

Provera identiteta

Autentifikacija \neq identifikacija \neq autorizacija

- autentifikacija (provera identiteta) = utvrđivanje identiteta korisnika
- identifikacija = utvrđivanje da li je korisnik poznat sistemu
- autorizacija = utvrđivanje prava koja korisnik ima nad resursima u sistemu

- autorizacija zahteva uspešnu autentifikaciju
 - autentifikacija prethodi autorizaciji
- autentifikacija podrazumeva identifikaciju
 - identifikacija je sastavni deo postupka autentifikacije

Učesnici u autentifikaciji

- onaj koji se predstavlja kao korisnik (**prover**, claimant)
- onaj koji proverava identitet (**verifier**, recipient)

Alati za autentifikaciju

- šifrovane poruke
 - ispravan šifrat potvrđuje autentičnost poruke i pošiljaoca
- checksum funkcije
 - izračunate nad porukom za autentifikaciju i tajnim ključem
- heš funkcije

Protokol za autentifikaciju

- proces u realnom vremenu kojim se utvrđuje identitet
- rezultat
 - korisnikov identitet je autentičan
 - korisnikov identitet nije autentičan
- karakteristike protokola za autentifikaciju
 - zanemarljiva verovatnoća da treći učesnik, predstavljajući se kao prover, može da navede verifier-a na pozitivan rezultat autentifikacije
 - verifier ne može da koristi informacije koje je dostavio prover kako bi se predstavio kao prover trećem učesniku

Sredstva za autentifikaciju

- korisnik se identifikuje pomoću određenog identifikatora (npr. username)
- svoj identitet potvrđuje pomoću više faktora
 - nečega što zna (lozinka, PIN, šifra sefa)
 - nečega što ima (smart kartica, ključ, generator vremenski zavisnog PIN-a)
 - nečega što je njegova karakteristika (otisak prsta, oblik lica, retina, prepoznavanje glasa)
- autentifikacija je pouzdanija ako se koristi više faktora
 - lozinka se može pogoditi
 - ključ se može izgubiti
 - prepoznavanje lica: "false positives" procenat je još uvek značajan
- banke koriste dvostruku autentifikaciju na bankomatima
 - traže i karticu i PIN

Klasifikacija šema za autentifikaciju

- slabe (weak) šeme
 - jednostavne za implementaciju
 - podložne napadima
 - lozinka, PIN
- jake (strong) šeme
 - zasnovane na challenge-response protokolu
 - obuhvataju kriptografske tehnike

Autentifikacija pomoću lozinke

- najrašireniji metod za autentifikaciju
- lozinka ~ tajni ključ
- postupak
 - korisnik se prijavljuje pomoću para (korisničko ime, lozinka)
 - server proverava da li dati par postoji u registru postojećih korisnika
 - registar = sistemski fajl, zaštićen od čitanja
- ako se lozinka smešta direktno u sistemski fajl, nema zaštite od privilegovanih korisnika sistema ili čitanja fajla iz bekapa
- rešenje: ne čuva se lozinka, već njen heš
 - prilikom prijavljivanja izračunava se heš i poredi sa onim koji je skladišten u fajlu
 - otvoreni tekst lozinke nigde se ne skladišti trajno

Autentifikacija pomoću lozinke

- treba izbegavati skladištenje lozinke kao otvorenog teksta
 - upravo ono što radi Windows u registry-ju
 - "usluga" korisniku da ne mora u toku rada da ponovo unosi svoju lozinku prilikom prijavljivanja na druge sisteme u mreži
 - na UNIX/Linux sistemima, heševi lozinki se čuvaju u `/etc/passwd` ili `/etc/shadow`

Autentifikacija pomoću lozinke

- napad pomoću rečnika (dictionary attack)
 - izračunava se heš svake reči iz rečnika (recimo milion stavki)
 - porede se izračunati heševi iz rečnika sa sadržajima skladištenim u sistemskom fajlu
- otežati napad pomoću "salt" vrednosti
 - salt: slučajan niz znakova koji se dodaje na lozinku pre izračunavanja heša
 - salt i heš se smeštaju u fajl
 - napad na konkretnu lozinku traži heširanje kompletnog rečnika za dati salt
 - i tako za svaku lozinku posebno
 - umesto da se rečnik hešira jednom, mora se heširati za svaku salt vrednost

Autentifikacija pomoću lozinke

- jednokratne lozinke
 - prilikom svakog prijavljivanja koristi se različita lozinka
 -
- naredna lozinka se određuje pomoću
 - liste unapred definisanih lozinki
 - ne koriste se sekvencijalno već sistem traži i -tu lozinku iz liste
 - sekvencijalnog ažuriranja
 - nakon pozitivne autentifikacije korisnik će poslati lozinku za naredno prijavljivanje
 - prva lozinka je unapred poznata
 - sekvencijalnog izračunavanja
 - prva lozinka je unapred poznata
 - svaka sledeća lozinka je heš prethodne lozinke: $h(h(\dots h(\text{pwd})\dots))$

Autentifikacija pomoću lozinke

- SKEY postupak
 - korisnik unese slučajan broj R
 - sistem generiše $f(R) = x_1$, $f(f(R)) = x_2$, ... $f(f(\dots f(R)\dots)) = x_{100}$ i daje niz x_{1-100} korisniku, a skladišti i x_{101}
 - prilikom prvog prijavljivanja korisnik šalje x_{100} , server izračunava $f(x_{100})$ i poredi ga sa x_{101}
 - ako su jednaki, korisnik je pozitivno autentifikovan
 - sistem zamenjuje x_{101} sa x_{100}
 - korisnik za dalje prijavljivanje koristi poslednji neiskorišćeni broj iz niza
- svaki broj se koristi samo jednom, a f je jednosmerna funkcija
- napadač ako i pročita x_{101} , taj podatak mu ne znači puno

Autentifikacija pomoću PIN-a

- PIN = personal identification number
- PIN = kratak niz cifara (4 do 10)
- PIN se skladišti na fizičkom uređaju npr. kartici ili tokenu
- broj mogućih vrednosti ključa je mali → ograničava se broj pokušaja korišćenja PIN-a

Jake šeme za autentifikaciju

- challenge-response princip
 - korisnik potvrđuje svoj identitet tako što demonstrira poznavanje tajne informacije
 - ako poznaje tajnu informaciju, smatra se da je autentičan
 - pri tome se tajna informacija ne odaje sistemu
- sistem korisniku šalje podatak koji se menja tokom vremena (može da uključuje
 - timestamp
 - random number
 - sequence number)
- korisnik izračunava odgovor na osnovu dobijenog podatka i tajne informacije
- ako je odgovor ispravan, sistem smatra da je korisnik autentičan

Jake šeme za autentifikaciju

- realizacija challenge-response postupka pomoću
 - kriptosistema sa tajnim ključem
 - kriptosistema sa javnim ključem
 - zero-knowledge tehnika

Challenge-response sa tajnim ključem

- korisnik i sistem dele tajni ključ i unapred dogovoreni simetrični algoritam
- postupak
 - sistem šalje korisniku slučajan broj r
 - korisnik šifruje r tajnim ključem i šalje ga sistemu
 - sistem dešifruje r i proverava da li je jednak originalnom
- broj r koji se šifruje može se dopuniti proizvoljnom porukom m , kako bi se sprečili replay napadi
- ako je dodatna poruka dugačka, može se šifrovati heš od $[r, m]$

Challenge-response sa tajnim ključem

- moguća je i obostrana autentifikacija
 - sistem šalje korisniku slučajan broj r
 - korisnik šalje poruku koja se sastoji iz šifrovanog broja r (tajnim ključem) i novog slučajnog broja r'
 - ako se dešifrovanjem dobije originalni broj r korisnik je autentičan
 - sistem šalje korisniku šifrat broja r'
 - ako se dešifrovanjem dobije originalni broj r' sistem je autentičan

Challenge-response sa javnim ključem

- prvi (naivan) pokušaj
 1. server šalje korisniku slučajan niz
 2. korisnik šifruje niz svojim tajnim ključem i vraća ga sistemu uz svoj identifikator
 3. sistem pronalazi javni ključ korisnika i dešifruje poruku
 4. ako je dešifrovani niz jednak originalnom, korisnik je autentičan
- međutim, setimo se napada na RSA:
 - Eve navodi Alice da potpiše podmetnutu poruku svojim tajnim ključem... (fajl 03, slajd 17)... i dolazi do Alicinog tajnog ključa

Challenge-response sa javnim ključem

- sigurniji način
 - sistem generiše slučajan broj r , računa njegov heš $h(r)$, šifruje r i proizvoljnu poruku m korisnikovim javnim ključem $c(r, m, pk)$ i šalje korisniku $[h(r), c(r, m, pk)]$
 - korisnik dešifruje šifrovani deo poruke (time dobija r), izračunava svoj heš $h(r)$ i poredi ga sa primljenim; ako su jednaki, šalje r sistemu
 - sistem proverava da li je primljeni r jednak originalnom

Challenge-response sa zero-knowledge tehnikom

- prethodni postupci mogu da odaju deo tajne informacije
 - ako se napadač predstavi kao sistem i šalje specifične challenge vrednosti pomoću kojih će doći do tajne informacije
- zero-knowlegde postupci bi trebalo da eliminišu ovu mogućnost tako što sistemu ne otkrivaju nikakvu informaciju

Challenge-response sa zero-knowledge tehnikom

- Fiat-Shamir algoritam: korisnik dokazuje sistemu da poznaje tajnu s u n iteracija 3-faznog procesa
 - faza 1
 - treća strana od poverenja generiše dva velika prosta broja, p i q
 - treća strana objavljuje $n = pq$, a p i q ostaju tajni
 - korisnik bira tajnu s tako da je uzajamno prosta sa n i
 - računa javni ključ kao $c = s^2 \bmod n$
 - registruje javni ključ kod treće strane

Challenge-response sa zero-knowledge tehnikom

- Fiat-Shamir algoritam
 - faza 2
 - korisnik generiše slučajan broj $r < n$, i razmenjuje sledeće poruke
 - korisnik → sistem: $x = r^2 \bmod n$
 - sistem → korisnik: slučajan boolean broj b
 - korisnik → sistem: $y = r s^b \bmod n$
 - ove poruke se razmene u $t < n$ krugova
- nakon t krugova
 - razmenjeno je $3t$ poruka
 - korisnik je generisao t slučajnih brojeva r_1, r_2, \dots, r_t
 - korisnik je izračunao t vrednosti x_1, x_2, \dots, x_t
 - korisnik je izračunao t vrednosti y_1, y_2, \dots, y_t
 - sistem je generisao t boolean vrednosti b_1, b_2, \dots, b_t
 - sistem je primio x_i i y_i takvi da su

Challenge-response sa zero-knowledge tehnikom

- Fiat-Shamir algoritam
 - faza 3
 - sistem prihvata korisnika ako za svako $i < t$ važi $y_i \neq 0$ i

$$y_i^2 = x_i \cdot c^{b_i} \bmod n$$

- jer je

$$y_i^2 = (r_i \cdot s^{b_i})^2 \bmod n = r_i^2 \cdot s^{2b_i} \bmod n = x_i \cdot s^{2b_i} \bmod n = x_i \cdot c^{b_i} \bmod n$$

Challenge-response sa zero-knowledge tehnikom

- Fiat-Shamir algoritam
 - slanjem poruka b_i (t puta) sistem proverava da li korisnik može da demonstrira poznavanje tajne s
 - i da spreči napadača koji se lažno predstavlja da izabere slučajan r i šalje npr. $x = r^2/c$
 - za primljeno $b = 0$ šalje se samo r jer je $y = r s^0 \bmod n$
 - za primljeno $b = 1$ šalje se $y = rs \bmod n$
 - što ne otkriva ništa o s jer je r slučajan broj

Napadi na mehanizme za autentifikaciju

- napadi sa lažnim predstavljanjem (*impersonation attacks*)
 - napadač dolazi u posed tajne informacije (lozinka, PIN, tajni ključ) i koristi je za autentifikaciju
- napadi sa ponovljenim porukama (*replay attacks*)
 - napadač prisluškuje komunikaciju u toku autentifikacije i ponavlja je
 - npr. ako nalog za plaćanje potpiše pravi korisnik banke, a napadač ponavo šalje istu poruku banci (time ponavlja plaćanje)
 - anti-replay mere: timestamp, slučajan broj, itd
- napadi sa uvođenjem kašnjenja (*forced delay attacks*)
 - napadač presretne poruku, sačeka određeni period vremena, i prosledi je na odredište
 - korisnik dobije timeout, misli da je transakcija otkazana, a ona je stigla na odredište

Napadi na mehanizme za autentifikaciju

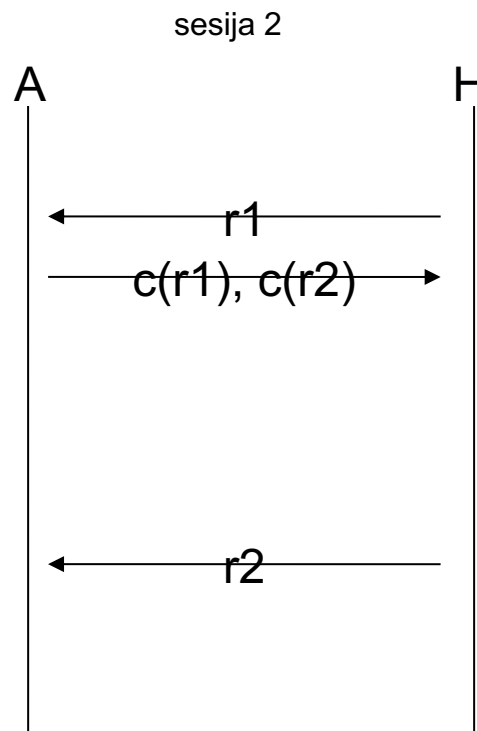
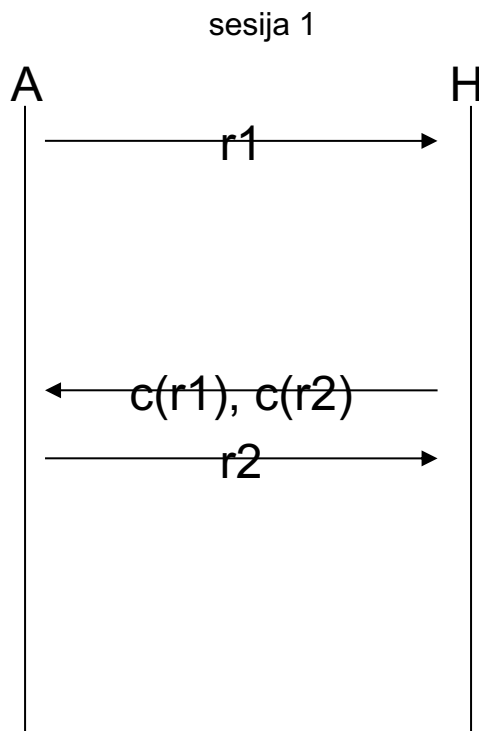
- napadi sa preplitanjem (*interleaving attacks*)
 - napadač koristi informacije iz više tekućih postupaka za autentifikaciju
 -
 - primer protokola:
 - korisnik→sistem: $c(k, r)$ (korisnik šalje sistemu šifrovani slučajan broj)
 - sistem→korisnik: $[r, c(k, r')]$ (sistem odgovara dešifrovanim brojem i šalje šifrat novog)
 - korisnik→sistem: r' (korisnik odgovara dešifrovanim novim brojem)
- primeri napada
 - oracle session attack
 - parallel session attack

Napadi na mehanizme za autentifikaciju

- *oracle session attack*
 - napadač H se predstavi kao korisnik A
 - H generiše broj i šalje ga kao da je to $c(k, r_1)$
 - sistem dešifruje poruku ključem k , generiše broj r_2 i šalje $[r_1, c(k, r_2)]$
 - korisnik A pokušava da se prijavi na sistem, ali H presreće komunikaciju i predstavlja se kao sistem
 - H šalje $c(k, r_2)$ korisniku A
 - A šalje $[r_2, c(k, r_3)]$ napadaču H
 - napadač H šalje r_2 sistemu (time se pozitivno autentifikovao)
 - i prekida komunikaciju sa korisnikom A
- način da se ovaj napad izbegne je da se druga poruka u protokolu formira ovako:
 $[c(k, r), c(k, r')]$
umesto ovako:
 $[r, c(k, r')]$

Napadi na mehanizme za autentifikaciju

- *parallel session attack*: napad na modifikovani protokol; odvija se u dve sesije
 - pretpostavlja se da u protokolu obe strane mogu da iniciraju autentifikaciju
 - napadač H presreće komunikaciju između korisnika A i B



Mere protiv napada na mehanizme za autentifikaciju

- *replay attacks*: koristiti sequence numbers
- *interleaving attacks*: sequence numbers
- *forced delay attacks*: slučajni brojevi i redukovani vremenski prozori

Mere protiv napada na mehanizme za autentifikaciju

- ako se koriste lozinke
 - definisati pravila vezana za dužinu i sadržaj lozinki, rok trajanja, vremenski period kada je dozvoljen pristup
- definisati bezbednosnu politiku koja opisuje uslove pod kojima se korisnički nalozi kreiraju, menjaju i brišu
- sprovoditi strategije za ublažavanje rizika
 - analiza mogućih scenarija napada
 - analizirati moguću štetu kod otkrivanja raznih vrsta tajnih informacija
- za dugo otvorene konekcije sprovoditi autentifikaciju periodično
- ako se autentifikacija koristi zajedno sa proverom integriteta, koristiti različite tajne informacije (ključeve)
- ako se koriste timestamps, neophodna je sinhronizacija časovnika
- ograničiti broj pokušaja za autentifikaciju

Standardi za autentifikaciju

- X.509
- Kerberos
- HTTP basic/digest auth
- OAuth

X.509

- hijerarhijski direktorijumski servis koji čuva informacije o korisnicima za potrebe autentifikacije
- informacije su smeštene u sertifikate
- definiše tri tipa autentifikacione procedure
 - one-way authentication
 - two-way authentication
 - three-way authentication

- one-way authentication
 - prover šalje verifieru informacije kojima se potvrđuje
 - da je prover formirao poruku
 - da je poruka upućena verifieru
 - da nije bila modifikovana ili ponovo poslata (replay) u prenosu
 - utvrđuje se samo identitet proverera
- poruka mora da sadrži minimalno
 - slučajan broj *rand*, kojim se sprečava *replay* napad
 - timestamp *tmp*, koji uključuje vreme formiranja poruke
 - identitet verifiera
 - potpisani podaci koji uključuju *rand* i *ttmp*, praćeni sertifikatom proverera
- a opciono i
 - podaci šifrovani javnim ključem verifiera koji mogu poslužiti za uspostavljanje session ključa

- two-way authentication
 - omogućava obostranu autentifikaciju
 - dodatna poruka kojom verifier potvrđuje svoj identitet
 - primljeni slučajni broj *rand*
 - novi slučajni broj *rand'*
 - potpisani podaci koji uključuju *rand*, *rand'* i identitet proveru kojima se potvrđuje identitet verifera

- three-way authentication
 - uključuje i treću poruku koju prover šalje verifieru, koja sadrži
 - slučajan broj *rand*
 - potpisane podatke koji služe za sinhronizaciju časovnika

X.509

- one-way i two-way autentifikacija se koriste na webu u okviru SSL protokola
- biće reči o tome kasnije

Kerberos

- nastao na MIT-u 1980-tih
- verzije 1-3 su se koristile samo interno na MIT-u
- verzija 4 u javnoj upotrebi
- verzija 5 ispravlja neke nedostatke i postaje široko rasprostranjena (Windows)
- Kerberos = centar za distribuciju ključeva ima tri „glave“
 - bazu podataka
 - server za proveru identiteta
 - server za izdavanje karata
- omogućava **single sign-on** (SSO): korisnik se prijavi samo jednom a potom (u skladu sa svojim pravima) ima pristup svim resursima na mreži
 - upravljanje velikim brojem korisničkih naloga
 - efikasan pristup pojedinačnim resursima
 - lozinke se nikad ne šalju kao otvoreni tekst

Kerberos

- centar za proveru identiteta → Key Distribution Center, KDC
 - baza u kojoj se čuvaju provereni parametri svih učesnika Kerberos sistema
 - svi učesnici mu bezuslovno veruju
 - svaki učesnik (korisnik, računar, mrežni servis) predstavljen je imenom u KDC bazi

Kerberos principal

- koncept principala: jednoznačno identifikuje učesnika
 - za principala je vezan tajni ključ za autentifikaciju učesnika kod KDC-a
- struktura principala: **identity**/**instance**@**realm**
 - *identity*: ime Kerberos učesnika; obavezno polje
 - *instance*: ime računara na kome se nalazi resurs, ili ime korisničke grupe; nije obavezno
 - *realm*: identifikator pojedinačnih Kerberos okruženja; po konvenciji formira se kao uppercase DNS ime domena organizacije, npr. FTN.UNS.AC.RS
- primeri principala:

goran/nastavnici@FTN.UNS.AC.RS	(korisnik goran član grupe nastavnici)
ssh/informatika.ftn.uns.ac.rs@FTN.UNS.AC.RS	(ssh servis na računaru informatika.ftn.uns.ac.rs)
zozon/zozon.ftn.uns.ac.rs@FTN.UNS.AC.RS	(računar zozon.ftn.uns.ac.rs)

Kerberos KDC

- sastoji se iz tri komponente
 - baze principala: evidencija svih principala i njihovih tajnih ključeva
 - implementacija baze: na Linuxu LDAP, na Windowsu Active Directory
 - server za autentifikaciju (*authentication server*, AS)
 - izdaje TGT kartu svim klijentima prilikom prijave na sistem
 - pomoću nje klijenti kasnije traže pristup resursima
 - TGT = ticket-granting ticket
 - server za dodelu karata (*ticket granting server*, TGS)
 - zadužen za izdavanje ST karata namenjenih pojedinim resursima
 - pomoću ST karata klijenti pristupaju resursima
 - ST = service ticket

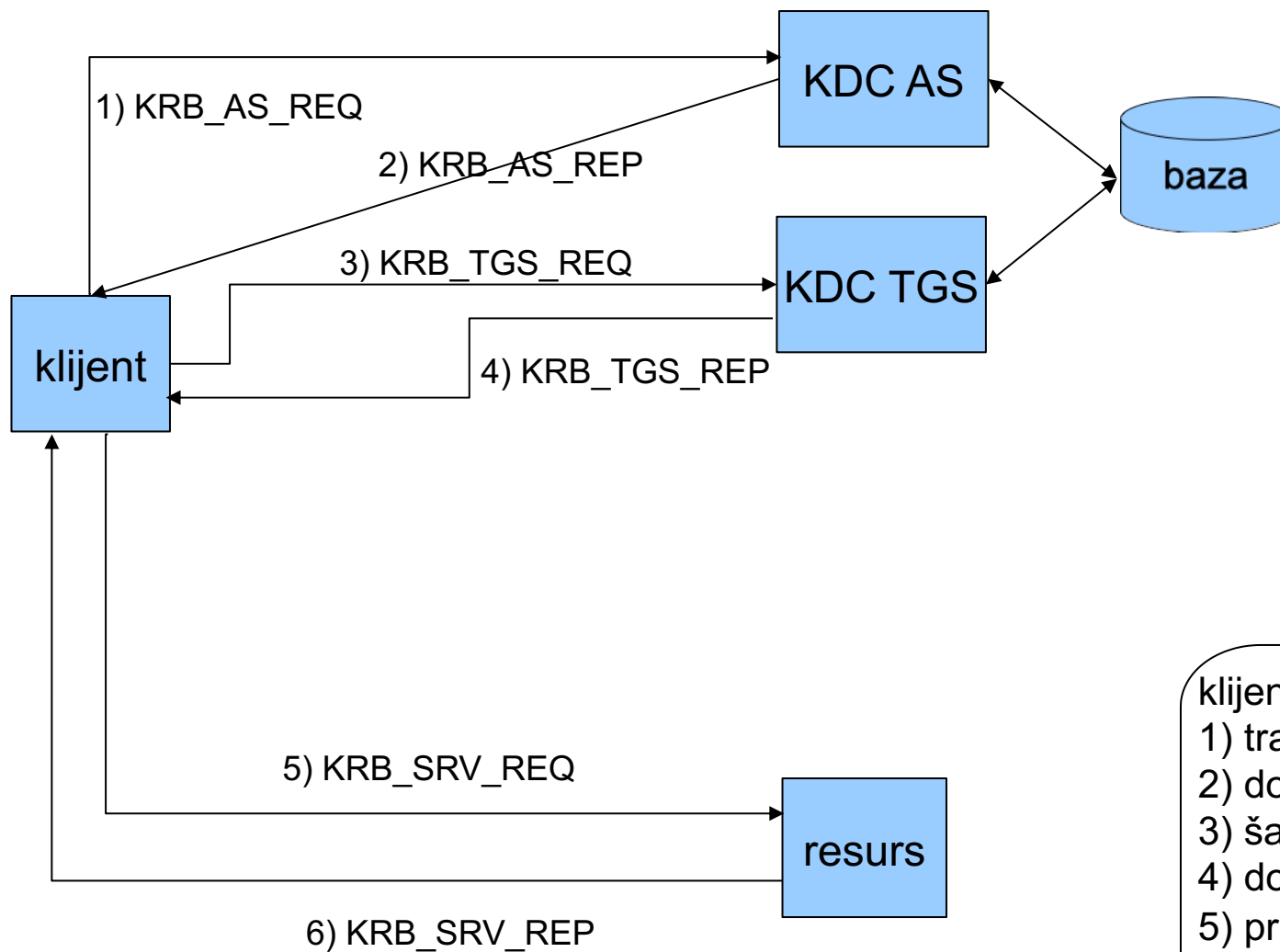
Kerberos KDC

- KDC mora da funkcioniše da bi klijenti mogli pristupati resursima
- potencijalno usko grlo i SPoF (single point of failure)
 - KDC na više servera
 - 1 master KDC
 - više slave KDC - preuzimaju ulogu ukoliko je primarni nedostupan
 - izmene baze samo na primarnom serveru!

Ticket-granting ticket

- TGT karta sadrži
 - ime principala koji traži pristup
 - ime principala kome se želi pristupiti
 - timestamp (trenutak formiranja zahteva)
 - rok važenja karte
 - listu IP adresa sa kojih se može koristiti karta
 - tajni ključ za komunikaciju sa resursom

Kerberos - tok komunikacije



klijent:

- 1) traži TGT
- 2) dobija TGT
- 3) šalje TGT, traži ST
- 4) dobija ST
- 5) pristupa uz ST
- 6) biva uslužen

Kerberos - tok komunikacije

1. KRB_AS_REQ zahtev - počinje autentifikaciju na KDC AS serveru
 - šalje se kao otvoreni tekst
 - sadrži
 - ime principala koji šalje zahtev
 - timestamp - vreme generisanja zahteva
 - ime principala TGS servera
 - traženi rok važenja karte

Kerberos - tok komunikacije

2. KRB_AS_REP odgovor

- AS proverava da li principal postoji u bazi
- ako postoji, proverava se da li je timestamp u dozvoljenom vremenskom okviru
- odgovor ima dva dela
 - prvi deo se šifruje tajnim ključem poznatom samo KDC-u i učesniku
- prvi deo sadrži
 - ključ sesije koji će klijent u nastavku koristiti za komunikaciju sa TGS serverom
 - ime principala TGS servera
 - rok važenja karte
- drugi deo sadrži
 - TGT kartu šifrovanu tajnim ključem koji koriste AS i TGS (TGS key)
 - session ključ za komunikaciju klijent-TGS
 - ime principala Kerberos klijenta
 - rok važenja karte
 - timestamp KDC servera
 - klijentova IP adresa
 - klijent ne može da pročita ovaj deo poruke!
 - klijent će je sačuvati u kešu i koristiti prilikom slanja zahteva za pristup resursima

Kerberos - tok komunikacije

3. KRB_TGS_REQ zahtev

- klijent dešifruje prvi deo KRB_AS_REP odgovora
- ključ sesije i šifrovanu TGT kartu smešta u keš
- zahtev za pristup konkretnom resursu klijent šalje TGS serveru
- sadrži
 - ime principala resursa kome klijent želi da pristupi
 - traženi rok važenja karte
 - TGT kartu iz prethodnog koraka
 - autentifikator
 - obezbeđuje jedinstvenost svakog zahteva za pristup resursu
 - potvrđuje da korisnik ima prethodno ugovoren tajni ključ sesije

Kerberos - tok komunikacije

- KRB_TGS_REP odgovor
- TGS server formira novi ključ sesije koji će klijent koristiti za komunikaciju sa resursom
- odgovor (ponovo) ima dva dela
- prvi deo je šifrovan ključem sesije koji je ustanovljen u koracima 1 i 2
 - ime principala resursa kome klijent želi da pristupi
 - rok važenja karte
 - ključ sesije za komunikaciju sa resursom kome klijent želi da pristupi
- drugi deo je ST karta za pristup resursu šifrovana tajnim ključem koji dele TGS i resurs
 - ključ sesije za komunikaciju klijent-resurs
 - ime principala klijenta
 - rok važenja karte
 - timestamp KDC servera
 - klijentova IP adresa
- ... klijent dalje pristupa resursu koristeći ST kartu

HTTP autentifikacija

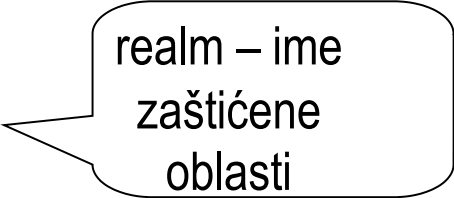
- postoje dva tipa autentifikacije po HTTP standardu
 - Basic Authentication
 - relativno često korišćena
 - Digest Access Authentication
 - vrlo retko korišćena
 - često nepravilno implementirana

HTTP autentifikacija

- postoje dva tipa autentifikacije po HTTP standardu
 - Basic Authentication
 - relativno često korišćena
 - Digest Access Authentication
 - vrlo retko korišćena
 - često nepravilno implementirana

HTTP Basic Authentication

- klijenti se identifikuju na osnovu korisničkog imena i lozinke
- izvodi se na sledeći način:
 - klijent traži željeni resurs
 - server proverí da li je pristup resursu ograničen
 - ako je ograničen, proverí da li je klijent već ranije poslao username:password
 - ako nije, kao odgovor mu se šalje:
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm=OBLAST
- klijent prikaže prozor za unos username:password i ponovo traži željeni resurs
- ako username:password nisu ispravni, dobija se isti odgovor (401)



realm – ime
zaštićene
oblasti

HTTP Basic Authentication

- username:password se šalju u okviru zaglavlja HTTP zahteva
GET HTTP/1.1
. . .
Authorization: Basic cHJvYmE6cHJvYmE=
- ime i lozinka se pakuju kao **ime:lozinka** (razdvojeni dvotačkom)
i kodiraju **Base64** algoritmom
- Base64 algoritam nije kriptografski algoritam
 - predstavlja brojeve u brojnom sistemu sa osnovom 64
 - najveća osnova brojnog sistema takva da se brojevi mogu predstaviti ASCII karakterima
 - služi za pakovanje binarnih sadržaja u tekstualni format
 - email poruke sa zakačenim binarnim fajlovima

HTTP Basic Authentication

- jednostavan mehanizam za autentifikaciju
- username:password putuju od klijenta do servera kao otvoreni tekst
- nema zaštite od prisluškivanja
- nizak nivo zaštite

HTTP Digest Access Authentication

- ispravlja osnovnu manu Basic metode
 - korisničko ime i lozinka se ne šalju preko mreže, već samo njihov hash kod
 - na serverskoj strani se ne mora čuvati ime i lozinka, već njihov hash kod
- radi po challenge-response principu:
 - server pošalje klijentu kodiranu informaciju
 - klijent vraća korisničko ime, lozinku i kodiranu informaciju

HTTP Digest Access Authentication

- tok komunikacije
 - klijent traži željeni resurs
 - server proverava da li je pristup resursu ograničen
 - ako jeste, proverava da li je klijent već ranije poslao username:password
 - ako nije, kao odgovor mu se šalje:
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest realm=OBLAST, nonce="...",
...
- klijent prikaže prozor za unos username:password i ponovo traži željeni resurs
- ako username:password nisu ispravni, dobija se isti odgovor (401)

HTTP Digest Access Authentication

- struktura WWW-Authenticate zaglavlja:

WWW-Authenticate: Digest

```
realm="IME",  
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",  
qop="auth,auth-int",  
opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

- realm – ime zaštićene oblasti
- nonce – jednokratna slučajna vrednost
- qop (quality of protection) – auth (authentication) i/ili auth-int (authentication with integrity protection)
- opaque – string koji bi klijent trebalo da pošalje za svaki naredni zahtev unutar iste oblasti

HTTP Digest Access Authentication

- odgovor klijenta stiže u sledećem zahtevu, u zaglavlju Authorization:

Authorization: Digest

```
username="proba",  
realm="IME",  
qop="auth",  
algorithm="MD5",  
uri="/protect.html",  
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",  
nc=00000001,  
cnonce="aa671f12a8587e2cffe73519863f9dbb",  
opaque="5ccc069c403ebaf9f0171e9517f40e41",  
response="f90a24d42f7d59fa0496cbbce6aacfe6"
```

- uri – zahtevani resurs
- nc – brojač zahteva za istim nonce poljem
- cnonce – koristi se za izračunavanje response
- response – sadrži hash kod za korisničko ime i lozinku

HTTP Digest Access Authentication

- izračunavanje response polja

HA1=MD5(username + ":" + realm + ":" + password)

HA2=MD5(http_method + ":" + uri)

response=MD5(HA1 + ":" + nonce + ":" + nc + ":" + cnonce + ":" + qop + ":" + HA2)

- MD5 – standardna kriptografska hash funkcija

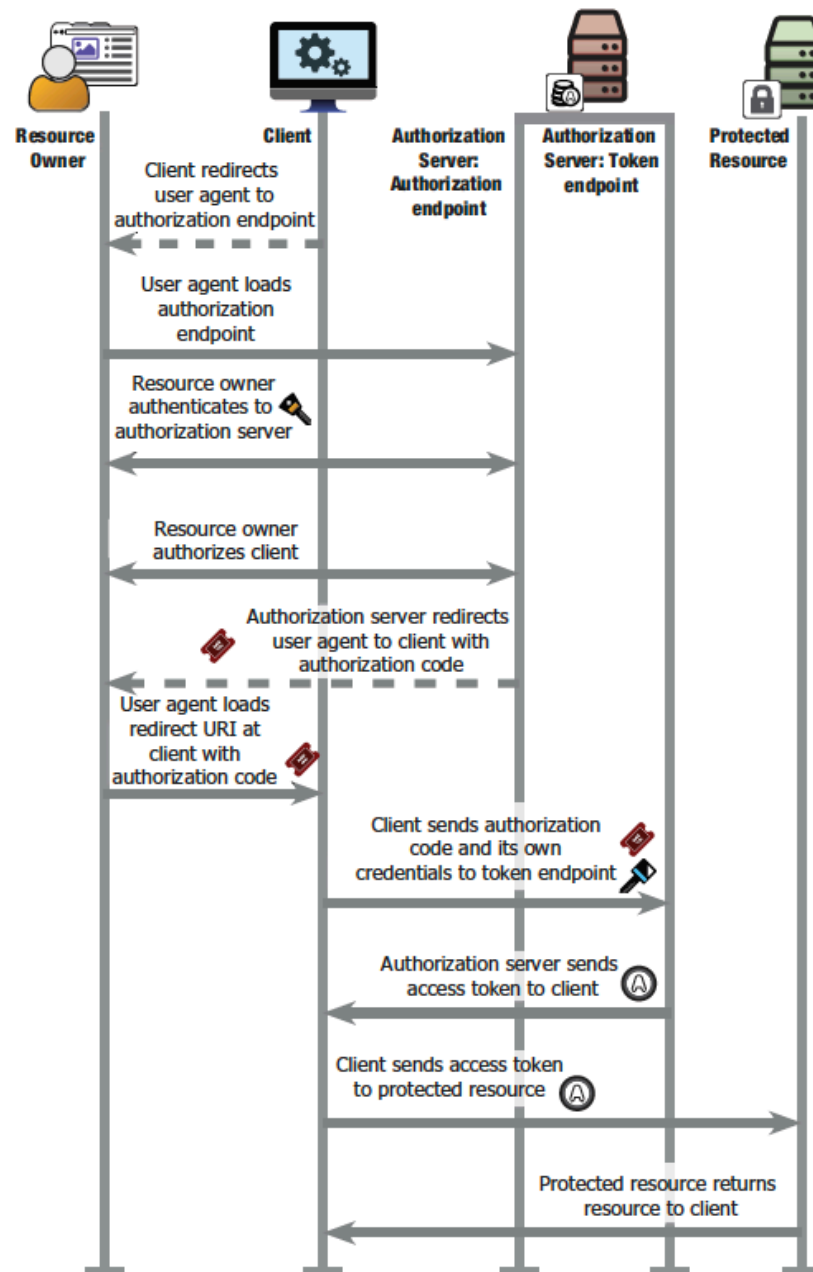
HTTP Digest Access Authentication

- lozinka ne putuje preko mreže, već samo njen hash
- otkrivanje lozinke prisluškivanjem nije moguće
- preneti sadržaji i dalje nisu zaštićeni od prisluškivanja (samo lozinka jeste)
- nema zaštite od man-in-the-middle napada

OAuth 2.0

- autorizacioni okvir koji omogućuje aplikacijama da pristupaju resursima u ime vlasnika resursa
- protokol za delegaciju
 - neko ko je vlasnik resursa dozvoljava nekom drugom da pristupi resursu u njegovo ime
 - nije baš protokol samo za autentifikaciju
- aplikacija od vlasnika zahtev autorizaciju i dobija token koji koristi za pristup
 - token = delegirano pravo pristupa
- napravljen za web sisteme
 - pre svega REST bazirane
- implicitno uključuje autentifikaciju

OAuth 2.0

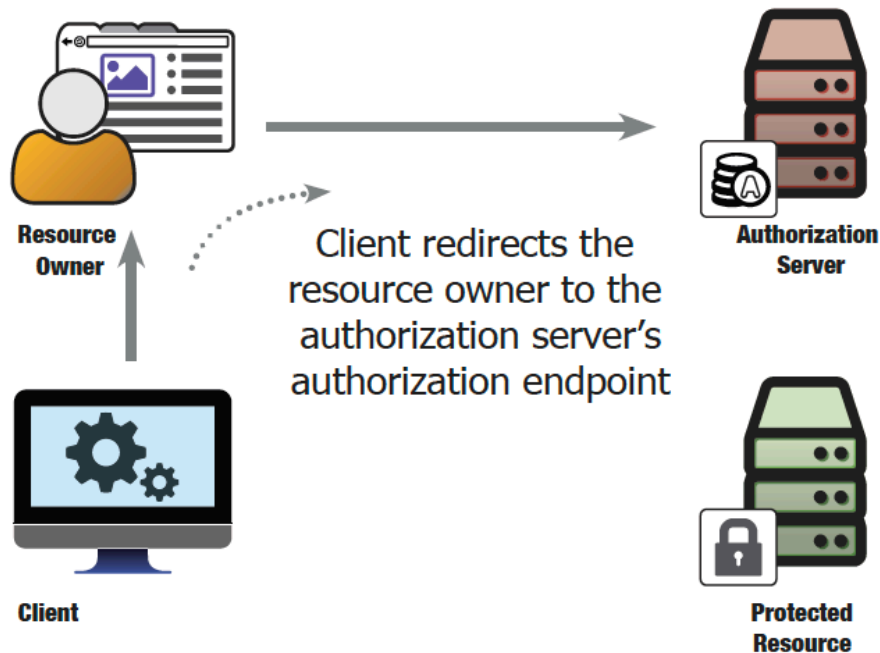


OAuth 2.0

- kada klijent zaključi da mu treba OAuth access token
redirektuje *resource owner-a* na autorizacioni server

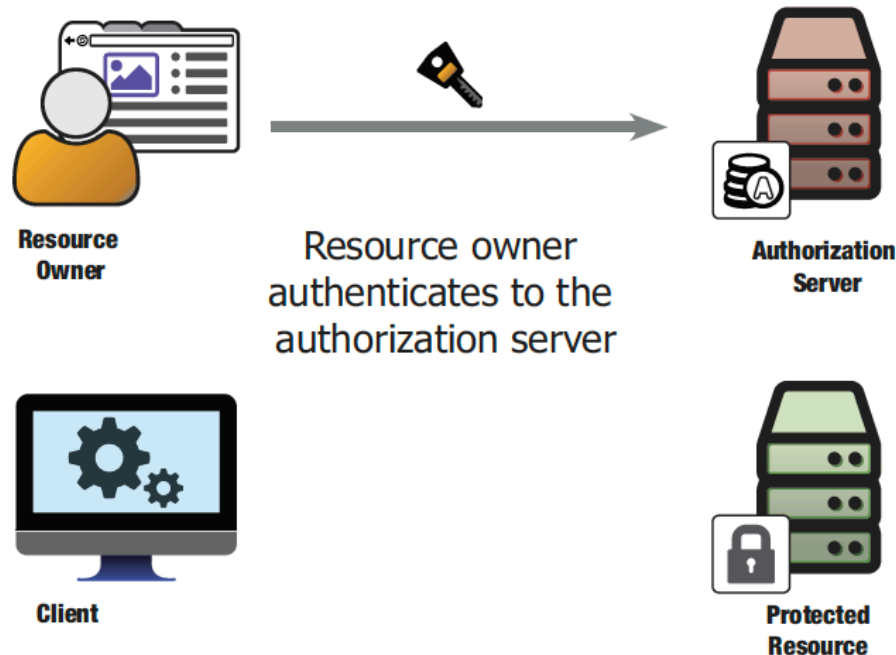
HTTP/1.1 302 Moved Temporarily

Location: `http://localhost:9001/authorize?response_type=code
&scope=foo&client_id=oauthclient1&redirect_uri=http%3A%2F%2Flocalhost%3A9000%2Fcallback&state=Lwt50DDQKUB8U7jtfLQCVGDL9cnmwHH1`



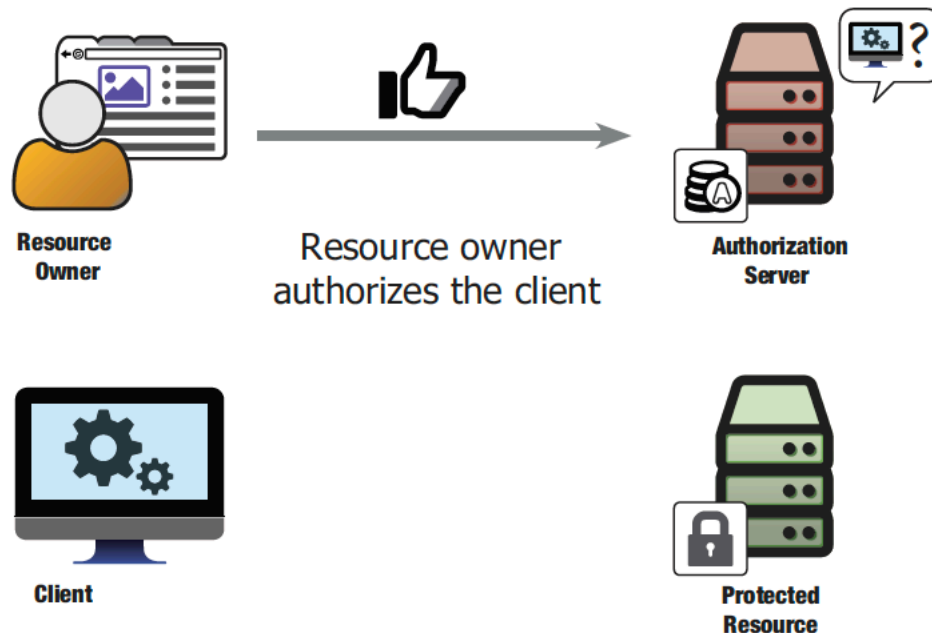
OAuth 2.0

- autorizacioni server zahteva od klijenta da se autentifikuje
 - ko je resource owner i šta može da delegira klijentu
 - autentifikacije uvek između korisnika i servera
 - klijent ne učestvuje
 - nema deljenja kredencijala sa klijentom
- način autentifikacije je proizvoljan



OAuth 2.0

- resource owner autorizuje klijentsku aplikaciju
 - deo prava se delegira klijentskoj aplikaciji
- server može da kešira odluke za buduće identične zahteve
- server može da „pregazi“ odobrenje resource owner-a

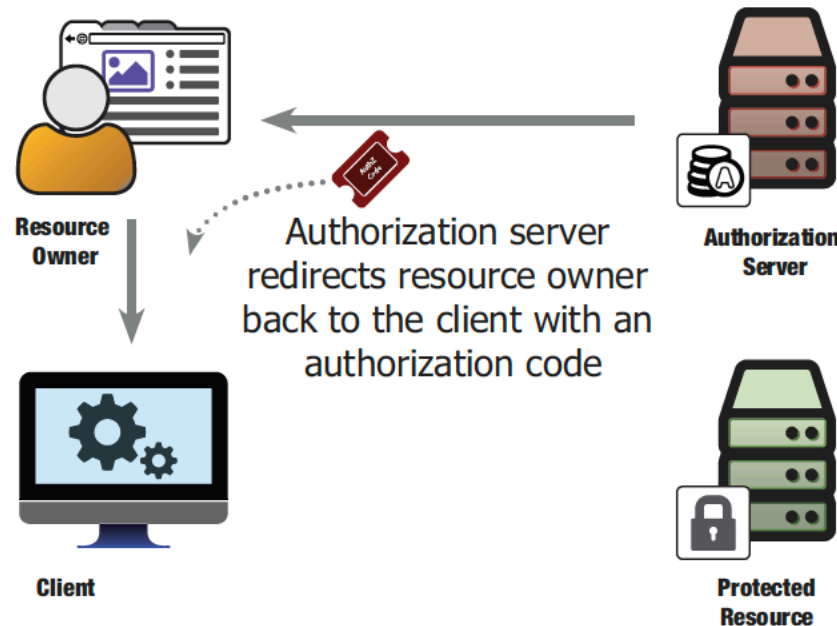


OAuth 2.0

- autorizacioni server redirektuje resource owner-a nazad na klijentsku aplikaciju
 - u primeru se kao grant type koristi *authorization code*
 - parametar *code* predstavlja one-time-use kredencijale
 - klijent ga koristi za kasnije aktivnosti
 - parametra *state* mora biti identičan kao na početku

HTTP 302 Found

Location: http://localhost:9000/oauth_callback?code=8V1pr0rJ&state=Lwt50DDQKUB8U7jtfLQCVGDL9cnmwHHI



OAuth 2.0

- klijent šalje vrednost parametra *code* autorizacionom serveru zajedno sa svojim kredencijalima

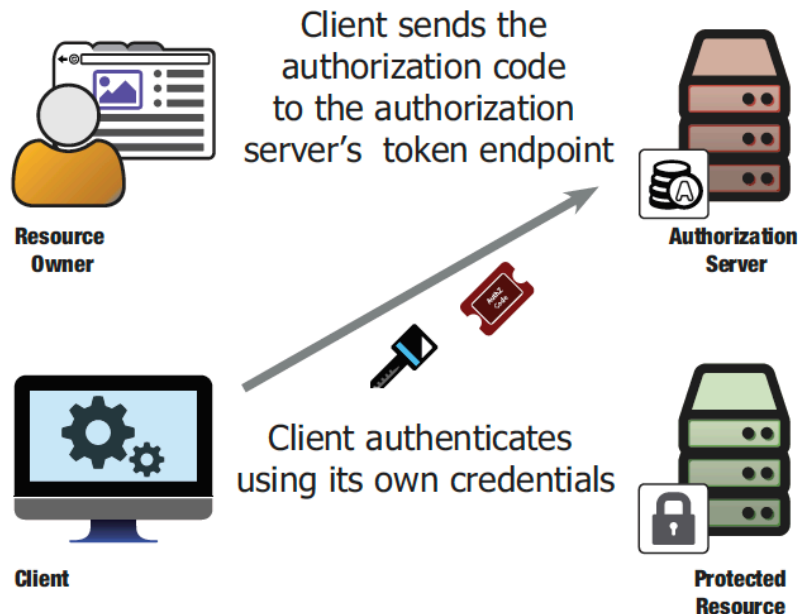
POST /token

Host: localhost:9001

Accept: application/json

Content-type: application/x-www-form-encoded

Authorization: Basic b2F1dGgtY2xpZW50LTE6b2F1dGgtY2xpZW50LXN
grant_type=authorization_code&redirect_uri=http%3A%2F%2Flocalhost%3A9000%2Fcallback&**code**=8V1pr0rJ



OAuth 2.0

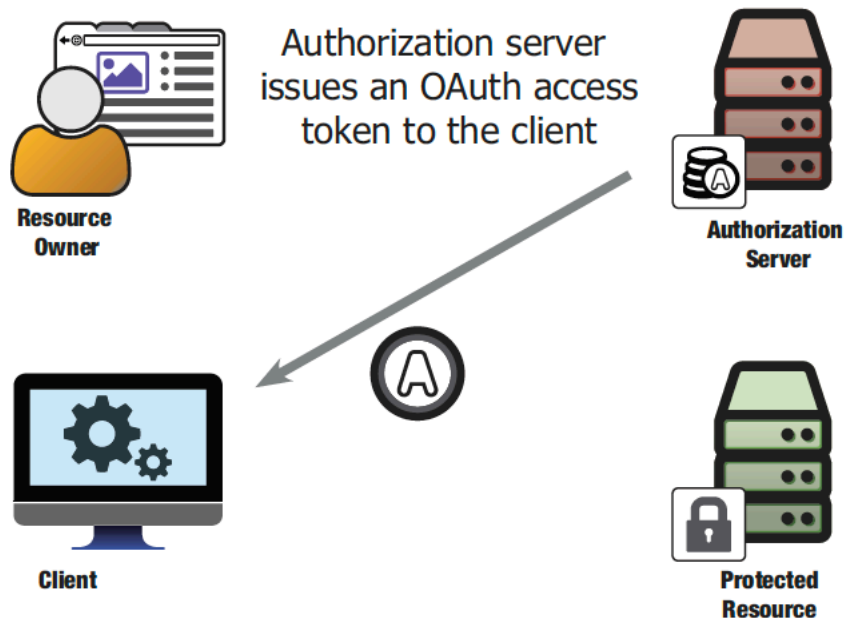
- autorizacioni server izdaje (access) token klijentu

HTTP 200 OK

Date: Fri, 31 Jul 2015 21:19:03 GMT

Content-type: application/json

```
{  
  "access_token": "987tghjkiu6trfghjuytrghj",  
  "token_type": "Bearer"  
}
```



OAuth 2.0

- sa dobijenim (access) tokenom klijent pristupa resursu
 - resurs parira i validira token

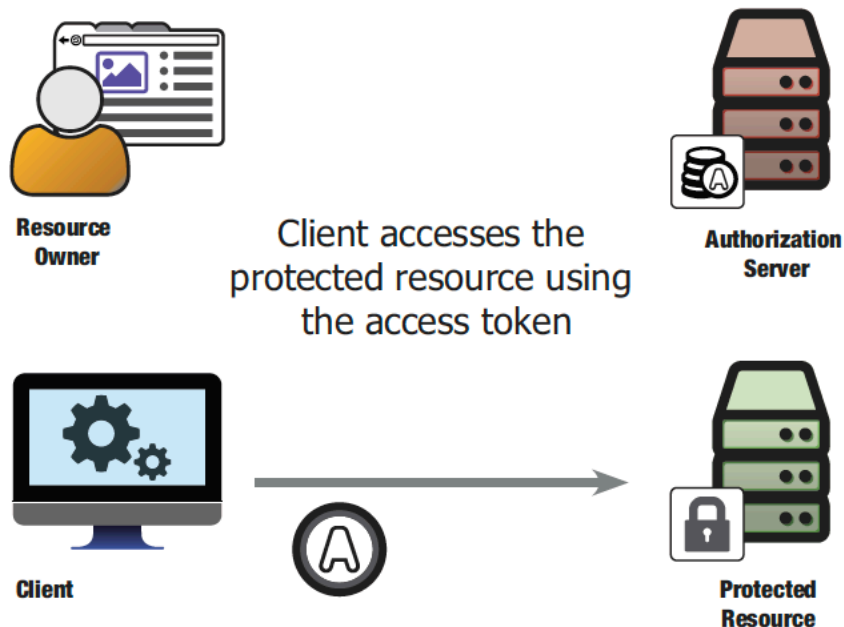
GET /resource HTTP/1.1

Host: localhost:9002

Accept: application/json

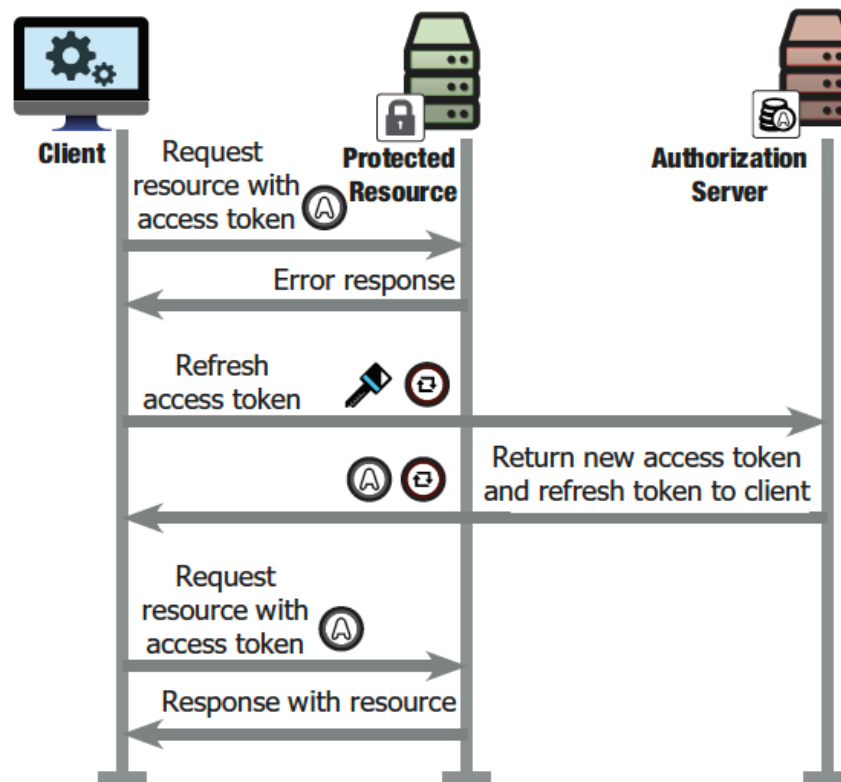
Connection: keep-alive

Authorization: Bearer 987tghjkiu6trfghjuytrghj



OAuth 2.0

- access token može biti povučen ili da ima vreme trajanja
- refresh token
 - služi da klijent zatraži novi access token bez kontaktiranja resource owner-a



OAuth 2.0

- *authorization token*
 - standard ne definiše format i sadržaj tokena
 - može da sadrži bilo šta
 - obično JSON format
 - klijent nema potrebe za razumevanje tokena
 - samog ga prosleđuje u zahtevu
 - jednostavnija implementacija
 - server mora biti u stanju da pročita i razume token
 - tri načina slanja
 - HTTP authorization header – preporučen način
 - form-encoded request body
 - URL/encoded param

OAuth 2.0

- *scope*
 - skup prava za zaštićeni resurs - grupisanje prava
 - jedan string ili sekvenca stringova razdvojeni razmakom
 - njihovo definisanje obično zavisi od API-ja

OAuth 2.0

- *refresh token*
 - klijent ga koristi za obnavljanje access tokena, kada on istekne bez potrebe da se u obnavljanje uključi korisnik

OAuth 2.0

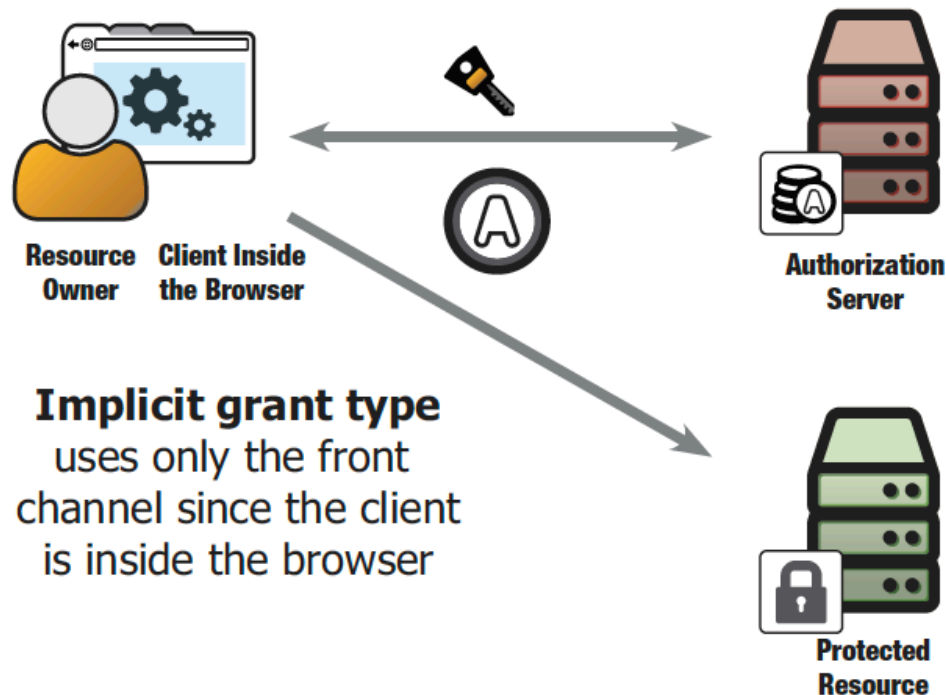
- *authorization grant*
 - metoda za dobijanje tokena
 - eksplicitni
 - implicitni
 - client credentials grant
 - resource owner credentials grant
 - assertion grant

OAuth 2.0

- *eksplicitni authorization grant*
 - klijent i korisnik su zasebne (odvojene komponente)
 - *response_type=code*
 - radi se i autentifikacija i autorizacija klijenta
 - prethodno opisani tok

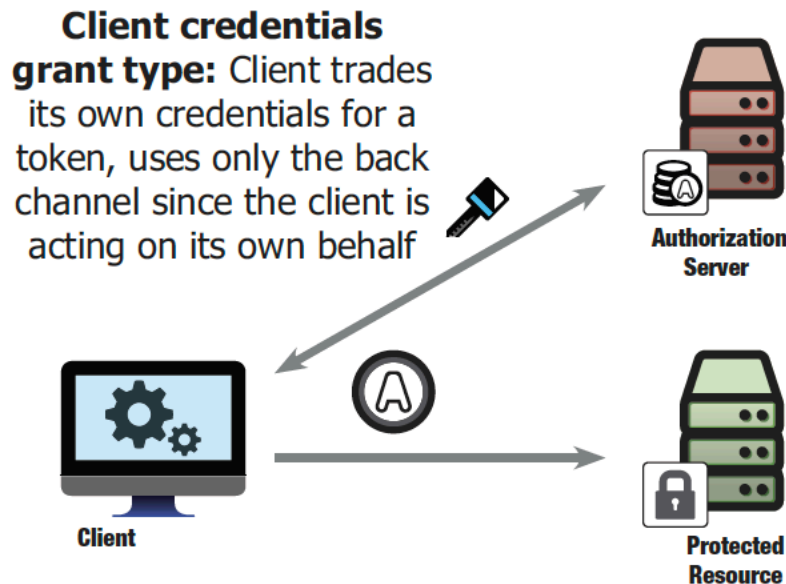
OAuth 2.0

- *implicit authorization grant*
 - klijent je korisnikov brauzer
 - *response_type=token*
 - autorizacioni server odmah generiše token



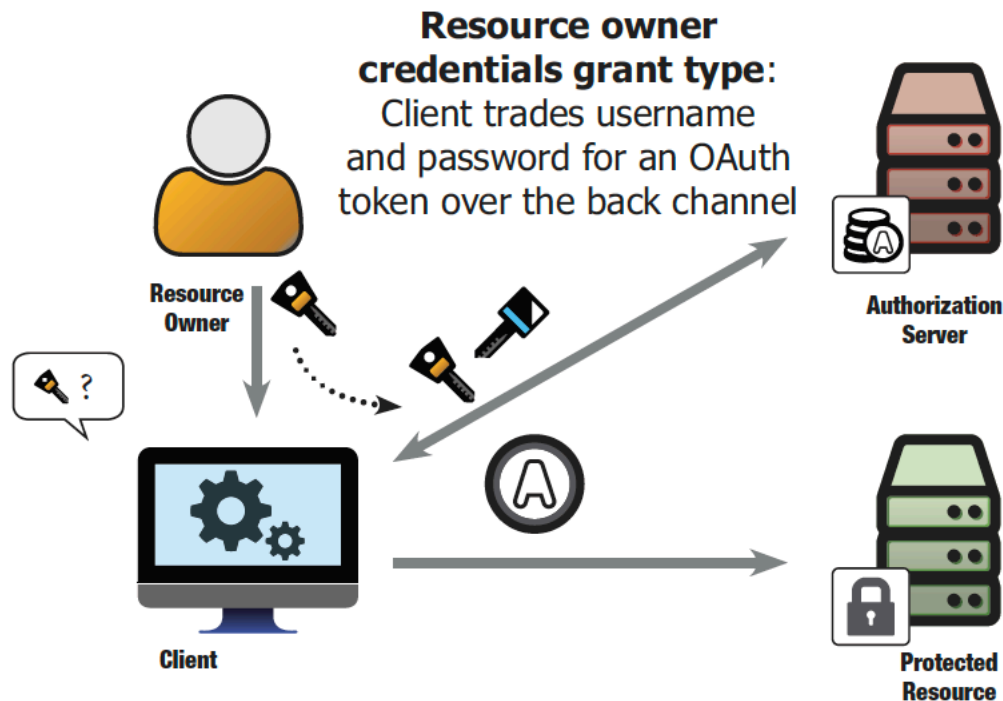
OAuth 2.0

- *client credentials authorization grant*
 - nema eksplicitnog vlasnika resursa
 - za server-server komunikacija
 - *grant_type= client_credentials* (kao i za eksplicitni tip)
 - ne postoji autorizacioni kod (kao kod eksplicitnog tipa)



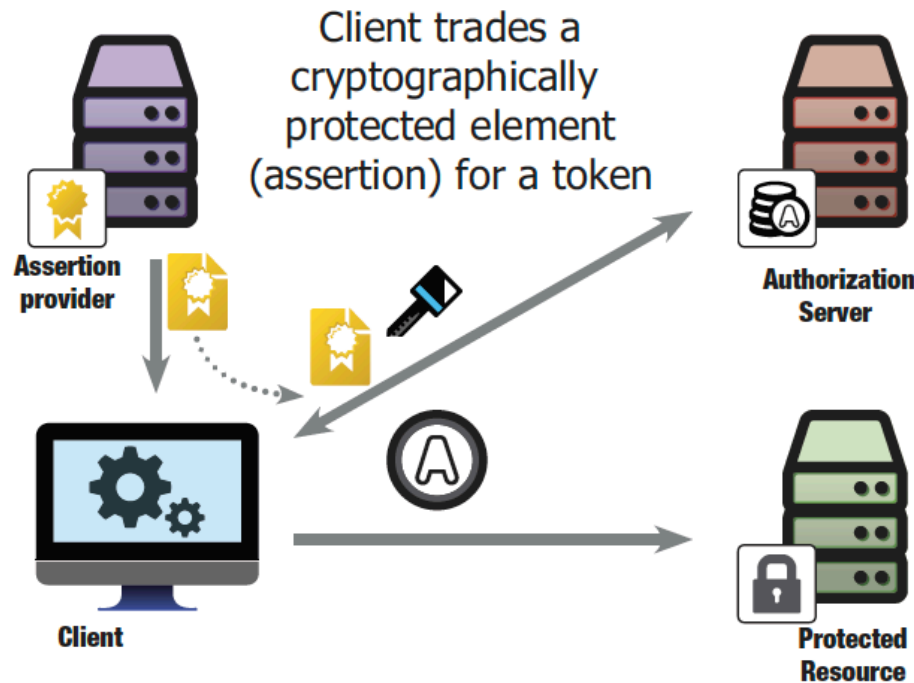
OAuth 2.0

- *resource owner credentials grant*
 - klijent od korisnika trazi kredencijale sa kojima se autentifikuje serveru
 - *grant_type= password* (npr.)
 - nije preporučljiv, osim ako je neophodan



OAuth 2.0

- *assertion grant*
 - klijent dobije kriptografski zaštićen podatka *assertation* koji koristi za autentifikaciju
 - Format SAML ili JWT
 - Asertacije mogu biti na nivou korisnika, konfiguracije, ...

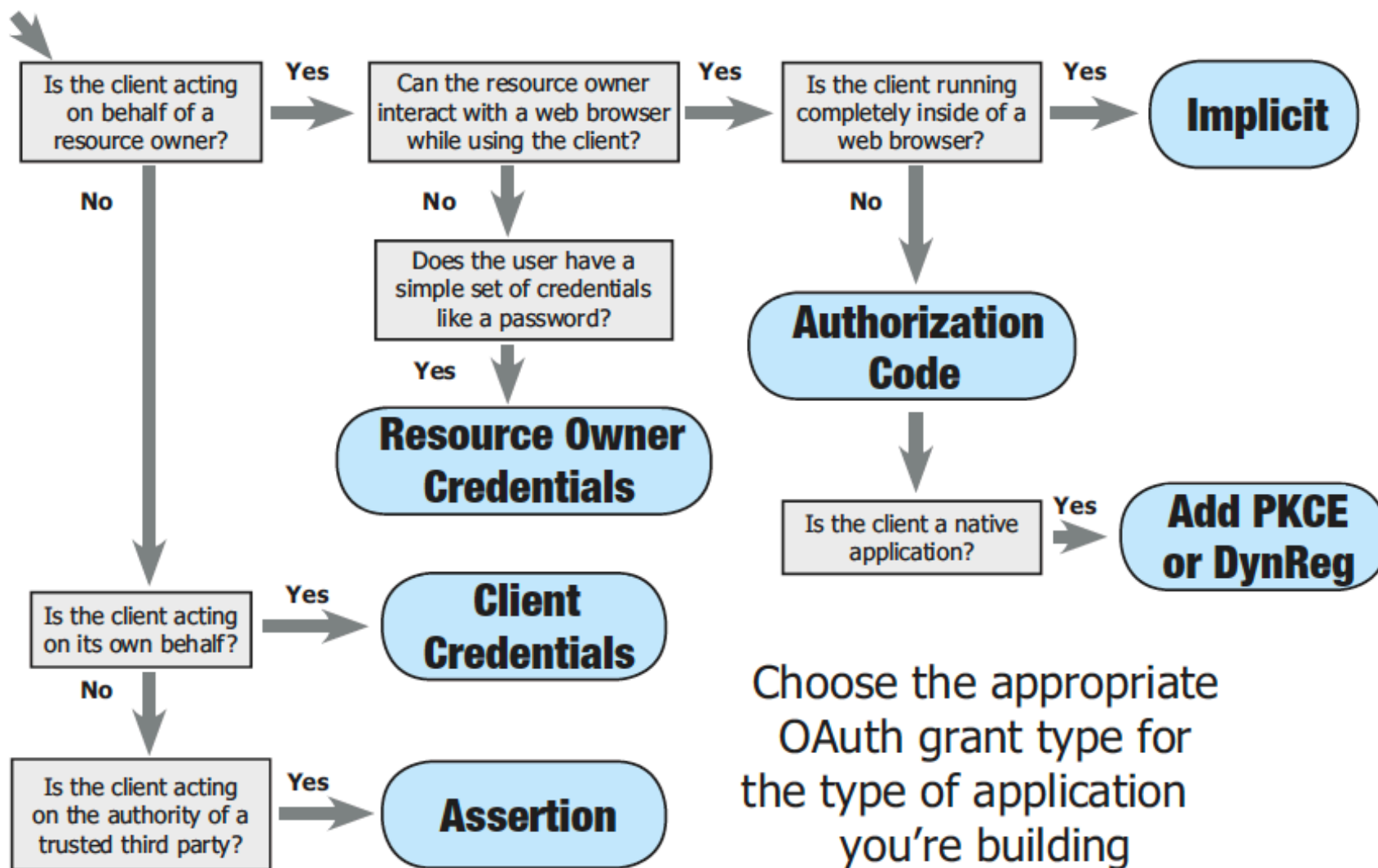


OAuth 2.0

- vrste klijenata
 - javni
 - ne vrši se autentifikacija klijenta
 - poverljivi
 - klijenti poseduje autentifikacione podatke
 - svaka instanca klijenta ima posebnu konfiguraciju

OAuth 2.0

- *koji tok (grant type) odabrati*



Poverenje u sisteme za autentifikaciju

- Ken Thompson: Reflections on Trusting Trust
 - govor povodom prijema Turingove nagrade
 - opisuje korišćenje samo-izmenjujućih programa za postavljanje zadnjih vrata (*backdoor*)
- korak 1: izmeniti C kompajler tako da kada prevodi UNIX login komandu za određenu lozinku korisniku daje sva prava
 - ovakav trojanski konj je lako uočljiv pregledom sors koda kompajlera
- korak 2: izmeniti C kompajler tako da kada prevodi C kompajler ugradi prethodnu izmenu
 - ovo se teže otkriva
 - niko ne proverava prevedeni kod kompajlera!
- korak 3: ukloniti prethodne izmene u sorsu C kompajlera i prevesti ga ponovo
 - "trojanske" izmene će postojati u prevedenom kodu kompajlera iako ih nema u sorsu!
- zaključak: ne možemo verovati nijednom programu koji nismo sami napisali
 - naročito ne programima koje proizvode kompanije koje zapošljavaju ljude kao što je K.T. ;)
 - "trojanca" je moguće podmetnuti i u disasembler tako da se izmena ne može videti ni inspekcijom prevedenog koda
 - jedino da sami pišemo disasembler, ali ko to još danas radi ;)