



Hash cracking

Day 23: You wanna know what happens to your hashes?

29.12.2024

—

Nikhil Kumar

Detailed Documentation: Password Cracking Task

Scenario Overview

Glitch, an expert in active and passive reconnaissance, retrieved a discarded tablet from Mayor Malware's garbage. Among the files on the tablet, a password-protected PDF caught Glitch's attention. The task is to uncover the password and retrieve evidence from the document, demonstrating techniques for cracking password hashes and understanding password security mechanisms.

Learning Objectives

1. Understand hash functions and hash values.
 2. Learn how passwords are stored securely.
 3. Explore methods for cracking hashed passwords.
 4. Apply tools to retrieve passwords from password-protected documents.
-

Understanding Hashed Passwords

Old Practices of Password Storage

- Initially, passwords were stored in plaintext alongside usernames.
- A data breach exposed passwords, leading to credential reuse across multiple platforms.

Hash Functions

- Hash functions produce a fixed-size output (hash value) for an input of any size.
- Examples:
 - **SHA-256**: Creates a 256-bit hash value.
 - **MD5**: Considered insecure due to vulnerabilities.

Modern Password Storage

1. **Hashing Passwords**: Instead of saving plaintext passwords, the hash value is stored.

- Example: Password **ASDF1234** with MD5 results in **ce1bccda287f1d9e6d80dbd4cb6beb60**.

```
VM Terminal
user@machine:~/AOC2024/example_files$ ls -lh
total 2.3G
-rw-rw-r-- 1 user user 2.3G Oct 24 15:05 Fedora-Workstation-Live-x86_64-1.4.iso
-rw-rw-r-- 1 user user 13 Nov 14 14:49 hello.txt
user@machine:~/AOC2024/example_files$ sha256sum *
a2dd3caf3224b8f3a640d9e31b1016d2a4e98a6d7cb435a1e2030235976d6da2  Fedora-Workstation-Live-x86_64-1.4.iso
03ba204e50d126e4674c005e04d82e84c21366780af1f43bd54a37816b6ab340  hello.txt
```

2. Salting:

- A random string (salt) is added to the password before hashing.
- Stored as **hash(password + salt)** to make cracking more difficult.

3. Issues in Implementation:

- Despite guidelines, some organizations still store plaintext passwords.
- **Example: A platform leaked 600 million plaintext passwords over seven years.**

Cracking Password-Protected Files

Data at Rest

- Data stored on devices like flash drives, laptops, and external storage must be encrypted.
- Digital forensic investigators need tools to access encrypted files during criminal investigations.

Passwords in Practice

- Users often select weak, easily guessable passwords.
- Common password patterns:
 - Substituting characters (e.g., **f1uffyc4t** instead of **fluffycat**).
 - Adding years or dates (e.g., **fluffy2024**).



Step 1: Analyze the Hash

- ```
 VM Terminal
```
- ```
user@machine:~$ cd AOC2024/  
user@machine:~/AOC2024$ cat hash1.txt  
d956a72c83a895cb767bb5be8dba791395021dcece002b689cf3b5bf5aaa20ac  
user@machine:~/AOC2024$ python hash-id.py  
  
#####  
#####  
#                               #  
#   _      _       _____    #  
# / \  \ / \     / \         / \  #  
#\ / \ / \ \_/_\_/ \_\_/ \_\_/ \  #  
# |___|_|__|\_____|_|__|\_____  #  
# \|/\|/\|/\|/\|/\|/\|/\|/\|/\|  #  
#  \|/\|/\|/\|/\|/\|/\|/\|/\|/\|  #  
#   V/V/V/V/V/V/V/V/V/V/V/V/V/V  v1.2 #  
#                                   By Zion3R #  
#                                 www.Blackexploit.com #  
#                                Root@Blackexploit.com #  
#####  
-----  
HASH: d956a72c83a895cb767bb5be8dba791395021dcece002b689cf3b5bf5aaa20ac  
  
Possible Hashes:  
[+] SHA-256  
[+] Haval-256  
  
Least Possible Hashes:  
[+] GOST R 34.11-94  
[+] RIPEMD-256  
[+] SNEFRU-256  
[+] SHA-256(HMAC)  
[+] Haval-256(HMAC)  
[+] RIPEMD-256(HMAC)  
[+] SNEFRU-256(HMAC)  
[+] SHA-256(md5($pass))  
[+] SHA-256(sha1($pass))  
-----
```

- _____

Step 2: Use a Wordlist for Cracking

- Wordlist: **rockyou.txt** (a real-world dataset of breached passwords).
- Command to crack the hash:
`john --format=raw-sha256 --wordlist=/usr/share/wordlists/rockyou.txt /home/user/AOC2024/hash1.txt`
- Monitor the results to identify the matching password.

```
user@ip-10-10-133-97:~/AOC2024$ john --format=raw-sha256 --wordlist=/usr/share/wordlists/rockyou.txt --rules=wordlist hash1.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA256 [SHA256 256/256 AVX2 8x])
Warning: poor OpenMP scalability for this hash type, consider --fork=2
Will run 2 OpenMP threads
Note: Passwords longer than 18 [worst case UTF-8] to 55 [ASCII] rejected
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
Enabling duplicate candidate password suppressor
fluffycat12 (?)
ig 0-00-00-16 DONE (2024-12-08 19:41) 0.06020g/s 2335Kp/s 2335Kc/s 2335Kc/s markie182..cherrylee2
Use the "--show --format=Raw-SHA256" options to display all of the cracked passwords reliably
Session completed.
user@ip-10-10-133-97:~/AOC2024$
```

Crack the hash value stored in hash1.txt. What was the password?

Correct Answer- fluffycat12

Enhanced cracking method.

John the Ripper to crack a password hash more effectively by applying rules that modify words from a wordlist. Here's how it works:

Command Overview

```
john --format=raw-sha256 --rules=wordlist
--wordlist=/usr/share/wordlists/rockyou.txt hash1.txt
```

1. **--format=raw-sha256**: Specifies the type of hash you're cracking (SHA-256 in this case).
2. **--rules=wordlist**: Activates a set of rules that change the wordlist entries, like:
 - Adding numbers to the beginning or end of words.
 - Replacing letters with symbols (e.g., **a** → **@**, **i** → **!**, **s** → **\$**).

- ## Why Use Rules?

How It Works

- John picks words from the wordlist.
 - It applies the rules to create variations of each word (e.g., "password" becomes "password123", "P@ssword", etc.).
 - Then it compares these guesses with the hash in **hash1.txt**
- The first thing you need to do is to convert the password-protected file into a format that john can attack. Luckily, John the Ripper jumbo edition comes with the necessary tools. The different tools follow the naming style "format2john". The terminal below shows a few examples:

So as in this event ***we are interested in a password-protected PDF***; therefore, **pdf2john.pl** should do the job perfectly for you. In the terminal below, you can see how to create a hash challenge from a PDF file. This hash value can later be fed to john to crack it.

[illegible]

- **private.pdf**: The PDF file you want to crack.
- **pdf.hash**: The output file containing the hash for John to process.

Using a Custom Wordlist to Crack a Password-Protected PDF

If common wordlists like `rockyou.txt` fail to crack the password, you can create a **custom wordlist** based on clues about the password owner's preferences. Here's how to do it:

- **Create a Custom Wordlist**

Since Mayor Malware values certain things, we create a personalized wordlist with the following words:

- Fluffy
- FluffyCat
- Mayor
- Malware
- MayorMalware

Save these words in a file called `wordlist.txt`.

Use the Custom Wordlist with John

- Run John the Ripper using the custom wordlist and additional rules for password transformation.

Example command:

```
john --rules=single --wordlist=/home/user/A0C2024/wordlist.txt pdf.hash
```

- **--rules=single**: Applies advanced transformation rules to the wordlist, such as appending/prepending characters or substituting symbols.
- **--wordlist=wordlist.txt**: Specifies the custom wordlist we created.
- **pdf.hash**: The hash file extracted from the password-protected PDF.

```

user@ip-10-10-133-97:~/A0C2024$ john --rules=single --wordlist=wordlist.txt pdf.hash
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Cost 1 (revision) is 3 for all loaded hashes
Will run 2 OpenMP threads
Note: Passwords longer than 10 [worst case UTF-8] to 32 [ASCII] rejected
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
Enabling duplicate candidate password suppressor
M4y0rM4lw4r3J (private.pdf)
lg 0:00:00:00 DONE (2024-12-08 19:45) 4.348g/s 5286p/s 5286c/s 5286C/s mayored..afluffy
Use the "--show --format=PDF" options to display all of the cracked passwords reliably
Session completed.

```

Output

If successful, John will crack the password and display it in the terminal.

Now for the password we need to run the command of the pdf file to extract the flag for the question provided below.

Step 3: Access the Encrypted PDF

1. Open the PDF using the identified password.
2. Verify its content to gather evidence against Mayor Malware.

```

user@ip-10-10-133-97:~/A0C2024$ pdftotext private.pdf -upw M4y0rM4lw4r3J
user@ip-10-10-133-97:~/A0C2024$ ls -l
total 188
drwxrwxr-x 2 user user 4096 Nov 14 14:50 example files
-rw-rw-r-- 1 user user 35345 Nov 3 04:17 hash-id.py
-rw-rw-r-- 1 user user 65 Nov 3 09:00 hash1.txt
-rw-rw-r-- 1 user user 205 Dec 8 19:44 pdf.hash
-rw-r--r-- 1 user user 87159 Oct 31 13:31 private.pdf
-rw-rw-r-- 1 user user 46809 Dec 8 19:45 private.txt
-rw-rw-r-- 1 user user 44 Nov 14 15:22 wordlist.txt
user@ip-10-10-133-97:~/A0C2024$ head private.txt
transactions
THM{do not GET CAUGHT}
date
transaction ref
type
amount usd
Feb 4, 2022
F9613FAA
incoming

```

- There with the command we were able to extract the `private.txt` file from `private.pdf`.
- We got the flag for solving this lab by using the linux `cat` command with the name of file (`private.txt`).

Key Concepts Demonstrated

Hash Functions in Security

- Hashing ensures that passwords are not stored in plaintext.
- Adding salt further enhances protection against brute-force attacks.

Password Cracking Techniques

- **Brute-Force Attacks:** Systematically testing all possible password combinations.
- **Dictionary Attacks:** Using precompiled wordlists like `rockyou.txt`.
- **Hash Identification:** Tools like `hash-id` simplify the process of identifying hash algorithms.

Security Insights

- Poor password practices expose systems to breaches.
- Adding complexity (e.g., salt, secure hash functions) increases security.

Learning Outcomes

- Ability to identify and analyze hash types.
 - Understanding secure password storage methods.
 - Experience with tools like `john` and `hash-id` for cracking hashes.
 - Awareness of common security pitfalls and mitigation strategies.
-
-

Demonstration Summary

1. **Locate the Hash:** Navigate to `/home/user/A0C2024` and view the `hash1.txt` file.
2. **Identify Hash Type:** Use `hash-id` to determine potential algorithms.
3. **Crack the Hash:** Use `john` with the `rockyou.txt` wordlist.
4. **Access the File:** Retrieve the password and open the PDF for evidence collection.

This documentation provides a comprehensive guide for uncovering the password of Mayor Malware's protected file, enabling Glitch to gather crucial evidence.