Documentation by: Dominic McElroy

Windows Password Cracking - Assignment 8 – Dominic McElroy

Date: 4/26/2025

Topic: ***Windows Password Cracking***

Overview:

We cracked three Windows password hashes employing three different methods in this assignment:

- *Dictionary Attack*
- *Brute-Force Attack*
- *Cryptanalysis with Rainbow Tables*

The motive for this exercise was to experiment with these methods and be aware of how proficient they are against each other.

Objectives:

✓*Show awareness of Windows password hashing.*

✓*Conduct attacks on Windows password hashes.*

Assumptions:

✓*Awareness of Unix-like environments.*

✓*Elementary knowledge of Linux command-line operations.*

Expected Outcomes:

✓*Know the relevance of strong passwords.*

✓*Describe methods of attacking hashed passwords.*

Background Information:

Documentation by: Dominic McElroy

Cryptography hash functions are used to secure user passwords in current applications. Though the majority of algorithms offer great security, the strength selected by the user has a substantial influence on their effectiveness. Weaker or outdated algorithms can also render even strong passwords vulnerable.

Two major password hashing algorithms are utilized by Windows:

• LM (Lan Manager Hash): Originally used for legacy versions of Windows. Password length restrictions and character restrictions create LM hashes relatively weak.

• NTLM (NT Lan Manager Hash): Improved hashing that does not divide the password and is based on MD4 and HMAC-MD5 hashes. NTLM is more robust than LM but vulnerable to contemporary cracking attacks.

Exercise Steps:

1. Setup

• Lunched the virtual machine.

• Logged in

• Opened the terminal application.

2. Examination of Hashes

```
gedit ~/Desktop/hashes/hashes2016.txt
```

• Observed the hashes were kept in PWDump format.

3. Dictionary Attack against Administrator Account

• Ran into the hashes directory:

```
cd ~/Desktop/hashes/
```

```
ls
```

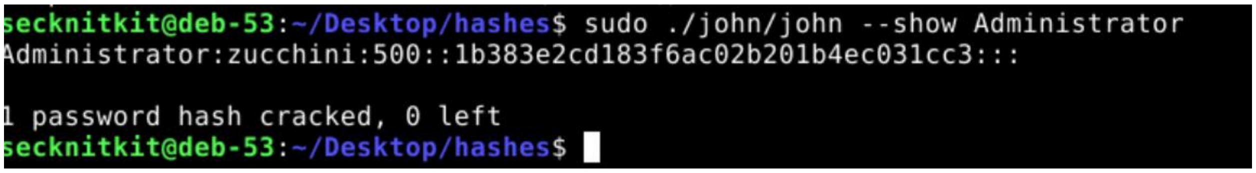
• Performed dictionary attack on Administrator hash:

```
sudo ./john/john --wordlist=Wordlist.txt --format=NT Administrator
```

• Showed cracked password:

```
sudo ./john/john --show Administrator
```

• Screenshot of Administrator's Cracked Password:



```
secknitkit@deb-53:~/Desktop/hashes$ sudo ./john/john --show Administrator
Administrator:zucchini:500::1b383e2cd183f6ac02b201b4ec031cc3:::

1 password hash cracked, 0 left
secknitkit@deb-53:~/Desktop/hashes$
```

---

4. Brute-Force Attack on Charles' Account

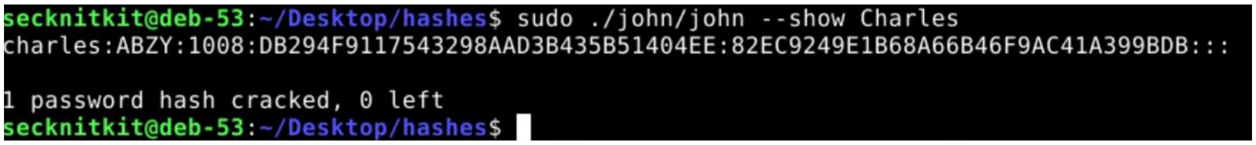
• Cracked Charles' LM hash by brute-force approach:

sudo ./john/john --format=LM Charles

• Showed cracked password:

sudo ./john/john --show Charles

• Screenshot of Charles' Cracked Password:



```
secknitkit@deb-53:~/Desktop/hashes$ sudo ./john/john --show Charles
charles:ABZY:1008:DB294F9117543298AAD3B435B51404EE:82EC9249E1B68A66B46F9AC41A399BDB:::

1 password hash cracked, 0 left
secknitkit@deb-53:~/Desktop/hashes$
```

---

5. Rainbow Table Attack on Alice's Account

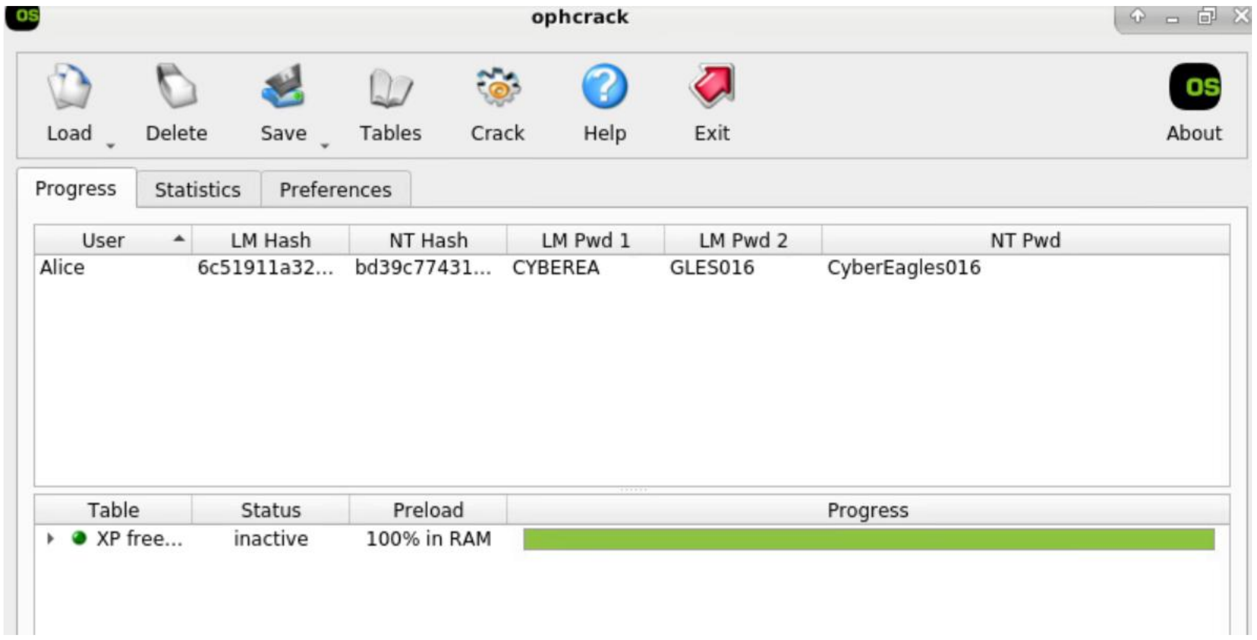
• Executed ophcrack:

sudo ophcrack

• Imported the PWDump file for Alice and cracked the password.

• Alice's Broken Password Screenshot:

Documentation by: Dominic McElroy



User	LM Hash	NT Hash	LM Pwd 1	LM Pwd 2	NT Pwd
Alice	6c51911a32...	bd39c77431...	CYBEREA	GLES016	CyberEagles016

Table	Status	Preload	Progress
XP free...	inactive	100% in RAM	

Post Assignment Questions:

1. What were the three passwords?

o Administrator: *zucchini*

o Charles: *ABZY*

o Alice: *CyberEagles016*

---

2. Give an example password where a dictionary attack would prove more efficient than a brute-force attack. Why?

Example: sunshine123

Reason:

A dictionary attack would be quicker because "sunshine" is a normal word that would most likely be included in a pre-existing wordlist. Dictionary attacks are intended to crack passwords made up of real words or regular patterns, so they are faster than attempting all combinations of characters like brute-force would.

---

3. Give an example password where brute force would succeed but a dictionary attack would fail. Why?

Example: x7!q9@w2#

Reason:

Brute-force would eventually win out because it systematically tries all possible combinations of characters. A dictionary attack would most likely fail, however, since x7!q9@w2# is a randomly chosen, non-dictionary-based password that would never be in a typical or even large wordlist.

---

4. How do you make your scheme technically more resistant so that traditional rainbow tables will not work? Why?

Answer:

With salting — selecting a random value and prepending that to the password before hashing — we can thwart traditionally rainbow tables. Salting ensures that two users who might use the same password will also have totally different hashes, so precomputed rainbow tables won't work because they'd have to be redone for every unique salt.

5. If you go to the "Charles" file, you will see both LM and NTLM hashes. From what you have learned so far, why did we attack the LM hash?

Answer:

We aimed at attacking the LM hash as it's considerably weaker than the NTLM hash. LM hashing is done by upper-casing all characters, truncating the password to two 7-character halves, and using the DES encryption and is highly vulnerable to even basic brute force or dictionary attack. Nevertheless, NTLM hashes use stronger hashing techniques along with support for the preservation of the password case and thus are less vulnerable.