Generowanie haseł

(nienajgorszych)

Łukasz Bromirski

lukasz@bromirski.net

Dobór dobrych haseł do różnego rodzaju zastosowań, oraz proste sposoby na ich generowanie i zapamiętywanie.

Spis treści

1. Hasło? Dobre hasło?

Problem doboru możliwie silnego sposobu uwierzytelniania się w systemach informatycznych jest tak stary jak pierwsze systemy wielożytkownikowe. Od prostego wpisywania hasła, informatyka przeszła do wpisywania identyfikatora i hasła, później pojawiły się inne mechanizmy uwierzytelniania - z obecnie istotniejszych i najpopularniejszych warto wymienić infrastrukturę klucza publicznego oraz mechanizmy używające haseł jednorazowych generowanych przez sprzętowe żetony bezpieczeństwa.

W tym artykule skupię się nad zagadnieniem wygenerowania silnego i trudnego do złamania hasła do użytku w aplikacjach takich jak np. `telnet` czy `ssh`.

1.1. Jak poznać dobre hasło?

Na każdym szkoleniu związanym z bezpieczeństwem, omawia się zagadnienie doboru dobrego hasła i przy okazji podaje jakich haseł nie wolno używać. Na pewno nie wolno używać haseł prostych, składających się z tylko jednej nazwy, czy stwierdzenia. Dzisiaj standardem jest już tworzenie haseł wieloczłonowych, w których używa się urozmaiconego sposobu zapisu danych. Dla przypomnienia poniżej mniej lub bardziej kompletna lista *niebezpiecznych* sposobów wymyślania haseł:

1. **Użycie tylko swoich informacji osobistych**

   Jeśli wydaje Ci się że Twoje imię czy nazwisko czy nawet data urodzenia nie zostaną sprawdzone - mylisz się. Sposobów na zdobycie informacji osobistych są setki, więc nie stosujemy imion, nazwisk, dat urodzenia ani swoich, ani osób nam bliskich (czy również, co częste, imion nadawanych rozmaitym zwierzątkom hodowanym w domu)

2. **użycie tylko nazw własnych**

   Coca Cola, Sprite, Mercedes, Macintosh - wszystkie te nazwy zostaną sprawdzone w wyniku ataku słownikowego. Nie używaj ich.

3. **użycie tylko cyfr, tylko małych lub dużych liter**

   Dobre hasło składać się będzie zarówno z cyfr, małych i dużych liter, jak i ze znaków specjalnych (jeśli mechanizm uwierzytelniania na to pozwala). Znaki specjalne to między innymi: `!`, `@`, `#`, `$`, `%`, `^`, `&`, `*`, `(`, `)`, `[`, `]`, `/`, `?`, `\`, `|`, `_` czy `~`.

W przeciwieństwie do złych haseł, *dobre hasło* będzie miało następujące cechy:

1. Będzie mieszanką małych i dużych liter, cyfr i znaków specjalnych, o długości nie mniejszej niż **8** znaków

   Kiedyś zakładano nie mniej niż 5, później 6 znaków. Należy założyć, że czym dłuższe hasło tym lepiej, zakładając oczywiście że jesteś w stanie je prawidłowo zapamiętać.

2. Nie zawiera informacji osobistych

   Własne dane wszelkiego rodzaju, dane rodziny, ulubionych zwierzątek, nazwy własne miejsc itp. - należy się ich wystrzegać, bo jeśli nawet zapisane będą wymyślnie, mogą osłabić jakość hasła.

3. Nie zostało nigdzie zapisane

   Nawet jeśli jest to niebywale tajne i osobiste miejsce, zawsze może dojść do jakiegoś nieprzewidzianego wypadku.

4. Nie ma podejrzeń, że mogło zostać podejrzane podczas wpisywania

   Dobrze wybrać długie hasło, ale jednocześnie takie, które jesteśmy w stanie wpisać szybko. Nie zawsze trafimy na przyzwoitych ludzi, którzy gdy będziemy się logować odwrócą głowę.

5. Nie znajduje się w słowniku

   Ani jego odmiana, ani odwrócenie, ani proste zamiany znaków miejscami, ani dodanie pojedyńczych cyfr itp. Programy do łamania haseł wykonują takie testy automatycznie.

6. Nie jest prostym wzorem

   Wpisy typu ‘`1234567890`’ czy ‘`qwertyuiop`’ też nie są dobrym pomysłem.

1.2. Generowanie haseł

Samo wygenerowanie bardzo trudnego hasła nie stanowi problemu, kwestia doboru hasła tak, by możliwe było jego zapamiętanie. Jeśli nie musisz wpisywać hasła ręcznie, a służy ono np. do automatycznego uwierzytelnienia podczas nawiązywania sesji IPsec, polecam coś takiego:

```
# openssl rand 20 | hexdump -e '20/1 "%02x"' > plik_z_hasłem
```

W wynikowym pliku '`plik_z_hasłem`' powinieneś otrzymać coś takiego:

```
# cat plik_z_haslem
6660090136d12f0109007032728eca9fb8e0886f
```

Krótkie, prawda? Lepsze będzie takie:

```
# openssl rand 128 | hexdump -e '128/1 "%02x"' > plik_z_hasłem
```

```
# cat plik_z_haslem
a3b075c25876c3b9e09fe9d6c7f4619fbe7ce339dc2924ffa4f7fc7afe1a197d
b21148c05063050564343d0d19909b42b76f40b97b2a54af7a29ca626f7a1ed3
b2277e2d8a82196d6bd2ae1e582dac7c10629da72ff68906d463f2b80bc9f88c
e1416d50523f295d314476cb599eef4635532157e68b24d29ddb7343ea3fefe7
```

Takie hasła są idealne jeśli chodzi o automatyzację pewnych procesów i przeprowadzanie ich w sposób bezpieczny (łatwo i szybko można wygenerować nowe, w dodatku samo hasło jest raczej trudne do "zgadnięcia"). Co jednak jeśli musisz je zapamiętać?

> **Ostrzeżenie**
>
> **Każde** podane tu hasło staje się automatycznie niedopuszczalne, ponieważ zostało publicznie opublikowane i może zostać dołączone do pliku słownikowego!

Najprościej a jednocześnie niekoniecznie najmniej bezpiecznie jest wymyślić sobie jakieś długie, dobrze zapadające w pamięć zdanie i wziąć z niego co *n*-tą literę. Przećwiczmy to:

```
pancernik to fajna książka o UNIXie ale kosztuje 70 PLN
```

Bierzemy pierwsze litery i całe cyfry i tworzymy hasło: `ptfkoUak70P` - z marszu trudne do zapamiętania, ale jeśli wykujesz zdanie - nie ma problemu. Drugie litery - też nie problemu: `aoasoNlo70L`.

Oczywiście uniwersalnego przepisu nie ma, ale ten jest całkiem niezły. Najlepiej jest wziąć w rękę ulubioną książkę, otworzyć na chybił-trafił i przepisać np. dwa zdania. Następnie wyuczyć się ich i już mamy przykład dla prawdziwych h4x0r0v:

```
Operatory ++ i -- działają tak samo jak w języku C. To znaczy, jeżeli są
umieszczone przed zmienną, to inkrementują lub dekrementują tę zmienną przed
zwróceniem wartości a gdy są umieszczone za zmienną, robią to dopiero po
zwróceniu wartości.
```

Perl: Programowanie, str. 87, Larry Wall, Tom Christiansen, Jon Orwant, O'Reilly. Wydanie polskie: Wydawnictwo RM, tłumaczenie Marcin Moskwa

Pierwsze litery: `O+i-dtsjwjC.Tzjsupztildtzpzwagsuzzrtdpzw.` - niezłe, prawda? A zdanie jest logiczne i można się go nauczyć ;)

Parę porad na koniec:

ZAWSZE używaj różnych haseł nie tylko dla każdej maszyny, na którą się logujesz, ale także dla różnych usług

> Jeśli na jednym serwerze masz zarówno konto SSH, FTP jak i IMAP4, staraj się dla każdego używać osobnego, różnego hasła. Nawiasem mówiąc, posiadając dostęp przez SSH powinieneś zrezygnować z FTP i używać SCP zawartego w pakiecie SSH. To samo dotyczy również innych usług uwierzytelniających - np. dostępu do zahasłowanych stron w ramach mechanizmu `.htaccess` na serwerze Apache itp.

Nie przechowuj haseł a nawet loginów w formie czytelnej dla postronnej osoby

> Przechowywanie w portfelu karteczki z hasłami to świetny pomysł, ale tylko jeśli chcesz narobić sobie problemów. Zdesperowany włamywacz, lub dobry socjotechnik poza jej zawartość prawdopodobnie dużo prościej i szybciej, niż gdyby miał włamywać się za pomocą różnych programów. Na tej samej zasadzie Palmy, komórki i inne notatniki elektroniczne nie są dobrym miejscem do przechowywania tajnych informacji. Ćwicz pamięć!

Wykaż się inwencją, nie bezmyślnością czy beztroską.

> Proste hasła, nawet na teoretycznie nieistotne maszyny, mogą prowadzić do problemów. Zwróć uwagę na to, że istnieje cała klasa problemów z bezpieczeństwem, które wymagają dowolnego konta lokalnego użytkownika na systemie, a które nie są do wykorzystania zupełnie z zewnątrz.

Przyzwyczaj się do regularnej zmiany haseł.

> W zależności od znaczenia danego konta, zmieniaj hasła codziennie (tak! są takie systemy!) lub co np. tydzień czy miesiąc. Postaraj się ocenić wartość algorytmu szyfrowania danych uwierzytelniających w danym systemie, by móc ocenić ilość czasu potrzebną na jego złamanie. Hasło zaszyfrowane kluczem DES (40-bitowym) można dzisiaj złamać w parę naście godzin, hasło zapisane algorytmem AES-256 będzie bronić się przy ataku *brute force* w dobrym przypadku parę miliardów lat...

2. Materiały

Establishing Good Password Policies

> http://www.onlamp.com/lpt/a/bsd/2001/01/17/FreeBSD_Basics.html

Jak skonstruować dobre hasło?

> http://www.cert.pl/PDF/dobre_haslo.pdf