**UNIVERZITET U BANJOJ LUCI**

**ELEKTROTEHNIČKI FAKULTET**

**Rajo Gajić**

**REALIZACIJA SSO SERVERA**

**Diplomski rad**

**Banja Luka, 2017**

**Tema: REALIZACIJA SSO SERVERA**

**Komisija: prof. dr Slavko Marić**

**prof. dr Zoran Đurić**

**mr Ognjen Joldžić**

**kandidat:**

**Rajo Gajić**

UNIVERZITET U BANJOJ LUCI

ELEKTROTEHNIČKI FAKULTET

KATEDRA ZA RAČUNARSKU TEHNIKU

Predmet: INTERNET PROGRAMIRANJE

Tema: REALIZACIJA SSO SERVERA

Zadatak:

Uvod. Autentifikacija i upravljanje identitetima. Autorizacija i kontrola pristupa. SAML. XACML. Dati pregled i analizu postojećih SSO rješenja. Realizovati SSO server, sa osnovnim SSO funkcionalnostima, koji omogućava autentifikaciju i autorizaciju korisnika. Dati uporednu analizu realizovanog rješenja sa postojećim rješenjima.

Mentor: prof. dr Zoran Đurić

Kandidat: Rajo Gajić (22/07)

Banja Luka, 2017

Sadržaj

[1 UVOD 1](#_Toc475293081)

[2 Single-sing-on (SSO) 2](#_Toc475293082)

[3 Autentifikacija 6](#_Toc475293083)

[3.1 Autentifikacija upotrebom lozinke 6](#_Toc475293084)

[3.2 Autentifikacioni faktori 7](#_Toc475293085)

[4 Upravljanje identitetima 7](#_Toc475293086)

[5 Autorizacija i kontrola pristupa 9](#_Toc475293087)

[5.1 Autorizacija 9](#_Toc475293088)

[5.2 Kontrola pristupa 10](#_Toc475293089)

[6 SAML (Security Assertion Markup Language) 10](#_Toc475293090)

[6.1 Principi 10](#_Toc475293091)

[6.2 Kako SAML funkcioniše 11](#_Toc475293092)

[6.2.1 Assertions (tvrdnje) 11](#_Toc475293093)

[6.2.2 Protocols (protokoli) 12](#_Toc475293094)

[6.2.3 Bindings (povezivanja) 12](#_Toc475293095)

[6.2.4 Profiles (profili) 13](#_Toc475293096)

[6.3 SAML 2.0 13](#_Toc475293097)

[7 XACML (Extensible Access Control Markup Language) 13](#_Toc475293098)

[7.1 Konstrukcije najvišeg nivoa: propisi (Policy) i skupovi propisa (PolicySet) 14](#_Toc475293099)

[7.2 Ciljevi (Targets) i pravila (Rules) 15](#_Toc475293100)

[7.3 Atributi (Attributes), vrijednosti atributa (Attribute Values) i funkcije (Functions) 15](#_Toc475293101)

[7.4 Primjer jednog propisa 17](#_Toc475293102)

[8 Izbor strategije za realizaciju SSO servera 18](#_Toc475293103)

[8.1 SAML 2.0 19](#_Toc475293104)

[8.1.1 SAML token (Token) vs. SAML tvrdnja (Assertion) 20](#_Toc475293105)

[8.1.2 SAML ograničenja 20](#_Toc475293106)

[8.1.3 Zaobilazno rješenje SAML HTTP POST uvezivanja 21](#_Toc475293107)

[8.2 OAuth 2.0 22](#_Toc475293108)

[8.2.1 OAuth2 nedostaci 24](#_Toc475293109)

[8.3 Rezime – SAML vs. OAuth2 24](#_Toc475293110)

[9 Implementacija SSO servera 26](#_Toc475293111)

[9.1 Web Browser SSO profil 26](#_Toc475293112)

[9.2 Integracija SSO klijenata sa SSO serverom 30](#_Toc475293113)

[9.2.1 Pozadinski proces prilikom pristupa zaštićenoj stranici na strani SP-a 31](#_Toc475293114)

[9.3 Konfigurisanje i pokretanje SP i IdP aplikacija 33](#_Toc475293115)

[10 Literatura 35](#_Toc475293116)

# UVOD

U današnje vrijeme gotovo svaki web sajt zahtjeva neki oblik provjere identiteta za pristup sadržajima i funkcionalnostima. Sa povećanjem broja sajtova i servisa, centralizovani sistem za prijavu korisnika postaje nužnost.

Koncept centralizovanog ili povezanog elektronskog identiteta poznat je kao *federated identity* (centralizovani identitet). Sistemi centralizovanog identiteta upravljaju sa sledećim problemima:

* Autentifikacija (provjera identiteta)
* Autorizacija (prava pristupa)
* Razmjena korisničkih atributa
* Upravljanje korisnicima

**Autentifikacijski** aspekt se zasniva na provjeri pristupnih podataka i na identifikaciji korisnika. **Autorizacija** je vezana za prava pristupa (npr. da li korisnik ima dozvolu da pristupi određenom resursu). **Razmjena korisničkih atributa** predstavlja dijeljenje informacija o korisniku između različitih sistema za upravljanje korisnicima. Na primjer, atribut „pravo ime“ može biti prisutan u nekoliko sistema. Sistem centralizovanog identiteta spriječava dupliranje podataka povezivanjem ekvivalentnih atributa u različitim sistemima. Na kraju, **upravljanje korisnicima** je vezano za administraciju (dodavanje, brisanje, ažuriranje) korisničkih naloga. Ovaj sistem omogućuje administratorima (ili korisnicima) da održavaju jedan korisnički nalog u okviru više domena ili podsistema.

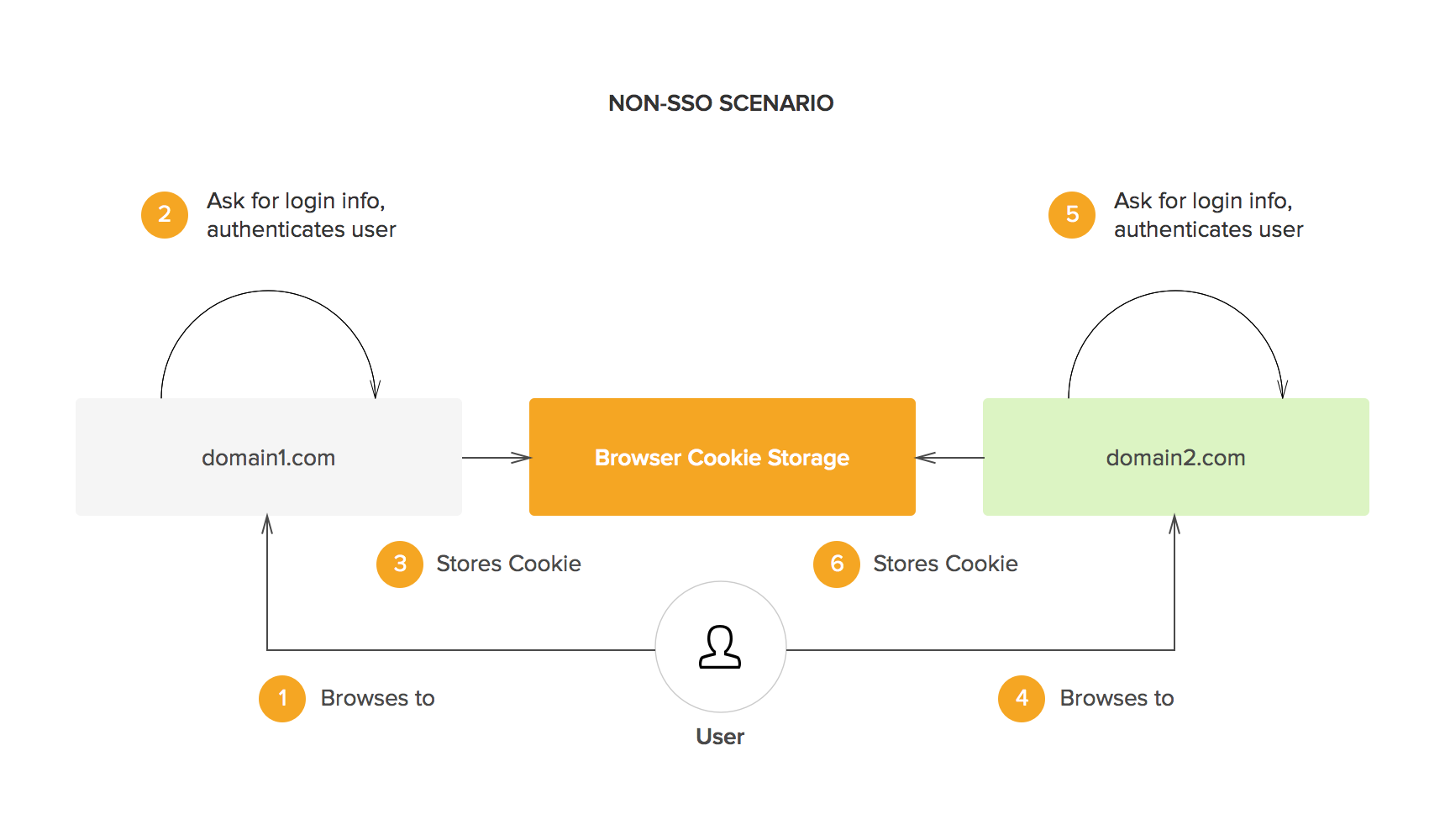
SSO je isključivo povezan sa **autentifikacijskim** dijelom sistema centralizovanog identiteta. Osnovni zadatak SSO servera je identifikacija korisnika i dijeljenje tih podataka sa svim podsistemima koji zahtijevaju podatke.

# Single-sing-on (SSO)

Prije ili kasnije razvojni tim naiđe na jedan problem: razvili ste aplikaciju na domenu X a sada želite da vaša nova aplikacija na domenu Y koristi iste pristupne podatke kao i domen X. U suštini, želite i više: da korisnici koji su već prijavljeni pod domenom X budu automatski prijavljeni i pod domenom Y. To je osnovni zadatak SSO servera.

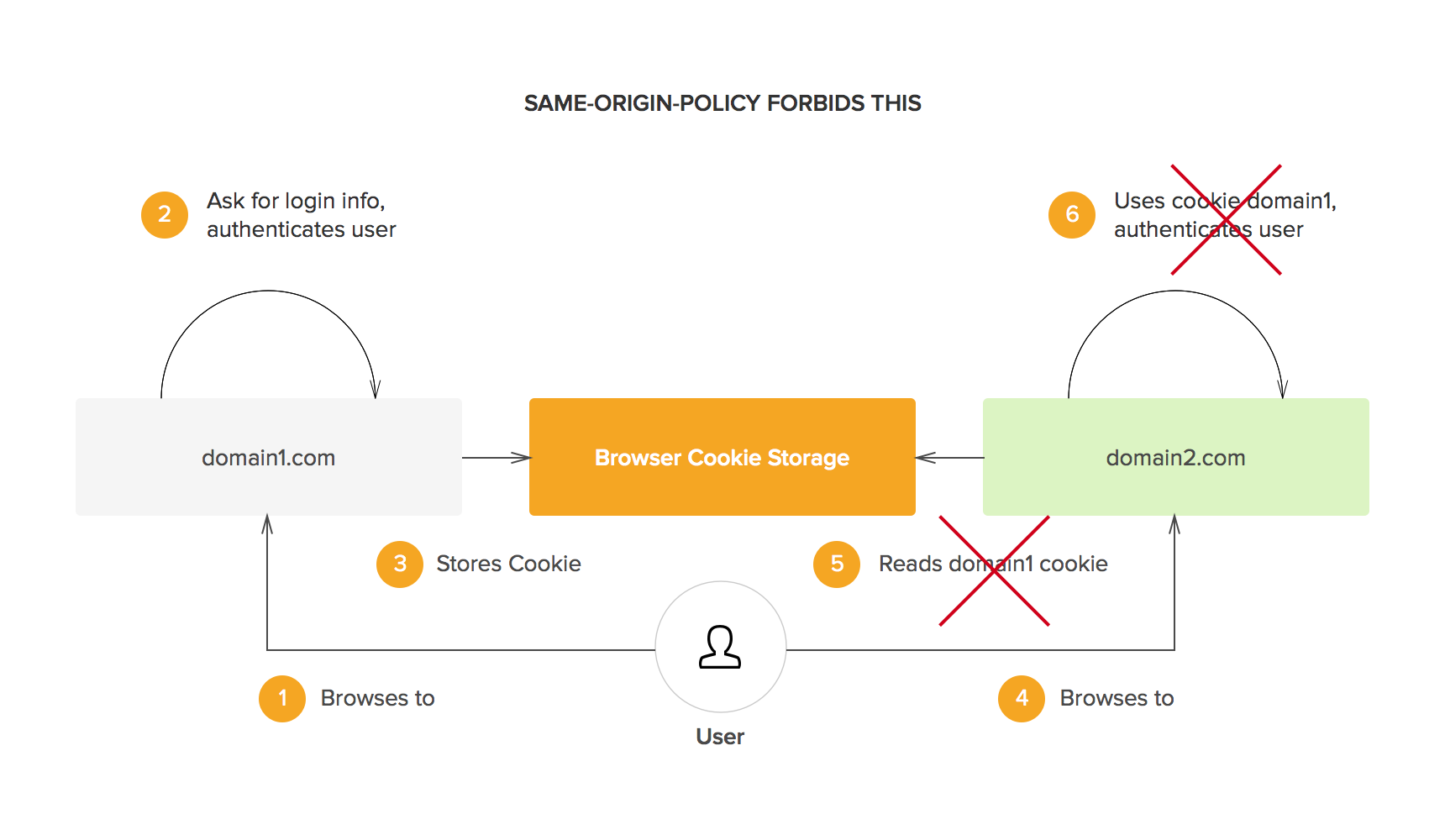
Npr. Google je realizovao SSO za svoje proizvode: Gmail, YouTube, Analytics itd. Kada na svom računaru pristupite Gmail aplikaciji, ulogujete se prvi put, a zatim posjetite YouTube, nećete morati ponovo unositi pristupne podatke nego ćete biti automatski prijavljeni.

Na **Slici 1** je prikazan scenario prijave korisnika na dvije različite aplikacije, pri čemu ne postoji SSO server.



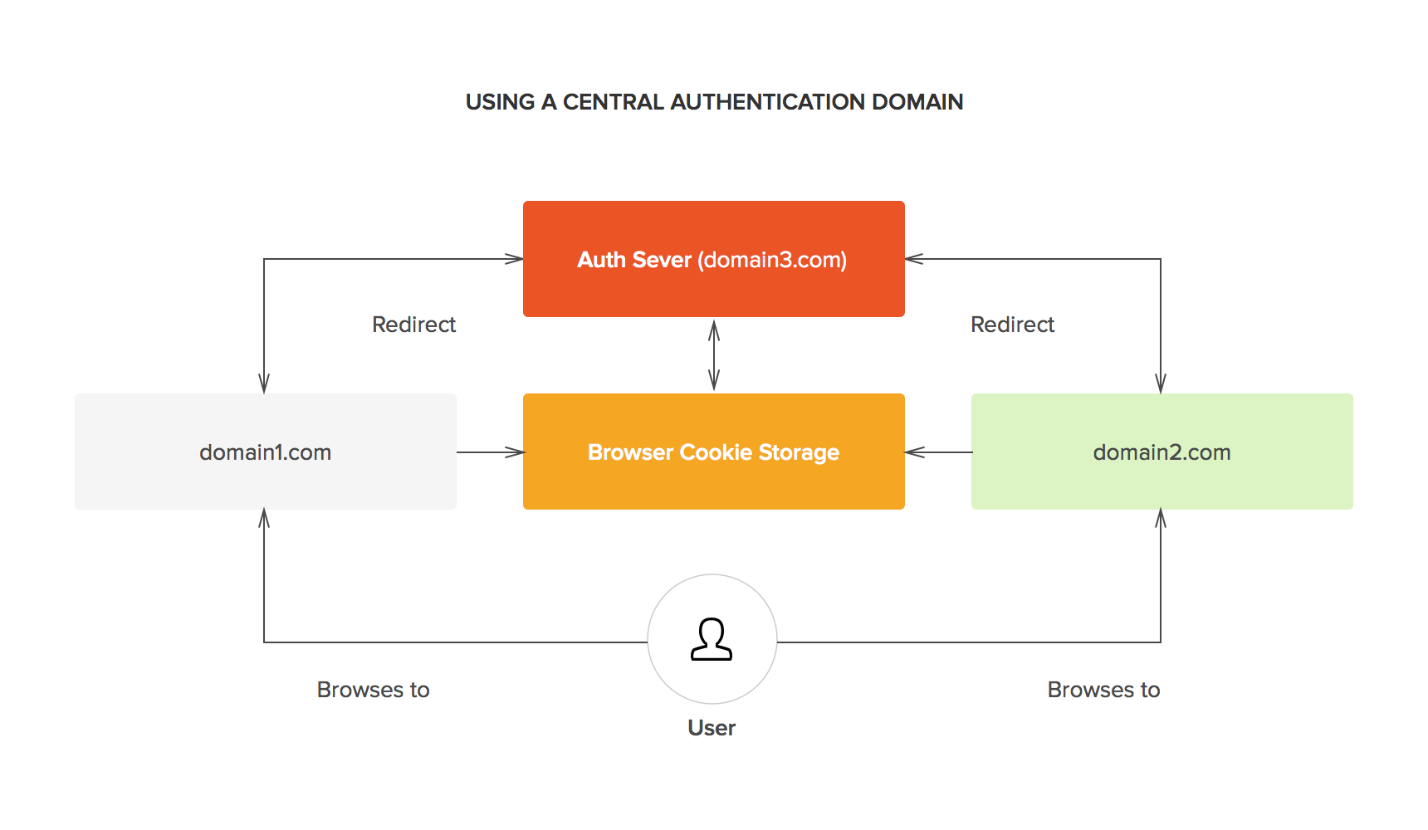
Slika 1 – Rješenje bez SSO servera

Očigledno rješenje ovog problema je dijeljenje informacija o sesijama između različitih domena, predstavljeno na **Slici 2**. Međutim, zbog sigurnosnih razloga, preglednik primjenjuje pravilo poznato kao *same-origin-policy* – pravilo istog porijekla. Ovo pravilo kaže da kolačićima (*cookies*) (i ostalim podacima lokalne pohrane podataka) može pristupiti samo onaj ko ih je napravio (tj domen koji je napravio zahtjev za čuvanje tih podataka). Drugim riječima, domen X ne može pristupiti kolačićima kreiranim od strane domena Y i obrnuto. To je ono što SSO rješava: dijeljenje sesijskih informacija između različitih domena.



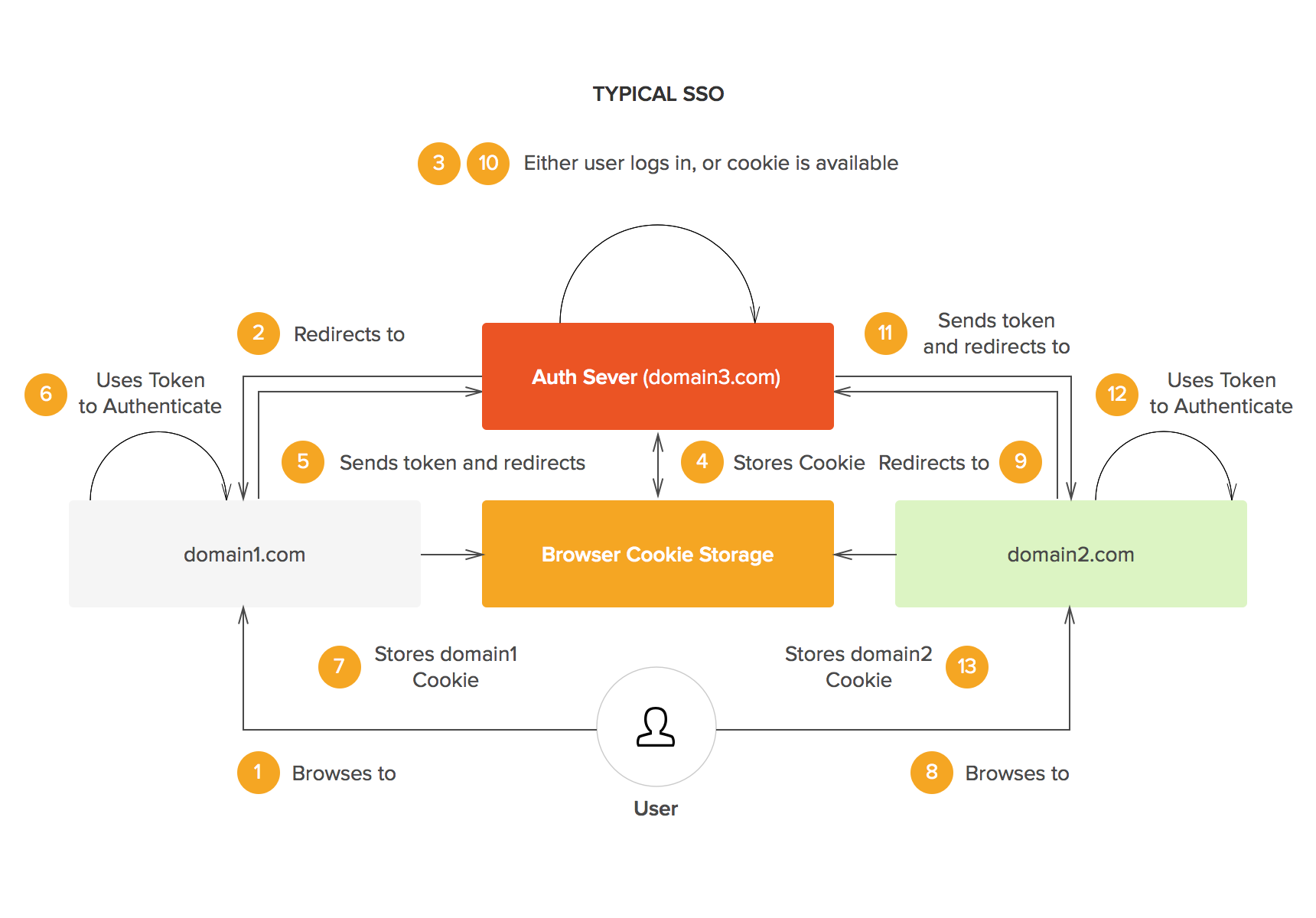
Slika 2 - Pravilo istog porijekla

Različiti SSO protokoli dijele sesijske podatke na različite načine, ali osnovni princip je isti – postoji centralni domen preko kog se vrši autentifikacija, nakon čega se sesija dijeli sa ostalim domenima na neki način. Npr. centralni domen generiše potpisani JSON Web Token (koji može biti šifrovan upotrebom JWE). Taj token se dalje može proslijediti klijentu i može se koristiti od strane autentifikacijskog domena kao i od strane ostalih domena. Token može biti proslijeđen izvornom domenu putem redirekcije, pri čemu on sadrži sve informacije potrebne za identifikaciju korisnika u okviru domena koji je zahtijevao autentifikaciju. S obzirom na to da je token potpisan, ne može biti izmijenjen od strane klijenta ni na koji način. Ovaj scenarion je prikazan na **Slici 3**.



Slika 3 - Upotreba centralnog autentifikacionog domena

Svaki put kada korisnik posjeti domen koji zahtjeva autentifikaciju, redirektovan je na autentifikacijski domen. S obzirom na to da je korisnik već ulogovan pod tim domenom, može odmah biti redirektovan na izvorni domen uz neophodni autentifikacijski token.



Slika 4 - Uobičajeni SSO scenario

Single sign-on (SSO) je oblik tehnologije koji olakšava proces autentifikacije kako za korisnike tako i za IT administratore. Ukoliko se koristi SSO sistem, dovoljno je da korisnik samo jednom unese korisničko ime i lozinku kako bi dobio pristup na vise aplikacija. Korisnicima se dodjeljuju prava pristupa određenim aplikacijama, i biće u stanju da pristupe tim aplikacijama kada unesu svoje pristupne podatke, što eliminiše višestruke unose i prijavu. SSO takođe smanjuje vrijeme i cijenu uređivanja velikog broja lozinki za IT osoblje.

SSO sistemi unaprijeđuju sigurnost time što se cjelokupna autentifikacija izvršava u okviru posebnog servera za tu namjenu. Svi autentifikacioni podaci moraju prvo proći kroz poseban server za SSO, koji zatim dalje proslijeđuje određene autentifikacione informacije koje posjeduje za datog korisnika. Ovakav način centralizovanog sistema za autentifikaciju će vjerovatno prije odoljeti napadima i zlonamjernom pristupu nego obični sistem autentifikacioni sistem. Dodatno, SSO sistemi obično obezbjeđuju bolju zaštitu osjetljivog materijala, s obzirom na to da su najčešće zaštićeni firewall-om.

SSO je takođe koristan za dokumentovanje pristupa i nadzor korisničkih naloga – npr. čišćenje neaktivnih korisničkih naloga zaposlenih i praćenje aktivnosti korisnika.

# Autentifikacija

Autentifikacija je proces utvrđivanja da li je neko stvarno onaj za koga se predstavlja. Pri tom procesu pristupni podaci se porede sa onim u bazi ili nekom drugom izvoru liste podataka registrovanih tj. ovlaštenih korisnika. Ako se pristupni podaci poklapaju, proces autentifikacije je uspješno završen i korisnik dobija ovlašćenje za pristup sistemu.

Autentifikacija može biti sprovedena upotrebom pristupnih podataka, SSO sistema, biometrijski, digitalnim sertifikatima i javnim ključevima.

Autentifikacija korisnika je ključna za osiguravanje odgovarajućih ovlašćenja i pristupa sistemima i uslugama, naročito od kada je sve učestalija krađa podataka i sve su veće prijetnje sigurnosti informacija. Iako autentifikacija ne može u potpunosti spriječiti krađu informacija i identiteta, možemo se pobrinuti da naši resursi budu zaštićeni upotrebom nekoliko autentifikacijskih metoda.

## Autentifikacija upotrebom lozinke

U privatnim i javnim računarskim mrežama (uključujući internet), autentifikacija se najčešće realizuje upotrebom korisničkog imena i lozinke. Pretpostavlja se da je poznavanje pristupnih podataka dovoljna garancija da je korisnik ispravan, tj. onaj za koga se prestavlja. Svaki korisnik se na početku upotrebe sistema registruje (ili ga je već neko registrovao, npr. sistem administrator), upotrebom dodijeljene ili lozinke po svom izboru. Međutim, autentifikacija na osnovu lozinke se ne smatra dovoljno sigurnom i pouzdanom za bilo koji sistem koji sadrži osjetljive podatke.

Korisnička imena su često kombinacija imena i prezimena osobe, zbog čega ih je lako pogoditi. Ako se ne postave pravila za kreiranje, ljudi često prave slabe lozinke – čak i jake lozinke mogu biti ukradene, slučajno otkrivene ili zaboravljane. Iz tog razloga, internet poslovanje i mnoge druge transakcije zahtijevaju strože procese i metode autentifikacije.

Nedostaci autentifikacije na osnovu lozinke mogu se nadoknaditi upotrebom kompleksnijih korisničkih imena koja nije lako pogoditi te definisanjem pravila za kreiranje sigurnije lozinke, kao što su minimalna dužina te kombinovanje velikih i malih slova sa brojevima i simbolima.

Uprkos tome, sistemi koji koriste autentifikaciju na osnovu lozinke i generalno autentifikaciju na osnovu „nečega što znamo“ su mnogo ranjiviji nego oni koji koriste nekoliko nezavisnih metoda autentifikacije.

Autentifikacioni faktor (metod) predstavlja način na koji se vrši provjera identifikacije korisnika. Tri najčešće kategorije su opisane kao „nešto što znamo“ (faktor znanja), „nešto što posjedujemo“ (faktor posjedovanja) i „nešto što jesmo“ (faktor nasleđivanja).

## Autentifikacioni faktori

* Faktori znanja – kategorija identifikatora koja sa sastoji od informacija koje korisnik posjeduje, kao što su lični identifikacioni broj (PIN-*Personal Identification Number*), korisničko ime, lozinka ili odgovor na tajno pitanje.
* Faktori posjedovanja – kategorija identifikatora koja se zasniva na predmetima koje korisnik ima uz sebe, najčešće u obliku fizičkog uređaja kao što je sigurnosni token ili mobilni telefon sa instaliranom aplikacijom koja generise token.
* Faktori nasleđivanja – kategorija identifikatora koja se sastoji od elemenata koji su svojstveni pojedincu, u obliku biometrijskih podataka kao što je npr. otisak prsta.

Lokacija korisnika i trenutno vrijeme su nešto što se smatra četvrtim i petim faktorom autentifikacije. Rasprostranjenost pametnih telefona olakšava primjenjivanje autentifikacije korisnika upotrebom više faktora. Većina pametnih telefona opremljena je GPS-om, tj. sistemom za lociranje pomoću satelita, omogućujući time relativno pouzdanu provjeru lokacije sa koje se korisnik prijavljuje. Manje pouzdane mjere su upotreba MAC adrese pristupne tačke ili provjera fizičke prisutnosti upotrebom kartica i ostalih elemenata autentifikacionog faktora posjedovanja.

# Upravljanje identitetima

Upravljanje identitetima je široka administrativna oblast koja se bavi identifikovanjem individua u sistemu i kontrolisanjem njihovog pristupa resursima u okviru tog sistema na osnovu povezivanja korisničkih prava sa utvrđenim identitetom.

Na najnižem nivou, upravljanje identitetima podrazumijeva definisanje onoga šta korisnik može raditi na mreži sa specifičnim uređajima i pod kojim okolnostima. Danas, mnogi sigurnosni proizvodi imaju naglasak na upravljaju mobilnog pristupa korporativnim sistemima. U enterprise okruženju, upravljanje identitetima se koristi zbog poboljšanja sigurnosti i produktivnosti, pri čemu se smanjuju troškovi i suvišan napor.

Upravljanje identitetima je postalo veoma bitno pitanje u IT svijetu. Kako su web aplikacije vremenom rasle kako bi podržale mnoštvo poslovnih procesa, povećao se i broj korisnika, grupa i uloga (*rola*) kojima treba upravljati. Kako se povećava broj sistema koji se uvode, povećava se i napor za sistem administratore i timove za podršku.

Što se tiče upravljanja identitetima (nalozima i korisničkim pristupom), postoje mnogi izazovi za pravilno upravljanje identitetima. Neki od njih uključuju kreiranje naloga, kao i njihovo održavanje i uklanjanje. Drugi izazov je, koji je najbolji način da se obezbijedi autentifikacija za korisnike. To može obuhvatati pravila vezana za lozinke i/ili biometriju. Kako potvrđujemo korisničke podatke na taj način da zaštitimo kompaniju, pri čemu istovremeno umanjujemo napor potreban kako bi se korisnik autentifikovao u okviru više sistema.

Uz povećanje broja sistema koji se koriste, povećava se i napor potreban za administraciju naloga. To je mjesto gdje SSO i sistem za upravljanje identitetima nastupaju zajedno. Odogvarajuća implementacija takvog sistema će značajno umanjiti cijelokupan administrativni posao neophodan pri upravljanju korisničkim informacijama u centralizovanom skladištu. Neki od problema koje ovakav sistem rješava:

* Prevelik broj timova uključen u administraciju korisnika
* Kreiranje naloga od strane zaposlenih koji nemaju odgovarajuća prava pristupa
* Nepostojanje standarda pri kreiranju korisničkih naloga, što dovodi do toga da jedan korisnik ima više različitih naloga sa različitim dozvolama u okviru istog sistema
* Spor odziv pri održavanju naloga, zbog postojanja birokratije u održavanju različitih sistema i koordinisanja različitih pristupnih pravila između više grupa
* Suvišne ili netačne informacije koje djelimično identifikuju direktorije i identitete

Možda najveći problem predstavlja to što neke kompanije adresiraju taj problem, ali ne poduzimaju ništa kako bi ga riješile, ne gledajući dugoročno i ne pokušavajući da se suoče sa stvarnim uzrokom problema.

Aplikacije mogu koristiti SSO kako bi obezbijedile korisnicima jednostavan i brz pristup sadržaju koji se čuva i uređuje na različitim vrstama sistema. To se postiže na taj način što se omogućava višestruka prijava sa istom lozinkom. SSO je metodologija koja obezbjeđuje jednu akciju za autentifikaciju i autorizaciju. Omogućuje korisnicima pristup svim računarima i sistemima na kojima imaju pravo pristupa, bez potrebe da unose lozinku više puta. Nudeći tu mogućnost, SSO značajno umanjuje ljudsku grešku koja je najčešći uzrok pada sistema. Moglo bi se reći da je najznačajnija stvar posjedovanja centralizovanog sistema, postojanje centralnog skladišta gdje se čuvaju sva korisnička imena i lozinke. SSO rješava problem postojanja pojedinačnih baza korisnika, lozinki i dozvola u okviru svake od aplikacija. Povezivanje grupe aplikacija upotrebom integrisanog sistema, omogućava i internim i eksternim korisnicima da koriste isto korisničko ime i lozinku za više različitih aplikacija.

Iz sigurnosnih razloga, alati za upravljanje identitetima trebalo bi da se izvršavaju kao aplikacija na posebnom mrežnom uređaju ili serveru, bilo lokalno ili *cloud*. Jezgro sistema za upravljanje identitetima čine pravila koja definišu koji uređaji i korisnici su dozvoljeni na mreži i šta korisnik može postići, zavisno od tipa njegovog uređaja, lokacije i drugih faktora. Npr. alarm se može aktivirati kada određeni korisnik pokuša pristupiti resursu za koji nema dozvolu. Izvještavanje proizvodi revizijski dnevnik u koji se bilježe sve specifične aktivnosti koje su izvršene.

# Autorizacija i kontrola pristupa

## Autorizacija

Autorizacija je proces dodjeljivanja dozvole nekome da uradi ili posjeduje nešto. U višekorisničkim računarskim sistemima, sistem administrator definiše koji korisnici imaju pristup sistemu i koje privilegije posjeduju (kao što su pristup određenim direktorijima, trajanje i vrijeme pristupa, količina skladišnog prostora itd.). Pretpostavimo da se neko prijavi u računarski operativni sistem ili aplikaciju, sistem ili aplikacija identifikuju kojim resursima korisnik može pristupiti tokom sesije. Prema tome, autorizaciju možemo posmatrati i kao početna podešavanja dozvola od strane sistem administratora i kao provjeru dozvola koje se dodjeljuju korisniku kada dobije pristup.

Autentifikacija provjerava identitet korisnika i omogućava autorizaciju. Autorizacijska pravila diktiraju šta korisnik ima pravo da izvršava. Npr. bilo koja mušterija banke može kreirati svoj nalog kako bi pristupila online sistemu banke, ali autorizaciona pravila banke moraju da se pobrinu da da samo taj korisnik ima pravo pristupa svom ličnom nalogu nakon što se potvrdi njegov identitet. Jednostavno rečeno, autorizacija je proces sprovođenja pravila: određivanje kojim vrstama ili funkcionalnostima aktivnosti, resursa ili usluga korisnik ima pristup. Uobičajeno, autorizacija se dešava u okviru procesa autentifikacije. Nakon što se korisnik autentifikuje, mogu mu se odobriti različiti tipovi pristupa ili aktivnosti.

Autorizacija može biti primijenjena u mnogo složenijem obliku nego što je to npr. pravo pristupa web sajtu ili internoj mreži kompanije. Korisnički nalog može biti uključen u grupu naloga koji dijele zajednička autorizacijska pravila. Npr. recimo da postoji baza podataka koja sadrži transakcije kupaca ali i njihove lične podatke i podatke o kreditnoj kartici. Trgovac bi mogao kreirati autorizacijsko pravilo za ovu bazu podataka, tako da dozvoli grupi marketinga pristup svim kupovinama mušterija ali pri tome spriječavajući pristup ličnim podacima kupaca i podacima njihovih kreditnih kartica, kako bi marketinška grupa mogla identifikovati popularne proizvode u cilju njihovog promovisanja ili rasprodaje.

Korisnici implicitno kreiraju autorizacijska pravila kada koriste socijalne medije: Facebook, LinkedIn ili Twiter autentifikuju stotine miliona korisnika, ali u izvjesnoj mjeri korisnik može sam odrediti pravila da li i u koliko mjeri ostvaruje kontakt sa drugim korisnicima. Isti je slučaj sa dijeljenjem datoteka, video klipova ili fotografija na sajtovima kao što su Google Docs, Dropbox, Instagram, Pinterest ili čak i kada napravite dijeljeni direktorijum na svom laptopu.

## Kontrola pristupa

Dok autorizacijska pravila definišu čemu određeni korisnik ili grupa korisnika može pristupiti, kontrola pristupa, takođe poznata pod nazivima dozvole ili privilegije, predstavlja metode koje sprovode autorizacijska pravila. Neki od primjera:

* Kroz podešavanja Facebook-a – Ko može vidjeti moje stvari ? Ko me može kontaktirati ? Ko me može pronaći ? – Nekim korisnicima ili javnosti dozvoljavamo ili onemogućujemo pristup onome što postavaljamo na svoj Facebook nalog.
* Podešavanja Google Docs-a omogućavaju nam postavljanje dozvola za izmjenu ili dijeljenje dokumenata koje zajednički koristimo.
* Flickr podešavanja nam dozvoljavaju da kreiramo ili podijelimo album ili sliku sa porodicom, prijateljima ili sa javnošću.

Ispravno podešavanje prava pristupa je ključna komponenta zaštite informacija od neodobrenog pristupa i zaštite računarskih sistema od zloupotrebe, ali posao podešavanja tih prava pristupa nije jednostavan.

# SAML (Security Assertion Markup Language)

SAML je otvoreni, standardizovani format poruka, zasnovan na XML-u koji služi za razmjenu autentifikacionih i autorizacionih podataka između strana koje komuniciraju, najčešće između provajdera identiteta (IdP) i servisnog provajdera (SP). SAML je razvijen od strane OASIS-a i datira još iz 2001 godine, s tim da se konstantno vrše nadogradnje i unapređenja.

Jedan od najvažnijih zahtjeva koje SAML adresira je *single sign-on (SSO)* putem internet preglednika.

## Principi

SAML specifikacija definiše tri uloge: *principal* (korisnik), *Identity Provide*r – IdP (provajder identiteta) i *Service Provider* – SP (servis provajder). U scenarijima koje SAML adresira, korisnik zahtjeva uslugu od servis provajdera. Servis provajder zahtjeva i dobavlja tvrdnje o identitetu korisnika od strane provajdera identiteta. Na osnovu te tvrdnje, servis provajder vrši odluku o pravu pristupa korisnika određenim resursima i uslugama.

Provajder identiteta, prije nego što dostavi tvrdnju o identitetu korisnika servisnom provajderu, može tražiti neke dodatne informacije od korisnika, kao što su korisničko ime i lozinka, kako bi utvrdio njegov identitet. SAML specifikuje tvrdnje izmedju tri strane, naročito poruke kojim se tvrdi identitet korisnika od strane provajdera identiteta prema servisnom provajderu. Po SAML specifikaciji, jedan provajder identiteta može slati tvrdnje različitim servisnim provajderima. Slično, jedan servis provajder se može oslanjati na i vjerovati tvrdnjama dobijenim od strane više nezavisnih provajdera identiteta.

SAML ne definiše način na koji će se korisnik autentifikovati na strani provajdera identiteta. Mgu se koristiti korisničko ime i lozinka ili neki drugi oblik autentifikacije, uključujući autentifikaciju u više koraka (*multi factor*).

## Kako SAML funkcioniše

Na koji način SAML rješava sve ? U suštini, SAML nije ništa više nego niz XML baziranih poruka koje govore da li je korisnik autentifikovan, kakva prava, uloge i pristupe posjeduje te na koji način može koristiti podatke i resurse na osnovu tih prava i uloga. Kompatibilan je sa HTTP, SMTP, FTP i SOAP protokolima, uz niz ostalih protokola i tehnologija.

Glavne komponente SAML specifikacije su:

* *Assertions* (tvrdnje)
* *Protocols* (protokoli)
* *Bindings* (povezivanja)
* *Profiles* (profili)

### *Assertions* (tvrdnje)

SAML tvrdnja predstavlja paket sigurnosnih informacija dobijenih od strane provajdera identiteta.

<saml:Assertion ...>

<saml:Issuer>...</saml:Issuer>

<saml:Subject>...</saml:Subject>

<saml:Conditions>...</saml:Conditions>

<saml:AuthnStatement>...</saml:AuthnStatement>

<saml:AttributeStatement>...</saml:AttributeStatement>

<saml:AuthorizationDecisionStatement>...

</saml:AuthorizationDecisionStatement>

</saml:Assertion>

SAML tvrdnje je obično prenose od strane provajdera identiteta prema sersisnom provajderu. Tvrdnje sadrže izjave (*statements*) koje servis provajder koristi kako bi donio odluku o pravima pristupa korisnika. Postoje 3 vrste izjava koje SAML prenosi:

1. ***Authentication statements* (autentifikacione izjave)** – govore servis provajderu da se korisnik autentifikovao na strani provajdera identiteta u određeno vrijeme i upotrebom određene metode za autentifikaciju
2. ***Attribute statements* (atributske izjave) –** govore koji određeni atributi su vezani za subjekta odnosno korisnika. Atribut predstavlja jednostavan par podataka, „naziv-vrijednost“. Oslanjajuće strane koriste atribute kako bi donijele odluke o dozvolama pristupa.
3. ***Authorization decision statements* (autorizacijske odluke) –** govore da li korisnik ima dozvolu da izvrši određenu akciju nad resursom. Izražavanje autorizacionih odluka je nešto gdje je SAML sa namjerom ograničen. Za naprednije slučajeve upotrebe preporučuje se korištenje XACML-a.

### *Protocols* (protokoli)

SAML protokol opisuje način na koji se određeni SAML elementi (uključujući tvrdnje) pakuju unutar SAML zahtjeva i odgovora, i definiše pravila kojih se SAML učesnici moraju pridržavati prilikom kreiranja ili konzumiranja istih. Najvećim dijelom, SAML protokol je jednostavan protokol koji se zasniva na slanju zahtjeva i primanju odgovora.

Najbitniji tip zahtjeva SAML protokola se naziva *query* (upit). Servis provajder šalje upit direktno provajderu identiteta putem sigurnog kanala. U skladu sa tri tipa izjava, postoje tri tipa SAML upita:

1. Autentifikacijski upit
2. Atributski upit
3. Upit autorizacijske odluke

Među navedenim, atributski upit je možda najvažniji. Rezultat ovog upita je SAML odgovor koji sadrži tvrdnju koja se sastoji od atributskih izjava.

### Bindings (povezivanja)

SAML povezivanje je mapiranje SAML poruka u standardne formate poruka i komunikacione protokole. Npr, SAML SOAP povezivanje specifikuje na koji način se SAML poruka enkapsulira u SOAP poruku, koja je sama po sebi vezana za HTTP poruku.

SAML 1.1 definiše samo jedno povezivanje – SAML SOAP. SAML 2.0 sadrži potpuno novu specifikaciju povezivanja koja definiše sledeće standarde:

* *SAML SOAP Binding (based on SOAP 1.1)*
* *Reverse SOAP (PAOS) Binding*
* *HTTP Redirect (GET) Binding*
* *HTTP POST Binding*
* *HTTP Artifact Binding*
* *SAML URI Binding*

Za *Web Browser SSO* profil najčešće se koriste *HTTP Redirect Binding* i *HTTP POST Binding*. Npr, servis provajder može koristiti HTTP redirekciju kako bi poslao zahtjev, dok provajder identiteta koristi HTTP POST za vraćanje odgovora.

### Profiles (profili)

SAML profil detaljno opisuje kako se kombinuju tvrdnje, protokoli i povezivanja za određeni slučaj upotrebe. Najvažniji SAML profil je *Web Browser SSO* profil.

Ovaj profil je detaljno opisan u sekciji 9.1

## SAML 2.0

# XACML (Extensible Access Control Markup Language)

XACML (Extensible Access Control Markup Language) je otvoreni standarizovani jezik zasnovan na XML-u, dizajniran za iskazivanje sigurnosnih propisa i prava pristupa podacima u okviru web servisa i poslovnih sigurnosnih aplikacija. Potvrđen od strane Organizacije za unapređenje uređenih informacionih standarda (OASIS - Organization for the Advancement of Structured Information Standards) u Februaru 2003 godine, XACML je razvijen s ciljem standarizovanja kontrole pristupa upotrebom XML-a, tako da npr. korisnik može pristupiti nekoliko uvezanih web sajtova samo jednom prijavom. XACML se nekad oslovljava i kao XACL (Extensible Access Control Language).

XACML je dizajniran tako da funkcioniše u kombinaciji sa SAML-om (Security Assertion Markup Language), drugim OASIS standardom. SAML definiše načine dijeljenja autorizacionih podataka, kao što su korisničke lozinke i bezbijednosne provjere, između sigurnosnih sistema. Komponenta za pravila (program koji ispituje ustanovljena pravila i predlaže ponašanje koje ide uz njih) zajedno sa propisima iskazanim u XCAML obliku, može porediti te informacije sa uspostavljenim kriterijumima kako bi se utvrdila prava korisnika. XACML specifikacija je razvijena zajedničkim naporima članica OASIS-a među kojima su IBM, Sun Microsystems i Entrust.

Uobičajena postavka je kada neko želi izvršiti neku akciju nad resursom. On će poslati zahtjev onome ko štiti taj resurs (kao što je fajl sistem ili web server), koga nazivamo PEP (Policy Enforcement Point). PEP će napraviti zahtjev na osnovu atribute naručioca, resursa koji se potražuju akcije koju želi izvršiti i ostalih informacija koje se odnose na zahtjev. PEP će zatim poslati to PDP-u (Policy Decision Point), koji će pogledati zahtjev i neke propise koji se odnose na njega, a zatim donijeti odluku da li pristup trebao biti odobren. Odgovor tj. odluka se vraća PEP-u, koji naručiocu može dozvoliti ili spriječiti pristup. PEP i PDP mogu biti dio jedne aplikacije ili raspoređeni na nekoliko servera.

Prednosti XACML-a:

* Standarizovan je. Upotrebom standarizovanog jezika, koristimo nešto što je provjereno od velikog broja stručnjaka i korisnika, te ne moramo svaki put praviti svoj sistem, razmišljati i nezgodnim problemima koji se pojavljuju pri stvaranju novog jezika. Uz to, kako XACML postaje sve rasprostranjeniji, sve je lakše upregnuti ga sa ostalim aplikacijama koje koriste isti standarizovani jezik.
* Generički je. To znači da umjesto pokušavanja obezbijeđivanja kotrole pristupa za neko određeno okruženje i posebnu vrstu resursa, može biti korišten u bilo kom okruženju. Jedan propis može biti zapisan a zatim se može koristiti u mnogim različitim vrstama aplikacija, i kada se koristi jedan zajednički jezik, upravljanje propisima postaje mnogo jednostavnije.
* Distribuisan je. To znači da propis može biti napisan tako da upućuje na druge propise koji se čuvaju na proizvoljnim lokacijama. Rezultat je da umjesto upravljanjem jednog monolitnog propisa, različite osobe ili grupe mogu po potrebi upravljati pod-elementima propisa, a XACML zna na koji način da kombinuje rezultate tih različitih propisa u jednu odluku.
* Moćan je. Dok postoje razni načini za proširenje osnovnog jezika, u većini okruženja neće biti potrebe za to. Standardni jezik već podržava širok spektar tipova podataka, funkcija i pravila o kombinovanju rezultata različitih propisa.

## Konstrukcije najvišeg nivoa: propisi (Policy) i skupovi propisa (PolicySet)

U korijenu svih XACML propisa je **propis** ili **skup propisa**. Skup propisa je spremište koje može sadržati druge propise ili skupove propisa, kao i reference (uputstva) prema propisima pronađenim u udaljenim lokacijama. **Propis** predstavlja jedan propis za kontrolu pristupa, izražen kroz skup **pravila** (**Rules**). Svaki XACML dokument propisa, sadrži tačno jedan korijenski XML tag propisa ili skupa propisa.

S obzirom na to da **propis** ili **skup propisa** može sadržati nekoliko propisa ili pravila, svaki od njih može dovesti to različitih odluka prava pristupa. XACML treba na neki način da uskladi svaku od odluka koje ti propisi donose. Ovo se vrši kroz kolekciju algoritama za kombinovanje (Combining Algorithms). Svaki algoritam predstavlja drugačiji način kombinovanja nekoliko odluka u jednu. Postoje algoritmi za kombinovanje propisa (Policy Combining Algorithm – koriste se od strane skupa propisa) i algoritmi za kombinovanje pravila (Rule Combining Algorithms – koriste se od strane propisa). Primjer jednog algoritma je algoritam – odbijanje nadglašava, koji kaže da bez obzira na sve, ako bilo koja procjena vrati status – odbijen (Deny), konačan rezultat je odbijanje. Ovi algoritmi za kombinovanje se koriste za izgradnju sve složenijih propisa, i dok postoji sedam standardnih algoritama kombinovanja, moguće je stvoriti i dodatne koji će odgovarati posebnim potrebama.

## Ciljevi (Targets) i pravila (Rules)

Dio zadatka XACML PDP-a je pronalaženje propisa koji se odnosi na upućeni zahtjev. Kako bi to izvršio, XACML obezbjeđuje još jednu karakteristiku pod nazivom **cilj** (**Target**). Cilj je u suštini skup pojednostavljenih uslova za subjekta, resurs i akciju koji moraju biti zadovoljeni kako bi se propis, skup propisa ili pravilo primijenili na dati zahtjev. Koriste funkcije istinitosti za poređenje vrijednosti pronađenih u zahtjevu sa onim koje se nalaze u cilju. Ako se zadovolje svi uslovi cilja, onda se propis, skup propisa ili pravilo koje je vezano za njega primjenjuje na zahtjev. Dodatno, da bi postajao način za provjeru primjenjivosti, cilj sadrži i informacije kojima se mogu označiti (indeksirati) propisi, što je korisno ukoliko će se čuvati veliki broj propisa a zatim vršiti brza pretraga kako bi se pronašli oni koji se odgovaraju. Npr. propis može sadržati cilj koji se odnosi samo na zahtjeve na određeni servis. Kada stigne zahtjev za pristup tom servisu, PDP će znati gdje da traži propise koji bi mogli odgovarati tom zahtjevu, zahvaljujući tome što su propisi označeni na osnovu ograničenja njihovih ciljeva. Cilj može biti označen i tako da je primjenjiv na bilo koji zahtjev.

Nakon što se propis pronađe i potvrdi da je odgovara zahtjevu, procjenjuju se njegova pravila. Propis može imati bilo koji broj pravila koja sadrže osnovnu logiku XACML propisa. Jezgro većine pravila je **uslov** (**Condition**), koji je funkcija tačnosti. Ako se uslov procijeni kao tačan, izvršiće se posljedica pravila. Procjena uslova može kao rezultat vratiti i grešku (Indeterminate) ili dovesti do zaključka da uslov nije primjenjiv na zahtjev (NotApplicable). Uslov može biti prilično složen, izgrađen od proizvoljnog broja ugnježdenih funkcija i atributa.

## Atributi (Attributes), vrijednosti atributa (Attribute Values) i funkcije (Functions)

Valuta kojom se XACML bavi su **atributi**. Atributi su imenovane vrijednosti poznatih tipova podatak koje mogu sadržati identifikator izdavaoca ili datum i vrijeme izdavanja. Atributi su karakteristike subjekta, resursa, akcije ili okruženja u kome je stvore zahtjev. Ime korisnika, njegova sigurnosna dozvola, datoteka kojoj želi pristupiti i trenutno vrijeme su atributske vrijednosti. Kada se zahtjev šalje od PEP-a prema PDP-u, on se sastoji gotovo isključivo od atributa koji će se kasnije porediti sa vrijednostima atributa u okviru propisa, kako bi se donijela odluka o dozvoli pristupa.

Propis razlučuje vrijednosti atributa iz zahtjeva ili iz nekog drugog izvora upotrebom dva mehanizma: **razvodnik atributa** (**AttributeDesignator**) i **odabirač atributa** (**AttributeSelector**). Razvodnik atributa dozvoljava propisu da naznači atribut sa određenim imenom i tipom, i eventualno i izdavačem, a zatim će PDP potražiti tu vrijednost u zahtjevu ili negdje drugo ukoliko je ne pronađe unutar zahtjeva. Postoje četiri vrste razvodnika, po jedan za svaku vrstu atributa u zahtjevu: **subjekat**, **resurs**, **akcija** i **okruženje**. Kako atribut subjekta može biti razdvojen u nekoliko različitih kategorija, **razvodnik atributa subjekta** (**SubjectAttributeDesignator**) može takođe navesti kategoriju u kojoj vršiti pretragu. Odabirač atributa dozvoljava propisu da pretražuje vrijednosti atributa kroz XPath upit. Tip podatka i XPath izraz se mogu koristiti za razriješavanje jednog skupa vrijednosti, bilo to u okviru zahtjeva ili negdje drugo.

I razvodnik i odabirač atributa moguu vratiti nekoliko vrijednosti (s obizirom na to da može postojati više poklapanja u zahtjevu), zbog čega XACML obezbjeđuje poseban atribut nazvan torba (Bag). Torba je neuređeni skup koji dozvoljava pojavu duplikata, te predstavlja ono što razvodnici i odabirači uvijek vraćaju, lak i se samo jedna vrijednost poklapa. U slučaju da nema poklapanja, vraća se prazna torba, iako razvodnik ili odabirač može postaviti oznaku koja u tom slučaju dovodi do greške umjesto vraćanja prazne torbe.

Nakon što se dopremi torba atributskih vrijednosti, treba ih analizirati na neki način kako bi se donije odluke o dozvoli za pristup. To se vrši kroz moćni sistem funkcija. Funkcije mogu raditi sa bilo kojom kombinacijom atributa, te mogu vratiti bilo koji tip vrijednosti atributa koji sistem podržava. Fnkcije takođe mogu biti ugnježdene, tako da je moguće imati funkcije koje operišu sa rezultatom druge funkcije, i ta hijerarhija može biti proizvoljne složenosti. Moguće je definisati prilagođene funkcije kako bi se obezbijedio još bogatiji jezik za izražavanje uslova pristupa.

Jedna napomena koju treba uzeti u obzir prilikom izgradnje ovih hijerarhija funkcija je to da većina funkcija radi sa određenim tipovima (kao što su nizovi karaktera ili cijeli brojevi) dok razvodnici i odabirači uvijek vraćaju torbe (Bags) vrijednosti. Kako bi to razriješio, XACML definiše skup standardnih funkcija koje prijamu torbu vrijednosti određenog tipa a zatim vraćaju jednu vrijednost tog tipa ukoliko se nalazi samo jedna vrijednost u torbi, ili grešku ukoliko se u torbi nalazi više ili ni jedna vrijednosti. To je jedna od funkcija koja se najčešće može vidjeti u okviru uslova. Takve funkcije nisu potrebne u ciljevima, pošto PDP automatski primjenjuje odgovarajuće funkcije svakom elementu torbe.

## Primjer jednog propisa

Navešćemo jedan jednostavan primjer **propisa** koristeći gore navedene osobine. Njegov **cilj** govori da se **propis** primjenjuje samo na zahtjeve prema serveru pod nazivom „UzorakServera“. **Propis** ima **pravilo** u okviru **cilja** koje zahtjeva akciju prijave i **uslov** koji se primjenjuje samo ako **subjekat** pokušava da se prijavi između 9:00 i 17:00. Ovaj primjer se može proširiti drugim **pravilima** za različite akcije. Ako prvo pravilo nije primjenjivo, biće primijenjeno podrazumijevano pravilo koje uvijek vraća odbijanje (Deny) (pravila se izvršavaju po redosledu).

<Policy PolicyId="SamplePolicy"

RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">

<!-- This Policy only applies to requests on the SampleServer -->

<Target>

<Subjects>

<AnySubject/>

</Subjects>

<Resources>

<ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">SampleServer</AttributeValue>

<ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"

AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>

</ResourceMatch>

</Resources>

<Actions>

<AnyAction/>

</Actions>

</Target>

<!-- Rule to see if we should allow the Subject to login -->

<Rule RuleId="LoginRule" Effect="Permit">

<!-- Only use this Rule if the action is login -->

<Target>

<Subjects>

<AnySubject/>

</Subjects>

<Resources>

<AnyResource/>

</Resources>

<Actions>

<ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">login</AttributeValue>

<ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"

AttributeId="ServerAction"/>

</ActionMatch>

</Actions>

</Target>

<!-- Only allow logins from 9am to 5pm -->

<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal"

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">

<EnvironmentAttributeSelector DataType="http://www.w3.org/2001/XMLSchema#time"

AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>

</Apply>

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue>

</Apply>

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal"

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">

<EnvironmentAttributeSelector DataType="http://www.w3.org/2001/XMLSchema#time"

AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>

</Apply>

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00</AttributeValue>

</Apply>

</Condition>

</Rule>

<!-- We could include other Rules for different actions here -->

<!-- A final, "fall-through" Rule that always Denies -->

<Rule RuleId="FinalRule" Effect="Deny"/>

</Policy>

# Izbor strategije za realizaciju SSO servera

Vjerovatno ste se već prijavljivali u aplikaciju (mobilnu ili web) klikom na dugme „Prijavi me upotrebom Facebook-a“. Ako koristite Instagram, Spotify ili Pinterest znate o čemu se radi.

Kao korisnik, ne interesuje vas na koji način funkcioniše SSO. Sve što želite je koristiti aplikaciju i imati uglađeno iskustvo, uz što manje pamćenje korisničkih imena i lozinki.

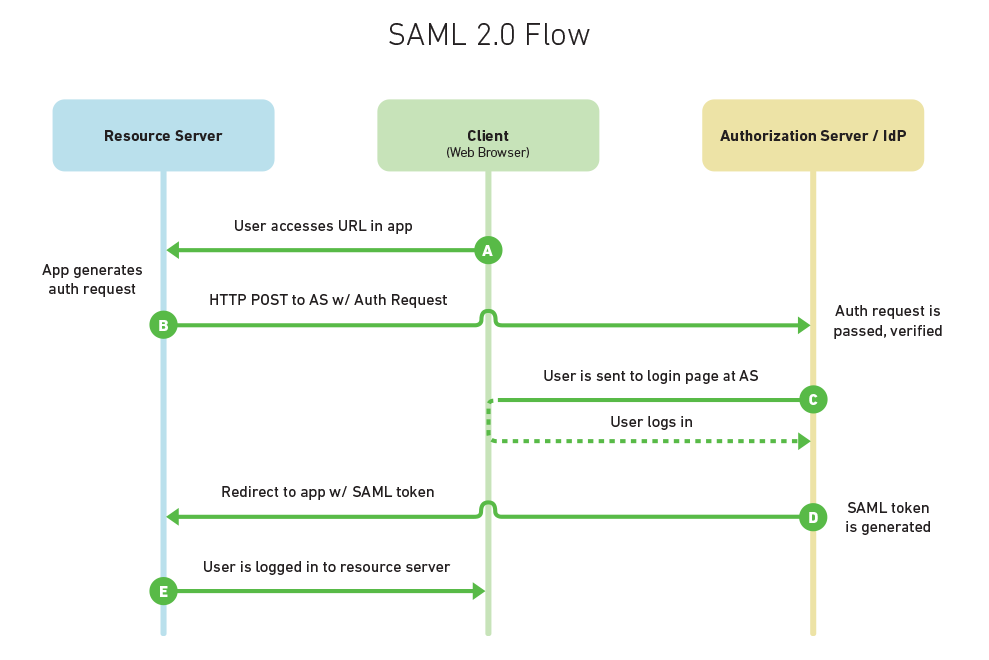
Kako bi se korisniku omogućilo takvo iskustvo, potrebno je implementirati SSO rješenje. Postoji mnogo strategija za implementaciju SSO-a, ali ćemo analizirati i porediti dvije najpopularnije – SAML i OAuth2.

## SAML 2.0

Prvo ćemo baciti pogled na SAML 2.0 koji je skup otvorenih standarda, među kojima je jedan posebno namijenjen i dizajniran za SSO.

SAML 2.0 specifikacija (nadalje SAML) obezbjeđuje „Web Browser SSO Profile“ koji opisuje kako se single-sign-on može postići za web aplikacije. Postoje tri ključna igrača u SAML-u:

* Davalac usluga (Service Provider - SP) – web server kome pokušavamo pristupiti
* Klijent (Client) – način na koji korisnik međudjeluje sa davaocem usluga, kao što je npr. web aplikacija koja je učitana u okviru web preglednika korisnika
* Dobavljač identiteta (Identity Provider - IdP) – server koji posjeduje identitet i pristupne podatke korisnika, tj. potvrđuje identitet korisnika.



Slika - Najčešći scenario upotrebe SAML-a

Primjer jednog slučaja upotrebe koji opisuje gornji dijagram:

* A – korisnik otvara web preglednik i posjećuje MyPhotos.com na kome se nalaze sve njegove slike. MyPhotos.com ne može sam vršiti autentifikaciju korisnika.
* B – kako bi autentifikovao korisnika, MyPhotos.com pravi SAML autentifikacijski zahtjev (SAML Authrequest), potpisuje ga, eventualno ga šifruje, a zatim kodira. (šifrovati znači sakriti sadržaj, dok kodirati znači obezbijediti njegovu upotrebo s ciljem izbjegavanja gubitka informacija ili grešaka u prenosu) Nakon toga, preusmjerava web preglednik korisnika na dobavljač identiteta (IdP) s ciljem autentifikacije. IdP prima zahtjev, dekodira ga, koliko je potrebno dešifruje, te provjerava potpis.
* C – Ukoliko je autentifikacijski zahtjev ispravan, IdP će korisniku prikazati stranicu za prijavu na kojoj može unijeti svoje korisničko ime i lozinku.
* D – Nakon što se korisnik uspješno prijavio, IdP generiše SAML token koji sadrži informacije o identitetu korisnika (kao što su korisničko ime, elektronska adresa itd.) IdP uzima taj token i preusmijerava korisnika nazad na davaoca usluga – SP (MyPhotos.com).
* E – MyPhotos.com provjerava SAML token, ukoliko je potrebno dešifruje ga i izdvaja informacije o identitetu korisnika – ko je taj korisnik i koje dozvole posjeduje. MyPhotos.com zatim korisnika prijavljuje u sistem, vjerovatno upotrebom kolačića ili kreiranjem nove sesije.

Na kraju procesa korisnik može koristiti MyPhotos.com kao prijavljeni korisnik. Pristupni podaci korisnika se nikada ne preslijeđuju kroz MyPhotos.com nego samo kroz IdP.

Postoji još detalja vezano za dijagram, ali ovo je opis onog što se dešava na visokom nivou.

### SAML token (Token) vs. SAML tvrdnja (Assertion)

Kada je SAML prvi put demonstriran, iznova i iznova se pojavljivao pojam *SAML token*. To u stvari i nije pojam definisan u okviru SAML specifikacije, ali je ostao u upotrebi iako njegovo značenje nije potpuno jasno.

Ispostavilo se da pojam *SAML token* predstavlja ustaljeni za upućivanje na SAML tvrdnju koja je često sabijen, kodirana, moguće i šifrovana i najčešće izgleda kao nešto zapetljano klupko. SAML tvrdnja je samo XML čvor sa određenim elementima.

### SAML ograničenja

SAML podržava koncept uvezivanja. To su u suštini sredstva kojim IdP preusmjerava korisnika nazad do SP. Npr. u koraku **D**, korisnik se preusmjerava nazad na MyPhotos.com, ali kako ?

Dva bitna tipa uvezivanja su HTTP preusmijeravanje (HTTP Redirect) i HTTP POST uvezivanja, navedena u SAML 2.0 specifikaciji. HTTP Redirect koristi HTTP preusmjeravanje kako bi poslao korisnika nazad na SP, u slučaju našeg primjera na MyPhotos.com

HTTP Redirekt je odličan za kratke SAML poruke, ali se ne preporučuje njegova upotreba ukoliko su poruke duge, kao što je SAML tvrdnja.

Preporučeni način uvezivanja tj. HTTP POST takođe ime svoje nedostatke. Npr. SAML specifikacija preporučuje da se u koraku D prikaže HTML forma sa akcijom koja upućuje nazad na SP.

Možete dodati dugmić na koji korisnik treba kliknuti kako bi poslao podatke sa forme ili koristiti JavaScript funkciju za automatizovano slanje. Zbog čega mora postojati forma koja se šalje? Razlog je što je u vremenu kada je izašla SAML 2.0 specifikacija bilo moguće to postići jedino upotrebom forme, tako da je bilo potrebno koristiti HTTP POST za slanje SAML tokena nazad do SP-a. U to vrijeme, to je bilo neophodno rješenje.

Međutim, to predstavlja problem u slučaju kada klijent nije web-bazirana aplikacije nego nativna aplikacija, kao što su mobilne aplikacije. Npr. recimo da smo instalirali MyPhotos aplikaciju za iPhone. Kada otvorimo aplikaciju, potrebna je autentifikacije posredstvom dobavljača identiteta. Nakon što se autentifikujemo, dobavljač identiteta treba poslati nazad SAML token do MyPhotos aplikacije.

Većina mobilnih aplikacija mogu biti pokrenute upotrebom prilagođenog URLa, kao što je npr. „my-photos://authenticate“, i najvjerovatnoje, dobavljač identiteta šalje formu koja sadrži SAML token na taj URL. MyPhotos aplikacija će se pokrenuti, ali nećemo biti ulogovani. Šta je razlog tome?

Mobilne aplikacije nemaju pristup HTTP POST tijelu. One imaju pristup samo URL-u koji se koristi za pokretanje aplikacije. To znači da ne možemo pročitati SAML token.

### Zaobilazno rješenje SAML HTTP POST uvezivanja

Ograničenje HTTP POST uvezivanja za mobilne aplikacije može biti riješeno na drugi način. Npr. moguće je koristiti ugrađene web prikaze, u okviru kojih se može napisati prilagođeni kod koji će nadgledati cijeli proces autentifikacije. Na završetku cjelokupnog potrebno je odstraniti nepotreban HTML kod i izvući SAML token.

Drugo zaobilazno rješenje je implementacija proxy servera koji može primiti HTTP POST, izvući SAML token, a zatim napraviti URL koji sadrži SAML token (npr. „myphotos://authenticate/?SAMLRequest=gfhvcmijtrra“). Proxy server bi zatim mogao upotrijebiti HTTP redirekciju kako bi doveo do toga da uređaj pokrene aplikaciju MyPhotos. Kako je sada SAML token dio URL-a, aplikacija MyPhotos ga može izdvojiti u upotrijebiti za prijavu.

Treće rješenje bi bilo zanemarivanje preporuke protiv upotrebe HTTP Redirect uvezivanja. Međutim ni malo se ne preporučuje jer je veoma lako napraviti pogrešan korak.

Još jedan pristup koji ne koristi zaobilazno rješenje je da se ne koristi SAML nego druga strategija kao što je OAuth 2.0.

## OAuth 2.0

Za razliku od SAML-a, OAuth 2.0 (nadalje OAuth2) je specifikacija koja je gotovo u potpunosti odbačena krajem 2012. godine. Ima olakšicu pošto je novija specifikacija i sve više se uzima u obzir kako se svijet tehnike mijenjao zadnjih osam godina.

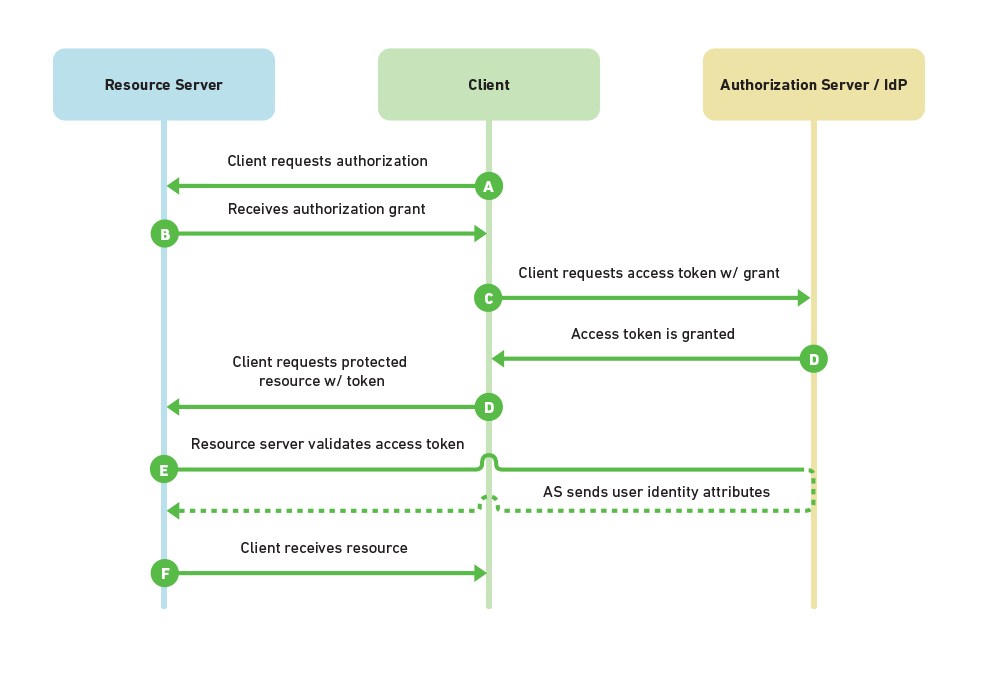
Mobilni uređaji i nativne aplikacije danas napreduju na način koji SAML nije mogao očekivati 2005. godine.

Ključni igrači OAuth2 su:

SAML i OAuth2 koriste slične pojmove za slične koncepte. Zbog poređenja, pored zvaničnog OAuth2 izraza u zagradama navodimo i ekvivalentan SAML izraz.

* Server resursa – Resource Server (Service Provider) – web server kome pokušavamo pristupiti.
* Klijent – Client – način na koji korisnik međudjeluje sa serverom resursa. To može biti web aplikacija u okviru pregledika, nativnam obilna aplikacija, desktop aplikacija serverska aplikacija itd.
* Autorizacijski server - Authorization Server (Identity Provider) - server koji posjeduje identitet i pristupne podatke korisnika, tj. potvrđuje identitet prava pristupa korisnika.

Na visokom nivou, OAuth2 scenario upotrebe se ne razlikuje mnogo od SAML-a:



Slika - OAuth2 scenario upotrebe

Primjer slučaja upotrebe korišten i kod ranije opisanog SAML-a:

* A – korisnik otvara web preglednik i posjećuje MyPhotos.com na kome se nalaze sve njegove slike. MyPhotos.com ne može sam vršiti autentifikaciju korisnika tako da se on preusmjerava na autorizacijski server sa zahtjevom za autorizaciju. Korisniku se prikazuje forma za prijavu i traži se potvrda da želi odobriti serveru sa resursima (MyPhotos.com) da djeluje u njegovo ime. Korisnik se prijavljuje i zatim preusmjerava nazad na MyPhotos.com.
* B – klijent prima kod za odobrenje autorizacije kao dio preusmjeravanja
* C – klijent zatim koristi taj autorizacijski kod kako bi zahtijevao pristupni token od autorizacijskog servera
* D – ukoliko je kod za odobrenje autorizacije ispravan, autorizacijski server odobrava pristupni token. Pristupni token se zatim koristi od strane klijenta pri zahtijevanju resursa sa servera sa resursima (MyPhotos.com).
* E – MyPhotos.com prima zahtjev za resursom i pristupni token. Kako bi se uvjerio da je token ispravan, šalje ga direktno autorizacijskom serveru zbog provjere. Ukoliko je token ispravan, autorizacijski server šalje nazad informacije o korisniku.
* F – nakon provjere korisničkog zahtjeva MyPhotos.com šalje nazad korisniku zahtijevani resurs

Ovo je najčešći OAuth2 tok: tok autorizacijskog koda. OAuth2 nudi tri dodatna toka (kako ih oni zovu odobrenja autorizacije) koji su namijenjeni malo drugačijim scenarijima, kao što su JavaScript aplikacije u okviru jedne stranice, nativne mobilne aplikacije, nativne desktop aplikacije, standardne web aplikacije, serverske aplikacije.

Veliku prednost OAuth2 tokova je to što se komunikacija od autorizacijskog servera nazad do klijenta i servera sa resursima vrši putem HTTP preusmjeravanja (HTTP Redirect) zajedno sa informacijama o tokenu dostupnim u obliku parametara upita. OAuth2 takođe ne pretpostavlja da je klijent u stvari web preglednik, dok SAML Web Browser SSO Profile to radi.

Nativne mobilne aplikacije će odmah raditi, bez potrebe za zaobilaznim rješenjima.

### OAuth2 nedostaci

OAuth2 specifikacija u mnogim slučajevima ne propisuje kako radi komunikacija između resursnog servera i autorizacijskog servra, kao npr. u slučaju provjere tokena. Takođe ne govori ništa o tome koje informacije o korisniku bi trebale biti vraćene ni u kom obliku.

Postoji dosta mjesta gdje OAuth2 specifikacija navodi da su stvari „izvan okvira ove specifikacije (outside the scope of this specification)“. To je dovelo do kritika na OAuth2 specifikaciju jer su mnoge stvari prepuštene implementaciji, što bi u jednom trenutku moglo dovesti do nekompatibilne implementacije.

OAuth2 je još uvijek veoma mlad, a već je široko prihvaćen od strane Google-a, Facebook-a, Salesforce-a, Twitter-a i mnogih drugih. Istinska ljepota OAuth2 se ogleda u njegovoj jednostavnosti. U stvari, OpenID Connect Basic Profile, izgrađen na OAuth2, popunjava neke oblasti koje sama OAuth2 specifikacije ne definiše.

## Rezime – SAML vs. OAuth2

SAML posjeduje svojstvo koje nedostaje OAuth2: SAML token sadrži informacije o identitetu korisnika. Upotrebom OAuth2 te informacije nisu odmah direktno dostupne, te zbog toga resursni server mora napraviti dodatni zahtjev za provjeru tokena sa autorizacijskim serverom.

Sa druge strane, upotrebom OAuth2 moguće je poništiti pristupni token na autorizacijskom serveru, i onemogućiti njegovu dalju upotrebu na resursnom serveru.

Oba pristupa imaju korisne mogućnosti i oba će dobro poslužiti za implementaciju SSO-a.

OAuth2 nudi jednostavnije i više standarizovano rješenje kojim se izbjegavaju zaobilazna rješenja za podršku nativnih aplikacija.

# Implementacija SSO servera

Zadatak praktičnog dijela ovo rada je implementacija SSO servera sa osnovnim funkcionalnostima, koji omogućava autentifikaciju i autorizaciju korisnika.

Implementacija praktičnog dijela se sastoji od SSO servera tj. IdP-a te dva klijenta tj SP-a koji sluze za demonstraciju funkcionisanja samog servera. Razmjena autentifikacionih i autorizacionih poruka se vrši upotrebom SAML 2.0 protokola, i to na način koji definiše SAML 2.0 Web Browser SSO profil.

## Web Browser SSO profil

Ovaj profil podrazumijeva postojanje provajrdera identiteta (IdP), provajdera servisa (SP) i korisničkog agenta. Zavisno od načina povezivanja kojih SP posjeduje 4 a IdP 3, postoji ukupno 12 različitih scenarija. Za ovu implementaciju je korišten najčešći scenario u kome SP vrši slanje SAML zahtjeva koristeći redirekciju dok IdP vrši slanje odgovora putem HTTP POST metode.

Servis provajder - SP

(SSO klijent)

Korisnički agent

(web preglednik)

Provajder identiteta - IdP

(SSO servis)

**8**

**7**

**6**

**5**

**4**

**3**

**Preusmjerava na željeni resurs**

**Preusmjerava na željeni resurs**

**Zahtjeva željeni resurs**

**Zahtjeva servis za konzumiranje odgovora**

**Odgovara XHTML formom**

**(Identifikuje korisnika)**

**Zahtjeva SSO servis**

**1**

**Zahtjeva željeni resurs**

**(Pronalaženje IdP-a)**

**Preusmjerava na SSO servis**

**2**

U narednim tačkama su objasnjeni gore navedeni koraci. Korištene se adrese i primjeri poruka relaizovane implementacije.

1. **Zahtjevanje željenog resursa**

Korisnik putem svog web preglednika zahtjeva željeni resurs na strani servis provajdera (koji predstavlja web aplikaciju)

https://localhost:8081/sp1/protectedResource

Servis provajder vrši provjeru prava pristupa željenom resursu. Ako već postoji važeći sigurnosni kontekst, preskaču se koraci od 2 do 7.

1. **Preusmjeravanje na IdP SSO servis**

Servis provajder generiše odgovarajući SAML zahtjev (i RelayState parametar), a zatim preusmjerava preglednik korisnika na SSO servis koristeci standardnu HTTP redirekciju.

302 Redirect

Location:https://localhost:8080/sso/Redirect?SAMLRequest=<request>&RelayState=<relystate>

SAMLRequest može biti potpisan, ali za onovni primjer nije. Parametar RelayState predstavlja neprozirnu referencu ka stanju informacija koje se čuvaju na strani servis provajdera. U ovoj implementaciji sluzi za čuvanje reference ka informaciji na strani servisnog provajdera koja predstavlja traženi resurs, kako bi se nakon autentifikacije na SSO strani korisnik automatski preusmjerio na taj resurs.

SAMLRequest parametar je kompresovan, base64 enkodiran i URL enkodirana vrijednost <samlp:AuthnRequest> elementa:



1. **Zahtjevanje SSO servisa na IdP strani**

U skladu sa HTTP standardom, web preglednik šalje GET zahtjev SSO servisu na IdP strani:

GET /Redirect?SAMLRequest=<request>&RelayState=

<relystate> HTTP/1.1

Host: localhost:8080/sso

Vrijednosti parametara SAMLRequest i RelayState su iste kao one proslijedjene od strane servis provajdera. SSO servis na IdP strani procesira <samlp:AuthnRequest> element tako što ga prvo URL-dekodira, base64-dekodira te dekompresuje (tim redosledom) a zatim vrši sigurnosnu provjeru. Ako korisnik nema važeći sigurnosni kontekst, IdP identifikuje korisnika nekim od mehanizama autentifikacije, tj u ovom slučaju login formom.

1. **Odgovor XHTML formom**

SSO servis nakon validiranja zahtjeva odgovara stranicom koja sadrži XHTML formu:

<form method=”post” action=”http://localhost:8081/sp1/saml”>

<input type=”hidden” name=”SAMLResponse” value=<samlres> />

<input type=”hidden” name=”RelayState” value=<relstate> />

...

<input type=”submit” value=”Submit”>

</form>

Vrijednost RelayState parametra je očuvana jos iz 3. koraka. Vrijednost SAMLResponse parametra je base64 enkodiran <samlp:Response> element:



1. **Zahtjevanje servisa za konzumiranje odgovara na SP strani**

Preglednik korisnika šalje POST zahtjev servisu za konzumiranje odgovora na strani servis provajdera.

POST /saml HTTP/1.1

Host: localhost:8081/sp1

Content-Type: application/x-www-form-urlencoded

SAMLResponse=response&RelayState=token

gdje su vrijednosti SAMLResponse i RelayState parametara uzete sa XHTML forme.

1. **Preusmjeravanje na željeni resurs**

Servis za konzumiranje odgovora obradjuje odgovor, stvara sigurnosni kontekst na strani servis provajdera i preusmjerava korisnika na željeni resurs koristeći referenu smještenju u RelayState parametru.

1. **Novi zahtjev za željeni resurs na strani SP-a**

Korisnik upotrebom preglednika ponovo zahtjeva novi resurs na strani servis provajdera:

https:// localhost:8081/sp1/protectedResource2

1. **Odgovor vraćanjem željenog resursa**

S obzirom na to da već postoji sigurnosni kontekst, servis provajder vraća zahtjevani resurs.

## Integracija SSO klijenata sa SSO serverom

U ovom dijelu objasnićemo od čega se sastoje klijent i server aplikacije, kako je izvršena integracije i konfigurisanje klijent aplikacija za konzumiranje SSO servera, te na koji način se stvara sigurnosni kontekst na strani klijenta.

Kao što je ranije navedeno, praktični dio ovo rada čine dvije ServiceProvider (SP) aplikacije koje predstavljaju klijente i IdentificationProvider koji predstavlja SSO server na koji se klijentske aplikacije povezuju.

SP aplikacije su implementirane su kao veoma jednostavne Spring Boot aplikacije koje služe za demonstrativnu srvhu, kako bi se pokazalo i testiralo autentifikovanje i autorizovanje putem SSO servera. Sadrže smo dvije stranice, tj početnu stranicu koja je nezaštićena tj pristup je omogućen svim korisnicima, te testnu stranicu koja je zaštićena i zahtjeva da korisnik bude autorizovan sa određenom rolom odnosno ulogom.

IdP aplikacija je takodje Spring Boot aplikacija, povezana sa bazom podataka čija se struktura vidi na narednoj slici. Sastoji se samo od login stranice koja se prikazuje ukoliko korisnik (preusmjeren na IdP od strane SP-a) još uvijek ne posjeduje važeći sigurnosni kontekst na IdP strani, tj nije autentifikovan. Za kreiranje SAML poruka koristi se OpenSAML biblioteka. IdP aplikacija sadrži Maven plugin koji kreira JAR paket koji se koristi u okviru klijentskih aplikacija kako bi se pripremile sve neophodne stvari za komunikaciju sa IdP-om. Ovaj paket sadrži metodu sa kreiranje AuthnRequest poruka i pripremu za slanje putem redirekcije, metodu za generisanje odgovarajuće HTTP redirekcije koja sadrži tačnu putanju do IdP-a te metodu za konvertovanje SAML odgovora u odgovarajući Response objekat iz OpenSAML biblioteke.



Tabele users i user\_roles su tabele koje Spring Security koristi kako bi authentifikovao i autorizovao korisnika na strani IdP-a, odnosno nad ovim tabelama se vrše upiti prilikom autentifikacije korisnika na SSO serveru nakon redirekcije od SP-a. Tabele target\_hosts i target\_authorities sadrže autorizacijske podatke koje autentifikovani korisnik posjeduje na određenim SP aplikacijama odnosno klijentima.

### Pozadinski proces prilikom pristupa zaštićenoj stranici na strani SP-a

1. Korisnik putem preglednika pristupa početnoj stranici prve SP1 aplikacije – localhost:8081/sp1, pri čemu mu se otvara stranica sa objašnjenjem i linkom za zaštićenu stranicu
2. Kada korisnik klikne na link za zaštićenu stranicu, aplikacija detektuje da ne postoji sigurnosni kontekst neophodan da se se korisniku prikaže stranica. Usled toga na SP strani se kreira SAML zahtjev koji predstavlja AuthnRequest element. Uz kreiranje zahtjeva, generiše se i RelayState vrijednost koja se čuva u mapi zajedno sa adresom prethodno traženog resursa tj stranice, kako bi se nakon povratka sa SSO servera korisnik mogao automatski redirektovati na nju. Kada je sve spremno, prethodno generisane vrijednosti se koriste kako bi se poslala HTTP redirekcija na SSO server.

SAML zahtjev se prvo kompresuje, base64 enkodira i URL enkodira i ta vrijednost se salje kao SAMLRequest parametar redirekcije. Uz to se dodaje i RelayState parametar koji predstavlja referencu na informacije u mapi na strani SP-a. Konačno, vrši se redirekcija korisnika na SSO server, i to na putanju koja je konfigurabilna, a predstavlja komponentu SSO servera koja obrađuje SAML zahtjeve.

1. Prilikom redirekcije, korisnik se šalje na IdP putanju koja vrši obradu AuthnRequest zahtjeva – localhost:8080/sso/Redirect. Medjutim, ova putanja je takodje zaštićena, a pošto korisnik još nema važeći sigurnosni kontekst na IdP strani, prikazuje mu se stranica za autentifikaciju na SSO server. Na toj stranici potrebno je unijeti pristupne podatke, tj korisničko ime i lozinku čije se postojanje zatim provjerava u bazi.

Nakon što se uspješno autentifikuje, korisnik dobija važeći sigurnosni kontekst na strani IdP-a, tako da dobija pristup localhost:8080/sso/Redirect putanji. Kako se na ovoj putanju nalazi komponenta koja vrši obradu zahtjeva, iz AuthnResponse-a se izvlači podatak o servisnom provajderu tj clijentu koji je uputio zahtjev za autorizaciju korisnika. Prije svega, AuthnResponse se mora prvo URL dekodovati, base64 dekodovati i dekompresovati. S obizirom na to da znamo ko je prethodno autentifikovani korisnik i koji servisni provajder je izdao zahtjev, IdP u bazi pronalazi koja prava pristupa korisnik ima na tom provajderu, tj prikuplja uloge odnosno role tog korisnika. Ova informacija se dodaje kao vrijednost Attribute elementa pod nazivom „role“ koji se nalazi pod AttributStatement elementom SAML odgovora. Kasnije će SP1 na osnovu ove vrijednosti znati koja su prava pristupa korisnika.

Nakon što je SAML Response kreiran i popunjen, nad njegovim String oblikom se vrši base64 enkodovanje. Ta vrijednost se dodaje kao atribut XHTML forme zajedno uz nepormijenjenu RelayState vrijednost, podaci se šalju nazad do SP-a HTTP POST metodom. Adresa na koju se šalju podaci je poznata tako što IdP izvači vrijednos Issuer elementa iz AuthnRequest-a.

1. Kada HTTP POST zahtjev stigne na ciljanu putanju na SP-u, u ovom slučaju na localhost:8081/sp1/saml koja predstavlja tzv „Assertion Consumer Service“, tj servis za konzumiranje SAML odgovora vrši se njegova obrada. Prvi korak obrade odgovora je njegova validacija koja se vrši provjeravanjem određenih informacija kao što su vrijeme kreiranja i krajnje vrijeme do kog je odgovor važeći, itd.

Ukoliko je odgovor validan, SP1 izdvaja role za datog korisnika i na osnovu tih rola za njega kreira odgovarajući sigurnosni kontekst, u našem slučaju Spring Securty kontekst sa datim rolama. Ovim je korisnik autentifikovan i autorizovan u okviru SP1 aplikacije. Nakon toga, na osnovu RelayState state vrijednosti koja je takodje primljena kao dio HTTP odgovora, SP pronalazi inicijalno željenu stranicu kojoj je korisnik želio pristupiti i vrši redirekciju korisnika na tu stranicu. Ukoliko se neka od rola koje korisnik posjeduje poklapa sa rolama potrebnim za pristup stranici, korisnik će je uspješno vidjeti, u suprotnom, dobiće stranicu sa porukom Access Denied.

1. Ako korisnik pokuša da pristupi nekoj drugoj zaštićenoj stranici u okviru SP1 aplikacije, izvršiće se provjera postojanja važećeg sigurnosnog konteksta u okviru te aplikacije. Ukoliko kontekst već postoji pošto je kreiran u prethodnom koraku, SP1 neće slati novi SAML zahtjev SSO serveru tj IdP-u nego će odlučiti o pravima pristupa željenoj stranici na osnovu postojećeg sigurnosnog konteksta.
2. Ukoliko taj isti korisnik sada pokuša da pristupi zaštićenoj stranici u okviru druge klijentske aplikacije SP2 na kojoj još nema sigurnosni kontekst, ponoviće se svi prethodno navedeni koraci, s tim da na IdP strani neće morati ponovo da unosi pristupne podatke jer već postoji sigurnosni kontekst za tog korisnika na strani IdP-a. Na taj način će korisnik biti automatski redirektovan na SP2, gdje će mu biti kreiran sigurnosni kontekst sa odgovarajućim rolama, odnosno biće autentifikovan i autorizovan i u okviru druge klijentske aplikacije.

## Konfigurisanje i pokretanje SP i IdP aplikacija

S obzirom da su u pitanju Spring Boot aplikacije, sama konfiguracija i pokretanje su veoma jednostavni. Pošto IdP koristi bazu za autorizaciju korisnika i izvlačenje rola, neophodno je da na računaru postoji SQL baza podataka. U okviru IdP projekta se nalaze skripte za inicijalizovanje baze (**schema.sql**) i popunjavanje tabela sa testnim podacima (**data.sql**).

Unutar **application.properties** fajlova u okviru sve tri aplikacije, promjenom **server.context** i **server.port** parametara moguće je prilagoditi kontekst i port na kom će server uslužiti aplikaciju. Za klijentske aplikacije je moguće podesiti putanju koja će vršiti obradu SAML odgovora, a trenutno je postavljena na **/saml**.

**Napomena:** Paziti da je u okviru **application.properties** fajlova koji se nalaze u klijentskim aplikacijama ispravno podešena adresa IdP-a tj SSO servera – **sso.idp.address**

Takođe, u slučaju promjene konteksta SP aplikacija, potrebno je prilagoditi vrijednosti SQL promijenjivih **@host1Url** i **@host2Url** unutar **data.sql** skripte u okviru IdP projekta, kako bi se mogle ispravno učitati role za testne korisnike.

Za pokretanje serverske i klijentskih aplikacija potrebno je izvršiti naredne korake:

1. Raspakovati **sso-project.zip** na zeljenu lokaciju na disku ili odraditi checkout projekta sa GitHub repozitorija - <https://github.com/racog3/sso-project.git>
2. Otvoriti direktorij pod nazivom **IdentificationProvider** i podesiti **application.properties** fajl tako da **spring.datasource** parametri imaju ispravne postavke za povezivanje na bazu podataka.
3. Izvršiti Maven komandu mvnw clean install. Ovim korakom će se izvršiti i pakovanje neophodne klijentske biblioteke 'IdentificationProvider-x.x-client.jar' te njeno instaliranje u lokalni maven repozitorij kako bi se mogla koristiti kao *dependency* u okviru klijentski aplikacija SP1 i SP2.
4. Pokrenuti SSO server tj IdP aplikaciju:

java -jar target/IdentificationProvider-0.0.9-SNAPSHOT.jar

1. Otvoriti i direktorije pod nazivima '**ServiceProvider**' i '**ServiceProvider2**' i u okviru njih takodje izvršiti mvnw clean install komandu.
2. Pokrenuti SSO klijente tj SP aplikacije:

java -jar target/ServiceProvider-0.0.1-SNAPSHOT.jar

java -jar target/ServiceProvider2-0.0.1-SNAPSHOT.jar

Podrazumijevane adrese pri lokalnom pokretanju aplikacija:

SP1 - <http://localhost:8081/sp1>

SP2 - <http://localhost:8082/sp2>

IdP (SSO) – http://localhost:8080/sso/

Predefinisani tesni korisnici u **data.sql** skripti:

1. Korisnik 1

username: [user1@etfbl.net](mailto:user1@etfbl.net)

password: 1234

Ima pristup zaštićenim stranicama na obije klijentske aplikacije:

<http://localhost:8081/sp1/protectedResource>

<http://localhost:8081/sp1/protectedResource2>

<http://localhost:8082/sp2/protectedResource>

<http://localhost:8082/sp2/protectedResource2>

1. Korisnik 2

username: [user2@etfbl.net](mailto:user2@etfbl.net)

password: 1234

Ima pristup zaštićenim stranicama na drugoj klijentskoj aplikaciji:

<http://localhost:8082/sp2/protectedResource>

<http://localhost:8082/sp2/protectedResource2>

ali nema pristup žaštićenim stranicama na prvoj klijentskoj aplikaciji:

<http://localhost:8081/sp1/protectedResource>

<http://localhost:8081/sp1/protectedResource2>

# Literatura

1. <https://auth0.com/blog/what-is-and-how-does-single-sign-on-work/>
2. <http://www.tomsitpro.com/articles/single-sign-on-solutions,2-853.html>
3. <http://docs.media.bitpipe.com/io_12x/io_122457/item_1111347/03-0615_HB_Making%20Sign-on%20Simple%20and%20Secure.pdf>
4. <https://www.icann.org/news/blog/what-is-authorization-and-access-control>
5. <https://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html>
6. <https://community.emc.com/docs/DOC-7314>
7. <https://www.mutuallyhuman.com/blog/2013/05/09/choosing-an-sso-strategy-saml-vs-oauth2/>
8. <http://tech-nugget.blogspot.rs/2013/09/saml-explained.html>
9. <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>
10. <http://wiki.servicenow.com/index.php?title=SAML_2.0_Web_Browser_SSO_Profile#gsc.tab=0>
11. <http://saml.xml.org/wiki/sp-initiated-single-sign-on-postartifact-bindings>
12. <https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-single-sign-on-protocol-reference>
13. [https://en.wikipedia.org/wiki/SAML\_2.0#SP\_POST\_Request.3B\_IdP\_POST\_Response](https://en.wikipedia.org/wiki/SAML_2.0%23SP_POST_Request.3B_IdP_POST_Response)