**UNIVERZITET U BANJOJ LUCI**

**ELEKTROTEHNIČKI FAKULTET**

**Rajo Gajić**

**REALIZACIJA SSO SERVERA**

**diplomski rad**

**Banja Luka, 2018**

**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. Autentikacija 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 autentikaciju i autorizaciju korisnika. Dati uporednu analizu realizovanog rješenja sa postojećim rješenjima.

Mentor: prof. dr Zoran Đurić

Kandidat: Rajo Gajić (22/07)

Banja Luka, 2018

Sadržaj

[1. UVOD 1](#_Toc511154413)

[2. AUTENTIKACIJA 2](#_Toc511154414)

[2.1 Autentikacija upotrebom lozinke 2](#_Toc511154415)

[2.2 Autentikacioni faktori 3](#_Toc511154416)

[3. UPRAVLJANJE IDENTITETIMA 4](#_Toc511154417)

[4. SINGLE-SIGN-ON (SSO) 6](#_Toc511154418)

[5. AUTORIZACIJA I KONTROLA PRISTUPA 10](#_Toc511154419)

[5.1 Autorizacija 10](#_Toc511154420)

[5.2 Kontrola pristupa 11](#_Toc511154421)

[6. SAML (SECURITY ASSERTION MARKUP LANGUAGE) 13](#_Toc511154422)

[6.1 Principi 13](#_Toc511154423)

[6.2 Kako SAML funkcioniše 14](#_Toc511154424)

[6.2.1 Tvrdnje (Assertions) 14](#_Toc511154425)

[6.2.2 Protokoli (Protocols) 15](#_Toc511154426)

[6.2.3 Povezivanja/uvezivanja (Bindings) 15](#_Toc511154427)

[6.2.4 Profili 16](#_Toc511154428)

[7. XACML (EXTENSIBLE ACCESS CONTROL MARKUP LANGUAGE) 17](#_Toc511154429)

[7.1 Arhitektura 17](#_Toc511154430)

[7.2 Elementi propisa 19](#_Toc511154431)

[7.2.1 Strukturni elementi 19](#_Toc511154432)

[7.2.2 Atributi i kategorije 19](#_Toc511154433)

[7.2.3 Ciljevi 19](#_Toc511154434)

[7.2.4 Uslovi 19](#_Toc511154435)

[7.2.5 Obaveze 20](#_Toc511154436)

[7.3 Primjer propisa 20](#_Toc511154437)

[8. Postojeća SSO rješenja 21](#_Toc511154438)

[8.1 Apereo CAS (Central Authentication Service) 23](#_Toc511154439)

[8.1.1 Osnovne karakteristike 23](#_Toc511154440)

[8.1.2 CAS protokol 24](#_Toc511154441)

[8.1.3 Arhitektura 25](#_Toc511154442)

[8.1.4 Protokoli 27](#_Toc511154443)

[8.2 Shibboleth 27](#_Toc511154444)

[8.2.1 Osnovne karakteristike 28](#_Toc511154445)

[8.2.2 Arhitektura 28](#_Toc511154446)

[8.2.3 Provajder identiteta – IdP 29](#_Toc511154447)

[8.2.4 Servis provajder – SP 30](#_Toc511154448)

[8.3 WSO2 Identity Server 31](#_Toc511154449)

[8.3.1 Osnovne karakteristike 31](#_Toc511154450)

[8.3.2 Arhitektura 32](#_Toc511154451)

[8.3.3 Servis provajder – SP 34](#_Toc511154452)

[8.3.4 Protokoli 34](#_Toc511154453)

[9. IZBOR STRATEGIJE ZA REALIZACIJU SSO SERVERA 35](#_Toc511154454)

[9.1 SAML 2.0 35](#_Toc511154455)

[9.1.1 SAML ograničenja 37](#_Toc511154456)

[9.1.2 Zaobilazno rješenje SAML HTTP POST uvezivanja 38](#_Toc511154457)

[9.2 OAuth 2.0 38](#_Toc511154458)

[9.2.1 OAuth2 nedostaci 40](#_Toc511154459)

[9.3 Rezime – SAML vs. OAuth2 40](#_Toc511154460)

[10. IMPLEMENTACIJA SSO SERVERA 42](#_Toc511154461)

[10.1 SAML 2.0 Web Browser SSO profil 44](#_Toc511154462)

[10.2 Arhitektura i način funkcionisanja 48](#_Toc511154463)

[11. UPOREDNA ANALIZA SSO RJEŠENJA 52](#_Toc511154464)

[12. ZAKLJUČAK 54](#_Toc511154465)

[13. LITERATURA 55](#_Toc511154466)

# UVOD

U današnje vrijeme IT sistemi se koriste za svakodnevno obavljanje poslova u organizacijama iz raznovrsnih grana industrije. Zavisno od same prirode poslova i procesa koji se obavljaju, takvi sistemi se primjenjuju u manjoj ili većoj mjeri, od čega zavisi i broj, tj. raznovrsnost aplikacija koje se pri tome koriste. Uobičajeno je da svaka od tih aplikacija kojoj korisnik ili zaposleni pristupa svakog radnog dana, zahtjeva i provjeru njegovog identiteta, iz sigurnosnih razloga. S obzirom na to da broj tih aplikacija može biti prilično velik, i da svaka od njih zahtijeva identifikaciju, korisnik mora pamtiti veliki broj pristupnih podataka odnosno lozinki za svaku od njih, što često predstavlja problem i povećava broj poziva prema korisničkoj podršci. To dovodi do nepotrebnog gubitka vremena predviđenog za konkretan rad, a uz to predstavlja i dodatni posao za tehničko osoblje kao i otežano održavanje korisničkih naloga, s obzirom na njihov veliki broj.

Gore navedeni problem, doveo je do razvoja procesa za autentikaciju korisnika upotrebom jednokratnog prijavljivanja, poznatijeg pod nazivom *Single-Sign-On (SSO).* Ovaj proces omogućava korisniku da se samo jednim unošenjem korisničkog imena i lozinke prijavi u više aplikacija. Najveće prednosti upotrebe ovakvog procesa su:

* Jednostavan i brz proces prijave korisnika
* Olakšano održavanje korisničkih naloga
* Povećana sigurnost sistema
* Istorija pristupa aplikacijama

Bitno je napomenuti da se SSO proces uveliko koristi i u svakodnevnom radu korisnika na internetu, s obzirom na činjenicu da većina vodećih IT kompanija poput Google-a, Facebook-a i ostalih, korisnicima nude različite aplikacije za određenu namjenu, zavisno od njihovih interesovanja. Uobičajeno je da korisnik može pristupiti svakoj od tih nezavisnih aplikacija, koje su pod vlasništvom jedne kompanije, upotrebom istih pristupnih podataka, tj. korišćenjem jednog istog naloga. Pored toga, takvi nalozi se mogu koristiti i za pristup aplikacijama drugih kompanija koje koriste njihov SSO servis. Često je moguće vidjeti opciju prijavljivanja na određenu *web* aplikaciju upotrebom postojećeg Facebook[[1]](#footnote-1) ili Google[[2]](#footnote-2) naloga.

U narednom poglavlju dat je detaljniji opis SSO procesa autentikacije kao i odgovarajućih sistema koji učestvuju u njemu. Osnovni podprocesi SSO procesa, kao što su autentikacija, upravljanje identitetima, te autorizacija i kontrola pristupa, koji grubo rečeno služe za utvrđivanje identiteta korisnika i njegovih dozvola na sistemu, opisani su u poglavljima 3, 4 i 5, respektivno. U poglavlju 6 i 7 izvršena je analiza SAML i XACML standarda koji se koriste pri razmijeni autentikacionih i autorizacionih poruka između sistema uključenih u SSO proces. Poglavlje 8 sadrži pregled i analizu postojećih SSO rješenja. Na kraju, u poglavljima 9, 10 i 11 dato je objašnjenje izbora strategije za realizaciju SSO servera te sami detalji njegove implementacije koja je praktični dio ovog rada, kao i uporedna analiza realizovanog rješenja sa postojećim rješenjima.

# AUTENTIKACIJA

Autentikacija je proces određivanja identiteta subjekta, najčešće fizičke osobe, upotrebom pristupnih podataka na osnovu kojih se može utvrditi da je subjekat upravo taj za koga se predstavlja. Pri tom procesu pristupni podaci se upoređuju sa podacima u bazi ili nekom drugom izvoru liste podataka registrovanih, tj. ovlašćenih korisnika. Ako se oni poklapaju, proces autentikacije je uspješno završen i korisnik dobija ovlašćenje za pristup sistemu.

Autentikacija može biti sprovedena upotrebom pristupnih podataka, SSO sistema, biometrijski, digitalnim certifikatima i javnim ključevima.

Autentikacija korisnika je ključna za osiguravanje odgovarajućih ovlašćenja i prava pristupa sistemima i uslugama, naročito od kada je sve učestalija krađa podataka i sve su veće prijetnje sigurnosti informacija. Iako autentikacija ne može u potpunosti spriječiti krađu informacija i identiteta, takve resurse moguće je zaštititi upotrebom jedne ili više autentikacijskih metoda.

## Autentikacija upotrebom lozinke

U privatnim i javnim računarskim mrežama (uključujući internet), autentikacija 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 pri prvoj upotrebi sistema registruje (ili ga je već neko registrovao, npr. sistem administrator) upotrebom korisničkog imena i lozinke, bilo predodređene ili po njegovom izboru. Međutim, autentikacija na osnovu korisničkog imena i 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. Pored toga, ljudi često prave i slabe lozinke, ukoliko se ne postave pravila za njihovo kreiranje. Međutim, čak i jake lozinke mogu biti ukradene, slučajno otkrivene ili zaboravljene. Iz tog razloga, internet poslovanje i mnoge druge transakcije zahtijevaju strože procese i metode autentikacije.

Nedostaci autentikacije 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 uz kombinovanje velikih i malih slova sa brojevima i simbolima.

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

Autentikacioni faktor, odnosno metod, predstavlja način na koji se vrši provjera identiteta korisnika. Tri najčešće kategorije su opisane kao „nešto što znamo“ (faktor znanja), „nešto što posjedujemo“ (faktor posjedovanja) i „nešto što jesmo“ (faktor nasljeđivanja). [1]

## Autentikacioni faktori

* Faktori znanja – kategorija identifikatora koja sa sastoji od informacija koje korisnik zna, kao što su lični identifikacioni broj (*Personal Identification Number - PIN*), 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 generiše token.
* Faktori nasljeđivanja – kategorija identifikatora koja se sastoji od elemenata koji su svojstveni pojedincu, u obliku biometrijskih podataka kao što su npr. otisak prsta, mrežnjača oka itd.

Lokacija korisnika i trenutno vrijeme su nešto što se smatra četvrtim i petim faktorom autentikacije. Rasprostranjenost pametnih telefona olakšava primjenjivanje autentikacije korisnika upotrebom više faktora. Većina pametnih telefona opremljena je GPS-om, tj. sistemom za lociranje pomoću satelita, omogućavajući 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 autentikacionog faktora posjedovanja.

Upotreba više faktora autentikacije postala je uobičajena praksa za prijavljivanje i izvršavanje transakcija u okviru sistema koji zahtijevaju visok nivo sigurnosti. Šta više, dvostepena autentikacija postaje standard za upotrebu u okviru uobičajenih sistema, dok se autentikacija sa tri ili više faktora koristi u sistemima koji imaju izuzetno visok stepen sigurnosti. Na ovaj način sigurnost se znatno uvećava, s obzirom na to da je mala vjerovatnoća da će napadač doći u posijed svim faktorima koji su potrebni za autentikaciju. [1]

# UPRAVLJANJE IDENTITETIMA

Upravljanje identitetima je administrativna oblast koja se bavi identifikovanjem subjekata u sistemu i kontrolisanjem njihovog pristupa resursima u okviru njega. Prava pristupa resursima i određenim funkcionalnostima sistema definišu se povezivanjem standardnih, unaprijed definisanih korisničkih prava sa utvrđenim identitetom subjekta.

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 upravljanju mobilnog pristupa korporativnim sistemima. U poslovnom okruženju, upravljanje identitetima se koristi zbog poboljšanja sigurnosti i produktivnosti.

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 (*roles*) 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. Neki od njih uključuju kreiranje naloga, kao i njihovo održavanje i uklanjanje. Drugi izazov predstavlja pronalaženje najboljeg načina da se obezbijedi autentikacija za korisnike. To može obuhvatati pravila vezana za lozinke i/ili biometriju te odgovor na pitanje kako provjeriti korisničke podatke na način da se zaštiti kompanija, a istovremeno umanji napor potreban kako bi se korisnik autentikovao u okviru više sistema.

Odgovor, odnosno rješenje, na prethodno navedene izazove nalazi se u upotrebi SSO-a i sistema za upravljanje identitetima. Odgovarajuća implementacija takvog sistema može značajno umanjiti cjelokupan administrativni posao neophodan pri upravljanju korisničkim informacijama u centralizovanom skladištu. Neki od problema koje ovakav sistem može riješiti su:

* 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 nekoliko grupa
* Suvišne ili netačne informacije koje djelimično identifikuju direktorije i identitete

Aplikacije mogu koristiti SSO proces kako bi obezbijedile korisnicima jednostavan i brz pristup sadržaju koji se čuva i uređuje na različitim vrstama sistema. To se postiže tako što se omogućava višestruka prijava upotrebom istog korisničkog imena i lozinke. Više detalja o samom SSO procesu i načinu funkcionisanja nalazi se u narednom poglavlju.

Iz sigurnosnih razloga, alati za upravljanje identitetima trebalo bi da se izvršavaju kao aplikacija na posebnom mrežnom uređaju ili serveru, bilo lokalno ili „u oblaku“ (*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. Velika prednost je i izvještavanje o aktivnostima sistema koje proizvodi revizijski dnevnik u koji se bilježe sve specifične aktivnosti koje su izvršene. [2] [3]

# SINGLE-SIGN-ON (SSO)

SSO je metodologija koja obezbjeđuje jednu akciju za autentikaciju i autorizaciju korisnika, čime im se omogućuje pristup svim računarima i sistemima na koje imaju prava, bez potrebe da se identifikuju nekoliko 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 u kom se čuvaju svi korisnički pristupni podaci i dozvole, čime se rješava problem njihovog postojanja u okviru svake od aplikacija pojedinačno. Povezivanje više različitih aplikacija upotrebom integrisanog sistema, omogućava i internim i eksternim korisnicima da koriste isto korisničko ime i lozinku za prijavu na svaku od njih.

Na **Slici 4.1** prikazan je scenario prijave korisnika na dvije različite aplikacije, bez postojanja SSO servera.

domen1.com

**Skladište kolačića preglednika**

domen2.com



Unos pristupnih podataka,

identifikovanje korisnika

**BEZ SSO SERVERA**

Unos pristupnih podataka,

identifikovanje korisnika

Odlazi na

Odlazi na

**Korisnik**

Čuva kolačić

Čuva kolačić

Slika 4.1 – Rješenje bez SSO servera

Kao što je moguće vidjeti na prethodnoj slici, korisnik nakon odlaska na svaki od domena mora da se prijavljuje pojedinačno. Nakon uspješne prijave, u skladište kolačića (*cookies*) njegovog preglednika čuva se kolačić koji služi kao identifikator sesije u okviru koje je korisnik autentikovan.

Očigledno rješenje ovog problema je dijeljenje informacija o sesijama između različitih domena, predstavljeno na **Slici 4.2**. Međutim, zbog sigurnosnih razloga, preglednik primjenjuje pravilo poznato kao pravilo istog porijekla (*same-origin-policy*). Ovo pravilo kaže da kolačićima 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 upravo problem koji SSO rješava: dijeljenje sesijskih informacija između različitih domena.

domen1.com

**Skladište kolačića preglednika**

domen2.com



Unos pristupnih podataka,

identifikovanje korisnika

**PRAVILO ISTOG PORIJEKLA**

**SPRIJEČAVA OVO**

Koristi kolačić domena1,

Identifikuje korisnika

Odlazi na

Odlazi na

**Korisnik**

Čuva kolačić

Čita kolačić domena1

Slika 4.2 - Pravilo istog porijekla

Različiti SSO protokoli dijele sesijske podatke na različite načine, ali osnovni princip je isti – postoji centralni domen preko kog se vrši autentikacija, nakon čega se sesija dijeli sa ostalim domenima na neki način. Npr. centralni domen generiše potpisani SAML token (koji može biti šifrovan). Taj token se dalje može proslijediti klijentu i može se koristiti od strane autentikacionog domena kao i od strane ostalih domena. Token može biti proslijeđen izvornom domenu putem preusmjeravanja, pri čemu on sadrži sve informacije potrebne za identifikaciju korisnika u okviru domena koji je zahtijevao autentikaciju. S obzirom na to da je token potpisan, ne može biti izmijenjen od strane klijenta ni na koji način. Ovaj scenario prikazan je na **slici 4.3**.

domen1.com

**Skladište kolačića preglednika**

domen2.com



**UPOTREBA DOMENA ZA CENTRALNU AUTENTIKACIJU**

Odlazi na

**Korisnik**

**Autentikacioni server**

(domen3.com)

Odlazi na

Preusmjeravanje

Preusmjeravanje

Slika 4.3 - Upotreba centralnog autentikacionog domena

Svaki put kada korisnik posjeti domen koji zahtjeva autentikaciju, preusmjeren je na autentikacioni domen. S obzirom na to da je korisnik već prijavljen pod tim domenom, može odmah biti preusmjeren na izvorni domen uz neophodni autentikacioni token. Uobičajeni scenario prijave korisnika upotrebom SSO servera, prikazan je na **slici 4.4**.

domen1.com

**Skladište kolačića preglednika**

domen2.com



Preusmijerava na

**UOBIČAJENI SSO**

Koristi token

za autentikaciju

Odlazi na

Odlazi na

**Korisnik**

Čuva kolačić

za domen1

**Autentikacioni server**

(domen3.com)

Čuva kolačić

za domen2

Koristi token

za autentikaciju

Šalje token i preusmijerava

Čuva kolačić

Preusmijerava na

Korisnik se identifikuje ili je kolačić dostupan

Šalje token i preusmijerava

Slika 4.4 - Uobičajeni SSO scenario

U uobičajenoj konfiguraciji SSO-a koja je navedena u gore prikazanom scenariju, korisnik prvo odlazi na domen1 na kom još uvijek nije prijavljen, zbog čega se vrši preusmjeravanje na autentikacioni domen odnosno SSO server. Ukoliko korisnik nije bio prijavljen u okviru njega (sesijski kolačić nije dostupan), korisnik se prvo autentikuje na centralnom autentikacionom serveru gdje dobija svoju sesiju i odgovarajući kolačić. Nakon toga, on se preusmjerava nazad na izvorni domen, zajedno sa autentikacionim tokenom na osnovu kog se vrši autentikacija i čuvanje kolačića u okviru domena1, s obzirom na to da token sadrži sve informacije potrebne za to. Sada, ukoliko korisnik posjeti domen2 u okviru kog još uvijek nije prijavljen, ponovo se vrši preusmjeravanje na SSO server. Međutim, kako je korisnik ovaj put već prijavljen u okviru njega, odnosno posjeduje sesijski kolačić koji drži informaciju o sesiji u okviru koje je on autentikovan na SSO server, nije potrebna ponovna identifikacija korisnika, već se on automatski preusmjerava na izvorni domen zajedno sa tokenom. Autentikacioni token ponovo sadrži informacije neophodne za autentikaciju korisnika, ali ovaj put u okviru domena2, tako da se token koristi za autentikaciju i u okviru njega.

SSO je oblik tehnologije koji olakšava proces autentikacije, 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 više aplikacija. Korisnicima se posle registracije dodjeljuju prava pristupa određenim aplikacijama, nakon čega su 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 korisničkih naloga koje IT osoblje mora da održava.

SSO sistemi unaprijeđuju sigurnost time što se cjelokupna autentikacija izvršava u okviru posebnog servera za tu namjenu. Svi autentikacioni podaci moraju prvo proći kroz poseban server za SSO, koji zatim dalje prosljeđuje određene autentikacione informacije koje posjeduje za datog korisnika. Ovakav način centralizovanog sistema za autentikaciju će vjerovatno prije odoljeti napadima i zlonamjernom pristupu nego obični autentikacioni 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. [4]

# AUTORIZACIJA I KONTROLA PRISTUPA

## Autorizacija

Autorizacija je proces dodjeljivanja dozvola nekome da uradi ili posjeduje nešto. U višekorisničkim računarskim sistemima, sistem administrator definiše koji korisnici imaju pristup sistemu i koje privilegije posjeduju u okviru njega (kao što su pristup određenim direktorijima, trajanje i vrijeme pristupa, količina skladišnog prostora itd.). Nakon prijave u računarski operativni sistem ili aplikaciju, identifikuje se kojim resursima korisnik može pristupiti tokom sesije. Prema tome, autorizaciju je moguće posmatrati i kao početna podešavanja dozvola definisana od strane sistem administratora, i kao provjeru dozvola koje se dodjeljuju korisniku nakon što je dobio pristup sistemu.

Autentikacija provjerava identitet korisnika i omogućava autorizaciju. Autorizacioni propisi diktiraju šta korisnik ima pravo da izvršava. Npr. bilo koja mušterija banke može kreirati svoj nalog kako bi pristupila online sistemu banke, ali autorizacioni propisi banke moraju da se pobrinu za to 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 autentikacije. Nakon što se korisnik autentikuje, 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* stranici ili internoj mreži kompanije. Korisnički nalog može biti uključen u grupu naloga koji dijele zajednička autorizaciona pravila. Npr. recimo da postoji baza podataka koja sadrži transakcije kupaca ali i njihove lične podatke i podatke o kreditnim karticama. Trgovac bi mogao kreirati autorizacioni propis 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 autorizacione propise kada koriste socijalne medije: Facebook, LinkedIn ili Twitter autentikuju 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 stranicama kao što su Google Docs, Dropbox, Instagram, Pinterest ili čak i sa pravljenjem dijeljenog direktorijuma na računaru. [5] [6]

## Kontrola pristupa

Uopšteno govoreći, kontrola pristupa obuhvata procese autentikacije i autorizacije. Međutim, preciznije rečeno, kontrola pristupa se odnosi na odobrenje pristupa, pri čemu sistem donosi odluku o tome hoće ili dozvoliti pristup već autentikovanom korisniku ili ne, na osnovu dozvola koje su mu dodijeljene. Dok autorizacioni propisi definišu čemu određeni korisnik ili grupa korisnika može pristupiti, kontrola pristupa predstavlja metode koje se koriste za njihovo sprovođenje.

U bilo kom modelu kontrole pristupa, postoje dvije različite vrste entiteta: oni koji mogu vršiti određene akcije na sistemu su subjekti, dok su objekti entiteti koji predstavljaju resurse kojim se treba kontrolisati pristup. Kontrola pristupa reguliše odluke i procese utvrđivanja, dokumentovanja i upravljanja, kako resursima kojim se kontroliše pristup, tako i subjektima kojima bi on trebao biti omogućen. Takođe, reguliše i metode i uslove pod kojim je pristup zahtijevanom resursu dozvoljen ili onemogućen.

Postoji više različitih modela kontrole pristupa:

* Kontrola pristupa zasnovana na atributima (*Attribute-based Access Control* - ABAC) – sprovodi se tako što se provjerava jesu li odgovarajući atributi subjekta, kao i atributi objekata kojima on pristupa, u skladu sa definisanim propisima.
* Diskreciona kontrola pristupa (*Discretionary Access Control* - DAC) – ograničavanje pristupa objektima vrši se na osnovu identiteta subjekta i grupa kojima on pripada. Kontrola je diskreciona u smislu da subjekat sa određenom dozvolom za pristup, može da prenese tu dozvolu na bilo koji drugi subjekat.
* Kontrola pristupa zasnovana na istoriji (*History-Based Access Control* - HBAC) – pristup se odobrava ili ograničava na osnovu trenutne analize istorije svih aktivnosti koje su se odvijale na sistemu
* Kontrola pristupa zasnovana na identitetu (*Identity-Based Access Control* - IBAC) –pristup objektu se odobrava ili ograničava na osnovu utvrđenog identiteta subjekta, zavisno od toga da li je na listi onih koji imaju tu dozvolu ili ne
* Obavezna kontrola pristupa – (*Mandatory Access Control* - MAC) – svaka operacija, koju izvršava bilo koji subjekat, nad bilo kom objektu, testira se prema skupu autorizacionih pravila odnosno propisa.
* Kontrola pristupa zasnovana na organizaciji (*Organization-Based Access Control* - OrBAC) – svi sigurnosni propisi su podešeni specifično za svaku od organizacija, pri čemu su definisani od strane same organizacije. Na taj način, moguće je istovremeno upravljati sa nekoliko sigurnosnih propisa, povezanih sa različitim organizacijama.
* Kontrola pristupa zasnovana na ulogama (*Role-Based Access Control* - RBAC) – ograničavanje pristupa objektima vrši se na osnovu uloga koja posjeduje određeni subjekat, odnosno koje su mu dodijeljene. Uloge se definišu u skalu sa opisom posla, autoritetom i odgovornostima u okviru organizacije. Razlika uloge u odnosu na grupu je to što u modelima koji koriste grupe, subjekat može biti član više grupa istovremeno, dok u ovom modelu subjektu može biti dodijeljena samo jedna uloga unutar iste organizacije.
* Kontrola pristupa zasnovana na pravilima (*Rule-Based Access Control* - RAC) – pristup objektu se dozvoljava ili sprječava u zavisnosti od skupa pravila koje je definisao administrator sistema. Pristupna pravila se čuvaju u odgovarajućoj listi, specifičnoj za taj objekat. [5] [6]

# SAML (SECURITY ASSERTION MARKUP LANGUAGE)

SAML je otvoreni, standardizovani format poruka, tj. protokol zasnovan na XML-u, koji služi za razmjenu autentikacionih i autorizacionih podataka. Komunikacija u kojoj se razmjenjuju takve poruke se najčešće vrši između provajdera identiteta i servisnog provajdera, u okviru SSO procesa putem internet preglednika, što i jeste jedna od najvažnijih uloga za koju se SAML koristi. Razvijen je od strane OASIS-a[[3]](#footnote-3) i datira još iz 2001. godine, s tim da se konstantno vrše nadogradnje i unapređenja. [7]

## Principi

Postoje tri različite uloge u scenarijima koje propisuje SAML specifikacija:

* Korisnik *(Principal)* - subjekat koji može biti autentikovan u okviru određenog sigurnosnog domena
* Provajder identiteta (*Identity Provider* - IdP) - sistem koji služi za dodavanje, održavanje i upravljanje informacijama o identitetu subjekta, kao i za njihovo izdavanje u obliku SAML tvrdnji koje su opisane u tački 6.2.1.
* Servis provajder (*Service Provider* - SP)- sistem odnosno servis čijim resursima korisnik želi pristupiti, i koji obavlja komunikaciju sa provajderom identiteta u cilju sprovođenja kontrole pristupa

Uobičajeni tok dešavanja u tim scenarijima počinje tako što korisnik traži pristup resursu ili određenoj usluzi na strani servis provajdera. On zatim od provajdera identiteta zahtjeva i dobavlja tvrdnje o identitetu korisnika, te na osnovu njih vrši odluke o pravima pristupa zahtijevanim resursima i uslugama.

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

SAML ne definiše način na koji će se korisnik autentikovati na strani provajdera identiteta. Mogu se koristiti korisničko ime i lozinka ili neki drugi oblik autentikacije, uključujući autentikaciju u više koraka, tj. višestepenu autentikaciju (*multi factor*). [8]

## Kako SAML funkcioniše

U suštini, SAML nije ništa više nego niz XML baziranih poruka koje daju informaciju o tome da li je korisnik autentikovan, kakva prava pristupa i uloge posjeduje, te na koji način može koristiti podatke i resurse na osnovu toga. Kompatibilan je sa HTTP, SMTP, FTP i SOAP protokolima, uz niz ostalih protokola i tehnologija. [8]

Glavne komponente SAML specifikacije su:

* Tvrdnje (*Assertions*)
* Protokoli (*Protocols*)
* Povezivanja/uvezivanja (*Bindings*)
* Profili (*Profiles*)

### Tvrdnje (Assertions)

SAML tvrdnja predstavlja skup sigurnosnih informacija dobijenih od strane provajdera identiteta. Primjer koji prikazuje strukturu SAML tvrdnje:

<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 se obično prenose od strane provajdera identiteta prema servis provajderu. Tvrdnje sadrže izjave (*statements*) koje servis provajder koristi kako bi donio odluku o pravima pristupa korisnika, nakon što on uputi zahtjev za pristup određenom resursu ili usluzi. Postoje tri vrste izjava koje SAML prenosi:

1. **Autentikacione izjave (*Authentication statements*)** – govore servis provajderu da se korisnik autentikovao na strani provajdera identiteta u određeno vrijeme i upotrebom određene metode za autentikaciju
2. **Atributske izjave (*Attribute statements*) –** govore koji specifični atributi su vezani za subjekat odnosno korisnika. Atribut predstavlja jednostavan par podataka, predstavljen u obliku „naziv-vrijednost“. Strana koja konzumira te poruke koristi atribute kako bi donijela odluke o dozvolama i pravima pristupa.
3. **Autorizacione odluke (*Authorization decision statements*) –** 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, detaljnije obrađenog u glavi 7. [8]

### Protokoli (Protocols)

SAML protokol opisuje način na koji se određeni SAML elementi (uključujući i tvrdnje) pakuju unutar SAML zahtjeva i odgovora, i definiše pravila kojih se učesnici u SAML komunikaciji 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 upit (*query*). Servis provajder upit šalje direktno provajderu identiteta putem sigurnog kanala. U skladu sa tri tipa izjava, postoje tri tipa SAML upita:

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

Među navedenim, atributski upit je možda najvažniji. Rezultat takvog upita je SAML odgovor koji sadrži tvrdnju koja se sastoji od gore navedenih atributskih izjava. [8]

### Povezivanja/uvezivanja (Bindings)

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, dok 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, koji je detaljnije obrađen u paragrafu 10.1, najčešće se koriste *HTTP Redirect Binding* i *HTTP POST Binding*. Npr., servis provajder može koristiti HTTP preusmjeravanje kako bi poslao zahtjev, dok provajder identiteta koristi HTTP POST metodu za vraćanje odgovora. [8]

### Profili

SAML profil definiše način na koji treba da se kombinuju tvrdnje, protokoli i povezivanja za određeni slučaj upotrebe SAML-a. Najvažniji SAML profil je *Web Browser SSO* profil, koji je u paragrafu 10.1 objašnjen kroz primjer, kao dio implementacije SSO servera koja se i zasniva na njemu. [8]

# XACML (EXTENSIBLE ACCESS CONTROL MARKUP LANGUAGE)

XACML (*eXtensible Access Control Markup Language*) je standard koji definiše jezik zasnovan na XML-u, dizajniran za iskazivanje sigurnosnih propisa i prava pristupa podacima u okviru web servisa i poslovnih sigurnosnih aplikacija. Pored samog jezika, definiše arhitekturu kao i proces obrade zahtjeva za pristup, u skladu sa pravilima definisanim unutar propisa. Odobren od strane OASIS-a u Februaru 2003. godine, XACML je razvijen s ciljem uspostavljanja zajedničke terminologije i kompatibilnosti između različitih implementacija sistema za kontrolu pristupa. Za XACML se nekada koristi i naziv XACL (*eXtensible Access Control Language*).

To je prvenstveno sistem koji vrši kontrolu pristupa na osnovu atributa. Atributi, koji se odnose na entitete sistema bilo u obliku korisnika, resursa, akcije ili okruženja, koriste se za donošenje odluka o tome da li bi korisnik trebao, na određeni način, dobiti pristup željenom resursu ili ne. Pored toga, XACML podržava i kontrolu pristupa na osnovu uloga, što znači da se odluke o pristupu korisnika određenom resursu donose na osnovu uloga koje su mu dodijeljene i koje sa sobom donose određene dozvole.

Ukoliko se donošenje odluka o pristupu vrši u okviru klijentske aplikacije, te dođe do promjene propisa i regulativa, veoma je teško ažurirati kriterijume koji se pri tome koriste. Iz tog razloga, XACML model preporučuje da se donošenje pristupnih odluka vrši van mjesta upotrebe, što znatno olakšava njihovo ažuriranje i omogućuje momentalnu primjenu. [9]

## Arhitektura

Arhitektura XACML-a i primjer odvijanja procesa autorizacije prikazani su na slici 7.1. Pojmovi koji se koriste pri opisu ovakvih procesa su sledeći:

* PIP (*Policy Administration Point*) – tačka upravljanja propisima je tačka koja upravlja propisima vezanim za autorizaciju, odnosno dodijelu prava pristupa
* PDP (*Policy Decision Point*) - tačka donošenja odluka je tačka koja prije izdavanja odluka o pristupu, procijenjuje zahtjev za pristup na osnovu autorizacionih propisa
* PEP (*Policy Enforcement Point*) - tačka izvršenja propisa je tačka koja presreće zahtjev korisnika za pristupu resursu, izdaje PDP-u zahtjev za donošenje odluke kako bi se utvrdilo da li je korisniku pristup odobren ili ne, a zatim postupa po toj odluci
* PIP (*Policy Information Point*) – informativna tačka je tačka koja služi kao izvor informacija o vrijednostima atributa nekog entiteta u okviru sistemu (npr. korisnik, resurs, akcija, okruženje)
* PRP (*Policy Retrieval Point*) - tačka pribavljanja propisa je tačka u kojoj su smješteni XACML propisi vezani za autorizaciju pristupa, najčešće baza podataka ili sistem datoteka



**PEP**

**PDP**

**PAP**

**PIP**

1.Učitaj resurs

2.Da li korisnik ima pristup resursu?

3. Procijena zahtijeva na osnovu propisa

Upravljanje

propisima

3.a..Pribavljanje

dodatnih atributa

4.Dozvola, korisnik

ima pristup resursu

5.Učitaj resurs

**PRP**

Pribavljanje

propisa

Slika 7.1 – Proces XACLML autorizacije

Odvijanje procesa autorizacije opisano je narednim koracima: [9]

1. Korisnik šalje zahtjev za pristup resursu, koji zatim biva presretnut od strane PEP-a, odnosno tačke za izvršenje propisa
2. PEP taj zahtijev konvertuje u XACML zahtjev za autorizaciju i prosljeđuje ga PDP-u, odnosno tački za donošenje odluka
3. PDP na osnovu podešenih propisa procijenjuje zahtijev. Propisi se dobavljaju pomoću tačke za pribavljanje propisa tj. PRP-a, dok se za upravljanje njima koristi PAP – tačka za upravljanje propisima.
   1. Ukoliko je potrebno, PDP takođe od informativnih tačaka (PIP) dobavlja vrijednosti atributa koji se odnose na entitete sistema
4. PDP donosi odluku o pristupu (dozvoljeno / odbijeno / nije primjenjivo / neodređeno) i vraća je nazad tački za izvršenje propisa (PEP)
5. PEP korisniku dozvoljava ili zabranjuje pristup, zavisno od prethodno dobijene odluke

## Elementi propisa

### Strukturni elementi

XACML je sačinjen od tri različita nivoa elemenata:

* Set propisa (*PolicySet*)
* Propis (*Policy*)
* Pravilo (*Rule*)

Set propisa može sadržati bilo koji broj propisa i setova propisa, dok propis može sadržati bilo koji broj pravila. [9]

### Atributi i kategorije

Propisi, setovi propisa, pravila i zahtijevi odnose se na entitete sistema, odnosno subjekte, resurse, okruženja i akcije.

* Subjekat je entitet koji zahtijeva pristup, te može imati jedan ili više atributa.
* Resurs je podatak, servis ili sistemska komponenta koja takođe može imati jedan ili više atributa
* Akcija definiše tip zahtjeva za pristup određenom resursu. Ima jedan ili više atributa.
* Okruženje može poslužiti kao dodatni izvor informacija. [9]

### Ciljevi

XACML omogućuje definisanje cilja, koji u stvari predstavlja set pojednostavljenih uslova koje subjekt, resurs i akcija moraju zadovoljiti, kako bi se set propisa, propis ili pravilo mogli primijeniti na određeni zahtjev. Ukoliko je taj cilj postignut, pravila definisana unutar propisa koristite se za donošenje odluke o pristupu i generisanje odgovora. Cilj takođe može biti definisan i tako da bude primjenjiv na bilo koji zahtjev. Cilj može biti sastavni dio bilo seta propisa, samog propisa ili pravila definisanog unutar njega. [9]

### Uslovi

Uslovi postoje samo u okviru pravila i u suštini predstavljaju samo napredniji oblik cilja. Razlika odnosno prednost u odnosu na cilj je to što mogu koristiti širi spektar funkcija, i još važnije, mogu se koristiti za poređenje dva ili više atributa. Upotrebom uslova moguće je sprovesti kontrolu pristupa zasnovanu na odnosima. [9]

### Obaveze

U okviru XACML-a moguće je koristiti koncept pod nazivom obaveze. To je direktiva od tačke donošenja odluka (PDP) do tačke izvršenja propisa (PEP) o tome šta se mora izvršiti prije ili posle odobrenja pristupa. Ukoliko tačka izvršenja propisa nije u stanju da se pridržava te direktive, odobreni pristup može ili ne mora biti realizovan. [9]

## Primjer propisa

U narednom dijelu dat je primjer propisa koji se odnosi na kontrolu pristupa. Korisnik gubi pristup sistemu ukoliko se na njega nije prijavio više od 30 dana, što bi u pseudo kodu izgledalo kao: **onemogući pristup ako su *trenutniDatumIVrijeme* > *datumIVrijemeZadnjegPristupa* + 30 dana**

**<xacml3:Rule** RuleId="f6637b3f-3690-4cce-989c-2ce9c053d6fa" Effect="Deny"**>**

**<xacml3:Description>**Use it or lose it: this policy denies access if lastLogin is more than 30 days away from today's date**</xacml3:Description>**

**<xacml3:Target/>**

**<xacml3:Condition** **>**

**<xacml3:Apply** FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of"**>**

**<xacml3:Function** FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than"**/>**

**<xacml3:Apply** FunctionId="urn:oasis:names:tc:xacml:3.0:function:dateTime-add-dayTimeDuration"**>**

**<xacml3:Apply** FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-one-and-only"**>**

**<xacml3:AttributeDesignator** Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" AttributeId="com.acme.user.lastLogin" DataType="http://www.w3.org/2001/XMLSchema#dateTime" MustBePresent="false"**/>**

**</xacml3:Apply>**

**<xacml3:AttributeValue** DataType="http://www.w3.org/2001/XMLSchema#dayTimeDuration"**>**P30D**</xacml3:AttributeValue>**

**</xacml3:Apply>**

**<xacml3:AttributeDesignator** Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment" AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-dateTime" DataType="http://www.w3.org/2001/XMLSchema#dateTime" MustBePresent="false"**/>**

**</xacml3:Apply>**

**</xacml3:Condition>**

**</xacml3:Rule>**

# Postojeća SSO rješenja

Ukoliko postoji potreba za uvođenjem SSO sistema u postojeću infrastrukturu, svakako prvo treba uzeti u obzir i razmotriti neka od gotovih rješenja. Za osnovne potrebe, u slučajevima gdje trenutni sistem nije previše komplikovan i zahtjevan, sigurno je moguće pronaći bar nekoliko od dole ponuđenih, koje će zadovoljiti sve potrebe. Naravno, ukoliko se radi o sistemu koji je kompleksan, ili ukoliko tim koji treba da vrši uvođenje SSO-a jednostavno želi da ima potpunu kontrolu u svojim rukama i ne želi da rizikuje da dođe u situaciju da na kraju izabrano rješenje ne zadovoljava sve potrebe, dobru odluku može predstavljati čak i implementacija sopstvenog SSO sistema, iako obično zahtijeva mnogo više vremena i truda. U tabeli 8.1 dat je pregled nekih od postojećih rješenja SSO sistema.

Tabela 8.1 - Neka od postojećih SSO rješenja[[4]](#footnote-4)

| **Naziv** | **Izdavač** | **Licenca** |
| --- | --- | --- |
| Access: One | Pirean | Proprietary |
| Accounts & SSO | Nokia, Intel,… | Free software |
| Active Directory Federation Services | Microsoft | Proprietary |
| Aerobase | Aerobase | Open source |
| Authen2cate | Authen2cate, LLC | Proprietary |
| AuthStack | Buckhill Ltd | Proprietary |
| Bitium | Bitium | Proprietary |
| CA SSO (formerly CA Siteminder) | CA Technologies | Commercial software |
| CAS / Central Authentication Service | Apereo | Free & Open Source |
| CoSign single sign on | University of Michigan | Academic |
| DistributedAccessControlSystem(DACS) | Distributed Systems Software | Free Software |
| Enterprise Sign On Engine | QueenslandUniversity of Technology | Free Software |
| Enterprise SSO, Web Access Manager | Evidian | Commercial software |
| Facebook connect | Facebook | Proprietary |
| Forgerock Access Manager | ForgeRock | Commercial |
| FreeIPA | Red Hat | Free Software |
| IBM Enterprise Identity Mapping | IBM | Free software |
| IBM Tivoli Access Manager | IBM | Proprietary |
| IceWall SSO | Hewlett-Packard Enterprise | Proprietary |
| ImprivataOneSign | Imprivata | Proprietary |
| JanrainFederate SSO | Janrain | Proprietary |
| JBoss SSO | Red Hat | Free Software |
| JOSSO | JOSSO | Free Software |
| Keycloak | Red Hat | Open source |
| LoginRadius | LoginRadius Inc. | Proprietary |
| LTPA | IBM | Proprietary |
| Microsoft account | Microsoft | Proprietary |
| myOneLogin | VMware | Proprietary |
| NetIQ Access Manager | NetIQ | Proprietary |
| Numina Application Framework | Numina Solutions | Proprietary |
| OneLogin | OneLogin Inc. | Proprietary |
| OpenAM | Open Identity Community | CDDL |
| OpenAthens | Eduserv | Proprietary |
| Oracle Identity Management | Oracle Corporation | Proprietary |
| ORY Hydra | ORY GmbH | Free software |
| PortalGuard | PistolStar, Inc. | Proprietary |
| SecureLogin | NetIQ | Proprietary |
| Shibboleth | Shibboleth | Free & Open Source |
| Ubuntu Single Sign On | Canonical Ltd. | Proprietary |
| Univention Corporate Server | Univention | Free & Open Source |
| WSO2 Identity Server | WSO2 | Free & Open Source |
| ZXID | ZXID | Free Software |

U narednom dijelu dat je pregled i izvšena analiza nekoliko od gore navedenih rješenja, izabranih po slobodnom izboru, uglavnom na osnovu njihove popularnosti i preporuke od strane relevantnih web portala. Jedan od kriterijuma pri izboru bio je podrška za SAML protokol, s obzirom na to da se ovaj rad, uključujući implementaciju servera kao praktičnog dijela, zasniva na njemu. Drugi kriterijum je bio tip licence, odnosno prednost su imala besplatna ili *OpenSource[[5]](#footnote-5)* rješenja. Na osnovu gore navedenog, za pregled i analizu su izabrana tri sledeća:

* Apereo CAS (Central Authentication Service)
* Shibboleth
* WSO2 Identity Server

## Apereo CAS (Central Authentication Service)

CAS predstavlja protokol i implementaciju servera za *Single Sign-On* na mreži. Projekat je započet 2001. godine na Yale univerzitetu, a njegov osnivač je Shawn Bazern, koji je tamo radio kao istraživač unutar grupe za tehnologiju i razvoj. Inicijalna verzija CAS 1.0 implementirala je osnovni *Single-Sign-On*, sa jednim stepenom autentikacije. Godinu nakon osnivanja, objavljena je verzija 2.0 koja je omogućavala višestepenu autentikaciju putem proksija. U decembru 2005. godine, CAS je postao projekat JASIG (*Java in Administration Special Interest Group*) grupe. Ubrzo je nastao CAS 3.0 koji je uveo novu arhitekturu, koja je omogućavala jednostavno proširivanje sistema i povezivanje na CAS server bez potrebe za izmjenom koda. Konačno, od Decembra 2012. godine, JASIG se spojio sa Sakai fondacijom, i ove dvije organizacije su ujedinjene u Apereo fondaciju koja i dalje radi na razvoju i održavanju CAS-a. [10]

### Osnovne karakteristike

* Java server zasnovan na Spring Framework-u
* Autentikacija u odnosu na LDAP, bazu podataka, ActiveDirectory itd.
* Podrška za različite protokole – CAS, SAML, WS-Federation, OAuth2, OpenID
* Podrška za višestepenu autentikaciju posredstvom provajdera kao što su DuoSecurity, Google Authenticator, Microsoft Azure itd.
* Podrška za delegaciju autentikacije vanjskim provajderima kao što su ADFS, Facebook, Twitter itd.
* Upravljanje lozinkama, nadzor ponašanja sistema, statistika u realnom vremenu
* Upravljanje i registrovanje klijentskih aplikacija sa određenim autentikacionim propisima
* Podrška za različite platforme – Java, .Net, PHP, Perl, Apache itd. [11]

### CAS protokol

CAS protokol je jednostavan *Single-Sign-On* protokol zasnovan na tiketima. Podrazumijeva postojanje jednog ili više klijenata i jednog servera. Klijenti su integrisani u aplikacije koje treba da komuniciraju sa CAS serverom, pa se one nazivaju CAS servisima.

* CAS server je zadužen za autentikaciju korisnika i odobravanje pristupa aplikacijama
* CAS klijent štiti CAS aplikaciju i dobavlja identitet korisnika od CAS servera

Za slanje zahtjeva i vraćanje odgovora koristi se HTTP GET metoda, pri čemu se za vraćanje odgovora, koji u suštini predstavlja XML sadržaj, može takođe koristiti i HTTP POST.

Osnovna CAS autentikacija koja omogućava *Single-Sign-On* funkcionalnost, implementirana još u verziji CAS 1.0, veoma je jednostavna i u osnovi funkcioniše na sledeći način:

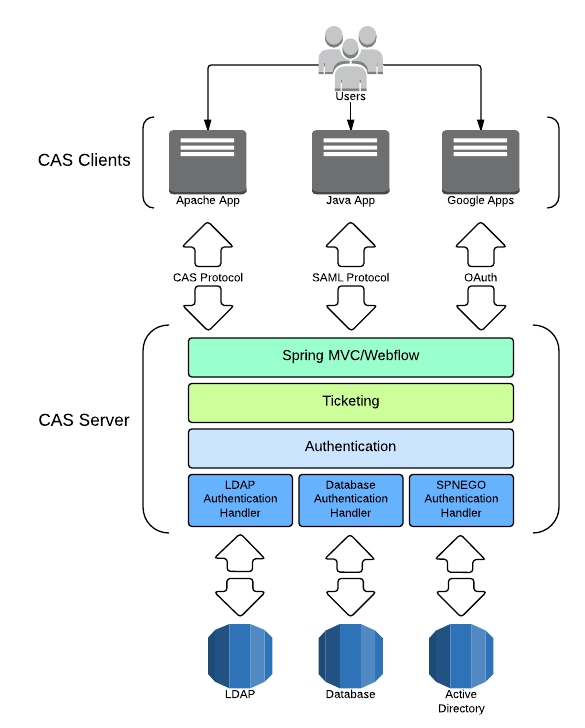
Kada klijent posjeti aplikaciju (CAS servis) koji zahtijeva autentikaciju, ona ga preusmjerava na CAS server koji vrši autentikaciju korisnika. To se najčešće vrši provjerom njegovih pristupnih podataka, odnosno korisničkog imena i lozinke, u odnosu na bazu podataka. Ukoliko su podaci ispravni, CAS server preusmjerava klijenta nazad na aplikaciju, zajedno sa servisnim tiketom, koji u suštini predstavlja jedinstven skup karaktera koje klijent koristi kao ključ za pristup servisu. Kada aplikacija primi tiket, ona ga putem sigurnog kanala šalje nazad CAS serveru na provjeru, pri čemu prosleđuje i svoj identifikator servisa. CAS server aplikaciji daje pouzdanu informaciju, u obliku XML odgovora, o tome da li je dati korisnik uspješno autentikovan ili ne. Na osnovu tog odgovora, CAS servis donosi odluku o tome da li korisniku treba dozvoliti pristup željenom resursu ili ne. [13]

Kao što je ranije pomenuto, CAS 2.0 verzija protokola uvela je novine kao što su podrška za višestepenu autentikaciju, te podrška za autentikaciju putem proksija, koja je omogućila da se jedan CAS servis ponaša kao proksi za neki drugi CAS servis, tako što mu prosleđuje identitet korisnika.

Najvažnije unapređenje u verziji CAS 3.0 u odnosu na verziju CAS 2.0 predstavlja mogućnost dobavljanja autentikacije i određenih atributa koji se odnose na nalog korisnika, putem nove pristupne tačke (*endpoint*-a). Ovo je korisno u slučaju kada aplikacija koja zahtijeva autentikaciju, osim kreiranja lokalne sesije na osnovu odgovora koji dobije od CAS servera, za autorizovanje pristupa mora da obavi još neke dodatne provjere identiteta korisnika, na osnovu njegovih atributa. [12]

### Arhitektura

Kao što je rečeno, CAS naravno postoji i kao server, što i jeste glavna tema ovog poglavlja, s obzirom na to da se odnosi na postojeća rješenja, tj. implementacije SSO sistema. CAS server i klijent su dvije osnovne komponente arhitekture CAS sistema, predstavljene na narednoj slici 8.1, i mogu da vrše komunikaciju različitim vrstama podržanih protokola, navedenih u tački 8.1.4. [14]



Slika 8.1 – Arhitektura CAS sistema[[6]](#footnote-6)

**CAS server** je Java servlet izgrađen na Spring Framework-u, a sastoji se od sledećih podsistema, koji predstavljaju tri različita sloja servera:

* Web (Spring MVC/Spring Webflow)
* Tiketi
* Autentikacija

Web sloj predstavlja pristupnu tačku koja vrši komunikaciju sa vanjskim sistemima, uključujući i CAS klijente. Oslanja se na podsistem tiketa, koji vrši generisanje tiketa za pristup CAS klijentima. Da bi se tiket izdao, korisnik mora biti uspješno autentikovan, tako da se ovaj podsistem dalje oslanja na podsistem za autentikaciju, koji najčešće samo obrađuje zahtjeve koji se šalju prilikom kreiranja SSO sesije. Podsistem za autentikacija provjerava pristupne podatke korisnika u odnosnu na bazu podataka, LDAP bazu, ActiveDirectory servis ili drugi od podržanih izvora autentikacionih podataka.

Kao glavne zadatke ima autentikaciju korisnika, te odobravanje pristupa CAS servisima, tj. klijentima, izdavanjem i provjeravanjem tiketa.

Pojam “**CAS klijent**” ima različito značenja zavisno od konteksta u kom se koristi. CAS klijent je bilo koja aplikacija koja je podešena da vrši komunikaciju sa CAS serverom putem nekog od podržanih protokola, i za nju se koristi i naziv CAS servis. Pored toga, CAS klijent predstavlja i softverski paket koji može biti integrisan u razne platforme i aplikacije kako bi one mogle da komuniciraju sa CAS serverom.

Podržane platforme:

* Apache Httpd Server
* Java
* .NET
* PHP
* Perl
* Python
* Ruby

Podržane aplikacije:

* Canvas
* Atlassian Confluence i JIRA
* Drupal
* Liferay
* UPortal
* ...

### Protokoli

Klijent komunicira sa serverom putem bilo kog od nekoliko podržanih protokola. Iako su oni uglavnom međusobno slični, svaki od njih posjeduje neke mogućnosti koje ga izdvajaju od ostalih, zbog čega predstavlja bolji izbor u određenom slučaju upotrebe. Npr., CAS protokol podržava autentikaciju posredstvom proksija, dok SAML protokol podržava *Single-Log-Out* (SLO). [14]

Podržani su sledeći protokoli:

* CAS (1.x, 2.x, 3.x)
* SAML (v1.1 i v2)
* OpenID / OpenID Connect
* OAuth 2.0
* WS Federation

## Shibboleth

Shibboleth je još jedna od *Open-Source* implementacija SSO sistema, namijenjena za upotrebu unutar jedne ili između više različitih organizacija. To je u suštini softverski paket koji kao dvije osnovne komponente, tj. aplikacije, sadrži sledeće - provajder identiteta (IdP) koji obezbjeđuje sigurnosne informacije o korisnicima, te servis provajder (SP) koji koristi te informacije i na osnovu njih odobrava pristup zaštićenom sadržaju. Pored toga, moguće je koristiti i treću komponentu - servis za otkrivanje (*Discovery Service* - DS) koji pomaže servis provajderima da otkriju tj. pronađu provajdere identiteta posredstvom kojih mogu vršiti autentikaciju korisnika. Kako bi obezbijedila *Single-Sign-On* funkcionalnost, implementacija se zasniva na široko prihvaćenih standardima vezanim za jedinstvenu identifikaciju korisnika, i to prvenstveno na SAML standardu. Korisnik se autentikuje tako što unosi svoje pristupne podatke u okviru provajdera identiteta, koji zatim servis provajderu šalje minimalnu količinu informacija o korisniku, potrebnu za donošenje odluka o pravima pristupa.

Shibboleth je započet 2000. godine kao projekat Internet2 zajednice, nakon čega je uvezan sa OASIS SAML razvojnom grupom. 2003. godine izbačena je prva verzija Shibboleth 1.0, koja je stekla je veliku popularnost i prihvaćenost od strane istraživačkih i obrazovnih zajednica širom svijeta. Druga verzija, Shibboleth 2.0, izašla je godinu dana nakon objavljivanja SAML 2.0 protokola, a glavna nadogradnja bila je upravo podrška za novu verziju protokola. Pored toga, značajnije promjene su i unapređena enkripcija, te definisanje podrazumijevanog trajanja sesije od 30 minuta. Treća verzija, Shibboleth 3.0, objavljena je 2014. godine, a fokus je stavljen na podršku za naprednu autentikaciju te veću fleksibilnost izmjene i podešavanja sistema. [15]

Trenutno, poslednja stabilna verzija Shibboleth provajdera identiteta je IdP 3.3.2[[7]](#footnote-7), dok je poslednja verzija servis provajdera SP 2.6.1[[8]](#footnote-8).

### Osnovne karakteristike

* Podrška za LDAP, Kerberos, JAAS, X.509, SPNEGO, Duo Security
* Podrška za čitanje korisničkih podataka iz LDAP direktorijuma i relacionih baza podataka
* Potpuna kontrola nad podacima koji se prosleđuju zavisnim sistemima (servis provajderima)
* Odlična skalabilnost – jedna instanca može obrađivati milione autentikacionih zahtijeva dnevno i može komunicirati sa stotinama servis provajdera
* Podrška za SAML 1.1, SAML 2.0 i CAS 2 SSO protokol
* Podrška za bilo koju implementaciju servis provajdera koja je kompatibilna sa SAML 1.1 i 2.0 protokolom
* Obiman API koji omogućava proširivanje funkcionalnosti u slučaju nekih posebnih potreba i slučajeva upotrebe
* Podrška za platforme - Linux, Windows, Mac OS X, Solaris i Java Servlete [16]

Kao što je navedeno u uvodu za Shibboleth SSO rješenje, njegova implementacije se prvenstveno zasniva na SAML protokolu, ali kako je on već detaljnije obrađen u poglavlju 6 - SAML, za razliku od prethodno pomenutog CAS protokola u okviru Apereo CAS rješenja, ovdje nema potrebe za dodatnim objašnjenjem samog protokola.

### Arhitektura

Gledajući na visokom nivou, Shibboleth arhitektura definiše interakciju između provajdera identiteta i servis provajdera, u svrhu postizanja Single Sign-On funkcionalnosti na mreži. Osnovne komponente Shibboleth softverskog paketa, pa tako i arhitekture, su sledeće:

* Provajder identiteta (*Identity Provider* - IdP) – vrši autentikaciju korisnika i prosleđuje informacije o njemu ka servis provajderu. Nalazi se u okviru organizacije koja je zadužena za održavanje korisničkih naloga.
* Servis provajder (*Service Provider* - SP) – štiti resurse od neodobrenog pristupa, a odluke o pravu pristupa donosi na osnovu informacija koje dobija od provajdera identiteta. Nalazi se u okviru organizacije koja nudi te resurse.
* Servis za otkrivanje (*Discovery Service* - DS) – pomaže servis provajderu da za datog korisnika otkrije tj. pronađe odgovarajući provajder identitet, kako bi posredstvom njega izvršio njegovu autentikaciju, sa napomenom da ova komponenta nije neophodna u svim konfiguracijama sistema.

S obzirom na to da se Shibboleth oslanja prvenstveno na SAML protokol, za nazive komponenti sistema upotrebljeni su pojmovi koji su standardni i u SAML specifikaciji.



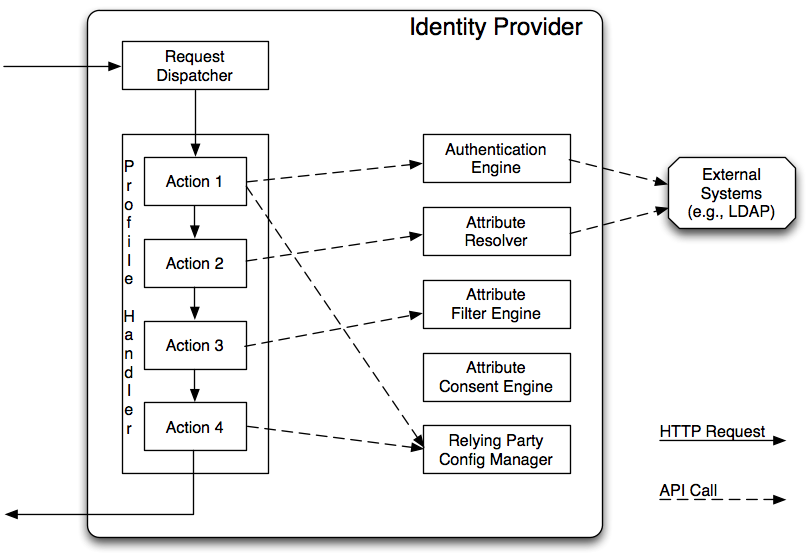
Slika 8.2 – Osnovna interakcija [17]

Na prethodnoj slici 8.2, prikazana je na visokom nivou interakcija koja se odvija između korisnika (posredstvom internet preglednika), provajdera identiteta i servis provajdera:

1. SP je ustanovio da određeni korisnik pokušava pristupiti zaštićenom resursu
2. SP preusmjerava korisnika na IdP, zajedno sa zahtjevom za autentikaciju koji je prethodno generisao
3. IdP autentikuje korisnika, nakon čega ga zajedno sa autentikacionim odgovorom šalje nazad na SP
4. SP provjerava odgovor koji je dobio od IdP-a, i ukoliko je sve u redu, dozvoljava korisniku da pristupi željenom resursu [17]

### Provajder identiteta – IdP

Kao i kod prethodnog rješenja, implementacija servera sastoji se od odvojenih dijelova koda uvezanih Spring Framework-om, u okviru kog se takvi dijelovi nazivaju komponentama (*Components*). One mogu biti manje ili veće cjeline, pri čemu više manjih komponenti mogu činiti jednu veću, koja u tom slučaju predstavlja servis.



Slika 8.3 – Arhitektura IdP-a [18]

Sama arhitektura IdP-a, prikazana na prethodnoj slici 8.3, prilično je jednostavna. Novi HTTP zahtjev dolazi prvo na *Request Dispatcher*, koji na osnovu atributa vrši njegovu analizu, i zatim ga prosleđuje *Profile Handler*-u. Njegov zadatak je da obradi zahtjev bilo kog od podržanih protokola. Pri tome se izvršavaju određene akcije, od kojih svaka vrši određeni dio cjelokupnog procesa obrade, kako bi se na kraju dobio odgovarajući odgovor. Taj odgovor se zatim vraća nazad podnosiocu zahtjeva. Same akcije koje se izvršavaju pri tom procesu, mogu biti vezane za autentikaciju korisnika, određivanje njegovih atributa, potpisivanje odgovora itd., pri čemu one takođe mogu pozivati i neke vanjske sisteme, ukoliko je to potrebno.

Za instalaciju IdP-a v3, neophodne su platforme Oracle ili OpenJDK Java 7/8, kao i Tomcat 7+ ili Jetty 8+ servlet kontejneri.[[9]](#footnote-9)

### Servis provajder – SP

Shibboleth servis provajder omogućava da se web aplikacije, napisane upotrebom bilo kog programskog jezika ili framework-a, brzo i lako integrišu u SSO sistem, bez potrebe za velikim promjenama unutar koda aplikacije.

Servis provajder, integrisan u okviru web aplikacije sa kojom zajedno predstavlja dio SSO sistema, ima sledeće osnovne zadatke:

* Presretanje zahtjeva ka zaštićenim resursima i pristupnim tačkama aplikacije
* Izdavanje SAML autentikacionih zahtjeva određenom provajderu identiteta
* Obrađivanje SAML odgovora i izvlačenje informacija o korisniku
* Sprovođenje lokalnih propisa i prikupljanje dodatnih podataka

Osim upotrebe u kombinaciji sa Shibboleth provajderom identiteta, ovaj servis provajder može se koristiti i u kombinaciji sa bilo kojom drugom implementacijom IdP-a koja je u skladu sa SAML standardom. [18]

Podržane platforme i sistemi:

* Linux OS
* Mac OS X
* Windows
* Solaris
* Java Servleti

## WSO2 Identity Server

WSO2 Identity Server (dalje u tekstu WSO2 IS) je *Open-Source* rješenje za upravljanje identitetima korisnika u okviru web aplikacija i servisa. Kao takvo, osim same SSO funkcionalnosti, nudi i brojne druge mogućnosti kao što su: kreiranje baze korisnika, autentikacija i autorizacija korisnika, delegacija pristupa, registracija korisnika, resetovanje lozinke, zaključavanje naloga itd. Kako je tema ovog rada vezana za SSO, dalji pregled i analiza će se uglavnom odnositi na SSO funkcionalnosti koju nudi ovo rješenje.

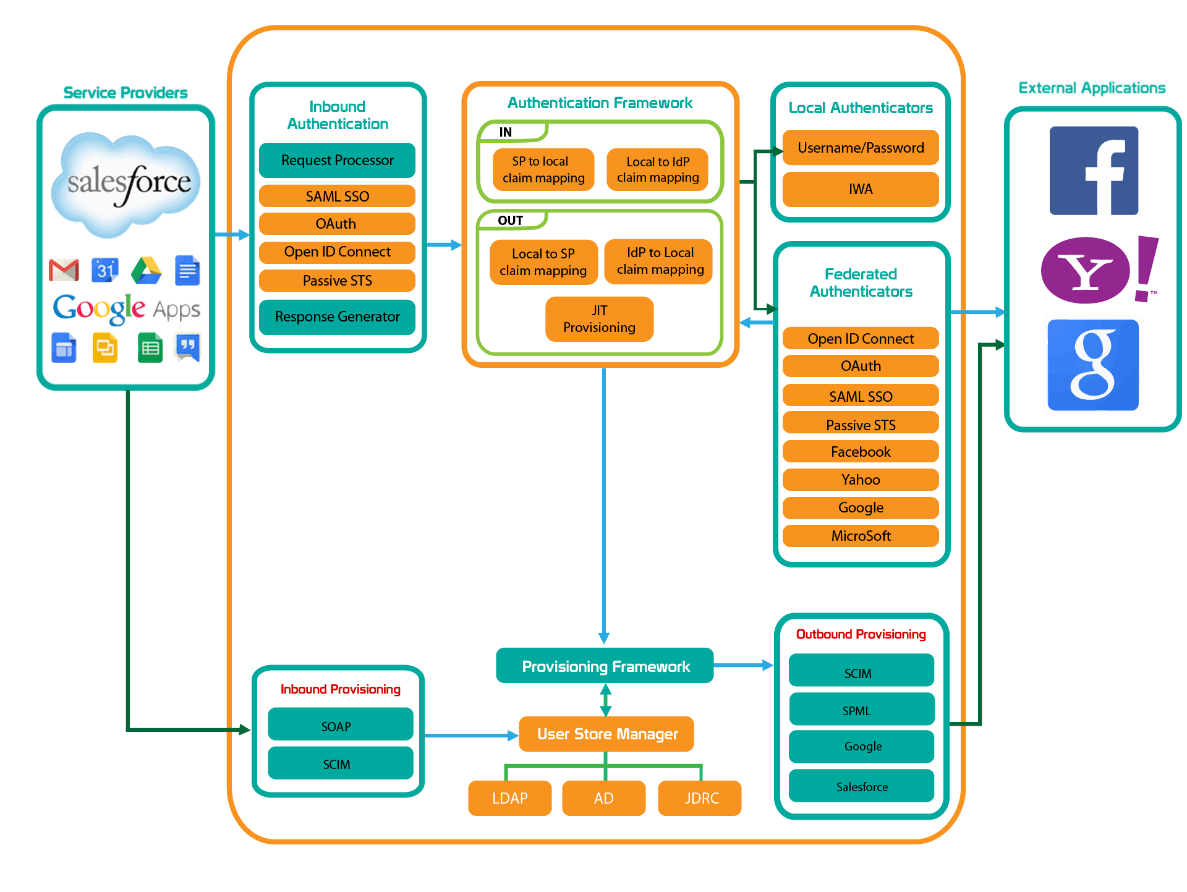
Kompanija WSO2, osnovana je u Avgustu 2005. godine od strane Sanjiva Weerawarana[[10]](#footnote-10) koji je bio istraživač u IBM-u, gdje je bio jedan od osnivača platforme web servisa[[11]](#footnote-11). WSO2 IS je samo jedan od nekoliko proizvoda ove kompanije, i njegova poslednja verzija je WSO2 IS 5.4.1[[12]](#footnote-12).

### Osnovne karakteristike

* Podrška za SSO protokole – OAuth2/OpenID Connect, SAML 2.0 i WS-Federation/Trust
* Podrška za višestepenu autentikaciju upotrebom jednokratke SMS/e-mail lozinke ili servisa kao što su FIDO, MePIN, Duo Security itd.
* Uređivanje korisničkih naloga i grupa
* Autorizacija sa XACML kontrolom pristupa zasnovanom na ulogama i atributima
* Analiza i izvještaj o aktivnostima na sistemu, kao što su prethodne autentikacije korisnika
* Podrška za delegiranu autentikaciju upotrebom WSO2 IS konektora [[13]](#footnote-13)

### Arhitektura

WSO2 IS posjeduje panel za upravljanje, čime se administratorima i ostalim korisnicima koji imaju odgovarajuće privilegije, omogućava da prijavljuju nove korisnike, dodjeljuju im odgovarajuće role, te vrše sve ostale akcije vezane za upravljanje identitetima. Međutim, taj dio funkcionalnosti sistema je van opsega *Single Sign-On*-a, ali se pored toga WSO2 IS može koristiti upravo i kao SSO server, odnosno kao provajder identiteta za nezavisne aplikacije koje imaju svoje posebne grupe korisnika, što i jeste teme ove glave.



Slika 8.4 – Arhitektura servera [19]

Na prethodnoj slici 8.4 prikazana je kompletna arhitektura servera identiteta, koja se na osnovu njegovih gore pomenutih funkcionalnosti, može podijeliti na dva velika dijela:

* Dio vezan za autentikaciju korisnika (*Autentication Framework*)
* Dio vezan za registraciju i dodavanje korisnika (*Provisioning Framework*)

Kako je tema ove glave SSO funkcionalnost postojećih rješenja, pa prema tome i autentikacija korisnika kao dio *Single Sign-On*-a, u narednom dijelu predstavljen je samo dio arhitekture vezan za autentikaciju, zajedno sa komponentama koje se odnose na taj proces, a to su sledeće:

* Servis provajderi (*Service providers*)
* Ulazni autentikatori (*Inbound authenticators*)
* Autentikacioni *framework* (*Authentication framework*)
* Lokalni autentikatori (*Local authenticators*)
* Eksterni autentikatori (*Federated authenticators*)
* Eksterne aplikacije (External applications)

**Servis provajderi** su *web* servisi koji se oslanjaju na provajder identiteta, kako bi posredstvom njega izvršili autentikaciju i autorizaciju korisnika. To se vrši slanjem odgovarajućeg zahtjeva, putem nekog od podržanih protokola među kojima se nalaze SAML 2.0, OpenID Connect i WS Federation/Trust.

**Ulazni autentikatori** su odgovorni za identifikovanje i parsiranje svih dolaznih zahtjeva za autentikaciju, kao i generisanje odgovarajućeg odgovora za primljeni zahtjev. Za svaki od podržanih protokola, postoji po jedan ulazni autentikator. Tako npr. ulazni autentikator za SAML 2.0 zahtjeve treba da prihvati SAML zahtjev od servis provajdera, provjeri ga, te u zajedničkom internom obliku proslijedi autentikacionom *framework*-u na dalju obradu. Takođe, kada od njega nazad dobije odgovor, treba da ga pretvori u odgovarajući SAML odgovor koji vraća servis provajderu.

**Autentikacioni *framework***iz zahtjeva koji je primio od ulaznog autentikatora izvlači podatak o tome koji servis provajder je ga je uputio, te na osnovu postavki koje odgovaraju tom provajderu, određuje način na koji treba izvršiti autentikaciju korisnika kako bi ona bila u skladu sa autentikacionim zahtjevom. Ova komponenta nije vezana ni za jedan određeni protokol, nego radi nad zajedničkim internim modelom u čijem obliku je i primila zahtjev od ulaznog autentikatora.

**Lokalni autentikatori** vrše autentikaciju korisnika na osnovu lokalno dostupnih kredencijala, odnosno pristupnih podataka. Autentikacija se najčešće vrši provjerom korisničkog imena i lozinke, koji su primljeni u okviru zahtjeva proslijeđenog od strane autentikacionog *framework*-a, u odnosu na bazu korisničkih naloga na koju je server povezan. Pored toga, moguće je razviti i sopstveni autentikator te ga integrisati u okviru servera. Ukoliko je autentikacija bila uspješna, lokalni autentikator će o tome informisati autentikacioni *framework*, koji će odlučiti da li je potrebna dodatna autentikacija ili ne, te proslijediti odgovor odgovarajućem ulaznom autentikatoru kako bi on izvršio generisanje odgovora koji je u skladu sa datim protokolom.

**Eksterni autentikatori** su odgovorni za autentikaciju korisnika posredstvom nekog od podržanih eksternih sistema, kao što su Facebook, Google, Yahoo, LinkedIn, Twitter ili bilo koji drugi provajder identiteta. Kada autentikacioni *framework* primi zahtjev od ulaznog autentikatora, on na osnovu zahtjeva i postavki podešenih za dati servis provajder pronalazi koji eksterni autentikatori su registrovani za njega, te komunicira sa njima u cilju eksterne autentikacije korisnika. Važi isto kao i za prethodnu komponentu - moguće je razviti i integrisati sa server sopstveni eksterni autentikator. Takođe, nakon uspješne autentikacije kontrola se prebacuje na autentikacioni *framework*, koji dalje prosleđuje odgovor ulaznom autentikatoru s ciljem generisanja odgovora u skladu sa datim protokolom.

**Eksterne aplikacije** su eksterni sistemi koji se mogu koristiti i kao provajderi identiteta u slučaju autentikacije upotrebom nekog od eksternih autentikatora. [19]

WSO2 Identity Server 5.4.0+, zahtijeva minimalno Oracle JDK 8 verziju Jave kako bi mogao da funkcioniše. Što se tiče baze podataka, podržava većinu standardnih baza, s tim da se preporučuje upotreba Oracle, PostgreSQL, MySQL, MS SQL itd.[[14]](#footnote-14)

### Servis provajder – SP

Za razliku od prethodnih SSO sistema, WSO2 ne obezbjeđuje gotovo rješenje (bilo u obliku aplikacije ili biblioteke) koje bi se moglo integrisati u okviru postojeće web aplikacije kao servis provajdera, s ciljem njegovog povezivanja na WSO2 provajder/server identiteta.

### Protokoli

U okviru SSO funkcionalnosti servera podržani su sledeći protokoli:

* SAML 2.0
* WS-Federation
* OAuth2 i OpenID Connect
* WS-Trust

# IZBOR STRATEGIJE ZA REALIZACIJU SSO SERVERA

Kao što je ranije rečeno u glavi 4 koja se odnosi na SSO, njegova glavna uloga je da uz minimalan broj unosa pristupnih podataka, korisnicima obezbijedi jednostavan i brz način prijave u više različitih sistema i aplikacija koje svakodnevno koriste u obavljanju svojih poslova.

Sa aspekta korisnika, njega ne interesuje na koji način funkcioniše SSO. Sve što on želi je da na jednostavan i brz način pristupi željenim aplikacijama, uz pamćenje što manjeg broja korisničkih imena i lozinki.

Kako bi se korisniku omogućilo takvo iskustvo, potrebno je implementirati odgovarajuće SSO rješenje. Postoji mnogo strategija za implementaciju SSO-a, među kojima su dvije popularne one zasnovane na upotrebi SAML i OAuth2, detaljnije opisane i upoređene u narednim paragrafima.

## SAML 2.0

SAML 2.0 predstavlja drugu verziju ranije opisanog skupa otvorenih standarda pod nazivom SAML, potvrđenu od strane OASIS u Martu 2005. godine kao zamjenu za SAML 1.1.

Najznačajnija promjena koju donosi SAML 2.0 specifikacija (nadalje SAML) je to što omogućava realizovanje SSO-a unutar mreže, između različitih domena. Profil koji opisuje kako se to može postići naziva se *Web Browser SSO Profile*, i ujedno je i najvažniji SAML profil. [8]

Posmatrajući na nekom nižem, tj. tehničkom nivou u odnosu na navedeno u paragrafu 6.1, učesnici u SAML scenarijima su sledeći:

* Klijent (*Client*) - web aplikacija učitana u okviru web preglednika korisnika, kojom korisnik međudjeluje sa servis provajderom
* Servis provajder (*Service Provider* - SP) – *web* server čijim uslugama i resursima korisnik pokušava pristupiti putem klijenta
* Provajder identiteta (*Identity Provider* - IdP) – server koji posjeduje identitet i pristupne podatke korisnika, i koristi se za utvrđivanje njegovog identiteta

Dijagram najčešćeg scenarija upotrebe SAML-a, dat je na narednoj slici 9.1.

**Servis provajder / SP**

**Klijent**

**Provajder identiteta / IdP**

**Korisnik posjećuje URL**

**Generiše**

**autorizacioni zahtjev**

**HTTP POST na IdP**

**Autorizacioni zahtjev usvojen**

**Korisnik poslan na stranicu za prijavu u okviru IdP-a**

**Korisnik se prijavljuje**

**SAML token generisan**

**Preusmjeravanje na SP**

**Korisnik prijavljen u okviru SP-a**

Slika 9.1 - Najčešći scenario upotrebe SAML-a

Primjer jednog slučaja upotrebe, koji opisuje gornji dijagram:

* A – korisnik otvara *web* preglednik i posjećuje servis provajdera (SP) na kome se nalaze njegovi dokumenti, međutim SP ne može sam vršiti autentikaciju korisnika.
* B – kako bi autentikovao korisnika, SP pravi SAML autentikacioni zahtjev (SAML AuthnRequest), potpisuje ga, eventualno šifruje, a zatim kodira (šifrovati znači sakriti sadržaj, dok kodirati znači obezbijediti njegovu upotrebljivost s ciljem izbjegavanja gubitka informacija ili grešaka u prenosu). Nakon toga, preusmjerava *web* preglednik korisnika zajedno sa SAML zahtjevom na provajder identiteta (IdP), s ciljem autentikacije korisnika. IdP prima zahtjev, dekodira ga, ukoliko je potrebno dešifruje, te provjerava potpis.
* C – Ukoliko je autentikacioni 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, email adresa itd.) i zajedno sa tokenom preusmjerava korisnika nazad na SP.
* E – SP provjerava SAML token, ukoliko je potrebno dešifruje ga i izdvaja informacije o identitetu korisnika – ko je taj korisnik i koje dozvole posjeduje. SP zatim korisnika prijavljuje u sistem, najčešće upotrebom kolačića ili kreiranjem nove sesije.

Na kraju jednog ovakvog procesa, korisnik može koristiti SP kao prijavljeni korisnik. Pristupni podaci korisnika se nikada ne prosljeđuju kroz SP nego samo kroz IdP. [20]

### SAML ograničenja

SAML podržava koncept uvezivanja. To je u suštini sredstvo kojim se vrši preusmjeravanje korisnika između SP-a i IdP-a. Npr. u koraku **D**, IdP preusmjerava korisnika nazad na SP.

Dva bitna tipa uvezivanja su HTTP preusmjeravanje (*HTTP Redirect*) i HTTP POST, uvedena u SAML 2.0 specifikaciji.

*HTTP Redirect* koristi HTTP preusmjeravanje kako bi se korisnik poslao nazad na SP. To uvezivanje je pogodno za kratke SAML poruke, ali se ne preporučuje njegova upotreba ukoliko su poruke duge, kao što je slučaj sa SAML tvrdnjama. Razlog tome je što većina *web* preglednika ima ograničenje od 2kB do 8kB za veličinu *HTTP Redirect* odgovora.

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

Forma može sadržati dugmić na koji korisnik treba kliknuti kako bi poslao podatke sa forme, ili se može koristiti JavaScript funkcija za automatizovano slanje. Zbog čega mora postojati forma koja se šalje? Razlog je taj što je u vrijeme kada je izašla SAML 2.0 specifikacija, to bilo moguće postići jedino upotrebom forme, tako da je bilo neophodno koristiti HTTP POST za slanje SAML tokena nazad do SP-a. Tada nije bilo drugog rješenja.

Međutim, to predstavlja problem u slučaju kada klijent nije *web*-bazirana aplikacije nego nativna aplikacija, kao što su mobilne aplikacije. Npr. recimo da je korisnik instalirao odgovarajuću aplikaciju za iPhone koja za prijavu koristi SSO upotrebom SAML-a. Posle pokretanja aplikacije, potrebna je autentikacija posredstvom provajdera identiteta, tj. IdP-a. Nakon što se korisnik autentikovao, IdP treba nekako poslati SAML token nazad do te aplikacije.

Većina mobilnih aplikacija mogu biti pokrenute upotrebom prilagođenog URL-a, kao što je npr. „my-app://authenticate“, i najvjerovatnije, provajder identiteta šalje formu koja sadrži SAML token na taj URL. Međutim, aplikacija će se pokrenuti, ali korisnik neće biti uspješno prijavljen.

Razlog tome je to što mobilne aplikacije nemaju pristup HTTP POST tijelu. One imaju pristup samo URL-u koji se koristi za pokretanje aplikacije, što znači da ne mogu pročitati SAML token. [20]

### 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 odgovarajući kod koji će nadgledati cijeli proces autentikacije. Po završetku cjelokupnog procesa, potrebno je odstraniti nepotreban HTML kod i izvući SAML token.

Drugo zaobilazno rješenje je implementacija proksi servera koji može primiti HTTP POST zahtijev, izvući SAML token iz tijela, a zatim napraviti URL koji u sebi sadrži taj token (npr. „my-app://authenticate/?SAMLRequest=gfhvcmijtrra“). Proksi server bi zatim mogao upotrijebiti HTTP preusmjeravanje kako bi doveo do toga da uređaj pokrene aplikaciju korištenjem generisanog URL-a. Kako je SAML token sada dio URL-a, aplikacija ga može izdvojiti i pročitati, te upotrijebiti za prijavu korisnika.

Još jedan pristup koji ne koristi zaobilazno rješenje je da se ne koristi SAML nego neka druga strategija koja se zasniva na upotrebi OAuth 2.0. [20]

## OAuth 2.0

Za razliku od SAML-a, OAuth 2.0 (nadalje OAuth2) je specifikacija koja je gotovo u potpunosti odbačena krajem 2012. godine. Objavljena je 2010. godine kao zamjena za inicijalnu verziju OAuth-a, koja je nastala 2007. godine kao autentikaciona metoda za Twitter API.[[15]](#footnote-15) Kao i prva verzija OAuth-a, OAuth2 omogućuje internet korisnicima da odobre *web* stranicama ili aplikacijama pristup svojim informacijama na drugim *web* stranicama, bez dijeljenja lozinke. Ima prednost u tome što je to novija specifikacija i što uzima u obzir napredak i razvoj svijeta tehnologije.

Mobilni uređaji i nativne aplikacije su u međuvremenu napredovale na način koji SAML nije mogao očekivati 2005. godine.

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

* Server resursa – *Resource Server* (*Service Provider*) – *web* server čijim informacijama korisnik pokušava pristupiti.
* Klijent – *Client* – način na koji korisnik međudjeluje sa serverom resursa. To može biti *web* aplikacija u okviru preglednika, nativna mobilna aplikacija, desktop aplikacija, serverska aplikacija itd.
* Autorizacioni server - *Authorization Server* (*Identity Provider*) - server koji posjeduje identitet i pristupne podatke korisnika, tj. utvrđuje njegov identitet i prava pristupa.

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

**Server resursa / SR**

**Klijent**

**Autorizacioni server / AS**

**Klijent zahtijeva autorizaciju**

**Dobija dozvolu za autorizaciju**

**Klijent zahtijeva pristupni token koristeći odobrenje**

**Klijent zahtijeva zaštićeni resurs koristeći pristupni token**

**SR provjerava pristupni token**

**Klijent prima resurs**

**Odobrava i izdaje pristupni token**

**Šalje informacije o korisniku**

Slika 9.2 - OAuth2 scenario upotrebe

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

* A – korisnik otvara *web* preglednik i posjećuje server resursa (SR) na kome se nalaze njegovi dokumenti, i traži određeni resurs. SR ne može sam vršiti autentikaciju korisnika, tako da preusmjerava korisnika na autorizacijski server (AS), zajedno sa zahtjevom za autorizaciju. Korisniku se prikazuje forma za prijavu i traži odobrenje da server sa resursima djeluje u njegovo ime. Korisnik se prijavljuje i zatim preusmjerava nazad na SR.
* B – SR prima kod odobrenja autorizacije kao dio preusmjeravanja i prosljeđuje ga klijentu
* C – klijent zatim koristi taj kod odobrenja autorizacije kako bi zahtijevao pristupni token od autorizacionog servera
* D – ukoliko je kod odobrenja autorizacije ispravan, autorizacioni server odobrava i izdaje pristupni token. Pristupni token se zatim koristi od strane klijenta pri zahtijevanju resursa od servera sa resursima.
* E – Server resursa prima zahtjev za resursom, zajedno sa pristupnim token. Kako bi se uvjerio da je token ispravan, šalje ga direktno autorizacionom serveru na provjeru. Ukoliko je token ispravan, autorizacioni server šalje nazad informacije o korisniku.
* F – nakon provjere korisničkog zahtjeva, SP korisniku šalje zahtijevani resurs.

Ovo je najčešći OAuth2 tok dešavanja: tok autorizacijskog koda. OAuth2 nudi tri dodatna toka dešavanja (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, i serverske aplikacije gdje korisnik nije direktno uključen u dešavanja već je dao odobrenje da aplikacija djeluje u njegovo ime.

Velika prednost OAuth2 tokova dešavanja je to što se komunikacija od autorizacionog servera nazad do klijenta i servera sa resursima vrši putem HTTP preusmjeravanja (HTTP Redirect), sa informacijama o tokenu dostupnim u obliku parametara zahtjeva. OAuth2 takođe ne pretpostavlja da je klijent u stvari web preglednik, dok *SAML Web Browser SSO Profil* ima tu pretpostavku.

Prema gore navedenom, OAuth2 se može koristiti u okviru nativnih mobilnih aplikacija, bez potrebe za zaobilaznim rješenjima kao što je to slučaj sa SAML-om. [20]

### OAuth2 nedostaci

OAuth2 specifikacija, za većinu situacija, ne propisuje kako treba da se vrši komunikacija između resursnog servera i autorizacionog servera, kao npr. u slučaju provjere tokena. Takođe, ne govori ništa o tome koje informacije o korisniku bi trebale biti vraćene i u kom obliku.

Postoji dosta toga gdje OAuth2 specifikacija navodi da su stvari „van okvira ove specifikacije“. To predstavlja jednu od mana OAuth2 specifikaciju jer su mnoge stvari prepuštene implementaciji, što bi u jednom trenutku moglo dovesti do nekompatibilne implementacije.

OAuth2 je ipak široko prihvaćen od strane Google-a, Facebook-a, Salesforce-a, Twitter-a i mnogih drugih. Najbolja osobina OAuth2 je njegova jednostavnost. Kao alternativa, OpenID Connect Basic Profile, izgrađen na OAuth2, popunjava neke oblasti i praznine koje sama OAuth2 specifikacija ne definiše. [20]

## Rezime – SAML vs. OAuth2

SAML posjeduje svojstvo koje nedostaje u OAuth2 specifikaciji: SAML token sadrži informacije o identitetu korisnika. Upotrebom OAuth2 takve informacije nisu odmah direktno dostupne, te kako bi provjerio token, resursni server mora napraviti dodatni zahtjev prema autorizacionom serveru.

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

Oba pristupa imaju svoje prednosti, i predstavljaju dobra rješenja za SSO implementaciju.

OAuth2 nudi jednostavnije i više standarizovano rješenje za podršku nativnih aplikacija, jer se njegovom upotrebom izbjegavaju zaobilazna rješenja, za razliku od SAML-a. Međutim, kako ovaj rad prvenstveno pokriva i sadrži detaljniju analizu SAML specifikacije, i pri tome ne uključuje integraciju mobilne nativne aplikacije u komunikaciju sa SSO serverom, za implementaciju koja predstavlja praktični dio rada i koja je detaljnije opisana u narednom poglavlju, korišten je SAML standard odnosno specifikacija.

# IMPLEMENTACIJA SSO SERVERA

Zadatak praktičnog dijela ovog rada je realizacija SSO servera sa osnovnim funkcionalnostima, odnosno servera koji omogućava autentikaciju i autorizaciju korisnika. Pored toga, server osim SSO funkcionalnosti podržava i *Single-Log-Out* (SLO) iako to nije dio zadatka. To je funkcionalnost koju takođe podržava i svako od analiziranih postojećih rješenja. Time se postiže još veća sigurnost korisnika, jer mu omogućava da se odjavom iz jedne od klijentskih aplikacija, odjavi sa svih aplikacija na koje je bio prijavljen upotrebom SSO servera. Na taj način se smanjuje mogućnost greške korisnika da se odjavi iz samo od jedne od aplikacija, zaboravljajući na ostale, ili misleći da će time biti odjavljen i iz svih ostalih aplikacija, što naravno u slučaju upotrebe SSO-a bez SLO-a nije tačno. Tako da u suštini implementacija SSO servera predstavlja kombinaciju SSO i SLO servera, ali dalje u tekstu će se zbog jednostavnosti koristiti samo skraćenica SSO.

Implementacija se sastoji od implementacije SSO servera, tj. provajdera identiteta (IdP-a), te dvije demonstrativne klijentske aplikacije koje predstavljaju servis provajdere (SP), koji vrše komunikaciju sa IdP-om u cilju njegove autentikacije i autorizacije. Naravno, ovaj sistem je moguće proširiti sa više klijentskih aplikacija koje se povezuju na server, uz njihovo podešavanje kako bi bile u stanju da komuniciraju sa njim.

Klijentske aplikacije su neophodne za demonstraciju funkcionisanja SSO servera, a služe i kao primjer za podešavanje nekog drugog servis provajdera koji treba povezati sa njim. Implementirane su kao veoma jednostavne *Spring Boot* aplikacije koje se sastoje od samo tri stranice. Prva odnosno početna stranica nije zaštićena, tj. pristup je omogućen svim korisnicima, dok su druge dvije stranice zaštićene i zahtijevaju da korisnik posjeduje odgovarajuću ulogu (*role*) kako bi im pristupio. Zbog lakše i jednostavnije integracije sa IdP-om, klijentske aplikacije koriste biblioteku generisanu od strane samog IdP-a, pri čemu ta biblioteka obezbjeđuje jednostavan i standardizovan način generisanja SAML poruka koje se razmijenjuju između klijenta i servera, kao i URL adresa na koje se te poruke šalju. Analizirani serveri Apereo CAS i Shibboleth takođe nude gotovo rješenje za integraciju sa IdP-om, dok to nije slučaj sa WSO IS-om.

IdP aplikacija je takođe *Spring Boot* aplikacija, i sadrži samo stranicu za prijavu koja se prikazuje ukoliko korisnik (preusmjeren na IdP od strane SP-a) još uvijek ne posjeduje važeći sigurnosni kontekst na IdP strani, tj. nije autentikovan. Kao što je gore navedeno, IdP aplikacija generiše klijentsku biblioteku koja olakšava njegovu integraciju sa serverom, a za pakovanje odgovarajućih klasa biblioteke u JAR datoteku, koristi se *Maven* *plugin*. Biblioteka sadrži metodu za kreiranje *AuthnRequest* poruke i njenu pripremu za slanje putem preusmjeravanja, zatim metodu za generisanje odgovarajućeg HTTP preusmjeravanja koje sadrži tačnu putanju do IdP-a, kao i metodu za konvertovanje SAML odgovora u odgovarajući *Response* objekat iz *OpenSAML* biblioteke.

S obzirom na to da IdP mora sadržati skladište podataka koje će da služi za čuvanje informacija o identitetu korisnika, kao i podataka o pristupnim pravima korisnika u okviru različitih klijentskih aplikacija koje su integrisane sa njim, IdP se povezuje na MySQL bazu podataka čija je struktura prikazana na narednoj slici 10.1.



Slika 10.1 – struktura baze podataka SSO servera

Tabele *users* i *user\_roles* su tabele koje *Spring Security* u okviru IdP-a koristi kako bi autentikovao i autorizovao korisnika, odnosno nad ovim tabelama se vrše upiti prilikom autentikacije korisnika na SSO server, nakon preusmjeravanja sa SP-a. Tabele *target\_hosts* i *target\_authorities* sadrže autorizacione podatke, odnosno uloge koje autentikovani korisnik posjeduje na odgovarajućim SP, tj. klijentskim aplikacijama.

Kao što se može i vidjeti na osnovu strukture baze podataka, ovaj sistem vrši kontrolu pristupa na osnovu uloga (*Role Based Access Control - RBAC*), što znači da se ograničavanje pristupa stranicama vrši na osnovu uloga (*roles*) koje korisnici, koji im pokušavaju pristupiti, posjeduju. U skladu sa testnim podacima kojim se inicijalizuju tabele IdP-a, postoje dvije različite uloge – *USER* i *GUEST*, pri čemu korisnici sa ulogom *USER* imaju pristup zaštićenim stranicama u okviru klijentskih aplikacija, dok oni sa ulogom *GUEST* nemaju. Naravno, uloge koje koriste različite klijentske aplikacije ne moraju biti iste niti dijeljene. Svaka od njih može imati uloge definisane u skladu sa potrebama same aplikacije, što se obično i vrši u skladu sa opisom posla, odgovarajućim odgovornostima u okviru aplikacije itd.

Komunikacija između klijenata i servera zasniva se na razmijeni autentikacionih i autorizacionih poruka isključivo putem SAML 2.0 protokola, za razliku od analiziranih implementacija od kojih svaka podržava bar još protokol, kao što je to slučaj sa Shibboleth-om sa podrškom za CAS, dok Apereo CAS i WSO2 IS pored njega podržavaju i OAuth, OpenID Connect, WS-Federation itd. Razmjena SAML poruka vrši se na način koji definiše *SAML 2.0 Web Browser SSO* profil, detaljnije opisan u narednom poglavlju 10.1. Za kreiranje SAML poruka koristi se besplatna *OpenSAML* biblioteka, razvijena od strane Shibboleth[[16]](#footnote-16) konzorcijuma upravo s ciljem da podrži programere koji rade sa SAML-om.

## SAML 2.0 Web Browser SSO profil

Ovaj profil podrazumijeva postojanje provajdera 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 scenario u kome SP vrši slanje SAML zahtjeva putem preusmjeravanja, dok IdP vrši slanje odgovora putem HTTP POST metode.

Servis provajder - SP

(SSO klijent)

Korisnički agent

(web preglednik)

Provajder identiteta - IdP

(SSO servis)

**1**

**Zahtjeva željeni resurs**

**(Pronalaženje IdP-a)**

**Preusmjerava na SSO servis**

**2**

**5**

**6**

**7**

7

**8**

**Zahtjeva servis za konzumiranje odgovora**

**Preusmjerava na željeni resurs**

**Zahtjeva željeni resurs**

**Preusmjerava na željeni resurs**

**Zahtjeva SSO servis**

**(Identifikuje korisnika)**

**Odgovara XHTML formom**

**3**

**4**

Slika 10.2 – Web Browser SSO profil

U narednim tačkama su objašnjeni gore navedeni koraci (korištene su adrese i primjeri poruka realizovane implementacije):

1. **Zahtijevanje željenog resursa**

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

http://localhost:8081/sp1/protectedResource

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

1. **Preusmjeravanje na IdP SSO servis**

Servis provajder generiše odgovarajući SAML zahtjev (i *RelayState* parametar), a zatim preusmjerava preglednik korisnika na SSO servis koristeći standardno HTTP preusmjeravanje.

302 Redirect

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

*SAMLRequest* može biti potpisan, ali za osnovni primjer to nije urađeno. Parametar *RelayState* predstavlja referencu ka stanju informacija koje se čuvaju na strani servis provajdera. U ovoj implementaciji služi za čuvanje reference ka informaciji koja predstavlja putanju do traženog resursa, kako bi se nakon autentikacije na SSO strani korisnik automatski preusmjerio na njega. *SAMLRequest* parametar je kompresovana, *base64* enkodirana i URL enkodirana vrijednost *<samlp:AuthnRequest>* elementa:



Slika 10.3 – primjer SAML AuthnRequest-a

1. **Zahtijevanje 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 proslijeđene 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 odgovarajući sigurnosni kontekst, IdP identifikuje korisnika nekim od mehanizama autentikacije, tj. u ovom slučaju formom za prijavu.

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* atributa je očuvana još iz 3. koraka, dok je vrijednost *SAMLResponse* atributa *base64* enkodiran *<samlp:Response>* element, koji u suštini predstavlja odgovor SSO servera na zahtijev koji je primio od servis provajdera.

Pored toga, neophodno je da se SAML odgovor šifruje ukoliko se vrši komunikacije putem HTTP a ne HTTPS protokola, međutim u ovom primjeru i implementaciji to nije urađeno zbog jednostavnosti. Ukoliko bi se server koristio u nekom konkretnom sistemu, bilo bi naravno potrebno ispuniti bar jedan od tih zahtijeva.



Slika 10.4 – primjer SAML Response-a

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

Nakon podnošenja XHTML forme na strani IdP-a, koja sadrži *SAMLResponse* i *RelayState* atribute*,* 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

1. **Preusmjeravanje na željeni resurs**

Servis za konzumiranje odgovora obrađuje odgovor, stvara odgovarajući sigurnosni kontekst na strani servis provajdera, te preusmjerava korisnika na željeni resurs koristeći referencu smještenu u *RelayState* parametru.

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

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

http://localhost:8081/sp1/protectedResource2

1. **Odgovor vraćanjem željenog resursa**

S obzirom na to da već postoji odgovarajući sigurnosni kontekst, servis provajder vraća zahtijevani resurs. [21]

## Arhitektura i način funkcionisanja

U narednom dijelu, kroz primjer kreiranja sigurnosnog konteksta u okviru dvije klijentske aplikacije (SP) koje se povezuju i vrše komunikaciju sa implementiranim SSO serverom (IdP), objašnjen je način funkcionisanja ovog SSO sistema. Pored toga, može se reći da naredni dio pokazuje i samu arhitekturu sistema, jer je iz njega moguće vidjeti kuda sve to prolaze zahtjevi, koje komponente vrše njihovu obradu, kao i kako se vrši i šalje generisanje odgovora. Takođe, objašnjava i način funkcionisanja SLO funkcionalnosti.

Kreiranje odgovarajućeg sigurnosnog konteksta i sprovođenje kontrole pristupa, nakon što korisnik zatraži pristup zaštićenoj stranici u okviru servis provajdera, odvija se u narednim koracima:

1. Korisnik putem preglednika pristupa početnoj stranici prve SP1 aplikacije – *localhost:8081/sp1*, pri čemu mu se otvara stranica sa objašnjenjem i linkom ka zaštićenoj stranici
2. Kada korisnik klikne na link ka zaštićenoj stranici, aplikacija detektuje da ne postoji odgovarajući sigurnosni kontekst neophodan za prikaz stranice. 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 preusmjeriti na nju. Kada je sve spremno, prethodno generisane vrijednosti se koriste kako bi se izvršilo HTTP preusmjeravanje korisnika na SSO server, što se dešava na sledeći način.

SAML zahtjev se prvo kompresuje, *base64* enkodira i URL enkodira, i ta vrijednost se šalje kao *SAMLRequest* parametar HTTP preusmjeravanja. Uz to se dodaje i *RelayState* parametar, kako bi se korisnik nakon povratka mogao automatski preusmjeriti na traženi resurs. Konačno, vrši se preusmjeravanje korisnika na SSO server, i to na putanju koja je podesiva, a predstavlja putanju do komponente SSO servera koja obrađuje SAML zahtjeve.

1. Prilikom preusmjeravanja, korisnik se šalje na IdP putanju koja vrši obradu *AuthnRequest* zahtjeva – *localhost:8080/sso/Redirect*, odnosno na SSO servis u okviru IdP-a. Međutim, kako je ta putanja takođe zaštićena, a korisnik još nema važeći sigurnosni kontekst na IdP strani, prikazuje mu se stranica za autentikaciju na SSO server. Na toj stranici potrebno je unijeti pristupne podatke, tj. korisničko ime i lozinku, čije se postojanje zatim provjerava u bazi podataka.

Nakon što se uspješno autentikovao, korisnik dobija odgovarajući 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 servis provajderu, tj. klijentu koji je uputio zahtjev za autorizaciju korisnika. Prije svega, *AuthnResponse* se mora prvo URL dekodovati, *base64* dekodovati i dekompresovati. S obizirom na to da je poznato ko je prethodno autentikovani korisnik i koji servis provajder je izdao zahtjev, IdP u bazi pronalazi odgovarajuća prava pristupa koja korisnik posjeduje u okviru njega, tj. prikuplja uloge (*roles*) 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 *SAMLResponse* kreiran i popunjen, nad njegovim *String* oblikom se vrši *base64* enkodovanje. Ta vrijednost se dodaje kao atribut XHTML forme zajedno uz nepromijenjenu *RelayState* vrijednost, i podaci se šalju nazad do SP-a HTTP POST metodom. Adresa na koju se šalju podaci je poznata tako što IdP očita vrijednost *Issuer* elementa iz *AuthnRequest*-a.

Kao što je u prethodnom poglavlju napomenuto, poruke zbog jednostavnosti nisu šifrovane, ali naravno u nekoj konkretnoj upotrebi to bi bilo neophodno uraditi.

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 uloge koje dati korisnik posjeduje, i na osnovu njih kreira odgovarajući sigurnosni kontekst. U ovom slučaju kreira se *Spring Security* kontekst u okviru kog su navedene i uloge koje korisnik posjeduje. Ovim je korisnik autentikovan i autorizovan u okviru SP1 aplikacije. Nakon toga, na osnovu *RelayState* vrijednosti koja je takođe primljena kao dio HTTP odgovora, SP unutar mape pronalazi inicijalnu stranicu kojoj je korisnik želio pristupiti, te ga preusmjerava na nju. Ukoliko se neka od uloga koje korisnik posjeduje, poklapa sa nekom od uloga kojoj je dozvoljen pristup stranici, korisnik će je uspješno vidjeti. U suprotnom, dobiće stranicu sa porukom da mu je pristup onemogućen (*Access Denied)*.

1. Kada korisnik pokuša da pristupi nekoj drugoj zaštićenoj stranici u okviru SP1 aplikacije, izvršiće se provjera postojanja odgovarajućeg sigurnosnog konteksta u okviru nje. S obzirom na to da kontekst već postoji, pošto je kreiran u prethodnom koraku, SP1 neće slati novi SAML zahtjev SSO serveru, nego će odlučiti o pravima pristupa željenoj stranici na osnovu postojećeg sigurnosnog konteksta i uloga definisanih u okviru njega.
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, korisnik će biti automatski preusmjeren na SP2 gdje će mu se kreirati sigurnosni kontekst sa ulogama koja posjeduje, odnosno biće autentikovan i autorizovan i u okviru druge klijentske aplikacije.

Pored SSO funkcionalnosti, server podržava i SLO, kao što je ranije rečeno u uvodnom dijelu opisa implementacije. Time se želi izbjeći da korisnik, slučajno ili zbog zabune, ostavi iza sebe žive sesije u okviru ostalih aplikacija na koje je prijavljen putem SSO servera, nakon što se odjavi iz samo jedne od njih. Na ovaj način, prilikom odjave korisnika iz bilo koje od aplikacija, vrši se njegova odjava sa svih ostalih aplikacija i to na sledeći način:

1. Korisnik posjećuje link za odjavu na drugoj SP2 aplikaciji - *localhost:8082/sp2/logout*, nakon čega se uništava sigurnosni kontekst, odnosno sesija, koju je posjedovao u okviru nje. Time je korisnik odjavljen iz SP2 klijentske aplikacije.
2. SP2 nakon toga preusmjerava preglednik korisnika na SLO servis IdP-a, zajedno sa zahtjevom za odjavu u obliku SAML *LogoutRequest*-a koji sadrži identifikator sesije na IdP strani.
3. Kada IdP primi zahtijev za odjavu, on vrši komunikaciju sa svim ostalim SP aplikacijama na koje je taj korisnik prijavljen, s ciljem uništavanje sesije i u okviru njih.

Po SAML specifikaciji[22], nakon što je IdP primio taj zahtjev, on određuje koje su to ostale aplikacije u okviru kojih je korisnik prijavljen putem njega, a zatim iterativno, za svaku od njih vrši sledeće:

* 1. Generiše novi zahtjev za odjavu – *LogoutRequest*
  2. Preusmjerava preglednik korisnika na prethodno podešenu adresu za odjavu na strani SP-a
  3. Čeka povratak sa SP-a i odgovor u obliku *LogoutResponse*-a

Kao što je moguće uočiti, a i sama definicija SLO-a u okviru SAML specifikacije na to upozorava, *Single-Log-Out* je teško implementirati, a i pored toga, neotporan je na eventualne greške do kojih može doći prilikom komunikacije i procesa odjave[22].

S obzirom na to da se odjava korisnika vrši preusmjeravanjem njegovog preglednika na svaki od SP-ova, lako je uvidjeti da u slučaju greške u bilo kom koraku procesa, lako dolazi do pucanja lanca. U tom slučaju, korisnik bi ostao i dalje prijavljen u okviru onih aplikacija koje nisu primile zahtjev za odjavu prije nego što je došlo do greške, iako je mogao biti uspješno odjavljen da se to nije desilo. Zbog toga se u ovoj implementaciji izdavanje zahtjeva za odjavu vrši na nešto drugačiji način, odnosno tako što SLO servis IdP-a šalje pojedinačne, nezavisne zahtjeve svim ostalim SP aplikacijama. Na taj način, i u slučaju greške u komunikaciji sa bilo kojom od njih, korisnik će biti uspješno odjavljen iz svih aplikacija gdje je to moguće, nezavisno od ostalih.

1. Na kraju, i sam IdP uništava sigurnosti kontekst vezan za tog korisnika, nakon čega je on uspješno odjavljen sa njega, kao i sa svih ostalih aplikacija koje su bile u stanju da uspješno prime i obrade zahtjev za odjavu.

Neke dodatne uobičajene funkcionalnosti koje ova implementacija ne posjeduje, a posjeduju je analizirana rješenja, su višestepena autentikacija kao i delegirana autentikacija koja omogućava da IdP autentikuje korisnika posredstvom nekog vanjskog sistema. Razlog tome je što je ova implementacija predstavlja osnovni SSO server sa minimalnim zahtjevima koje kao takav mora imati, odnosno funkcionalnosti autentikacije i autorizacije. Takođe, razlog izostavljanja dodatnih funkcionalnosti je to što je u pitanju „pilot“ projekat koji treba da demonstrira načina funkcionisanja jednog SSO servera. Naravno, bilo bi moguće proširiti postojeću verziju sa tim dodatnim funkcionalnostima ukoliko bi to bilo potrebno.

# UPOREDNA ANALIZA SSO RJEŠENJA

U narednoj tabeli nalazi se uporedna analiza postojećih rješenja, i rješenja koje je implementirano kao praktični dio rada. U prvoj koloni navedene su karakteristike svojstvene SSO serveru na osnovu kojih je i izvršena uporedna analiza, dok su u zaglavljima narednih kolona navedena sama rješenja, prvo postojeća a na kraju i implementirano. Neke karakteristike su takođe grupisane, kao što je to slučaj sa podržanim protokolima, tako da grupa opisuje aspekt na koji se te karakteristike odnose. Ukoliko neko od rješenja posjeduje određenu karakteristiku, na presjeku njegove kolone sa redom koji odgovara toj karakteristici, odnosno u odgovarajućoj ćeliji, nalazi se simbol (✓), u suprotnom je ćelija prazna.

Tabela 11.1 – Uporedna analiza rješenja

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Karakteristika** | | **Apereo CAS** | **Shibboleth** | **WSO2 Identity Server** | **Impl. rješenje** |
| **Podržani protokoli** | SAML | ✓ | ✓ | ✓ | ✓ |
| CAS | ✓ | ✓ | ✓ |  |
| OpenID | ✓ |  |  |  |
| OpenID Connect | ✓ |  | ✓ |  |
| OAuth | ✓ |  | ✓ |  |
| WS-Federation | ✓ |  | ✓ |  |
| WS-Trust |  |  | ✓ |  |
|  | Višestepena autentikacija | ✓ | ✓ | ✓ |  |
|  | Delegirana autentikacija | ✓ | ✓ | ✓ |  |
|  | *Single Log-Out* (SLO) | ✓ | ✓ | ✓ | ✓ |
|  | Dostupna klijentska aplikacija ili biblioteka | ✓ | ✓ |  | ✓ |
|  | *OpenSource* licenca | ✓ | ✓ | ✓ | ✓ |

Prva grupa karakteristika odnosi se na **podržane protokole**, pa su navedeni oni protokoli koji su podržani u bar jednom od rješenja.

Naredna karakteristika je **podrška za višestepenu autentikaciju**, odnosno mogućnost podešavanja servera da zahtijeva od korisnika autentikaciju upotrebom bar još jednog autentikacionog faktora, npr. upotrebom jednokratne lozinke primljene putem SMS-a/email-a ili odobravanjem pristupa putem aplikacije na mobilnom telefonu, kao što su npr. Google Authenticator, Duo Security itd.

**Delegirana autentikacija** se odnosi na mogućnost podešavanja servera da delegira tj. proslijedi autentikaciju korisnika nekom vanjskom provajderu identiteta, kao što su npr. Google, Facebook, Twitter itd.

***Single-Log-Out*** (SLO) je funkcionalnost koja omogućuje da se odjavom iz bilo koje od aplikacija, korisnik automatski odjavi iz svih ostalih aplikacija u koje se prijavio upotrebom SSO servera.

Karakteristika koja je bitna programerima je i **dostupnost klijentske aplikacije ili biblioteke**, jer se time značajno olakšava integraciju željene aplikacije, odnosno servis provajdera, u SSO sistem.

Na kraju, takođe bitnu karakteristiku, naročito za menadžment organizacije koja uvodi SSO sistem, predstavlja i to da li je u pitanju **rješenje sa *OpenSource* licencom**, koje je samim tim besplatno.

# ZAKLJUČAK

Uvođenje SSO sistema, naročito u sklopu organizacija sa velikim brojem korisnika i aplikacija kojima oni svakodnevno pristupaju za obavljanje svojih poslova, donosi brojne prednosti kao što su olakšano održavanje korisničkih naloga, povećana sigurnost sistema, veće zadovoljstvo korisnika pri korištenju sistema, itd.

SAML, kao jedan od standardnih protokola koji se koriste za razmjenu autentikacionih i autorizacionih poruka, i dalje predstavlja dobar izbor za realizaciju SSO sistema. Glavni razlozi za to su to što postoji već niz godina i podržan je u okviru većine postojećih implementacija. Iako je nastao 2001. godine, i dalje se konstantno razvija, a popularnost je stekao uglavnom zbog toga što nudi visok stepen sigurnosti, te se pokazao kao pouzdan protokol zbog čega je vremenom sve više i više korišten od strane brojnih organizacija što je i dovelo do toga da njegova popularnost dodatno raste zbog velike rasprostranjenosti.

Kako je SAML prvenstveno predviđen za razmjenu autentikacionih poruka, te autorizacionih poruka za osnovne, jednostavnije slučajeve upotrebe, OASIS organizacija je 2003. godine objavila XACML specifikaciju za potrebe iskazivanja sigurnosnih propisa i pravila za pristupanje podacima u okviru aplikacija, odnosno za implementaciju sistema za kontrolu pristupa.

Što se tiče postojećih SSO rješenja, kao što je moguće vidjeti iz tabele 8.1 dostupan ih je veliki broj, pri čemu je dobar dio njih čak i besplatan. Mnoga od njih podržavaju nekoliko različitih protokola, nude razne funkcionalnosti koje prevazilaze i mnogo više nego što je domen samog SSO servera, tako da bi trebalo da zadovoljavaju sve potrebe većine organizacija koje uvode SSO u svoj sistem. S obzirom na to, može se reći da bi implementacija sopstvenog rješenja trebala biti poslednja opcija, odnosno ukoliko nije moguće pronaći rješenje koje zadovoljava sve potrebe u okviru definisanog budžeta koji je na raspolaganju.

Za razliku od postojećih, rješenje implementirano u okviru rada u trenutnoj verziji nije upotrebljivo u nekom realnom sistemu, ali predstavlja osnovnu za realizaciju konkretnog rješenja koje bi se zasnivalo na SAML protokolu i nudilo osnovne, minimalne funkcionalnosti koje treba da posjeduje jedan SSO server. Naravno, bitno je napomenuti da bi bilo potrebno osigurati da se komunikacija u okviru njega vrši isključivo putem HTTPS protokola, ili da se implementacija doradi tako da se vrši enkripcija/dekripcija poruka koje se razmjenjuju između SP-a i IdP-a.

# LITERATURA

1. <https://searchsecurity.techtarget.com/definition/authentication>
2. Phillip J. Windley, Ph.D., *Understanding Digital Identity Management*
3. <https://searchsecurity.techtarget.com/definition/identity-management-ID-management>
4. <https://auth0.com/blog/what-is-and-how-does-single-sign-on-work/>
5. <https://www.icann.org/news/blog/what-is-authorization-and-access-control>
6. <https://www.owasp.org/index.php/Category:Access_Control#Definition>
7. <https://searchsecurity.techtarget.com/definition/SAML>
8. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard, 15 March 2005
9. <https://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html>
10. <https://wiki.jasig.org/display/CASUM/Introduction>
11. <https://apereo.github.io/cas/5.2.x/index.html>
12. <https://apereo.github.io/cas/5.2.x/protocol/CAS-Protocol.html>
13. <https://calnetweb.berkeley.edu/calnet-technologists/cas/how-cas-works>
14. [https://apereo.github.io/cas/4.2.x/planning/Architecture.html#architecture](https://apereo.github.io/cas/4.2.x/planning/Architecture.html%23architecture)
15. <https://www.shibboleth.net/index/>
16. <https://www.shibboleth.net/products/identity-provider/>
17. [https://wiki.shibboleth.net/confluence/display/CONCEPT#app-switcher](https://wiki.shibboleth.net/confluence/display/CONCEPT%23app-switcher)
18. [https://wiki.shibboleth.net/confluence/display/IDP30/GeneralArchitecture#GeneralArchitecture-OverallArchitecture](https://wiki.shibboleth.net/confluence/display/IDP30/GeneralArchitecture%23GeneralArchitecture-OverallArchitecture)
19. <https://docs.wso2.com/display/IS540/Architecture>
20. <https://www.mutuallyhuman.com/blog/2013/05/09/choosing-an-sso-strategy-saml-vs-oauth2/>
21. <http://wiki.servicenow.com/index.php?title=SAML_2.0_Web_Browser_SSO_Profile#gsc.tab=0>
22. Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard, 15 March 2005

1. https://developers.facebook.com/docs/facebook-login/multiple-providers [↑](#footnote-ref-1)
2. https://developers.google.com/identity/sign-in/web/cross-platform-sign-in [↑](#footnote-ref-2)
3. *Organization for the Advancement of Structured Information Standards* (OASIS) je međunarodna neprofitna organizacija koja radi na razvoju, širenju i usvajanju standarda u oblasti sigurnosti [↑](#footnote-ref-3)
4. https://en.wikipedia.org/wiki/List\_of\_single\_sign-on\_implementations [↑](#footnote-ref-4)
5. Softverska licenca koja označava to da softver može biti besplatno korišten, promijenjen i dijeljen. [↑](#footnote-ref-5)
6. https://apereo.github.io/cas/5.2.x/planning/Architecture.html [↑](#footnote-ref-6)
7. https://shibboleth.net/downloads/identity-provider/ [↑](#footnote-ref-7)
8. http://shibboleth.net/downloads/service-provider/latest/ [↑](#footnote-ref-8)
9. https://wiki.shibboleth.net/confluence/display/IDP30/SystemRequirements [↑](#footnote-ref-9)
10. https://wso2.com/blogs/thesource/2015/08/wso2-ten-years-it-takes-two-to-tango/ [↑](#footnote-ref-10)
11. https://www.w3.org/TR/2006/CR-wsdl20-20060327/wsdl20-z.pdf [↑](#footnote-ref-11)
12. https://wso2.com/products/carbon/release-matrix/ [↑](#footnote-ref-12)
13. https://docs.wso2.com/display/IS510/Identity+Server+Features [↑](#footnote-ref-13)
14. https://docs.wso2.com/display/IS541/Installation+Prerequisites [↑](#footnote-ref-14)
15. https://oauth.net/about/introduction/ [↑](#footnote-ref-15)
16. [↑](#footnote-ref-16)