CSCI 331:
Introduction to Computer Security

Lecture 5: Passwords

Instructor: Dan Barowy

Williams

Topics

Office hours:
Tuesday 1:10-2:25pm (**TBL 301**)
Thursday 4-6pm (**TBL 301**)

Lab 1

Reading discussion (Schneier)

Password systems

Crypto primer

Your to-dos

1. Lab 1 **due Sunday 9/26** by 11:59PM.
2. Reading response (Oechslin) **due Wed, 9/29**.
    1. This one is a *technical review*.
3. Project part 1 due **Sunday, 10/3**.

Lab 1

- Small typo in lab handout—fixed.
- `fflush`—no, sorry, my mistake.
- Please use the libraries, e.g., database.
- How does a pseudoterminal work, really?
- `gdb` is very helpful.
- Don't use `sudo` if you don't need to!

2.8.1 *pty, a pseudoterminal demo*

The `ptyhelper.c` program supplied in your starter code demonstrates how to create a pseudoterminal and attach it to a program you want to control. Chapter 3 explains how to use the helper code to create a program that controls another program.

In this part, you will create a file called `pty.c`. You should be able to compile this program by typing `make pty`, which should produce a binary file called `pty`. Since the supplied `Makefile` does not have a rule to do this, you will need to modify it to add a `pty` target. *Take note* that, when compiling with `gcc`, any program that makes use of the `ptyhelper` library must include the `-lutil` flag. The `-lutil` flag tells `gcc` to find several of the pseudoterminal functions in the `libutil.so` system library. [2]

Specification:

1. Write a program that attaches to `login0`. Call this program `pty`.
2. Call `exec_on_pty` with an appropriately constructed `argv`.
3. Manipulate the file descriptor returned by `exec_on_pty` using `read` and `write` system calls. [3]
4. `pty` should supply a single correct username and password (look in the `password.db` file) to `login0`, print `It worked!` when `login0` returns `ACCESS GRANTED`, and then quit with a non-zero exit code.

[2] `ptyhelper.c` calls the `openpty` C library function, which is not normally in `gcc`'s library search path. Appending `-lutil` tells `gcc` to search for the implementation of this function elsewhere. How did I know to do this? `man openpty` told me to do it.

[3] The `read` and `write` system calls can read and write arbitrary data, including binary data. This means that, if you're reading and writing strings, those calls do not know and they do not help you handle strings. Recall that C strings must always be null-terminated. Does `read` ensure that strings are null-terminated? Read `$ man 2 read` to find out!

Reading discussion

3 vs 4: **thoughtfulness**

Next week: **technical review**

Four major security concerns

- Confidentiality
- Integrity
- Authenticity
- Availability

Confidentiality

Confidentiality is the property that information is **not made available** or disclosed to **unauthorized** individuals, entities, or processes.

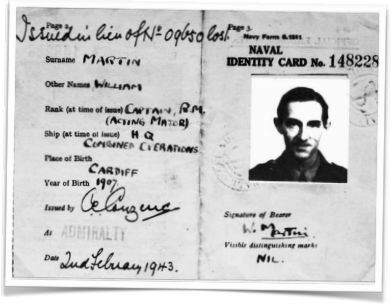
Integrity

Integrity is the property that information is **accurate**, **complete**, and **consistent** over its entire lifecycle. Importantly, information **should not be modifiable** by an **unauthorized party** or in an **undetected manner**.

Ferris changes his grade in "Ferris Bueller's Day Off."

Authenticity

Authenticity is the property that a **fact** or **identity** is **true** or **genuine**.



"Operation Mincemeat"

Operation Mincemeat

- Successful British intelligence operation (1943)
- Fooled Nazi military into believing that allied troops would invade Italy via Sardinia instead of Sicily.
- Body of deceased sailor ("Capt. William Martin") set afloat from submarine HMS Seraph with forged identity documents.
- Body was actually Glyndwr Michael, a homeless Welsh man who died after eating rat poison.
- Spanish fishermen found body; passed on to Nazi intelligence.
- Nazis redirected troops to Sardinia; allies invaded via Sicily.

Non-Repudiation

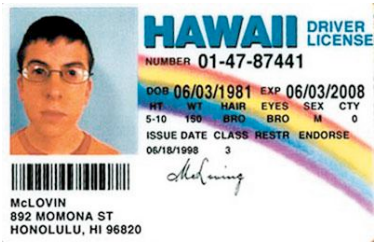
Non-repudiation is the property that an **action** can be associated with a **unique actor** (e.g., an individual or process). Such actors **cannot dispute the association**.

Non-Repudiation

Non-repudiation is the property that an **action** can be associated with a **unique actor** (e.g., an individual or process). Such actors **cannot dispute the association**.

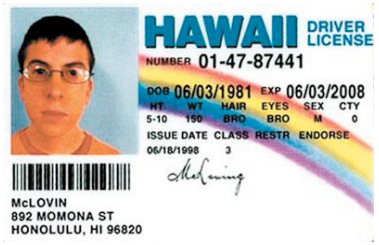
Non-Repudiation

Non-repudiation is the property that an **action** can be associated with a **unique actor** (e.g., an individual or process). Such actors **cannot dispute the association**.



Non-Repudiation

Non-repudiation is the property that an **action** can be associated with a **unique actor** (e.g., an individual or process). Such actors **cannot dispute the association**.



Availability

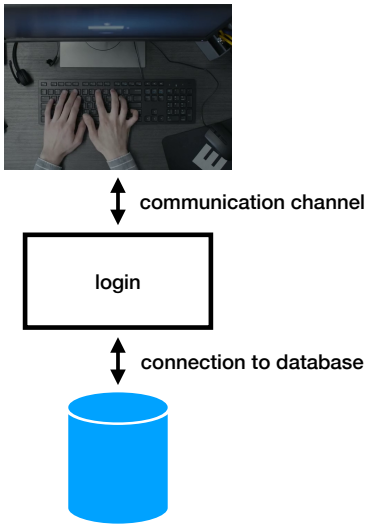
Availability is the **proportion of time** that a resource is in **functioning condition**.

Demonstration

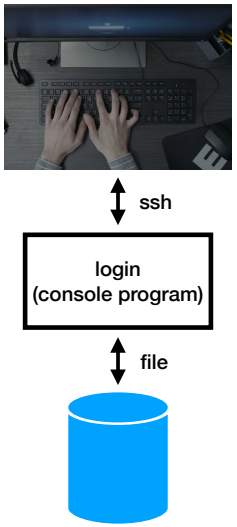
What **properties** are violated here?

Password Databases

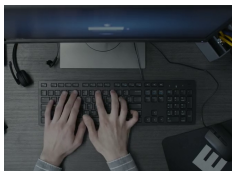
How a Password Database Works

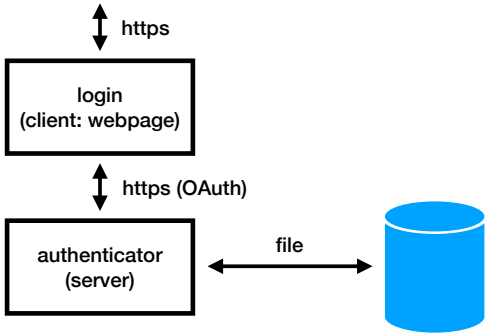


Example

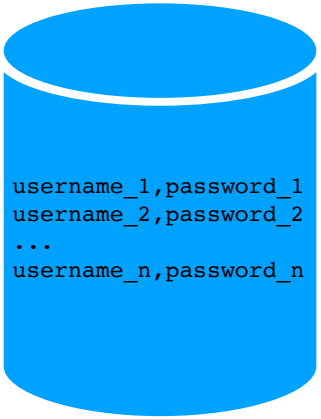


Example





Form of a password database



Kept in sorted order by username (allows fast lookups).

Short cryptography primer

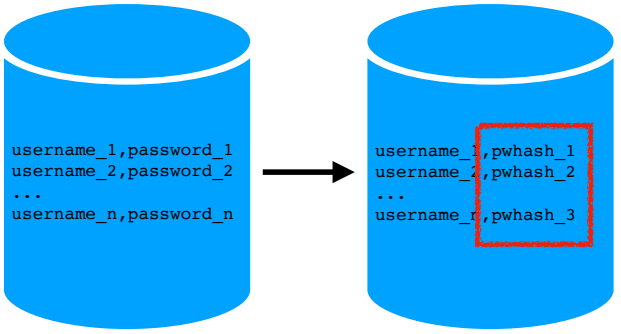
Encryption is the **process of encoding a message** so that it can be read only by the sender and the **intended recipient**.

- A **plaintext** p is the original, unobfuscated data. This is information you want to protect.
- A **ciphertext** c is encoded, or encrypted, data.
- A **cipher** f is an algorithm that converts **plaintext** to **cipertext**. We sometimes call this function an **encryption function**.
  - ✳More formally, a cipher is a function from plaintext to ciphertext, $f(p)=c$. The properties of this function determine what kind of encryption scheme is being used.
- A **sender** is the person (or entity) who enciphers or encrypts a message, i.e., the party that converts the plaintext into cipertext. $f(p)=c$
- A **receiver** is the person (or entity) who deciphers or decrypts a message, i.e., the party that converts the ciphertext back into plaintext. $f^{-1}(c)=p$

See the reading Why Stolen Password Databases are a Problem for a little more nuance.

A Common Attack

Entire password database **leaked** (bug; misconfiguration; theft by authorized personnel).

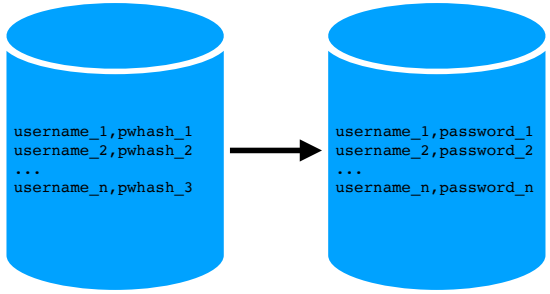


We keep password databases in **encrypted** form.

Password databases are encrypted

But the details of the encryption may still leave it open to attack.

A **dictionary attack** is a form of **brute force attack** technique for **recovering passphrases** by systematically **trying all likely possibilities**, such as words in a dictionary.



Recap & Next Class

Today we learned:

- CIAA
- Password systems
- Cryptography primer
- Dictionary attacks

Next class:

- Precomputed hash chain attack (PCHC)