



UNIVERZITET SINGIDUNUM

FAKULTET ZA INFORMATIKU I RAČUNARSTVO

Autentifikacija i autorizacija klijenta u sistemimaza rad sa više servisa

- Diplomski rad -

Beograd, 2015.

FAKULTET ZA INFORMATIKU I RAČUNARSTVO

UNIVERZITET SINGIDUNUM

FAKULTET ZA INFORMATIKU I RAČUNARSTVO

Beograd, Danijelova 32

Kandidat: Dalibor Marijančević

Broj indeksa: 2011/200933

Smer: Informatika i računarstvo

Tema: Autentifikacija i autorizacija klijenta u sistemima zarad sa više servisa

MENTOR

Datum odobrenja rada:
Beograd, __. __. ____.

Doc. dr Saša Adamović

DEKAN

Prof. dr Dragan Cvetković

Autentifikacija i autorizacija klijenta u sistemima za rad sa više servisa

Sažetak: U ovom radu bavićemo se razdvajanjem servisa autentifikacije od servisa autorizacije. Prednosti ovog pristupa se ogledaju u boljim performansama i jednostavnijem sprečavanju dobro poznatih napada na mreži (Denial of Service). Zbog nepostojanja dvosmerne veze između ova dva servisa, servise je moguće nezavisno implementirati. Administratori autentifikacionih servera konfigurišu željeni nivo zaštite preko konfiguracionih parametara koji se odnose na ključeve, sertifikate, kriptološke algoritme i mrežne parametre, dok autorizacioni server koristi skup pravila definisanih od strane autentifikacionog servera i sam definiše parametre zasebno za svaki servis, koji se odnose na autorizaciju korisnika. Razvojem ovakvog dizajna sistema autentifikacije i autorizacije, korisniku servisa obezbeđen je bezbedan pristup servisima u lokalnoj mreži ili servisima na Internetu.

Ključne reči: autentifikacija, autorizacija, veb servis

Client Authentication and Authorisation for Systems with Multi-Service Support

Abstract: This paper discusses a separation of service of authentication from the service of authorisation. Advantages to this approach are better performance and easier prevention of known attacks, such as Denial of Service. The services can be implemented independently due to a lack of two-way connection in between. The authentication server administrator (pre)sets/(pre)defines a required level of security through configuration parameters which are keys, certificates, cryptological algorithms and network. The authorisation server then uses these rules to define its own behaviour with regard to each service pertaining to user authorisation. Such a design of authentication and authorisation services provides users with a secure access to local or web services.

Keywords: authentication, authorisation, webservice

SADRŽAJ

1. Uvod	- 4 -
1.1. Predmet istraživanja	- 4 -
1.2. Doprinos rada	- 4 -
1.3. Struktura rada	- 5 -
2. Pregled u oblasti istraživanja	- 6 -
2.1. Opis problema	- 6 -
2.2. Autorizacioni okvir OAuth 2.0	- 6 -
2.2.1. Tip dodele prava	- 7 -
2.2.2. Tip klijenta	- 7 -
2.2.3. Realizovanje autorizacije	- 8 -
2.2.4. Izdavanje pristupnog tokena	- 10 -
2.2.5. OS aplikacije (<i>native applications</i>)	- 10 -
2.2.6. Bezbednost	- 11 -
2.2.6.1. Autentifikacija	- 11 -
2.2.6.2. Lažiranje klijenata	- 11 -
2.2.6.3. <i>Man in the middle attack</i>	- 12 -
2.2.6.4. <i>Phishing</i> napad	- 12 -
3. Teorijske osnove	- 13 -
3.1. Kriptografski protokoli	- 13 -
3.1.1. <i>SSH – Secure Shell</i>	- 13 -
3.1.2. <i>IPSec – IP Security</i>	- 14 -
3.1.3. Kerberos	- 15 -
3.2. Kriptografski servisi	- 15 -
3.3. Razmena ključa	- 16 -
3.3.1. <i>Diffie–Hellman</i> protokol	- 17 -
3.3.2. Razmena ključa na fizičkom sloju	- 17 -
3.4. <i>SSL</i> protokol	- 17 -
4. Pregled predloženog rešenja	- 19 -
4.1. Učesnici u protokolu	- 19 -
4.1.1. Provajder	- 19 -

4.1.2.	Klijent.....	- 20 -
4.1.3.	Krajnji korisnik.....	- 20 -
4.2.	Algoritmi u protokolu	- 21 -
4.3.	Podaci u protokolu	- 21 -
4.3.1.	Kriptološki parametri	- 21 -
4.3.2.	Parametri izvora zahteva i prosleđivanja odgovora	- 22 -
4.3.3.	Autentifikacioni i autorizacioni parametri	- 22 -
4.4.	Šema protokola	- 22 -
4.4.1.	Diskusija o pojedinim koracima protokola.....	- 26 -
4.4.1.1.	Generisanje izazova – Korak 2.....	- 26 -
4.4.1.2.	Kreiranje sesije i beleženje izazova – Korak 3	- 27 -
4.4.1.3.	Provera domena iz zahteva – Korak 7	- 27 -
4.4.1.4.	Preuzimanje izazova – Korak 8	- 27 -
4.4.1.5.	Provera korisnika – Korak 12.....	- 27 -
4.4.1.6.	Odgovor provajdera – Korak 18	- 28 -
4.5.	Odnos između učesnika u protokolu	- 28 -
4.5.1.	Kako provajder posmatra klijenta i krajnjeg korisnika.....	- 28 -
4.5.1.1.	Tip klijenta	- 28 -
4.5.1.2.	Rangiranje klijenta	- 28 -
4.5.1.3.	<i>Krajnji korisnik</i>	- 29 -
4.5.2.	Kako klijent posmatra provajdera	- 30 -
4.6.	Napadi.....	- 30 -
4.6.1.	Napad ponavljanjem poruke.....	- 30 -
4.6.2.	Napad preusmeravanja.....	- 31 -
4.6.3.	Preuzimanje tokena	- 32 -
4.6.4.	Lažiranje zahteva	- 32 -
4.6.4.1.	<i>Funkcionalnosti koje obrađuje klijent</i>	- 33 -
4.6.4.2.	<i>Funkcionalnosti koje obrađuje provajder sa nižim nivoom bezbednosti..</i> -	33 -
4.6.4.3.	<i>Funkcionalnosti koje obrađuje provajder sa visokim nivoom bezbednosti</i>	- 34 -
5.	Zaključak	- 34 -
5.1.	Sumarni ciljevi rada i ostvareni rezultati.....	- 34 -

5.2. Predlog za budući rad	- 35 -
Dodatak – programski kodovi	- 36 -
LITERATURA.....	- 40 -

Spisak slika

Slika 1 - Implicitni tok.....	- 9 -
Slika 2 - Šema protokola	- 23 -
Slika 3 - Log izvršavanja protokola	- 24 -
Slika 4 - Provera kredencijala i kreiranje parametara zahteva kod provajdera.....	- 36 -
Slika 5 - Provera parametara zahteva kod klijenta.....	- 37 -
Slika 6 - Preuzimanje ključeva, kodovanje i šifrovanje podataka	- 38 -
Slika 7 - Digitalno potpisivanje i verifikacija potpisa	- 39 -

Spisak tabela

Tabela 1 - Koraci u izvršavanju protokola	- 26 -
---	--------

1. Uvod

1.1. Predmet istraživanja

Komunikacija, odnosno razmena informacija nezaobilazan je proces koji jedinke povezuje u društvo. Nekada su to javne poruke koje je potrebno da čuje što više ljudi, a nekada poverljivi razgovori ili parametri na osnovu kojih se donose važne odluke. To dalje podrazumeva da u zavisnosti od stepena otvorenosti poruke zavisi stepen zaštite kanala preko koga se prenosi i da li će informacija koju sadrži biti uspešno upotrebljena. Tako nastaju različiti protokoli za različite potrebe otvorenosti poruka.

S obzirom na to da se razmena informacija uglavnom odigrava na internetu, ovaj rad posvećen je bezbednosnim problemima u tom vidu komunikacije.

Pregledom u oblasti istraživanja utvrđeni su učesnici i način uređivanja njihove komunikacije. Protokol koji je tema rada detaljnije razlikuje funkcionalnosti koje mogu biti potrebne u komunikaciji. Takođe, kao podrška protokolu biće implementirano sopstveno rešenje servisa za autentifikaciju i autorizaciju. Istovremeno, rešenje predstavlja i skup obaveznih koraka i minimum zahteva saobraćaja i resursa koje protokol propisuje.

1.2. Doprinos rada

Pored sopstvenog rešenja koje može služiti kao polazna tačka u razvoju sopstvenog sistema za zaštitu informacija, doprinos rada je i ukazivanje na osetljive tačke iz standardnih rešenja i razvoj svesti kako kod proizvođača, tako i kod korisnika aplikacija i usluga da internet nije bezbedan kanal komunikacije i da moraju da budu oprezni u radu. Posebna pažnja mora biti posvećena krajnjim korisnicima i potrebno ih je upoznati sa svim kritičnim tačkama.

- Daćemo pregled klijent – serverarhitekture kod servisa za autentifikaciju i autorizaciju, gde se servisi izvršavaju u serverskoj aplikaciji, a klijent je aplikacija koja koristi uslugu navedenih servisa.
- Ukazaćemo na bezbednosne probleme u veb okruženju, kao i napade koji iz toga proizilaze.
- Dizajniraćemo sopstveni protokol za zaštitu informacija pri korišćenju servisa autentifikacije i autorizacije.
- Protokol će podržavati klijente koji koriste više servisa.

-
- Biće predstavljen novi servis koji sadrži potpuno nove elemente ugrađene po protokolu koji predviđa manje slobode klijentskim servisima, a sve u cilju zaštite korisničkih informacija i resursa.
 - Predstavićemo i programsku implementaciju protokola sa najvažnijim delovima koda.

1.3. Struktura rada

U prvom poglavlju upoznaćemo se sa predmetom istraživanja rada. Ovo poglavlje obuhvata i ciljeve i strukture rada.

Drugo poglavlje donosi pregled u oblasti istraživanja i u njemu je predstavljen okvir koji će poslužiti kao osnova za većinu rešenja u upotrebi. Okvir čine potrebni parametri i poznati napadi, kao i neophodni koraci za sprečavanje istih.

Treće poglavlje o teorijskim osnovama predstavlja protokole koji, samostalno ili u celosti, mogu biti upotrebljeni u realizaciji dela procesa autentifikacije i autorizacije. Posebno su predstavljeni protokoli za razmenu ključeva.

U četvrtom poglavlju prikazan je pregled predloženog rešenja, učesnici i šema predloženog protokola. U zavisnosti od željenih funkcionalnosti, biće predstavljeni obavezni koraci ili resursi koje treba dodati na osnovnu verziju protokola. Na kraju poglavlja razmotrićemo načine kojima protokol sprečava neke poznate napade.

U petom poglavlju izvedeni su zaključci na osnovu sumiranih rezultata i ostvarenih ciljeva u radu. Na kraju poglavlja daćemo predloge za buduće istraživanje.

Na kraju se nalazi dodatak sa prikazanim kodovima programskog rešenja. Kodovi predstavljaju najvažnije korake u protokolu koji se odnose na zaštitu podataka.

2. Pregled u oblasti istraživanja

2.1. Opis problema

Problem autentifikacije i autorizacije nastao je sa prvom organizacijom koja je funkcionisala na hijerarhijskom principu, kada je bilo potrebno obezbediti ko može, a ko ne možeda prisustvuje sastancima, kao i koja prava ima, da li može da govori ili samo da sluša i da li može da učestvuje u odlučivanju ili ne. Načini provere su se menjali, ali suština je i danas ostala ista. Kako biti siguran da li je određenom resursu pristupio samo onaj kome je to dozvoljeno i da li raspolaže pravima koja su mu odobrena?

Sa razvojem računarskih mreža, postalo je neophodno da utvrdimo ko je korisnik koji želi da pristupi zaštićenom servisu[1]. Nekoliko velikih svetskih tehnoloških kompanija koje su razvile servise autentifikacije za svoje potrebe, razvile su module preko kojih i proizvođači servisa mogu da autentifikuju svoje korisnike. Većina rešenja se oslanja na autorizacioni okvir OAuth 2.0.

2.2. Autorizacioni okvir OAuth 2.0

Ovaj okvir naslednik je OAuth 1.0 koji je bio informacioni dokument. Iako dele neke osnovne ideje i načela, OAuth 2.0 nije kompatibilan sa svojim prethodnikom[2].

Obično su za korišćenje nekog resursa potrebni neki parametri. To mogu biti korisničko ime ili lozinka, a ponekad i sama putanja. Razvojem veba pojavili su se alati i aplikacije koji bolje ili samo drugačije manipulišu sa onim što je napravljeno u osnovnom alatu. Proizvođači alata (naročito oni koji daju besplatne usluge) rado dozvoljavaju da druge aplikacije koriste njihove resurse pošto to povećava i funkcionalnost njihovog proizvoda.

Tu nastaje problem pristupa treće strane nekom zaštićenom resursu. Dakle, potrebno je dozvoliti da treća strana može da pristupi, ali da ne dobije parametre preko kojih korisnik inače dobija pristup. Autorizacioni okvir u tu svrhu uvodi pristupni akreditiv (*access token*). U zavisnosti od tipa dodele prava mogu se pojaviti još neki tipovi tokena kao što je *refresh token* koji služi za produžavanje pristupnog tokena.

Da bi se jasno razdvojilo ko zahteva, ko odobrava, a ko izdaje tokene definisane su četiri uloge u procesu:

- vlasnik resursa – može biti bilo ko ko treba da odobri pristup resursu; može biti i osoba, i tada se naziva krajnji korisnik,

-
- server na kome se nalazi resurs – server koji treba da prihvati token i da, shodno značenju koje sadrži, odgovori isporukom resursa,
 - klijent – aplikacija koja u ime i sa odobrenjem vlasnika resursa postavlja zahtev za pristup tom resursu; to može biti bilo koji tip aplikacije (serverska, desktop, veb ili bilo koja druga),
 - autorizacioni server – server koji nakon autentifikacije svih strana u procesu izdaje pristupni token klijentu.

2.2.1. Tip dodele prava

Dodela prava je akreditiv vlasnika resursa na osnovu koga autorizacioni server izdaje klijentu pristupni token. OAuth 2.0 prepoznaje četiri vrste dodele: autorizacioni kôd, implicitna dodela, lozinka vlasnika resursa i akreditivi klijenta, kao i dodatke koji omogućavaju razvoj dodatnih tipova.

U ovom radu bavićemo se veb i OS aplikacijama koje pristupaju veb serveru. Za veb aplikacije obavezno je korišćenje implicitne dodele (engl. *Implicit Grant*), a sam režim se naziva implicitni tok (engl. *Implicit Flow*).

Implicitni tok se primenjuje za krajnje korisnike koji upotrebljavaju veb pregledače i posledično skript jezike. Kod implicitne dodele izdaje se samo pristupni token. Ovo je pojednostavljen pristup s obzirom na to da klijent odmah dobija pristupne podatke, ali to uprošćenje proizvodi nekoliko negativnih aspekata. Token može biti izložen klijentu ili drugoj aplikaciji koja ima pristup pregledaču, tako da je obavezna upotreba *TLS* protokola, što smanjuje inicijalno dobre performanse.

2.2.2. Tip klijenta

Pri registraciji kod autorizacionog servera klijent mora da definiše:

- tip klijenta,
- jedinstveni identifikator resursa za redirekciju (*redirection URI*) potreban posle autorizovanja od strane vlasnika resursa,
- sve podatke koje je autorizacioni server propisao (naziv aplikacije, veb-sajt, logo i sl.)

OAuth 2.0 u odnosu na mogućnost klijenta da se bezbedno autentifikuje kod autorizacionog servera (ovo se odnosi na sposobnost da održava poverljivost klijentskih kredencijala) definiše dva tipa:

- poverljivi – klijent sposoban da održi poverljivost,

-
- javni – klijent koji nije u mogućnosti da garantuje poverljivost (izvršava se na uređaju vlasnika resursa, recimo OS aplikacija ili veb aplikacija koja se izvršava u pregledaču).

Ova specifikacija tipova je razvijena za sledeće profile klijenata:

- veb aplikacija – poverljivi klijent koji se izvršava na veb serveru; klijentovi akreditivi ostaju na serveru i vlasnik resursa ne može da dođe do njih,
- korisnička veb aplikacija – javni klijent koji se izvršava na računaru vlasnika resursa u pregledaču, tako da su podaci o klijentu dostupni vlasniku resursa,
- OS aplikacija (*native application*) – javni klijent koji se izvršava na računaru krajnjeg korisnika i smatra se da može da pristupi svim podacima o klijentu.

Pri registraciji autorizacioni server izdaje klijentu identifikator koji je jedinstven i koji odslikava podatke koje je klijent prosledio pri zahtevu za registraciju. Podatak je javan i ne sme se koristiti kao jedini faktor autentifikacije. Takođe, autorizacioni server mora da propiše dužinu niza znakova koji predstavlja identifikator.

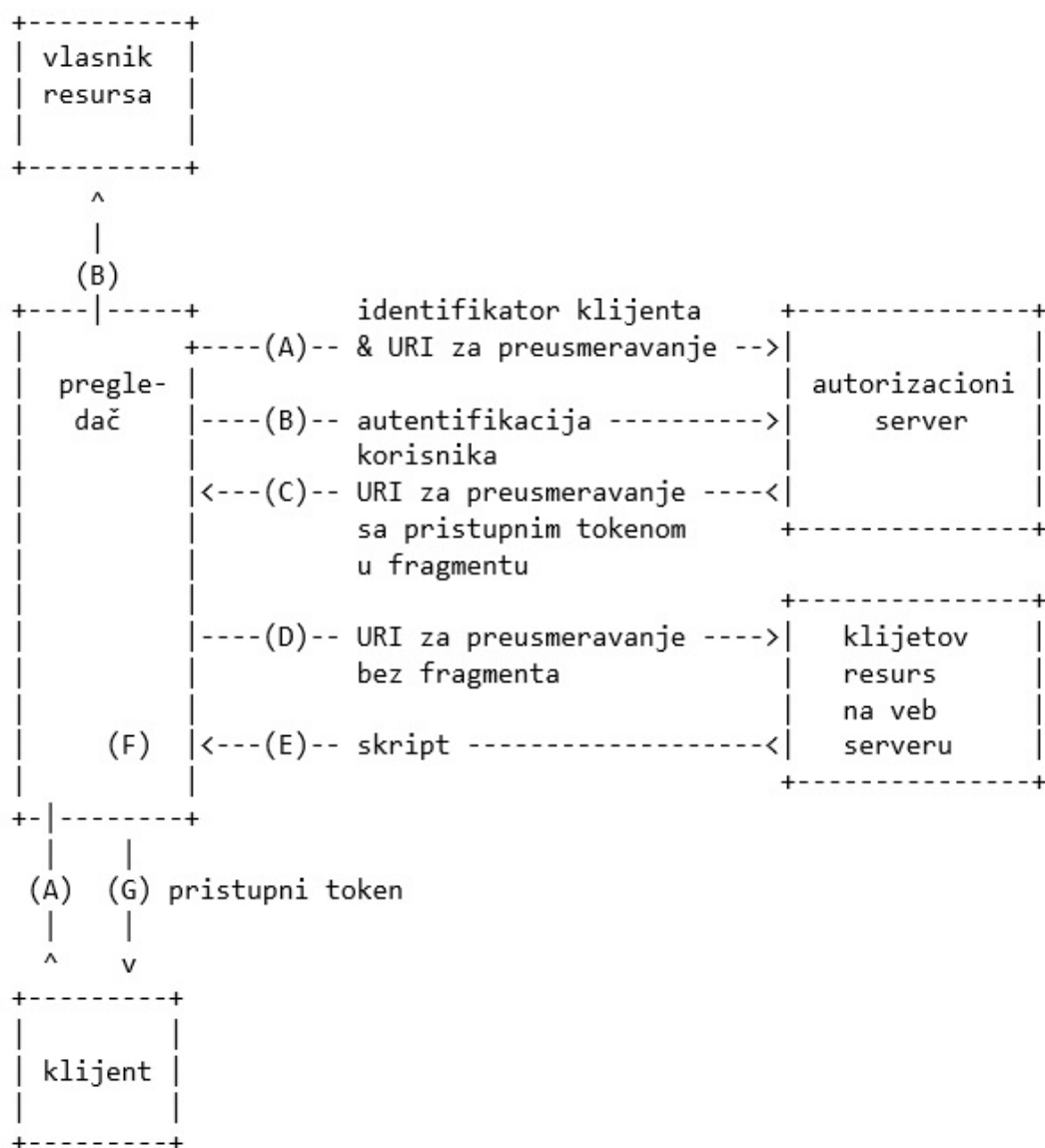
2.2.3. Realizovanje autorizacije

Implicitni tok koristi se za dobijanje pristupnog tokena i ne podržava izdavanje tokena za produžavanje (*refresh token*). Optimizovan je za javne klijente koji su obično implementirani u pregledaču i koriste skript jezike kao što je *JavaScript*.

Pošto je ovo tok zasnovan na preusmeravanju, klijent mora biti u mogućnosti da komunicira sa pregledačem i da preko redirekcije prima zahteve od autorizacionog servera.

- A. Klijent započinje tok upućivanjem korisničkog agenta vlasnika resursa na autorizacionu tačku. Klijent dodaje svoj identifikator i *URI* za preusmeravanje na koji će autorizacioni server poslati korisnika kada autorizacija bude dobijena.
- B. Autorizacioni server autentifikuje vlasnika resursa i utvrđuje da li vlasnik dozvoljava ili odbija klijentov zahtev za pristup.
- C. Ako je pristup odobren, autorizacioni server preusmerava korisnika nazad do klijenta koristeći *URI* koji je dostavljen ranije. *URI* za preusmeravanje uključuje pristupni token u *URI* fragmentu.
- D. Korisnički agent prati korake za preusmeravanje i upućuje zahtev veb serveru gde se nalazi resurs. Ovaj zahtev ne uključuje fragment pošto ga korisnički agent zadržava kod sebe.

- E. Veb server vraća skript potreban za pristup adresi resursa i računanje pristupnog tokena i ostalih parametara koji se nalaze u fragmentu.
- F. Korisnički agent izvršava skript i dobija pristupni token.
- G. Korisnički agent prosleđuje token klijentu.



Slika 1- Implicitni tok

2.2.4. Izdavanje pristupnog tokena

Ako je zahtev za pristupnim tokenom validan i autorizovan, autorizacioni server izdaje pristupni token. Odgovor se šalje u *HTTP*-usa statusnim kodom 200 (OK) u *JSON* formatu i sadrži sledeće parametre:

- pristupni token – obavezan,
- tip tokena – obavezan – klijent ne sme da koristi token ako ne razume koji je tip i čemu služi,
- trajanje – preporučeni parametar – vreme trajanja tokena; ako ne postoji, trebalo bi da autorizacioni server obezbedi podrazumevanu vrednost kroz dokumentaciju ili na neki drugi način,
- opseg – opcioni – ako se opseg poklapa sa onim koji je zahtevao klijent, ako ne, onda je obavezan parametar.

2.2.5. OS aplikacije (*native applications*)

OS aplikacije izvršavaju se na uređaju vlasnika resursa i zahtevaju posebno razmatranje u vezi sa bezbednošću, mogućnostima platforme i korisničkim iskustvom.

Autorizacija zahteva komunikaciju između klijenta i korisničkog agenta vlasnika resursa. OS aplikacije mogu da pozovu spoljnog korisničkog agenta ili nekog koji je ugrađen u samu aplikaciju.

- spoljni korisnički agent – OS aplikacija može da: a) prihvati odgovor od autorizacionog servera koristeći *URI* za preusmeravanje preko šeme koja je ugrađena u operativni sistem, b) kopira ručno kredencijale ilic) instalira ekstenziju itd.
- ugrađeni korisnički agenti – OS aplikacija uzima odgovor direktno iz komunikacije sa ugrađenim korisničkim agentom prateći promenu stanja ili pristupajući kolačiću korisničkog agenta.

Prilikom biranja između ugrađenog ili spoljnog agenta trebalo bi razmotriti sledeće:

- spoljni agent bi mogao da ubrza proces tako što bi za već aktivnu sesiju korisnika završio i proces autentifikacije. To obezbeđuje ugodno korisničko iskustvo i funkcionalnost. Vlasnik resursa bi takođe mogao da se osloni na razne mogućnosti kao što je funkcionalnost za upravljanje lozinkama.
- ugrađeni korisnički agent nudi pogodnost da se ne mora menjati kontekst i otvaraju novi prozori.

-
- ugrađeni korisnički agent predstavlja izazov za bezbednost pošto se vlasnik resursa autentifikuje u nepoznatom prozoru bez vizuelne zaštite koja postoji u spoljnom korisničkom agentu. Ugrađeni korisnički agent navikava krajnjeg korisnika da treba da veruje nepoznatim zahtevima za autentifikaciju što omogućava lakše izvršavanje tzv. *phishing* napada.

2.2.6. Bezbednost

S obzirom na to da bezbednost zavisi od mnogih faktora postoji nekoliko aspekata na koje treba obratiti pažnju.

2.2.6.1. Autentifikacija

Preporučljivo je da se autorizacioni server ne oslanja samo na lozinku pri autentifikaciji klijenta. Klijentska veb aplikacija mora da osigura poverljivost klijentove lozinke i drugih akreditiva.

Autorizacioni server ne sme da dozvoli vidljivost klijentove lozinke ili drugih kredencijala OS aplikaciji ili bilo kom drugom korisničkom agentu. Odstupanje od ovog dozvoljeno je samo u slučaju posebne instalacije OS aplikacije na posebnom uređaju.

Kada autentifikacija klijenta nije moguća, autorizacioni server bi trebalo da uključi druge načine autentifikacije kao što je *URI* za preusmeravanje, iako to nije dovoljno za njenu verodostojnost.

2.2.6.2. Lažiranje klijenata

Autorizacioni server mora da autentifikuje klijenta kad god je to moguće. Ako priroda klijenta to ne omogućava, autorizacioni server mora da zahteva registraciju svakog *URI*-ja za preusmeravanje koji služi za korišćenje zaštićenog resursa. Jedan od načina je da autorizacioni server traži pomoć od vlasnika resursa radi autentifikovanja klijenta i njegovog porekla.

Autorizacioni server trebalobi eksplicitno da traži autentifikaciju od vlasnika resursa i da mu obezbedi informacije od klijentao zahtevanom opsegu i veku trajanja zahteva. Sam vlasnik resursa trebalo bi potom da pregleda date podatke i odluči da li će autorizovati zahtev konkretnog klijenta.

Autorizacioni server ne bi trebalo da izvršava ponovljene zahteve automatski bez saglasnosti vlasnika resursa. Da bi se utvrdilo da li je ponovljeni zahtev upućen od stvarnog klijenta, potrebna je njegova autentifikacija ili neki drugi način utvrđivanja identiteta.

2.2.6.3. *Man in the middle attack*

Da bi se sprečio ovaj napad, autorizacioni server mora da zahteva bezbednu konekciju korišćenjem TLS (*Transport Layer Security*) protokola za svaki zahtev za autorizaciju. Klijent mora da proveri TLS sertifikat autorizacionog servera.

2.2.6.4. *Phishing* napad

Tokom vremena krajnji korisnici korišćenjem ovakvih usluga navikavaju se da budu preusmeravani na sajtove gde im se zahteva unos korisničkog imena i lozinke. Ako nisu pažljivi, mogu postati žrtva napada.

Servis koji pruža usluge trebalo bi da upozna korisnika sa rizicima i mogućnošću napada i da mu obezbedi mehanizme preko kojih će jednostavno utvrditi autentičnost sajtova.

3. Teorijske osnove

Postupak zaštite informacionih sistema podrazumeva definisanje nekoliko parametara. To su kriptografski protokol, različiti kriptografski algoritmi koji će se koristiti u okviru protokola, način implementacije protokola. Sam protokol, kao i kriptografski algoritam uglavnom su javno dostupne informacije s obzirom na to da se sigurnost sistema oslanja na jačinu i tajnost ključa, kako i govori Kerkhofov princip.

Polazeći od ove činjenice jasno je da je najvažnija uloga protokola da obezbedi upravo te dve stvari, tj. da se generiše ključ koji je zasnovan na nekom slučajnom prirodnom izvoru i da se takav razmeni i čuva na što bezbedniji način. Takođe, određeni protokol može sadržati druge protokole.

3.1. Kriptografski protokoli

Postoji nekoliko protokola koji imaju široku primenu u savremenim komunikacijama. Neki su definisani tako da mogu da se koriste u bilo kom načinu komunikacije, a neki kao što su *WEPI* ili *WPA2* samo za bežične komunikacije. Neki od protokola su: *SSH*, *IPSec*, *Kerberos* i *SSL/TLS*.

3.1.1. *SSH – Secure Shell*

Zadatak *SSH* protokola jeste da obezbedi sigurnosni tunel između strana u komunikaciji. Pod tunelom se podrazumeva komunikacija u kojoj su poruke zaštićene [servisima](#) tajnosti i integriteta[3].

S obzirom na to da se za razmenu simetričnog ključa koristi [Diffie – Hellman](#) protokol, strane u komunikaciji moraju međusobno da se autentifikuju. U tu svrhu može da se upotrebi lozinka, sistem javnih ključeva ili digitalni sertifikati.

Protokol sadrži nekoliko koraka:

- jedna od strana šalje poruku koja sadrži njen identitet, kriptografske parametre i izazov,
- druga strana bira ponuđene parametre i šalje ih zajedno sa svojim izazovom,
- prva strana generiše i šalje svoje *Diffie – Hellman* parametre,
- druga strana generiše svoje *Diffie – Hellman* parametre i šalje ih sa svojim sertifikatom i digitalnim potpisom svih dotadašnjih poruka; takođe, sada poseduje sve podatke za računanje simetričnog ključa za dalju komunikaciju,

-
- prva strana dobija sve parametre za računanje simetričnog ključa i šalje svoj sertifikat i digitalni potpis do tada razmenjenih poruka,
 - druga strana verifikuje potpis i ako je u redu, dalja komunikacija može da se nastavi sa izračunatim kriptološkim ključem.

3.1.2. IPSec – IP Security

IPSec [4] protokol radi na sloju mreže *OSI* modela i kao takav je deo operativnog sistema. Najčešće se koristi za obezbeđivanje zaštićene tunelovane komunikacije u virtuelnim privatnim mrežama[4].

IPSec se sastoji iz dva dela: *IKE (Internet Key Exchange)* i *ESP/AH (Encapsulating Security Payload and Authentication Header)*.

IKE deo protokola se koristi za obezbeđivanje uzajamne autentifikacije i sesijskih ključeva.

ESP deo se koristi za šifrovanje i zaštitu integriteta *IP* paketa, a *AH* obezbeđuje samo integritet.

IPSec protokol ne zahteva da su prethodno razmenjeni digitalni sertifikati učesnika u komunikaciji. Postoje dva režima rada prve faze *IKE* protokola: glavni i agresivni.

Namena glavnog režima je da obezbedi anonimnost, ali uz dvostruko veće korišćenje resursa u odnosu na agresivni režim koji ne vodi računa o tom delu.

U glavnom režimu rada prvo se razmenjuju kriptološki parametri, zatim izazovi i *Diffie – Hellman* parametri i na kraju digitalni potpisi svih do tada razmenjenih poruka.

Agresivni režim kriptološke i *Diffie – Hellman* parametre, kao i izazove, razmenjuje odjednom, a potom digitalne potpise ostvarene komunikacije.

Nakon završetka prve faze, pristupa se računanju simetričnog ključa. I tu postoje ista dva režima rada kao i u prvoj fazi.

U ovoj fazi agresivni režim pokazao se kao bolje rešenje, s obzirom na to da se anonimnost glavnog režima zasniva na *IP* adresi koja pri tome mora da bude tajna i statička. Shodno postavci da agresivni režim ne obezbeđuje anonimnost, isti efekat zaštite postiže se kao i kod glavnog, ali su performanse znatno bolje.

Drugi deo *IPSec* protokola odnosi se na saobraćaj nakon razmenjenog ključa. Tu postoje dva moguća režima: transportni i režim tunelovanja.

Transportni režim postavlja novo *ESP/AH* zaglavlje između *IP* zaglavlja i podataka.

Režim tunelovanja enkapsulira ceo *IP* paket u novi *IP* paket.

AH tip zaglavlja koristi se samo za integritet, ali ne podržava šifrovanje.

ESP tip zaglavlja podržava i integritet i šifrovanje, ali se primenjuje samo na deo paketa koji se odnosi na podatke izvan *IP* zaglavlja.

3.1.3. Kerberos

Protokol Kerberos zasnovan je na trećoj strani od poverenja i koristi se u lokalnim mrežama. U ovom protokolu sva komunikacija ide preko servera koji za svakog učesnika ima poseban ključ [5].

Prizapočinjanju komunikacije server izdaje glavne tikete svim korisnicima koje oni posle koriste u zahtevu za pristup određenim resursima. Svi tiketi su šifrovani sa jedinstvenim ključem koji je poznat samo serveru. Takođe, tiketi su vremenski ograničeni, tako da je potrebno implementirati i *NTP (Network Time Protocol)*.

S obzirom na to da se u protokolu ne koriste sertifikati, inicijalni ključ generiše se na osnovu klijentove lozinke korišćenjem neke jednosmerne funkcije. Server kreira sesijski ključ, a korisnik na osnovu inicijalnog ključa računa taj sesijski ključ i glavni tiket. Kada korisnik dobije glavni tiket, može ga koristiti za naredne zahteve ka resursima.

3.2. Kriptografski servisi

Osnovni servisi koje treba da obezbedi većina kriptografskih protokola su:

- servis tajnosti,
- servis integriteta,
- servis neporecivosti,
- servis autentifikacije.

Većina kriptografskih protokola ima zadatak da obezbedi servis tajnosti. Pod servisom tajnosti podrazumevamo upotrebu šifarskih algoritama između dve strane koje koriste protokol. Servis tajnosti može da se zasniva na simetričnim i asimetričnim šiframa. U odnosu na tip izabrane šifre, vrši se i odabir kriptografskih protokola koji su zaduženi za razmenu tajnog ključa ili digitalnog sertifikata. Servis tajnosti zasnovan na simetričnim šifarskim algoritmima je nekoliko puta efikasniji od asimetričnog šifarskog sistema. Iz tog razloga se asimetrični algoritmi koriste na početku protokola, u inicijalnoj fazi za razmenu tajnog ključa koji će kasnije poslužiti kao osnova za generisanje sesijskih ključeva preko *Diffie – Hellman* protokola [6].

Servis integriteta obavezan je za sve protokole. Uobičajena praksa pre projektovanja sistema je formiranje zahteva kako sistem treba da se ponaša. Ti zahtevi sadrže listu servisa koji treba da budu podržani. Servisi mogu biti različiti, ali gotovo da ne postoji situacija gde na toj listi ne bi stajao servis integriteta. Cilj komunikacije je prenos poruka. Ako postoji način da neko u toku transfera promeni neki deo ili zameni ceo sadržaj, tada osnovni cilj nije postignut. Servis integriteta implementira se korišćenjem jednosmernih funkcija i asimetrične kriptografije. Prvo se izabere odgovarajuća jednosmerna funkcija, a zatim se rezultat digitalno potpisuje. Primalac poruke će upotrebiti istu funkciju i posle verifikacije potpisa, moći će da poređenjem rezultata utvrdi da li je poruka izmenjena.

Servis neporecivosti naslanja se na servis integriteta. Ako smo sigurni da je integritet očuvan i da poruka nije menjana, sledeći zahtev bio bi da pošiljalac ne može da opovrgne da je on poslao poruku. S obzirom na to da se za implementaciju servisa integriteta koristi asimetrična kriptografija, odnosno digitalno potpisivanje, pri čemu pošiljalac koristi svoj privatni ključ koji samo on poseduje, jasno je da ne može kasnije da kaže da je poruku poslao neko drugi.

Servis autentifikacije nije obavezan za sve sisteme. Većina javnih servisa pruža uslugu tako što korisnik samo pristupa podacima i sadržajima bez potrebe da priloži neki dokaz svog identiteta. Naravno, ovo ne mora uvek biti slučaj, naročito pri upotrebi veb servisa. Neki sistemi koriste razne alate i algoritme da prepoznaju da li je korisnik već koristio uslugu. Postoje tri načina da se neko predstavi, nečim što zna, nečim što ima ili nečim što jeste. Primer nečega što korisnik zna može biti lozinka, nečega što ima može biti kartica, a nečega što jeste – biometrija. S obzirom na to da se kartica može kopirati, nije preporučljivo da se takav vid zaštite koristi samostalno kao ostala dva načina, već najmanje u kombinaciji sa jednim od njih. Takav vid zaštite naziva se višefaktorska autentifikacija.

3.3. Razmena ključa

Pretpostavka pri projektovanju sistema je da su informacije o korišćenim algoritmima, protokolima i samoj implementaciji javno dostupne. Jedina tajna je ključ. Po Kerkhofovim principima snaga sistema zasniva se na snazi ključa. Iz toga proizilazi da je upravljanje ključevima kritična tačka i da se tome mora posvetiti najveća pažnja. Najbolji način za razmenu ključa je da to ne bude istim kanalom kojim ide i komunikacija. Prilikom preuzimanja kartice u banci, posebno dobijamo samu karticu, a posebno *PIN*. Državne institucije imaju još rigoroznije protokole za razmenu ključeva. Ovakav način razmene ključa podrazumeva da se dve strane poznaju na neki način. Na internetu je takva situacija retka, te je potrebno naći neke druge načine za sigurnu komunikaciju.

Određivanjem jačine ključa bavi se teorija informacija[7].

3.3.1. *Diffie–Hellman* protokol

Protokol *Diffie–Hellman* za generisanje simetričnog ključa odvija se na istom kanalu kao i glavna komunikacija. Snaga ovog protokola je korišćenje diskretnog algoritma koji se za dovoljno velike parametre može smatrati neizračunljivim. Loša strana je što je neophodno koristiti velike vrednosti koje samim tim troše resurse i poskupljuju samu operaciju. Takođe, s obzirom na to da pruža visoku tajnost, učesnici se pre započinjanja protokola moraju identifikovati, tako da je bar jednostrana autentifikacija neophodna. Obično je ta strana servis[8].

3.3.2. Razmena ključa na fizičkom sloju

Razmena ključa na fizičkom sloju podrazumeva da dve strane u komunikaciji same generišu ključ na osnovu podataka iz nekog zajednički dostupnog informacionog izvora. Ideja ovog načina je da će sve strane, uključujući i potencijalnog malicioznog napadača usled grešaka u prenosu dobiti različite podatke, tako da svako sa svakim ima zajednički, ali i svoj jedinstven deo informacije. Kada prikupe podatke, dve strane određenim transformacijama pronalaze ono što je njima zajedničko i na taj način isključuju treću stranu. Važno je napomenuti da je greška u prijemu neophodan uslov za ovaj protokol. U slučaju da znamo da će prijem biti dobar, potrebno je generisati grešku pre započinjanja transformacija[9].

3.4. *SSL* protokol

Razvojem elektronskog poslovanja postalo je neophodno da se razvije sistem u koji će imati poverenja svi učesnici u komunikaciji. Asimetrična kriptografija je u to vreme već bila razvijena, ali je problem iskoristiti prednosti koje nudi. Ovo se odnosi na izdavanje i čuvanje ključeva koji se koriste u algoritmima. Dodatnu kompleksnost unosi potreba da za svaku operaciju šifrovanja i dešifrovanja koristimo različite ključeve, tako da je neophodno posedovati minimum par ključeva da bi se uspešno izvela komunikacija. Isto tako je važno da sve strane mogu da provere čiji su ključevi. Tu dolazimo do pojma digitalnog sertifikata, kao i institucije koja ga izdaje korisniku, koja se naziva sertifikaciono telo[10].

Na digitalnom sertifikatu, pored ostalih parametara stoji javni ključ i identitet vlasnika sertifikata, a takođe i digitalni potpis sertifikacionog tela kao izdavaoca. Sertifikaciono telo je institucija u koju svi imaju poverenja. To poverenje može biti narušeno kompromitovanjem podataka o klijentima ili nekim sličnim propustima u njihovom radu. Ali sve ovo još nije dovoljno da bi komunikacija bila bezbedna, s obzirom

ns to da, iako su digitalni sertifikati javno dostupni, običan korisnik nema načina da ih bezbedno preuzme. Tu ulogu su, kao korisničke aplikacije, na sebe preuzeli veb pregledači, tako što oni za korisnika proveravaju da li je sertifikat validan. Pored ostalih karakteristika, jedna od najvažnijih osobina pri izboru koji pregledač ćemo koristiti jeste i to da li su dobro implementirane ove provere, pošto je i tu, kao i kod sertifikacionih tela, najvažniji faktor poverenje korisnika.

Uvođenjem sertifikacionih tela, digitalnih sertifikata i pregledača u priču stekli su se uslovi za osmišljavanje protokola koji bi mogao da ponudi zaštitu komunikacije. Protokol koji se najviše koristi u ove svrhe jeste *Secure Socket Layer*.

Suština ovog protokola je uspostavljanje bezbedne komunikacije između serverske i klijentske aplikacije (pregledača), pre nego što korisnik krene sa radom. Važno je napomenuti da u ovoj fazi postoji samo jednostrana autentifikacija. Korisnik zna ko je servis, dok servis ne zna ništa o korisniku.

Prvi korak je da servis koji nudi uslugu preuzme digitalni sertifikat od sertifikacionog tela. Sledeći korak je da podesi svoju aplikaciju tako da radi preko protokola koji obaveštava pregledač da je to bezbedan servis. Uloga pregledača je da proveriti da li je to stvarno tačno. Kada korisnik pokuša da ode na veb adresu i ako je na toj aplikaciji podešen bezbedan protokol, to je signal pregledaču da proveriti kod sertifikacionog tela da li za taj domen stvarno postoji validan digitalni sertifikat. Ako je sve u redu, pregledač će nastaviti sa protokolom, u suprotnom prikazaće korisniku poruku da nastavlja na svoju odgovornost. U tom trenutku korisnik je autentifikovao servis. Dalje se preko različitih parametara i protokola i javnog ključa servisa generiše simetrični ključ koje se upotrebljava zašifrovanje dalje komunikacije.

Preporučljivo je, a najčešće se to tako i implementira, da generisani simetrični ključ važi samo za tu konkretnu sesiju. Ovo, pored izbegavanja problema sa čuvanjem ključeva, omogućava perfektnu bezbednost unapred. U nekom trenutku mi možemo biti sigurni da napadač ne može da dešifruje šifrat, ali nikada ne znamo šta nosi budućnost i da li će se pronaći uspešan napad na šifarski sistem. Ako koristimo isti ključ za svaku sesiju, napadač bi imao mogućnost da pronalaženjem samo jednog ključa ima uvid u kompletnu komunikaciju koja se ostvarila u nekom periodu. Sesijskim ključem ograničavamo napadača da ima pristup komunikaciji samo za tu konkretnu sesiju za koju je rekonstruisao ključ.

4. Pregled predloženog rešenja

Predlog protokola koji je tema ovog rada pokušava da predupredi ili nekim slučajevima dodatno objasni krajnjem korisniku rizike po privatnost ili bezbednost na internetu ili bilo kojoj mrežnoj komunikaciji. Sama implementacija uključuje minimum bezbednosnih pretpostavki kako bi se što transparentnije mogli prikazati navedeni rizici.

Protokol definiše redosled kojim se razmenjuju informacije. To podrazumeva da se definišu učesnici i tip podataka koji, grupisani ili transformisani na određeni način, predstavljaju te informacije.

Podaci koji se razmenjuju predstavljaju autentifikacione i autorizacione parametre učesnika u komunikaciji. Transformacije su različiti algoritmi za šifrovanje ili kompresiju. Učesnici su aplikacije preko kojih se izvršava protokol. U daljem tekstu biće objašnjeno ko su učesnici, kao i kojim podacima i algoritmima moraju da raspolazu da bi se protokol uspešno implementirao i kasnije odvijao.

4.1. Učesnici u protokolu

Sistem u kome se odvija protokol uključuje:

- server ili provajder – glavna serverska aplikacija koja čuva i proverava tajne podatke,
- klijent – aplikacija koja koristi usluge provajdera,
- krajnji korisnik – korisnik usluga klijentske ili serverske aplikacije, ovaj naziv se nekada izjednačava sa aplikacijom koja je posrednik sa ostalim učesnicima u komunikaciji i uglavnom predstavlja veb pregledač; nekada je međutim potrebno naglasiti razliku između ova dva entiteta.

4.1.1. Provajder

Sa aspekata kompleksnosti i važnosti to je najbitniji deo sistema, ali s obzirom na to da se nalazi na računaru ili u okviru mreže nad kojom bi onaj ko implementira ceo protokol trebalo da ima punu kontrolu, možda ipak jednostavniji za upravljanje pošto postoji uvid u ceo kôd.

Uloga provajdera je da na bezbedan način pomogne u razmeni podataka između krajnjeg korisnika i klijenta. Ta razmena podrazumeva da svi dobiju tačno onoliko informacija koliko im je potrebno za obavljanje određenog posla. Pojednostavljeno rečeno, provajder je učesnik u kog obe strane imaju poverenja. Da bi zaslužio i održavao visok nivo poverenja, provajder mora da ispunjava određene zahteve:

-
- podaci koje mu ostali učesnici predaju na čuvanje moraju biti zaštićeni na najbolji mogući način,
 - pružanje bezbednog načina za dostavu i razmenu tih podataka.

S obzirom na to da se protokol uglavnom odnosi na komunikaciju u okviru internet pregledača, provajder je obavezan da na svom serveru implementira neki od bezbednosnih protokola koje internet pregledači podržavaju. Za većinu slučajeva to je minimalno upotreba *SSL/TLS* protokola. Dodatna zaštita može se postići i korišćenjem *HSTS* protokola (*HTTP Strict Transport Security*). Na ovaj način bi trebalo da se prenos podatka koje klijent i kranji korisnik predaju provajderu na čuvanje i kasniju upotrebu obavi na bezbedan način. Takođe, ovako se štite i podaci koje provajder daje klijentu na čuvanje.

Implementacija za potrebe ovog rada ne uključuje navedene protokole iz razloga transparentnijeg pregleda informacija i paketa koji se razmenjuju.

4.1.2. Klijent

Klijent je aplikacija koja nudi sadržaj za krajnjeg korisnika. Da bi izbegao upotrebu bezbednosnih mehanizama autentifikacije i autorizacije, klijent se obraća provajderu i zahteva uslugu za navedene servise. Pri registraciji kod provajdera dobija podatke koji su mu kasnije neophodni za sprovođenje protokola. Po protokolu, sa bezbednosnog aspekta, za implementaciju servisa autentifikacije i inicijalne autorizacije na klijentskoj aplikaciji dovoljna je obična *HTTP* konekcija. Za servis autorizacije, odnosno za pristup krajnjeg korisnika poverljivim podacima ili resursima, klijent ima dva izbora:

- da doda iste zaštitne protokole kao i provajder,
- da zahteva od provajdera servis za autorizaciju svakog zahteva koji podrazumeva poverljivosti koji će se dalje obaviti po istom protokolu kao i autentifikacija.

4.1.3. Krajnji korisnik

Krajnji korisnik konzumira sadržaj koji nudi klijent. On pristupa servisima klijenta i provajdera uglavnom preko internet pregledača. Krajnji korisnik se smatra učesnikom koji ne poseduje nikakva informatička znanja, posebno ona koja se odnose na bezbednost. Takođe, sa aspekta konkurentnosti, proizvođači aplikacija se trude da mu pruže što bolje korisničko iskustvo koje je obično u suprotnosti sa dobrim praksama u bezbednoj komunikaciji. Uzimajući ovo u obzir, dodatna uloga provajdera i klijenta, kako je to napomenuo i sam autor [OAuth2](#) okvira, jeste da upoznaju korisnike sa tim aspektima.

Protokol krajnjeg korisnika tretira samo kao kurira izmedju provajdera i klijenta.

4.2. Algoritmi u protokolu

Algoritmi koji se koriste u protokolu su:

- *RSA* – za šifrovanje i digitalno potpisivanje,
- *Base64* – za kodovanje *HTTP* zahteva,
- *SHA1* – za kreiranje otiska poruke.

4.3. Podaci u protokolu

Postoje dva načina rada provajdera i shodno tome potrebno je manje ili više parametara:

- samo servis autentifikacije – klijent pošalje zahtev za autentifikaciju krajnjeg korisnika, provajder odgovara i tu se komunikacija završava.
- servis autentifikacije i autorizacije – posle autentifikacije, klijent nastavlja komunikaciju kada mu zatreba autorizacija za pojedinačne zahteve krajnjeg korisnika.

4.3.1. Kriptološki parametri

Provajder pre prihvatanja prvog klijenta, odnosno pre samog započinjanja sa radom, mora da generiše svoj par ključeva za digitalno potpisivanje. Ovo je neophodno za servis autentifikacije i opcionu inicijalnu autorizaciju.

U slučaju da želi da proširi uslugu i na potpun servis autorizacije, odnosno da klijent i krajnji korisnik preko provajdera žele da ostvaruju pristup resursima, neophodno je da generiše još jedan par ključeva za šifrovanje zahteva za autorizaciju.

Pri registraciji klijenta, potrebno je napraviti, opet u zavisnosti od željene usluge autorizacije, jedan ili dva para ključeva.

Kada se klijent registruje, dostavljaju mu se samo neophodni ključevi i to:

a) U slučaju servisa autentifikacije:

- klijentov privatni ključ – za dešifrovanje odgovora provajdera,
- provajderov javni ključ – za proveru integriteta i izvora poruke.

S obzirom na to da je komunikacija šifrovana samo u jednom pravcu, klijentu nije potreban njegov javni ključ.

-
- b) U slučaju punog servisa autorizacije, koji uključuje autentifikaciju – neophodno je da ima oba para svojih ključeva, zbog kasnijih zahteva za autorizaciju.

4.3.2. Parametri izvora zahteva i prosleđivanja odgovora

Pri registraciji klijent dostavlja provajderu adrese sa kojih će stizati zahtevi i adrese na koje će se prosleđivati odgovori. Ovo je neophodno da ne bi dolazilo do obrade lažnih zahteva ili prosleđivanja odgovora na pogrešne adrese kojima u najvećem broju slučajeva upravljaju nedobronamerni klijenti. Takođe, klijent mora da ima adresu sa koje će stizati odgovori provajdera iz istog razloga.

4.3.3. Autentifikacioni i autorizacioni parametri

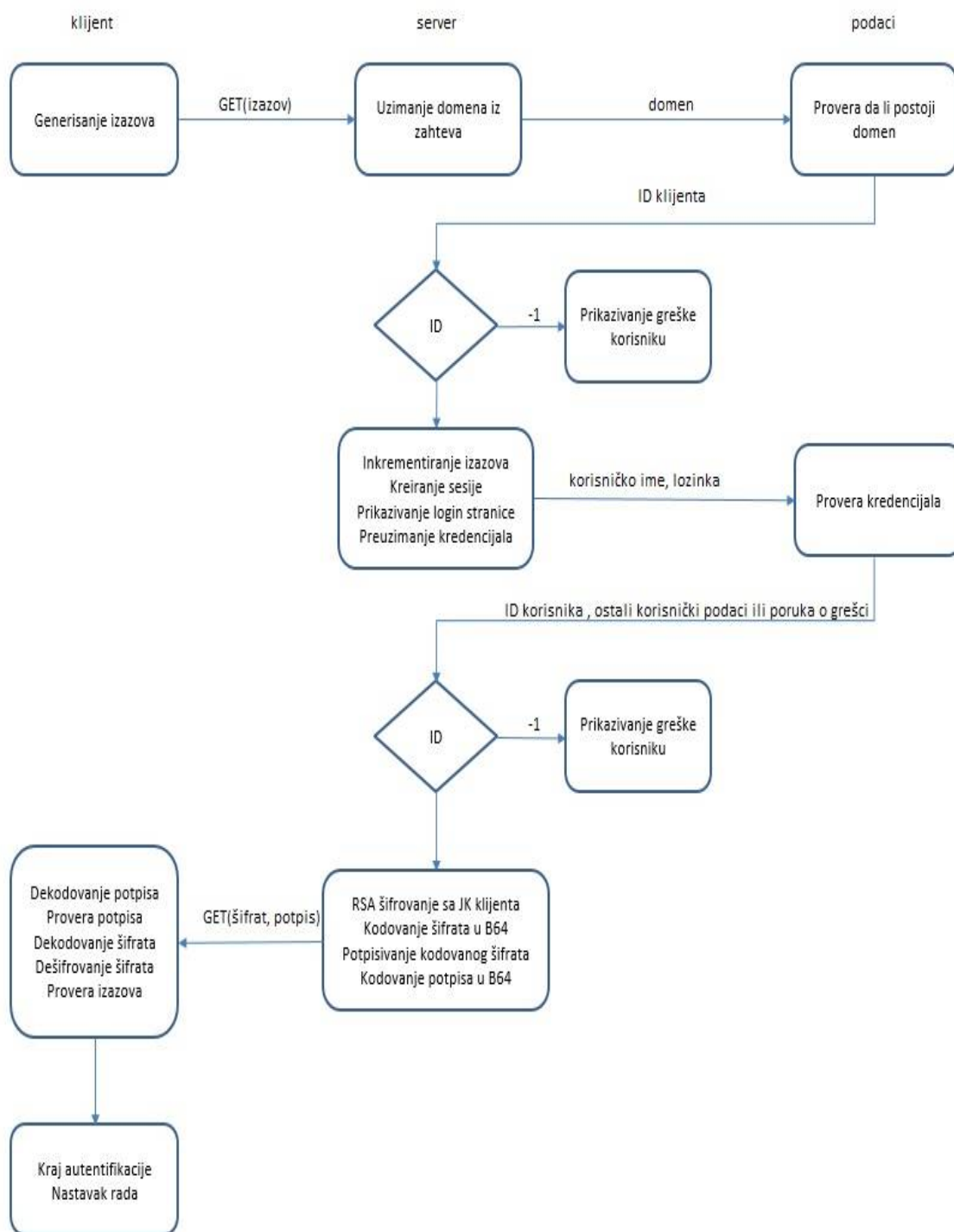
Protokol predviđa jednofaktorsku autentifikaciju, tako da su autentifikacioni parametri, kako za klijenta tako i krajnjeg korisnika, korisničko ime ili adresa elektronske pošte i lozinka.

Inicijalni autorizacioni parametar je niz karaktera koji predstavlja privilegije krajnjeg korisnika u okviru klijentske aplikacije. Klijent može da reši ovaj problem u svojoj aplikaciji preko tipova ili uloga korisnika, tako da ovaj parametar nije obavezan.

U slučaju punog servisa autorizacije, može biti potrebno da klijent dostavi provajderu algoritam ili mapu, koji omogućavaju proveru da li zahtev treba da bude ispunjen.

4.4. Šema protokola

Na slici 2 prikazana je šema protokola. Na slici nije prikazan krajnji korisnik kao entitet. Njegova uloga, odnosno internet pregledača kao korisničke aplikacije je da prenosi poruke između klijenta i provajdera.



Slika 2 - Šema protokola

Na slici 3 prikazan je log izvršavanja protokola. Server predstavlja provajdera.

```

Info: Klijent: Izazov i postavljanje izazova u sesiju: -69932230
Info: Klijent: Redirekt: http://dacha:8080/StwWebAuth/Login?idn=-69932230
Info: Server: Uzimam naziv domena iz URL-a: GPSPonger
Info: Server: Proveravam da li postoji i ako postoji uzimam ID i URL gde teba da vratim korisnika
Info: Server: ID: 2
Info: Server: Ako provera vrati ID = -1, prosledjuje se na stranicu sa greskama i obavestava da sajt nije registrovan
Info: Server: Uzimam izazov i inkrementiram: -69932229
Info: Server: Kreiram sesiju i postavljam atribut:
Info: Server: identifier: -69932229
Info: Server: name: GPSPonger
Info: Server: urlTO: GPSPonger/ClientAuthed
Info: Server: i prosledjujem na login stranicu.
Info: Server: Preuzimam podatke iz posta i kreiram get parametre
Info: Server: ID=1&name=neki_korisnik&role=user&authr=[{setting1:5,setting2:6}]&idn=-69932229
Info: Server: Sifrat sa javnim kljucem klijenta:
Info: Server: .NwZÅ:šqê46"-ÊôjÆ;5ð%ÊÖæK®,âiWâŽ'jôš-žh_ô%|+||À-|I8?üxu·xb|+x>+ÔRVF<{ ·QUûiTâiV1c0ÛsIÛ#?é'@æðŹÁÄD»E?¹ÇOUy@âúJÐæŹÛ÷B-:G..P
'Zæw
Info: Server: Sifrat kodovan u Base64:
Info: Server: Lk4Pv8A6mnHqEPOZfC2j9GrGDaE18L32ytXma66C4u4P5I4EGfSKrY5oX/Y1BSsTwBoXzjiQ/Hh1t9difIZ4PobUULZGPHsgt1FV
+e5U5MxWŹWOp3HPP/COD6ZJAnPQUwREu4yducFvVf2p4PpK0LzmFNn3QI06R4VQtL205nc=
Info: Server: Potpisujem sa svojim privatnim kljucem:
Info: Server: 3t>?iF8-ÄÄ|eðBµêD/ü«Äæ"z·Nfðe)z#PDgôigüÜŹ_æ*PŽ||'êKê+Z...eÇæ³;KææTcàZ---ÛJ?T%ôí!!ÍÆÛñ'44·Fý4÷\LYÄæð+?k
%ÊÜwDèN<:JiZ,|K·*âZæðNDÄæ
Info: Server: i kodujem u Base64:
Info: Server: M7G7nWlG3y3Cwxll9UK16UQvf3WrwcaUPHqVToP4ZS16HaVQ0Gf07Ue13CXcX4kM3o4TkupLHuuHwVlX4zos79rupxUY
+Ba1635So9UJfXtE83G3PGRoJ53Rv0091wV3cOM+IaPayXK3HFQ6E6L0kppjgKmS7cq5DKc+FdEwZw=
Info: Server: Kreiram GET sa dva parametra 1 i 2:
Info: Server: ?1=Lk4Pv8A6mnHqEPOZfC2j9GrGDaE18L32ytXma66C4u4P5I4EGfSKrY5oX/Y1BSsTwBoXzjiQ/Hh1t9difIZ4PobUULZGPHsgt1FV
+e5U5MxWŹWOp3HPP/COD6ZJAnPQUwREu4yducFvVf2p4PpK0LzmFNn3QI06R4VQtL205nc=&2=M7G7nWlG3y3Cwxll9UK16UQvf3WrwcaUPHqVToP4ZS16HaVQ0Gf07Ue13CXcX4k
M3o4TkupLHuuHwVlX4zos79rupxUY+Ba1635So9UJfXtE83G3PGRoJ53Rv0091wV3cOM+IaPayXK3HFQ6E6L0kppjgKmS7cq5DKc+FdEwZw=
Info: Server: Redirekt URL:
Info: Server: http://dacha:8080/GPSPonger/ClientAuthed?
1=Lk4Pv8A6mnHqEPOZfC2j9GrGDaE18L32ytXma66C4u4P5I4EGfSKrY5oX/Y1BSsTwBoXzjiQ/Hh1t9difIZ4PobUULZGPHsgt1FV
+e5U5MxWŹWOp3HPP/COD6ZJAnPQUwREu4yducFvVf2p4PpK0LzmFNn3QI06R4VQtL205nc=&2=M7G7nWlG3y3Cwxll9UK16UQvf3WrwcaUPHqVToP4ZS16HaVQ0Gf07Ue13CXcX4k
M3o4TkupLHuuHwVlX4zos79rupxUY+Ba1635So9UJfXtE83G3PGRoJ53Rv0091wV3cOM+IaPayXK3HFQ6E6L0kppjgKmS7cq5DKc+FdEwZw=
Info: Server: Ukidam sesiju i saljem klijentu.
Info: Klijent: GET:
Info: Klijent: ?1=Lk4Pv8A6mnHqEPOZfC2j9GrGDaE18L32ytXma66C4u4P5I4EGfSKrY5oX/Y1BSsTwBoXzjiQ/Hh1t9difIZ4PobUULZGPHsgt1FV
+e5U5MxWŹWOp3HPP/COD6ZJAnPQUwREu4yducFvVf2p4PpK0LzmFNn3QI06R4VQtL205nc=&2=M7G7nWlG3y3Cwxll9UK16UQvf3WrwcaUPHqVToP4ZS16HaVQ0Gf07Ue13CXcX4k
M3o4TkupLHuuHwVlX4zos79rupxUY+Ba1635So9UJfXtE83G3PGRoJ53Rv0091wV3cOM+IaPayXK3HFQ6E6L0kppjgKmS7cq5DKc+FdEwZw=
Info: Klijent: Dekodujem drugi parametar iz b64 (potpis servera):
Info: Klijent: 3t>?iF8-ÄÄ|eðBµêD/ü«Äæ"z·Nfðe)z#PDgôigüÜŹ_æ*PŽ||'êKê+Z...eÇæ³;KææTcàZ---ÛJ?T%ôí!!ÍÆÛñ'44·Fý4÷\LYÄæð+?k
%ÊÜwDèN<:JiZ,|K·*âZæðNDÄæ
Info: Klijent: Dodajem prvi parametar u signature buffer.
Info: Klijent: Proveravam sa javnim kljucem servera:
Info: Klijent: Verifikovanje uspešno!
Info: Klijent: Dekodujem parametre iz B64 (Parametar 1 iz GET-a):
Info: Klijent: .NwZÅ:šqê46"-ÊôjÆ;5ð%ÊÖæK®,âiWâŽ'jôš-žh_ô%|+||À-|I8?üxu·xb|+x>+ÔRVF<{ ·QUûiTâiV1c0ÛsIÛ#?é'@æðŹÁÄD»E?¹ÇOUy@âúJÐæŹÛ÷B-:G..P
'Zæw
Info: Klijent: Desifrujem za svojim tajnim kljucem:
Info: Klijent: ID=1&name=neki_korisnik&role=user&authr=[{setting1:5,setting2:6}]&idn=-69932229
Info: Klijent: Preuzimam uvecani izazov iz GET-a i dekrementiram
Info: Klijent: -69932230
Info: Klijent: Uzimam izazov iz sesije:
Info: Klijent: -69932230
Info: Klijent: Ako su u redu:
Info: Klijent: Ovde klijentski sajt odlucuje sta ce dalje da radi.
Info: Klijent: Ako nisu:
Info: Klijent: Ukida se sesija za postavljenim izazovom.

```

Slika 3 - Log izvršavanja protokola

U tabeli 1 prikazan je detaljniji redosled koraka. Radi jednostavnosti krajnji korisnik predstavljen je samo kao korisnik. Predstavljene koraci važe za opciju gde postoji samo servis autentifikacije i inicijalne autorizacije ako postoji.

Korak	Izvršilac	Opis
1	Korisnik	Pristupa adresi klijenta
2	Klijent	Generiše izazov za provajdera
3	Klijent	Kreira sesiju u koju beleži izazov
4	Klijent	Konstruiše <i>GET</i> zahtev u koji postavlja izazov
5	Korisnik	Odlazi na adresu iz zahteva
6	Provajder	Iz zaglavlja preuzima domen sa koga je stigao zahtev
7	Provajder	Proverava da li je taj domen registrovan i ako jeste preuzima adresu na koju će poslati odgovor
8	Provajder	Ako je domen validan, uzima i inkrementira izazov iz zahteva
9	Provajder	Kreira sesiju u koju beleži provajdera, adresu za odgovor i inkrementirani izazov
10	Provajder	Prosleđuje korisnika na stranicu za prijavljivanje
11	Provajder	Korisnik ukucava svoje kredencijale
12	Provajder	Proverava kredencijale, kao i da li je korisnik prijavljen kod datog klijenta
13	Provajder	Kreira prvi parametar odgovora u koji postavlja korisničko ime, inicijalnu autorizaciju i izazov
14	Provajder	Šifruje parametar sa javnim ključem klijenta
15	Provajder	Koduje šifrat u <i>Base64</i> format
16	Provajder	Digitalno potpisuje kodovani šifrat
17	Provajder	Koduje potpis u <i>Base64</i> format

18	Provajder	Formira <i>GET</i> zahtev sa adresom odgovora i dva parametra, kodovani šifrat i kodovani potpis, ukida sesiju i prosleđuje zahtev korisniku
19	Korisnik	Odlazi na adresu iz zahteva
20	Klijent	Proverava da li je odgovor stigao sa validne adrese
21	Klijent	Dekoduje iz <i>Base64</i> drugi parametar iz zahteva koji predstavlja potpis provajdera
22	Klijent	Verifikuje potpis
23	Klijent	Dekoduje iz <i>Base64</i> prvi parametar iz zahteva
24	Klijent	Dešifruje parametar
25	Klijent	Proverava da li se izazov poklapa sa izazovom u sesiji
26	Klijent	Ako je sve u redu, prosleđuje korisnika dalje u aplikaciju

Tabela 1 - Koraci u izvršavanju protokola

4.4.1. Diskusija o pojedinim koracima protokola

S obzirom na to da osnovna postavka protokola predviđa samo neophodne korake, u ovom delu biće razmotreni neki od ovih koraka, mogući napadi i predloženi dodaci za poboljšanje bezbednosti.

4.4.1.1. Generisanje izazova – Korak 2

Klijent generiše slučajnu celobrojnu vrednost iz dva razloga.

Prvi je da bi imao identifikator.

Drugi razlog je što su parametri koje provajder vraća klijentu nakon provere korisnika u većini slučajeva identični (korisničko ime, privilegije i slično). Na ovaj način se uvodi dodatno obezbeđenje od napada ponavljanja poruke tako što će provajder ovaj izazov pridružiti tim parametrima i na taj način svaki potpis provajdera biće različit.

Dodatno, provajder može pridružiti i vremenski pečat da se još poveća jedinstvenost zahteva.

4.4.1.2. Kreiranje sesije i beleženje izazova – Korak 3

Ovde se klijent oslanja na pregledač i očekuje da se posle provere vrati isti korisnik koji je i poslat kod provajdera.

4.4.1.3. Provera domena iz zahteva – Korak 7

Prvi razlog je delom, iz ugla protokola, možda i trivijalan. Ako provajder naplaćuje uslugu, mora proveriti da li je ta usluga trenutno i plaćena.

Drugi deo, mnogo važniji, jeste da li je zahtev stigao od validnog klijenta.

U isto vreme preuzima se i adresa za odgovor. Ovo je veoma važan deo protokola. Ako se desi da pristigne zahtev sa lažiranim zaglavljem, korisnik će ipak biti poslat na pravu adresu klijenta. U tom slučaju prijem na aplikaciju biće mu odbijen, ali klijent to može da ubeleži i da formira statistiku u koju će taj zahtev upisati kao potencijalni napad.

Takođe, kada kasnije korisnik da svoje kredencijale na stranici za prijavu, provajder će pored istih, proveravati i da li je korisnik prijavljen kod tog klijenta.

4.4.1.4. Preuzimanje izazova – Korak 8

Uloga izazova je sprečavanje napada sa ponavljanjem poruke.

Pored dodatka vremenskog pečata od strane provajdera da bi klijent mogao da odredi vremenski okvir u kome korisnik mora da se vrati i tako zaštititi sebe, provajder može dodatno da beleži izazove i da proverava da li se i na koje vreme ponavljaju, kako bi zaštitio krajnjeg korisnika u slučaju da je klijentski sajt nepouzdan ili maliciozan.

4.4.1.5. Provera korisnika – Korak 12

Provera korisničkog imena ili adrese elektronske pošte sa pridruženom lozinkom, kao i da li je korisnik prijavljen kod klijenta. Ovo je sadržaj koraka u slučaju da je zahtevana samo autentifikacija i opciono inicijalna autorizacija.

U slučaju da se klijent prijavi i za pun servis autorizacije, potrebno je da dostavi provajderu podatke (u vidu algoritma ili mape) uz pomoć kojih će biti proverljivo da li konkretan korisnik ima privilegije za ispunjenje tog konkretnog zahteva.

Primer može biti *Cross-site request forgery* ili neki sličan [napad lažiranja](#). Ako algoritam ili mapa ne pružaju dovoljnu zaštitu, onda je potrebna autorizacija za svaki zahtev, kao i dodatno edukovanje korisnika o mogućim opasnostima.

4.4.1.6. Odgovor provajdera – Korak 18

Ukidanje sesije. Ako je klijent prijavljen samo za servis autentifikacije, ovo je sasvim logično rešenje. Međutim, ukoliko je zahtevao i servis autorizacije, onda postoje dva scenarija.

Prvi je i da autorizacija ide posebno za svaki zahtev kao i autentifikacija, ali to za veliki broj korisnika ili čestih zahteva može biti neizvodljivo.

Drugi je da provajder ostavi sesiju otvorenom neko vreme koje propiše i da generiše token sa vremenskim pečatom isteka tokena. Taj token pridodaje se prvom parametru odgovora i klijent ga po prijemu upisuje u sesiju korisnika na svom serveru. Dalji zahtevi mogu se autorizovati preko tog tokena, ali u tom slučaju neophodno je i da klijent dobije oba para svojih ključeva za *RSA* algoritam, s obzirom na to da ti zahtevi moraju biti šifrovani i digitalno potpisani.

4.5. Odnos između učesnika u protokolu

Polazna pretpostavka protokola je da klijent i krajnji korisnik imaju poverenja u provajdera.

U ovom i sledećem delu biće izneti neki stavovi koji možda deluju suviše autoritarno i sa mnogim ograničenjima, ali uslovi za bezbednost moraju biti transparentni. Krajnji korisnik treba da bude upoznat sa trenutnim i upozoravan na nove rizike koji se pojavljuju. S obzirom na to da se na internetu sve odigrava brzo, informacije i poruke kroz koje se korisnici obaveštavaju su kratke, jasne i precizne.

4.5.1. Kako provajder posmatra klijenta i krajnjeg korisnika

4.5.1.1. Tip klijenta

Kako to propisuje i [OAuth 2.0 okvir](#) uz činjenicu da se većina aplikacija izvršava na vebu, klijent se proglašava javnim. To znači da provajder nema garanciju, bar ne bez neke dodatne provere ili sertifikacije, da je klijent sposoban da čuva ključeve i tokene.

4.5.1.2. Rangiranje klijenta

Javni klijenti su ogromna većina, tako da ta osobina ne govori mnogo. Provajder bi trebalo da rangira klijente prema sopstvenoj proceni i da taj rang prikazuje korisniku. Ako se sprovodi rangiranje, parametri na osnovu kojih se vrši su javni. Postoje različite vrste aplikacija koje traže različite podatke od korisnika. Isto tako, klijent može nuditi usluge skladištenja fajlova korisnika. Sve ovo može biti kompromitovano ako je korisnik lakomislen i ne proverava kome je predao svoje resurse na upotrebu i čuvanje.

Prvi podatak koji može biti izložen je adresa elektronske pošte. Tu provajder može da odigra ulogu tako što će za svakog svog korisnika otvoriti adresu, tako da korisnici klijentima daju tu adresu. Na taj način stvarne adrese korisnika, koje inače koriste za komunikaciju, bile bi poznate samo provajderu. Ukoliko bi klijent zahtevao dodatne adrese, dobijao bi negativne poene u rangiranju. Ova informacija je lako merljiva, odnosno lako se može prikazati krajnjem korisniku.

Druga, mnogo komplikovanija stvar za procenu je merenje koliko je sajt klijenta bezbedan. Ovo se posebno odnosi na servis autorizacije i [napad lažiranja](#). Nijedna važna operacija ne sme da se obavlja samo na osnovu *URI*-ja, naročito u režimu rada sa otvorenom sesijom kod provajdera i tokenom za autorizaciju.

Ako je token izložen korisničkoj aplikaciji, tada je izložen i napadaču i može biti zamenjen pri izvršavanju akcije. Klijentska aplikacija mora biti dizajnirana tako da se samo stavke akcije nalaze u *URI*-ju, recimo kod prebacivanja novca može biti samo količina, kao i kome se prebacuje. Kada se takva akcija prosledi klijentskom serveru, aplikacija proverava da li je iznos veći od prethodno odobrenog maksimuma od strane korisnika. Takođe, aplikacija može biti dizajnirana tako da korisnik prvo mora da označi račune na koje će prebacivati novac, i da samo izabrane može da bira u transakcijama. Time se značajno sužava broj malicioznih korisnika koji bi mogli da zloupotrebe sistem. Ceo ovaj algoritam može preko tipova zahteva biti prosleđen provajderu, da bi bio u stanju da autorizuje zahtev. Za sve što nije predviđeno algoritmom, provajder će tražiti dodatnu autorizaciju od korisnika uz kratko i precizno objašnjenje zašto je potrebna. Na taj način provajder može u mnogo većoj meri da garantuje korisniku bezbednost klijentske aplikacije i to dodatno može uticati na rang klijenta.

4.5.1.3. Krajnji korisnik

Kako prosečan korisnik ne poseduje potrebna informatička znanja, smatra se najslabijom karikom u lancu. Sa druge strane, najvažniji je učesnik, s obzirom na to da se ceo sistem razvija da bi mogao da konzumira sadržaj ili završi neki posao.

Korisniku je potrebno jasno naznačiti za koje usluge je potrebna dodatna autorizacija. Kako se to kosi sa korisničkim iskustvom, potrebno je naći pravu meru i usmeravati klijenta da tom problemu posveti pažnju. Nepotrebni zahtevi za autorizaciju mogu da prouzrokuju prestanak korišćenja usluge. Takođe, korisnik za premošćavanje tih problema može posegnuti za korišćenjem nekog dodatnog softvera koji će mu olakšavati proces, a time se izlaže dodatnom riziku, kako po pitanju izlaganja svojih kredencijala, tako i automatizovanjem autorizacije koja time gubi smisao. Postoji velika verovatnoća da se kod korisnika stvori navika stalnog autorizovanja zahteva. Ta navika može, i kod iskusnijih korisnika, proizvesti pad pažnje u trenutku napada.

4.5.2. Kako klijent posmatra provajdera

Klijent se od početka nalazi u podređenom položaju zbog poverenja koje mora da pokloni provajderu. To se dodatno produbljuje ako preda algoritam za autorizaciju zahteva. Međutim, klijent ovo može i drugačije da posmatra. Poštovanjem zahteva provajdera izmešta veliki deo posla oko bezbednosti sa svoje platforme i može da se bavi samo svojim servisom.

4.6. Napadi

U [diskusijio koracima protokola](#) dato je kratko objašnjenje na koji način sprečavaju pojedine napade. U narednih nekoliko tačaka ti scenariji biće razmotreni detaljnije. Sa aspekta poštovanja pravila postoje različite vrste klijenata. Većina će verovatno postupiti po propisanim ograničenjima. Međutim, postoje i nespretni, koji će možda iz neznanja isključiti deo koda ako im smeta, a postoje i oni zlonamerni. Efekti rada obe navedene grupe mogu dovesti do kompromitovanja sistema, pa se oni smatraju neprijateljskim klijentima.

4.6.1. Napad ponavljanjem poruke

Parametri koje provajder prosleđuje klijentu po autentifikaciji krajnjeg korisnika se u velikoj većini slučajeva ne menjaju. Ti parametri sadrže uglavnom korisničko ime i neke dodatne podatke kao što su tip klijenta ili podešavanja prikaza. Takve vrste zahteva podložne su napadu ponavljanja poruka. Da bi se sprečio takav napad uvodi se izazov. Korišćenje izazova podrazumeva generisanje, slanje, inkrementiranje na prijemu i proveru po dobijanju odgovora. S obzirom na to da se pri autentifikaciji koriste uglavnom statički parametri, dodavanjem izazova i propuštanjem poruke kroz izabranu jednosmernu funkciju dobija se jedinstven otisak poruke.

Ako je provajder u kodu koji je predao klijentu na izvršavanje obezbedio kriptografski sigurnu slučajnu vrednost za izazov ili dodao vremeski pečat, tu bi problem bio rešen. Međutim, ako je u pitanju neprijateljski klijent, mora se posvetiti dodatna pažnja. Takav klijent bi mogao da isključi generisanje kriptografski sigurne slučajne vrednosti i da je pretvori u običnu pseudo slučajnu. To može učiniti iz želje da smanji trošenje svojih resursa, ali namere mogu biti i daleko malicioznije.

Jedan od načina za praćenje ponašanja je formiranje baze izazova koji stižu od klijenata i provera da li zadovoljavaju uslove koje je ispunjavao kôd koji je poslat klijentskoj strani. Na taj način, provajder može da proverava da li su vrednosti onakve kakve su propisane. Dodatno, ako su vrednosti ispravne, može ih kasnije koristiti kao izvor za generisanje svojih slučajnih vrednosti.

4.6.2. Napad preusmeravanja

Ovaj napad izvršava se zamenom adrese na koju treba odgovoriti po obradi zahteva. Napadač u najvećem broju slučajeva umetne svoj kôd u već isporučeni koji se nalazi na korisničkoj strani. Na taj način, čak iako postoji dodatna autorizacija, odgovor će stići na adresu koju je odredio napadač.

Da bi se ovo sprečilo potrebno je da provajder proverava odredišnu adresu pre prosleđivanja odgovora. Ovo se može obaviti na dva načina. Prvi je da pri svakom zahtevu proverava da li je domen odgovarajući, ali osim što troši resurse, taj metod pretpostavlja da adresa bude poslata sa zahtevom, što protokol isključuje.

Drugi način, koji je i primenjen u ovom protokolu, jeste da klijent pri registraciji prijavi na koju adresu će za odgovarajući tip usluge provajder slati odgovore. Tako se izbegava proveravanje adrese pri svakom zahtevu. Sa prilaganjem adrese pri registraciji, proveru izvršava upravo klijent.

Ako klijent želi dinamičku adresu odgovora, ima obavezu da prosledi provajderu algoritam na osnovu koga će moći da se izvrši validacija adrese. Uvođenje algoritma bi u velikom broju slučajeva podrazumevalo dodatne parametre u inicijalnom zahtevu. Dodatni parametri usložnjavaju postavku.

U osnovnom režimu rada, odnosno u režimu gde se klijent prijavio samo za servis autentifikacije, klijent u zahtevu šalje samo izazov. To je podatak koji ne nosi nikakvu informaciju i može se poslati i običnom nezaštićenom konekcijom. Takođe, klijent u tom režimu poseduje samo svoj privatni ključ za dešifrovanje i javni ključ provajdera za proveru digitalnog potpisa odgovora.

Dodavanjem parametara za dinamičku proveru odredišne adrese odgovora, zahtevu se dodaju i informacije, a ne samo jednokratni podaci. Ova činjenica prouzrokuje promenu u zahtevima protokola na taj način da i početni zahtev mora biti zaštićen. To se može postići uvođenjem *HTTPS* protokola na klijentskoj aplikaciji ili prelaskom režima protokola u rad sa potpuno šifrovanim saobraćajem i digitalnim potpisivanjem. Taj režim podrazumeva da klijent dobije sve potrebne ključeve za obavljanje takvog saobraćaja.

Moglo bi se postaviti pitanje zašto je potrebno da se adresa prijavljuje posebno, a ne da se dodaje zahtevu. Postoje dva razloga za primenu ovog metoda. Prvi je da se deo logike prebacuje na provajdera koji po pretpostavci ima zaštićen sistem, tako da deo kôda protokola na klijentskom serveru ostaje isti za sve klijente i ne daje mnogo informacija napadaču, kome tako treba mnogo više izviđanja da bi mogao da nađe neki propust. Drugi je da provajder zna nešto o klijentskoj aplikaciji te stoga može da formira i neki vrednosni sud o samoj bezbednosti klijentske aplikacije.

4.6.3. Preuzimanje tokena

Tokeni se koriste za autorizaciju zahteva i njihova upotreba podrazumeva zaštićen saobraćaj. Token je autorizacioni string koji omogućava da se detaljnije odrede prava korisnika u odnosu na ulogu koju ima tip korisničkog naloga pod kojim je prijavljen na aplikaciju.

Postoji nekoliko načina da se tokeni distribuiraju. Jedan je da se token čuva u korisničkoj aplikaciji i da se pridodaje zahtevima za autorizaciju. S obzirom na to da je jezik na kome se klijentska aplikacija izvršava u korisničkoj aplikaciji transparentan, ovaj metod je prilično nepouzdan, sa više aspekata, kako sa strane mogućeg napada, tako i po pitanju malicioznih korisnika koji bi mogli da tvrde da nisu oni izvršili neku radnju i da je token bio zloupotrebljen.

Sledeći način je da se taj token čuva u sesiji na serveru. Kada korisnik izvrši neku akciju, klijentska aplikacija dodaje token na zahtev tako da provajder može da utvrdi da li korisnik ima pravo da izvrši tu operaciju.

4.6.4. Lažiranje zahteva

U prethodnoj tački razmotren je napad preuzimanja tokena i čini se da je problem sa zloupotrebom tokena rešen. Međutim, postoje napadi lažiranja zahteva sa već autentifikovanim i autorizovanim korisnikom. U tom scenariju napadač zloupotrebljava istu onu prednost veb pregledača koju koristi i protokol za prepoznavanje korisnika, a to je sesija.

Pretpostavka napada je da je korisnik već autentifikovan i autorizovan, da postoji sesija u pregledaču, kao i na aplikacijama klijenta i provajdera, zajedno sa pripadajućim tokenom. Napadač može, preko adrese elektronske pošte ili umetanja u stranicu, da prevari korisnika tako da aktivira akciju preko hiperveze čiji naziv sakriva prave namere:

`<a href="zlonamerna akcija">Dobili ste nagradu</a>`

Zlonamerna akcija može predstavljati bilo koju funkcionalnost koja je inače legitimna za aplikaciju, od običnog veštačkog podizanja broja poseta do prosleđivanja osetljivih resursa. Kadakorisnik klikne na takav link, klijentska aplikacija dodaje token na zahtev koji dalje prosleđuje na autorizaciju, koja u opštem slučaju može biti i na klijentovoj i provajderovoj strani. Ako autorizacija predstavlja samo upoređivanje pristiglog tokena sa postojećim u sesiji, akcija će biti odobrena i maliciozni kôd će se izvršiti.

Najekstremniji metod za rešavanje ovog problema može biti prisiljavanje korisnika da autorizuje svaku akciju, ali to nije dobro rešenje sa stanovišta korisničkog iskustva s

obzirom na to da bi rad aplikacije na taj način bio potpuno blokiran i korisnici bi vrlo brzo prestali da je koriste. Takođe, više puta je istaknuto da se tako korisnik navikava da često mora da autorizuje zahteve što može dovesti do toga da ne primeti kada to učini napadač.

Suptilniji načini zahtevaju malo više angažovanja kako klijenta, tako i provajdera. Provajder treba da propiše različite tipove autorizacije, od jednostavnog upoređivanja tokena do najkomplikovanijeg tipa koji može biti i višefaktorski. Shodno tome, klijent sa svoje strane treba da rasporedi svoje funkcionalnosti i da im pridruži tipove koje je propisao provajder. Kada stigne zahtev, klijent pored tokena pridružuje i tip tako da provajder može da prepozna na koji način će izvršiti autorizaciju.

Prva podela funkcionalnosti sadrži tri stavke:

- funkcionalnosti koje će obrađivati klijentska aplikacija i neće biti ni prosleđivani provajderu,
- funkcionalnosti koje zahtevaju nizak nivo autorizacije, ali je ipak potrebno da ih proveriti još jedna strana; ti zahtevi će uglavnom biti pridruženi tipu koji samo upoređuje tokene,
- funkcionalnosti koje zahtevaju visok nivo autorizacije.

4.6.4.1. Funkcionalnosti koje obrađuje klijent

U ovoj grupi nalazi se najveći broj zahteva. Tu spada većina funkcionalnosti koja je vezana za sam tip i privilegije korisničkog naloga. To su oni na koje su krajnji korisnici navikli u njih spadaju navigacija, čitanje sadržaja, prikazivanje slika, objavljivanje postova, komentara i slično. To su akcije koje svojim izvršavanjem ne povlače za sobom preuzimanje resursa, prenos vlasništva ili privilegija.

Ako postoje neki od ovakvih sadržaja koji ipak zahtevaju privatnost, klijent mora da ih izdvoji kao posebnu kategoriju i kao takve svrsta pod tip koji će imati viši nivo bezbednosti.

4.6.4.2. Funkcionalnosti koje obrađuje provajder sa nižim nivoom bezbednosti

Ovde spadaju akcije koje mogu da uključe treću stranu za deljenje sadržaja, ali ti sadržaji ne spadaju u kategoriju koji zahtevaju viši nivo privatnosti. To su javni postovi, fotografije, stavovi i slično. Ovde je važno istaći da se sama autorizacija više odnosi na verifikaciju te treće strane, nego na sam sadržaj. Većina aplikacija uključuje skripte velikih tehnoloških kompanija ili društvenih mreža.

Korisnici su navikli na ove skripte i prozore za unos kredencijala kada kliknu na dugme sa poznatim logotipom. Napadač može da umetne svoj kôd ispod poznatog

logotipa tako da korisnik klikom na dugme otvori njegovu umesto legitimne stranice. Jedan od načina za sprečavanje ovog vida napada je da klijentska aplikacija pri renderovanju stranice na serveru sve hiperveze predstavi tako da može da se proveriti da li u njima postoje veze ka drugim domenima. Takvi zahtevi dalje se prosleđuju provajderu na validaciju.

4.6.4.3. Funkcionalnosti koje obrađuje provajder sa visokim nivoom bezbednosti

Poslednja velika grupa funkcionalnosti predstavljaju deljenje poverljivih resursa, prebacivanje novca i slično i uključuje i provere iz prethodnih tačaka. Jedan scenario može predstavljati prenos novca ili neke druge vrednosti sa jednog člana klijentske aplikacije na drugog. Taj napad može biti izveden posle popunjavanja obrasca na korisničkoj aplikaciji menjanjem jedinstvenog identifikatora člana ili prosto lažiranjem zahteva.

Kada krajnji korisnik stigne sa takvim zahtevom kod provajdera prvo slede provere nižeg nivoa bezbednosti. Ako su provere uspešne, u zavisnosti od tipa zahteva može se zahtevati dodatna autentifikacija i autorizacija samog korisnika. Dalje, korisniku se ponovo prikazuju parametri zahteva. Ako je zamenjen jedinstveni identifikator kome treba da se proslede sredstva ili resursi, biće prikazano ime tog izmenjenog člana i korisnik će lako prepoznati da treba da se poništi zahtev. To takođe može biti jedan od znakova upozorenja da je klijent nepouzdan.

Isto tako, korisniku može biti ponuđeno da sačuva pregledane parametre. Ovo postaje važno u trenutku kada još nije utvrđeno da je klijent nepouzdan, bilo iz razloga da je maliciozan ili ima slabu zaštitu svog servera, s obzirom na to da promena identifikatora može da se odigra i posle autorizovanog zahteva po povratku korisnika na klijentsku aplikaciju. U tom slučaju korisnik može biti prevaren, ali je ostao trag preko kojeg može da se diskredituje klijent.

Ovo su razlozi zašto je za rizične transakcije neophodno da ih korisnik autorizuje na aplikaciji provajdera. Na taj način otežava se prevara korisnika od strane nepouzdanih klijenata. Korisnici neće verovati aplikacijama koje ih za važne zahteve ne upućuju na provajdersku aplikaciju. U tu svrhu, provajder treba korisnika pri registraciji i svakom prijavljivanju da podseća na opasnosti sa kojim može da se susretne.

5. Zaključak

5.1. Sumarni ciljevi rada i ostvareni rezultati

U radu je prikazana šema protokola za zaštitu saobraćaja pri autentifikaciji i autorizaciji. Za zaštitu, odnosno šifrovanje, korišćena je asimetrična kriptografija, uz prethodno razmenjene ključeve preko SSLprotokola. Nakon diskusije o osnovnoj šemikoja

se odnosi na autentifikaciju predstavljeni su obavezni koraci koji prethode servisu autorizacije. Takođe, predstavljeni su poznati napadi i načini kako ih sprečiti.

Shodno očekivanim doprinosima koje smo naveli u prvom poglavlju, konstatujemo da su u ovom diplomskom radu ostvareni sledeći:

- Prikazan je *OAuth* 2.0 okvir koji ima najširu primenu za servise autentifikacije i autorizacije.
- S obzirom na to da prikazani okvir obuhvata sve tipove mrežne komunikacije, u ovom radu izdvojili smo one bezbednosne rizike i napade koji se odnose na servise koji kao korisničku aplikaciju koriste veb pregledač.
- Dizajniran je sopstveni protokol za zaštitu informacija korisnika, koji se u serverskom delu, odnosno u aplikaciji koja pruža usluge autentifikacije i autorizacije oslanja na poznate načine zaštite informacija, dok klijentske aplikacije za osnovni servis autentifikacije nemaju potrebu za korišćenjem tih resursa.
- Kako protokol koristi jedinstveno korisničko ime, klijent može imati više aplikacija koje su prijavljene za servise.
- Nakon diskusije o osnovnoj šemi koja se odnosi na autentifikaciju predstavljeni su obavezni koraci koji prethode servisu autorizacije. Takođe, prikazani su poznati napadi i načini kako ih sprečiti.
- Predstavljen je potpuno novi tip servisa. Postojeća rešenja daju više slobode programerima klijentskih aplikacija, a takođe i dozvoljavaju, može se reći i podržavaju, naviku korisnika da ne vode računa o tome gde ostavljaju svoje poverljive podatke.
- Posebna pažnja posvećena je obaveštavanju krajnjeg korisnika na opasnosti po privatnost koje postoje pri korišćenju servisa na internetu. Iako ne može biti deo samog protokola, znanje korisnika može biti jedan od važnih faktora za sprečavanje napada.
- Predstavljeni su programski kodovi koji se koriste za zaštitu podataka.

5.2. Predlog za budući rad

Programsko rešenje kao podrška protokolu, kako na strani provajdera, tako i klijenta, razvijeno je u *Java Web* okruženju. U planu je razvoj klijentskog dela na drugim veb platformama, što se prvenstveno odnosi na *php*. Takođe, s obzirom na to da rad, odnosno protokol, pretpostavlja postojanje ključeva, potrebno je projektovati skladište za ključeve. Nakon toga, može se pristupiti razvoju potpunog servisa autorizacije.

Dodatak – programski kodovi

```
String un = request.getParameter("un");
String pw = request.getParameter("pw");
if (un.equalsIgnoreCase("")) {
    out.println("nije uneto korisnicko ime");
} else if (pw.equalsIgnoreCase("")) {
    out.println("nije uneta lozinka");
} else {
    KeyValueList rez = Check.loginUserProvider(idp, un, pw);
    if (rez.getID() == -1) {
        Check.redirectError(request, response, rez);
    } else {
        rez.add(new KeyValue("idn", request.getSession().getAttribute("identifier").toString()));
        String params = rez.toParams("&", "&", "=");
        AS.asda("Preuzimam podatke iz posta i kreiram get parametre");
        AS.asda(params);
        try {
            params = Proba.encryptToBase64(params, Proba.getEncKeyPath((String) request.getSession().getAttribute("name")));
        } catch (Exception ex) {
            Logger.getLogger(Auth.class.getName()).log(Level.SEVERE, null, ex);
        }
        AS.asda("Sifrat kodovan u Base64: ");
        AS.asda(params);
        String signToB64=null;
        try {
            signToB64=Proba.signTo64(params, Proba.getSigKeyPath());
        } catch (Exception ex) {
            Logger.getLogger(Auth.class.getName()).log(Level.SEVERE, null, ex);
        }
        params = "?1=" + params+"&2="+signToB64;
        AS.asda("Kreiram GET sa dva parametra 1 i 2:");
        AS.asda(params);
        String location = null;

        if (Check.isSame()) {
            location = Check.host(ClientServerURLS.getLoginServerClientTo() + params);
        } else {
            location = Check.host(request.getSession().getAttribute("urlTO").toString() + params);
        }
        request.getSession().invalidate();
        AS.asda("Redirekt URL:");
        AS.asda(location);
        AS.asda("Ukidam sesiju i saljem klijentu.");
        Check.redirect302(response, location);
    }
}
```

Slika 4 - Provera kredencijala i kreiranje parametara zahteva kod provajdera

```

String url = ClientCheck.getFromURL(request);

if (stw.auth.client.ClientCheck.validateLoginClientServerFrom(url)) {

    String b64 = request.getParameter("1");
    String b64Signature = request.getParameter("2");
    b64 = b64.replace(" ", "+");
    b64Signature = b64Signature.replace(" ", "+");
    String uri="?1="+b64+"&2="+b64Signature;
    AS.asda("GET:");
    AS.asda(uri);
    boolean status = false;
    try {
        status = verifySignatureFrom64(b64, b64Signature, Proba.getSigKeyPath());
    } catch (Exception ex) {
        Logger.getLogger(ClientAuthed.class.getName()).log(Level.SEVERE, null, ex);
    }

    if (status) {
        AS.asda("Verifikovanje uspešno!");
    } else {
        AS.asda("Verifikovanje nije uspešno!");
    }

    if (status) {
        String params = null;
        try {
            params = Proba.decryptFromBase64(b64, Proba.getDecKeyPath());
        } catch (Exception ex) {
            Logger.getLogger(ClientAuthed.class.getName()).log(Level.SEVERE, null, ex);
        }
        List<String[]> ls = AS.stringToList(params, "=", "&", "=");
        Iterator<String[]> it = ls.iterator();
        String id = it.next()[1];
        String un = it.next()[1];
        String role = it.next()[1];
        String authr = it.next()[1];
        String identifierFrom = "" + (Integer.parseInt(it.next()[1]) - 1);
        AS.asda("Preuzimam uvecani isasov iz GET-a i dekrementiram");
        AS.asda(identifierFrom);
        HttpSession session = request.getSession();
        String identifierSession = session.getAttribute("identifier").toString();
        AS.asda("Usimam isasov iz sesije: ");
        AS.asda(identifierSession);
        AS.asda("Ako su u redu:");
        AS.asda("Orde klijentski sajt odlucuje sta ce dalje da radi.");
        AS.asda("Ako nisu:");
        AS.asda("Ukida se sesija na postavljenim isasovom.");
        if (identifierSession.equalsIgnoreCase(identifierFrom)) {
            session.setAttribute("ID", id);
            session.setAttribute("name", un);
            session.setAttribute("role", role);
            session.setAttribute("authr", authr);
            ClientCheck.redirect302(response, ClientCheck.getHomeURL());
        } else {
            session.removeAttribute("identifier");
            session = null;
            ClientCheck.redirect302(response, ClientCheck.getHomeURL());
        }
    } else {
        HttpSession session = request.getSession();
        session.removeAttribute("identifier");
        session = null;
        ClientCheck.redirect302(response, ClientCheck.getHomeURL());
    }
} else {
    out.println("Authed: Bad redirect");
}

```

Slika 5 - Provera parametara zahteva kod klijenta

```
private static byte[] readKey(String file) throws Exception {
    FileInputStream fis = new FileInputStream(file);
    ObjectInputStream ois = new ObjectInputStream(fis);
    return (byte[]) ois.readObject();
}

private static PublicKey getPublicKey(String keyFile) throws Exception {
    byte[] pbkey = readKey(keyFile);
    byte[] key = new BigInteger(pbkey).toByteArray();
    X509EncodedKeySpec keySpecPublic = new X509EncodedKeySpec(key);
    KeyFactory factory = KeyFactory.getInstance("RSA");
    return factory.generatePublic(keySpecPublic);
}

private static PrivateKey getPrivateKey(String keyFile) throws Exception {
    byte[] pbkey = readKey(keyFile);
    byte[] key = new BigInteger(pbkey).toByteArray();
    PKCS8EncodedKeySpec keySpecPrivate = new PKCS8EncodedKeySpec(key);
    KeyFactory factory = KeyFactory.getInstance("RSA");
    return factory.generatePrivate(keySpecPrivate);
}

private static byte[] encrypt(String transformation, PublicKey publicKey, String plainText)
    throws Exception {
    Cipher cipher = Cipher.getInstance(transformation);
    cipher.init(Cipher.ENCRYPT_MODE, publicKey);
    byte[] message = plainText.getBytes();
    return cipher.doFinal(message);
}

private static byte[] decrypt(String transformation, PrivateKey privateKey, byte[] message)
    throws Exception {
    Cipher cipher = Cipher.getInstance(transformation);
    cipher.init(Cipher.DECRYPT_MODE, privateKey);
    return cipher.doFinal(message);
}

private static String encodeB64(byte[] bytes) {
    return java.util.Base64.getEncoder().encodeToString(bytes);
}

private static byte[] decodeB64(String text) throws Exception {
    return java.util.Base64.getDecoder().decode(text);
}
```

Slika 6 - Preuzimanje ključeva, kodovanje i šifrovanje podataka

```
public static String signTo64(String params, String keyFile) throws Exception {
    Signature rsa = Signature.getInstance("SHA1withRSA");

    rsa.initSign(getPrivateKey(getSigKeyPath() + ".pkkey"));

    InputStream poruka = new ByteArrayInputStream(params.getBytes());
    BufferedInputStream bufPoruka = new BufferedInputStream(poruka);

    byte[] buffer = new byte[1024];
    int len;
    while (bufPoruka.available() != 0) {
        len = bufPoruka.read(buffer);
        rsa.update(buffer, 0, len);
    }
    bufPoruka.close();
    poruka.close();

    byte[] digitalniPotpis = rsa.sign();
    String to64 = encodeB64(digitalniPotpis);
    return to64;
}

public static boolean verifySignatureFrom64(String params, String signatureB64, String keyFile)
    throws Exception {
    Signature signature = Signature.getInstance("SHA1withRSA");
    signature.initVerify(getPublicKey(getSigKeyPath() + ".pkkey"));
    InputStream porukaVer = new ByteArrayInputStream(params.getBytes());
    BufferedInputStream bufPorukaVer = new BufferedInputStream(porukaVer);
    byte[] digitalniPotpis = decodeB64(signatureB64);
    AS.asda("Dekodujem drugi parametar iz b64 (potpis servera):");
    AS.asda(new String(digitalniPotpis));
    byte[] buffer = new byte[1024];
    int len;
    while (bufPorukaVer.available() != 0) {
        len = bufPorukaVer.read(buffer);
        signature.update(buffer, 0, len);
    }
    AS.asda("Dodajem prvi parametar u signature buffer.");
    bufPorukaVer.close();
    boolean status = signature.verify(digitalniPotpis);
    AS.asda("Proveravam sa javnim kljucem servera:");
    return status;
}
```

Slika 7 - Digitalno potpisivanje i verifikacija potpisa

LITERATURA

- [1] M. Veinović, A. Jevremović, M. Šarac / G. Šimić, Zaštita u računarskim mrežama, Beograd: Univerzitet Singidunum, 2014, ISBN: 978-86-7912-565-1.
- [2] D. Hardt, „The OAuth 2.0 Authorization Framework,“ Oktobar 2012. [Na mreži]. Available: <https://tools.ietf.org/html/rfc6749>.
- [3] T. Ylonen / C. Lonvick, „The Secure Shell (SSH) Transport Layer Protocol,“ Januar 2006. [Na mreži]. Available: <https://tools.ietf.org/html/rfc4253>.
- [4] S. Kent / K. Seo, „Security Architecture for the Internet Protocol,“ Decembar 2005. [Na mreži]. Available: <https://tools.ietf.org/html/rfc4301>.
- [5] C. Neuman, T. Yu, S. Hartman / K. Raeburn, „The Kerberos Network Authentication Service (V5),“ Jul 2005. [Na mreži]. Available: <https://www.ietf.org/rfc/rfc4120.txt>.
- [6] M. Veinović / S. Adamović, Kriptologija 1, Beograd: Univerzitet Singidunum, 2013, ISBN: 978-86-7912-469-2.
- [7] M. Milosavljević / S. Adamović, Osnovi teorije informacija i kodovanja, Beograd: Univerzitet Singidunum, 2013, ISBN: 978-86-7912-506-4.
- [8] S. Adamović, Zaštita informacionih sistema, Beograd: Univerzitet Singidunum, 2015, ISBN: 978-86-7912-589-7.
- [9] M. Milosavljević / S. Adamović, Kriptologija 2, Beograd: Univerzitet Singidunum, 2014, ISBN: 978-86-7912-537-8.
- [10] A. O. Freier, P. Karlton / P. C. Kocher, „The Secure Sockets Layer (SSL) Protocol Version 3.0,“ Avgust 2011. [Na mreži]. Available: <https://tools.ietf.org/html/rfc6101>.