



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

INFORMACINIŲ TECHNOLOGIJŲ SAUGOS METODAI

Praktinis darbas nr. 2 - Saugi WEB autentifikacija

Atliko: ITSfm-22 grupės studentai:

Aurimas Šakalys

Aleksandr Prišmont

Vytautas Rastenis

Tikrino: lektorius Vitalijus Gurčinas

Vilnius, 2022

Turinys

1	Santrumpos	3
2	Įvadas	3
3	Kas yra <i>TLS/SSL</i>?	3
3.1	Kas yra <i>CA</i> ?	3
3.2	Kaip vykdomas sertifikato patvirtinimas?	4
4	Darbo užduoties įgyvendinimas	4
4.1	<i>CA</i> atsakomybės	4
4.1.1	Vartotojų sertifikatų generavimas	5
4.1.2	Vartotojų sertifikatų atkūrimas	5
4.1.3	Sertifikato galiojimo patikrinimas	6
4.1.4	Sertifikato pasirašymas	6
4.2	Serverio ir kliento autentikacija	7
5	2FA sprendžiamos problemos	8

Iliustracijų sąrašas

1	CSR pasirašymo diagrama	4
2	Bendro sistemos vaizdo diagrama	5
3	Kliento autentikavimo serveryje diagrama	8

Ištraukų sąrašas

1	Vartotojo sertifikato generavimo JSON užklausa	5
2	Vartotojo sertifikato generavimo JSON atsakymas	5
3	Vartotojo sertifikato atkūrimo JSON užklausa	6
4	Vartotojo sertifikato atkūrimo JSON atsakymas	6
5	Vartotojo sertifikato galiojimo patikrinimo JSON užklausa	6
6	Vartotojo sertifikato galiojimo patikrinimo JSON atsakymas	6
7	Serverio sertifikato pasirašymo JSON užklausa	7
8	Serverio sertifikato pasirašymo JSON atsakymas	7

1. Santrumpos

- *TLS - Transport Layer Security;*
- *SSL - Secure Socket Layers;*
- *SAN - Subject Alternate Name;*
- *CA - Certificate Authority;*
- *CSR - Certificate Signing Request;*
- *JSON - JavaScript Object Notation;*
- *2FA - Two Factor Authentication.*

2. Įvadas

Šio darbo tema - realizuoti *TLS/SSL* autentikaciją ir papildomą mechanizmą. Darbe formuluojami papildomi uždaviniai, norint gauti maksimalų balą:

1. Įgyvendinti atsarginę autentikacijos sprendimą;
2. Atstatyti vartotojo kredencialus;
3. Centralizuotas vartotojų prisijungimo duomenų panaikinimas arba atstatymas.

3. Kas yra *TLS/SSL*?

TLS (Rescorla, 2018) (anksčiau žinomas kaip *SSL* (Freier ir kt., 2011)) - kriptografinis protokolas, leidžiantis saugų duomenų apsikeitimą tarp interneto mazgų. *TLS* naudoja tiek simetrinį, tiek asimetrinį šifravimą. Kadangi asimetrinis šifravimas reikalauja daugiau resursų, nei simetrinis, asimetrinis šifravimas *TLS* protokole naudojamas apsikeisti simetrinio šifravimo raktu.

TLS protokole yra naudojami sertifikatai, kurie gali atlikti kelias funkcijas, kaip identifikavimas, viešojo ar privataus rakto laikmena. *TLS* sertifikatai skirstomi į kategorijas pagal apimamą domenų sritį:

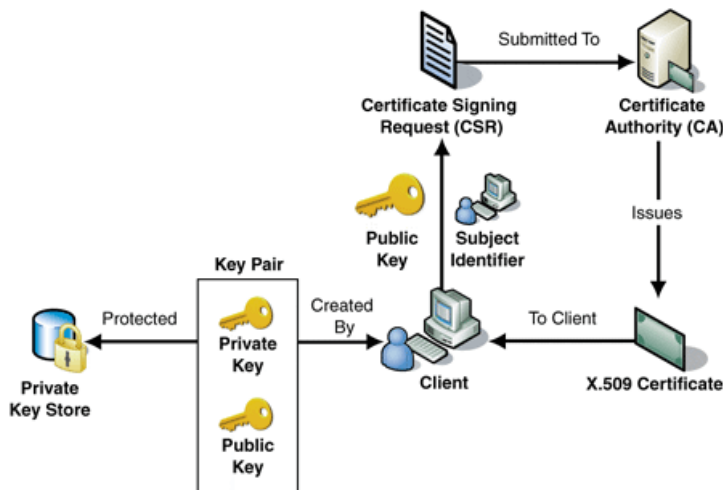
1. Vieną domeną apimantis - *SSL* sertifikatas;
2. Apimantis daugiau nei vieną domeną/subdomeną - *SSL SAN* sertifikatas;
3. Apimantis visus domeno subdomenus - *SSL Wildcard* sertifikatas;

3.1. Kas yra *CA*?

CA (Housley ir kt., 1999) yra institucija, kuri tvirtina ir išduoda skaitmeninius sertifikatus, skirtus identifikuoti tam tikrus subjektus, kaip tinklalapius, el. pašto adresus, juridinius ar fizinius subjektus.

3.2. Kaip vykdomas sertifikato patvirtinimas?

Norint gauti skaitmeninį sertifikatą valdomam subjektui, tai galima atlikti keliais būdais. Vienas jų - sertifikatą sugeneruoti ir patvirtinti pačiam. Tokiu būdu, sau išduotas sertifikatas suteikia visas kriptografines funkcijas, kaip CA pasirašytas sertifikatas. Pagrindinis skirtumas tas, jog jungiantis į interneto mazgą, kuris naudoja pačio pasirašytą *SSL* sertifikatą, mums bus patarta, jog sertifikatas nėra pasirašytas CA ir užmegzti ryšį su šio mazgu gali būti pavojinga. Taip yra todėl, kad sertifikatų grandinė baigiasi CA, kuris nėra įtrauktas į globalius patikimų CA sąrašus.



1 pav. CSR pasirašymo diagrama

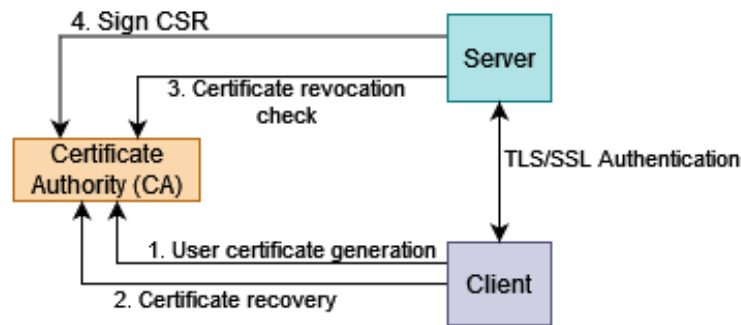
Norint, kad sugeneruotą sertifikatą pasirašytų CA, mums reikia sugeneruoti CSR sertifikatą, ir šį pateikti CA. CA turintis CSR sertifikatą, gali pasirašyti jį naudojant jų privatų raktą ir šį sertifikatą pateikti užsakovui. Šiuo atveju, jei bandoma užmegzti ryšį su mazgu, naudojančiu CA pasirašytą sertifikatą, ryšys bus užmegztas automatiškai, asimetrinio šifravimo pagalba apsikeista simetrinio šifravimo raktais, ir simetrinio šifravimo raktų pagalba, užmegztas ryšys paverčiamas saugiu.

4. Darbo užduoties įgyvendinimas

Užduotis įgyvendinta sukuriant tris posistemes: CA, serveris ir klientas.

4.1. CA atsakomybės

CA posistemė yra atsakinga už didžiąją dalį sistemos funkcionalumo. Visa komunikacija su CA posistemiui yra atliekama naudojant *HTTP RESTful API*.



2 pav. Bendro sistemos vaizdo diagrama

4.1.1. Vartotojų sertifikatų generavimas

Ši funkcija yra atsakinga už sertifikatų generavimą vartotojams. Vartotojas */generate* resursui išsiunčia *POST* užklausą, pateikdamas norimo subjekto vardą (ištrauka 1). Jei šis vardas jau yra užimtas, vartotojui pateikiamas atitinkamas klaidos pranešimas. Jei vardas nėra užimtas, gaunamas atsakymas su sugeneruotu sertifikatu *cert_pem JSON* lauke bei sertifikato atkūrimo kodu *cert_recovery JSON* lauke (ištrauka 2).

Ištrauka 1: Vartotojo sertifikato generavimo JSON užklausa

```

1 {
2   "subject": "DidysisChungusas"
3 }
  
```

Ištrauka 2: Vartotojo sertifikato generavimo JSON atsakymas

```

1 {
2   "cert_pem": "-----BEGIN CERTIFICATE -----...",
3   "cert_recovery": "1Nxp..."
4 }
  
```

4.1.2. Vartotojų sertifikatų atkūrimas

Ši funkcija atsakinga už vartotojų sertifikatų atkūrimą, naudojant sugeneruota sertifikato atkūrimo kodą (skyrius 4.1.1). Vartotojas */recover* resursui išsiunčia *POST* užklausą, pateikdamas subjekto vardą ir atkūrimo raktą (ištrauka 3). Jei atkūrimo raktas jau buvo panaudotas, ar atkūrimo raktas nėra skirtas nurodytam subjektui, vartotojui pateikiamas atitinkamas klaidos pranešimas. Jei pateikti duomenys yra teisingi, gaunamas atsakymas su sugeneruotu sertifikatu *cert_pem JSON* lauke bei sertifikato atkūrimo kodu *cert_recovery JSON* lauke (ištrauka 4).

CA serveris gavęs neklaidingus duomenis, jau išduotą sertifikatą paverčia negaliojančiu ir vartotojui sugeneruoja naują. Kadangi išduoti sertifikatai negali būti modifikuojami, sertifikato galiojimo nutraukimas yra įgyvendinamas įrašant sertifikatą į nebegaliojančių sertifikatų sąrašą.

Atkuriant sertifikatą, būtinybė pateikti subjekto pavadinimą padidina atkūrimo saugą. Jei *blogiečiai* sužino tik atkūrimo kodą, negali padaryti žalos, nes nežino subjekto vardo.

Ištrauka 3: Vartotojo sertifikato atkūrimo JSON užklausa

```
1 {
2   "subject": "DidysisChungusas",
3   "key": "1Nxp..."
4 }
```

Ištrauka 4: Vartotojo sertifikato atkūrimo JSON atsakymas

```
1 {
2   "cert_pem": "-----BEGIN CERTIFICATE-----...",
3   "cert_recovery": "KyVP..."
4 }
```

4.1.3. Sertifikato galiojimo patikrinimas

Ši funkcija atsakinga už vartotojų sertifikatų galiojimo patikrinimą. Sertifikatų galiojimo laikas yra užkoduojamas pačiame sertifikate, tačiau jei sertifikato galiojimas buvo nutrauktas anksčiau laiko, egzistuojančių sertifikatų neįmanoma modifikuoti. Dėl šios priežasties, galiojimo nutraukimui CA duomenų bazėje saugo nebegaliojančių sertifikatų sąrašą.

Vartotojas */revoked* resursui išsiunčia *POST* užklausa, pateikdamas tikrinamą sertifikatą (ištrauka 5). Priklausomai nuo sertifikato galiojimo, gaunamas atitinkamas atsakas (ištrauka 6).

Galiojimo patikrinimas sistemos įgyvendinime vykdomas CA serverio pusėje. Yra galimybė perduoti šį patikrinimą, autentikaciją validuojančiam serveriui, pateikiant nebegaliojančių sertifikatų sąrašą. Šio metodo trūkumas, kad galimai būtų siunčiamas didelis kiekis duomenų, taip pat autentikuojantis serveris gali klaidingai nustatyti, ar sertifikatas jau nebėra galiojantis.

Ištrauka 5: Vartotojo sertifikato galiojimo patikrinimo JSON užklausa

```
1 {
2   "certificate": "-----BEGIN CERTIFICATE-----..."
3 }
```

Ištrauka 6: Vartotojo sertifikato galiojimo patikrinimo JSON atsakymas

```
1 {
2   "revoked": false
3 }
```

4.1.4. Sertifikato pasirašymas

Ši funkcija atsakinga už serverio sertifikatų pasirašymą. CA serveris atlieka 3.2 skyriuje aptartą CSR pasirašymo mechanizmą. Sertifikatas pasirašomas tik *SSL* ir *SSL SAN* sertifikatams, norint

užtikrinti, kad autentikuojantys serveriai negalėtų apsimesti vienas kitu.

Vartotojas `/sign` resursui išsiunčia *POST* užklausą, pateikdamas *CSR* sertifikatą (ištrauka 7) . Jei *CSR* sertifikate pateikiami netinkami duomenys, vartotojui pateikiamas atitinkamas klaidos pranešimas. Jei *CSR* sertifikatas yra tvarkingas, serveris gauna pasirašytą sertifikatą *cert_pem JSON* lauke (ištrauka 8) .

Ištrauka 7: Serverio sertifikato pasirašymo JSON užklausa

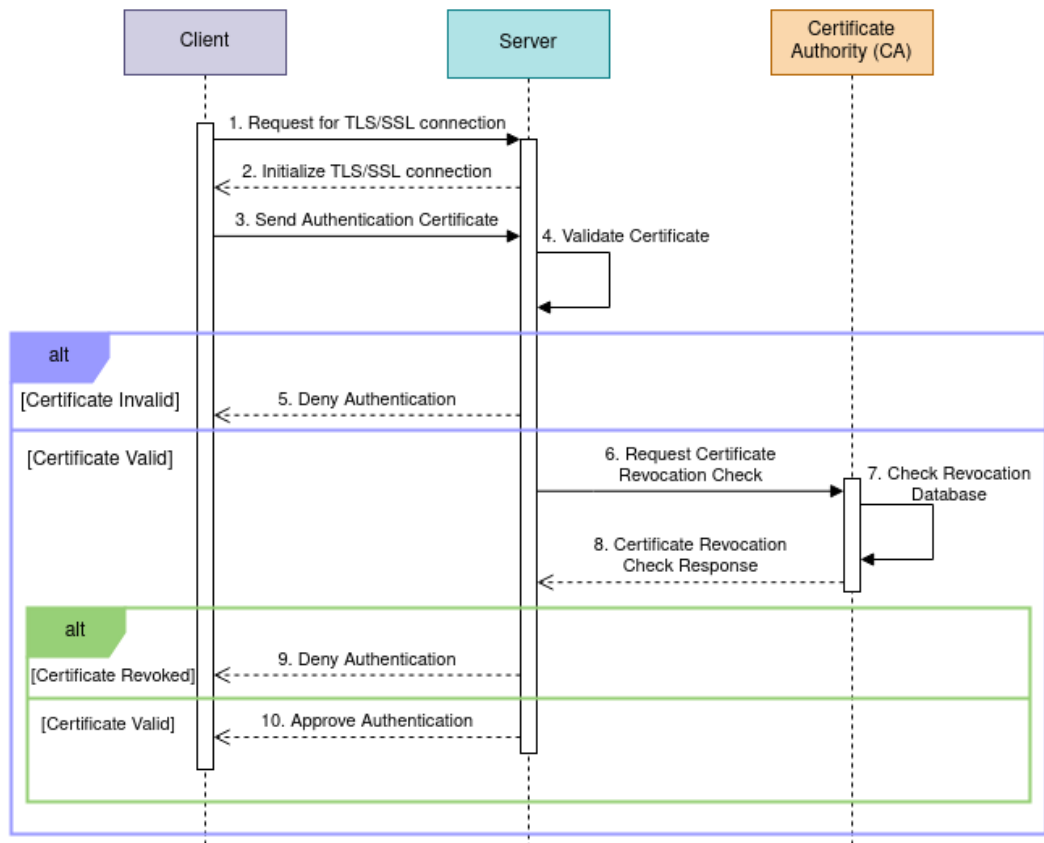
```
1 {  
2   "certificate": "-----BEGIN CERTIFICATE-----..."  
3 }
```

Ištrauka 8: Serverio sertifikato pasirašymo JSON atsakymas

```
1 {  
2   "cert_pem": "-----BEGIN CERTIFICATE-----..."  
3 }
```

4.2. Serverio ir kliento autentikacija

Klientas, jungdamasis prie serverio inicijuoja įprastą *TLS* ryšį. Iš serverio gaunamas *CA* pasirašytas sertifikatas (ištrauka 8) , kurį klientas gali validuoti naudojantis *CA* serverio sertifikatu. Jei serverio sertifikatas yra tvarkingas, užmezgamas saugus *TLS* ryšys ir pradedama vartotojo autentikacija.



3 pav. Kliento autentikavimo serveryje diagrama

Vartotojas pateikia sugeneruotą vartotojo sertifikatą (ištrauka 2). Serveris patikrina, ar sertifikatas buvo sugeneruotas CA, ar sertifikatas vis dar galioja. Jei patikrinimas yra nesėkmingas, vartotojo autentikavimas nutraukiamas. Kitu atveju, autentikavimas tęsiamas, serveris išsiunčia sertifikato galiojimo patikrinimo užklausą (ištrauka 5) CA serveriui. Jei sertifikato galiojimas nutrauktas anksčiau laiko, vartotojo autentikavimas nutraukiamas. Kitu atveju, vartotojo autentikacija baigta, vartotojas gali pasiekti serverio resursus.

5. 2FA sprendžiamos problemos

Nors užduoties įgyvendinime, 2FA nėra įgyvendintas, tačiau norint toliau didinti autentikacijos ir atstatymo procesų saugumą, turėtume prijungti 2FA implementaciją. Šį procesą įdiegus, autentikacijos metu padidėja saugumas - jei kažkoku būdu kitas asmuo bando prisijungti prie sistemos, negali šio proceso užbaigti, nes neturi prieigos prie 2FA įrenginio. Taip pat padidinamas sertifikato atkūrimo saugumas, kadangi *blogiečiams* gavus tiek atkūrimo kodą, tiek sertifikatą (ar tik subjektą), jie negalėtų padaryti žalos, neturint prieigos prie 2FA įrenginio.

Literatūra

- Rescorla, E. (2018). *RFC8446: The transport layer security (TLS) protocol version 1.3* (techn. atask.).
- Freier, A., Karlton, P., & Kocher, P. (2011). *RFC6101: The secure sockets layer (SSL) protocol version 3.0* (techn. atask.).
- Housley, R., Ford, W., Polk, W., & Solo, D. (1999). *RFC2459: Internet X. 509 public key infrastructure certificate and CRL profile* (techn. atask.). RFC Editor.