**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 2](#_Toc511154416)

[3. UPRAVLJANJE IDENTITETIMA 4](#_Toc511154417)

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

[5. AUTORIZACIJA I KONTROLA PRISTUPA 9](#_Toc511154419)

[5.1 Autorizacija 9](#_Toc511154420)

[5.2 Kontrola pristupa 10](#_Toc511154421)

[6. SAML (SECURITY ASSERTION MARKUP LANGUAGE) 12](#_Toc511154422)

[6.1 Principi 12](#_Toc511154423)

[6.2 Kako SAML funkcioniše 13](#_Toc511154424)

[6.2.1 Tvrdnje (Assertions) 13](#_Toc511154425)

[6.2.2 Protokoli (Protocols) 14](#_Toc511154426)

[6.2.3 Povezivanja/uvezivanja (Bindings) 14](#_Toc511154427)

[6.2.4 Profili 14](#_Toc511154428)

[7. XACML (EXTENSIBLE ACCESS CONTROL MARKUP LANGUAGE) 21](#_Toc511154429)

[7.1 Arhitektura 21](#_Toc511154430)

[7.2 Elementi propisa 23](#_Toc511154431)

[7.2.1 Strukturni elementi 23](#_Toc511154432)

[7.2.2 Atributi i kategorije 23](#_Toc511154433)

[7.2.3 Ciljevi 23](#_Toc511154434)

[7.2.4 Uslovi 23](#_Toc511154435)

[7.2.5 Obaveze 23](#_Toc511154436)

[7.3 Primjer propisa 23](#_Toc511154437)

[8. Postojeća SSO rješenja 26](#_Toc511154438)

[8.1 Apereo CAS (Central Authentication Service) 28](#_Toc511154439)

[8.1.1 Osnovne karakteristike 28](#_Toc511154440)

[8.1.2 CAS protokol 28](#_Toc511154441)

[8.1.3 Arhitektura 29](#_Toc511154442)

[8.1.4 Protokoli 30](#_Toc511154443)

[8.2 Shibboleth 32](#_Toc511154444)

[8.2.1 Osnovne karakteristike 32](#_Toc511154445)

[8.2.2 Arhitektura 33](#_Toc511154446)

[8.2.3 Provajder identiteta – IdP 34](#_Toc511154447)

[8.2.4 Servis provajder – SP 35](#_Toc511154448)

[8.3 WSO2 Identity Server 35](#_Toc511154449)

[8.3.1 Osnovne karakteristike 35](#_Toc511154450)

[8.3.2 Arhitektura 36](#_Toc511154451)

[8.3.3 Servis provajder – SP 38](#_Toc511154452)

[8.3.4 Protokoli 38](#_Toc511154453)

[9. IZBOR STRATEGIJE ZA REALIZACIJU SSO SERVERA 39](#_Toc511154454)

[9.1 SAML 2.0 39](#_Toc511154455)

[9.1.1 SAML ograničenja 39](#_Toc511154456)

[9.1.2 Zaobilazno rješenje SAML HTTP POST uvezivanja 39](#_Toc511154457)

[9.2 OAuth 2.0 39](#_Toc511154458)

[9.2.1 OAuth2 nedostaci 39](#_Toc511154459)

[9.3 Rezime – SAML vs. OAuth2 39](#_Toc511154460)

[10. IMPLEMENTACIJA SSO SERVERA 39](#_Toc511154461)

[10.1 SAML 2.0 Web Browser SSO profil 41](#_Toc511154462)

[10.2 Arhitektura i način funkcionisanja 47](#_Toc511154463)

[11. UPOREDNA ANALIZA SSO RJEŠENJA 50](#_Toc511154464)

[12. ZAKLJUČAK 52](#_Toc511154465)

[13. LITERATURA 53](#_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, zahtijeva 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 je doveo 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.

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 2, 3 i 5, respektivno. Sam opis SSO procesa, kao i odgovarajućih sistema koji učestvuju u njemu, nalazi u 4. poglavlju. U poglavljima 6, 7 i 8 izvršena je analiza SAML, OAuth i XACML standarda/protokola koji se mogu koristiti pri razmijeni autentikacionih i autorizacionih poruka između sistema uključenih u SSO proces. Poglavlje 9 sadrži pregled i analizu postojećih SSO rješenja. Na kraju, u 10. poglavlju su dati detalji implementacije SSO servera koji je praktični dio ovog rada, kao i njegova uporedna analiza 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 sertifikatima 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.

## Autentikacioni faktori

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):

* 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 posjed svim faktorima koji su potrebni za autentikaciju.

### 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 predstavlja. Svaki korisnik se pri prvoj upotrebi sistema registruje (ili ga je već neko registrovao, npr. administrator sistema) 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. [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 što 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 administratore sistema 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,
* jednostavnija dodjela/uskraćivanje prava i privilegija pristupa različitim dijelovima informacionog sistema u okviru organizacije,
* kreiranje naloga od strane zaposlenih koji nisu stručni i odgovorni za to,
* 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 mehanizam za autentikaciju korisnika koji omogućava da se on upotrebom jednog para korisničkog imena i lozinke prijavi na više različitih aplikacija (kojima ima prava pristupa) povezanih na SSO server, bez potrebe da ih unosi svaki put. Unosi ih samo pri autentikaciji na strani SSO servera, koja se vrši nakon što prvi put pristupi nekoj od aplikacija koje su na njega povezane. Nakon toga, prilikom pristupa svim ostalim aplikacijama, korisnik će biti automatski prijavljen posredstvom servera, sve dok u okviru njega posjeduje važeću sesiju u okviru koje je autentikovan. Time se rješava i problem postojanja pojedinačnih pristupnih podataka za svaku od aplikacija, što u slučaju njihovog velikog broja može predstavljati veliki problem za korisnika koji ih mora pamtiti. Takođe, postojanje centralizovanog sistema, kao i skladišta u kom se čuvaju svi pristupni podaci i dozvole korisnika, znatno olakšava posao njihovog uređivanja.

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

Aplikacija1

**Internet preglednik (skladište kolačića)**



Autentikacija korisnika

Pristup

Pristup

**Korisnik**

Čuvanje kolačića

Čuvanje kolačića

**Zaštićeni resurs**

**Forma za prijavu**

Aplikacija2

Autentikacija korisnika

**Zaštićeni resurs**

**Forma za prijavu**

Slika 4.1 – Rješenje bez SSO servera

Sa prethodne slike je moguće vidjeti da se korisnik prilikom pristupa svakoj od aplikacija mora ponovno prijaviti, koristeći odgovarajuće pristupne podatke. To može oduzeti dosta vremena i predstavljati dodatnu distrakciju za korisnika, naročito ako se radi o poslovnom okruženju gdje korisnik svakodnevno koristi veliki broj različitih aplikacija i alata za obavljanje svojih poslova. Dakle, cilj je olakšati proces prijave korisnika u različite aplikacije, bez njegove direktne interakcije. Na najjednostavniji način, to bi se moglo postići dijeljenjem kolačića (*cookies*) koji se u okviru preglednika kreiraju na zahtjev servera aplikacija koja ih šalje, a služe za čuvanje podataka o korisniku koji su relevantni za nju. Uobičajeno je da nakon uspješne prijave korisnika, aplikacija kreira sesijski kolačić[[3]](#footnote-3) koji sadrži određene informacije o njemu, ili identifikator sesije na strani servera aplikacije, koja takođe sadrži informacije o korisniku. Kolačić se od strane preglednika automatski šalje uz svaki naredni zahtjev ka toj aplikaciji koja ga je i kreirala. Na osnovu toga, aplikacija zna o kom korisniku se radi, ili ukoliko kolačić predstavlja identifikator sesije, koja je njegova odgovarajuća sesija na strani servera u okviru koje se nalaze podaci o njemu. Ukoliko bi kolačić sadržao podatak koji identifikuje korisnika u okviru jedne aplikacije, npr. korisničko ime ili email adresu, druga aplikacija bi teoretski to mogla pročitati iz kolačića, i na osnovu te informacije mogla bi automatski identifikovati i prijaviti korisnika. Međutim, iz sigurnosne mjere, odnosno pravila u okviru preglednika poznatog kao „pravilo istog porijekla“ *(same-origin-policy*), pristup kolačićima je dozvoljen samo onim aplikacijama koje su izdale zahtjev za njihovo kreiranje. Prema tome, aplikacija2 ne može pristupiti kolačiću kreiranom od strane aplikacije1, kao što je i predstavljeno na **Slici 4.2**.

**Internet preglednik (skladište kolačića)**



Autentikacija korisnika

Pristup

Pristup

**Korisnik**

Čuvanje kolačića

Čitanje kolačića aplikacije1

**Zaštićeni resurs**

**Forma za prijavu**

Aplikacija2

Autentikacija korisnika upotrebom kolačića aplikacije1

**Zaštićeni resurs**

**Komponenta za autentikaciju**

Aplikacija1

Slika 4.2 - Pravilo istog porijekla

Gore navedeni problem, odnosno problem dijeljenja autentikacionih podataka o korisniku, može se riješiti upotrebom SSO servera kao centralnog autentikacionog servera u okviru kog je on autentikovan. Kao takav, on sadrži određene informacije o korisniku, na osnovu kojih se putem njega može izvršiti prijava korisnika i na sve ostale aplikacije koje su na njega povezane. prikazana na narednoj **Slici 4.4**aplikaciju1koju kao identifikator odgovarajuće sesije sesijskiu aplikaciju i kreiranje odgovarajuće sesije aplikacije1aplikaciju2jeuaplikaciju(aplikacija2) aplikacije2nje

**Internet preglednik (skladište kolačića)**



Preusmjeravanje na

Autentikacija upotrebom tokena

Pristup

Pristup

**Korisnik**

Čuvanje kolačića

za aplikaciju1

**Autentikacioni / SSO server**

Čuvanje kolačića

za aplikaciju2

Autentikacija upotrebom tokena

Preusmjeravanje i slanje tokena

Čuvanje kolačića za SSO server

Preusmijeravanje na

Identifikacija korisnika ili očitavanje postojećeg kolačića koji sadrži identifikator sesije

Preusmjeravanje i slanje tokena

Aplikacija1

**Zaštićeni resurs**

**Komponenta za autentikaciju**

Aplikacija2

**Zaštićeni resurs**

**Komponenta za autentikaciju**

Slika 4.4 - Uobičajeni SSO scenario

Pored centralizovanog SSO sistema koji je naveden u prethodnom dijelu, SSO može postojati i u decentralizovanoj konfiguraciji. To znači da ne postoji samo jedan, odnosno centralni, autentikacioni server koji je predefinisan u okviru aplikacije kao posrednik za autentikaciju korisnika, nego ona može vršiti posredstvom bilo kog povjerljivog autentikacionog servera koji podržava određeni standard/protokol. Jedan od najpoznatijih protokola za decentralizovanu autentikaciju je OpenID, koji korisniku omogućava da on sam odabere svog davaoca identiteta u okviru kog je registrovan, i posredstvom kog će se vršiti potvrđivanje njegovog identiteta kada davalac usluge to zatraži, nakon što korisnik unese svoj OpenID URL koji predstavlja njegov identifikator.

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 koje posredstvom njega vrše autentikaciju. Korisnicima se posle registracije dodjeljuju odgovarajuća prava pristupa određenim aplikacijama, nakon čega su u stanju da pristupe tim aplikacijama kada prvi unesu svoje pristupne podatke, što eliminiše višestruke unose i prijavu. SSO takođe smanjuje vrijeme i cijenu uređivanja velikog broja korisničkih naloga koje administratori sistema moraju da održavaju, a koristan je i za dokumentovanje pristupa te praćenje aktivnosti korisnika.

Potencijalni sigurnosni problem upotrebe SSO sistema za autentikaciju korisnika, u odnosu na postojanje više nezavisnih sistema u okviru kojih korisnik mora pojedinačno da se prijavljuje, ogleda se u znatno većoj šteti koja nastaje ukoliko napadač dođe do njegovih pristupnih podataka. U tom slučaju, napadač bi automatski imao pristup svim aplikacijama koje korisnika autentikuju posredstvom tog SSO sistema.

# AUTORIZACIJA I KONTROLA PRISTUPA

## Autorizacija

Autorizacija korisnika je proces utvrđivanja prava koje on posjeduje u okviru sistema, nakon što se prethodno autentikovao, odnosno potvrdio svoj identitet. Sistem na osnovu identiteta korisnika pronalazi neke dodatne informacije o njemu, kao npr. da li pripada nekoj određenoj grupi korisnika, koja je njegova uloga u okviru sistema itd., te na osnovu toga sprovodi pravila, odnosno određuje kojim resursima on može pristupiti i koje akcije su mu dozvoljene. Kako bi sistem bio u stanju da sprovodi ta pravila, on osim informacija o samom korisniku, mora imati definisane propise koji određuju prava i dozvole koje npr. određene grupe i tipovi korisnika imaju ka resursima ili funkcionalnostima sistema. Uobičajeno je da se korisniku nakon registracije, bilo automatski ili od strane administratora sistema, dodjeljuje pripadnost odgovarajućoj grupi (grupama) i/ili odgovarajuća uloga, čime on stiče određena prava pristupa koja data grupa/uloga ima ka određenim resursima i funkcionalnostima sistema.

Autorizacija je bitna iz razloga što većina sistema, bilo poslovnih ili namijenjenih javnosti, posjeduje određene dijelove/ogranke koji ne bi trebali biti dostupni svima. Zavisno od veličine sistema, on može posjedovati određeni broj ogranaka koji sadrže osjetljive resurse i podatke, i kojima bi trebali moći pristupiti samo korisnici koji pripadaju tom ogranku, odnosno određenoj grupi. Nakon što se u okviru sistema/organizacije definišu grupe korisnika, kao npr. u okviru banke - klijenti, menadžment, finansije, tehničko osoblje, računovodstvo itd., potrebno je definisati kojim resursima i funkcionalnostima njihovi članovi imaju pristup i koje akcije oni mogu vršiti nad njima. Npr. u slučaju online sistema banke, korisnik iz grupe klijenata ima pristup pregledu isključivo svojih transakcija kao i mogućnost obavljanja novih, dok korisnik iz grupe finansija ima samo mogućnost pregleda transakcija svojih klijenata. Kako bi gore navedeno bilo moguće ostvariti i definisati, sigurnosni sistemi određenog tipa kao što je npr. IAM (*Identity and Access Management*) posjeduju funkcionalnost koja administratorima nudi mogućnost kreiranja/uređivanja autorizacionih pravila, uključujući sve ono što je potrebno za to, odnosno kreiranje/uređivanje korisničkih naloga, grupa, uloga, privilegija, dozvola itd.

Pored gore navedenog načina definisanja autorizacionih propisa, koji predstavlja njihovo eksplicitno kreiranja od strane administratora sistema upotrebom odgovarajućih alata, oni mogu biti i implicitno kreirani od strane korisnika sistema, kao što je to npr. slučaj sa socijalnim mrežama. Nekada ranije su te mogućnosti bile prilično ograničene i svodile su se na samo nekoliko pravila koje korisnik može definisati. Međutim, u današnje vrijeme je povjerljivost informacija koje korisnik dijeli o sebi veoma bitna tema, pa on kroz određena podešavanja u okviru sistema na raspolaganju ima dosta veći broj sigurnosnih propisa koje može implicitno podesiti kako bi se zaštitio od neželjenog pristupa, odnosno kako bi informacije o sebi podijelio isključivo sa ciljnom grupom korisnika i to na određeni način, ili im potpuno onemogućio pristup. [5] [6]

## Kontrola pristupa

Uopšteno govoreći, kontrola pristupa obuhvata procese autentikacije i autorizacije. Međutim, preciznije rečeno, kontrola pristupa se odnosi na odobrenje pristupa već autentikovanom korisniku ka nekom konkretnom resursu, pri čemu sistem tu odluku donosi na osnovu dozvola koje su mu dodijeljene kao i na osnovu atributa samog resursa. Pored toga, može se oslanjati i na neke dodatne informacije vezane za uslove/okruženje u kojim se sprovodi, kao npr. lokaciju korisnika, njegovu IP/MAC adresu, trenutni datum i vrijeme itd. 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 kojim objektom, 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 skladu 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 sprečava u zavisnosti od skupa pravila koje je definisao administrator sistema. Pristupna pravila se čuvaju u odgovarajućoj listi, specifičnoj za taj objekat.

# 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 davaoca identiteta i davaoca usluge, 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[[4]](#footnote-4) 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,
* davalac 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,
* davalac usluge (*Service Provider* - SP)- sistem odnosno servis čijim resursima korisnik želi pristupiti, i koji obavlja komunikaciju sa davaocem identiteta u cilju sprovođenja kontrole pristupa.

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

Davalac identiteta, prije nego što dostavi tvrdnju o identitetu korisnika davaoca usluge, 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 davaoca identiteta prema davaoca usluge. Po SAML specifikaciji, jedan davalac identiteta može slati tvrdnje različitim davaocima usluge. Slično, jedan davalac usluge može se oslanjati i vjerovati tvrdnjama dobijenim od strane više nezavisnih davalaca identiteta.

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

## Način funkcionisanja SAML-a

SAML je 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 davaoca 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 davaoca identiteta prema davaoca usluge. Tvrdnje sadrže izjave (*statements*) koje davalac usluge 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 davaoca usluge da se korisnik autentikovao na strani davaoca 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 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*). Davalac usluge upit šalje direktno davaocu identiteta putem sigurnog kanala (npr. upotrebom HTTPS protokola), čime se obezbjeđuje da će podaci biti zaštićeni od neovlašćene upotrebe. 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 iz razloga što se atributske izjave često koriste za pružanje podataka o korisniku na osnovu kojih strana koja je izdala upit može nadalje samostalno donositi odluke o njegovim pravima pristupa. Npr. autorizacijski upit koji kao rezultat vraća osnovne informacije o korisniku zajedno sa ulogama koje on posjeduje u okviru sistema koji je izdao taj upit, na osnovu čega za tog korisnika kreira odgovarajući sigurnosni kontekst.

### 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 najčešće 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.*

## 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 davaocem usluge
* Davalac usluge (*Service Provider* - SP) – *web* server čijim uslugama i resursima korisnik pokušava pristupiti putem klijenta
* Davalac 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 6.1**.

**Davalac usluge / SP**

**Klijent**

**Davalac 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 6.1 - Najčešći scenario upotrebe SAML-a

Primjer jednog slučaja upotrebe, koji opisuje gornji dijagram: [20]

* A – korisnik otvara *web* preglednik i posjećuje davaoca usluge (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 davalac 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.

### SAML ograničenja

SAML podržava koncept uvezivanja. To je u suštini sredstvo kojim se vrši „preusmjeravanje“ korisnika između SP-a i IdP-a u cilju razmjene SAML autentikacionih poruka, kako bi se izvršila njegova autentikacija. Npr. u koraku **D**, IdP „preusmjerava“ korisnika nazad na SP kako bi mu (posredstvom korisnika) poslao SAML token kao odgovor na autentikacioni zahtijev. Dva bitna tipa uvezivanja su HTTP preusmjeravanje (*HTTP Redirect (GET)*) i HTTP POST, uvedena u SAML 2.0 specifikaciji.

*HTTP Redirect (GET)* uvezivanje predstavlja korištenje 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** u kom se vrši slanje informacija o korisniku (u obliku SAML tokena) od strane IdP-a ka SP-u (posredstvom korisnika), korist upravo HTTP POST uvezivanje.To podrazumijeva da IdP korisniku usluži HTML stranicu sa formom koja sadrži SAML token, ima akciju koja upućuje nazad na SP (na pristupnu tačku za obradu SAML odgovora), te koja kao metodu za slanje koristi „POST“. Forma može sadržati dugmić na koji korisnik treba kliknuti kako bi je poslao, ili se može koristiti JavaScript funkcija za automatizovano slanje, što će u oba slučaja rezultovati slanjem HTTP POST zahtjeva od strane korisnika, na lokaciju definisanu kao akcija forme. Međutim, to predstavlja problem u slučaju kada klijent nije *web*-bazirana aplikacija već nativna aplikacija (npr. mobilna aplikacija), iz razloga što ona kao takva ne može pristupiti sadržaju tijela HTTP POST zahtijeva, odnosno ne može iz njega izvući SAML token neophodan za autentikaciju. Recimo da je korisnik instalirao odgovarajuću aplikaciju za iPhone koja za prijavu koristi SSO upotrebom SAML-a. Nakon pokretanja aplikacije koja sada predstavlja davaoca usluge (SP), potrebna je autentikacija posredstvom davaoca identiteta (IdP), što će rezultovati time da aplikacija preusmjerava korisnika na stranicu za prijavu u okviru IdP-a, koja se otvara u okviru internet preglednika na telefonu. Nakon što se korisnik autentikovao, IdP ga treba nekako poslati zajedno sa SAML tokenom nazad do aplikacije, odnosno kao što je gore objašnjeno usluživanjem stranice sa odgovarajućom formom. U ovom slučaju, s obzirom na to da korisnika treba vratiti na mobilnu aplikaciju, IdP će generisati formu koja kao akciju ima prilagođeni URL za pokretanje te aplikacije (npr. „my-app://authenticate“ – dostupan iz inicijalnog zahtjeva), dok kao metodu za slanje i dalje koristi POST jer je u pitanju HTTP POST uvezivanje. S obzirom na gore navedeno, aplikacija će se pokrenuti ali korisnik neće biti uspješno prijavljen, zbog čega je to upravo jedan od nedostataka ovog uvezivanja. [20]

Međutim, ograničenje HTTP POST uvezivanja za mobilne aplikacije može biti riješeno na više načina. 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 iz tijela HTTP POST zahtjeva. 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/?SAMLResponse=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.

# OAuth 2.0

Za razliku od SAML-a, OAuth 2.0 (nadalje OAuth2) je specifikacija koja je bila gotovo u potpunosti odbačena krajem 2012. godine, međutim vremenom je ipak zaživjela pa je danas koriste i Facebook, Google, Twitter te mnogi drugih. Objavljena je 2010. godine kao zamjena za inicijalnu verziju OAuth-a, koja je nastala 2007. godine kao autentikaciona metoda za Twitter API.[[5]](#footnote-5) Kao i prva verzija OAuth-a, OAuth2 omogućava internet korisnicima da odobre *web* stranicama ili aplikacijama pristup svojim informacijama na drugim *web* stranicama, bez dijeljenja lozinke. Ima prednost u odnosu na SAML u tome što je to novija specifikacija i što uzima u obzir napredak i razvoj svijeta tehnologije.

Iako kao zadatak rada nije navedena analiza OAuth2-a, on je u okviru ove glave ukratko predstavljen, iz razloga što predstavlja dobru alternativu SAML-u i pokriva neke njegove nedostatke, kao što je standardizovana podrška za nativne mobilne aplikacije koje su danas izuzetno bitne i rasprostranjene. SAML i OAuth2 koriste slične pojmove za slične koncepte. Zbog poređenja, pored 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, 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:

Slika 7.1 - OAuth2 scenario upotrebe

**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**

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.

Prednost OAuth2 tokova dešavanja u odnosu na SAML tokove, je to što se slanje pristupnog tokena od autorizacionog servera ka serveru sa resursima (posredstvom klijenta) vrši upotrebom HTTP preusmjeravanja (*HTTP Redirect (GET)*), pri čemu se on prosljeđuje kao parametar zahtjeva. To predstavlja prednost u odnosu na tok dešavanja naveden kod SAML-a, iz razloga što se na taj način izbjegava upotreba HTTP POST uvezivanja koje sa sobom nosi problem pristupa tijelu HTTP POST zahtjeva od strane nativnih (npr. mobilnih) aplikacija, čime se olakšava njihova integracija sa SSO serverom. 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]

* + 1. *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 specifikacije 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]

* 1. Poređenje sa SAML-om

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. Takođe, OAuth2 nudi jednostavnije i više standarizovano rješenje za podršku nativnih aplikacija, jer se njegovom upotrebom izbjegavaju zaobilazna rješenja, za razliku od SAML-a. Prema tome, oba pristupa imaju svoje prednosti u odnosu na određene potrebe i zahtjeve, i predstavljaju dobra rješenja za implementaciju SSO sistema.

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 10. poglavlju, korišten je SAML standard odnosno specifikacija.

# 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.

## Arhitektura

Arhitektura XACML-a i primjer odvijanja procesa autorizacije prikazani su na **Slici 8.1**. Pojmovi koji se koriste pri opisu ovakvih procesa su sledeći:

* PAP (*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. Učitavanje resursa

2. Provjera da li korisnik ima pristup resursu?

3. Procijena zahtijeva na osnovu propisa

Upravljanje

propisima

3.a..Pribavljanje

dodatnih atributa

4. Odobravanje pristupa ka resursu

5. Učitavanje resursa

**PRP**

Pribavljanje

propisa

Slika 8.1 – 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 zahtjev 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.

### Atributi i kategorije

Propisi, setovi propisa, pravila i zahtijevi odnose se na entitete sistema, odnosno subjekte, resurse, okruženja i akcije. [9]

* 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

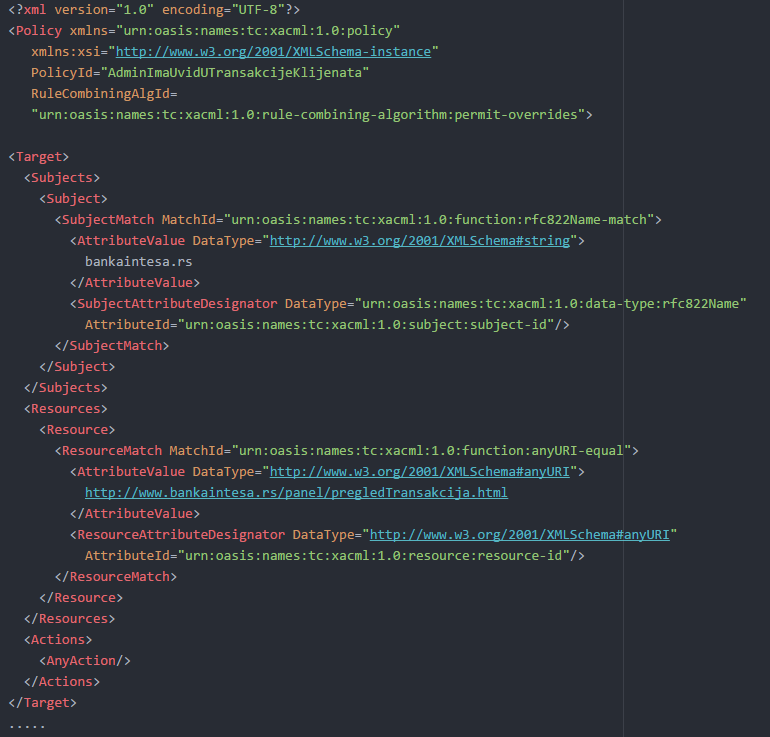
### Ciljevi, uslovi i obaveze

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 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]

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]

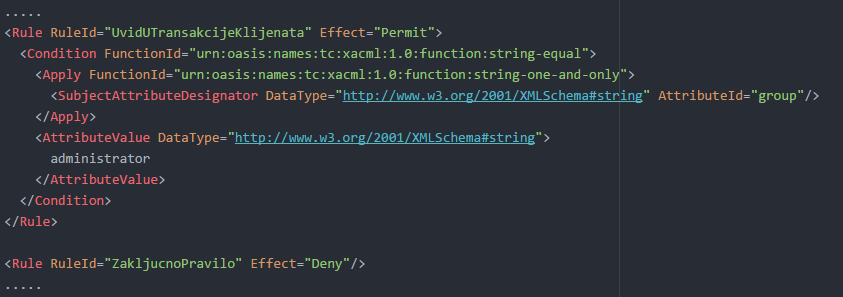
U narednom dijelu su kao sastavni elementi primjera jednog propisa, pojedinačno navedeni i objašnjeni po jedan cilj, pravilo (sa uslovom) i obaveza. Za primjer je uzet propis koji kaže da korisnici čija se email adresa završava sa „@bankaintesa.rs“ i koji pripadaju grupi administratora, imaju pravo na uvid u sve transakcije klijenata banke, dostupne na adresi „http://www.bankaintesa.rs/panel/pregledTransakcija.html“. Ukoliko email adresa korisnika nije sa datog domena ili on ne pripada grupi administratora, pristup će mu biti onemogućen (uz pretpostavku da je sistem podešen tako da je pristup podrazumijevano onemogućen). Pored toga, biće evidentirani svi odbijeni zahtjevi na koje je dati propis bio primjenjiv.



Slika 8.2 - primjer cilja

Na prethodnoj **Slici 8.2** prikazan je cilj propisa iz navedenog primjera. Kao što je moguće vidjeti, da bi propis i pravila definisana u okviru njega uopšte bili primijenjeni na određeni zahtjev, on mora biti upućen ka stranici za uvid u sve transakcije klijenata, i to od strane korisnika čija se email adresa završava sa „@bankaintesa.rs“.

Naredna **Slika 8.3** predstavlja pravilo definisano u okviru propisa, koje sadrži uslov da pravo pristupa u uvid svih transakcija klijenata banke imaju samo korisnici koji pripadaju grupi administratora. U suprotnom, uslov pravila „UvidUTransakcijeKlijenata“ neće biti zadovoljen i neće biti postignut efekt odobrenja pristupa (*Permit*). Nakon toga će biti primijenjeno zaključno, tzv. „propadajuće“ (*fall-through*) pravilo, koje kao što i sam propis nalaže, osigurava da u slučaju nezadovoljavanja prethodnih pravila pristup bude odbijen (*Deny*).



Slika 8.3 - primjer pravila sa uslovom

Poslednji element propisa predstavlja obaveza koja nalaže da se svi odbijeni pokušaji pristupa (oni koji su rezultovali efektom *Deny*) evidentiraju.



Slika 8.4 - primjer obaveze

# 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 9.1** dat je pregled nekih od postojećih rješenja SSO sistema.

Tabela 9.1 - Neka od postojećih SSO rješenja[[6]](#footnote-6)

| **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 navedenih rješenja, izabranih 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[[7]](#footnote-7)* rješenja. Na osnovu gore navedenog, za pregled i analizu su izabrana tri sljedeća:

* 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

Neke od osnovnih karakteristika ovog SSO rješenja su sledeće: [11]

* 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.

### Arhitektura

CAS server i klijent su dvije osnovne komponente arhitekture CAS sistema, predstavljene na **Slici 9.1**, i mogu da vrše komunikaciju različitim vrstama podržanih protokola, navedenih u tački 8.1.3. [14]



Korisnici

**CAS klijenti**

**CAS protokol**

Apache aplikacija

Java aplikacija

Google aplikacija

**SAML protokol**

**OAuth protokol**

**CAS server**

Spring MVC/Webflow

Upravljanje tiketima

Autentikacija

LDAP autentikator

DB autentikator

SPNEGO autentikator

LDAP

DB

Active Directory

Slika 9.1 – Arhitektura CAS sistema

**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 autentikaciju 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 (Apache Httpd Server, Java, .NET, PHP, Perl, Pyton, Ruby) i aplikacije (Canvas, Atlassian Confluence/Jira, Drupal, Liferay, UPortal...) kako bi one mogle da komuniciraju sa CAS serverom.

### 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

### 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

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 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]

Preusmjeravanje klijenta od strane aplikacije ka CAS serveru kako bi se izvršila autentikacija, obično rezultuje slanjem HTTP GET zahtjeva (ili nekog drugog zavisno od tipa inicijalnog zahtjeva koji je klijent uputio aplikaciji prije nego što ga je ona preusmjerila na server) ka pristupnoj tački u okviru servera, namijenjenoj za autentikaciju korisnika (npr. */login*). Kao opcioni parametri tog zahtjeva mogu se proslijediti *service* i *method* parametri. Taj zahtjev će, ukoliko korisnik još nije prijavljen u okviru servera, rezultovati prikazom forme za prijavu. Nakon što korisnik formu popuni ispravnim pristupnim podacima i pošalje je serveru (obično putem HTTP POST zahtjeva), on će biti uspješno prijavljen u okviru njega. Zavisno od parametara proslijeđenih u okviru inicijalnog zahtjeva za prijavu na server (na */login*) i njihovih vrijednosti, on će nakon uspješne prijave korisnika, na određeni način vratiti „odgovor“ u smislu preusmjeravanja klijenta/korisnika nazad na aplikaciju (ukoliko je *service* parametar koji obično predstavlja putanju do aplikacije dostupan) zajedno sa servisnim tiketom koji mu se dodjeljuje. Ukoliko *service* parametar nije dostupan, server će prikazati poruku da je korisnik uspješno prijavljen. Što se tiče *method* parametra, ukoliko nije dostupan, server će uobičajeno izvršiti preusmjeravanje korisnika na putanju proslijeđenu kao vrijednosti *service* parametra (dodajući i servisni tiket kao parametar). Međutim, za aplikacije koje zahtijevaju „vraćanje odgovora“ (servisnog tiketa) na *service* putanju upotrebom HTTP POST zahtjeva (u okviru njegovo tijela), server će korisniku uslužiti stranicu sa formom koja sadrži tiket, te kao akciju forme ima putanju proslijeđenu u okviru *service* parametra, a kao metodu za slanje ima podešenu vrijednost „POST“. Ova stranica će obično sadržati i poruku za korisnika da mora poslati/potvrditi formu kako bi se vratio nazad na željenu aplikaciju (CAS servis), ili će se to slanje forme izvršiti automatski nakon određenog vremena upotrebom JavaScript-a.

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]

## Shibboleth

Shibboleth je još jedna od *Open-Source* implementacija SSO sistema, namijenjena za upotrebu unutar jedne ili između više različitih organizacija. To je softverski paket koji kao dvije osnovne komponente, tj. aplikacije, sadrži davalac identiteta (IdP) koji obezbjeđuje sigurnosne informacije o korisnicima, te davalac usluge (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 davaocima usluge da otkriju tj. pronađu davaoce 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 davaoca identiteta, koji zatim davaoca usluge 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 davaoca identiteta je IdP 3.3.2[[8]](#footnote-9), dok je poslednja verzija davaoca usluge SP 2.6.1[[9]](#footnote-10).

### Osnovne karakteristike

Neke od osnovnih karakteristika Shibboleth sistema su sledeće: [16]

* 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 (davaocima usluge)
* odlična skalabilnost – jedna instanca može obrađivati milione autentikacionih zahtijeva dnevno i može komunicirati sa stotinama davaoca usluge
* podrška za SAML 1.1, SAML 2.0 i CAS 2 SSO protokol
* podrška za bilo koju implementaciju davaoca usluge 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

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 davaoca identiteta i davaoca usluge, u svrhu postizanja Single Sign-On funkcionalnosti na mreži. Osnovne komponente Shibboleth softverskog paketa, pa tako i arhitekture, su davalac identiteta (IdP), davalac usluge (SP) i servis za otkrivanje davaoca identiteta (DS). 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.



**Korisnik**

*Internet preglednik*

**Davalac usluge (SP)**

**Davalac identiteta (IdP)**

Slika 9.2 – Osnovna interakcija [17]

Na prethodnoj **Slici 9.2**, prikazana je na visokom nivou interakcija koja se odvija između korisnika (posredstvom internet preglednika), davaoca identiteta i davaoca usluge: [17]

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

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.

***Request Dispatcher***

Profile Handler

Akcija 1

Akcija 2

Akcija 3

Akcija 4

***Authentication Engine***

***Attribute Resolver***

***Attribute Filter Engine***

***Attribute Consent Engine***

***Relying Party Config Manager***

Davalac identiteta (IdP)

**Eksterni sistemi (npr. LDAP)**

**HTTP zahtjev**

**API poziv**

Slika 9.3 – Arhitektura IdP-a

Sama arhitektura IdP-a, prikazana na **Slici 9.3**, 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. 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.

Shibboleth davalac usluge (SP) podržava nekoliko različitih platformi i sistema (Linux, Mac OS X, Windows, Solaris, Java Servleti), čime omogućava da se *web* aplikacije, napisane upotrebom bilo kog programskog jezika ili okruženja (*framework*), brzo i lako integrišu u SSO sistem, bez potrebe za velikim promjenama unutar koda aplikacije.

Davalac usluge, 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 davaocu identiteta
* Obrađivanje SAML odgovora i izvlačenje informacija o korisniku
* Sprovođenje lokalnih propisa i prikupljanje dodatnih podataka

Osim upotrebe u kombinaciji sa Shibboleth davaocem identiteta, ovaj davalac usluge može se koristiti i u kombinaciji sa bilo kojom drugom implementacijom IdP-a koja je u skladu sa SAML standardom.

## 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-13) koji je bio istraživač u IBM-u, gdje je bio jedan od osnivača platforme *web* servisa[[11]](#footnote-14). WSO2 IS je samo jedan od nekoliko proizvoda ove kompanije, i njegova poslednja verzija je WSO2 IS 5.4.1[[12]](#footnote-15).

### Osnovne karakteristike

U narednim tačkama su navedene neke od osnovnih karakteristika WSO2 IS-a: [[13]](#footnote-16)

* 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

### 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 uloge, 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 davalac identiteta za nezavisne aplikacije koje imaju svoje posebne grupe korisnika, što i jeste teme ove glave. , aš

**Davaoci usluge**

**SP2**



**SP1**



**ExtApp21**



**ExtApp1**



**Eksterne aplikacije**

**Lokalni autentikatori**

**Korisničko/Lozinka**

**IWA**

**Eksterni autentikatori**

**Open ID Connect**

**OAuth**

**SAML SSO**

**Passive STS**

**Facebook**

**Google**

**Ulazni autentikatori**

**Procesor zahtjeva**

**Generator odgovora**

**SAML SSO**

**OAuth**

**Open ID Connect**

**Passive STS**

**AUTENTIKACIONO**

***okruženje***

Slika 9.4 – Arhitektura servera [19]

Na **Slici 9.4** 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*)

U narednim tačkama navedene su komponente servera koje se odnose na proces autentikacije:

* davaoci usluge (*Service providers*),
* ulazni autentikatori (*Inbound authenticators*),
* autentikaciono okruženje (*Authentication framework*),
* lokalni autentikatori (*Local authenticators*),
* eksterni autentikatori (*Federated authenticators*),
* eksterne aplikacije (External applications).

**Davaoci usluge** su *web* servisi koji se oslanjaju na davalac 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 davaoca usluge, provjeri ga, te u zajedničkom internom obliku proslijedi autentikacionom okruženju na dalju obradu. Takođe, kada od njega nazad dobije odgovor, treba da ga pretvori u odgovarajući SAML odgovor koji vraća davaoca usluge.

**Autentikaciono okruženje** iz zahtjeva koji je primilo od ulaznog autentikatora izvlači podatak o tome koji davalac usluge 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 okruženja, 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 autentikaciono okruženje koje će odlučiti da li je potrebna dodatna autentikacija, što podrazumijeva da se korisnik mora autentikovati upotrebom još jedne ili više autentikacionih metoda, zavisno od toga koji lokalni autentikatori su podešeni za davaoca usluge na kog se taj zahtjev odnosi. Nakon toga, autentikaciono okruženje će odgovor proslijediti odgovarajućem ulaznom autentikatoru, kako bi on dalje 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 davalac identiteta. Kada autentikaciono okruženje primi zahtjev od ulaznog autentikatora, ono na osnovu zahtjeva i postavki podešenih za dati davalac usluge 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 te integrisati sa serverom sopstveni eksterni autentikator. Takođe, nakon uspješne autentikacije kontrola se prebacuje na autentikaciono okruženje, koje 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 davaoci identiteta u slučaju autentikacije upotrebom nekog od eksternih autentikatora.

Što se tiče davaoca usluge (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, s ciljem njenog povezivanja na WSO2 davalac/server identiteta.

### Protokoli

U okviru SSO funkcionalnosti servera podržani su sledeći protokoli:

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

## Uporedna analiza postojećih rješenja

U narednoj **Tabeli 9.2,** nalazi se uporedna analiza odabranih postojećih rješenja, predstavljenih i analiziranih u okviru ove glave. 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. 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.

Prva grupa karakteristika odnosi se na **podržane protokole**, pa su navedeni oni protokoli koji su podržani u bar jednom od rješenja. Kao što je moguće vidjeti, CAS i WSO2 nude podršku za većinu danas popularnih i zastupljenih protokola – SAML, CAS, OpenID (samo Apereo CAS), OpenID Connect, OAuth, WS-Federation, WS-Trust (samo WSO2), dok to nije slučaj sa Shibboleth rješenjem koje osim svog primarnog protokola - SAML, nudi podršku još samo za CAS protokol. Međutim, manjak te podrške u okviru Shibboleth rješenja ne mora nužno predstavljati veliku manu, npr. ukoliko su zahtjevi takvi da treba implementirati SSO sistem koji koristiti isključivo SAML/CAS protokol, i koji neće biti potrebno proširivati u smislu integracije sa nekim vanjskim sistemima koji možda zahtijevaju upotrebu drugog protokola.

Tabela 9.2 – Uporedna analiza postojećih rješenja

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Karakteristika** | | **Apereo CAS** | **Shibboleth** | **WSO2 Identity Server** |
| **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 | ✓ | ✓ | ✓ |

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. Po ovoj tački su izabrana rješenja izjednačena, jer svako od njih nudi tu mogućnost, koja je u današnje vrijeme veoma bitna i prihvaćena i od velikog broj običnih korisnika, iako oni nisu dio nekog poslovnog sistema koji zahtjeva visok nivo sigurnosti i zaštite informacija. Razlog tome je možda i rasprostranjena upotreba socijalnih mreža, u okviru kojih postoje sigurnosne prijetnje kao što su npr. krađa identiteta i neovlašten pristup ličnim podacima.

**Delegirana autentikacija** se odnosi na mogućnost podešavanja servera da delegira tj. proslijedi autentikaciju korisnika nekom vanjskom davaocu identiteta, kao što su npr. Google, Facebook, Twitter itd.Isto kao u slučaju prethodne, ova mogućnost je takođe podržana u okviru sva tri odabrana rješenja. Takođe bi se moglo reći da je potreba za njom porasla zbog popularnost socijalnih mreža u okviru kojih korisnici već posjeduju nalog, zbog čega se često odlučuju da taj isti nalog upotrijebe i za autentikaciju u okviru neke druge, nezavisne aplikacije koja nudi tu mogućnost. Sa aspekta korisnika je to veoma zgodna opcija, jer na taj način jedan nalog mogu koristiti za veliki broj aplikacija, čime se izbjegava pamćenje dodatnih lozinki.

***Single-Log-Out*** (SLO) je funkcionalnost koju nude sva tri rješenja, a ona korisniku omogućava da se odjavom iz bilo koje od aplikacija, automatski odjavi iz svih ostalih aplikacija u koje se prijavio upotrebom SSO servera. Iako ne zvuči kao nešto što bi trebalo predstavljati bitnu karakteristiku *Single-Sign-On* sistema, za korisnika i njegovu sigurnost je značajna iz razloga što umanjuje šansu da on nesvjesno ostane prijavljen unutar određenih aplikacija na koje se prijavio putem SSO-a, što bi moglo dovesti do neovlaštenog pristupa njegovim „živim“ sesijama, npr. ukoliko je koristio dijeljeni/javni računar i nije zatvorio internet preglednik.

Karakteristika koja je bitna programerima je i **dostupnost klijentske aplikacije ili biblioteke**, jer znatno olakšava integraciju željene aplikacije, odnosno davaoca usluge, u SSO sistem. Ovu karakteristiku ne posjeduje jedino WSO2 rješenje, što znači da povezivanje davaoca usluge na njega iziskuje dodani napor i vrijeme od strane tima koji radi na realizaciji SSO sistema. Što se tiče druga dva rješenja, Shibboleth pored dostupnosti same klijentske aplikacije/biblioteke, pruža podršku za bilo koju implementaciju davaoca usluge (SP-a) koja je je kompatibilna sa SAML 1.1 i 2.0 protokolom, bez potrebe za njegovim prepravkama i upotrebom dodatnih biblioteka/aplikacija. Za razliku od njega, CAS zahtjeva dodatni posao, odnosno da se davalac usluge „upakuje“ unutar CAS klijenta kako bi se posredstvom njega postigla integracija sa CAS serverom, što je još poznato pod nazivom CAS-ifikacija.

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. S obzirom na to da je to bio jedan od kriterijuma za odabir, sva tri rješenja posjeduju ovu karakteristiku.

# 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. davaoca identiteta (IdP-a), te dvije demonstrativne klijentske aplikacije koje predstavljaju davaoce usluge (SP), koji vrše komunikaciju sa IdP-om u cilju njegove autentikacije i autorizacije. Sistem je moguće proširiti sa više klijentskih aplikacija koje se povezuju na server.

Klijentske aplikacije su neophodne za demonstraciju funkcionisanja SSO servera, a služe i kao primjer za podešavanje nekog drugog davaoca usluge koji treba povezati sa njim. Implementirane su kao veoma jednostavne *Spring Boot* aplikacije koje se sastoje od samo tri stranice. 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*). 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.

Komunikacija između klijenata i servera zasniva se na razmjeni 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 konzorcijuma[[14]](#footnote-21) upravo s ciljem da podrži programere koji rade sa SAML-om.

## Arhitektura i način funkcionisanja

i slanje SAMLu drugom dijelu ovog poglavlja objašnjen je (*Single-Log-Out*)

Kao što je ranije navedeno, razmjena SAML poruka vrši se na način definisan *SAML 2.0 Web Browser SSO* profilom*,* tako da je on ujedno predstavljen i pojašnjen kroz naredni primjer. Njegova upotreba podrazumijeva postojanje davaoca identiteta (IdP), davaoca servisa (SP) i korisničkog agenta. Zavisno od načina uvezivanja, kojih SP posjeduje 4 a IdP 3, postoji ukupno 12 različitih scenarija. Za ovu implementaciju je korišten scenario u kom se slanje SAML zahtjeva od SP-a vrši upotrebom *HTTP Redirect (GET)* uvezivanja, odnosno preusmjeravanjem korisnika na IdP. Sa druge strane, IdP vrši slanje odgovora, u smislu prosljeđivanja SAML tokena ka SP-u posredstvom korisnika, upotrebom HTTP POST uvezivanja. To podrazumijeva da IdP korisniku usluži stranicu na kojoj se nalazi forma koja sadrži SAML token, i koja je generisana tako da njeno potvrđivanje rezultuje slanjem HTTP POST zahtjeva od korisnika ka SP-u.

U skladu sa gore navedenim SAML profilom, kreiranje odgovarajućeg sigurnosnog konteksta i sprovođenje kontrole pristupa, nakon što korisnik zatraži pristup zaštićenoj stranici u okviru davaoca usluge, odvija se u narednim koracima predstavljenim i na **Slici 10.2** (korištene su adrese i primjeri poruka realizovane implementacije):

Davalac usluge - SP

(SSO klijent)

Korisnički agent

(web preglednik)

Davalac identiteta - IdP

(SSO servis)

**1**

**Zahtijeva željeni resurs**

**(Pronalaženje IdP-a)**

**Preusmjerava na SSO servis**

**2**

**5**

**6**

**7**

7

**8**

**Zahtijeva servis za konzumiranje odgovora**

**Preusmjerava na željeni resurs**

**Zahtijeva željeni resurs**

**Preusmjerava na željeni resurs**

**Zahtijeva SSO servis**

**(Identifikuje korisnika)**

**Odgovara XHTML formom**

**3**

**4**

Slika 10.2 – Web Browser SSO profil

1. **Zahtijevanje željenog (zaštićenog) resursa na strani SP1**

Korisnik putem preglednika pristupa početnoj stranici prve SP1 aplikacije (davaoca usluge) – *localhost:8081/sp1*, pri čemu mu se otvara stranica sa objašnjenjem i linkom ka zaštićenoj stranici.

http://localhost:8081/sp1/protectedResource

vrši provjeru prava pristupa željenom resursu. Ako već postoji odgovarajući sigurnosni kontekst, preskaču se koraci 2-7, u suprotnom prelazi se na naredni korak.

1. **Preusmjeravanje na IdP SSO servis**

Aplikacija SP1 generiše odgovarajući SAML zahtjev (u obliku *AuthnRequest* elementa), kao i *RelayState* vrijednost koja generalno predstavlja referencu ka informacijama koje se čuvaju na strani davaoca usluge. Te vrijednosti služe za generisanje parametara koji se dodaju na adresu na koju se vrši HTTP preusmjeravanje korisnika, odnosno slanje SAML poruke (zahtjeva) na SSO server posredstvom korisnika, što u suštini predstavlja upotrebu SAML *HTTP Redirect (GET)* uvezivanja.

302 Redirect

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

U ovoj implementaciji, *RelayState* parametar služi za čuvanje reference ka informaciji (u okviru SP-a) koja predstavlja putanju do željene stranice, kako bi se korisnik nakon povratka sa SSO servera mogao automatski preusmjeriti na nju. *SAMLRequest* parametar predstavlja kompresovanu, *base64* enkodiranu te URL enkodiranu vrijednost SAML zahtjeva, odnosno *<samlp:AuthnRequest>* elementa prikazanog na **Slici 10.3**. Pored toga, zahtjev može biti i potpisan, ali u ovoj implementaciji zbog jednostavnosti to nije slučaj. Međutim, ukoliko bi SAML zahtjev bio potpisan od strane SP-a, to bi se vršilo upotrebom privatnog dijela ključa iz para ključeva koje on generiše, dok bi potpis validirao primalac, odnosno IdP, upotrebom javnog dijela ključa generisanog od strane SP-a.



Slika 10.3 – primjer SAML AuthnRequest-a

Konačno, vrši se preusmjeravanje korisnika na SSO server, i to na putanju koja je podesiva, a predstavlja putanju do komponente servera koja obrađuje SAML zahtjeve.

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

U skladu sa HTTP standardom, kao rezultat preusmjeravanja iz prethodnog koraka, *web* preglednik korisnika šalje GET zahtjev na putanju IdP-a (*localhost:8080/sso/Redirect*) na kojoj se nalazi SSO servis zadužen za obradu SAML zahtjeva. Međutim, kako je ta putanja zaštićena, da bi korisnik mogao pristupiti navedenom SSO servis u cilju obrade SAML zahtjeva, on mora imati važeći sigurnosni kontekst u okviru IdP-a. Kako to ovdje nije slučaj, korisnik se prvo mora autentikovati u okviru IdP-a, što upravo i jeste naredni korak.

1. **Autentikacija korisnika u okviru IdP-a**

S obzirom na to da je u prethodnom koraku utvrđeno da je neophodna autentikacija korisnika, što se vrši upotrebom jednog ili više mehanizama, u ovom primjeru/implementaciji IdP vraća stranicu na kojoj se nalazi forma za prijavu. Nakon što je korisnik popuni unosom svojih pristupnih podataka, odnosno korisničkog imena i lozinke, te pošalje serveru, izvršiće se njihova provjera u odnosu na bazu podataka.

Ukoliko je autentikacija bila uspješna, korisnik dobija odgovarajući sigurnosni kontekst na strani IdP-a, tako da dobija pristup SSO servisu koji treba da vrši obradu njegovog SAML zahtjeva primljenog u prethodnom koraku.

1. **Obrada SAML zahtjeva**

Nakon uspješne autentikacije korisnika, SSO servis prima i obrađuje njegov zahtjev iz 3. koraka.

Vrijednosti parametara *SAMLRequest* i *RelayState* su one proslijeđene od strane aplikacije SP1 (davaoca usluge) posredstvom korisnika, odnosno njegovim preusmjeravanjem. SSO servis iz zahtjeva izdvaja *SAMLRequest* parametar koji zatim URL-dekodira, *base64*-dekodira te dekompresuje (tim redoslijedom), nakon čega dobija originalnu vrijednost *<samlp:AuthnRequest>* elementa.

Nadalje, sama obrada SAML *AuthnRequest*-a, odvija se na sledeći način. Iz njega se izvlači podatak o davaocu usluge, tj. klijentu koji je uputio zahtjev za autorizaciju korisnika, što je u ovom slučaju aplikacija SP1. S obizirom na to da je poznato ko je prethodno autentikovani korisnik i koji davalac usluge (SP) je izdao taj zahtjev, IdP u bazi pronalazi odgovarajuće uloge (*roles*) koje korisnik posjeduje u okviru njega. One se koriste pri generisanju SAML odgovora (*SAMLResponse*), u okviru kog se upotrebljavaju kao vrijednost *Attribute* elementa pod nazivom *role,* koji se nalazi pod *AttributStatement* elementom. Kasnije, SP1 na osnovu tih uloga koje korisnika posjeduje u okviru njega, može odrediti koja su njegova prava pristupa ka određenim resursima.

1. **Odgovor XHTML formom**

Nakon validiranja zahtjeva i generisanja SAML odgovora, SSO servis korisniku vraća stranicu koja sadrži XHTML formu, kako bi se on zajedno sa SAML odgovorom vratio nazad na SP1 aplikaciju, što u suštini predstavlja SAML *HTTP POST* uvezivanje. Adresa na koju se forma šalje poznata je jer IdP može da očita vrijednost *Issuer* elementa iz inicijalnog *AuthnRequest*-a, odnosno SAML zahtjeva.

<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 (prikazan na **Slici 10.4**), koji u suštini predstavlja odgovor SSO servera na zahtjev koji je primio od davaoca usluge.

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 ta dva zahtjeva.



Slika 10.4 – primjer SAML Response-a

1. **Zahtijevanje servisa za konzumiranje odgovara na SP1 strani**

Nakon podnošenja XHTML forme na strani IdP-a, koja sadrži *SAMLResponse* i *RelayState* atribute*,* preglednik korisnika šalje POST zahtjev servisu za konzumiranje odgovora (*Assertion Consumer Service*) na strani davaoca usluge, koji se u ovom slučaju nalazi na putanji *localhost:8081/sp1/saml*.

POST /saml HTTP/1.1

Host: localhost:8081/sp1

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

SAMLResponse=response&RelayState=token

Nakon prijema tog zahtjeva, vrši se obrada SAML odgovora (*Response* elementa) proslijeđenog u okviru njega kao vrijednost *SAMLResponse* atributa, pri čemu je prvi korak obrade validacija 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.

1. **Preusmjeravanje na željeni resurs na strani SP1**

Nakon uspješne autentikacija i autorizacije izvršene u prethodnom koraku, na osnovu *RelayState* vrijednosti koja je takođe primljena kao dio HTTP POST zahtjeva, 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. **Novi zahtjev za željeni (zaštićeni) resurs na strani SP1**

Kada korisnik pokuša da pristupi nekoj drugoj zaštićenoj stranici u okviru SP1 aplikacije, izvršiće se provjera postojanja odgovarajućeg sigurnosnog konteksta u okviru nje.

http://localhost:8081/sp1/protectedResource2

1. **Odgovor SP1 vraćanjem željenog resursa**

S obzirom na to da je odgovarajući sigurnosni kontekst već kreiran u 7. koraku, SP1 neće slati novi SAML zahtjev SSO serveru, nego će odluku o pravima pristupa korisnika željenoj stranici donijeti na osnovu postojećeg sigurnosnog konteksta i uloga koje korisnik posjeduje u okviru njega.

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 (*Single-Log-out*), 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 odmah vrši komunikaciju sa svim ostalim SP aplikacijama na koje je taj korisnik prijavljen, s ciljem uništavanje sesije i u okviru njih. Komunikacija između IdP-a i SP-ova se vrši direktno, bez posredstva korisnika tj. njegovog preusmjeravanja, tako što SLO servis IdP-a šalje pojedinačne, nezavisne zahtjeve svim ostalim SP aplikacijama (direktna komunikacija između servisa).

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, odnosno gore navedeni način, slanjem pojedinačnih, nezavisnih zahtjeve ka 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.

## Poređenje sa postojećim rješenjima

Naredna **Tabela 10.1,** predstavlja proširenje postojeće **Tabele 9.2** (**Paragraf 9.4**) u kojoj su navedene osnovne karakteristike izabranih postojećih SSO rješenja, tako što se u njoj sada nalaze i karakteristike gore navedenog implementiranog rješenja.

Tabela 10.1- Uporedna analiza implementiranog rješenja sa postojećim

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 | ✓ | ✓ | ✓ | ✓ |

Kao što je moguće vidjeti, implementirano rješenje podržava SAML protokol, isto kao i ostala rješenja predstavljena u 8. poglavlju, međutim ne podržava ni jedan drugi - CAS, OpenID, OpenID Connect, OAuth, WS-Federation i WS-Trust. Prema tome, može se reći da ima najslabiju podršku za protokole, s napomenom da je u pitanju prototip koji treba da predstavi realizaciju SSO sistema zasnovanog na SAML protokolu. Sa druge strane, kao i sva ostala rješenja, podržava SLO funkcionalnost koja korisniku omogućava brzu i laku odjava iz svih aplikacije na koje je prijavljen posredstvom SSO servera. Pored toga, posjeduje čak i jednu karakteristiku koju nema WSO2, a to je dostupnost klijentske biblioteke koja znatno olakšava integraciju željene aplikacije u SSO sistem.

Međutim, funkcionalnosti koje ova implementacija ne nudi za razliku od analiziranih rješenja, jesu višestepena autentikacija kao i delegirana autentikacija koja omogućava da davalac identiteta autentikuje korisnika posredstvom nekog vanjskog sistema. Razlog je to što ova implementacija predstavlja osnovni SSO server sa minimalnim funkcionalnostima koje kao takav mora imati, odnosno sa funkcionalnostima autentikacije i autorizacije. Dodatni razlog je i 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.

Na kraju, što se tiče poslednje karakteristike odnosno tipa licence, u pitanju je naravno *OpenSource* licenca pod kojom su izdana i ostala rješenja, a sam kod je javno dostupan na na adresi <https://github.com/racog3/sso-project>.



# 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. Linda Rosencrance, *Authentication*, <https://searchsecurity.techtarget.com/definition/authentication>
2. Phillip J. Windley, Ph.D., *Understanding Digital Identity Management*, 2003.
3. Linda Rosencrance, Craig Mathias, *Identity Management (ID management)*, <https://searchsecurity.techtarget.com/definition/identity-management-ID-management>
4. Sebastián Peyrott, *What is and how does Single Sign On Authentication work?*, <https://auth0.com/blog/what-is-and-how-does-single-sign-on-work/>
5. Dave Piscitello, *What is Authorization and Access Control?*, <https://www.icann.org/news/blog/what-is-authorization-and-access-control>
6. OWASP, *Access Control*, <https://www.owasp.org/index.php/Category:Access_Control#Definition>
7. Gerard Enter and Peter Loshin, *SAML (Security Assertion Markup Language)*, <https://searchsecurity.techtarget.com/definition/SAML>
8. Security Services Technical Committee (OASIS), *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*, OASIS Standard, Mart 2005.
9. Security Services Technical Committee (OASIS), *A Brief Introduction to XACML*, <https://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html>
10. Scott Battaglia, *CASUM Introduction*, <https://wiki.jasig.org/display/CASUM/Introduction>
11. Apereo CAS Documentation, *CAS Enterprise Single Sign-On*, <https://apereo.github.io/cas/5.2.x/index.html>
12. Apereo CAS Documentation, *CAS Protocol*, <https://apereo.github.io/cas/5.2.x/protocol/CAS-Protocol.html>
13. Berkeley University, *How CAS Works*, <https://calnetweb.berkeley.edu/calnet-technologists/cas/how-cas-works>
14. Apereo CAS Documentation, *Architecture*, <https://apereo.github.io/cas/4.2.x/planning/Architecture.html#architecture>
15. Shibboleth, *What’s Shibboleth?*, <https://www.shibboleth.net/index/>
16. Shibboleth, *Identity Provider*, <https://www.shibboleth.net/products/identity-provider/>
17. Scott Cantor (Shibboleth Wiki), *Concept*, <https://wiki.shibboleth.net/confluence/display/CONCEPT#app-switcher>
18. Chad La Joie (Shibboleth Wiki), *GeneralArchitecture*, <https://wiki.shibboleth.net/confluence/display/IDP30/GeneralArchitecture#GeneralArchitecture-OverallArchitecture>
19. WSO2 Documentation, *Architecture*, <https://docs.wso2.com/display/IS540/Architecture>
20. Zach Dennis, *Choosing an SSO Strategy: SAML vs OAuth2*, <https://www.mutuallyhuman.com/blog/2013/05/09/choosing-an-sso-strategy-saml-vs-oauth2/>
21. ServiceNow Wiki, *SAML 2.0 Web Browser SSO Profile*, <http://wiki.servicenow.com/index.php?title=SAML_2.0_Web_Browser_SSO_Profile#gsc.tab=0>
22. Security Services Technical Committee (OASIS), *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*, OASIS Standard, Mart 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. Kolačić koji se automatski briše nakon zatvaranja internet preglednika, odnosno uništavanja sesije [↑](#footnote-ref-3)
4. *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-4)
5. https://oauth.net/about/introduction/ [↑](#footnote-ref-5)
6. https://en.wikipedia.org/wiki/List\_of\_single\_sign-on\_implementations [↑](#footnote-ref-6)
7. Softverska licenca koja označava to da softver može biti besplatno korišten, promijenjen i dijeljen. [↑](#footnote-ref-7)
8. https://shibboleth.net/downloads/identity-provider/ [↑](#footnote-ref-9)
9. http://shibboleth.net/downloads/service-provider/latest/ [↑](#footnote-ref-10)
10. https://wso2.com/blogs/thesource/2015/08/wso2-ten-years-it-takes-two-to-tango/ [↑](#footnote-ref-13)
11. https://www.w3.org/TR/2006/CR-wsdl20-20060327/wsdl20-z.pdf [↑](#footnote-ref-14)
12. https://wso2.com/products/carbon/release-matrix/ [↑](#footnote-ref-15)
13. https://docs.wso2.com/display/IS510/Identity+Server+Features [↑](#footnote-ref-16)
14. Shibboleth konzorcijum je organizacija koja obezbjeđuje stalni razvoj i održavanje Shibboleth aplikacija, među kojima se nalazi i ranije analizirana implementacija SSO rješenja pod nazivom “Shibboleth” [↑](#footnote-ref-21)