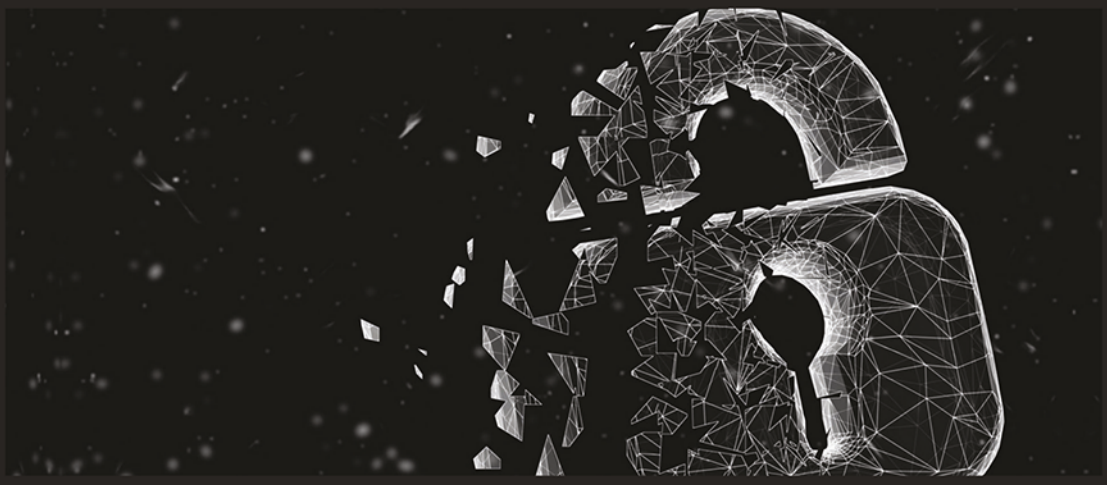
Helion

packt



# Etyczne łamanie haseł

## John the Ripper, hashcat i inne zaawansowane techniki

JAMES LEYTE-VIDAL

Tytuł oryginału: Ethical Password Cracking: Decode passwords using John the Ripper, hashcat, and advanced methods for password breaking

Tłumaczenie: Małgorzata Dąbkowska-Kowalik i Witold Sikorski

ISBN: 978-83-289-2222-8



Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
*helion.pl/user/opinie/etlaha*
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

# Spis treści

# Przechowywanie haseł — matematyka, prawdopodobieństwo i złożoność

Rozdział 1

Od czasu narzędzi takich jak Cain and Abel po nowoczesne narzędzia takie jak hashcat łamanie haseł stało się nieodłącznym elementem testowania zabezpieczania informacji. Podczas gdy narzędzia i techniki zmieniały się z upływem lat, zasady leżące u podstaw łamania haseł pozostały w znacznej mierze bez zmian.

Łamanie haseł może obejmować wiele praktycznych przypadków, od odzyskiwania dostępu do systemu po odejściu użytkownika z firmy, po testowanie penetracyjne i przypadki wykorzystania przez zespoły „czerwone", gdzie wykorzystujemy łamanie hasła, aby udowodnić bezpieczeństwo mechanizmów kontroli dostępu lub stwierdzić jego nieskuteczność.

W tym rozdziale omawiamy następujące główne zagadnienia:

- Czym jest łamanie haseł?
- Jak hasła są używane i przechowywane?
- Dlaczego niektóre hasła są łatwiejsze do złamania od innych?

## Czym jest łamanie haseł?

Łamanie haseł to proces odtwarzania sekretu na podstawie zakodowanego (zwykle zaszyfrowanego lub skróconego) tekstu. Termin ten jest bardzo szeroki i obejmuje wiele metod przechowywania i mieszania haseł. Dlatego nie wszystkie działania związane z łamaniem haseł są równorzędne — niektóre hasła, a także metody przechowywania haseł są łatwiejsze do złamania niż inne. Będziemy to gruntownie omawiać w całej tej książce.

W łamaniu haseł można wyodrębnić dwa różne podejścia do próby odzyskania sekretu:

- oparte na słowniku,
- kombinowane,
- siłowe,
- hybrydowe,
- oparte na częściowej wiedzy, zwane też atakami z maską.

Omówmy po kolei wszystkie te podejścia.

## Ataki słownikowe

**Ataki słownikowe**, jak można się domyślić na podstawie nazwy, wykorzystują listę słów lub fraz jako kandydatów na hasło — potencjalne hasło — które będziemy sprawdzać, aby zobaczyć, czy będą one poprawnymi hasłami. Taka lista jest nieformalnie określana jako **słownik**, choć może wcale nie zawierać słów ze słownika. Lista słów może w ogóle nie przypominać słownika. Termin ten jest utrzymywany od czasów, gdy wiele haseł opierało się na słowach ze słowników, zanim powszechne stały się wymagania dotyczące złożoności haseł (takie jak dodawanie do hasła wielkich liter, liczb i znaków specjalnych).

Gdy mówimy o wymaganiach dotyczących złożoności, wydaje się, że tradycyjne słowa ze słownika nie byłyby skuteczne jako kandydaci na hasło podczas łamania haseł, gdyż wymogi złożoności stają się coraz bardziej powszechne. Zajmiemy się tym w kolejnych podrozdziałach.

Budowa listy słów podczas ataku słownikowego może być prosta lub czasochłonna. Jednak w wielu przypadkach poświęcenie czasu na początku przy tworzeniu dobrej listy dopasowanej do celu może przynieść korzyści podczas łamania. Kompromis polega tu na tym, że przygotowana lista słów może nie nadawać się do ponownego zastosowania w innych przypadkach łamania haseł. Omawiamy to w rozdziale 2. „Po co łamać, jeśli wystarczy OSINT?", wykorzystując **biały wywiad** (**OSINT**) do pomocy przy tworzeniu listy słów.

Dobra i dość duża lista słów, którą możemy wykorzystać na początek, to **RockYou**. Nazwa pochodzi od nazwy firmy RockYou, do której włamano się w 2009 roku, ujawniając ponad 32 miliony uwierzytelnień użytkowników. Lista ta dostępna jest w kilku miejscach w internecie, ale popularną lokalizacją, z której można ją pobrać, jest *https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt*. Lista ta zawiera 14 milionów unikatowych kandydatów na hasła, a jednocześnie wiele popularnych dystrybucji testów penetracyjnych, takich jak Kali Linux (dostępny pod adresem *https://www.kali.org/get-kali/#kali-platforms*) oraz Slingshot Linux (dostępny pod adresem *https://www.sans.org/tools/slingshot/*).

## Ataki kombinowane

**Ataki kombinowane** pobierają dwie listy słów jako dane i scalają (łączą ze sobą) po jednym kandydacie na hasło z każdej listy, tworząc kandydata na hasło do celów testowania. Na przykład jedna lista może zawierać słowa `słowo1` i `słowo2`, a druga `słowo3` i `słowo4`. W tym scenariuszu atak kombinowany wykorzysta obie listy, tworząc potencjalnych kandydatów na hasło w postaci `słowo1słowo3`, `słowo1słowo4`, `słowo2słowo3` oraz `słowo2słowo4`.

Obecne zalecenia amerykańskiej organizacji **NIST** (National Institute of Standards and Technology) mówią o tym, że dla zapewnienia lepszej odporności na złamanie długość hasła jest ważniejsza od jego złożoności. Pomaga to zachęcić użytkowników, aby tworzyli hasła łatwe do zapamiętania, ale trudne do złamania i zgodne z bieżącymi zaleceniami NIST. Można tego dokonać, łącząc ze sobą kilka słów ze słownika i dodając mnemonikę pomagającą w zapamiętaniu hasła. Jest to tylko jedno z podejść do tego tematu, ale przykład ten pokazuje — w powiązaniu z obecnymi zaleceniami NIST — że kombinacyjne

podejście do łamania haseł może być bardziej skuteczne, gdyż wielu użytkowników, kierując się tymi zaleceniami, przechodzi na hasła złożone z fraz.

Jednocześnie niektóre standardy mogą spowolnić przechodzenie na dłuższe, mniej skomplikowane frazy. Na przykład standard **PCI-DSS** (ang. *Payment Card Industry Data Security Standard*), który jest wymagany przy przetwarzaniu danych kart kredytowych w placówkach handlowych, wymaga hasła złożonego z 12 znaków, a także liter i cyfr przy hasłach powiązanych z kontami, które mają dostęp do danych posiadacza karty.

## Ataki siłowe

**Ataki siłowe** robią dokładnie to, na co wskazuje ich nazwa — każda pozycja kandydatów na hasło jest wypełniana każdym możliwym znakiem w tej pozycji. Jeśli na przykład hasło może mieć jedynie długość ośmiu znaków, podejście siłowe będzie wypróbowywać jako kandydata `aaaaaaaa`, potem `aaaaaaab` i tak dalej, aż do *wyczerpania* wszystkich możliwości. Problem związany z tym podejściem polega na tym, że gdy hasło osiąga rozsądną długość, czas potrzebny na wykonanie tego rodzaju ataku staje się niemożliwy do realizacji. Ponadto liczba zbiorów znaków dostępnych do wykorzystania (małe litery, wielkie litery, cyfry i symbole) także znacznie zwiększa liczbę prób wymaganych przy tym ataku.

Dobrą wiadomością z punktu widzenia łamania haseł jest możliwość złamania tym sposobem każdego hasła. Jednak czas potrzebny na to przy dzisiejszej mocy obliczeniowej komputerów sprawia, że jest to ułudą przy większych hasłach lub bardziej skomplikowanych algorytmach (wymagających więcej czasu przy każdej próbie).

## Ataki hybrydowe

**Ataki hybrydowe** łączą ze sobą pewne cechy ataków kombinowanych i ataków siłowych. Atak hybrydowy wykorzystuje jako podstawę listę słów, a następnie modyfikuje słowa z listy, dodając do nich jeden lub więcej znaków i próbując ataku siłowego w przestrzeni powiązanych z tym znaków. Dla przykładu powiedzmy, że mam następujące słowo z listy słów:

```
banana
```

Wiem jednak, że reguły haseł w moim celu wymagają, aby w każdym haśle była cyfra. Mogę spróbować ataku hybrydowego, który wykorzystuje słowo z mojej listy i dodaje po nim cyfrę. Teraz moi kandydaci na hasło wyglądają następująco:

```
banana1
banana2
banana3
banana4
```

Pozwala to na bardziej efektywne testowanie środowisk, których użytkownicy często dodają na końcu lub na początku swojego hasła jakieś podstawowe słowo ze słownika.

## Ataki oparte na częściowej wiedzy, zwane też atakami z maską

**Ataki z maską** wykorzystują pomysł, że jeśli częściowo rozumiemy format użyty do budowy hasła, to tworzymy atak o podejściu podobnym do siłowego, który jest przyspieszony dzięki założeniom, jakie przyjmujemy na temat formatu hasła.

Pomocny będzie tu przykład. Przyjmijmy, że testujemy hasła w firmie, która wymaga w hasłach jednej wielkiej litery, jednej małej litery i cyfry. Jest dość powszechne w wielu firmach wymaganie związane ze złożonością haseł i wielu użytkowników spełnia ten warunek, stawiając wielką literę na początku słowa ze słownika i dodając na końcu jedną lub dwie cyfry.

Okazuje się, że takie wymagania co do hasła w powiązaniu ze zmianą hasła co 90 dni mogą prowadzić do koszmarnego hasła *pora-rok*, kiedy to użytkownicy ustawiają hasło na porę roku (wiosna, lato itd.), dodając dwu- lub czterocyfrowe oznaczenie roku (Wiosna22/Wiosna2022/Lato22/Lato2022 itd.).

Może to prowadzić do utworzenia maski na hasło, która zakłada, że użytkownik wybierze hasło z wielką literą na początku, potem kilkoma małymi literami i dwiema lub czterema cyframi na końcu (od 0 do 9). Maska ta będzie dobrą próbą ataku siłowego na każde hasło o podobnej długości spełniające te kryteria. Nie doprowadzi to do uzyskania każdego hasła z listy, ale podejście to wykazuje historycznie duży procent złamanych haseł, gdyż zasady te są często przyjmowane przez użytkowników podczas tworzenia haseł.

**Ważna uwaga**

Lepsze sposoby tworzenia i ograniczania haseł pokażemy w rozdziale 11.

## Jak hasła są używane i przechowywane?

Sposób przechowywania haseł może wydawać się sprawą prostą, ale może to mieć ogromny wpływ na możliwości ich odzyskiwania za pomocą łamania haseł i na czas trwania takiej operacji.

**Nie zawsze trzeba łamać hasła!**

Większość haseł jest przechowywana w systemach uwierzytelnień za pośrednictwem jakiegoś procesu, który utrudnia ich odzyskanie. Jednak nie jest niczym niezwykłym natknięcie się na systemy, które nie chronią odpowiednio danych uwierzytelniających użytkowników. Wcześniej była w tym rozdziale mowa o złamaniu systemu RockYou. W tym przypadku firma ta przechowywała hasła w postaci tekstowej (bez skrótów ani szyfrowania), co sprawiło, że uzyskanie haseł było trywialne. Oznaczało to, że gdy hasła użytkowników stały się dostępne publicznie, były w pełni narażone — nie było wymagane żadne łamanie haseł ani inne skomplikowane działania. Hasła były po prostu do wzięcia.

Pomówmy teraz o dwóch rodzajach przechowywania haseł, z którymi zwykle mamy do czynienia: **skrótach** oraz **szyfrowaniu**.

## Skróty

Idea tworzenia skrótów jest taka, że nie mogą zostać one przez nikogo odzyskane. Istnieje kilka zalet tego podejścia:

- Firmie, która przechowuje hasła, daje to wysoki poziom staranności i *może* zapewnić legalnie pewną ochronę.
- Haseł nie można przekształcić ze skrótu do postaci tekstowej (oryginalnego hasła), co oznacza, że złośliwe osoby z wewnątrz firmy z dostępem do magazynu haseł nie mogą ich uzyskać.
- Istnienie standardowych funkcji skrótu w wielu strukturach aplikacji sprawia, że jest to łatwe do implementacji.

W swojej istocie funkcja skrótu przyjmuje tekst jawny (hasło) i przekształca go na ciąg o stałej długości zawierający nieczytelne dane. Wartości tej nie można ponownie przekształcić w tekst, co stanowi podstawową różnicę pomiędzy wykonywaniem skrótów a szyfrowaniem. Proces ten zawsze da taką samą wartość dla tych samych danych wejściowych, co oznacza, że proces jest **deterministyczny**. Niektóre metody tworzenia skrótu dodają **sól**, co wprowadza do generowania wartości skrótu dodatkową entropię (losowość). Sól jest inna dla każdego hasła, co uniemożliwia skuteczność **ataku z użyciem wstępnego wyznaczania** — rodzaju ataku, który generuje wszystkie możliwe skróty przed wykonaniem łamania (być może słyszeliście o tęczowych tablicach — technice stosowanej w tego rodzaju atakach). Różne algorytmu skrótów zostaną omówione dalej, gdy zagłębimy się w różne sposoby odzyskiwania haseł.

W przypadku wykonania skrótu hasła są weryfikowane podczas procesu uwierzytelniania przez pobranie hasła od użytkownika, wykonanie skrótu i porównanie go z przechowywanym skrótem. Jeśli są one identyczne, hasło jest poprawne, a jeśli jest inaczej, to wprowadzone hasło jest błędne. Tak więc podczas tego procesu wykonanie skrótu chroni jawne hasło, sprawiając, że po wykonaniu skrótu jawne hasło nigdy nie jest obsługiwane przez system.

## Szyfrowanie

Szyfrowanie różni się od tworzenia skrótów (mieszania), gdyż szyfrogram (produkt algorytmu szyfrowania) może zostać odwrócony i dać oryginalny tekst jawny (hasło). W tym celu musi zostać wygenerowany jeden lub więcej kluczy szyfrujących używanych do szyfrowania i odszyfrowywania.

Szyfrowanie ma pewne wady w kwestii przechowywania haseł. Najistotniejszy jest fakt, że szyfrogram jest odwracalny, co oznacza, że złośliwy wewnętrzny lub zewnętrzy napastnik może odzyskać jawne hasła, jeśli uda mu się uzyskać szyfrogram i klucze szyfrujące. Ponadto klucze, które są używane podczas szyfrowania i odszyfrowywania, muszą być możliwe do uzyskania, co jeszcze zwiększa potencjalne ryzyko niewłaściwego zarządzania kluczami lub ich ujawnienia.

**Łatwe sprawdzenie, czy użyto szyfrowania zamiast skrótu (lub gorzej)**

Czy zdarzyło Ci się zapomnieć hasła i wykorzystać w aplikacji link do funkcji *Zapomniano hasła*? Zapewne zdarzyła się taka sytuacja. Każdy, kto używał takiej funkcji i otrzymał swoje hasło przesłane za pomocą e-maila lub innej metody używającej jawnego tekstu (zamiast otrzymać prośbę o ustanowienie nowego hasła), powinien wiedzieć, że hasło przechowywane jest w systemie w postaci zaszyfrowanej. Jeśli wykorzystywany jest skrót hasła, nikt nie może odzyskać Twojego jawnego hasła. Jest też jeszcze inna możliwość — system przechowuje Twoje hasło jawnym tekstem, podobnie jak to robiła firma RockYou. Wiemy już, jak zły jest taki pomysł, lecz niestety czasami mamy do czynienia z takim postępowaniem.

W przypadku uwierzytelniania za pomocą zaszyfrowanych haseł szyfrogram może być porównywany (podobnie jak w przypadku uwierzytelniania skrótem) lub hasło może zostać odszyfrowane i porównane w celu jego potwierdzenia z hasłem dostarczonym przez użytkownika.

Wspomniano tu o szyfrowaniu, aby przedstawić całą paletę możliwości, ale nie jest to optymalna metoda przechowywania haseł i nie jest zalecana przez standardy NIST 800-53.

# Dlaczego niektóre hasła są łatwiejsze do złamania od innych?

Istnieje kilka powodów tej sytuacji, ale sprowadza się to do tego, ile czasu zajmuje systemowi poprawne odgadnięcie hasła. Jeśli potrafimy utworzyć hasła wydłużające czas na to potrzebny, tworzymy hasła trudniejsze do złamania. Jeśli tworzymy hasła skracające czas potrzebny na pomyślne odgadnięcie hasła, tworzymy hasło, które jest łatwiejsze do złamania.

Jakie czynniki sprawiają, że hasło jest łatwiejsze (lub trudniejsze) do złamania? Oto niektóre najważniejsze z nich:

- długość hasła,
- złożoność hasła,
- czas potrzebny na wykonanie skrótu lub zaszyfrowanie hasła.

Omówmy po kolei te czynniki.

## Długość hasła

Długość hasła jest często traktowana przez użytkownika końcowego w kategoriach wymaganego minimum. Innymi słowy, jeśli system wymaga, aby minimalna długość hasła wynosiła osiem znaków, to użytkownik wybierze hasło ośmioznakowe.

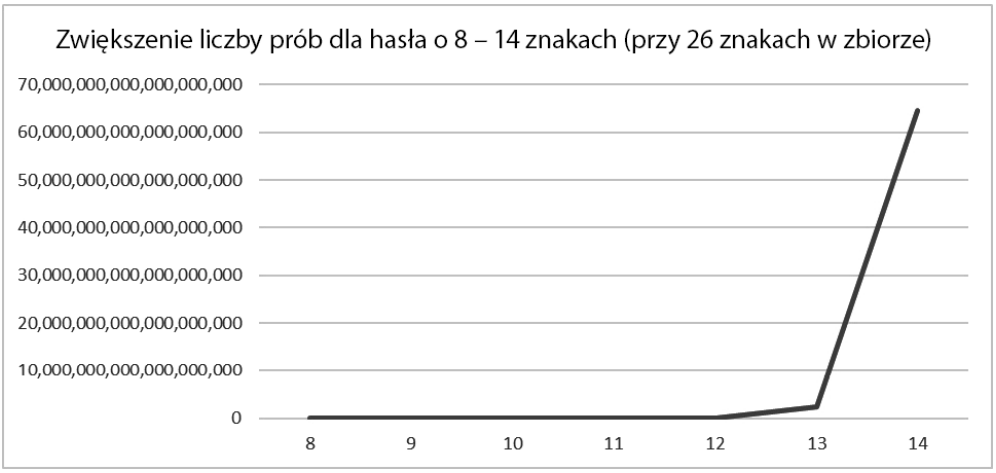
W czasie gdy pisana była ta książka, NIST utrzymywał rekomendacje dla haseł jako minimum osiem znaków. Jest to zapisane w specjalnej publikacji SP 800-53B NIST, która jest od czasu do czasu aktualizowana. Jednak NIST zauważa także, że systemy powinny akceptować hasła użytkownika o długości co najmniej 64 znaków.

Pomyślmy przez chwilę o ośmioznakowym haśle. Ile prób musimy podjąć, aby odnaleźć czyjeś hasło złożone z ośmiu znaków? Odpowiedź, podobnie jak w wielu przypadkach zabezpieczeń informacji, brzmi: *to zależy*. Zacznijmy od prostego zestawu znaków złożonego z 26 małych liter alfabetu łacińskiego. Liczbę prób potrzebnych, aby odgadnąć to hasło, można obliczyć jako x do potęgi y, czyli $x^y$, gdzie x to możliwy znak na każdej z pozycji hasła, a y to całkowita liczba znaków w haśle. Dla hasła wykorzystującego 26 małych liter alfabetu, które ma długość ośmiu znaków, będzie to $26^8$ prób, czyli 208 827 064 576 prób. Zauważmy, że jest to *maksymalna* liczba prób — reprezentuje sytuację, w której ktoś wypróbowuje wszystkie możliwe hasła i trafia dopiero w ostatniej próbie. To dużo możliwych prób! Czy oznacza to jednak, że hasło jest bezpieczne? Znowu — to zależy. Jak szybko możemy próbować i oceniać, czy coś jest lub nie jest hasłem? Nawet milisekundy mniej lub więcej na każdą próbę mogą mieć ogromny wpływ na całkowity czas przejścia przez wszystkie możliwości.

Co będzie, jeśli wybierzemy długość hasła większą niż minimum zalecane przez NIST? Co będzie w przypadku dziewięciu znaków z alfabetu łacińskiego? To będzie $26^9$ prób, czyli 5 529 503 678 978. Jest to, jak można się było spodziewać, 26 razy więcej prób niż było potrzebnych dla hasła złożonego z ośmiu znaków.

Gdy dojdziemy do hasła złożonego z 12 przy tym samym zbiorze 26 znaków, patrzymy na $26^{12}$ prób, czyli liczbę 95 428 956 661 682 176 (określaną również jako 95 trylionów prób). Jest to 456 976 razy więcej niż liczba prób potrzebna w przypadku hasła o ośmiu znakach!

Wizualizując to na wykresie (patrz rysunek 1.1), możemy zobaczyć wykładniczy wzrost liczby prób potrzebnych przy zwiększeniu długości hasła o każdy kolejny znak.



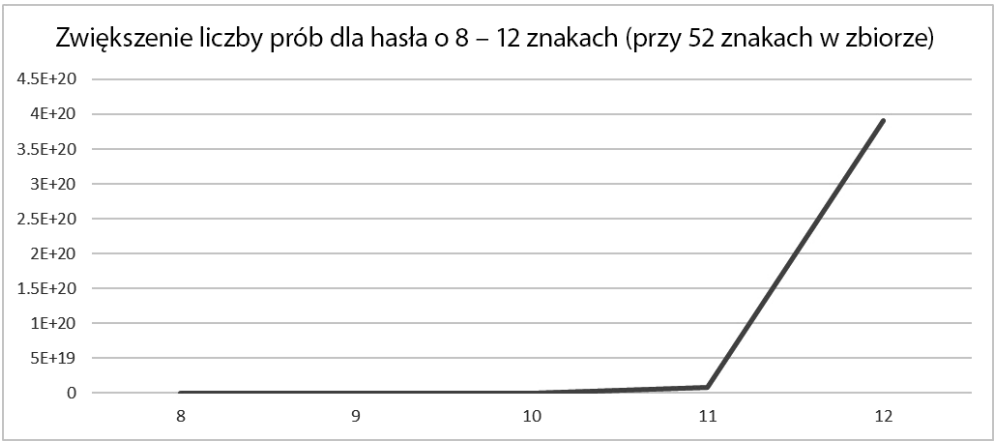

**Rysunek 1.1. Liczba prób dla hasła liczącego 8 – 14 znaków (przy 26 możliwych znakach)**

Jest to dobra wiadomość dla wszystkich tworzących bezpieczne systemy i oznacza, że przy długości hasła liczy się każdy dodatkowy znak. Im dłuższe jest hasło, tym dłużej trwa jego łamanie i jest ono lepiej zabezpieczone (odporne na złamanie).

# Złożoność hasła

Podobnie jak długość hasła, jego złożoność ma sprawić, że będzie ono bardziej odporne na złamanie. Jednak złożoność stanowi inne podejście — zwiększamy liczbę możliwych znaków, które mogą być używane na każdej pozycji w haśle. Zobaczmy, jak to działa w praktyce, wracając do matematyki z poprzedniego punktu.

Jeśli do naszych małych liter dodamy wielkie litery łacińskiego alfabetu, otrzymamy 52 możliwe znaki. Tak więc teraz nasze ośmioznakowe hasło będzie wymagało $52^8$ prób, czyli 53 459 728 531 456. Tak więc dodatkowych 26 znaków znacząco zwiększa liczbę prób. Ponadto, ponieważ działanie to jest wykładnicze, zwiększenie liczby prób na znak można podobnie jak hasło oparte na 26 znakach przedstawić w funkcji zwiększania długości (patrz rysunek 1.2).




**Rysunek 1.2. Liczba prób dla hasła liczącego 8 – 12 znaków (przy 52 możliwych znakach)**

Jak widać na rysunku 1.1, zwiększanie długości hasła zwiększa ogólną liczbę prób potrzebną do identyfikacji hasła. Podobnie na rysunku 1.2 możemy zobaczyć, że zwiększenie złożoności hasła zwiększa liczbę potrzebnych prób, a zwiększenie jednocześnie długości i złożoności zwiększa liczbę potrzebnych prób jeszcze szybciej! Co jest więc lepsze? W tym celu musimy spojrzeć na matematykę i wesprzeć nasze rozważania psychologią.

Ośmioznakowe hasło zawierające wielkie i małe litery wymaga maksymalnie 53 459 728 531 456 prób (53 biliony prób). Dziesięcioznakowe hasło zawierające tylko małe litery będzie wymagało niemal trzy razy więcej prób — 141 bilionów. Przejdźmy teraz do psychologii. Co jest łatwiejsze do zapamiętania przez człowieka — ciąg znaków złożony z samych małych liter czy ciąg małych i wielkich liter? Zwykle zdarzy się jedna z dwóch rzeczy:

- Użytkownik utworzy hasło, które jest łatwe do zapamiętania, ustawiając na początku wielką literę, a resztę hasła pisząc małymi. Jest to trywialne do rozwiązania przy łamaniu i sprawia, że dodanie kolejnego znaku nie ma sensu. Jeśli pierwsza litera hasła jest wielka, to mamy 26 różnych możliwości, co oznacza, że liczba prób jest taka sama jak przy użyciu małych liter.

Jeśli pozostałe litery będą małe, to także mammy do wyboru 26 znaków. W tym scenariuszu przy ośmioznakowym haśle mamy 268 możliwości zamiast 528 — *tyle samo, co gdyby hasło składało się z samych małych liter!*

- Druga możliwość jest taka, że użytkownik tworzy trudne do zapamiętania hasło i zapisuje je na kartce lub w menedżerze haseł. Wprawdzie korzystanie z menedżera haseł jest ogólnie pożądanym zachowaniem, ale zapisanie hasła w miejscu, gdzie może zostać znalezione, nie jest dobre.

Do czego więc doszliśmy? Ludzki umysł łatwiej zapamięta hasło pisane małymi literami niż ciąg liter małych i wielkich, liczb i symboli. Możemy zwiększać długość hasła pisanego małymi literami i także uzyskać hasło odporne na złamanie. To stanowi aktualną rekomendację NIST — ostatnia wersja SP 800-53B sugeruje, aby tworzyć hasła, które *nie* wymagają stosowania reguł złożoności (patrz podpunkt 5.1.1.2).

## Pora na wykonanie skrótu lub zaszyfrowanie hasła

Trzeci ważny czynnik przy tworzeniu haseł, które są odporne ma złamanie, nie leży w wyborze samego hasła, lecz dotyczy działań obliczeniowych przy tworzeniu skrótu i czasu, który jest na to potrzebny. Pomyśl o liczbie prób potrzebnych przy różnych rodzajach hasła, o czym była mowa wyżej. Każda z tych prób wymaga niezerowego czasu na wykonanie. Musimy obliczyć skrót kandydata na hasło, a następnie porównać go za znanym skrótem, aby zobaczyć, czy są one takie same (co oznacza, że nasz kandydat na hasło jest hasłem).

Jeśli działanie to trwa pełną sekundę zamiast pół sekundy, to całkowity czas złamania hasła się podwaja. W rzeczywistości próby zachodzą znacznie szybciej, ale dla celów ilustracyjnych można zobaczyć, jak dużą różnicę to robi względem liczby prób, z którymi mamy do czynienia w tych scenariuszach.

Algorytmy tworzenia skrótu są zaprojektowane tak, aby były szybkie. Funkcja skrótu to popularne obliczenia matematyczne służące do porównywania i chcemy, aby były szybkie. Jednak tworzenie skrótów (mieszanie) haseł powinno być szczególnie wolne — chcemy, aby było tak wolne, jak to jest możliwe. Im wolniejsze jest tworzenie skrótu, tym implementacja będzie bardziej odporna na złamanie, gdyż każda próba będzie potencjalnie droższa obliczeniowo. Algorytmy mieszania haseł, jak np. PBKDF2, wykorzystują popularny algorytm SHA-512, ale wykonują wiele rund algorytmu, aby zwiększyć czas tworzenia skrótu hasła.

Podczas gdy zwiększony czas na wykonanie skrótu będzie skutkował wolniejszymi działaniami podczas łamania, osoba łamiąca hasło może próbować to zniwelować, zwiększając liczbę skrótów wykonywanych na sekundę albo zwiększając stosowaną moc obliczeniową, albo rozkładając obciążenie związane z łamaniem na wiele maszyn prowadzących obliczenia. W części II „Zbieranie i łamanie" przyjrzymy się ogólnie szybkości różnych działań związanych z łamaniem zależnie od rodzaju skrótu, który jest łamany.

## Trochę na temat „etycznego” łamania haseł

Niezależnie od podejścia celem tej książki jest pomoc w użytkowaniu narzędzi i technik potrzebnych do odzyskiwania haseł, bez względu na to, czy pracujesz w zespole testującym możliwość penetracji systemu, czy odzyskujesz hasło użytkownika, który niestety odszedł z tego świata, czy też masz inny powód.

Ważne zastrzeżenie jest takie, że książka skupia się na etycznym łamaniu haseł. Celem jej nie jest pomoc w obchodzeniu prawa lub wykonywaniu nielegalnych działań. Jej celem jest danie Ci narzędzia potrzebnego do tego, aby z powodzeniem wskazywać błędy w testach penetracyjnych lub innych zatwierdzonych działaniach koniecznych przy prowadzeniu biznesu.

Zanim zaczniesz stosować te techniki w swojej firmie, na wszelki wypadek skonsultuj się z firmowym radcą prawnym lub prawnikiem.

## Podsumowanie

W tym rozdziale przedstawiliśmy pojęcie łamania haseł, różne typy ataków w celu złamania hasła, sposoby przechowywania i używania haseł oraz przeprowadziliśmy analizę tego, co sprawia, że hasło jest silniejsze. Mając tę wiedzę, masz podstawy do rozpoczęcia pracy nad różnymi rodzajami łamania haseł.

Czy jednak nie byłoby prościej, gdybyś nigdy nie musiał łamać hasła? W niektórych przypadkach można tego uniknąć, gdyż mamy dostępne informacje o wcześniejszych wyciekach danych i złych praktykach związanych z hasłami, jak np. ich ponowne wykorzystanie. W następnym rozdziale przeanalizujemy sposoby wykorzystania OSINT, aby uzyskać informacje z wcześniejszych wycieków lub zbudować dostosowaną do potrzeb listę słów dla określonych celów.

# Myślisz, że Twoje hasło jest bezpieczne?

Umiejętność łamania haseł przydaje się nie tylko przestępcom, ale również specjalistom do spraw bezpieczeństwa. Jest cenna w wielu sytuacjach, na przykład w razie konieczności odzyskania dostępu do systemu po odejściu użytkownika z firmy, w testowaniu penetracyjnym i w obszarze działalności zespołów czerwonych. W takich przypadkach próba złamania hasła ma udowodnić wysoki poziom mechanizmu kontroli dostępu albo jego nieskuteczność.

Dzięki tej praktycznej książce dogłębnie zrozumiesz zagadnienia związane z ochroną haseł i odzyskiwaniem chronionych nimi danych. Rozpoczniesz od zapoznania się z zasadami przechowywania danych uwierzytelniających i matematycznymi podstawami technik łamania haseł. Następnie nauczysz się posługiwać różnymi narzędziami ułatwiającymi odzyskiwanie haseł, by potem zająć się typowymi przypadkami ich łamania, odzyskiwania skrótów i pokonywania zabezpieczeń. Przyjrzysz się działaniu metod siłowych i słownikowych, dowiesz się także, jak stosować je przy różnych sposobach przechowywania danych uwierzytelniających. Poszczególne zagadnienia zostały zilustrowane licznymi rzeczywistymi przykładami. Pod koniec lektury przekonasz się, że potrafisz z łatwością łamać najpopularniejsze typy danych uwierzytelniających.

## W książce między innymi:

- koncepcje łamania haseł i popularnych typów skrótów
- identyfikowanie, wyodrębnianie i łamanie skrótów haseł systemów Windows i macOS
- architektura WPA/WPA2
- popularne menedżery haseł, takie jak KeePass, LastPass i 1Password
- formatowanie skrótów dla portfeli bitcoin, litecoin, Ethereum i ich łamanie

**James Leyte-Vidal** od 20 lat działa w branży bezpieczeństwa komputerowego. Doradzał wielu firmom z listy Fortune 100 w zakresie architektury bezpieczeństwa, testów penetracyjnych czy polityki bezpieczeństwa. Jest instruktorem w SANS Institute, a także współautorem kursów SANS.

Helion
helion.pl
HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-289-2222-8
9 788328 922228
Cena: 59,90 zł