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

Iliustracijų sąrašas

Ištraukų sąrašas

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*;
- *HSM - Hardware Security Module*;
- *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 atsarginį 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:

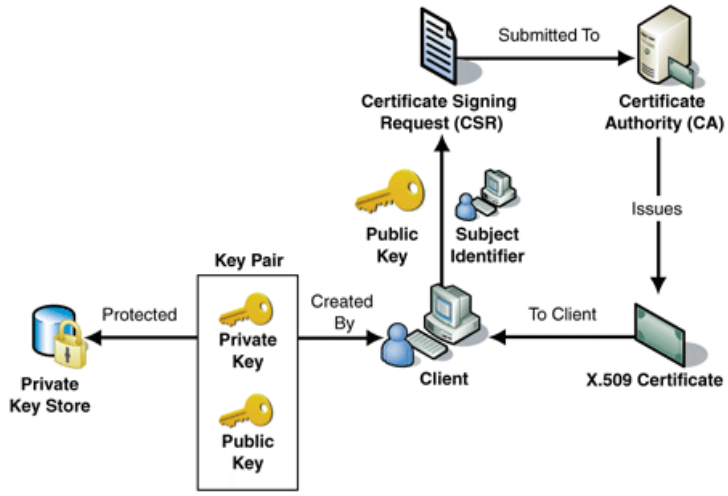
1. Vieną domeną apimantis - *SSL* sertifikatas;
2. Apimantis daugiau nei vieną domeną/subdomeną - *SSL SAN* sertifikatas;
3. Apimantis visus domeno subdomenus - *SSL Wildcard* sertifikatas;

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* pasirašytų sertifikatu, ryšys bus užmegztas automatiškai, asimetrinio šifravimo pagalba apsikeista simetrinio šifravimo raktais, ir simetrinio šifravimo raktų pagalba, užmegztas ryšys paverčiamas saugiu.

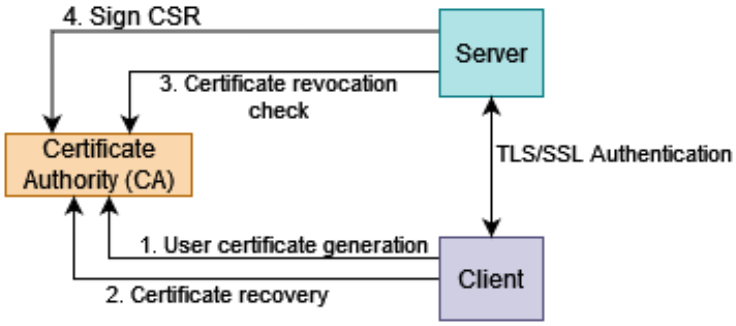
4. Darbo užduoties įgyvendinimas

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* posistemiu yra atliekama naudojant *HTTP RESTful API*.

Prieš dislokuojant *CA* posistemę, būtina sugeneruoti viešąjį ir privatų raktus, bei *CA* sertifikatą. Šių objektų reikia, norint elektroniniu parašu pasirašyti vartotojams generuojamus sertifikatus, taip pat patvirtinti ir pasirašyti sistemai perduodamas *CSR* užklausas. Praktiniu požiūriu, realioje *CA* įmonėje, raktų generavimas ir pasirašymas vyktų *HSM* prietaise. Vystomoje sistemoje raktai yra sugeneruojami prieš *CA* sistemos dislokaciją, naudojantis pagalbiniais *Python* skriptais.



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

```
{
    "subject": "DidysisChungusas"
}
```

Ištrauka 2: Vartotojo sertifikato generavimo JSON atsakymas

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

Generuojant vartotojo autentikacijos sertifikatą, užkraunami *CA* sertifikatas ir privatus raktas, naudojantis šiais, yra sugeneruojamas galutinės esybės sertifikatas. Sugeneravus galutinės esybės sertifikatą, sugeneruojamas sertifikato atkūrimo raktas.

Sertifikato atkūrimo raktas generuojamas imant atsitiktinį skaičių iš pseudo-atsitiktinių skaičių generatoriaus, jį užkoduojant *Hashid* bibliotekos pagalba. Būtent šis raktas yra pateikiamas vartotojui, kaip atkūrimo raktas.

Prie gauto rezultato prijungiant subjekto vardą, gautas rezultatas išmaišomas naudojant *SHA256* maišos funkcija ir išsaugomas į duomenų bazę.

Tiek sugeneruotas sertifikatas, tiek sugeneruotas atkūrimo kodas yra perduodami *JSON* formatu (ištrauka 4) .

4.1.2. Vartotojų sertifikatų atkūrimas

Ši funkcija atsakingą 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

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

Ištrauka 4: Vartotojo sertifikato atkūrimo JSON atsakymas

```
{
    "cert_pem": "-----BEGIN CERTIFICATE-------...",
    "cert_recovery": "KyVP..."
}
```

CA sistemai gavus užklausą, duomenų bazėje, pagal subjekto vardą, randami sertifikato metaduomenys. Subjekto vardo ir atkūrimo rakto kombinacija yra sumaišoma naudojant *SHA256* maišos funkciją, rezultatas yra palyginamas su metaduomenyse saugomu rezultatu. Jei rezultatai sutampa, dabartinis sertifikatas yra įrašomas į nebegaliojančių sertifikatų sąrašą ir yra sugeneruojamas naujas vartotojo sertifikatas ir atkūrimo raktas, naudojantis 4.1.1 skyriuje aprašytu generavimo metodu.

4.1.3. Sertifikato galiojimo nutraukimas

Ši funkcija vystomoje *CA* sistemoje tiesiogiai neegzistuoja, tačiau 4.1.1 skyriuje aprašytame metode minima, jog prieš naujo sertifikato generavimą, yra nutraukiamas prieš tai buvusio sertifikato galiojimas. Šią funkciją būtų galima lengvai įgyvendinti sistemoje.

Tiesiogiai nesant šiai funkcijai, šį funkcionalumą galima pasiekti pateikus sertifikato atkūrimo užklausą, neišsaugant *CA* sistemos pateikiamo sertifikato ir atkūrimo rakto. Niekam neturint sugeneruoto sertifikato (*CA* sistema saugo tik sertifikato metaduomenis) ir sertifikato atkūrimo rakto, sertifikatą būtų galima laikyti negaliojančiu.

Iš *CA* serverio pusės, sertifikato galiojimo nutraukimas būtų atliekamas į nebegaliojančių sertifikatų sąrašą įtraukiant sertifikatą, kurio galiojimą norima nutraukti. Vienas trūkumas dabartinėje sistemos būsenoje - vykdant sertifikato atkūrimą, nėra atsižvelgiama į tai, kad sertifikatas gali būti nebegaliojančių sertifikatų sąraše. Jei *CA* įtraukė sertifikatą į negaliojančių sertifikatų sąrašą, o šis sertifikatas yra atkuriamas, staiga, atkurtas sertifikatas tampa galiojančiu. Norint nutraukti sertifikato galiojimą, kartu tektų sugadinti ar pašalinti to sertifikato metaduomenis duomenų bazėje, kadangi neturint sertifikato metaduomenų, sertifikato atkūrimas tampa neįmanomu.

4.1.4. Sertifikato galiojimo patikrinimas

Ši funkcija atsakingą 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* užklausą, 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

```
{
    "certificate": "-----BEGIN CERTIFICATE-------..."
}
```

Ištrauka 6: Vartotojo sertifikato galiojimo patikrinimo JSON atsakymas

```
{
    "revoked": false
}
```

Galiojimo patikrinimas atliekamas gavus sertifikatą. Atliekama šio sertifikato paieška nebegaliojančių sertifikatų sąraše. Atitinkamai, radus pateikiamas atsakymas, jog sertifikatas yra nebegaliojantis, neradus - jog galiojantis.

4.1.5. Sertifikato pasirašymas

Ši funkcija atsakingą 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

```
{
    "certificate": "-----BEGIN CERTIFICATE-------..."
}
```

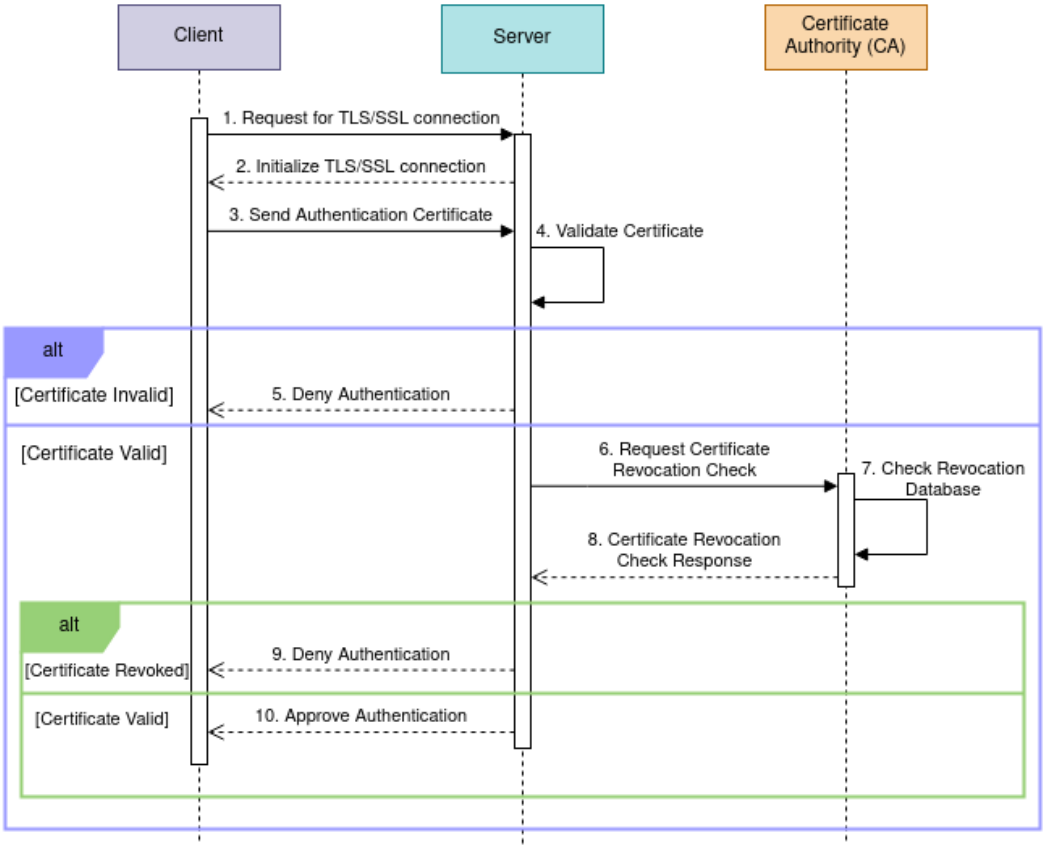
Ištrauka 8: Serverio sertifikato pasirašymo JSON atsakymas

```
{
    "cert_pem": "-----BEGIN CERTIFICATE-------..."
}
```

CA sistemai gavus *CSR* užklausą, užkraunami *CA* sertifikatas ir privatus raktas. Yra patikrinama, ar pateiktas subjektas yra unikalus. Į naująjį sertifikatą yra įrašomi pasirašančios *CA* duomenys, taip pat nukopijuojami *CSR* užklausoje pateikti duomenys, ir naujasis sertifikatas pasirašomas *CA* privačiu raktu. Sugeneruotas sertifikatas pateikiamas užsakovui.

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 kažkokiu 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 negalėtu 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.