CROSS-PLATFORM HASH ANALYSYS TOOL

Table of Contents

Project Overview

The Cross-Platform Password Cracking and Hash Analysis Platform is designed to assess password security across multiple operating systems (Linux, Windows, MacOS). This tool can:

1. Extract password hashes from different OS environments.

2. Analyze and attempt to crack weak passwords.

3. Provide real-time monitoring of password files for changes.

4. Generate comprehensive reports on password strength and security recommendations.

This platform aims to identify vulnerabilities related to weak passwords and help enhance organizational password policies.

Similar tools in industry:

1.) **L0pthcrack**: A cross-platform password auditing and recovery tool that analyzes password strength for Windows and other platforms with support for various hash types and password policies.

2.) **KeePassXC's Audit Feature**: A cross-platform password manager with built-in auditing features to analyze stored passwords for weak or reused entries, ensuring strong password practices.

3.) **COPS** (Computer Oracle and Password System): A Unix-based vulnerability scanner that includes password strength auditing by analyzing '/etc/passwd' and '/etc/shadow' files. It identifies weak passwords and system misconfigurations to improve overall security.

What makes this tool unique: it combines real-time monitoring of password hashes with active reporting and alerts for weak passwords, tailored for both Linux and Windows environments. Unlike most existing tools, which primarily focus on auditing static files or cracking hashes, this tool emphasizes proactive system health by integrating live hash analysis, cracking feedback, and actionable guidance for strengthening passwords. This real-time, cross-platform focus makes it distinct and more user-centric for ongoing system security.

limitations:

- Operating System Scope: Initially planned to support Linux, Windows, and macOS. However, the project now only focuses on Linux and Windows due to time constraints, excluding macOS.
- Hash Types and Extraction Methods: The project aimed to support various hash types like SHA-512, NTLM, and MD5. It now focuses only on SHA-512 hashes for Linux and NTLM hashes for Windows, limiting its versatility.
- Tool Choices for Hash Extraction and Cracking: Originally, multiple tools were considered for flexibility in hash extraction. The final project uses shadow file and John the Ripper for Linux and Mimikatz for NTLM extraction on Windows, reducing tool diversity.
- Password Cracking Constraints: Expected to offer comprehensive password-cracking capability with multiple wordlists. A time limit is now imposed to avoid lengthy cracking sessions, potentially missing complex passwords.
- Cross-Platform Integration of Results: Intended to provide centralized, unified results across platforms. Currently, each OS saves results locally, requiring manual consolidation if cross-platform results are needed.

Summary:

With proper research and analysis the project was taken forward with required environmental setup and necessary control over the system.

Environmental setup:

1.) Administrative access for windows and Linux OS.
2.) Necessary libraries or packages for smooth functionality of tool.
3.) Python to run the program which analyze your hashes.
4.) Configuration setup according to hash type.

Necessary libraries:

Here are the list of necessary libraries and packages:

Python Libraries:

1. ‘os’
2. ‘subprocess’
3. ‘platform’
4. ‘inotify_simple’ (for Linux)
5. ‘argparse’

Linux Packages:

1. ‘john’ (John the Ripper)
2. ‘rockyou.txt’ wordlist (Kali Linux default or manually added)

Windows Tools:

1. ‘Mimikatz’
2. ‘John the Ripper’ (Windows version)
3. Python (with added ‘Scripts’ path in the environment variables)

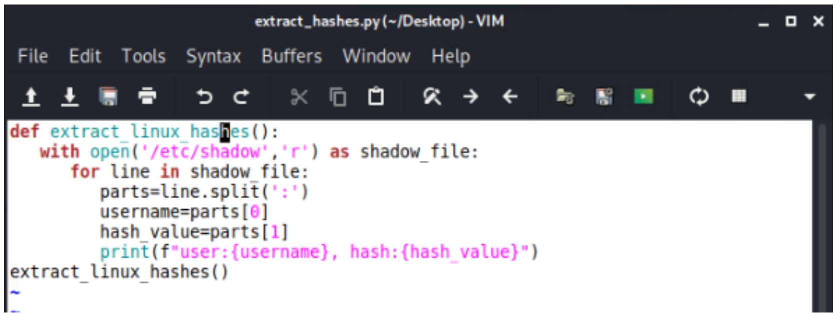
Configurational setup: for testing purposes I just used SHA 512 hashing. We can change our code according to our default hashing from our etc/pam.d dictionary in Linux. Windows takes NTLM by default. Permissions for shadows file for real-time monitoring purpose.

After necessary changes in your systems, make sure to restart them to update your changes.

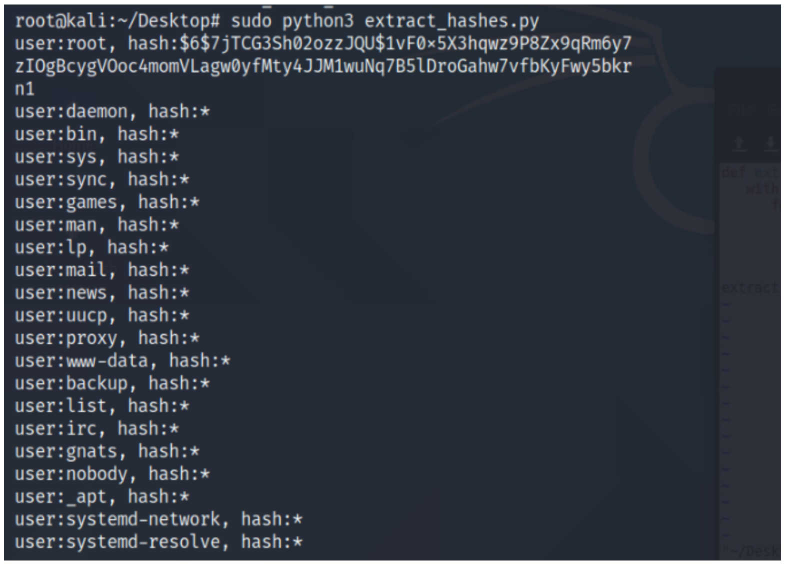
Steps involved:

1. Hash Extraction

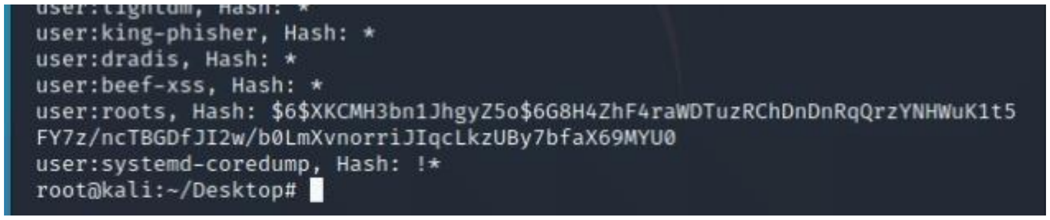
Linux: Successfully extracted password hashes from the ‘/etc/shadow’ file using Python.


extract_hashes.py (~/Desktop) - VIM
File Edit Tools Syntax Buffers Window Help

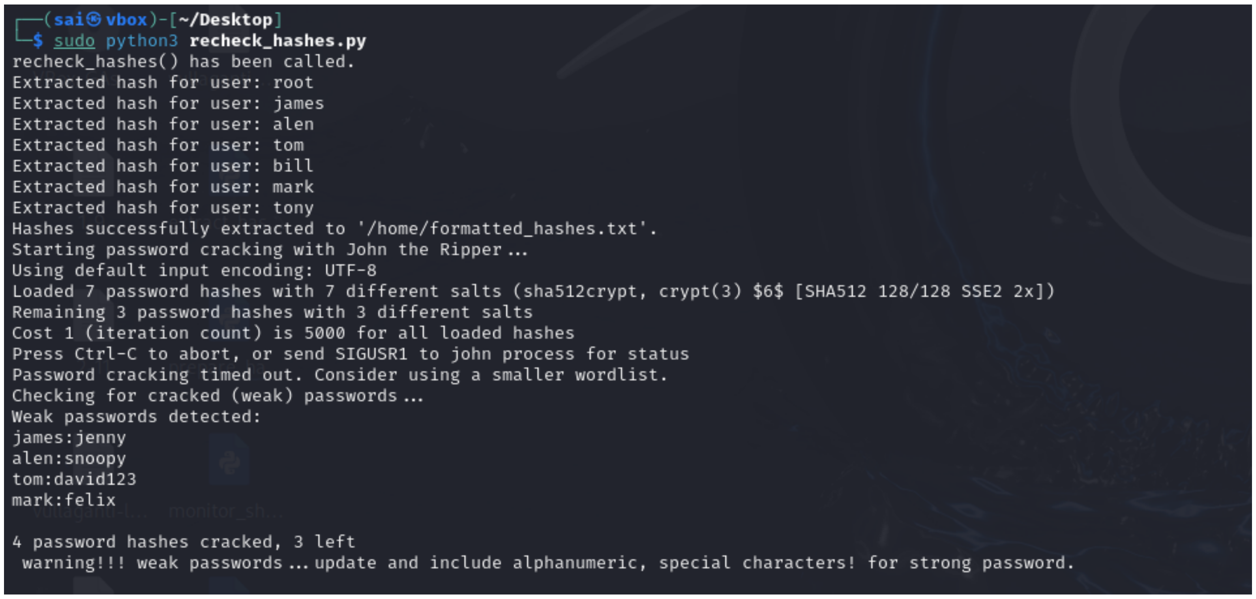
```
def extract_linux_hashes():
   with open('/etc/shadow','r') as shadow_file:
      for line in shadow_file:
         parts=line.split(':')
         username=parts[0]
         hash_value=parts[1]
         print(f"user:{username}, hash:{hash_value}")
extract_linux_hashes()
~
~
```



```
root@kali:~/Desktop# sudo python3 extract_hashes.py
user:root, hash:$6$7jTCG3Sh02ozzJQU$1vF0×5X3hqwz9P8Zx9qRm6y7
zIOgBcygVOoc4momVLagw0yfMty4JJM1wuNq7B5lDroGahw7vfbKyFwy5bkr
n1
user:daemon, hash:*
user:bin, hash:*
user:sys, hash:*
user:sync, hash:*
user:games, hash:*
user:man, hash:*
user:lp, hash:*
user:mail, hash:*
user:news, hash:*
user:uucp, hash:*
user:proxy, hash:*
user:www-data, hash:*
user:backup, hash:*
user:list, hash:*
user:irc, hash:*
user:gnats, hash:*
user:nobody, hash:*
user:_apt, hash:*
user:systemd-network, hash:*
user:systemd-resolve, hash:*
```



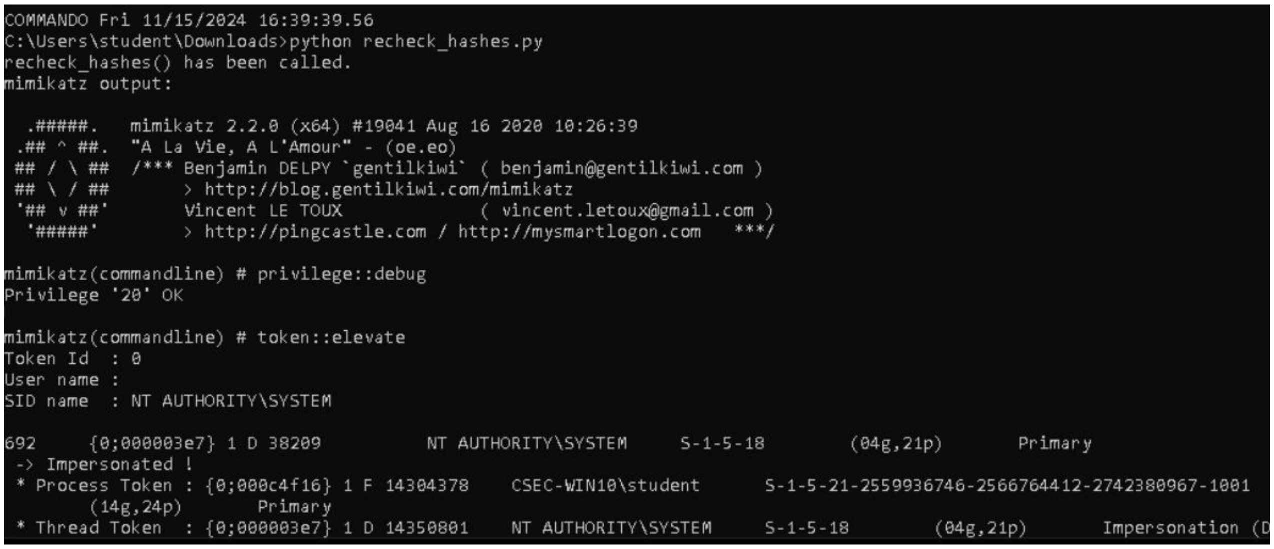
```
user:lightdm, Hash: *
user:king-phisher, Hash: *
user:dradis, Hash: *
user:beef-xss, Hash: *
user:roots, Hash: $6$XKCMH3bn1JhgyZ5o$6G8H4ZhF4raWDTuzRChDnDnRqQrzYNHWuK1t5
FY7z/ncTBGDfJI2w/b0LmXvnorriJIqcLkzUBy7bfaX69MYU0
user:systemd-coredump, Hash: !*
root@kali:~/Desktop# 
```



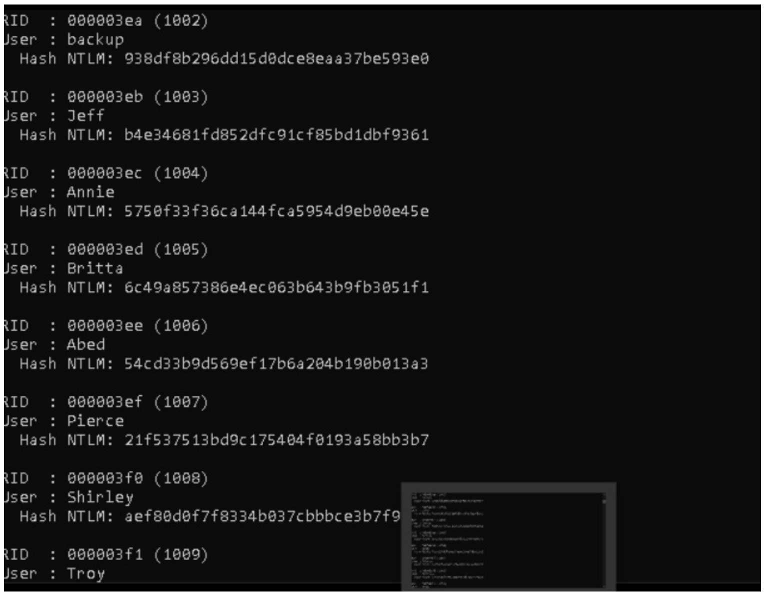
```
┌──(sai㉿vbox)-[~/Desktop]
└─$ sudo python3 recheck_hashes.py
recheck_hashes() has been called.
Extracted hash for user: root
Extracted hash for user: james
Extracted hash for user: alen
Extracted hash for user: tom
Extracted hash for user: bill
Extracted hash for user: mark
Extracted hash for user: tony
Hashes successfully extracted to '/home/formatted_hashes.txt'.
Starting password cracking with John the Ripper...
Using default input encoding: UTF-8
Loaded 7 password hashes with 7 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])
Remaining 3 password hashes with 3 different salts
Cost 1 (iteration count) is 5000 for all loaded hashes
Press Ctrl-C to abort, or send SIGUSR1 to john process for status
Password cracking timed out. Consider using a smaller wordlist.
Checking for cracked (weak) passwords...
Weak passwords detected:
james:jenny
alen:snoopy
tom:david123
mark:felix

4 password hashes cracked, 3 left
 warning!!! weak passwords... update and include alphanumeric, special characters! for strong password.
```

Windows: Encountered challenges with Windows SAM file extraction using Impacket. So, approaching other efficient way of extracting hashes through mimikatz.

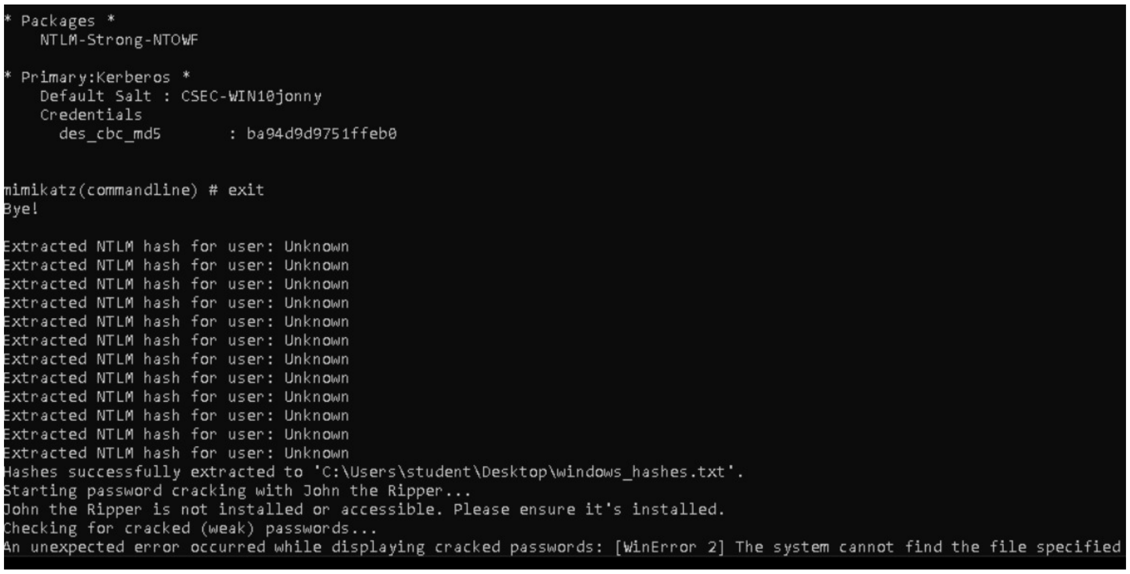
Steps followed:



```
COMMANDO Fri 11/15/2024 16:39:39.56
C:\Users\student\Downloads>python recheck_hashes.py
recheck_hashes() has been called.
mimikatz output:

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 16 2020 10:26:39
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # privilege::debug
Privilege '20' OK

mimikatz(commandline) # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

692     {0;000003e7} 1 D 38209          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Primary
 -> Impersonated !
 * Process Token : {0;000c4f16} 1 F 14304378    CSEC-WIN10\student      S-1-5-21-2559936746-2566764412-2742380967-1001
        (14g,24p)       Primary
 * Thread Token  : {0;000003e7} 1 D 14350801    NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Impersonation (D
```



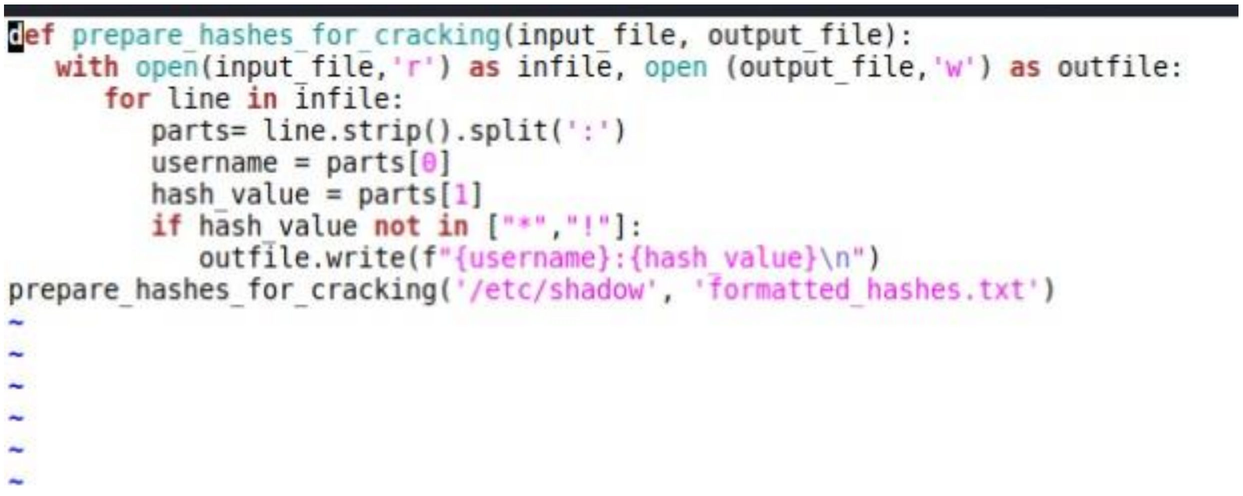
```
RID  : 000003ea (1002)
User : backup
  Hash NTLM: 938df8b296dd15d0dce8eaa37be593e0

RID  : 000003eb (1003)
User : Jeff
  Hash NTLM: b4e34681fd852dfc91cf85bd1dbf9361

RID  : 000003ec (1004)
User : Annie
  Hash NTLM: 5750f33f36ca144fca5954d9eb00e45e

RID  : 000003ed (1005)
User : Britta
  Hash NTLM: 6c49a857386e4ec063b643b9fb3051f1

RID  : 000003ee (1006)
User : Abed
  Hash NTLM: 54cd33b9d569ef17b6a204b190b013a3

RID  : 000003ef (1007)
User : Pierce
  Hash NTLM: 21f537513bd9c175404f0193a58bb3b7

RID  : 000003f0 (1008)
User : Shirley
  Hash NTLM: aef80d0f7f8334b037cbbbce3b7f9

RID  : 000003f1 (1009)
User : Troy
```



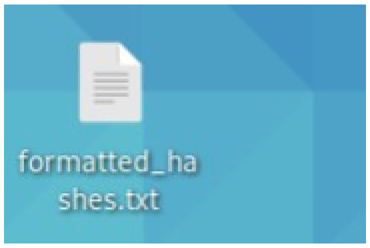
```
* Packages *
    NTLM-Strong-NTOWF

* Primary:Kerberos *
    Default Salt : CSEC-WIN10jonny
    Credentials
      des_cbc_md5       : ba94d9d9751ffeb0


mimikatz(commandline) # exit
Bye!

Extracted NTLM hash for user: Unknown
Extracted NTLM hash for user: Unknown
Extracted NTLM hash for user: Unknown
Extracted NTLM hash for user: Unknown
Extracted NTLM hash for user: Unknown
Extracted NTLM hash for user: Unknown
Extracted NTLM hash for user: Unknown
Extracted NTLM hash for user: Unknown
Extracted NTLM hash for user: Unknown
Extracted NTLM hash for user: Unknown
Extracted NTLM hash for user: Unknown
Extracted NTLM hash for user: Unknown
Hashes successfully extracted to 'C:\Users\student\Desktop\windows_hashes.txt'.
Starting password cracking with John the Ripper...
John the Ripper is not installed or accessible. Please ensure it's installed.
Checking for cracked (weak) passwords...
An unexpected error occurred while displaying cracked passwords: [WinError 2] The system cannot find the file specified
```

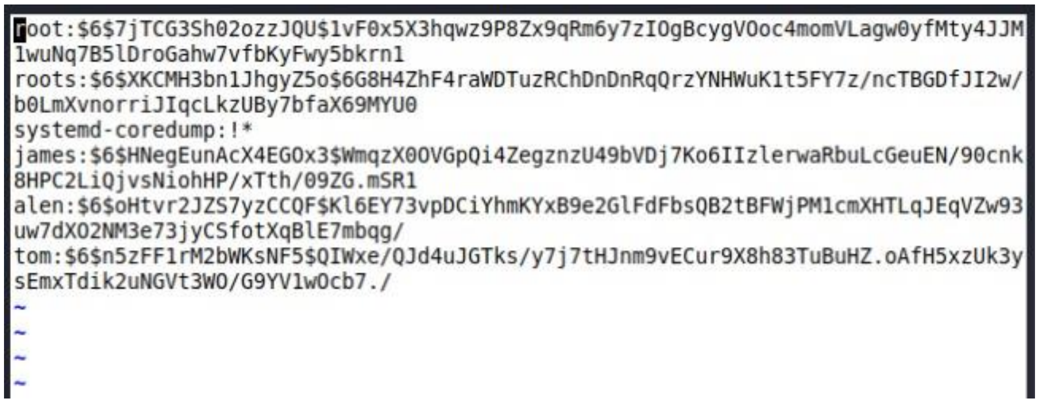
2. Hash Preparation and Formatting

Created a formatted file (‘formatted_hashes.txt’) for the extracted hashes, readying them for use with password-cracking tools.

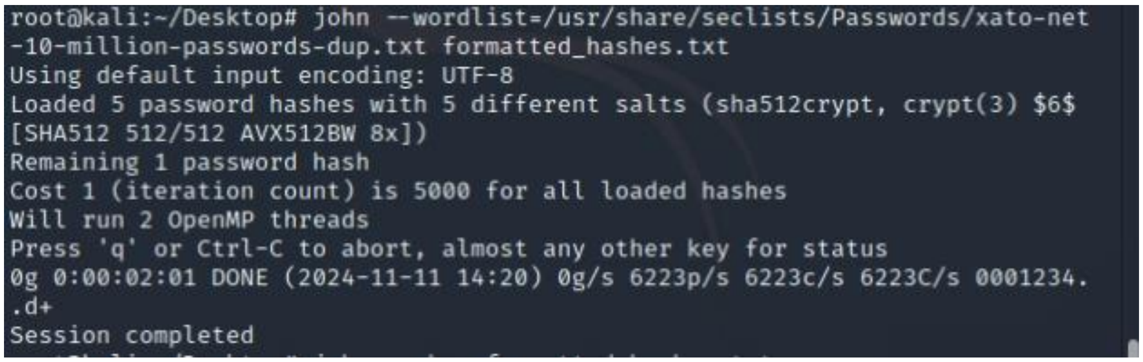


```
def prepare_hashes_for_cracking(input_file, output_file):
    with open(input_file,'r') as infile, open (output_file,'w') as outfile:
        for line in infile:
            parts= line.strip().split(':')
            username = parts[0]
            hash_value = parts[1]
            if hash_value not in ["*","!"]:
                outfile.write(f"{username}:{hash_value}\n")
prepare_hashes_for_cracking('/etc/shadow', 'formatted_hashes.txt')
~
~
~
~
~
~
```





```
root:$6$7jTCG3Sh02ozzJQU$1vF0x5X3hqwz9P8Zx9qRm6y7zIOgBcygVOoc4momVLagw0yfMty4JJM
1wuNq7B5lDroGahw7vfbKyFwy5bkrn1
roots:$6$XKCMH3bn1JhgyZ5o$6G8H4ZhF4raWDTuzRChDnDnRqQrzYNHWuK1t5FY7z/ncTBGDfJI2w/
b0LmXvnorriJIqcLkzUBy7bfaX69MYU0
systemd-coredump:!*
james:$6$HNegEunAcX4EGOx3$WmqzX0OVGpQi4ZegznzU49bVDj7Ko6IIzlerwaRbuLcGeuEN/90cnk
8HPC2LiQjvsNiohHP/xTth/09ZG.mSR1
alen:$6$oHtvr2JZS7yzCCQF$Kl6EY73vpDCiYhmKYxB9e2GlFdFbsQB2tBFWjPM1cmXHTLqJEqVZw93
uw7dXO2NM3e73jyCSfotXqBlE7mbqg/
tom:$6$n5zFF1rM2bWKsNF5$QIWxe/QJd4uJGTks/y7j7tHJnm9vECur9X8h83TuBuHZ.oAfH5xzUk3y
sEmxTdik2uNGVt3WO/G9YV1wOcb7./
~
~
~
~
```


```
root@kali:~/Desktop# john --wordlist=/usr/share/seclists/Passwords/xato-net
-10-million-passwords-dup.txt formatted_hashes.txt
Using default input encoding: UTF-8
Loaded 5 password hashes with 5 different salts (sha512crypt, crypt(3) $6$
[SHA512 512/512 AVX512BW 8x])
Remaining 1 password hash
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:02:01 DONE (2024-11-11 14:20) 0g/s 6223p/s 6223c/s 6223C/s 0001234.
.d+
Session completed
```

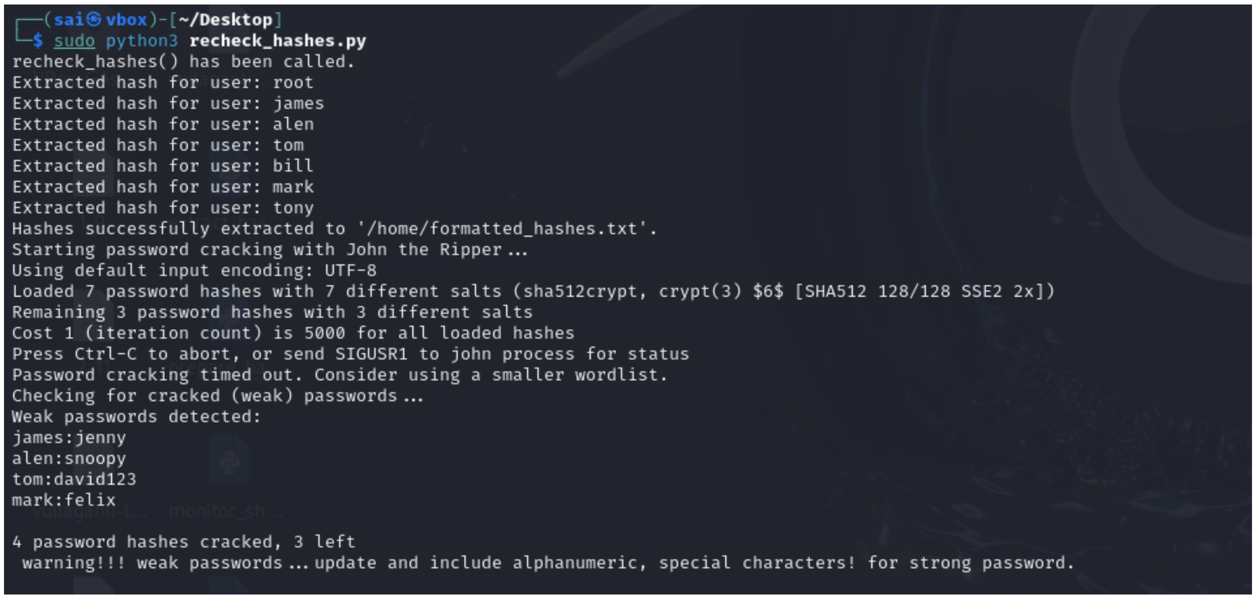
Ensured compatibility with cracking tools by formatting entries with only usernames and hashes.

3. Password Cracking

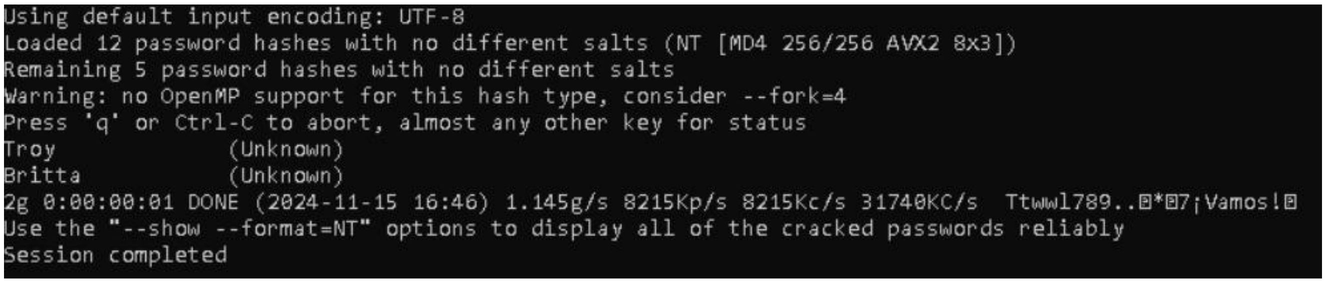
Used John the Ripper to crack hashes, successfully cracking weaker passwords.

Experimented with wordlists like ‘rockyou.txt’ to maximize success and identified patterns in weaker passwords.

Hash analysis in Linux


```
┌──(sai㉿vbox)-[~/Desktop]
└─$ sudo python3 recheck_hashes.py
recheck_hashes() has been called.
Extracted hash for user: root
Extracted hash for user: james
Extracted hash for user: alen
Extracted hash for user: tom
Extracted hash for user: bill
Extracted hash for user: mark
Extracted hash for user: tony
Hashes successfully extracted to '/home/formatted_hashes.txt'.
Starting password cracking with John the Ripper ...
Using default input encoding: UTF-8
Loaded 7 password hashes with 7 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])
Remaining 3 password hashes with 3 different salts
Cost 1 (iteration count) is 5000 for all loaded hashes
Press Ctrl-C to abort, or send SIGUSR1 to john process for status
Password cracking timed out. Consider using a smaller wordlist.
Checking for cracked (weak) passwords ...
Weak passwords detected:
james:jenny
alen:snoopy
tom:david123
mark:felix

4 password hashes cracked, 3 left
 warning!!! weak passwords ... update and include alphanumeric, special characters! for strong password.
```

Hash analysys in windows:

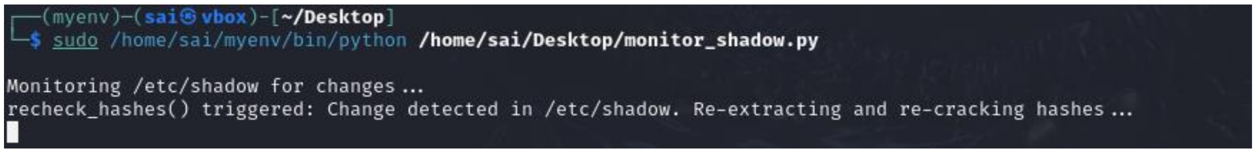


```
Using default input encoding: UTF-8
Loaded 12 password hashes with no different salts (NT [MD4 256/256 AVX2 8x3])
Remaining 5 password hashes with no different salts
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
Troy             (Unknown)
Britta           (Unknown)
2g 0:00:00:01 DONE (2024-11-15 16:46) 1.145g/s 8215Kp/s 8215Kc/s 31740KC/s  Ttwwl789..□*□7¡Vamos!□
Use the "--show --format=NT" options to display all of the cracked passwords reliably
Session completed
```

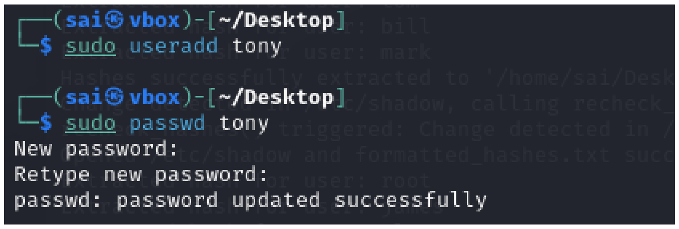
4. Real-Time Monitoring

Planned a monitoring system for Linux's '/etc/shadow' file to detect any changes in real time, such as new users or password updates. Can make it

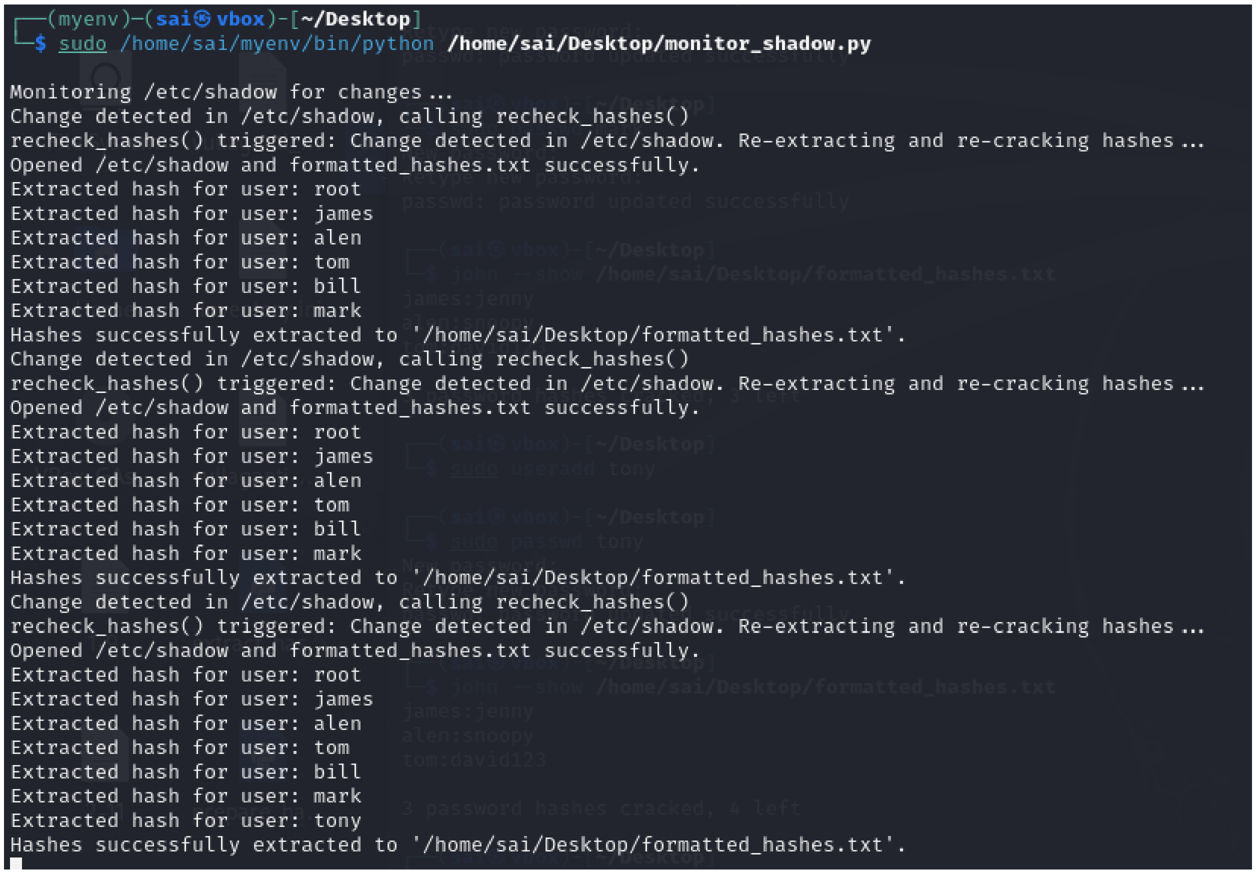
Monitoring in linux



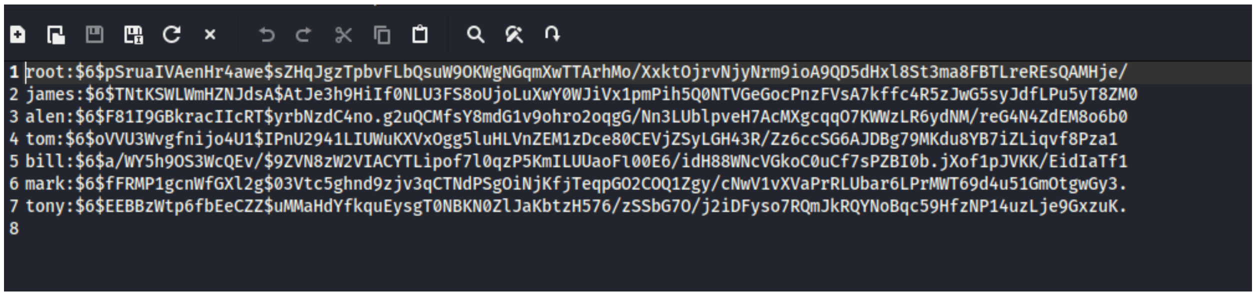
```
┌──(myenv)─(sai㉿vbox)-[~/Desktop]
└─$ sudo /home/sai/myenv/bin/python /home/sai/Desktop/monitor_shadow.py

Monitoring /etc/shadow for changes...
recheck_hashes() triggered: Change detected in /etc/shadow. Re-extracting and re-cracking hashes...
```



```
┌──(sai㉿vbox)-[~/Desktop]
└─$ sudo useradd tony

┌──(sai㉿vbox)-[~/Desktop]
└─$ sudo passwd tony
New password:
Retype new password:
passwd: password updated successfully
```



```
┌──(myenv)─(sai㉿vbox)-[~/Desktop]
└─$ sudo /home/sai/myenv/bin/python /home/sai/Desktop/monitor_shadow.py

Monitoring /etc/shadow for changes...
Change detected in /etc/shadow, calling recheck_hashes()
recheck_hashes() triggered: Change detected in /etc/shadow. Re-extracting and re-cracking hashes...
Opened /etc/shadow and formatted_hashes.txt successfully.
Extracted hash for user: root
Extracted hash for user: james
Extracted hash for user: alen
Extracted hash for user: tom
Extracted hash for user: bill
Extracted hash for user: mark
Hashes successfully extracted to '/home/sai/Desktop/formatted_hashes.txt'.
Change detected in /etc/shadow, calling recheck_hashes()
recheck_hashes() triggered: Change detected in /etc/shadow. Re-extracting and re-cracking hashes...
Opened /etc/shadow and formatted_hashes.txt successfully.
Extracted hash for user: root
Extracted hash for user: james
Extracted hash for user: alen
Extracted hash for user: tom
Extracted hash for user: bill
Extracted hash for user: mark
Hashes successfully extracted to '/home/sai/Desktop/formatted_hashes.txt'.
Change detected in /etc/shadow, calling recheck_hashes()
recheck_hashes() triggered: Change detected in /etc/shadow. Re-extracting and re-cracking hashes...
Opened /etc/shadow and formatted_hashes.txt successfully.
Extracted hash for user: root
Extracted hash for user: james
Extracted hash for user: alen
Extracted hash for user: tom
Extracted hash for user: bill
Extracted hash for user: mark
Extracted hash for user: tony
Hashes successfully extracted to '/home/sai/Desktop/formatted_hashes.txt'.
```



```
root:$6$pSruaIVAenHr4awe$sZHqJgzTpbvFLbQsuW9OKWgNGqmXwTTArhMo/XxktOjrvNjyNrm9ioA9QD5dHxl8St3ma8FBTLreREsQAMHje/
james:$6$TNtKSWLWmHZNJdsA$AtJe3h9HiIf0NLU3FS8oUjoLuXwY0WJiVx1pmPih5Q0NTVGeGocPnzFVsA7kffc4R5zJwG5syJdfLPu5yT8ZM0
alen:$6$F81I9GBkracIIcRT$yrbNzdC4no.g2uQCMfsY8mdG1v9ohro2oqgG/Nn3LUblpveH7AcMXgcqqO7KWWzLR6ydNM/reG4N4ZdEM8o6b0
tom:$6$oVVU3Wvgfnijo4U1$IPnU2941LIUWuKXVxOgg5luHLVnZEM1zDce80CEVjZSyLGH43R/Zz6ccSG6AJDBg79MKdu8YB7iZLiqvf8Pza1
bill:$6$a/WY5h9OS3WcQEv/$9ZVN8zW2VIACYTLipof7l0qzP5KmILUUaoFl00E6/idH88WNcVGkoC0uCf7sPZBI0b.jXof1pJVKK/EidIaTf1
mark:$6$fFRMP1gcnWfGXl2g$03Vtc5ghnd9zjv3qCTNdPSgOiNjKfjTeqpGO2COQ1Zgy/cNwV1vXVaPrRLUbar6LPrMWT69d4u51GmOtgwGy3.
tony:$6$EEBBzWtp6fbEeCZZ$uMMaHdYfkquEysgT0NBKN0ZlJaKbtzH576/zSSbG7O/j2iDFyso7RQmJkRQYNoBqc59HfzNP14uzLje9GxzuK.
```

Unanticipated Challenges and Solutions

Cracking Limitations: Some passwords were tougher to crack using a standard wordlist. To improve results, we may incorporate more advanced cracking rules or brute-force methods if needed.

Preliminary Findings and Observations

Password Weakness Patterns: Initial analysis shows that weak passwords often include predictable patterns (e.g., common words, short lengths). This highlights the need for stronger password guidelines.

Security Implications: Weak passwords across systems can pose significant security risks, emphasizing the importance of enforcing complex password requirements and educating users on best practices.

achieved objectives:

1. Extract password hashes from different OS environments.

2. Analyze and attempt to crack weak passwords.

3. Provide real-time monitoring of password files for changes.

4. Generate comprehensive reports on password strength and security recommendations.

Future work:

1. Adding MAC OS compatibility and real time monitoring for it.
2. Making it user friendly through a simple application which analyzes and reports system passwords.
3. Adding more features, like able to crack password irrespective of its hash type.
4. Adding more packages to crack hard passwords.

Resources:

1) John the Ripper: Password cracking tool supporting multiple hash formats. https://www.openwall.com/john/ .
2) 2. Mimikatz: Tool for extracting NTLM hashes on Windows. https://github.com/gentilkiwi/mimikatz
3) RockYou Wordlist: Preinstalled in Kali Linux (` /usr/share/wordlists/rockyou.txt`) A widely used wordlist for password cracking.
4) NIST Password Guidelines: Best practices for password security and management. https://pages.nist.gov/800-63-3/
5) Python Subprocess Module: For executing system commands in Python scripts.

https://docs.python.org/3/library/subprocess.html

6) Linux Shadow File Documentation: Information about `/etc/shadow` file structure. https://linux.die.net/man/5/shadow
7) SecLists: Repository of wordlists for password and hash cracking https://github.com/danielmiessler/SecLists.

GitHub: https://github.com/saipradyumna16/Cross-Platform-Hash-Analysis-Tool