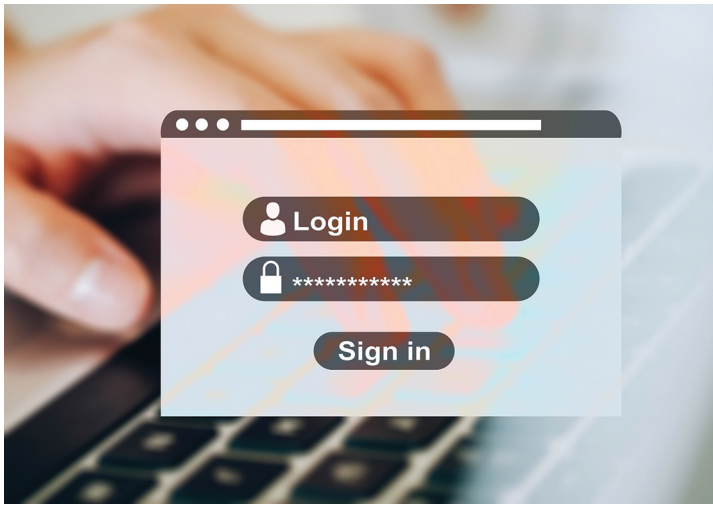
Lozinke i matematika

Sandra Gračan, Zagreb



U digitalnom svijetu kojim se danas gotovo svatko od nas koristi često smo prisiljeni kreirati ili promijeniti svoju lozinku kao svojevrsnu zaštitu za pristup nekim internetskim sadržajima ili uslugama. Trebamo li za lozinku odabrati niz znakova koji ćemo relativno lako pamtiti ili pak niz znakova odabranih na sasvim slučajan način? Kako znati koliko nam je lozinka jaka te kako funkcioniraju hakeri i njihovi programi za probijanje lozinka? U odgovorima na ova pitanja skriva se matematika.

Pri određivanju lozinke često ćemo dobiti poruku o njezinoj jačini i neke savjete za odabir slova i simbola. Sastavljamo li primjerice lozinku od 6 znakova, birajući neka od 26 malih slova s tipkovnice na slučajan način, tada je to jedna od ukupno $26^6 = 308\,915\,776$ mogućih lozinka. Upisujemo li mala slova, velika slova i simbole, tj. biramo li znakove iz skupa od 72 elementa pa ako još k tome kreiramo lozinku duljine 12 znakova, tada postoji ukupno 72^{12} mogućnosti, što je približno jednako broju $19 \cdot 10^{21}$ (19 trilijardi) ili oko 63 bilijuna ($63 \cdot 10^{12}$) puta više nego u prvom slučaju. Računalo koje bi ispitivalo sve moguće lozinke s ciljem da otkrije pravu, imalo bi sada itekako više posla nego u prvom slučaju.

Broj elemenata domene, tj. ukupan broj svih mogućih lozinka prvi je važan faktor sigurnosti naše lozinke. Označimo li broj svih mogućih lozinka sa N, duljinu lozinke sa k, a broj svih znakova koje smijemo upotrijebiti pri sastavljanju te lozinke sa n, tada vrijedi:

$$N = n^k.$$

Svaka se lozinka u računalu prevodi u binarni zapis, odnosno niz nula i jedinica određene duljine. Stoga se pri računanju jačine pojedine lozinke zapravo gleda broj bitova koji ta lozinka zauzima u binarnom zapisu. Naravno, on direktno ovisi o broju elemenata domene. Označimo li broj bitova slovom b, tada vrijedi:

$$b = 1 + \lfloor \log_2 N \rfloor.$$

Tako za $N = 26^6$ dobivamo $b = 29$, odnosno lozinka sastavljena od 6 iz skupa od 26 znakova ima u binarnom zapisu duljinu 29 bitova. Za veću domenu, primjerice za $N = 72^{12}$, dobivamo $b = 75$, odnosno za binarni zapis lozinke zapisane s 12 znakova iz skupa od 72 znaka potrebno je 75 bitova.

Sandra Gračan, dipl. inž. matematike, Element d. o. o., Zagreb, **sandra@element.hr**

Matematičari će reći da skup svih mogućih lozinka ima **entropiju** 29 ili 75.

Francuska Nacionalna agencija za kibernetičku sigurnost (ANSSI) preporuča duljinu od najmanje 100 bitova za sasvim sigurne lozinke. Lozinka od 128 bitova garantira nam sigurnost nekoliko godina, dok lozinka od 64 bita predstavlja jako slabu sigurnost. Lozinke duljine između 64 i 80 bitova su slabe, a lozinke duljine od 80 do 100 bitova su srednje ili umjereno jake lozinke.

Trebamo imati na umu i to da snaga računala stalno raste, gotovo da se udvostručuje svake dvije godine. Ono što je danas srednje jaka ili jaka lozinka, za neko vrijeme više sigurno neće biti.

Dakle, za zaista jaku lozinku treba nam niz od 16 slova i simbola odabranih iz skupa od ukupno 200 znakova. Pa na tipkovnici niti nema toliko znakova. Takva lozinka zauzimala bi 123 bita, a gotovo sigurno je ne bismo mogli zapamtiti. Stoga sistemski inženjeri najčešće korisnicima dopuštaju lozinke slabe ili srednje sigurnosti i tada se koriste drugim načinima borbe protiv zloporabe lozinka, kao što je primjerice blokiranje pristupa sustavu nakon triju pogrešnih unosa lozinke. Jake lozinke zahtijevaju se samo kada ih automatski generiraju sami sustavi pa ih korisnici niti ne trebaju pamtiti i upisivati.

Vrijeme pretraživanja

Neka je, dakle, k duljina lozinke, n ukupan broj svih znakova koje smijemo upotrijebiti, a N broj svih mogućih lozinka ($N = n^k$). Označimo sada sa T broj sati koje hakeri, odnosno njihova računala moraju provesti u ispitivanju *svih* mogućnosti, isprobavajući jednu po jednu lozinku. Taj broj ovisit će o brzini rada računala. Pretpostavit ćemo da računala mogu ispitati milijardu (10^9) lozinka u jednoj sekundi. Imajući na umu da se ta brzina svake godine udvostuči, računat ćemo još i G, odnosno broj godina za koje će računala moći ispitati cijeli skup od n lozinka za manje od sat vremena. Dakle, vrijedi:

$$N = n^k$$
$$T = \frac{N}{10^9 \cdot 3600} = \frac{n^k}{10^9 \cdot 3600}$$
$$G = 2\log_2 t = 2\log_2\left(\frac{N}{10^9 \cdot 3600}\right).$$

Za $n = 26$ i $k = 6$, odnosno $N = 308\,915\,776$ dobivamo $T = 0.0000858$ sati potrebnih za ispitivanje svih mogućih lozinka, a $G = 0$ jer je već moguće ispitati sve lozinke za manje od sat vremena, točnije za manje od 0.31 sekunde.

Za $n = 26$ i $k = 12$ vrijedi da je $N = 9.5 \cdot 10^{16}$ i potrebno je $T = 26\,508$ sati za ispitivanje, a $G = 29$, odnosno za 29 godina to će vrijeme biti kraće od jednoga sata.

Za $n = 100$ i $k = 10$ vrijedi da je $N = 10^{20}$ i potrebno je $T = 27\,777\,777$ sati za ispitivanje, a $G = 49$ godina. Za $n = 100$ i $k = 15$ dobivamo $N = 10^{30}$, $T = 2.7 \cdot 10^{17}$ sati i $G = 115$ godina, a za $n = 200$ i $k = 20$ dobivamo $N = 1.05 \cdot 10^{46}$, $T = 2.7 \cdot 10^{33}$ sati i $G = 222$ godine.

Hakerski trikovi

Naravno, sav ovaj izračun temelji se na pretpostavci da lozinku kreiramo odabirući znakove na sasvim slučajan način. Ali znamo da u stvarnosti nije tako. Kako bismo ih lakše pamtili, lozinke uglavnom biramo tako da imaju neki smisao ili značenje, što predstavlja veliki sigurnosni rizik i hakerima znatno olakšava posao. Hakeri se koriste gotovim listama često korištenih lozinka, tzv. **rječnicima lozinka** koji omogućuju napadačima da sistematski isprobavaju jednu po jednu lozinku s popisa i pritom imaju veliku šansu da će pronaći pravu. Odabiremo li samo imena, nadimke, neke riječi ili kratke rečenice, jako smo smanjili veličinu domene, tj. broj N. Pa čak i kad mislimo da smo lozinku odabrali na sasvim slučajan način, skloni smo (svjesno ili nesvjesno) raditi uzorke, volimo simetriju i jednostavne kombinacije slova i brojki, poput "lozinka123", drugim riječima, prilično smo predvidljivi. Rječnici lozinka razlikuju se od jezika do jezika, ali čak i bez

njih, analizom frekvencije pojavljivanja nekih slova i otkrivanjem bilo kakvih pravilnosti hakeri dodatno ubrzavaju vrijeme otkrivanja lozinke.

Evo nekih najčešće korištenih lozinka: 123456, password, 12345678, qwerty, 12345, 123456789, admin, login, abc123, 123123, hello, qaywsx[1] itd. Za četveroznamenkaste brojevne lozinke kakve koristimo primjerice za podizanje novaca na bankomatima ili pak kao PIN kodove za SIM kartice ili pametne telefone, situacija je još gora. Najupotrebljavaniji niz je 1234, te 1111 i 0000, a zatim slijede 1212 i 7777.

Kako zagorčati posao hakerima

Brojni internetski servisi čuvaju naša korisnička imena i lozinke kako bi nam olakšali pristup i korištenje njihovim uslugama. Pritom su svjesni opasnosti zloporabe tih podataka pa su razvili poseban način zaštite. Oni zapravo ne spremaju korisnička imena i lozinke već korisnička imena i **otiske**, odnosno nizove znakova izvedenih iz lozinka. U slučaju napada hakerima su ti otisci beskorisni jer iz njih ne mogu doći do originalnih lozinka korisnika. Ili ipak mogu?

Dobivanje otisaka transformacija je koja se odvija s pomoću algoritama poznatih kao kriptografske ***hash*-funkcije**. Ti pametni algoritmi preslikavaju neku datoteku F, ma koliko ona velika bila, u niz znakova fiksne duljine $h(F)$. Primjerice, *hash*--funkcija SHA256 preslikat će izraz "dobar dan" u niz:

AFC875450034E75B9BAE426CD5A7CC2A8E235-CF22723BBBE71F7E178B3016F09,

a promijenimo li samo jedno slovo – stavimo li primjerice veliko prvo slovo: "Dobar dan" – tada dobivamo sasvim drugi SHA256 zapis:

9B71B25F0E1BEF5474A16CB2777D60822952B-E3186C45843F074839CB70A8987.

Dobiveni zapisi su heksadecimalni, tj. u bazi 16, odnosno u računalu zauzimaju 256 bitova. Na internetskoj stranici `https://passwordsgenerator.net/sha256-hash-generator/` i sami možete isprobati te transformacije.

Dobre *hash*-funkcije stvaraju otiske lozinka kao da smo ih birali na potpuno slučajan način. Točnije, za svaki mogući slučajno odabrani rezultat $h(F)$ nemoguće je otkriti podatak F u nekom razumnom vremenu. Već sada postoji nekoliko generacija *hash*-funkcija. Generacije SHA0 i SHA1 zastarjele su i ne preporučuju se, a funkcije generacije SHA2, uključujući SHA256, smatraju se sigurnima. Najnoviji član familije SHA-standarda je SHA3, koji je objavljen 2015. godine. Kratica SHA dolazi iz engl. *Secure Hash Algorithm*.

Savjet korisnicima

Dobro dizajnirane internetske stranice analizirat će lozinku koju upisujemo i odbiti one koje bi se lako mogle probiti. To nas može iznervirati, ali za naše je dobro. Iz gore navedenog nameće se zaključak da naše lozinke moramo birati potpuno nasumice. I za to nam se nude gotovi programi, ali i pri njihovu korištenju valja biti na oprezu jer mogu biti (namjerno ili ne) loši.

Na internetu možemo provjeriti čak i to je li naša lozinka već bila provaljena. Postoji alat koji se zove *Pwned Passwords* (`https://haveibeenpwned.com/Passwords`). Primjerice, lozinka "e=mc2e=mc2" bila je provaljena 119 puta, a lozinka "aaaaaa" čak 400 874 puta. Pokušajte upisati svoju lozinku i dobit ćete poruku je li ona (i koliko puta) bila provaljena.

I na kraju, nikako nemojte svoje lozinke zapisati u neki dokument i pohraniti ga na svojem računalu kao običan tekst. To možete učiniti samo ako taj tekst šifrirate, a ključ za dešifriranje upamtite ili pohranite negdje izvan digitalnog svijeta.

[1] Nizovi slova "qwerty" i "qaywsx" dolaze od rasporeda slova na (lijevom) dijelu tipkovnice.

Dobri pružatelji internetskih usluga za spremanje korisničkih imena i lozinka svojih klijenata trebaju upotrebljavati *hash*-funkcije. Kad se korisnik želi spojiti, server će pročitati upisane podatke, izračunati otisak lozinke i utvrditi nalazi li se on na listi svih spremljenih korisničkih imena i otisaka. Čak i ako hakeri dođu do tog popisa, on im je beskoristan jer ne mogu otkriti lozinku, niti generirati neku novu lozinku kako bi zavarali servere i upali na stranice umjesto korisnika.

Kako bi spriječili hakere da zloupotrijebe liste otisaka (jer se i one mogu ukrasti), sistemski inženjeri koriste se još jednom metodom koju nazivaju **soljenje**. To je postupak dodatne zaštite kojim se svakom otisku dopisuje dodatni niz slučajno odabranih znakova. Tako će i korisnici s istom lozinkom sigurno imati različite otiske lozinka jer se "sol" stalno može mijenjati i biti drukčija za svakog korisnika, a čak i kad su lozinke slabe, metoda soljenja dodatno komplicira hakerima posao i znatno im produljuje vrijeme provaljivanja.

Hakeri uzvraćaju udarac

U doba interneta, superračunala i računalnih mreža, znanost o lozinkama i njihovu razbijanju nastavlja se razvijati, kao i neprestana borba između onih koji štite i čuvaju lozinke i onih koji ih žele ukrasti i zloupotrijebiti. I hakeri imaju svoje trikove. Oni su u dilemi: ili im trebaju izuzetno jaka računala ili računala s izuzetno puno memorije.

Pretpostavite da ste haker i da želite iskoristiti neku bazu podataka koju ste ukrali s nekog servera. Podatci se sastoje od parova oblika (korisničko ime, otisak lozinke), a znate koja se *hash*-funkcija koristila za dobivanje tih otisaka. Znate i to da se lozinka sastoji od 12 malih slova što odgovara 56-bitnim podatcima iz skupa od 26^{12} mogućnosti. Vaš je cilj otkriti lozinku jednog korisnika.

Možete ispitivati jednu po jednu mogućnost, odnosno za svaku lozinku L iz domene morate računati njezin otisak $h(L)$ te ispitati pojavljuje li se taj otisak među ukradenim podatcima. Za to vam treba jako puno vremena. Ako računalo izvodi milijardu operacija uspoređivanja u sekundi, trebat će vam $\frac{26^{12}}{10^9 \cdot 3600 \cdot 24} = 1104$ dana ili oko tri godine za traženje jednog otiska. S obzirom na to da ne želite pohraniti sve te otiske, ne trošite previše računalne memorije, ali poželite li otkriti lozinku nekog drugog korisnika, sav posao kreće ispočetka. Cilj nije nedostižan u slučaju da imate na raspolaganju mrežu od 1000 računala – tada biste ta ispitivanja odradili za jedan dan.

Druga metoda jest da sve lozinke i dobivene otiske spremate u jednu veliku tablicu. Za to bi vam trebalo $(9.54 \cdot 10^{16}) \cdot (7 + 32) \approx 3.7 \cdot 10^{18}$ bajtova memorije (7 bajtova za lozinku i 32 bajta za otisak ako se on sastoji od 256 bitova jer je dobiven *hash*-funkcijom SHA256). To je dakle 3.7 milijuna hard diskova kapaciteta jedan terabajt.

Obje su metode problematične: ili se svaka nova lozinka predugo ispituje ili spremanje svih mogućnosti zahtijeva previše memorije. Ali, postoji kompromis. 1980. godine Martin Hellman sa Sveučilišta Stanford osmislio je metodu koju su 2003. godine i kasnije doradili Philippe Oechslin sa švicarskog Federalnog instituta za tehnologiju *ETH Zürich* i Gildas Avoine s francuskog Nacionalnog instituta za primijenjene znanosti *INSA Rennes*. Oni su osmislili takozvane **dugine tablice**.

Ljepota duginih tablica

Evo o čemu je riječ. Potrebna nam je najprije funkcija M koja svakom otisku $h(L)$ pridružuje novu lozinku $M(h(L))$ koja pripada skupu svih mogućih lozinka, tj. domeni. Zamislimo li primjerice otiske kao brojeve zapisane u binarnom sustavu, lako možemo zamisliti i funkciju M koja te nizove nula i jedinica pretvara natrag u brojeve zapisane u sustavu sa k znamenaka, gdje je k broj dozvoljenih znakova za zapis lozinke.

Proces kreiranja dugine tablice kreće od prve moguće lozinke L_0 za koju načinimo njezin otisak $h(L_0)$. Zatim s pomoću funkcije M "izračunamo" drugu moguću lozinku $M(h(L_0))$, koja postaje L_1. Ovaj postupak sada ponavljamo za L_1 i tako dobivamo niz lozinka $L_1, L_2, \ldots$ i njihovih otisaka

$h(L_1), h(L_2), \ldots$ sve dok ne dođemo do otiska koji započinje s 20 nula. Takav se otisak sigurno pojavljuje jednom u otprilike 1 000 000 otisaka jer je rezultat *hash*-funkcije sličan nasumičnom odabiru, a 2^{20} je približno jednako milijun. Taj otisak označimo sa $h(L_n)$. Par $(L_0, h(L_n))$ koji se sastoji od početne lozinke i konačnog otiska koji započinje s 20 nula sprema se u duginu tablicu.

Na isti se način "računa" i u tablicu zapisuje i sljedeći par, koji je krenuo od lozinke koja se nije pojavila nigdje unutar prethodnog lanca kao neka vrijednost $M(h(L_i))$, $i = 1, \ldots, n$. U konačnici, tablica će sadržavati niz takvih parova, a svaki par zapravo predstavlja čitav niz lozinka $L_0, L_1, \ldots, L_n$ i njihovih otisaka, ali se podatci dobiveni u međukoracima ne pamte.

U tako dobivenim tablicama mogu se dogoditi neka preskakanja, odnosno neke se lozinke možda neće pojaviti niti u jednom lancu. Pa ipak, za dobru bazu podataka dobivaju se tablice gotovo bez "rupa" koje sada zauzimaju milijun puta manje računalne memorije nego tablice s podatcima svih korisnika. To su manje od četiri tvrda diska od jednog terabajta.

Uz poznate funkcije h i M i s pomoću takvih tablica hakerima je moguće odgonetnuti lozinku iz njezina otiska u nekoliko sekundi. Evo kako.

Pretpostavimo da opisano preračunavanje pokriva sve lozinke iz domene. Ako je ukraden otisak h_0, računa se $h(M(h_0))$, odnosno otisak od lozinke dobivene iz ukradenog otiska s pomoću funkcije M. Označimo ga sa h_1, pa zatim računamo $h(M(h_1)) = h_2$ i tako dalje. Postupak ponavljamo sve dok ne dobijemo otisak h_i koji započinje s 20 nula. I sada taj otisak potražimo u duginoj tablici te pročitamo početnu lozinku L_0 koja mu je pridružena.

Počevši od lozinke L_0, sada možemo izračunati novi niz otisaka $h_1, h_2, \ldots$. Prije ili kasnije u tom lancu, u nekom koraku j, dobit ćemo otisak h_j koji je jednak ukradenom otisku h_0. Lozinka koju tražimo je upravo ona iz koje smo dobili taj otisak, imali smo je u prethodnom koraku, odnosno to je $M(h_{j-1})$.

Potrebno vrijeme računanja je ono za traženje otiska h_i u tablici plus vrijeme potrebno za računanje dvaju nizova otisaka $(h_1, h_2, \ldots, h_i)$ i $(h_1, h_2, \ldots, h_j)$, što je milijun puta kraće od vremena potrebnog za kreiranje same tablice.

Nizovi

$$A_0 \xrightarrow{h} h(A_0) \xrightarrow{M} A_1 \xrightarrow{h} h(A_1) \xrightarrow{M} A_2 \xrightarrow{h} \ldots \xrightarrow{h} h(A_n)$$
$$B_0 \xrightarrow{h} h(B_0) \xrightarrow{M} B_1 \xrightarrow{h} h(B_1) \xrightarrow{M} B_2 \xrightarrow{h} \ldots \xrightarrow{h} h(B_n)$$
$$\vdots$$
$$Z_0 \xrightarrow{h} h(Z_0) \xrightarrow{M} Z_1 \xrightarrow{h} h(Z_1) \xrightarrow{M} Z_2 \xrightarrow{h} \ldots \xrightarrow{h} h(Z_n)$$

prikazuju pojedine lance koji vode od lozinka A_0, B_0 i tako dalje do Z_0 preko njihovih otisaka i novih lozinka sve do otiska koji počinju s 20 nula. Poznavanjem samo početka i kraja svakog lanca hakeri će, uz nešto preračunavanja, probiti bilo koju lozinku.

LITERATURA

1/ Jean-Paul Delahaye (2019.): *The Mathematics of (Hacking) Passwords*, objavljeno 12. 4. 2019. `https://www.scientificamerican.com/article/the-mathematics-of-hacking-pašwords/`

2/ K. G. Orphanides: *How to create a genuinely strong pas|sword for your digital life*, objavljeno 22. 2. 2018. `https://www.wired.co.uk/article/how-secure-is-my-password-good-strong-password-ideas`