Passbolt: a bold use of `';--have i been pwned?`

Qui bene amat bene castigat

Philippe Teuwen

Quarkslab

Who am I? A user of Passbolt

Open Source Password Manager with sharing features for teams

Uses *Pwned Passwords*, part of `';--have i been pwned?`

About 1 500 000 000 leaked passwords,
maintained by Troy Hunt, cached by CloudFlare

"[...] it never gains enough information about a non-breached password hash to be able to breach it later."

Sounds good, right?

Pwned Password API

```
$ echo -n p@ssword | sha1sum
36e618512a68721f032470bb0891adef3362cfa9
\___/\_________________________________/
```

⇒ send 36e61

```
$ wget -q -O - https://api.pwnedpasswords.com/range/36e61
[...]
8512A68721F032470BB0891ADEF3362CFA9:21804
[...]
```

If no hit, we're fine
Nothing to learn from 2.5 bytes leak, right?

Passbolt UX

Passbolt UX

Wait, is it querying the API even for 1 char??

Sniffing API usage

Typing password “`123456789AB`” ⇒

- 1 to 7 → nothing
- 8 → API query with `7C222` (SHA1[0:5] of `12345678`)
- 9 → API query with `F7C3B` (SHA1[0:5] of `123456789`)
- A → API query with `BE472` (SHA1[0:5] of `123456789A`)
- B → API query with `4A3C4` (SHA1[0:5] of `123456789AB`)

300 ms debounce ⇒ If typing at 3 chars/s max, we get all queries

Sniffing API usage

Typing password “`123456789AB`” $\Rightarrow$

- 1 to 7 $\rightarrow$ nothing
- 8 $\rightarrow$ API query with `7C222` (SHA1[0:5] of `12345678`)
- 9 $\rightarrow$ API query with `F7C3B` (SHA1[0:5] of `123456789`)
- A $\rightarrow$ API query with `BE472` (SHA1[0:5] of `123456789A`)
- B $\rightarrow$ API query with `4A3C4` (SHA1[0:5] of `123456789AB`)

300 ms debounce $\Rightarrow$ If typing at 3 chars/s max, we get all queries

8-char from 92-char alphabet: ~ 52 bits
Learned 20 bits of leak, remain 5 billion possibilities...

Maths...

- After 8 chars,
  $\mathcal{H} = log_2(92^8) = 52.2$ bits, but $\mathcal{L} = log_2(16^5) = 20$ bits $\Rightarrow \mathcal{H} = 32.2$ bits
- After 9 chars,
  $\mathcal{H} = log_2(92^9) = 58.7$ bits, but $\mathcal{L} = 2\ log_2(16^5) = 40$ bits $\Rightarrow \mathcal{H} = 18.7$ bits
- After 10 chars,
  $\mathcal{H} = log_2(92^{10}) = 65.2$ bits, but $\mathcal{L} = 3\ log_2(16^5) = 60$ bits $\Rightarrow \mathcal{H} = 5.2$ bits
- After 11 chars,
  $\mathcal{H} = log_2(92^{11}) = 71.8$ bits, but $\mathcal{L} = 4\ log_2(16^5) = 80$ bits $\Rightarrow \mathcal{H} = 0$
  $\Rightarrow$The password can be fully recovered!

Strategy...

- Generate the 5 million of billion of 8-char candidates
- 1st hash → 1 / 1 000 000
- Extend to 9 chars
- 2nd hash → 1 / 1 000 000
- Extend to 10 chars
- 3rd hash → 1 / 1 000 000
- Extend to 11 chars
- 4th hash → fully recovered!
- etc.

PoC||GTFO: Hashcat module

- 4 partial hashes
- Assume API calls on 8th, 9th, 10th and 11th char
- Crack the 11-char password

Then extend to any length at no cost

PoC||GTFO

Worst case: 5h on a 8x RTX 4090 instance at $4/h

But we're breaking passwords chosen and typed by humans...

Worst case: 5h on a 8x RTX 4090 instance at $4/h

But we're breaking passwords chosen and typed by humans...

Much simpler case example:

`iwashere$&@!2=[#)`

⇒ 6 s on my laptop to recover `iwashere`
⇒ +33 ms for the full password

CVE-2024-33669 timeline glimpse

- $2024/03/22$ – Vuln report
- $2024/03/30$ – Chrome extension fixed
- $2024/04/03$ – Firefox extension fixed
- $2024/04/04$ – Edge extension fixed
- $2024/04/11$ – Windows application fixed
- $2024/04/17$ – Synchronized publication & ping Troy Hunt
- $2024/04/20$ – Pwned Passwords APIv3 documentation updated to include warning

Applied fix: API call only on form submission, only if $\mathcal{H}(pwd) > 60$ bits

`https://blog.quarkslab.com/passbolt-a-bold-use-of-haveibeenpwned.html`

Thank you

https://blog.quarkslab.com

Quarkslab