

Lecture Notes – Computer Security

Author: Dr. Aaron Striegel
University of Notre Dame striegel@nd.edu

Topic: Authentication

Passwords Something you know

Having the correct password confirms the subject's identity

Password space: Set of all sequences of characters

Problem: Need to validate the password but do not want the user to see the cleartext password

Solution: 1-way hash

UNIX Password

```
dmmcnally:EhYpHWagUoVhM:0:1:DOGBERT:~/bin/false
```

Contents

User Name	dmmcnally
Encrypted Password	
User ID	0
Group Number	1
Home directory	DOGBERT
Shell	/bin/false

Up to 8 ASCII letters, anything beyond 8 is ignored

Actual Password	->	Hashed Password	11 letters +
		Salt	2 letters (what to use)

Salt = 4,096 combinations of functions

E table in DES is perturbed one of 4096 possible ways\

Brute force impact:

A	Set of authentication info	6.9 x 10 ¹⁶
C	Set of complementary info (13 chars, alphabet of 64)	3.0 x 10 ²³

Store C in either /etc/passwd or /etc/shadow

/etc/passwd	Readable by all	more /etc/passwd
/etc/shadow	Readable only by superuser	

Try it on darwin.helios.nd.edu

Protecting Password Info

1. Hide enough such that a, c, or f cannot be found
 /etc/shadow Cannot see encrypted password or salt
2. Prevent access to functions
 No root access from the network, only from the console

Attacking a Password System

Dictionary attack Guessing a password by trial and error

What did we learn from the homework?

 Must avoid easy guesses

Other thoughts

 How do you attack in the first place?

 Must have access and get user names

 Mine from web pages :)

 Use well-known accounts

 root, guest, etc.

 Always double check well-known account points

 Set usernames that you do not want people logging into
 with a shell of /bin/false or nobody permissions

 More info: Recent changes to OpenSSH

 Privilege separation

Thwarting Guessing

 How strong?

 Anderson's formula generalized

 P = Probability that an attacker guesses a password in a unit of time

 T = Number of time units one can guess

 G = Number of guesses that can be tested in a single time unit

 N = Number of possible passwords

$$P \geq (TG) / N$$

 Why is this \geq ?

 This is the worst case (i.e. attacker is only brute forcing). The
 attacker can be much smarter.

Example

4 digit ATM PIN $10 * 10 * 10 * 10 = 10,000$ combos

Time unit = 1 hour

T = 24

24 hours

G = 60

60 guesses per hour

N = 10,000

$P \geq 0.144$

14.4%

What about a computer testing the same?

If we can test 5x more guesses (easy), will crack the code

Random Password

T is the time that someone can guess the password

Choose a regeneration time to keep P sufficiently low

Randomly generate new passwords

One password = OK

Multiple random passwords = problem

Slight alternative

Write down a transformed (encrypted version of password)

Ex: Capitalize the third letter, append the letter 2 and shift all by 3

Pronounceable Passwords

Phoneme

helgoret vs. przbqxdf

Downside: Brute force space (potential key space) gets reduced

User selectable passwords

Bottom line:

People are bad

Low hanging fruit:

Account name + number

Account name + delimiters

User names (initials)

All lower or upper case

Reverse of name

1st Initial + last name reversed

Computer names

Dictionary words

Friends/family/etc.

...

Proactive Password Checker

Any time a password is changed, verify against easy cracking

Password Aging

Force replacement of password after X time

Challenge-Response

Passwords -> re-usable

Potentially vulnerable to replay

Challenge/Response Force uniqueness (think nonce)

U wants to log in to S

Have an agreed upon secret function

S sends a random message m to U

U transforms the message using the secret function

U replies with the transformation $r = f(m)$

S validates r by computing it separately

Is there a problem if the channel is not secure?

Similar to IFF (identification of friend or foe)

Can also combine with traditional password scheme

One-Time Password

Similar to One-Time Pad

Use once, discard, and move onto next

Example: RSA SecureID

Small device, number changes every 60 seconds

Number from 0 to $10^N - 1$

Server knows about device and what number it should be displaying

Must send re-usable password + device info

Biometrics

ID by physical characteristics

Something you are

Fingerprints

Capacitive techniques to detect whorls on finger

Convert to graph with ridges represented by vertices

Graph matching problem

Voice

Train on fixed phrases

Eyes

Iris or retina scan

Iris – camera

Retina – blood vessels on back of eye Intrusive

Face

Look for locations of items eyes, nose, mouth

Place face in pre-determined position

Hand Geometry

Combinations

Can combine multiple methods Dr. Bowyer & Dr. Flynn

Words of caution

Biometrics are not perfect Inaccuracy

Very hard problem to do without fixed conditions

Relies on medium / other methods being tamperproof

Think man-in-the-middle after the fingerprint is taken

What happens if biometric info gets hacked?

Can you change who you are?

Uses

London Considerable use for facial recognition

Face scanning at airports Not really ready

International airports have feature based on iris

Get registered, avoid waiting in line

Physical

Think CIA / NSA / DoD

Don't let people log in remotely or use GPS to verify

PAM – Pluggable Authentication Mechanism

auth sufficient pam_afs.so try_first_pass ignore_root

auth required pam_stack.so service=system-auth

auth required pam_nologin.so

account required pam_stack.so service=system-auth

password required pam_stack.so service=system-auth

session required pam_stack.so service=system-auth

session required pam_limits.so

session optional pam_console.so