Bezpieczeństwo komputerowe

semestr letni 2023/24

Lista nr 2

(laboratorium)

Bartosz Biegalski, 272347

1.1 Łamanie haseł złożonych jedynie z cyrf - funkcja hashująca MD5

Hashcat z łatwo i szybko radzi sobie z hasłami do 6 cyfr, nawet bez podania patternów:

```
$ hashcat -m 0 -a 3 e10adc3949ba59abbe56e057f20f883e
```

```
e10adc3949ba59abbe56e057f20f883e:123456

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: e10adc3949ba59abbe56e057f20f883e
Time.Started.....: Sat Mar 30 17:12:03 2024 (0 secs)
Time.Estimated...: Sat Mar 30 17:12:03 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?2?2?2?2?2 [6]
Guess.Charset....: -1 ?l?d?u, -2 ?l?d, -3 ?l?d*!$@_, -4 Undefined
Guess.Queue......: 6/15 (40.00%)
Speed.#1.........:   405.4 MH/s (7.40ms) @ Accel:512 Loops:512 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 3145728/3748902912 (0.08%)
Rejected.........: 0/3145728 (0.00%)
Restore.Point....: 0/1679616 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-512 Iteration:0-512
Candidate.Engine.: Device Generator
Candidates.#1....: sanier -> hs9oma
Hardware.Mon.#1..: Temp: 54c Util: 63%

Started: Sat Mar 30 17:12:01 2024
Stopped: Sat Mar 30 17:12:04 2024
```

Z hasłem “123456” i z informacją o tym, że użyty został algorytm MD5 i atak bruteforce, poradził sobie w 3 sekundy

Za to hasło 10 cyfrowe z podaną maską:

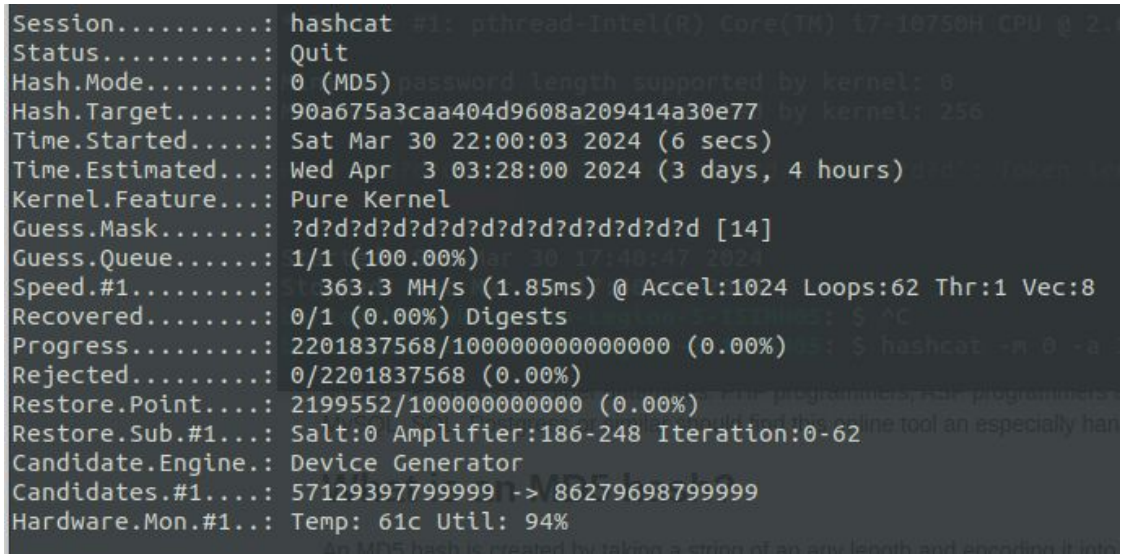
```
hashcat -m 0 -a 3 e807f1fcf82d132f9bb018ca6738a19f ?d?d?d?d?d?d?d?d?d?d
```

```
e807f1fcf82d132f9bb018ca6738a19f:1234567890

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: e807f1fcf82d132f9bb018ca6738a19f
Time.Started.....: Sat Mar 30 21:56:28 2024 (3 secs)
Time.Estimated...: Sat Mar 30 21:56:31 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?d?d?d?d?d?d?d?d?d?d [10]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   386.5 MH/s (3.83ms) @ Accel:1024 Loops:125 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 996864000/10000000000 (9.97%)
Rejected.........: 0/996864000 (0.00%)
Restore.Point....: 995328/10000000 (9.95%)
Restore.Sub.#1...: Salt:0 Amplifier:0-125 Iteration:0-125
Candidate.Engine.: Device Generator
Candidates.#1....: 1238925734 -> 9129826990
Hardware.Mon.#1..: Temp: 61c Util: 97%

Started: Sat Mar 30 21:56:27 2024
Stopped: Sat Mar 30 21:56:32 2024
```

Z hasłem “1234567890” i z informacją o tym, że użyty został algorytm MD5 i atak bruteforce potrwał 5 sekund

Dla podobnego hasła dłuższego tylko o jeden znak czas wydłużył się do 3 dni.

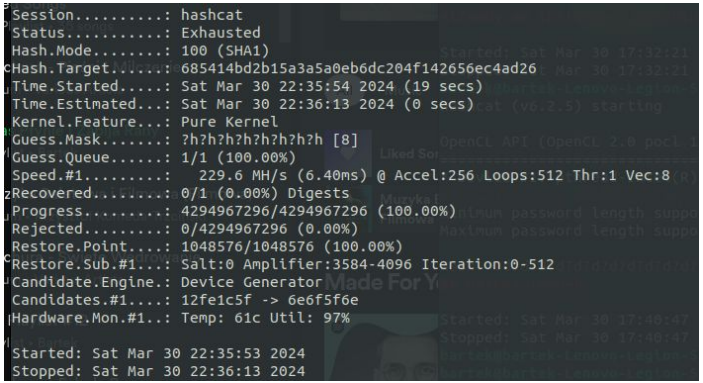
Wydaje się więc, że pomimo ograniczonej ilości znaków do 10, hasła tylko numeryczne od 14 - 15 znaków metodą bruteforce zajmują dużo czasu (14 -> 3 dni, 15 -> 30 dni). W rozsądnym czasie więc mój komputer jest w stanie poradzić sobie do haseł numerycznych długich do ok. 13 znaków.



1.2 Powtórzone doświadczenia dla kombinacji małych i wielkich cyfr, dodałem też inne funkcję phashującą SHA1. Polecenia podobne jak poprzednio

	Długość (ilość znaków)	6	10	14
Algorytm				
MD5		4 sekundy	ok. 1 godzina	ok. 9 lat
SHA1		ok. 12 minut	(Exhausted)	(Exhausted)

Okazuje się, że SHA1 generuje dłuższe hashe, które są już zbyt wymagające dla bruteforcowego podejścia hashcata. Nawet podanie maski nie pomaga. Wynik dla hasha o długości 8 (12345678)



1.3 Próba ataku na podrasowane “słabe” hasło

Spośród 100 haseł w pliku wybrałem hasło “midnight” i zmodyfikowałem je do “midnight!98”. Spodziewałem się, że hashcat dość szybko złamie hasło gdy zapewnie mu plik ze 100 hasłami oraz maskę, ale myliłem się. Po pierwsze:

- hash okazał się bardzo długi (dla MD5: b7b4e45d6a0adbe379139aa62ae147b2)
- nie mogłem zapewnić dłuższej maski niż tej w podanym poleceniu:

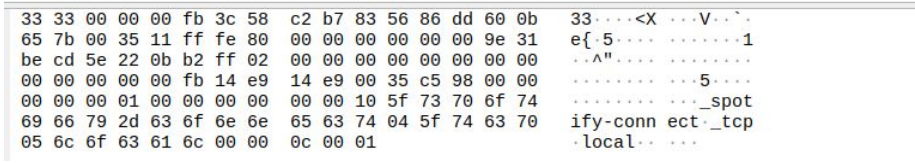
```
hashcat -a 6 -m 0 b7b4e45d6a0adbe379139aa62ae147b2 passwd.txt dictionary.txt ?h?h?h?h?h?h
```

```
Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 0 (MD5)
Hash.Target......: b7b4e45d6a0adbe379139aa62ae147b2
Time.Started.....: Sat Mar 30 22:56:18 2024 (13 mins, 18 secs)
Time.Estimated...: Sat Mar 30 23:09:36 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (dictionary.txt), Left Side
Guess.Mod........: Mask (?h?h?h?h?h?h?h) [7], Right Side
Guess.Queue.Base.: 2/2 (100.00%)
Guess.Queue.Mod..: 1/1 (100.00%)
Speed.#1.........: 34831.5 kH/s (0.27ms) @ Accel:256 Loops:128 Thr:1 Vec:8
Recovered........: 0/1 (0.00%) Digests
Progress.........: 26843545600/26843545600 (100.00%)
Rejected.........: 0/26843545600 (0.00%)
Restore.Point....: 100/100 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:268435328-268435456 Iteration:0-128
Candidate.Engine.: Device Generator
Candidates.#1....: 12345617e7f6f -> angel16e6f5f6
Hardware.Mon.#1..: Temp: 58c Util: 80%

Started: Sat Mar 30 22:52:14 2024
Stopped: Sat Mar 30 23:09:37 2024
```

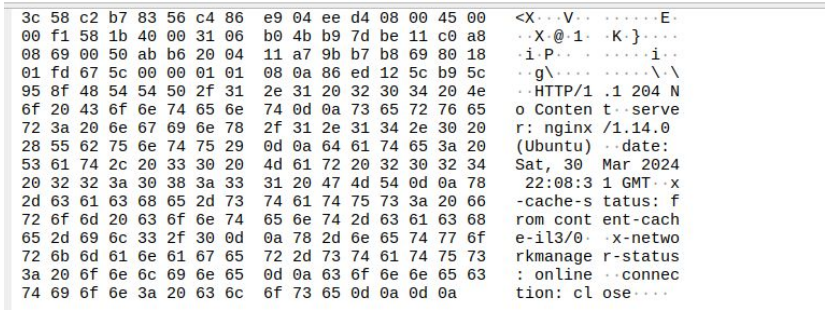
Hashcat działał ok. 17 minut, po czym stwierdził że wykracza to poza jego moce (exhausted)

2. Sniffer (Wireshark)

1. Możliwości filtrowania danych - możemy wyświetlać tylko połączenia za pomocą szczególnych protokołów lub adresów (ipv6 only, ipv4 only, UDP only), możemy też szukać takich, które mają włączone zabezpieczenie w połączeniu (https) bądź nie (http)
2. W większości są to zahashowane informacje, których nie da się zrozumieć dosłownie, gdyż różne protokoły zakrywają informacje żeby nie były jawne
3. Dlatego wireshark może być użyteczny podczas połączeń niezabezpieczonych (http)



Jawna informacja o mojej aktywności w sieci jest dostępna po włączeniu filtra “ipv6”



Jawna informacja o mojej aktywności, kiedy i z jakiego urządzenia się łączę