Implementacija SSO na bazi open source rješenja

Zoran I. Đurić, *Elektrotehnički fakultet Banjaluka, Univerzitet u Banjaluci*, Dijana R. Vuković, *Univerzitetski računarski centar, Univerzitet u Banjaluci*

***Sadržaj* — U ovom radu dat je primjer centralizovanog sistema za autentikaciju koji obezbjeđuje SSO (Single Sign-On) funkcionalnost. Sistem je baziran na open source CAS (Central Authentication Service) serveru. Ukazano je na neke dobre i loše osobine SSO rješenja i dat je opis rada CAS servera na konkretnom primjeru. U radu je opisana implementacija CAS servera u testnom okruženju sa više web aplikacija. Razmatrani su i načini implementacije modula za autentikaciju i dati su primjeri implementacije u testnom okruženju. Posebno su razmatrani problemi koji se javljaju pri prilagođavanja postojećih aplikacija za korištenje CAS servera, ali i drugih SSO rješenja.**

***Ključne reči* — autentikacija, CAS, SSO.**

I. UVOD

POSLJEDNJIH nekoliko godina broj web i SOA (Service Oriented Architecture) aplikacija značajno se povećava. Web portali postaju sve sofisticiraniji i ispunjavaju sve komplikovanije tehničke i funkcionalne zahtjeve. Postoje različite vrste alata i framework-a koji omogućavaju brz razvoj manje komplikovanih web portala. Korištenje ovakvih alata za brz razvoj ne rješava brojna pitanja, posebno pitanja sigurnosti aplikacija i njihove integracije sa drugim sistemima. Jedno od otvorenih pitanja je i pitanje autentikacije u okruženju sa velikim brojem web i SOA aplikacija. Rješavanje problema autentikacije u ovakvom okruženju nije jednostavno, jer svaka aplikacija ima vlastiti mehanizam za autentikaciju koji se obično baziraju na kombinaciji korisničkog imena i lozinke. Ovi mehanizmi koriste različite sisteme za skladištenje podataka, kao i različite sigurnosne mehanizme. Postojanje različitih mehanizama za autentikaciju podrazumijeva i postojanje više različitih podataka koje korisnici moraju znati i/ili posjedovati. Jasno je da ovakvo okruženje komplikovanije za održavanje i da negativno utiče na sigurnost čitavog sistema [1].

Rješenje koje, pored jednostavnijeg održavanja, obezbjeđuje i jednostavan pristup svim aplikacijama i centralizovanu sigurnost jeste SSO (Single Sign-On) [3]. SSO rješenje donosi brojene prednosti, kako za krajnje korisnike aplikacije, tako i za administratore. Ovakva rješenja omogućavaju korisnicima da se prijave samo jednom i da nakon toga mogu ostvariti pristup svim aplikacijama za koje su autorizovani.

U ovom radu dat je primjer centralizovanog sistema za autentikaciju koji obezbjeđuje SSO funkcionalnost. Sistem je baziran na open source CAS (Central Authentication Service) serveru [2]. U sekciji 2 ukratko je ukazano na neke dobre i loše osobine SSO rješenja. U sekciji 3 dat je opis rada CAS servera na konkretnom primjeru. U sekciji 4 opisana je implementacija CAS servera u testnom okruženju. U ovoj sekciji opisani su način implementacije modula za autentikaciju i problemi koji se javljaju pri prilagođavanja postojećih aplikacija za korištenje CAS servera. Na kraju rada dat je zaključak.

II. DOBRE I LOŠE OSOBINE SSO RJEŠENJA

Na osnovu uvodnog teksta jasno je da implementacija SSO u ciljom okruženju donosi određene prednosti. Kao dobre osobine SSO mogu se izdvojiti: povećanje produktivnosti korisnika, povećanja produktivnosti razvojnih timova, jednostavnije održavanje i smanjenje troškova. U ovakvom okruženju od korisnika se ne zahtijeva da pamte različite kombinacije korisničkih imena i lozinki, kao i da se više puta prijavljuju na različite aplikacije. Razvojni timovi ne moraju da razvijaju module za autentikaciju, niti da vode računa o skladištenju podataka za autentikaciju. Administracija korisničkih naloga u okruženju sa SSO je znatno pojednostavljena. Stepen pojednostavljenosti zavisi od toga da li SSO obezbjeđuje samo autentikaciju, što je njegova osnovna namjena, ili i autorizaciju korisnika. U slučaju da SSO ne obezbjeđuje autorizaciju, svaka od aplikacija će zahtijevati podešavanje određenih atributa korisničkih naloga, poput privilegija pristupa. Smanjenje troškova je logična posljedica prethodno navedenog.

Kao neki od najznačajnijih nedostataka SSO mogu se navesti: problem prilagođavanja postojećih aplikacija kako bi mogle koristiti SSO, problem „ostavljenog“ računara i problem jedne tačke za napad na sistem. U zavisnosti od SSO rješenja, ali i drugih faktora, poput dostupnosti izvornog koda i razvojnog tima, moguće je da će prilagođavanje postojećih aplikacija za SSO biti komplikovano i da će trajati značajan vremenski period. Problem „ostavljenog“ računara spada u domen socijalnog inžinjeringa i podrazumijeva situaciju u kojoj maliciozni korisnik putem računara drugog korisnika ostvari pristup

Z. Đurić, Elektrotehnički fakultet Banjaluka, Patre 5, 78000 Banjaluka, Republika Srpska, BiH (telefon: 387-51-221820; faks: 385-51-211408; e-mail: zoran.djuric@etfbl.net).

D. Vuković, Univerzitetski računarski centar, Bulevar Vojvode Stepe Stepanovića 75A, 78000 Banjaluka, Republika Srpska, BiH (telefon: 387-51-465595; e-mail: dijana.vukovic@etfbl.net).

resursima za čije je korišćenje drugi korisnik autorizovan. Iako ovaj problem postoji i u okruženju bez SSO, jasno je da je on u okruženju sa SSO multipliciran. Kako u okruženju sa SSO sve aplikacije koriste centralni sistem za autentikaciju, on postaje atraktivna meta za različite vrste napada malicioznih korisnika.

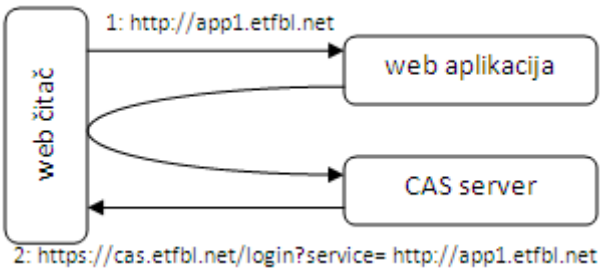
Bez obzira na ove nedostatke koji nisu zanemarljivi, prednosti koje ovakav sistem donosi iz ugla korisnika, administratora i razvojnih timova su mnogo značajnije.

III. CAS

CAS predstavlja servis za autentikaciju koji je razvijen na Yale univerzitetu s ciljem obezbjeđivanja centralnog mjesta za autentikaciju korisnika za pristup web aplikacijama. CAS ne posjeduje vlastitu bazu podataka u kojoj se čuvaju podaci za autentikaciju korisnika, već koristi vanjske sisteme za skladištenje podataka za autentikaciju korisnika. CAS podržava različite tipove autentikacije, uključujući autentikaciju putem baza podataka, LDAP (Lightweight Directory Access Protocol) infrastrukture, X.509 digitalnih sertifikata i JAAS (Java Authentication and Authorization Service). Web aplikacije koje koriste CAS mogu od ovog sistema da dobiju rezultate autentikacije korisnika i informacije o atributima autentikovanih korisnika. U procesu autentikacije CAS koristi fundamentalne web tehnologije poput kolačića (cookie), HTTP redirekcije i Java skripta.

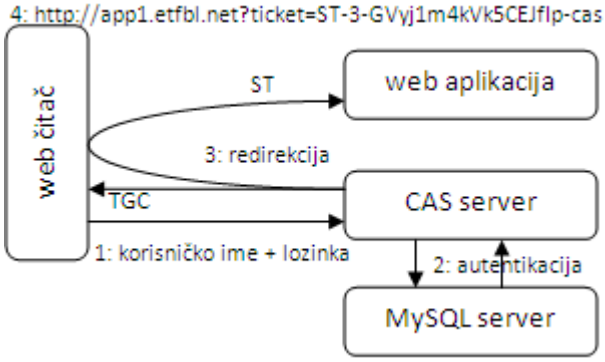
Mehanizam za autentikaciju CAS-a koristi sljedeća dva tiketa (ticket): TGC (Ticket Granting Cookie) i ST (Service Ticket). TGC je kolačić koji CAS server generiše za svaki od autentikovanih korisnika (web čitača), a autentikovani web čitač prilikom pristupa web aplikaciji šalje ST kao URL parametar.

Protokol za autenikaciju koji koristi CAS sastoji se iz sljedeća 3 koraka. Kada korisnik putem web čitača pokuša pristupiti web aplikaciji bez ST (Sl. 1, oznaka 1), CAS klijentski modul ugrađen u web aplikaciju inicira HTTP redirekciju na CAS server, prosljeđujući URL zahtijevane aplikacije kao parametar (Sl. 1, oznaka 2). Korisnikov web čitač (u kojem je sada učitana forma za prijavu) i CAS server komuniciraju putem HTTPS konekcije.



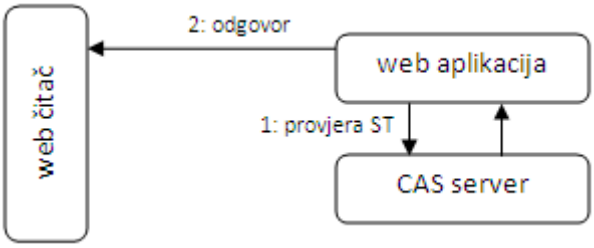
Sl. 1. CAS protokol za autentikaciju, korak 1.

Korisnik unosi korisničko ime i lozinku i submit-uje formu za prijavu (Sl. 2, oznaka 1). CAS server pokušava da autentikuje korisnika koristeći neki od raspoloživih sistema za autentikaciju korisnika (Sl. 2, oznaka 2). Za potrebe pilot projekta implementacije CAS servera u testnom okruženju, implementirana je podrška za SQL baze podataka i LDAP infrastrukturu. Ako autentikacija nije uspjela aplikacija kojoj je korisnik pokušao da pristupi u koraku 1 za to neće ni znati. Ako CAS server uspješno autentikuje korisnika, generisaće TGC, poslaće ga web čitaču i iniciraće redirekciju na zahtijevanu web aplikaciju (Sl. 2, oznaka 3). U ovom koraku ST se ubacuje u redirektovanu URL adresu (Sl. 2, oznaka 4).



Sl. 2. CAS protokol za autentikaciju, korak 2.

Na ovaj način ciljna web aplikacija dolazi u posjed ST-a i može izvršiti njegovu verifikaciju šaljući ga CAS serveru, kroz HTTPS konekciju, zajedno sa nazivom aplikacije (Sl. 3, oznaka 1). CAS provjerava da li je ST validan i povezan sa odgovarajućom aplikacijom. Ako se provjera okonča uspješno, CAS server će aplikaciji vratiti korisničko ime, a pristup aplikaciji od strane web čitača korisnika će biti omogućen (Sl. 3, oznaka 2).

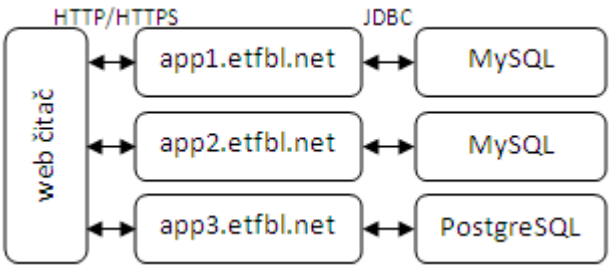


Sl. 3. CAS protokol za autentikaciju, korak 3.

IV. Implementacija CAS servera u testnom okruženju

Za potrebe pilot projekta implementacije CAS servera, izvršena je njegova implementacija u testnom okruženju. Testno okruženje sačinjavale su tri web aplikacije (slika 4). Dvije aplikacije implementirane su korišćenjem JSP (Java Server Pages) tehnologije, a jedna je bila bazirana na JSF (Java Server Faces) tehnologiji. Prijava na svaku od aplikacija odvijala se na takav način da je korisnik morao da unese svoje pristupne podatke u formi korisničkog imena i lozinke. Svaka aplikacija posjedovala je namjenski razvijene module za autentikaciju, koristila je nazavisne baze podataka, na različitim sistemima za upravljanje bazama podataka. Na ovaj način svaki od korisnika morao je da pamti tri različite kombinacije korisničkog imena i lozinke i morao je da prođe proces autentikacije pri pokušaju pristupa svakoj od aplikacija. U slučaju da su sve aplikacije koristile jedinstvenu bazu sa korisničkim nalozima, korisnik bi morao pamtiti samo jednu kombinaciju korisničkog imena i lozinke, ali bi i dalje morao da prođe proces autentikacije pri pokušaju pristupa

svakoj od aplikacija.



Sl. 4. Testno okruženje prije implementacije SSO rješenja.

Za potrebe implementacije CAS u testnom okruženju implementirana su dva modula za autentikaciju korisnika. Ova dva modula razvijena su proširenjem CAS-ove autentikacione arhitekture, konkretno implementacijom AuthenticationHandler interfejsa iz paketa org.jasig.cas.authentication.handler. Izvorni Java kod ovog interfejsa dat je na slici 5.

```
package org.jasig.cas.authentication.handler;

import org.jasig.cas.authentication.principal.Credentials;

public interface AuthenticationHandler {
        boolean authenticate(Credentials credentials)
                        throws AuthenticationException;
        boolean supports(Credentials credentials);
}
```

Sl. 5. AuthenticationHandler interfejsa iz paketa org.jasig.cas.authentication.handler.

Prvi (DBUsernamePasswordAuthenticationHandler) modul implementira autentikaciju korisnika putem korisničkog imena i lozinke na neku od JDBC (Java Database Connectivity) kompatibilnih baza podataka, dok drugi (LDAPUsernamePasswordAuthenticationHandler) modul implementira autentikaciju korisnika na LDAP infrastrukturu koristeći JNDI (Java Naming and Directory Interface). Primjena ova dva modula na CAS sistemu izvršena je na taj način da su oni zamjenili originalni, testni, CAS modul za autentikaciju korisnika. Zamjena je izvršena promjenom parametara u deployerConfigContext.xml konfiguracionoj datoteci (Sl. 6).

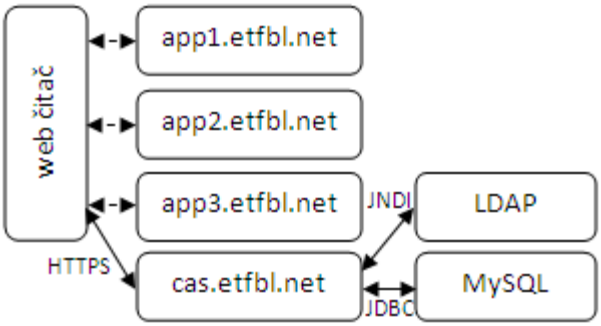
```
<!--testni modul za autentikaciju-->
<bean class="SimpleTestUsernamePasswordAuthenticationHandler" />

<!--namjenski razvijeni moduli za autentikaciju-->
<bean class="DBUsernamePasswordAuthenticationHandler" />
<bean class="LDAPUsernamePasswordAuthenticationHandler" />
```

Sl. 6. Zamjena modula za autentikaciju na CAS sistemu.

Nakon implementacije i zamjene modula za autentikaciju, bilo je neophodno prilagoditi postojeće aplikacije za korištenje SSO mehanizma. Aplikacije koje su imale samo jedan nivo privilegija korisničkih naloga nisu zahtijevale bilo kakvu modifikaciju izvornog koda za korištenje SSO mehanizma.

Kako CAS server nije dizajniran da podržava autorizaciju, JSF aplikacija koja je posjedovala više nivoa privilegija korisničkih naloga zahtijevala je modifikaciju izvornog koda modula za autorizaciju. Modul za autorizaciju je bilo potrebno prilagoditi na takav način da sada koristi korisničko ime dobijeno od CAS servera u trećem koraku CAS protokola za autentikaciju (Sl. 3), te da na bazi dobijenog korisničkog imena i interne baze podataka određuje privilegije korisnika. Nakon ovih modifikacija testno okruženje je u potpunosti moglo da koristi SSO mehanizam (Sl. 7), tj. od ovog trenutka autentikacija korisnika vršila se na centralnom mjestu – CAS serveru.



Sl. 7. Testno okruženje poslije implementacije SSO rješenja.

Na osnovu iskustava stečenih pri prilagođavanju postojećih aplikacija može se zaključiti da se statički web sadržaj, aplikacije koje nemaju module za autentikaciju, kao i aplikacije koje imaju modul za autentikaciju i samo jedan nivo privilegija korisničkih naloga mogu veoma lako prilagoditi za korištenje SSO mehanizma. Kod aplikacija koje imaju vlastite mehanizme za autorizaciju korisnika (sa više nivoa privilegija korisničkih naloga) neophodno je modifikovati module za autorizaciju. Poseban problem predstavljaju aplikacije koje ne mogu biti modifikovane iz bilo kojeg razloga, na primjer zato što ne postoji izvorni kod. Iako ovaj problem nije predmet ovog rada, jasno je da bi njihovo prilagođavanje za korištenje SSO mehanizma bilo znatno komplikovanije i uključivalo bi komplikovanije akcije, poput presretanja zahtjeva za autentikaciju i njihovo prosljeđivanje centralnom serveru za autentikaciju.

V. ZAKLJUČAK

Rješavanje problema autentikacije u okruženju sa velikim brojem web i SOA aplikacija nije jednostavno, jer svaka aplikacija ima vlastiti mehanizam za autentikaciju. Ovakva okruženja imaju brojne nedostatke, poput postojanje više različitih autentikacionih podataka koje korisnici moraju znati i/ili posjedovati, koji dovode do niskog nivoa sigurnosti čitavog sistema. Implementacija SSO rješenja u ovakvom okruženju može otkloniti mnoge nedostatke.

U ovom radu je analiziran problem implementacije SSO rješenja u okruženju sa velikim brojem web aplikacija. Na praktičnom primjeru implementacije CAS servera u testnom okruženju sa tri web aplikacije ukazano je na

potencijalne probleme prilagođavanja postojećih aplikacija za korištenje SSO mehanizma.

LITERATURA

[1] W. Timothy van der Horst, Kent E. Seamons, "Simple authentication for the web", Proceedings of the 16th international conference on World Wide Web, May 2007.

[2] Central Authentication Service, http://www.jasig.org/cas/, 2009.

[3] A. Pashalidis, C.J. Mitchell, "Impostor: a single sign-on system for use from untrusted devices", in: Proc. IEEE Globecom 2004, IEEE Press (2004), Volume 4, pp.2191-2195, 2004.

ABSTRACT

In this paper an example of central authentication system for SSO (Single Sign-On) is presented. This system is based on CAS (Central Authentication Service), an open source solution. CAS server implementation in test environment with several web applications is given. Implementation of authentication modules is also considered and examples of their use in test environment are presented. Some common existing problems, when readjustment of existing applications for use in SSO environment is in question, are especially addressed.

SSO IMPLEMENTATION BASED ON OPEN SOURCE SOLUTION

Z. Đurić, D. Vuković