Avtentikacija, avtorizacija, enkripcija

- openID, OAuth

Avtentikacija:

- ugotavljamo identiteto uporabnika; "ti si Janez"
- ne želimo, da vsi lahko vidijo tisto, kar mi vidimo
- ne želimo, da se lahko kdorkoli lahko predstavi kot kdorkoli
- ime in geslo, certifikati (si pass), MFA, passkey
- problem MFA, če imaš preko SMS:
  - nekdo ti lahko ukrade SMS avtentikacijo, če hijacka tvoj ISP (SS7)
  - sim swapping
- kako prisluškovati GSM klicem:
  - postaviš svojo anteno in rečeš, da boš ti dal key
- kaj lahko Mirko naredi, ko ugotovimo, da je Mirko - avtorizacija nam pove, katere operacije lahko posamezni uporabnik oz. skupine delajo
- vedno, ko avtentikacija sloni na neki tretji entiteti (sigenca), je to spet ranljivo, ker nimamo nadzora nad tem sistemom

Avtorizacija:

- npr. nočemo, da si študent sam vpiše oceno
- potrjevanje/omogočanje operacije
- npr. potrjevanje transkacije v banki:
  - banka ima nek ruleset, kaj je normalna transakcija
  - če si ravnokar v Ljubljani in je prišla neka transakcija iz Londona, te bo banka poklicala, če si to res ti naredil
  - avtorizacija ni nujno samo od ene osebe (ti sebi avtoriziraš transakcijo), ampak lahko tudi od več oseb (banka)
- od tu naprej že verjamem, da je Jože res Jože; vseeno je lahko morebitna dodatna preverba
- ne zagotavlja ali je kanal, po katerem delam to operacijo res varen - za to rabimo enkripcijo

Enkripcija:

- samo jaz in ti veva, o čem se pogovarjava
- jaz z zasebnim ključem podpišem, on pa z javnim ključem preveri, da sem res jaz podpisal

- včasih smo še vedno ranljivi s človeškim faktorjem, ki lahko izda zaupne podatke
- HTTPS certifikati:
  - pogleda datum veljavnosti certifikata
  - Certification Authority izdaja certifikate, mu zaupamo, je vse v redu, dokler jim ne ukradejo ključev, ker potem si lahko v njihovem imenu izdamo certifikat za neko spletno stran, ki ni zares naša
- Enkripcija
  - Kupimo izdelek preko spleta
- Avtentikacija
  - Kupimo kosilo s študentskim bonom (študentska izkaznica)
- Avtorizacija
  - Na koncertu želimo brezplačno pijačo/dostop do golden ringa s svojo karto

Primeri storitev in standardov:

- (PHP) session, OpenID, OAuth
- nekega uporabnika ne želimo ves čas spraševati, če je to res on
- ne želimo za vsak app, ki ga naredimo imeti še svojega sistema za avtentikacjo
- želimo minimizirati effort, da ne implementiramo vedno skoraj iste stvari - single sign on (npr. od Google, Githuba)
- ko uporabimo SSO, povemo, do česa vsega lahko dostopa aplikacija, ki jo avtoriziramo (npr. pri Github SSO ji damo dostop, da lahko dela commite, ne pa da dela nove repozitorije); omejimo katere podatke o tebi lahko bere neka aplikacija (npr. Google ji bo dal tvoje ime, ampak ne tvoje telefonske)

PHP session:

- `$_SESSION` spremeljivka

Stara šola
- Spremenljivka `$_SESSION`
- Zaledni sistem drži podatke o:
  - Avtentikaciji (uporabnik)
  - Veljavnosti (časovno pogojeno)
- Proces
  - `Session_start()`
  - `Session_unset()`
  - `Session_destroy()`

Kako deluje?
- Vsakič preverimo identifikator
  - session_id
- Vrednosti hranimo v polju `$_SESSION`
- Na klientu držimo podatek v PHPSESSION piškotku

Problemi:

- na vsakem sistemu se rabiš na novo avtorizirati, ves čas moraš iste podatke dajati na različne sisteme
- to je problem tudi za firme, ker nimaš nekega uporabnika poenotenega
- poskus reševanja s caching serverjem, ki hrani seje:
  - še vedno omejeno samo na produkte ene firme (enititete)
- problematike:
  - kraja piškotov
  - več sej se breaka, ko imamo več tabov

OAuth:

- spletni standard za avtorizacijo
- JWT token - JSON objekt

SAML:

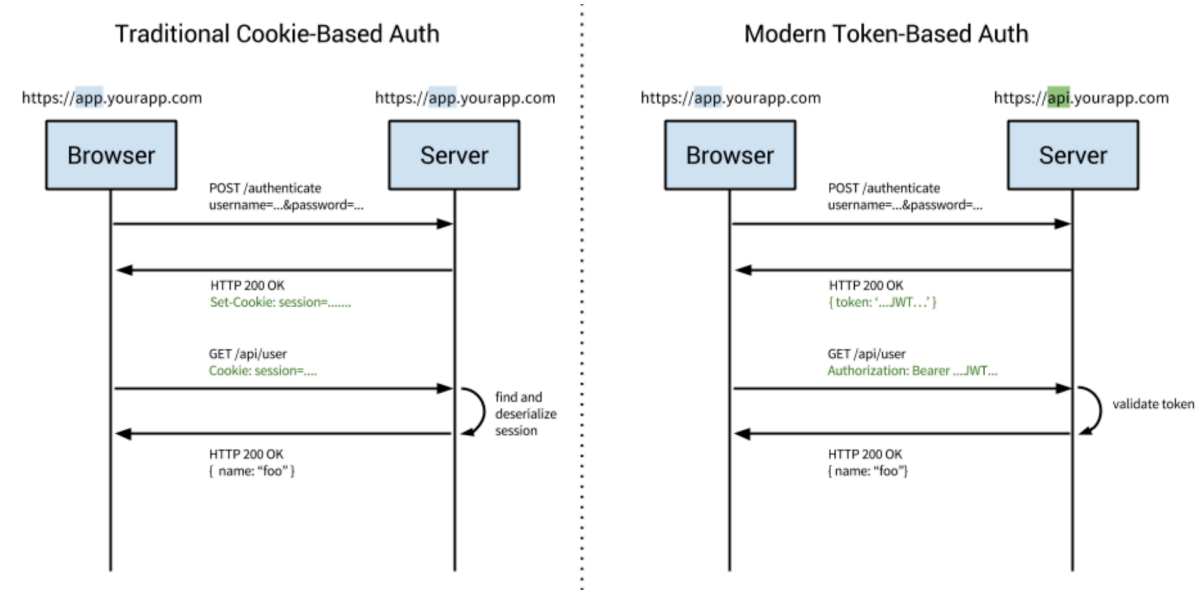
- XML

OpenID:

- JWT
- žetoni za dostop third party aplikacij z SSO

JWT:

- včasih je bil cookie based auth - lahko ukradeš cookie
- token based auth



- header:

    - lahko določimo enkripcijski algoritem, ker različne naprave lahko različne algoritme uporabljajo in ker se enkripcijski algoritmi razvijajo
- payload - dejanski podatki
- podpis:
    - zahashiran header in payload
- zakaj JWT:
    - ločimo kdo nam zagotavlja samo storitev in kdo zagotavlja avtentikacijo - lahko je problem, ker damo zaupanje tudi na SSO, ne samo na ponudnika storitve
- lahko imamo druge osebne podatke v storitvi, zato ji ne zaupamo 100%
- umikamo se stran od tega, da bi vse podatke hranili v neki SQL bazi, ampak v JWT tokenu:
    - v primeru napada na storitev ni na sistemu podatkov o uporabniku
    - lahko imitiramo druge uporabnike, če sistem uporablja simetrično enkripcijo
- po tem ko imamo JWT token, ga uporabljamo za vso avtentikacijo
- v telesu imamo neke atribute, ki jih lahko poljubno mnogo dodajamo
- ne rabiš več svojega auth serverja za vsako storitev v firmi
- token nam zagotavlja avtentikacijo, ker je podpisan - če nimamo ključa, ne vemo, kaj je noter v tokenu:
    - če spreminjam naše podatke, signature ne bo več veljaven, torej server lahko samo preverja signature
    - vsi serverji imajo public keye, da lahko preverijo, da je signature pravilen
- običajno pošiljaš JWT prek API, lahko je tudi Http-Only cookie
- lahko imamo več encryption algoritmov in public key za vsakega

JWT napadi:

- none algoritem:
    - ne kriptiraš podatkov
    - ima namen za komunikacijo pri zalednih sistemih, ampak v produkciji naj tega ne bi uporabljal; namenjen že preverjenim žetonom v zalednih sistemih
    - lahko poljubno spreminjaš podatke v žetonu
    - zaščita: eksplicitno definiraš algoritme, ki jih podpiraš
- zamenjava algoritmov:
    - iz asimetrične enkripcije zamenjaš na simetrično
    - prej smo s public keyem lahko preverili, da je tisto kar je podpisano s private keyem prava stvar
    - JWKS - identity provider, ki ima nek well-known url, ki nam pošlje vse public keye - ko postaviš nov server, ne boš rabil na vsakem serverju dodati tega public keya, ampak mu rečeš, da uporablja JWKS in imamo na enem serverju vse public keye

- nimamo direktno public keyev na sistemu, ampak imamo na JWKS serverju
- če zamenjamo hashing algoritem, signature ni več veljaven
- simetrično enkripcijo bo probal preverjati tako, header in data signa s public keyem
- rešitev: če uporabljaš JWKS, ga uporabljaj povsod; uporabljaš ali samo simetrične algoritme ali pa samo asimetrične
-