

Computer Security

Introduction

Contact Email: AdrianEugen.Musuroi@gmail.com

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

Weekly sessions:

Other information: will be added to Virtual Campus

Lab organization

1. Discussion (ONLINE) [20-30 minutes]
 - a. Theoretical background
 - b. Lab contents
2. Exercises (OFFLINE) [40-50 minutes]
 - a. Homework exercises, available on Virtual Campus
 - b. Condition for lab attendance
3. Questions (ONLINE) [20-30 minutes]
 - a. Questions about homework (can be also asked on forum)
 - b. Discuss proposed topics

Authentication

Authentication

Q: What is authentication?

A: The action/process of proving the origin or authorship of information

Examples of 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

File / Database	
User_1	*password_user_1*
User_2	*password_user_2*
User_3	*password_user_3*
User_4	*password_user_4*
User_5	*password_user_5*
User_6	*password_user_6*
Adrian	bananas
...	

Attempt #1 (cont.)

Threat analysis

Q: Who can be the adversary?

File / Database	
User_1	*password_user_1*
User_2	*password_user_2*
User_3	*password_user_3*
User_4	*password_user_4*
User_5	*password_user_5*
User_6	*password_user_6*
Adrian	bananas
...	

Attempt #1 (cont.)

Threat analysis

Q: Who can be the adversary?

A: E.g.: the superuser

Q: What is the risk?

File / Database	
User_1	*password_user_1*
User_2	*password_user_2*
User_3	*password_user_3*
User_4	*password_user_4*
User_5	*password_user_5*
User_6	*password_user_6*
Adrian	bananas
...	

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

File / Database	
User_1	*password_user_1*
User_2	*password_user_2*
User_3	*password_user_3*
User_4	*password_user_4*
User_5	*password_user_5*
User_6	*password_user_6*
Adrian	bananas
...	

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

File / Database	
User_1	OWF(*password_user_1*)
User_2	OWF(*password_user_2*)
User_3	OWF(*password_user_3*)
User_4	OWF(*password_user_4*)
User_5	OWF(*password_user_5*)
User_6	OWF(*password_user_6*)
Adrian	e4ba5cbd251...
...	

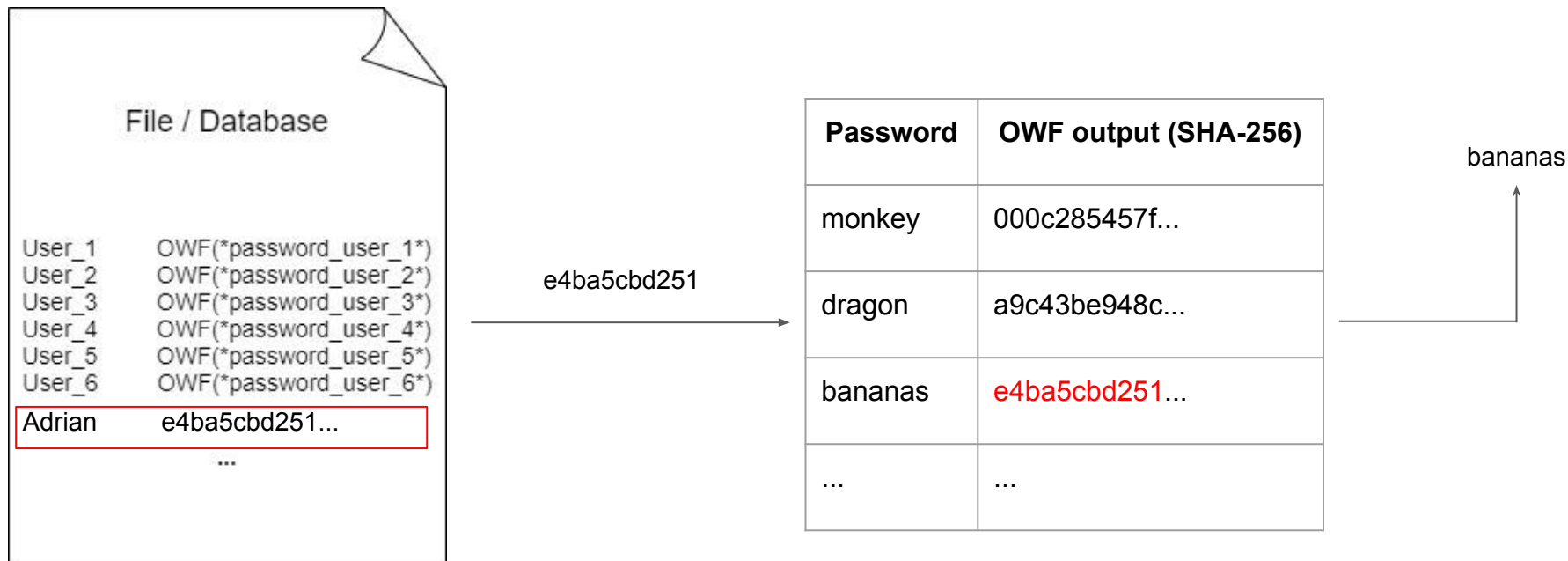
Attempt #2 (cont.)

Q: What can go wrong?

Attempt #2 (cont.)

Q: What can go wrong?

A: Dictionary attacks!



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@1**password”
 - **32a141078487ce78eef3ff09a0638ffad1a46d4e3379a1cedff0b03c25e95a33**