Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.738.5:339
DOI: https://doi.org/10.24867/04BE16Sulja

VREMENSKO I LOKACIJSKO PROŠIRENJE KONTROLE PRISTUPA BAZIRANE NA ULOGAMA U SPRING SECURITY OKRUŽENJU

TEMPORAL AND LOCATION-BASED EXTENSION OF ROLE-BASED ACCESS CONTROL WITHIN SPRING SECURITY FRAMEWORK

Jan Šulja, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je prikazana implementacija vremenskog i lokacijskog proširenja kontrole pristupa bazirane na ulogama u Spring Security okruženju. Implementacija je podeljena na dve celine, gde je prva celina generički modul za kontrolu pristupa, a druga, veb aplikacija koja taj modul koristi za kontrolu pristupa.*

Ključne reči: *Java, Spring, kontrola pristupa, autorizacija*

Abstract – *This paper presents the implementation of temporal and location-based extension of role-based access control within Spring Security framework. Implementation is divided into two units, where the first unit is a generic access control module, and the second unit is a web application that uses this module to control the access.*

Keywords: *Java, Spring, access control, authorization*

1. UVOD

Vreme i lokacija su važni atributi koji mogu odrediti politiku kontrole pristupa u organizaciji. Brzi porast korišćenja uređaja koji imaju ugrađen GPS, senzora i druge mobilne opreme, doveo je do pojave značajnih istraživanja u oblasti kontrole pristupa na osnovu lokacije i vremena pristupa. Politika kontrole pristupa na osnovu lokacije proširuje bezbednost aplikacije tako što dozvoljava ili zabranjuje pristup resursu sa određenih lokacija. Sa druge strane, vremensko proširenje pruža mogućnost zabrane ili dozvole pristupa resursu u željenim vremenskim intervalima [1].

Autori rada [2] daju predlog modela koji proširuje RBAC prostornim i vremenskim elementima. Njihov predlog predstavlja model za izdvajanje zahteva kontrole pristupa izuzetno mobilnih i dinamičnih okruženja kako bi obezbedili kontrolu pristupa zasnovanu na lokaciji i vremenu. U radu je predstavljen obiman prostorni i vremenski RBAC model. Pojam vremena je jedinstven nad svim entitetima RBAC modela (korisnici, uloge, dozvole) dok kontekst lokacije može biti različit za svaki od ovih entiteta.

Prema tome, korisnici koji se nalaze na nekoj lokaciji L1 mogu dobiti ulogu koja je omogućena u lokaciji L2 za dobijanje dozvola za operacije nad objektima koji se nalaze na lokaciji L3, pri čemu lokacije L1, L2 i L3 ne moraju biti iste lokacije [2].

Još jedan predlog modela koji opisuje vremensko i prostorno proširenje kontrole pristupa na osnovu uloga predstavljen je u radu [1]. U radu je dat predlog prostorno vremenskog modela kontrole pristupa zasnovanog na ulogama (STRBAC – *Spatio Temporal Role-Based Access Control*), kao proširenje prvobitnog RBAC modela. STRBAC model podržava dinamičko aktiviranje uloga i dozvola u zavisnosti od navedenih ograničenja. Takođe, daje novi pravac za kategorizaciju logičke lokacije i vremena, a pogodan je za organizacije koje zahtevaju kontrola pristupa na osnovu vremena i lokacije [1].

2. KONTROLA PRISTUPA

Pod pojmom kontrola pristupa, podrazumeva se dodela prava pristupa, kontrole ili upravljanja objektima od značaja zainteresovanom subjektu. Subjekat na osnovu precizno određenih pravila može dobiti pravo na vršenje određenih operacija nad traženim objektom ukoliko su zadovoljeni tačno definisani uslovi. Subjekat i objekat predstavljaju entitete u modelima kontrole pristupa, gde je subjekat entitet koji vrši operacije nad objektom, a objekat resurs sistema [3].

Način na koji se pravila kontrole pristupa definišu, opisuje se modelom kontrole pristupa, a neki od najzastupljenijih modela su:

- RBAC - role based access control [4]
- ABAC - attribute based access control [5]
- DAC - discretionary access control [6]
- MAC - mandatory access control [7]

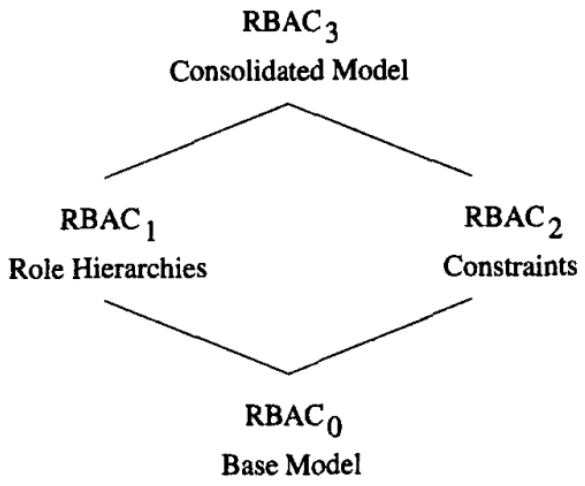
2.1. Kontrola pristupa na osnovu uloga

Kod ove kontrole pristupa administratori sistema kreiraju uloge u skladu sa poslovnim pozicijama unutar kompanije ili organizacije, odobravaju dozvole za te uloge, i dodeljuju subjektima uloge na osnovu njihovih radnih obaveza i kvalifikacija. Uloga predstavlja sposobnost subjekta da obavlja specifičan zadatak. Kombinacija korisnika sa dozvolama dodeljenim putem uloga, se obično često menja vremenom. Sa druge strane, veze između dozvole i uloge su stabilnijeg karaktera, ređe se menjaju nego ljudi koji rade posao predstavljen tom ulogom. Zbog toga se prilikom administracije bezbednosti koristi uloga, a ne dozvola. Ukoliko su promene potrebne, korisnicima se mogu jednostavno dodeljivati ili brisati uloge. Slično tome kako se sistem u kompaniji razvija, ulogama mogu biti dodeljivane nove ili uklonjene neke

---

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Goran Sladić, vanr. prof.

stare dozvole [8]. Neke varijacije ove kontrole pristupa uključuju sposobnost uspostavljanja odnosa između uloga, kao i između korisnika i uloga.
Na primer, dve uloge mogu biti uzajamno isključive, istom korisniku nije dozvoljeno da ima dodeljene obe uloge. Uloge mogu takođe sticati i veze nasleđivanja, pri čemu jedna uloga nasleđuje dozvole dodeljene drugoj ulozi.
Veze između uloge i dozvole mogu biti a obično i jesu predefinisane, što olakšava dodelu uloga korisnicima.
Politika kontrole pristupa na osnovu uloga obuhvaćena je vezama kao što su uloga-dozvola, korisnik-uloga i uloga-uloga.
Ove veze kolektivno određuju da li je korisniku dozvoljen pristup određenom delu sistema i mogu direktno biti konfigurisane od strane administratora sistema ili indirektno putem odgovarajućih uloga koje delegira administrator sistema. Politika koja se sprovodi u datom sistemu, rezultat je specifične konfiguracije ovih veza koje navodi administrator sistema.
Pošto politika kontrole pristupa može, a obično se i menja tokom životnog ciklusa sistema, kontrola pristupa na osnovu uloga pruža mogućnost modifikacije kontrole pristupa kako bi organizacione potrebe bile zadovoljene u svakom trenutku životnog ciklusa sistema [8]. RBAC modeli kontrole pristupa definisani su familijom od četiri modela (*slika 1*). $RBAC_0$ je osnovni model i predstavlja minimalni deo koji mora da bude zadovoljen od strane sistema, da bi se za taj sistem moglo reći da koristi RBAC model za kontrolu pristupa.
Komponente $RBAC_0$ modela su: korisnici, uloge, dozvole i sesije. $RBAC_1$ i $RBAC_2$ uključuju osnovni RBAC model, ali mu dodaju mu neke nezavisne funkcionalnosti. $RBAC_1$ dodaje koncept hijerarhije uloga (uloge mogu naslediti dozvole iz drugih uloga). $RBAC_2$ dodaje ograničenja (na primer uzajamno isključive uloge).
Ujedinjeni model $RBAC_3$ uključuje $RBAC_1$, $RBAC_2$ i tranzitivno $RBAC_0$ [8].

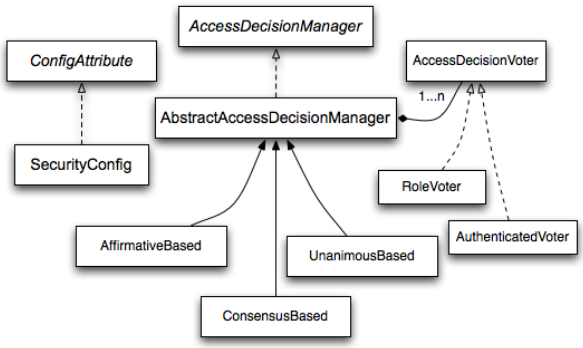


Slika 1. *Veze RBAC modela*

2.2. Kontrola pristupa u spring okruženju

U spring okruženju interfejs *AccessDecisionManager* je zadužen za kontrolu pristupa (*slika 2*). Utvrđivanje prava pristupa zasnovano je na glasanju, a postoje tri konkretne implementacije *AccessDecisionManager-a*:

- *AffirmativeBased* – pristup je dozvoljen ukoliko je bar jedan glasač dao glasao pozitivno
- *UnanimousBased* – pristup je dozvoljen ako su svi glasači bili uzdržani ili glasali pozitivno
- *ConsensusBased* - pristup je dozvoljen ako je većina glasača glasala pozitivno



Slika 2. *Arhitektura autorizacije u Spring okruženju*

Najčešće se koristi glasač tipa *RoleVoter*, kod kontrole pristupa na osnovu uloga. Ovaj glasač će glasati ukoliko postoji bar jedan konfiguracioni atribut koji počinje prefiksom „ROLE_“. Ako korisnik ima dodeljenu ulogu definisanu u konfiguracionom atributu, glasač će glasati pozitivno, u suprotnom glasač daje negativan glas. U slučaju da u konfiguracionim atributima ne postoji atribut koji počinje prefiksom „ROLE_“ glasač neće glasati, odnosno biće uzdržan [9].

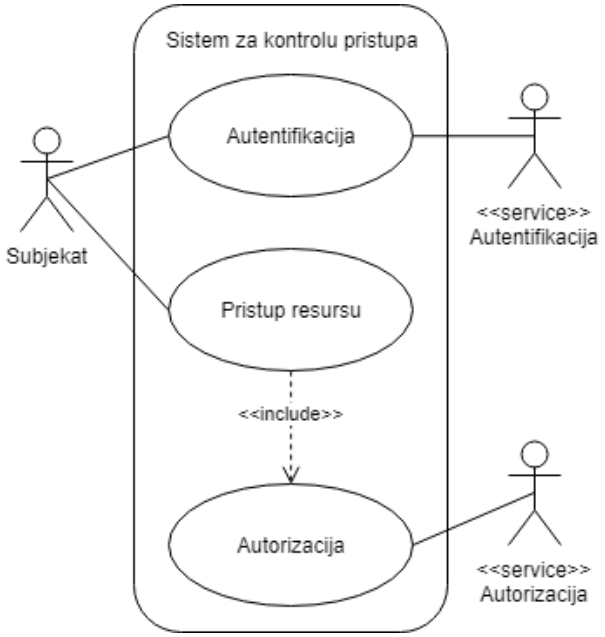
3. MODEL SISTEMA

Projektovanje modula za kontrolu pristupa zasnovano je na ideji da se za svaki resurs u sistemu kome je kontrola pristupa potrebna definiše lista „brava“ koje je potrebno otključati tačno određenim „ključevima“. Pristup resursu se dozvoljava samo ako su sve brave na resursu uspešno otključane, u suprotnom zahtev za pristup traženom resursu se odbija. Brava je u modulu za kontrolu pristupa modelovana interfejsom *AccessLock*, dok ključ može biti atribut subjekta, atribut sistema, tip *AccessTicket* kao što je slučaj u veb aplikaciji ovog rada ili neke druge vrednosti.
Prilikom procesa odluke o dodeli prava pristupa resursu, svaki od *AccessLock* tipova definisanih za taj resurs prolazi kroz lanac odgovornosti, gde će ga odgovarajuća komponenta koja je zadužena za konkretan *AccessLock* obraditi. Ukoliko uslov potreban za dobijanje prava pristupa nije zadovoljen baca se izuzetak, u suprotnom izvršavanje procesa obrade zahteva se nastavlja dalje dok neka komponenta ne baci izuzetak ili proces obrade ne dođe do kraja lanca obrade. Ukoliko ni jedna komponenta ne baci izuzetak u ovom procesu obrade pristup resursu će biti uspešno odobren.

3.1 Učesnici u sistemu

U sistemu postoje tri učesnika, subjekat, servis za autentifikaciju i servis za autorizaciju. Na slici 3 prikazan je dijagram slučajeva korišćenja za pristup resursima. Ukoliko se pristupa javnom resursu koji je dostupan svima, servisi za autentifikaciju i autorizaciju se neće koristiti. Ako je resurs kome se pristupa zaštićen, onda subjekat pre pristupa resursu obavlja autentifikaciju a nakon uspešne autentifikacije sa token-om koji mu je

dodeljen može da zatraži pristup resursu. Pre dobijanja dozvole pristupa, servis za autorizaciju proverava da li subjekat ima prava da dobije pristup traženom resursu, ako je odgovor da, subjekat dobija pravo pristupa u suprotnom pristup je zabranjen.



Slika 3. *Dijagram slučajeva korišćenja pristupa resursu*

3.2. Model podataka

Osnovna ideja modela podataka je podržavanje više različitih vrsta privilegija za kontrolu pristupa, uz jednostavno dodavanje novih tipova privilegija kontrole pristupa. Jedan od ciljeva ovog modela je i da se omogući proizvoljno modelovanje kompleksnih hijerarhija privilegija kontrole pristupa, tako da može na najbolji mogući način da zadovolji potrebe što većeg broja različitih aplikacija, čime se dobija i mogućnost proširenja kontrole pristupa na osnovu uloga vremenskim i prostornim komponentama što je i zadatak ovog rada. Pored pomenutih komponenti, vrlo je jednostavno proširiti sistem drugim vrstama privilegija za kontrolu pristupa.

4. IMPLEMENTACIJA SISTEMA

Kada govorimo o tehnologijama koje su korišćene u ovom radu, moramo napomenuti da aplikaciju možemo podeliti na dve celine, prva celina predstavlja generičko rešenje koje omogućava kreiranje različitih tipova kontrole pristupa, dok druga celina predstavlja samo korišćenje alata koje nudi prva celina. Nakon što smo to ustanovili možemo zasebno opisati tehnologije koje su korišćene u ove dve celine.

Prva celina koja nudi generičko rešenje kontrole pristupa je implementirana koristeći Java [10] programski jezik, stoga može biti korišćena u bilo kojoj drugoj aplikaciji koja se bazira na Java programskom jeziku, bilo da je to veb bazirana aplikacija, Desktop aplikacija, mobilna aplikacija ili aplikacija nekog drugog tipa.

Druga celina predstavlja rešenje koje demonstrira primenu različitih tipova kontrole pristupa uz mogućnost proširenja baznih implementacija kontrole pristupa i implementirano je u Java programskom jeziku koristeći Spring okruženje [11].

Komponente zadužene za odlučivanje o dodeli prava pristupa date su kroz projektni obrazac Lanac odgovornosti, sa osnovnom klasom *AuthorizationHandler* (*listing 1*) koju nasleđuju sve konkretne komponente u lancu odgovornosti. Klasa *AuthorizationHandler* sadrži jedan atribut *successor* istog tipa koji predstavlja referencu na sledeću komponentu u lancu, dve apstraktne metode i jednu metodu sa implementacijom.

Prva metoda *doAuthorization(AccessLock)* zadužena je za obradu *AccessLock-a*, odnosno odlučivanju o dodeli ili zabrani pristupa resursu koji je zaštićen određenim *AccessLock* tipom.

Druga metoda *getResponsibility()* ima za ulogu da vrati konkretni tip *AccessLock-a* za koji je ta konkretna komponenta zadužena.

Treća metoda *handleRequest(AccessLock)* odlučuje na osnovu odgovornosti *AccessLock-a* da li je trenutna komponenta u lancu odgovorna za taj *AccessLock* pozivom prve metode *getResponsibility()*, ako jeste, poziva dalje drugu metodu *doAuthorization(AccessLock)* i dalje poziva *handleRequest(AccessLock)* nad sledećom komponentom u lancu, ovim se obezbeđuje obrada jednog konkretnog *AccessLock* tipa od strane više komponenti u lancu, a ako nije odgovorna u tom slučaju se samo poziva *handleRequest(AccessLock)* nad sledećom komponentom u lancu.

Metoda *doAuthorization(AccessLock)* odlučuje da li će pristup biti dozvoljen ili zabranjen, ukoliko dođe do zabrane ta metoda baca izuzetak.

```
public abstract class AuthorizationHandler {

	protected AuthorizationHandler successor;

	protected abstract void doAuthorization(AccessLock lock) throws Exception;
	protected abstract Class<? extends AccessLock> getResponsibility();

	public void setSuccessor(AuthorizationHandler successor) {
		this.successor = successor;
	}

	public void handleRequest(AccessLock lock) throws Exception {

		if(lock.getClass().equals(this.getResponsibility())) {
			this.doAuthorization(lock);
		}

		if(nonNull(successor)) {
			successor.handleRequest(lock);
		}

}
```

Listing 1 *Klasa AuthorizationHandler*

Kako bi se omogućilo korišćenje modula za kontrolu pristupa registruje se novi filter koji će automatski biti dodat posle Spring security filter chain-a. Za donošenje odluke o kontroli pristupa koristi se *AuthorizationService* pozivom metode *authorize*. Ukoliko ova metoda baci izuzetak, kao odgovor se vraća poruka o grešci sa HTTP status kodom 401.

Servis klasa *AuthorizationService* je zadužena za konfigurisanje modula za kontrolu pristupa. Konfigurisanje se sastoji iz konstrukcije lanca odgovornosti i delegiranja zahteva za obradu u lanac odgovornosti.

Klasa *SimpleAuthorizationService* predstavlja jednu od mogućih implementacija ovog servisa (*listing 2*).

Korišćenjem Spring constructor dependency injection mehanizma u trenutku kada Spring okruženje instancira ovaj servis, u konstruktoru kreiramo lanac odgovornosti i inicijalizujemo atribut *securityConfig* koji sadrži konfiguraciju za kontrolu pristupa resursima.

```
@Service
public class SimpleAuthorizationService implements AuthorizationService {

  private SecurityConfig securityConfig;
  private AuthorizationHandler authorizationChain;

  public SimpleAuthorizationService(SecurityConfig securityConfig,
                                    List<AuthorizationHandler> links) {

    this.securityConfig = securityConfig;
    this.authorizationChain = links.stream()
                                   .reduce(null, (a,b) -> { b.setSuccessor(a); return b; });

  }

  @Override
  public void authorize(HttpServletRequest request) throws Exception {

    securityConfig.getLocks(request)
                  .stream()
                  .forEach(this::authorize);

  }

  @SneakyThrows
  private void authorize(AccessLock lock) {

    authorizationChain.handleRequest(lock);

  }
}
```

Listing 2. *Implementacija servisa za autorizaciju*

Nakon inicijalizacije lanca odgovornosti atribut *authorizationChain* predstavlja referencu prve komponente u lancu. Kao rezultat ovog načina inicijalizacije lanca odgovornosti, dobili smo vrlo jednostavno registrovanje novih komponenti u lanac odgovornosti. Za dodavanje nove autorizacione komponente nije potrebno menjati postojeću logiku za kreiranje lanca odgovornosti, potrebno je samo naslediti klasu *AuthorizationHandler* sa anotacijom @Component a prilikom sledećeg pokretanja aplikacije ova komponenta će automatski biti dodata u lanac odgovornosti, čime smo dobili vrlo jednostavno dodavanje novih autorizacionih komponenti u postojeći sistem.

Uloga javne metode *authorize* je da za resurs za koji se proverava kontrola pristupa preuzme iz *SecurityConfig* objekta sve *AccessLock* tipove koji definišu pravila o kontroli pristupa i da svaki od njih pošalje u lanac odgovornosti na kontrolu. Ova javna metoda se poziva iz filtera *AuthorizationFilter* prilikom svakog primljenog HTTP zahteva.

Za konfiguraciju kontrole pristupa koristi se klasa SecurityConfig a način konfigurisanja kontrole pristupa je sličan kao u Spring okruženju. U prvom delu koda (*listing 3*) data je konfiguracija *SecurityConfig* klase, koja predstavlja deo rešenja ove aplikacije, dok drugi deo prikazuje konfiguraciju Spring Security modula, dakle dozvoljena je kombinacija korišćenja obe implementacije kontrole pristupa.

```
securityConfig
       .antMatcher("/access_control_test/test3")
              .access(rbac("ADMIN"))
       .antMatcher("/access_control_test/test4")
              .access(rbac("USER"))
       .antMatcher("/access_control_test/test5")
              .access(ibac("178.220.0.0-178.223.255.255"))
       .antMatcher("/access_control_test/test6")
              .access(tbac())
       .antMatcher("/access_control_test/test7")
              .access(ibac("178.220.0.0-178.223.255.255"), tbac());

http.authorizeRequests()
       .antMatchers("/access_control_test/test2")
              .authenticated();
```

Listing 3. *Konfiguracija kontrole pristupa*

5. ZAKLJUČAK

U radu je prikazana implementacija vremenskog i lokacijskog proširenja kontrole pristupa bazirane na ulogama. Pored glavnog zadatka čiju je implementaciju pratio ovaj rad prikazano je generičko rešenje modula za kontrolu pristupa koji je omogućio jednostavnu implementaciju ovog tipa kontrole pristupa i otvorilo vrata za implementaciju bilo koje druge vrste kontrole pristupa. Pored ovih implementacija dat je prikaz konceptualnog modela i dijagrama klasa ovih rešenja, kao i demonstracija upotrebe ovih aplikacija kroz interaktivni korisnički interfejs. Prikazan je i primer jednostavnog proširenja modula za kontrolu pristupa, slično tom primeru ovaj modul se može proširiti bilo kojom novom vrstom kontrole pristupa.

6. LITERATURA

[1] Kumar, Mahendra / Newman, Richard E. *STRBAC - AN APPROACH TOWARDS SPATIO-TEMPORAL ROLE-BASED.* Gainesville, FL 32608 : CISE Department Gainesville, FL 32608, 2006.

[2] *LoT-RBAC: A Location and Time-Based RBAC Model.* Chandran, Suroop Mohan / Joshi, J. B. D. Department of Information Sciences and Telecommunications, University of Pittsburgh : s.n., 2005.

[3] Access control. *Wikipedia.* [Na mreži] https://en.wikipedia.org/wiki/Access_control.

[4] *Role based access control.* [Na mreži] https://en.wikipedia.org/wiki/Role-based_access_control.

[5] *Attribute based access control.* [Na mreži] https://en.wikipedia.org/wiki/Attribute-based_access_control.

[6] *Discretionary access control.* [Na mreži] https://en.wikipedia.org/wiki/Discretionary_access_control.

[7] *Mandatory access control.* [Na mreži] https://en.wikipedia.org/wiki/Mandatory_access_control.

[8] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. Role-based access control models. [Na mreži] 1996. https://ieeexplore.ieee.org/abstract/document/485845.

[9] Authorization. *Spring.* [Na mreži] https://docs.spring.io/spring-security/site/docs/current/reference/html/authorization.html.

[10] *Java.* [Na mreži] https://docs.oracle.com/javase/8/docs/api/.

[11] *Spring.* [Na mreži] https://spring.io/.

Kratka biografija:

Jan Šulja je rođen 17.12.1993 godine u Zrenjaninu. Osnovnu školu "Maršal Tito" u Padini je završio 2008. godine. Elektrotehničku građevinsku školu „Nikola Tesla" u Zrenjaninu završio je 2012. godine. Osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu završio je 2016. godine na studijskom programu Računarstvo i automatika, smer Primenjene računarske nauke i informatika. Godine 2016. upisuje master akademske studije na Fakultetu tehničkih nauka u Novom Sadu, na studijskom programu Računarstvo i automatika, smer Primenjene računarske nauke i informatika, modul Elektronsko poslovanje.