Mitigating Password Compromise with Honey Passwords

1

1

Decoys

- Decoys are fake objects that look real, are a time-honored counterintelligence tool.
- In computer security, we have "honey objects":
  - Honeypots
  - Honey-tokens, honey accounts
  - Decoy documents
- Honey objects seem undervalued.

2

2

Key questions today

- How to apply honey objects to pressing computer security / privacy problems?
  - Password breaches in the cloud
  - Compromise of mobile device data
- How to use honey objects in a *principled* manner?
  - Can we progress from the art of deception to science of deception?

Larger research vision: How to deal with powerful adversaries who will sooner or later compromise systems?

3

3

HONEYWORDS: MAKING PASSWORD CRACKING DETECTABLE

Ari Juels (RSA) and Ron Rivest (MIT), ACM CCS'13

4

4

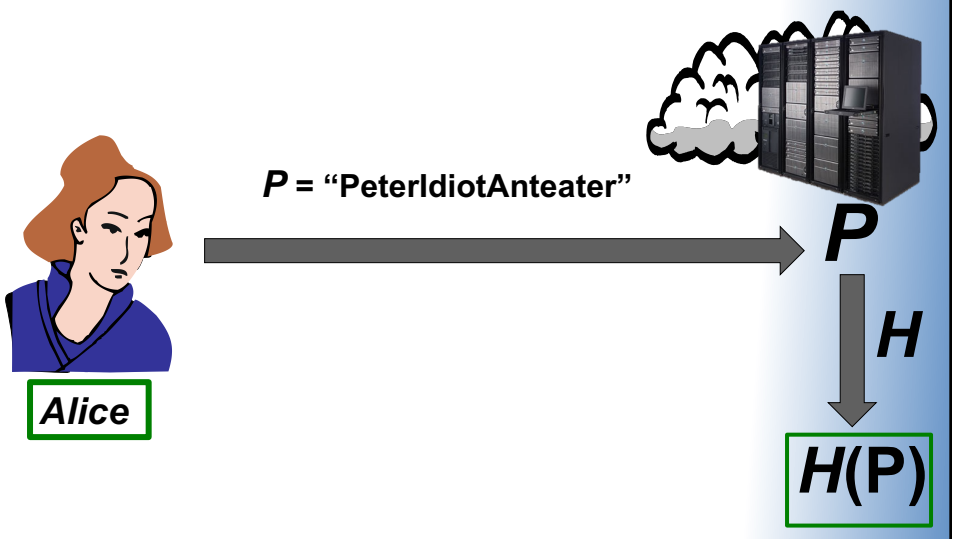
Good news and bad about password breaches

- The good news: when giving a talk about password (or PII) breaches, convenient examples crop up.
  - October 2013: Adobe lost 130 million ECB-encrypted passwords

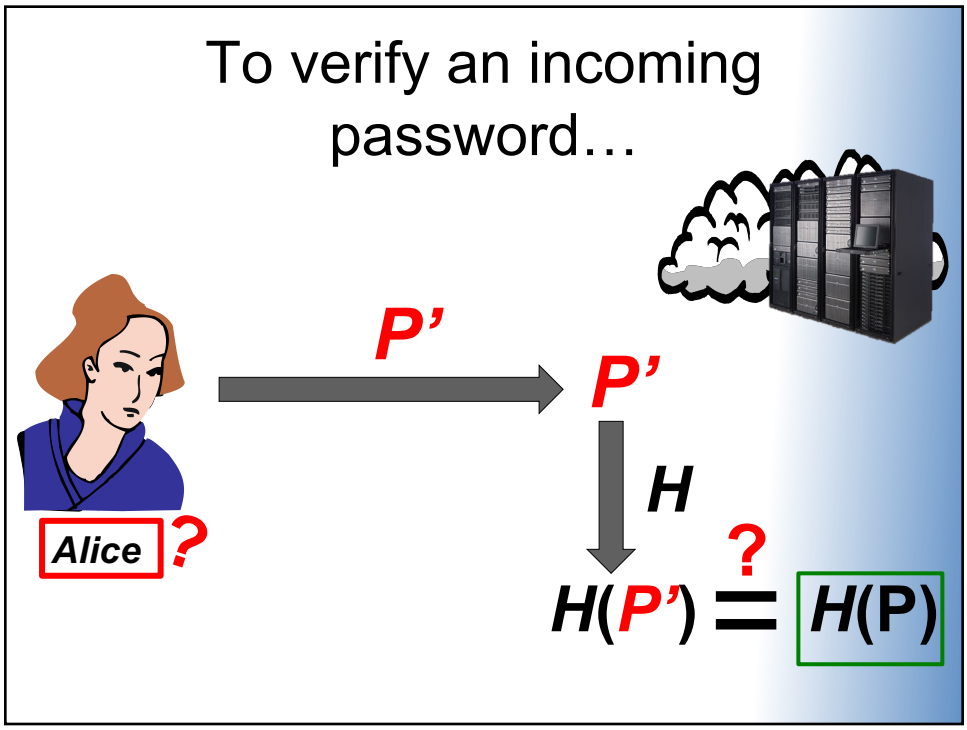
- The bad news: This is all **bad news**.

5

Passwords are protected via *hashing*



6



7

Password hashing

- Hashing (plus salting) forces attacker that learns hashes to determine passwords by brute-force (offline) guessing
- Brute-force guessing means: attacker guesses *P'* and checks if *H*(*P'*) = *H*(*P*)
- Additionally, hashing can be hardened (slowed) in various ways (e.g., bcrypt)
- This all seems good, but…

8

Password hashing

- Real passwords are weak and easily guessed.
  - Nice study of 69+ million Yahoo! passwords shows that:
    - w = 1.08% of users had the same password
    - About 50% of passwords at best equivalent to 22-bit strength keys
      - $\mu_{1/2} \approx 2^{22}$
- Good password-hash crackers now model real users' password selection.
  - E.g., [WAdMG09] (probabilistic context-free grammar)
  - Crackers leverage, e.g., RockYou 2009 database of 32 million passwords
- Even good (& salted) hashes are often inadequate.
- Let's assume that salted hashes can be cracked and passwords are effectively in the clear.

9

Adversarial game

<u>"Smash-and-grab" attack</u>

- adversary compromises the system ephemerally (usually passively)
- Steals a copy of password file
- Impersonates user(s)

Alice:

P

10

Adversary always wins

Alice:

P

"Alice", P

11

Honeywords

Alice:

P_1

P_2

…

P_n

12

Honeywords

Alice:

P_1
P_2
…
$P_i = P$
…
P_n

True password

13

Honeywords

Alice:

P_1
P_2
…
$P_i = P$
…
P_n

Honeywords (decoys)

14

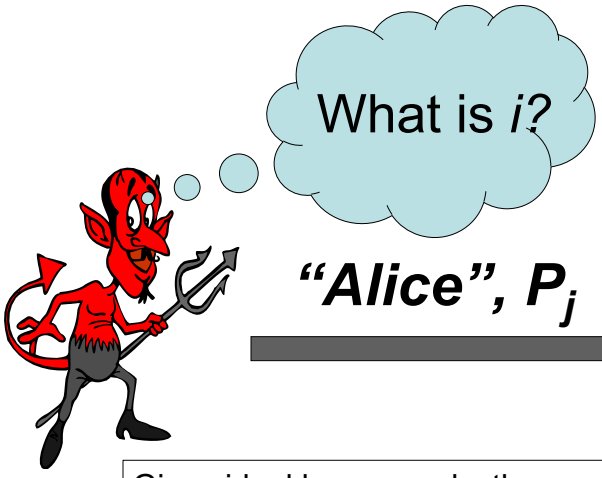
Honeywords

Alice:

Sweetwords $\left\{ \begin{array}{l} P_1 \\ P_2 \\ \ldots \\ P_n \end{array} \right.$

15

The adversarial game



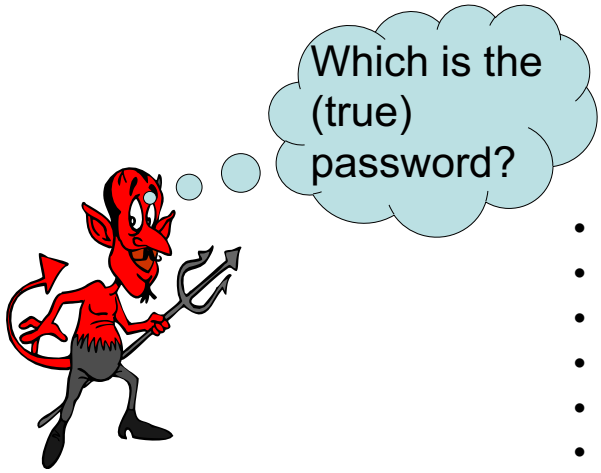
Alice:

P_1
P_2
…
P_n

Given ideal honeywords, the attacker will guess correctly, $j = i$, with probability about $1/n$!

16

The adversarial game



Alice:

- `5512lockerno.`
- `tribal_3`
- `cshcsh.meowr.18`
- `28/07/89rm`
- `anto_2001_jesu`
- `CRFRALAASS$4`
- `!v0nn3`
- `ponk.m4t`

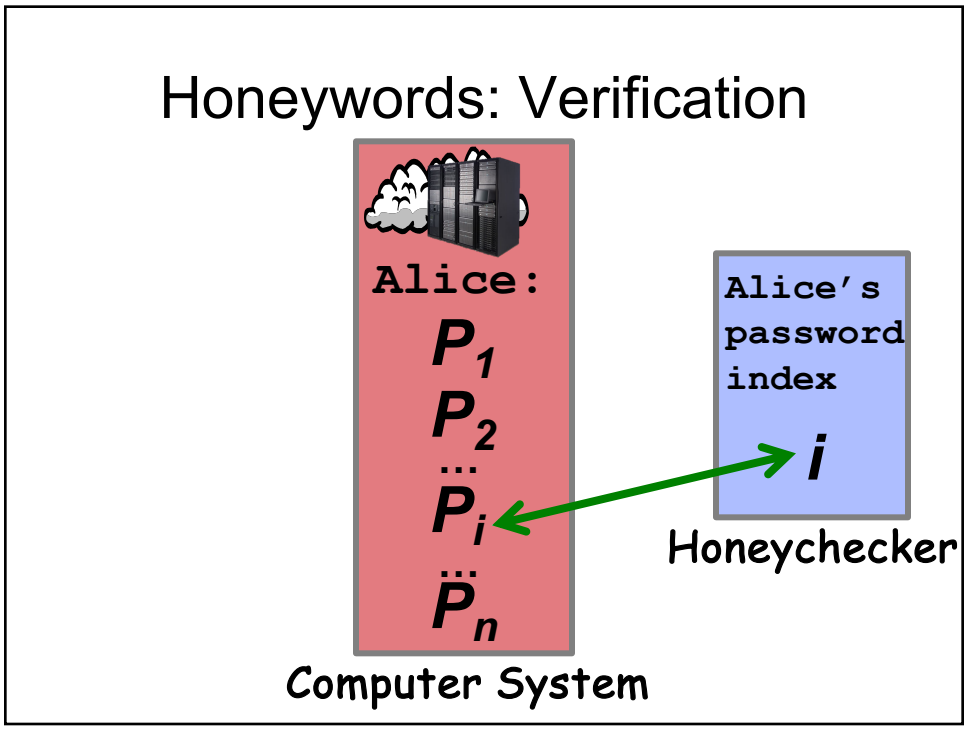
17

Design questions

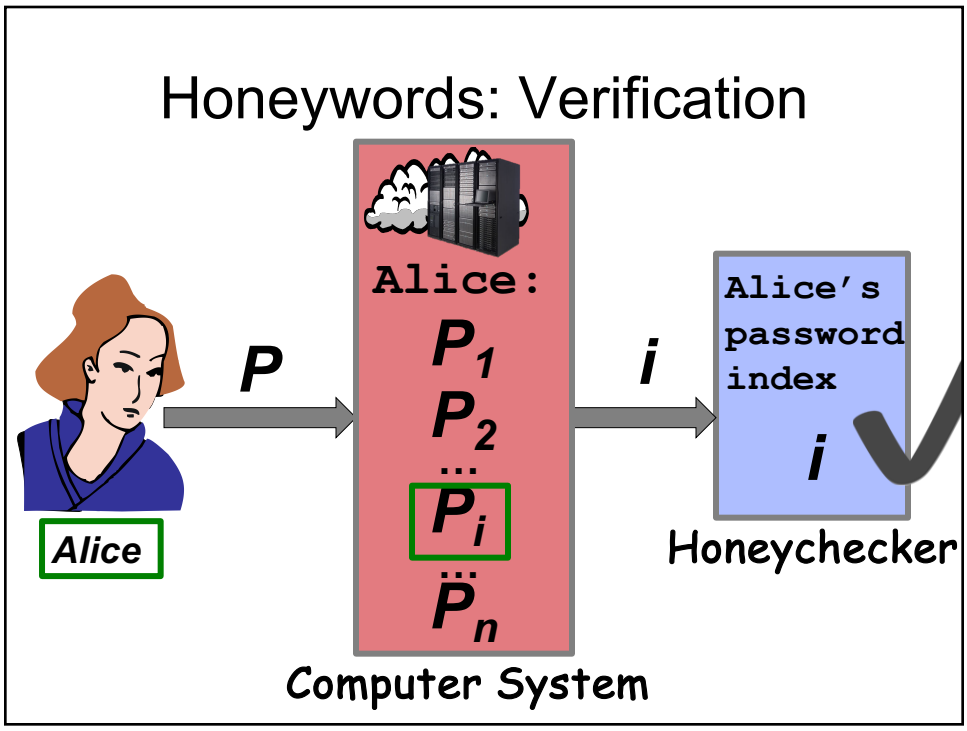
1. **Verification:** How does the check whether a *submitted* password *P'* is the *true* password P_i?
   - How is index *i* verified without storing *i* alongside passwords?
2. **Generation:** How are honeywords generated?
   - How do we make bogus passwords look real?

(Many other questions, e.g., how to respond when breach is detected using honeywords…)

18

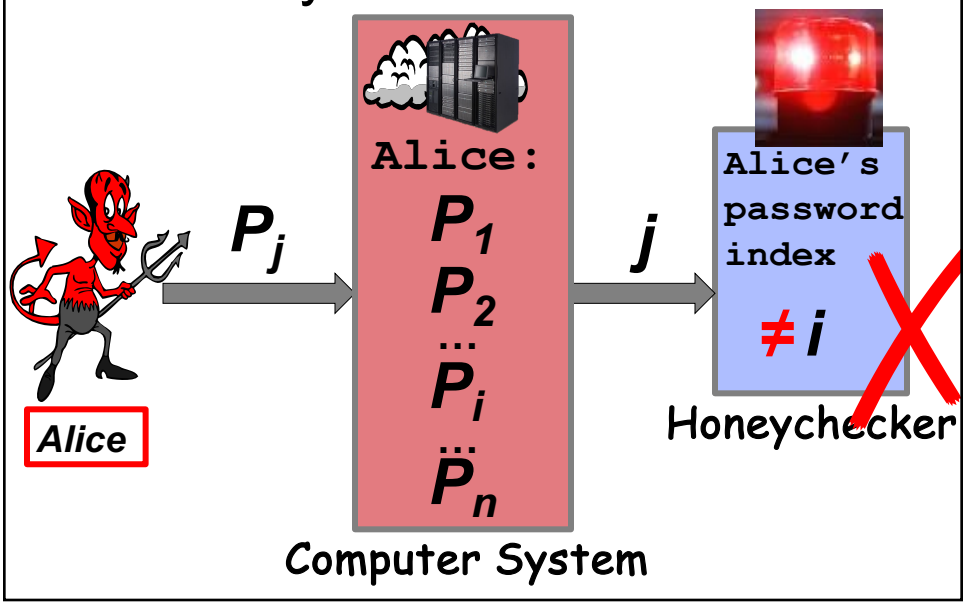

Honeywords: Verification
Alice:
P_1
P_2
...
P_i
...
P_n
Computer System
Alice's password index
i
Honeychecker

19


Honeywords: Verification
Alice
P
Alice:
P_1
P_2
...
P_i
...
P_n
Computer System
i
Alice's password index
i
Honeychecker

20

Honeywords: Verification



21

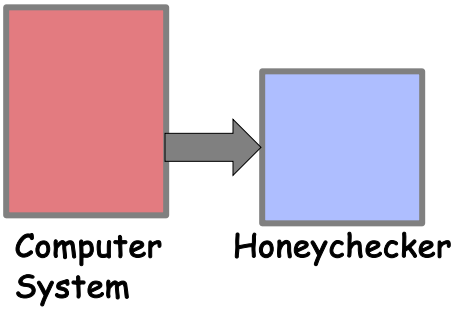
Honeywords: Verification Rule

- If the true *password* P_i is submitted, the user is authenticated.
- If a password $P' \notin \{P_1 \ldots P_n\}$ is submitted, it's treated as a normal password authentication failure.
- If a *honeyword* $P_j \neq P_i$ is submitted, an alarm is raised by the honeychecker.
  - This is likely to happen only after a breach!
    - Only if P_j-s are all non-trivial?
  - Honeywords (if properly chosen) will rarely be submitted otherwise.
- **Note: No change in the user experience!**

22

Some nice features of this design

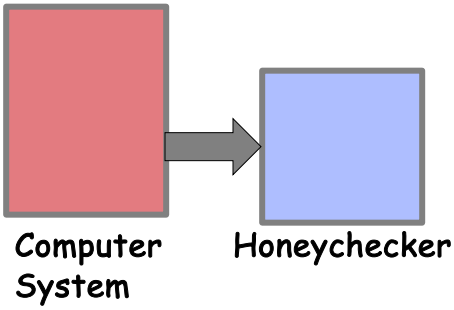
- Computer system does nothing but transmit sweetword index *j*
  - *Little modification needed*
- We get the benefits of distributed security
  - Compromise of either component isn't fatal
  - No single point of compromise
  - Compromise of both brings us back to hashed case
- Honeychecker can be minimalist, (nearly) *input-only*
  - Only (rare) output is alarm



23

Some nice features of this design

- Honeychecker can be offline
  - E.g., honeychecker sits downstream in security operations center (SOC)
  - Not active in authentication itself, but gives rapid alert in case of breach
  - If honeychecker goes down, users can still authenticate (security no worse than without it)



24

Honeyword generation

Which is Alice's real password?

Alice:

- QrMdmkQt
- AP9LXEEa
- m7xnQVV4
- kingeloi
- y5BJKWhA

25

Honeyword generation: Chaffing with a password model

- Password-hash crackers learn model from lexicon of breached passwords (e.g., RockYou database)
  - Make guesses from model probability distribution
- Idea: Repurpose cracker as generator!
- Works assuming user's real pw "fits" the lexicon

Alice:

- qivole
- paloma
- 123asdf
- Compaq
- asdfway

26

But there are problem cases…

Which is Alice's real password?

Alice:

- hi4allaspls
- #Down-with-Elon-Musk
- Travis46
- #1bruinn
- KJGS^!*ss

27

Honeyword generation: Chaffing by tweaking

- [ZMR10] observed users tweak passwords during reset (e.g., HardPassword1, HardPassword2, …)
  - Proposed tweak-based cracker
- Idea: Tweak password to generate honeywords!
- E.g., tweak numbers in true password…

Alice:

- yamahapacificer3214567898765432 1
- yamahapacificer12345678987654321
- yamahapacificer12345678901234567
- yamahapacificer62145678987654322

28

Honeyword generation: A research challenge

- Blink-182 is a rock band
- This password is *semantically significant*
  - Tweaking would break it
  - Generation is unlikely to yield it
- Dealing with such passwords is a special challenge—relates to natural language processing
- Possible cure:
  - use *other people's passwords* as honeywords… How?

Alice:

- `Blink123`
- `Graph128`
- `Froggy%71`
- `Blink182`
- `Froggy!83`

29

The larger landscape

- Honeywords are a kind of poor-man's distributed security system
- Honeywords strike an attractive balance between ease of deployment and security
  - Little modification to computer system
  - Honeychecker is minimalist
  - Conceptually simple

30

Follow-up Questions/Issues

- Can honey password scheme be extended to raise an alarm when Alice is under duress (e.g., gun to the head)
  - Supplies Pj where j tells the honey-checker that Alice is in trouble

- What if the Adversary's goal is to cause havoc?
  - Learns password file, breaks (brute-forces) all users' passwords.
  - For each user X, attempts to log in using P^x_k – one of X's honey passwords
  - Everyone is locked out, must change all passwords ASAP → chaos
  - Excellent Denial-of-Service attack!

31

31