Password Selection and Alternatives

CMSC 426 - Computer Security

1

Outline

- User password selection and rules
- Bloom Filters
- Token-based Authentication

2

Password Rules

- Users pick terrible passwords - just look at the beginning of the RockYou list:

  `123456`, `12345`, `123456789`, `password`, `iloveyou`, `princess`, `1234567`, `rockyou`, `12345678`, `abc123`, **etc.**

- Primary defenses are education and enforcement of password rules.

3

Other Options

- *Computer generated passwords* or pass-phrases *can* be done well but are often unpopular with users.

  Personal experience: pass-phrases are better.

- *Reactive password checking* - try to crack users' passwords on your system.
- *Proactive password checking* - check password at the time the user selects it.

4

Proactive Checking

- *Rule-based* - check length, proper mix of character classes, etc.

  Better than nothing, but annoying for users.

- *Dictionary-based* - do not allow passwords from a dictionary of "bad" passwords.

  Need a **big** dictionary, and it is slow.

- *Bloom Filters* - clever technique...

5

Bloom Filters

- Need k independent hash functions $H_i(x)$.
- Each $H_i(x)$ takes values in {0, 1, ..., N-1}.
- Need N-bit table $T = (b_0, b_1, ..., b_{N-1})$.
- For each word in dictionary of "bad" passwords, compute the k hashes and set the corresponding bits in T.

6

Example

- $H_0(x)$ = first nibble of MD5 hash of x.
- $H_1(x)$ = second nibble of MD5 hash of x.
- $H_2(x)$ = third nibble of MD5 hash of x.
- $H_3(x)$ = fourth nibble of MD5 hash of x.
- T starts as all zeros

7

- MD5 of `123456` is `f447...` so set bits 15, 4, and 7 in T.
- MD5 of `12345` is `d577...` so set bits 13, 5, and 7 in T.
- MD5 of `123456789` is `b2cf...` so set bits 11, 2, 12, and 15 in T.

T = (0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1)

8

- User selects new password *x*.
- System computes $H_0(x)$, $H_1(x)$, $H_2(x)$, $H_3(x)$ and checks corresponding bits in *T*.
- If **all** of the bits are set (1), then reject the password *x*.
- Guaranteed that `123456`, `12345` and `123456789` will be rejected.

- Continuing with the example...

  T = (0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1)

  - Suppose user selects password “blargh”. MD5 hash is `5f71...`
  - Check bits 5, 15, 7, and 1 in *T*
  - 5, 15, and 7 are set, but 1 **is not set**, so we accept the password.

Real Parameters

- *k* in the range 2 - 6 is reasonable.
- *N* is large, an order of magnitude times larger than the dictionary size.
- *False Positive* - reject a password that is **not** in the dictionary. Want to minimize these!

False Positive Rate

- Let *R* be ratio of *N* to the dictionary size *D*, that is $R = N / D$.
- Let *p* be the probability of a false positive.

  $$p = (1 - e^{-k/R})^k$$

  which gives

  $$R = -k / \ln(1 - p^{1/k})$$

Example

- Suppose I have a dictionary of 1,000,000 words and want to implement a Bloom Filter with $k = 6$ and false positive probability of $p = .001$. What should N be?

$$R = -6 / \ln(1 - .001^{1/6}) = 15.78406$$

so N needs to be 15,784,060, or approximately 16 million bits.

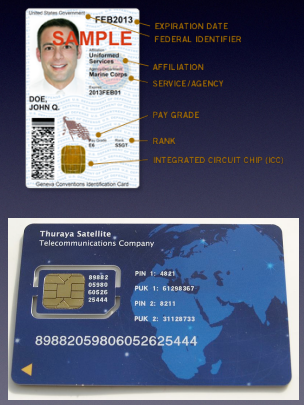
Token-based Authentication

Something you have...

- A token is a physical device that is used as part of the user authentication process.
- User must be in possession of token to be authenticated to the system.
- We'll look at two types of token: smart cards and one-time password generators.

Smart Cards

- If your mobile phone has a SIM, that is a smart card.
- Smart cards contain full-fledged processors with CPU, memory, and I/O.



Images are Public Domain

FIPS 201

- FIPS 201, *Personal Identity Verification*, defines the ways in witch a smart ID card can be used to verify identity.
- We only care about Authentication Using Asymmetric Cryptography.

17

User enters PIN to activate card

Card Reader	Smart Card
Has access to public keys for the Certification Authority	Stores Private Key, Public Key, and Certificate (cert)

Sends challenge to smart card → Signs challenge with private key

Verifies signature and validates cert ← Returns signed challenge and cert to reader

18

- Once the reader has verified the signature and validated the certificate, it extracts the user identity from the cert and forwards it to the authorization service.
- The public key algorithm would typically be RSA with a 2048-bit modulus.

19

One-Time Passwords

- Eample: RSA SecurId
- Device generates a passcode every minute
- Server knows how to generate code to verify user's input
- May be used in conjunction with usual id and password

Image is Public Domain

20

Types of OTP

- RSA SecurId is a *Time-Based OTP* system since the creation of the passcode is based on time (well, there's also a secret key...)
- *HMAC-Based OTP* (HOTP) defined in RFC 4226.

  Uses a counter synchronized between the client and server. OTP derived from HMAC of the counter.

21

- *Password-Based OTP* (my term) defined in RFC 2289.

  User receives random seed from server, hashes this along with password *N* times, saving hashes. OTP derived from hashes *used in reverse order*, i.e. use the N^{th} hash, next time *N*-1^{st}, etc.
- **Why reverse order?**

22

Finished. See the website for exercises.

23