Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Monitorovanie validity, funkčnosti a bezpečnosti internetovych SSL certifikátov

Diplomová práca

2023
Bc. Ján Kelemen

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Monitorovanie validity, funkčnosti a bezpečnosti internetovych SSL certifikátov

Diplomová práca

Študijný program:	Aplikovaná Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	FMFI.KAI - Katedra aplikovanej informatiky
Školiteľ:	prof. RNDr. Roman Ďurikovič, PhD.
Konzultant:	Peter Mihálik

Bratislava, 2023
Bc. Ján Kelemen

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Ján Kelemen
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Monitorovanie validity, funkčnosti a bezpečnosti internetovych SSL certifikátov
Monitoring the validity, functionality and security of internet SSL certificates

Anotácia: Výsledkom práce je webová aplikácia, slúžiaca na monitorovanie validity, funkčnosti a bezpečnosti SSL certifikátov internetových služieb, ktoré podporujú komunikáciu pomocou protokolu https. Pravidelne kontroluje zmeny v certifikačnej reťazi monitorovaných služieb. Vo svojej databáze uchováva všetky monitorované certifikáty, vrátane ich certifikačnej reťaze a zároveň aj ich kompletnú históriu. V prípade blížiacej sa exspirácie certifikátu alebo detekcii inej udalosti, kvôli ktorej už certifikát nebude platný alebo všeobecne považovaný za bezpečný, notifikuje používateľa.

Cieľ:
- aké certifikačné autority sú najčastejšie
- priemerný čas do exspirácie certifikátu, vrámci ktorého sa obnoví
- výskyt prípadov, kedy sa certifikát neobnoví načas, kvoli čomu služba vypadne
- vykonávanie rizikovej analýzy na základe metadát certifikátu a certifikačnej reťaze
- hodnotenie rizikového profilu vydavateľa certifikátov (niektorí vydavatelia certifikátov môžu mať zle zvládnuté manažovanie životného cyklu certifikátu, čo môže viesť k častejším neželaným exspiráciam)
- skúmať metadáta v certifikátoch a certifikačných reťaziach a hľadať také, z ktorých je možné vytvoriť čo najpresnejší rizikový profil (vydavateľ, typ certifikátu, zloženie certifikačnej reťaze, čas výmeny certifikátu pred časom jeho exspirácie, atď)
- monitorovať vydanie nových certifikátov (https://crt.sh) a merať čas, ktorý uplynie medzi ich vydaním a nasadením (často vzniká problém, kedy napr. Let's Encrypt skriptom obnoví certifikát, ale služba samotná neposkytuje klientom nový, lebo nedošlo k jeho opätovnému načítaniu z filesystemu, atď.)
- upozorňovanie klientov, ktorých identifikujeme ako rizikových na možné problémy pri obnove certifikátu

Literatúra:
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3425554
https://conferences.sigcomm.org/imc/2011/docs/p427.pdf
https://eprint.iacr.org/2013/538.pdf

Vedúci: prof. RNDr. Roman Ďurikovič, PhD.
Konzultant: Mgr. Peter Mihálik
Katedra: FMFI.KAI - Katedra aplikovanej informatiky

15903807

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 05.12.2022

Dátum schválenia: 05.12.2022 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Sem príde poďakovanie.

Abstrakt

Tu bude abstrakt po slovensky.

Abstract

Here will be the abstract in English.

Obsah

Zoznam obrázkov

Úvod

V dnešnej dobe sa nikto z nás nezaobíde bez internetu. Každý deň navštevujeme množstvo internetových portálov, ktorým poskytujeme svoje citlivé informácie a preto je dôležité, aby tieto citlivé informácie neskončili v rukách útočníka. Jedným z najdôležitejších nástrojov ako predísť tomu, aby sa útočník mohol vydávať za nami navštevovanú webovú lokalitu je internetový protokol https, nadstavba na protokol http, ktorý prenáša dáta zašifrované a na svoje fungovanie využíva SSL certifikáty. SSL používa na overenie totožnosti asymetrickú šifru. Počas takzvaného SSL handshaku medzi používateľom a severom prebehne výmena kľúčov a tým sa overí identita serveru. Tieto kľúče sú podpisované certifikačnými autoritami, ktorými môžu byť napríklad komerčné alebo neziskové organizácie, prípadne štátne orgány.

Tieto SSL certifikáty ale po určitom čase exspirujú alebo môžu byť počas svojho života revokované autoritou, ktorá ho vydala. Prípadne môžu byť zle nasadené na server alebo zle vydané, pretože držiteľ certifikátu zle vyplnil údaje pri žiadaní o certifikát.

Bratislavská informatická spoločnosť v ktorej pracujem, bonet.systems, sa primárne zaoberá správou a údržbou internetových služieb iných firiem. Z mnohoročných skúseností zamestnancov bonet.systems vyplynulo, že väčšina našich zákazníkov, vrátane veľkých korporácií, si nijakým systematickým spôsobom nesleduje životný cyklus certifikátov. Prípadne údaje o nich majú zaznamenané len v Excel tabuľke, čo vedie k mnohým haváriám, keď certifikáty prestanú byť platné. Preto sa ma moja spoločnosť rozhodla poveriť vývojom systému, ktorý by pravidelne vykonával monitorovanie a analýzu SSL certifikátov zákazníkov, ktorí o to požiadajú.

Celý systém ktorý vyvíjam pozostáva zo štyroch častí. Prvá časť je webová aplikácia, na ktorej si zákazník vedie pozrieť prehľad všetkých svojich monitorovaných certifikátov. Druhá časť je knižnica, ktorá na vstup dostane certifikáty, analyzuje ich a na výstup pošle výsledok analýzy. Tretia časť sú mikroslužby, ktoré pravidelne touto knižnicou skenujú certifikáty uložené v databáze a posledná štvrtá časť je aplikácia takzvaného agenta, ktorá sa dá nainštalovať na servery zákazníkov a ktorá v pravidelných intervaloch odosiela nájdené certifikáty do centrálnej databázy na analýzu mikroslužbami.

Vývoj tohto systému začal už približne pred rokom a pol a v čase písania tejto diplomovej práce sa už jeho prvá verzia ponúka zákazníkom, ktorí ho testujú na svojich

internetových službách. V tejto diplomovej práci som sa rozhodol zozbierané dáta od zákazníkov, ktoré sa nachádzajú v databáze analyzovať. Taktiež okrem certifikátov našich zákazníkov som zanalyzoval čo najväčší počet internetových portálov s doménou .sk, aby som získal širší pohľad na to, aké chyby vznikajú v certifikátoch aj pri iných slovenských webových lokalitách. Následne plánujem porovnať výsledné dáta z analýzy slovenských portálov s podobnými prácami z iných krajín, ktoré analyzovali certifikáty internetových stránok v svojej domovskej krajine.

Na začiatok v prvej kapitole si ukážeme ako prebieha už spomínaný SSL handshake, chyby, ktoré môžu pri certifikátoch nastať a už existujúce riešenia, ktoré tieto chyby odhaľujú.

V druhej kapitole opíšem návrh celého systému - od webovej aplikácie cez miktroslužby a aplikáciu agenta až po knižnicu, ktorú tieto mikroslužby a agent využívajú. Zároveň tiež opíšem výzvy spojené s analýzou veľkého počtu certifikátov a ako som sa s nimi vysporiadal. Ukážem aj ako webová aplikácia vyzerá, ako prebieha nasadzovanie aplikácie agenta u zákazníkov a akú má štruktúru databáza. Taktiež uvediem zopár ukážok kódu z implementácie jednotlivých komponentov systému. Väčšina týchto komponentov je písaných v jazyku Go, ktorý síce nie je na Slovensku až tak veľmi rozšírený, avšak čitateľ ktorý je schopný programovať v jazykoch C a Python by tento kód mal vedieť pochopiť.

V ďalšej tretej kapitole opíšem metódy, ktoré som použil pri zbere dát, ktoré ďalej používam v mojej analýze. Opíšem ako som získal zoznam všetkých slovenských domén, ako som z nich posťahoval ich SSL certifikáty a ako ich následne môj systém analyzoval.

S štvrtej kapitule už pracujem so zozbieranými dátami z analýz certifikátov všetkých webových stránok s doménou .sk a zákazníkov bonet.systems. Zozbierané dáta zagregujem, znázorním na grafoch a urobím z nich závery, ktoré z nich vyplývajú a tiež odporúčania, ktoré by sa dali implementovať, aby nedochádzalo k najčastejším chybám, ktorá moja analýza odhalila.

V piatej, poslednej kapitole porovnám moje výsledky s výsledkami iných podobných prác, ktoré analyzovali veľký počet certifikátov na národnej úrovni v iných krajinách.

Kapitola 1

Základné pojmy, chyby v certifikátoch a existujúce riešenia

Dnešná internetová doba si taktiež vyžaduje je obzvlášť dôležitá internetová bezpečnosť. Keď nadviažeme spojenie s internetovou stránkou, prípadne keď náš program nadviaže spojenie s nejakým iným programom dostupným cez internet je dôležité, aby sme dokázali overiť, že komunikujeme naozaj so severom, s ktorým sme chceli komunikovať. Pokiaľ nedokážeme identitu webového servera overiť, môže sa stať, že to využije útočník, ktorý sa môže zmocniť našich osobných údajov.

Na overenie totožnosti slúžia SSL certifikáty, ktoré vydávajú certifikačné autority, ktoré sú overené vyššími certifikačnými autoritami. Vzniká tu akási reťaz dôvery, ktorá ide od vydaného certifikátu pre konkrétnu doménu alebo skupinu domén až po najvyššiu koreňovú certifikačnú autoritu, ktorých je v čase písania tejto diplomovej práce na svete okolo 150.

Tieto certifikáty alebo môžu exspirovať, byť revokované, zle nasadene, prípadne pri nich môžu nastať iné defekty, ktoré sa v tejto kapitole pokúsim ozrejmiť. Na odhaľovanie týchto chýb už existujú programy, ale ako sa pokúsim vysvetliť, neexistuje žiadne také riešenie, ktoré by spĺňalo požiadavky, ktoré forma v ktorej pracujem požaduje od tohoto systému.

1.1 SSL handshake a šifrované spojenie

Overenie identity pomocou SSL certifikátu začína takzvaným SSL handshakom, ktorý je výpočtovo najzložitejší a zároveň kľúčový v celkom procese šifrovanej komunikácie. V súčasnoti sa používa namiesto SSL štandardu používa jeho novšia forma TLS, preto teraz popíšem ako prebieha šifrované spojenie v TLS. Konkrétne vo verzií 1.2, pretože na nej sa najlepšie demonštruje ako spomínaný štandard funguje. Existuje aj novšia verzia, v čase písania tejto práce je najnovšia verzia 1.3, ale tá ešte nie je až tak

veľmi rozšírená, podľa Internet Engineering Task Force ju používa len 27% webových spojení [4] a podľa mňa pridáva navyše len zopár malých vylepšení oproti 1.2, čo by len zbytočne komplikovalo vysvetľovanie.

SSL handshake prebieha v nasledujúcich piatich krokoch. Taktiež sa predpokladá, že čitateľ ovláda ako funguje Diffieho–Hellmanova výmena kľúčov.

1. V prvom kroku client odošle správu, takzvané ,,Client Hello'', v ktorej špecifikuje najvyššiu verziu TLS, ktorú podporuje a súbor šifier (anglicky cipher suite). Server si vyberie jednu z týchto šifier, ktorá predstavuje algoritmus pomocou ktorého šifrovaná komunikácia prebieha, napríklad RSA alebo ECDH a verziu TLS pomocou ktorej chce komunikovať. Následne odpovie správou ,,Hello Server'', v ktorej tieto voľby zahrnie. Pokiaľ server nepodporuje žiadnu zo šifier ktorú klient poslal, pošle naspäť TLS allert a spojenie sa preruší.

2. Server následne pošle TLS certifikát spolu s jeho privátnym kľúčom. Pošle tiež ,,server-key exchange'' správu, v ktorej sa nachádzajú hodnoty ,,g'' a ,,n'' potrebné pre Diffie–Hellman asymetrickú komunikáciu a hashsum predošlých správ podpísaný jeho privátnym kľúčom servera. Pošle aj server hello done správu, ktorá znamená, že už nevie ponúknuť žiadne dodatočné informácie.

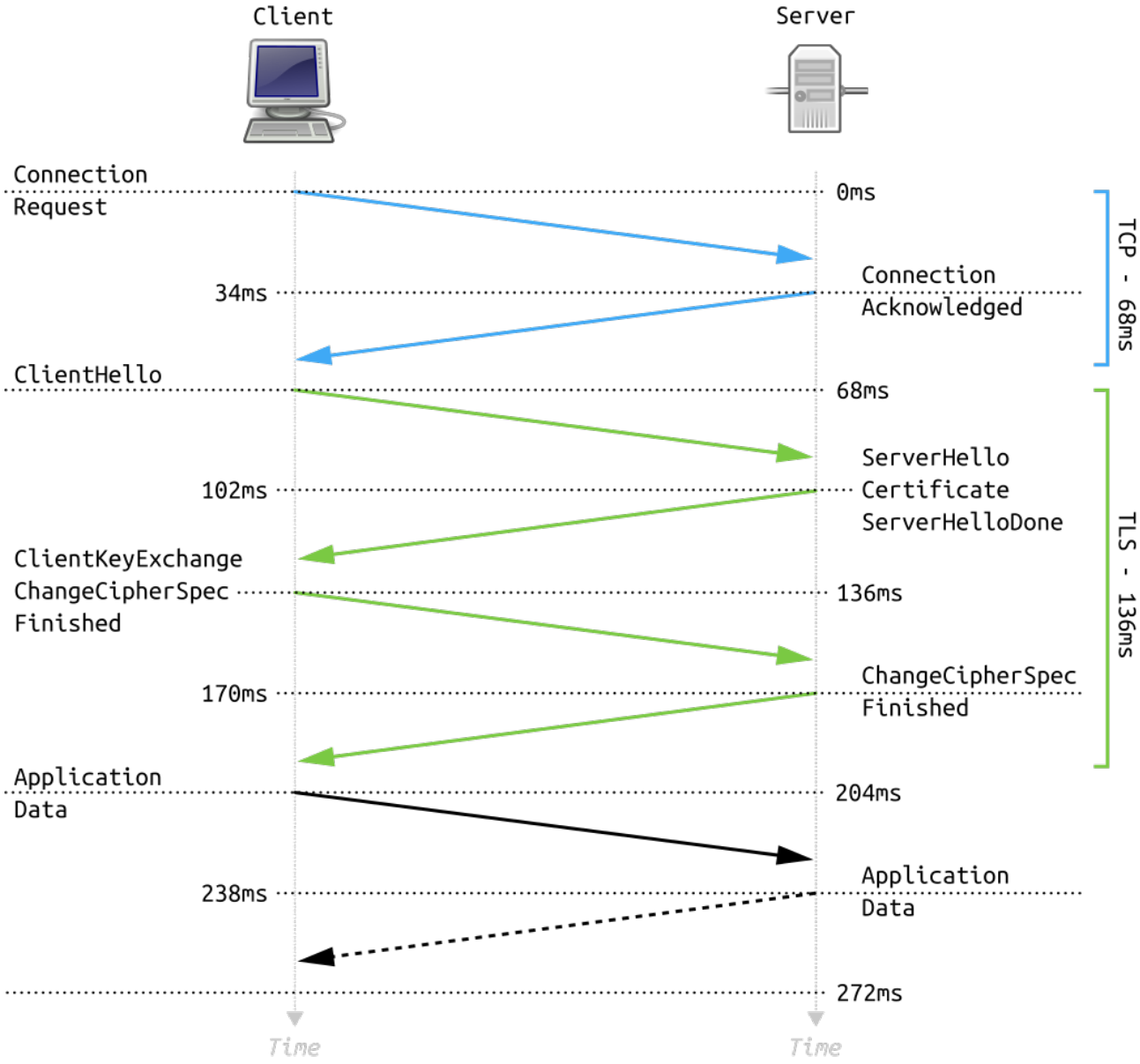
3. Klient sa pozrie či hodnoty v certifikáte a checksum sú v poriadku. Klient má u seba zoznam všetkých najvyšších koreňových autorít, teda si vie poskladať reťaz dôvery k jednej z nich. Ak sa táto reťaz nedá vytvoriť znamená to, že certifikát je neplatný, server nevie overiť svoju identitu a šifrované spojenie nemôže nastať.

4. Po úspešnom vytvorení reťaze, klient pošle ,,client key exchange'', v ktorom pošle verejnú časť svojho kľúča. Tiež pošle ,,change cypher spec'' správu, ktorá signalizuje, že ďalšie správy od klienta už budú len šifrované pomocou dohodnutého šifrovania. Ako tretiu vec pošle finished správu, v ktorej sa nachádza checksum všetkých predošlých správ.

5. Na záver server pošle tiež ,,cypher spec'' správu a ,,finished'' správu, ktoré reprezentujú to isté ako pri klientovi. Prvotné verzie SSL nemali tieto ,,finished'' správy, ale neskôr sa implementovali aby sa predišlo útokom, kedy útočník naruší proces handshaku a klient nadviaže šifrované spojenie s útočníkom, nie so serverom.

Po úspešnom handshaku prebieha už symetrická šifrovaná komunikácia pomocou dohodnutého alogitmu na šiforvanie, ktorá je oveľa rýchlejšia na výpočet ako asymetrická.

Sýstém ktorý je výsledkom tejto práce ukladá každý certifikát, ktorý dostane od servera, aj taký pomocou ktorého sa nedá nadviazať https spojenie. Chyby ktoré sa v



Obr. 1.1: Diagram SSL handshaku

tomto certifikáte nájdu sú následne hlásené používateľovi, ktorý danú webovú lokalitu monitoruje.

1.2 Chyby v certifikátoch

Chyby v certifikátoch môžu nastať rôzne. Rozdelím ich do troch kategórií.

1.2.1 Chyby spôsobené zlým nasadením na server

Tieto chyby viem odhaliť len u klientov, ktorý používajú náš firemný produkt na monitorovanie certifikátov, keďže pri návšteve iných stránok sa nemám ako dostať k

informáciám v akých priečinkoch server uchováva certifikáty.

Súbor s certifikátom a jeho privátnym kľúčom by mali byť uložené na takom mieste, aby sa k ním mohol webový server poskytujúci službu pre ktoré sú certifikáty určené dostať. Môže sa stať, že webový server do tohoto priečinka nemá prístup a SSL spojenie nemôže nastať. O tejto chybe sa dozvieme z logov webového servera alebo z jeho konfiguračného súboru, kde je špecifikovaná cesta k priečinku s certifikátom.

Taktiež sa môže stať, že bol certifikát z nejakého dôvodu vymazaný alebo uložený do zlého priečinku. Tieto všetky chyby by mala aplikácia agenta odhaliť, keďže ona je nainštalovaná u zákazníkov na serveri a analyzuje nájdené certifikáty a hľadá nové.

1.2.2 Chyby spôsobené chybným vydaním certifikátu

Pri žiadaní o certifikát od autority správca serveru vypĺňa dotazník, do ktorého dáva rôzne údaje, napríklad webovú lokáciu servera. Pokiaľ webovú lokáciu zle napíše, certifikát je neplatný. Vydaný certifikát môže mať tiež rôzne defekty:

- Napríklad mu môžu chýbať niektoré políčka, ktoré sú povinné. Povinné sú také, ktoré sú špecifikované ako povinné v štandarde RFC 5280 [5].

- Nachádza sa v ňom zlá Signed Certificate Timestamp (SCT). SCT môžeme chápať ako databázu všetkých vydaných certifikátov a pokiaľ sa v nej náš certifikát nenachádza znamená to, že buď bol certifikát označený ako zlý danou autoritou alebo autorite bolo odobrané povolenie vydávať certifikáty.

- Certifikát bol vydaný sebou samým. Toto nie je problém pokiaľ bol certifikát vydaný koreňovou najvyššou autoritou, s čím musí môj systém počítať, ale v ostatných prípadoch je takýto certifikát neplatný.

- Certifikát bol odvolaný autoritou, ktorá ho vydala. Väčšinou sa to stáva ak dané webové sídlo obsahuje ilegálny obsah alebo obsah, s ktorým autorita nechce byť spájaná.

1.2.3 Chyby spôsobené zanedbaním monitorovania

Každý certifikát má svoj životný cyklus. Je vydaný, používaný a po určitom čase exspiruje. Jedna z mikroslužieb v mojom systéme má za úlohu pravidelne monitorovať či niktorý certifikát nejde exspirovať za špecifikovaný čas a ak áno, pošle notifikáciu používateľovi, Okrem špecifikovaného dátumu dokedy je certifikát platný má certifikát tiež políčko odkedy je platný. Pokiaľ je certifikát nasadený skôr ako je platný, používateľ taktiež dostane notifikáciu od systému.

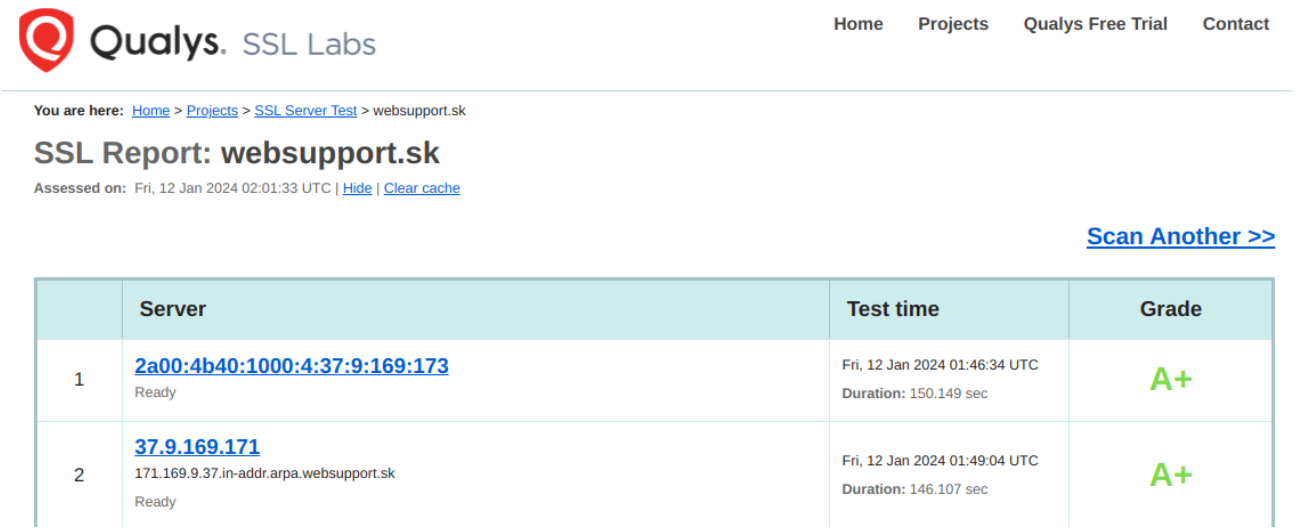
Obr. 1.2: Ukážka z používteľského webového rozhrania sytému. Certifikát ukázanej stránky čoskoro exspiruje

1.3 Podobné už existujúce riešenia

Napriek tomu, že presne také riešenie aké moja firma požadovala neexistuje, existujú podobné riešenia, ktorá taktiež riešia validáciu SSL certifikátov.

1.3.1 SSL Labs od Qualys

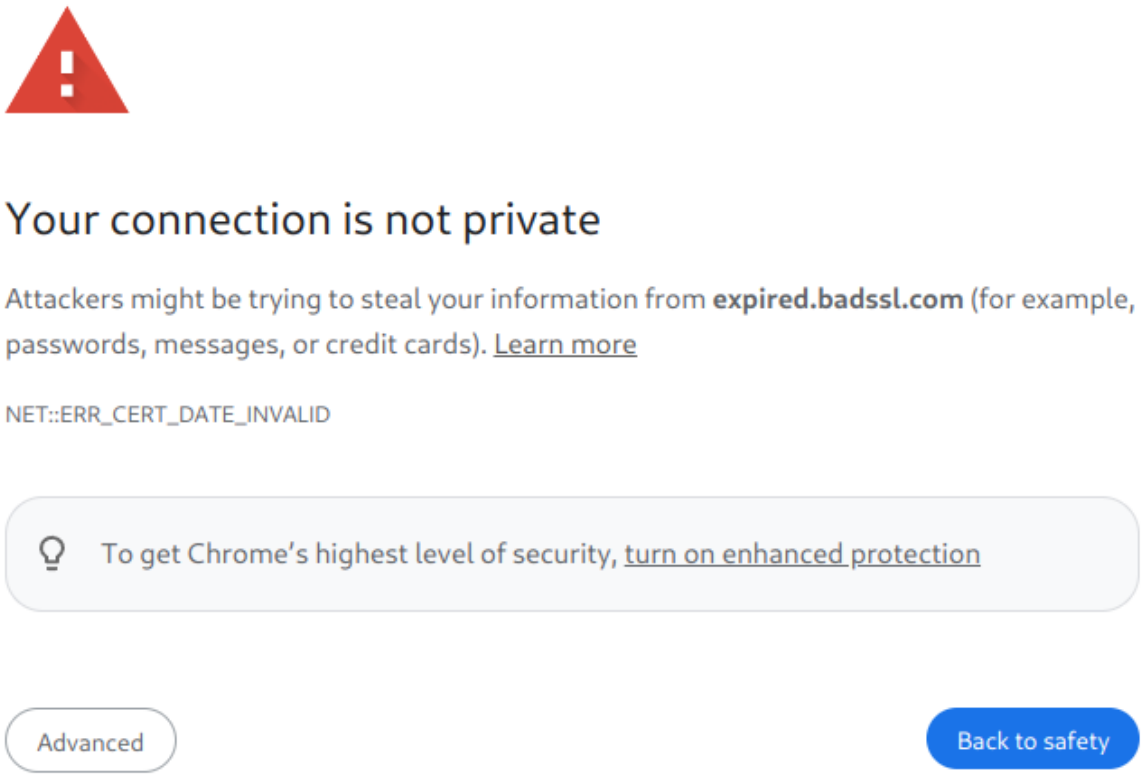
SSL Labs [9] je closedsource platené riešenie od spoločnosti Qualys. Toto riešenie je veľmi robustné, ale oproti mnou vyvíjanému systému neposkytuje aplikáciu agenta, ktorá odosiela zo servera zákazníka na analýzu certifikáty, ktoré inak z internetu nie sú viditeľné. Taktiež agent disponuje mechanizmami, ktoré vedia odhaliť certifikát na disku. Tiež čas ktorý trvá SSL Labs skontrolovať jednu webovú stránku je pomerne dlhý, až niekoľko minút, čo je oproti môjmu riešeniu niekoľkonásobne pomalšie, čo zapríčiňuje, že jednotlivé webové lokality nemôžu byť až tak často monitorované a ich certifikáty analyzované. Dlhé medzery medzi jednotlivými kontrolami môžu spôsobiť neskoré odhalenie problému.



Obr. 1.3: Ukážka z aplikácie SSL Labs od Qualyse

1.3.2 Existujúce webové priehliadače

Všetky súčasné prehliadače majú opensource renderovacie jadro. Firefox má jadro nazývané Gecko [6], Google Chrome má Chromium [7] a Safari má Webkit [8]. Ako prvé som zamýšľal na kontrolu certifikátov používať niektoré z nich, to sa ale ukázalo ako nie veľmi dobré rozhodnutie, pretože prehliadače si kontrolu certifikátov implementovali podľa seba a nepostupovali pritom podľa RFC štandardov. Taktiež množstvo kódu, ktoré sa nachádza v týchto jadrách je obrovské, čo komplikuje vývoj kvôli dlhým kompilačným časom.



Obr. 1.4: Ukážka chybovej správy, ktorú Chromium zobrazí pri detekcií zlého certifikátu

1.3.3 Openssl

Openssl [10] je opensource program, ktorý pracuje s SSL certifikátmi. Problém ale je ten, že vždy vracia prvú chybu ktorú nájde, teda pokiaľ certifikát obsahuje viac ako jednu chybu, odhalí len jednu. Taktiež napríklad nevie hlásiť viac granulárne chyby. Napríklad pokiaľ v certifikáte chýba povinné políčko, nezahlási, že aké konkrétne chýba.

```
SSL handshake has read 5003 bytes and written 452 bytes
Verification error: certificate has expired
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES128-GCM-SHA256
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
```

Obr. 1.5: Ukážka z výpisu programu Openssl po zadaní príkazu openssl s_client -connect 104.154.89.105:443 -servername expired.badssl.com

Kapitola 2

Návrh systému pre zber a analýzu SSL certifikátov

2.1 Použité technológie

2.1.1 Technológie použité na strane servera

Technológie použité na strane servera sa taktiež označujú pojmom ,,backend'' alebo ,,server side''.

Go

Golang [11][12][13], alebo len Go, je open-source typový, kompilovaný programovací jazyk, ktorý vytvorili Robert Griesemer, Rob Pike a Ken Thompson v spoločnosti Google v roku 2009. Tento jazyk je syntakticky podobný jazyku C, ale na rozdiel od C používa garbage collection a taktiež je memory safe. Memory safe znamená, že sa počas runtime nedovolí pristupovať k nesprávnym adresám v pamäti.

V tomto jazyku mám naprogramovanú najväčšiu časť backendu.

Medzi hlavné výhody tohto jazyka patrí:

- Jednoduchá syntax. Go má tak isto ako C veľmi jednoduchú syntax. Je možné sa naučiť celý tento jazyk v priebehu jedného týždňa. Vďaka jednoduchej syntaxi je potom napísaný kód ľahšie čitateľný.

- Dobrá podpora konkurencie. Na konkurenciu používa goroutiny, čo sú Go implementácia green threadov, ktoré medzi sebou komunikujú pomocou posielania správ, takzvaných kanálov. Taktiež z nekonkurentného programu sa veľmi ľahko, pridaním zopár riadkov, dá urobiť konkurentný program.

- Extrémne rýchle kompilovanie. Go kompilátor používa bootstrapping, čo znamená, že celé Go je napísané v Go, vďaka čomu dokáže byť väčšina aplikácií skompilovaná do pár sekúnd. Go sa taktiež defaultne kompiluje do statických binárnych súborov, takže spolu s binárnym súborom nie je potrebné dodávať aj knižnice, ktoré boli používané.
- V štandardnej knižnici sa nachádza veľmi veľa užitočných balíkov. Napríklad balík net poskytuje funkcionalitu pre vytváranie http servera, balík flag slúži na vytváranie command line aplikácií, alebo pomocou balíka testing sa dajú písať jednotkové testy.
- Prehľadná online dokumentácia, v ktorej sa ľahko orientuje.

Hlavné nevýhody tohto jazyka:

- Jednoduchosť tohto jazyka je zároveň aj jeho nevýhoda. Go napríklad neobsahuje žiadne funkcie vyššieho rádu, ktoré by umožňovali prácu s poliami, ako napríklad map alebo filter.
- Error handling je riešený pomocou návratovej hodnoty, podobne ako v jazyku C, a nie pomocou try-catch blokov ako vo väčšine moderných programovacích jazykov. Toto má za príčinu časté opakovanie bloku

```
if err != nil {
    return err
}
```

, čo môže robiť kód menej prehľadným.

Laravel

Laravel [21] patrí v súčasnoti medzi najpoužívanejšie php frameworky. Využívam ho vo webovej aplikácií na autentifikáciu používateľa, poskytovanie REST API aplikácií agenta a obsluhovanie celej webovej aplikácie. V práci bolo odo mňa požadované tento framework používať, pretože jeho popularita zabezpečuje, že bezpečnosť celej webovej aplikácie je na dobrej úrovni.

PostgreSQL

PostgreSQL [16], alebo len Postgres, je open-source relačný databázový systém, ktorý je kompatibilný s SQL štandardom.

V relačnej databáze sú v každej tabuľke uložené údaje týkajúce sa jednej oblasti. Tabuľky sa v relačnej databáze nazývajú reláciami. Tieto relácie sú potom na základe vzájomných vzťahov navzájom prepojené.

Postgres som si zvolil spomedzi databázových serverov, pretože je podľa mňa v súčastnosti najvyspelejším relačným databázovým systémom, a taktiež mám s ním už skúsenosti zo školy.

V mojej aplikácií mám v Postgrese uložené dáta o používateľoch a certifikátoch.

Redis

Remote Dictionary Server [15], alebo tiež Redis, je in-memory key–value databáza. Defaultne neposkytuje durabilitu, čiže negarantuje, že dáta zapísané prebehnutými transakciami prežijú pád systému. Durabilita sa ale dá aktivovať na úkor efektivity databázového serveru. Podporuje taktiež veľa dátových typov, ako sú napríklad zoznamy, množiny, asociatívne polia a JSON objekty.

Pre svoju vysokú efektivitu sa Redis najčastejšie používa ako cache. V mojej aplikácií ho využívam pri asynchrónnej komunikácií s aplikáciou agenta.

Ubuntu server

Ubuntu server [17] je v súčastnosti najpopulárnejšia serverová linuxová distribúcia. Dve časti môjho systému - webová aplikácia a mikroslužby bežia na Ubuntu serveri 22. Verzia 22 je označená ako long-term support verzia, čo znamená, že bude podporovaná najmenej do roku 2032. Linux som si vybral z dôvodu, že je to najpopulárenjší operačný systém, ktorý sa používa na strane servera.

Nginx

Nginx [18] je open-source softwarový webový server s load managmentom a reverznou proxy, vytvorený Igorom Sysoevom a vyvíjaný od roku 2004. Zmeriava sa predovšetkým na vysoký výkon a nízke pamäťové nároky. Spolu s Apache Server patria medzi najpoužívanejšie http a reverz proxy servery. Ja ho v mojej aplikácií používam na presmerovanie https requestov na moju webovú aplikáciu, ktorá beží na localhoste, čiže ho používam ako reverz proxy.

Certbot

Certbot [19] je open-source nástroj pre automatické vytváranie a obnovovanie X.509 certifikátov. Dokáže spolupracovať s Nginx, ktorý potom tieto certifikáty používa pri spracovávaní https requestov. Certbot získava certifikáty od Let's Encrypt.

Let's Encrypt je non-profit certifikačná autorita patriaca pod Internet Security Research Group, ktorá udeľuje X.509 certifikáty zadarmo. V súčastnosti to je najväčšia certifikačná autorita, až 10% všetkých udelených certifikátov pochádza od nej.

2.1.2 Technológie použité na strane klienta

Tieto technológie sa taktiež označujú ako ,,frontend'' alebo ,,client side''.

Alpine.js

Alpine.js [20] je ľahký, open-source frontendový framewrok, ktorý pridáva interaktivitu na webovú stránku. Syntakticky je dosť podobný napríklad frameworku Vue.js.

Bootstrap

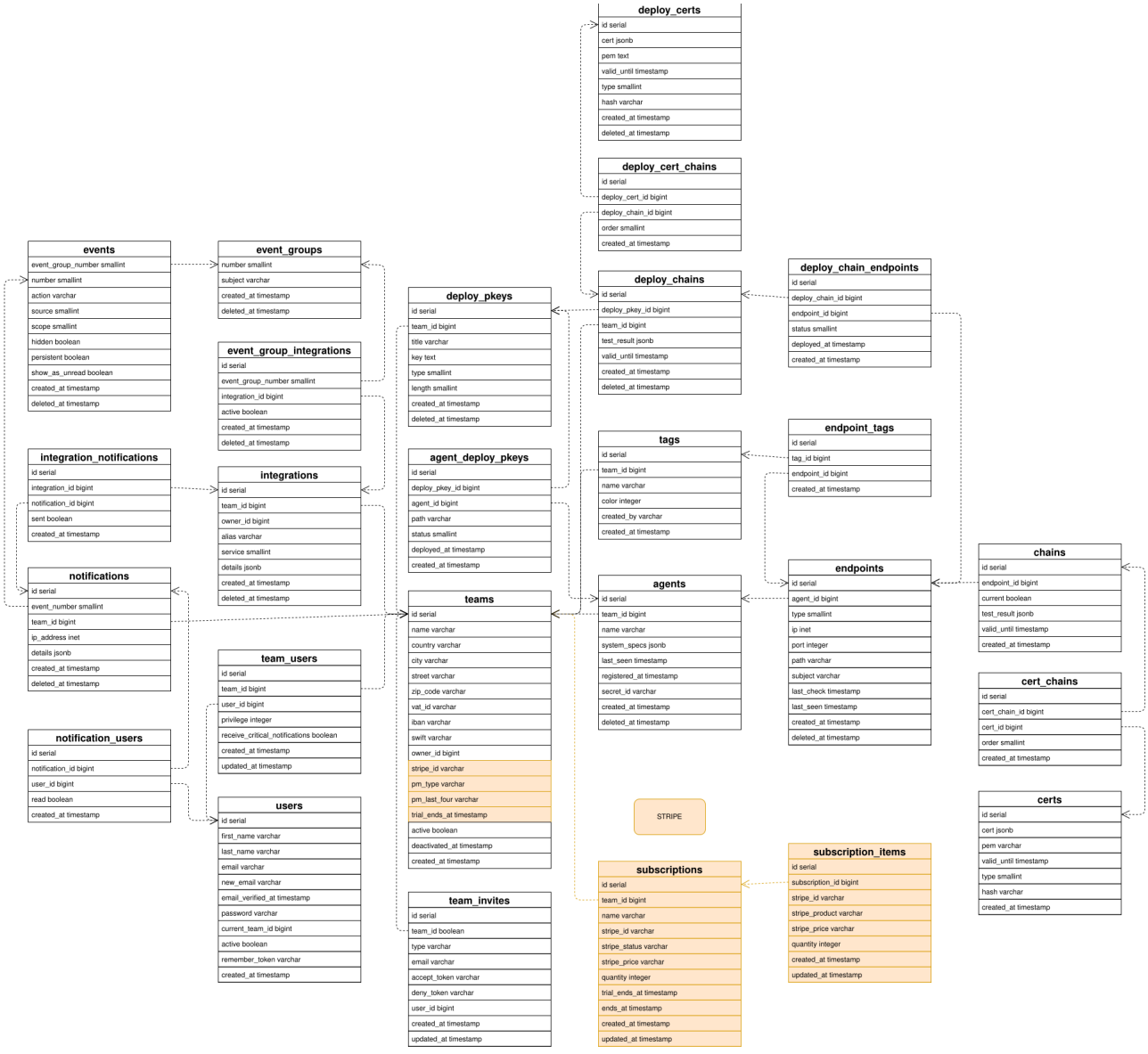
Bootstrap je bezplatný a open-source front-end framework na vývoj webových aplikácií. Je napísaný v jazykoch HTML, CSS a JavaScript a poskytuje súbor predpripravených komponentov a nástrojov, ktoré možno použiť na vytváranie responzívnych návrhov. Spolu s alpine.js mi značne uľahčuje písanie javascriptu a css.

2.2 Implementácia webovej aplikácie

2.3 Implementácia mikroslužieb

2.4 Implementácia aplikácie agenta

2.5 Implementácia relačnej databázy



Obr. 2.1: Entitno relačný model databázy

Kapitola 3

Zbieranie dát

V tejto kapitole opíšem ako som zoznam všetkých webových stránok s doménou .sk, posťahoval z nich SSL certifikáty, uložil ich do databázy a spustil nad nimi analýzu, ktorú systém pravidelne vykonáva. Taktiež popíšem výzvy, ktorým som čelil pri takomto veľkom objeme dát.

Kapitola 4

Analýza zozbieraných dát

...

Kapitola 5

Porovnanie mojich výsledkov s inými prácami

...

Záver

Vo mojej práci som sa venoval ....

Možné vylepšenia v budúcnosti

...

Nadobudnuté poznatky

...

Literatúra

[1] A Complete Study of P.K.I. (PKI's Known Incidents). Dosptupné on-line: papers.ssrn.com/sol3/papers.cfm?abstract_id=3425554, citované v januári 2024.

[2] The SSL Landscape – A Thorough Analysis of the X.509 PKI Using Active and Passive Measurements. Dosptupné on-line: conferences.sigcomm.org/imc/2011/docs/p427, citované v januári 2024.

[3] Practical Issues with TLS Client Certificate Authentication. Dosptupné on-line: eprint.iacr.org/2013/538.pdf, citované v januári 2024.

[4] TLS 1.3: One Year Later on-line: ietf.org/blog/tls13-adoption, citované v januári 2024.

[5] RFC 5280 on-line: datatracker.ietf.org/doc/html/rfc5280, citované v januári 2024.

[6] Gecko - Firefox on-line: firefox-source-docs.mozilla.org/setup/index.html, citované v januári 2024.

[7] Chromium on-line: chromium.googlesource.com/chromium/src.git, citované v januári 2024.

[8] Webkit on-line: webkit.org, citované v januári 2024.

[9] SSL Labs on-line: ssllabs.com/ssltest, citované v januári 2024.

[10] Openssl on-line: openssl.org, citované v januári 2024.

[11] Go dokumentácia. Dosptupné on-line: go.dev/doc, citované v januári 2024.

[12] Alan A. A. Donovan a Brian Kernighan, The Go Programming Language, Addison-Wesley Professional, 2015.

[13] Alex Edwards, Let's Go, 2020.

[14] Pgx dokumentácia, Dosptupné on-line: github.com/jackc/pgx, citované v januári 2024.

[15] Redis dokumentácia. Dosptupné on-line: redis.io/documentation, citované v januári 2024.

[16] PostgreSQL dokumentácia. Dosptupné on-line: www.postgresql.org/docs, citované v januári 2024.

[17] Ubuntu dokumentácia. Dosptupné on-line: ubuntu.com/server/docs, citované v januári 2024.

[18] Nginx dokumentácia. Dosptupné on-line: nginx.org/en/docs, citované v januári 2024.

[19] Certbot dokumentácia. Dosptupné on-line: eff-certbot.readthedocs.io/en/stable, citované v januári 2024.

[20] Alpine.js dokumentácia. Dosptupné on-line: alpinejs.dev/start-here, citované v januári 2024.

[21] Laravel dokumentácia. Dosptupné on-line: laravel.com/docs/10.x/readme, citované v januári 2024.