Sphinx es a bruteforszorny

stef

<2021>

emlekezteto

sphinx: olyan jelszo "tarolo", amely rengeteg elonnyel rendelkezik a hagyomanyos tarolokkal szemben.

`rwd = argon2i(pwd || hash-to-curve(pwd)*k)`

mindezt ugy, h. **k** a szerveren van tarolva, es a szerver semmit nem tud meg a felhasznalo mester es account jelszavarol - es emiatt az nsa is hostolhatna toron keresztul[1]

[1] (persze az nsa torolheti a **k**-dat, es ezzel DoSol teged, meg statisztikat keszithet melyik k-t milyen gyakran hasznaljak milyen ipcimrol ha nem torozol)

bruteforce tamadoi modell

harom szereplo, negy vagyontargy:

- kliens $\leftarrow$ a koronaekszer a **mesterjelszo**!
- kliens $\leftarrow$ kliens mestertitok
- sphinx szerver $\leftarrow$ k
- account server $\leftarrow$ felhasznaloi-adatbazis

a kliens mester titok

kliens mester titok nelkul nem lehet tudni a sphinx serveren melyik rekord milyen user/host paroshoz tartozik. Sphinx recordok IDje igy szamolodik:

```
hash(user+|+host, hash(const, mestertitok))
```

Ha leakelodik, csak akkor tudja a tamado a sphinx szerveren melyik rekordot kell tamadni.

leakelt sphinx server adatbazis

db = userid $\rightarrow$ {k,authpubkey} mappingje

```
for k in userdb:
  for pwd in dictionary:
     query_google("moriczka", hash(pwd+hash(pwd)*k))
```

- ennek ismereteben egy tamado - feltetlezve, h. a moriczka@gmail.com cimhez tartozo jelszohoz tartozo **k** a sphinx server db-ben van
- kenytelen bruteforcolni a mester jelszot minden egyes db beli **k** ellen, es az eredmenyt a gmail loginjanal tesztelni. azaz csak online bruteforce tamadas lehetseges.

leakelt account server user db

userdb = userid $\rightarrow$ {pwd-hash} mappingje

```
for pwd in dictionary:
    if query_sphinx(id, pwd) == userdb['moriczka'][pwd-hash]
        print("found master password: %s\n", pwd)
```

ha kileakelodik egy user db, a benne levo sphinxbol jovo jelszo vedett, hiszen 32byte entropia. az ezt generalo sphinx master jelszo pedig csak online bruteforszolhato a sphinx szervert querizve es osszehasonlitva a valaszat a db beli jelszavakkal. de ehhez persze tudni kell a sphinx szerverbeli useridt, ami a kliens mestertitka nelkul kb lehetetlen.

leak kombinaciok

- mester titok & sphinx db - online bruteforce account server ellen
- mester titok & account user db - online bruteforce sphinx server ellen
- sphinx db & account user db: shotgun offline bruteforce - osszes sphinx K vs known user
- mindharom: celzott offline bruteforce!

vedekezes

- az hogy az account server mit csinal, nincs kontrollunk alatt.
- a sphinx szervert viszont vedhetjuk.

blokkolas n sikertelen kiserlet utan

- sajnos a get muveletnel a szerver nem tudja, h. sikeres volt-e es helyes jelszo generalodott. igy get muveletnel nem tudunk accountot blockolni n sikertelen kiserlet utan.
- management muveletek authentikaltak es fuggoen az rwd_keys beallitastol, fugghetnek az adott account mester jelszavatol - ezt a szerver azonban nem tudja. ha fugg a mester jelszotol, es a szerver tudna, akkor blokkolhatna n sikertelen auth utan.
- a blokkolas sajnos DoS-ra is felhasznalhato, igy ez nem fasza (nem ugrunk seggest)

ratelimiting

- ip alapjan nem realis, mivel ipv6 alatt ilyen /48-/64-es tartomanyok lehetsegesek, egyebkent meg ugye felkeszult tamado botnettel jon.
- userid alapjan mar lehet, felteve h. bevallaljuk, h. ezaltal online userid enumeracio lehetove valik a valasz alapjan. persze vakon 32byteos useridt kitalalni eleg valoszinutlen, igy ez nem nagy riziko.
- kesleltetes ratelimiting, bejovo kapcsolatra csak n ido mulva valaszolunk, ha userid alapjan tesszuk, ehhez azonban mar olvasni kell a requestet, es ez szerver oldali socketet es memoriat foglal. ezaltal DoS tamadast tesz lehetove.
- request/sliding window: percenkent max 2, orankent max 100 request. ehhez useridnkent nyilvan kell tartani a requestek beerkezesenek idopontjat. Es ez is DoS-hoz vektor ismert userid eseten.

client puzzles

- proof-of-work: minden requestnek kell egy u.n. client puzzlet megoldania. ez a client puzzle lehetoseg szerint konfiguralhato nehezsegu, es a helyes megoldas ellenorzese lehetoleg nagyon olcso.
- Egy naiv talalj 28byteos prefixhez olyan 4byteos posztfixet ami argon2ivel hashelve nbit leading zeroval kezdodik, bar jol hangzik. azonban a megoldas ellenorzese a szerver szamara is legalabb 0.1s szamitast igenyel.
- ehelyett:

equihash client puzzles

find distinct n-bit $i_1, i_2, \ldots\ i_{2^k}$
to satisfy $H(i_1) \oplus H(i_2) \oplus \ldots\ \oplus H(i_{2^k}) = 0$
such that $H(i_1 \mid\mid i_2 \mid\mid \ldots\ \mid\mid i_{2^k})$ has d leading zeros.

- `https://www.cryptolux.org/images/b/b9/Equihash.pdf`
- konfiguralhato nehezsegu - adaptivan nehezitheto es konnyitheto
- memoryhard[2, 3]
- nagyon olcso ellenorizni, ilyen 0.000005-0.000022 masodperc nagysagrendu, szemben a naiv argon2i 0.1s-ehez kepest gen8as intelen.

---

[2] A Note on the Security of Equihash - `https://dl.acm.org/doi/10.1145/3140649.3140652`

> *The main purpose of this short note is to raise awareness that Equihash should be considered a heuristic scheme with no formally proven security guarantees.*

[3] Improved Quantum Algorithms for the k-XOR Problem: `https://eprint.iacr.org/2021/407`

benchmarks intel i7-8550

n	k	mem	solve (s)	verify (s)
60	4	320KiB	0.0191246s	5.01e-06s
70	4	1280KiB	0.0990458s	7.04e-06s
80	4	5120KiB	0.534211s	8.54e-06s
90	4	20480KiB	2.11532s	7.87e-06s
90	5	3200KiB	0.53583s	1.268e-05s
100	5	6400KiB	1.12532s	1.271e-05s
110	5	25600KiB	4.90879s	1.24e-05s
120	5	102400KiB	19.5753s	1.182e-05s
120	6	15360KiB	12.1717s	2.171e-05s

benchmarks intel Celeron J4115

n	k	mem	solve (s)	verify (s)
60	4	320KiB	0.0231862s	1.437e-05s
70	4	1280KiB	0.141695s	1.651e-05s
80	4	5120KiB	0.688328s	1.624e-05s
90	4	20480KiB	2.9004s	1.616e-05s
90	5	3200KiB	0.69292s	2.413e-05s
100	5	6400KiB	1.4349s	2.433e-05s
110	5	25600KiB	6.59735s	2.438e-05s
120	5	102400KiB	27.1087s	2.441e-05s
120	6	15360KiB	14.8045s	4.121e-05s

benchmarks arm64

n	k	mem	solve (s)	verify (s)
60	4	320KiB	0.0566198s	1.771e-05s
70	4	1280KiB	0.282635s	1.81e-05s
80	4	5120KiB	1.16754s	2.025e-05s
90	4	20480KiB	5.00207s	1.685e-05s
90	5	3200KiB	1.27173s	2.625e-05s
100	5	6400KiB	2.61081s	2.541e-05s
110	5	25600KiB	10.0288s	2.837e-05s
120	5	102400KiB	39.1474s	3.147e-05s
120	6	15360KiB	23.026s	4.721e-05s

benchmarks raspi 1b

n	k	mem	solve (s)	verify (s)
60	4	320KiB	0.9797s	0.00082489s
70	4	1280KiB	4.3772s	0.0008222s
80	4	5120KiB	17.0851s	0.00081515s
90	4	20480KiB	65.466s	0.00080718s
90	5	3200KiB	17.8408s	0.00148526s
100	5	6400KiB	33.4936s	0.00150453s
110	5	25600KiB	141.49s	0.00151378s
120	5	OOM		
120	6	15360KiB	351.63s	0.00282118s

libequihash

`https://github.com/stef/libequihash`

python bindinggel lehet shellscriptelni!

Peppered Account server

- facebook es google is hasznal valami ilyesmit[4]
- internal sima szimmetrikus
- external sphinx!

[4] https://vnhacker.blogspot.com/2020/09/why-you-want-to-encrypt-password-hashes.html