
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ**
**DEPARTMENT OF COMPUTER SYSTEMS**


# POČÁTEČNÍ AUTOMATICKÁ KONFIGURACE V NETCONF SERVERU

**ZERO TOUCH PROVISIONING IN NETCONF SERVER**

**DIPLOMOVÁ PRÁCE**
**MASTER'S THESIS**


**AUTOR PRÁCE** — **Bc. PAVOL VICAN**
**AUTHOR**

**VEDOUCÍ PRÁCE** — **Ing. JAN WRONA**
**SUPERVISOR**



**BRNO 2018**

**Vysoké učení technické v Brně - Fakulta informačních technologií**

Ústav počítačových systémů Akademický rok 2017/2018

# Zadání diplomové práce

Řešitel: **Vican Pavol, Bc.**
Obor: Počítačové sítě a komunikace
Téma: **Počáteční automatická konfigurace v NETCONF serveru**
**Zero Touch Provisioning in NETCONF Server**
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s modelovacím jazykem YANG, konfiguračním protokolem NETCONF a jejich implementací v rámci projektu Sysrepo.
2. Podrobně nastudujte mechanismus Zero Touch.
3. Navrhněte vhodný způsob implementace a integrace mechanismu Zero Touch do projektu Sysrepo.
4. Navržený postup zrealizujte.
5. Jako součást implementace vypracujte sadu testů a uživatelskou dokumentaci.
6. Zhodnoťte dosažené výsledky a diskutujte možnosti pokračování projektu.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Wrona Jan, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

prof. Ing. Lukáš Sekanina, Ph.D.
*vedoucí ústavu*

# Abstrakt

Administrátori počítačových sietí potrebujú pokročilé nástroje na konfiguráciu sieťových zariadení. Pre tento účel vznikol protokol NETCONF a modelovací jazyk YANG. Cieľom tejto práce je implementovať mechanizmu zero touch, ktorý slúži na počiatočnú konfiguráciu NETCONF servera.Tento mechanizmus bude integrovaný do dátového úložiska sysrepo.



# Abstract

Computer network administrators need advanced tools to configure network devices. For this purpose, the NETCONF protocol and the YANG modeling language were developed. The aim of this thesis is to implement the zero touch mechanism that serves the initial configuration of the NETCONF server. This mechanism will be integrated into the sysrepo datastore.



# Kľúčové slová

NETCONF, YANG, sysrepo, Zero Touch, počiatočná konfigurácia

# Keywords

NETCONF, YANG, sysrepo, Zero Touch


# Citácia

VICAN, Pavol. *Počáteční automatická konfigurace v NETCONF serveru.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Wrona

# Počáteční automatická konfigurace v NETCONF serveru

## Prehlásenie

Prehlasujem, že som túto semestrálnu prácu vypracoval samostatne pod vedením pána Ing. Jána Wrony. Ďalšie informácie mi poskytli Ing. Michal Vaško a RNDr. Radek Krejči. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....

Pavol Vican

23. mája 2018

# Obsah

# Kapitola 1

# Úvod

Počítačové siete sú v dnešnej dobe komplexné a často nevyhnutné pri poskytovaní rôznych služieb zákazníkom. S narastajúcou veľkosťou sietí vzniká problém konfigurácie jednotlivých sieťových zariadení. Správcovia sieti museli vykonávať inštaláciu a konfiguráciu zariadení vrátane aktualizácie operačného systému manuálne. Je to často časovo náročný proces, najmä pri aplikovaní konfigurácie na viacero zariadení, ktoré je potrebné nainštalovať, nakonfigurovať alebo spravovať v prostrediach ako sú dátové centrá. Tento prístup spravovania sietí je neefektívny a zároveň finančne nákladný. Aby bolo možné ušetriť ľudské a aj finačné zdroje, je nutné prejsť k automatizácii procesu konfigurácie. Výrobcovia sieťových zariadení poskytovali rôzne rozhrania pre konfiguráciu a nebolo jednoduché tieto rozhrania spojiť do jedného rozhrania, ktoré by sa využívalo pre automatizáciu. Preto vznikol protokol NETCONF, ktorý umožní automatizáciu v konfigurovani a poskytne mechanizmus konfigurácie pre softvérovo definované siete [6].

Vďaka automatizácii a využitia protokolu NETCONF, ktorý umožňuje rozširovať funkcionalitu, je možné využiť ďalších mechanizmov, ktoré pomáhajú znižovať náklady. Medzi tieto mechanizmy patrí aj Zero Touch. Ponúka poskytovateľom komunikačných služieb urýchlenie aktivácii nových služieb. Ako príklad si môžeme predstaviť zákazníka, ktorý si objedná službu IPTV. Poštou dostane zariadenie, ktoré len pripojí ksieti a napájaniu. Zariadenie sa samo autentizuje voči konfiguračnému serveru a stiahne si najnovší softvér a vhodnú konfiguráciu. Tento mechanizmus znižuje náklady, zrýchľuje aktiváciu služby a v poslednom rade zvyšuje spokojnosť zákazníka. Navyše tento mechanizmus je plne bezpečný, pretože využíva osvedčené kryptografické metódy na bezpečnú autentifikáciu zariadenia a zabezpečenie integrity softvéru a konfiguračných dát.

Cieľom diplomovej práce je implementovať počiatočnú automatickú konfiguráciu NETCONF servera. Tento mechanizmus je známy pod pojmom Zero Touch a pre protokol NETCONF je len zatiaľ v návrhu štandardu, čo komplikovalo návrh implementácie a aj samotnú implementáciu. Diplomová práca bude rozširovať funkcionalitu už existujúceho NETCONF servera Netopeer2, ktorý je vyvíjaný organizáciou CESNET. Netopeer2 využíva dátové úložisko sysrepo, pričom podpora počiatočnej automatickej konfigurácie bude integrovaná práve do tohto dátového úložiska.

Štruktúra práce bude rozdelená do siedmich kapitol. Kapitola 2 sa zaoberá protokolom NETCONF, RESTCONF a modelovacím jazykom YANG. V kapitole 3 rozoberá existujúca implementácia NETCONF protokolu a dátového úložiska. Taktiež je zobrazená architektúra NETCONF serveru Netopeer2 a dátového úložiska sysrepo. Kapitola 4 popisuje samotný mechanizmus Zero Touch, naviazanie a sťahovanie konfiguračných dát a overovanie ich platnosti. Taktiež zobrazuje zdroje, z ktorých môžu byť tieto dáta stiahnuté. Kapitoly 5 a 6

sa zobrazujú návrhom a implementáciou mechanizmu Zero Touch a integráciou do dátového úložiska sysrepo. V kapitole 7 je rozoberaný spôsob testovania a možné prípady získania počiatočných konfiguračných dát.

# Kapitola 2

# Použité technológie

Pri spravovaní veľkých sieti nie je možné efektívne reagovať na zmeny bez automatizácie procesu konfigurácie. Preto bolo potrebné, aby vznikli protokoly podporujúce automatizáciu konfigurácie. Preto vznikol protokol NETCONF, ktorý umožní konfigurovanie sieťových zariadení zautomatizovať. Aby bolo možné automatizovať konfiguráciu, je nutné mať konfiguračné dáta zobrazené čitateľne pre správcu sietí a zároveň vhodne štruktúrované pre automatickú konfiguráciu. Preto vznikol nový modelovací jazyk YANG, ktorý vychádza z týchto požiadavkov. Po uplynutí pár rokov používania protokolu NETCONF sa dospelo k tomu, že pre niektoré prípady užitia je zbytočne zložitý najmä implementáciou. Z tohto dôvodu vznikol protokol RESTCONF, ktorý je založený na HTTP protokole a v súčastnosti je medzi programátormi obľúbený.

## 2.1 YANG

YANG je modelovací jazyk, ktorý vznikol pri špecifikácii protokolu NETCONF. Využíva sa na definovanie štruktúry konfiguračných dát a špecifických funkcií protokolu NETCONF napr. notifikácie a RPC. Kľúčovou vlastnosťou jazyka je jasnosť a čitateľnosť pre človeka. Prvá špecifikácia jazyka YANG bola štandardizovaná v roku 2010 v RFC 6020 [4]. S rozvojom protokolu NETCONF prišla aktualizácia tohto jazyka na verziu 1.1 v RFC 7950 [5].

Jazyk YANG umožňuje rozdeliť popis dát do logických celkov, ktoré sa nazývajú moduly. Každý modul je identifikovaný pomocou svojho názvu a URI (*Uniform Resource Identifier*). Modul sa môže ešte rozdeliť do submodulov, čo zjednodušuje udržovanie komplexných modulov. Tieto submoduly sú viditeľné len z modulu ku ktorému patria. Navyše jazyk YANG dovoľuje skladať moduly do seba a tak vytvárať hierarchiu modulov.

Definícia dát a aj samotné dáta sú usporiadané do stromovej štruktúry, ktorá napomáha rýchlemu vyhľadávaniu pomocou XPATH výrazov. Jazyk YANG ukladá všetky hodnoty dát pomocou uzlov *leaf* a *leaf-list*. Na udržanie štruktúry dát sa využíva uzol *container*. Pre definovanie komplexnej štruktúry sa používa uzol *list*, ktorý môže mať viacero instancii v dátovom strome. Pre identifikáciu určitej instancii sa využíva kľúč, ktorý je uložený minimálne v jednom *leaf* uzle. Uzol *RPC* definuje vstupné a výstupné parametre pre vzdialené volanie procedúr a uzol *notification* definuje štruktúru dát pre asynchrónne správy. V tomto jazyku existujú ešte niekoľko výrazov, pomocou ktorých sa upravuje definícia dát.

Jazyk YANG podporuje mechanizmus, ktorý označuje určitú časť modulu ako voliteľnú. Tento mechanizmus napomáha návrhárovi modulu rozdeliť dátový model popisujúci kom-

plexný systém do množiny voliteľných vlastností. Každá vlastnosť je identifikovaná svojím názvom, pričom každé zariadenie môže povoliť alebo zakázať túto vlastnosť.

Kľúčovou charakteristikou jazyka YANG je rozšíriteľnosť (anglicky extension). Vďaka tejto vlastnosti je možné pridávať funkcionalitu do tohto jazyka. Umožňuje vytvárať nové jazykové konštrukcie, meniť sémantický význam určitých jazykových konštrukcií a pod.

Pre grafické znázornenie stromovej štruktúry časti YANG modulu sa budú požívať nasledujúce symboly:

- zátvorky `[]` — označuje kľúč identifikujúci instanciu uzla *list*
- zátvorky `{}` — označuje názov vlastnosti, ktorá musí byť povolená, aby daný podstrom bol dostupný
- skratky pred názvami dátových uzlov: `rw`(čítanie a zápis) predstavuje konfiguračné dáta, `ro`(iba na čítanie) predstavujú stavové dáta a `x` predstavuje spustenie RPC
- symboly za názvami dátových uzlov: `?` znamená voliteľný uzol, `*` označuje uzol *leaf-list* s viacerými hodnotami
- symbol `...` — informuje o podstrome, ktorý nie je zobrazený

Pre parsovanie jazyka YANG existuje viacero implementácii (pyang, libyang a pod.). V projekte sysrepo implementujúce dátové úložisko konfiguračných dát sa využíva knižnica libyang. Táto implementácia dokáže parsovať moduly jazyka YANG, validovať konfiguračné dáta voči danému modelu a podporuje spracovanie operácii, ktoré sú definované v protokole NETCONF. Zaujímavosťou tejto implementácie je vyriešenie podpory rozšírenia jazyka YANG. Pre podporu jednotlivých rozšírení treba implementovať plugin, ktorý sa pri spustení knižnice libyang načíta. Tento plugin kontroluje vhodné umiestnenie rozšírenia v danom module, syntax a sémantiku rozšírenia.

## 2.2 NETCONF

Protokol NETCONF (*NETwork CONFiguration*) slúži na vzdialenú konfiguráciu a správu sieťových zariadení. Poskytuje jednoduchý mechanizmus, ktorý umožňuje prácu s konfiguráciou sieťového zariadenia. Vďaka nemu sa dajú nastaviť ľubovoľné parametre daného zariadenia, získať konfiguračné dáta alebo stavové informácie.

Tento protokol bol štandardizovaný v roku 2006 v RFC 4741 organizáciou IETF (*Internet Engineering Task Force*). V roku 2011 bol aktualizovaný na novú verziu 1.1, ktorá bola publikovaná v RFC 6241. V súčasnosti sa protokol NETCONF rýchlo rozvíja a aktualizuje, vznikajú nové štandardy YANG modulov a rozširuje sa funkcionalita o nové mechanizmy ako napríklad Zero Touch.

Protokol NETCONF využíva pre prenos dát formát XML (*Extensible Markup Language*), ktorý poskytuje usporiadanie dát do hierarchickej štruktúry, pričom zostávajú stále čitateľné pre človeka. Toto hierarchické usporiadanie využíva protokol pre vytváranie a posielanie správ vzdialeného vykonávania procedúr RPC (*Remote Procedure Call*). Štruktúra dát je definovaná pomocou modelovacieho jazyka YANG, ktorý je popísaný v kapitole 2.1.

Ďalšou výhodou využitia XML formátu je, že viacero aplikácii je schopné pracovať s týmito dátami. To umožňuje previazať NETCONF s ďalšími aplikáciami a vytvoriť komplexný systém (napr. pre automatickú konfiguráciu rozsiahlych sietí).

V protokole NETCONF je použitá klasická komunikácia typu klient-server. Klient je agent, z ktorého vychádzajú požiadavky. Väčšinou je to administrátor, ktorý používa nástroj na konfiguráciu využívajúci tento protokol. Server je zariadenie, ktoré chceme nakonfigurovať, prípadne z ktorého chceme získať požadované informácie. Server vykoná operáciu, ktorá bola poslaná klientom. Po vykonaní operácie server odošle odpoveď, ktorá môže obsahovať požadované dáta alebo len potvrdzuje úspešné/neúspešne vykonanie operácie.

Jedna z hlavných vlastností protokolu NETCONF je rozšíriteľnosť. Samotný protokol ponúka malú množinu operácii, ktoré skoro výhradne pracujú s konfiguráciou ako s celkom. Dokáže vytvárať, kopírovať alebo mazať konfiguráciu. Táto malá množina operácii je výhodná z dôvodu dostupnosti operácii na každom zariadení podporujúce NETCONF protokol. V prípade nutnosti špecifických operácii, ktoré poskytne výrobca zariadenia, sa môže využiť vlasnosť protokolu NETCONF - rozšíriteľnosť (*capabilities*). Aby bolo možné využiť tieto rozšírenia, musí ich podporovať klient aj server. Preto klient pri inicializácii spojenia získava informácie od servera, v ktorých sa nachádzajú podporované rozšírenia.

Existuje viacero implementácii tohto protokolu, pričom v projekte sysrepoje použitá knižnica libnetconf2. Táto knižnica je implementovaná v jazyku C. Knižnica podporuje transportné protokoly SSH, TLS (*Transport Layer Security*) a rozšírenie Call Home.

### 2.2.1 Architektúra protokolu

Architektúru protokolu NETCONF môžeme rozdeliť do 4 logických častí, ktoré sú zobrazené na obrázku 2.1. Konfiguračné dáta sú posielané v nejakej operácii, ktorú vykoná server. Medzi základné operácie protokolu NETCONF patrí <get>, <get-config>, <edit-config>, <copy-config>, <delete-config>, <lock>, <unlock>, <kill-session>, <close-session>. Tieto operácie sú zaobalené do vzdialeného volania procedúr. Aby bolo možné zaručiť bezpečnosť vzdialeného volania procedúr, boli kladené nasledovné podmienky na transportný protokol:

- identifikácia typu sedenia (server/klient),
- spoľahlivosť a sekvečne doručovanie dát
- autentizácia, autentifikácia, integrita,
- trvalé spojenie.

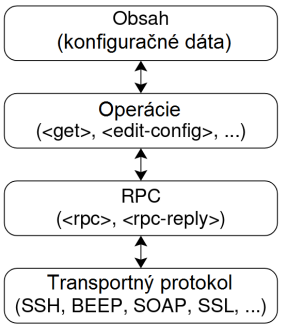
Vhodný protokol, ktorý splňuje tieto požiadavky, je SSH (*Secure Shell*) [2], ktorý je nutne aj podporovať. Ďalší vhodný protokol pre prenášanie NETCONF dát je TLS, ktorý je v súčasnosti používaný pri bezpečnom spojení HTTPS.

### 2.2.2 Dátové úložisko konfiguračných dát

Protokol NETCONF definuje v základnom modely jedno dátové úložisko konfiguračných dát, ktoré sa nazýva *running*. V prípade potreby je možné rozšíriť dátové úložiska na *startup* a *candinate*. Tejto dátové úložiska musia byť implementované pomocou rozšírenia (capabilities). Na obrázku 2.2 sú zobrazené spôsoby použitia jednotlivých dátových úložísk.

#### Dátové úložisko - running

Toto úložisko obsahuje aktuálne konfiguračné dáta, podľa ktorých je zariadenie nakonfigurované. V prípade existencie dátového úložiska *startup*, sa pri štarte zariadenia mali prekopírovať konfiguračné dáta z tohto dátového úložiska do *running* dátového úložiska.



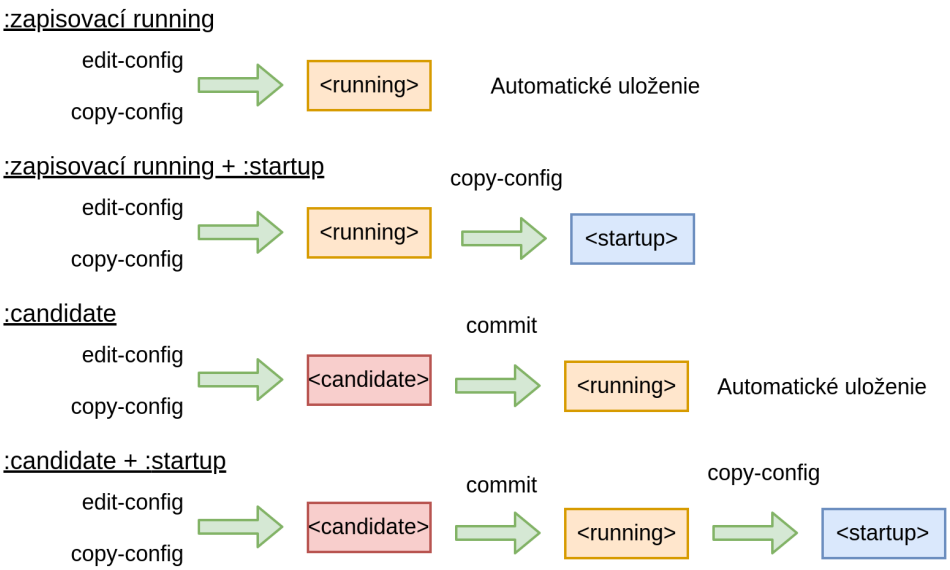

Obr. 2.1: Architektúra protokolu NETCONF

### Dátové úložisko - startup

Toto úložisko obsahuje konfiguračné dáta, ktoré sa použijú pri štarte zariadenia. Toto úložisko je dostupné v prípade, že server podporuje rozšírenie *startup*. Zároveň s týmto dátovým úložiskom server nebude automaticky ukladať dátové úložisko running na nevolatívnu pamäť. Preto treba zavolať operáciu <copy-config> nad dátovým úložiskom *running*.

### Dátové úložisko - candidate

Toto úložisko je dostupné, ak server podporuje rozšírenie *candidate*. V tomto dátovom úložisku sa nachádzajú rovnaké dáta ako v dátovom úložisku *running*. Zbiera všetky zmeny, ktoré sú vyžiadané operáciou <edit-config>. Klient môže vytvoriť zmeny a može požiadať server o validáciu zmenených dát. Po všetkých zmenách sa musí uložiť do dátového úložiska *running* pomocou operácie <commit>.



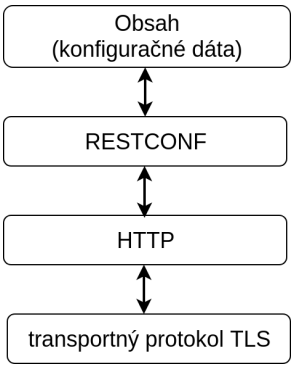

Obr. 2.2: Spôsoby konfigurácie dátových úložísk

## 2.3 RESTCONF

Webové aplikácie často vyžadujú prístup ku konfiguračným dátam, stavovým dátam, vzdialeným volaniam procedúr (RPC) alebo upozornenie na udalosti v rámci sieťového zariadenia. Protokol NETCONF bol pre tento účel nevhodný, lebo je náročný na implementáciu. Preto vznikol protokol RESTCONF [3].

Protokol RESTCONF (*REST CONFiguration*)je založený na populárnom architektonickom štýle REST (*REpresentational State Transfer*), ktorý využíva vlastností protokolu HTTP. Vďaka tomuto návrhu protokol RESTCONF umožňuje aplikáciam jednoduchý prístup a manipuláciu s dátami v textovej forme pomocou jednotného rozhrania a množinou predefinovaných bezstavových operácií. Zjednodušuje to implementáciu na strane klienta, kde klient je väčšinou integrovaný do systému monitorovania sieťových zariadení.

Architektúra protokolu RESTCONF je zobrazená na obrázku 2.3. Tento protokol poskytuje zjednodušené rozhranie k prístupu dátového úložiska. Tento protokol sa snaží zjednodušiť rozhranie k prístupu dátového úložiska. Pre zaručenie bezpečnosti sa používa transportný protokol TLS. RESTCONF klient by sa mal autentizovať pomocou klientského TLS certifikátu a zároveň overiť reťazec certifikátov poslaných serverom. V prípade, že server nevie autentizovať klienta mal by poslať chybový stav.




Obr. 2.3: Architektúra protokolu RESTCONF

Pre prístup k jednotlivým zdrojom (konfiguračné dáta, stavové dáta, ...) sa využíva URI. RESTCONF server môže predefinovať koreňový uzol, od ktorého bude dostupné rozhranie pre RESTCONF protokol. Toto rozhranie je zobrazené na obrázku 2.4, kde si môžeme všimnúť, že všetky konfiguračné dáta sa nachádzajú v uzle *data*, vzdialené volanie procedúr definované YANG modulom v uzle *operation* a *yang-library-version* obsahuje dáta modulu *ietf-yang-library*. Klient musí na začiatku komunikácie získať URI, ktorý odpovedá tomuto koreňovému uzlu.

```
+---- {+restconf}
          +---- data
          | ...
          +---- operations?
          | ...
          +--ro yang-library-version string
```

Obr. 2.4: Stromový diagram štruktúry RESTCONF rozhrania

Ďalší rozdiel medzi protokolom NETCONF a RESTCONF je formát posielaných správ. Protokol NETCONF využíva len XML formát. V prípade protokolu RESTCONF umožňuje vybrať formát poslaných a prijatých dát v HTTP hlavičke. K dispozícii sú dva dostupné formáty:

- XML - application/yang-data+xml
- JSON (*JavaScript Object Notation*) - application/yang-data+json

V protokole NETCONF je definovaná množina operácii s dátovým úložiskom (napr. vytvorenie, čítanie, aktualizovanie, vymazanie dát). V protokole RESTCONF sa na tieto operácie využívajú HTTP metódy GET, POST, HEAD, PUT, DELETE a PATCH. V tabuľke 2.1 sú zobrazené HTTP metódy k jednotlivým NETCONF operáciám.

| RESTCONF operácie | NETCONF operácie |
|---|---|
| HEAD | <get-config>, <get> |
| GET | <get-config>, <get> |
| POST | <edit-config>(nc:operation="create") |
| POST | vyvolať RPC operáciu |
| PUT | <copy-config>(PUT na dátové úložisko) |
| PUT | <edit-config>(nc:operation="create/replace") |
| PATCH | <edit-config>(nc:operation zavisí na PATCH obsahu) |
| DELETE | <edit-config>(nc:operation="delete") |

Tabuľka 2.1: Porovnanie operácii v NETCONF a RESTCONF protokole

# Kapitola 3

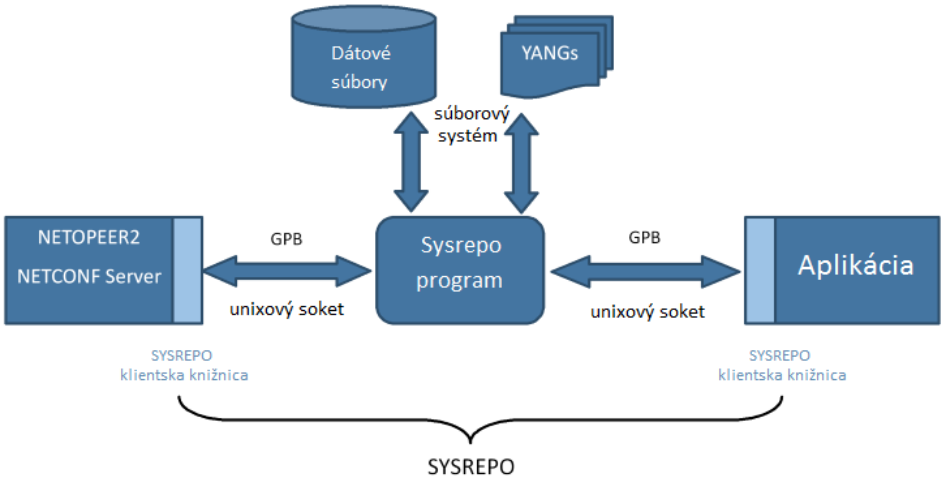
# NETCONF server s dátovým úložiskom

V tejto kapitole bude popísaná architektúra dátového úložiska sysrepo [1]. Toto dátové úložisko je integrované do implementácie NETCONF servera Neetopeer2.

## 3.1 Sysrepo

Veľa programov a démonov potrebujú konfiguračné súbory, aby mohli správne fungovať. Na ukladanie konfigurácii sa často používa plochý model, ktorý ukladá konfiguráciu ako celok do súboru. Tento prístup je problematicky pri zmene a validácii konfiguračných dát. Preto sa projekt sysrepo snaží zmeniť tento prístup.

Cieľom sysrepa je vytvoriť dátové úložisko, ktoré bude definované pomocou modelovacieho jazyka YANG. Umožňuje rozlišovať konfiguračné a stavové dáta a pracovať s nimi. Sysrepo povoľuje vývojárom aplikácii sa zamerať na dôležitú funkcionalitu, pokým sysrepo sa stará o predávanie konfiguračných a stavových dát. Taktiež zabezpečuje konzistenciu uložených dát v dátovom úložisku a vynúti obmedzenia, ktoré sú definované pomocou YANG modulov. Vďaka intergrácie NETCONF servera Netopeer 2 sa aplikácie stanú vzdialene ovládateľné pomocou protokolu NETCONF, čo je zobrazené na obrázku 3.1.



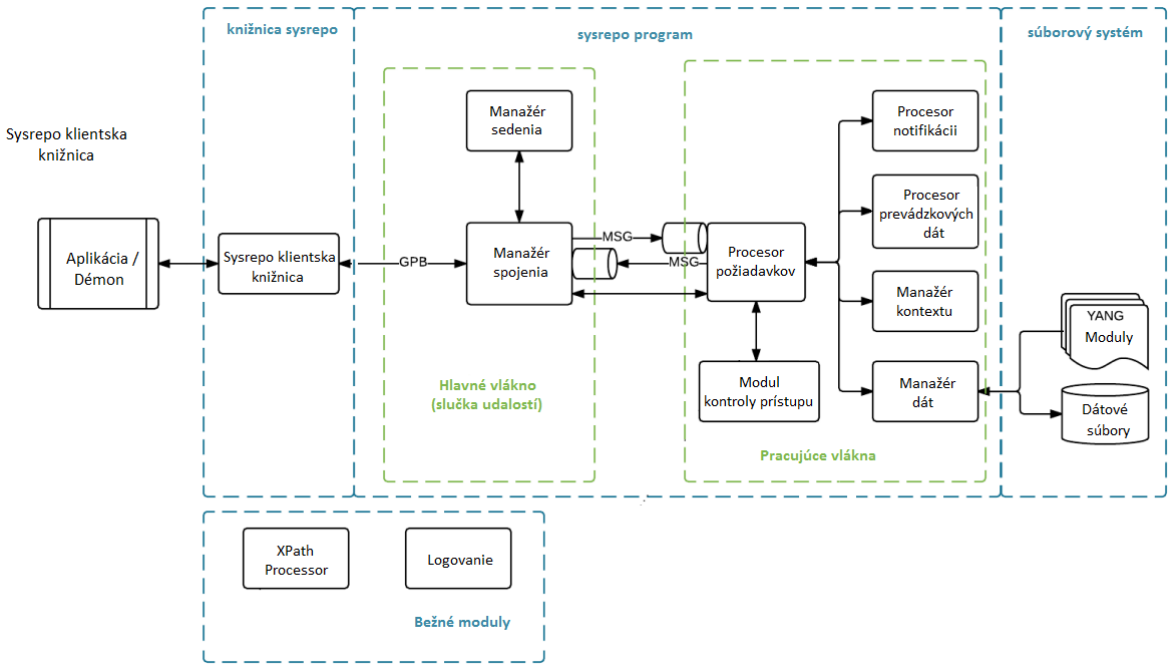

Obr. 3.1: Architektúra dátového úložiska sysrepo

Jedna z kľúčových vlastností sysrepa je viacero bodov zlyhania. V štandardnom režime klientská knižnica sysrepo komunikuje cez spustený `sysrepod(8)`, ktorý riadi všetky operácie v dátovom úložisku. Ak `sysrepod(8)` nebeží z rôznych dôvodov (napríklad nie je spustený alebo ešte nie je inicializovaný, spadol z rôznych dôvodov), klientská knižnica môže vykonať väčšinu funkcionality prístupu k dátam sama.

Sysrepo pozostáva z dvoch hlavných komponent – klientská knižnica a `sysrepod(8)`. Klientská knižnica poskytuje API (Application Programming Interface) pre aplikácie a komunikáciu so `sysrepod(8)`. V prípade nemožnosti komunikácie so `sysrepod(8)`, klientská knižnica si vytvorí vlastnú instanciu `sysrepod(8)` a vykoná požadované operácie. Keď aplikácia sa odpojí od `sysrepod(8)`, lokálna instancia sa ukončí.

Na obrázku 3.2 je zobrazená detailná architektúra sysrepa. Klientská knižnica komunikuje so `sysrepod(8)` pomocou GPB (*Google Protocol Buffer*). Tento mechanizmus umožňuje flexibilnú, efektívnu serializáciu štruktúrovaných dát. Výhodou tohto mechanizmu je jazyková a platformová neutralita. Štruktúra dát je definovaná v konfiguračnom súbore, pričom sa pri kompilácii programu vygenerujú štruktúry a funkcie v jazyku C.

**Manažér spojenia** sa stará o vytvorenie nového spojenia, ukončenie spojenia a komunikáciu so spomínanou klientskou knižnicou. Aby bolo možné riadiť jednotlivé spojenia, musí byť spustená slučka zachytávajúca udalosti. Táto slučka riadi všetky spojenia a preposiela správy od klientskej knižnice do procesoru požiadavok. Spustenie slučky sa vykoná pomocou funkcie *cm_start*. Manažér spojenia môže fungovať v dvoch režimoch: *démon* alebo *lokálna instancia*. Hlavná odlišnosť medzi týmito režimami je rozdielne spustenie slučky udalostí. V režime démon beží slučka v hlavnom vlákne a je blokujúca, pokým nepríde príkaz zastavenia pomocou funkcie *cm_stop*. Keď je manažér spojenia v režime lokálna instancia, beží slučka udalostí na dedikovanom vlákne, pričom je neblokujúca pre volajúce vlákno.




Obr. 3.2: Detailná architektúra dátového úložiska sysrepo

**Manažér sedenia** má za úlohu spravovať jednotlivé sedenia (session). Každé spojenie môže mať niekoľko sedení. Pre rýchle vyhľadávanie sedení sa používa index sedenia, ktorý je unikátny. Tento index sa získa pri vytvorení sedenia. Taktiež má uložený identifikátor sysrepo užívateľa.

**Procesor požiadavkov** spracováva individuálne požiadavky od klienta, ktoré sú doručované pomocou manažéra spojenia. Komunikácia medzi manažérom spojenia a procesorom požiadavok je tiež založená na sedeniach. Manažér spojenia využíva funkcie *rp_session_start* a *rp_session_stop*, ktoré oznamujú začiatok a koniec sedenia pre procesor požiadavkov.

**Modul prístupu kontroly** poskytuje autorizáciu požadovaných operácii na dátové úložisko. Umožňuje dočasne vymeniť identitu procesu podľa poskytnutého užívateľského poverenia. Tento modul je dôležitý, lebo YANG moduly sú uložené na súborov systéme, kde sú nastavené oprávnenia prístupu k danému súboru. Navyše kontroluje aj nastavené prístupové práva v protokole NETCONF.

**Procesor notifikácii** spracováva notifikácie a odosiela ich klientovi. Notifikácie sú asynchrónne správy, ktoré generuje server. Pomocou nich môže server informovať klienta o nepredvídaných udalostiach (napr. spadnutie linky).

**Procesor prevádzkových dát** je zodpovedný za ukladanie stavu spojeného s YANG modulom, ktorý by mal zotrvať aj pri ukončení programu sysrepo. Sú to informácie, ktoré sa nepovažujú za konfiguračné dáta, ale sú nevyhnutné pre YANG modul (napr. povolené vlastnosti, aktívny odberatelia notifikácii a pod.).

**Manažér dát** sprístupňuje dáta z dátových úložísk. Sysrepo podporuje všetky tri typy dátových úložísk, ktoré boli spomenuté v sekcii 2.2.2. Pre každý YANG modul, ktorý má byť implementovaný, sú vytvorené súbory *running*, *startup* a *persist*. Každé vytvorené sedenie pomocou procesoru požiadavok môže pristupovať len jednému dátovému úložisku súčasne. Manažér dát si ukladá do svojho kontextu zmenené dáta z jednotlivých dátových úložísk, pričom uloženie dát sa vykoná pri operácii <commit>, <copy-config> a <edit-config>.

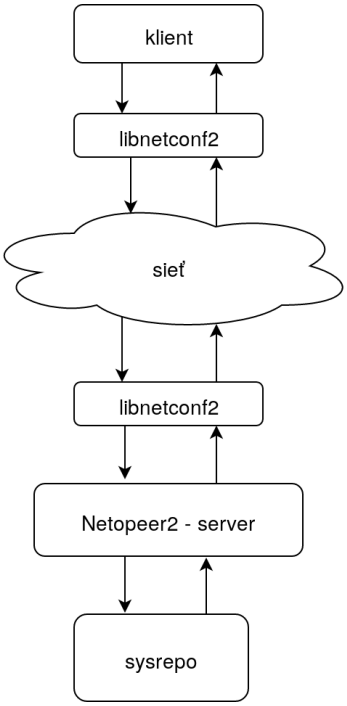
## 3.2 Netopeer2

Netopeer2 je open-source projekt, ktorý je vyvíjaný organizáciou CESNET. Je to množina nástrojov implementujúca konfiguráciu sieťových zariadení pomocou protokolu NETCONF. Pre transportnú vrstvu využíva knižnicu libnetconf2, ktorá poskytuje prenos dát pomocou bezpečných transportných protokolov SSH a *Transport Layer Security* (TLS). Na obrázku 3.3 je zobrazená komunikácia klienta s NETCONF serverom Netopeer2. Netopeer2 je možné rozdeliť do dvoch logických celkov: serverová časť a klientská časť.

**Serverová časť** implementuje základné operácie, ktoré sú definované v protokole NETCONF. Server umožňuje pripojenie viacerých klientov súčasne. Z tohto dôvodu je komunikácia rozdelená do sedení, pričom jedno sedenie odpovedá jednému klientovi. Počet pripojených klientov je neobmedzený. Pre dátové úložisko je použitá implementácia sysrepo, čo prináša mnoho výhod (napr. využitie identifikácie klienta na základe sedenia, rýchle vyhľadanie dát, ochrana a validácia dát).

**Klientská časť** obsahuje jednoduché konzolové rozhranie pre pripojenie klienta na NETCONF server. Pre grafické rozhranie je vytvorený špeciálny projekt Netopeer2GUI[1], ktorý ponúka webové rozhranie pre management zariadení, ktoré podporujú protokol NETCONF.

[1] https://github.com/CESNET/Netopeer2GUI




Obr. 3.3: Komunikácia klienta s NETCONF serverom Netopeer2

# Kapitola 4

# Mechanizmus Zero Touch

Mnohý zákazníci, ktorý si objednajú službu od poskytovateľa komunikačných služieb, nevedia správne nakonfigurovať sieťové zariadenie. Preto je nutné poslať technika k zákazníkovi, aby nainštaloval a nakonfiguroval sieťové zariadenie. Ale poskytovateľ komunikačných služieb sa snaží znižovať potrebné náklady pri zavedení sieťového zariadenia k novému zákazníkovi. Preto potrebuje mechanizmus, ktorý by dokázal splniť tento cieľ. Zároveň poskytovateľ komunikačných služieb ma širokú ponuku služieb, pričom môže využívať mnoho sieťových zariadení, ktoré treba automatizovane spravovať. K tomuto účelu môže použiť protokol NETCONF, ktorý bol popísaný v sekcii 2.2. Výrobcovia sieťových zariadení ponúkajú proprietárne riešenia pre automatickú konfiguráciu. Preto sa organizácia IETF začala zaoberať s týmto problémom a snaží sa vytvoriť štandard, ktorý by popisoval spôsob automatickej konfigurácie. Vznikol návrh štandardu s názvom *draft-ietf-netconf-zerotouch* [10], ktorého autorom je Kent Watson.

Tento návrh štandardu ponúka mechanizmus samozavádzania (anglicky bootstrapping) pre zariadenia podporujúce NETCONF protokol. Tento mechanizmus umožní získať bezpečnou formou počiatočnú konfiguráciu sieťových zariadení. Bezpečnosť je vynútená využívaním kryptografickej štruktúry CMS (*Cryptographic Message Syntax*) [8], ktorá dokáže podpísať a aj zašifrovať dáta. Taktiež sa využíva bezpečné spojenie s RESTCONF serverom pomocou transportného protokolu TLS. Tento mechanizmus sa spustí pri prvom zapnutí zariadenia, ktoré je v továrenskom nastavení.

## 4.1 Typy správ a objektov mechanizmu

Zero Touch mechanizmus využíva 2 typy správ, resp. informácií pri procese samozávádzania:

- informácia s presmerovaním (redirect information),
- informácia s konfiguráciou (onboarding information).

Informácia s presmerovaním obsahuje dáta, ktoré sú využité pri naviazaní spojenia s iným samozavádzacím serverom. Táto informácia je uložená v štruktúre ktorá je definovaná pomocou jazyka YANG. Jej definícia je zobrazená na obrázku 4.1. Táto informácia obsahuje zoznam údajov pozostavajúcich z IP adresy alebo doménového názvu, portu a dôveryhodného certifikátu. Informácia s presmerovaním je buď dôveryhodná alebo nedôveryhodná. Informáciu považujeme za dôveryhodnú, keď je podpísaná certifikátom vlastníka zariadenia alebo je získaná pomocou zabezpečeného spojenia k samozavádzaciemu serveru. V inakšom prípade je nedôveryhodná.

```
+--:(redirect-information)
   +--ro redirect-information
      +--ro bootstrap-server* [address]
         +--ro address inet:host
         +--ro port? inet:port-number
         +--ro trust-anchor? binary
```

Obr. 4.1: Stromový diagram štruktúry presmerovacie informácie

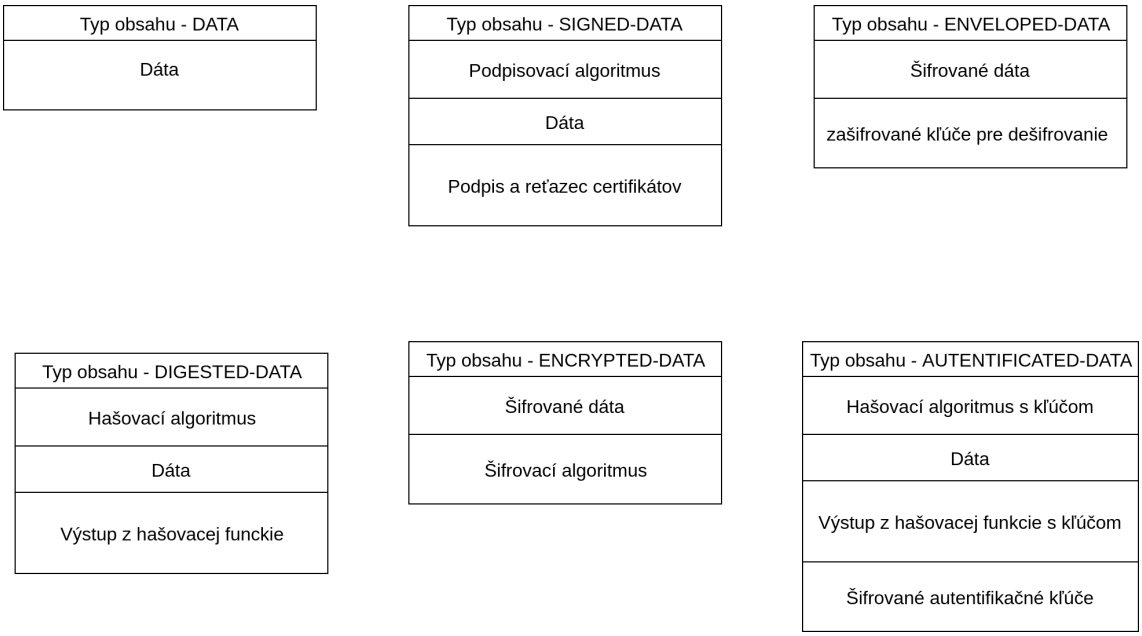
Informácia s konfiguráciou poskytuje všetky potrebné dáta, ktoré bude zariadenie potrebovať, aby fungovalo správne. Tieto dáta obsahujú informáciu o zavádzanom obraze, ktorý musí byť spustený, inicializačnú konfiguráciu a voliteľné skripty, ktoré musia byť úspešne vykonané. Na obrázku 4.2 je zobrazená štruktúra tejto informácie, ktorá je definovaná pomocou jazyka YANG. Rozhodovanie dôveryhodnosti informácie s konfiguráciou je totožné ako pri informácii s presmerovaním. V prípade nedôveryhodnej informácii musí byť ignorovaná a pokračovať v procese získavania dôveryhodnej informácie s konfiguráciou.

```
+--:(onboarding-information)
   +--ro onboarding-information
      +--ro boot-image
      | +--ro name string
      | +--ro (hash-algorithm)
      | | +--:(sha256)
      | | +--ro sha256? string
      | +--ro uri* inet:uri
      +--ro configuration-handling enumeration
      +--ro pre-configuration-script? script
      +--ro configuration?
      +--ro post-configuration-script? script
```

Obr. 4.2: Stromový diagram štruktúry informácie s konfiguráciou

Pre zaručenie bezpečnosti týchto informácii sa použije štruktúra CMS. Táto kryptografická štruktúra popisuje spôsob zapúzdrenia dát, ktorá zabezpečí ich ochranu pomocou digitálneho podpisu alebo šifrovania. CMS štruktúry sa môžu zapúzdrovať do seba, čo umožňuje lepšie chrániť citlivé dáta. Tento spôsob ochrany umožňuje zaručiť integritu dát vďaka digitálnemu podpisu a následne ho zašifrovať, čo umožní posielanie dát cez nezabezpečené spojenie. Navyše umožňuje ukladanie ďalších ľubovoľných atribútov, ktoré sú spojené s overovaním podpisu dát (napr. čas digitálneho podpisu, reťazec certifikátov vedúcich k známemu dôveryhodnému certifikátu, zoznam odvolaných certifikátov a pod.). CMS štruktúra je kódovaná v ASN.1 (*Abstract Syntax Node*), pričom hodnoty znakov sú 8 bitové. V prípade prenosu dát je vhodné prekonvertovať tieto hodnoty na ASCII znaky pomocou base64 kódovania. CMS štruktúra používa šesť typov zápuzdrenia dát, ktorých formát štruktúry je zobrazený na obrázku 4.3. Typy zapuzdrení sú nasledovné:

- **data** (ID - 1.2.840.113549.1.7.1) - je to základný typ CMS štruktúry. V obsahu tejto štruktúry sa nachádzajú užívateľské dáta. Tento typ je zapúzdrovaných do ďalších typov
- **signed-data** (ID - 1.2.840.113549.1.7.2) - tento typ je používaný na digitálne podpisovanie dát. Môže obsahovať viacero podpisov
- **enveloped-data** (ID - 1.2.840.113549.1.7.3) - tento typ obsahuje zašifrované dáta a zašifrované šifrovacie kľúče pre príjemcu
- **digested-data** (ID - 1.2.840.113549.1.7.4) - tento typ obsahuje dáta a výstup hašovacej funkcie pre overenie integrity obsahu
- **encrypted-data** (ID - 1.2.840.113549.1.7.5) - tento typ obsahuje zašifrované dáta
- **authenticated-data** (ID - 1.2.840.113549.1.7.6) - tento typ obsahuje dáta, výstup z hašovacej funkcie s kľúčom a šifrované autentifikačné kľúče




Obr. 4.3: Štruktúra jednotlivých typov zapúzdrenia

Z dôvodu použitia kryptografickej štruktury CMS je potrebné zadefinovať objekty, s ktorými bude mechanizmus Zero Touch pracovať. Tieto objekty sú zobrazené v nasledujúcom zozname:

- zero touch informácia (zero touch information),
- certifikát vlastníka (certificate owner),
- doklad vlastníctva (ownership voucher).

Objekt *zero touch informácia* obsahuje samotné samozavádzacie dáta (informácie), ktoré sú popísane vyššie. Pre nepodpísané dáta musí byť typ obsahu CMS štruktúry nastavený na hodnotu *id-data*. Samotné samozavádzacie dáta môžu byť uložené vo formáte XML

alebo JSON (*JavaScript Object Notation*), pričom voľba tohto formátu sa dohodla počas komunikácie. V prípade podpísaných dát musí byť typ obsahu CMS štruktúry nastavený na hodnotu *id-signedData*. Zároveň vnútorný typ obsahu musí odpovedať nepodpísaným dátam.

Objekt *certifikát vlastníka* je certifikát vo formáte X.509, ktorý sa používa pre overenie podpisu objektu *zero touch informácie*. Certifikát môže byť podpísaný ľubovoľnou certifikačnou autoritou. CMS štruktúra musí obsahovať samotný certifikát a všetky vedľajšie certifikáty, ktoré vedú dôveryhodnému certifikátu označovaného ako *pinned-domain-cert*, ktorý sa nachádza v doklade vlastníctva. Voliteľne môže obsahovať aj dôveryhodný certifikát *pinned-domain-cert*. Z toho vyplýva, že typ obsahu je nastavený na hodnotu *id-signedData*. Navyše, zariadenie nasadené v súkromných sieťach môže obsahovať taktiež v tejto štruktúre nové CRL (*Certification Revocation List*), v ktorom sú uložené odvolané certifikáty niektorou certifikačnou autoritou. Výhodou dostupnosti týchto informácii je, že nenúti zariadenie dynamicky sťahovať odvolané certifikáty z CRL distribučného bodu.

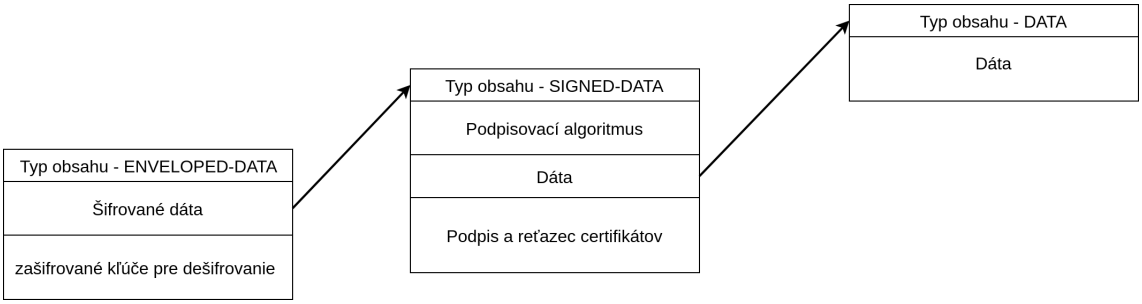
Posledný objekt *doklad vlastníctva* je použitý pre bezpečnú identifikáciu vlastníka zariadenia. Doklad vlastníctva je podpísaný výrobcom zariadenia. Využíva sa pri verifikácii certifikátu vlastníka, ktorý je získaný s dokladom vlatníctva a zero touch informácie zo zdroja samozavádzacích dát, ktoré budú popísané v sekcii 4.2. Táto verifikácia spočíva v overení reťazca dôveryhodnosti od certifikátu vlastníka až k dôveryhodnému certifikátu *pinned-domain-cert*. Táto verifikácia sa vykoná aj v prípade, keď certifikát vlastníka je podpísaný sám sebou. Tento objekt musí byť podpísaný a preto musí mať nastavený typ obsahu na hodnotu *id-signedData*. Jeho vnútorný typ obsahu odpovedá hodnote *id-data*, pričom formát obsahu je dohodnutý počas komunikácie. V prípade hodnoty odpovedajúcej *animaJSONVoucher* (1.2.840.113549.1.9.16.1) sú dáta v JSON formáte. Štruktúra dát je definovaná v RFC 8366 [11]. Na obrázku 4.4 je zobrazená táto štruktúra definovaná pomocou jazyka YANG.

```
yang-data voucher-artifact:
   +---- voucher
      +---- created-on yang:date-and-time
      +---- expires-on? yang:date-and-time
      +---- assertion enumeration
      +---- serial-number string
      +---- idevid-issuer? binary
      +---- pinned-domain-cert binary
      +---- domain-cert-revocation-checks? boolean
      +---- nonce? binary
      +---- last-renewal-date? yang:date-and-time
```

Obr. 4.4: Stromový diagram štruktúry dokladu vlastníctva

Popísané objekty sú zatiaľ odolné len proti zmene. V prípade prenosu dát, ktoré sa považujú za citlivé, je nutno zaistiť ich ochranu proti nedovolenému čítaniu. Preto CMS štruktúra podporuje aj zašifrovanie dát. Každý z 3 objektov môže byť individuálne zašifrovaný. Na zašifrovanie sa použije zvlášť certifikát zariadenia, ktorý je určený na tento účel. Ale je možné mať totožné certifikáty pre overovanie totožnosti klienta v TLS spojení a na šifrovanie dát. Zašifrované dáta musia mať typ obsahu CMS štruktúry nastavený na

hodnotu *id-envelopedData*, a jeho vnútorný typ obsahu musí odpovedať podpísaným dátam. Z toho vyplýva že dáta nemôžu byť najskôr šifrované a potom podpísané.




Obr. 4.5: Príklad zapúzdrenia CMS štruktúry

## 4.2 Zdroje samozavádzacích dát

Mechanizmus definuje niekoľko zdrojov, z ktorých môže získavať informácie, ktoré sú spomenuté v sekcii 4.1. Zoznam možných zdrojov sa v budúcnosti môže rozšíriť o ďalšie zdroje.

### Vymeniteľne zariadenie

Po pripojení vymeniteľného zariadenia (napríklad USB kľúča) sa môžu získať samozavádzacie dáta vrátane certifikátu vlastníka a dokladu vlastníctva. Tento spôsob získania samozavadzacích dát je veľmi výhodný, lebo nevyžaduje žiadnu externú infraštruktúru k svojej činnosti. Zariadenie, ktoré podporuje tento zdroj, musí vedieť detegovať pripojenie vymeniteľného zariadenia a pripojiť jeho súborový systém. Názvy a umiestnenie súborov závisí od implementácie tohto mechanizmu, pričom každý objekt musí byť uložený vo svojom vlastnom súbore. Získanie dát týmto spôsobom sa považuje za nedôveryhodné a dáta musia byť podpísané alebo obsahujú informácie s presmerovaním na iný samozavádzací server.

### DNS server

Pomocou odpovedí DNS (*Domain Name System*) môže získať samozavádzacie dáta. Výhodou tohto zdroja je ľahké nasadenie do existujúcej infraštruktúry DNS. Zariadenie môže vykonať multicastové DNS dotazy pre hľadanie služby *_zerotouch._tcp.local*. Prípadne zariadenie môže vykonať normálny DNS dotaz na DNS server, ktorý je získaný z DHCP služby. V prípade nepoužitia DNSSEC sú dáta považované za nedôveryhodné, čiže musia byť znova podpísané alebo sú to informácie s presmerovaním. Tieto dáta sú uložené v TXT záznamoch. Mapovanie objektov na záznamy TXT v DNS je nasledovné:

- zero touch informácia - mapované na *zt-info*
- certifikát vlastníka - mapované na *zt-cert*
- doklad vlastníka - mapované na *zt-voucher*

### DHCP server

Pomocou DHCP možností zariadenie môže získať samozavádzacie dáta. Využitie tohto zdroja umožňuje ľahké nasadenie do existujúcej DHCP infraštruktúry a je považovaný za nedôveryhodný. Protokol DHCP (najmä pre IPv4) je limitovaný pre prenos dát, pričom podpísané dáta by sa nezmestili do tohto limitu. Z tohto dôvodu sú prenášané len nepodpísané informácie s presmerovaním.

### Samozavádzajúci server

Samozavádzací (bootstrap) server je definovaný ako RESTCONF server. Pre komunikáciu využíva RESTCONF protokol, ktorý je popísaný v podkapitole 2.3. V prípade úspešného overenia serverového certifikátu je naviazané bezpečné spojenie, čo umožňuje považovať dáta za dôveryhodné. V inakšom prípade musia byť dáta podpísané, ak majú byť dôveryhodné. Tento spôsob získania dát môže uľahčiť nasadenie v niektorých situáciach.

Na rozdiel od ostatných zdrojov, ktoré boli spomenuté, zariadenie dokáže informovať server o postupe samozavádzacieho procesu (napr. varovania, chyby) pomocou RPC *report-progress*. Na obrázku 4.6 je zobrazená štruktúra dát RPC *report-progress* a RPC *get-bootstrapping-data*, pomocou ktorého sa získavajú samozavádzacie dáta.

```
+---x get-bootstrapping-data
| +---w input
| | +---w untrusted-connection? empty
| | +---w hw-model? string
| | +---w os-name? string
| | +---w os-version? string
| | +---w nonce? binary
| +--ro output
| +--ro zerotouch-information cms
| +--ro owner-certificate? cms
| +--ro ownership-voucher? cms
+---x report-progress
   +---w input
      +---w progress-type enumeration
      +---w message? string
      +---w ssh-host-keys
      | +---w ssh-host-key* []
      | +---w format enumeration
      | +---w key-data string
      +---w trust-anchors
         +---w trust-anchor* []
            +---w certificate cms
```

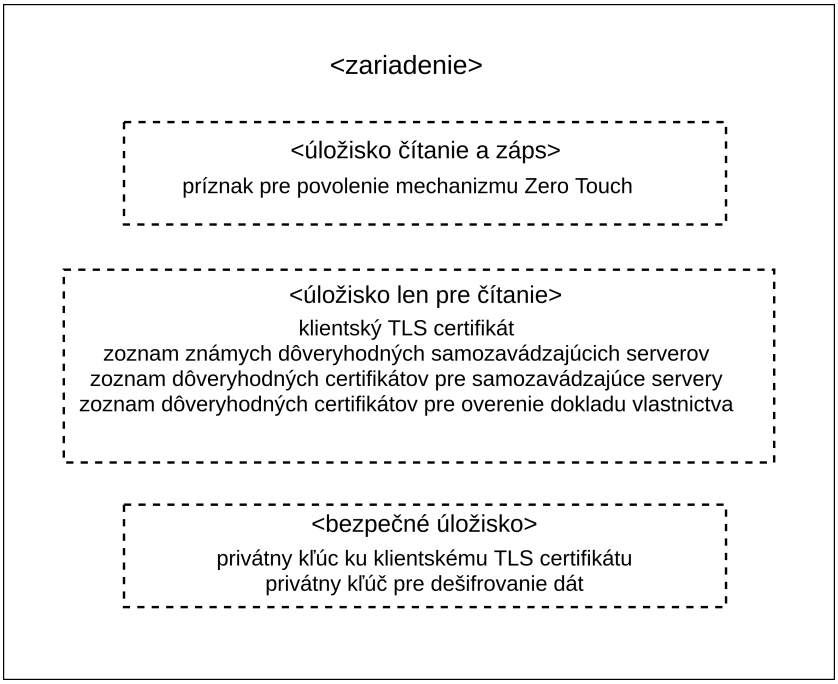
Obr. 4.6: Stromový diagram rpc štruktúr samozavádzajúceho servera

## 4.3 Architektúra zariadenia

Zariadenie podporujúce Zero Touch mechanizmus musí mať predkonfigurovaný stav, ktorý odpovedá štruktúre dát zobrazených na obrázku 4.7. Zo štruktúry dát plyne, že konfigurovateľná premenná povoľuje alebo zakazuje spustenie Zero Touch mechanizmu. Pri prvom spustení zariadenia je táto premenná nastavená na hodnotu true, ktorá odpovedá povoleniu spustenia mechanizmu. Po úspešnom nainštalovaní konfigurácie sa zmení táto hodnota na false.

Aby sa zariadenie mohlo autentizovať voči samozavádzaciemu serveru, obsahuje klientský TLS certifikát a všetky potrebné certifikáty k známej certifikačnej autorite. Zároveň slúži aj ako identifikácia zariadenia. V bezpečnom úložisku sa nachádza súkromný kľúč pre dešifrovanie samozavádzacích dát a súkromný kľúč, ktorý je využívaný pre naviazanie TLS spojenia. Pre bezpečné úložisko môže byť využitý kryptografický procesor (napr. TPM).

Ak zariadenie podporuje získanie dát zo známych samozavádzacích serverov, musí obsahovať zoznam týchto serverov a dôveryhodné certifikáty, pomocou ktorých sa overí certifikát servera pri naviazaní TLS spojenia.



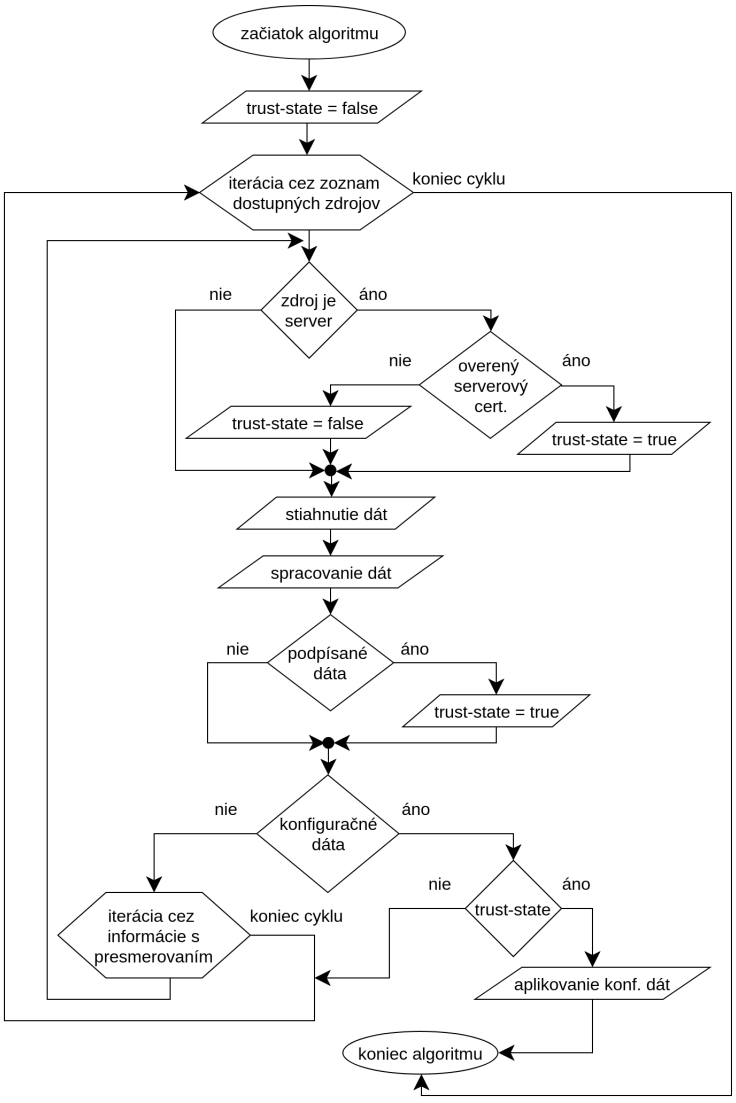

Obr. 4.7: Architektúra zariadenia podporujúca zero touch mechanizmus

## 4.4 Mechanizmus overovania a sťahovania samozavádzacích dát

Po zapnutí zariadenia sa skontroluje premenná, ktorá určuje, či sa má zariadenie naštartovať normálne alebo spustiť Zero Touch mechanizmus. V prípade spustenia Zero Touch mechanizmu sa prechádza cez zoznam podporovaných zdrojov, z ktorých sa môžu získať samozavádzacie dáta. Využíva sa princíp algoritmu prehľadávania do hĺbky. Je to z dôvodu

použitia informácií s presmerovaním. Na obrázku 4.8 je zobrazený vývojový diagram tohto algoritmu.

Algoritmus používa globálnu premennú, ktorá sa nazýva *trust-state*. Táto premenná vyjadruje aktuálny stav dôveryhodnosti dát. Premenná je inicializovaná na hodnotu false a pri spracovaní dát s konfiguráciou musí byť táto premenná nastavená na hodnotu true, inak konfiguračné dáta musia byť ignorované.




Obr. 4.8: Vývojový diagram algoritmu na sťahovanie dát

V prípade získania dát zo samozavádzacieho servera je zariadenie schopné autentizovať server pomocou jeho certifikátu. Overuje sa certifikačná cesta vedúca k dôveryhodným certifikátom, ktoré boli predkonfigurované alebo získané pomocou presmerovanej informácie. Ak je overenie úspešné, premenná *trust-state* sa nastaví na hodnotu true. V inakšom prípade sa hodnota premennej nastaví na false a zrušia sa všetky dôveryhodné certifikáty, ktoré sa zariadenie naučilo.

Ak sú získané dáta podpísané, overí sa certifikát dokladu vlastníctva. Toto overovanie prebieha pomocou predkonfigurovaných dôveryhodných certifikátov. V prípade overenia cer-

tifikátu dokladu vlastníctva sa vytiahne položka s *pinned-domain-cert* certifikátom, s ktorým sa overí certifikát vlastníka. Po úspešnom overení vlastníka sa môže overiť podpis zero touch informácie. V prípade úspešného overenia podpisu sa nastaví hodnota premennej *trust-state* na true. Ak v priebehu overovania týchto certifikátov nastane chyba, zariadenie vymaže získané dáta a pokračuje v ďalšom hľadaní vhodných dát.

Pri získaní dát s konfiguráciou musí byť nastavená premenná *trust-state* na hodnotu true. Spracujú sa konfiguračné dáta a zmení sa príznak povolenia Zero Toch mechanizmu na false. V inakšom prípade sa získané dáta vymažú a pokračuje sa v ďalšom hľadaní vhodných dát. Ak sa využijú všetky dostupné zdroje a zariadenie sa nenakonfigurovalo, môže začať algoritmus od začiatku alebo čakať na manuálnu konfiguráciu. Rozhodnutie záleží od implementácii mechanizmu.

# Kapitola 5

# Návrh implementácie

Hlavná myšlienka mechanizmu Zero Touch je počiatočná automatická konfigurácia. Preto je nutné tento mechanizmus integrovať do dátového úložiska sysrepo. Architektúra samotného sysrepa je popísaná v kapitole 3.1. Sysrepo ponúka 2 spôsoby integrácie aplikácii:

- natívna sysrepo aplikácia
- sysrepo plugin

Natívna sysrepo aplikácia využíva ako primárne dátové úložisko sysrepo. Pomocou rozhrania klientskej knižnice sysrepo aplikácia komunikuje s dátovým úložiskom. Preto musí byť linkovaná k binárnemu súboru aplikácie. Aplikácie ukladajú konfiguráciu do sysrepa, ale v mnohých prípadoch ju len načítavajú.

Aplikácia používajúca sysrepo plugin využíva pôvodný spôsob ukladania konfigurácie, pričom sysrepo môže využívať pre validáciu konfiguračných dát. Sysrepo plugin vytvára rozhranie medzi sysrepom a pôvodným spôsobom konfigurácie aplikácie. Vyžaduje synchronizáciu medzi nimi. Nevýhodou tohto prístupu je nutnosť spustenia `sysrepo-plugind(8)`, aby bolo možné komunikovať so sysrepom.

Pre integráciu mechanizmu Zero Touch nevyhovuje ani jeden ponúkaný spôsob. Pri týchto spôsoboch sa vyžaduje, aby bol spustený minimálne `sysrepod(8)`. To umožňuje akejkoľvek aplikácii čítanie alebo zápis dát do dátového úložiska. Ale mechanizmus Zero Touch má poskytnúť počiatočnú konfiguráciu daného zariadenia. Preto som sa rozhodol integrovať Zero Touch mechanizmus priamo ako komponentu sysrepa.

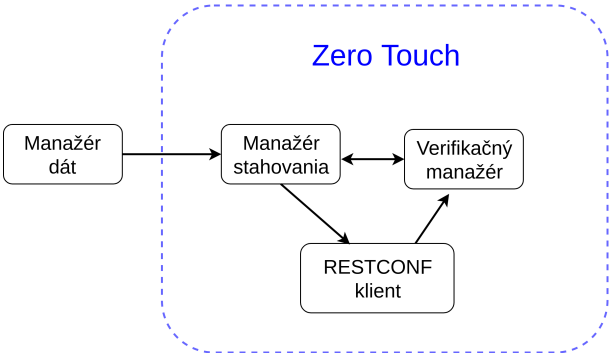
Vytvoril som komponentu Zero Touch, ktorá je zobrazená na obrázku 5.1. Táto komponenta sa spustí pri prvom spustení `sysrepod(8)`. Pomocou dátového manažéra sa získajú konfiguračné dáta pre YANG modul *zerotouch-device*. Skontroluje sa uzol *enabled*, či je povolené spustenie algoritmu na získanie samozávadzajúcich dát. Komponenta Zero Touch sa skladá z ďalších troch častí:

- manažér sťahovania
- RESTCONF klient
- verifikačný manažér

**Manažér sťahovania** sa spustí v prípade, keď je povolený mechanizmus Zero Touch. Táto komponenta riadi celý proces získavania samozavádzajúcich dát. Umožňuje sťahovať dáta z vymeniteľného zariadenia USB. Ak toto zariadenie nie je dostupné, pokračuje sa v naviazaní spojenia so známymi samozavádzajúcimi servermi. Pretože naviazanie spojenia

a stiahnutie dát bude zložitejšie z tohto zdroja, vytvoril som špeciálnu komponentu **RESTCONF klient**. Táto komponenta naviaže bezpečné spojenie, pošle požiadavok na získanie dát RESTCONF serveru a počká na odpoveď. Po úspešnom stiahnutí dát je nutné overiť ich validitu. Na tento účel sa využije posledná komponenta - verifikačný manažér.

**Verifikačný manažér** slúži najmä na vykonanie kryptografických operácii. Umožňuje validovať stiahnuté podpísané dáta, verifikovať certifikáty a poskytuje certifikáty pre bezpečné naviazanie spojenia so samozavádzajúcim serverom pomocou transportného protokolu TLS.




Obr. 5.1: Integrácia Zero Touch mechanizmu do sysrepa

Pri tomto návrhu vznikol ešte problém možného čítania a zápisu konfiguračných dát z neinicializovaného dátového úložiska. Tento prípad môže nastať, keď `sysrepod(8)` nebol ani raz spustený, tým pádom sa nespustil Zero Touch mechanizmus a nejaká aplikácia bude chcieť čítať alebo zapisovať konfiguráciu z dátového úložiska. Aby sa vyriešil tento problém, musí sa upraviť klientská knižnica sysrepa. Ak klientská knižnica sysrepa zistí, že nie je spustený `sysrepod(8)`, vytvorí si vlastnú instanciu sysrepa. V tejto instancii sa skontroluje, či už bol vykonaný mechanizmus Zero Touch. V prípade nevykonania mechanizmu si uloží tento stav a pri každom pokuse o čítanie alebo uloženie dát z dátového úložiska sa skontroluje tento stav. Ak stav reprezentuje nespustenie mechanizmu, skontroluje si znova v YANG modulu, či sa stav nezmenil. V inakšom prípade povolí čítanie alebo zápis z dátového úložiska.

## 5.1 Výber vhodnej technológie

Sysrepo je implementované v jazyku C. Z dôvodu návrhu integrácie mechanizmu Zero Touch ako komponentu, musí byť implementovaná taktiež v jazyku C. Preto je nutné nájsť vhodnú knižnicu, ktorá by vedela pracovať s kryptografickou štruktúrou CMS. Taktiež by mala zvládať prácu s X509 certifikátmi a ich overovanie. Existuje niekoľko riešení, ktoré vyhovujú zadaným podmienkam. Tieto riešenia sú zobrazené v tabuľke 5.1.

| | Open-source | Licencia | Balíček |
|---|---|---|---|
| OpenSSL | áno | BSD | áno |
| GnuTLS | áno | GPL | áno |
| WolfSSL | áno | GPL | áno - rôzne názvy balíčkov |

Tabuľka 5.1: Tabuľka zobrazujúca knižnice pre prácu s certifikátmi

Knižnica WolfSSL[1], ktorá sa v minulosti volala CyaSSL, je odľahčená verzia OpenSSL. Je zameraná na vstavané systémy. Táto knižnica nie je vyhovujúca kvôli použitej licencie. Aplikácia využívajúca túto knižnicu musí mať taktiež GPL licenciu, pričom sysrepo má licenciu Apache 2.0. Ďalšia knižnica GnuTLS[2] dokáže pracovať so štruktúrou CMS, ale len v obmedzenom režime. Vykoná všetky operácie, ktoré boli definované štandardom PKCS#7, pričom CMS štruktúra dokáže aj zašifrovať svoje dáta. Implementácia tejto knižnici to neumožňuje. Z tohto dôvodu bola zamietnutá. Zostala knižnica OpenSSLfootnotehttps://www.openssl.org/, ktorá najviac vyhovuje. Umožňuje všetky operácie, ktoré sú potrebné pre prácu s CMS štruktúrou. Navyše do projektu sysrepa sa nezavádza ďalšia závislosť, čo je veľká výhoda pri kompilácii projektu sysrepa a jeho spustení.

### OpenSSL

Projekt OpenSSL je robustný, hodne populárny open-source nástroj pre prácu s TLS a SSL (*Secure Sockets Layer*) protokolmi. Taktiež implementuje množstvo známych kryptografických algoritmov (napr. AES, RSA, DSA, eliptické krivky) a štruktúr (napr. X509 certifikáty, PKCS#7/CMS). V skutočnosti sa OpenSSL skladá z dvoch hlavných knižníc: *libcrypto* a *libssl*. Knižnica libcrypto implementuje kryptografické algoritmy a štruktúry. Umožňuje dáta šifrovať a dešifrovať, vytvárať kľúče pre symetrické a asymetrické šifrovanie, spravovať X509 certifikáty a mnoho ďalších funkcií. Knižnica libssl implementuje SSL a TLS protokoly, pričom na kryptografické operácie využíva knižnicu libcrypto. Navyše projekt OpenSSL obsahuje nástroj `openssl(1)`, ktorý využíva obidve knižnice. Vďaka tomuto nástroju je možné rýchlo a ľahko vykonať kryptografické operácie nad užívateľskými dátami (napr. šifrovať a dešifrovať dáta, vygenerovať certifikát a pod.).

Pri implementácii overovania dát vo verifikačnom manažéri sa využije API z knižnice *libcrypto*. Aktuálna verzia knižnice libcrypto je 1.1.0. Táto knižnica je celkovo rozsiahla a dokumentácia k jednotlivým funkciám je často nedostačujúca. Preto bolo potrebné vyhľadať informácie v samotnom kóde projektu OpenSSL, ktorý je dostupný na githube[3].

[1] https://www.wolfssl.com/
[2] https://www.gnutls.org/
[3] https://github.com/openssl/openssl

# Kapitola 6

# Implementácia

Táto kapitola popisuje samotnú implementáciu Zero Touch mechanizmu. Aby tento mechanizmus mohol pracovať, musela byť upravená knižnica libyang, ktorá doteraz nepodporovala dáta definované v rozšíreniach. Následne je popísaná implementácia sťahovania samozavádzacích dát, pripojenie na RESTCONF server a overenie samozavádzacích dát pomocou knižnice OpenSSL.

## 6.1 Podpora extension v dátovom strome

Knižnica libyang slúži na parsovanie a overovanie YANG modulov. Keď sa načíta úspešne YANG modul, môžu sa načítať dáta, ktoré odpovedajú YANG modulu. Tieto dáta sú vo formáte XML alebo JSON. Zero Touch mechanizmus používa rozšírenie *yang-data*. Toto rozšírenie je definované v YANG module *ietf-restconf* a využíva sa pre špecifikovanie šablóny dát. Rozšírenie *yang-data* je implementované ako plugin do knižnice libyang. Tento plugin kontroluje výskyt tohto rozšírenia v YANG module, pričom je povolený len na najvyššej úrovni. V inakšom prípade sa ignoruje.

Knižnica libyang neumožňuje parsovať dátový strom, ktorého definícia je v rozšírení. Preto som musel rozšíriť knižnicu libyang o toto parsovanie. Samotné rozšírenie *yang-data* definuje, že schéma musí začínať uzlom *container*, ktorý reprezentuje uzol na najvyššej úrovni v XML dokumente. Knižnica libyang pri parsovaní dát z pamäte alebo zo súboru vyhľadáva uzol v module podľa jeho názvu. Pri tomto vyhľadávaní môže nastať problém, lebo názov začiatočného uzlu v rozšírení *yang-data* môže byť rovnaký ako uzol na najvyššej úrovni v danom module. Preto som pridal príznak *LYD_OPT_DATA_TEMPLATE*, ktorý je vyžadovaný pri parsovaní dát rozšírenia *yang-data*. Rôzne instancie rozšírenia *yang-data* môžu obsahovať uzol s rovnakým názvom na najvyššej úrovni stromu. Funkcie na parsovanie dát (`lyd_parse_xml()` a `lyd_parse_json()`) by sa nevedeli rozhodnúť, ktorú instanciu majú použiť. Preto bolo nutné pridať názov instancie ako parameter týmto funkciám.

Pri vytváraní, mazaní alebo vyhľadávaní dátového uzla sa využíva cesta, ktorá identifikuje daný uzol. Aby bolo možné rozlíšiť uzol definovaný v rozšírení od uzla s rovnakým názvom v klasickom dátovom strome, musel som vložiť do cesty fiktívny uzol odpovedajúci názvu danej šablóny zo začiatočným znakom `#`. Funkcie, ktoré pracujú s touto cestou, dokážu odstrániť fiktívny uzol a nájsť v module správnu instanciu tohto rozšírenia. Navyše som pridal špeciálnu funkciu `lyd_new_yangdata()`, ktorá vytvorí uzol na najvyššej úrovni k danej šablóne. Na obrázku 6.1 je zobrazený príklad definície instancii rozšírenia *yang-*

*data* a odpovedajúce dáta. K týmto dátam môžeme vytvoriť XPATH výraz na uzol ip: `/test:#data/server/ip`.

```
rc:yang-data data {
  container server {
    leaf ip {
         type string;
    }
    leaf port {
      type uint16;
    }
  }
}
```

```
<server xmlns="urn:example:test">
  <ip>192.168.0.1</ip>
</server>
```

Obr. 6.1: Príklad využitia rozšírenia yang-data

## 6.2 Načítanie predkonfigurovaného stavu

Pre správnu činnosť mechanizmu Zero Touch je nutné mať predkonfigurovaný stav, ktorý bol spomenutý v kapitole 4.3. Z toho vyplýva nutnosť uloženia certifikátov a súkromných kľúčov. Pre tento účel slúži ďalší návrh štandardu *draft-ietf-netconf-keystore* [9], ktorý umožňuje ukladanie certifikátov do dátového úložiska. Súkromné kľúče väčšinou bývajú uložené v kryptografických procesoroch (napr. TMP). V prípade sysrepa sa súkromné kľúče ukladajú do špeciálneho adresára, do ktorého ma právo čítať a zapisovať len root užívateľ.

NETCONF server, ktorý je implementovaný v projekte Netopeer2, používa sysrepo plugin na získanie certifikátov a odpovedajúce kľúče k nim. V tomto prípade musí byť spustený `sysrepod(8)` a `sysrepo-plugind(8)`, ktorý načítava pluginy pre sysrepo. V prípade mechanizmu Zero Touch je `sysrepod(8)` len v inicializačnej časti a preto sa nedá využiť daný plugin. Pre získanie certifikátov sa využije funkcia `dm_get_data_info()` z manažéra dát, ktorý sprístupní dáta zo *STARTUP* dátového úložiska pre YANG modul *ietf-keystore*. Pri kompilácii projektu sysrepa je možné konfigurovať adresár pomocou premennej *KEYSTORED_KEYS_DIR*, do ktorého sa budú ukladať súkromné kľúče. Ak táto premenná nebude nastavená, použije sa prednastavená hodnota operačného systému *sysconfdir*.

Pre ukladanie súkromných kľúčov a certifikátov je využitý formát PEM. Výhodou využitia tohto formátu je base64 kódovanie, čo umožňuje zobraziť a ukladať binárne dáta pomocou tlačiteľných ASCII znakov. Taktiež tento formát využíva jazyk YANG na ukladanie binárnych dát.

Aby sa dalo pohodlne pracovať s certifikátmi a súkromným kľúčom vo verifikačnom manažéri, tak sa uložia do viacerých štruktúr. Štruktúra `zt_tls_certificate`, ktorá je zobrazená na obrázku 6.2, zoskupuje certifikáty pre bezpečné naviazanie spojenia s RESTCONF serverom. Ak sa pri naviazaní spojenia nepodarí overiť certifikát servera pomocou dôveryhodných certifikátov, musia sa zrušiť naučené certifikáty. Ďalšie certifikáty uložené v dátovom úložisku sú použité pre overovanie podpisu dokladu vlastníctva. Tieto certifikáty sú uložené v štruktúre `X509_STORE`.

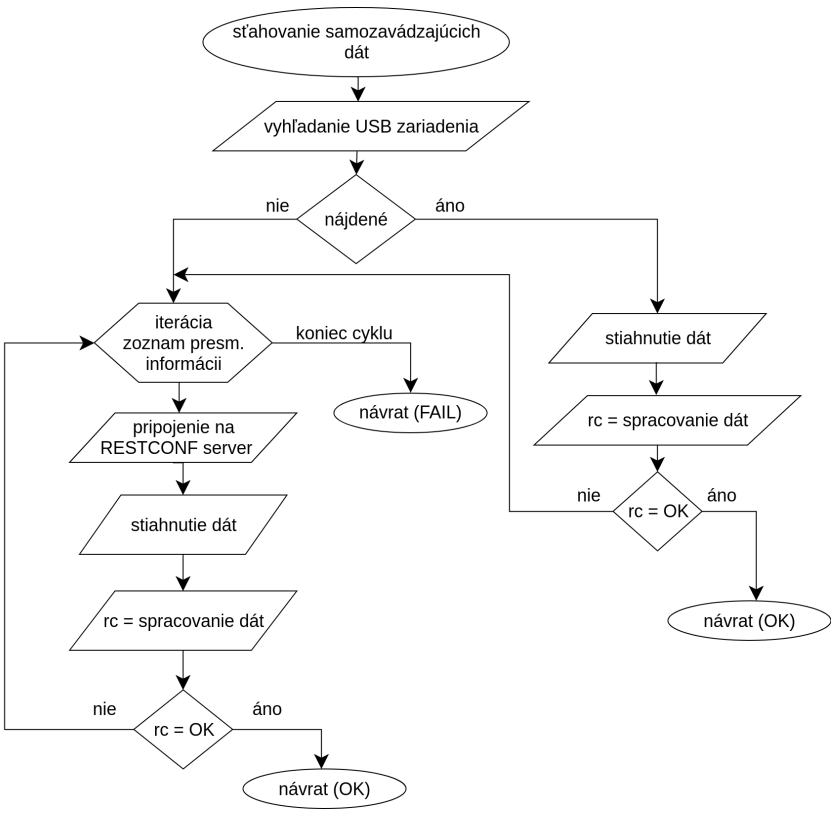
```
struct zt_tls_certificate {
    X509 *client_cert;
    EVP_PKEY *client_key;
    STACK_OF(X509) *configure_certs;
    STACK_OF(X509) *learned_certs;
    STACK_OF(X509) *client_intermediate_certs;
};
```

Obr. 6.2: Štruktúra dát pre TLS spojenie

## 6.3 Manažér sťahovania

V kapitole 4.4 je zobrazený všeobecný algoritmus pre úspešné stiahnutie samozavádzacích dát, ktorý bude implementovaný v manažéri sťahovania. Pri inicializovaní dátového manažéra instancie `sysrepod(8)` sa načítajú dáta pre YANG modul *zerotouch-device*. Z tohto modulu sa skontroluje uzol *enabled*. Ak je nastavený na hodnotu true, vykoná sa funkcia `dwn_run_zerotouch()`, ktorá obsahuje algoritmus na získavanie samozavádzacích dát. V inakšom prípade sa pokračuje v spúšťaní instancie `sysrepod(8)`.

Algoritmus začína s inicializovaním OpenSSL knižnice, ktorá sa využíva vo verifikačnom manažéri pomocou funkcie `OpenSSL_add_all_algorithms()`. Načíta sa predkonfigurovaný stav, ktorý bol popísaný v kapitole 6.2. Časť algoritmu na vyhľadávanie a stiahnutie samozavádzacích dát je zobrazený na obrázku 6.3.

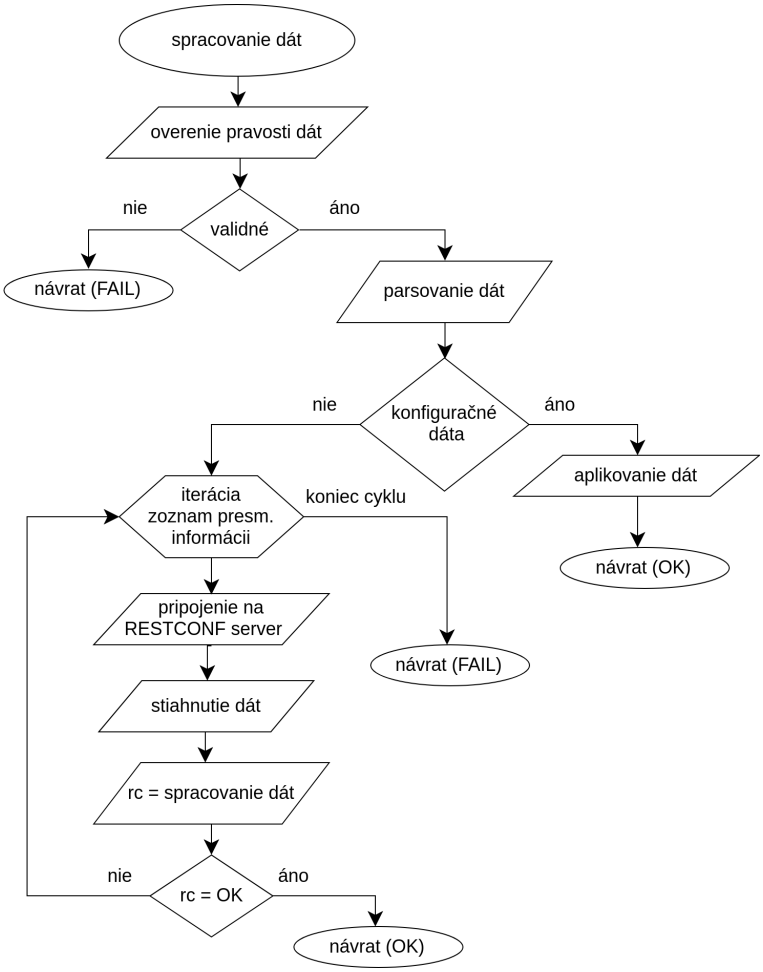



Obr. 6.3: Vývojový diagram sťahovania samozavádzacích dát

Algoritmus sa pokúsi zistiť, či je dostupné vymeniteľné USB zariadenie pomocou funkcie `is_available_usb()`. V prípade dostupnosti vymeniteľného USB zariadenia sa pripojí súborový systém, ktorý by mal byť podporovaný operačným systémom (napr. FAT32, NTFS, EXT4). V koreňovom adresári tohto zariadenia sa vyhľadávajú súbory obsahujúce samozavádzacie dáta. Každý súbor musí byť vo formáte PEM a obsahuje CMS štruktúru. Názvy súborov sú definované v nasledujúcom zozname:

- *zerotouch-information.pem* - zero touch informácia obsahujúca konfiguračné dáta v XML alebo JSON formáte
- *owner-cerrtificate.pem* - certifikát vlastníka
- *ownership-voucher.pem* - doklad vlastníctva

V prípade úspešného načítania súborov sa dáta začnú spracovávať, pričom táto časť algoritmu je zobrazená na obrázku 6.4. Pre overenie pravosti dát sa využije verifikačný manažér, ktorý je popísaný v kapitole 6.5. Po overení platnosti dát je nutné ich sparsovať pomocou funkcie `lyd_parse_mem()`, ktorá vyžaduje poznatok o typu formátu dát. V prípade naviazania spojenia s RESTCONF serverom, je možné určiť tento typ z HTTP hlavičky.




Obr. 6.4: Vývojový diagram spracovania samozavádzacích dát

Ale v prípade vymeniteľného USB zariadenia sa musí tento typ odhadnúť. Načíta sa prvý čitateľný znak a porovná sa zo znakom `<`, ktorý indikuje formát XML. V inakšom prípade budú dáta uložené v JSON formáte. Po úspešnom spracovaní dát funkciou `lyd_parse_mem()` je možné určiť typ informácie.

V prípade informácie s presmerovaním sa začne prechádzať cez zoznam samozavádzacích serverov. Každá informácia o samozavádzacom serveri obsahuje IP adresu alebo doménu. Pre preloženie domény na IP adresu som použil funkciu `gethostbyname()`, ktorá zavolá DNS resolver. Ten zistí IP adresu a vráti ju. Ak k jednej doméne odpovedá viacero IP adries, musí sa postupne prechádzať cez tento zoznam. Aby bolo možné uskutočniť pripojenie na RESTCONF server je nutné poznať port a dôveryhodný certifikát voči ktorému sa bude server overovať. Prednastavený port je 443, ktorý odpovedá službe HTTPS. Celá komunikácia s RESTCONF serverom je implementovaná v komponente RESTCONF klient, ktorá je popísaná v kapitole 6.4. Ako je vidieť z obrázku 6.4, spracovanie dát vedie k rekurzívnej funkcii.

V prípade konfiguračných dát sa skontroluje premenná *trusted-state*. Ak je nastavená na hodnotu true, začnú sa aplikovať konfiguračné dáta. Ak sa v konfiguračných dátach nachádza informácia o názve a verzii obrazu systému, musí sa najskôr skontrolovať, či sa zhoduje s aktuálne spusteným systémom. V prípade nezhody sa musí zariadenie aktualizovať. Pre tento účel sa očakáva od výrobcu zariadenia plugin, pomocou ktorého sa táto aktualizácia vykoná. Tento plugin by sa mal nachádzať pri sysrepo pluginoch, pričom jeho názov by mal byť `zerotouch_update.so`. Názov vstupnej funkcie je `zerotouch-update()`. V prípade nenájdenia pluginu sa preruší aplikovanie konfiguračných dát a pokračuje sa v hľadaní vhodných konfiguračných dát. Po úspešnom aplikovaní konfiguračných dát sa algoritmus ukončí.

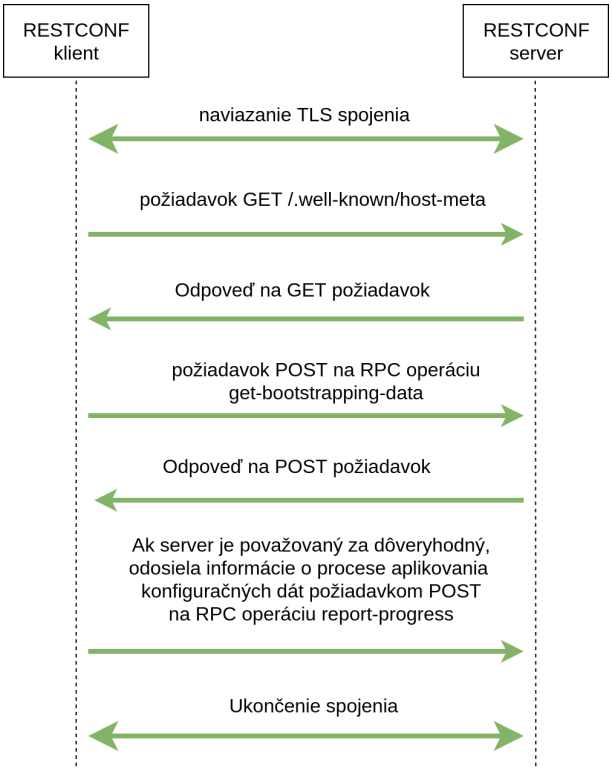
## 6.4 RESTCONF klient

Komponenta RESTCONF klient sa stará o naviazanie bezpečného spojenia s RESTCONF serverom, stiahnutie dát a prípadne posielanie notifikácii. Implementácia RESTCONF klienta musí byť v jazyku C, pretože je implementovaný v sysrepe. Bolo nutné nájsť knižnicu podporujúcu HTTP protokol. Dostupných knižníc bolo mnoho, ale vybral som si známu knižnicu *libcurl*[1]. Táto knižnica má mnoho výhod (napr. dostupný balíček, jednoduché rozhranie). Navyše podporuje verziu protokolu HTTP/2, ktorý je využívaný v testovacom serveri. Testovací RESTCONF server jetconf[2] používa iba verziu HTTP/2, preto bolo nutné, aby aj RESTCONF klient podporoval túto verziu protokolu.

Pomocou funkcie `rc_download()` sa spustí požiadavka na pripojenie RESTCONF servera. Na začiatku komunikácie sa musia nastaviť základné parametre pomocou funkcie `rc_setup()` ako sú napr. callback funkcie pre čítanie a zápis HTTP hlavičky a tela, nastavenie klientského TLS certifikátu a dôveryhodného certifikátu voči ktorému sa bude server overovať. Na obrázku 6.5 je zobrazený diagram komunikácie klienta s RESTCONF serverom.

Pri naviazaní bezpečného spojenia sa overuje serverový certifikát. V prípade úspešného overenia certifikátu sa server považuje za dôveryhodný a prijaté dáta nemusia byť podpísané. V inakšom prípade je to nedôveryhodný zdroj. Vypne sa validácia serverového certifikátu a naviaže sa znova spojenie.

[1] https://curl.haxx.se/libcurl/
[2] https://github.com/CZ-NIC/jetconf




Obr. 6.5: Diagram komunikácie RESTCONF klienta

Pošle sa požiadavok `GET /.well-known/host-meta`[7] na získanie koreňového uzlu rozhrania RESTCONF API. V odpovedi sa bude nachádzať uzol *Link* obsahujúci atribút restconf. V prípade odpovede s chybovým stavom klient zašle novú požiadavku `GET /.well-known/host-meta.json`. Ak klient znova dostane chybový stav, tak sa spojenie ukončí. Tento stav reprezentuje, že na danej IP adrese nie je dostupný RESTCONF server.

Po získaní koreňového uzlu klient zadá operáciu `get-bootstrapping-data` z YANG modulu *ietf-zerotouch-bootstrap-server* pomocou metódy POST. Keď je RESTCONF server považovaný za nedôveryhodný musí byť zaslaný vstupný parameter pre RPC v POST metóde, ktorý je zobrazený na obrázku 6.6.

```
{
   ietf-zerotouch-bootstrap-server:input : {
     untrusted-connection : [null]
   }
}
```

Obr. 6.6: Vstupný parameter RPC operácie

Klient dostane odpoveď od serveru, v ktorej sa nachádzajú dáta v textovej podobe alebo chybový stav. V prípade chybového stavu sa ukončí spojenie, inak sa spracujú dáta. Formát dát je definovaný v HTTP hlavičke odpovedi. Aby bolo možne s dátami pracovať, musia sa naparsovať pomocou knižnice libyang. Knižnica libyang nedokáže naparsovať RPC odpoveď bez odpovedajúceho RPC uzlu. Preto je nutné tento uzol vytvoriť pomo-

cou funkcie `lyd_new_path()`. Táto funkcia vyžaduje YANG modul, v ktorom sa vyhľadá RPC uzol. Naparsované dáta obsahujú CMS štruktúry v base64 formáte. Treba ich previesť pomocou funkcie `base64_to_cms()` do binárnej podoby, aby mohol s nimi pracovať verifikačný manažér.

## 6.5 Verifikačný manažér

V prechádzajúcich sekciách sa rozoberalo sťahovanie samozavádzacích dát. Aby bolo možné použiť tieto dáta je nutné ich overiť. Na tento účel slúži verifikačný manažér.

Pre začatie verifikačného procesu je nutné zavolať funkciu `vm_verify_zt_data()`, ktorá vykonáva tento proces. Na začiatku sa skontrolujú dáta, či sú šifrované. Toto overenie prebieha prečítaním typu obsahu CMS štruktúry pomocou funkcie `OBJ_obj2txt()`. Táto hodnota sa porovná s hodnotou odpovedajúcou šifrovaným dátam. V prípade rovnosti sa dešifrujú dáta pomocou funkcie `CMS_decrypt()`. Súkromný kľúč pre dešifrovanie bol načitaný z predkonfigurovaného stavu.

V prípade overenia podpísaných dát je nutné mať dostupný certifikát vlastníka a doklad vlastníctva. Ak tieto dáta nie sú dostupné, funkcia vráti chybový stav. Inak sa začne overovať podpis v doklade vlastníctva. K tomuto overovaniu je vytvorená funkcia `vm_verify_voucher()`. Overuje sa certifikačná cesta až k dôveryhodnému certifikátu, ktorý bol načítaný z predkonfigurovaného stavu. V prípade úspešnosti sa skontroluje ešte vnútorný obsah CMS štruktúry a vrátia sa dáta obsahujúce doklad vlastníctva.

Doklad vlastníctva je väčšinou vo formáte JSON. Preto je potrebné naparsovať tieto dáta pomocou funkcie `lyd_parse_mem()`. Na začiatku sa skontroluje uzol *created_on*, ktorý reprezentuje dátum vytvorenia dokladu vlastníctva. Tento dátum musí byť skorší než aktuálny. Ak sa nachádza uzol *expired_on*, musí sa skontrolovať, či doklad vlastníctva už expiroval. V prípade expirácie sa získané dáta musia zrušiť. Ďalej sa porovnáva sériové číslo zariadenia s uzlom *serial_number*, ktoré musia byť totožné. V niektorých prípadoch nemusí byť sériové číslo unikátne, preto je možné definovať identifikáciu vydavateľa IDDev certifikátu v uzle *idevid-issuer*. Táto hodnota sa musí rovnať s hodnotou v klientskom TLS certifikáte. Uzol *pinned-domain-cer* obsahuje dôveryhodný certifikát, ktorý sa využije pri overení certifikátu vlastníka. Po úspešnom overení všetkých podmienok sa môže začať spracovávať certifikát vlastníka.

Certifikát vlastníka je uložený v CMS štruktúre. Táto štruktúra môže obsahovať certifikáty, ktoré sú v certifikačnej ceste k dôveryhodnému certifikátu, ktorý bol získaný z dokladu vlastníctva. Preto je nutné získať tieto certifikáty a vložiť ich do CMS štruktúry obsahujúcej Zero Touch informáciu. To umožní overiť podpis Zero Touch informácie. Ak sú dáta overené, vrátia sa naspäť manažérovi sťahovania.

V prípade nepodpísaných dát sa celá verifikácia preskočí a vrátia sa dáta naspäť manažérovi sťahovania.

## 6.6 Inicializácia mechanizmu Zero Touch

V predchádzajúcich sekciách bola popísaná implementácia Zero Touch mechanizmu, ktorý bol integrovaný do dátového úložiska sysrepo. Aby poskytovateľ komunikačných služieb mohol využívať zariadenie podporujúce Zero Touch mechanizmus, musí nastaviť preddefinovanú konfiguráciu tomuto mechanizmu. Pomocou existujúcich nástrojov `sysrepocfg(1)` a `sysrepoctl(1)` nebolo možné nakonfigurovať tento stav. Je to z dôvodu, že požadované

dáta sú stavové a nemôžu byť uložené v dátovom úložisku. Do dátového úložiska sa uložia iba certifikáty a názvy súkromných kľúčov. Súkromné kľúče budú uložené v špeciálnom adresári, do ktorého má prístup len užívateľ root. Preto som vytvoril nástroj `sysrepozt(1)`, ktorý umožní nakonfigurovať Zero Touch mechanizmus.

`sysrepozt(1)` je konzolová aplikácia, ktorá sa ovláda pomocou prepínačov zadaných pri spustení aplikácie. Zoznam prepínačov a ich význam je zobrazený v tabuľke 6.1. Po povolení

| Prepínač | Význam prepínača |
|---|---|
| -h, --help | Zobrazenie nápovedy. |
| -v, --version | Zobrazenie verzie. |
| -r, --remove <názov certifikátu> | Vymaže certifikát s daným názvom. |
| -i, --insert <TLS\|ENCRYPT\|BOOTSTRAP> | Uloží certifikát pre daný účel. |
| --nameCert <názov certifikátu.pem> | Definuje názov súboru, z ktorého sa načíta certifikát. |
| --nameKey <názov súkromného kľúču.pem> | Definuje názov súboru, z ktorého sa načíta súkromý kľúč. |
| -s, --show | Zobrazí nakonfigurované certifikáty a ich využitie. |
| --same | Použitý v prípade rovnakého certifikátu a súkromné kľúča pre TLS a šifrovanie/dešifrovanie dát |
| -a, --add <IP alebo hostname> [port] | Pridanie nového samozavádzajúceho servera do zoznamu. |
| -d, --delete <IP alebo hostname> | Vymazanie samozavádzajúceho servera zo zoznamu. |
| -p, --print | Vymazanie samozavádzajúceho servera zo zoznamu. |
| -e, --enable | Povolenie mechanizmu Zero Touch |
| -d, --disable | Zakázanie mechanizmu Zero Touch |

Tabuľka 6.1: Dostupné prepínače nástroja sysrepozt(1)

Zero Touch mechanizmu nie sú povolené zmeny v certifikátoch, súkromných kľúčoch a ani v upravovaní zoznamu známych samozavádzacích serverov.

Pre správne fungovanie je potrebné, aby bol dostupný klientský TLS certifikát, v ktorom je uložené sériové číslo zariadenia. Sériové číslo je uložené v položke *serialNumber* X509 certifikátu. V prípade, že dáta sú šifrované, musí byť dostupný súkromný kľúč pre dešifrovanie týchto dát. Tento kľúč môže byť rovnaký ako v prípade klientského TLS certifikátu, keď sa zadá prepínač *same* pri ukladaní klientského TLS certifikátu.

Pri importovaní certifikátov je nutné zadať prepínače *nameCert* a *nameKey*. Pomocou prepínača *nameCert* sa definuje názov súboru, v ktorom je uložený certifikát. Môžu sa tu vyskytovať aj certifikáty, ktoré vedú k známemu dôveryhodnému alebo koreňovému certifikátu, ktorý je dostupný v samozavádzajúcom serveri. Tieto certifikáty musia byť zoradené od požadovaného certifikátu až po dôveryhodný alebo koreňový certifikát. Pomocou prepínača *nameKey* sa definuje názov súboru so súkromným kľúčom, ktorý odpovedá certifikátu. Obidva súbory musia byť vo formáte PEM.

# Kapitola 7

# Testovanie

Testovanie je nevyhnuteľnou súčasťou každého vývoja softvéru. Umožňuje overiť správne fungovanie a korektnosť implementácie, odhaliť množstvo chýb, ktoré by mohol programátor prehliadnuť. Preto som pri vývoji využil automatické a integračné testy. Výhoda automatických testov je, že každá komponenta môže byť testovaná zvlášť.

Existuje množstvo frameworkov, ktoré by sa dali použiť na automatické testovanie. V projekte sysrepo je využitý framework cmocka[1], ktorý použijem pre automatické testy. Vytvorené automatické testy sú integrované do testov sysrepa. Tieto testy overujú funkcionalitu jednotlivých funkcii.

## 7.1 Integračné testy

Po úspešnom otestovaní samostatných komponent pomocou automatický testov, bolo nutné ešte otestovať systém ako celok. K tomuto účelu slúžia integračné testy, ktoré pozostávajú z niekoľkých prípadov užitia. Pri týchto testoch bolo nutné mať dostupný RESTCONF server, ktorý obsahuje samozavádzacie dáta. K dispozícii je zopár implementácii RESTCONF protokolu (YumaPro, jetconf), pričom vhodnejší je **jetconf** od združenia CZ.NIC. Pretože projekt jetconf je open-source.

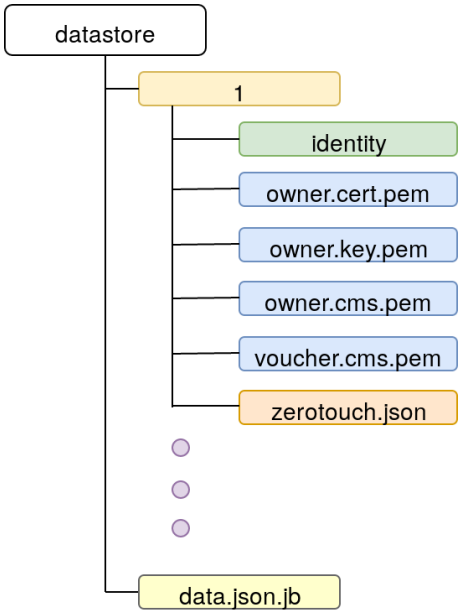
Jetconf je implementovaný v jazyku python. Užívateľská aplikácia, ktorá využíva jetconf, musí vytvoriť separátny python balíček, ktorý obsahuje minimálne 4 moduly:

- *usr_conf_data_handlers* - callback funkcie pre konfiguračné dáta
- *usr_state_data_handlers* - callback funkcie na stavové dáta
- *usr_op_handlers* - calback funkcie na RESTCONF operácie - RPC
- *usr_datastore* - inicializácia, funkcie pre načítanie a uloženie dátového úložiská

Pomocou týchto modulov je možné reagovať na zmenu konfiguračných dát, stavových dát, vykonať RPC a spravovať dátové úložisko. Pre testovacie účely Zero Touch mechanizmu postačí implementovať callback funkcie na RESTCONF operácie a inicializáciu dátového úložiska.

Jetconf používa pre autentizáciu klienta certifikát, ktorý bol poslaný pri naviazaní TLS spojenia. Z tohto certifikátu sa uloží emailová adresa. Používa sa k autorizácii užívateľa. Po autorizácii získa prístup dátam v dátovom úložisku. Emailovú adresu som taktiež využil

[1] https://cmocka.org/




Obr. 7.1: Adresárová štruktúra inicializačných dát jetconf servera

pre identifikáciu zariadenia v mechanizme Zero Touch. Vďaka nej je možné vrátiť odpoveď so správnymi dátami na zadané RPC. Na obrázku 7.1 je zobrazená navrhnutá adresárová štruktúra, ktorá je využitá pri inicializácii dátového úložiska. Certifikáty, súkromné kľúče a CMS štruktúry sú uložené v PEM formáte. Výhoda tohto spôsobu uloženia dát oproti uloženiu informácii do jedného JSON súboru je ľahká manipulácia s dátami. To umožňovalo vyskúšať rôzne prípady užitia.

Klient po úspešnom pripojení na jetconf server posiela požiadavok na vykonanie RPC *get-bootstrapping-data*. Na serveri je definovaná callback funkcia *get_bootstrapping_data_op*, ktorá vyhľadá dáta pre mechanizmus Zero Touch a vráti ich ako výsledok operácie. Ak sa klientovi nepodarilo overiť certifikát servera, musí poslať vo vstupných parametroch uzol *untrusted-connection*, ktorý reprezentuje tento stav. Server zistí, že je považovaný za nedôveryhodný zdroj, preto musí vrátiť podpísané dáta. Vyhľadá správny certifikát v dátovom úložisku a pomocou nástroja *openssl cms* sa podpíšu dáta. Ak server nemôže nájsť správny certifikát, server odošle chybový stav.

## 7.2 Testovacie prípady užitia

Pre otestovanie celého systému sa vytvorí virtuálny stroj, ktorý je spustiteľný vo VirtualBoxe. Virtuálny stroj beží na operačnom systéme Fedora 28. Obsahuje všetky závislosti, ktoré sú potrebné pre kompiláciu a beh projektu sysrepa.

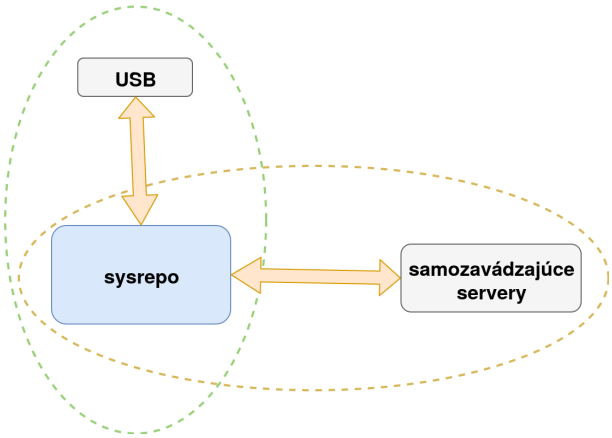
Implementácia Zero Touch mechanizmu, ktorá je popísaná v kapitole 6, dokáže získavať samozavádzacie dáta len z dvoch zdrojov: vymeniteľné zariadenie USB a samozavádzací server. S kombináciou týchto zdrojov môžeme vytvoriť 3 hlavné testovacie prípady užitia, ktoré sú zobrazené na obrázku 7.2.

Prvý prípad užitia využíva len vymeniteľné zariadenie USB. Tento zdroj je považovaný za nedôveryhodný a dáta musia byť podpísane. Na začiatku testu sa inicializuje sysrepo, kde sa nastavia certifikáty pre overenie dokladu vlastníctva. Spustí sa `sysrepod(8)`, čo naštar-

tuje mechanizmus Zero Touch. Po vykonaní Zero Touch mechanizmu sa ukončí `sysrepod(8)` a pomocou `sysrepoctl(1)` sa vyexportuje *STARTUP* dátové úložisko. Pomocou nástroja diff sa porovnajú súbory a overí sa, že Zero Touch mechanizmus prebehol úspešne.

V druhom prípade užitia sa využijú len samozavádzacie servery. Na začiatku testu sa inicializuje sysrepo. V tejto inicializácii sa vykoná nastavenie certifikátov pre TLS spojenie a certifikátu pre overovanie dokladu vlastníctva. V tomto testovacom prípade sa otestuje naviazanie na samozavádzací server, ktorý obsahuje informácie s presmerovaním na ďalší samozavádzací server, ktorý už obsahuje samozavádzacie dáta. Po skončení Zero Touch mechanizmu sa skontroluje úspešnosť testu rovnakým spôsobom ako v prvom prípade.

Tretí prípad užitia je kombinácia prvého a druhého prípadu užitia. Na vymeniteľnom zariadení USB sa budú nachádzať informácie s presmerovaním. Tento test sa spustí 2-krát, pričom v prvom raze bude vypnutý samozavádzací server, na ktorý bude odkazovať informácia s presmerovaním. V tomto prípade sa začnú načítavať dáta zo známych samozavádzacích serverov, ktoré sa pri inicializácii nakonfigurovali. V druhom raze bude už spustený samozávadzací server, z ktorého sa stiahnu samozavádzacie dáta. Overenie úspešnej konfigurácie sa vykoná rovnakým spôsobom ako v prvom prípade.




Obr. 7.2: Testovacie prípady užitia

# Kapitola 8

# Záver

Výsledkom diplomovej práce je implementácia Zero Touch mechanizmu, ktorý je zatiaľ v návrhu štandardu. Tento mechanizmus je integrovaný do dátového úložiska sysrepo a slúži na počiatočnú konfiguráciu sieťových zariadení. Hlavným cieľom mechanizmu je bezpečné nakonfigurovanie zariadenia. Klade sa dôraz na bezpečnosť a integritu dát. Počas implementácie tohto mechanizmu sa tento návrh hodne zmenil z hľadiska využitia kryptografických štruktúr. Na základe týchto zmien, nebolo možné aplikovať konfiguračné dáta do dátového úložiska. S čím súvisí neúplná implementácia nástroja pre konfigurovanie tohoto mechanizmu.

V rámci práce bola naštudovaná problematika konfigurácie sieťových zariadení pomocou protokolu NETCONF, RESTCONF, modelovacieho jazyku YANG a samotného mechanizmu Zero Touch. Taktiež bola naštudovaná problematika dátového úložiska sysrepo.

Pri návrhu integrácie mechanizmu Zero Touch som musel naštudovať architektúru dátového úložiska sysrepo. Samotnú implementáciu mechanizmu Zero Touch je možné rozdeliť do niekoľkých častí. V prvej časti bola upravená knižnica libyang, pretože neobsahovala podporu parsovania dátového stromu, ktorý bol definovaný v rozšírení. V ďalších krokoch sa upravovalo dátové úložisko sysrepo. Úpravy spočívali v implementácii algoritmu na sťahovanie samozavádzacích dát z vymeniteľného USB zariadenia alebo z RESTCONF serveru. Po úspešnom stiahnutí dát bolo nutné overiť ich validitu. Pre tento účel bola využitá knižnica OpenSSL.

Aby mohol správne fungovať Zero Touch mechanizmus je nutné ho nakonfigurovať. Táto konfigurácia spočíva v nastavení certifikátov pre overenie dát, bezpečného spojenia s RESTCONF serverom a možný zoznam známych samozavádzacích serverov.

Na základe už zmienených zmien kryptografických štruktúr nebolo možné vykonať korektné testovanie Zero Touch mechanizmu.

Cieľom ďalšej práce bude implementovať novinky kryptografických štruktúr a tým s aktualizovať už vytvorenú implementáciu. Vytvoriť testy, ktoré overia správnosť aktualizované implementácie.

# Literatúra


[1] *Sysrepo*. [cit. 2017-01-15].
URL http://www.sysrepo.org/

[2] Bider, D.; Baushke, M.: SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol. RFC 6668, RFC Editor, July 2012.
URL http://www.rfc-editor.org/rfc/rfc6668.txt

[3] Bierman, A.; Bjorklund, M.; Watsen, K.: RESTCONF Protocol. RFC 8040, RFC Editor, January 2017.

[4] Bjorklund, M.: YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). RFC 6020, RFC Editor, October 2010.
URL http://www.rfc-editor.org/rfc/rfc6020.txt

[5] Bjorklund, M.: The YANG 1.1 Data Modeling Language. RFC 7950, RFC Editor, August 2016.
URL http://www.rfc-editor.org/rfc/rfc7950.txt

[6] Chappell, C.: *Creating the Programmable Network: The Business Case for NETCONF/YANG in Network Devices*. Tech. rep. tail-f, Říjen 2003.
URL http://www.tail-f.com/wordpress/wp-content/uploads/2013/10/HR-Tail-f-NETCONF-WP-10-08-13.pdf

[7] Hammer-Lahav, E.; Cook, B.: Web Host Metadata. RFC 6415, RFC Editor, October 2011.
URL http://www.rfc-editor.org/rfc/rfc6415.txt

[8] Housley, R.: Cryptographic Message Syntax (CMS). STD 70, RFC Editor, September 2009.
URL http://www.rfc-editor.org/rfc/rfc5652.txt

[9] Watsen, K.: YANG Data Model for a "Keystore"Mechanism. Internet-Draft draft-ietf-netconf-keystore-04, IETF Secretariat, October 2017.
URL
http://www.ietf.org/internet-drafts/draft-ietf-netconf-keystore-04.txt

[10] Watsen, K.; Abrahamsson, M.; Farrer, I.: Zero Touch Provisioning for NETCONF or RESTCONF based Management. Internet-Draft draft-ietf-netconf-zerotouch-19, IETF Secretariat, October 2017.
URL
http://www.ietf.org/internet-drafts/draft-ietf-netconf-zerotouch-19.txt

[11] Watsen, K.; Richardson, M.; Pritikin, M.; aj.: A Voucher Artifact for Bootstrapping Protocols. RFC 8366, RFC Editor, May 2018.
URL http://www.rfc-editor.org/rfc/rfc8366.txt

# Príloha A

# Obsah CD

Priložené CD obsahuje:

- priečinok **text** obsahuje zdrojové súbory textu bakalárskej práce pre LaTeXvrátane obrázkov
- priečinok **source** obsahuje zdrojové súbory dátového úložiska sysrepa, knižnice libyang a python balíčku pre jetconf server