Single sign-on **L**ooking **E**asy - **SLEASY**

Autori:
Darko Grabar
Mate Boban

Prošireni sažetak:

U današnje vrijeme imamo situaciju u kojoj u svakom poduzeću pa tako i na fakultetu postoji veća količina različitih Web aplikacija u konačnici namijenjena istom skupu korisnika. Najčešće svaka od tih aplikacija ima svoje autentifikacijske i autorizacijske mehanizme što dovodi do toga da za svaku od tih aplikacija korisnik mora pamtiti svoje pristupne podatke. Neka istraživanja su pokazala da ukoliko korisnik ima različite pristupne podatke za pojedine dijelove sustava razina sigurnosti se u stvari smanjuje umjesto povećava. Razlog je vrlo jednostavan. Ukoliko korisnici imaju veći broj različitih pristupnih podataka oni da izbjegnu problem njihovog pamćenja pribjegnu najjednostavnijem mogućem rješenju, zapišu ih. I to većinom na najgore moguće mjesto poput papirića na svojem radnom stolu ili još gore u svoje osobno računala (najčešće u datoteku na desktopu). Drugo, bolje rješenje je da svaka od tih aplikacija koristi jedinstveni autentifikacijski mehanizam poput LDAP-a. To je značajan napredak u odnosu na prethodan primjer ali opet ne rješava problem kontinuiranog ponavljanja autentifikacijske procedure prilikom pristupa svakoj od aplikacija. Kao jedno do mogućih rješenja ovog problema je implemetacija SSO mehanizma.
Upotrebom SSO-a postižemo da se korisnik mora autentificirati samo jednom te time automatski dobiva pristup svim aplikacijama unutar SSO domene. Open Group definira SSO kao mehanizam putem kojeg korisnik samo s jednom autentifikacijskom i autorizacijskom procedurom može koristiti sva računala i sustave za koje ima pravo pristupa bez potrebe za ponovnim unosom lozinke.

Ovakav pristup ima brojne prednosti u odnosu na gore navedene:

1. **za krajnjeg korisnika:**
    - samo jedan autentifikacijski mehanizam za pamćenje
    - nema potrebe za ponavljanjem autenitifikacije
2. **za administratore**
    - jedinstvena baza korisničkih podataka
    - jedinstven način upravljanja tim podacima
    - jedinstvena sigurnosna infrastruktura
3. **sigurnost**
    - zajednička sigurnosna infrastruktura za sve sustave kojom je onda lakše upravljati
    - laka provjera korisničkih podataka te njihovo ažuriranje
    - podizanje sigurnosti korištenjem samo jedne lozinke (manje su šanse da ju korisnici negdje zapišu)

Upravo zbog svih ovih gore navedenih razloga na Fakultetu organizacije i informatike se pristupilo izradi SSO sustava kojim će se povezati nekolicina manjih Web aplikacija koje

se koriste u svakodnevnom radu a do sada su zahtijevale odvojene autentifikacijske procedure. Prije donošenja odluke o izradi vlastitog rješenja proučeno je nekoliko Open source rješenja poput Shibboleth. Razlozi zbog kojih je odlučeno da se krene u implementaciju vlastitog rješenja su višestruki:

1. **Kompleksnost postojećih sustava**
   Postojeći sustavi su dizajnirani da se mogu implementirati u velikom rasponu situacija što ih čini previše kompleksnim za naše potrebe. Ujedno njihova kompleksnost zahtjeva značajne promjene u autentifikacijskim mehanizmima postojećih aplikacija te povećava složenost njihove prilagodbe specifičnim potrebama.
2. **Tehnologija**
   Tokom pretraživanja postojećih rješenja nije pronađeno zadovoljavajuće rješenje izvedeno samo u PHP-u. Naime svi kvalitetni SSO sustavi baziraju se većinom na Java platformi. Naravno da je problem interoperabilnosti Java i PHP aplikacija zanemariv ali jedan od naših zahtjeva je bio da se naše rješenje može implementirati u okviru standardnih hosting paketa što onda isključuje upotrebu Java aplikacija.

Zbog ovih razloga smo pristupili izradi vlastitog SSO rješenja koje će biti jednostavno, lako za implementaciju te prilagodljivo specifičnim potrebama.

Arhitektura sustava SLEASY

Kao što je već napomenuto naš sustav se bazira na:

- Apache Web poslužitelju
- PHP programskom jeziku (OpenSSL)
- LDAP-u
- PostgreSQL/MySQL bazi podataka

Jedan od najvećih problema kod implementacije SSO rješenja je ovisnost o centralnoj SSO aplikaciji. Ukoliko dođe do pada tog sustava korisniku postaju nedostupne sve aplikacije unutar SSO domene. Da bi spriječili ovisnost o centralnoj aplikaciji jedan od zahtjeva na naš sustav je bio da aplikacije i dalje mogu koristiti svoju standardnu autentifikacijsku proceduru ukoliko dođe do pada SSO sustava.

Osnovni princip rada:

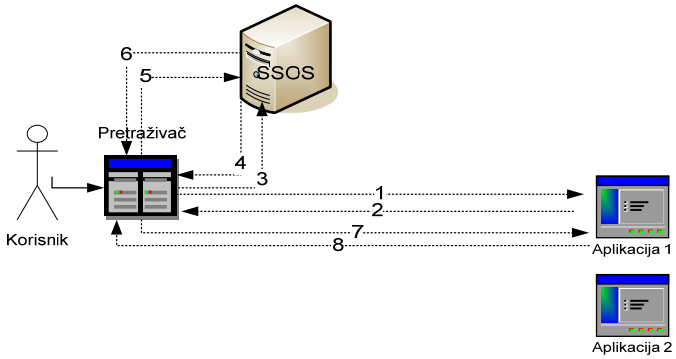
a) Prva prijava u jednu od aplikacija u SSO sustavu

Pristup prvoj aplikaciji u nizu: dakle, korisnik do sada nije pristupao niti jednoj od aplikacija u SSO sustavu, tako da ga se mora autentifcirati na SSO poslužitelju:

Koraci:

1. Korisnik zahtijeva pristup aplikaciji
2. Aplikacija 1 usmjerava Web preglednik na SSO poslužitelj zajedno s zahtjevom za autentifikaciju kriptiranim na sljedeći način:
   $K_s^+(K_a^-(\text{poruka}))$, gdje je K_s^+ javni ključ SSO poslužitelja, a K_a^- privatni ključ aplikacije 1.

3. Web preglednik pristupa SSO poslužitelju



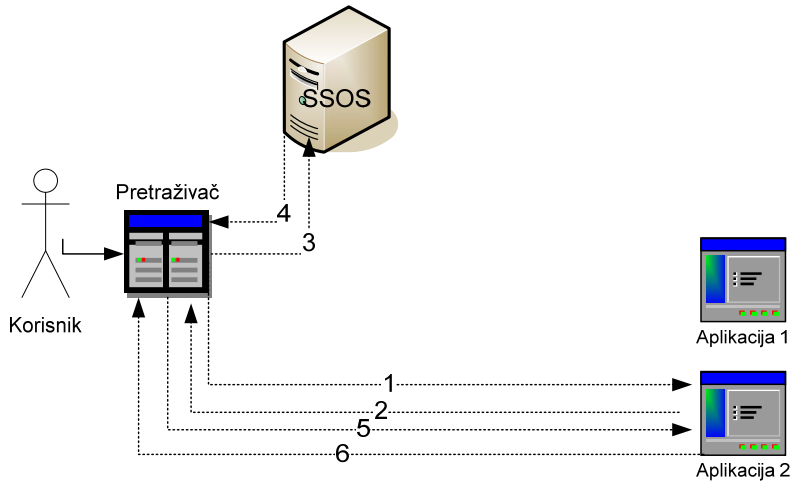
4. Budući da korisnik nije prijavljen, SSO poslužitelj dekriptira poruku te Web pregledniku prosljeđuje formu za prijavu
5. Korisnik unosi svoje korisničko ime i lozinku u formu za prijavu
6. Nakon uspješne autentifikacije, SSO poslužitelj pokreće redirekciju natrag na aplikaciju 1 te prosljeđuje poruku koja je kriptirana na slijedeći način:

$K_a^+(K_s^-(\text{poruka}))$, gdje je K_a^+ javni ključ aplikacije 1, a K_s^- privatni ključ SSO servera. Poruka sadrži podatke o uspješnoj prijavi korisnika.

7. Web preglednik proslijeđuje kriptiranu poruku aplikaciji 1.
8. Nakon što je dekriptirala poruku, aplikacija 1 potvrđuje da je korisnik uspješno autentificiran, te korisniku šalje željeni resurs.

<u>b) Naknadne prijave na jednu od aplikacija u SSO sustavu</u>

Prijave u jednu od aplikacija nakon što se korisnik autentificirao na SSO poslužitelju, odnosno nakon što je već prije pristupio jednoj od aplikacija:



Koraci:

1. Korisnik zahtijeva pristup aplikaciji 2
2. Aplikacija 2 usmjerava Web preglednik na SSO poslužitelj zajedno s zahtjevom za autentifikaciju kriptiranim na sljedeći način:

$K_s^+(K_a^-(\text{poruka}))$, gdje je K_s^+ javni ključ SSO poslužitelja, a K_a^- privatni ključ aplikacije 2.

3. Pretraživač pristupa SSO poslužitelju
4. Budući je korisnik već prije prijavljen, SSO poslužitelj dekriptira poruku, pokreče redirekciju natrag na aplikaciju 2 te prosljeđuje poruku koja je kriptirana na slijedeći način:

   $K_a^+(K_s^-(\text{poruka}))$, gdje je K_a^+ javni ključ aplikacije 2, a K_s^- privatni ključ SSO poslužitelja.

5. Web preglednik prosljeđuje kriptiranu poruku aplikaciji 2.
6. Nakon što je dekriptirala poruku, aplikacija 2 potvrđuje da je korisnik uspješno autentificiran, te korisniku šalje željeni resurs.