FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Chytrý strážce domácí sítě
Student:	Peter Páleník
Vedoucí:	Ing. Jiří Smítka
Studijní program:	Informatika
Studijní obor:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Proveďte rešerši existujících IDS (Intrusion Detection Systems) vhodných pro malé sítě. Porovnejte je se zaměřením na detekční schopnosti IDS, na cenu systému a na uživatelskou přívětivost. Navrhněte vlastní IDS pro uživatele s minimálními znalostmi z oboru počítačových sítí.

Navrhovaný systém by měl:
- monitorovat datové toky v lokální síti,
- umět odepřít přístup zařízením v lokální síti a v Internetu na základě vyhodnocení rizik či na žádost administrátora,
- detekovat některé známé útoky,
- být modulárně rozšiřitelný v zájmu detekce nově popsaných útoků,
- být jednoduše ovladatelný, levný a bezpečný.

Navrhovaný detekční systém nemusí být přehnaně inteligentní, musí však umožnit pozdější rozšíření v zájmu vylepšení
jeho detekčních schopností.

Navržený systém implementujte a otestujte.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 8. února 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalárska práca

Chytrý strážce domácí sítě

Peter Páleník

Katedra počítačových systémů
Vedúci práce: Ing. Jiří Smítka

15. mája 2018

Poďakovanie

V prvom rade sa chcem poďakovať Ing. Jiřímu Smítkovi za rady a pomoc pri vedení mojej práce. Ďalej ďakujem svojej manželke za pomoc a morálnu podporu pri tvorbe tejto práce. Vďaka patrí aj mojim rodičom za podporu v štúdiu.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 15. mája 2018 .....

Odkaz na túto prácu

Páleník, Peter. *Chytrý strážce domácí sítě*. Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Práca sa zaoberá návrhom a tvorbou systému, ktorý sa snaží riešiť komplexnú bezpečnosť domácej siete. Najväčší dôraz je kladený na detekciu nových (a neznámych) zariadení na sieti, detekciu útokov a hrozieb v domácej sieti a v neposlednom rade na bezpečnosť detí a na kontrolu ich prístupu k Internetu. Cieľom práce je vytvoriť systém bežiaci na platforme Raspberry Pi 3, ktorý je čo najjednoduchší na nasadenie a používanie bežným užívateľom bez znalostí z odboru počítačových sietí. Systém má byť schopný spolupracovať s domácim routrom, ku ktorému má byť pripojený jediným ethernetovým káblom. Systém pracuje v prostredí GNU/Linux, jeho jadro je Java aplikácia slúžiaca ako server, ktorý zároveň obaľuje ďalšie existujúce open-source sieťové nástroje (Nmap, Suricata, a pod.). Výsledkom je systém spĺňajúci požadované vlastnosti, ktorý bol otestovaný v ostrom prostredí domácej siete.

Klíčová slova bezpečnosť domácej siete, sieťová bezpečnosť, rodičovská kontrola, detekcia útokov, Intrusion Detection System, monitorovanie dátového toku, WiFi bezpečnosť, IoT bezpečnosť, Raspberry Pi

Abstract

This thesis is about design and creation of a system, which tries to solve complex security of a home network. Emphasis is mostly placed on detection of new (and unknown) devices on the network, detection of attacks and threats in a home network and last but not least security of children and controlling their access to the Internet. The aim of this thesis is to create a system running on Raspberry Pi 3 platform, which is easy to deploy and use by an ordinary user without knowledge in the field of computer networks. System should be able to cooperate with home router, to which it should be connected by a single ethernet cable. System runs in GNU/Linux enviroment, the core of the system is Java application serving as a server, which also encapsulates other existing open-source network tools (Nmap, Suricata, etc.). The result is a system meeting the reqirements, which has been tested in home network enviroment.

Keywords home network security, network security, parental control, attack detection, Intrusion Detection System, traffic monitoring, WiFi security, IoT security, Raspberry Pi

Obsah

Úvod

V dnešnom svete sa kladie veľký dôraz na bezpečnosť. Každý pamätá na bezpečnosť v základnom zmysle slova – zamykáme si autá, domy, byty, dávame si pozor na osobné veci v mestskej doprave, strážime naše deti aby sa im nič nestalo. Jednoducho povedané, myslíme na každú hrozbu a snažíme sa ju eliminovať. Na čo však dnešný človek zabúda je nová oblasť hrozieb, ktorá sa nám otvorila príchodom informačných technológií. Ľuďom sa nové technológie páčia a radi si ich kupujú a inštalujú do svojich domovov, aby si zjednodušili život. Chytré zámky, chytré žiarovky, chytré telefóny, chytré hračky a mnoho iných „chytrých zariadení“, často súhrnne označované ako Internet of Things, sa dnes stávajú súčasťou našich domácností, z ktorých sa stávajú chytré domácnosti. S nimi sa však stávajú naše domovy stále viac a viac náchylnejšie na kyberútoky. Čo ak niekto na diaľku ovládne chytrý zámok a otvorí si moje dvere? Čo ak niekto získa prístup k môjmu chytrému toasteru a začne ma pomocou neho odpočúvať? To sú otázky ktoré si dnes ľudia nekladú tak často. Nie sú zvyknutí premýšľať o bezpečnosti z hľadiska IT. Ďalším problémom je, že oblasť kyberbezpečnosti je tak široká, že nie je prakticky možné, aby ju bežný človek, ktorý sa touto témou nezaoberá, bol schopný obsiahnuť a stal sa „odolný“ voči kyberútokom. Preto sú dnes tak časté rôzne antivírusové programy a podobné bezpečnostné systémy – aby ľudia nemuseli premýšľať nad bezpečnosťou a mohli sa venovať svojej činnosti.

A to je dôvod, prečo vzniká táto práca, ktorá si kladie za cieľ vytvoriť bezpečnostný systém pre domácu sieť, ktorý pokrýva čo najširšiu oblasť bezpečnostných problémov hroziacich užívateľom v domácej sieti od ochrany pred malvérom, cez detekciu neznámych zariadení na sieti až po kontrolu prístupu detí k Internetu a filtrovnie obsahu.

Dnešné domáce siete sa stávajú čoraz komplexnejšie a väčšie čo do počtu zariadení. Často ani nevieme, koľko rôznych „vecí“ máme pripojených k sieti, koľko je v sieti otvorených portov, koľko sietových služieb a serverov nám doma beží (veď obýčajná sieťová tlačiareň má bežne aj 6 služieb). Bežný

človek ani nie je schopný pochopiť, čo to vôbec je sieťová služba a nie ešte aby vedel načo ktorá služba slúži a ako ju zabezpečiť. Systém má byť schopný dať užívateľovi prehľad o zariadeniach, ktoré má na sieti a poskytnúť mu o každom zariadení čo najdetailnejšie informácie, aby bol schopný posúdiť, o aké zariadenie sa jedná a či je dané zariadenie legitímne, alebo na sieti iba parazituje (napríklad nejaký sused). Zariadenia sa často pripájajú k rôznym serverom a môžu o nás posielať citlivé informácie. Systém má dať užívateľovi informácie o danom spojení a možnosť zablokovať takéto spojenie.

Ďalšou oblasťou, ktorá je v dnešnej dobe problematická, je bezpečnosť detí na Internete. Dnes už malým deťom dávame do rúk rôzne zariadenia, ktoré im umožňujú prístup na Internet. Deti sú zvedavé a Internet je obrovským zdrojom informácií a zábavy, ktorý chcú prirodzene preskúmať. Bohužiaľ si často neuvedomujeme, čo tam na ne môže číhať. Pornografia, rôzni podozriví ľudia na sociálnych sieťach či nebezpečné hry ako nedávno takzvaná „modrá veľryba“ sú iba špičkou ľadovca zo všetkých vecí, ktoré deťom na Internete hrozia. Preto sa navrhovaný systém snaží adresovať aj túto oblasť domácej bezpečnosti. Umožniť rodičom kontrolu nad časom prístupu k internetu, dať možnosť filtrovať webový obsah a kontrolovať históriu surfovania na webe a to aj vrámci šifrovaného protokolu HTTPS a v neposlednom rade notifikovať rodiča o rôznych podozrivých aktivitách ale aj o aktivitách ako príchod dieťaťa domov a podobne.

Práca sa venuje vysvetleniu problematiky bezpečnosti domácej siete, popisuje možné hrozby a typy útokov a predkladá existujúce bezpečnostné riešenia a systémy na detekciu a prevenciu útokov. Nakoniec sa venuje návrhu a implementácií popísaného bezpečnostného systému.

Kapitola 1

Cieľ práce

Táto práca si kladie za cieľ vytvoriť bezpečnostný systém pre domácu sieť, ktorý čo možno najlepšie pokrýva bezpečnostné problémy v domácej sieti. Systém má byť schopný monitorovať dátové toky všetkých (alebo vybraných) zariadení v lokálnej sieti, ku ktorej (k domácemu routru) sa pripojí pomocou jediného ethernetového rozhrania. Systém má byť čo najjednoduchší na nasadenie bežným človekom a zároveň modulárne rozšíriteľný na detekciu novo popísaných útokov. Hlavnými črtami systému má byť:

- Detekcia nových zariadení na sieti, ich klasifikácia a čo najobsiahlejší popis pre užívateľa.
- Detekcia čo možno najviac útokov hroziacich v domácej sieti.
- Kontrola prístupu zariadení zo siete do Internetu a z Internetu do siete.
- Monitorovanie sieťových spojení a možnosť dané spojenie ukončiť.
- Notifikovanie užívateľa pri rôznych udalostich (nové zariadenie na sieti, prebiehajúci útok a pod.).
- Rodičovská kontrola nad prístupom detí k Internetu – konkrétne filtrovanie webového obsahu.
- Nízka cena.

Kapitola 2

Analýza problematiky

2.1 Prehľad bezpečnostných hrozieb v domácej sieti

Domáce siete sa od korporátnych sietí líšia v značnej miere. Sú väčšinou oveľa menšie, obsahujú menší počet zariadení, často obsahujú iba jediný sieťový prvok – domáci router – ktorý je kombináciou switcha, routra a access pointu. Tak isto aj útoky, ktoré v domácej sieti hrozia, sú často iného rázu. V tejto sekcií sú vymenované nielen útoky ale aj iné bezpečnostné hrozby, ktoré sa najviac týkajú domácich sietí.

2.1.1 Zvnútra siete

Odpočúvanie Za predpokladu, že je útočník pripojený k sieti obete, môže použiť rôzne metódy aby boli pakety preposielané cez jeho zariadenie. To sa nazýva Man-In-The-Middle (MITM) útok. Vďaka tomu, že pakety prechádzajú jeho zariadením, môže komunikáciu monitorovať a sledovať obsah protokolov, ktoré nie sú šifrované.

ARP Spoofing Tiež zvaný ARP[1] cache poisoning, je typ útoku, ktorým útočník modifikuje položku ARP tabulky zariadenia v sieti pomocou falošných ARP odpovedí, ktoré pošle danému zariadeniu. Tie prepíšu skutočné mapovanie IP adresy na MAC[2] adresu. To môže útočník využiť tak, že namapuje IP adresu DG[3] na svoju MAC adresu. Ak bude následne zariadenie chcieť komunikovať smerom do Internetu cez DG, bude vskutočnosti komunikovať s počítačom útočníka. Ten môže následne komunikáciu predávať na DG, čím si obeť nič nevšimne, ale úťočník môže odpočúvať všetku komunikáciu v jednom

[1] Address Resolution Protocol
[2] Media Access Control
[3] Default Gateway

smere. Ak chce útočník odpočúvať komunikáciu aj v opačnom smere, môže rovnakým spôsobom „oklamať“ DG, ktorá bude následne všetky prichádzajúce pakety posielať na útočníkov počítač.[1, 2]

Falošný DHCP server DHCP[4] server má na sieti za úlohu pridelovať zariadeniam IP adresy a podať im základné informácie o sieti aby mohli komunikovať do Internetu. Medzi tieto informácie patrí aj adresa DG a DNS[5] servera. Ak útočník spustí svoj vlastný DHCP server, ktorý označí ako DG sám seba a podarí sa mu, aby jeho server zvíťazil nad legitímnym serverom, je schopný zachytávať pakety v smere od obete do Internetu (avšak nie naspať). Ak by chcel zachytávať aj prichádzajúce pakety, môže zmeniť nastavenia svojho DHCP servera tak, že vytvorí novú podsieť v existujúcej sieti s inými adresami, a následne spustí NAT. Tým dosiahne to, že keď bude obeť komunikovať s Internetom, bude jej adresa preložená útočníkovým zariadením a pri komunikácií s legitímnou DG bude vystupovať iba adresa útočníka avšaknie adresa obete. Tým bude legitímna GW nútená posielať pakety späť na adresu útočníka, ktorý ich následne prepošle obeti. Tento typ útoku predpokladá vyradenie legitímneho DHCP servera. To sa dá dosiahnuť napríklad pomocou takzvaného DHCP starvation útoku, ktorý spočíva v tom, že útočník „vyhladuje“ DHCP server – vyčerpá všetky adresy ktoré je server schopný priradiť. Ůtočník začne generovať DHCP žiadosti s falošnými MAC adresami a ak server nie je príliš podozrievavý, začne tieto adresy prideľovať z domnienkou, že ich prideluje reálnym zariadeniam. Keďže čas, za ktorý DHCP server začne adresu znova považovať za voľnú býva často rádovo v desiatkach minút, hodinách až dňoch, môže tento útok zraziť DHCP server na kolená na pomerne dlhý čas. Útočník môže náasledne útok znova opakovať, takže je schopný permanentne držať DHCP server hladný.[3]

Úprava nastavení routru Drvivá väčšina dnešných domácich routrov je konfigurovateľná cez webové rozhranie. Prístup však často býva podmienený zadaním správneho mena a hesla. Mnoho užívateľov však tieto hodnoty necháva v „továrenskom“ stave, takže sa môžeme často stretnúť s užívateľským menom *admin* a heslom *admin* atp. Znamená to, že ktokoľvek na sieti (kto je dosť chytrý na to, aby uhádol meno a heslo) môže pristupovať k nastaveniam routru a meniť ich. To predstavuje riziko, pretože útočník je schopný napríklad otvárať a presmerovať porty, zmeniť nastavenia DG DHCP servera, či úplne ovládnuť sieť zmenou SSID[6] a hesla.

Prelomenie hesla do systému Útočník sa môže pokúsiť „uhádnuť“ heslo nejakého systému v domácej sieti, ku ktorému nemá oprávnenie. Môže to

[4]Dynamic Host Configuration Protocol
[5]Domain Name System
[6]Service Set Identifier – inak povedané názov WiFi siete

byť napríklad NAS server, konfiguračné rozhranie domáceho routru, či iného zariadenia v domácej sieti. Technika, ktorú na to môže použiť môže byť hrubá sila (skúšanie všetkých permutácií znakových reťazcov) alebo slovníkový útok (skúšanie hesiel z predpripraveného zoznamu hesiel – slovníka).

2.1.2 Zvonka siete

Otvorené porty – neoprávnený vzdialený prístup Mnohé dnešné routre podporujú takzvaný UPnP port mapping čo je funkcia ktorá umožňuje zariadeniam v sieti otvárať vonkajšie porty routra a presmerovávať ich na vybrané zariadenia. Taktiež sa môže stať, že užívateľ má doma nejaké zariadenie, ku ktorému chce vzdialene pristupovať tak otvorí port, cez ktorý služba na danom zariadení komunikuje, avšak z neznalosti toto zariadenie nezabezpečí, a tak môže každý z Internetu komunikovať s týmto zariadením. Toto môže byť veľmi nebezpečné v prípade rôznych IP kamier alebo zdieľaných adresárov. Kamera obete sa tak ľahko môže dostať na zoznam ako `http://www.insecam.org`.

Botnety Užívateľ si môže stiahnuť stiahnuť software, ktorý je trójskym koňom obsahujúcim botnetového klienta a jeho zariadenie sa stane súčasťou botnetu. Jeho zariadenie je potom ovládané na diaľku z Command and Control servera, ktorý mu dáva príkazy. Zariadene tak môže nechcene vykonávať DDoS útoky, rozposielať spam či robiť iné nekalé činnosti. To, že je zariadenie súčasťou botnetu sa môže prejaviť tým, že začne komunikovať s obskúrnymi adresami na zvláštnych portoch, začne generovať neočakávané dátové toky pri vykonávaní DoS útoku, alebo začne vysielať DNS dotazy na zvláštne domény.

Spyware Ďalšou hrozbou pre užívateľov je spyware. Jedná sa o malware, ktorého cieľom je získať rôzne informácie o užívateľovi, ktoré potom môžu byť použité na cielenú reklamu, predaj tretím stranám a pod. Či už sa jedná o trójskeho koňa alebo o software predinštalovaný pochybným výrobcom zariadenia, užívateľom hrozí únik citlivých informácií, ktoré môžu byť odosielane na pochybné servery, často umiestnené v krajinách s nie príliš demokratickým systémom.

2.1.3 Hrozby pre bezdrôtové siete

Prelomenie hesla a neoprávnený prístup Ak má užívateľ slabé alebo dokonca žiadne heslo na svojej bezdrôtovej sieti, môžu sa k jeho sieti pripájať zariadenia, ktoré tam nemajú čo hľadať, a môžu zneužívať a zahltiť jeho internetové pripojenie. Čo je však horšie, útočník môže odpočúvať dátové toky zariadení v sieti napríklad pomocou ARP spoofingu, pristupovať k zdieľaným adresárom a iným zdrojom a zariadeniam na sieti.

Dnešné siete používajú protokol WPA2 ktorý poskytuje dostatočné zabezpečenie a šifrovanie dát v sieti (ak zariadenia nie sú zraniteľné voči KRACK

útoku). Najslabším článkom býva často heslo, ktoré si užívateľ zvolí. Pri prelamovaní WPA hesla sa útočník snaží zachytiť tzv. WPA handshake, čo je prvotná komunikácia medzi AP a klientom, kde si dohodnú kľúče, ktorými budú šifrovať dáta. V tejto úvodnej podobe sa v istej forme nachádza aj zahashované heslo. Útočník potom môže skúšať hádať heslo a porovnávať ho sozískaným hashom. Ak teda užívateľ používa jednoduché slovníkové heslo, môže byť prelomenie otázkou pár minút. Aby útočník získal WPA handshake, musí zachytiť komunikáciu medzi zariadením a AP počas pripájania. To môže trvať dlho kým sa pripojí zariadenie, a preto môže útočník použiť takzvané deautentikačné rámce ktoré odošle klientovi a AP a tieto zariadenia na základe protokolu 802.11 ukončia komunikáciu. To spôsobí odpojenie zariadenia zo siete. Väčšina zariadení má funkciu automatického pripojenia, takže sa zariadenie pravdepodobne pokúsi znovu pripojiť. Tým útočník získa handshake a môže zahájiť samotný proces prelamovania hesla.

Evil Twin útok Tento typ útoku spočíva vo vytvorení AP s rovnakým SSID ako pôvodný AP. Pri nezabezpečenej sieti môžeme týmto spôsobom ľahko donútiť obeť pripojiť sa k nášmu AP a to napríklad tak že obeť zahltíme deautentikačnými rámcami a pri troche šťastia a dostatočne silnom signále a ak zahltíme pôvodný AP, obeť sa pripojí k nášmu AP a my môžeme odpočúvať jeho komunikáciu. Zabezpečený AP to útočníkovi skomplikuje, pretože ak nevie heslo, nedokáže plne sklonovať AP a zariadenia sa k tomu jeho nebudú automaticky pripájať. Útočník však môže použiť nástroj ako `wifiphisher`, ktorý vytvorí nezabezpečený Evil Twin AP a zahltí legitímny AP. Obete sa postupne začnú pripájať k falošnému AP, kde regulérne dostanú IP adresu od DHCP servera a pri pokuse o prístup na Web sa zobrazí captive portál, ktorý sa od obete bude snažiť vymámiť heslo od legitímneho AP so zámienkou aktualizície firmware alebo podobnou technikou sociálneho inžinierstva.[4]

2.1.4 Hrozby pre deti

Nejedná sa síce o klasickú oblasť IT bezpečnosti, avšak dovolím si tvrdiť, že v tomto kontexte stojí za zmienku. Deti dnes pristupujú k Internetu už od útleho veku a domáca sieť je pravdepodobne najčastejším miestom, kde sa tak deje. Preto by mala byť dostatočne bezpečná na to, aby deti boli schopné zdravého vývoja.

Internet je skvelý informačný zdroj a prostriedok pre tých, ktorí ho vedia používať. Bohužiaľ, deti toho ešte nie sú schopné, a tak sa ľahko stane, že namiesto toho Internet ovláda ich. Môžu tak stratiť zmysel pre realitu a stať sa na Internete „závislé“. To môže ďalej viesť k mnoho psychickým problémom ako napríklad nízke sebavedomie, samota, depresia, ... [5]

Bohužiaľ, najjednoduchšie riešenie – odoprieť im prístup k Internetu – nemusí automaticky fungovať[5]. Rovnako ani špehovať ich aktivitu na Internete

nemusí byť riešením, pretože sa deti môžu rodičom odcudziť. Je nutné hľadať strednú cestu a hlavne otvorený dialóg s deťmi. [6].

Veľkým problémom je taktiež pornografia a podobný obsah na Internete. Keďže deti zatiaľ nerozumejú ľudskej sexualite a jej zmyslu, môže na nich vystavenie takémuto materiálu zachovať neblahé následky. Môže to ovplyvniť ich pohľad na opačné pohlavie, na vzťahy a hodnoty a rovnako sa môže zhoršiť ich správanie. [7, 6]

2.2 Prehľad existujúcich bezpečnostných riešení

V tejto sekcií sú prebrané existujúce riešenia, ktoré sa snažia adresovať problém bezpečnosti zariadení v sieti. Zameriavam sa na systémy, ktoré riešia bezpečnosť všetkých zariadení na sieti a nevyžadújú si inštaláciu na tieto zariadenia.

2.2.1 Riešenia pre domáce siete

2.2.1.1 Komerčné riešenia

Rattrap Je to zariadenie sediace medzi routrom a modemom, takže mimo domácej siete za NAT[7]. Je to akýsi chytrý firewall ktorý analyzuje hlavičky paketov (nie samotné dáta) a na základe toho vykonáva potrebné opatrenia. Používa cloudové ML techniky na vylepšenie svojich detekčných schopností. Je jednoduchý na inštaláciu, zapojí sa jedným káblom do modemu a druhým do routru. Disponuje dvoma Gigabitovými ethernetovými portmi a je schopný pracovať s prenosovou rýchlosťou 150-180 Mbps. Kvôli tomu, že sa zariadenie nachádza mimo domácu sieť, nie je schopné odlíšiť dátové toky jednotlivých zariadení. Nerieši ani bezdrôtovú bezpečnosť a nie je schopné detekovať útoky vo vnútri siete, keďže sa nachádza mimo sieť. Cena zariadenia je 199.00 $. Na to, aby malo plnú funkcionalitu je nutné mesačne platiť 9 $ s tým, že prvých 12 mesiacov je zdarma. [8, 9]

Cujo IDS[8]/IPS[9]/Firewall riešenie zamerané na neskúsených užívateľov, chová sa ako MITM[10] voči všetkým zariadeniam komunikujúcim s routrom. Od užívateľa si vyžiada, aby vypol DHCP[11] server na svojom routri a následne spustí vlastný DHCP server, ktorým ovládne sieť. Monitoruje dátové toky jednotlivých zariadení a snaží sa v nich odhaliť možné útoky alebo hrozby. Zároveň získané dáta posiela na cloud, kde sú ďalej analyzované a zároveň sa pomocou nich rozširuje databáza hrozieb, čím sa neustále vylepšuje detekčná

[7] Network Address Translation
[8] Intrusion Detection System
[9] Intrusion Prevention System
[10] Man In The Middle
[11] Dynamic Host Configuration Protocol

schopnosť. Zariadenie je ovládané a komunikuje s užívateľom cez mobilnú aplikáciu. Taktiež disponuje schopnosťou rodičovskej kontroly, nerieši však bezpečnosť bezdrôtovej siete. Cena zariadenia je 249 $ bez nutnosti mesačných platieb za cloudové služby. [10, 11]

Dojo IDS riešenie, ktoré sa snaží detekovať anomálie v sieti. Najprv sa „učí" tým, že sleduje bežnú sieťovú prevádzku voči ktorej následne zisťuje anomálie a notifikuje používateľa pomocou mobilnej aplikácie. Tiež ovláda sieť pomocou vlastného DHCP servera. Disponuje aj cloudovou zložkou, ktorá analyzuje dáta pomocou ML[12]. Zariadenie nerieši bezdrôtovú bezpečnosť, ani rodičovskú kontrolu. Cena zariadenia je 129.99 $. Na to, aby malo plnú funkcionalitu je nutné mesačne platiť 9.99 $ s tým, že prvých 12 mesiacov je zdarma. [12]

FingBox Fingbox je narozdiel od predchádzajúcich zariadení zameraný najmä na detekciu útokov z vnútra siete, najmä detekcia neznámych zariadení na sieti a zameriava sa aj na neznáme bezdrôtové zariadenia v okolí siete[13]. Je schopný detekovať rôzne klientské zariadenia, ale aj iné access pointy a tak odhaľovať útoky typu Evil Twin. Narozdiel od predchádzajúcich riešení nepresmerováva všetok sieťový tok cez seba aby ho analyzoval, preto neznižuje rýchlosť pripojenia. Namiesto toho používa ARP spoofing, ktorým selektívne presmerováva toky vybraných zariadení cez seba[14]. Touto technikou tiež môže blokovať jednotlivých užívateľov od internetu, takže môže podporovať aj rodičovskú kontrolu. Cena je 129 $.

KoalaSafe Toto zariadenie je čiste zamerané na rodičovskú kontrolu. Funguje ako samostatný WiFi router popri primárnom domácom routri. Deti sa k nemu pripoja a sú pod kontrolou. Zariadenie je ovládané pomocou mobilnej aplikácie, kde sa dajú nastaviť rôzne obmedzenia, sledovať aktivitu detí či blokovať nechcené stránky. Zariadenie však nie je schopné využívať existujúcu sieťovú infraštruktúru a spolieha sa na vysielanie vlastného signálu, ktorý všk nemusí byť dostatočne silný na to, aby pokryl celú domácnosť. To môže byť však aj výhodou, keďže zariadenie si môže rodina vziať so sebou na dovolenku a mať bezpečnosť pod kontrolou kdekoľvek sú avšak nie komplexnú bezpečnosť. Cena zariadenia je 99.99 $.[15]

2.2.1.2 Opensource riešenia

FalconGate Jedná sa o opensource systém pre bezpečnosť domácej siete, zamerané hlavne na beh na zariadení Raspberry Pi. V jeho jadre sa nachádza viacero opensource nástrojov ako Nmap či Bro. Zariadenie prevezme kontrolu nad sieťou tak, že spustí vlastný DHCP server a presmerováva dátový tok cez seba, ktorý následne analyzuje. Má aj cloudovú časť, ktorá spravuje databázu

[12]Machine Learning

malvéru. Systém sa zameriava najmä na detekciu malvéru, ale aj na blokovanie nežiadúcich reklám, detekciu zariadení na sieti, šifrovanie DNS dotazov či detekciu defaultných hesiel zariadení v sieti. Je ovládaný pomocou webového rozhrania. Systém však napríklad vôbec nerieši bezpečnosť bezdrôtovej siete a rodičovskú kontrolu.[16]

2.2.2 IDS/IPS systémy pre všeobecné použitie

Snort Je to jeden z najúspešnejších a najviac overených IDS systémov. Vykonáva realtime analýzu paketov v sieti vrátane ich hlavičiek a obsahu a tak je schopný odhaliť mnohé útoky alebo nekalú komunikáciu na sieti. Analýza je vykonávaná na základe zoznamu pravidiel (signatúr), ktoré si užívateľ môže stiahnuť alebo si ich sám napísať. Výhodou je široká komunita a to, že sa jedná o rokmi overený systém. Nevýhodou však je, že používa jedno vlákno, a tak nie je schopný využiť viac jadier procesora. ďalšou nevýhodou je, že nedetekuje protokoly na neštandardných portoch[17, 18]

Suricata Je to systém podobný Snortu schopný pracovať dokonca s rovnakými pravidlami, avšak engine na ktorom beží je úplne iný a podporuje mnohé ďalšie funkcie, ktoré Snort nepodporuje ako napríklad multi-threading, hardwarová akcelerácia pomocou grafických kariet či extrakcia súborov z http komunikácie. Je to omnoho novší systém oproti Snortu. Výhodou je spomínaný multithreading, ktorý umožnuje programu využiť všetky jadrá procesoru. Ďalej je schopný detekcie základných protokolov na neštandardných portoch, a vo veľkej miere je kompatibilný so Snortom, či už v pravidlách alebo vo formáte výstupu. Nevýhodou je vyššie využívanie RAM a menšia komunita. [18, 19]

Bro Jedná systém ktorý nie je úplne klasickým IDS systémom pracujúcim len na základe signatúr. Je obohatený o detekciu anomálií v dátovom toku a behaviorálnu analýzu. Navyše je to plne rozšíriteľný systém, ktorý poskytuje užívateľovi mocný skriptovací jazyk, vďaka ktorému si môže užívateľ nadefinovať vlastné úlohy a kontroly. Nevýhodou je však ťažšia konfigurácia.[20, 18]

Kapitola 3

Návrhované riešenie

Keďže žiadne existujúce riešenie nespĺňa požadované vlastnosti – nerieši bezpečnosť domácej siete v plnom rozsahu (od bezpečnosti detí, cez nežiadúce zariadenia na sieti až po útoky z Internetu) – navrhujem v tejto kapitole systém, ktorý tieto vlastnosti bude spĺňať.

V tejto kapitole sú rozobrané požiadavky na systém a navrhované riešenia, ktoré sú odpoveďou na dané požiadavky. Následne sa venuje výberu konkrétnych riešení a technológií, a napokon predstavuje architektúru navrhovaného systému.

3.1 Analýza požadovaných vlastností

V nasledujúcej sekcí sú rozobrané požiadavky na systém a sú navrhnuté spôsoby a možnosti riešenia.

3.1.1 Schopnosť monitorovať dátové toky všetkých zariadení na sieti

Základnou požadovanou schopnosťou systému je monitorovanie dátových tokov všetkých zariadení na sieti za účelom ich analýzy. To vyžaduje, aby všetky zariadenia preposielali pri komunikácií s Internetom svoje pakety cez náš systém a zároveň aby pakety, ktoré sa vracajú naspäť, prechádzali tiež našim zariadením.

Navrhované riešenie:

ARP spoofing ARP spoofing je technika, ktorou útočník môže vykonať MITM útok na obeť a sledovať je dátový tok. My ju však môžme využiť, aby sme legitímne monitorovali dátové toky všetkých alebo vybraných zariadení na sieti. Výhodou je, že z hľadiska siete nevyžaduje žiadne dodatočné nastavenia, stačí nám jediné sieťové rozhranie, nachádzame

sa vo vnútri siete, nevýhodou však je zaplavovanie siete ARP paketmi (je nutné „oklamať“ aj dané zariadenie a aj router), čo môže byť zachytávané rôznymi antivírusovými programamai, ktoré môžu byť voči tomuto typu útoko odolné, a tak nezískame kontrolu nad všetkými zariadeniami.

Umiestnenie systému medzi ISP[13] a domáci router Zo sieťového pohľadu je toto riešenie jedno z najčistejších. Nevytvára žiadnu nadbytočnú komunikáciu a nenarušuje existujúcu logickú architektúru. Zariadenie je umiestnené mimo sieť, medzi router a modem, kde monitoruje komunikáciu. Výhodou je čistota riešenia a jednoduchosť nasadenia, nevýhodami je, že naše zariadenie musí byť vybavené dvoma sieťovými rozhraniami (alebo byť pripojené k zariadeniu, ktoré duplikuje dátový tok a preposiela ho nášmu zariadeniu, napríklad rozbočovač) a taktiež naše zariadenie sa nenachádza vo vnútri siete, a tak nie je schopné rozlíšiť dátové toky jednotlivých zariadení. Problémom môže byť taktiež spojený modem a router do jedného zariadenia, kde sa nedokážeme medzi ne „dostať“.

Náhrada domáceho routru Toto riešenie je asi najčistejšie a najelegantnejšie, ale má najväčšie HW požiadavky na naše zariadenie. Je to výhodné z toho hladiska, že dátové toky idú automaticky cez naše zariadenie, a nie je nutné ich žiadnym spôsobom presmerovávať. Avšak naše zariadenie by muselo plnohodnotne nahradiť domáci router: malo by obsahovať dostatočne silné WiFi rozhranie, aby pokrylo domácnosť podobne, ako domáci router, ďalej dostatočný počet ethernetových rozhraní a napokon by malo implementovať všetky funkcie, ktoré podporuje domáci router.

Vytvorenie vlastného DHCP serveru Ak na sieti vypneme pôvodný DHCP server a spustíme vlastný DHCP server, sme schopný distribuovať DHCP odpovede v ktorých bude položka Gateway IP address vyplnená IP adresou nášho zariadenia. Tým pádom budú zariadenia posielať všetky dáta namiesto pôvodnému routru nám a my ich ďalej prepošleme pôvodnému routru. Problémom však je, že odpoveď už nepôjde cez nás, ale priamo ku komunikujúcemu zariadeniu, pretože sieť zariadenia je v smerovacej tabuľke pôvodného routru označená ako priamo pripojená. To však môžme vyriešiť tak, že vytvoríme vlastnú podsieť s úplne odlišným IP rozsahom ako je pôvodná sieť, a využijeme preklad adries (NAT) z nášho rozsahu na adresu nášho zariadenia z pôvodného rozsahu. Tým donútime pôvodný router, aby dáta preposielal nám a my ich následne prepošleme komunikujúcemu zariadeniu. Výhodou je nízke zaťaženie siete prebytočnými paketmi, prítomnosť nášho zariadenia vo vnútri siete, postačuje jediné sieťové rozhranie, nevýhody sú nutnosť prekladu adries, nutnosť vytvorenia nového rozsahu v rovnakom

sieťovom segmente a nutnosť vypnúť pôvodný DHCP server domáceho routru, čo môže byť náročnejšie pre neskúseného človeka. Teoreticky by sa tento problém dal riešiť útokom DHCP starvation, ktorým by sme mohli udržať DHCP server routru „hladný“ – bez IP adries, ktoré by mohol prideliť. Toto by však malo za následok zbytočnú záťaž siete DHCP komunikáciou, avšak umožnilo by to jednoduchšie nasadenie.

3.1.2 Monitorovanie zariadení pripojených k sieti

Detekcia a monitorovanie zariadení pripojených k sieti je kľúčová k zaisteniu bezpečnosti na sieti. Ak vieme, zariadenia akého typu sa nachádzajú v sieti, vieme predpokladať ako by sa dané zariadenie malo správať. Ak napríklad IP kamera začne vysielať desiatky HTTP dotazov na nejaký server, vieme, že niečo nie je v poriadku, ale jedná sa pravdepodobne o DDoS[14] útok.

Ďalším dôvodom na monitorovanie zariadení môže byť schopnosť odhaliť zariadenia, ktoré na sieť nepatria. Môže sa jednať napríklad o útočníka, ktorý prelomil (alebo iným spôsobom zistil) heslo k WiFi sieti a získal k nej prístup, čo je asi najbežnejší typ „útoku“ na domácu sieť.

Je taktiež dobré, aby sme vedeli aké služby sa nachádzajú v našej sieti, aké porty majú naše zariadenia otvorené. To nie je úplne nutné pre bezpečnosť siete, avšak v prípade, že sa útočník dostane do vnútra siete, môže zohrať rolu, či naše zariadenia používajú nové verzie služieb, alebo staršie, ktoré nemajú záplaty a sú zraniteľné voči útokom.

Navrhované riešenie:

Monitorovanie ARP dotazov a odpovedí Arp dotaz je vysielaný vždy, keď chce zariadenie na sieti zahájiť komunikáciu, a nevie akú MAC adresu má zariadenie s ktorým chce komunikovať. Tento dotaz je broadcastový – odoslaný všetkým zariadeniam. To znamená, že ak tieto dotazy budeme zaznamenávať, môžme zistiť, že sa na sieti vyskytlo zariadenie, ktoré sme predtým nezaznamenali. Nevýhodou môže byť, že nie sme schopní zistiť presný moment, kedy sa zariadenie odpojilo, iba keď zistíme, že sme prestali dostávať ARP dotazy.

Monitorovanie DHCP dotazov a odpovedí Domáce siete dnes spoliehajú na DHCP server. Zariadenie sa pripojí na sieť a vyžiada si od DHCP serveru, aby mu bola pridelená IP adresa. Keďže tieto požiadavky sú broadcastové, môžeme ich zachytiť a zaznamenávať si ich. Tak môžeme odhaľovať nové neznáme zariadenia. Nevýhodou je, že nevieme presne, akú IP adresu dané zariadenie obdržalo, predože DHCP odpoveď je posielaná ako unicast (ak by však DHCP serverom bol náš systém,

[14] Distributed Denial of Service

túto informáciu vieme získať). Taktiež nevieme presný moment odpojenia, ibaže by zariadenie pri odpojení odoslalo DHCP release, čo však štandard nevyžaduje. Asi najväčším problémom je, že útočník sa na sieť môže pripojiť bez použitia DHCP – pomocou staticky nastavenej IP adresy – čo týmto spôsobom nezaznamenáme.

Monitorovanie bezdrôtovej komunikácie Bezdrôtová sieť je dnes už súčasťou každej domácej siete. To znamená, že môžme odpočúvať bezdrôtovú komunikáciu a sledovať, ktoré zariadenia sú pripojené ku ktorému AP. A tak si môžme zaznamenávať ktoré zariadenia sú pripojené k nášmu AP. Nevýhodou je nutnosť dostatočne silnej antény na zachytenie všetkých zariadení. Nevýhodou je, že nemáme prehľad o IP adresách zariadení a taktiež nie sme schopní detekovať zariadenia pripojené káblom.

Periodické scanovanie siete Toto riešenie vyžaduje aktívnu periodickú činnosť na sieti – spúšťanie akéhosi skenovacieho programu, ktorý odhalí aké zariadenia sa na sieti nachádzajú a prípadne aké porty majú zariadenia otvorené, aké služby na nich bežia, aký operačný systém používajú a akého druhu je dané zariadenie. Oproti predchádzajúcim spôsobom odhaľovania zariadení na sieti má výhodu, že dokáže zistiť omnoho viac informácií o zariadeniach. Nevýhodou však je zaťažovanie siete dátovým tokom scannera a tento scanner musí byť aktívne spúšťaný v nejakých časových intervaloch. Ak je príliš krátky, sieť bude zbytočne preťažovaná. Ak zasa dlhý, môže byť nové zariadenie odhalené príliš neskoro alebo vôbec, ak sa stihlo odpojiť. Za tento čas mohol byť už dávno daným zariadením vykonaný útok vovnútri siete.

3.1.3 Detekcia útokov hroziacich v domácej sieti

Domáca sieť je iného rázu než podniková sieť. A taktiež aj útoky, ktoré tu môžu nastať sú často iného rázu.

Navrhované riešenie:

Implementácia vlastného IDS riešenia Táto možnosť by si vyžadovala komplikovanú implementáciu. Systém by musel byť dostatočne rýchly aby dokázal reagovať na možné hrozby

Použitie existujúceho IDS systému Výhodou by bolo, že by nebolo nutné implementovať vlastný IDS systém. Takéto systémy sú zároveň už otestované a funkčné, takže sa na ne dá spoľahnúť.

3.1.4 Kontrola prístupu zariadení zo siete do internetu a naopak – Firewall

Navrhované riešenie:

Použitie nástroja iptables Tento nástroj slúži na konfiguráciu modulov jadra Linuxu, ktoré súvisia s rôznymi sieťovými funkciami. Konkrétne by bol použitý modul `iptable_filter`, ktorý je spojený s tabuľkov `filter` programu iptables. Pridávaním pravidiel do tejto tabuľky pomocou nástroja `iptables` (konkrétne do „chainu“ FORWARD) je možné definovať firewallové pravidlá.

3.1.5 Monitorovanie sieťových spojení a možnosť dané spojenie ukončiť

Navrhované riešenie:

Použitie nástroja `conntrack` Tento nástroj úzko súvisí s modulom jadra `nf_conntrack`, ktorý, ako názov napovedá, slúží na sledovanie sieťových spojení. Vďaka nemu je možné s týmto modulom komunikovať – dotazovať sa na existujúce spojenia, spojenia ukončovať či modifikovať.

3.1.6 Notifikovanie užívateľa pri udalostiach na sieti

V prípade, že na sieti nastane nejaká udalosť, je vhodné aby bol o tom administrátor oboznámený. Jeho akcia môže byť oveľa účinnejšia, než akcia systému, pretože ako človek si na sieť dokáže vytvoriť komplexný pohľad (k čomu mu nôže pomôcť aj tento systém, vďaka ktorému napríklad zistí, aké zariadenia má na svojej sieti).

Navrhované riešenie:

Odosielanie udalostí do klientskej aplikácie Ak by bola ako klient použitá mobilná aplikácia, mohol by systém priamo odosielať notifikácie do tejto aplikácie. Výhodou by bolo okamžité notifikovanie administrátora, napríklad v porovnaní s webovou aplikáciou, kam by sa musel najprv prihlásiť a následne by videl udalosti. Nevýhodou však je komplikovanejšia implementácia/konfigurácia v prípade, že chce administrátor dostávať notifikácie aj keď nie je doma.

E-mail Administrátor môže byť notifikovaný o udalostiach E-mailom. Výhodou je jednoduchšia implementácia, avšak za cenu toho, že notifikačný systém nie je integrovný do aplikácie.

SMS Asi najspoľahlivejší spôsob, je odoslanie SMS do mobilného telefónu administrátora pomocou nejakej existujúcej webovej služby. Takéto riešenie však pravdepodobne nebude bezplatné.

3.1.7 Rodičovská kontrola nad prístupom detí k Internetu

3.1.7.1 Filtrovanie webového obsahu

Systém má byť schopný administrátorovi poskytnút možnosť, ako detekovať pre deti nebezpečný webový obsah a definovať akcie, ktoré majú byť vykonané pri zistení takéhoto obsahu.

Navrhované riešenie:

DNS monitorovanie Ak by systém monitoroval DNS dotazy a kontroloval ich obsah, prípadne bol schopný blokovať odpovede, bolo by takto možné filtrovať webový obsah. Systém by musel mať nejaký blacklist domén, ktoré sú označené ako nebezpečné alebo slovník slov, ktoré ak sa vyskytnú v doméne, bude sa považovať za nebezpečnú. Následne by bola odpoveď zahodená a tým pádom by komunikujúce zariadenie nebolo schopné preložiť danú doménu na IP adresu a zobraziť danú webovú stránku.

Výhodou tohto riešenia je, že je pomerne jednoduché na implementáciu. Nevýhodou však je, že systém nie je schopný reálne nahliadnuť to obsahu HTTP protokolu, a tak nemusí byť schopný odfiltrovať nebezpečný obsah v prípade, že sa doména nejaví podozrivá.

Monitorovanie obsahu HTTP a HTTPS Ak by bol systém schopný monitorovať obsah HTTP protokolu, bolo by možné označiť obsah za nebezpečný s vyššou mierou úspešnosti, než pri čistom monitorovaní domén. Tento spôsob umožňuje nielen nahliadnutie do HTML kódu stránky, ale aj do obsahu hlavičiek HTTP protokolu, kde sa nachádza napríklad aj URL. Toto sa dá dosiahnuť pomocou proxy serveru v režime *intercept*.

Problémom tohto riešenia však je, že väčšina webových stránok už dnes používa HTTPS a ako vieme, protokol HTTPS je šifrovaný pomocou TLS. Dešifrovať ho je v rozumnej dobe hrubou silou nemožné. Spôsob, ako sa však dá monitorovať komunikácia, je použiť proxy server schopný pracovať nielen s HTTP ale aj s HTTPS. Takýto proxy klientovi dodá falošný certifikát pre navštevovanú stránku, ktorý podpíše za behu svojím self-signed koreňovým certifikátom a so serverom potom komunikuje pomocou legitímneho certifikátu. Komunikácia potom prebieha šifrovane v dvoch nezávislých kanáloch, medzi ktorými je proxy server, ktorý vidí celú komunikáciu v nešifrovanej podobe.

Problémom však je, že klient dokáže detekovať dodaný falošný certifikát, ktorý je podpísaný neznámou autoritou ktorú klient nepozná. Klient teda odmietne komunikovať so serverom, pretože jeho certifikát je falošný. Väčšinou webový prehliadač notifikuje užívateľa o falošnom

certifikáte, a v niektorých prípadoch mu ponúkne túto bezpečnostnú politiku obísť a stránku napriek tomu navštíviť. Ak však na klienta nainštalujeme koreňový certifikát proxy serveru, klient začne takéto certifikáty akceptovať, lebo už pozná jeho koreňový certifikát, a tak takáto komunikácia prebehne bez problémov a bez varovania klienta. Na popísaný proces sa dajú využiť nasledujúce programy:

SSLsplit Jedná sa o nástroj určený na vyššie popísaný proces. Terminuje TLS spojenia, klientovi pošle falošný certifikát a na základe neho s klientom komunikuje. Zároveň iniciuje vlastné spojenie k serveru. Dešifrované dáta potom zapisuje do zvoleného súboru, napríklad aj do pomenovanej rúry[15]. Je schopný monitorovať aj iné TLS spojenia než HTTPS. Výhodou je, že to je dedikovaný nástroj určený práve na túto činnosť. Nevýhodou je, že tento -nástroj automaticky prijme všetky certifikáty (aj falošné) od serveru. Znamená to že ak nejaký iný útočník použije tento nástroj niekde medzi nami a serverom, náš SSLsplit dostane falošný certifikát, avšak automaticky ho príjme. Klient následne dostane nami vygenerovaný certifikát a všetko funguje ako má avšak netuší, že ho niekto odpočúva.

Squid Je to open-source proxy server na všeobecné použitie. Slúži hlavne ako cacheovací server pre urýchlenie HTTP protokolu. Je však schopný pracovať aj s protokolom HTTPS, keď je kompilovaný so správnymi prepínačmi. Podporuje široký rozsah nastavení prístupových práv a má širokú komunitu. Ďalšou výhodou je, že od verzie 3.3 podporuje takzvanú imitáciu serverového certifikátu. Táto funkcia umožňuje proxy serveru vytvoriť falošný certifikát, ktorého polia čo možno najviac kopírujú polia originálneho certifikátu[21]. Tým pádom dostane klient certifikát, ktorý je čo najviac podobný pôvodnému, a ak sa v ňom vyskytne nejaká chyba a klient ju rozpozná, odmietne certifikát. Navyše samotný squid dokáže detekovať chybné a neplatné certifikáty a klientovi poskytne chybovú stránku namiesto skutočnej, ktorá mohla byť napadnutá. Nevýhodou Squidu v tejto oblasti je, že neexistuje jednoduchý spôsob, ako si zobraziť zachytený dátový tok okrem HTTP hlavičiek, ktoré sú ukladané do logu. Ďalšou nevýhodou je, že nie je zvyčajne poskytovaný s HTTPS podporou a užívateľ si ho musí sám skompilovať sám. [22]

SSLstrip Nástroj trošku z iného súdka, než predchádzajúce. Funguje podobne ako SSLsplit, ale namiesto toho, aby s klientom komunikoval pomocou HTTPS, komunikuje s ním pomocou HTTP a nedovolí, aby bola komunikácia povýšená na HTTPS. Nevýhodou však

[15]angl. named pipe, je to súbor, ktorý slúži ako FIFO fronta

je, že ak klientský prehliadač začne komunikáciu v HTTPS, nedá sa už ponížiť na HTTP a tak dáta nemožno dešifrovať.

3.1.8 Rozšíriteľnosť detekčných schopností

Navrhované riešenie:

- Použitie existujúceho IDS systému, ktorý má rozšíriteľný zoznam signatúr
- Využitie strojového učenia

3.1.9 Nízke náklady na systém

Navrhované riešenie:

- Využitie open-source softvéru
- Použitie nízkonákladovej hardwarovej platformy

3.2 Zvolené riešenia a technológie

V tejto sekcií sú popísané zvolené riešenia a technológie a dôvody, prečo som ich zvolil.

3.2.1 Platforma

Ako softvérovú platformu pre tento systém som zvolil operačný systém GNU/Linux, ktorý je vo vysokej miere konfigurovateľný, je open-source, a podporuje požadované sieťové schopnosti, najmä čo sa týka frameworku Netfilter[16]. Vďaka tomu bude naše zariadenie schopné fungovať ako router, vykonávať preklad adries, definovať firewallové pravidlá a pod.

Ako hardvérovú platformu som zvolil zariadenie Raspberry Pi 3, ktoré je ľahko dostupné za pomerne nízku cenu, obsahuje jedno ethernetové 100 Mbit/s rozhranie a jedno WiFi rozhranie. Má štvorjadrový procesor s frekvenciou 1,2 GHz a 1 GB RAM, čo je dostatočné na beh aplikácie a všetkých komponentov zmienených nižšie.

3.2.2 Monitorovanie dátových tokov

Na monitorovanie dátových tokov zariadení som sa rozhodol použiť techniku vytvorenia vlastného DHCP serveru, ktorý bude distribuovať adresy z vlastného, nezávislého rozsahu, čo bude mať výhodu, že budeme mať pod

[16]Množina modulov Linux kernelu, ktoré poskytujú rôzne sieťové funkcie, ako napríklad firewall, sledovanie spojení, NAT, ...

kontrolou všetky toky zariadení, a budeme ich vedieť rozlíšiť. Zariadeniu bude stačiť jediné sieťové rozhranie. Nevýhodou pre administrátora však bude nutnosť deaktivovať DHCP server na svojom domácom routri.

Implementáciou DHCP servera bude Dnsmasq, ktorý je pomerne rýchy a nenáročný na systémové zdroje. Funguje zároveň aj ako caching DNS server, čo môže urýchliť DNS dotazy v sieti.

NAT bude implementovaný pomocou `iptable_nat` modulu Netfiltru, a bude konfigurovaný nástrojom `iptables`.

Routovanie medzi našou podsieťou a pôvodnou sieťou bude zaistené kernelom operačného systému.

3.2.3 Detekcia útokov

Na detekciu útokov bude systém používať IDS systém Suricata, ktorého výhodou je viacvláknovosť, ktorá sa nám hodí na viacjadrovom procesore Raspberry Pi 3. Zároveň bude zaistená kompatibilita so Snortom, za ktorým bude možné v prípade potreby Suricatu nahradiť. Použitie existujúceho IDS je výhodné, pretože tak nie je nutné implementovať vlastný IDS systém, čo by bolo extrémne náročné. Tento systém však detekuje útoky až od 4. vrstvy OSI modelu, avšak niektoré útoky (napríklad ARP spoofing) fungujú na nižších protokoloch.

Bude nutné použiť aj vlastné riešenie na detekciu paketov z nižších vrstiev. Na to bude použitý program `tcpdump` ktorý dokáže zachytávať pakety zvoleného protokolu na sieťovom rozhraní.

3.2.4 Rodičovská kontrola

Rodičovská kontrola bude zatiaľ realizovaná v jednom aspekte – filtrovanie webového obsahu – neskôr bude rozšírená na viac modulov a bude poskytovať voac konfiguračných možností. Administrátor bude môcť vybrať zariadenia, na ktorých bude chcieť vykonávať rodičovskú kontrolu.

3.2.4.1 Filtrovanie obsahu

Filtrovanie obsahu bude riešené blokovaním známych domén s nevhodným obsahom. Okrem toho však bude použitý intercepting proxy server, ktorý bude zachytávať HTTP požiadavky klienta a predávať ich serveru. Tým pádom bude mať prehľad o obsahu, ktorý je preposielaný medzi klientom a serverom.

Keďže v dnešnej dobe už väčšina webových serverov používa HTTPS, monitorovanie iba protokolu HTTP by bolo nepostačujúce. Preto bude zariadenie mať možnosť pre vybrané zariadenia zvoliť dešifrovanie HTTPS protokolu. To bude zaistené proxy serverom, ktorý je schopný za behu podpisovať falošné certifikáty a predávať ich klientovi. Ako implementáciu tohoto proxy serveru som vybral Squid, ktorý má veľké množstvo konfiguračných možností, má širokú komunitu a oproti ostatným preskúmaným serverom je bezpečnejší z hľadiska

validácie serverových certifikátov. Nevýhodou tohto programu je, že vyžaduje vlastnú kompiláciu na to, aby podporoval HTTPS a na filtrovanie obsahu musí byť použitý samostatný ICAP[17] server.

ICAP server bude súčasťou hlavného serveru. Squid bude komunikovať s týmto ICAP serverom pomocou ICAP protokolu. Keďže ICAP server a hlavný server budú jedna a tá istá aplikácia, budú môcť navzájom komunikovať, a tak bude môcť server reagovať na obsah prenášaný HTTP protokolom, a to napríklad tak, že dá ICAP serveru pokyn, aby komunikáciu zastavil (zobrazí sa iná stránka, než požadoval HTTP klient).

Obsah bude systémom kontrolovaný na základe slovníka „nebezpečných" slov, ktoré si užívateľ bude môcť nadefinovať. Slová z tohto slovníka budú následne vyhľadávané v obsahu HTTP protokolu a pri zhode bude vykonaná užívateľom definovaná akcia.

3.2.5 Monitorovanie pripojených zariadení

Monitorovanie pripojených zariadení je jedna z najdôležitejších súčastí systému. Vďaka tomu bude mať administrátor prehľad o počte pripojených zariadení, type zariadenia a pod.

Na detekciu novo pripojených zariadení bude použité monitorovanie ARP dotazov pomocou nástroja `tcpdump`. To umožní okamžitú detekciu novo pripojeného zariadenia. Zariadenia si bude systém ukladať na základe ich MAC adresy. V prípade zaznamenania novej adresy bude vykonaná definovaná akcia. Zariadenia, ktoré sú aktuálne pripojené budú kontrolované pomocou periodického scanovania nástrojom `arp-scan`, čím sa bude overovať ich prítomnosť na sieti.

Na hlbšiu analýzu pripojených zariadení bude použitý nástroj `nmap`, pomocou ktorého bude možné zistiť viac informácií o danom zariadení, čo poslúži administrátorovi na bližšiu klasifikáciu typov zariadení, ktoré sú na jeho sieti, portov, ktoré majú otvorené, a verzií služieb, ktoré na nich bežia. Zároveň tak bude vedieť „ktoré zariadenie je ktoré", inak povedné, nebude zahltený iba zoznamom IP a MAC adries zariadení, ale bude o každom zariadení vedieť ďalšie informácie ako OS, otvorené porty či typ zariadenia.

3.2.6 Monitorovanie bezdrôtových zariadení v okolí

Na monitorovanie bezdrôtových zariadení, bude použité WiFi rozhranie zariadenia Raspberry Pi 3, prípadne iné pripojené bezdrôtové rozhranie podporujúce takzvaný monitor mode. Rámce protokolu 802.11 bude dekódovať nástroj `airodump-ng`, ktorého výstup budeme periodicky parsovať. V tomto výstupe sa nachádzajú klientské zariadenia a aj prístupové body, plus informácie ku každému zariadeniu zozbierané zo zachytených rámcov.

[17] Internet Content Adaptation Protocol, popísaný v RFC 3507, slúži na kontrolu a modifikáciu obsahu HTTP protokolu.

Tieto informácie budú administrátorovi slúžiť na to, aby mal predstavu aké siete a aké zariadenia sa nacháadzajú v jeho susedstve, prípadne aké zariadenia sú pripojené k jeho bezdrôtovej sieti.

3.2.7 Znemožnenie zariadeniu komunikovať

V prípade pokusu o obídenie bezpečnostnej politiky nastolenej systémom v sieti alebo pri neoprávnenom pripojení do siete bude na žiadosť administrátora systém schopný zastaviť, prípadne obmedziť komunikáciu zariadenia v sieti.

V prípade, že zariadenie o ktoré sa jedná bude pripojené k sieti bezdrôtovo, systém použije nástroj `aireplay-ng`, ktorý vyšle 802.11 deautentikačné rámce, ktoré spôsobia odpojenie zariadenia od siete.

3.2.8 Server

Na zariadení bude bežať server, ktorý bude dá sa povedať „srdcom“ celého systému. Tento server bude spravovať všetky zmienené komponenty, vykonávať potrebné akcie a komunikovať s administrátorom.

Ako implementačný jazyk som zvolil jazyk Java a to najmä z dôvodu existencie veľkého množstva externých knižníc. To v prípade potreby umožní neskoršie rozšírenie funkcionalít systému. Ďalším dôvodom je bezpečnosť. Keďže Java aplikácie bežia v sandboxe, ktorým je JVM[18], nemajú priamy prístup k systémovým zdrojom. To je výhodné z hľadiska bezpečnosti, pretože tu nemôžu vzniknúť zraniteľnosti typu buffer overflow, ktoré nastávajú z nepozornosti programátora. Samozrejme, vždy môže byť chyba v implementácií JVM. Avšak pravdepodobnosť nájdenia takejto chyby je podľa môjho názoru menšia, než pravdepodobnosť toho, že by som do implementácie serveru napríklad v jazyku C++ nevniesol zraniteľnosť typu buffer overflow ja sám. Ďalší dôvod pre výber tohto jazyka bol nutnosť rýchleho vývoja. Neskôr by systém mohol byť prípadne prepísaný do natívnejšieho jazyka. Avšak java obsahuje takzvané JNI – Java Native Interface, ktoré umožňuje niektoré metódy deklarovať s kľúčovým slovom `native` a implementovať ich v jazyku C/C++, následne preložiť na .so súbor a ten načítať počas behu JVM. Keď bude táto metóda zavolaná, JVM zavolá jej implementáciu z .so súboru. Takýmto spôsobom je kritickú časť kódu, vyžadujúcu rýchlosť, možné spúšťať v natívnom kóde priamo na CPU.

Ako behové prostredie som zvolil OpenJDK 8, ktoré podporuje mnoho optimalizačných možností, ako napríklad JIT[19] kompiláciu. JIT kompilácia príde vhod najmä pri dlho bežiacich programoch (akým server istotne je), kedy JVM postupne zbiera štatistiky o behu interpretovaných metód. Po dostatočnom množstve interpretovaných behov (väčšinou 10000), JVM danú

[18] Java Virtual Machine

[19] Just In Time, kompilácia za behu programu

metódu skompiluje do strojového kódu daného CPU a optimalizuje ju na základe zozbieraných štatistík. Pri následnom volaní sa táto metóda už nebude interpretovať, ale zavolá sa priamo natívny kód, ktorý ju reprezentuje.

3.2.9 Klient

Na administráciu serveru bude použitá Android aplikácia. Bude slúžiť ako tenký klient, čo znamená, že všetku dôležitú prácu bude musieť odviesť server. Toto riešenie som zvolil, aby mohol byť neskôr implementovaný aj webový klient. Samozrejme, keďže sa bude jednať o Android aplikáciu, klient nemusí byť „úplne tenký“, pretože aplikácia si môže niektoré dáta ukladať do vlastnej cache aby nemusela zaťažovať server. Taktiež môže použiť vlastné súborové úložisko, aby bol administrátor schopný zobrazovať dáta o sieti aj keď bude offline. Výhodou použitia Android aplikácie je aj to, že časť kódu bude môcť byť spoločná so serverom (keďže android používa Javu) a nebude nutné reimplementovať to isté v inom jazyku.

Pomocou tejto aplikácie bude zároveň administrátor notifikovaný ohľadom sieťových udalostí.

3.2.10 Sieťový protokol

Sieťový protokol na komunikáciu medzi klientom a serverom bude riešený pomocou odosielania objektov vo formáte JSON. Komunikovať sa bude pomocou tcp spojenia, ktoré bude zabezpečené TLS. Pri prvom pripojení sa vytvorí serverový certifikát, ktorý si následne klient zapamätá a na základe neho sa bude k serveru pripájať.

3.3 Architektúra navrhovaného systému

Architektúra systému vyplýva z riešení zvolených v predchádzajúcej sekcií. Hlavnou časťou systému bude server bežiaci v JVM, ktorý bude spravovať všetky ostatné komponenty, spúšťať externé programy, komunikovať s klientom a reagovať na udalosti.

Externými komponentami, ktoré bude server ovládať, budú nasledujúce programy. Budú inštalované ako systémové služby, takže o ich správu sa bude starať OS. V prípade zmeny konfigurácie alebo nutnosti službu vypnúť či zapnúť bude použitý nástroj `systemctl`. Server bude pre tieto služby generovať konfiguračné súbory a tak bude môcť reagovať na rôzne sieťové konfigurácie.

Suricata IDS systém – S hlavným serverom bude komunikovať pomocou takzvaného „fastlogu“, ktorý bude zapisovaný do pomenovanej rúry, z ktorej bude server čítať. Týmto spôsobom sa budú serveru predávať výstražné hlášky (alerts), na ktoré bude môcť server reagovať.

Squid proxy server – S hlavným serverom bude komunikovať prostredníctvom ICAP protokolu.

Dnsmasq DHCP a DNS server – S hlavným serverom bude komunikovať pomocou logu, ktorý bude reprezentovaný pomenovanou rúrou. Pomocou neho sa budú serveru predávať informácie o DHCP aktivite na sieti.

Najdôležitejšími externými programami, ktoré bude server spúšťať budú nasledujúce programy. Server ich bude spúšťať ako svoje subprocesy. Pri spúšťaní im predá potrebné argumenty a následne bude schopný čítať ich štandardný výstup, chybový výstup a zapisovať na ich štandardný vstup.

Arp-scan zisťovanie prítomnosti zariadení na sieti

Nmap skenovanie zariadení na sieti

Iptables konfigurácia modulov Netfiltru

Conntrack monitorovanie a ukončovanie pripojení

Airodump-ng monitorovanie bezdrôtových zariadení

Aireplay-ng vysielanie 802.11 deautentikačných rámcov

Tcpdump zachytávanie paketov na sieťovom rozhraní

Systemctl konfigurácia služieb OS

Bližšie je architektúra načrtnutá na obrázku 3.4.

3.4 Umiestnenie systému v sieti

Zariadenie bude jediným fyzickým ethernetovým rozhraním pripojené k LAN portu domáceho routru. Tým pádom bude umiestnené v sieťovom segmente domácej siete spolu so všetkými zariadeniami v sieti. Toto fyzické rozhranie však bude rozdelené na viac virtuálnych rozhraní čo zaistí, že zariadenie bude môcť mať viac IP adries na danom fyzickom rozhraní. Vďaka tomu sa bude môcť zariadenie nachádzať vo viacerých podsieťach. Jednou z nich bude pôvodná podsieť spravovaná domácim routrom a jeho DHCP serverom. Ďalšou podsieťou v tomto segmente bude podsieť vytvorená našim zariadením. Na tejto podsieti sa budú nachádzať všetky ostatné zariadenia okrem routru, ktorý bude v pôvodnej podsieti.

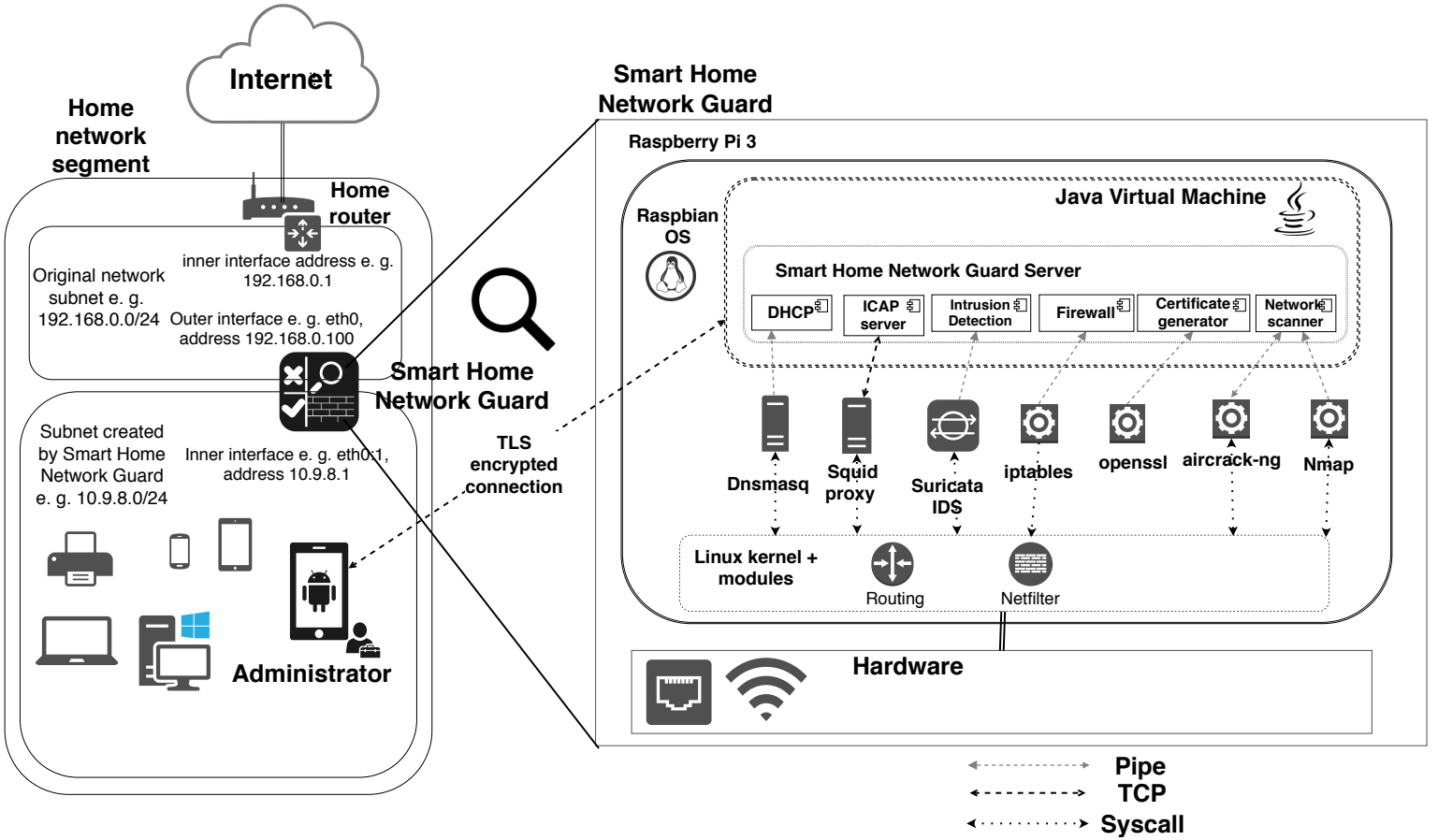
Na to, aby sme zariadenia dostali z pôvodnej podsiete na našim zariadením vytvorenú podsieť, bude nutné, aby bol na pôvodnom routri administrátorom vypnutý DHCP server, ktorý prideľoval zariadeniam IP adresy. Jeho funkciu preberie DHCP server nášho zariadenia, ktorý však nebude rozdávať adresy

z rozsahu pôvodnej podsiete, ale z úplne iného rozsahu – rozsahu ním vytvorenej podsiete – ktorý sa žiadnym spôsobom nesmie prekryvať s pôvodným.

Naše zariadenie sa teda bude jedným rozhraním nachádzať v pôvodnej podsieti a druhým, virtuálnym rozhraním v ním vytvorenej podsieti a vďaka tomu bude môcť fungovať ako router medzi týmito podsieťami. Zároveň bude vykonávať preklad adries (NAT) medzi nimi a taktiež bude naše zariadenie fungovať ako default gateway pre zariadenia v ním vytvorenej podsieti. Takáto konfigurácia bude mať za následok, že ak bude nejaké zariadenie chcieť pristupovať k Internetu, bude musieť komunikovať cez naše zariadenie, vďaka čomu budeme mať kontrolu nad prístupom zariadení k Internetu a taktiež bude možné monitorovať dáta, ktoré odosielajú a prijímajú.

Nevýhodou tejto konfigurácie však je, že zariadenia od routra nebudú oddelené fyzicky, iba logicky tým, že im DHCP server pridelí adresu. Ak by si zariadenie samo nastavilo IP adresu z rozsahu podsiete v ktorej sa nachádza router, mohlo by priamo komunikovať so zariademiami v Internete. Takéto obídenie však vyžaduje pokročilejšie znalosti z odboru sietí, a tak bude takáto konfigurácia dostatočne neobíditeľná pre drvivú väčšinu užívateľov na sieti (vačšina užívateľov domácich sietí nie sú odborníci na PC siete ani hackeri). Avšak na to, aby sa náš systém mohol proti tomuto skutočne brániť, bude nutné sledovať zariadenia, ktoré sú k sieti pripojené a v prípade zistenia zariadenia, ktoré si samo pridelilo IP adresu notifikovať administrátora a prípadne na toto zariadenie vykonať nejaký typ útoku, ktorý zariadeniu znemožní komunikovať (napríklad ARP spoofing alebo odpojenie od siete pomocou 802.11 deautentikačných rámcov v prípade bezdrôtového zariadenia).

Bližšie je sieťová architektúra načrtnutá na obrázku 3.4.



Obr. 3.1: Diagram architektúry systému a jeho nasadenia v domácej sieti. Vľavo je načrtnutné umiestnenie systému v sieti a vpravo je priblížené na samotný systém a jeho architektúru.

Kapitola 4

Realizácia

Táto kapitola popisuje najdôležitejšie časti implementácie systému.

4.1 Implementácia v jazyku Java

Keďže som sa rozhodol použiť jazyk Java, bolo možné zdielať časť kódu medzi klientskou Android aplikáciou a serverom. Vytvoril som balíček, do ktorého som umiestnil spoločný kód. Jedná sa hlavne o pomocné triedy, ďalej takzvané „utilty“ triedy a triedy slúžiace na komunikáciu.

Najdôležitejšie použité knižnice:

RxJava 2 Umožnuje asynchrónne programovanie na základe udalostí.

BouncyCastle Kryptografická knižnica.

GSON Serializácia a deserializácia objektov do JSON formátu.

Aho-Corasick Implementácia algoritmu Aho-Corasick pre efektívne vyhľadávanie veľkého množstva stringov.

4.1.1 Protokol

Komunikačný protokol je založený na dotazoch a odpovediach. Hlavným komunikačným objektom je trieda `Message`, od ktorej ďalej dedia ostatné triedy posielané medzi klientom a serverom. Medzi ne patria napríklad `CommandMessage`, `ResponseMessage` či `PingMessage`. Dôležitými triedami, ktoré pracujú s objektami typu `Message` sú:

MessageSender Slúži na odosielanie správ.

ResponseCache Zapamätáva si odoslané správy a očakáva odpovede na ne. V prípade doručenej odpovede notifikuje odosielateľa a predá mu `ResponseMessage`.

Na samotnú sieťovú komunikáciu je použitý protokol TCP. Bol zvolený port 4911, keďže sa jedná o nepriradený port. Komunikácia je zabezpečená pomocou TLS, administrátor si pri prvom spustení vygeneruje privátny kľúč a certifikát (kým to neurobí, server mu nedovolí vykonávať iné akcie), ktoré sú následne použité na šifrovanie komunikácie a autentizáciu serveru voči klientovi (taktiež na podpisovanie falošných certifikátov Squid proxy serverom). Klient je voči serveru autentizovaný pomocou 128 bitového UUID, ktoré mu pridelí server pri prvom pripojení. Pri prvom pripojení sa neoveruje identita klienta ani serveru a je použitý predgenerovaný kľúč a certifikát. Preto nie je prvotné párovanie bezpečné, avšak je ho možné uskutočniť cez USB kábel pomocou funkcie Androidu zvanej USB tethering, kedy nehrozí riziko odpočúvania.

4.2 Inštalácia

Systém je zameraný na beh v prostredí OS Raspbian. Na inštaláciu som preto vytvoril dva skripty, ktoré sú zamerané na tento operačný systém. Skript `install.sh` má za úlohu vytvoriť užívateľa, pod ktorým bude server bežať, dať mu práva spúšťať progrogramy ako `root` pomocou programu `sudo` bez hesla, vytvoriť potrebné adresáre a pomenované rúry, nastaviť im potrebné práva a pridať príkaz, ktorý bude spúštať server, do súboru `/etc/rc.local` (ktorý je vykonaný po štarte OS). Zároveň tento skript spustí skript `installDependencies.sh`, ktorý pomocou programu `apt`, slúžiaceho na správu balíčkov, nainštaluje potrebné programy.

Neskôr by mal byť systém distribuovaný ako obraz disku, avšak kvôli dodržaniu licencií, vzťahujúcich sa na OS Raspbian a jeho súčsti, som zatiaľ musel zvoliť toto riešenie ktoré je bohužiaľ pre užívateľa komplikovanejšie.

4.3 Spúšťanie serveru

Ako bolo spomenuté v 4.2, server je spúšťaný skriptom `/etc/rc.local`. V tomto skripte je príkaz, ktorý spustí skript `startup.sh` pod úžívateľom, ktorý bol vytvorený pri inštalácií. Tento skript má na starosti spustenie serveru, zapnutie NAT a spustenie IP forwardingu (routing). Zároveň je ním spustený skript `checkServerRunning.sh`, ktorý periodicky (každých 10 minút) testuje, či server beží. Prvý test prebehne po 10 sekundách po spustení. Ak server nebeží, pokúsi sa o jeho opätovné spustenie, a ak sa mu ani to nepodarí, je spustený skript `fallback.sh`, ktorý prepne systém do pohotovostného režimu, aby bola sieť funkčná aj bez bežiaceho serveru.

4.4 Trieda Server

Jedná sa o najdôležitejšiu triedu serveru. Má na starosti inicializáciu všetkých súčastí systému, konfiguráciu systému a napokon spustenie serverového socketu, ktorý čaká na pripojenia od klientov.

4.5 Trieda ProcessExecutor

Táto trieda slúži na spúšťanie procesov. Program s argumentami je prestavovaný objektom triedy `CommandLine`, ktorý ich má uložené ako text v obojstrannej fronte. To má výhody, že argumenty je možné pridávať aj na začiatok aj na koniec. `ProcessExecutor` má metódu `executeProcess()`, ktorá prijíma `CommandLine` ako parameter. Ďalšie parametre popisujú, či má byť daný proces spustený pod *root* užívateľom, či má byť jeho výstup bufferovaný v blokoch alebo riadkoch a napokon hodnotu v milisekundách, po ktorej má byť proces ukončený.V prípade, že má byť proces spustený privilegovane, metóda pridá na začiatok príkaz `sudo`. V prípade, že má proces skončiť po určitom čase, pridá na začiatok príkaz `timeout` s argumentom, reprezentujúcom čas. V prípade, že má byť výstup bufferovaný po riadkoch, pridá na začiatok príkaz `unbuffer`. Následne je objekt `CommandLine` preložený na pole stringov, ktoré je následne predané metóde triedy `Runtime exec()`, ktorá vo vnútri obsahuje volania systému `fork()` a `exec()`.

Tento spôsob spúšťania je výhodný z hľadiska bezpečnosti, pretože nie je volaný žiaden shell, ktorý by programy spúšťal, ale argumenty sú predávané priamo spúšťanému programu pomocou systémového volania. Volanie shellu by mohlo byť nebezpečné, pretože ak by vstup nebol dobre ošetrený , moflo by dochádzať k takzvanému „command injection“ – užívateľ by mohol „prepašovať“ svoje príkazy cez dáta do shellu, kde by boli vykonané, čo je obrovské bezpečnostné riziko.

4.6 Trieda NetworkPropertiesMiner

Táto trieda obsahuje metódy na zisťovanie stavu siete s sieťových nastavení. Jedná sa hlavne o zisťovanie adresy GW, zisťovanie akému sieťovému rozhraniu prísluší daná adresa, zisťovanie typu sieťových rozhraní, hľadanie DHCP servera na sieti a zisťovanie adresy a masky sieťového rozhrania.

4.7 Trieda NetworkActions

Ako napovedá názov, táto trieda umožňuje serveru vykonávať rôzne sieťové akcie, akými sú napríklad nastavenie adresy a masky na rozhraní, vypnutie a zapnutie rozhrania, DHCP konfigurácia rozhrania, úprava ARP tabuľky

a v neposlednom rade vytváranie súboru `/etc/network/interfaces`, ktorý slúži systému na konfiguráciu sieťových rozhraní.

4.8 Trieda Component

Táto abstraktná trieda predstavuje istý komponent systému. Má obstraktnú metódu `initComponent()`, ktorú musia potomkovia implementovať a je zavolaná pri inicializácií komponentov.

4.9 Trieda Service

Na administráciu programov, ktoré sú inštalované ako systémová služba (Dnsmasq, Suricata a Squid) slúži abstraktná trieda `Service` dediaca od triedy `Component`. Od triedy `Service` dedia triedy reprezentujúce konkrétne služby (`Dhcp, IntrusionDetection a Proxy`). Trieda obaľuje nástroj `systemctl`, ktorý slúži práve na administráciu služieb OS. Trieda poskytuje metódy ako `start(), stop(), restart()` pomocou ktorých je možné ovládať danú službu. Implementuje metodu `initComponent()`, v ktorej testuje, či je daná služba v systéme prítomná.

4.10 Generovanie kľúčov a certifikátov

Na generovanie klúčov a certifikátov používaných ICAP serverom a TLS protokolom som použil program `openssl`. Server ho spúšťa s požadovanými parametrami a na jeho štandardný vstup zapisuje informácie o certifikáte. Na spracovanie certifikátu a kľúča v aplikácií (kde sú nutné pri vytváraní SSL socketu) používam triedy balíčka *BouncyCastle*.

4.11 Ukladanie zariadení

Systém si ukladá všetky zariadenia, ktoré sa pripojili k sieti. Zariadenie v systéme reprezentuje trieda `Device`. Na ich ukladanie slúži trieda `DeviceStorage`. Táto trieda slúži zároveň iným komponentám systému aby sa mohli dotazovať na zariadenia ktoré sú uložené. Tak je možné efektívne riešiť napríklad preklady IP na MAC adresu a naopak a nie je nutné ich zisťovať zo siete vždy, keď to nejaký komponent vyžaduje. S triedou `DeviceStorage` komunikujú všetky komponenty, ktoré slúžia na skenovanie zariadení. Tak sú vždy zariadeni, ktoré má táto trieda uložené, vždy aktuálne.

Systém má ďalej triedu `ConnectedDevices`, ktorá si pamätá zariadenia, ktoré sú aktuálne pripojené. Spolupracuje s `DeviceStorage`, ktorá ju informuje o nových zariadeniach. Trieda potom sleduje, či sú zariadenia stále pripojené tak, že ich periodicky „pinguje“. Ak zistí, že zariadenie už nie je on-

line, ešte 3 krát sa pokúsi zistiť jeho prítomnosť a následne ho zabudne. Táto trieda slúži najmä na zobrazovanie aktuálne pripojených zariadení užívateľovi. Neskôr však môže byť využitá aj inými komponentami systému, ktoré si vyžadujú poznatok o aktuálne pripojených zariadeniach.

4.12 Zachytávanie paketov

Na zachytávanie paketov slúži trieda `Tcpdump`, ktorá obaľuje program `tcpdump` a parsuje jeho výstup. Zatiaľ je implementované len zachytávanie a parsovanie ARP paketov, ktoré sú reprezentované abstraktnou triedou `ArpPacket` a jej podtriedami `RequestArpPacket` a `ResponseArpPacket`.

4.13 Detekcia nových zariadení

Detekcia nových zariadení je riešná hlavne detekciou ARP dotazov. Na to slúži trieda `ArpChecker`, ktorá dostáva objekty typu `ArpPacket` od triedy `Tcpdump`. pri obdržaní objektou `RequestArpPacket` systém vytvorí nový objekt `Device` a predá ho triede `DeviceStorage`. Takto sa pri zistení nového ARP dotazu okamžite notifikuje trieda `ConnectedDevices` a tak má systém okamžite informáciu o pripojení nového zariadenia.

4.14 Trieda Aircrack

Táto trieda obaľuje programy balíčka `aircrack-ng`. Jej hlavnou úlohou je prepínanie sieťového rozhrania do monitor módu, na čo používa program `iw`, monitorovanie okolitých bezdrôtových staníc a AP pomocou `airodump-ng` a v neposlednom rade deautentikácia zariadení zo siete pomocou nástroja `airodump-ng`.

4.15 DHCP server

Ako DHCP server som použil Dnsmasq, ktorý som nainštaloval ako systémovú službu. V programe je reprezentovaný triedou `Dhcp`, ktorá má na starosti jeho spúšťanie a vypínanie, komunikáciu s ním a vytvára pre neho konfiguračné súbory. Trieda s programom komunikuje pomocou pomenovanej rúry, do ktorej Dnsmasq zapisuje informácie o DHCP dotazoch. Tie trieda parsuje a na základe nich vykonáva požadované akcie. Napríklad pri obdržaní DHCPDISCOVER dotazu trieda vytvorí nový objekt typu `Device` a predá ho triede `DeviceStorage` podobne ako je popísané v 4.13.

4.16 Skenovanie siete

Na skenovanie zariadení na sieti slúži primárne trieda `MultiPurposeNetworkScanner`, ktorá na skenovanie používa iné triedy obaľujúce konkrétne skenovacie programy. Jedná sa o triedy `NmapScanner`, `ArpScanner` a `PingScanner`.

4.16.1 Trieda NmapScanner

Slúži na hlboké skenovanie zariadení, keďže program `nmap` obsahuje množstvo konfiguračných možností a typov skenov. Z výstupu programu sú následne požadované údaje parsované a predávané volajúcemu.

4.16.2 Trieda ArpScanner

Slúži na zisťovanie prítomnosti zariadení na sieti pomocou programu `arp-scan`, ktorý vysiela zariadeniam ARP otazy a čaká na ich odpovede. Táto metóda je pomerne efektívna, pretože zariadenie teoreticky môže ignorovať ICMP Echo (ping), ale na to, aby mohlo komunikovať, musí akceptovať a odpovedať na ARP pakety.

4.16.3 Trieda PingScanner

Táto trieda je spoločná aj s klientskou aplikáciou, kde zatiaľ nie je vyuzitá, avšak neskôr bude môcť taktiež aj aplikácia vykonávať isté typy skenov. Táto trieda obaľuje nástroj `ping` a zároveň obsahuje metódy na preklad IP adresy na MAC pomocou vyslania paketu na danú IP adresu a následného parsovania ARP tabuľky.

4.17 Perzistencia

Na ukladanie a prístup k dátam som použil DAO objekty, ktoré poskytujú rozhrania (interface) na manipuláciu s dátami. Samotná ich implementácia je však zatiaľ veľmi jednoduchá. Dáta sú uložené v obyčajných dátových štruktúrach ako `Map` alebo `List` poskytovaných Javou a ich obsah je každú hodinu serializovaný do súboru pomocou serializácie poskytovanej Java Serialization API. Toto riešenie má výhodu v jednoduchej implementácií a v rýchlosti, pretože sú dáta uložené priamo v RAM, avšak veľké množstvo nevýhod, ako napríklad malá odolnosť voči chybám, a preto bude v budúcnosti použité lepšie riešenie – napríklad SQL databáza.

4.18 Udalosti

Na prácu s udalosťami systém používa abstraktnú triedu `Event`, od ktorej dedia rôzne podtriedy, reprezentujúce konkrétne udalosti (napr. `NewDeviceDiscoveredEvent` alebo `IntrusionDetectionEvent`). Ukladanie a predávanie udalostí klientovi rieši trieda `EventHandler`, ktorá obsahuje metódu na registráciu novej udalosti. V prípade registrácie novej udalosti táto trieda odošle správu o tejto udalosti všetkým pripojeným klientom.

4.19 Komunikácia s modulmi Netfiltru

4.19.1 Nástroj conntrack

Nástroj `conntrack` je oobalený triedou `NatConnections`, ktorá je potomkom triedy `Component` a poskytuje metódu na zobrazenie aktuálne aktívnych pripojení, metódu ktorá notifikuje pri vytvorení nového pripojenia a metódu na ukončenie vybraného pripojenia.

4.19.2 Nástroj iptables

Na prácu s týmto nástrojom slúži trieda `Iptables`, ktorá je potomkom triedy `Component`. Obsahuje metódy na pridávanie a odoberanie pravidiel a zisťovanie, či pravidlo existuje. Pravidlo je reprezentované triedou `Rule`, ktorá má uložené pravidlo v podobe kľúč-hodnota štruktúry argumentov programu. Kľúčami sú jednotlivé prepínače programu `iptables` (napr. kľúč: `-p`, hodnota: `tcp`). Trieda je vďaka tomu schopná vytvárať svoje inštancie parsovaním výstupu príkazu `iptables -S`. Zároveň obsahuje metódu `compile()`, ktorá z objektu vytvorí inštanciu triedy `CompiledRule`, ktorá môže byť následne predaná metódam triedy `Iptables` a vykonaná (čo sa vykoná záleží na metóde triedy `Iptables`, ktorá bola zavolaná).

Trieda `Iptables` zároveň rozširuje funkcionalitu programu `iptables` tak, že umožňuje vytvárať časované pravidlá – pravidlo sa pridá do modulu kernelu a po definovanom čase sa opäť zavolá program `iptables` a zmaže toto pravidlo. Tak je možné napríklad na určitý čas zablokovať nejakú IP adresu a následne ju znova automaticky povoliť bez nutosti ďalšej interakcie s triedou `Iptables` (trieda to spraví sama).

4.20 Trieda SpecialRule

Táto abstraktná trieda obaľuje triedu `Rule` a špecializuje ju na konkrétny typ pravidla tým, že skrýva jej metódy. Jej podtriedy potom odkrývajú niektoré z metód triedy `Rule`, čo závisí na konkrétnom type pravidla.

Najdôležitejšie podtriedy sú:

FirewallRule Reprezentuje pravidlo firewallu.

RedirectRule Slúži na presmerovanie paketov na špecifikovaný port systému.

4.21 Detekcia útokov

4.21.1 Suricata IDS

Je inštalovaný ako systémová služba. Reprezentuje ho trieda `IntrusionDetection` dediaca od `Service`. Táto trieda má na starosti jeho spúšťanie, vytváranie konfiguračných súborov a napokon parsovanie logu. Hlavným výstupom IDS je súbor fast.log, do ktorého systém zapisuje upozonenia v prípade zachytenia nejakej udalosti. Formát výstupu je nasledovný:

```
04/10/2018-12:13:31.936426 [**] [1:2018919:3] ET POLICY
possible Xiaomi phone data leakage HTTP [**] [Classification:
Potential Corporate Privacy Violation] [Priority: 1] TCP
10.9.8.109:54050 -> 52.76.188.200:80
```

Tento výstup je triedou parsovaný a v prípade nového upozornenia z neho systém vytvorí objekt typu `SendableAlert`. Ten je potom zaobalený do objektu `IntrusionDetectionEvent` a predaný `EventHandler`u. Ak je pole `Priority` rovné 1, systém dané spojenie ukončí a vytvorí nové firewallové pravidlo, ktoré zablokuje danú zdrojovú IP adresu a cieľovú IP adresu s portom na 60 sekúnd.

Signatúry sú ponechané v predvolenom stave a sú spravované nástrojom `suricata-oinkmaster-updater`, ktorý je periodicky denne spúšťaný a aktualizuje ich.

4.21.2 Detekcia ARP spoofingu

Detekciu ARP spoofingu systém rieši tak, že pri pripojení nového zariadenia, keď je mu priradená IP adresa DHCP serverom, si systém zapamätá, aká IP adresa zodpovedá danej MAC adrese. Systém ďalej zachytáva ARP odpovede v sieti a v prípade zachytenia ARP odpovede, v ktorej je mapovanie MAC na IP adresu rozdielne než zapamätané, systém to označí za ARP spoofing a zaregistruje túto udalosť. Túto detekciu rieši trieda `ArpChecker`.

4.21.3 Detekcia falošného DHCP servera

Systém periodicky scanuje sieť. Súčasťou jedného zo scanov je aj DHCP discover dotaz. V prípade, že nejaký server odpovie, systém ho označí za falošný DHCP server a zaregistruje túto udalosť.

4.21.4 Detekcia použitia statickej IP adresy

Ak sa nejaké zariadenie pokúsi o obídenie systému tým, že si nastaví IP adresu z pôvodného sieťového rozsahu a začne komunikovať, trieda `ArpChecker` to zachytí a zaregistruje novú udalosť. Neskôr bude užívateľ môcť zvoliť akciu, ktorá bude vykonaná pri zistení takéhoto zariadenia. Jedna z akcií by mohlo byť odpojenie zariadenia zo siete pomocou 802.11 deautentikačných rámcov, ak sa jedná o bezdrôtové zariadenie.

4.22 Firewall

Je reprezentovaný triedou s rovnakým názvom – `Firewall`. Táto trieda má na starosti správu firewallových pravidiel. Poskytuje metódy `append()`, `prepend()` a `delete()` slúžiace na manipuláciu s pravidlami.

Firewall je implementovaný pridávaním a odoberaním pravidiel pomocou triedy `Iptables`.

Pravidlá sú reprezentované triedou `SendableFirewallRule`, ktorá slúži zároveň na preposielanie a prijímanie pravidiel z klientskej aplikácie. Tie sú následne prevedené na objekty typu `FirewallRule`, ktoré sú ďalej kompilované a predávané triede `Iptables`, aby boli aplikované do modulu kernelu.

Pravidlá reprezentované triedou `SendableFirewallRule` umožňujú zvoliť zdrojovú a cieľovú IP adresu (prípadne rozsah adries), cieľový port (prípadne rozsah portov), protokol (konkrétne TCP a UDP) a napokon akciu, ktorá má byť vykonaná (ALLOW a DENY).

Firewallové pravidlá zatiaľ nie sú citlivé na smer vytvorenia spojenia. Plánujem však pridať možnosť, aby mohlo byť spojenie iniciované z jednej strany povolené, avšak spojenie z opačnej strany blokované. Implementácia bude jednoduchá, pretože triedy `Iptables a Rule` toto podporujú (bude to riešené pomocou „state“ extenzie programu `iptables`).

4.23 Filtrovanie obsahu HTTP a HTTPS protokolu

Filtrovanie je v systéme riešené triedou `Proxy` dediacou od `Service`, ktorá zaobaľuje Squid proxy server a ICAP server reprezentovaný triedou `IcapServer`.

Trieda `Proxy` má tiež na starosti presmerovávanie HTTP a HTTPS komunikácie vybraných zariadení cez Squid proxy. Na to používa pravidlá typu `RedirectRule`, ktorými presmeruje TCP pakety smerujúce do internetu na port 80, na *localhost* a port 3128, kde počúva Squid proxy. Rovnako port 443 je presmerovaný na port 3129, na ktorom počúva Squid, avšak s podporou HTTPS. Vďaka tomu je možné pomocou Squid proxy monitorovať HTTP a HTTPS komunikáciu vybraných zariadení.

Ďalej trieda zabezpečuje vytváranie konfiguračného súboru pre Squid a vytváranie udalostí súvisiacich s webovým obsahom.

Certifikát, ktorý používa Squid na podpisovanie falošných certifikátov predávaných webovému klientovi je ten, ktorý si užívateľ vygeneroval pri prvom pripojení.

Squid má vo svojom konfiguračnom súbore nastavené, aby preposielal komunikáciu ICAP serveru.

Jedným problémom, ktorý bolo treba vyriešiť, bolo obchádzanie proxy serveru QUIC protokolom. Je to protokol od Googlu, ktorý pracuje cez UDP a jeho cieľom je urýchliť pripojenia. Je podporovaný webovými klientmi od Googlu. Keďže však proxy server nepracuje s UDP, musel som tento protokol zakázať aby mohla byť komunikácia monitorovaná. Vychádzal som z metódy popísanej v [23]. Keďže QUIC protokol používa UDP porty 80 a 443, stačí ich jednoducho zablokovať pre vybrané zariadenia. To nezablokuje komunikáciu, iba spôsobí jej spomalenie, keďže nebude umožnený prechod na QUIC. Na blokovanie som použil pravidlá typu `FirewallRule`.

4.23.1 ICAP server

Jedná sa o najkomplikovanejšiu časť serveru. Implementoval som ho podľa RFC 3507. Slúži na komunikáciu so Squid proxy serverom, od ktorého dostáva obsah HTTP komunikácie vrátane hlavičiek. Komunikácia prebieha cez TCP na *localhoste*. Obsah je predávaný pomocou tzv. „chunked transfer encoding“ definovaného pre HTTP protokol. Znamená to, že server dostáva v blokoch („chunk“) obsah HTTP protokolu. Bolo nutné implementovať dekódovanie týchto blokov a ich skladanie do celistvého bloku dát. Nad tým je následne vykonaná slovníková kontrola obsahu. Tú má na starosti trieda `AhoCorasickMatcher`, ktorá dostane pri svojom vytvorení pole objektov `KeywordInfo`. Každý z týchto objektov popisuje jedno slovo, kategóriu daného slova a taktiež či slovo vyžaduje pevný zažiatok alebo pevný koniec. Ak napríklad je pre slovo *java* vyžadovaný pevný koniec, slovo *javascript* nebude pre neho zhodou avšak ak nie je vyžadovaný koniec, slovo je zhodou. Vyhľadávanie slov je riešené tak, aby ignorovalo veľkosť písmen a diakritiku. To má však za následok stratu výkonu, pretože text a hľadané slová musia byť pred kontrolou skonvertované do jednotnej podoby bez diakritiky a veľkých písmen. Na samotné vyhľadávanie bola použitá knižnica Aho-Corasick, pretože je nutné v jednom dlhom texte hľadať množstvo slov čo najefektívnejšie a algoritmus, ktorý táto knižnica popisuje je jedným z najefektívnejších algoritmov vhodných na tento účel. Následne sú zhody prekonvertované do objektov typu `Match`, ktoré popisujú kde bola zhoda nájdená a aký bol kontext daného slova v texte. Spolu s obsahom HTTP protokolu je týmto spôsobom kontrolovaná aj URL navštívenej stránky.

Všetkym zhodám sú následne priradené kategórie z objektov `KeywordInfo`. Na základe týchto kategórií je následne vykonaná administrátorom definovaná

akcia. Zatiaľ sú implementované dve akcie – notifikácia administrátora a zablokovanie stránky. Ku každej akcií je definovaná aj jej negácia, aby bolo možné definovať komplikovaniešie filtrovanie. Ak bude napríklad definovaná kategória *BLOCK_PROGRAMMING_LANGUAGES*, kde budú slová *c++, java, php, python* a akcia tejto kategórie bude zablokuj stránku, bude možné zadefinovať kategoriu *WARN_IF_JAVA*, ktorá bude obsahovať slovo *java* a jej akcie budú neblokuj stránku a notifikuj administrátora. Ak sa teda v stránke objaví napríklad *c++*, bude zablokovaná, avšak ak sa v nej objaví *java*, bude iba notifikovaný administrátor a stránka nebude zablokovaná.

Ak je vykonaná akcia zablokuj stránku, proxy serveru sa odošle stránka s obsahom signalizujúcim zablokovanie stránky. V opačnom prípade je proxy serveru naspäť odoslaná stránka v nezmenenej podobe, čo je pomerne neefektíve – odosiela sa naspať rovnaký obsah. ICAP protokol na toto myslí a podporuje tzv. „preview“, čo znamená, že ICAP server si môže vypýtať časť stránky, a ak je časť v poriadku, môže si ďalej vypýtať ďalšiu časť alebo odoslať odpoveď „204 - No modifications needed“, čo pre proxy server znamená, že môže komunikujúcemu klientovi predať stránku bez zmeny. To však zatiaľ môj ICAP server neimplementuje (avšak z hľadiska RFC 3507 to nie je nutné).

Problémom pri kontrole je však typ prenášaného obsahu. Ak nechceme, aby systém zbytočne pálil CPU cykly na hľadanie textu v obrázkoch alebo videách, je nutné tieto typy rozlišovať. Našťastie hlavička HTTP protokolu obsahuje položku `Content-Type:`, v ktorej je uvedený tzv. „media type“ (tiež zvaný MIME type). Ten definuje typ prenášaného obsahu. ICAP server teda ignoruje typy, pri ktorých by kontrola obsahu nedávala zmysel. Ak je však kontrola pre daný obsah zmysluplná, ale obsah je príliš veľký (nad 1,1 MB), systém prestane načítavať ďalší obsah a vykoná kontrolu nad obsahom, ktorý má a na základe neho vykoná potrebnú akciu.

Ďalším problémom, ktorý bolo potrebné vyriešiť je kódovanie znakov. Väčšina stránok dnes používa UTF-8, avšak na to sa nedá spoliehať. Predvolené kódovanie pre HTTP protokol je ISO-8859-1. Preto je nutné čítať položku `charset`, v ktorej je uvedené použité kódovanie. Na základe neho je následne nutné previesť prenášaný obsah na text.

HTTP protokol podporuje kompresiu prenášaného obzahu napríklad pomocou gzip. To by však znamenalo, že by ICAP server musel obsah dekomprimovať predtým, než by ho analyzoval a navyše by musel podporovať všetky režimy kompresie. Preto som do konfiguračného súboru Squidu pridal riadok `request_header_access Accept-Encoding deny all`, ktorý zjednodušene povedané spôsobí, že obsah nebude komprimovaný. Neskôr však plánujem implemetovať dekompresiu, aby bol prenos rýchlejší.

4.24 Odolnosť systému voči útokom

Jedným z mojich cieľov je, aby bol systém odolný voči možným útokom. Preto som sa na bezpečnosť snažil myslieť už od návrhu. Komunikácia medzi klientom je zabezpečená protokolom TLS. Server je autentikovaný pomocou certifikátu, klient pomocou 128 bitovej hodnoty UUID, čo zamedzí MITM útokom, ktoré by sa snažili komunikáciu odpočúvať a zároveň neumožní pripojenie klienta k falošnému serveru a falošného klienta k serveru.

Systém je odolný voči ARP spoofingu vďaka použitiu statických ARP záznamov pri priradení adresy DHCP serverom.

Keďže systém vykonáva akcie na základe ARP a DHCP komunikácie, bolo by možné vykonať DoS útok na sýstém tak, že vygenerujeme veľké množstvo paketov týchto protokolov a systém ho nebude schopný spracovať. To server rieši pomocou triedy `io.reactovex.Flowable` (z knižnice RxJava 2), ktorá umožňuje bufferovanie udalostí, a v prípade, že ich producent generuje príliš mnoho a konzument ich nestíha spracovávať, udalosti sa podľa definovanej stratégie začnú zahadzovať.

Procesy spúšťané serverom sú spúšťané pomocou volania `exec()`, ktorému sú priamo predané parametre (nie cez shell), a tak nemôže dôjsť ku tzv. „command injection“.

ICAP server počúva na adrese 127.0.0.1, takže je k nemu možné pristúpiť iba z lokálneho zariadenia. Vďaka tomu nie je možné sa k nemu pripojiť z iných zariadení a tak naň vykonať napríklad DoS útok.

4.25 Klient

Android aplikácia, ktorá funguje ako klient, podporuje všetky popísané operácie. Je to však stále iba prototyp, obsahuje množstvo debug výpisov a vyžaduje si UI a UX zmeny na to, aby mohla byť publikovaná. Avšak cieľom tejto práce nebolo vytvoriť peknú a príjemnú aplikáciu, ale funkčný systém, ktorý beží `za ňou` a ktorý jej poskytne potrebné komunikačné rozhranie na to, aby mohla byť čo najjednoduchšia a najpríjemnejšia pre užívateľa. Najzaujímavejšie triedy:

ServerConnector Trieda, ktorá má za úlohu vyhľadávať server a pripájať sa k nemu.

ServerConnectableActivity Aktivita, od ktorej dedia ostatné aktivity, ktoré sa potrebujú pripájať k serveru.

SortedAndFiltredListAdapter Trieda, ktorá slúži na vytváranie zoradeného a filtrovaného scrollovateľného zoznamu, do ktorého je možné „za živa“ pridávať položky. Aplikuje sa napríklad pri zobrazení WiFi zariadení v okolí (zariadenia sú zoradené podľa sily signálu).

4.26 Cena hardware

Tu sú približné ceny hardware komponentov, ktoré sú potrebné na nasadenie systému:

RPi 3 1000 Kč

Obal na RPi 3 100 Kč

4 GB MicroSD karta 150 Kč

USB adaptér 200 Kč

Ethernetový kábel 20 Kč

Celková cena sa teda pohybuje okolo 1470 Kč. Avšak ceny komponentov, ktoré som uviedol nie sú zďaleka najnižšie na trhu. Ak by boli komponenty nakúpené za veľkoobchodné ceny, mohla by sa cena systému rapídne znížiť. Rovnako by tomu pomohlo aj prejsť z platformy RPi 3 na vlastný hardware. Takto by celková cena podľa môjho názoru klesla hlboko pod 1000 Kč, avšak vyžadovalo by si to výrobu v obrovského množstva dosiek. Výhodou by bolo aj to, že by bolo možné prispôsobiť si hardware podľa svojich požiadaviek.

Kapitola 5

Testovanie

Systém bol počas vývoja podrobený množstvu testov. V tejto kapitole sú popísané najdôležitejšie z nich.

5.1 Detekcia útokov

Pri testovaní detekčných schopností systému som sa zameral najmä na útoky, ktoré nerieši IDS systém Suricata, ale ich detekciu som implementoval sám (ARP spoofing a falošný DHCP server). Vychádzam z predpokladu, že IDS systém bol dôkladne otestovaný jeho vývojármi.

Detekciu ARP spoofingu som testoval pomocou programu `arpspoof`, ktorý slúži na vykonávanie tohto útoku. Použitý príkaz vyzeral nasledovne:

```
arpspoof -i wlan0 -t 10.9.8.109 -r 10.9.8.1
```

IP adresa 10.9.8.1 je brána – náš systém – a adresa 10.9.8.109 je zariadenie, ktorého dátový tok chceme presmerovať. Tento príkaz začne vysielať nevyžiadané ARP odpovede obom IP adresám so sfalšovanou „is-at“ MAC adresou. To spôsobí, že zariadenia (ak nemajú implementovanú ochranu voči tomuto útoku) na základe týchto odpovedí aktualizujú svoju ARP tabuľku (keďže protokol ARP je bezstavový[2]), čo som si experimentálne overil na Linuxe a Androide. Test bol úspešný a systém naozaj tieto pakety zachytil a zaregistroval udalosť ako ARP spoofing útok. Zároveň sa systém týmto útokom nenechal zmiasť, lebo používa statické ARP záznamy.

Detekciu falošného DHCP servera som otestoval tak, že som na sieti spustil ďalší DHCP server. Tento test bol tiež úspešný a systém zaregistroval udalosť ako DHCP útok.

Tiež som testoval, či je systém schopný zachytiť zariadenie, ktoré si priradilo IP adresu samo a snaží sa obísť systém. Na počítači som nastavil IP adresu z pôvodného rozsahu siete a ako GW som zvolil domáci router. Pri zahájení komunikácie systém okamžite zaznmenal ARP dotaz a zaregistroval túto udalosť, čo sa ihneď prejavilo v klientskej aplikácií.

5.2 Rýchlosť

Raspberry Pi 3 má 100 Mbit/s ethernetové rozhranie, takže maximálna prenosová rýchlosť je cca 10 MB/s. Úzkym hrdlom systému je toto rozhranie. Ďalším prvkom, ktorý môže spomalovať rýchloť prenosu je IDS systém, keďže permanentne monitoruje dátový tok a porovnáva ho so všetkými signatúrami, ktoré má nadefinované. Rýchlosť som testoval prenášaním toku náhodných dát na porte 80, keďže podľa môjho názoru je tento port pomerne široko obsiahnutý signatúrami IDS systému a tak by mal byť prenos spomalovaný v maximálnej miere.

Výsledok testu je, že prenosová rýchlosť kolíše na hodnote okolo 8,2 MB/s a IDS systém používal okolo 200 % (2 jadrá) procesora. Ďalej bolo prvé jadro procesora plne zaťažené procesom jadra `softirqd`, ktorý má na starosti spracovanie prerušení. Tie sú spracovávané prvým jadrom procesora a to je dôvod, prečo je toto jadro plne zaťažené. Po vypnutí IDS systému vzrástla rýchlosť na 9,8 MB/s.

Test dopadol úspešne. Rýchlosť poklesla len minimálne – necelých 20 %, čo je s prihliadnutím na výkon Raspberry Pi 3 podľa môjho názoru slušné. Avšak pre niektorých užívateľov to nemusí byť postačujúce a preto by bolo vhodné použiť silnejší hardware s 1 Gbit/s rozhraním a na rozdelenie prerušení medzi jadrá by pomohol nástroj `irqbalance`.

Ďalej som testoval rýchlosť spracovania dát ICAP serverom. Na lokálnej sieti som vytvoril webový server, s veľkosťou stránky 300 KB, čo je bežná veľkosť. Ďalej som sa k tomuto serveru pripájal s vypnutým webovým filtrovaním a následne so zapnutým. Použil som slovník s 2000 náhodnými slovami. Na meranie rýchlosti som použil nasledujúci príkaz:

```
time curl -q 192.168.0.1/index.html &> /dev/null
```

Pri vypnutom filtrovaní bol výstup nasledovný:

```
real 0m0,222s
user 0m0,033s
sys  0m0,005s
```

Zaujíma nás čas *real*, ktorý odzrkadľuje čas od začiatku do konca behu programu.

Po spustení filtrovania sa výstup zmenil nasledovne:

```
real 0m0,590s
user 0m0,036s
sys  0m0,000s
```

Ukázalo sa, že čas vzrástol približne trojnásobne. Vybral som slová, pri ktorých som si bol istý, že sa nenachádzajú v danej stránke. Ak som použil slová, ktoré sa v stránke nachádzajú vo vysokej miere, čas stúpol rapídne:

```
real 0m2,465s
user 0m0,014s
sys  0m0,009s
```

To je spôsobené najmä spôsobom, ako ukladám pozitívne zhody do dátových štruktúr. Tento spôsob je pomerne výpočtovo náročný. Predpokladal som však, že pozitívne zhody budú menej časté ako prípady, kedy sa na stránke nenašla žiadna zhoda a preto táto časť algoritmu nie je kritická. Neskôr by bolo vhodné tento algoritmus vylepšiť – možno napríklad rozdeliť prácu na viac jadier, keďže algoritmus pracuje sekvenčne ale dá sa paralelizovať.

Všetky testy som skúšal viacnásobne a vybral som približné stredné hodnoty výstupov.

Rýchlosť je taktiež ovplyvnená generova-ním a podpisovaním certifikátu za behu. Spôsobuje to isté spomalenie pri pripájaní k webovým serverom.

5.2.1 Kontrola webového obsahu

Testoval som aj samotnú funkčnosť filtrovania webového obsahu. Test prebiehal tak, že som si vytvoril slovník s určitými slovami a potom som navštevoval stránky, kde som buď pedpokladal alebo nepredpokladal výskyt daného slova. Následne som kontroloval výstupy, ktoré server dával aby som zistil, či sú výskyty daných slov relevantné.

Test ukázal že filtrovanie webového obsahu a ICAP server fungujú korektne, problémom však je rozpoznávanie obsahu HTTP protokolu, keďže slová sú vyhľadávané nielen v obsahu stránky ale a aj v jej kóde. Znamená to, že systém generuje hlášky, ktoré nie sú relevantné, ak dané slovo nájde v kóde stránky. To ma priviedlo k miernemu vylepšeniu regulárneho výrazu, ktorý zisťuje, či sa jedná o zhodu a či nie.

5.3 Inštalácia

Inštaláciu som testoval na čerstvo stiahnutom obraze disku Raspbian OS, ktorý som spustil na RPi 3. Test ukázal niekolko logických chýb v inštalačnom skripte, ktoré som opravil a napokon inštalácia prebehla úspešne a systém sa spustil a bol funkčný. Test ukázal, že systém sa nedokáže pripojiť k API, ktoré mu poskytuje mapovanie z MAC adresy na výrobcu. Ukázalo sa, že problém spočíva v OpenJDK 9, ktorý je pre Debian chybne nakonfigurovaný (má chybne konfigurované certifikáty) a pri pokuse o pripojenie vyhadzuje výnimku. Preto je nutné prejsť na OpenJDK 8 alebo chybu obísť manuálnou konfiguráciou certifikátov.

5.4 Využitie RAM

Využitie pamate RAM osciluje okolo 50 %. Najväčší podiel z toho tvorí Suricata – okolo 22 %, ďalej JVM v ktorom beží server – okolo 15 % a ďalej ostatné programy s využitím RAM pod 5 %. Bolo by vhodné obmedziť množstvo signatúr IDS systému, čo by znížilo využitie RAM a rovnako by vzrástla aj rýchlosť systému, avšak nie je to kritické.

5.5 Bezpečnosť

Bezpečnosť som testoval hlavne na základe bodov z 4.24. Pokšal som sa pripojiť falošným klientom, vytvoriť falošný server, ku ktorému by sa klient pripojil, skúšal som systém oklamať ARP spoofingom a vykonať DoS útok na server pomocou zahltenia ARP a DHCP komunikáciou. Systém obstál a tak bol test úspešný.

5.6 Ostrý test v sieti

Počas celého vývoja bol systém súčasťou reálnej domácej siete, čo malo výhodu, že systém sa nachádzal v reálnych podmienkach. To mi pomohlo včas odhaľovať nečakané chyby. Testoval som všetky funkcionality systému a snažil som sa nájsť ich bod zlomu, čo sa viacnásobne podarilo a viedlo to k vylepšeniu systému.

Záver

Cieľom tejto práce bolo vytvoriť bezpečnostný systém pre domácu sieť. Navrhnutý systém je schopný monitorovať dátové toky všetkých zariadení na sieti, ktoré presmeruje cez seba pomocou popísanej techniky vytvorenia vlastného DHCP serveru. Systém je schopný z týchto dát extrahovať metadáta, ktoré môžu byť v neskoršom rozvoji použité na cloudovú analýzu hrozieb.

Blokovanie IP adries a portov a monitorovanie a ukončovanie sieťových spojení je implementované pomocou frameworku Netfilter, konkrétne pomocou nástrojov `iptables` a `conntrack`. Spojenie je možné ukončiť na žiadosť administrátora alebo v prípade niektorých nebezpečných útokov systém dokáže zasiahnuť sám – ukončí spojenie a zablokuje danú zdrojovú IP adresu a cieľovú IP adresu a port na istý čas.

Systém taktiež podporuje detekciu o 802.11 bezdrôtových zariadení a poskytuje užívateľovi informácie o zariadeniach v dosahu bezdrôtového rozhrania. Systém je schopný vyslať 802.11 deautentikačné rámce zariadeniam na vlastnej sieti, a tak ich zo siete odpojiť na žiadosť administrátora.

Detekcia nových zariadení na sieti je implementovaná monitorovaním ARP protokolu v kombinácií s detekciou bezdrôtových zariadení a monitorovaním DHCP dotazov. Systém automaticky notifikuje administrátora pri objavení nového zariadenia na sieti.

Detekcia útokov je riešená pomocou IDS systému Suricata, ktorý je založený na detekcií signatúr útokov. Vďaka tomu sú detekčné schopnosti systému vo vysokej miere rozšíriteľné. Ďalej som implementoval detekciu útokov na lokálnej sieti, konkrétne ARP spoofingu a falošného DHCP servera.

Systém dokáže na žiadosť či periodicky scanovať jednotlivé zariadenia a zbierať informácie o nich. Administrátor si tak môže zobraziť zoznam zariadení na jeho sieti s popisom jednotlivých zariadení. To mu môže pomôcť identifikovať jednotlivé zariadenia, najmä ak používa rôzne IoT zariadenia, ktorých môže byť na sieti veľké množstvo. V kombinácií s monitorovaním sieťových pripojení tak môže identifikovať zariadenia, ktoré sa potenciálne

pripájajú k serverom tretích strán a poskytujú im informácie o užívateľovi alebo v kombinácií so zabudovaným IDS systémom môže identifikovať ktoré z jeho zariadení je napríklad „nakazené“ malwarom.

Všetky tieto programy sú ovládané serverovou aplikáciou bežiacou v JVM, ktorá odpovedá na požiadavky klienta, či vykonáva časovo naplánované akcie alebo odpovedá na sieťové udalosti, ktoré boli zaznamenané.

Systém je administrovaný pomocou klienta ktorým je Android aplikácia, avšak jedná sa iba o prototyp a bolo by vhodné ho vylepšiť. Do budúcnosti by bolo vhodné vytvoriť aj multiplatformné administračné rozhranie, napríklad webové rozhranie.

Z licenčných dôvodov systém zatiaľ nie je vydávaný ako obraz disku, ale ako inštalátor. Plánujem však detailne preskúmať licenčné možnosti OS Raspbian a, ak to bude možné, distribuovať systém spolu s OS ako obraz disku (neskôr možno spolu s hardware), čo bude jednoduchšie na nasadenie pre bežných užívateľov.

Systém je nasaditeľný v praxi v každej domácej sieti, kde postačuje rýchlosť 100 Mbit/s. Užívateľ však musí byť schopný sa dostať do nastavení svojho routru, kde je nutné vypnúť DHCP server, aby systém mohol prevziať kontrolu nad sieťou a spustiť vlastný DHCP server. Následne je systém plne funkčný a pripravený monitorovať dátový tok. Rozsah pokrytia bezpečnostných hrozieb prevyšuje schopnosti popísaných komerčných riešení a rovnako aj cena je výrazne nižšia. Niektore funkcie systému, ako napríklad kontrola obsahu HTTPS protokolu (ktorú som v žiadnom existujúcom systéme nenašiel), výrazne prevyšujú funkcionalitu komerčných riešení.

Výhľadom do budúcna by bolo vytvorenie inštalátora pre rôzne distribúcie OS Linux, aby bolo možné systém nasadiť na výkonnejšie zariadenia. Ďalej by bolo dobré vylepšiť odolnosť súborového systému voči chybám napríklad pri výpadku elektrického prúdu a taktiež zdokonaliť perzistenciu dát, zdokonaliť komunikačný protokol, implementovať ďalšie režimy operácie ako napríklad náhrada routru (zariadenie by nahradilo domáci router a fungovalo by aj ako WiFi prístupový bod) a určite by bolo vhodné použiť strojové učenie spojené s cloudom na zvýšenie detekčných schopností systému. Ďalej plánujem prispôsobiť signatúry IDS systému Suricata pre potreby domácej siete (čo odľahčí využitie RAM), implementovať detekciu vírusov v sťahovaných súboroch, zefektívniť ICAP serveru (a extrahovať ho do samostatného balíčka, ktorý by som chcel zverejniť, pretože som nenašiel existujúcu implementáciu ICAP serveru pre Javu), vytvoriť silné, viacjazyčné slovníky na detekciu nevhodného webového obsahu a v neposlednom rade vylepšiť klientskú aplikáciu z UI a UX hľadiska, aby bolo možné v plnej miere využiť sunkcionalitu poskytovanú serverom.

Pri tvojbe tejto práce som sa osobne výrazne obohatil poznatkami a skúsenosťami z oblastí konfigurácie modulov Netfiltru pomocou nástroja Iptables, Squid proxy serveru, Suricata IDS, Dnsmasq DHCP a DNS serveru a implementácie ICAP serveru. V neposlednom rade som sa výrazne zdoko-

nalil v programovaní v jazyku Java (aj pre Android) a používaní frameworku RxJava 2.

Bibliografia

1. UHLMANN, Stephan. *ARP cache poisoning / ARP spoofing* [online]. 2003. Dostupné tiež z: http://su2.info/doc/arpspoof.php [cit. 2018-4-16].
2. JAJODIA, S.; MAZUMDAR, C. *Information Systems Security: First International conference, ICISS 2005, Kolkata, India, December 19-21, 2005, Proceedings*. Springer Berlin Heidelberg, 2005. Lecture Notes in Computer Science. ISBN 9783540324225. Dostupné tiež z: https://books.google.cz/books?id=biwGCAAAQBAJ.
3. MUKHTAR, Husameldin; SALAH, Khaled; IRAQI, Youssef. Mitigation of DHCP starvation attack. *Computers and Electrical Engineering*. 2012, roč. 38, č. 5, s. 1115–1128. ISSN 0045-7906. Dostupné z DOI: https://doi.org/10.1016/j.compeleceng.2012.06.005. Special issue on Recent Advances in Security and Privacy in Distributed Communications and Image processing.
4. *Wifiphisher: Automated phishing attacks against Wi-Fi networks* [online]. Dostupné tiež z: https://wifiphisher.org/docs.html [cit. 2018-4-16].
5. VITELLI, Romeo. Can Compulsive Internet Use Affect Adolescent Mental Health? [online]. 2016. Dostupné tiež z: https://www.psychologytoday.com/us/blog/media-spotlight/201601/can-compulsive-internet-use-affect-adolescent-mental-health. [cit. 2018-5-8].
6. KANG, Shimi. Online Pornography and Young Minds [online]. 2016. Dostupné tiež z: https://www.psychologytoday.com/intl/blog/the-dolphin-way/201610/online-pornography-and-young-minds. [cit. 2018-5-8].
7. ROSS, Carolyn. Overexposed and Under-Prepared: The Effects of Early Exposure to Sexual Content [online]. 2012. Dostupné tiež z: https://www.psychologytoday.com/us/blog/real-healing/201208/

overexposed-and-under-prepared-the-effects-early-exposure-sexual-content. [cit. 2018-5-8].

8. IOT DEFENSE, INC. *Frequently asked questions* [online]. 2017. Dostupné tiež z: https://www.myrattrap.com/faq [cit. 2018-4-16].
9. IOT DEFENSE, INC. *Technology* [online]. 2017. Dostupné tiež z: https://www.myrattrap.com/technology [cit. 2018-4-16].
10. CRUZ, Bethuel. *CUJO vs DOJO vs KEEZEL vs RATTRAP vs AKITA vs BITDEFENDER BOX 2* [online]. 2018. Dostupné tiež z: https://homealarmreport.com/cujo-dojo-vs-keezel [cit. 2018-4-16].
11. CUJO LLC. *How it works* [online]. 2018. Dostupné tiež z: https://www.getcujo.com/how-it-works/ [cit. 2018-4-16].
12. *Dojo - FAQ* [online]. Dostupné tiež z: https://dojo.bullguard.com/dojo-by-bullguard/faq/ [cit. 2018-4-16].
13. VIOLET. *Fingbox vs Smart Firewalls : CUJO, Bitdefender Box, Dojo, Disney's Circle* [online]. 2018. Dostupné tiež z: https://www.fing.io/support/fingbox-vs-firewalls-cujo-bitdefender-box-dojo-disneys-circle/ [cit. 2018-4-16].
14. CROSSTALK SOLUTIONS. *Fingbox* [online]. 2017. Dostupné tiež z: https://www.youtube.com/watch?v=4Hc3YGsaW8U [cit. 2018-4-16].
15. KOALASAFE, INC. *How it works* [online]. 2017. Dostupné tiež z: https://koalasafe.com/how-it-works.html [cit. 2018-4-16].
16. *FalconGate* [online]. 2018. Dostupné tiež z: https://raw.githubusercontent.com/A3sal0n/FalconGate/master/README.md. [cit. 2018-4-16].
17. ROESCH, Martin. *Snort: open source network intrusion detection system* [online]. 2011. Dostupné tiež z: http://www.manpagez.com/man/8/snort/. [cit. 2018-4-16].
18. SCHREIBER, Joe. *Open Source Intrusion Detection Tools: A Quick Overview* [online]. 2014. Dostupné tiež z: https://www.alienvault.com/blogs/security-essentials/open-source-intrusion-detection-tools-a-quick-overview [cit. 2018-4-16].
19. *All features* [online]. 2018. Dostupné tiež z: https://suricata-ids.org/features/all-features/ [cit. 2018-4-16].
20. THE BRO PROJECT. *Open Source Intrusion Detection Tools: A Quick Overview* [online]. 2018. Dostupné tiež z: https://www.bro.org/sphinx/intro/ [cit. 2018-4-16].
21. ROUSSKOV, Alex. *Feature: Mimic original SSL server certificate when bumping traffic* [online]. 2014. Dostupné tiež z: https://wiki.squid-cache.org/Features/MimicSslServerCert [cit. 2018-4-16].

22. *What is Squid?* [online]. Dostupné tiež z: `http://www.squid-cache.org/Intro/` [cit. 2018-4-16].

23. VOINOV, Yuri. *Block QUIC protocol* [online]. 2016. Dostupné tiež z: `https://wiki.squid-cache.org/KnowledgeBase/Block%20QUIC%20protocol` [cit. 2018-4-16].

Dodatok **A**

Zoznam použitých skratiek

ARP Address Resolution Protocol

CPU Procesor

DG Default gateway

DHCP Dynamic Host Configuration Protocol

DNS Domain Name System

GW Gateway

HTTP Hypertext Transfer Protocol

HTTPS HTTP Secure

ICAP Internet Content Adaptation Protocol

IDS Intrusion Detection System

IoT Internet of Things

IP Internet Protocol

IPS Intrusion Prevention System

JIT Just In Time

JVM Java Virtual Machine

MAC Media Access Control

MITM Man In the Middle

ML Machine Learning

NAT Network Address Translation

OS Operačný Systém

RAM Random Access Memory

RPi Raspberry Pi

TLS Transport Layer Security

UPnP Universal Plug and Play

URL Uniform Resource Locator

VM Virtual Machine

Dodatok **B**

Obsah priloženého CD

- `README`
- `INSTALL` ..... pokyny pre inštaláciu a zostavenie
- `bin` ..... adresár s inštalačnými súbormi
- `src` ..... adresár so zdrojovými kódmi
  - `libs` ..... knižnica spoločných tried pre klienta a server
  - `app` ..... android aplikácia
  - `server` ..... serverová časť
- `text` ..... text práce
  - `thesis.pdf` ..... text práce vo formáte PDF
  - `thesis.tex` ..... zdrojová forma práce vo formáte LaTeX