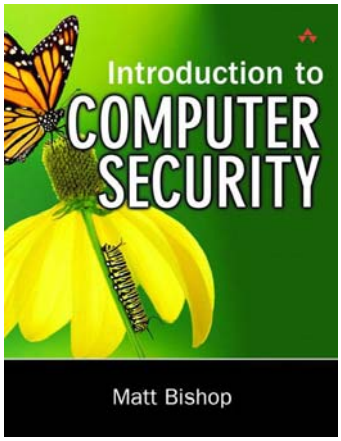
These Slides are prepared from
Matt Bishop slides and book "Introduction to Computer Security"
Benefiting from the Slides posted by Ahmad Al-Mulhem

Authentication

Chapter 11



Adnan Gutub
gutub@kfupm.edu.sa

Computer Engineering Department
King Fahd University of Petroleum & Minerals
Dhahran, Saudi Arabia

Chapter 11: Authentication

- Basics
- Passwords
- Challenge-Response
- Biometrics
- Location
- Multiple Methods

Basics

Authentication: binding of identity to subject

- Identity is that of external entity (my identity, Matt, *etc*.)
- Subject is computer entity (process, *etc*.)
  - Authentication is establishing or confirming something (or someone) is authentic (real or genuine).

Types

- Authentication in communication
  - End user ! End user (Alice and Bob)
  - Computer ! Computer (database)
- Authentication to a single system
  - End user ! Local computer (login)
  - End user ! Remote computer (web login)

Establishing Identity

Confirming a User Identity by one or more of the following:

- What entity knows: Information he knows
  - (*eg.* Password, PIN, mother's name)
- What entity has: Something he has
  - (*eg.* ID, Passport, smart card)
- What entity is: Something he is
  - (*eg.* fingerprints, voice, face)
- Where entity is: Where he is
  - (*eg*. In front of a particular terminal)

Passwords

Passwords is a type of authentication techniques based on what people ***knows***

Sequence of characters
- Examples: 10 digits, a string of letters, *etc*.
- Generated randomly, by user, by computer with user input

Sequence of words
- Examples: pass-phrases

Algorithms
- Examples: challenge-response, one-time passwords

Passwords Generation

- User-Generated: created by the user
  - Easy to remember
  - Easy to guess!
    - dictionary words
    - obvious personal information
    - keyboard pattern (aaaaa, qwerty)
    - reused password
- Computer-Generated: created by the computer
  - stronger (random)
  - change periodically (password aging)
  - pronounceable (why55Go60)
  - some users will write it on a paper!

Passwords Storage

Store as cleartext

- If password file compromised, *all* passwords revealed

Encipher file

- Need to have decipherment, encipherment keys in memory
- Reduces to previous problem

Store one-way hash of password

- If file read, attacker must still guess passwords or invert the hash
- possible problem: same password = same hash!

Getting the password

- Try all possible passwords (exhaustive, brute-force)
- Try all frequent used passwords (dictionary)
- Try passwords likely used by the user (his name, his city name)
- Get it from the user!

src: palisade.plynt.com

Getting the password: Exhaustive search

- T
  -
  -
- I
  - Try all possible passwords
  - Should be computationally infeasible
- T
  -
  -
  -
  - Cain (windows)

Getting the password: From the user!

- Write on a stick-it note
- share with a workmate (laz
- Social engineering

Normal Password User Selec
passwords – easy to remer
- Based on account names, us
- Dictionary words (also reve characters, "elite-speak", co Torah/Bible/Koran/… word
- Too short, digits only, letters only
- License plates, acronyms, social security numbers
- Personal characteristics or foibles (pet names, nicknames, job characteristics, *etc*.

Picking Good Passwords

"LlMm*2^Ap"
- Names of members of 2 families

"OoHeO/FSK"
- Second letter of each word of length 4 or more in third line of third verse of Star-Spangled Banner, followed by "/", followed by author's initials

What's good here may be bad there
- "DMC/MHmh" bad at Dartmouth ("Dartmouth Medical Center/Mary Hitchcock memorial hospital"), ok here

Why are these now bad passwords? ☹

Proactive Password Checking

Analyze proposed password for "goodness"
- Always invoked
- Can detect, reject bad passwords for an appropriate definition of "bad"
- Discriminate on per-user, per-site basis
- Needs to do pattern matching on words
- Needs to execute subprograms and use results
  - Spell checker, for example
- Easy to set up and integrate into password selection system

Proactive Password Checking Examples:

Goal: check passwords against large dictionaries quickly

- Run each word of dictionary through k different hash functions $h_1, \ldots, h_k$ producing values less than n
- Set bits $h_1, \ldots, h_k$ in OPUS dictionary
- To check new proposed word, generate bit vector and see if *all* corresponding bits set
  - If so, word is in one of the dictionaries to some degree of probability
  - If not, it is not in the dictionaries

Provides little language to describe proactive checking

- test length("$p") < 6
  - If password under 6 characters, reject it
- test infile("/usr/dict/words", "$p")
  - If password in file /usr/dict/words, reject it
- test !inprog("spell", "$p", "$p")
  - If password spelled correct, reject it

Password Salting

Adjusting a user password by introducing small change called *salt*

Same passwords result in different stored passwords!

Goal: slow dictionary attacks

Example:

Original password: mike

System change it to: mikeA#8$

Unix Passwords

Vanilla UNIX method

- Use DES to encipher 0 message with password as key; iterate 25 times
- Perturb E table in DES in one of 4096 ways
  - 12 bit salt flips entries 1–11 with entries 25–36

Alternate methods

- Use salt as first part of input to hash function

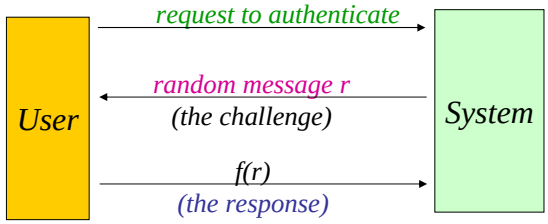
Password Aging

Force users to change passwords after some time has expired

- How do you force users not to re-use passwords?
  - Record previous passwords
  - Block changes for a period of time
- Give users time to think of good passwords
  - Don’t force them to change before they can log in
  - Warn them of expiration days in advance

Challenge-Response Authentication

- User & system share a secret function *f*
- Password changes each time (no replay)
- In practice, *f* is a known function with unknown parameters, such as a cryptographic key
- One-time password



Examples of *f(r)*
f(r) = r + 1 : simple mathematical function
f(r) = r (r) : seed a shared random generator
$f(a_1a_2a_3a_4a_5a_6) = a_3a_1a_1a_4$: mapping pattern
f(E(r)) = E(D(E(r)) + 1) : encryption algorithm (DES/AES/RSA)

One-Time Passwords

Password that can be used exactly *once*
- After use, it is immediately invalidated

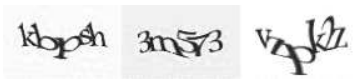
Challenge-response mechanism
- Challenge is number of authentications; response is password for that particular number

Problems
- Synchronization of user, system
- Generation of good random passwords
- Password distribution problem

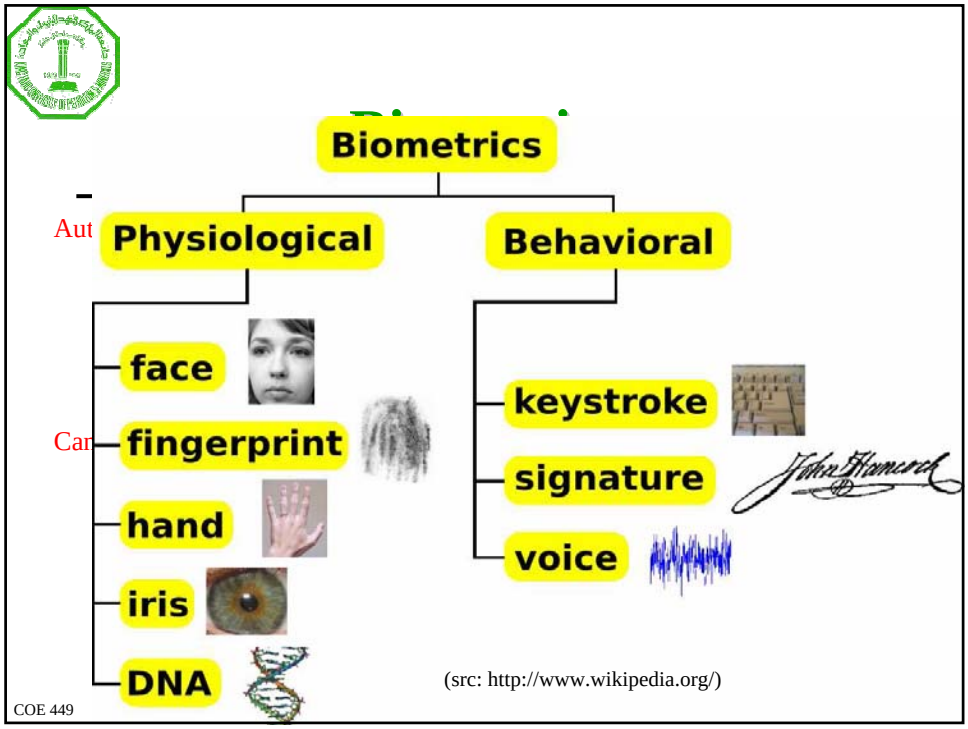
Hardware Support Challenge Response

(src: http://www.wikipedia.org/)

CAPTCHA



- A challenge-response to distinguish humans from computers
- A human can solve easily, but a computer can not
- Completely Automated Public Turing test to tell Computers and Humans Apart (Carnegie Mellon University, 2000)



(src: http://www.wikipedia.org/)

Cautions

These can be fooled!

- Assumes biometric device accurate *in the environment it is being used in!*
- Transmission of data to validator is tamperproof, correct

Biometrics Criteria (wikipedia)

- Universality: Each person should have the characteristic
- Uniqueness: How well the biometric separates individually from another.
- Permanence: How well a biometric resists aging.
- Collectability: Ease of acquisition for measurement.
- Performance: Accuracy, speed, and robustness of technology used.
- Acceptability: Degree of approval of a technology.
- Circumvention: Ease of use of a substitute.

Biometrics Issues

New (trust)

Expensive

False readi

Attacks

Privacy co

Danger to u

Location-Based Authentication

If you know where user is, validate identity by location: seeing if person is where the user is

Check if user is logging-in from an approved location (e.g. office)

- Requires special-purpose hardware to locate user
  - GPS (global positioning system) device gives location signature of entity
  - Host uses LSS (location signature sensor) to get signature for entity

Combining Authentication Methods

- Two-factor authentication is a system wherein two different methods are used to authenticate
- Delivers a higher level of authentication assurance.
- Called *strong authentication*
- Multifactor authentication causes inconveniences!

Key Points

Authentication is not cryptography
- You have to consider system components

Passwords are here to stay
- They provide a basis for most forms of authentication

Protocols are important
- They can make hidden harder

Biometrics can be useful

Authentication methods can be combined