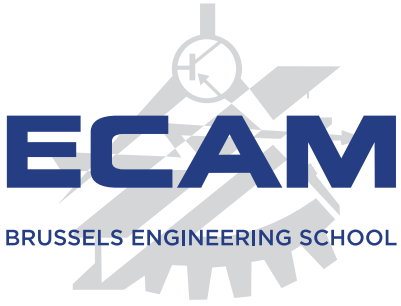


SE507μ Hacking Password Hashes
with Rainbow Tables

Session 1

Hash Function and Password Storage

Sébastien Combéfis

Winter 2020

Objectives

- Understand user authentication and how to implement it

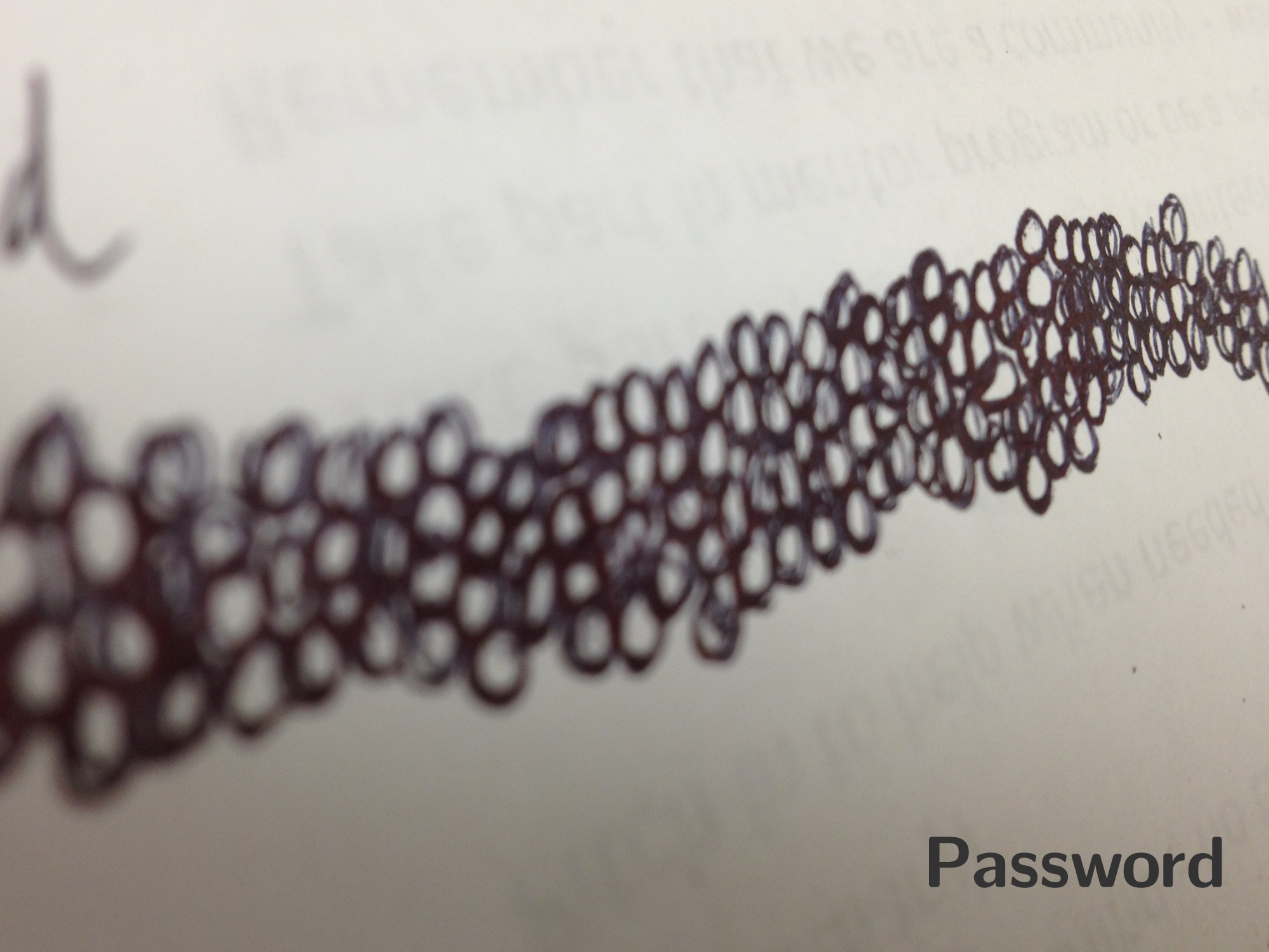
  Discover how identifiers and passwords are used for that

- Understand what are (cryptographic) hash functions

  Discover data integrity, collision and compact information storage

- Understand how to securely store passwords in a database

  Password storage and verification to authenticate users


Password

Authentication

- Important to be able to authenticate users of a system

  Checking that the identity of a user is authentic

- Three possible elements to authenticate a user
  - **Possession**: a key, a card, etc.
  - **Knowledge**: an identifier with a password, etc.
  - **Attribute**: a fingerprint, a signature, etc.
- Authentication process consists in two steps
  - **Identification**: presenting an identifier to the system
  - **Verification**: proving relation between entity and identifier

Password

- Authentication with a username and the associated password

  Password compared with the one stored in the system

- Different kinds of associations for passwords
  - One password for each object to protect
  - Protecting a set of access rights with the same password

- One password should only be used for one access

  Not a good practice to use the same password for the same user

Password Vulnerability

- Using passwords is not secure at all
  - Could be easy to guess, can be exposed, sniffed, etc.
  - Can be illegally transfered to an unauthorised user!
- Only 10,000 possibilities with a four-digit pin code

  Only 5,000 attempts on average (only 5s if one test/ms)
- A password can be seen while it is exposed

  Shoulder surfing, network sniffing, keylogger, etc.

Securing Password

- Passwords should be secretly stored in the system

  But it must be possible to check whether a password is correct

- The storage should be protected against theft

  Use disk encryption, physical protection, backups, etc.

- Passwords should not be stored in clear in the database

  Typically storing them in a secured hashed form

Brute-Force Attack

- Brute-force attack tries all the possible passwords

  The passwords space should be large enough

- Using botnets to make the attack legitimate

  Simulate multiple users trying to access the resource

- Require an access to the system and the possibility to connect

  Not always possible to do so, in particular for remote access

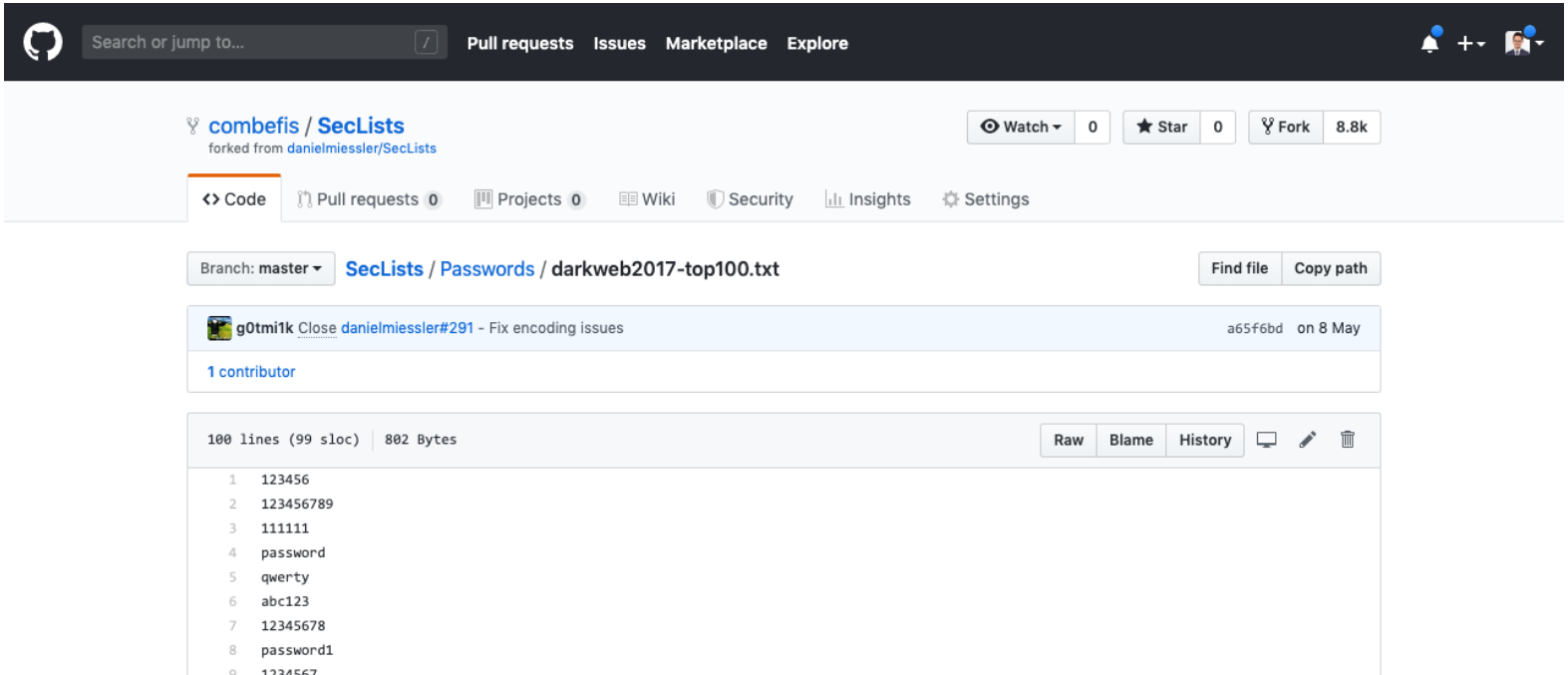
Dictionary Attack

- Try all the passwords from a dictionary

  It is an improvement of brute-force attack, with fewer trials

- Most people use common words as passwords

  Dictionary can be general or specialised for a particular target



Hash Function

Hash Function

- Initially used to check for data integrity

  Computing a digital fingerprint for a given data

- Using a hash function h to get a fingerprint $y = h(x)$

  For any x, a binary sequence of arbitrary length

- A fingerprint is a binary sequence (typically 160 bits)
  - Storing the data x and its fingerprint $h(x)$ separately
  - The fingerprint $h(x)$ should be stored in a secure place

Collision

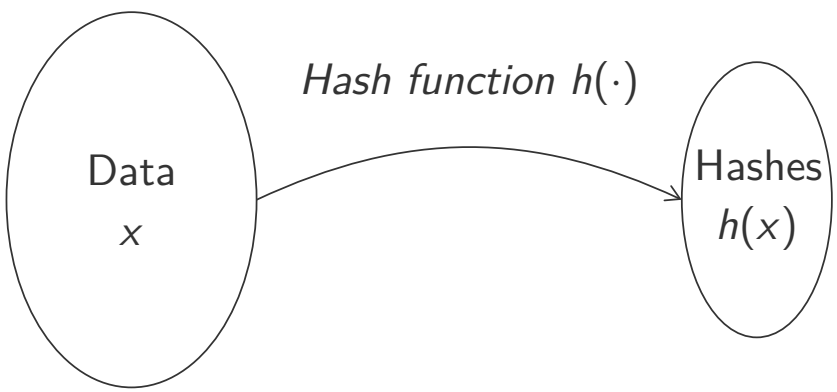
- Hash function does some compression of the data

  The domain of the function is larger than its image

- Two different data x and x' can result in the same fingerprint

  This is known as collision and is expected from hash functions

- Ideally collisions must be minimised



Hash Function Security

- A pair (x, y) is valid if and only if $h(x) = y$

  It means that y is a fingerprint for x

- Three problems hard to solve for a hash function to be good

  - **Preimage**

    Given $h(\cdot)$ and y,
    find x such that $h(x) = y$ $\qquad$ ((x, y) *valid*)

  - **Second preimage**

    Given $h(\cdot)$ and x,
    find x' such that $x' \neq x$ et $h(x') = h(x)$ $\qquad$ ($(x', h(x))$ *valid*)

  - **Collision**

    Given $h(\cdot)$,
    find x, x' such that $x' \neq x$ and $h(x') = h(x)$

Cryptographic Hash Function

- The only way to find $h(x)$ for x is by computing the function

  Even if you already have several computed values $h(x_1), h(x_2), ...$

- Adding a secret key as an input to the function

  The key K is required in addition to the data x: $h_K(x)$

- The secret key is only known by the people communicating

  Very important to store securely this secret

Hash Function Example

- MD5 only used for checksum to test data integrity

  Even if it has been designed to be a cryptographic hash function

- LM Hashes used in the Windows world for passwords

  Very weak hashes that can be easily reversed

- SHA-1, SHA-2, SHA-3 form the Secure Hash Algorithm family

  The last one has been released on August 5, 2015

VirtualBox

search...
Login Preferences

Download VirtualBox

About
Screenshots
Downloads
Documentation
End-user docs
Technical docs
Contribute
Community

Here you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

By downloading, you agree to the terms and conditions of the respective license.

If you're looking for the latest VirtualBox 5.2 packages, see VirtualBox 5.2 builds. Please also use version 5.2 if you still need support for 32-bit hosts, as this has been discontinued in 6.0. Version 5.2 will remain supported until July 2020.

VirtualBox 6.0.14 platform packages

- ⇨Windows hosts
- ⇨OS X hosts
- Linux distributions
- ⇨Solaris hosts

The binaries are released under the terms of the GPL version 2.

See the changelog for what has changed.

You might want to compare the checksums to verify the integrity of downloaded packages. *The SHA256 checksums should be favored as the MD5 algorithm must be treated as insecure!*

- SHA256 checksums, MD5 checksums

Note: After upgrading VirtualBox it is recommended to upgrade the guest additions as well.

VirtualBox 6.0.14 Oracle VM VirtualBox Extension Pack

- ⇨All supported platforms

Support for USB 2.0 and USB 3.0 devices, VirtualBox RDP, disk encryption, NVMe and PXE boot for Intel cards. See this chapter from the User Manual for an introduction to this Extension Pack. The Extension Pack binaries are released under the VirtualBox Personal Use and Evaluation License (PUEL). *Please install the same version extension pack as your installed version of VirtualBox.*

VirtualBox 6.0.14 Software Developer Kit (SDK)

- ⇨All platforms

User Manual

The VirtualBox User Manual is included in the VirtualBox packages above. If, however, you would like to take a look at it without having to install the whole thing, you also access it here:

- ⇨User Manual (HTML version)

You may also like to take a look at our frequently asked questions list.

VirtualBox older builds

The binaries in this section for VirtualBox before version 4.0 are all released under the VirtualBox Personal Use and Evaluation License (PUEL). As of VirtualBox

```
c8a5cc980c9c94cdac3d94e23cf159c2433aae76b416dbfb5b1a918758f21e63 *Oracle_VM_VirtualBox_Extension_Pack-6.0.14-133895.vbox-extpack
c8a5cc980c9c94cdac3d94e23cf159c2433aae76b416dbfb5b1a918758f21e63 *Oracle_VM_VirtualBox_Extension_Pack-6.0.14.vbox-extpack
8313828f66aec0cab46d5de503692797552fd18a878979c69961264fb70f06d4 *SDKRef.pdf
e8fd7b98890b6505904bc30e0860e69b39ecb8ffda4279887a5ec6ce54e3b3b7 *UserManual.pdf
cb82f43a71f91a50b97aa27063c5c8743772a623eb0073ab8daf77e840d73eb1 *VBoxGuestAdditions_6.0.14.iso
af5de90ee25e78993e78ab099298628ab1222586274bfd5b4993182c7ce48db0 *VirtualBox-6.0-6.0.14_133895_el6-1.x86_64.rpm
9ca02945ed5c79d42de4f041a24870136f82efedeffde1b2551038ed5f5b5540 *VirtualBox-6.0-6.0.14_133895_el7-1.x86_64.rpm
6992794ae70a551ae3f7f05000317c8830717adaecf5b549f40e7534e6c266b8 *VirtualBox-6.0-6.0.14_133895_el8-1.x86_64.rpm
04a28d5399bfa18f2ba3ed1607437a6312c9aeebab542af03abb4e2b3355b01c *VirtualBox-6.0-6.0.14_133895_fedora26-1.x86_64.rpm
fd4aa9ed3cb43397c4708250303c3e683ebf9a710381705ee416bbc740ce1e6e *VirtualBox-6.0-6.0.14_133895_fedora29-1.x86_64.rpm
134aad285d64a56682904ccb879151c234bd8b51935d63090efc07f30ac69907 *VirtualBox-6.0-6.0.14_133895_openSUSE132-1.x86_64.rpm
18c73f6640d0708e724f093518fb98eb677914867ea29fd55ca4974fced57dcc *VirtualBox-6.0-6.0.14_133895_openSUSE150-1.x86_64.rpm
e6a63037caf3bc5ced1bb384b2a7fcf86b9fca5a467101ece4c5f7bf38edc4fe *VirtualBox-6.0.14-133895-Linux_amd64.run
ea3ebf2457908495128178f9b6676a77b120fa7878fd626d2baf0a8c45921370 *VirtualBox-6.0.14-133895-OSX.dmg
bda7f8baac26b00dd618b38abb64aa4fbb5d4625c170ab8bde8f4052f7a3fb81 *VirtualBox-6.0.14-133895-SunOS.tar.gz
a24b3b3fd406766c409e5e2e8cf6d88cc7a50ad6694925e640c25e0d8d913063 *VirtualBox-6.0.14-133895-Win.exe
5e12b14f0c38bf195d9592d76a9e0a128df2d38cd77c26ce5b5488397715dd0c *VirtualBox-6.0.14.tar.bz2
c7b848034939fa65be095752109009ad0c3733e5d10c09fd26b5addaf58ba159 *VirtualBoxSDK-6.0.14-133895.zip
2c6a791e783997d8156363a6a0533f3460c7d768b0c650be7c862f4e34eb81b1 *virtualbox-6.0_6.0.14-133895~Debian~buster_amd64.deb
5497e00a9cf261e15c664961da2e83817ea642ba0974ffe58f4a1fe71f56fad6 *virtualbox-6.0_6.0.14-133895~Debian~jessie_amd64.deb
fd669e70e738ea7a3fd0f10d7208ff03ae69a525bcf9cdb1c7382a002f5bd0ca *virtualbox-6.0_6.0.14-133895~Debian~stretch_amd64.deb
44178005a2085b87c751a72c6b9b8500d813f4f974279d58a675d6ef90a6b3d6 *virtualbox-6.0_6.0.14-133895~Ubuntu~bionic_amd64.deb
d6a6776e42001eda5659a7f09a8c9a8aeeaced3c9a1e40bd538ab0b48f1f9f37 *virtualbox-6.0_6.0.14-133895~Ubuntu~trusty_amd64.deb
a8f707caca8dbb113b9ad61504f6aa6d98009639531053646cd02ead54411904 *virtualbox-6.0_6.0.14-133895~Ubuntu~xenial_amd64.deb
```

Password Storage

Password Hashes

- Storing password hashes is more secure

  As it is virtually impossible to reverse them

- Only cryptographic hash functions are practically irreversible

  Brute-force is only possible, but will take long time

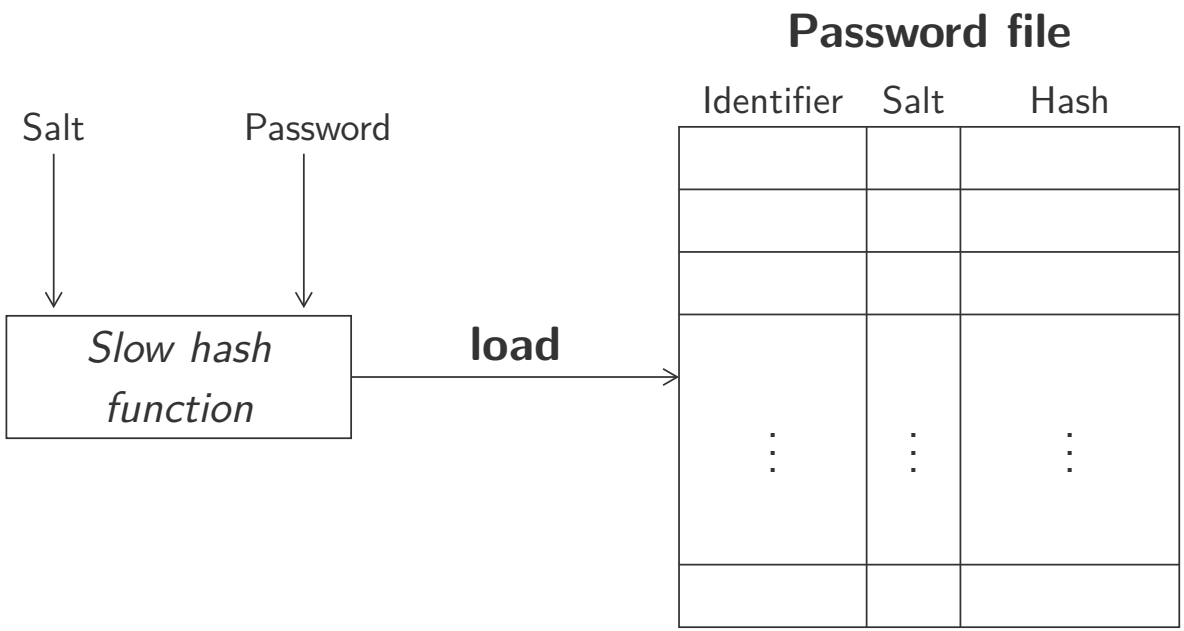
- It also helps to preserve users' confidentiality

  As the administrators do not have access to the passwords

Unix Password Scheme (1)

- Adding a new password in the system

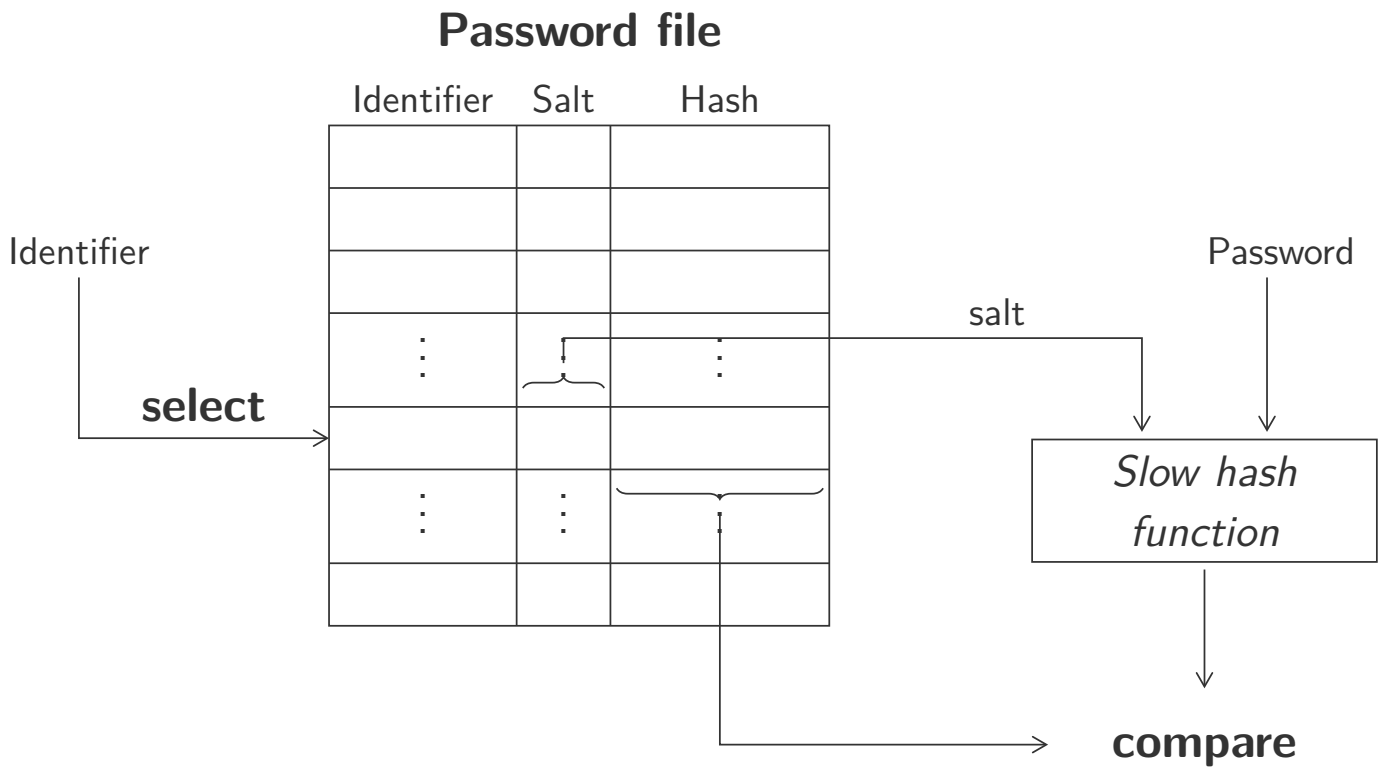
  The salt is chosen by the system and used to compute the hash



Unix Password Scheme (2)

- Checking a password for a given identifier

  The goal is to authenticate a user



Password Database

- Databases used to store password hashes should be secured

  Likely to have some database theft nowadays

- Password hashes can be reversed to guess the password

  Brute-force, dictionary, rainbow tables attacks

References

- William Stallings, & Lawrie Brown (2004). *Computer Security: Principles and Practice*, Pearson, ISBN: 978-1-29222-061-1.
- Douglas R. Stinson, & Maura B. Paterson (2017). *Cryptography: Theory and Practice* (Fourth Edition), CRC Press, ISBN: 978-1-138-19701-5.
- Defuse Security (2019). *Salted Password Hashing - Doing it Right*, June 5, 2019, https://crackstation.net/hashing-security.htm.
- Dan Arias (2018). *Hashing Passwords: One-Way Road to Security*, April 25, 2018, https://auth0.com/blog/hashing-passwords-one-way-road-to-security.
- Danny Denenberg (2019). *A Guide to Password Hashing: How to Keep your Database Safe*, August 28, 2019, https://medium.com/swlh/a-guide-to-password-hashing-671c9ab923bd.

Credits

- Jack Acecroft, March 4, 2013, https://www.flickr.com/photos/jackace/8663584323.
- stu_spivack, March 3, 2010, https://www.flickr.com/photos/stuart_spivack/4425612269.
- Mike Mozart, August 17, 2014, https://www.flickr.com/photos/jeepersmedia/14764162497.