Tečajevi naprednog računarstva u Križevcima

HACK2020 | Hub for Advanced Computing Križevci

Osnove računarstva visokih performansi

Vol. 8: Zadatak – *Brute-force* multithreadingom

Andrej Dundović

Križevci, 16. 7. 2021.

Organizator: udruga point križevci

Pokrovitelj: Embassy of the United States Zagreb - Croatia

Što je kriptografska raspršna funkcija (hash)?

- *hash* funkcija je bilo koja funkcija koja može mapirati podatke proizvoljen duljine u vrijednosti fiksne duljine
- vrijednost fiksne duljine: hash vrijednost, hash kod, digest (sažetak)
- najjednostavniji primjer "hash funkcije" za cijele brojeve: $f_n(x) = x \bmod n$
- kolizija je kada dva različita podatka daju istu hash vrijednost
- dobra hash funkcija statistički minimizira šansu za kolizijom
- hash funkcija koja je korisna za spremanje zaporki je *spora* (zato da bi se otežalo otkrivanje inverza, odnosno podatka iz hasha)
- primjeri kriptografskih hash funkcija: MD5, WHIRLPOOL, SHA1, SHA256, SHA512

 andrej.dundovic@udruga-point.hr

SHA1 hash zaporke

- SHA-1 (Secure Hash Algorithm 1), standard od 1993, ali od 2005 više se ne smatra sigurnom hash funkcijom (napušta se od 2011)
- iako se u praksi više ne bi trebale koristiti, pogotovo ne za hash zaporki, ovdje ćemo se pozabaviti upravo navedenom jer se njezine hash vrijednosti relativno brzo računaju na modernom hardveru
- generiranje SHA1 hasha u GNU/Linuxu

```
echo -n "abc" | sha1sum | awk '{ print $1 }' >> hashes.txt
```

- Zadatak: naći izvorne zaporke iz hash vrijednosti u `hashes.txt`

“Brute force”

- “Brute-force” napad (*napad sirovom snagom*) temelji se na isprobavanju mnogih kombinacija u nadi da će se pogoditi ona točna
- takav napad nije baš efikasan, štoviše, najneefikasniji mogući, no upravo zato što je takav, pogodan je u našem kontekstu demonstriranja tehnika računarstva visokih performansi
- inače se za neautorizirano dobivanje zaporki koristi socijalni inženjering, a ako se već koristi “brute-force”, koriste se rječnici ili baze uobičajenih zaporki kao odabrani podskup za pogađanje
- broj svih mogućih kombinacija je n^ℓ gdje je n skup svih mogućih znakova, a ℓ duljina zaporke
- primjer: za velika i mala slova engleskog alfabeta + brojevi (0–9) imamo $n = 26 \times 2 + 10$, pa za zaporke duljine 3 imamo 238k kombinacija, 4 – 15M kombinacija, 5 – 916M kombinacija, 6 – 56G kombinacija, itd. - razbijanje hash vrijednosti u principu za neselektivni “brute-force” napad postaje vrlo teško nakon 7-8 znakova, ovisno o hash funkciji

Primjer (elegantnog) rješenja u Rustu...

...bazirano na rekurziji...

```
fn get_comb(&self, k: u32) -> Vec<String> {
    if k == 1 { return self.all_chars.clone(); }
    if k > 4 { panic!("Not enough memory for k > 4"); }

    let capacity = self.all_chars.len().pow(k);
    let mut vec: Vec<String> = Vec::with_capacity(capacity);

    for c in &self.all_chars {
       vec.extend(Comb::get_comb(self, k - 1)
        .into_iter().map(|s| s + &c));
    }
    vec
}
(...)
let results: Vec<&str> = comb.iter()
    .filter_map(|s| check_hashes(&s, &hashes)).collect();
```

...koje ne radi jer nakon duljine veće od 4 vektor kombinacija zauzima previše memorije :-(

Drugi pristup preko iteratora...

...koji broji trenutnu kombinaciju i povećava je svaki puta za jedan, a knjigovodstvo vršimo preko $(d_{l-1}, \ldots, d_1, d_0)$:

$$n_c = d_{l-1} n_s^{l-1} + d_{l-2} n_s^{l-2} + \cdots + d_1 n_s^1 + d_0 n_s^0 \tag{1}$$

$$\Rightarrow d_0 = n_c \bmod n_s \tag{2}$$

$$\Rightarrow d_1 = \frac{(n_c - d_0)}{n_s} \bmod n_s \tag{3}$$

$$\Rightarrow \ldots \tag{4}$$

- Kako to paralelizirati?

 andrej.dundovic@udruga-point.hr

Mjerenje performansi

hyperfine – A command-line benchmarking tool.[1]

Korištenje:

```
$ hyperfine './target/release/password_cracker hashes.txt 4'

Benchmark #1: [User: 4.585 s, System: 0.193 s]
  Time (mean +/- std):     4.786 s +/-  0.077 s
  Range (min ... max):     4.647 s ...  4.910 s    10 runs
```

[1] github.com/sharkdp/hyperfine

Usporedba

Password length	Single thread	Multithreading with mutex	Multithreading without mutex
3	55.0 ms ± 1.8 ms	53.9 ms ± 1.0 ms	16.7 ms ± 1.2 ms
4	3.507 s ± 0.144 s s	4.078 s ± 0.333 s	1.081 s ± 0.039 s
5	224.253 s ± 1.612 s s	365.073 s	87.138 s ± 10.118 s

Figure: Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz (4 cores/8 threads)

Password length	Single thread	Multithreading with mutex	Multithreading without mutex
3	74.7 ms ± 0.2 ms	216.9 ms ± 4.4 ms	8.5 ms ± 1.0 ms
4	4.574 s ± 0.026 s	13.573 s ± 0.248 s	330.0 ms ± 35.7 ms s
5	284.679 s ± 0.656 s	894.230 s	17.668 s ± 0.125 s

Figure: Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz (10 cores/20 threads)

Usporedba

Password length	Single thread	Multithreading with mutex	Multithreading without mutex
3	81.7 ms $\pm$ 11.5 ms	83.5 ms $\pm$ 3.5 ms	14.8 ms $\pm$ 0.6 ms
4	3.965 s $\pm$ 0.100 s	4.534 s $\pm$ 0.137 s	482.1 ms $\pm$ 5.5 ms
5	281.810 s $\pm$ 3.876 s	689.11s	26.864 s $\pm$ 0.855 s

Figure: AMD(R) Ryzen(R) 7 2700X (8 cores/16 threads)

Usporedba

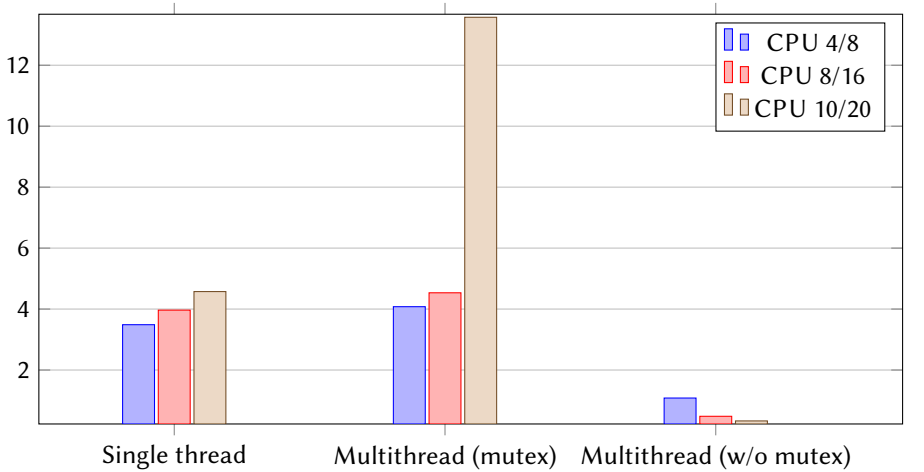

Figure: Usporedba vremena (single/multi+mutex/multi-wo-mutex) na različitim procesorima za duljinu zaporke 4

Podsjetnik

- Spajanje na udaljeno računalo putem SSH-a (Secure Shell Protocol)
- Klijenti za: za Windows – Putty, za Linux/Mac – ssh
- Otvoriti račun na poslužitelju za svakog polaznika (`username`)
- Zaporka: naslov današnje radionice (etherpad, sve malim slovima)

```
ssh username@vavra.krizevci.info -p2212
```

- Promijeniti zaporku s `passwd`
- Zauzeće resursa možemo pogledati: `free -h`, `uptime` (load average), `htop`
- `screen`: (Ctrl+a c, Ctrl+a d), `screen -r`