LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

TĪKLA PLŪSMU MONITORINGA SISTĒMAS IZSTRĀDE

BAKALAURA DARBS

Autors: **Guntars Pužulis**
Studenta apliecības Nr.:gp13022
Darba vadītājs: Dr. dat., docents, Leo Trukšāns

RĪGA 2017

ANOTĀCIJA

Darba mērķis ir izpētīt veidus, kā var savākt tīkla plūsmu datus priekš analīzes, metodes ar kādām analizēt datus, izpētīt cik daudz informācijas var iegūt analizējot savāktos datus, izmantojot uz lietotāju paradumu bāzētu pieeju.

Darba praktiskajā daļā tika izveidota sistēma, kas automātiski savāc datus un veic analīzi savāktajiem datiem. Tika izveidota privāta vietne, kur var apskatīties pārraudzības datus.

Darba rezultātā tika sasniegti izvirzītie mērķi, kā arī izveidota sistēma, kas palīdz noteikt lietotāju paradumus, interneta izmantojumu.

Atslēgvārdi: paradumu analīze, pārraudzība, sistēma, tīkla plūsma

ABSTRACT

Development of network flow monitoring system

The aim of this work is to research possibilities to get network flow data for analysis, research possibilities in data analysis, explore how much information can be obtained using behavior based approach.

System that can automatically collect and analyze data, has been made in practical part of this work. Private web page has been made to access monitored data.

As result of this work, introduced objectives have been achieved and system, which helps to analyze user behavior of internet usage, has been made.

Key words: behavior, monitoring, network flow, system

SATURS

Apzīmējumu saraksts

Multicast - Ziņojumu pārraides veids datoru tīklos, kad viens un tas pats ziņojums vienlaicīgi tiek nosūtīts vairākām, bet ne visām datoru tīkla stacijām.

Tīkla plūsma – paku sekvence no mērķa datora uz galamērķa datoru, kas var būt arī cits resursdators, multicast grupa vai apraides domēns.

Tīkla tapa *(Network TAP)* – Specializēta ierīce, ar kuras palīdzību var piekļūt datiem, kas plūst pa tīklu. Tā kopē tīkla datus un nosūta tos citai sistēmai, lai tā varētu tos, piemēram, analizēt, pārsūtīt tālāk.

VLAN *(virtuālais LAN)* - jebkurš apraides domēns, kas ir sadalīts un izolēts datortīklā un datu savienojuma slānī (*OSI 2*. slānī).

SPAN – Portu spoguļattēls *Cisco* sistēmās – Maršrutētāja Portu Analizators (*Switch Port Analyzer*). Tiek lietots, lai maršrutētājos pārsūtītu datu plūsmas kopiju no viena porta (vai visa *VLAN*) uz citu tīkla monitoringa savienojumu vai citu maršrutētāju. Citiem ražotājiem ir savādāki nosaukumi portu spoguļattēla veidošanai, piemēram *Roving Analysis Port – RAP* priekš 3com maršrutētājiem

RTT *(round-trip time)* – laika sprīdis, kas nepieciešams signālam, lai to aizsūtītu un saņemtu ziņojumu, ka tas ticis saņemts.

SNMP – Interneta standarta protokols priekš datu savākšanas no monitorējamajām iekārtām.

RMON – ir standarta monitorēšanas specifikācija, kas iekļauj dažādus tīkla novērošanas un consoļu sistēmas, lai mainītos ar tīkla monitorējamajiem datiem.

Netflow – funkcija, kas tika ieviesta *Cisco* rūteros, kas piedāvā iespēju savākt IP tīkla datu plūsmas datus, kad tie ienāk vai iziet no interfeisa.

NMS *(network monitoring system)* – sistēma, kas patstāvīgi novēro datortīklu priekš lēnām ierīcēm vai komponentēm kuru kļūdu parādīšanās ir būtiska un par to paziņo tīkla administratoram, ja notiek kādas kļūdas.

IEVADS

Mūsdienās internets arvien biežāk tiek izmantots, lai sazinātos, veiktu darījumus, skatītos jaunumus, ziņas un citas lietas. Tas ir nozīmīgs rīks, gan brīvajā laikā, gan arī darbā. Interneta lietojumam palielinoties, palielinās gan cilvēku skaits, kas ir iesaistīti datu iegūšanā un izmantošanā, gan to kas ir iesaistīti interneta un datu drošības risku radīšanā. Gadu no gada uzbrukumu veidi attīstās un pilnveidojas, tāpēc ir svarīgi pilnveidoties līdzi un veidot ar vien jaunus un labākus veidus kā sekot līdzi tam, kas notiek internetā.

Tā kā interneta lietojums arvien pieaug, palielinās arī nepieciešamība apstrādāt arvien lielākus datu apjomus.

Tiek izstrādāti dažādi monitoringa risinājumi, kas samazina drošības riskus, taču mainoties interneta tehnoloģijām, rodas arvien jauni ievainojamību veidi.

Darba mērķis ir izpētīt kā savākt tīkla plūsmu datus un veidus kā tos analizēt.

Darba uzdevumi mērķa sasniegšanai izstrādāt sistēmu, kas ļauj monitorēt tīkla plūsmas un tīkla lietotāju paradumus.

Pētījuma metodes:

- Aprakstošā jeb monogrāfiskā

  Literatūras analīze, lai izpētītu tīkla plūsmu monitoringa iespējas un eksistējošos risinājumus paradumu (anomāliju) noteikšanā.

- Salīdzinošā jeb komparatīvā

  Eksistējošu risinājumu analīze un jaunās anomāliju bāzētās tīkla sistēmas metodoloģijas sintēze.

- Modelēšana un imitācija

  Jaunās tīkla plūsmu monitoringa sistēmas pārbaude

1. TEORĒTISKAIS APSKATS

Literatūrā ir pieejami samērā daudz pētījumi kā savākt datus un pārraudzīt tīkla plūsmas, izmantojot dažādus tīkla protokolus, tīkla topoloģijas, ierīču funkcionalitātes.

Taču katram no aprakstītajiem datu ievākšanas veidiem un esošajām monitoringa sistēmām ir gan savas priekšrocības gan trūkumi.

1.1. Datu savākšana izmantojot dažādus tīkla protokolus

Literatūrā ir aplūkoti veidi, kā var iegūt datus izmantojot specifiskus tīkla protokolus, kas domāti tīkla monitorēšanai un izvērtēšanai.

1.1.1. Vienkāršais tīkla pārvaldības protokols

Visbiežāk tiek izmantots "Vienkāršais tīkla pārvaldības protokols (*SNMP; Simple Network Management Protocol*). Tas ir interneta standarta protokols, lai pārvaldītu ierīces *IP*

tīklos. Ierīces, kas atbalsta *SNMP*, ietver maršrutētājus, komutatorus, serverus, darba stacijas,printerus" [19].

Mūssdienās tas ir pats populārākais dažādu komerciālo, universitāšu un pētniecisko iestāžu apvienoto tīklu protokols. *SNMP* ir samērā vienkāršs protokols, taču tā raksturojumu kopa ir pietiekami spēcīga dažādu problēmu risināšanai utt. " [19].

"Tas ir lietojuma slāņa protokols, kurš ir paredzēts lai atvieglotu pārvaldības informācijas apmaiņu starp tīkla iekārtām. tīkla pārvaldības protokolus parasti izmanto lai administrators varētu noteikt pakešu skaitu sekundē un tīkla kļūmju koeficientu, tādējādi administratori spēj vieglāk pārvaldīt tīkla veiktspēju, konstatēt un risināt problēmas. *SNMP* aģenti ir programmas vai aparatūras moduļi, kuri strādā vadāmajās iekārtās, vāc informāciju par vadāmajām iekārtām, kurās tie strādā, un padara šo informāciju pieejamu tīklu pārvaldības sistēmām (*network management system - NMS*) ar *SNMP* protokola palīdzību" [19].

"Operācijas - *SNMP* ir vienkāršais pieprasījuma/atbildes protokols. Ir noteiktas sekojošas četras *SNMP* operācijas: *Get* (saņem), *Get-next* (saņem nākamo), *Set* (uzstādi),*Trap* (slazds)" [19].

Šī protokola pielietojumi: Lielā hardonu paātrinātāja *NMS, Zabbix, PRTG, Solarwinds, Nagios* un daudzi citi [16,14,15,17,18].

1.1.2. Attālinātā monitorēšana

Attālināta monitorēšana *Remote Monitoring (RMON).* "Attālinātās tīkla monitoringa ierīces ir instrumenti, kas eksistē, lai varētu pārraudzīt / pārvaldīt tīklu. Bieži vien attālinātās ierīces ir atsevišķas ierīces un velta ievērojamus iekšējos resursus priekš vienīgā nolūka - tīkla pārvaldīšanas. Organizācija var izmantot vairākas šādas ierīces, līdz vienai ierīces vienā tīkla segmentā, lai pārvaldītu tā internetu. Piedevām šādas ierīces var tikt izmantotas ģeogrāfiski attālinātos tīklos, kā piemēram priekš operatora tīkla pārvaldības atbalsta centr, lai pārvaldītu klientu tīklu vai priekš centralizētas atbalsta organizācijas, priekš uzņēmuma, lai pārvaldītu attālinātu vietu" [20].

"Mūsdienās, daži no jaunākajiem *RMON* ietvariem atļauj vairākas tehnikas kā ievākt datus, kur visu paku savākšana ir tikai viens no daudzajiem variantiem.

Daži no *RMON* mērķiem:

- Bezsaites operācija

RMON atļauj ierīci konfigurēt, lai veiktu diagnostiku un savāktu statistiku nepārtraukti, pat ja savienojums ar pārvaldības staciju var nebūt iespējams vai efektīvs. Tātad pat tad, ja savienojums ar vadības staciju un ziņotāju nav patstāvīgs, kļūdas, veiktspēja un konfigurācijas informācija var tikt patstāvīgi sakrāta un sazināta ar vadības staciju pārliecinoši un efektīvi" [20].

- Aktīvā monitorēšana

"Ņemot vērā resursus uz monitora, tas ir potenciāli noderīgi priekš patstāvīgas diagnostikas veikšanas un tīkla veiktspējas žurnalēšanas [20].

- Datu pievienotā vērtība

Piemēram izcelt tos resursdatorus tīklā, kuri ģenerē visvairāk datu pārraides kļūdas, datu savācēj ierīce var dot vadības stacijai precīzu informāciju, kas nepieciešama lai atrisinātu problēmu klases" [20].

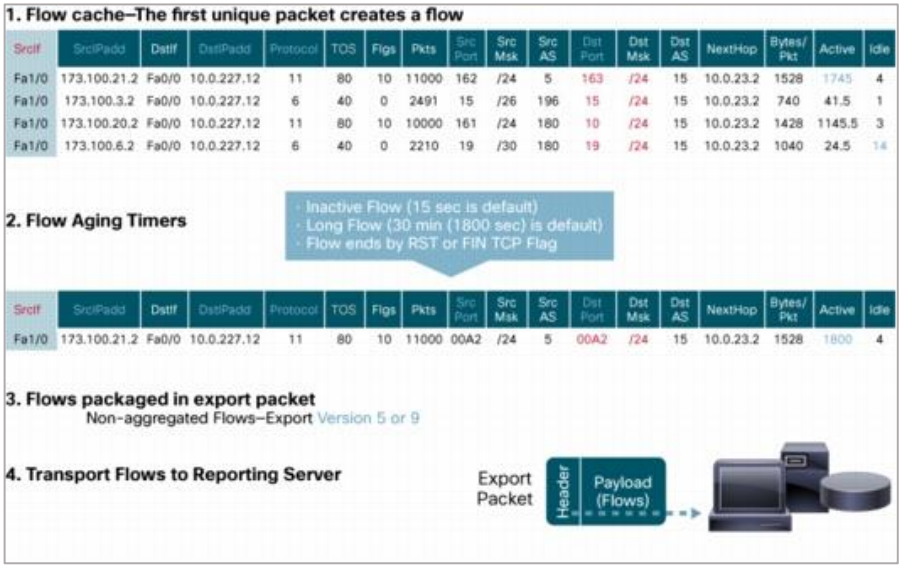
1.1.3. *Netflow* protokols

Netflow ir funkcija, "kas tika ieviesta *Cisco* rūteros, kas piedāvā savākt *IP* tīkla datu plūsmu, kad tas ienāk vai iziet no interfeisa. Analizējot *Netflow* datus, tīkla administratori var

noteikt avota un galamērķa datu plūsmas, servisa klasi un pārslogojuma cēloņus. Tipiska monitorēšanas montāža (*Netflow*) sastāv no trīs galvenajām komponentēm:

- Tīkla plūsmas eksportēšana: tīkla pakas tiek apkopotas tīkla plūsmās un tiek pārraidītas uz plūsmu savācēj ierīci;
- Plūsmu savākšana: atbildīga par plūsmu datu uzņemšanu, glabāšanu un priekšapstrādi, kas saņemta no plūsmu eksportēšanas;
- Analīzes lietotne: analizē saņemto plūsmu datus piemēram ievainojamību atklāšanā, datplūsmas veiktspējā ” [25].

Netflow ”ļoti līdzīga principa risinājumi ir arī citiem ražotājiem kā piemēram: *NetStream, Cflowd, Rflow, AppFlow, Jflow, sFlow* ” [25].



1.1. att. ***Netflow*** **kešatmiņas piemērs ar savāktajiem datiem [26]**

Dažādās *netflow* versijās ir dažāda informācija, ko var iegūt no protokola lietošanas. Attēlā 1.1. redzams, ka tiek savākta informācija par avota, mērķa adresēm, protokola identifikatoru, pakešu skaitu, pārsūtīto apjomu, aktīvo laiku un citām vienībām. Tāpat arī redzams, ka ir tīkla datu plūsmas pārtraukumu uzstādīšana plūsmām.

1.1.4. Izvērtējums

Visi no iepriekš aplūkotajiem risinājumiem ir labi, bet katram ir savs pielietojums. Autors neizvēlējās nevienu no šiem risinājumiem kā par pamatu tīkla plūsmu monitoringa sistēmas

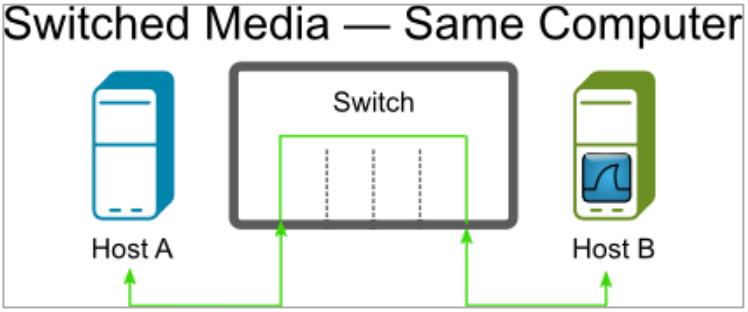
datu ievākšanas avotam. Tam iemesls bija, ka Autora sistēmu bija plānots papildināt ar funkcionalitāti, kas vērtē / analizē pašus pakešdatus. Līdz ar to neviens no šiem risinājumiem nav īsti piemērots. Šie risinājumi balstās uz to, ka tiek sūtīta maza informācijas porcija par apkopotiem datiem, piemēram, skaitītāji. No šādiem datiem viennozīmīgi var saprast tīkla kopējo situāciju un šīs pieejas iedvesmoja Autora risinājumu, datu kopsavilkuma veidošanā.

1.2. Datu savākšana izmantojot dažādas tīkla topoloģijas un tīkla ierīču funkcionalitāti

Priekšrocības un trūkumi tiek rakstīti ar domu, ka tiek apskatīts scenārijs piekļūt visiem lietotāja datoriem, līdz ar to, piemēram, par tīkla tapām tiek uzskatīts, ka tiek izvietotas starp klienta ierīci un maršrutētāju vai rūteri, nevis starp rūteri un maršrutētāju vai kā savādāk, ko protams ir iespējams izdarīt. Tas tiek darīts ar nolūku redzēt visus datus, jo savādāk nevarētu redzēt maršrutētājam pieslēgto ierīču savstarpējās tīkla plūsmas. Šie risinājumi ir labi ar to, ka var iegūt patvaļīgu informāciju par tīkla lietojumu, pretēji no, piemēram, SNMP pieejas, kur tiek definēts, kas tiek iegūts un sūtīts tikai tas, kas ir vajadzīgs.

1.2.1. Uz klienta ierīces bāzētā pieeja

Uz klienta ierīces bāzēta pieeja nozīmē to, monitoringa sistēma tiek uzstādīta uz klienta ierīces ar ko tiek savākti tīkla plūsmu dati un apstrādāti (skatīt 1.2. attēlu). Un tad pastāvīgi ik pēc kāda laika savāktā informācija tiek sūtīta uz sistēmu, kas visus klienta datorus apkopo.



1.2. att. **Tīkla plūsmu datu savākšana uz lietotāja datora [4]**

Apkopojot pieejas priekšrocības un trūkumus (skatīt 1.1. tabulu) šāds risinājums der sistēmām, kurām vajadzīga detalizēta datu analīze, vai arī nepieciešams monitorēt lielu datortīklu ar lielu datu plūsmu. Šādā veidā datu apstrāde notiek decentralizēti un nav vājās vietas, lai apstrādātu lielu kopējo datu plūsmu, ja to vēlas uzkrāt vienā vietā. Šāds risinājums ir

plaši izmantots tīkla monitoringa sistēmās kā *PRTG, Solarwinds, Zabbix* un daudzās citās [14,15,17].

1.1. tabula

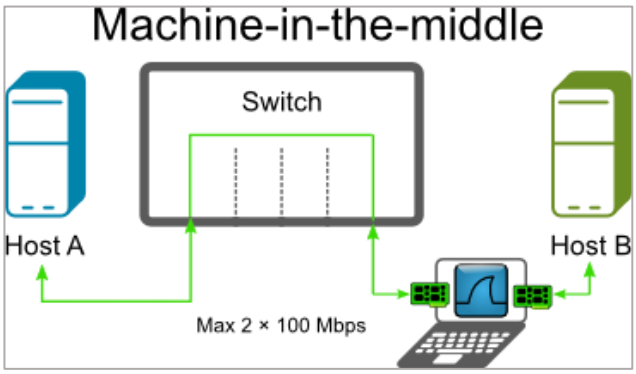
Uz klienta ierīces bāzētas datu savākšanas pieejas priekšrocības un trūkumi

Priekšrocības	Trūkumi
Monitoringa sistēmai maza noslodze, būtiskā daļa datu ir tikai konkrētās ierīces	Var redzēt lielākoties datus, kas ir vērsti tikai šai ierīcei. Līdz ar to jāuzstāda katrā klienta ierīcē
Risinājums ir labs lai noteiktu tīkla darbības veiktspēju, atrastu vājās vietas, pārsūtot datus piemēram lokālajā tīklā	Nepieciešamība pārsūtīt datus sakrātos datus, līdz ar to palielināt tīkla noslodzi.
	Noslogo klienta datoru

Klients var redzēt, ka tiek savākti dati. Ja runa par klienta darba laika kontroli, tad šāds risinājums var iedrošināt klientu pilnvērtīgāk strādāt. Vai arī var iedrošināt mēģināt apiet šādu risinājumu.

1.2.2. Starpniekierīce

Risinājums ar starpniekierīci nozīmē to, ka pa vidu starp maršrutētāju un klienta datoru šajā gadījumā, tiek izvietota ierīce, kas vāc datus. Šādai ierīcei ir nepieciešamas 2 tīkla kartes (skatīt 1.3. attēlu).



1.3. att. **Tīkla plūsmu datu savākšana uz starpniekierīces [4]**

Izvērtējot šādu pieeju (skatīt 1.2. tabulu), Autors apgalvo, ka šāda pieeja ir nepraktiska pastāvīgai monitorēšanai. Šāda pieeja ir vairāk noderīga īslaicīga skaidra defekta atrašanai un

novēršanai tīklā, nevis monitoringa sistēmām, jo vajadzīgas izmaiņas tīklā un speciāla aparatūra.

1.2. tabula

Uz starpniekierīces bāzētas datu savākšanas pieejas priekšrocības un trūkumi

Priekšrocības	Trūkumi
Monitoringa sistēmai maza noslodze, būtiskā daļa datu ir tikai konkrētās ierīces	Var redzēt lielākoties datus, kas ir vērsti tikai šai ierīcei. Līdz ar to jāpielieto katrai klienta ierīcei
Risinājums ir labs lai noteiktu tīkla darbības veiktspēju, atrastu vājās vietas	Sarežģītāka uzstādīšana, nepieciešamas izmaiņas esošā tīklā
	Nepieciešams daudz speciālas aparatūras (ierīces ar 2 tīkla kartēm). Lai savāktu datus no daudziem lietotājiem

"Ierīce ar divām tīkla kartēm var tikt lietota kā caurspīdīgs tilts, kas redz visus pārsūtītos datus no vienas konkrētas ierīces vai tīkla segmenta. *Linux* sistēmās - *brctl* (*brcfg* – vecākās sistēmās) var tikt lietots. *BSD* tipa operētājsistēmās – *brconfig*. Windows operētājsistēmas arī atļauj šāda tilta uzstādīšanu" [4].

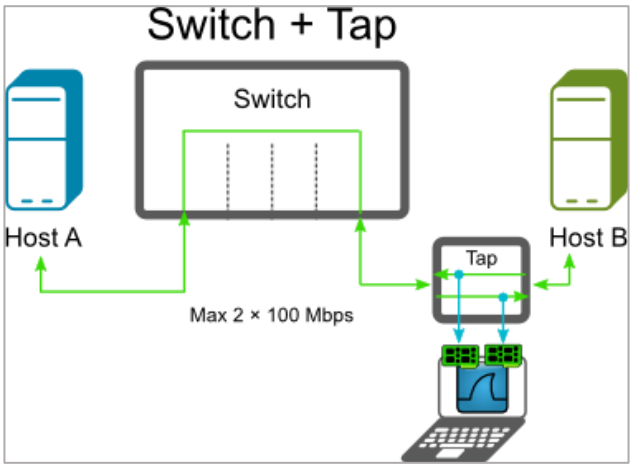
Pietiek ar to, ka dati tiek savākti uz vienas no tīkla kartēm starpniekierīcē, lai savāktu visu caurejošo datu plūsmu.

Daudzi portatīvie datori piedāvā vienu tīkla karti un otra var tikt pievienota kā *PC card*. Stacionārie datori ir viegli papildināmi ar papildus tīkla kartēm.

"Tilts ir caurredzams *IP* līmenī un līdzīgos protokolos. Un tas ir gandrīz caurredzams *Ethernet* līmenī – tas rada mazu aizturi pakešu pārraidīšanā un *Ethernet* adreses no 2 tīkla kartēm starpniekierīce var atbildēt uz daļu no apraides ziņojumiem "[4].

1.2.3. Datu savākšana izmantojot papildus ierīci – Network Tap

Risinājums, kas izmanto tīkla tapu (*Network Tap*) prasa papildus ierīci, kas tiek uzstādīta šajā gadījumā starp maršrutētāju un klienta datoru (skatīt 1.4. attēlu). Atkarībā no tā kādus režīmus tīkla tapā izmanto, ierīcei, kas pieslēgta tīkla tapai jābūt vai nu vienai vai divām tīkla kartēm.



1.4. att. **Tīkla plūsmu datu savākšana izmantojot Network TAP[4]**

Apkopojot priekšrocības un trūkumus šādai datu savākšanas pieejai (skatīt 1.3. tabulu) Autors secina, ka šāda pieeja nav ekonomiska priekš tīkla monitorēšanas sistēmas izveides. Tīkla tapas, gan ir lielisks veids liela mēroga tīkliem ar lieliem datu apjomiem, kas mērāmi Gb.

1.3. tabula

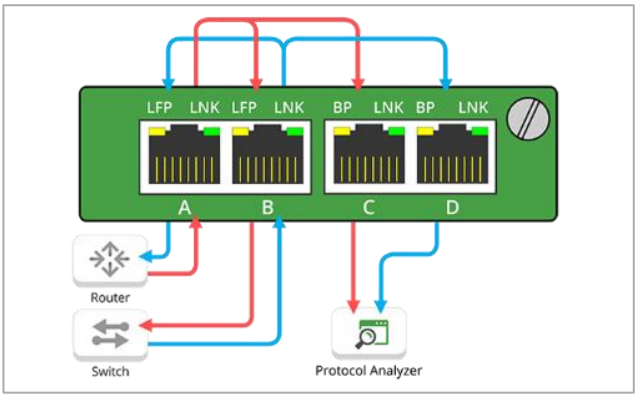
Datu savākšana pieeja izmantojot tīkla tapu priekšrocības un trūkumi

Priekšrocības	Trūkumi
Monitoringa sistēmai maza noslodze, būtiskā daļa datu ir tikai konkrētās ierīces	Var redzēt lielākoties datus, kas ir vērsti tikai šai ierīcei. Līdz ar to jāpielieto katrai klienta ierīcei
Nav riska, ka datu paketes tiks pazaudētas [7].	Nepieciešama papildus ierīce – tīkla tapa, kas ir dārgas.
Tiek savāktas visas datu paketes – arī ar datortehnikas (*hardware*) kļūdām gan MAC, gan tīkla vides kļūdām.	Nav īsti piemērota, lai novērotu šauras datu plūsmas [7].
Risinājums ir labs lai noteiktu tīkla darbības veiktspēju, atrastu vājās vietas	Nepieciešamas izmaiņas jau esošā tīkla topoloģijā

Dažādi ražotāji piedāvā tīkla tapas, kas var tikt izmantotas kā starpniekierīce, priekš tīkla plūsmu datu savākšanas. Ir dažādu veidu tīkla tapas, ierīces kurās iespējami dažādi režīmi tīkla tapām kā:

1.2.3.1. Breakout

Breakout režīms ir tāds, ka datu plūsmas A uz B un B uz A tiek nokopētas atsevišķi uz diviem dažādiem izejas portiem. Piemēram plūsma virzienā no A uz B tiek nokopēta datu plūsma uz C un plūsma no B uz A tiek nokopēta uz D (skatīt 1.5. attēlu).

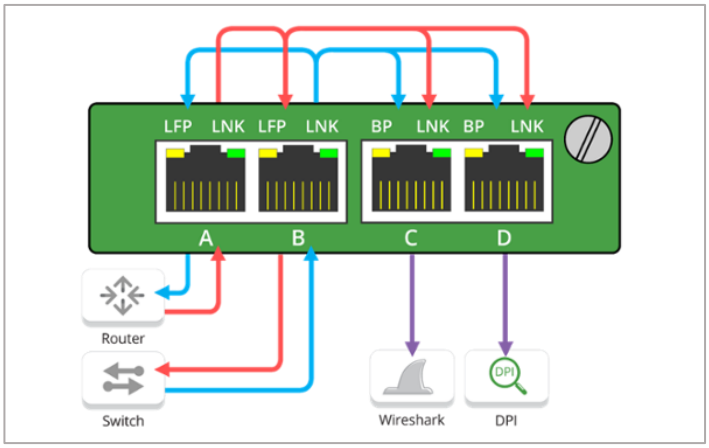


1.5. att. ***Breakout*** **režims [5]**

"Šādas veids parasti tiek lietots tad, kad pievienotie savienojumi ir pietiekami noslogoti, ja tīkla tapa tiktu pārslogota, ja tiktu apvienotas izejošās un ienākošās plūsmas kopā vienā monitoringa portā. Šāds risinājums prasa 2 tīkla kartes priekš ienākošajām un izejošajām datu plūsmām iekārtai, kurā uzstādīta monitorēšanas programmatūra" [5].

1.2.3.2. Agregācija

Agregācijas režīms ir tāds, ka tīkla tapa datu plūsmas A uz B un B uz A apkopo vienā vai vairākos izejas portos. Tas nozīmē, ka vairākām ierīcēm var viegli pieslēgt A un B kopējo datu plūsmu (skatīt 1.6. attēlu).

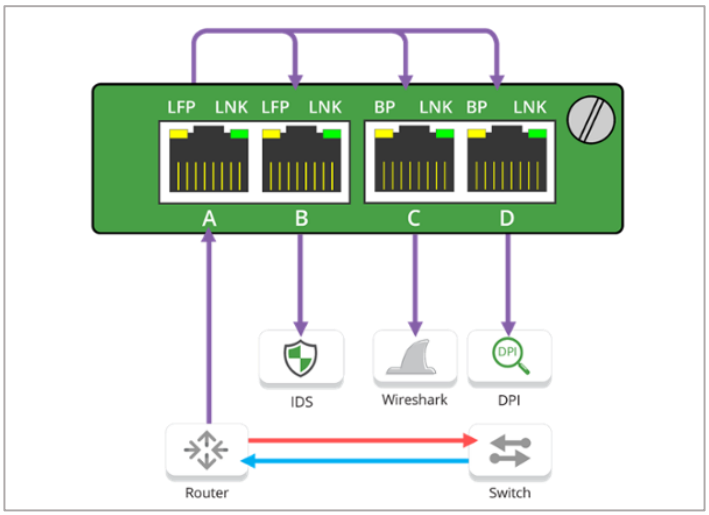


1.6. att. **Agregējošais režīms[5]**

"Tīkla tapa var tikt pārslogota, ja ienākošās un izejošās datu plūsmas ir pārlieku lielas. Tīkla plūsmas – ienākošās un izejošās var tikt apvienotas un pārsūtītas kopā, skatīt attēlu. Līdz ar to nav nepieciešamības priekš 2 tīkla kartēm vai 2 monitoringa ierīcēm" [5].

1.2.3.3. SPAN (Port mirror) – Spoguļattēls

Tīkla tapas spoguļattēls ir veids kā kopēt datu plūsmu no viena porta uz kādu citu vai vairākiem, piemēram no A uz B, C un D (skatīt 1.7. attēlu)



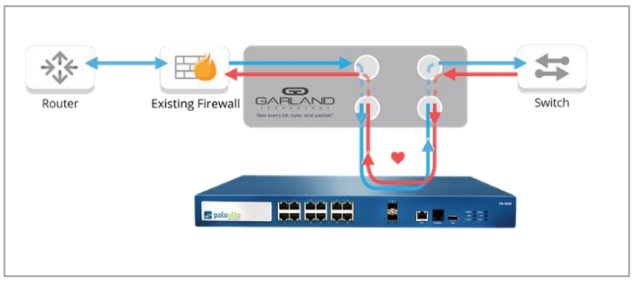
1.7. att. **Spoguļattēls[5]**

“ Bieži vien nav pietiekami daudz *SPAN* portu uz tīkla maršrutētāja vai rūtera, lai pieslēgtu vairākus analīzes rīkus. Pārliecinošs veids, kā atrisināt šādas problēmas ir novadīt SPAN izvadu no maršrutētāja vai rūtera uz replicējošu Tapu, atļaujot vienus un tos pašus datus pārsūtīt vairākām analīzes iekārtām vienlaicīgi.

Svarīgs novērojums ir, ka atšķirībā no *breakout* un agregācijas režīmiem, nav nepieciešams uztraukties par to, ka tīkla tapa ir vājais punkts. Pats savienojums netiek vadīts cauri tīkla tapai. Vienīgais, kas tiek sūtīts uz tīkla tapu ir tīkla plūsmas kopija.” [5]

1.2.3.4. Bypass

“Nozīmīgs rīks lai novietotu aktīvus tiešlaika risinājumus kritiskā savienojumā, neieviešot tīklā riska vietu. Scenārijā, kad ir vēlme uzstādīt uz tīkla līnijas bāzētu sistēmu, kā nākamās paaudzes ugunsmūri vai ievainojamību novēršanas sistēmu, ierīcei jābūt patstāvīgi uz tīkla līnijas, lai izdarītu patstāvīgi nepieciešamās darbības, piemēram, bloķētu apdraudējumus (skatīt attēlu 1.8) [5].



1.8. att. ***Bypass*** **tapas pielietojums tīklā[5]**

"Taču izmantojot šādu risinājumu uz tīkla varētu ieviest apdraudējumu, ka tīkls nestrādā, ja uzstādītais risinājums atsakās darboties, vai vajag noņemt sistēmu priekš atjaunināšanas, kļūdu labošanas.

Kļūdu drošā *Bypass* tīkla tapa novērš šādu scenāriju. Tā ievieto sirdspukstu (*heartbeat*) pakas, kas tiek sūtītas caur uz līnijas esošo sistēmu un kamēr uz līnijas esošā sistēma darbojās, sirdspukstu paka, tiks atgriezta tīkla tapai. Tapa izdzēsīs sirdspukstu paku, pirms sūtīt to tālāk tīklā" [5].

1.2.4. Maršrutētāja spoguļattēls - *SPAN (Port mirroring)*

"Spoguļattēla režīms maršrutētājos dublicē tīkla izejošo un ieejošo plūsmu no pieslēgtajiem *VLAN* un pārsūta iegūto uz monitoringa portu priekš lokālas monitorēšanas vai uz citu *VLAN* priekš attālinātas monitorēšanas. Šāds risinājums tiek izmantots priekš sistēmām, kas analizē tīkla plūsmas, īstenojot lietotāju politikas novērošanu, nevēlamu drošības risku monitorēšanas, tīkla plūsmu lietojuma modeļa paredzēšanai, u.c"[2].

Spoguļattēla režīms ir "nepieciešams maršrutētājos, jo parasti maršrutētājs sūta datus tikai uz nepieciešamo portu, kur ir piestiprināta konkrēta ierīce.

Jāpatur prātā, ka uzstādot spoguļattēla režīmu, maršrutētāja monitoringa ports var tikt pārslogots, līdz ar to parasti tiek vāktas datu plūsmas no specifiskiem *VLAN* un vai tiek uzstādīts datu plūsmu filtrs izmantojot ugunsmūra filtrus. Tikai nepieciešamo datu plūsmu savākšana samazina maršrutētāja noslodzi.

Spoguļattēla darbība ir ar mazāku prioritāti nekā pārējā maršrutētāja datu pārsūtīšana no porta uz portu "[7].

Izanalizējot priekšrocības un trūkumus (skatīt 1.4. tabulu), Autors secināja, ka šis ir labs veids kā monitorēt tīkla datus.

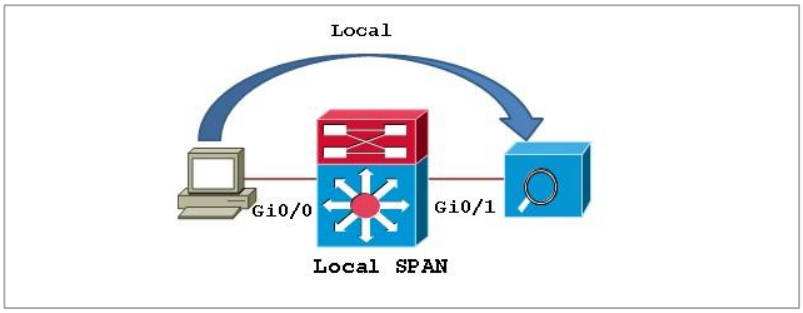
1.4. tabula

Uz maršrutētāja spoguļattēla bāzētas datu savākšanas pieejas priekšrocības un trūkumi

Priekšrocības	Trūkumi
Monitoringa sistēmai ir samērā liela noslodze, jo tiek piegādāts liels datu apjoms	Maršrutētāja apstrādes prioritātes dēļ var netikt piegādāti pilnīgi visi no datiem, kas plūst caur maršrutētāju, pie lielākām noslodzēm.
Nav nepieciešamas papildus ierīces priekš datu plūsmu savākšanas	Paku ietvara laiks var tikt nedaudz izmainīts.
Var redzēt visus klientu datus, kas ir tieši pieslēgti maršrutētājam.	
Labs risinājums kā redzēt datu plūsmas, kur notiek savstarpēja saziņa starp maršrutētāja iekšējām ierīcēm	

1.2.4.1. Vietējais spoguļattēls

Lokālais spoguļattēls ir bieži lietots veids kā kopēt tīkla datus.

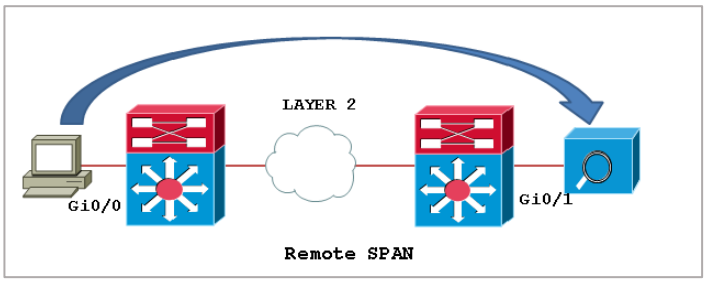


1.9. att. **Tīkla topoloģija, lietojot vietējo spoguļattēlu pieeju [3]**

Lokālais spoguļattēls ir veids kā uzstādīt datu kopēšanu no portiem uz kādu citu portu – monitoringa portu (skatīt 1.9. attēlu). Var tikt kopēti vai nu porta ienākošie dati, izejošie dati, izejošie un ienākošie.

1.2.4.2. Attālinātais spoguļattēls

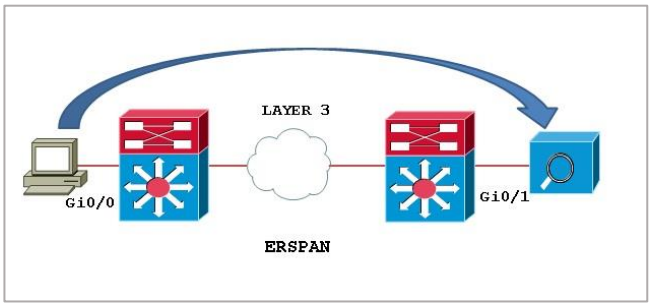
Kā papildinājumu vietējam spoguļattēlam ir attālinātais spoguļattēls (skatīt 1.10. attēlu).



1.10. att. **Tīkla topoloģija izmantojot *RSPAN* pieeju [3]**

"Attālinātā spoguļattēla veidošana *(Remote SPAN (RSPAN)):*Papildinājums lokālajam spoguļattēlam, kuru sauc par attālināto spoguļattēlu. Attālinātais spoguļattēls piedāvā monitorēt datu plūsmas no vairākiem maršrutētājiem vienlaicīgi. Tas nozīmē to, ka ir iespējams centralizēt datu ievākšanu. Attālinātais spoguļattēls darbojas dublicējot datu plūsmu no *RSPAN* sesijas maršrutētāja portiem uz *VLAN* portu, kas ir atvēlēts priekš *RSPAN* sesijas. Šis *VLAN* ir pievienots citiem maršrutētājam, atļaujot *RSPAN* sesijas datu plūsmu pārraidīt starp vairākiem maršrutētājiem. Uz maršrutētāja kurš satur galamērķa portu priekš sesijas, no *RSPAN* sesijas *VLAN* datu plūsma tiek dublicēta uz galamērķa portu." [3]

1.2.4.3. Iekapsulētais attālinātais spoguļattēls



1.11. att. **Tīkla topoloģija izmantojot *ERSPAN* pieeju [3]**

Iekapsulētais attālinātais *SPAN (Encapsulated Remote SPAN - ERSPAN): RSPAN*, kā jau nosaukums norāda, piedāvā vispārīgu maršrutēšanas iekapsulēšanu *(generic routing encapsulation - GRE)* priekš visas caurejošās datu plūsmas un atļauj to paplašināt pāri 3. Slāņa domēniem (skatīt 1.11. attēlu).

ERSPAN ir *Cisco* patentēta funkcija, kas ir pieejama arī ne visos *Cisco* maršrutētājos [3].

1.2.5. Izvērtējums

Izvērtējot risinājumus Autors secināja, ka vislabāk autora vajadzībām un iespējām atbilst *SPAN* veids kā savākt datus no tīkla.

Tas tiek pamatots ar to, ka priekš anomāliju analīzes vajag tomēr salīdzinoši daudz datus. Līdz ar to risinājums "Starpniekierīce" būtu neekonomisks, jo vajadzētu daudz specializētas ierīces un viennozīmīgi vajadzētu kādas izmaiņas – papildinājumus esošajā tīkla struktūrā, lai risinājumu ieviestu.

Uz klienta ierīces balstīts risinājums savā ziņā bija slikts divu iemeslu dēļ.

Pirmkārt tāpēc, ka jāuzstāda uz katras ierīces un tajā tīklā, kurā Autoram bija iespēja savākt īstus tīkla datus, tas netika atzīts par labu esam.

Otrkārt, tas būtu liels samērā liels darbs notestēt un pārliecināties, ka uz katras ierīces savācēj sistēma darbojas kā vajag. Pārsūtot datus būtu jārēķinās ar dažādām papildus problēmām, kā datu dublēšanās, ja uzstāda savācēj sistēmu uz datora A un B. Tad sūtot no A uz B, ieraksti būtu gan no A, gan no B. Tāpat arī jāpiemin, ka tas radītu esošā tīklā papildus datu plūsmas, kas varētu nebūt atbalstāms.

Tīkla tapas līdzīga iemesla dēļ Autoram nebija piemērotas, jo prasa izmaiņas jau esošā tīklā, nepieciešams daudz specializētas aparatūras.

Autors izvēlējās maršrutētāja *SPAN* – spoguļattēlu, jo tas Autoru visvairāk apmierināja šādos kritērijos:

- neprasīja izmaiņas jau esošā tīklā
- piedāvāja labu resursdatoru redzamības apgabalu
- tam nevajadzēja papildus speciālu aparatūru
- minimāla konfigurēšana
- resursdatori neredz to, ka tīkla plūsmu monitoringa sistēma ievāc datus, pretstatā piemēram ar variantu "uz klienta ierīces bāzēts risinājums".

1.3. Esošie tīkla monitoringa risinājumi

Tīkla monitoring risinājumu ir samērā daudz, nākamajās apakšnodaļās tiks uzskaitīti pēc Autora domām populārākie un veiksmīgākie bezmaksas risinājumi.

1.3.1. Monitoringa rīks *Wireshark*

Wireshark ir ”pasaules galvenais un plašāk lietotais tīkla protokolu analīzes rīks. Tas ļauj redzēt to kas notiek tīklā mikroskopiskā līmenī un ir de facto (bieži vien de jure) standarts daudzās komerciālās un bezpeļņas organizācijās, valdības aģentūrās un izglītības iestādēs. *Wireshark* attīstība plaukst, pateicoties brīvprātīgam ieguldījumam no tīkla ekspertiem no visas pasaules un tas ir turpinājums iesāktajam projektam, kuru iesāka Gerald Combs, 1998. ” [6].

“*Wireshark* piedāvā plašu klāstu ar iespējām iekļaujot:

- Dziļu izpēti vairākiem simtiem protokolu, un vairāk pievienotiem ar laiku;
- Tiešlaika datu savākšanu un *offline* analīzi;
- Standarta trīs logu paku pārlūku;
- Vairāk platformu atbalstu: *Windows*, *Linux*, *MacOS*, *Solaris*, *FreeBSD*, *NetBSD* un daudzi citi;
- Savāktie dati var tikt pārlūkoti caur GUI vai arī ar komandrindas rīku *Tshark;*
- Visspēcīgākie atlases filtri industrijā;
- Bagāta *VoIP* analīze;
- Lasīšana / Rakstīšana daudziem datu uzglabāšanas formātiem kā: *tcpdump (libpcap), Pcap NG, Catapult DCT2000, Cisco Secure IDS iplog, Microsoft Network Monitor, Network General Sniffer® (compressed and uncompressed), Sniffer® Pro, and NetXray®, Network Instruments Observer, NetScreen snoop, Novell LANalyzer, RADCOM WAN/LAN Analyzer, Shomiti/Finisar Surveyor, Tektronix K12xx, Visual Networks Visual UpTime, WildPackets EtherPeek/TokenPeek/AiroPeek*, un daudziem citiem.
- Saglabātie datu faili var tikt saspiesti ar *gzip* kompresiju tiešlaikā.
- Tiešlaika dati var tikt nolasīti no *Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI*, un citiem (atkarībā no platformas)
- Dešifrēšanas atbalsts priekš daudziem protokoliem kā: *IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, un WPA/WPA2*
- Krāsošanas rīki var tikt pielietoti, priekš ātras un intuitīvas analīzes
- Izejas dati var tikt eksportēti uz *XML, PostScript®, CSV*, vai arī tekstā ”[6].

1.3.2. Monitoringa rīks *Zabbix*

Zabbix ir ”augsti integrēts tīkla monitoringa risinājums, kas piedāvā daudzveidīgas iespējas vienā sistēmā. *Zabbix* ir biznesa klases atvērtā koda izplatīts risinājums.

Zabbix ir programmatūra, kas monitorē neskaitāmus parametrus no tīkla un veselību un integritāti serveros. *Zabbix* lieto elastīgu apziņošanas mehānismu, kas ļauj lietotājiem konfigurēt e-pasta brīdinājumus par gandrīz jebkādu notikumu. Tas atļauj ātri reaģēt uz servera problēmām. *Zabbix* piedāvā lieliskus pārskatus un datu vizualizācijas iespējas bāzētas uz saglabātajiem datiem. Visi *Zabbix* ziņojumi, atskaites ir pieejami caur tīmekļa pārlūku.

- Pieejamība un veiktspējas pārbaudes
- *SNMP* atbalsts (gan ievākšanai, gan trapiem), *IPMI, JMX, VMware* monitorings.
- Parametru sliekšņu uzstādīšana
- Automatizēta ierīču atpazīšana tīklā
- Domāts uzstādīt gan *Linux,* gan *Windows* sistēmās” [14].

1.3.3. Monitoringa rīks *Snort*

Snort ir ”atvērtā koda tīkla ievainojamību novēršanas sistēma, kas ir spējīga veikt reāla laika analīzi un IP tīklu pakešu saglabāšanu. Tas spēj veikt protokolu analīzi, satura meklēšanu / saskaņošanu, un tas var tikt izmantots, lai pamanītu dažāda klāsta uzbrukumus, un tīkla datu savākšanām, *stealth port scans, CGI attacks, SMB probes, OS fingerprinting* un daudzus citus.

Snort sistēmai ir trīs galvenie pielietojumi:

- Tas var tikt lietots kā tiešs datu plūsmu ievācējs kā, piemēram *tcpdump*;
- Tīkla pakešu žurnalēšanas rīks (lietderīgs priekš tīkla plūsmu atkļūdošanas u.c);
- Pilnībā spēcīga tīkla ievainojamību novēršanas sistēma.

Snort atbalsta šobrīd četrus protokolus, kuriem veica analīzi uz aizdomīgu rīcību: *TCP, UDP, ICMP un IP*. Nākotnē, iespējams, varētu būt vēl tādi protokoli kā *ARP, IGRP, GRE, OSPF, RIP, IPX* un citi ” [13].

1.3.4. Monitoringa rīks *Ntopng*

”Tīkla plūsmu rīks *ntop* - augstas veiktspējas tīmekļa vietnes bāzēta datu analīze un tīkla plūsmu datu ievākšana.

Ntopng ir nākamās paaudzes versija no orģinālā ntop, tīkla plūsmu rīks, kas parāda tīkla lietojumu, līdzīgi kā populārā *Unix* komanda – top. *Ntopng* ir bāzēts uz *libpcap* un ir rakstīts tādā veidā, lai tas varētu virtuāli strādāt uz visām *Unix* platformām, *MacOSX* un *Windows*

sistēmām. *Ntopng* lietotāji var lietot interneta pārlūku, lai pārlūkotu caur ntop (tas uzvedās kā *web* serveris) datu plūsmu informāciju un apskatītu tīkla statusu. Pēdējā laikā ntopng vairāk var tikt uzskatīts kā vienkāršs *RMON*-līdzīgu aģentu ar iestrādātu *web* interfeisu"" [12].

"*Ntopng* piedāvā:

- Kārtot tīkla datu plūsmas atkarībā no daudziem kritērijiem iekļaujot *IP* adresi, portu, *L7* protokolus, caurplūdumu, u.c.;
- Parāda *IPv4/v6* aktīvās tīkla datplūsmās resursdatorus;
- Piedāvā izveidot ilgtermiņa pārskatus par dažādiem tīkla parametriem kā caurplūdumu, lietojumslāņa protokoliem;
- Lielākie X runātāji / klausītāji, lielākie galapunkti, lielākie *L7* slāņa lietojumi;
- Priekš katras sarunu datu plūsmas savāc datus par *TCP* statistiku (atkārtotu pārraidi, pārsūtītās paketes), baiti/paketes, aizturi/*RRT;*
- Saglabā uz diska patstāvīgu datu plūsmas statistiku *RRD* formātā;
- Resursdatoru ģeolokācija un atskaites par resursdatora novietojumu;
- Lietojumslāņa protokolu analīze izmantojot *nDPI*, ntop *DPI* ietvaru;
- Raksturo *HTTP* datu plūsmu piesaistot raksturojumu servisus no *Google* un *HTTP Blacklist*;
- Parāda *IP* datu plūsmas izkliedi dažādiem tīkla protokoliem;
- Analizē *IP* datuplūsmu un kārto to pēc avota / galamērķa;
- *IP* datuplūsmas apakštīkla matrica (kas ar ko runā?);
- *IP* protokolu lietojums kārtots pēc protokolu tipa" [12].

1.3.5. Monitoringa rīks *PRTG*

PRTG ir "tīkla monitoringa rīks priekš Windows operētājsistēmas, kas piedāvā neskaitāmas iespējas priekš tīkla joslas noslogojuma un datu plūsmu monitoringa. Kad monitorē datu pakas, *PRTG* lieto *SNMP*, *Netflow*, *WMI*, un pakešu savākšanu.

PRTG monitorē visas sistēmas, ierīces, datuplūsmu izmantojot šādas tehnoloģijas:

- *SNMP*
- *WMI* un *Windows* Veiktspējas skaitītāji
- SSH: priekš *Linux*/*Unix* un *MacOS* sistēmām
- Tīkla datu plūsmu un paku savākšana
- Ping

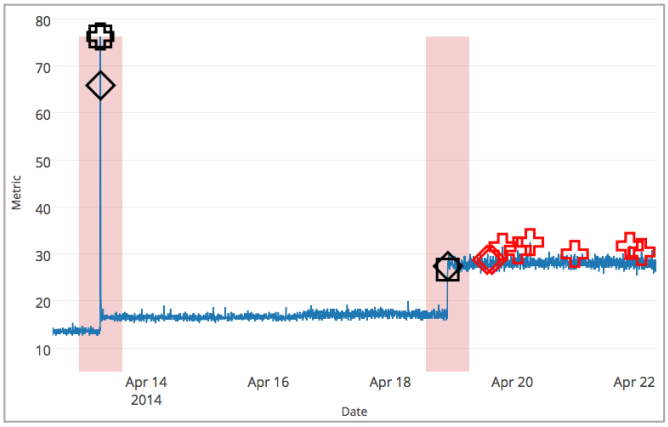
- *HTTP* pieprasījumi un push dati
- *SQL*
- Un daudzas citas”[15].

1.3.6. Izvērtējums

Autors vispirms apskatīja kā risinājumus izmantot *Winpcap*, taču redzot, ka ir pieejami atvērtā koda risinājumi, kam ir laba pieeja pie datu *API*, Autors izvēlējās ņemt gatavus risinājumus datu plūsmu iegūšanā. Autors arī apskatīja risinājumus esošus risinājumus kā piemēram *libtins*. Taču *Wannacry* ievainojamības *SMB* protokolā [27], autoru tikai pārliecināja, ka pareizais ceļš ir apskatīt vairumu lietoto protokolu, nevis monitorēt tikai dažus kā to piedāvā gan *libtins*, gan *snort*.

Zabbix un līdzīgus risinājumus, kas balstās, piemēram uz *SNMP* protokolu Autors neizvēlējās, līdzīga iemesla dēļ kas ir aprakstīts secinājumos par datu savākšanu izmantojot dažādas tīkla protokolu pieejas.

1.4. Analīzes metodes un to salīdzinājums



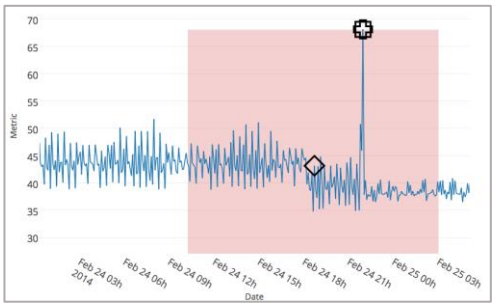
1.12. att. **“Anomāliju noteikšanas piemērs uz produkcijā esoša servera bāzēts uz Centrālā procesora noslodzi. Marķieri, atbilst dažādiem anomāliju noteicējiem: *HTM, Skyline* un *ADVec* ir rombs, kvadrāts un pluss atbilsotši. Melnā krāsā iekrāsoti patiesā anomālijas un ar sarkano nepatiesās anomālijas” [10].**

Noteikt anomālijas ir svarīgi, jo ne vienmēr viss ir nosakāms ar normālas sistēmas darbības sliekšņiem, kurus lietotājs var pats uzstādīt. Un vēlāk salīdzināt sliekšņus ar pašreizējām vērtībām, lai noteiktu sistēma darbojas pareizi. Šie sliekšņi var mainīties un palikt

tādi uz ilgu laiku. Piemēram, ja ir ieinstalēta ziņu aplikācija, kas visu laiku fonā skatās vai nav jaunu ziņu, tad tā patērēs ik brīdi vairāk datus, līdz ar to jaunā normas robeža būs pa visam cita (skatīt 1.12. attēlu). Līdz ar to nepieciešams būtu patstāvīgi pielāgoties izmaiņām.

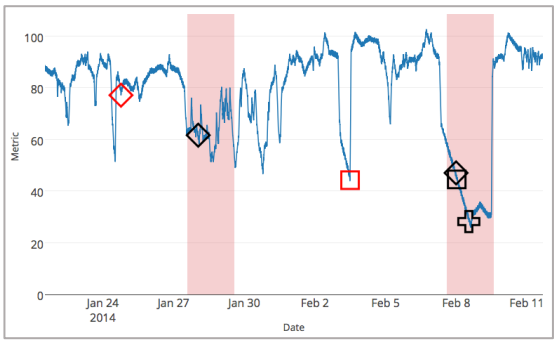
Ideja balstās uz faktu, ka lai noteiktu veiksmīgu tīkla darbību nepietiek ar to, ka tiek uzstādīts statisks slieksnis kuru pārkāpjot tiek apgalvots, ka sistēmā varbūt ir radusies problēma. Šādi sliekšņa veida uzstādīšanas risinājumi ir mūsdienās izplatīti risinājumos, kas monitorē iekārtas caur *SNMP* protokolu, piemēram *PRTG* [8] vai *NAGIOS* [9]. Šāda veida risinājumi mazāk noder, ja runa ir par tīkla plūsmu analīzi, jo tīkla plūsmas ir dažkārt nevienmērīgas, bet ar noteiktu raksturu. Līdz ar to anomāliju bāzēta sistēma te var palīdzēt saprast, vai tīkla darbība ir normāla, kaut arī tīklā nereti ir svārstības lietojuma ziņā.

Motivācija radīt anomāliju bāzētu sistēmu rodas no fakta, ka arvien vairāk un vairāk tiek izmantotas datu plūsmas, kas tiek šifrētas un līdz ar to būtu jāizmanto gudrāki risinājumi, kas prot atšķirt labu tīkla darbību no sliktas.



1.13. att. **"Anomāliju noteikšanas rezultāti, kuri parāda iepriekšēju paredzēšanu. Lai gan visi anomāliju noteicēji pareizi identificē anomāliju, *HTM* to novēro jau 3 h pirms *Skyline* un *ADVec* sliekšņa veida novirzei parādoties" [10].**

"Atkarībā no analīzes tipa pielietota ievainojamību atrašanas sistēmās ir balstītas vai nu signatūras bāzētas vai arī anomāliju bāzētas. Signatūras bāzētas (sauc arī par nepareizas lietošanas bāzētas) meklē šablonus vai signatūras analizējamajos datos. Priekš šādas pieejas, signatūras bāzēta datubāze ar atbilstībām uz zināmiem uzbrukumiem ir jau iepriekš nodefinēta. No otras puses, anomāliju bāzēti risinājumi mēģina paredzēt "normālu" sistēmas uzvedību, lai aizsargātu to un ģenerē anomālijas paziņojumu, ja novirze no šobrīd novērotā un normālās uzvedības pārsniedz noteiktu slieksni. Vēl viena iespēja ir modelēt nenormālu sistēmas uzvedību un paziņot, ja starpība starp novēroto uzvedību un paredzēto uzvedību ir mazāka nekā kāds iepriekš definēts slieksnis"[1].



1.14. att. **Anomāliju noteikšana temperatūras datiem [10].**

Abi šie attēli (skatīt 1.13 attēlu un 1.14. attēlu) ilustrē ļoti biežu situāciju, kas atgadās reālajā dzīvē. "Kā jau demonstrēts *NAB* (*Numenta Anomaly Benchmark*) datu failos, īslaicīgas izmaiņas lietojumu paradumos bieži paredz lielas, viegli nosakāmas nobīdes. Temporārās un virkņu bāzētās anomāliju noteikšanas tehnikas var palīdzēt noteikt anomālijas datu plūsmā pirms viņas ir viegli redzamas. Tas dod cerību, ka šādi algoritmi var tikt izmantoti produkcijā, lai jau agri ziņotu par anomālijām un palīdzētu izvairīties no katastrofām paļaujoties daudz vairāk nekā uz tradicionālajām tehnikām" [10].

1.5. tabula

Ievainojamību noteikšanas metožu salīdzinājums [21]

	Signatūras bāzētas (zināšanu bāzētas)	**Anomāliju bāzētas (paradumu bāzētas)**	**Pakešu protokolu analīze (specifikācijas bāzētas)**
Priekšrocības	• Vienkāršākā un efektīva metode lai detektētu zināmus uzbrukumus • Padziļināta konteksta analīze.	• Efektīvs jaunu un iepriekš neredzētu ievainojamību atklāšanā • Mazāk atkarīgs no Operētājsistēmas • Sekmē privilēģiju ļaunprātīgas izmantošanas noteikšanu	• Var zināt un izsekot līdzi protokola stāvokļiem • Var atpazīt neparedzētas komandu sekvences
Trūkumi	• Neefektīva pret nezināmiem uzbrukumiem ,	• Vāja profilu precizitāte pateicoties tam, ka	• Resursu ietilpīga protokolu

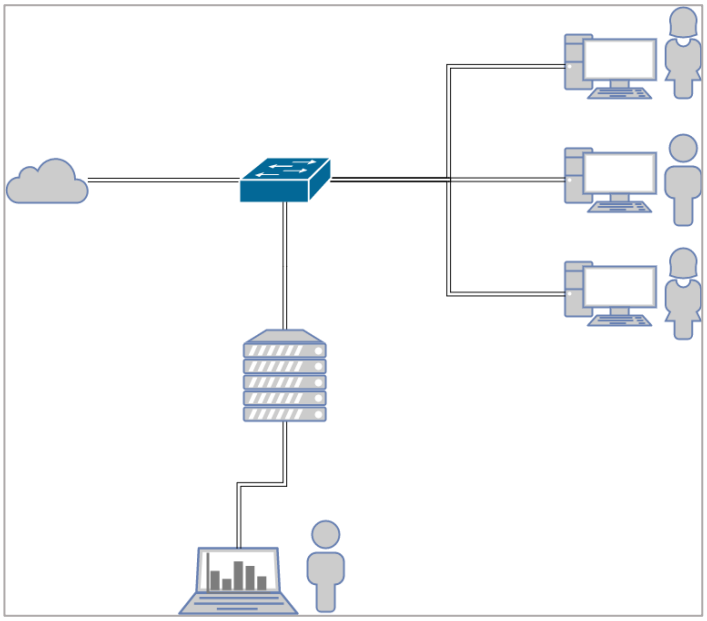
	izvairīšanās (*evasion*) uzbrukumiem un zināmu uzbrukumu variantiem. • Maza saprašana par stāvokļiem un protokoliem • Grūti uzturēt signatūras un šablonus atjauninātus • Laikietilpīgs lai uzturētu zināšanas	novērotie notikumi patstāvīgi mainās • Nav pieejams, kad tiek pārrēķināti uzvedības profili • Grūti paziņot par izmaiņām pareizajā laikā.	stāvokļu izsekošana un izmeklēšana • Nevar izmeklēt uzbrukumus, kas izskatās kā labvēlīga protokola uzvedība • Var būt nesaderīga ar izvēlēto *OS* vai programmatūru

Autora motivācija veidot anomāliju bāzētu risinājumu patiesībā bija divu iemeslu pēc. Pirmais ir tāds, ka tīkla monitoringa sistēmās tas nav bieži izmantots, jo būtībā tiek izmantoti risinājumi, kas balstās uz zināšanu vai specifikācijas bāzes un otrkārt, tāpēc, ka tam ir liels potenciāls, esošo risinājumu papildināšanai (skatīt 1.5. tabulu).

2. TĪKLA PLŪSMU MONITORINGA SISTĒMA

Tiek glabāta informācija par datu plūsmu noteiktā laika periodā, ziņojumiem, kam ir sesijas identifikators, piemēram 10 sekundes. Līdz ar to, piemēram, *TCP* sesijas ietvaros ik pēc 10 sekundēm būs tikai viens ieraksts.

Risinājums tika veidots uz sistēmas *Windows*, taču pašas komponentes (*Wireshark, Nodejs, Python*), kas tika izmantotas risinājumā, ir domātas vairākām operētājsistēmām un ar nelielu piepūli vajadzētu būt strādājošam risinājumam šādās operētājsistēmās (kā piemēram *Linux, MaxOS, Windows, FreeBSD, Solaris, OpenBSD …*).



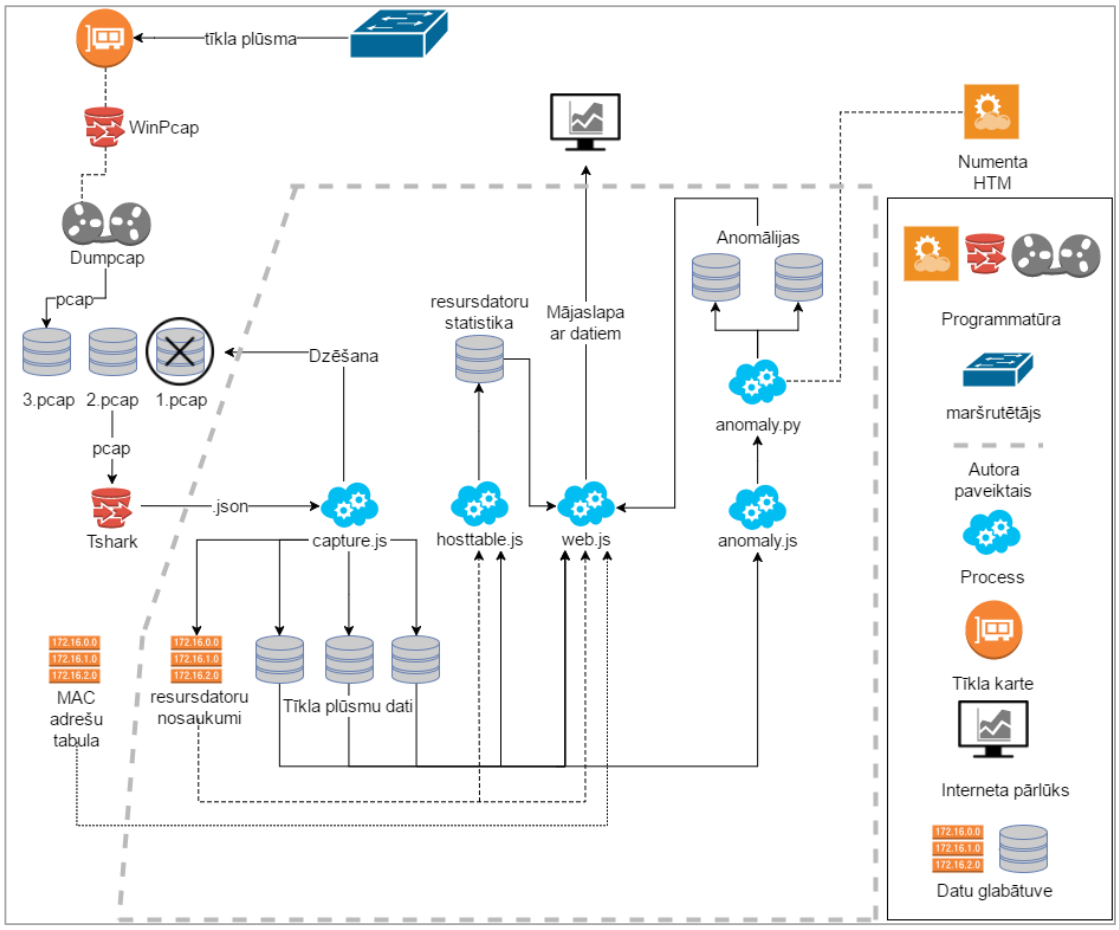
2.1. att. **Tīkla plūsmu monitoringa sistēmas kopskats, kas attēlo datu plūsmu.**

Attēlā 2.1. attēlots no kreisās puses augšā: Interneta tīkls, maršrutētājs, Lietotāji. Un zemāk: Tīkla plūsmu monitoringa sistēmas serveris un lietotājs, kas to izmanto. Šī darba sistēmas datu ievākšanai tiek izmantota maršrutētāja vietējais spoguļattēls, lai iegūtu datus (skatīt 2.1. attēlu).

Tīkla monitoringa sistēmas datus un analīzes rezultātus var apskatīt priekš Tīkla plūsmu monitoringa sistēmas Autora izveidotā vietnē (skatīt 7. pielikumu).

2.1. Risinājuma implementācija

Tīkla plūsmu monitoringa sistēmas izveidei ir izmantotas bibliotēkas / rīki / tehnoloģijas kā *Tshark* (*Wireshark*), *Dumpcap* (*Wireshark*), *Nodejs, Python nupic* (*Numenta*).



2.2. att. **Autora izveidotās tīkla plūsmu monitoringa sistēmas detalizēts attēlojums.**

Tīkla plūsma no maršrutētāja monitoringa porta tiek padota tīkla kartei. *Winpcap* draiveris saņem datus no tīkla kartes un pēc tam *Dumpcap* lietojumprogrammai. *Dumpcap* ieraksta saformatētus tīkla datus *pcap* formātā. *Tshark* patstāvīgi lasa saglabātos *pcap* datus un nosūta jau saformatētus un izanalizētus datus par paku slāņiem *capture.js* servisam caur standart izvadu izmantojot procesu caurules *(Process Pipes). Hostable.js* serviss apkopo datus pa resursdatoru statistiku. *Web.js* apkopo visu nepieciešamo informāciju no visiem savāktajiem un pieejamajiem datiem, lai parādītu to pārlūkā. *Anomaly.js* serviss veido anomāliju ierakstus no tīkla datu plūsmām.

2.2. Datu savākšana no dzīvā interfeisa

Datu savākšanai tiek izmantots *Wireshark* rīks *Dumpcap*, kas iegūst datus no tīkla izmantojot *libpcap*/*Winpcap* bibliotēkas un tos ieraksta failā uz diska *pcap* / *pcap-ng* formātā. [22]. Lietotāji sūta savus datus uz maršrutētāju. Maršrutētājs savukārt, visus saņemtos datus aizsūta monitoringa portam, kurš ir pieslēgts pie Monitoringa sistēmas ierīces. Monitoringa ierīcē ar *Winpcap* palīdzību *Dumpcap* process savāc datus, saformatē un ieraksta **.pcap* failā.

Faili tiek rotēti pēc 100 Mb. Tas vajadzīgs, lai nesakrātos daudz informācijas, jo tīkla plūsmu monitoringa sistēmas viens no mērķiem ir monitorēt ilgtermiņā un neaizņemt daudz vietas.

2.3. Savākto datu protokolu analīze

Tīkla iegūto datu protokolu analīzei tika izmantots *Tshark* rīks, kas ir *Wireshark* apakšrīks. Autors darba gaitā atklāja to, ka *Tshark* rīks nedarbojas no dzīvās tīkla plūsmas un neatrada veidu, kā panākt, lai *Tshark* datus ievāktu uzreiz no piemēram *libpcap* / *Winpcap,* bez failu starpniecības. No otras puses jāteic, ka failu starpniecības mehānisms ir labs, ja datu ir tiešām daudz, jo analizējot tos tiešlaikā varētu gadīties situācija, kad būtu vairāk izkritušu datu paku, piemēram, ja izmanto *SPAN* metodi datu savākšanai. Apstrādāta un izanalizēta *Tshark* rīka atgrieztais datu *JSON* formāta paraugs *(skatīt 6. pielikumu)*.

Šis rīks ir teicams, jo spēj atpazīt vairāk kā 206000 laukus 2000 protokolos virs versijas 2.2.5 [24]. Līdz ar to ir lielas iespējas šādu risinājumu papildināt ar jaunu funkcionalitāti.

2.4. Izanalizēto datu filtrācija un vajadzīgo datu saglabāšana ilgtermiņā

Dati, kas iegūti no *Tshark* tiek izanalizēti *Nodejs* rakstītā programmatūrā, no kuras tiek atlasīta vajadzīgā informācija priekš sistēmas. Dati tiek glabāti binārā formātā, lai nodrošinātu mazāku datu glabāšanas apjomu un ātrāku datu pēcapstrādi kā filtrāciju, datu izgūšanu periodā, u.c. *TCP* sesijām papildus tiek veikta apkopošana ar periodu, piemēram 10 sekundes. Pārējie dati tiek atstāti nepārveidotā veidā. Pārējo datu nepārveidošana ir saistīta ar to, ka pārējo tīkla datu neapkopots apjoms ir salīdzinoši mazs.

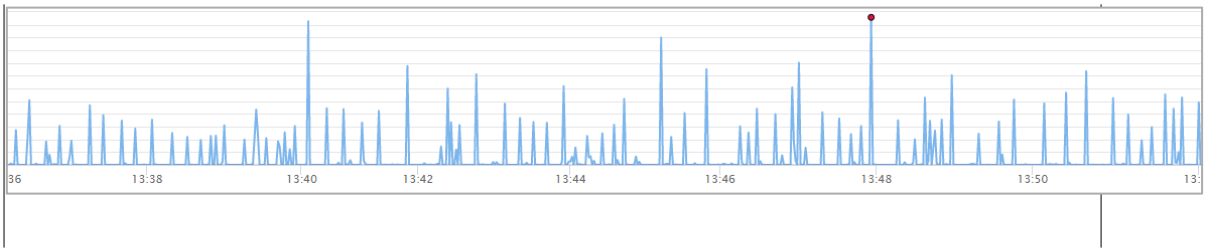
2.5. Datu lietojuma analīze

Datu lietojuma analizēšanai Autors izvēlējās lietot *Numenta HTM* (*Hierarchical temporal memory*) bibliotēkas *python* implementāciju – *nupic*, kas attaisnoja autora cerības datu lietojuma paredzēšanā laikrindas datos, arī tādos kuros ir liels troksnis, anomāliju precizitāte bija ievērojami laba. Dati priekš anomālijām tiek apkopoti pa sekundei. Tas tiek darīts tāpēc, ka datu ir ievērojami daudz un arī vienas sekundes laikā var tikt veiktas neskaitāmas tīkla plūsmu datu apmaiņas.

Daži no svarīgakajiem konfigurācijas iestatījumiem, ar kuriem Autors sasniedza labus rezultātus:

- Kolonnu skaits iekš *Spacial Pooler* : 2048,

- *Spacial Pooler* maksimālais kolonu skaits reģiona izejā : 40,
- *Spacial Pooler* jūtīgo kolonnu skaits : 0.8,
- *Spacial Pooler* savienojuma slieksnis : 0.1,
- Īslaicīgās atmiņas kolonnu skaits : 2048,
- Īslaicīgās atmiņas rezervēto šūnu skaits uz kolonnu : 32,
- Īslaicīgās atmiņas segmenta aktivācijas slieksnis : 12,
- Īslaicīgās atmiņas jauno elementu mācīšanās skaits : 3.

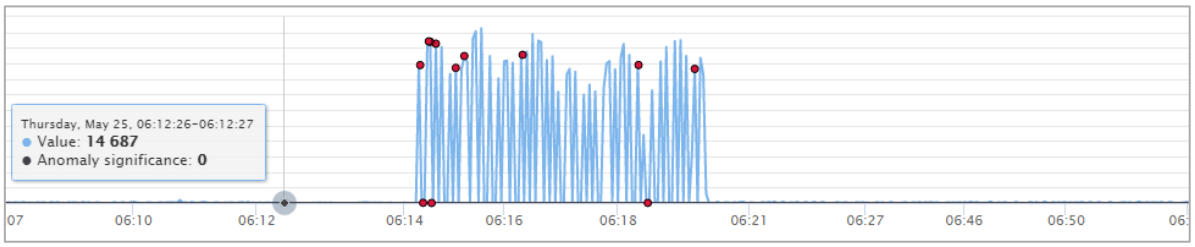


2.3. att. ***ARP* protokola lietojuma grafiks ar vienu sarkanu aplīti**

Datos, kur ir svarīgi atrast anomālijas, ir nepieciešams atrast anomālijas arī datos, kuros ir izteikts lietojuma troksnis kā piemēram (2.3. attēlā). Var redzēt (2.4. attēlā), ka anomāliju noteikšanas sistēma darbojas pēc Autora cerībām, jo atpazīst sagaidītās anomālijas.

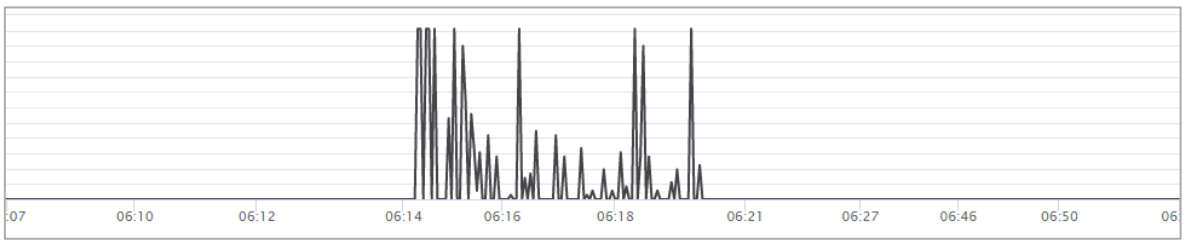
2.4. att. **Anomāliju varbūtību novērtējums 4.3. attēlā redzamajiem datiem *ARP* protokola lietojuma statistikā**

Uz īstiem datiem Hierarhiskā Īslaicīgā atmiņa parādīja labas spējas anomāliju atpazīšanā arī tur kur tas bija tiešām acīmredzams un svarīgs (skatīt 2.5. attēlu).



2.5. att. ***HTTP*** **protokola lietojuma grafiks ar izteiktu anomāliju**

Šādai situācijai (skatīt 2.5. attēlu), kas izceļās stipri no pārējā fona ir nepieciešams būt pamanītai, kas arī veiksmīgi tiek pamanīta (skatīt 2.6. attēlu). Turklāt šāds lietojums ir kā redzams ļoti izcēlies jo citos punktos lietojuma līmenis ir nevis zems bet normāls (skatīt 2.5. attēlu) kā to var redzēt rīkjoslā. Pie tam anomālijas lielumu norāda arī fakts, ka vairākās vietās ir norādīts augsts anomālijas apjoms.



2.6. att. **Iepriekšējā attēla (2.5. attēla) datu anomāliju novērtējums**

2.6. Tīkla plūsmu monitoringa sistēmas glabātie dati

Tiek savākta informācija par datu plūsmām. Datu plūsmu dati glabājās bināri un ir iedalīti pēc tīkla pakas slāņu nosaukumiem, lai nodrošinātu mazāku datu uzglabāšanas apjomu un ātrāku datu filtrēšanu. TCP protokolam dati tiek apkopoti 10 sekunžu intervālā pēc sesijas un galapunktiem. Glabājamais saturs (skatīt 2. un 4. pielikumu, daļējs atspoguļojums – daļa no zemāk aprakstītajiem datiem):

- Informācija par ierakstu – piemēram, adreses tips – *IPv4*/*IPv6*/*MAC*
- Pieprasījuma iekārtas adrese
- Galamērķa iekārtas adrese
- Sākuma laiks
- Beigu laiks
- Ziņojumu kopskaits laika intervālā
- Ziņojumu kopējais apjoms laika intervālā

- Ziņojumu sesijas identifikators, ja tāds ir
- Pieprasījuma ports
- Galamērķa ports,

Tiek glabāti dati par resursdatoriem csv formātā, kā pāru atbilstība IP adrese -> Resursdatora nosaukums. Tas ir vajadzīgs, lai vēlāk vietnē tiktu parādīti draudzīgi resursdatoru nosaukumi (skatīt 3. pielikumu).

Īstermiņā arī tiek glabāti dati vieglākai atspoguļošanai sistēmas vietnē dati kā:

- Resursdatoru statistika, kurā apkopota informācija par:
    - Resursdatora nosaukums
    - Adrese
    - Kopējais pārsūtītais datu apjoms
    - Kopējais pārsūtītais paku skaits
    - Protokolu kopējā izmantojuma statistika
    - Kopējais tīkla lietojums salīdzinājumā ar citiem lietotājiem

Šāda resursdatoru statistikas informācija ir svarīga, lai uzzinātu kopskatu par ierīcēm tīklā un redzētu, kuras ierīces ir tieši pieslēgtas tīklam un kuras ierīces ir tīklā, bet ne tieši pakļautas monitoringa sistēmas darbībai.

- Dati par anomālijām, kur tiek glabāti gan anomāliju kopsavilkuma dati, gan arī pilni dati, kas piekārto datu ierakstam anomālijas lielumu.

2.7. Resursdatora nosaukumu iegūšana:

Tiek savākta informācija par resursdatoru nosaukumiem (*Hostname*). Kas ļauj vieglāk orientēties tajā, kas ar ko sazinās.

"*DNS* resursdatora nosaukumus var noteikt izmantojot *DNS* serveri. Sistēmas izsaukums *gethostname()* mēģinās pārveidot adresi uz nosaukumu. Lai to izdarītu vispirms tas prasīs sistēmas resursdatoru nosaukumu failam (piemēram */etc/hosts*), ja nu tas atrod atbilstošu ierakstu. Ja tas neatrod, tad tas prasīs nokonfigurētajiem *DNS* serveriem(-im)."[11]

Mac adreses tiek pārvērstas par "lasāmāku tekstu", to var izdarīt ar MAC adrešu tabulu, lai pārvērstu pirmos 3 baitus no *Ethernet* adreses uz ražotāja abreviētu nosaukumu, kas ir *IEEE* noteikta, piemēram (e.g. 00:09:5b:01:02:03 → *Netgear*_01:02:03) [11].

REZULTĀTI

Autors ir izveidojis tīkla plūsmu monitoringa sistēmu un iekštīkla privātu vietni, kur var apskatīties vēsturiskos datus, iekšējā tīkla saziņas galapunktus pēc protokolu lietojuma (skatīt 5. pielikumu), protokolu lietojuma grafiku, resursdatora kopējās statistikas salīdzinājumu pret citiem lietotājiem un protokolu izmantojuma statistiku, protokolu lietojuma statistiku, protokola lietojuma anomāliju un salīdzināšanas grafikus (skatīt 1. pielikumā).

Tīkla plūsmu monitoringa sistēmas lietotāji var noteikt kurā laikā ir vislabāk veikt apkopes darbus sistēmām. Vieglāk noteikt laiku priekš sistēmu restarta, labošanas un atjaunošanas, ja tīkls tiek izmantots patstāvīgi ar nelieliem pārtraukumiem, gan dienā, gan nakts laikā.

Nav nepieciešamas specializētas aparatūras, datu glabāšanai un sistēmas darbināšanai, jo datu apjoms, kas tiek glabāts ir salīdzinoši mazs – vidēji 7 dienu laikā, kur katrā dienā tiek pārsūtīti apmēram 30 gb, kopumā tiek glabāts tikai apmēram 1 gb.

Var veikt analīzes, par lietotāju paradumiem – kad parasti darbinieks ierodas un iet prom no darba. Kādus interneta resursus tas izmanto un cik daudz.

Uzraugāmais lietotājs neredz, ka tiek novērots, jo sistēma paredzēta lietošanai pie *Routera* vai *Switch Port mirror* režīmā.

SECINĀJUMI

Visi uz tīkla protokoliem bāzētie risinājumi ir labi, bet katram ir savs pielietojums. Autors neizvēlējās nevienu no šiem risinājumiem kā par pamatu tīkla plūsmu monitoringa sistēmas datu ievākšanas avotam. Tam iemesls bija, ka Autora sistēmu bija plānots papildināt ar funkcionalitāti, kas vērtē / analizē pašus pakešdatus. Līdz ar to neviens no šiem risinājumiem nav īsti piemērots. Šie risinājumi balstās uz to, ka tiek sūtīta maza informācijas porcija par apkopotiem datiem, piemēram skaitītāji. No šādiem datiem viennozīmīgi var saprast tīkla kopējo situāciju un šīs pieejas iedvesmoja Autora risinājumu, datu kopsavilkuma veidošanā.

Autors vispirms apskatīja kā risinājumus izmantot *Winpcap*, taču redzot, ka ir pieejami atvērtā koda risinājumi, kam ir laba pieeja pie datu *API*, Autors izvēlējās ņemt gatavus risinājumus datu plūsmu iegūšanā. Autors arī apskatīja risinājumus esošus risinājumus kā piemēram *libtins*. Taču *Wannacry* ievainojamības *SMB* protokolā [27], autoru tikai pārliecināja, ka pareizais ceļš ir apskatīt vairumu lietoto protokolu, nevis monitorēt tikai dažus kā to piedāvā gan *libtins*, gan *snort*.

Zabbix un līdzīgus risinājumus, kas balstās, piemēram uz *SNMP* protokolu, Autors neizvēlējās, līdzīga iemesla dēļ kas ir aprakstīts secinājumos par datu savākšanu izmantojot dažādas tīkla protokolu pieejas.

Autors izvēlējās maršrutētāja *SPAN* – spoguļattēlu par veidu kā savākt datus, jo tas Autoru visvairāk apmierināja šādos kritērijos:

- neprasīja izmaiņas jau esošā tīklā
- piedāvāja labu resursdatoru redzamības apgabalu
- tam nevajadzēja papildus speciālu aparatūru
- minimāla konfigurēšana
- resursdatori neredz to, ka tīkla plūsmu monitoringa sistēma ievāc datus, pretstatā piemēram ar variantu "uz klienta ierīces bāzēts risinājums".

Tīkla plūsmu monitoringa sistēmas uzlabošanai būtu noderīgi uztaisīt modeli, kurš tiek uztrenēts atpazīt kādiem *TCP, UDP* portiem atbilst kāda noteikta informācijas struktūra un saturs. Piemēram, *MySQL* tipiski tiek piedāvāts 3306 ports. Līdz ar to varētu uztrenēt *SQL* valodas pieprasījumu izskatu. Un pamīšus ņemt pakešu datus no kādas tīkla sesijas un skatīties vai atbilst saturs atbilst portam. Kā arī varētu papildināt esošo risinājumu ar uz satura analīzes bāzēts anomāliju risinājumu, kas atpazīst "labus" datus no "sliktiem". Galapunktu ģeolokācija un savienojuma laika analīzes paradumu noteikšana būtu noderīgi, lai skatītos paradumus par lietotāju saziņas galapunktiem. Vēl viens veids kā uzlabot esošo risinājumu būtu *Tshark*

integrācija – modificēt *Tshark* atvērtā koda programmatūru pielāgojot to savām vajadzībām un vācot tikai izmantotos datus vai arī izvēlēties citu risinājumu datu ieguvei : *libtins*, *scapy*, savs risinājums . Autors saskārās ar to, ka šāds esošs risinājums ir labs priekš maziem tīkliem ar datu plūsmu aptuveni < 5 Mb/s. Tas ir tāpēc, ka pakešu protokolu atkodēšana / analīze par ko atbild *Tshark*, prasa daudz laika.

PATEICĪBAS

Paldies visiem par atbalstu un ieteikumiem.

Paldies darba vadītājam Leo Trukšānam par būtisku atbalstu un sniegtajām konsultācijām darba izveidei.

Paldies AS “SAF Tehnika” un Intam Bukovskim par palīdzību materiālu un datu nodrošināšanā.

IZMANTOTĀ LITERATŪRA UN AVOTI

[1] GARCIA-TEODORO, Pedro, et al. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 2009, 28.1: 1

[2] Juniper Tech Library Understanding Port Mirroring. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://www.juniper.net/documentation/en_US/junos/topics/concept/port-mirroring-qfx-series-understanding.html.

[3] Cisco Support Understanding SPAN,RSPAN,and ERSPAN. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://supportforums.cisco.com/document/139236/understanding-spanrspanand-erspan.

[4] Wireshark Ethernet capture setup. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://wiki.wireshark.org/CaptureSetup/Ethernet.

[5] Garland tehnology The 101 Series: A Primer On Network TAPs. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://www.garlandtechnology.com/2014/01/17/a-test-access-point-tap-primer.

[6] About Wireshark. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://www.wireshark.org/.

[7] Performance vision HOW TO CAPTURE TRAFFIC ? (SPAN VS TAP). [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] http://blog.performancevision.com/eng/earl/how-to-capture-traffic-span-vs-tap.

[8] PAESSLER, THRESHOLD MONITORING, ADVANCED NETWORK MONITORING WITH FLEXIBLE ALERTING OPTIONS. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://www.paessler.com/threshold_monitoring.

[9] Ntopng Integration with Nagios. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] http://www.ntop.org/guides/ntopng-integration-with-nagios/.

[10] LAVIN, Alexander; AHMAD, Subutai. Evaluating Real-Time Anomaly Detection Algorithms--The Numenta Anomaly Benchmark. In: *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*. IEEE, 2015. p. 38-44.

[11] Wireshark name resolution. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://www.wireshark.org/docs/wsug_html_chunked/ChAdvNameResolutionSection.html.

[12] ntopng High-Speed Web-based Traffic Analysis and Flow Collection. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] http://www.ntop.org/products/traffic-analysis/ntop/.

[13] SNORT Users Manual. [Tiešsaiste] [Citēts: 2017. gada 26. maijs] https://www.snort.org/documents/snort-users-manual.

[14] Zabbix documentation. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://www.zabbix.com/documentation/3.4/.

[15] PAESSLER PRTG. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://www.paessler.com/prtg.

[16] LIU, Guoming; NEUFELD, Niko. Management of the LHCb network based on SCADA system. *Proceedings of ICALEPCS09*, 2009.

[17] SolarWinds, Server & Application Monitor. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] http://www.solarwinds.com/documentation/apm/docs/SAMAdminGuide.pdf.

[18] Nagios Core Version 3.x Documentation. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://assets.nagios.com/downloads/nagioscore/docs/nagios-3.pdf.

[19] Wikipedia, Vienkāršais tīkla pārvaldības protokols. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://lv.wikipedia.org/wiki/Vienk%C4%81r%C5%A1ais_t%C4%ABkla_p%C4%81rvald%C4%ABbas_protokols.

[20] WALDBUSSER, Steve; ROMASCANU, Dan; KALBFLEISCH, Carl W. Introduction to the remote monitoring (RMON) family of MIB modules. 2003.

[21] LIAO, Hung-Jen, et al. Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications, 2013, 36: 16-24.

[22] Wireshark, Dump network traffic. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://www.wireshark.org/docs/man-pages/dumpcap.html.

[23] Wireshark, tshark - Dump and analyze network traffic. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://www.wireshark.org/docs/man-pages/tshark.html.

[24] Wireshark Display Filter Reference. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://www.wireshark.org/docs/dfref/.

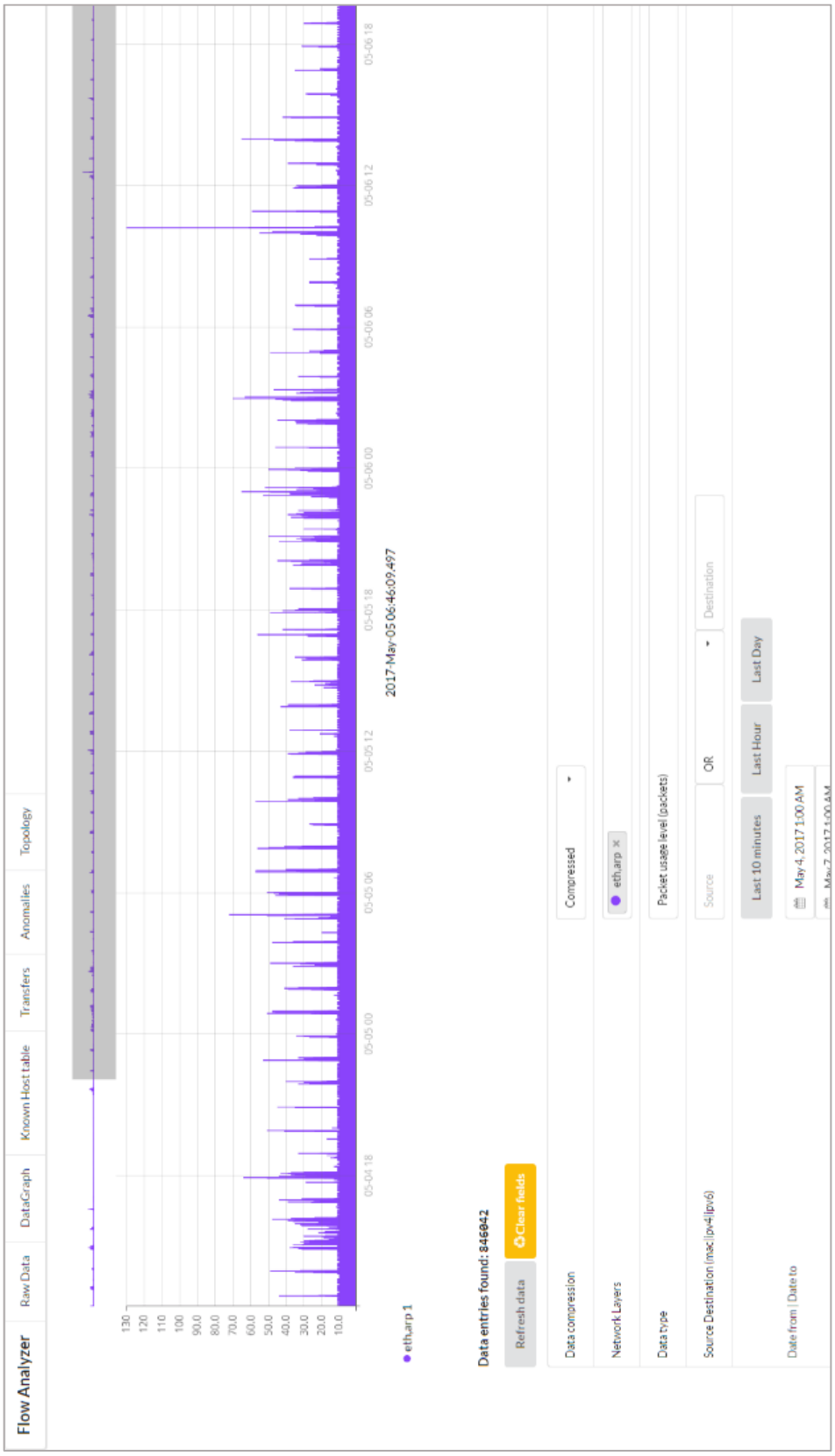
[25] Wikipedia, NetFlow. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://en.wikipedia.org/wiki/NetFlow.

[26] Introduction to Cisco IOS NetFlow - A Technical Overview. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html.

[27] WannaCry ransomware attack. [Tiešsaiste] [Citēts: 2017. gada 26. maijs.] https://en.wikipedia.org/wiki/WannaCry_ransomware_attack.

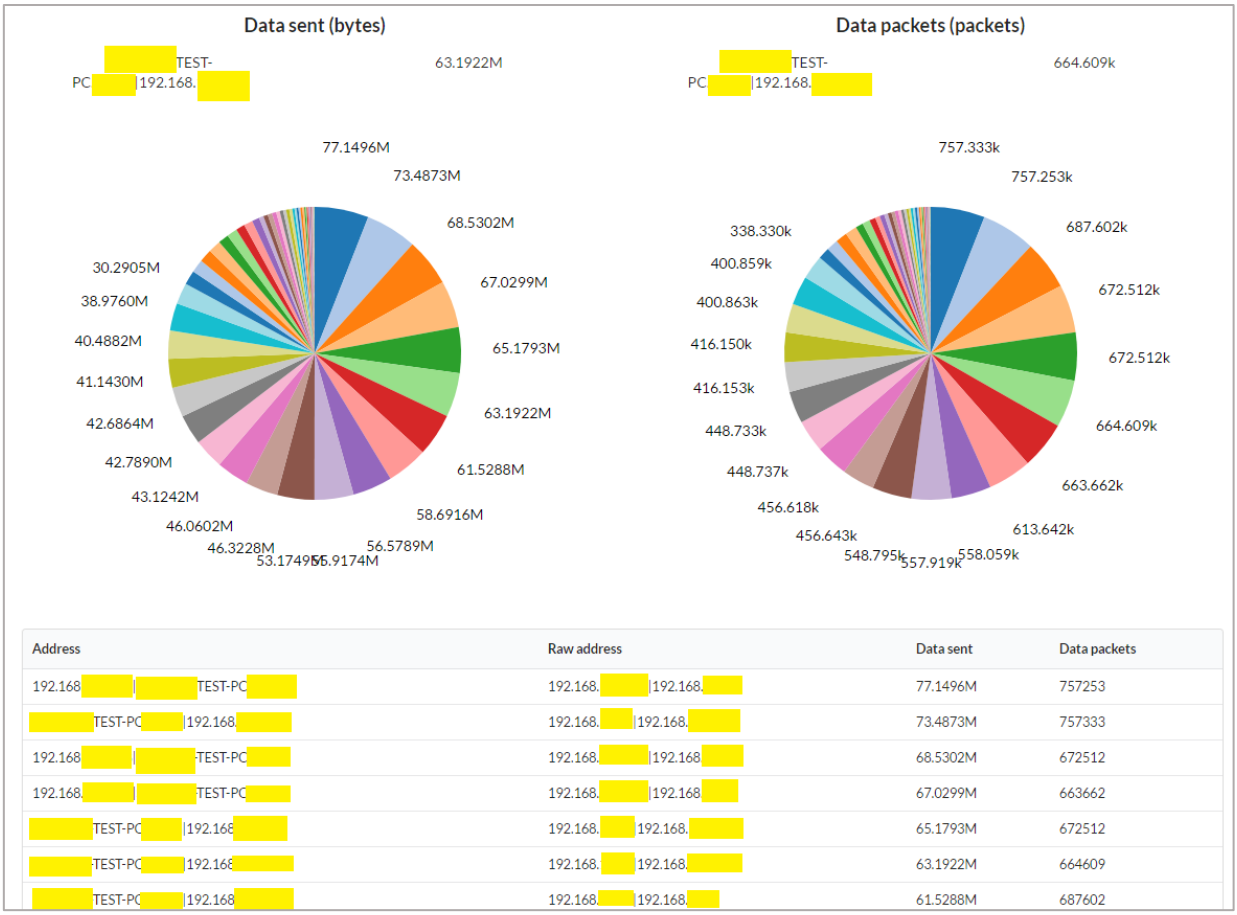
PIELIKUMI

1. pielikums ARP protokola lietojuma grafika piemērs.



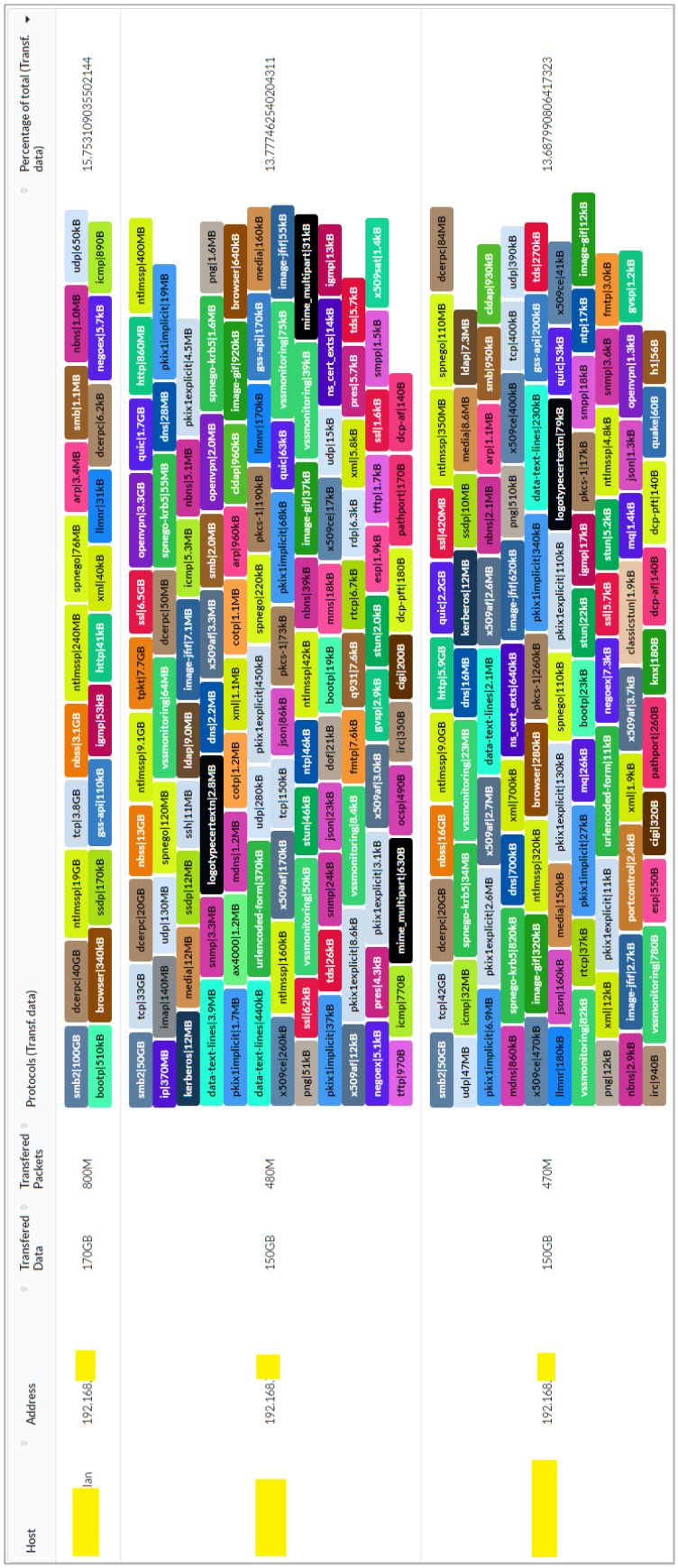
2. pielikums Tīkla plūsmu monitoringa sistēmas neapstrādātu datu piemērs TCP portokolam

3. pielikums Tīkla plūsmu analīzes sistēmas lietotāju datu lietojums periodā SNMP protokolam

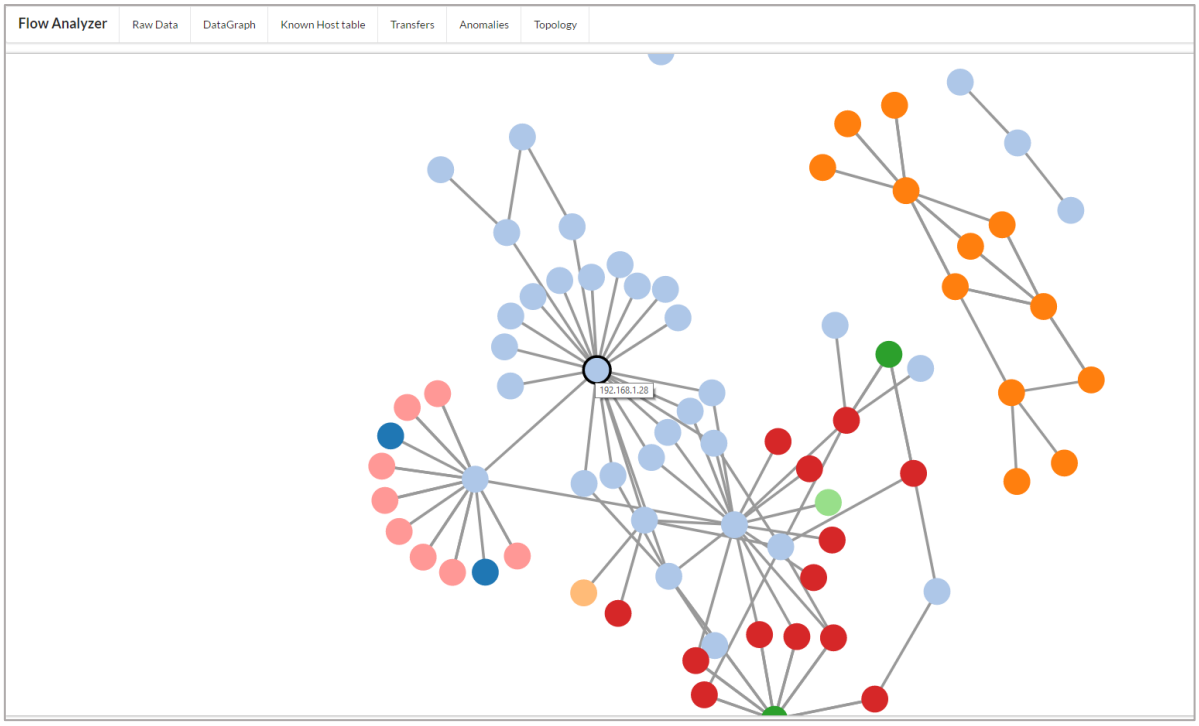


Address	Raw address	Data sent	Data packets
192.168 | TEST-PC	192.168. |192.168.	77.1496M	757253
TEST-PC |192.168.	192.168. |192.168.	73.4873M	757333
192.168 | -TEST-PC	192.168. |192.168	68.5302M	672512
192.168. | -TEST-PC	192.168. |192.168	67.0299M	663662
TEST-PC |192.168	192.168. |192.168.	65.1793M	672512
-TEST-PC |192.168	192.168. |192.168.	63.1922M	664609
TEST-PC |192.168	192.168. |192.168.	61.5288M	687602

4. pielikums Tīkla plūsmu analīzes sistēmas resursdatoru nosaukumi un kopējā lietojuma tabula



5. pielikums Tīkla plūsmu analīzes sistēmas SNMP protokola iekšējā tīkla resursdatoru saziņas karte



6. pielikums Tshark atgrieztās datu struktūras piemērs

Tshark rīka apstrādes piemērs datu pakai uz adresi apollo.lv. Tshark rīkam tiek padotas papildus opcijas "-T ek", lai iegūtu JSON formātu. Var pamanīt izanalizētos pakas slāņus: Ethernet, IP, TCP, HTTP, DATA-TEXT-LINES


```
{
 "timestamp": "1495717852577",
 "layers": {
  "frame": {
   "frame_frame_interface_id": "0",
   "frame_frame_encap_type": "1",
   "frame_frame_time": "May 25, 2017 16:10:52.577996000 FLE Summer Time",
   "frame_frame_offset_shift": "0.000000000",
   "frame_frame_time_epoch": "1495717852.577996000",
   "frame_frame_time_delta": "0.000002000",
   "frame_frame_time_delta_displayed": "0.000002000",
   "frame_frame_time_relative": "3.842081000",
   "frame_frame_number": "79",
   "frame_frame_len": "646",
   "frame_frame_cap_len": "646",
   "frame_frame_marked": "0",
   "frame_frame_ignored": "0",
   "frame_frame_protocols": "eth:ethertype:ip:tcp:http:data:data:data-text-lines"
  },
  "eth": {
   "eth_eth_dst": "60:a4:4c:71:91:7b",
   "eth_dst_eth_dst_resolved": "AsustekC_71:91:7b",
   "eth_dst_eth_addr": "60:a4:4c:71:91:7b",
   "eth_dst_eth_addr_resolved": "AsustekC_71:91:7b",
   "eth_dst_eth_lg": "0",
   "eth_dst_eth_ig": "0",
   "eth_eth_src": "d8:50:e6:42:ca:78",
   "eth_src_eth_src_resolved": "AsustekC_42:ca:78",
   "eth_src_eth_addr": "d8:50:e6:42:ca:78",
   "eth_src_eth_addr_resolved": "AsustekC_42:ca:78",
   "eth_src_eth_lg": "0",
   "eth_src_eth_ig": "0",
   "eth_eth_type": "0x00000800"
  },
  "ip": {
   "ip_ip_version": "4",
   "ip_ip_hdr_len": "20",
   "ip_ip_dsfield": "0x00000000",
   "ip_dsfield_ip_dsfield_dscp": "0",
   "ip_dsfield_ip_dsfield_ecn": "0",
   "ip_ip_len": "632",
   "ip_ip_id": "0x000004a1",
   "ip_ip_flags": "0x00000002",
   "ip_flags_ip_flags_rb": "0",
   "ip_flags_ip_flags_df": "1",
```

```
            "ip_flags_ip_flags_mf": "0",
            "ip_ip_frag_offset": "0",
            "ip_ip_ttl": "58",
            "ip_ip_proto": "6",
            "ip_ip_checksum": "0x00004644",
            "ip_ip_checksum_status": "2",
            "ip_ip_src": "78.28.227.182",
            "ip_ip_addr": "192.168.1.32",
            "ip_ip_src_host": "78.28.227.182",
            "ip_ip_host": "192.168.1.32",
            "ip_ip_dst": "192.168.1.32",
            "ip_ip_dst_host": "192.168.1.32",
            "ip_text": "Destination GeoIP: Unknown"
          },
          "tcp": {
            "tcp_tcp_srcport": "80",
            "tcp_tcp_dstport": "63585",
            "tcp_tcp_port": "63585",
            "tcp_tcp_stream": "3",
            "tcp_tcp_len": "592",
            "tcp_tcp_seq": "43801",
            "tcp_tcp_nxtseq": "44393",
            "tcp_tcp_ack": "418",
            "tcp_tcp_hdr_len": "20",
            "tcp_tcp_flags": "0x00000018",
            "tcp_flags_tcp_flags_res": "0",
            "tcp_flags_tcp_flags_ns": "0",
            "tcp_flags_tcp_flags_cwr": "0",
            "tcp_flags_tcp_flags_ecn": "0",
            "tcp_flags_tcp_flags_urg": "0",
            "tcp_flags_tcp_flags_ack": "1",
            "tcp_flags_tcp_flags_push": "1",
            "tcp_flags_tcp_flags_reset": "0",
            "tcp_flags_tcp_flags_syn": "0",
            "tcp_flags_tcp_flags_fin": "0",
            "tcp_flags_tcp_flags_str":
"\u00c2\u00b7\u00c2\u00b7\u00c2\u00b7\u00c2\u00b7\u00c2\u00b7\u00c2\u00b7\u00c2\u00b7\u00c2\u00b7\u00c2\u00b7\u00c2\u00b7AP\u00c2\u00b7\u00c2\u00b7\u00c2\u00b7\u00c2\u00b7
",
            "tcp_tcp_window_size_value": "31",
            "tcp_tcp_window_size": "15872",
            "tcp_tcp_window_size_scalefactor": "512",
            "tcp_tcp_checksum": "0x000060d6",
            "tcp_tcp_checksum_status": "2",
            "tcp_tcp_urgent_pointer": "0",
            "tcp_tcp_analysis": "",
            "tcp_analysis_tcp_analysis_initial_rtt": "0.008572000",
            "tcp_analysis_tcp_analysis_bytes_in_flight": "3512",
            "tcp_analysis_tcp_analysis_push_bytes_sent": "44392",
            "tcp_tcp_segment_data": "74:d1:e8:05:51:5e:34:64:41:32:4a:86:02:ba:4e:66:94:37:e3:3c:db:95:dc:bf:b5:ff:14:cf:97:54:a9:56:
... "
          },
          "tcp_segments": "",
```

```
            "tcp_segments_tcp_segment": "79",
            "tcp_segments_tcp_segment_count": "31",
            "tcp_segments_tcp_reassembled_length": "44392",
            "tcp_segments_tcp_reassembled_data":
"48:54:54:50:2f:31:2e:31:20:32:30:30:20:4f:4b:0d:0a:53:65:72:76:65:72:3a:20:6e:67:69:6e: ... ",
            "http": {
              "http_text": "Content-encoded entity body (gzip): 43980 bytes -> 266337 bytes",
              "text__ws_expert": {
                "_ws_expert_http_chat": "",
                "_ws_expert__ws_expert_message": "HTTP\/1.1 200 OK\\r\\n",
                "_ws_expert__ws_expert_severity": "2097152",
                "_ws_expert__ws_expert_group": "33554432"
              },
              "text_http_request_version": "HTTP\/1.1",
              "text_http_response_code": "200",
              "text_http_response_phrase": "OK",
              "http_http_server": "nginx\/1.2.1",
              "http_http_response_line": "Content-Encoding: gzip\r\n",
              "http_http_date": "Thu, 25 May 2017 13:10:52 GMT",
              "http_http_content_type": "text\/html; charset=utf-8",
              "http_http_transfer_encoding": "chunked",
              "http_http_connection": "keep-alive",
              "http_http_set_cookie": "APOLLOSESSID=95ovdgjln2ubcaukmlhm7n9100; path=\/",
              "http_http_cache_control": "no-store, no-cache, must-revalidate, post-check=0, pre-check=0",
              "http_http_content_encoding": "gzip",
              "http_http_response": "1",
              "http_http_response_number": "1",
              "http_http_time": "0.079176000",
              "http_http_request_in": "24",
              "text_text": "\\r\\n",
              "text_http_chunk_size": "0",
              "text_data": {
                "data_data_data": "4a:fd:8b:66:2c:89:ae:07:5c:7b:5c:9a:54:6a:18:a7:49:07:5c:54:28:b3:69:d5:f0:2a:ab:d5:ec:e4:2b:f4:2e ... ",
                "data_data_len": "19394"
              },
              "text_http_chunk_boundary": "0d:0a",
              "http_http_file_data": "<!DOCTYPE html>\n<html lang=\"lv\" class=\"lv\" xmlns=\"http:\/\/www.w3.org\/1999\/xhtml\"
xmln ... "
            },
            "data-text-lines": {
              "data-text-lines_text": "<\/html>"
            }
          }
        }
```

7. pielikums Tīkla plūsmu monitoringa sistēmas analīzes piemērs