

Class activity

A hands-on exercise from the class-activities appendix. The full text also appears in Appendix A of the book.

Accompanies *Computer Security: The Foundations*, Krerk Piromsopa, Ph.D., 2026.
<https://www.cp.eng.chula.ac.th/~krerk/books/ComputerSecurity/>

Authorized use only

These exercises involve real attack techniques. Carry them out only against systems you own, or that you have written permission from the owner to test. Unauthorized access, scanning or interception is a criminal offence under Thailand's Computer Crime Act and its equivalents elsewhere, whether or not any damage results. If you are unsure whether a target is in scope, ask your instructor before you start.

Activity 2: Hacking Password

Objectives

To understand the concepts of hashing and salting.

Overview

This activity demonstrates the fundamentals of password security. Several hacking techniques are illustrated throughout the exercises. In particular, you will learn: brute-force attack, rainbow-table attack, and password analysis.

We will use a free password dictionary from the following URL as our dictionary:

```
https://github.com/danielmiessler/SecLists/blob/master/Passwords/
Common-Credentials/10k-most-common.txt
```

Prerequisites

Prepare a computer with Python installed, and install bcrypt (hashlib is part of the Python standard library, so it needs no installation):

```
$ pip install bcrypt
```

Here is sample code that may be useful in this activity.

```
import hashlib
import bcrypt

# SHA-1
m = hashlib.shal(b"Chulalongkorn").hexdigest()
print(m)

# MD5
m = hashlib.md5(b"Chulalongkorn").hexdigest()
print(m)
```

```
# bcrypt returns bytes, not a hex digest
salt = bcrypt.gensalt()
m = bcrypt.hashpw(b"Chulalongkorn", salt)
print(m.decode())
```

Exercises

1. **Objective:** Understand how attackers use pre-built word lists (dictionaries) to crack the hashes of common passwords.

Scenario: You have discovered a SHA-1 hash on a compromised system:

d54cc1fe76f5186380a0939d2fc1723c44e8a5f7

You suspect the password is a simple, common word, possibly with some character substitutions.

Task: Write a Python program that reads a list of words, applies common substitutions, hashes the result, and checks whether it matches the target hash. Note that you may want to include substitutions in your code (lowercase, uppercase, digit-for-letter: 'o'→0, 'l'→1, 'i'→1).

2. **Objective:** To understand why modern password hashing algorithms such as bcrypt are more secure than older ones such as MD5 and SHA-1.

Task: Design and run an experiment to measure how many hashes each algorithm can compute in a fixed amount of time. The code must test at least MD5, SHA-1, and bcrypt. (You may also try additional algorithms such as SHA-256, SHA-512, scrypt, or Argon2.)

Hint: Use the time module in Python.

3. **Objective:** To apply the performance measurements to understand the importance of password length and algorithm choice.

Task: Based on the measurements from Exercise 2, estimate how long it would take an attacker to brute-force a password of a given length. You may assume that the password contains only upper-case letters, lower-case letters, digits, and symbols.

What does this suggest about the length of a proper password? (That is, a password that would take more than a year to brute force.)

4. If a given hash value is produced by the bcrypt algorithm, is it practical to mount a brute-force attack?
5. If a given hash value is produced by the bcrypt algorithm, is it practical to perform a rainbow-table attack?
6. You have to store a password in a database. Explain your design and strategy for storing it securely. (Hints: proper hash function, salting, cost factor, database security.)

From *Computer Security: The Foundations*, Kerk Piromsopa, Ph.D.
Reproduce and adapt freely for non-commercial classroom instruction, with attribution. Full terms: LICENSE.txt in the student package.