ECE458/750: Computer Security
Assignment 1

Prepared by Chenghao Tan *

Out: May 29th, 2026
Due: June 12th

1 Instructions

This assignment is broken into the three parts described below. The parts are broken up based on feedback type with Part A giving fast automated feedback, Part B being an interactive game, and Part C being a more traditional assignment where you can write out your reasoning.

A) Learn Quiz - this is a set of assignment questions where the answers can be entered onto Learn and immediately graded.

B) VM-based Password Game - You will be breaking into a game-based website that is protected by a range of password-related protections.

C) Crowdmark Questions - open-ended questions where the answers need to be uploaded onto Crowdmark.

2 Part A: Learn Quiz

"Assignment 1 Part A" Quiz on Learn

This is completed on Learn. If you want to do it offline, we recommend starting the learn quiz and downloading the page of questions. You can take the Learn quiz as many times as you like. You can also fill out only a single question on the quiz to get feedback on questions one at a time. This part of the assignment is designed to provide you with fast feedback.

3 Part B: VM-based password game

Details about where to download the VM will be announced soon.

You will be playing a game on a Virtual Machine (VM) that involves launching various offline and online password attacks against a website. Because launching such attacks against a live server could be problematic, we have placed the game on a VM with a local web server.

There are no questions to fill out for grading for this part. The game structure is designed to make it clear when you succeed in passing levels. It also provides hints and feedback.

Feel free to talk about the game over Piazza or with others in the course. The password you need to advance to the next level should be different for different students. So the only real way to cheat on the game is to share password-hacking code.

*Based on an earlier assignment by Tanmayi Jandhyala, Kami Vaniea and Cameron Hadfield

4 Part C: Crowdmark Questions

The questions below should be filled out and uploaded onto Crowdmark. You can reference the questions here or see them on Crowdmark. This part of the assignment will not be graded till the assignment deadline.

Question 1. *Authentication*

a) The BBC wants to create an authentication mechanism that works well for children who are too young to create or remember passwords. It would be expensive to send every child a physical token like a YubiKey, and most young children do not have cell phones. For privacy reasons they also do not want to record biometrics about the children.

To solve the problem, they decide to ask each child to draw a picture on a piece of paper as part of the account creation process, the child then shows the picture to the camera which records it. To login in the future, the child only needs to show the camera the exact same physical drawing.

i) State the kind of authentication (something you...) is being described above by using drawings to log in. Explain your reasoning

Solution: Something you have. Because the drawing is unique object, but it can be used by others that "have" it.

ii) Is the above approach better or worse than passwords for the intended audience? Justify your answer.

Solution: Probably worse than passwords. Implementation-wise its is about the same as face recognition. The password space is also large (good). But the target population is quite likely to lose the drawing (bad). The drawing is also something that would need to be carried around without damaging it (bad). Anyone with access to the drawing can also log in (bad). Furthermore, visual recognition of the drawing would be prone to errors. The account is also likely always/mostly logged into from the same device, so a cookie would possibly have been more seamless.

iii) Describe the threat model that the BBC is likely envisioning when they created this authentication scheme. What are capabilities they assume the attacker has, what are the capabilities they assume the attacker does not have?

Solution: They assume a remote online attacker. Likely a stranger. A child drawing, while simple looking, is actually quite complex from the perspective of machine learning. Exactly duplicating a bunch of crayon scribbles is actually fairly hard especially if the reference drawing is not available. They assume that the attacker does not have physical access. They may also be assuming that the child does not know the attacker, as the child can be easily convinced to show their drawing to someone they know.

iv) What might a backup authentication scheme look like for this approach?

Solution: The most obvious answer is to have a parent set a password during account setup to use as a backup. If a child-friendly solution is needed, then possibly allowing them to use their favorite toy as the backup, though the security would be poor since many children have the same toys.

Question 2. *Cryptography.*

(a) Consider the below advice in the context of using encryption to protect "data in motion" like Internet network connections, as well as "data at rest" such as database storage.

(i) Why do cryptologists recommend changing these encryption keys from time to time?

(ii) Should old keys be completely discarded? Explain your answer.

Solution: Keys need to be changed or "rotated" to limit damage in case they are recovered or compromised. The second question: unless keys are ephemeral, old key IDs need to be preserved for the system to be able to decrypt previously encrypted data. It is to ensure availability. This can be termed

key versioning. The idea of changing keys and changing passwords comes from a similar standpoint of keeping data secure against key or password compromises. Passwords are changed because they're low-entropy, reused, or guessed; encryption keys, on the other hand, are usually high-entropy and randomly generated. Also, password changing can be easily done from the user's end. Changing keys might not be as straightforward and would require for users to give the right parameters to ensure they're properly changed or rotated. A system that relies on users to rotate keys is, in general, not good system design. These aspects should be engineered and audited frequently for fail safes. When deciding how often to rotate keys, one should consider the sensitivity of the data, volume of encrypted content per key, compliance requirements, and whether the encryption system supports rekeying. In high-stakes or long-term security scenarios (e.g., healthcare or financial systems), shorter rotation intervals are generally safer.

(b) State two situations in which using a shorter key length, such as a 128-bit key length, could be security problematic compared to using a longer key length. (*Hint: Think about different threat models, different capabilities of attackers, and the range of security needs of system designers.*)

Solution:

(a) For data that needs to remain secure for decades (like healthcare or national security archives), 128-bit keys may not provide sufficient protection against future advances in computing or cryptanalysis. In these cases, using AES-256 provides a longer margin of safety.

(b) Quantum Threats: Grover's algorithm theoretically reduces brute-force complexity from 2^n to $2^{n/2}$, so a 128-bit key would offer about 64-bit quantum security. This is considered borderline for long-term confidentiality. AES-256 would offer 128-bit post-quantum security under Grover.

(c) If a system uses AES-128 with poor practices such as key reuse, poor entropy sources, or insecure modes like ECB, the fixed key length becomes irrelevant because the implementation introduces vulnerabilities.

(d) NIST regulations now requires for AES-256. Even if AES-128 is technically secure, using it may be non-compliant in those settings.

(c) Encrypt the following three strings using AES in ECB mode and record the ciphertext lengths (in bytes or hex digits). Use a tool like CyberChef and ensure the mode is set to AES-ECB. Use a consistent key.

- `this_is_user42`
- `this_is_user42_and_friends`
- `this_is_user42_and_many_more_friends`

What do you observe about the ciphertext hex characters?

Question 3. *Introduction to password cracking on Linux*

In this problem, you will learn a bit about how Linux stores passwords for authentication purposes. For all the questions in this problem, we will be referring to the `root` password from the provided a1-shadowfile.txt on the course website:

(a) Find the line of a1-shadowfile.txt that is associated with the root user. Using the information in that line, answer the following:

- What algorithm was used
- What salt was used
- What is the user's hashed password

Solution: In order,

- yescrypt
- j9T$t4HYYraTPjT8AtgercDbi
- RUA3FujaNm1e.R1zmYlosZcONtD3sUv/UTujtUoywu5

(b) Why does Linux use a salt when storing passwords?

Solution: Salt is used by the system to add an element of randomness to hashes so that the hashed passwords on one system are not directly comparable to those of another. It improves Authentication by making guessing passwords harder and take longer thereby ensuring that we can accurately authenticate people.

(c) Describe two threats: one where salting the password hashes helps keep users safe and one where salting hashes has no impact on user safety.

Solution: 1. Offline attacker. This is the threat model that hashes are intended to protect against.

2. Online attacker, this attacker never interacts with hashes, so salting does not impact them.

(d) What is root's plaintext password? What method did you use to find it? *HINT: For simplicity, it is one of the top 10 passwords given on the slides.*

Solution: The password is 'password' and it is hashed under yescrypt. The following program can be used to verify this:

```
#include <stdio.h>
#include <unistd.h>
#include <crypt.h>
int main() {
    const char *password = "password";  //you can also make a dictionary and iterate
    const char *salt = "$y$j9T$t4HYYraTPjT8AtgercDbi.";
    char *hash = crypt(password, salt);
    printf("Hash: %s\n", hash); //match the hash with the one in (a)

    return 0;
}
```

5 Crowdmark Graduate Student Section

The following questions are intended for graduate students taking the course. Anyone is welcome to attempt the questions. And like other assignments grading for this one does not count towards the final grade in the course.

Question 4. *MD5 Hashing and its Problems*

MD5 is a hash function widely used in cryptographic applications and was found to be vulnerable to computationally feasible attacks wherein an adversary could break MD5's property of `collision resistance`, which is a necessary property of cryptographic hash functions.

To complete this segment, students should follow the SEED lab on MD5 Collisions: https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_MD5_Collision/.

Question 5. *Password Cracking*

`John the Ripper` is a well-known password-cracking tool[1], commonly used for cracking Linux account passwords stored in the shadow file.

We provided a shadow file to run John the Ripper against, in `shadowfile.txt`.

Solution: These are the passwords, all taken from Rockyou (except for obligatory)

mintenjoyer: candycane
sneaky: zxcvbnm,./
secure: !!w2rockyou!!
ridiculous: !(()*)@*)&*
obligatory: correct-horse-battery-staple
passmgr: 1password

Question 6. *Symmetric-Key Encryption*

This section asks you to perform several activities with basic encryption. We recommend that you complete the Secret-Key Encryption SEED lab Tasks 1-5 which will help you understand how to answer the questions in this section.
SEED lab: https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_Encryption/
Encrypted file: a1-ciphertext.txt

1. Perform Frequency Analysis on the file provided in the assignment folder on Learn: `a1-ciphertext.txt` and answer the following questions:

   (a) What are the 5 highest frequency 1-grams, 2-grams and 3-grams of the text?
   Solution:

   - 1-grams
     i. n
     ii. k
     iii. u
     iv. y
     v. i
   - 2-grams
     i. kw
     ii. wn
     iii. yx
     iv. ni

[1] "openwall/john." Openwall, May 14, 2024. Accessed: May 14, 2024. [Online]. Available: https://github.com/openwall/john

v. in

- 3-grams
  i. kwn
  ii. gjk
  iii. igj
  iv. cig
  v. ydx

(b) Based on your analysis, what is the key used in the substitution?

Solution: The key is ubcrnvmwyqtapxdjoieksflhgz. Note that this key is the one used to substitute the plaintext to the ciphertext.

(c) The original text was taken from the web. What website was it most likely copied from, provide the URL.

Solution: https://en.wikipedia.org/wiki/Cryptography

2. Please define the key differences between ECB mode and CBC mode encryption. What are the performance implications of this difference?

   Solution: ECB mode treats each block as a different encryption, whereas CBC uses the result of the previous block to encrypt the current block. For this reason, ECB is fully parallelizable, as each block encryption can happen simultaneously, but CBC mode cannot be parallelized.

3. Using AES-128, and encrypting a 1000-byte file, how many bytes can be recovered if the 100th byte is corrupted under the following modes:

   (a) ECB

   (b) CBC

   (c) CTR

   Solution:

   - In ECB mode, it will affect only the block in which the corruption occurred. In AES-128, this should be 16 bytes. (984 bytes recoverable)
   - In CBC mode, it will affect the block where the corruption occurred, and the following one in decryption (because the error propagates to the next block's decryption). Though this is how it would make sense, this isn't quite true. 1 block and 1 additional byte are corrupted, thus 983 bytes are recoverable.
   - In CTR mode, all but one byte will be properly recovered. Meaning, the total difference is 1 byte (999 bytes are recoverable).