Hascat Kurulumu

Resmi sitesinden hashcat binaries'i (*hashcat-2.00.7z* dosyasını) indir. Çalıştırmak için;

> cd hashcat-2.00

> ./hashcat-cli64.bin

Sözlük İndirme

Kapsamlı bir sözlük olan rockyou.txt'i indir:

> wget http://scrapmaker.com/data/wordlists/dictionaries/rockyou.txt

Hashcat Kullanımı

1.

Varsayalım ki bismillahirrahmanirrahim parolasının online MD5 generator ile hash'ini alıp hash.txt'e koyup rockyou.txt sözlüğüyle kırmak istiyoruz. Bu durumda;

hash.txt:

819f8e0921e1d367db3d8df1ec219fdd

Terminal:

> cd hashcat-2.00

> ./hashcat-cli64.bin -m 0 -a 0 hash.txt rockyou.txt

Output:

…

819f8e0921e1d367db3d8df1ec219fdd:bismillahirrahmanirrahim

All hashes have been recovered

...

Görüldüğü üzere hash kırıldı ve parolanın bismillahirrahmanirrahim olduğu sözlük ile anlaşıldı.

NOT:Sözlük bismillahirrahmanirrahim kelimesini içermekte! Sublime ile CTRL+F yapıp bakabilirsin

Hashcat Hakkında Detaylı Açıklamalar

1.

hashcat'in parametre olarak alacağı options'lardan bazısı burada bahsedilecektir:

--hash-mode=NUM veya -m parametresi *(Kırılacak hash'in tipini alır)*

-m 0 MD5 algoritması ile hash'i kırmaya çalış denmiş olur

-m 1700 SHA512 algoritması ile hash'i kırmaya çalış denmiş olur.

…

Tüm algoritmaların numaralarına

> ./hashcat-cli64.bin --help

ile ulaşabilirsin. Hashcat varsayılan olarak MD5 algoritması ile hash'leri kırmaya çalışır.

--attack-mode=NUM veya -a parametresi *(Hash'i kırmak için kullanılacak method)*

-a 0 Straight

Sözlükteki tüm kelimeler hash'lenerek parola hash'iyle karşılaştırılır.

-a 1 Combination

Sözlükteki kelimelerin birbirleriyle kombinasyonları alınarak karşılaştırma yapılır. Mesela test ve hasan kelimelerini birleştirir ve hash'ini alıp parolanın hash'iyle karşılaştırır.

-a 2 Toggle-Case

Her bir satırın büyük küçük harf dönüşümleriyle kombinasyonları parola hash'iyle karşılaştırılır (Örn; test, Test, tEst, teSt,...)

-a 3 Brute-Force

Belirtilen koşullara (karakter setine, parola uzunluğuna vs...) uyan her türlü kombinasyon denenir.

-a 4 Permutation

Kelimelerden harfleri alır ve permütasyonlarını dener. Örneğin sözlüğün bir satırındaki kelime abc olsun. Permütasyonları abc, acb, bca, bac, cab, cba. (--perm-min=SAYI ve --perm-max=SAYI ile permütasyona sokulacak kelimelerin harf sayısı belirtilebilir).

-a 5 Table-Lookup

Kelimeyi harflerine ayırır ve --table-file ile belirtilen dönüşümleri harflere uygular. Mesela test kelimesi t3St olabilir. Bu yeni kombinasyonları parola hash'iyle karşılaştırarak parolayı kırmaya çalışır. Bunun bir örneğine birazdan değinilecektir.

2.

Table-Lookup Modunda Hash Kırma Örneği

İlk olarak kendimize ait bir table oluşturalım (Dilenilirse hashcat'in mevcut table dosyaları da kullanılabilir):

deneme.table

```
t=t
t=T
t=7
e=e
e=E
e=3
s=s
s=S
s=5
```

Bu table dosyamızı hascat-2.00/tables/ dizinine taşıyalım. Ardından hash dosyamıza MD5 algoritmasından geçmiş beş tane parola koyalım.

hash2.txt

```
751ec45015a704a39dc403001c963e97
e86e107b113b0f830b9b817b4a9addb8
f61954c634231f8bbf0264d9797df9fa
a0c58991fd9a340393d6a021bc490540
cc03e747a6afbbcbf8be7668acfebee5
05a671c66aefea124cc08b76ea6d30bb
```

Yukarıdaki her bir satır bir parola anlamına gelmektedir. Şimdi bu hash'leri oluşturduğumuz table dosyasının rockyou.txt sözlüğüne katacağı ekstra kombinasyonlarıyla beraber kırmaya çalışalım:

Terminal:

```
> ./hashcat-cli64.bin -m 0 -a 5 --table-file=tables/deneme.table hash2.txt rockyou.txt
```

Output:

```
cc03e747a6afbbcbf8be7668acfebee5:test123
05a671c66aefea124cc08b76ea6d30bb:testtest
e86e107b113b0f830b9b817b4a9addb8:t3st
a0c58991fd9a340393d6a021bc490540:73s7
f61954c634231f8bbf0264d9797df9fa:TEsT
```

Görüldüğü üzere rockyou.txt sözlüğünde olmayan t3st, 73s7 gibi kelimeler table sayesinde oluşturuldu ve hash'lerle kıyaslandı. Sonuç olarak hash'ler bu kelimelerle kırılmıştır.

Kaynak: Tez Raporları/İncelenmiş Makaleler/BGA/Sızma Testlerinde Brute Force.docx

Hashcat Kullanım Örnekleri

a) Hashcat ile MySQL Password'ü Kırma

Varsayalım ki ele geçirdiğimiz MySQL parola özeti şu şekildedir:

mysql_hash.txt

6691484EA6B50DDDE1926A220DA01FA9E575C18A

Bu parolayı kırabilmek için hashcat'ten yararlanabiliriz:

> hashcat --help | grep MySQL

Output:
200 = MySQL
300 = MySQL4.1/MySQL5

Görüldüğü üzere kullanabileceğimiz iki alternatif mod vardır. Eğer birinci mod işe yaramazsa ikincisini kullanabilirsiniz. Böylece parolayı kırmış olacaksınız:

> sudo su

>./hashcat-cli64.bin -m 200 -a 0 /home/hefese/hashcat-2.00/hash.txt /home/hefese/hashcat-2.00/rockyou.txt
>./hashcat-cli64.bin -m 300 -a 0 /home/hefese/hashcat-2.00/hash.txt /home/hefese/hashcat-2.00/rockyou.txt

Output:
[…]

6691484ea6b50ddde1926a220da01fa9e575c18a:**abc123**

All hashes have been recovered

[…]

Görüldüğü üzere şifrenin abc123 olduğu sözlük saldırısı ile tespit edilmiştir.

Kaynak: Tez Raporu/Literatür Taraması/İncelenmiş Makaleler/BGA/MySQL Sızma Testi.docx

b) Brute Force ile MD5 hash'ini Kırma

Diyelim ki “deneme” string'inin MD5 hash'ini aldık.

> echo -n "deneme" | openssl md5 // -n means do not output the newline

Output:

8f10d078b2799206cfe914b32cc6a5e9

Bu hash'i brute force ile kırmak istiyoruz . Bu durumda önce oluşturduğumuz hash'i hash_md5.txt adlı bir dosyaya atalım.

hash_md5.txt

8f10d078b2799206cfe914b32cc6a5e9

Ardından hashcat ile brute force atağını başlatalım:

> cd hashcat-2.0.0/

> ./hashcat-cli64.bin -m 0 -a 3 hash_md5.txt --increment --increment-min=1 --increment-max=6 ?l?l?l?l?l?l

?l demek "abcdefghijklmnopqrstuvwxyz" karakterlerinden birisi olabilir demektir. Ardı ardına 6 tane ?l demek ise minimum 1 karakterliden maksimum 6 karakterliye kadarki tüm karakterlerin sadece a'dan z'ye harf alabileceğini söylemek demektir. Bu kurallar doğrultusunda Hashcat ile brute force atağı başlatılır ve sonunda şu çıktı bizi karşılar:

8f10d078b2799206cfe914b32cc6a5e9:deneme

All hashes have been recovered

Input.Mode: Mask (?l?l?l?l?l?l) [6]

Index.....: 0/1 (segment), 308915776 (words), 0 (bytes)

Recovered.: 1/1 hashes, 1/1 salts

Speed/sec.: - plains, 36.53M words

Progress..: 142446888/308915776 (46.11%)

Running...: 00:00:00:04

Estimated.: 00:00:00:04

Görüldüğü üzere Brute Force atağı %46 aşamasındayken kırmaya çalıştığı hash'in karşılığını bulmuştur ve ekrana basmıştır:

8f10d078b2799206cfe914b32cc6a5e9:deneme

Aslında --increment, --increment-min ve --increment-max argumanları olmadan da brute force atağı yapabilirdik. Örneğin;

> ./hashcat-cli64.bin -m 0 -a 3 hash_md5.txt ?l?l?l?l?l?l

Yukarıdaki kodlama ile minumum 6 karakterli ve maksimum 6 karakterli, her karakteri sadece a'dan z'ye harf alabilen bir brute force atağı yapmış oluruz. Ancak biz 1'den 6'ya doğru arttırmalı bir brute force atağı yapmak istediğimiz için --increment, --increment-min ve --increment-max parametrelerini kullandık. Bu üç parametre varken brute force sırasında ?l?l?l?l?l?l maskının gösterdiği 6 karakterliden daha az karakterli kombinasyonlar denendiğinde charset tayini için şöyle bir method izlenir: Örneğin 2 karakterli kombinasyonlarda ?l?l?l?l?l?l charset'inin (**?l?l**)?l?l?l?l kısmı kullanılır. 3 karakterli kombinasyonlarda ?l?l?l?l?l?l charset'inin (**?l?l?l**)?l?l?l kısmı kullanılır. Bu şekilde arttırmalı kombinasyonların charset'leri belirlenir.

increment'in farkını anlayabilmek için "den" string'inin hash'ini hash2_md5.txt dosyasına koyup aşağıdaki gibi kırmayı deneyelim.

> ./hashcat-cli64.bin -m 0 -a 3 hash2_md5.txt ?l?l?l?l?l?l

Output:

```
Input.Mode: Mask (?l?l?l?l?l?l) [6]
Index.....: 0/1 (segment), 308915776 (words), 0 (bytes)
Recovered.: 0/1 hashes, 0/1 salts
Speed/sec.: 36.52M plains, 36.52M words
Progress..: 308915776/308915776 (100.00%)
Running...: 00:00:00:08
Estimated.: --:--:--:--
```

Görüldüğü üzere yukarıdaki kod "den" string'ine ait hash'i kıramamıştır. Çünkü yukarıdaki kod minimum 6 karakterli ve maksimum 6 karakterli bir brute force yapmaktadır. Bizim şifremiz ("den") ise üç karakterlidir. Şimdi kodumuza increment ekleyelim ve bir karakterliden altı karakterliye doğru kırma işlemi yapalım:

> ./hashcat-cli64.bin -m 0 -a 3 hash2_md5.txt --increment --increment-min=1

--increment-max=6 ?l?l?l?l?l?l

Output:

32ce9c04a986b6360b0ea1984ed86c6c:den

All hashes have been recovered

Input.Mode: Mask (?l?l?l) [3]
Index.....: 0/1 (segment), 17576 (words), 0 (bytes)
Recovered.: 1/1 hashes, 1/1 salts
Speed/sec.: - plains, 15.49k words
Progress..: 15487/17576 (88.11%)
Running...: 00:00:00:01
Estimated.: --:--:--:--

Görüldüğü üzere hash'in karşılığının "den" string'i olduğu tespit edilebilmiştir. Dolayısıyla diyebiliriz ki sadece mask kullanıldığında yapılan brute force saldırısının hem minimum değeri hem de maksimum değeri mask'ın uzunluğu kadar oluyorken mask'la birlikte increment kullanıldığında ise yapılan brute force saldırısının hem minimum değeri hem de maksimum değeri elimizle değiştirilebilmektedir.

Not: --increment-max değeri en fazla mask uzunluğu kadar olmalıdır. Eğer --increment-max değeri mask'ı geçerse esas alınan maksimum karakter limiti uzunluğu mask'ın uzunluğu olacaktır. Ayrıca hashcat'te Brute force için mask belirtmek zorunludur.

Not 2: ?l gibi daha başka charset'ler de vardır:

?l = abcdefghijklmnopqrstuvwxyz
?u = ABCDEFGHIJKLMNOPQRSTUVWXYZ
?d = 0123456789
?s = «space»!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
?a = ?l?u?d?s

Bu charset'lerin karması da kullanılabilmektedir. Örneğin;

?l?l?l?u?u?d?d

Böylece her karakterin charset'ini ayrı ayrı belirtmiş oluruz ve brute force atağını ona göre daha spesifik yapabiliriz.

> ./hashcat-cli64.bin -m 0 -a 3 hashDosyasi.txt ?l?l?l?u?u?d?d // Bu uygulanmamıştır (!)
// Temsil için konmuştur.

Yukarıdaki brute force minimum 6 karakterli, maksimum 6 karakterli ve belirtilen charset çeşitliliğine uygun olarak gerçekleşecektir. Peki ya arttırmalı brute force parametrelerini ekleseydik bu charset çeşitliliği nasıl okunurdu dersin? Bu durumda örneğin üç karakterli kombinasyonlar için yukarıdaki maskın (**?l?l?l**)?u?u?d?d kısmı charset olarak kullanılacakken dört karakterli kombinasyonlar için yukarıdaki maskın (**?l?l?l?u**)?u?d?d kısmı charset olarak kullanılacaktır. Yani yine aynı mantığa göre mask'ın belirttiği charset'ler kombinasyonun karakter sayısına göre uygun şekilde kullanılacaktır.

Uyarı: Normalde echo ile "deneme" string'ini openssl'e verdiğimizde şöyle bir hash ekrana gelmektedir:

> echo -n "deneme" | openssl md5 // -n : do not output the newline

Output:

8f10d078b2799206cfe914b32cc6a5e9

Ancak "deneme" stringini openssl'e aşağıdaki gibi direk verdiğimizde farklı bir hash ekrana gelmektedir:

> openssl md5 "deneme"

Output:

83ff036ac533544ddf84fb22edf62c2d

Bu farklılığın nedeni muhtemelen openssl'in direk kullanımında oluşturacağı hash'e salt değeri ekliyor oluşundandır. Online md5 generator'lar echo ile oluşturulan hash'in aynısını verdikleri için ve online md5 generator'lar openssl'in direk kullanımında oluşan hash'i vermedikleri için doğru hash echo ile oluşturulan hash'tir diyebiliriz. Zaten echo ile oluşturulan hash hashcat tool'u ile kırılabilmiştir, ancak direk openssl'in ürettiği hash hashcat tool'u ile kırılamamıştır. Dolayısıyla openssl ile şifre oluşturup kırma denemeleri yapacaksak echo ile kullanmaya özen göstermeliyiz.

Yararlanılan Kaynak

https://hashcat.net/wiki/doku.php?id=mask_attack

https://hashcat.net/wiki/doku.php?id=frequently_asked_questions#i_do_not_know_the_password_length_how_can_i_increment_the_length_of_the_password_candidates

c) Custom Charset ile Brute Force Yapmak

Diyelim ki bir NTLM hash'ini brute force ile kırmak istiyoruz.

5e35c54306bd41dcad571ec5b915a15c

Not: NTLM hash'inin Windows 7 sistemini Kali Live boot edip nasıl elde edildiğine dair olan yazı için bkz. İnternetten Edinilmiş Kıymetli Bilgiler/Elden Geçirdiğim Notlar/Kali Live'den SAM Dosyasını Çekme 2.docx

Önceki maddede Hashcat'te halihazırda var olan charset'leri kullanarak brute force yapmıştık. Halihazırda var olan charset'ler şu şekildeydi:

?l = abcdefghijklmnopqrstuvwxyz
?u = ABCDEFGHIJKLMNOPQRSTUVWXYZ
?d = 0123456789
?s = «space»!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
?a = ?l?u?d?s

Bu maddede biz kendi charset'imizi oluşturalım ve onunla brute force yapalım. Tanımlayacağımız charset alfabedeki harflerden ve sayılardan oluşsun. Hashcat'teki mevcut charset'lere bakacak olursak isteğimiz doğrultusunda bir charset'in olmadığı görülecektir. İsteğimize en yakın görünen ?a charset'i hem küçük harfleri hem büyük harfleri hem sayıları hem de özel karakterleri kapsadığından işimizi uzatacaktır. O nedenle kendi charset'imizi oluşturmak tek çıkar yoldur. Charset'imizi aşağıdaki parametrelerden herhangi biriyle oluşturabiliriz.

--custom-charset1=CS
--custom-charset2=CS
--custom-charset3=CS
--custom-charset4=CS

Bu parametreleri kısaca -1, -2, -3 ve -4 şeklinde yazabiliriz. Örneğin;

-1 CS
-2 CS
-3 CS
-4 CS

Şimdi yukarıdaki syntax'a uygun şekilde hem alfabeden oluşan hem de sayılardan oluşan charset'imizi tanımlayalım:

-1 ?l?d

Yukarıdaki ifade ile hem alfabeyi hem de sayıları charset'imiz yapmış olduk. Peki bu charset'i mask'a nasıl aktaracağız? O da şu şekildedir:

?1?1?1?1

Dikkat edilecek olursa soru işaretlerinden sonra 1 rakamı kullanılmıştır. 1 rakamı az önce bizim hem alfabe hem de sayı olarak tanımladığımız charset'ti ifade ediyor. Şimdi bunları birleştirelim:

hash.txt

5e35c54306bd41dcad571ec5b915a15c // NTLM Hash'i

Console

// NTLM Hash'leri -m 1000 ile kırılır.

> ./hashcat-cli64.bin -m 1000 -a 3 --increment --increment-min=1 --increment-max=5 hash.txt -1 ?l?d ?1?1?1?1?1

^ ^
charset mask

Output

5e35c54306bd41dcad571ec5b915a15c:**cem92**

All hashes have been recovered

Input.Mode: Mask (?1?1?1?1?1) [5]
Index.....: 0/1 (segment), 60466176 (words), 0 (bytes)
Recovered.: 1/1 hashes, 1/1 salts
Speed/sec.: - plains, 38.27M words
Progress..: 36301260/60466176 (60.04%)
Running...: 00:00:00:01
Estimated.: --:--:--:--

Started: Thu Dec 15 15:54:13 2016
Stopped: Thu Dec 15 15:54:14 2016

Görüldüğü üzere NTLM hash'inin karşılık geldiği şifrenin **cem92** olduğu görülmüştür.

Az önce var olan ?l ve ?d charset'lerini kombin edip bir charset oluşturmuştuk. Peki karakter karakter belirleyerek bir charset tanımlamasında bulunabilir miyiz dersen bu da mümkündür. Örneğin;

-1 abcdefg012345

Yukarıdaki charset tanımlaması ile diyoruz ki brute force sırasında sadece abcdefg012345 karakterlerini kullan, başka karakter kullanma. Bu şekilde daha spesifik brute force saldırılarında bulunabiliriz.

Hashcat aynı charset'leri farklı yollarla oluşturma esnekliğine de sahiptir. Örneğin aşağıdaki dört charset tanımlaması da aynı charset'i oluşturur:

-1 ?l?d

-1 abcdefghijklmnopqrstuvwxyz0123456789

-1 abcdefghijklmnopqrstuvwxyz?d

-1 ?l0123456789

d) Sözlük Saldırısı ile IncludeKarabuk Hash Kırma

includekarabuk_hash.txt

Çıktı:

$2y$10$/h7WyHABSwWrr3FvRjAtDeZ1LLmxjiZG1/WfPblXtoW5.tawue21S

hash type:

https://hashes.com/en/tools/hash_identifier

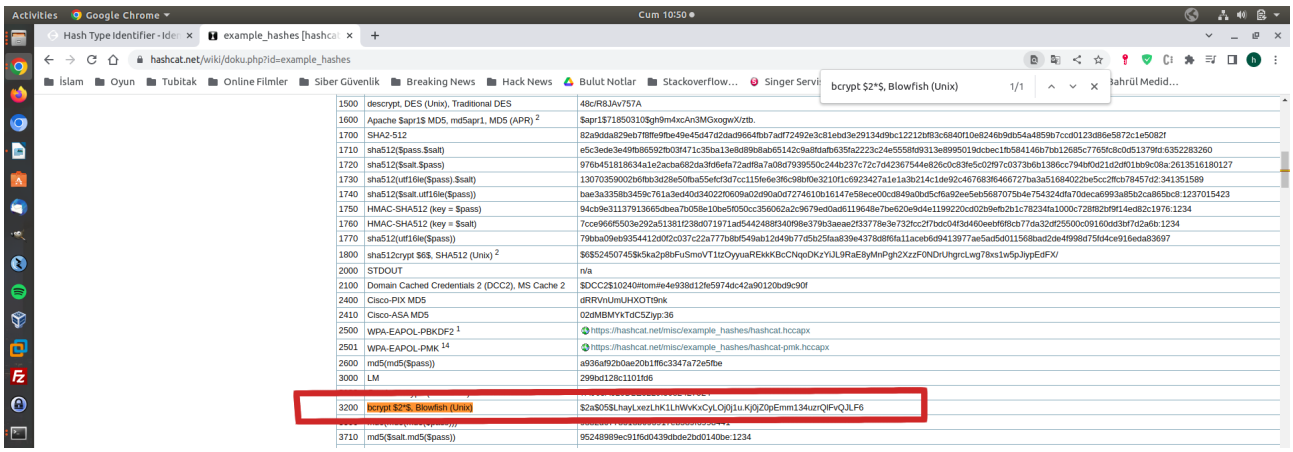
Çıktı:

bcrypt $2*$, Blowfish (Unix)

hash type (teyit):

https://hashcat.net/wiki/doku.php?id=example_hashes

Çıktı:



Hashcat ile Sözlük Saldırısı:

Ubuntu 18.04 LTS Terminal:

> ./hashcat-cli64.bin -m 3200 -a 0 includekarabuk_hash.txt rockyou.txt

Çıktı:

$2y$10$/h7WyHABSwWrr3FvRjAtDeZ1LLmxjiZG1
/WfPblXtoW5.tawue21S:**Qazwsx123.**

All hashes have been recovered

Hashcat Hatalar

a) “Token length exception - No Hashes Loaded”

(+) Birebir DVWA ve WebGoat - Ubuntu 14.04 LTS VM’deki mysql sunucudan root hash’i alınmıştır ve hashcat ile kırılmaya çalışıldığında bu hatayla karşılaşılmıştır.

Kali Linux sanal makinadan mysql istemcisi ile DVWA ve WebGoat - Ubuntu 14.04 LTS VM’deki mysql sunucusuna bağlantı kurulmuştur ve mysql root hash’i alınıp hashcat ile kırma denemesi yapılmıştır. Ayrıntılı bilgi için bkz. Tez Raporu/Literatür Taraması/İncelenmiş Makaleler/Okunmuşlar/MySQL Sızma Testi [Version 1].docx # Madde 9)). Fakat orijinal hash hashcat ile kırılamamıştır. Ardından sorunun çözümü bulunmuştur.

HATA OLAYI:

Ele geçirdiğimiz MySQL parola özeti şu şekildedir:

mysql_hash.hash

*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19

Bu parolayı kırabilmek için hashcat'ten yararlanabiliriz:

> hashcat --help | grep MySQL

Output:

200 = MySQL
300 = MySQL4.1/MySQL5

Görüldüğü üzere kullanabileceğimiz iki alternatif mod vardır. Eğer birinci mod işe yaramazsa ikincisini kullanabiliriz.

> sudo su
> hashcat -m 200 -a 0 mysql_hash.hash rockyou.txt
> hashcat -m 300 -a 0 mysql_hash.hash rockyou.txt

Output:

[…]

Hashfile 'mysql_hash.hash' on line 1

(*2470C...6DEE42FD1618BB99005ADCA2EC9D1E19): **Token length exception**

No hashes loaded.

[…]

Görüldüğü gibi hash length exception hatası vermiştir ve no hashes loaded diyerek kırma işlemi başlayamamıştır.

Kali Linux'taki hashcat yerine ubuntu 18.04 LTS ana makinadaki eski versiyon hashcat'le aynı kırma işlemi tatbik edildiğinde aynı hata gelmiştir:

> ./hashcat-cli64.bin -m 200 -a 0 mysql_hash.hash rockyou.txt

> ./hashcat-cli64.bin -m 300 -a 0 mysql_hash.hash rockyou.txt

Output:

Initializing hashcat v2.00 with 8 threads and 32mb segment-size...

Skipping line: *2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19 **(line length exception)**

No hashes loaded

AÇIKLAMA:

MySQL 5.x versiyonunda hash'ler * ile başlayan ve devamında 40 karakter alan formattadırlar. Yani * ile beraber devamında gelen 40 karakterle toplamda 41 karakterdirler (bkz. http://download.nust.na/pub6/mysql/doc/refman/5.5/en/password-hashing.html).

Hash doğrudan mysql veritabanı sunucusundaki mysql adlı veritabanının users tablosundan alınmıştır. Dolayısıyla hash'te problem olamaz. Zaten Kali'deki hashid ile hash'in türü tespit edildiğinde hash'in Mysql 5 olduğu aynı zamanda görülebilir.

Kali Linux:

> hashid "*2470C...6DEE42FD1618BB99005ADCA2EC9D1E19"

Çıktı:

Analyzing '*2470C...6DEE42FD1618BB99005ADCA2EC9D1E19'

[+] MySQL5.x

[+] MySQL4.1

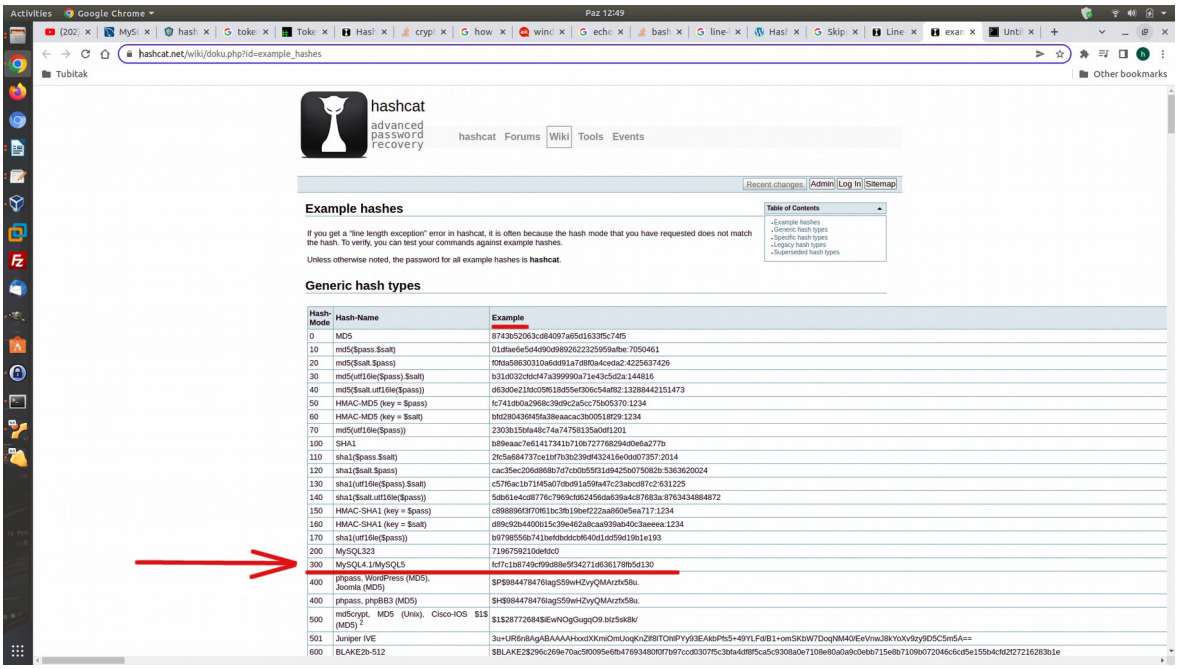
Dolayısıyla hashcat'te -m parametresine verilen -m 300 ile hash türü doğrudur. Fakat hash length hatası vermiştir ve kırılamamıştır.

ÇÖZÜM:

Hashcat'te -m ile hash türü belirtildiğinde hashcat'in ilgili hash türü için kabul ettiği hash dizgesi formatı listesi resmi sitesinde sunulmutur:

https://hashcat.net/wiki/doku.php?id=example_hashes

Bu listede örneğin -m 300 (MySQL 5.x) hash türü için 40 karakter dizgeli hash kabul ettiğini söylemiştir.



Yani orijinal hash'in en başındaki * karakterini omit ederek bu hash türünü kabul ettiğini tool beyan etmiştir.. Bu nedenle hashcat'e hash'in * karakterini kaldırarak vermek gerekir. Böylece length hatası gidecektir ve hash kırma işlemi başlayabilecektir.

Çıktı:

Initializing hashcat v2.00 with 8 threads and 32mb segment-size...

Added hashes from file ../Desktop/mysql_hash.hash: 1 (1 salts)

Activating quick-digest mode for single-hash

2470c0c06dee42fd1618bb99005adca2ec9d1e19:password

All hashes have been recovered

Input.Mode: Dict (../rockyou.txt)

Index.....: 1/5 (segment), 3627098 (words), 33550345 (bytes)

Recovered.: 1/1 hashes, 1/1 salts

Speed/sec.: - plains, 3.79M words

Progress..: 3173719/3627098 (87.50%)

Running...: --:--:--:--

Estimated.: --:--:--:--

Started: Sun Jun 5 12:52:56 2022

Stopped: Sun Jun 5 12:52:57 2022

Sonuç olarak verilen hash için hashcat length hatası verdiğinde hashcat'in kabul ettiği dizgede mi hash veriliyor kontrolü yapılabilir. Bunun için her hash türüne karşılık gelen örnek hash değeri sayfasına bakılabilir:

https://hashcat.net/wiki/doku.php?id=example_hashes

Kaynaklar:

https://hashcat.net/wiki/doku.php?id=example_hashes

https://hashcat.net/forum/thread-7202.html

Referans:

Tez Raporu/Literatür Taraması/İncelenmiş Makaleler/BGA/MySQL Sızma Testi.docx