Fundamentals of Information & Network Security
ECE 471/571

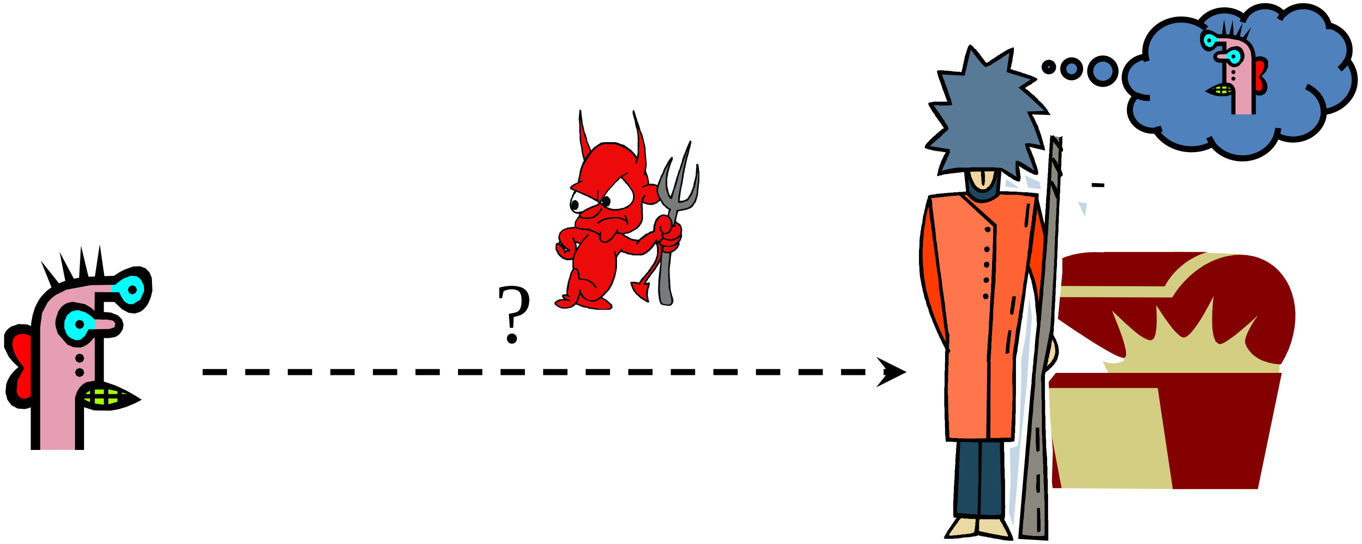
Lecture #29: User Authentication & Passwords

Instructor: Ming Li

Dept of Electrical and Computer Engineering

University of Arizona

Basic Problem



How do you prove to someone that you are who you claim to be?

Any system with access control must solve this problem

Authentication

- Binding of an identity to a subject (e.g., a person or a machine)
- Proof of information
  - What you know
  - What you have
  - What you are
  - Where you are
- Who is being authenticated?
  - Authenticate a person to a server
  - Authenticate a machine to a machine
  - Authenticate both a person and a machine to a server
  - A machine stores high-quality secret; a person memorizes low-quality password

Authentication of People

- User authentication: a computer verifies that you are who you claim to be
- Main techniques
  - What you know: password, SSN, Birth date
  - What you have: physical keys credit cards, smart card
  - What you are: biometrics

Password-Based Authentication

- *What you know*
- User has a secret password.
  System checks it to authenticate the user.
- Password in plaintext is sent over for authentication.
- Problems
    - Eavesdropping
    - Database reading
    - Password guessing: on-line, off-line (dictionary attack)
- telnet, ftp

Storing User Password

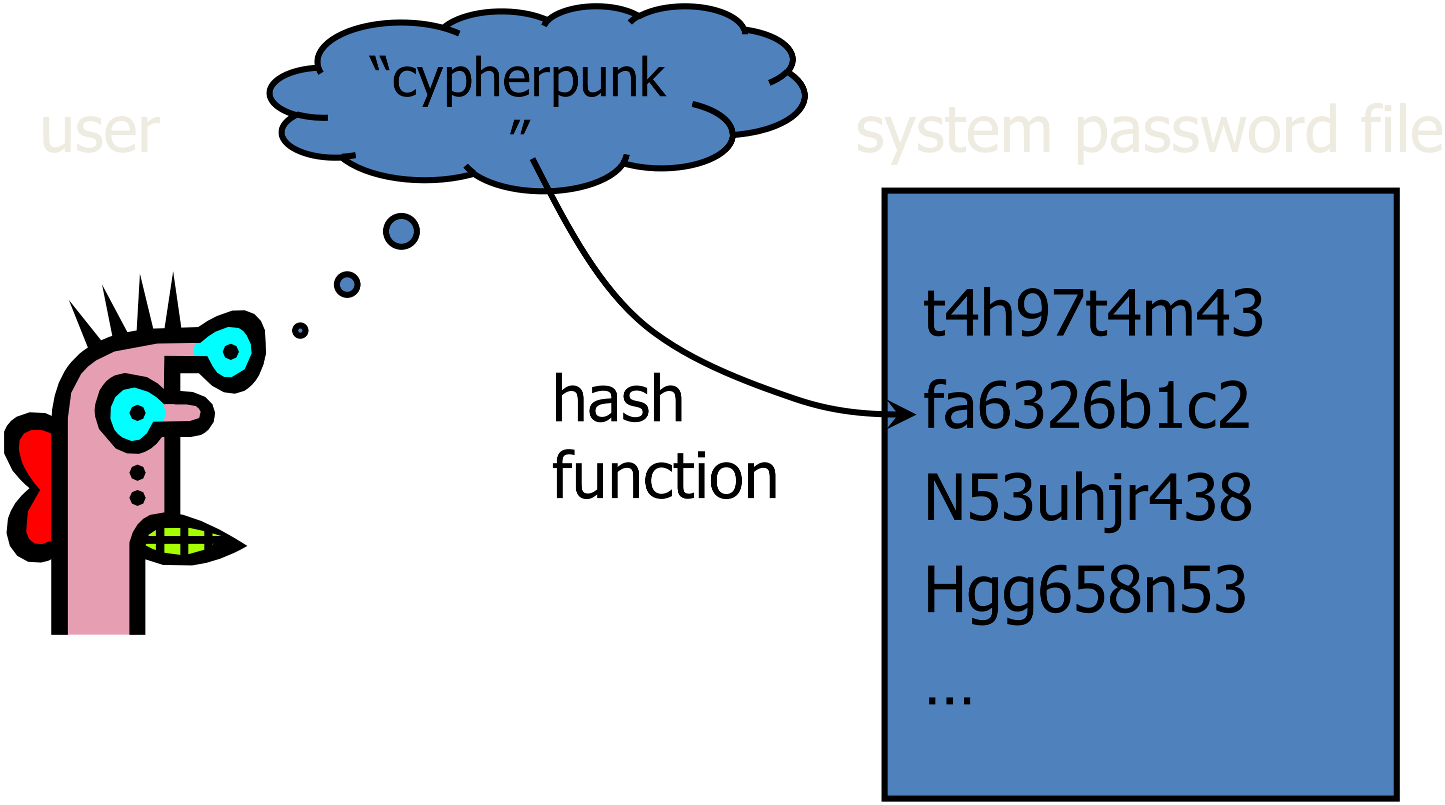
Where to store

- Store passwords individually on each server
- Store all passwords at authentication storage node
  - Retrieve password information from storage node; authentication is done at server node
- Store all passwords at authentication facilitator node
  - Send the login info received to facilitator node; authentication is done at facilitator node; result is sent back to server node.

In what format

- Store plain passwords
- Store hashes of passwords
  - Password guessing
- Store encrypted passwords
  - Involves high-quality key, cannot do password guessing; vulnerable to insider attack.

UNIX-Style Passwords



user

“cypherpunk
”

hash
function

system password file

t4h97t4m43
fa6326b1c2
N53uhjr438
Hgg658n53
…

Password Hashing

- Instead of user password, store H(password)
- When user enters password, compute its hash and compare with entry in password file
  - System does not store actual passwords!
- Hash function H must have some properties
  - One-way: given H(password), hard to find password
    - No known algorithm better than trial and error
  - Collision-resistant: given H(password1), hard to find password2 such that H(password1)=H(password2)

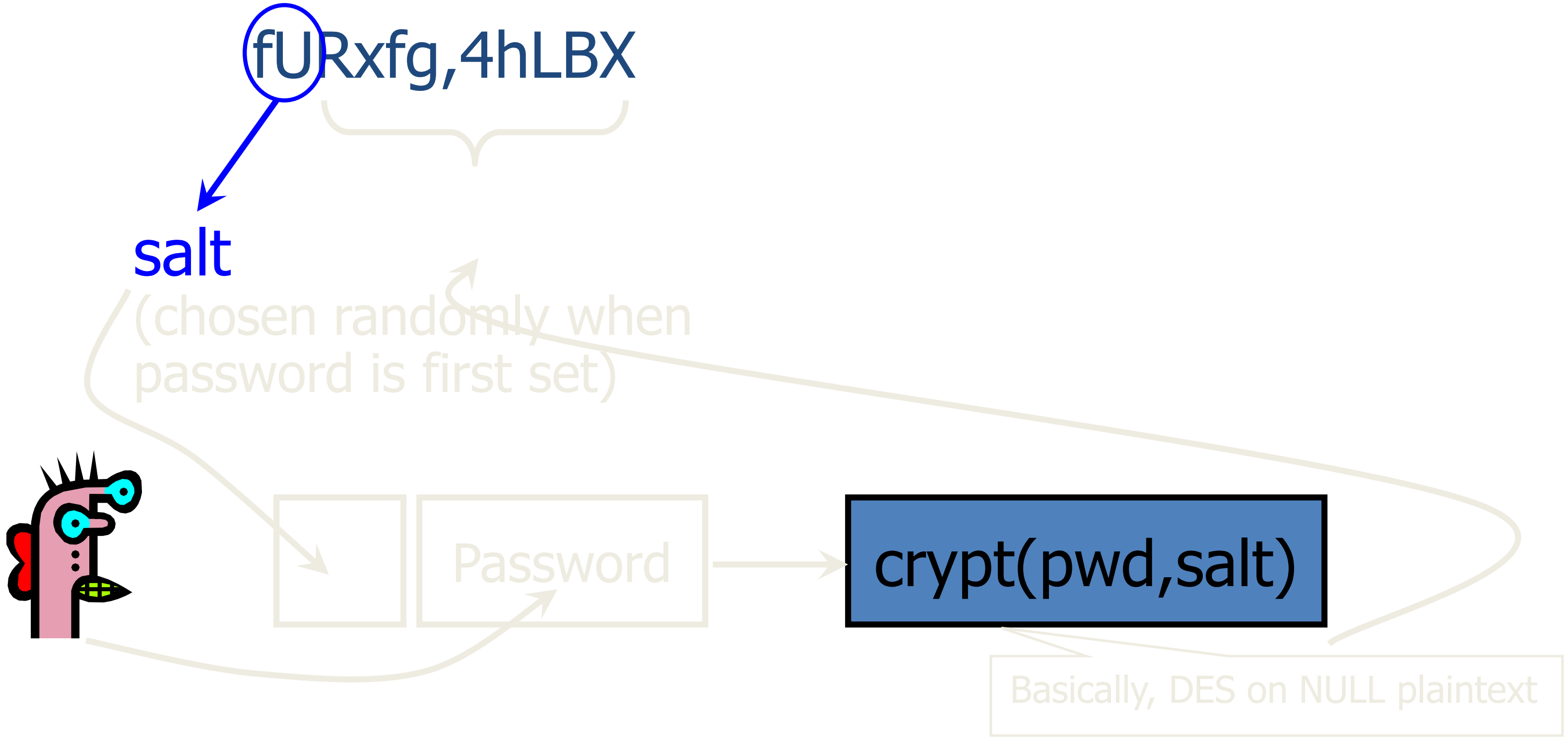
UNIX Password System

- Problem: passwords are not truly random
  - With 52 upper- and lower-case letters, 10 digits and 32 punctuation symbols, there are $94^8 \approx$ 6 quadrillion possible 8-character passwords
- Uses modified DES as if it were a hash function
  - Encrypt NULL string using password as the key
    - Truncates passwords to 8 characters!
  - Artificial slowdown: run DES 25 times
  - Humans like to use dictionary words, human and pet names ≈ 1 million common passwords

Dictionary Attack

- Password file /etc/passwd is world-readable
- Contains user IDs and group IDs which are used by many system programs
- Dictionary attack is possible because many passwords come from a small dictionary
  - Attacker can compute H(word) for every word in the dictionary and see if the result is in the password file
  - With 1,000,000-word dictionary and assuming 10 guesses per second, brute-force online attack takes 50,000 seconds (14 hours) on average
    - This is very conservative. Offline attack is much faster!

Salt



fURxfg,4hLBX

salt

(chosen randomly when password is first set)

Password

crypt(pwd,salt)

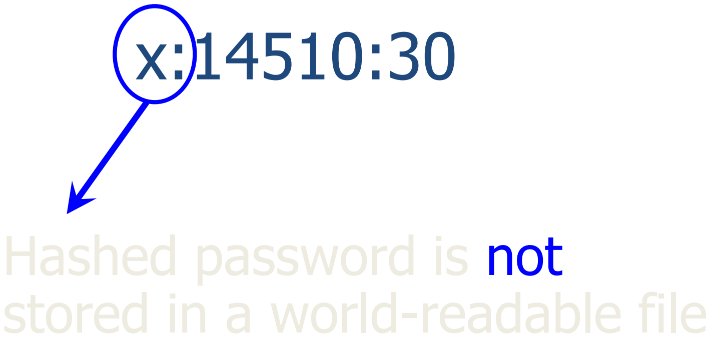
Basically, DES on NULL plaintext

- Users with the same password have <u>different</u> entries in the password file
- Dictionary attack is still possible!

Advantages of Salting

- Without salt, attacker can pre-compute hashes of all dictionary words once for all password entries
  - Same hash function on all UNIX machines
  - Identical passwords hash to identical values; one table of hash values can be used for all password files
- With salt, attacker must compute hashes of all dictionary words once for each password entry
  - With 12-bit random salt, same password can hash to 2^{12} different hash values
  - Attacker must try all dictionary words for each salt value in the password file

Shadow Passwords



x:14510:30

Hashed password is not stored in a world-readable file

- Store hashed passwords in /etc/shadow file which is only readable by system administrator (root)
- Add expiration dates for passwords