Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Využitie certifikátov a CT logov na komunikáciu

Bakalárska práca

2023
Matej Jurčák

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Využitie certifikátov a CT logov na komunikáciu

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	doc. RNDr. Martin Stanek, PhD.

Bratislava, 2023
Matej Jurčák

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Matej Jurčák
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Využitie certifikátov a CT logov na komunikáciu
Using Certificates and CT Logs for communication

Anotácia: Certifikáty a Certificate Transparency (CT) logy sú dôležitou súčasťou bezpečnej komunikácie na Internete. Cieľom bakalárskej práce je preskúmať a navrhnúť využitie týchto mechanizmov na skrytú komunikáciu medzi dvoma alebo viacerými účastníkmi. CT logy môžu byť použité ako forma verejnej nástenky. V práci budú analyzované limity, najmä priepustnosť skrytého komunikačného kanála s využitím politík verejnej certifikačnej autority Let's Encrypt. Súčasťou práce bude aj implementácia konceptu a diskusia o možnostiach detekcie takýchto komunikačných kanálov.

Vedúci: doc. RNDr. Martin Stanek, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Dátum zadania: 18.10.2022

Dátum schválenia: 18.10.2022 doc. RNDr. Dana Pardubská, CSc.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Ďakujem vedúcemu svojej bakalárskej práce, docentovi Martinovi Stanekovi za návrh témy, ktorá ma veľmi zaujala, za konštruktívne pripomienky pri písaní práce a za to, že mi bol počas celého obdobia nápomocný. Taktiež ďakujem RNDr. Jaroslavovi Janáčkovi za poskytnutie doménového mena a servera, vďaka ktorým sme mohli CT kanál implementovať.

Abstrakt

Certifikáty a Certificate Transparency (CT) logy sú dôležitou súčasťou bezpečnej komunikácie na Internete. V našej práci sme navrhli skrytý komunikačný kanál, ktorý zneužíva mechanizmy certifikátov a CT logov na komunikáciu — nazvali sme ho „CT kanál“. Predostreli sme modelový scenár, ktorý by bol vhodný na použitie CT kanála, a detailne sme opísali funkčnosť CT kanála. Odosielateľ bude správu vkladať do vhodne vybratých atribútov certifikátu a prijímateľ správu nájde v jednom z CT logov, ktoré v komunikácii pomocou CT kanála slúžia ako forma verejnej nástenky. Zanalyzovali sme niektoré vlastnosti a a limitácie, ktoré kanál má a s použitím certifikačnej autority Let's Encrypt sme implementovali funkčný koncept tohto kanála.

Kľúčové slová: certifikát, Certificate Transparency (CT), CT logy, skrytý komunikačný kanál

Abstract

Certificates and Certificate Transparency (CT) logs are an important part of secure communication on the Internet. In our work, we have proposed a covert channel that exploits the mechanisms of certificates and CT logs for communication — we called it „CT channel“. We presented a model scenario that would be suitable for using the CT channel, and described the functionality of the CT channel in detail. The sender would embed the message in suitably selected attributes of the certificate, and the receiver would find the message in one of the CT logs, which serve as a form of public bulletin board in CT channel communication. We have analyzed some of the properties and limitations that the channel has and implemented a working concept of the channel using the Let's Encrypt certificate authority.

Keywords: certificate, Certificate Transparency (CT), CT logs, covert channel

Obsah

Zoznam obrázkov

Zoznam ukážok

Úvod

V rámci komunikácie na Internete sa bežne stretávame s bezpečnostnými podmienkami, ktoré na danú komunikáciu kladú jej účastníci. Jednou z najbežnejších podmienok je, aby bol obsah komunikácie známy iba pre osoby, ktoré sa na komunikácii podieľajú. Na zabezpečenie takejto podmienky sa používajú rôzne kryptografické schémy, ktorými sa obsah komunikácie zašifruje a dešifrovať ju majú možnosť len účastníci komunikácie. Internetoví útočníci si však často kladú ešte vyššie požiadavky na ich vzájomnú komunikáciu a to napr. z dôvodu blokovania ich komunikácie pomocou rôznych bezpečnostných monitorovacích systémov a firewallov. V takom prípade útočníkom nestačí, že sa žiadna tretia strana nedostane k obsahu ich komunikácie, navyše potrebujú, aby o ich komunikácii nikto nevedel. Na takýto typ komunikácie sa používajú skryté komunikačné kanály, čo sú také komunikačné kanály, ktoré neboli navrhnuté na žiadny typ komunikácie [1].

Cieľom bakalárskej práce je navrhnúť a preskúmať skrytý komunikačný kanál, ktorý bude „zneužívať“ certifikáty a systém Certificate Transparency (CT) na skrytú komunikáciu medzi dvoma účastníkmi. Nami navrhnutý skrytý komunikačný kanál — pomenovali sme ho „CT kanál“ — bude využívať na odosielanie správ vhodne vybrané atribúty certifikátu a prijímateľ správy bude môcť certifikát so správou nájsť v CT logu.

V prvej kapitole detailnejšie priblížíme základné komponenty — skryté komunikačné kanály, certifikáty a CT logy — v miere postačujúcej na pochopenie navrhovaného skrytého komunikačného kanála. V druhej kapitole opíšeme ako CT kanál funguje, predostrieme modelový scenár komunikácie a vyhodnotíme priepustnosť tohto skrytého kanála s využitím politík verejnej certifikačnej autority Let's Encrypt. V tretej kapitole otestujeme koncept CT kanála, pričom opíšeme jednotlivé komponenty a ich implementáciu. Bližšie rozoberieme problémy, s ktorými sme sa pri implementácii stretli a priložíme ukážku funkčnosti konceptu.

Kapitola 1

Základné komponenty

V tejto kapitole sa pozrieme na základné komponenty našej práce, ktorými sú skryté komunikačné kanály, certifikáty a CT logy. Pokúsime sa ich vysvetliť v miere postačujúcej na pochopenie skrytého komunikačného kanála, ktorý budeme opisovať a analyzovať v nasledujúcej kapitole.

1.1 Skryté komunikačné kanály

Jednou zo základných požiadaviek, ktorú si na vzájomnú komunikáciu kladú jej účastníci, je, aby sa k obsahu správy nemohla dostať žiadna tretia osoba. Na zabezpečenie takejto podmienky sa používajú rôzne kryptografické schémy, ktorými sa obsah komunikácie zašifruje a dešifrovať ju majú možnosť len účastníci komunikácie. V niektorých prípadoch im však nestačí, že komunikácia je šifrovaná, navyše potrebujú, aby o nej nikto nevedel. Už samotná existencia komunikácie alebo zmena v komunikačných vzoroch správania, akou je napríklad zvýšená frekvencia v posielaných správach, prípadne ich veľkosť, môže byť dôvodom na podozrenie a následné blokovanie ich komunikácie. Skryté komunikačné kanály sa teda snažia ukryť samotnú komunikáciu pred okolitým svetom.

Pojem skrytý komunikačný kanál (*covert channel*) nemá striktnú definíciu. Ako prvý s týmto termínom prišiel ešte v roku 1973 Butler Lampson, ktorý označil skrytý komunikačný kanál za taký kanál, „ktorý vôbec nie je určený na prenos informácií“ [1]. Jemne pozmenenú definíciu možno nájsť v tzv. „Oranžovej knihe“, štandarde Ministerstva obrany Spojených štátov amerických, ktorý stanovuje základné požiadavky na hodnotenie účinnosti počítačovej bezpečnosti v počítačových systémoch. V ňom je skrytý komunikačný kanál definovaný ako kanál, „ktorý môže byť zneužitý na prenos informácií spôsobom, ktorý porušuje bezpečnostné zásady systému“ [2]. Narozdiel od Lampsonovej definície neprihliada na to, či bol takýto kanál navrhnutý na komunikáciu alebo nie. V nadväznosti na „Oranžovú knihu“ vyšla „Svetlo ružová kniha“, ktorá nesie

názov „Príručka na pochopenie analýzy skrytých kanálov dôveryhodných systémov“ [3]. Tá sa snaží o formálnejšiu definíciu, pričom berie do úvahy obe vyššie spomenuté definície. Okrem spomínaných definícií existuje ešte mnoho ďalších, aj keď menej známych, ktoré sa líšia podľa toho, v akom kontexte je skrytý komunikačný kanál používaný.

Pomenovanie „skrytý“ vychádza z toho, že daný kanál nemožno detegovať kontrolnými mechanizmami systému, cez ktorý komunikácia prebieha, keďže kanál nepoužíva legitímne mechanizmy prenosu údajov a preto ho bezpečnostné mechanizmy nevedia ľahko odhaliť.

1.1.1 Druhy skrytých kanálov

Väčšina skrytých komunikačných kanálov sa dá zaradiť do dvoch hlavných kategórií, ktorými sú úložné kanály a časové kanály.

- **Úložné kanály:** Formálne sú to kanály, ktoré „umožňujú priamy alebo nepriamy zápis úložného miesta jedným procesom a jeho priame alebo nepriame čítanie iným procesom“ [2]. Utajované údaje sú teda vložené odosielateľom do niektorej časti úložného priestoru, ktorá na to nebola navrhnutá a zároveň ku ktorej má prístup aj prijímateľ správy. Úložné kanály môžu údaje skrývať priamo manipuláciou zdieľaného zdroja, napr. obsahu súborov, diskových blokov alebo fyzickej pamäti, ale taktiež môžu využiť atribút zdieľaného zdroja, akým je napr. názov súboru. Dokonca aj požiadanie o súbor, ktorý neexistuje, umožňuje uložiť dáta do chybového statusu vráteného súborovým systémom [4]. V komunikácii cez počítačovú sieť sa údaje bežne vložia medzi viaceré vrstvy modelu TCP/IP, najčastejším miestom na ukladanie údajov sú však hlavičky paketov. Príkladom skrytého komunikačného kanálu využívajúceho počítačovú sieť je tzv. *Packet length-based covert channel*, ktorý na prenos utajovaných údajov využíva dĺžku sieťových paketov.

- **Časové kanály:** Takéto kanály zahŕňajú všetky nástroje, „ktoré by umožnili jednému procesu signalizovať údaje moduláciou svojho vlastného použitia systémových zdrojov takým spôsobom, že zmena reakčného času pozorovaná druhým procesom by mu poskytla údaje.“ [2]. V časových kanáloch teda nedochádza k žiadnej modifikácií údajov, namiesto toho sú skryté v dĺžke časových intervalov medzi dvoma akciami, akou je napr. poslanie paketu. Mohli by sme napríklad vytvoriť kanál založený na časovaní, ktorý vysiela bity podľa časového intervalu medzi dvoma paketmi — 0 pre krátky interval, 1 pre dlhší. Na základe zdroja použitého sieťového pripojenia možno časové kanály rozdeliť na dva typy:

  - **Pasívne kanály:** Využívajú na prenos údajov už existujúce spojenie. Ich nevýhodou je obmedzená kapacita na prenos údajov, ktorá je daná priepust-

nosťou základného spojenia. Keďže však nevytvárajú žiadne ďalšie spojenia, sú menej náchylné na odhalenie.

– **Aktívne kanály:** Vytvárajú samostatné spojenie na prenos utajovaných údajov. V porovnaní s pasívnymi časovými kanálmi sú schopné dosiahnuť výrazne vyššiu priepustnosť. Keďže však útočník musí vytvoriť vlastné pripojenie, sú aktívne kanály náchylnejšie na odhalenie.

Skrytý komunikačný kanál, ktorý navrhneme v nasledujúcej kapitole, bude úložného typu, keďže našu správu budeme ukladať do rôznych atribútov certifikátu.

1.1.2 Prevencia a detekcia skrytých kanálov

Z bezpečnostného hľadiska predstavujú skryté kanály s malým objemom prenášaných údajov nižšie riziko než tie s veľkým objemom. Samotná existencia skrytých kanálov v systéme je dôsledkom nedokonalého návrhu systému alebo dôsledkom bezpečnostných slabín v použitých technológiách. V ideálnom scenári sa potenciálne všetky skryté kanály úspešne eliminujú ešte pri návrhu systému. U mnohých typov skrytých kanálov však majú techniky na ich elimináciu, prípadne techniky na znižovanie objemu prenášaných údajov pod určitú úroveň, aj vedľajší účinok znižovania efektívnosti systému, keďže sa niektoré automatizované procesy musia nahradiť manuálnymi. Z tohto dôvodu sa musí urobiť kompromis medzi efektivitou systému a veľkosťou objemu prenášaných dát cez skryté kanály.

V „Oranžovej knihe“ poskytujú autori predstavu o tom, aký objem prenášaných dát sa považuje za veľký a ktorý za malý. Objem, ktorý presahuje rýchlosť 100 bitov za sekundu, považujú za veľký, pretože sa podľa nich „nezdá vhodné nazývať počítačový systém bezpečným, ak informácie môžu byť kompromitované rýchlosťou rovnakou ako štandardná výstupná rýchlosť bežne používaného zariadenia.“ Zároveň dodávajú, že v akomkoľvek počítačovom systéme existuje niekoľko relatívne málo objemových skrytých kanálov, ktorých existencia je hlboko zakorenená v návrhu systému. Vzhľadom na veľké potenciálne náklady na znižovanie objemu takýchto skrytých kanálov považujú za prípustné tie, ktoré majú maximálny objem prenášaných dát menší ako jeden bit za sekundu.

Potreba eliminácie skrytých kanálov viedla k štúdiu metód ich detekcie, ale zatiaľ ich bolo navrhnutých pomerne málo. Najvýznamnejšou je matica zdieľaných zdrojov (*Shared Resource Matrix*, SRM), ktorú v roku 1983 predstavil Richard Kemmerer [5]. V „Svetlo ružovej knihe“ [3] sú detailnejšie popísané aj ďalšie metódy na detekciu skrytých kanálov, ktorými sú *Syntactic Information-Flow Analysis* a *Noninterference Analysis*.

1.2 Certifikát

Certifikáty (*Public Key Certificates*), taktiež známe pod pojmom digitálne certifikáty, sú elektronické dokumenty, ktoré sa používajú v kryptografii na overenie platnosti verejného kľúča [6]. Certifikáty existujú vrámci infraštruktúry verejných kľúčov (*Public Key Infrastructure*, PKI), ktorá je súborom úloh, politík a postupov potrebných na vytváranie, správu, distribúciu, používanie, ukladanie a rušenie digitálnych certifikátov. Účelom PKI je uľahčiť bezpečný elektronický prenos informácií pre účastníkov komunikácie na internete. Existuje viacero typov certifikátov, v tejto práci sa však budeme stretávať výhradne s tzv. TLS[1] certifikátmi. Pre jednoduchosť budeme ďalej v práci označovať TLS certifikáty všeobecnejšie „certifikáty".

1.2.1 Na čo slúžia (TLS) certifikáty

Predpokladom bezpečnej komunikácie medzi klientom a serverom je jej dôvernosť, keďže si chcú medzi sebou posielať dosť často citlivé dáta. Z toho dôvodu je nutné zabezpečiť, aby sa k obsahu správy nedostala žiadna cudzia osoba. Na to sa využíva metóda šifrovania správ, počas ktorej si obe strany vymenia svoje verejné kľúče. Potenciálny problém nastáva práve v tejto výmene. Útočník môže pomocou útoku *Man-in-the-Middle* (MitM) pred klientom predstierať, že je server a pred serverom, že je klient. S oboma subjektami nadviaže dôveryhodné spojenie a potom je schopný voľne odchytávať komunikáciu medzi nimi. Tento problém rieši práve protokol TLS, ktorého nezanedbateľnou časťou sú certifikáty.

Keď sa klient a server dohodnú na používaní protokolu TLS, prebehne ešte pred výmenou správ tzv. TLS handshake, ktorý slúži na dohodnutie rôznych parametrov používaných pri vytvorení zabezpečeného spojenia. Vrámci neho musí server poskytnúť klientovi platný certifikát, ktorý potvrdzuje vlastníctvo verejného kľúča pre uvedenú doménu certifikátu. Klientský softvér, ktorým je najčastejšie webový prehliadač, overí, či je certifikát platný a či sa náhodou server nevydáva za niekoho iného a následne môže začať zabezpečená komunikácia. Keď sa pri komunikácii medzi klientom a serverom používa TLS protokol, ktorého súčasťou sú aj certifikáty, je útočníkovi výrazne sťažený spôsob pre MitM útoky. Pre dosiahnutie takéhoto útoku by totiž musel klientovi poskytnúť certifikát pre doménové meno, ku ktorému nemá prístup.

1.2.2 Proces vydania certifikátu

Procesu vydávania certifikátu sa zúčastňujú dve strany — žiadateľ a Certifikačná Autorita (ďalej aj ako CA). Je dôležité si uvedomiť, že aj napriek tomu, že o certifikát

[1] TLS protokol (často zvaný aj SSL, po jeho predchodcovi) je kryptografický protokol určený na zabezpečenie komunikácie v počítačovej sieti, napr. ako súčasť protokolu HTTPS.

žiada žiadateľ, vydaný je pre konkrétnu doménu, nad ktorou má žiadateľ kontrolu.

Certifikačná Autorita je entita, ktorá podpisuje a vydáva certifikáty, pričom v procese overuje identitu žiadateľa. Aby bola úloha certifikačnej autority v TLS ekosystéme zmysluplná, musia jej dôverovať obe strany — tak ako žiadateľ o certifikát, tak aj klient, ktorý sa na certifikát spolieha. Takýmito klientmi sú zväčša webové prehliadače a operačné systémy, pričom každý z nich môže mať iný zoznam certifikačných autorít, ktoré považuje za dôveryhodné[2].

Žiadateľ môže byť ľubovoľný používateľ internetu, ktorý má kontrolu nad doménou, pre ktorú žiada o certifikát.

Samotný proces vydávania certifikátu prebieha nasledovne:

1. Žiadateľ o certifikát si vygeneruje pár verejného a súkromného kľúča.

2. Žiadateľ vytvorí žiadosť o podpísanie certifikátu (CSR), čo je digitálny dokument, ktorý obsahuje jeho verejný kľúč a identifikačné informácie o žiadateľovi (napríklad jeho meno a organizáciu).

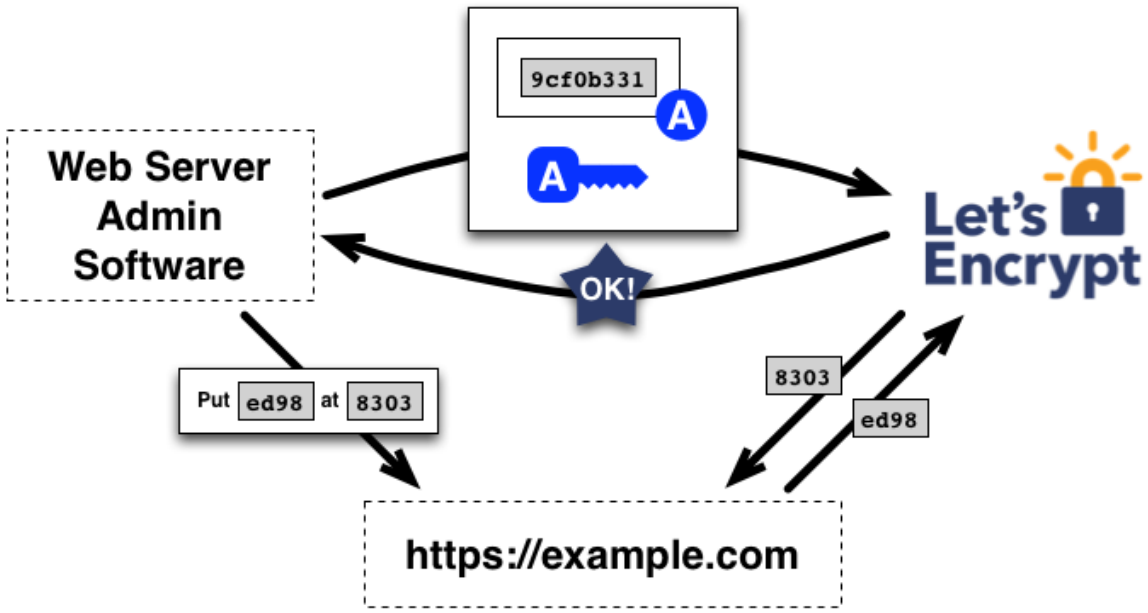
3. Žiadateľ predloží CSR dôveryhodnej CA, ktorá overí totožnosť žiadateľa a vykoná ďalšie kontroly, aby sa uistila, že žiadosť o certifikát je oprávnená.

4. Po úspešnom overení totožnosti vydá CA podpísaný certifikát žiadateľovi.

Overenie totožnosti žiadateľa (3. krok) si vyžaduje skontrolovať, či je žiadateľ vlastníkom danej domény. Tento proces sa nazýva validácia domény (*Domain Validation*). Dokázať vlastníctvo domény pred certifikačnou autoritou sa dá viacerými spôsobmi, my sa pozrieme bližšie na dva z nich — prvým je úprava DNS záznamu pre doménu a druhým je poskytnutie HTTP prostriedku pod známou adresou na doméne. Spôsob úpravy DNS záznamu spočíva v tom, že do DNS záznamu typu TXT žiadateľ vloží kus textu vygenerovaný certifikačnou autoritou. Tá následne skontroluje obsah záznamu. Druhá možnosť spočíva v nahratí HTTP súboru, ktorý mu poskytne certifikačná autorita, na známu adresu na doméne, napr. na `www.example.com/8303`. Certifikačná autorita sa potom pokúsi stiahnuť súbor z webového servera a skontroluje, či má očakávaný obsah. Pri oboch spôsoboch validácie domény navyše certifikačná autorita poskytne aj `nonce`[3], ktoré musí žiadateľ podpísať svojím súkromným kľúčom. Certifikačná autorita tak okrem prístupu k doméne skontroluje aj žiadateľov prístup v súkromnému kľúču.

[2]Trh s globálne dôveryhodnými certifikačnými autoritami je veľmi úzky kvôli značným prekážkam vstupu naň v dôsledku náročných technických požiadaviek na implementáciu.

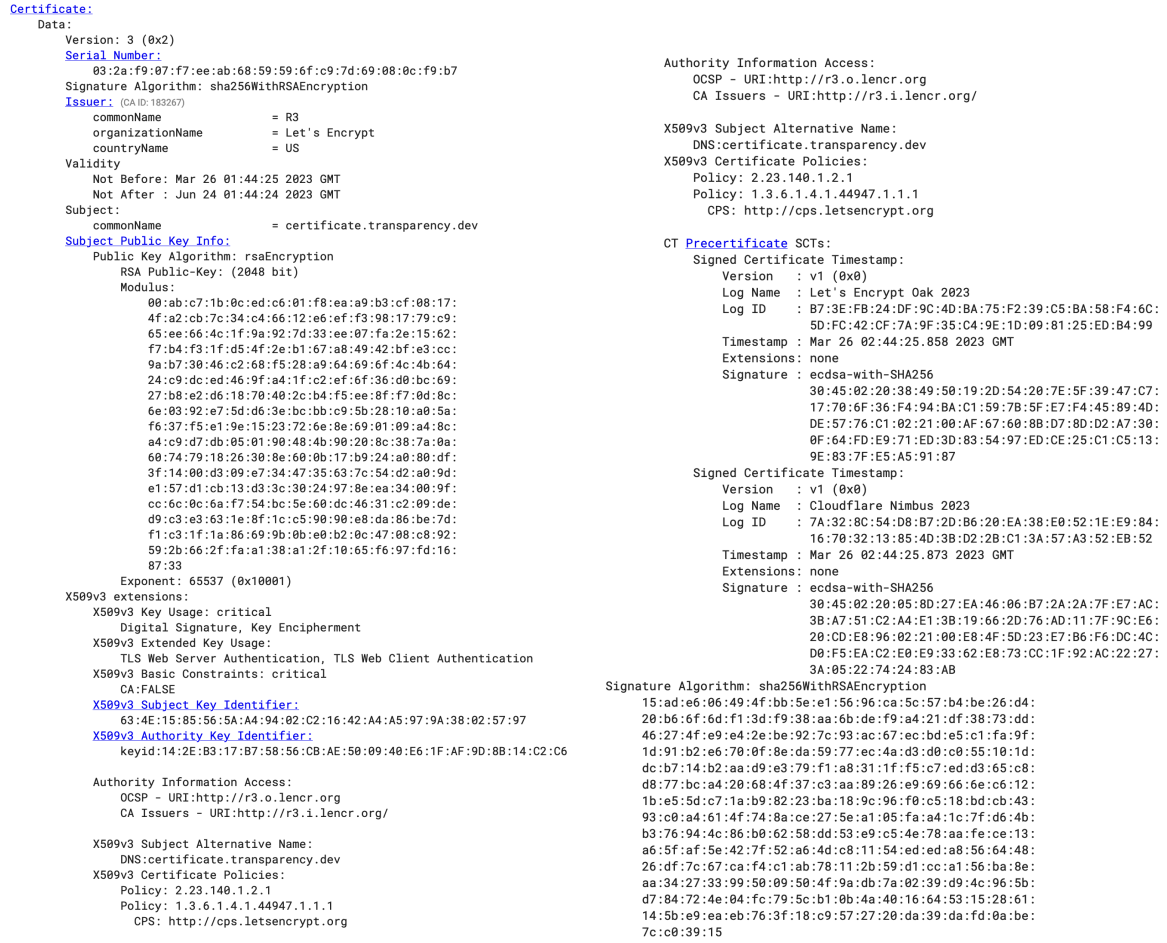
[3]V kryptografii je `nonce` náhodné číslo, ktoré sa vo vzájomnej komunikácii používa jednorázovo, aby sa zabezpečila unikátnosť spojenia a aby sa predchádzalo útokom opakovaním.



Obr. 1.1: Proces validácie domény poskytnutím HTTP súboru. Zdroj: `https://letsencrypt.org/how-it-works/`

1.2.3 Základná štruktúra certifikátu

Certifikát obsahuje striktne definovanú množinu atribútov [6], nad väčšinou ktorej má kontrolu len certifikačná autorita. Atribúty sa delia na základné a tzv. rozšírenia (*Extensions*). Základné atribúty musia byť obsiahnuté v každom certifikáte a patria medzi ne napr. doména subjektu, názov vydávajúcej certifikačnej autority, sériové číslo certifikátu, algoritmus použitý na podpísanie certifikátu, dátum vydania certifikátu, dátum exspirácie certifikátu, verejný kľúč (súkromný kľúč zostáva tajný) a digitálny podpis certifikačnej autority. Dátum exspirácie certifikátu sa môže líšiť v závislosti od CA, napr. certifikačná autorita Let's Encrypt ho má nastavený na 90 dní od vydania certifikátu. Po vypršaní platnosti je nutné požiadať o nový, štandardne je však tento proces automatizovaný. Množina rozšírení sú nepovinné atribúty, ktoré obsahujú dodatočné informácie od žiadateľa. Jedným z častých rozšírení je atribút *Subject Alternative Name* (SAN). Tento atribút je užitočný v prípade, keď chce žiadateľ použiť ten istý certifikát pre viaceré subdomény hlavnej domény. Nemusí žiadať o certifikát pre každú subdoménu zvlášť, stačí, ak pošle jednu žiadosť o certifikát pre hlavnú doménu a všetky ostatné subdomény uvedie v atribúte SAN.



Obr. 1.2: Príklad certifikátu pre doménu `https://certificate.transparency.dev/`.

1.3 CT logy

1.3.1 Certificate Transparency (CT)

Certificate Transparency (CT) je ekosystém, ktorý bol vyvinutý v roku 2013 ako reakcia na rastúce obavy o bezpečnosť certifikátov. Systém slúži na ochranu a monitorovanie nesprávneho vydávania certifikátov, ku ktorému dochádza, keď CA vydá certifikát niekomu, kto na to nemá oprávnenie. Môže sa tak stať v dôsledku omylu alebo úmyselného útoku a môže to mať dôsledky pre bezpečnú komunikáciu na Internete. Pred vývojom CT bola bezpečnosť certifikátov do veľkej miery závislá od certifikačných autorít. Existovalo len málo mechanizmov na odhalenie alebo zabránenie nesprávnemu vydaniu certifikátu. Nesprávne vydaný certifikát sa mohol napríklad použiť na zachytenie šifrovanej komunikácie alebo na vydávanie sa za webovú stránku/server.

Potreba bezpečnejšieho prístupu k vydávaniu a správe certifikátov sa stala zrejmou po sérii závažných narušení bezpečnosti v roku 2011, keď bola napadnutá holandská certifikačná autorita DigiNotar a útočníci vydávali podvodné certifikáty pre stránky ako sú Google, Skype a Facebook. Tento incident vyvolal obavy o bezpečnosť certifikátov a

zraniteľnosť internetu voči podobným útokom. V dôsledku týchto obáv spoločnosť Google v roku 2013 ako riešenie navrhla koncept Certificate Transparency. Cieľom bolo vytvoriť ekosystém, ktorý by zabezpečil väčšiu transparentnosť a zodpovednosť pri vydávaní certifikátov, čo by uľahčilo odhaľovanie a zrušenie nesprávne vydaných certifikátov. Od svojho zavedenia sa CT stala dôležitou zložkou internetovej bezpečnosti a v súčasnosti ju vo veľkej miere používajú hlavní výrobcovia prehliadačov, certifikačné autority a prevádzkovatelia webových stránok. Ekosystém CT pomohol zvýšiť transparentnosť a zodpovednosť pri vydávaní certifikátov, čím sa uľahčilo odhaľovanie a reakcia na nesprávne vydávanie certifikátov.

Systém CT stojí na troch hlavných pilieroch, ktorými sú logy, monitory a audítori. My budeme v našej práci využívať hlavne logy, preto sa ne pozrieme bližšie.

1.3.2 Čo sú CT logy a na čo slúžia

CT logy sú verejne prístupné distribuované databázy, v ktorých sa uchovávajú informácie o certifikátoch a ich súvisiacich metadátach. Ich účelom je poskytnúť mechanizmus na odhaľovanie a prevenciu vydávania podvodných alebo neoprávnených certifikátov. Logy spravujú certifikačné autority, ktoré sú zodpovedné za zabezpečenie integrity a dostupnosti logu. Od certifikačných autorít sa vyžaduje, aby uverejnili záznam o každom certifikáte, ktorý vydali, do jedného alebo viacerých verejných CT logov. Tieto záznamy obsahujú informácie o certifikáte, ktoré sú bližšie opísané v podkapitole 1.3.3. Vďaka zverejneniu týchto záznamov môže ktokoľvek vyhľadávať a analyzovať záznamy s cieľom identifikovať potenciálne podvodné alebo neoprávnené certifikáty.

Implementácia CT logov je postavená na základe dátovej štruktúry Merkleho stromu[4] (*Merkle tree*), ktorá umožňuje verejné zaznamenávanie certifikátov spôsobom, ktorý je kryptograficky bezpečný a odolný voči manipulácii. Každý záznam v logu je kryptograficky podpísaný a pokus o zmenu alebo vymazanie záznamu je možné identifikovať. Tým sa zabezpečí, že log zostane presný a dôveryhodný aj napriek útokom alebo pokusom o manipuláciu.

1.3.3 Informácie v CT logoch

CT logy uchovávajú informácie o certifikátoch vydaných certifikačnými autoritami. Tieto informácie obsahujú názov domény, pre ktorú bol certifikát vydaný, vydavateľa (CA) certifikátu, čas vydania a vypršania platnosti certifikátu, verejný kľúč certifikátu a iné. Okrem toho niektoré logy uchovávajú aj informácie o stave zrušenia certifikátu, čo umožňuje zvýšiť bezpečnosť a transparentnosť používania certifikátov. Log obsahuje aj heš certifikátu známy ako *Merkle Tree Hash*, ktorý kompaktne reprezentuje certifi-

[4] `https://en.wikipedia.org/wiki/Merkle_tree`

kát a jeho pozíciu v strome. Neoprávnená zmena certifikátu v CT logu sa dá odhaliť pomocou rekonštrukcie Merkleho stromu, ktorý sa buduje odspodu a hešujú sa dvojice uzlov, až kým sa nedosiahne koreň stromu. Následne sa overí, či heš v koreni skonštruovaného stromu sedí s hešom, ktorý je v originálom strome. Ak heše nesedia, došlo k neoprávnenej zmene.

1.3.4 Proces vkladania certifikátov do CT logov

Proces vkladania certifikátov do CT logov sa skladá z nasledujúcich krokov:

1. Vlastník domény požiada CA o certifikát.

2. Certifikačná autorita overí, či má vlastník domény právo žiadať o certifikát, a následne vytvorí tzv. predcertifikát (*Precertificate*). Predcertifikát obsahuje rovnaké informácie, ako sú aj v certifikáte, naviac však obsahuje špeciálne rozšírenie, vďaka ktorému ho webové prehliadače budú vedieť rozoznať od certifikátu a neakceptovať ho. Predcertifikáty slúžia na vyviaznutie z deadlocku v systéme CT — predtým, ako CA zaregistruje certifikát, potrebuje do certifikátu vložiť atribút SCT (*Signed Certificate Timestamp*), čo je podpísaná časová pečiatka certifikátu. Aby však certifikát získal SCT, musí byť najskôr nahratý do logu.

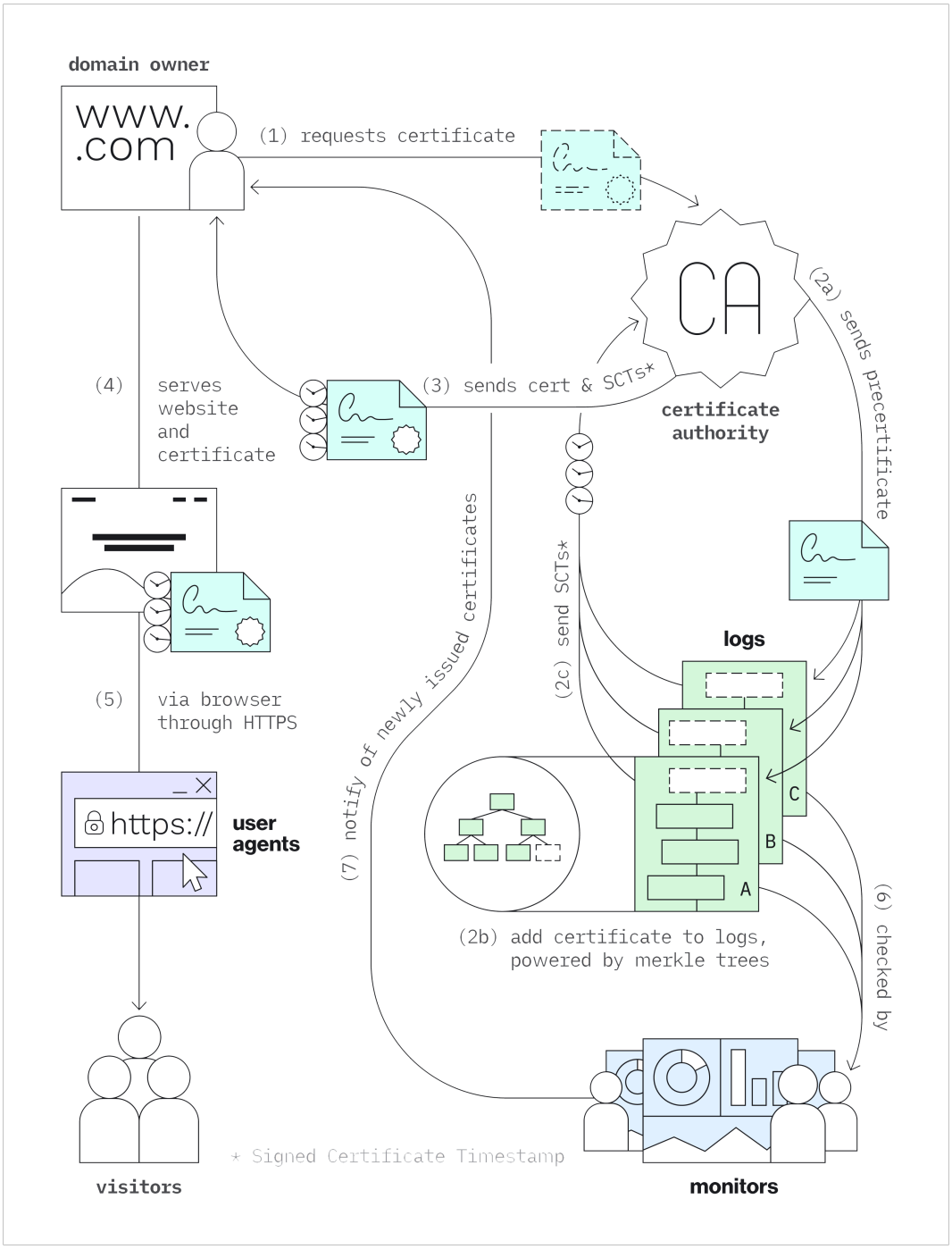
3. Certifikačná autorita pošle požiadavku o vloženie predcertifikátu do logu, ktorý je tam následne vložený.

4. Log vráti podpísanú časovú pečiatku certifikátu certifikačnej autorite. Každý log okamžite vracia SCT certifikačnej autorite so záväzkom nahrať certifikát v rámci maximálneho oneskorenia vloženia (*Maximum Merge Delay*, MMD). Ide o prísľub, že certifikát bude pridaný do logu v tomto časovom období. Zvyčajne je MMD 24 hodín — tento časový interval je určený na to, aby mali prevádzkovatelia logov, v prípade problémov s ich dostupnosťou, dostatok času na odstránenie týchto problémov.

5. Certifikačná autorita odošle plnohodnotný certifikát, ktorého súčasťou je aj podpísaná časová pečiatka certifikátu, vlastníkovi domény.

1.3.5 Pristupovanie k informáciám v CT logoch

Jednou z možností ako získavať informácie o certifikátoch z CT logov je prostredníctvom ich natívneho rozhrania API [7]. Takýto prístup nám umožňuje programátorsky získať informácie o certifikátoch, čo vie byť užitočné na automatizáciu procesu monitorovania nových certifikátov alebo integráciu informácií z CT logov do iných bezpečnostných



Obr. 1.3: Proces vkladania certifikátu do CT logu. Zdroj: `https://certificate.transparency.dev/howctworks/`

nástrojov. Alternatívne môžeme k certifikátom v CT logoch pristupovať aj nepriamo prostredníctvom vyhľadávačov certifikátov od tretích strán. Medzi najpoužívanejšie vyhľadávače certifikátov patria `crt.sh` a služba od spoločnosti Entrust. Ich výhodou je okrem prívetivého používateľského rozhrania aj poskytovanie funkcionalít na vyhľadávanie certifikátov, ktoré natívne API nepodporuje. Viac sa obom týmto prístupom venujeme v časti 2.4.

Kapitola 2

CT kanál

CT kanál je skrytý komunikačný kanál, ktorý na komunikáciu využíva certifikáty a CT logy. V tejto kapitole opíšeme, ako CT kanál funguje, predostrieme modelový scenár komunikácie a zanalyzujeme vlastnosti tohto skrytého kanála s dôrazom na jeho priepustnosť.

2.1 Modelový scenár komunikácie

Majme útočníka A, ktorý chce odosielať príkazy entite B. Typicky môže ísť o *Command And Control* (C&C alebo aj C2) útok, kde sa útočník snaží pomocou malware-u infiltrovať na cudzí počítač, prípadne na celú sieť, a následne s malware-om komunikovať a odosielať mu príkazy. Tomuto sa snažia zabrániť monitorovacie systémy a firewall-y, ktoré vedia vhodne blokovať podozrivú komunikáciu. CT kanálom sa budeme snažiť vyhnúť problému blokovania vzájomnej komunikácie. Hlavná myšlienka, na ktorej stojí komunikácia pomocou CT kanála, je predpoklad, že posielanie requestov na CT logy je jedna z menej podozrivých komunikácii na internete a že blokovacie systémy majú v súčasnosti nízku motiváciu tieto requesty blokovať alebo analyzovať. V našej práci sa budeme zaoberať len prípadom jednosmernej komunikácie, kde A bude správy iba odosielať a B ich bude iba prijímať.

Komunikácia pomocou CT kanálu bude vo všeobecnosti fungovať nasledovne: účastník A vloží správu, ktorú chce poslať účastníkovi B, do vhodne vybraných atribútov žiadosti o certifikát. Certifikát sa následne štandardným spôsobom objaví v CT logu a keďže tie sú verejne dostupné, B si daný certifikát v CT logu nájde (pošle request na CT log) a správu prečíta. Treba podotknúť, že nami opisovaný CT kanál využíva certifikáty a CT logy do miery, v ktorej je to povolené pre každého používateľa a teda dodržuje všetky stanovené limity a štandardy.

2.2 Vkladanie správy do certifikátu

Certifikát má striktne definovanú štruktúru (pozri časť 1.2.3), preto nie je možné správu do certifikátu vložiť triviálne. Musíme sa pozrieť na atribúty certifikátu a nájsť tie, ktoré má pod kontrolou žiadateľ a ktoré by sa dali „zneužiť“ na zakódovanie správy.

2.2.1 Vhodné atribúty certifikátu

Hlavnými atribútmi certifikátu, ktoré má pod kontrolou žiadateľ, sú *Subject Name* (SN), čo je doménové meno, pre ktoré žiada o certifikát a nepovinný atribút *Subject Alternative Name* (SAN) [6], čo je množina doménových mien, pre ktoré žiadateľ chce, aby platil ten istý certifikát. Najčastejšie sa v SAN atribúte nachádzajú práve subdomény hlavnej domény. Predpokladajme, že žiadateľ má prístup k doméne `example.com` a chce odoslať správu `M`. Využiť by mohol práve atribút SAN, kde by vložil subdoménu v tvare `M.example.com`. Ak by sa mu to podarilo, prijímateľ správy by vedel správu ľahko prečítať. Na to, aby mohol žiadateľ vložiť subdoménu do atribútu SAN, musí certifikačnej autorite dokázať, že k nej má prístup — rovnako, ako v prípade atribútu SN. Dokázať prístup k doméne sa dá viacerými spôsobmi (pozri časť 1.2.2), pri spôsobe validácie domény pomocou úpravy DNS záznamov však doména nemusí vrámci Internetu existovať. To žiadateľovi vyhovuje, pretože subdoménu v tvare `M.example.com` chce využiť len na odoslanie správy, nemusí sa tak starať o to, aby bola doména funkčná.

Žiadateľ taktiež generuje pár verejného a súkromného kľúča, ktoré sú viazané na certifikát. Verejný kľúč sa vkladá do atribútu *Subject Public Key*, ale jeho využitie na vloženie správy je problematické, pretože je to matematický objekt obsahujúci parametre, ktoré umožňujú kryptografické operácie a taktiež má špecifický formát a štruktúru. V práci sme sa rozhodli implementovať CT kanál, ktorý bude na posielanie správy využívať výhradne atribút SAN.

2.2.2 Delenie správy medzi SAN-y

Všeobecne sa doménové meno delí na časti (*labels*), ktoré sú oddelené bodkou — napr. `sub.example.com` má tri časti. Štandard pre doménové mená obmedzuje dĺžku doménového mena aj dĺžku jeho časti (bližšie opísané sú tieto limity v časti 2.5.1), preto je nutné navrhnúť spôsoby, akými vieme správu rozdeliť medzi viaceré doménové mená v atribúte SAN. Pozrieme sa na dva z nich.

V prvom ide o minimalizáciu počtu doménových mien. Správu rozdelíme do jednotlivých častí doménových mien, pričom najskôr sa budeme snažiť využiť celý priestor v prvom doménovom mene a potom sa posunieme na ďalšie.

$$
\begin{aligned}
&M_{11}.M_{12}.\cdots.M_{1x}.example.com\\
&M_{21}.M_{22}.\cdots.M_{2x}.example.com\\
&\cdots\\
&M_{y1}.M_{y2}.\cdots.M_{yx_y}.example.com
\end{aligned}
$$

kde $M = M_{11}\cdots M_{1x}M_{21}\cdots M_{2x}\cdots M_{y1}\cdots M_{yx_y}$ je celá správa a M_{ij} je časť správy. V druhom spôsobe sa budeme snažiť využiť maximálny počet doménových mien v atribúte SAN, pričom kódovaná správa sa rovnomerne rozdelí medzi všetky z nich.

Neformálne povedané, prvý spôsob maximalizuje šírku doménových mien, čím minimalizuje hĺbku atribútu SAN a druhý maximalizuje hĺbku atribútu SAN, čím minimalizuje šírku doménových mien. Druhá forma je menej podozrivá, pretože je bežné mať certifikáty s maximálnym počtom doménových mien v atribúte SAN, na rozdiel od veľmi zriedkavých neobvykle dlhých názvov doménových mien.

Aby vedel prijímateľ správne dekódovať takto rozdelenú správu, musí byť v CT logu zachované poradie domén, v akom ich odosielateľ vložil do žiadosti o certifikát. CT logy však nutne nevracajú domény v tom istom poradí a preto je potrebné pri vkladaní správy ku každej doméne ešte pridať informáciu o jej poradí. Jedným zo spôsobov, akým to docieliť, je pridať na začiatok domény krátky prefix, ktorý bude určovať jej poradie. Samotná dĺžka prefixu bude ovplyvnená počtom doménových mien. Prijímateľovi potom stačí usporiadať domény lexikograficky:

$$
\begin{aligned}
&\mathbf{a}M_{11}.\cdots.example.com\\
&\mathbf{b}M_{21}.\cdots.example.com\\
&\mathbf{c}M_{31}.\cdots.example.com\\
&\cdots
\end{aligned}
$$

2.3 Kódovanie správy

Použitia kódovania správy je nutné z dôvodu obmedzenej znakovej sady, ktorú je dovolené používať v doménových menách. V štandarde [8] sa píše, že časti domény, ktoré sú oddelené bodkou „musia začínať písmenom, končiť písmenom alebo číslicou a znaky medzi musia byť len písmená, číslice a pomlčky“. Výber kódovania je dôležitý aj z pohľadu priepustnosti CT kanála, pretože kódovaný text môže mať rôznu dĺžku pre rôzne kódovania. Pozrieme sa teda na niektoré spôsoby kódovania a vhodnosť ich použitia.

2.3.1 Base64/Base64Url

Kódovanie `base64` reprezentuje binárne údaje vo formáte textového reťazca a preto sa na prvý pohľad javí ako vhodné kódovanie. Používa 65-znakovú sadu [9], skladajúcu sa z číslic, malých a veľkých písmen a špeciálnych znakov `+`, `/` a `=`[1]. Prvým problémom, pre ktorý kódovanie nie je možné použiť, sú práve špeciálne znaky, pretože tie sa v doménovom mene nemôžu nachádzať. Z časti tento problém rieši modifikované kódovanie `base64url`, ktoré bolo navrhnuté na kódovanie URL do `base64`. V ňom sú znaky `+` a `/` nahradené znakmi `-`, resp. `_`. Znak `_` patrí do znakovej sady, ktorá je v URL povolená, do doménového mena ho však nie je možné vložiť. Na zakódovanie správy je preto `base64url` taktiež nepoužiteľné. Druhým problémom oboch kódovaní je používanie veľkých aj malých písmen. V štandarde doménových mien [8] je uvedené, že „hoci sú v názvoch domén povolené veľké aj malé písmená, dva doménové mená s rovnakým názvom, ale s rôznymi veľkými písmenami sa považujú za identické“. Aj napriek uvedeniu doménového mena s veľkými písmenami v žiadosti o certifikát sa do CT logov doména nahrá s malými písmenami. Zakódovanú správu by kvôli tejto konverzii nebolo možné pomocou `base64` dekódovať.

2.3.2 Base32

Kódovanie `base32` používa znakovú sadu, ktorá obsahuje písmená A-Z, číslice od 2-7 a znak `=`. Problém s malými a veľkými písmenami je v tomto kódovaní vyriešený, keďže využíva iba veľké písmená. Stále platí, že do CT logov sa nahrajú len malé písmená, ale pri dekódovaní stačí celé doménové meno skonvertovať na veľké písmená, čoho výsledkom bude pôvodný zakódovaný reťazec. Problém so špeciálnymi znakmi sa zredukoval na znak `=`, ktorý sa používa len ako výplň na konci reťazca kvôli doplneniu dĺžky na najbližší násobok 8. Tomuto znaku sa dá vyhnúť tak, že po zakódovaní správy sa z konca reťazca odstránia znaky `=` a takto modifikovaný reťazec sa vloží do žiadosti o certifikát. Pred procesom dekódovania správy prijímateľ doplní rovnaké množstvo znakov `=` na koniec reťazca. Oproti kódovaniu `base64` zaberá kódovanie `base32` väčší priestor kvôli menšiemu množstvu používaných znakov. Pre pôvodnú správu S, ktorá má $|S|$ bajtov, bude mať text $S_{b_{32}}$ v `base32` kódovaní výslednú veľkosť v bajtoch:

$$|S_{b_{32}}| = \left\lceil \frac{|S|}{5} \right\rceil \cdot 8$$

V priemere obsahuje kódovanie cez `base32` o 60% viac bajtov oproti pôvodnej správe. V implementácii CT kanála, ktorú opisujeme bližšie v nasledujúcej kapitole, sme použili práve toto kódovanie. Poznamenajme, že pri skutočnom použití CT kanála by

[1] Znak `=` slúži len ako výplň na konci kódovaného textu.

sa správa ešte pred samotným kódovaním zašifrovala, ale tomuto aspektu komunikácie sa v práci nevenujeme.

2.4 Hľadanie certifikátu v CT logu

V našom scenári má prijímateľ správ vopred informáciu o doménovom mene, pre ktoré žiada odosielateľ správy o certifikát. Vďaka tomu vie certifikát vyhľadať v CT logu. Každý CT log musí poskytovať API endpoint-y definované v štandarde [7]. Naša predstava v počiatočnej fáze výskumu bola, že existuje natívna podpora pre vyhľadávanie certifikátov v CT logu podľa doménového mena. Vďaka tomu by si mohol prijímateľ správy pomerne triviálne z CT logu vypýtať posledný certifikát pre dohodnuté doménové meno. Po dôkladnom preštudovaní dokumentácie sa však ukázalo, že túto funkciu CT logy natívne nepodporujú, keďže ani neboli navrhnuté na nami zamýšľaný typ komunikácie. Pozrieme sa preto bližšie na možnosti, akými sa vie odosielať dostať k certifikátu.

2.4.1 Certificate Search (crt.sh)

Prvou z možností je použitie služieb tretích strán, ktoré slúžia na vyhľadávanie certifikátov vydávaných dôveryhodnými certifikačnými autoritami. Tieto služby zbierajú v pravidelných intervaloch dáta o novo nahratých certifikátoch v sledovaných CT logoch. Ich hlavnou výhodou je okrem jednoduchého používateľského prostredia poskytovanie viacerých funkcií, ktoré natívne logy nepodporujú. Jednou z nich je práve nami želané vyhľadávanie pomocou doménového mena.

Existuje viacero takýchto služieb, avšak zďaleka najznámejšou a najpoužívanejšou je služba `crt.sh`[2] od spoločnosti Sectigo. Okrem webového rozhrania poskytuje API a dokonca aj prístup na PostgreSQL databázu zaindexovaných certifikátov. Hlavnou nevýhodou služby `crt.sh` je jej nespoľahlivosť. Služba je populárna a intenzívne využívaná, preto je často nedostupná a nezriedka býva problém aj s neskorším nahratím certifikátov z niektorých CT logov. Môže teda nastať situácia, kedy CA vydá certifikát, ten sa v momente objaví v príslušnom CT logu, ale v službe `crt.sh` sa certifikát zjaví o pár hodín, niekedy dokonca až o pár dní.

Prijímateľ správy by teda mohol robiť v pravidelných intervaloch HTTPS requesty[3] na službu `crt.sh` a kontrolovať, či v nej nepribudol nový certifikát pre hľadané doménové meno. Využitie služby, akou je `crt.sh`, považujeme aj napriek jej zjavným

[2]Všetky CT logy, ktoré služba `crt.sh` aktuálne monitoruje, je možná nájsť na: `https://crt.sh/monitored-logs`.

[3]Priame pripojenia na databázu by pravdepodobne vzbudilo pozornosť a neprechádzalo by cez obvyklým spôsobom konfigurované proxy.

výhodám za nevhodný spôsob získania správy z spomedzi opisovaných možností a to z dvoch dôvodov. Prvým z nich je nutnosť robenia requestov mimo natívnych logov, čo bol dôležitý predpoklad nepodozrivej komunikácie pri návrhu CT kanála. Druhým dôvodom je nestabilnosť a nespoľahlivosť týchto služieb.

2.4.2 Natívne vyhľadávanie v CT logoch

Ako sme už spomenuli, CT logy nemajú v štandarde definovaný API endpoint na vyhľadávanie pomocou doménového mena. Jediné definované volanie na získanie záznamov (certifikátov) z logu je `/get-entries` [7], ktoré vráti certifikáty zo špecifikovaného rozsahu indexov. Prijímateľ správy však nemá ako vedieť, ktorý index v logu prislúcha certifikátu od odosielateľa správy. Bez dodatočnej informácie by musel neustále prehľadávať CT log, čo je neprípustné kvôli obrovskému objemu prenášaných dát. Nezanedbateľné by bolo aj množstvo requestov, ktoré by musel prijímateľ správy poslať, keďže väčšina logov má definovaný limit na počet záznamov, ktoré vráti v jednom requeste. Štandard hovorí, že „logy môžu obmedziť počet záznamov, ktoré možno získať na jeden request.“ Väčšina logov má tento limit nastavený v rozsahu 32 až 256 záznamov na jeden request. Tempom, ktorým sa nahrávajú nové certifikáty do logov[4], by tak prijímateľ mohol mať problém stíhať prehľadávať log.

Nepodporovanie vyhľadávania pomocou doménového mena v natívnych logoch nás viedlo k pridaniu dodatočnej informácie, ktorú bude mať prijímateľ správy vopred k dispozícii a tou je čas odoslania prvej správy (čas požiadania o certifikát). Komunikácia vie potom prebiehať tak, že odosielateľ v i. správe zahrnie informáciu o čase poslania $i + 1$. správy. Pozrieme sa na možné spôsoby, ako vie prijímateľ s touto dodatočnou informáciou nájsť certifikát v logu.

2.4.3 Binárne vyhľadávanie

Prvým nápadom bolo použitie binárneho vyhľadávania. Prijímateľ by sa pozeral na vybrané certifikáty z logu a porovnával by časy ich vydania s časom, kedy mala byť poslaná správa. Na nájdenie by mu stačilo $\log_2 T$ porovnaní, kde T je množstvo certifikátov v logu. Logy obsahujú rádovo stovky miliónov certifikátov a tak by mu nájdenie trvalo v priemere pár desiatok requestov. Tento počet by sa dal znížiť tým, že by si prijímateľ pamätal index certifikátu s predošlou správou a na skoršie vydané certifikáty by sa ani nepozeral.

Pokus o implementáciu takéhoto riešenia však ukázal, že použitie binárneho vyhľadávania nemôže byť úspešné. Problémom je rozdiel medzi časom, kedy sa žiadalo o certifikát (*Not before*), a časom, kedy bol certifikát nahraný do logu. Prijímateľ má

[4]Bližšie sa na to pozrieme v implementačnej časti.

informáciu len o čase, v ktorom odosielateľ požiadal o certifikát, ale ten sa môže značne líšiť od času, kedy sa certifikát objaví v logu[5]. V dôsledku toho nie sú záznamy v logu usporiadané podľa času vydania certifikátu (pozri Tabuľku 2.1 nižšie), čo znemožňuje certifikát binárne vyhľadať.

Tabuľka 2.1: Úsek certifikátov z CT logu `argon2023`

Log ID	Not before	Domain names
1058068821	2023-05-07 09:04:53	www.expressiveverticals.com
1058068822	2023-05-07 09:05:01	www.swpowersystems.net,swpowersystems.net
1058068823	2023-05-07 09:04:56	www.milk.furniture
1058068824	2023-05-07 09:04:56	*.crumpling-crier.click,crumpling-crier.click
1058068825	2023-05-07 09:05:00	vikingflowerpatch.com
1058068826	2023-05-07 09:04:52	stockholmykt.ru,www.stockholmykt.ru
1058068827	2023-05-07 09:04:53	*.partner-massage.net,partner-massage.net

2.4.4 Zúženie intervalu

Prijímateľ môže využiť informáciu o čase odoslania správy na zúženie intervalu certifikátov, ktoré si potrebuje vypýtať z CT logu. Tesne predtým ako odosielateľ požiada o certifikát v dohodnutom čase, si prijímateľ pozrie aktuálnu veľkosť logu cez volanie `/get-sth`. Následne začne sekvenčne prehľadávať log od zapamätaného indexu až kým nenarazí na certifikát so správou. Aj v takomto prístupe môže nastať problém s oneskorením medzi časom požiadania o certifikát a jeho nahratím do CT logu, pričom tento problém môže byť rozdielne veľký naprieč rôznymi CT logmi. Pri implementácii funkčného CT kanála, ktorú opisujeme bližšie v nasledujúcej kapitole, sme však narazili buď na žiadne alebo zanedbateľne malé oneskorenie.

2.4.5 Manuálne nahratie certifikátu do CT logu

Certifikačná autorita, ktorá s CT logmi spolupracuje, má svoj zoznam logov, do ktorých automaticky nahráva certifikáty po ich vydaní. Komunikácia pomocou CT kanála je tak v prípade spoliehania sa na automatické nahratie do logu obmedzená na konkrétny typ logu, ktorý používa CA. Existuje však možnosť, ako si nahrať certifikát do ľubovoľného logu manuálne, a to pomocou volania `/add-chain`.

Toto volanie očakáva usporiadanú reťaz certifikátov (*Certificate chain/path*). Reťaz sa musí začínať certifikátom, ktorý chceme nahrať do CT logu a končiť koreňovým certifikátom certifikačnej autority (*Root certificate*). Koreňový sa nazýva z toho dôvodu,

[5]Log sa zaväzuje nahrať certifikát do istého času, zväčša do 24 hodín od požiadania o certifikát, pozri časť 1.3.4.

že si ho podpísala sama certifikačná autorita. Spôsobom, ktorému sa nebudeme venovať detailnejšie, CT log overí túto reťaz a následne vloží certifikát do logu.

Vďaka manuálnemu nahrávaniu tak môže komunikácia prebiehať v ľubovoľnom CT logu, nezávisle od toho, kde ho automaticky nahrá certifikačná autorita. Odosielateľ si teda môže požiadať o certifikát od CA, ktorá ho nahrá do jej príslušného logu, a potom si ho nahrať do, pre komunikáciu cez CT kanál, „výhodnejšieho“ CT logu. Výhodnejší log je ten, ktorý vracia väčší interval záznamov alebo ktorému v priemere pribúda menšie množstvo certifikátov v danom časovom rozsahu.

Podotýkame, že volanie `/add-chain` nie je masovo používanou funkciou, pretože vkladanie certifikátov do CT logov je v prvom rade zodpovednosťou certifikačných autorít. V štandarde sa nespomína, či je CT log nutný nahrať akúkoľvek korektnú reťaz certifikátov, avšak my sme sa pri testovaní tohto volania nestretli so žiadnym CT logom, ktorý by nahratie certifikátu nedovoľoval.

2.5 Vlastnosti CT kanála

Dôležitou vlastnosťou CT kanála je počet requestov, ktoré musí prijímateľ spraviť na CT log, aby sa dostal k správe. Čím je tento tento počet vyšší, tým existuje väčšia šanca, že si túto komunikáciu všimnú monitorovacie systémy a vyhodnotia ju ako podozrivú. Tento počet však výrazne ovplyvňuje implementácia CT kanála, pričom výraznými faktormi v znižovaní počtu requestov je výber CT logu, cez ktorý bude komunikácia prebiehať a spôsob vyhľadávania v CT logu. V časti 3.2 analyzujeme počet vykonaných requestov pre konkrétnu implementáciu.

Pri opisovaní skrytého komunikačného kanála je vhodné zanalyzovať jeho priepustnosť. Certifikát má striktne definovanú štruktúru s danými limitmi, čo prirodzene ovplyvňuje veľkosť správ, ktoré vieme cez CT kanál posielať. Okrem toho si každá CA nastavuje limity na množstvo certifikátov, o ktoré vie žiadateľ za istý časový úsek požiadať. Pozrieme sa preto na jednotlivé parametre, ktoré ovplyvňujú priepustnosť CT kanála a spravíme konkrétnu analýzu pre certifikačnú autoritu Let's Encrypt.

2.5.1 Parametre ovplyvňujúce priepustnosť

Na zanalyzovanie priepustnosti skrytého kanála potrebujeme vedieť dve informácie — akú dlhú správu vieme vložiť do certifikátu a ako často vieme požiadať o vydanie certifikátu.

Dĺžka správy. Správu budeme vkladať len do atribútu SAN. Pri ňom nás zaujímajú dve hodnoty — koľko sa do neho zmestí doménových mien a aké dlhé môžu byť jednotlivé doménové mená. Štandard, ktorý popisuje atribúty certifikátu a aj ich limity [6] nehovorí nič o maximálnom počte doménových mien v atribúte SAN. Tento

limit si nastavujú certifikačné autority. Maximálnu dĺžku domény popisuje štandard [8], kde sa spomína, že doména môže mať najviac 255 bajtov. Vrámci DNS sa na označenie začiatku prenosu a dĺžky doménového mena používa dodatočný prvý bajt a na označenie konca prenosu sa používa dodatočný bajt na konci, takže praktický limit dĺžky domény je **253** bajtov/znakov, ktorá sa potom prenáša ako 255 bajtový paket. Okrem toho existuje limit aj na jednotlivé časti domény oddelené bodkou, ktoré môžu mať dĺžku maximálne **63** znakov. Niektoré CA majú nastavený limit aj na počet častí domény.

Množstvo certifikátov. Každá CA má svoje limity na počet vydaných certifikátov pre konkrétnu doménu za určený časový úsek. Rozlišuje sa, či sa žiada o nový certifikát alebo ide len o žiadosť o predĺženie platnosti už vydaného certifikátu.

2.5.2 Analýza pre Let's Encrypt

Analýzu urobíme pre certifikačnú autoritu Let's Encrypt (LE), pretože ju využívame pri implementácii CT kanála. Do jedného certifikátu vieme cez LE vložiť maximálne **100** doménových mien. Čo sa týka množstva certifikátov, je možné požiadať o najviac **50** certifikátov za týždeň pre registrovanú doménu (*Registered Domain*). Registrovaná doména je vo všeobecnosti časť domény, ktorá bola zakúpená od registrátora doménových mien. Napríklad v názve `www.example.com` je registrovaná doména časť `example.com`. V komunikácii pomocou CT kanála bude odosielateľ žiadať o certifikát vždy pre tú istú, vopred dohodnutú, registrovanú doménu. Žiadosti sa budú líšiť len v atribúte SAN, čo sa však bude rátať ako nový certifikát, keďže je to iná množina doménových mien. Pri implementácii CT kanála sme zistili, že Let's Encrypt má nastavený limit aj na počet častí doménového mena oddelených bodkou, konkrétne ich nemôže byť viac ako **10**.

Keď už poznáme limity, vieme vypočítať priepustnosť CT kanála s použitím certifikačnej autority Let's Encrypt:

$$\begin{aligned} \mathsf{CRT} &= 50 \text{ certifikátov/týždeň} \\ \mathsf{SAN} &= 100 \text{ domén/certifikát} \\ \mathsf{DN} &= 253 \text{ bajtov/doména} \end{aligned}$$

$$\begin{aligned} \mathsf{P} &= \mathsf{CRT} \cdot \mathsf{SAN} \cdot \mathsf{DN} \\ \mathsf{P} &= 50 \cdot 100 \cdot 253 \\ \mathsf{P} &= \mathbf{1.265}\ \frac{\mathsf{MB}}{\text{týždeň}} \end{aligned}$$

kde P je priepustnosť. Poznamenajme, že toto je iba teoretický limit, keďže počet bajtov v doménovom mene využitých na správu bude nižší ako 253, v závislosti od

dĺžky registrovanej domény, keďže tá sa nepoužíva na kódovanie správy a je vždy rovnaká, a počtu častí domény. Čím viac je častí, tým viac miesta zaberajú bodky, ktoré tieto časti oddeľujú a tým menej miesta zostane na kódovanie správy.

Kapitola 3

Overenie konceptu

V tejto kapitole otestujeme koncept CT kanála, pričom opíšeme jednotlivé komponenty a ich implementáciu. Bližšie rozoberieme problémy, s ktorými sme sa pri implementácii stretli a priložíme ukážku funkčnosti konceptu.

3.1 Komponenty CT kanála

Pri overovaní konceptu CT kanála sme implementovali tri komponenty, z toho dva slúžia odosielateľovi a tretí slúži prijímateľovi. Každý z komponentov je spustiteľný skript napísaný v jazyku `Python`, ktorý sme si vybrali kvôli existencii vhodných knižníc pre prácu s certifikátmi.

- `send_message.py` (odosielateľ) — Skript berie ako argument správu, ktorú zakóduje pomocou vhodného kódovania. Takto zakódovanú správu vloží do doménových mien, pričom dbá na dodržanie všetkých limitov a štandardov, ktoré sa ich týkajú. Nakoniec pošle certifikačnej autorite Let's Encrypt žiadosť o vydanie certifikátu s vytvorenými doménovými menami.
- `add_chain.py` (odosielateľ) — Skript berie ako argument reťaz certifikátov, ktorá začína certifikátom, ktorý chce odosielateľ vložiť do CT logu. Túto reťaz upraví na požadovaný tvar a pošle ju do vybraného CT logu, vďaka čomu sa v ňom certifikát zjaví.
- `receive_message.py` (prijímateľ) — Skript si najskôr poznačí posledný index v CT logu a začne od zapamätaného indexu sekvenčne prehľadávať vybraný CT log až kým nenájde hľadaný certifikát.

3.1.1 Implementácia komponentu `send_message.py`

Implementáciu komponentu výrazne ovplyvnilo rozhodnutie o výbere certifikačnej autority. My sme sa pre vydávanie certifikátov rozhodli použiť certifikačnú autoritu **Let's**

Encrypt, pretože je bezplatná a dostupná pre každého. V procese žiadania o certifikát je nutné preukázať kontrolu nad doménovými menami, z čoho vyplýva, že **skript je možné spúšťať len na serveri**, ktorý má prístup k DNS záznamom pre doménové mená zo žiadosti. Nám bol na účely otestovania CT kanála poskytnutý vlastný nameserver a doménové meno `ctlr.seclab.dcs.fmph.uniba.sk`.

Kódovanie správy

Na zakódovanie správy používame kódovanie `base32`, pričom z konca zakódovaného reťazca dodatočne odoberáme znaky `=`, ktoré sa v doménovom mene nemôžu vyskytovať. Skript potom skontroluje, či dĺžka zakódovaného reťazca neprekračuje maximálnu možnú dĺžku, ktorú vieme do doménových mien vložiť. V prípade, že túto dĺžku prekračuje, o tom dá skript odosielateľovi vedieť a skončí s chybou.

Odosielateľovi sa nedá vopred poskytnúť informácia o maximálnej dĺžke správy v znakoch, ktorú je schopný odoslať, pretože v kódovaní UTF-8 môžu dve správy s rovnakým počtom znakov zaberať rozdielny počet bajtov z dôvodu premenlivého kódovania znakov. Musíme teda správu najskôr zakódovať do `base32` a až potom overiť, či sa takto zakódovaná správa zmestí do atribútu SAN.

```
   Pôvodná správa: ahoj
            Base32: MFUG62Q=
Zakódovaná správa: MFUG62Q
```

Ukážka 3.1: Kódovanie správy

Tvorba doménových mien

V časti 2.2.2 sme navrhli dva spôsoby, akými vieme vložiť zakódovaný text do doménových mien. Pri implementácii sme použili spôsob, ktorý maximalizuje dĺžky doménových mien. Každá časť doménového mena môže mať maximálne 63 znakov, avšak začínať sa môžu len písmenom. Pri delení správy na jednotlivé časti domény by sa však mohlo stať, že by sa niektorá časť začínala číslom. Aby sme sa tomuto problému vyhli, natvrdo pridáme na začiatok každej časti (s výnimkou prvej) znak `x`, čím síce jemne znížime priepustnosť kanála, ale zbavíme sa tým zbytočných implementačných detailov. Posledným krokom je pridanie prefixu, ktorý bude určovať poradie správy, na začiatok každého doménového mena. Certifikačná autorita Let's Encrypt podporuje až 100 doménových mien v atribúte SAN a keďže časť doménového mena sa môže začínať len písmenom (ktorých je spolu 52), musí mať prefix aspoň dva znaky — v našom prípade je prvým znakom písmeno a druhým číslo.

```
--- Base32 ---
JBQXMZJAPFXXKIDSMVQWY3DZEB2HE2LFMQQHI3ZAMRSWG33EMUQGS5BA6CP2JFB7EBESA2DPOBSSA6LPOUQ
GC4TFEBXG65BAMRUXGYLQOBXWS3TUMVSC4ICBOBQXE5BAMZZG63JAORUGC5BMEBESA53POVWGIIDMNFVWKI
DUN4QGEZLMNFSXMZJAORUGC5BAENGEMQZAO5UWY3BANVQWWZJANF2CA5DPEBKE6UBUFQQGE5LUEBESAZ3VM
VZXGIDJOQQGS4ZAORUGKIDIN5YGKIDUNBQXIIDLNFWGY4ZAPFXXKLQ
--- SAN ---
a0JBQXMZJAPFXXKIDSMVQWY3DZEB2HE2LFMQQHI3ZAMRSWG33EMUQGS5BA6CP2J.xFB7EBESA2DPOBSSA6L
POUQGC4TFEBXG65BAMRUXGYLQOBXWS3TUMVSC4ICBOBQ.xXE5BAMZZG63JAORUGC5BMEBESA53POVWGIIDM
NFVWKIDUN4QGEZLMNFSXMZ.ctlr.seclab.dcs.fmph.uniba.sk

a1JAORUGC5BAENGEMQZAO5UWY3BANVQWWZJANF2CA5DPEBKE6UBUFQQGE5LUEBE.xSAZ3VMVZXGIDJOQQGS
4ZAORUGKIDIN5YGKIDUNBQXIIDLNFWGY4ZAPFXXKLQ.ctlr.seclab.dcs.fmph.uniba.sk
```

Ukážka 3.2: Vkladanie zakódovaného textu do atribútu SAN

Posielanie žiadosti o certifikát

Keď už máme prichystané doménové mená, môžeme poslať žiadosť o vydanie certifikátu certifikačnej autorite Let's Encrypt. Tá využíva protokol **ACME** [10], ktorý slúži na automatizáciu procesu vydávania certifikátov a overovania kontroly nad doménovými menami, ktoré sú uvedené v žiadosti. Existuje mnoho softvérov[1] v rôznych programovacích jazykoch, ktoré používajú protokol ACME, odporúčaný je však softvér `certbot`[2], na ktorý je potrebné mať prístup k terminálu.

Klient `certbot` podporuje oba typy validácie domén, ako `http`, tak aj `dns` (pozri časť 1.2.2). My sme sa rozhodli použiť spôsob úpravy DNS záznamov a to z toho dôvodu, že sa tým vyhneme konfigurovaniu funkčného servera, čo je pri spôsobe `http` nutné. Po poslaní žiadosti o certifikát certifikačnej autorite, vygeneruje `certbot` pre každé doménové meno v žiadosti unikátny reťazec, ktorý treba vložiť ako DNS záznam typu TXT do zónového súboru.

```
_acme-challenge.example.com. 300 IN TXT "gfj9Xq...Rg85nM"
```

Ukážka 3.3: DNS záznam typu TXT na overenie prístupu k doméne

Štandardne tento proces prebieha interaktívne v termináli, pričom zónový súbor musí žiadateľ upravovať manuálne. Tomu sa chceme v komunikácii pomocou CT kanála vyhnúť, pretože je to zložitejšie na používanie a trvá to dlho (čím dlhšie vydanie certifikátu trvá, tým väčší úsek certifikátov musí prijímateľ prejsť v CT logu). Pre účel automatizácie úpravy DNS záznamov poskytuje `certbot` prepínač `--manual-auth-hook` očakávajúci skript, ktorý nahrá TXT záznam do zónového súboru pre každé doménové meno v žiadosti. My sme ho implementovali v jazyku `Bash`.

[1] `https://letsencrypt.org/docs/client-options/`
[2] `https://certbot.eff.org/`

```
#!/bin/bash

echo "_acme-challenge.${CERTBOT_DOMAIN}. 300 IN TXT \"${CERTBOT_VALIDATION}\"" >> /
    etc/bind/zones/example
rndc reload
```

Ukážka 3.4: Obsah súboru `cert_dns_hook.sh`

Okrem procesu validácie domény môže nastať interakcia v termináli aj v prípade, že už existuje platný certifikát pre niektoré doménové meno v žiadosti. `Certbot` sa na to snaží upozorniť a dáva možnosť sa rozhodnúť medzi ponechaním starého certifikátu alebo pridaním ostatných doménových mien do nového certifikátu. Akákoľvek interakcia je pre naše účely nežiadúca, preto sú vo výslednom volaní `certbota` aj prepínače[3] `--reinstall`, `--break-my-certs`, `--renew-with-new-domains`, `--expand` a `--force-renewal`, ktoré sa interakcie zbavujú a zabezpečujú vydanie nového certifikátu bez ohľadu na to, aké boli doménové mená v predošlom vydanom certifikáte. Výsledné volanie `certbota` teda vyzerá nasledovne:

```
certbot certonly --manual --reinstall --break-my-certs --renew-with-new-domains
--expand --force-renewal --manual-auth-hook cert_dns_hook.sh
--preferred-challenges dns -d "example.com"
```

Ukážka 3.5: Žiadosť o certifikát bez interakcie

Skript `send_message.py` po vytvorení doménových mien so zakódovanou správou zavolá volanie z Ukážky 3.5. pomocou metódy `subprocess.run()`, ktorá sa používa na spustenie príkazu v systémovom príkazovom riadku. Na implementovanie posielania žiadosti sme mohli využiť aj `ACME` klienta v jazyku `Python`, ale keďže sme už mali funkčný a otestovaný `certbot` program, rozhodli sme sa nemeniť ho.

3.1.2 Implementácia komponentu `add_chain.py`

V prípade, že účastníkom komunikácie nevyhovuje CT log, do ktorého certifikáty automaticky nahráva certifikačná autorita, môžu sa rozhodnúť využiť možnosť manuálneho nahratia certifikátu do iného CT logu. Na to slúži komponent `add_chain.py`, ktorý môže odosielateľ spustiť potom, ako pomocou `send_message.py` vloží správu do certifikátu. Skript očakáva reťaz certifikátov, na ktorej začiatku je certifikát, ktorý chce odosielateľ vložiť do CT logu a na konci je koreňový certifikát certifikačnej autority. Výhodou použitia `certbot` klienta pri žiadaní o certifikát je, že automaticky túto reťaz vygeneruje. Nie je tak potrebné implementovať časť s hľadaním tejto reťaze. Štandardne je táto reťaz generovaná vo formáte `.pem`, pričom CT logy pri manuálnom nahrávaní certifikátov očakávajú upravenú reťaz certifikátov vo formáte `JSON`. Skript

[3] `https://eff-certbot.readthedocs.io/en/stable/using.html#certbot-command-line-options`

každý certifikát v reťazi upraví do vhodného tvaru a pošle POST request na `/add-chain` do vybraného CT logu, čím sa do neho nahrá certifikát.

```
-----BEGIN CERTIFICATE-----
MIIErjCCA5agAwIBAgISBE7i/8I
HlMu6McX9FrGTzyusMA0GCSqGSI
         ...
3f+Lod+HJTlsJ6U0eQNZiL09q0v
6Ht+S2mbLD/LA=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIFFjCCAv6gAwIBAgIRAJErCEr
         ...
```

Ukážka 3.6: Reťaz v tvare PEM

```
{
  "chain": [
    "MIIErjCC...mbLD/LA=",
    "MIIFFjCC...q7hHwg==",
    "MIIFYDCC...dy753ec5"
  ]
}
```

Ukážka 3.7: Reťaz v tvare JSON

Alternatívna implementácia by mohla využívať službu `crt.sh`. Vhodná by bola napr. v prípade, kedy by chcel odosielateľ využiť takého ACME klienta, ktorý reťaz automaticky negeneruje. Táto služba poskytuje volanie `/gen-add-chain` na nájdenie hľadanej reťaze. Stačí poslať POST request s parametrom `b64cert` obsahujúcim certifikát vo formáte `.pem` a služba nájde a vráti hľadanú reťaz vo formáte, ktorý je možné priamo nahrať do nami zvoleného CT logu.

3.1.3 Implementácia komponentu `receive_message.py`

Tento komponent slúži prijímateľovi na nájdenie certifikátu so správou v CT logu a na dekódovanie tejto správy. Pri jeho implementácii sme predpokladali, že sa bude spúšťať tesne pred dohodnutým časom, v ktorom sa má správa odoslať. Po spustení skript zistí aktuálnu veľkosť logu pomocou volania `/get-sth`, aby vedel, odkiaľ ho má začať sekvenčne prehľadávať. Pred začatím vyhľadávania pár sekúnd počká pre prípad, že by sa vydávanie certifikátu predĺžilo a certifikát sa ešte nestihol zjaviť v CT logu. Prehľadávanie potom vyzerá nasledovne:

- Skript si vypýta úsek certifikátov `<index, index + max_block_size>` pomocou volania `/get-entries`, kde `index` je na začiatku zapamätaný index, ktorý sa každým volaním zvyšuje a hodnota `max_block_size` obsahuje maximálny počet záznamov, ktoré CT log vráti pre jeden request.

- Záznamy z úseku musí dekódovať, keďže CT logy ich vracajú v kódovaní `base64`. Na to používa knižnicu `construct`, ktorá sa využíva na vytváranie a parsovanie binárnych dátových štruktúr. Štruktúry, použité na dekódovanie záznamu z CT logu sú zobrazené v Ukážke certstruct. Inšpiráciou pri implementácii tejto časti bol nástroj `AdzeMan`.

- Každý záznam, ktorý obsahuje certifikát, načíta do natívneho `Python`-ovského objektu pomocou metódy `load_der_x509_certificate()` a vytiahne z neho niektoré atribúty, najdôležitejší je však atribút SAN.
- Skontroluje, či sa v atribúte SAN nenachádza vopred dohodnuté doménové meno. Ak áno, úspešne našiel hľadaný certifikát a skončí, ak nie, posunie sa na ďalší úsek a proces zopakuje.

```
CertEntry = Struct(
    "Length" / Int24ub,
    "CertData" / Bytes(this.Length)
)

CertChain = Struct(
    "ChainLength" / Int24ub,
    "Chain" / GreedyRange(CertEntry),
)

PreCertEntry = Struct(
    "LeafCert" / CertEntry,
    "Chain" / GreedyRange(CertEntry),
    Terminated
)
```

Ukážka 3.8: Binárne štruktúry na dekódovanie záznamu z CT logu

Po nájdení certifikátu z neho skript vytiahne doménové mená a pokúsi sa z nich dekódovať správu. Zoznam doménových mien najskôr usporiada lexikograficky, keďže CT logy ich v tomto poradí nemusia vracať, potom z každého z nich odstráni prefix, ktorý slúži len na určenie poradia, a z ostatných častí doménových mien znak `x`. Koncovú časť s registrovanou doménou, ktorú majú všetky doménové mená rovnaké, odignoruje, pretože tá sa na vkladanie správy nepoužíva. Časti kódovanej správy postupne spája za seba a keď ju už má celú, vykoná `encoded_message.upper()`, čím sa malé znaky v správe zmenia na veľké. To je nutné urobiť, pretože kódovanie `base32` pozná len veľké písmená a CT logy pri nahrávaní certifikátov konvertujú veľké písmená v doménových menách na malé. Nakoniec správu dekóduje a vypíše na výstup do konzoly. V prípade, že skript nenájde hľadaný certifikát v dostatočne dlhom úseku, skončí s chybou.

3.2 Konkrétne vlastnosti implementácie

Certifikačná autorita Let's Encrypt nahráva ňou vydané certifikáty do viacerých CT logov. Pomocou služby `crt.sh` sme zistili, že jedným z týchto CT logov je `argon2023` od spoločnosti Google. Let's Encrypt prevádzkuje aj svoj vlastný CT log `oak2023`, avšak v

ňom sa nám naše certifikáty nepodarilo nájsť. Domnievame sa, že LE nahráva do svojho logu len predcertifikáty, čo sa nám nepodarilo potvrdiť, ale silno tomu nasvedčuje fakt, že vo viacerých súvislých úsekoch, ktoré mali `100000+` záznamov, sa nachádzali len desiatky certifikátov (tie sa tam mohli dostať napr. manuálnym nahratím), zvyšok boli predcertifikáty.

Dôležitou vlastnosťou CT kanála je počet requestov, ktoré musí odosielateľ spraviť na CT log, aby našiel hľadaný certifikát. Keď sme sa snažili nájsť certifikát v CT logu `argon2023`, do ktorého LE certifikát nahráva automaticky, museli sme prejsť v priemere úsek dĺžky ≈ 4000 záznamov. Na jeden request vráti `argon2023` maximálne 32 záznamov, preto sa počet requestov na nájdenie certifikátu pohyboval medzi **120** až **150**. Manuálnym nahratím do CT logu `oak2023` sa nám tento počet dokázalo radikálne znížiť na **jednotky requestov**. Je to z dôvodu väčšieho intervalu, kde `oak2023` vracia až 256 záznamov, a nižšej frekvencie nahrávania certifikátov do tohto CT logu.

Čo sa priepustnosti týka, pomocou nami implementovaného CT kanála sa do jedného certifikátu vôjde *kódovaná* správa o veľkosti 18.2 kB. Pri použití kódovania `base32` má kódovaná správa v priemere o 60% bajtov viac ako pôvodná správa, preto môže odosielateľ poslať približne **11.38 kB** veľkú správu cez jeden certifikát.

3.3 Ukážka funkčnosti konceptu

Predpokladajme, že sú účastníci komunikácie dohodnutí na čase `2023-06-20 10:00:00`, v ktorom sa má správa odoslať. Ak neplánuje odosielateľ nahrávať certifikát do vybraného CT logu manuálne, tak mu stačí spustiť skript `send_message.py` v dohodnutom čase, ktorý správu zakóduje a pošle žiadosť o certifikát certifikačnej autorite, pozri Ukážku 3.9. Po úspešnom vydaní certifikátu skript ešte vytvorí `.log` súbor s dodatočnými informáciami o zakódovaní správy a doménových menách, pozri Ukážku 3.10.

V prípade, že chce odosielateľ nahrať certifikát do CT logu manuálne, musí najskôr zavolať skript `send_message.py`, čím dostane od CA certifikát so správou a potom v dohodnutom čase spustiť skript `add_chain.py`, čím ho manuálne nahrá do CT logu, pozri Ukážku 3.11.

Prijímateľ správy spustí v čase `2023-06-20 09:59:59` skript `receive_message.py`, ktorý si zapamätá index, od ktorého začne prehľadávať CT log a potom vypisuje na konzolu prehľadávané záznamy, až kým nenájde hľadaný certifikát, pozri Ukážku 3.12.

```
$ python send_message.py "I'm sending a message through the CT channel. I will send
    another one on 30.6.2023 at 9:00:00 UTC."

Saving debug log to /var/log/letsencrypt/letsencrypt.log
Requesting a certificate for ctlr.seclab.dcs.fmph.uniba.sk and
    a0jetw2idtmvxgi2lom4qgcidnmvzxgylhmuqhi2dsn52wo2baorugkicdkqqgg.
    x2dbnzxgk3boebesa53jnrwca43fnzscaylon52gqzlsebxw4zjan5xcamzqfy3.
    xc4mrqgizsayluea4tumbqhiydaicvkrbs4.ctlr.seclab.dcs.fmph.uniba.sk
Hook '--manual-auth-hook' ran with output:
 server reload successful

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/ctlr.seclab.dcs.fmph.uniba.sk/
    fullchain.pem
Key is saved at: /etc/letsencrypt/live/ctlr.seclab.dcs.fmph.uniba.sk/privkey.pem
This certificate expires on 2023-09-17.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the
    background.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
 * Donating to EFF:                  https://eff.org/donate-le
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Certificate with encoded message was successfully issued.
You can find additional data about message encoding in the log file: logs/2023-06-20
_10.00.00.log
```

Ukážka 3.9: Vkladanie správy do certifikátu

```
$ cat logs/2023-06-20_10.00.00.log

Common name: ctlr.seclab.dcs.fmph.uniba.sk
Message: I'm sending a message through the CT channel. I will send another one on
    30.6.2023 at 9:00:00 UTC.
Base32: JETW2IDTMVXGI2LOM4QGCIDNMVZXGYLHMUQHI2DSN52WO2BAORUGKICDKQQGG2DBNZXGK3BOEBE
SA53JNRWCA43FNZSCAYLON52GQZLSEBXW4ZJAN5XCAMZQFY3C4MRQGIZSAYLUEA4TUMBQHIYDAICVKRBS4

SANs:
a0JETW2IDTMVXGI2LOM4QGCIDNMVZXGYLHMUQHI2DSN52WO2BAORUGKICDKQQGG.
    x2DBNZXGK3BOEBESA53JNRWCA43FNZSCAYLON52GQZLSEBXW4ZJAN5XCAMZQFY3.
    xC4MRQGIZSAYLUEA4TUMBQHIYDAICVKRBS4.ctlr.seclab.dcs.fmph.uniba.sk
```

Ukážka 3.10: Dodatočné informácie o zakódovaní správy

```
$ python add_chain.py fullchain.pem --log-name oak2023

{
    "sct_version": 0,
    "id": "tz77JN+cTbp18jnFulj0bF38Qs96nzXEnh0JgSXttJk=",
    "timestamp": 1684223537493,
    "extensions": "",
    "signature": "BAMARzBFAiEA/6NWvotSG68wOolNruz+b0nv4IjqfMv0wX8/
        UYKIfFwCIA88q7qYyVvaBkqtb6t2+gBYN65p5SENHgnRC6uy0ohI"
}
Certificate was successfully inserted inside oak2023 CT log.
```

Ukážka 3.11: Manuálne nahratie certifikátu do CT logu

```
$ python receive_message.py ctlr.seclab.dcs.fmph.uniba.sk --log-name argon2023

Looking for certificate inside argon2023 from index 1076325998.

1076326000 | 2023-06-20 09:59:56 | shop.humask.com
1076326002 | 2023-06-20 09:59:59 | lockedroomcrafts.com
...
...
...
1076326160 | 2023-06-20 10:00:02 | www.halfchuboutfitters.com
1076326162 | 2023-06-20 10:00:00 | ctlr.seclab.dcs.fmph.uniba.sk,
    a0JETW2IDTMVXGI2LOM4QGCIDNMVZXGYLHMUQHI2DSN52WO2BAORUGKICDKQQGG.
    x2DBNZXGK3BOEBESA53JNRWCA43FNZSCAYLON52GQZLSEBXW4ZJAN5XCAMZQFY3.
    xC4MRQGIZSAYLUEA4TUMBQHIYDAICVKRBS4.ctlr.seclab.dcs.fmph.uniba.sk

Successfully found the message sent through the CT channel: I'm sending a message
    through the CT channel. I will send another one on 30.6.2023 at 9:00:00 UTC.
```

Ukážka 3.12: Hľadanie certifikátu so správou v CT logu

Záver

V našej práci sme sa venovali návrhu a implementácii skrytého komunikačného kanála, ktorý zneužíva mechanizmy certifikátov a CT logov na vzájomnú komunikáciu — nazvali sme ho CT kanál.

Detailne sme popísali, ako by CT kanál mohol fungovať. Na odosielanie správy sa využívajú certifikáty, pričom správa sa vhodným spôsobom vkladá do atribútu SAN. V atribúte SAN sa nachádzajú dodatočné doménové mená, pre ktoré žiadateľ chce, aby platil ten istý certifikát. Správa sa musí do atribútu SAN vhodne zakódovať, pretože v doménových menách sa môže nachádzať len obmedzená sada znakov. Opísali sme viaceré kódovanie a implementovali sme kanál, ktorý používa kódovanie `base32`. Navrhli sme aj dva spôsoby, ktorými vieme správy deliť medzi viaceré doménové mená. Po odoslaní správy vie prijímateľ nájsť certifikát so správou v CT logu, avšak na to potrebuje mať informáciu o čase, v ktorom bude správa poslaná. Pozreli sme sa na viaceré spôsoby prehľadávania CT logov, pričom sme implementovali spôsob, ktorý sekvenčne prehľadáva vhodne zvolený úsek. Na konci práce sme sa venovali implementačným detailom funkčného konceptu, ktorý využíva certifikačnú autoritu Let's Encrypt a vyhodnotili sme jeho vlastnosti.

Napriek snahe o podrobnú analýzu CT kanála, zostalo mnoho otvorených otázok, na ktoré sme sa v tejto práci nepozreli. Venovali sme sa len scenáru s jednosmernou komunikáciou dvoch účastníkov. Bolo by užitočné pozrieť sa aj na obojsmernú komunikáciu alebo komunikáciu medzi viacerými účastníkmi. Ďalej by bolo vhodné analyzovať vlastnosti CT kanála aj pre iné certifikačné autority a CT logy a pozrieť sa na to, ako ovplyvní výber CA jednotlivé vlastnosti kanála. Za zváženie stojí aj použitie iných typov kódovania, prípadne by bolo možné navrhnúť vlastné kódovanie, prispôsobené možnostiam a obmedzeniam pre doménové mená, ktoré by mohlo byť efektívnejšie ako bežne používané kódovania. Nevenovali sme sa ani šifrovaniu správy pred jej kódovaním, čo by mohlo ovplyvniť priepustnosť CT kanála.

Zaujímavou oblasťou, ktorej sme sa v práci nevenovali, je aj detekcia CT kanála. Detailnejšie by sme sa mohli pozrieť do CT logov a hľadať, či už sa niekto nesnažil zneužiť mechanizmus CT logov a certifikátov pred nami. Na to by bolo potrebné zozbierať dostatočne veľkú štatistickú vzorku z CT logov a potom v nej hľadať podozrivé certifikáty, napr. tie, ktoré majú dlhé doménové mená alebo maximálny počet doménových

mien v atribúte SAN.

Literatúra

[1] Butler W Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.

[2] Department of Defense. DEPARTMENT OF DEFENSE TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA. Technical report, 1985.

[3] DIANE Publishing Company. *A Guide to Understanding Covert Channel Analysis of Trusted Systems*. DIANE Publishing Company, 1994.

[4] Nitish Salwan, Sandeep Singh, Suket Arora, and Amarpreet Singh. An insight to covert channels. *CoRR*, abs/1306.2252, 2013.

[5] Richard A. Kemmerer. Shared resource matrix methodology: An approach to identifying storage and timing channels. *ACM Trans. Comput. Syst.*, 1(3):256–277, aug 1983.

[6] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, and David Cooper. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, May 2008.

[7] Ben Laurie, Adam Langley, and Emilia Kasper. Certificate Transparency. RFC 6962, June 2013.

[8] Domain names - implementation and specification. RFC 1035, November 1987.

[9] Simon Josefsson. The Base16, Base32, and Base64 Data Encodings. RFC 4648, October 2006.

[10] Richard Barnes, Jacob Hoffman-Andrews, Daniel McCarney, and James Kasten. Automatic Certificate Management Environment (ACME). RFC 8555, March 2019.

Príloha A: Obsah elektronickej prílohy

V elektronickej prílohe priloženej k práci sa nachádzajú zdrojové kódy ku jednotlivým komponentom. Pred ich spustením je treba vykonať nasledujúce príkazy:

```
$ python -m venv venv
$ source venv/bin/activate
$ (venv) pip install -r requirements.txt
```

Tým sa vytvorí virtuálne prostredie na prácu s `Python` programami a následne sa nainštalujú všetky potrebné knižnice. Okrem samotných komponentov sa v elektronickej prílohe nachádzajú aj pomocné súbory `ct_logs.py`, v ktorom je uložený zoznam CT logov s ich URL adresami a `certstruct.py`, v ktorom sú uložené binárne dátové štruktúry na dekódovanie záznamu z CT logu. Spúšťanie samotných komponentov je detailne opísané v Prílohe B.

Príloha B: Používateľská príručka

V tejto prílohe sa nachádzajú užívateľské príručky ku jednotlivým programom, ktoré slúžia na komunikáciu pomocou CT kanála.

```
$ (venv) python send_message.py --help

Usage: send_message.py [OPTIONS] MESSAGE [SUBJECT_NAME]

Arguments:
  MESSAGE       Message that will be encoded in the certificate. [required]
  [SUBJECT_NAME] Subject Name (SN) of the requested certificate, e.g.
                example.com. [default: ctlr.seclab.dcs.fmph.uniba.sk]

Options:
  --production / --no-production If true, it generates production certificate
                                 instead of testing certificate. [default:
                                 production]
  --verbose / --no-verbose       If true, it prints additional information
                                 about encoding and content of SANs.
                                 [default: no-verbose]
  --dry / --no-dry               If true, it just mimics running a certbot
                                 script. [default: no-dry]
  --help                         Show this message and exit.
```

Ukážka 3.13: Zakódovanie správy do certifikátu

```
$ (venv) python add_chain.py --help

Usage: add_chain.py [OPTIONS] CHAIN_FILE

Arguments:
  CHAIN_FILE Path to the .pem file with chain of certificates. Typically, the
             file is located at /etc/letsencrypt/live/.../fullchain.pem
             [required]

Options:
  --log-name TEXT         The CT log in which the certificate should be
                          inserted. [default: oak2023]
  --verbose / --no-verbose If true, it prints additional information about
                          extracted chain and response. [default: no-
                          verbose]
  --help                  Show this message and exit.
```

Ukážka 3.14: Manuálne vloženie certifikátu do CT logu

```
$ (venv) python receive_message.py --help

Usage: receive_message.py [OPTIONS] [SUBJECT_NAME]

Arguments:
  [SUBJECT_NAME] Subject Name (SN) of the wanted certificate, e.g. example.com.
                 [default: ctlr.seclab.dcs.fmph.uniba.sk]

Options:
  --log-name TEXT The CT log in which the certificate should be found.
                  [default: argon2023]
  --help          Show this message and exit.
```

Ukážka 3.15: Nájdenie certifikátu so správou v CT logu