Słowem wstępu

Łamiąc ostatnio hashe Hashcatem czegoś mi brakowało w słownikach. Słownik wyrazów polskich jest dosyć obszerny. Słowniki imion, nazw miejscowości, nazwisk też pomagają w łamaniu haseł, (głównie z popularnymi cyframi na końcu (zwłaszcza) lub początku).Można stworzyć słownik z z kombinacją samych małych liter. Ale już przy sześciu znakach słownik zawiera 308,915,776 wyrazów, jeżeli doliczyć polskie diakrytyki, to jest już 1,838,265,625. Dosyć sporo. Oczywiście, żeby łamać takie kombinacje, nie trzeba tworzyć słowników, ale z mojego doświadczenia Hashcat dłużej to robi w trybie brute-force, niż przy zwykłym słowniku.

Zastosowałem metodę pośrednią. Coś pomiędzy brute-force, a łamaniem popularnych słów, czyli łamanie sylabowe. Wg mnie łatwiej jest zapamiętać hasło, które składa się z sylab typu „kolademore" niż „kdsfehyseadd". Obu słów nie ma w słowniku. Być może jest program do tworzenia wyrazów z sylab, ale nie mogłem znaleźć, znalazłem za to bibliotekę Pyphen, którą użyłem w programie.
Więc, żeby stworzyć słownik sylabowy i nie tylko napisałem program Sylladic. (https://github.com/kerszl/sylladic)

Program tworzy sylaby ze słownika „dict/dic.pl.txt", który to został stworzony z innego słownika (https://sjp.pl/slownik/odmiany/)

Jak tego użyć?

Ściągamy repozytorium programu i instalujemy wg instrukcji, która jest w pliku README.md

Słownik i pojedyncze sylaby zostały już wcześniej stworzone i znajdują się w katalogu „dict", ale możemy wrzucać tam nowe słowniki i sylaby.

Komendą poniżej tworzymy pojedyncze sylaby, które są zapisywane do pliku syll.słownik.txt:

./sylladic.py -d dict/słownik.txt pl

-d - tworzenie sylab ze słów
slownik.txt – słownik z którego tworzymy sylaby

pl – język z którego tworzymy sylaby

Jeżeli chcemy „podwoić” sylabę, czyli np. z la stworzyć lala, lama… używamy komendy:

./sylladic.py -m dict/nasze_sylaby.txt 4

-m – zwielokrotnienie liter, sylab, znaków
nasze_sylaby.txt – plik do zwielokrotnienia
-4 – zwielokrotnienie 4 razy

Plik z sylabami (syll.pl.txt) zawiera 17653 sylaby, najdłuższa sylaba ma 10 znaków, najkrótsza 1. Jeżeli to dwa razy zwielokrotnić, to wychodzi już 311.628.409 różnych sylab. Tego drugiego pliku niestety nie wrzuciłem do repozytorium, bo zajmuje około 3 GB, ale można go samemu stworzyć.
Do repozytorium i do katalogu dict wrzuciłem również sylaby 2,3,4 literowe oraz kończące się na na samogłoskę. Oczywiście to wszystko możesz zwielokrotnić.

Jak to wygląda w praktyce?

Ściągnąłem jakieś polskie hashe, najpierw zacząłem je łamać słownikiem rockyou.txt, potem słownikiem imion, miejscowości i na końcu pojedynczymi sylabami. Jaki efekt? Prawdę mówiąc średni. Oczywiście jakieś hasła zostały złamane, ale głównie te które, zawierały cyfry na końcu lub na początku.

Złamane hashe dzięki sylabom z pliku syll.pl.txt

Złamane hashe dzięki sylabom 3 literowym

Zacząłem też łamać hashe poprzez podwójne polskie sylaby + cyfry na końcu, ale niestety trzeba mocnych kart graficznych. Mi wyświetliło 162 dni, a to tylko część tych sylab.

```
Session..........: hashcat
Status...........: Running
Hash.Name........: md5crypt, MD5 (Unix), Cisco-IOS $1$ (MD5)
Hash.Target......:
Time.Started.....: Wed May 12 14:46:04 2021 (1 min, 24 secs)
Time.Estimated...: Thu Oct 21 22:56:55 2021 (162 days, 8 hours)
Guess.Base.......: File (sylaby/syll.pl.x2it.001)
Guess.Mod........: Rules (rules/popular_digits.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  8417.3 kH/s (281.39ms) @ Accel:64 Loops:1000 Thr:1024 Vec:1
Recovered........: 62032/149796 (41.41%) Digests, 62032/149795 (41.41%) Salts
Remaining........: 87764 (58.59%) Digests, 87763 (58.59%) Salts
Recovered/Time...: CUR:0,N/A,N/A AVG:0,0,0 (Min,Hour,Day)
Progress.........: 2300716970/201512394831600 (0.00%)
Rejected.........: 816981930/2300716970 (35.51%)
Restore.Point....: 0/49824240 (0.00%)
Restore.Sub.#1...: Salt:21 Amplifier:26-27 Iteration:0-1000
Candidates.#1....: 11aa -> 11afootrzeźw
Hardware.Mon.#1..: Temp: 76c Util:100% Core:1125MHz Mem:5000MHz Bus:8
```

Podsumowując:

Pojedyncze sylaby w pliku syll.pl.txt mają od 1-10 znaków, podwójne już od 2-20 znaków.

Te słowniki mniej zajmują niż wszystkie kombinacje z brute-force, ale nie zawierają słów, które nie są sylabami. Z tradycyjnymi słownikami jest różnie, czasami zajmują więcej, czasami mniej.

Łamanie sylabowe może czasem Ci pomóc złamać zapomniane hasło, jeżeli użyłeś już różnych słowników i to nie przyniosło to efektu.

Program jeszcze wymaga dopracowania, zwłaszcza przetestowania kombinacji odpowiednich sylab. Na końcu zamieszczam tabelkę z iteracji znaków, jest dostępna w programie poprzez komendę:

./sylladic.py -g

char	1	2	3	4	5	6	7	8
d	10	100	1,000	10,000	100,000	1,000,000	10,000,000	100,000,000
l	26	676	17,576	456,976	11,881,376	308,915,776	8,031,810,176	208,827,064,576
l+u	52	2,704	140,608	7,311,616	380,204,032	19,770,609,664	1,028,071,702,528	53,459,728,531,456
l+pl	35	1,225	42,875	1,500,625	52,521,875	1,838,265,625	64,339,296,875	2,251,875,390,625
l+u+pl	70	4,900	343,000	24,010,000	1,680,700,000	117,649,000,000	8,235,430,000,000	576,480,100,000,000
prin...	100	10,000	1,000,000	100,000,000	10,000,000,000	1,000,000,000,000	100,000,000,000,000	10,000,000,000,000,000

1..8 - iterations, d - digits, l - lower chars, u - upper chars,
pl - polish chars, prin.. - all printable chars

Jeżeli masz jakieś sugestie pisz na: kerszi@protonmail.com