ENEE 457: Computer Systems Security

Lecture 13
Password Authentication and Rainbow Tables

Charalampos (Babis) Papamanthou

Department of Electrical and Computer Engineering
University of Maryland, College Park

Passwords and Authentication

- Passwords are not stored in the clear but as
  - username, HASH(password) (HASH is one-way)
- How can you invert a HASH(password) where the password has *n* bits (N=2^n)?

TIME	SPACE
O(1)	O(N)
O(N)	O(1)

- What if you chose from a dictionary?

TIME	SPACE
O(1)	O(|dictionary|)
O(|dictionary|)	O(1)

Can we do better?

- Rainbow tables

TIME	SPACE
O(N^{2/3})	O(N^{2/3})

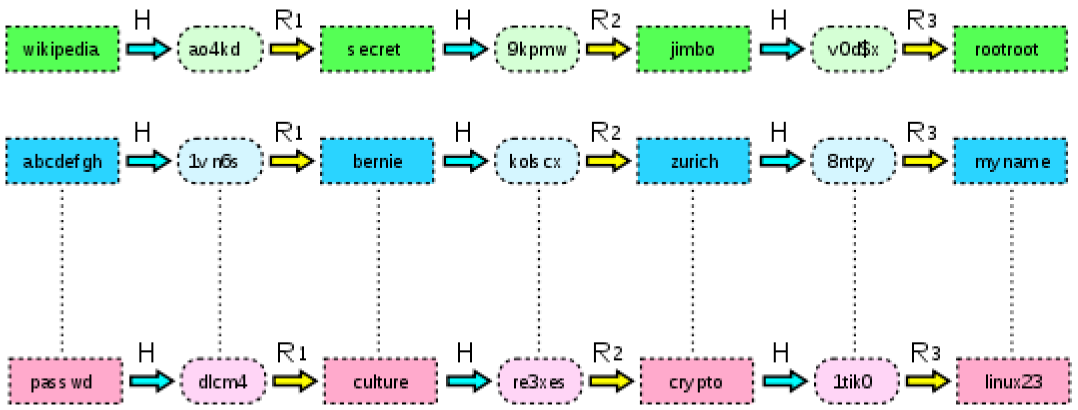
- Try it at http://project-rainbowcrack.com/

Basic Idea

- Assume h is a one-way hash function mapping n bits to n bits (password to hashes or hashes to passwords), where $N = 2^n$
- Assume h cycles through all the values of the domain: Namely if you begin with a password p, then applying h(.) N times will cycle through all the possible values of $\{0,1\}^n$. How you can use this fact to crack a password using sqrt(N) space in sqrt(N) time?
- Answer:
- Start with an arbitrary password p, compute $h_1, h_2, \ldots h_N$ and store
  - h_1 $h_{\sqrt{N}}$
  - $h_{\sqrt{N}+1}$ $h_{2\sqrt{N}}$
  - …
  - $h_{(\sqrt{N}-1)\sqrt{N}}$ h_{N}
- Store these in a hash table
- Now given a hash to break h_i, start computing $h(\ldots h(h(h_i))\ldots)$ and you are guaranteed to hit an endpoint of the above. Get the starting point and start developing the chain until you hit the password.
- Clearly this requires sqrt(N) time and sqrt(N) space

Rainbow tables

- Having a hash function that cycles is a big assumption. Hash functions typically have collisions.
- For that we need rainbow tables
- Pick m passwords p1 p2 … pm and start developing chains, each one having t elements. Store the start points and the endpoints. Then given a hash h, start developing chains until you hit an endpoint and then go the start point to retrieve the password



Hash function is not enough…

- Let D be the domain of the passwords and H be the domain of the hash fuction
- h: D → H
- r: H → D
- Before:
  - p → h(p)→h(h(p))→…→h(h(…h(p)…))
- Now
  - p → h(p)→ r(h(p)) → h(r(h(p))) → r(h(r(h(p))))…
  - Example reduction function: If your password is 16 bits and the hash is 256 bits, keep 16 equally distributed bits from the 256 bits

Problem 1

- You might not hit an endpoint after t evaluations. This can be the case if the hash you started with is not in the collection of the values that were generated

Problem 2

- Even if you do, it might be the case that there is a collision. Namely hashing many times H(sp_i) will never give you the hash you started with, so you cannot retrieve the password

Any theoretical guarantees?

- Hellman (Turing award winner, 2016) proved that if $mt^2 = N$, then the probability of retrieving the password using the above approach is
  - $\mathbf{mt/(2N)=1/(4t)}$
- Typical setting of the parameters: $m=N^{1/3}$, $t=N^{1/3}$ (non-constant probability)
- Can we increase this probability?
- Generate 4t independent tables
- Then the probability that the given password is covered is
  - $1 – (1-1/4t)^{4t} \sim = 1-e^{-1}=0.63$

How to generate independent tables?

- For each one of 4t tables, pick a random reduction function r_i
- Before:
  - p → h(p)→r(h(p))→…→h(r(…h(p)…))
- Now (for table i)
  - p → h(p)→r_i(h(p))→…→h(r_i(…h(p)…))

Complexities?

- Space: $O(mt)=O(N^{2/3})$
- Time: Search every table separately, so $O(t*t)=O(N^{2/3})$
- How can we keep the same probability, while reducing the time to search?
- Use mt chains of t size each but use a different reduction function per step
- So in the worst case you do space $t+(t-1)+(t-2)+\ldots 1 = t(t-1)/2$

Countermeasures

- Use salting
- Store h(r||p), r, instead of just h(p)
- Then the password space becomes too big to be stored.

Question

- What if you store passwords as h(p||r), where r is randomness of 128 bits?
- Will the O(N) space solution work?
- No, because the data structure is built on the password space, and not on the randomness+password space (that would require too much space!)