COM-402: Information Security and Privacy 0x03 Access Control

Mathias Payer (infosec.exchange/@gannimo)

Switcheroo: Swap Class/Exercises and Quiz Reminder

- 10/02: Class 0x14 Data Security + PAKE (instead of 07/10)
- 10/07: Quiz, Exercise for 0x03 and 0x14
- 10/09: Class 0x15 PL Security
- 10/14: No class
- 10/16: Exercise for 0x15 (back to normal)

Quiz:

- No extra material allowed
- Bring a blue or black pen
- Multiple-choice format
- Duration: 15min in CO01

What do you prefer?

Exercise, then quiz at 18h30
— or —
Quiz at 16h15 and then exercises?

Learning Goals

Know authentication factors (password, device, biometrics), strengths and risks

Differentiate and identify access control policies (RBAC, DAC, MAC)

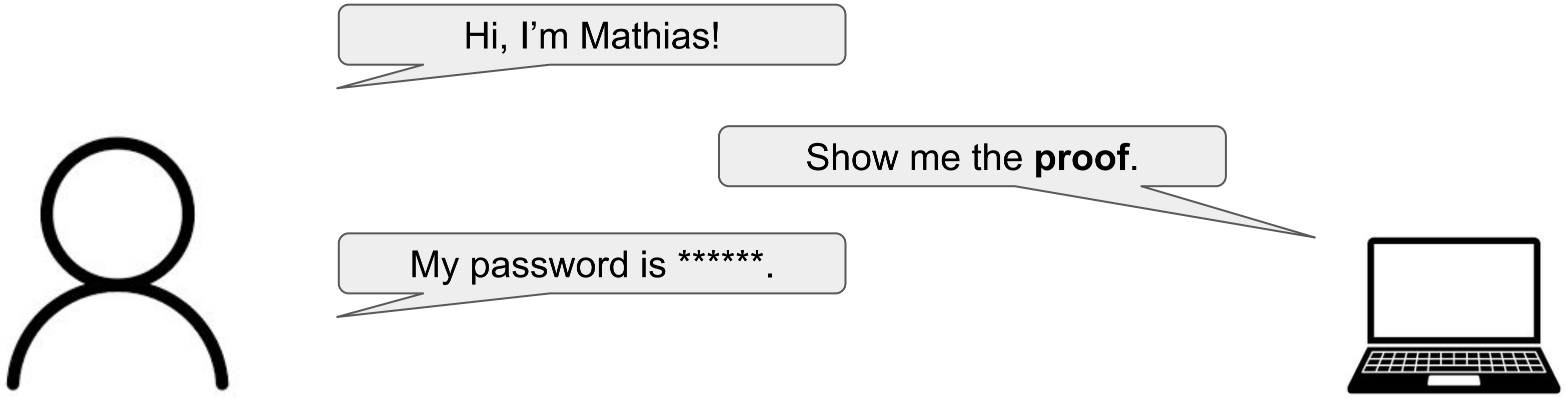
Understand the value of authentication protocols and delegated authentication

Authentication

What Is Authentication?

*Authentication is the process of **verifying** someone's or something's **identity***

Identification is the act of identifying a particular user, often through a username.

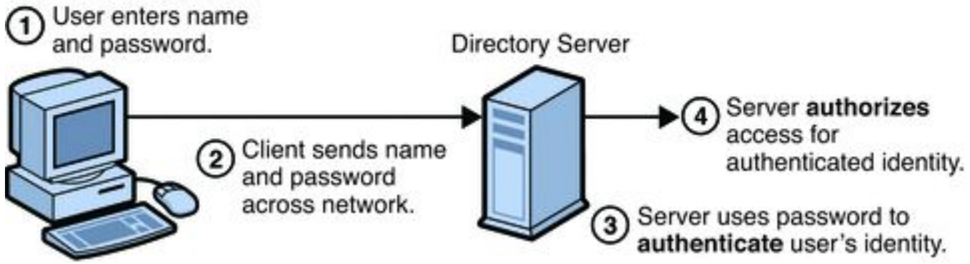


Password-Based Authentication

- Simplest form of *Authentication*
- User sends both username and password to the server, server authenticates user's identity.

What if the server is compromised (3/4)?

What if the network is compromised (2)?


1 User enters name and password.
Directory Server
2 Client sends name and password across network.
4 Server **authorizes** access for authenticated identity.
3 Server uses password to **authenticate** user's identity.

https://docs.oracle.com/cd/E19424-01/820-4811/gdzeq/index.html

Authentication Factors

Access control only makes sense if **subjects** are authenticated

There are 3 common flavors to authenticate subjects:

- Something you **know**: passwords, pin codes
- Something you **own**: paper card, smartphone, certificate, electronic token
- Something you **are**: biometrics

Two factor authentication (aka 2FA) requires the use of two factors

- A password is often one of the two factors

The Password Is Dead...

“They just don’t meet the challenge for anything you really want to secure.”

(Bill Gates, 2004)

“Within 5 years, you’ll never need a password again”

(IBM, 2011)

“Passwords are done at Google”

(H. Adkins, manager of Information Security at Google, 2013)

Truth: passwords are still the main form of authentication on the web

“No other single technology matches their combination of cost, immediacy and convenience”
(C. Herley, P. van Oorschot)

USERNAME
USER123
PASSWORD
HEY BRAIN, DO YOU REMEMBER MY PASSWORD FOR THIS?
I REMEMBER EVERY PASSWORD YOU'VE EVER CREATED!
SO WHICH ONE IS IT?
OWLTURD.COM

Even Strong Passwords Are at Risk

A stolen password can be replayed by anybody

An attacker can:

- Steal passwords using client-side malware
- Obtain passwords by cracking hashes stolen from a server (next lecture)
- Phish passwords with a fake website
- Eavesdrop the password (remember, that's why we need TLS)

Credential stuffing

- Lists of usernames and passwords are distributed online
- Hackers try the same credentials on many different sites
- Sony and Gawker were both hacked: "Two thirds of people with accounts at both Sony and Gawker reused their passwords" (source: Troy Hunt)

Password Managers

Store all your credentials in encrypted form

- Your **master password** is used to decrypt the credentials
- To be used on different devices, the encrypted credentials must be accessible online

Where are passwords stored?

- Password manager works **offline**, uses a local file (e.g., Keepass)
  - You can choose to host this file in the cloud
- Password manager talks to **a server in the cloud** (e.g., Lastpass)
  - Browser plug-in or app downloads credentials and decrypts them locally
  - You must trust them to not steal your master password
- Best of both: the password manager comes with an open source server that you can host where you want (e.g., Bitwarden)

Authentication/Session Cookies

After logging in a web application the server sends a cookie, stored in the client

The cookie is kept by the client and used for authenticating against the server

The cookie should be unique for every subject

The cookie can be used by illegal clients through forging

- If the server encrypts the cookie it limits forging (HMAC)

Something You Own

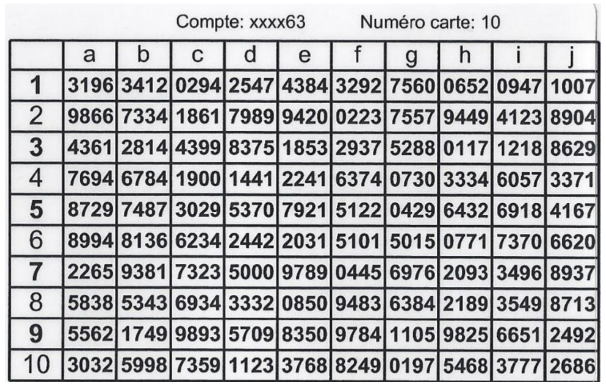
Typically called a (hardware/software) token

Bingo card

- Proves that the user owns the card
- Can be easily copied without being detected

One time password (OTP) token

- Displays a password to be used only once
- The password changes with time or click
- Proves that user owns token at time of login
- Cannot be copied easily (secure hardware)
- OATH standard



Compte: xxxx63 Numéro carte: 10

	a	b	c	d	e	f	g	h	i	j
1	3196	3412	0294	2547	4384	3292	7560	0652	0947	1007
2	9866	7334	1861	7989	9420	0223	7557	9449	4123	8904
3	4361	2814	4399	8375	1853	2937	5288	0117	1218	8629
4	7694	6784	1900	1441	2241	6374	0730	3334	6057	3371
5	8729	7487	3029	5370	7921	5122	0429	6432	6918	4167
6	8994	8136	6234	2442	2031	5101	5015	0771	7370	6620
7	2265	9381	7323	5000	9789	0445	6976	2093	3496	8937
8	5838	5343	6934	3332	0850	9483	6384	2189	3549	8713
9	5562	1749	9893	5709	8350	9784	1105	9825	6651	2492
10	3032	5998	7359	1123	3768	8249	0197	5468	3777	2686

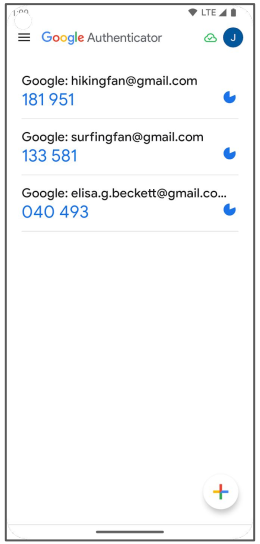
Something You Own

TAN generator

- A calculator that generates a number based on user input
- Can be used to sign a transaction (Transaction Authentication Number, TAN)
- Proves that user owns the generator
- Based on a smartcard (secure hardware), typically your bank card

Smartphone

- OTP is sent by SMS or
- OTP is generated by app, as with an OTP token
- Proof that user owns the phone (or sim card, secure hardware?)



Google: hikingfan@gmail.com
181 951

Google: surfingfan@gmail.com
133 581

Google: elisa.g.beckett@gmail.co...
040 493

Something You Own

A private key in a hardware token

- Token signs a challenge
- Different keys for different websites
- Proof that user owns the token
- Secure hardware
- Universal 2nd Factor (U2F) standard

OATH, Generation of OTP

OATH is a standard that describes:

- how OTPs are generated from a seed
- an XML format for importing the seeds into an authentication server

Standard OATH tokens exist in both hardware or software form

- E.g., the Google Authenticator app

Generation of the next OTP is either counter or time based (**RFC 4226**)

OATH Algorithms

Counter based:

$$\text{hotp}(k, c) = \text{truncate}(\text{hmac-sha512}(k, c))$$

Time-based (initial time $t0$, time interval x):

$$\text{totp}(k, t) = \text{hotp}(k, (t - t0)/x)$$

What are the tradeoffs between hotp and totp?

RSA SecureIDs

Secure IDs implement a form of totp

Requires a random seed to be synced between devices and customers

In a massive hack, attackers stole ALL seeds from RSA

The first true supply chain attack

https://www.wired.com/story/the-full-story-of-the-stunning-rsa-hack-can-finally-be-told/

U2F, FIDO2

Universal 2nd Factor is a standard developed by the **FIDO** alliance

Latest version is FIDO2

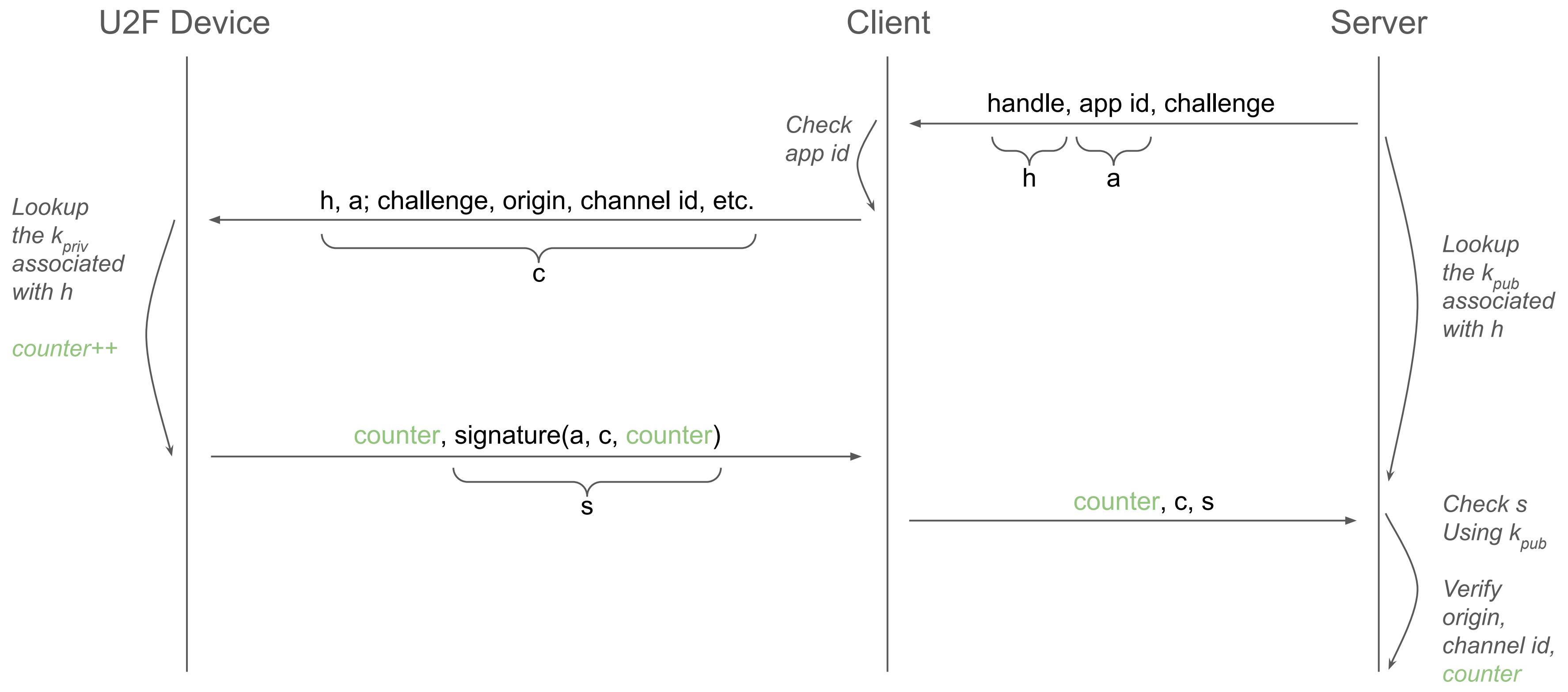
For each application, the token generates a key pair and gives the public key to the server

- On login, the server sends a random challenge to the client
- The client signs (the challenge + the domain name of the server + a signature counter)
- The client sends the data and signature to the server
- The server verifies the signature

Adding the domain prevents phishing attacks

Adding a counter detects cloning of the private key/replay attacks

U2F Authentication Workflow


U2F Device
Client
Server
handle, app id, challenge
h
a
Check app id
h, a; challenge, origin, channel id, etc.
c
Lookup the k_{priv} associated with h
counter++
counter, signature(a, c, counter)
s
Lookup the k_{pub} associated with h
counter, c, s
Check s Using k_{pub}
Verify origin, channel id, counter

https://developers.yubico.com/U2F/Protocol_details/Overview.html

U2F, FIDO2 (WebAuthn)

FIDO2 standardizes U2F within the browser with JavaScript (called WebAuthn)

Additionally makes use of **CTAP** to access the signature

CTAP, the Client to Authenticator Protocol, describes how an application can ask an authenticator to generate an assertion (signature)

- Authenticator can be a
  - USB token
  - Smartphone connected by Bluetooth
  - Authentication module of platform (e.g., phone’s biometric authentication)

The assertion contains information whether the user

- was present (e.g., clicked on a button)
- was verified (e.g., pin or fingerprint)

U2F, FIDO2 Pros

No problem if the server gets hacked:

- It is an asymmetric system. The information stored on the server cannot be used to log in

No problem if the client gets hacked

- The private key stays in secure hardware of the client
- Usage of the key is only possible with user interaction

Very convenient

- It can use the native authentication system of the platform (iPhone faceID, Microsoft Hello biometrics, Google fingerprint)

Biometrics

Motivation

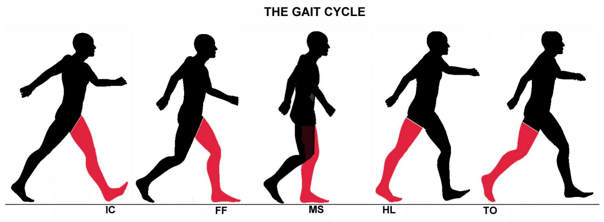
- Something you know could be guessed
- Something you own could be stolen
- Nothing to remember, nothing that can be lost

Physiological

- Iris
- Retina
- Fingerprint
- Shape of head
- Shape of hand

Behavioral

- Speech
- Keystroke timing
- Handwritten signature on tablet
- Gait



Biometrics: Authentication Process

Registration

- Acquisition
- Extraction of characteristics
- Storage of characteristics

Authentication

- Acquisition
- Comparison
- Decision

Limitations

- Acquisition is never exact
- Comparison is never a perfect match
  - Biometric information cannot be hashed (due to imprecision in reading out data)
- Decision is always error prone

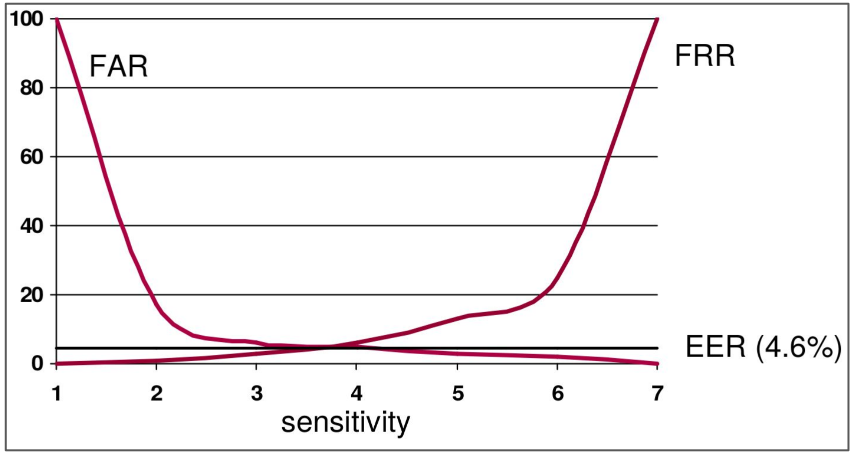
Biometrics: Error Rates

The decision algorithm must accept a certain error; sensitivity can be tuned

FAR False acceptance rate: the system declares a match when it wasn't

FRR False rejection rate: the system declares a non-match although it was a match

EER Equal error rate, when the system is tuned such that FAR=FRR



Biometrics: Fingerprints

The fingerprint is read by a sensor

An image of the ridges is created

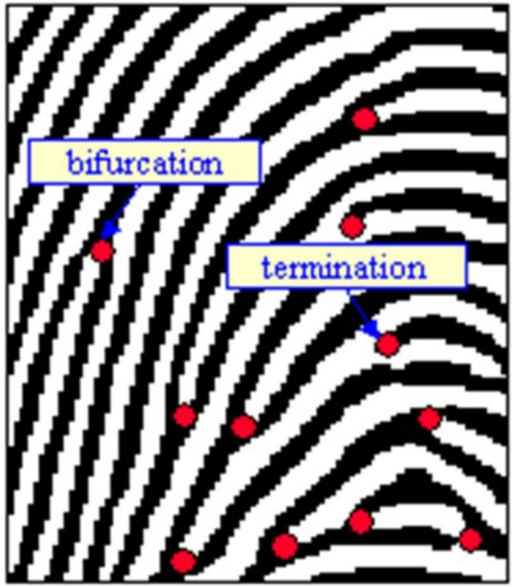
The **minutiae** are extracted at

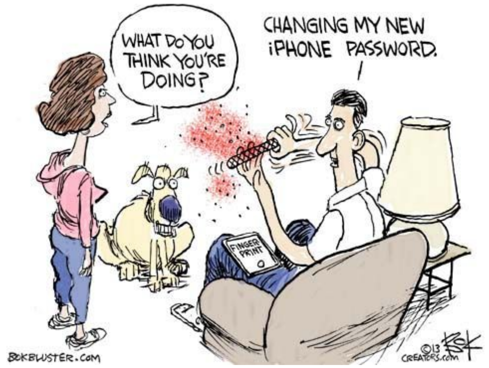
- termination of a ridge
- bifurcation of a ridge

A list of coordinates (x,y) and angles is generated

The list is compared with a stored list

- the minutiae are shifted and rotated to get the best match

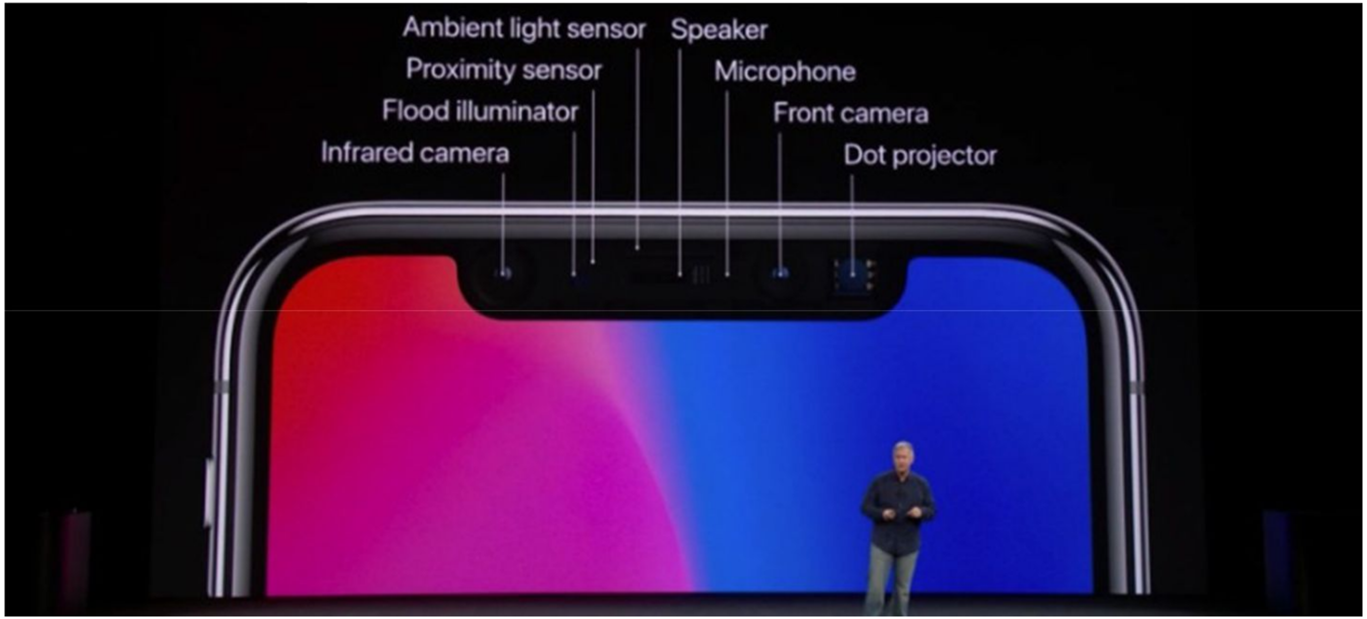



WHAT DO YOU THINK YOU'RE DOING?
CHANGING MY NEW iPHONE PASSWORD.
FINGERPRINT
BOKBLUSTER.COM
©13 CREATORS.com

Biometrics: FaceID

30,000 points are projected on the face and read from a different angle

A specialized and isolated processor constructs a 3D image and compares it to the registered face



Biometrics: FaceID

A specialized and isolated processor is used to store and compare images

- The main processor never sees the 3D data

The communication with the sensor (camera) is encrypted

- cannot be intercepted and replayed

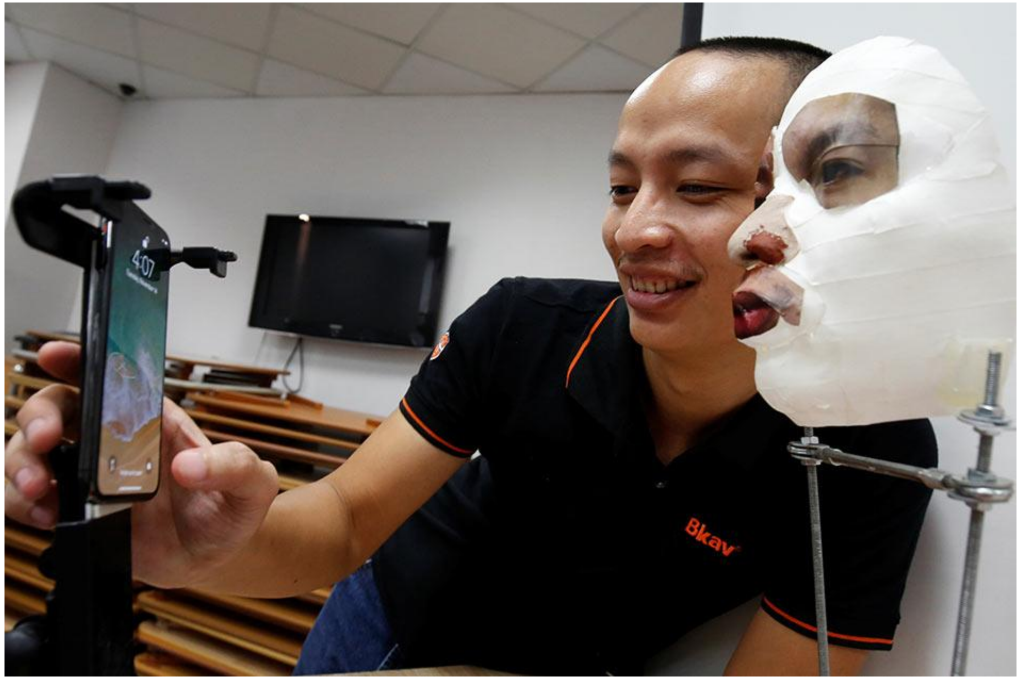
This is the same for iPhone fingerprint scanning (Touch ID)

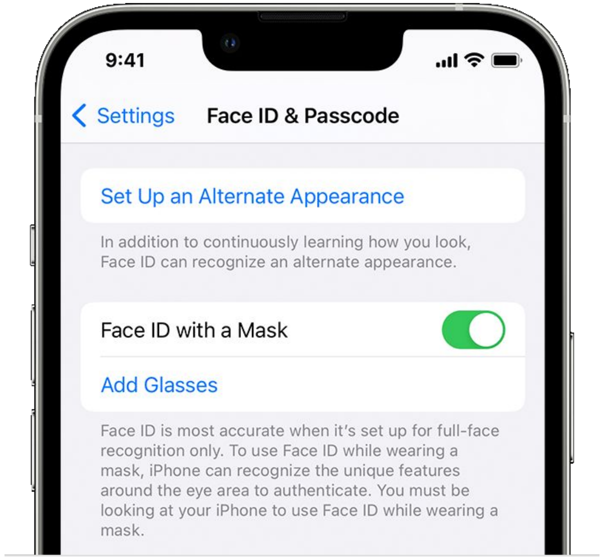
- this is why you can’t just replace a broken sensor
- it has to be paired with the processor

Note: you still need to type a password when booting the phone

- used to decrypt the storage

Biometrics: FaceID Limitations





Face ID & Passcode

Set Up an Alternate Appearance

In addition to continuously learning how you look, Face ID can recognize an alternate appearance.

Face ID with a Mask [✓ On]

Add Glasses

Face ID is most accurate when it's set up for full-face recognition only. To use Face ID while wearing a mask, iPhone can recognize the unique features around the eye area to authenticate. You must be looking at your iPhone to use Face ID while wearing a mask.

Biometrics Discussion

Biometrics and authentication

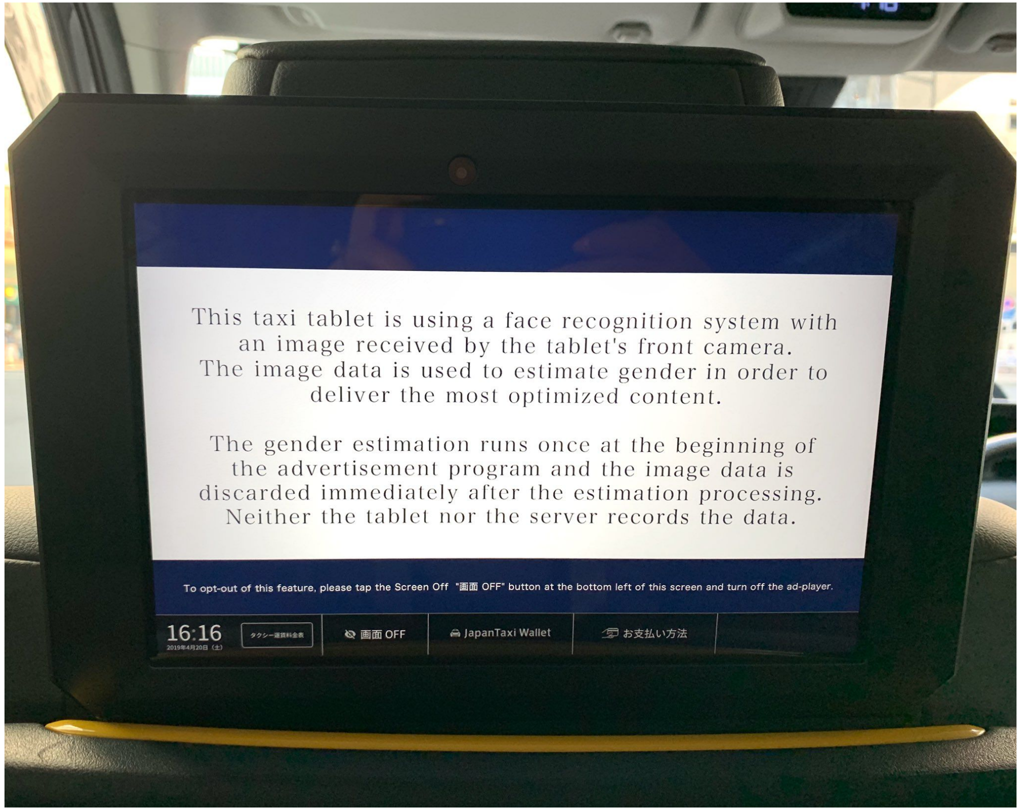
- No hashing possible (neither for transmission nor for storage), risk: theft
- It is impossible to change a stolen fingerprint!
- Best to store the biometric data locally in protected hardware
- Example:
  - Biometrics are stored in passports, not at the customs offices
  - Smartphones store fingerprint data in separate, secure hardware
- Some sensors can be fooled or replaced
- Not ideal for remote authentication
  - Rather use it for local access to authentication key (e.g. U2F)

Biometrics Discussion

Biometrics and privacy

- Biometric data is considered sensitive data by European data protection laws and the Swiss law
- Biometric data can reveal health issues (e.g., eye pathologies)
- Biometric data can exclude some people (absence of fingerprints or fingers)
- A lot of more or less serious research is done with ML and AI to extract information from faces
  - Detection of Propensity for Aggression based on Facial Structure
  - Privacy fears as Tokyo taxis use facial recognition cameras to guess riders' age and gender for targeted advertisements

Biometrics Discussion



This taxi tablet is using a face recognition system with an image received by the tablet's front camera.
The image data is used to estimate gender in order to deliver the most optimized content.

The gender estimation runs once at the beginning of the advertisement program and the image data is discarded immediately after the estimation processing.
Neither the tablet nor the server records the data.

To opt-out of this feature, please tap the Screen Off "画面 OFF" button at the bottom left of this screen and turn off the ad-player.

Source: Twitter @rosa

Access control

Access Control Basics

After we **authenticate** a subject…

Access control *defines and enforces* ***operations*** *that* ***subjects*** *can do on* ***objects***

- Bob (subject) has permission to read/write (operation) from a socket (object)
- Implies that the subject has been authenticated first

Access Control Basics

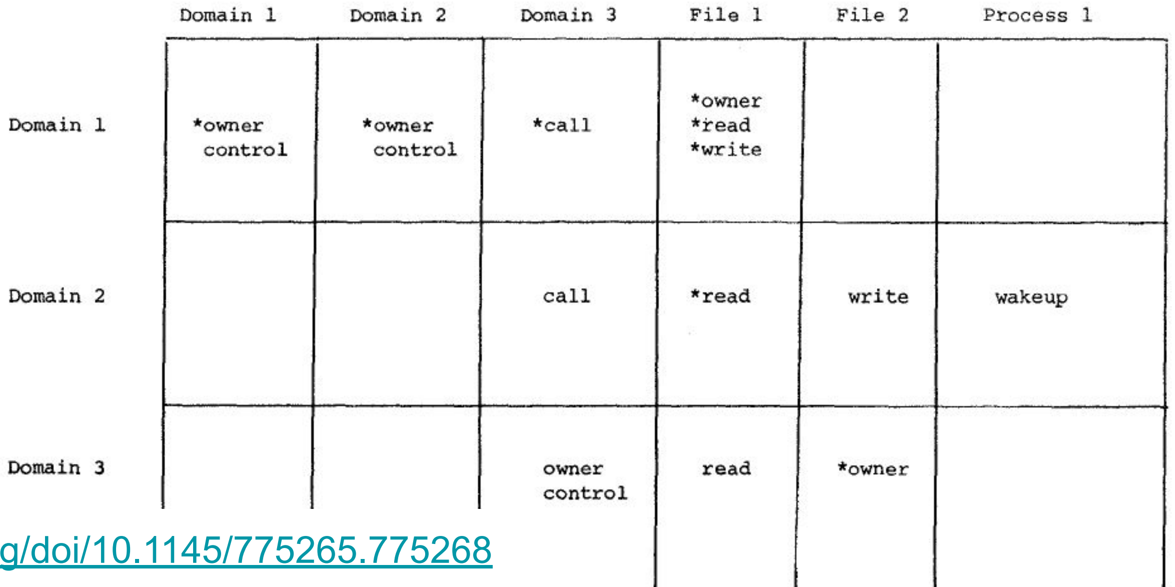
Access control defines and enforces **operations** that **subjects** can do on **objects**

- Bob (subject) has permission to read/write (operation) from a socket (object)
- Implies that the subject has been authenticated first

Security Policy

Access rights (aka permissions, privileges) describe which subjects can do what operations on what objects

A **security policy** is a collection of access rights. Security policies can be represented as an **access control matrix**



	Domain 1	Domain 2	Domain 3	File 1	File 2	Process 1
Domain 1	*owner control	*owner control	*call	*owner *read *write		
Domain 2			call	*read	write	wakeup
Domain 3			owner control	read	*owner	

https://dl.acm.org/doi/10.1145/775265.775268

Security Mechanism

Security mechanisms try to prevent operations that are not authorized by the security policy

For example, the kernel implements an access check to see if a user has permissions to open a file. If the access check fails, the rights are not granted.

Principle of Least Privilege (PoLP)

- Subjects only have the minimum rights required for their job
- I.e., subjects are only allowed minimal operations on objects per task
- This limits the impact if anything should go wrong

The challenge in access control is to have a system that is simple to implement and manage and that is close to the principle of least privilege

There is no “one size fits all” solution and often different approaches to access control are combined to achieve the best results

Most important
access control principle

Multiple Levels of Access Control

Network level access control

- Subjects are connections or data packets
- They are identified by source/destination IP addresses and protocol ports
- Typical operations are pass, block, or tag
- Example: a database server only accepts traffic from inside EPFL (source addr `128.*.*.*`) connecting to TCP port `3306` (`mysql`)

Typically enforced with

- Network equipment: firewalls
- The servers:
  - Local firewall on the server (e.g., `ufw` in Linux)
  - Configuration of the server software

What are the tradeoffs of filtering at the different levels?

Multiple (Confusing?) Levels of Access Control

Access control at the operating system

- Which user can start/stop the DB engine?
- Who can read/modify the files of the DB?

Access control in the application

- Which user of the application can edit user profiles?
- Who can see financial data?

Access control within the enterprise

- Which employees can access the application?
- Which applications are limited to human resources, which to marketing?

Multiple Approaches to Access Control

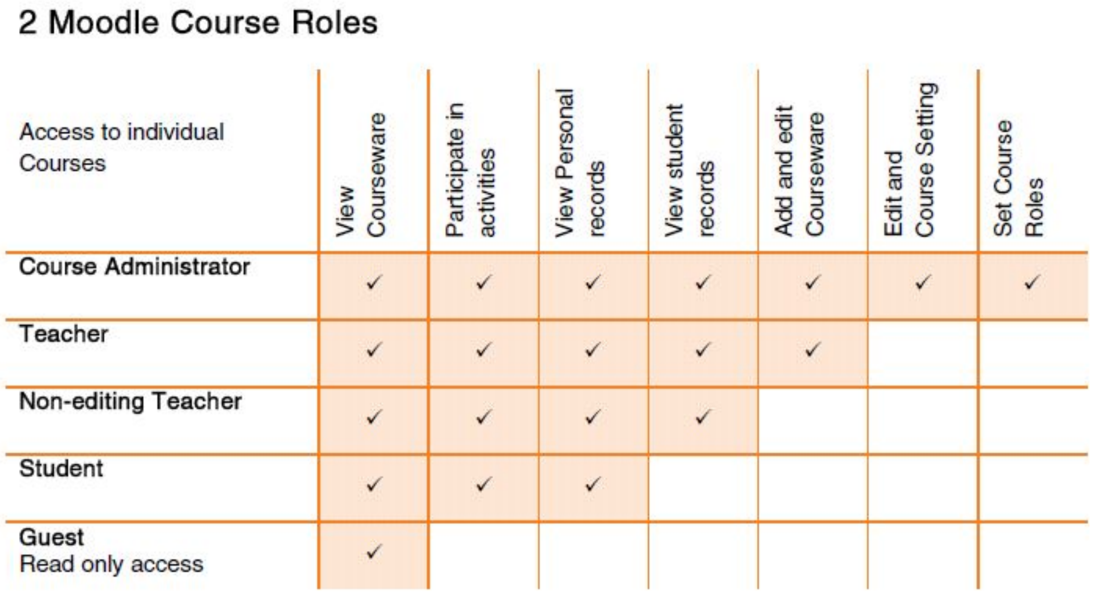
Three common varieties:

- Role-based Access Control (RBAC)
- Discretionary Access Control (DAC)
- Mandatory Access Control (MAC) → not message authentication code!

Role-based Access Control (RBAC)

Simplifies the specification of permissions by grouping users into roles

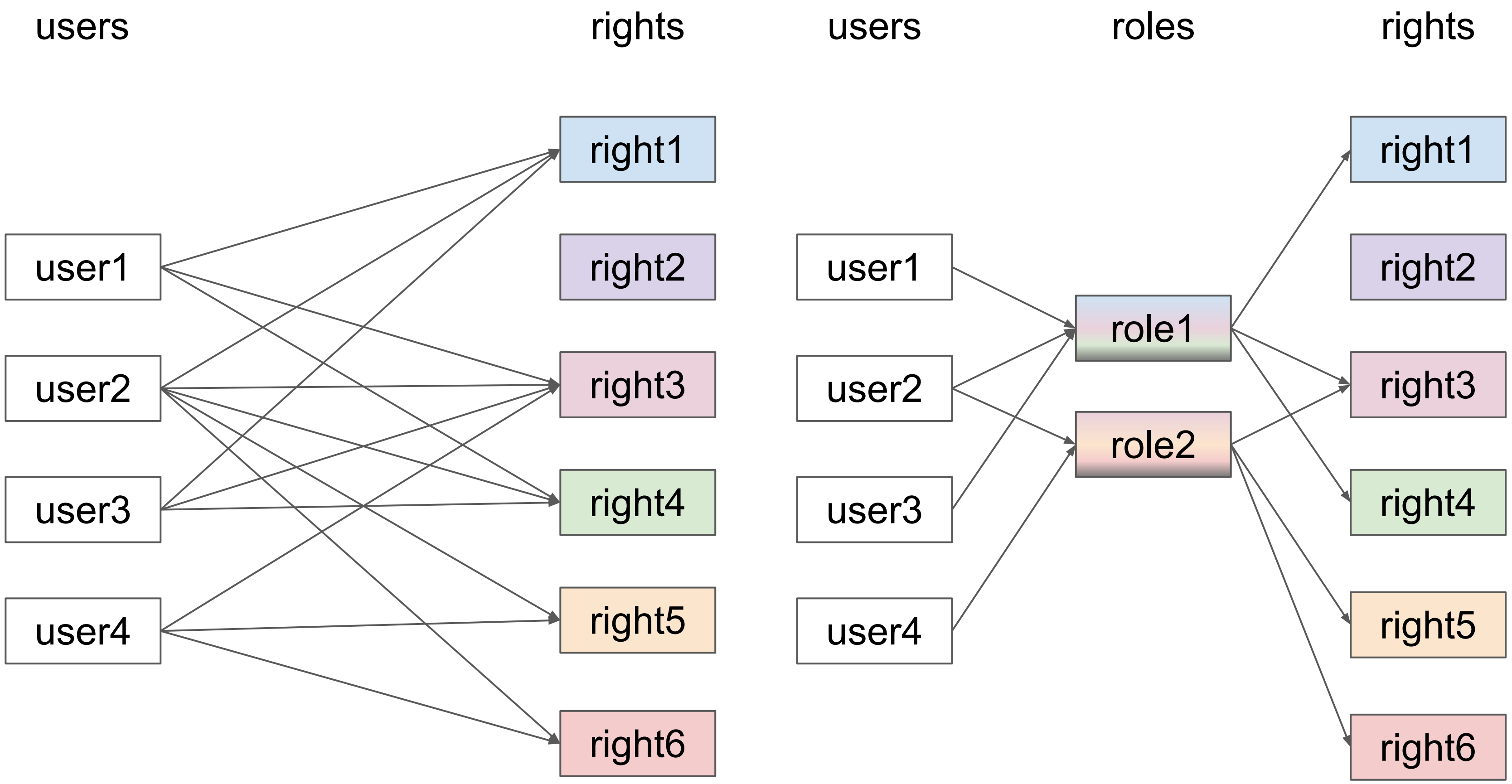
Centered on user roles (a role can contain multiple permissions)



2 Moodle Course Roles

Access to individual Courses	View Courseware	Participate in activities	View Personal records	View student records	Add and edit Courseware	Edit and Course Setting	Set Course Roles
Course Administrator	✓	✓	✓	✓	✓	✓	✓
Teacher	✓	✓	✓	✓	✓		
Non-editing Teacher	✓	✓	✓	✓			
Student	✓	✓	✓				
Guest Read only access	✓						

RBAC Example: Roles Simplify Management



RBAC Implementation

Example: Operating Systems

- Most operating systems have the notion of groups
- Groups can be given a set of permissions
- Users can be added to groups
- Examples:
    - Debian/Ubuntu: audio group can access mic and loudspeakers, wireshark group can sniff network traffic
    - Windows: "remote desktop user" group can access desktop remotely

Which groups does your user belong to?

```
$ id gannimo
uid=1000(gannimo) gid=1000(gannimo)
groups=1000(gannimo),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),108(netdev),
113(bluetooth),117(lpadmin),120(scanner)
```

RBAC Pros

Easy to grasp the idea of roles

Easy to manage

- Roles decouple digital entities from permissions
- Simply assign roles to a new subject
  - No need to decide for each object
- Easy to revoke authorizations by removing role

Easy to tell through roles which permissions a subject has and why

- Typically centrally managed

RBAC Cons

Difficult to decide on the granularity of roles

- Do we need different roles for modifying and deleting client information?
- Leads either to role explosion or roles that are too broad (not least privilege)

Role meaning is fuzzy

- Employee position in company may be different from RBAC role (think developers in same team working on different subjects)

Unclear if roles can be shared across different departments

- For “manager” roles, in general, is a finance IT manager the same as a marketing IT manager?

Discretionary Access Control (DAC)

Access control is at the discretion of the object owner

- owner specifies policies to access resources it owns

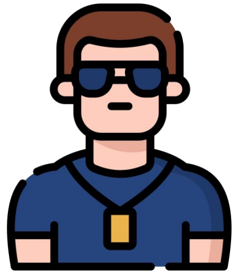
Access control matrix represents rules

- stored by **column**: Access Control List (ACL) stored with object
- stored by **row**: capabilities stored with subjects

	/stud/grades.txt	/hw1/grade.sh	/sensitive
stud1	r–	–x	–
TA1	rw-	rwx	r-x

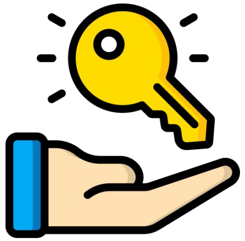
ACL vs Capabilities

Think of a door protected by a bouncer vs. a door protected by a lock



ACL (bouncer, tied to object):

- The bouncer knows exactly who can get in
- People don't know where they can get in and where they can't



Capabilities (key, tied to subject):

- Doors don't know who will show up with a key
- People know exactly for which doors they have a key

ACL is practical when you often have to create or modify rights on objects

Capabilities, when you often create or change rights of subjects or roles

DAC in Unix file systems

Typically done with ACL

Stored **in the target object**, e.g., in the metadata of files in the file system

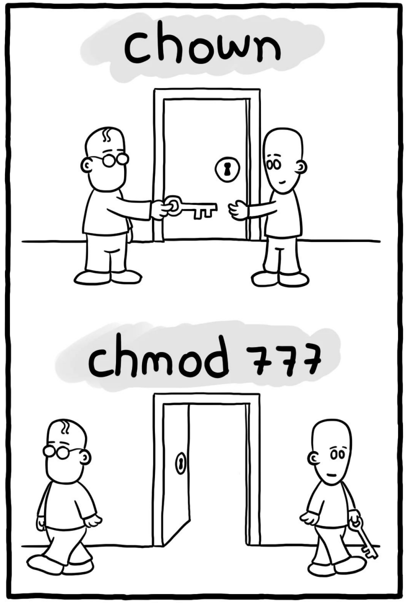
Subjects are grouped in three categories : owner, group, other.

Three access rights: (r)ead, (w)rite, e(x)ecute

- For directories:
  - r: directory can be listed
  - w: directory can be modified (create, delete, rename files)
  - x: directory can be accessed by the `cd` command

The three rights and three groups are stored in 9 bits

- represented as three octal digits
- owner `rwx`, group `rx`, others `r`: `rwx|r-x|r--` = 754
- the permissions of the first matching category dominate



Daniel Stori {turnoff.us}

ACLs in Unix: Example

Remember the Ubuntu local “uncomplicated” firewall GUI gufw:

- Only root can read or write the files and the directory:

```
$ ls -l /etc/gufw/
total 44
drwxr-xr-x 2 root root 28672 avr 24 08:17 app_profiles
-rw------- 1 root root 73 avr 23 15:10 gufw.cfg
-rw------- 1 root root 1079 avr 23 11:09 Home.profile
-rw------- 1 root root 76 avr 18 11:40 Office.profile
-rw------- 1 root root 78 avr 18 11:40 Public.profile
```

- Everybody can read the application profiles, but only root can modify them:

```
$ ls -l /etc/gufw/app_profiles/ssh.gufw_service
-rw-r--r-- 1 root root 213 mai 24 2017 gufw/app_profiles/ssh.gufw_service
```

ACLs in Unix: `setuid/setgid`

If a program has setuid bit set, it will be run with the permissions of the owner of the file instead of the permissions of the user running the program

Very useful to give users more privileges in specific cases

Example: the passwd command allows a user to modify the `/etc/passwd` and `/etc/shadow` files

- passwd can be read by all but only modified by root
- shadow can be read by root and shadow and modified by root

```
$ ls -l /etc/{passwd,shadow}
-rw-r--r-- 1 root root 2786 avr 23 11:09 /etc/passwd
-rw-r----- 1 root shadow 1489 avr 23 11:09 /etc/shadow
```

- the program /usr/bin/passwd has the setuid bit

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 59640 jan 25 2018 /usr/bin/passwd
```

ACLs in Unix: `setuid/setgid`

When user Jane runs the program passwd, the process is run as root:

```
$ ps -ef | grep passwd
root 16003 26651 0 09:26 pts/1 00:00:00 passwd
```

The setgid bit does the same for groups: the group of the process running the program is set to the group of the owner of the program

setuid and setgid is displayed as s instead of x in the access rights of the file

Example: this program has both setuid and setgid bits set:

```
$ ls -l ~/test
-rwsrwsr-x 1 user user 0 Sep 25 08:36 test
```

ACLs in Unix: `setuid/setgid`

Finding files that have the setuid bit set:

```
$ find / -perm /u=s 2>/dev/null
/bin/fusermount
/bin/mount
/bin/su
/bin/umount
/bin/ping
...
```

setuid is very practical, because it lets unprivileged users execute some well defined privileged actions

Quiz!

Setuid can be very dangerous, why?

Programs with setuid bit set are privileged. Any bug in these programs could be used to escalate the privileges of the user. As the user controls input to the program running as root, they may use that input to trigger the bug and gain arbitrary code execution.

Capabilities in Linux

Capabilities are permissions that are related to a subject, not to an object

Linux supports capabilities for processes. Some examples are:

- `CAP_CHOWN`: make arbitrary changes to file user ID and group ID
- `CAP_DAC_OVERRIDE`: Bypass file read, write, and execute permission checks
- `CAP_SYS_BOOT`: use reboot or kexec (load a new kernel)

Example: dumpcap is the program used by wireshark to sniff network traffic

- It can only be run by user root and members of the wireshark group
- It does not have the setuid bit

```
$ ls -l /usr/bin/dumpcap
-rwxr-xr-- 1 root wireshark 104688 jan 19 06:23 /usr/bin/dumpcap
```

Capabilities in Linux

Let's check the capabilities of dumpcap:

```
$ sudo getcap /usr/bin/dumpcap
/usr/bin/dumpcap = cap_net_admin,cap_net_raw+eip
```

The process assumes `net_admin` and `net_raw` capabilities when launching `dumpcap`: it can read and write to all network interfaces

The program can do this while running in the name of the user:

```
$ ps -ef | grep dumpcap
user 24342 26651 0 13:10 pts/1 00:00:00 /usr/bin/dumpcap
```

This is much safer than using setuid

If there was a bug in dumpcap allowing to execute arbitrary commands, setuid would run these commands as root!

DAC Pros and Cons

Pros

- Flexible
- Intuitive
- Easy to manage (owners get to set the permissions themselves)

Cons

- Depends on the owners judgment
- Only works if programs are benign and users make no mistakes
- Vulnerable to the "Trojan" / declassification problem*

✓ ✗

*A malicious program run by an authorized user can read a protected file and write an unprotected copy of that file. Anybody can now read the file.

Mandatory Access Control

Tries to ensure that even someone with access cannot leak the data

Historically associated with military-grade information security

- Multilevel security: e.g., unclassified, confidential, secret, top-secret

The system labels both subjects and objects with security labels

- Can only be modified by trusted administrators via trusted software

Security policy:

- Example: Subjects can only access objects of the same or lower level

subjects\objects	top-secret	secret	confidential	unclassified
top-secret	read, write	read	read	read
secret		read,write	read	read
confidential			read, write	read
unclassified				read, write

Mandatory Access Control

Depends on trusted software and admins for

- Keeping the system in a protected state, by preventing operations that violate the rules of the matrix
- Labeling new subjects and objects
- Perform transitions of labels (e.g., when a document is declassified)

Can be used in conjunction with DAC or RBAC

Quiz!

Why is it a bad idea to allow secret subjects to write confidential documents?

Remember the order of “secrecy”: unclassified, confidential, secret, top-secret.

A secret subject may read a secret file and write the contents into a confidential file, thereby reclassifying the secrecy level and downgrading it.

This is also called the “no-write-down” problem.

MAC Confidentiality vs Integrity

MAC and Confidentiality

- When protecting confidentiality, we don’t want users to write to a lower level (**no write-down**)
    - Prevents leaking information from higher to lower levels ("trojan")
- Typical scenario: network access control
    - Network split in zones: internet, internal, secret
    - Firewalls only allow data to flow from lower zones to higher zones

MAC and Integrity

- We don’t want users from lower levels to write into higher levels (**no write-up**)
    - Prevents unauthorized modification of objects
- Typical scenarios: operating systems
    - Users can read and execute OS programs, but they cannot modify them

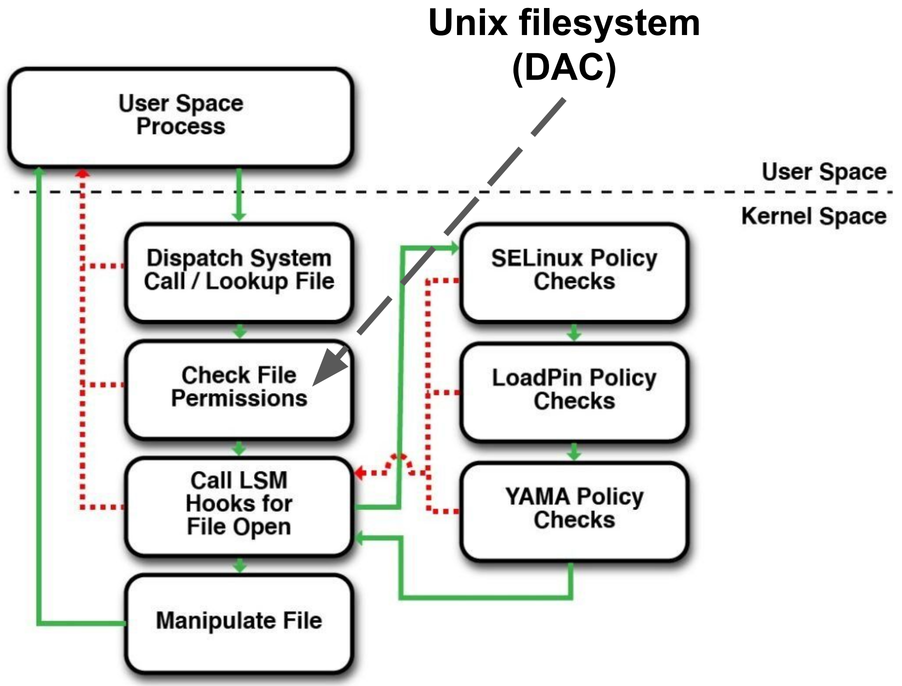
MAC Linux Examples

SELinux and AppArmor are two MAC systems for Linux

They are both based on the generic Linux Security Module (LSM)

LSM sits in the kernel and is called just after standard DAC checks and before access is given

LSM allows implementation of secondary security policies



A Brief Tour of Linux Security Modules - starlab.io

MAC Linux Examples: SELinux

- Every user has a context made of name, role, and domain
- Files, ports and other objects can be labeled with name, role and type
- Rules can be defined to allow certain actions

SELinux is leveraged in Android: better isolation of apps and generic services

MAC Linux Examples: AppArmor

- Also based on LSM
- Uses profiles to define access rights to files, network and capabilities
- There are no labels or security levels
- Profiles basically define the same rules that can be defined with DAC, but they cannot be modified at the discretion of the owner of the objects or subjects
- Profiles can be generated by observing a running application
- AppArmor is enabled by default in Ubuntu

AppArmor demo: copy a PDF file into your `.ssh` directory and try to open it with evince!

MAC Pros and Cons

Pros

- Addresses the limitations of DAC
- Easy to scale

Cons

- Can be too restrictive, prevent legitimate tasks
- Not flexible

Summary of Access Control

Different types of access control (RBAC, DAC, MAC), usage depends on situation

- Aim: achieve least privilege at minimal complexity

Modern OSes make use of all of these types

- DAC with ACLs for files and most objects
- DAC with capabilities for privileged operations
- Using groups to implement RBAC (users, admins, hr, marketing)
- MAC for protecting the integrity of the system

Authentication protocols

When sending a password over TLS is not enough

Motivation

Why is the naive Password-based Authentication insufficient?

- Sending passwords over network is risky, leaks to adversary when the network is compromised
- Attacker may break encryption
- Server may be compromised or even store passwords too
- Challenge-response Authentication

Why do we need authentication protocols?

- Standardization and save duplicated efforts implementing authentication.
- Kerberos

Challenge-Response

Rather than sending the password to the server

- The server sends a random challenge to the client
- The client uses the password hash to create a response
  - E.g., encryption or HMAC of the challenge

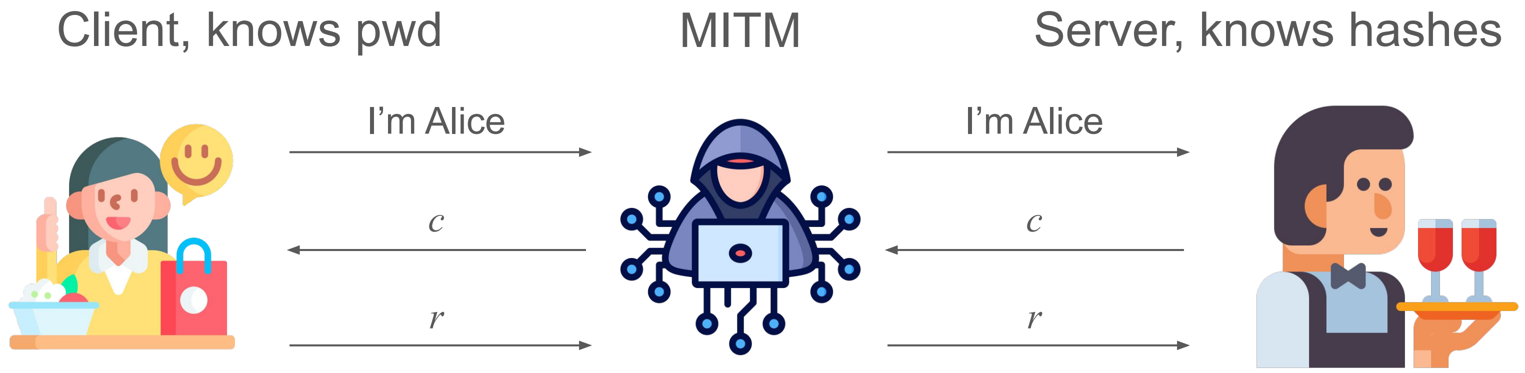
Example, Microsoft ad-hoc networking with SMBv1 (not Kerberos)

Client, knows pwd　　　　　　　　Server, knows hashes

$$\begin{tikzcd}
NTHash = \mathrm{MD4}(pwd) \arrow[r, "\text{I'm Alice}"] & \text{Lookup } NTHash \text{ of Alice} \\
\text{Calculate response} & \text{Choose random challenge } c \arrow[l, "c"'] \\
r = \mathrm{ENC}_{NTHash}(c) \arrow[r, "r"] & r \mathrel{?=} \mathrm{ENC}_{NTHash}(c)
\end{tikzcd}$$

Challenge-Response

Typically, challenge response protocols are vulnerable to MITM attacks



Microsoft introduced the signature of the packets with a key derived from the pwd hash (it is actually an HMAC)

- The MITM does not know the key, cannot send any packets

Challenge-Response: Mutual Authentication

The server and client can both use a challenge: they can authenticate each other

Examples:

- Newer versions of Microsoft challenge-response (SMBv2)
- WiFi WPA, WPA2, and WPA3

Challenge-response protocols may be eavesdropped or suffer cracking attacks

- The attacker records a challenge and a response
- They try all possible passwords to find which would yield the same response

Kerberos

Kerberos provides authentication and authorization across a network

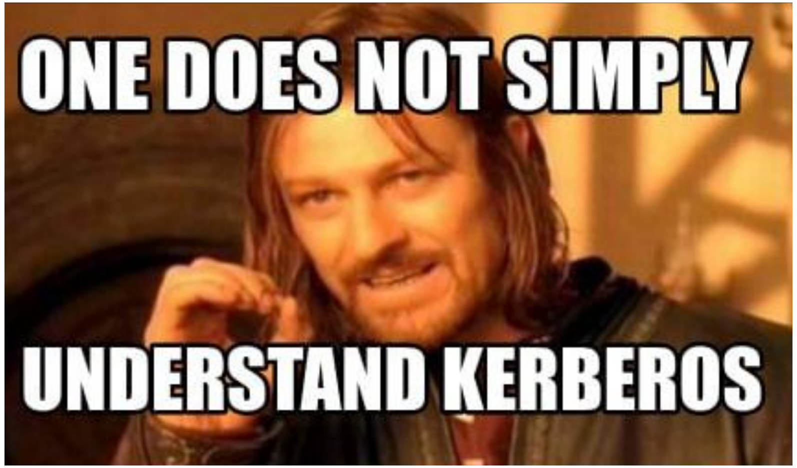
Subjects receive tickets that they use to access objects

Exclusively based on symmetric keys

Developed at MIT in the 80’s

Initially deployed in Unix

The main authentication protocol in Windows LAN networks



Motivation

Key idea: delegation

- Users authenticate once and then access **multiple services** without needing to repeatedly re-enter credentials

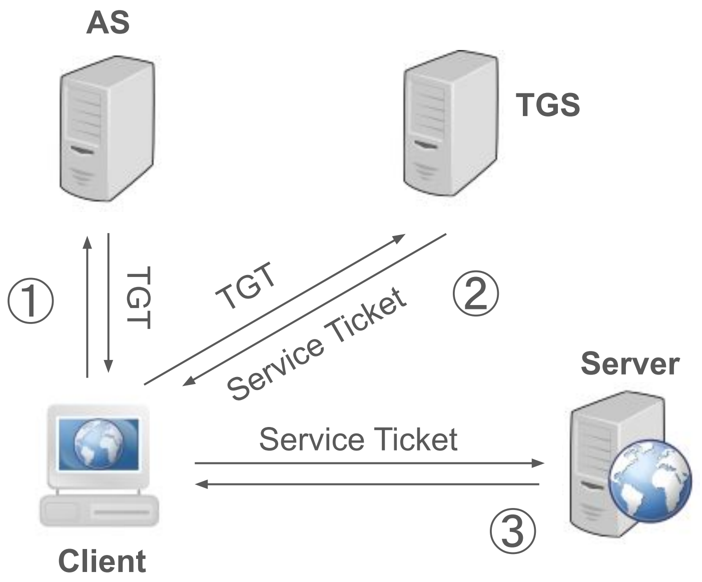
Separation of concerns

- Authentication
- Access control (Authorization)
- Providing actual service

Overview

Kerberos uses a three-phase approach

①: an authentication server (AS) authenticates the client and delivers a ticket granting ticket (TGT)
②: the client can then present the TGT to the ticket granting server (TGS) to get a service-specific ticket
③: the client can access the service



Role of AS, and TGS

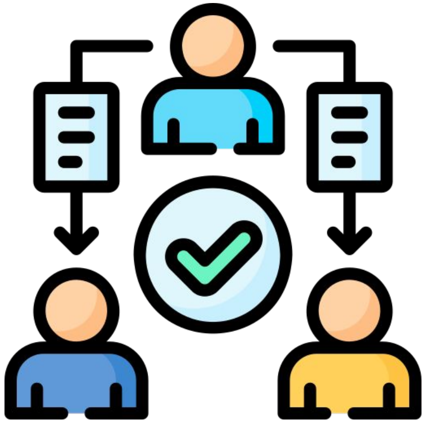
The **AS** handles **authentication** (user identity verification and issuing the TGT).

- This step is solely about **confirming** the user's **identity**.

The **TGS** handles **access control** (authorization, issuing service-specific tickets based on the TGT)

- Once the TGT is issued by the AS, the user doesn’t need to go back to the AS for each service access. Instead, the TGT can be used multiple times to request service-specific tickets from the TGS.

Delegated authentication



OAUTH2

Kerberos is a great but targets the intranet. Following the idea of delegated auth…

Oauth2 (RFC 6749) is used for delegated authentication on the **Internet**

- Oauth2 providers like Facebook, Google, or X/Twitter can be used to authenticate and access other applications

For example, you can log in to Pinterest with Facebook or Google

You can also authorize Pinterest to access your photos on Facebook

Remember TLSv1.2 vs 1.3? OAUTH2 is much simpler than Kerberos

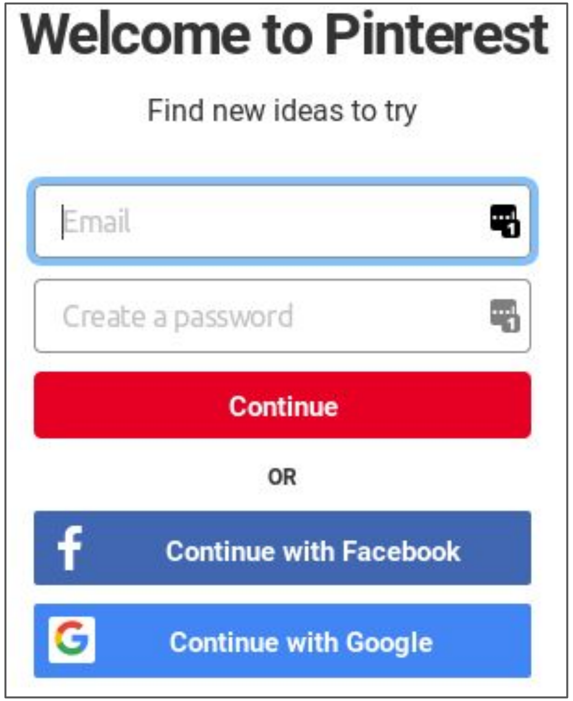
Roles

Client: application that wants to use authentication and possibly access the user's data (Pinterest)

Resource server: server that has user's data that client wants to use (Facebook's server containing the user's photos)

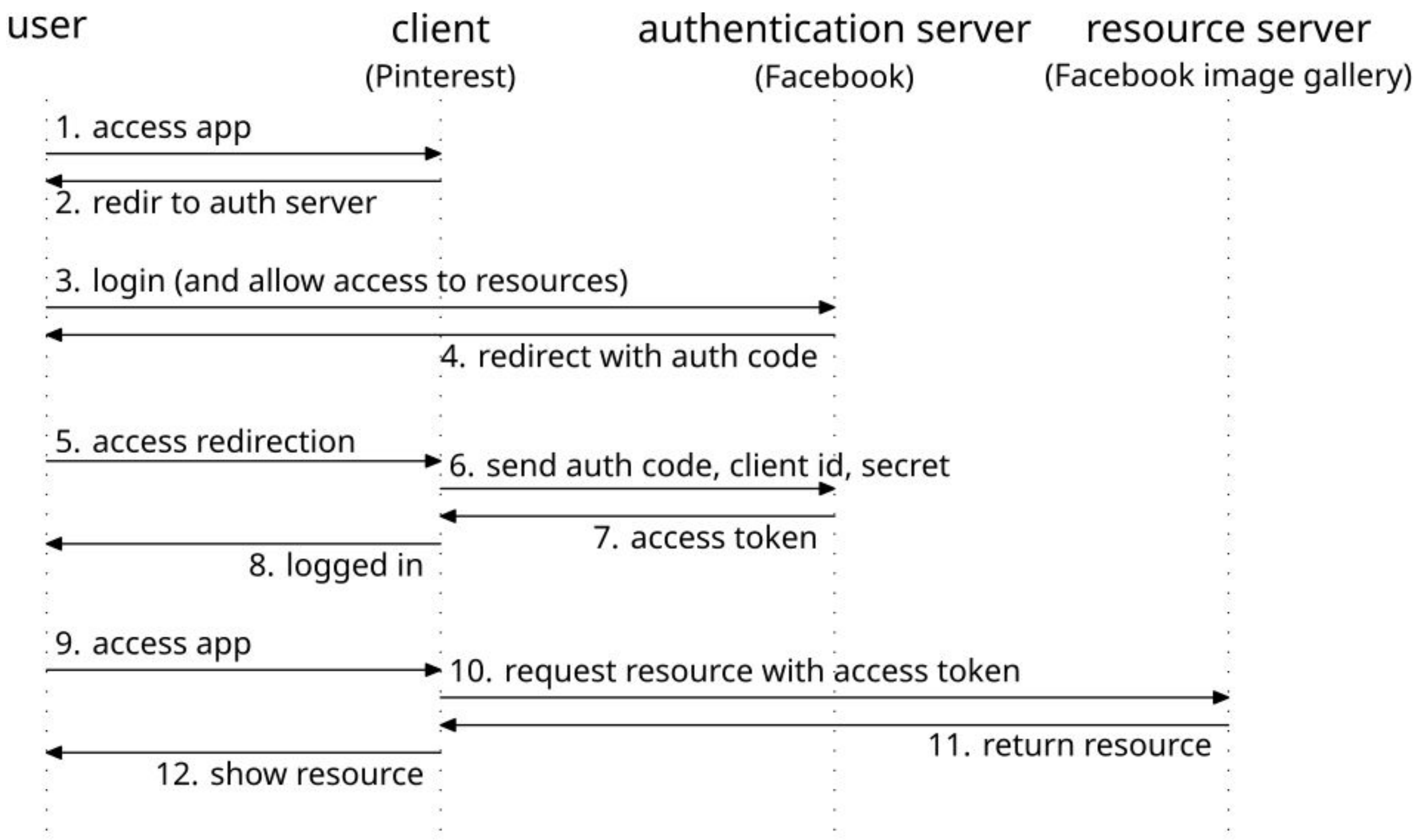
Authorization server: server on which the user authenticates (Facebook authentication server)

User: Owner of the account and resources on resource server (wants to log into Pinterest with his Facebook account)


Welcome to Pinterest

Find new ideas to try

Typical Oauth2 Flow



Typical Oauth2 Flow

The client and the authentication server have a shared secret

- the client thus has to register with the authentication server before being able to offer this service
- the secret is used when the client exchanges the **authentication code** for an **access token**
- the authentication code is not sufficient to get access to the resources
- It can only be used by the client and nobody else

Oauth2 can be used by browsers or in apps

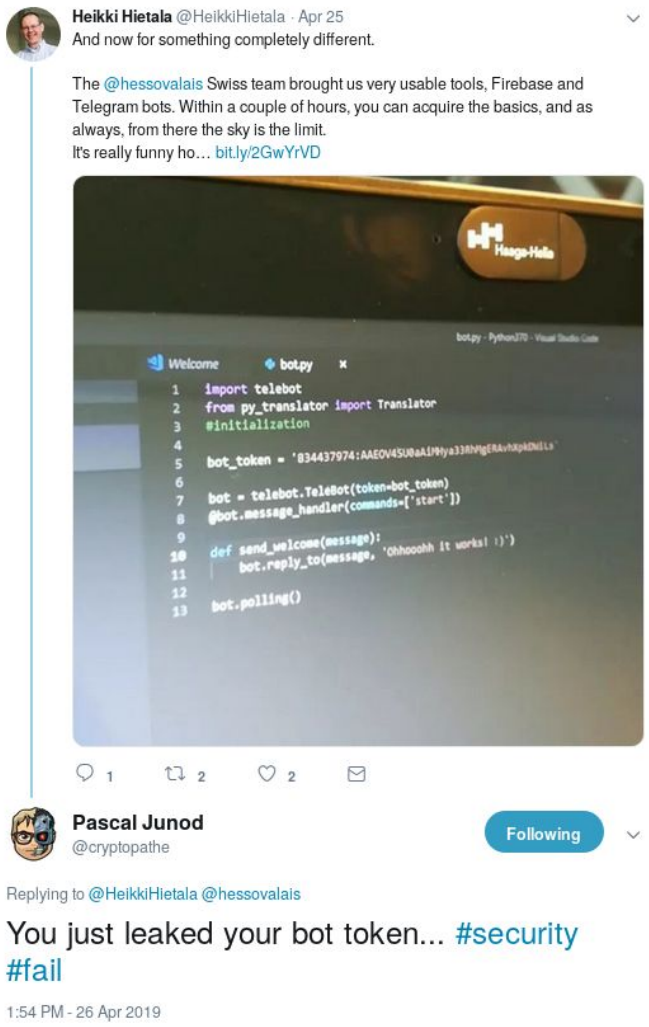
- in an app a redirection for authentication can be either
  - opening a browser within the app (called a webview)
    - not very safe as the app could be spying while you login
  - switching to the other app (e.g. facebook) and then back

Oauth2 Authentication Only

If Oauth is only used for logging in, then the flow can stop after message 8

Most apps in smartphones (e.g., X, Instagram, Gmail) do not store your passwords

- They use Oauth2 to request an access token and use it
- When you change your password you don't need to type your new password into all your devices



Heikki Hietala @HeikkiHietala · Apr 25
And now for something completely different.

The @hessovalais Swiss team brought us very usable tools, Firebase and Telegram bots. Within a couple of hours, you can acquire the basics, and as always, from there the sky is the limit.
It's really funny ho… bit.ly/2GwYrVD

```
import telebot
from py_translator import Translator
#initialization

bot_token = '834437974:AAEOV4SU0aAiMHya33RhMgERAvhXpkDWiLs'

bot = telebot.TeleBot(token=bot_token)
@bot.message_handler(commands=['start'])

def send_welcome(message):
    bot.reply_to(message, 'Ohhooohh it works! :)')

bot.polling()
```

Pascal Junod
@cryptopathe

Replying to @HeikkiHietala @hessovalais

You just leaked your bot token... #security #fail

1:54 PM - 26 Apr 2019

Authentication Summary

Passwords can go a long way, especially if you use a password manager

For critical accounts, 2FA significantly raises the bar for attackers

- U2F is secure and user-friendly

Challenge-response protocols authenticate a user without sending the password

- can be vulnerable to MITM, in particular if there is no mutual authentication

Kerberos uses tickets to authenticate users across a network

- Authentication is separated from authorization

Oauth is used to delegate authentication on the Internet

- it has no crypto at all, relies on communications being made over TLS

Challenge-response protocols we have seen and Kerberos use symmetric crypto

For the Homework

You will play with cookie for authentication

- Cookie tampering (attack)
- HMAC for cookies (defense)

You will get confident with the OTP algorithm internals

- HOTP algorithm, based on HMAC algorithm
- TOTP algorithm, an extension of the HOTP algorithm

In the Upcoming Lecture…

How does Access Control apply to data storage?

- Server level
- Application level
- Data level
- Network level

How can we securely store passwords?

- Salt
- Memory hard functions

How to verify a password and exchange a key?

- Password Authenticated Key Exchange (PAKE)

Summary

Access Control has multiple approaches depending on the policy definition

Security mechanisms prevent the violation of the policy

Authentication lets subjects to identify themselves via something they own/know/are

Authentication protocols let a user authenticate in a network without sending their passwords in cleartext

Authentication can be delegated to third parties