Computer Security

Introduction

Contact Email: AdrianEugen.Musuroi@gmail.com

Course/Lab website: http://www.aut.upt.ro/~bgroza/

Weekly sessions: Tuesday & Wednesday from 8:00 A.M.

Planning

Weeks 1-6	->	Cryptography labs
Week 7	->	Quiz #1 (short questions + exercises)
Weeks 8-12	->	Software security labs
Week 13	->	Quiz #2 (short questions)
Week 14	->	Quiz #1 + #2

Lab sessions

1. Discussion (ONLINE) [20-30 minutes]
    a. Theoretical background
    b. Lab contents
2. Assignments (OFFLINE) [40-50 minutes]
    a. These will be uploaded to Virtual Campus
    b. Please check deadlines
3. Questions (ONLINE) [20-30 minutes]
    a. Questions related to the previous lab
    b. Please email me your questions 1-2 days before

User authentication

Authentication

Q: What is authentication?
A: The action/process of proving the origin or authorship of information

Examples of user authentication systems?

- Password-based authentication
- Biometric authentication
- Token-based authentication
- Etc.

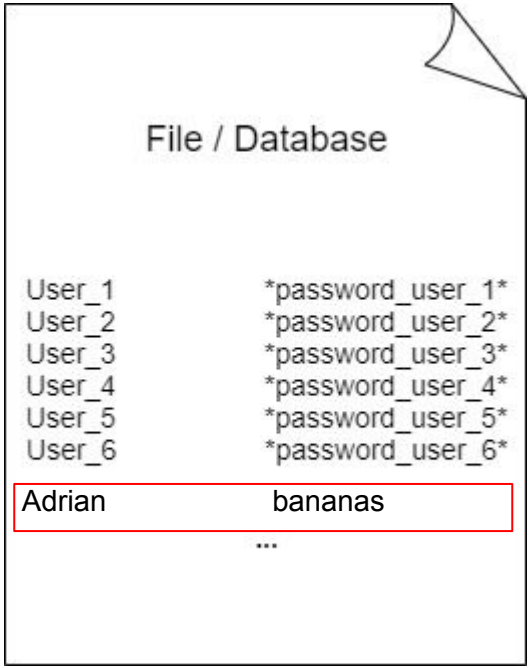
In this lab: Password-based authentication in UNIX

Goal: get insights of how to correctly design a user-password authentication system and what mistakes to avoid

Method: learn by making the mistakes

Attempt #1

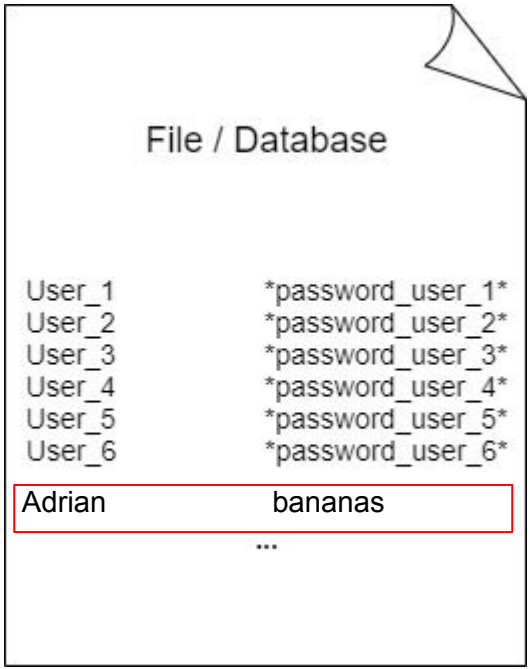
- Use a system file to store auth. information
- One user-password entry for each user
- Authentication steps
  - Find the user entry in the file/database
  - Compare input password with password from file/database
    - .. if match, authentication succeeded
    - .. else, error



Attempt #1 (cont.)

Threat analysis

Q: Who can be the adversary?



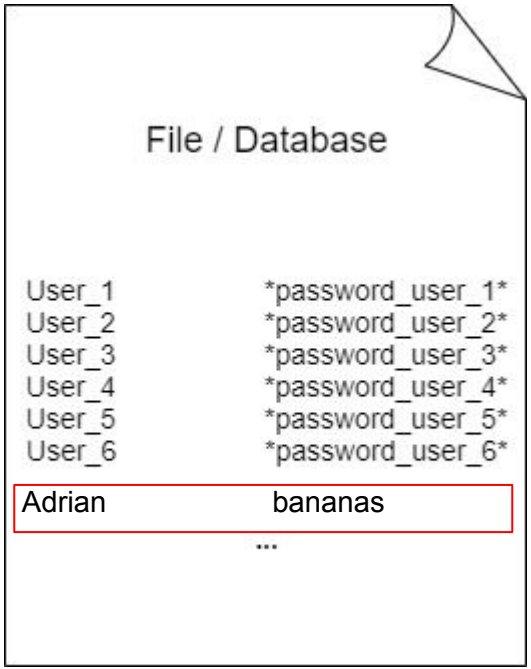
Attempt #1 (cont.)

Threat analysis

Q: Who can be the adversary?

A: E.g.: the superuser (root)

Q: What is the risk?



Attempt #1 (cont.)

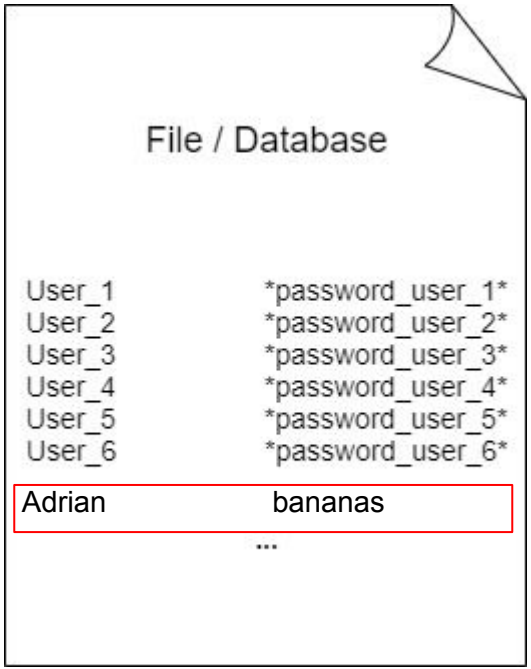
Threat analysis

Q: Who can be the adversary?

A: E.g.: the superuser

Q: What is the risk?

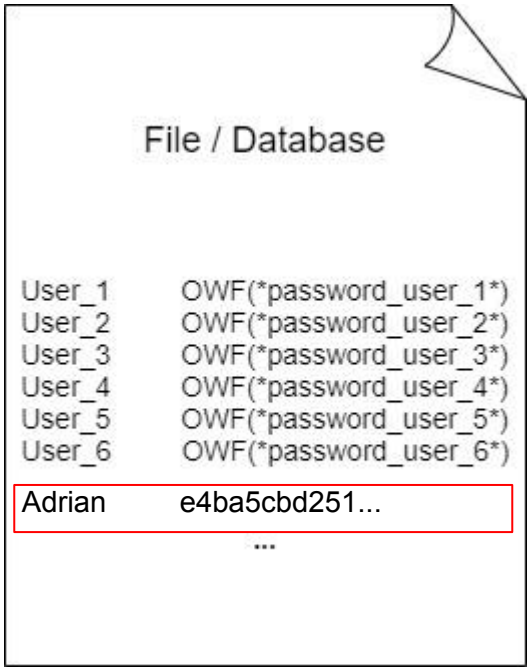
A: Anybody that can access the file will see all passwords



Attempt #2

- Replace the passwords with the output of a OWF

- Authentication steps
  - Find the user entry in the file/database
  - Compare OWF(input password) with value from file/database
    - .. if match, authentication succeeded
    - .. else, error



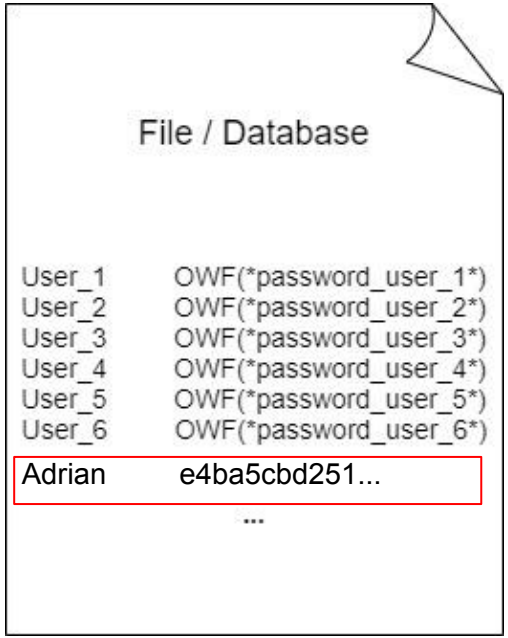
Attempt #2 (cont.)

Q: What can go wrong?

Attempt #2 (cont.)

Q: What can go wrong?

A: Dictionary attacks!



e4ba5cbd251

Password	OWF output (SHA-256)
monkey	000c285457f...
dragon	a9c43be948c...
bananas	e4ba5cbd251...
...	...

bananas

Preventing dictionary attacks

Solution

- Append a *salt* to the password before hashing
- Salt
  - Generated random
  - Not a secret

Example (SHA-256 used as OWF)

- Hash value of **“password”**
  - **5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8**
- Hash value of **“j3naup@1password”**
  - **32a141078487ce78eef3ff09a0638ffad1a46d4e3379a1cedff0b03c25e95a33**