Security and Privacy

Authentication & authentication protocols

30.04.2019

slide credits: Cristina Basescu, Ph. Oechslin

EPFL

Outline

- Authentication
  - Passwords, password managers
  - Token, universal 2nd factor
  - Biometrics
- Authentication protocols
  - Challenge response protocols
  - Kerberos
  - Oauth2

Authentication

Authentication factors

- Access control only makes sense if we can authenticate subjects
- There are 3 common factors that are used for authentication:
  - Something you know: passwords, pin codes
  - Something you own: paper card, smartphone, certificate, electronic token
  - Something you are: biometrics
- Tow factor authentication (aka strong authentication) implies the use of two factors
  - Password are 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)
  - Note: MS uses no salt, no iterations...
- '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, 2015)

Even strong passwords are at risk

- A stolen password can be replayed by anybody
- An attacker can
  - steal password using client-side malware
  - Steal and crack hashes stolen from a server (lecture 4b)
  - Phish passwords with a fake site
  - Eavesdrop the password
- Credential stuffing
  - Lists of usernames and passwords are published or sold on the Internet
  - Hackers try the same credentials on may 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

- Stores 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
- Options for storage
  - Password manager works off-line, uses a local file (e.g. Keepass).
    - You can chose to host this file in the cloud
  - Password manager talks to a server in the cloud (e.g. Lastpass)
    - browser plug-in or mobile 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 a open source server that you can host where you want (e.g. Bitwarden)

Something you own

Something you own

- Typically called a (hardware/software) token

- Bingo cards
  - proves that the user owns the card
  - can easily be copied without being detected

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	7467	3029	5370	7921	5122	0429	6432	6918	4167
6	8994	8136	6234	2442	2031	5101	5015	0771	7370	6620
7	2265	9381	7323	5900	9789	0445	6976	2093	3498	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

- One time password (OTP) tokens
  - displays a password to be used only once
  - password changes with time or click
  - proves that user owns token at time of login
  - can not be copied (secure hardware)
  - OATH standard

Something you own

- TAN generators
  - a calculator that generates a number based on an input typed by the user
  - 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

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

Something you own

- A private key in a hardware token
  - Token signs a challenge
  - Different key for different web sites
  - Proof that user owns token
  - Secure hardware
  - Universal 2nd Factor (U2F) standard

OATH, standard 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 as hardware and software
  - E.g the Google Authenticator app
- Generation of the next OTP is either counter or time based
  - the algorithm is described in RFC 4226
- Counter based:

$$\mathrm{hotp}(k, c) = \mathrm{truncate}(\mathrm{hmac{-}sha1}(k, c))$$

- Time based (initial time t_0, time interval x)

$$\mathrm{totp}(k, t) = \mathrm{hotp}(k, (t - t_0)/x)$$

U2F standard usage of private keys

- 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 name of the domain 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

U2F FIDO2 (webauthn)

- FIDO2 standardizes the use of 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
    - usb token
    - smartphone connected by Bluetooth
    - authentication module of platform (e.g. biometric authentication of a phone or a laptop)
- The assertion contains information whether the user
  - was present (e.g. clicked on a button)
  - was verified (e.g. pin or fingerprint)

U2F FIDO2

- Advantages
  - 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
    - Private key stays in secure hardware of 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

Biometrics: Introduction

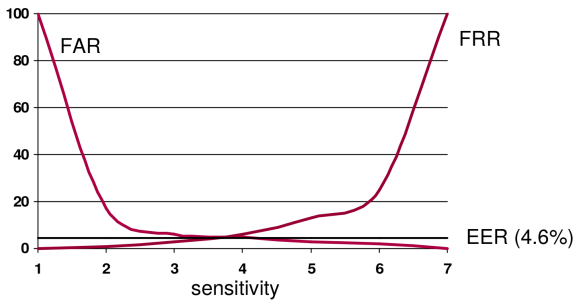
- Motivation
  - Something you know could be guessed
  - Something you own could be stolen
  - Nothing to remember, nothing that can be lost
- Physiological
  - Iris
  - Retina
  - Fingerprint
  - Shape of head,
  - Shape of hand
- Behavioral
  - Speech
  - Keystroke timing
  - Handwritten signature on tablet
  - Gait

Biometrics authentication process

- Registration
  - acquisition
  - extraction characteristics
  - storage of characteristics
- Authentication
  - acquisition
  - comparison
  - decision
- Limitations
  - Acquisition is never exact
  - Comparison is never perfect match
  - Decision is always error prone

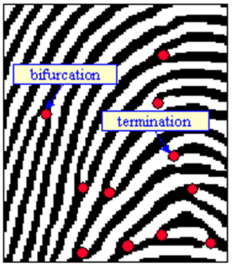
Biometrics: error rates

- The decision algorithm must accept a certain error
  - the sensitivity can be tuned
- FAR False acceptance rate: the system declare a match when it wasn't
- FRR False rejection rate: the system declare a non-match although it was a match
- EER Equal error rate, when the system is tuned such that FAR=FRR

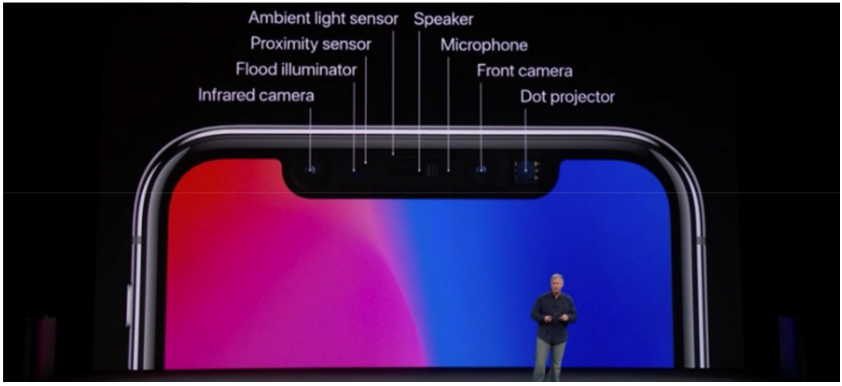


Fingerprints

- The fingerprint is read by a sensor
- An image of the ridges is created
- The minutiae are extracted
  - 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



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.

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
- You still need to type a password when booting the phone
  - used to decrypt the storage

FaceID limitations

source: **bkav**

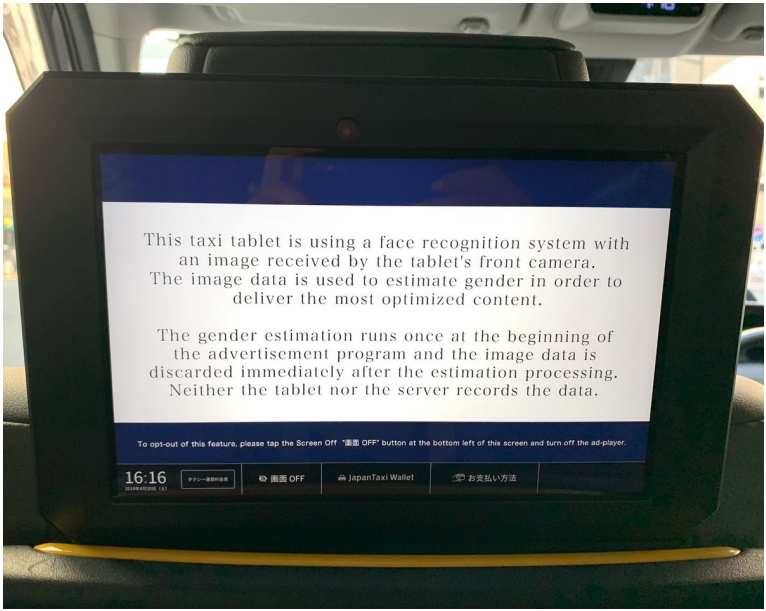
Biometrics: discussion

- Biometrics and authentication
  - no hashing possible (neither for transmission nor for storage)
    - risk of theft
  - It is impossible to change a stolen finger...
  - Best to store the biometric data locally in protected hardware
  - Example: biometric data of Swiss citizens is stored in their passports, not in the customs offices
  - Some sensors can be fooled or replaced
  - Not ideal for remote authentication
    - rather use it for local access to authentication key (U2F)

Biometrics: discussion

- Biometrics and privacy
  - biometric data is considered sensitive data by European data protection laws and the future Swiss law
    - considered sensitive under current Swiss law if it reveals race or health
  - 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 IA to extract information from faces
    - Detection of Propensity for Aggression based on Facial Structure
    - Row over AI that 'identifies gay faces'
    - Privacy fears as Tokyo taxis use facial recognition cameras to guess riders' age and gender for targeted advertisements

Biometrics: discussion



source: **Twitter: Rosa Golijan**

Biometrics: discussion

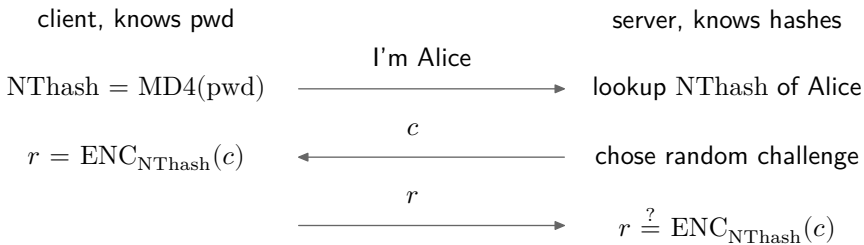
- Swiss law example
  - The KSS sports center (swimming pool) had a central database with user profiles including their biometric data (face)
  - They had to change the system
    - biometric data is now stored in a separate DB without a link to the user's profiles
    - a key stored on the user's member cards is needed to link and decrypt the biometric data of a user
    - thus biometric data can not be used without consent of the user
  - details in this **article** of the Swiss data protection officer.

Authentication protocols

When sending a password by TLS is not enough

Challenge-response

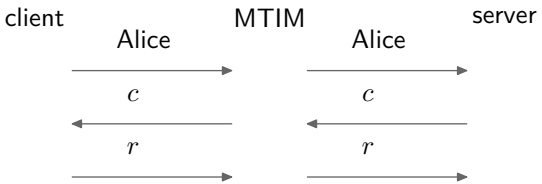
- Rather than sending the password to the server,
  - the server sends a random challenge to the client
  - the client uses the password hash to create a response
    - e.g. encryption or HMAC of the challenge
- Example, Microsoft ad-hoc networking with SMBv1 (not Kerberos)



Alice is authenticated if the response matches

Challenge-response

- Typically, challenge response protocols are vulnerable to MITM attacks



Server now thinks MITM is Alice

- Microsoft introduced the signature of the packets with a key derived from the pwd hash.
  - ➡ The MITM does not know the key, can not send any packets

Challenge-response: mutual auth

- The server and the client can both use a challenge
  - thus the client can also authenticates the server
- Examples:
  - Newer versions of Microsoft challenge-response (SMBv2)
  - WiFi WPA and WPA2
- Challenge response protocols may be vulnerable to eavesdropping and cracking attacks
  - The attacker records a challenge and a response
  - They try all possible passwords to find which pwd would yield the same response
- SMBv1 used no salt to generate the response $\rightarrow$ rainbow tables!
- WPA and WPA2 use the SSID as salt, and 4096 iterations
  - only simple passwords can be cracked

Kerberos

- Kerberos provides authentication and authorization across a network
- Developed at MIT in the 80's
- Exclusively based on symmetric keys
- Subjects receive tickets that they use to access objects
- Initially deployed in Unix
- The main authentication protocol in Windows network

Tickets, authenticators

- By presenting a tickets a client get access to a service from a server
- The authenticator proves that the client is the legitimate owner of the ticket
- The ticket contains
  - c: the client's identity
  - a: his IP address
  - v: a validity period
  - $K_{c,s}$: a session key used between the client and the destination server
- The ticket is encrypted with the key of the destination server

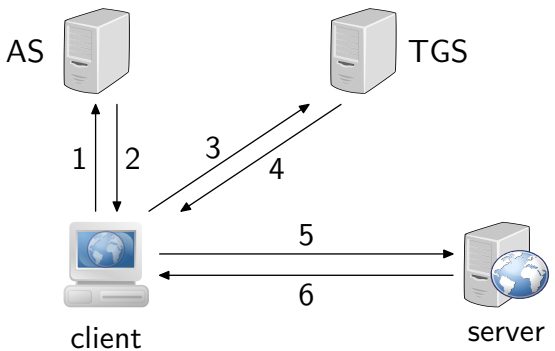
$$T_{c,s} = \mathrm{ENC}_{K_s}\{c, a, v, K_{c,s}\}$$

- The authenticator is an id and a timestamp encrypted with a session key

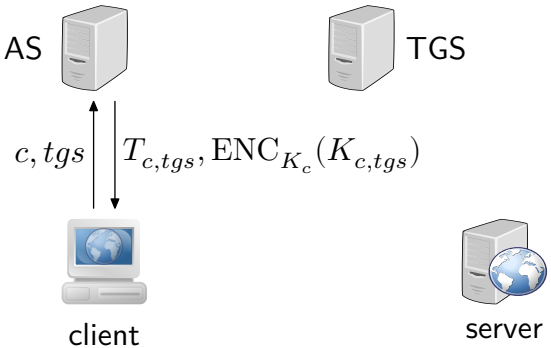
$$A_{c,s} = \mathrm{ENC}_{K_{c,s}}\{c, t\}$$

Authentication server, Ticket server

- Kerberos uses a three step approach
  - First, an authentication server (AS) authenticates the client and delivers a ticket granting ticket (TGS)
  - The client can the present the TGT to the ticket granting server (TGS) to get a ticket for the services he wants to use
  - Then he can access the service



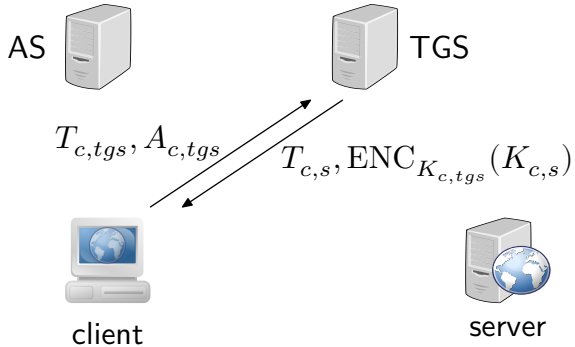
Authentication



- the password hash is used as key K_c
  1. the client asks for a ticket for the TGS, sends c, tgs
  2. the AS responds with the ticket $T_{c,tgs}$ and an encrypted session key $\mathrm{ENC}_{K_c}(K_{c,tgs})$

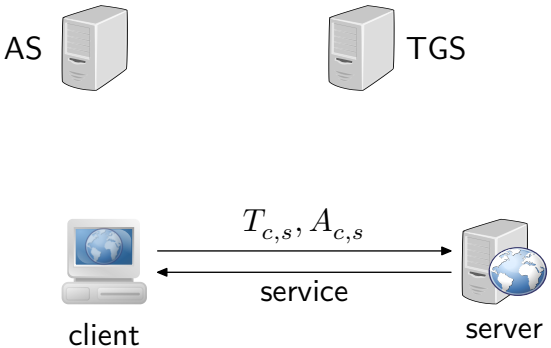
- the client knows K_c and can decrypt the session key

Authorization



- the client creates an authenticator with $K_{c,tgs}$
  3. the client sends the ticket $T_{c,tgs}$ and $A_{c,tgs}$ to the TGS
  4. the TGS verifies the authenticator and responds with a service ticket $T_{c,s}$ and an encrypted session key $\mathrm{ENC}_{K_{c,tgs}}(K_{c,s})$
- the client knows $K_{c,tgs}$ and can decrypt the session key

Access



- the client creates an authenticator with $K_{c,s}$

  5. the client sends the ticket $T_{c,s}$ and $A_{c,s}$ to the server
  6. the server verifies the authenticator

- the client can now use the service

Kerberos: discussion

- The ticket obtained from the AS is called the ticket granting ticket (TGT) because it is used to get other tickets
- To validate a ticket, a server
  - decrypts the ticket
  - validates the IP address and the validity period
  - uses the included session key to
    - decrypt the authenticator
    - verify that it contains the same id c as the ticket
  - verifies that the authenticator is fresh (e.g. timestamp less that 5min)
  - verifies that the same authenticator has not been used in the last 5min
- Tickets are typically valid for eight hours

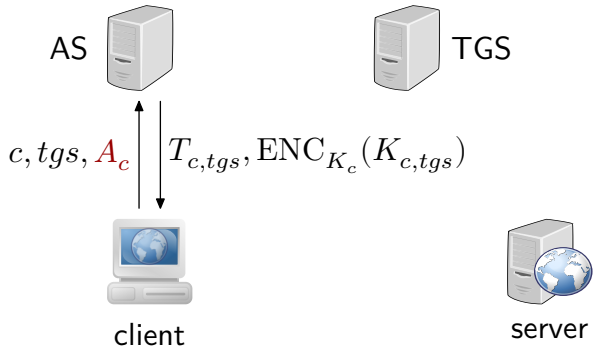
Kerberos: Attacks

- MITM attacks, not possible
  - The attacker does not know the session keys (K_c, $K_{c,tgs}$ and $K_{c,s}$)
  - The attacker does not know the keys K_{tgs}, K_s used to encrypt the tickets
  - The attacker can not create/modify any ticket or authenticator
- Replay attack
  - Attacker can not replay an old authenticator, only fresh ones are accepted
  - Attacker can not replay fresh authenticators, servers keep a list of the last authenticators received.

Kerberos pre-authentication

- Anybody can request a ticket for a used c from the AS
  - They can then brute force the password by trying to decrypt the session $K_{c,tgs}$ with password hashes
- To prevent this, pre-authentication can be used
  - The user must send an authenticator when requesting a TGT
- You can still try to bruteforce the authenticator
  - but you be there to observe the authenticator when the user logs in
- ✓ Microsoft uses pre-authentication
  - K_c is not the password hash
  - ✓ It is derived from the password hash using salt and 4069 iterations of SHA1 hashing
- be like Microsoft !

Pre-Authentication

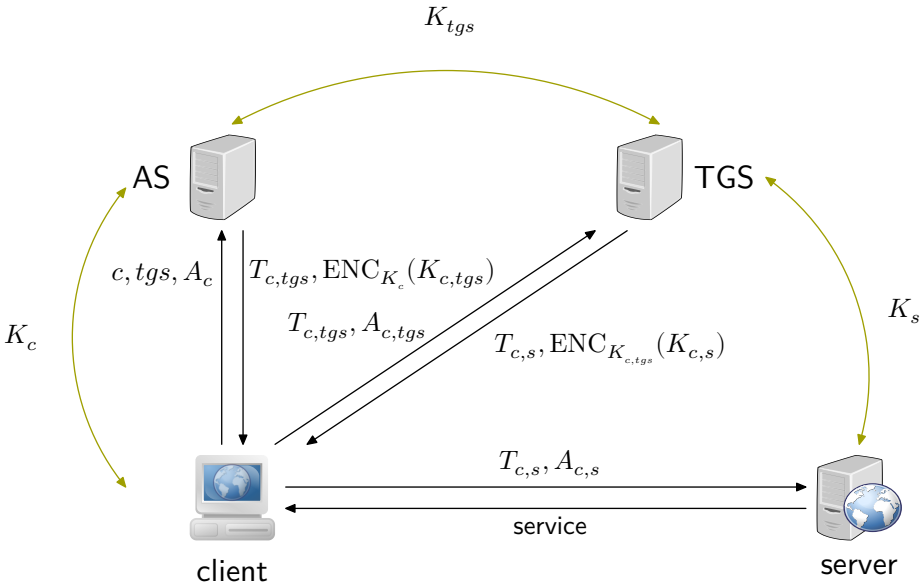


- the client creates an authenticator with K_c (pwd hash, salt, iterations)
  1. the client asks for a ticket for the TGS, sends c, tgs and A_c
  2. the AS responds with the ticket $T_{c,tgs}$ and an encrypted session key $\mathrm{ENC}_{K_c}(K_{c,tgs})$, if A_c is valid
- the client knows K_c and can decrypt the session key

Kerberos: symmetric keys

- The security of Kerberos relies on symmetric keys
  - the password hash of the user(s)
  - the symmetric keys used to encrypt the tickets
    - between the AS and the TGS(s)
    - between the TGS and the server(s)

Kerberos summary



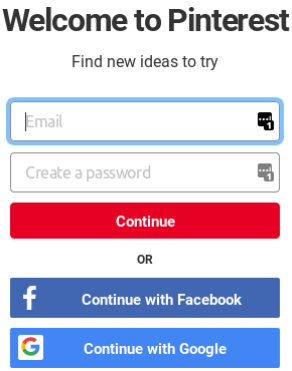
Delegated authentication

OAuth2

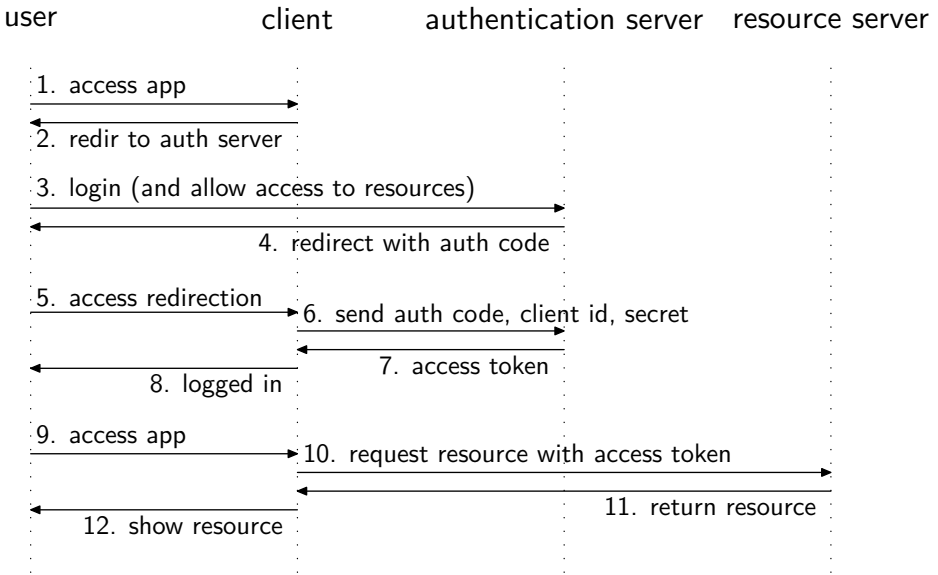
- Oauth2 is a protocol used for delegated authentication on the Internet
  - Facebook, Google, Twitter, ...can be used to authenticate and access other applications
- For example, you can log in to Pinterest with Facebook or Google
- You can also authorize Pinterest to access your photos on Facebook
- Oauth2 is specified in RFC 6749

Roles

- Client: application that want 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 account and resources on resource server (wants to log into Pinterest with his Facebook account)



Typcial Oauth2 flow



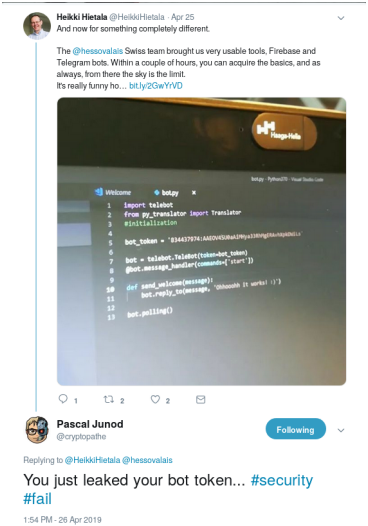
Typical Oauth2 flow

- 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

- Oauth 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 (Twitter, Instagram ...) do not store your password.
  - They use Oauth2 to request a access token and use this
  - When you change your password you don't need to type you new password into all your devices



Conclusions & Questions

Conclusions

- Passwords can go a long way, especially if you use a password manager
- For critical accounts, two factor authentication significantly raises the bar for attackers
  - U2F is secure and user friendly
- Challenge response protocols can authenticate a user without sending the password
  - they can be vulnerable to MITM, in particular if there is no mutual authentication
- Kerberos uses tickets to authenticate users across a network
  - Authentication is separate from authorization

Conclusions

- Oauth is used to delegate authentication on the internet
  - it has no crypto at all, relies on communications being made over TLS
- The challenge-response protocols we have seen and Kerberos use symmetric crypto
  - If you can sniff the authentication, you can bruteforce the password
- In session 4c, we saw SRP, a Password Agreed Key Exchange (PAKE)
  - It is asymmetric, based on Diffie-Hellman
    ➡ it can't be cracked, even if the attacker sees all messages
  - WPA3 (the next WiFi security protocol) also uses an asymmetric key exchange
    ➡ no more possible to crack WiFi passwords!

Questions

- Name a second authentication factor that can also be used to sign transaction?
- Why does U2F add a domain name in the signature of the random challenge?
- What is the big advantage (security wise) of using U2F versus OTP?
- Is biometric authentication system with FAR of 0.01% a good system ?
- Why do we need authenticators in Kerberos?
  - Why is it important to synchronize the clocks of the machines using Kerberos?
- How does pre-authentication increase security in Kerberos?