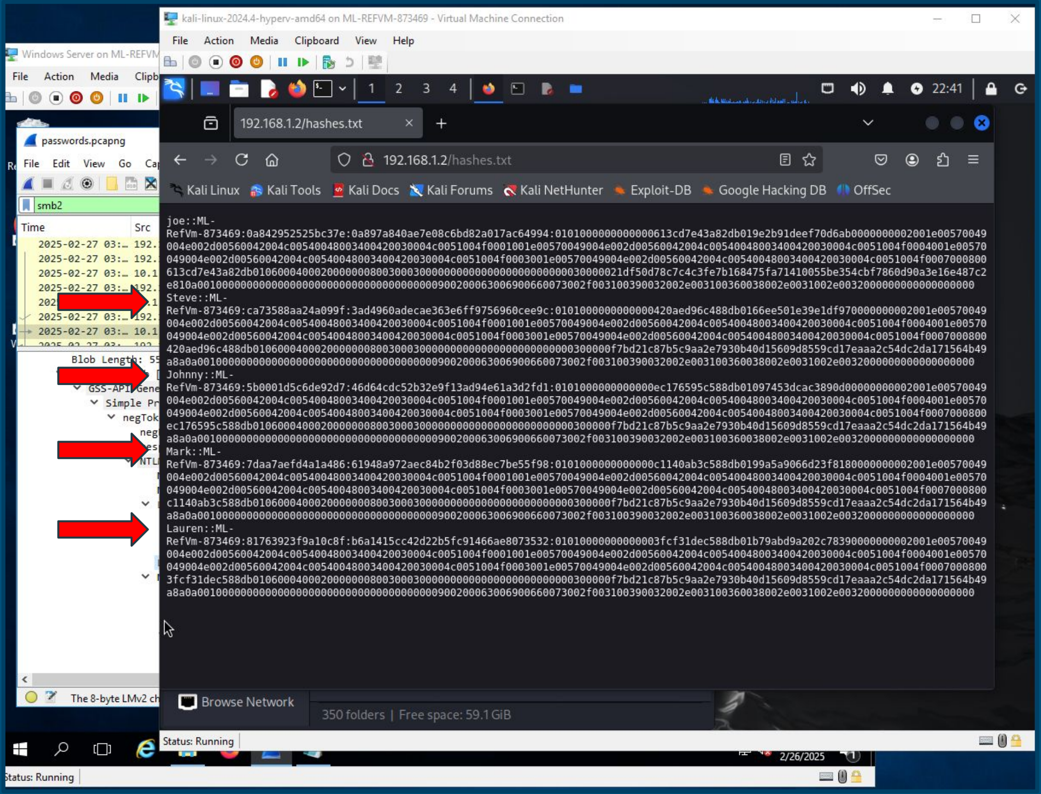
- **Doesn't meet complexity rules**
- Too complex for most password checkers
- Secure: Difficult to guess via brute force, but not allowed by Windows due to missing special characters

Lauren: O;dC1>}db9I83

- **Meets complexity rules**
- Contains uppercase, lowercase, numbers, and special characters
- Secure: Not easily cracked, not found in common password lists

Attack Preparation

- Using the NTLM Protocol data we **eavesdropped** in on via Wireshark, I compiled a **list** of hashes
- These hashes contain **user names,** host names, server challenges, and the **hash itself**
- I can then take this data and **run it against my dictionary of common passwords**
- The most common is the **RockYou list,** which contains 14,000,000+ passwords



```
2025-02-27 03:… 192.
2025-02-27 03:… 192.
2025-02-27 03:… 10.1
2025-02-27 03:… 192.
202
2025-02-27 03:… 192.
2025-02-27 03:… 10.1
```

```
joe::ML-
RefVm-873469:0a842952525bc37e:0a897a840ae7e08c6bd82a017ac64994:0101000000000000613cd7e43a82db019e2b91deef70d6ab0000000002001e00570049
004e002d00560042004c00540048003400420030004c0051004f0001001e00570049004e002d00560042004c00540048003400420030004c0051004f0004001e00570
049004e002d00560042004c00540048003400420030004c0051004f0003001e00570049004e002d00560042004c00540048003400420030004c0051004f0007000800
613cd7e43a82db010600040002000000080030003000000000000000000000000030000021df50d78c7c4c3fe7b168475fa71410055be354cbf7860d90a3e16e487c2
e810a001000000000000000000000000000000000009002000630069006600730002f003100390032002e003100360038002e0031002e0032000000000000000000
Steve::ML-
RefVm-873469:ca73588aa24a099f:3ad4960adecae363e6ff9756960cee9c:0101000000000000420aed96c488db0166ee501e39e1df970000000002001e00570049
004e002d00560042004c00540048003400420030004c0051004f0001001e00570049004e002d00560042004c00540048003400420030004c0051004f0004001e00570
049004e002d00560042004c00540048003400420030004c0051004f0003001e00570049004e002d00560042004c00540048003400420030004c0051004f0007000800
420aed96c488db0106000400020000000800300030000000000000000000000000300000f7bd21c87b5c9aa2e7930b40d15609d8559cd17eaaa2c54dc2da171564b49
a8a0a001000000000000000000000000000000000009002000630069006600730002f003100390032002e003100360038002e0031002e0032000000000000000000
Johnny::ML-
RefVm-873469:5b0001d5c6de92d7:46d64cdc52b32e9f13ad94e61a3d2fd1:0101000000000000ec176595c588db01097453dcac3890d00000000002001e00570049
004e002d00560042004c00540048003400420030004c0051004f0001001e00570049004e002d00560042004c00540048003400420030004c0051004f0004001e00570
049004e002d00560042004c00540048003400420030004c0051004f0003001e00570049004e002d00560042004c00540048003400420030004c0051004f0007000800
ec176595c588db0106000400020000000800300030000000000000000000000000300000f7bd21c87b5c9aa2e7930b40d15609d8559cd17eaaa2c54dc2da171564b49
a8a0a001000000000000000000000000000000000009002000630069006600730002f003100390032002e003100360038002e0031002e0032000000000000000000
Mark::ML-
RefVm-873469:7daa7aefd4a1a486:61948a972aec84b2f03d88ec7be55f98:0101000000000000c1140ab3c588db0199a5a9066d23f8180000000002001e00570049
004e002d00560042004c00540048003400420030004c0051004f0001001e00570049004e002d00560042004c00540048003400420030004c0051004f0004001e00570
049004e002d00560042004c00540048003400420030004c0051004f0003001e00570049004e002d00560042004c00540048003400420030004c0051004f0007000800
c1140ab3c588db0106000400020000000800300030000000000000000000000000300000f7bd21c87b5c9aa2e7930b40d15609d8559cd17eaaa2c54dc2da171564b49
a8a0a001000000000000000000000000000000000009002000630069006600730002f003100390032002e003100360038002e0031002e0032000000000000000000
Lauren::ML-
RefVm-873469:81763923f9a10c8f:b6a1415cc42d22b5fc91466ae8073532:01010000000000003fcf31dec588db01b79abd9a202c78390000000002001e00570049
004e002d00560042004c00540048003400420030004c0051004f0001001e00570049004e002d00560042004c00540048003400420030004c0051004f0004001e00570
049004e002d00560042004c00540048003400420030004c0051004f0003001e00570049004e002d00560042004c00540048003400420030004c0051004f0007000800
3fcf31dec588db0106000400020000000800300030000000000000000000000000300000f7bd21c87b5c9aa2e7930b40d15609d8559cd17eaaa2c54dc2da171564b49
a8a0a001000000000000000000000000000000000009002000630069006600730002f003100390032002e003100360038002e0031002e0032000000000000000000
```

The red arrows indicate the four users that I set up with passwords of varying strengths

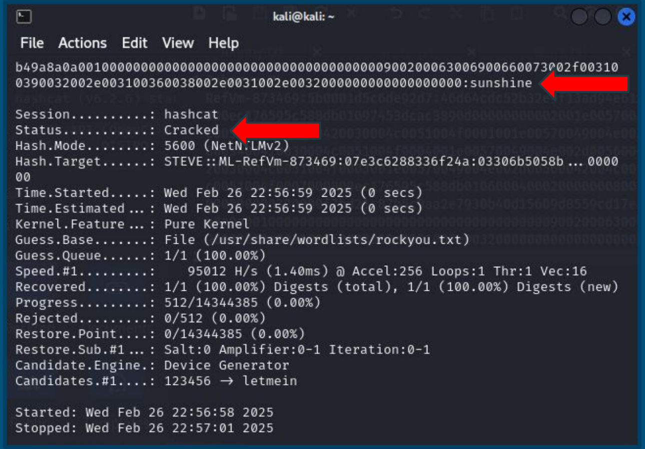
Successful Dictionary Attacks

- In a Kali workstation, I used this command to **successfully** crack the two **insecure** passwords:

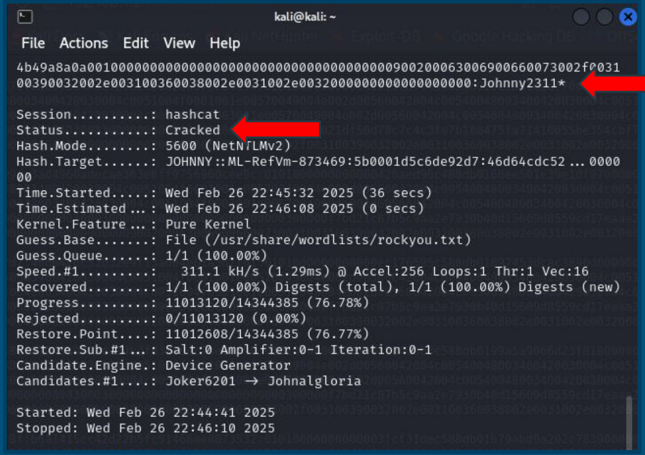
```
hashcat -a 0 -m 5600 hashes.txt /usr/share/wordlists/rockyou.txt
```

- -a 0: Specifies a **dictionary** attack
- -m 5600: Specifies the hash **type** (NTLMv2 for Windows passwords)
- hashes.txt: The **file** containing the NTLMv2 password hashes
- /usr/share/wordlists/rockyou.txt: The **wordlist used** to attempt cracking the passwords

Hashcat cracked **both** Steve's and Johnny's passwords because they were **found** in the RockYou wordlist, highlighting the **importance** of using **unique** and **complex** passwords **beyond common lists.**



```
b49a8a0a00100000000000000000000000000000000000090020006300690066007302f00310
0390032002e003100360038002e0031002e00320000000000000000000:sunshine

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: STEVE::ML-RefVm-873469:07e3c6288336f24a:03306b5058b ... 0000
00
Time.Started.....: Wed Feb 26 22:56:59 2025 (0 secs)
Time.Estimated ...: Wed Feb 26 22:56:59 2025 (0 secs)
Kernel.Feature ...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    95012 H/s (1.40ms) @ Accel:256 Loops:1 Thr:1 Vec:16
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 512/14344385 (0.00%)
Rejected.........: 0/512 (0.00%)
Restore.Point....: 0/14344385 (0.00%)
Restore.Sub.#1 ...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 123456 → letmein

Started: Wed Feb 26 22:56:58 2025
Stopped: Wed Feb 26 22:57:01 2025
```



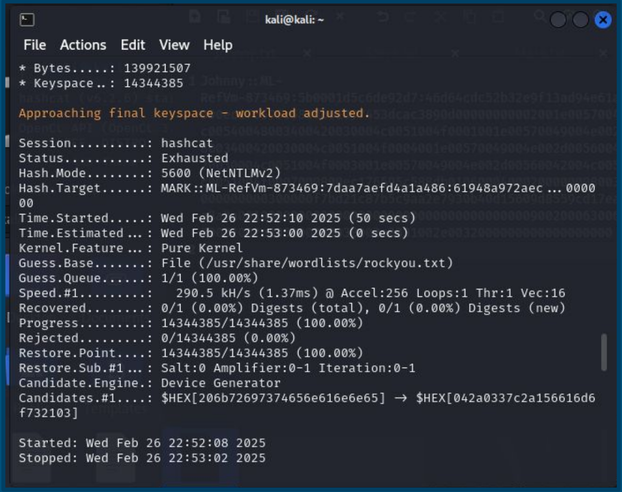
```
4b49a8a0a00100000000000000000000000000000000000090020006300690066007302f0031
00390032002e003100360038002e0031002e00320000000000000000000:Johnny2311*

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: JOHNNY::ML-RefVm-873469:5b0001d5c6de92d7:46d64cdc52 ... 0000
00
Time.Started.....: Wed Feb 26 22:45:32 2025 (36 secs)
Time.Estimated ...: Wed Feb 26 22:46:08 2025 (0 secs)
Kernel.Feature ...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   311.1 kH/s (1.29ms) @ Accel:256 Loops:1 Thr:1 Vec:16
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 11013120/14344385 (76.78%)
Rejected.........: 0/11013120 (0.00%)
Restore.Point....: 11012608/14344385 (76.77%)
Restore.Sub.#1 ...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: Joker6201 → Johnalgloria

Started: Wed Feb 26 22:44:41 2025
Stopped: Wed Feb 26 22:46:10 2025
```

Unsuccessful Dictionary Attacks

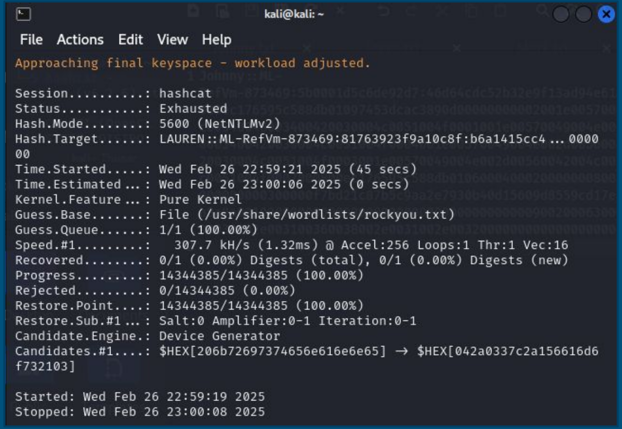
I attempted to crack Mark and Lauren's **secure** passwords using the **same method** with Hashcat and the RockYou wordlist, but was **unsuccessful**.

- **Mark's password ("bGlRoctCTht"):**
  - **Complex** and long, **not** found in the RockYou wordlist
  - Due to its **randomness** and lack of common patterns, it **couldn't be cracked** using a dictionary attack
- **Lauren's password ("O;dC1>}db9I83"):**
  - **Met all complexity rules** and contained a **mix of characters**
  - The password's unique combination made it **resistant to cracking through the wordlist**

These attempts demonstrate that **passwords following best practices** (length, randomness, and complexity) are **much harder to crack** using basic dictionary attacks.

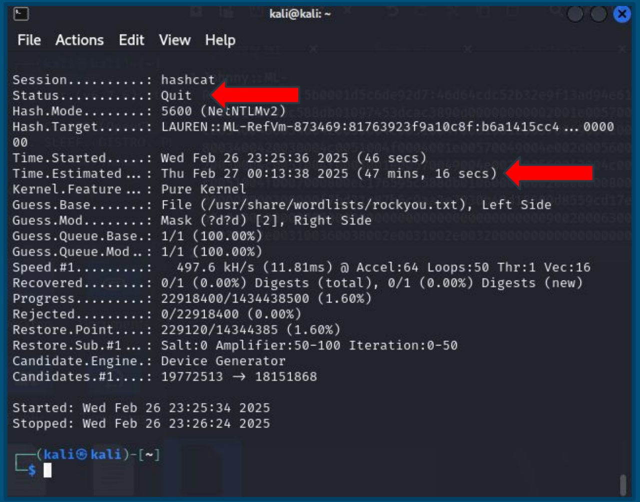


```
* Bytes.....: 139921507
* Keyspace..: 14344385

Approaching final keyspace - workload adjusted.

Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: MARK::ML-RefVm-873469:7daa7aefd4a1a486:61948a972aec...0000
00
Time.Started.....: Wed Feb 26 22:52:10 2025 (50 secs)
Time.Estimated...: Wed Feb 26 22:53:00 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   290.5 kH/s (1.37ms) @ Accel:256 Loops:1 Thr:1 Vec:16
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 14344385/14344385 (100.00%)
Rejected.........: 0/14344385 (0.00%)
Restore.Point....: 14344385/14344385 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: $HEX[206b72697374656e616e6e65] → $HEX[042a0337c2a156616d6
f732103]

Started: Wed Feb 26 22:52:08 2025
Stopped: Wed Feb 26 22:53:02 2025
```



```
Approaching final keyspace - workload adjusted.

Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: LAUREN::ML-RefVm-873469:81763923f9a10c8f:b6a1415cc4...0000
00
Time.Started.....: Wed Feb 26 22:59:21 2025 (45 secs)
Time.Estimated...: Wed Feb 26 23:00:06 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   307.7 kH/s (1.32ms) @ Accel:256 Loops:1 Thr:1 Vec:16
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 14344385/14344385 (100.00%)
Rejected.........: 0/14344385 (0.00%)
Restore.Point....: 14344385/14344385 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: $HEX[206b72697374656e616e6e65] → $HEX[042a0337c2a156616d6
f732103]

Started: Wed Feb 26 22:59:19 2025
Stopped: Wed Feb 26 23:00:08 2025
```

Hybrid Cracking Attack



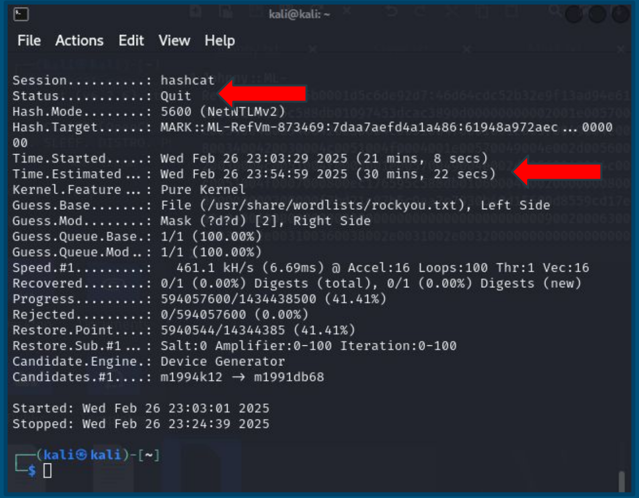
```
Session..........: hashcat
Status...........: Quit
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: LAUREN::ML-RefVm-873469:81763923f9a10c8f:b6a1415cc4...0000
00
Time.Started.....: Wed Feb 26 23:25:36 2025 (46 secs)
Time.Estimated...: Thu Feb 27 00:13:38 2025 (47 mins, 16 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt), Left Side
Guess.Mod........: Mask (?d?d) [2], Right Side
Guess.Queue.Base.: 1/1 (100.00%)
Guess.Queue.Mod..: 1/1 (100.00%)
Speed.#1.........:   497.6 kH/s (11.81ms) @ Accel:64 Loops:50 Thr:1 Vec:16
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 22918400/1434438500 (1.60%)
Rejected.........: 0/22918400 (0.00%)
Restore.Point....: 229120/14344385 (1.60%)
Restore.Sub.#1...: Salt:0 Amplifier:50-100 Iteration:0-50
Candidate.Engine.: Device Generator
Candidates.#1....: 19772513 → 18151868

Started: Wed Feb 26 23:25:34 2025
Stopped: Wed Feb 26 23:26:24 2025

┌──(kali㉿kali)-[~]
└─$ 
```

The **estimated time** for completion was high, indicating the **complexity and strength of their passwords**

I attempted to crack Mark and Lauren's passwords using a **hybrid attack**, which is done by appending **two random digits** to the RockYou wordlist with this command:

hashcat -a 6 -m 5600 hashes.txt rockyou.txt '?d?d'

It adds a random pattern (e.g., two digits: ?d?d) to each word from the list, trying combinations like password12 or sunshine99



```
Session..........: hashcat
Status...........: Quit
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: MARK::ML-RefVm-873469:7daa7aefd4a1a486:61948a972aec...0000
00
Time.Started.....: Wed Feb 26 23:03:29 2025 (21 mins, 8 secs)
Time.Estimated...: Wed Feb 26 23:54:59 2025 (30 mins, 22 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt), Left Side
Guess.Mod........: Mask (?d?d) [2], Right Side
Guess.Queue.Base.: 1/1 (100.00%)
Guess.Queue.Mod..: 1/1 (100.00%)
Speed.#1.........:   461.1 kH/s (6.69ms) @ Accel:16 Loops:100 Thr:1 Vec:16
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 594057600/1434438500 (41.41%)
Rejected.........: 0/594057600 (0.00%)
Restore.Point....: 5940544/14344385 (41.41%)
Restore.Sub.#1...: Salt:0 Amplifier:0-100 Iteration:0-100
Candidate.Engine.: Device Generator
Candidates.#1....: m1994k12 → m1991db68

Started: Wed Feb 26 23:03:01 2025
Stopped: Wed Feb 26 23:24:39 2025

┌──(kali㉿kali)-[~]
└─$ 
```

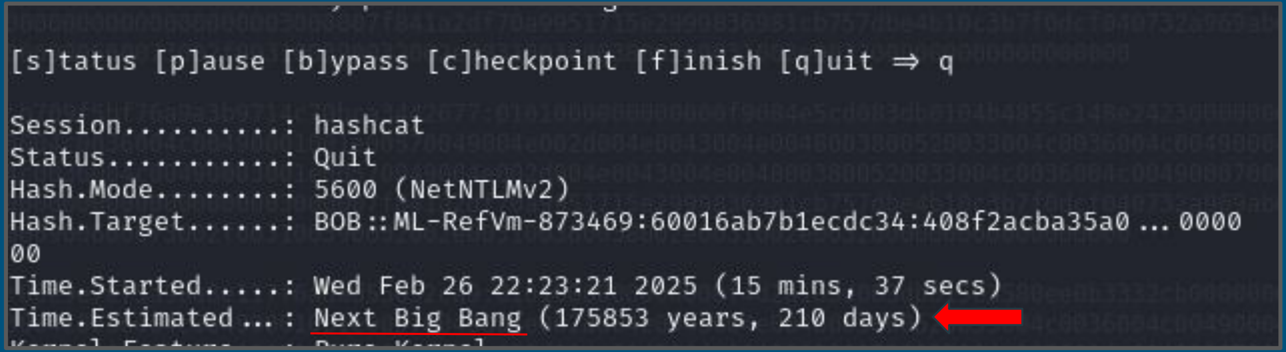
This method highlights that **even with additional random digits**, complex and unique passwords **remain resistant** to cracking without **significant computational power and time**

Brute Force Attack

- Brute force attack tries **all possible character combinations**
- Used Hashcat with this command for a 7-character lowercase password:

  hashcat -a 3 -m 5600 hashes.txt ?l?l?l?l?l?l?l -i

- Also attempted variations of the brute force, using different characters



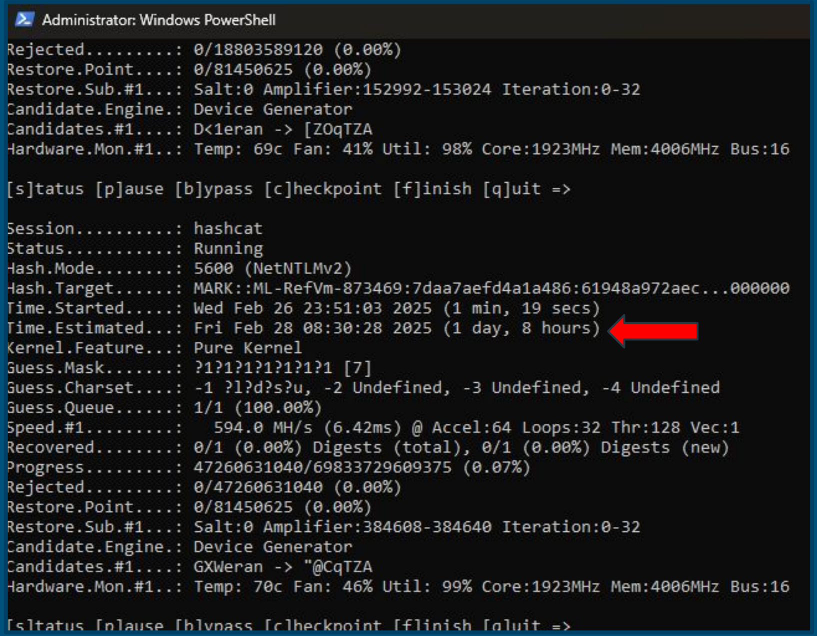
```
[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit => q

Session..........: hashcat
Status...........: Quit
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: BOB::ML-RefVm-873469:60016ab7b1ecdc34:408f2acba35a0...0000
00
Time.Started.....: Wed Feb 26 22:23:21 2025 (15 mins, 37 secs)
Time.Estimated...: Next Big Bang (175853 years, 210 days)
```

Attempting this on a Kali Virtual Machine, with no GPU (processing power) resulted in an estimated of 175,853 years!

The next Big Bang would occur before it was able to crack their strong passwords!

Brute Force Attack with a Strong GPU

- Out of curiosity, I tried it on my gaming computer with a 1070 Ti GPU
  - My NVIDIA GTX 1070 Ti is a **high-performance** graphics card, **speeding up password cracking** by using **parallel processing** but still requiring significant time for complex passwords
- Kali Linux VM: Estimated time to crack passwords was 175,000 years
- Gaming Computer (1070 Ti): Estimated time dropped to 1.5 days
- Both results show how **time-consuming** brute force attacks can be on **strong passwords**



```
Rejected.........: 0/18803589120 (0.00%)
Restore.Point....: 0/81450625 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:152992-153024 Iteration:0-32
Candidate.Engine.: Device Generator
Candidates.#1....: D<1eran -> [ZOqTZA
Hardware.Mon.#1..: Temp: 69c Fan: 41% Util: 98% Core:1923MHz Mem:4006MHz Bus:16

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit =>

Session..........: hashcat
Status...........: Running
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: MARK::ML-RefVm-873469:7daa7aefd4a1a486:61948a972aec...000000
Time.Started.....: Wed Feb 26 23:51:03 2025 (1 min, 19 secs)
Time.Estimated...: Fri Feb 28 08:30:28 2025 (1 day, 8 hours)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?1?1?1?1?1?1 [7]
Guess.Charset....: -1 ?l?d?s?u, -2 Undefined, -3 Undefined, -4 Undefined
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   594.0 MH/s (6.42ms) @ Accel:64 Loops:32 Thr:128 Vec:1
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 47260631040/69833729609375 (0.07%)
Rejected.........: 0/47260631040 (0.00%)
Restore.Point....: 0/81450625 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:384608-384640 Iteration:0-32
Candidate.Engine.: Device Generator
Candidates.#1....: GXWeran -> "@CqTZA
Hardware.Mon.#1..: Temp: 70c Fan: 46% Util: 99% Core:1923MHz Mem:4006MHz Bus:16

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit =>
```

Recommendations for Secure, Usable Passwords

Recommendations:

1. Avoid common passwords
2. Use passphrases
3. Include multiple character types
4. Length *does* actually matter
5. Use unique passwords for each account
6. Consider a password manager

It’s not always your own data at risk. Threat actors can use your insecure passwords to gain access and move laterally in a system, gaining access to organizational data.

Why they are recommended:

1. Don't use easily guessable words like "password" or names
2. Combine random words or phrases, making them long and complex (e.g., “PurpleMountain!78Horse$”)
3. Mix uppercase, lowercase, numbers, and symbols to increase complexity
4. Aim for 12+ characters; the longer the password, the harder it is to crack
5. Reusing passwords increases risk if one is cracked
6. Helps you store and generate strong, unique passwords without the need to remember them all