## Lab 5: How to Crack Passwords with Hashcat

### Scenario

As a cybersecurity practitioner for CyberFort Security Solutions, you have been engaged to evaluate the resilience of the company’s password policies and authentication practices. During your assessment, you obtained a file containing password hashes from a compromised Linux system. These hashes were extracted from a target machine during a simulated breach and represent user account credentials stored using the MD5 hashing algorithm. Since weak or commonly used passwords can put the organization at risk, you have been tasked with testing whether these hashes can be cracked using an offline attack. The purpose of this exercise is to demonstrate how attackers can leverage password-cracking techniques to recover plaintext credentials if strong password policies are not enforced.

### Solution

CyberFort Security Solutions assigned you, as a certified cybersecurity practitioner, the responsibility of using Hashcat to perform a dictionary-based password cracking attack. You first prepared a list of target hashes stored in a file and selected the popular wordlist rockyou.txt for the attack. By executing Hashcat with the appropriate options, you attempted to match each hash against potential passwords from the wordlist. During the test, Hashcat successfully cracked several hashes, revealing their plaintext passwords, while others remained uncracked due to the absence of matching entries in the wordlist. This exercise highlights the importance of implementing complex, unique passwords, enforcing multi-factor authentication, and conducting regular password audits to mitigate the risk of password cracking attacks in real-world environments.

|  |  |
| --- | --- |
| 1. Turn on **ParrotOS** virtual machine. Open a **Terminal**, and execute the **sudo su** command to run programs with root privileges.    2. We will create a sample set of hashes and then use a dictionary to crack these hashes. Execute the **pluma target\_hashed.txt**. Copy and paste the below-provided hashes in a text editor, then save the file and close it. These hashes comprise 7 different passwords, which we will attempt to crack.   |  | | --- | | dc647eb65e6711e155375218212b3964  eb61eead90e3b899c6bcbe27ac581660  958152288f2d2303ae045cffc43a02cd  2c9341ca4cf3d87b9e4eb905d6a3ec45  75b71aa6842e450f12aca00fdf54c51d  031cbcccd3ba6bd4d1556330995b8d08  b5af0b804ff7238bce48adef1e0c213f |   3. Execute the following command: **hashcat -h | more** to see all options that are available in Hashcat.  4. Open any browser in the ParrotOS virtual machine. In the search type, **download rockyou.txt** and press **Enter**. Then click on the first link, **Rockyou.txt wordlist** of GitHub, to download a text file with password wordlists.    5. Execute the following command: **hashcat -m 0 -a 0 -o cracked.txt target\_hashes.txt Downloads/rockyou.txt** to crack the password hashes.  Let us examine each of these options in detail.   * The **-m 0** option specifies that Hashcat will attempt to crack MD5 hash types * The **-a 0** option indicates the use of a dictionary attack * The **-o cracked.txt** option designates the output file where the cracked passwords will be stored * The **target\_hashes.txt** file contains the list of hashes to be cracked * The **/usr/share/wordlists/rockyou.txt** file serves as the dictionary wordlist for the attack   6. Execute the following command: **cat cracked.txt** to see all the passwords that are cracked by Hashcat.  We can observe that for each successfully cracked password, Hashcat displays both the hash and its corresponding plaintext value. In this case, Hashcat successfully cracked 5 out of the 7 hashes. The remaining two passwords were not present in the wordlist and, therefore, could not be cracked.  An examination of the attack details shows that Hashcat processed over 4 million password candidates in just one second. This highlights the critical importance of using strong, secure passwords to reduce the risk of compromise in the event of a security breach. |