ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«Ғ.ДАУКЕЕВ АТЫНДАҒЫ АЛМАТЫ ЭНЕРГЕТИКА ЖӘНЕ БАЙЛАНЫС УНИВЕРСИТЕТІ»
коммерциялық емес акционерлік қоғамы
телекоммуникациялық желілер және жүйелер кафедрасы

«ҚОРҒАУҒА ЖІБЕРІЛДІ»
Кафедра меңгерушісі Темырканова Э.К, PhD докторы, доцент
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «____» _____2020ж.
(қолы)

**ДИПЛОМДЫҚ ЖОБА**

Тақырыбы: Трафикке талдау жасаудың қосымша әдістерін әзірлеу үшін DPI технологиясын қолдану
Мамандығы 5В071900-Радиотехника, электроника және телекоммуникациялар
Орындаған Ахметбек Шыңғыс Алтайұлы Тобы РЭТ(ИКТ)-16-1
(Т.А.Ж.)
Ғылыми жетекшісі Чежимбаева Катипа Сламбаевна, т.ғ.к., профессор
(ғылыми дәрежесі, атағы, Т.А.Ж.)
Кеңесшілер:
экономикалық бөлім бойынша: Тузелбаев Бакберген Ибадиллаевич, доцент
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «_02_» _____06_____2020ж.
(қолы)
өмір тіршілігі қауіпсіздігі бойынша:
Жандаулетова Фарида Рустембековна, доцент
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «_14_» _____05_____2020ж.
(қолы)
есептеу техникасын қолдану бойынша: Чежимбаева Катипа Сламбаевна, т.ғ.к., профессор
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «_01_» _____06_____2020ж.
(қолы)
Нормабақылаушы: Мухамеджанова Альмира Далелханкызы, доцент
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «__04__» _____06_____2020ж.
(қолы)
Пікір беруші: _____
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «____» _____2020ж.
(қолы)

Алматы 2020

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«Ғ.ДАУКЕЕВ АТЫНДАҒЫ АЛМАТЫ ЭНЕРГЕТИКА ЖӘНЕ БАЙЛАНЫС УНИВЕРСИТЕТІ»
коммерциялық емес акционерлік қоғамы

Ғарыштық инженерия және телекоммуникациялар институты
телекоммуникациялық желілер және жүйелер кафедрасы

Мамандығы 5В071900 – Радиотехника, электроника және телекоммуникациялар

Дипломдық жобаны орындауға берілген

**ТАПСЫРМА**

Студент Ахметбек Шыңғыс Алтайұлы
(Т.А.Ж.)

Жобаның тақырыбы Трафикке талдау жасаудың қосымша әдістерін әзірлеу үшін DPI технологиясын қолдану

2019 ж. «11» __11__ № 147 университет бұйрығымен бекітілді.

Аяқталған жобаны тапсыру мерзімі «25» __05__ 20 20 ж.

Жобаға алғашқы деректер (талап етілетін зерттеу (жоба) нәтижелерінің параметрлері және зерттеу нысанының алғашқы деректері):
TCP/IP дестесінің құрылымдық сұлбасы
DPI жүйесінің байланыс операторының желісіне қосылу сызбанұсқасы
Желілік деңгей үшін TCP хаттамасының тұрақтысы PROTOCOL_TCP = 6
Желілік деңгей үшін UDP хаттамасының тұрақтысы PROTOCOL_UDP = 17
Саздақ типті топырақтың есептік үлестік кедергісі ρ = 100 Ом·м
i – түрлі электржабдықтың қуатты пайдалану коэффициенті $K_i$ = 0,7

Диплом жобасындағы әзірленуі тиіс мәселелер тізімі немесе диплом жобасының қысқаша мазмұны:
Кіріспе
1. Deep Packet Inspection технологиясы
2. Python арқылы желілік дестелерді алу
3. Алынған дестелік трафикті талдау
4. Өмір тіршілігі қауіпсіздігі
5. Экономикалық бөлім
Қорытынды
Әдебиеттер тізімі

Графикалық материалдардың (міндетті түрде дайындалатын сызбаларды көрсету) тізімі:

TCP – сегментінің пішімі

Таңбаланған желі үлгісі

DPI жүйесін байланыс оператордың желісіне қосу сызбанұсқасы

Bypass серверінің жұмыс істеу тәртібінің үлгісі

P2NMAP-Capture.py бағдарламасының жұмыс істеу алгоритмі

IPObservationDictionary класының алгоритмі

ISObservationDictionary класының алгоритмі

P2NMAP-Capture.py бағдарламасының шығыс нәтижесі

P2NMAP-Anaysis.py бағдарламасының коды

Белгілі бір талдау әдістерін ұсынатын бағдарламаның мәзірі

Сервер / клиент байланыс гистограммасы

Деректерді өңдеу орталығының орналасу жоспары

Негізгі ұсынылатын әдебиеттер:

1. Горнак А. Технология DPI для широкополосного доступа // Технологии и средства связи. 2010. № 4, сентябрь. с. 66 – 67.

2. Гольдштейн Б.С. Глубокая инспекция пакетов DPI: проблемы и подходы // Вестник связи. 2018. № 9, сентябрь. с. 5 – 10.

3. Chet Hosmer. Python Passive Network Mapping P2NMAP 1st Edition. USA: Syngpress Imprint, 2015, ISBN: 978-0-12-802721-9 – 162 p.

4. Сенченко Ю. Некоторые аспекты высокоскоростной обработки трафика // Технологии и средства связи. 2013. № 1, январь. с. 8 – 9

5. Берлин А.Н. Основные протоколы Интернет: Учебное пособие / А.Н. Берлин – М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2012. – 504 с.: ил., табл.

6. Гольдштейн А.Б., Гольдштейн Б.С. Технология и протоколы MPLS. – СПб.: БХВ-Петербург. 2005. 304 с.

7. TCP/IP и DNS в теории и на практике. Полное руководство / Пер. с чеш. Рус. изд. под ред. М.В. Финкова и А.В. Анисимова. Серия «Полное руководство». – СПб.: Наука и Техника, 2006. – 608 с., ил

Жоба бойынша жобаның бөлімдеріне қатысты белгіленген кеңесшілер

| Бөлімдері | Кеңесшілері | Мерзімі | Қолы |
|---|---|---|---|
| Экономика | Тузелбаев Б.И. | 02.06.2020 | |
| Ө.Т.Қ. | Жандаулетова Ф.Р. | 14.05.2020 | |
| Негізгі бөлім | Чежимбаева К.С. | 17.02.2020-25.05.2020 | |
| Есептеу техникасы | Чежимбаева К.С. | 01.06.2020 | |
| Нормабақылау | Мухамеджанова А.Д. | 04.06.2020 | |
| | | | |
| | | | |

Диплом жобасын дайындау

КЕСТЕСІ

| № р/с | Бөлімдердің атауы, әзірленетін мәселелердің тізімі | Ғылыми жетекшіге ұсыну мерзімдері | Ескерту |
|---|---|---|---|
| 1 | Deep Packet Inspection технологиясы | 17.02.20-19.02.20 | Орындалды |
| 2 | Телекоммуникациялық желілердегі трафикті талдау | 20.02.20-24.02.20 | Орындалды |
| 3 | ISO OSI өзара әрекеттесу моделі | 25.02.20-27.02.20 | Орындалды |
| 4 | TCP/IP хаттамалар стегі | 28.02.20-03.03.20 | Орындалды |
| 5 | Дестелік трафикті талдау тереңдігінің деңгейлері | 04.03.20-06.03.20 | Орындалды |
| 6 | DPI технологиясының құрылымы | 09.03.20-11.03.20 | Орындалды |
| 7 | Python арқылы желілік дестелерді алу | 12.03.20-16.03.20 | Орындалды |
| 8 | Желілік белсенділікті бақылау | 17.03.20-19.03.20 | Орындалды |
| 9 | Python тілінде жазылған дестелік трафикті алатын негізгі бағдарламаны әзірлеу | 20.03.20-24.03.20 | Орындалды |
| 10 | Алынған деректерді сақтау | 25.03.20-27.03.20 | Орындалды |
| 11 | Python сөздіктерін қолдану | 30.03.20-01.04.20 | Орындалды |
| 12 | Алынған дестелік трафикті талдау | 02.04.20-06.04.20 | Орындалды |
| 13 | Талдау тәсілдерін әзірлеу | 07.04.20-09.04.20 | Орындалды |
| 14 | Желілік трафиктің талдау нұсқаларын баптау | 10.04.20-14.04.20 | Орындалды |
| 15 | Кәсіпорындағы еңбек жағдайларын талдау | 15.04.20-17.04.20 | Орындалды |
| 16 | Жасанды қорғаныстық жерлендіруді есептеу | 20.04.20-22.04.20 | Орындалды |
| 17 | Эвакуация уақытын есептеу | 23.04.20-27.04.20 | Орындалды |
| 18 | ҒЗЖ кеткен шығындарды есептеу | 28.04.20-29.04.20 | Орындалды |
| 19 | ҒЗЖ ықтимал (келісімді) бағасын анықтау | 30.04.20-04.05.20 | Орындалды |
| 20 | Қорытынды | 05.05.20-09.05.20 | Орындалды |

Тапсырманың берілген уақыты «17» __02__ 20 20 ж.

Кафедра меңгерушісі _____ Темырканова Эльвира Кадылбековна

(қолы) (Т.А.Ж.)

Жобаның ғылыми жетекшісі _____ Чежимбаева Катипа Сламбаевна

(қолы) (Т.А.Ж.)

Орындалатын тапсырманы қабылдаған студент_Ахметбек Шыңғыс Алтайұлы

(қолы) (Т.А.Ж.)

**Аңдатпа**

Дипломдық жұмыста DPI технологиясы, оның байланыс операторларының желілерінде жұмыс істеу принципі және функционалдық мүмкіндіктері қарастырылады. Желілік трафикті талдау үшін серверде жұмыс істейтін, жоғары деңгейлі Python тілінде жазылған бағдарлама әзірленді. «Өмір тіршілігі қауіпсіздігі» бөлімінде телекоммуникациялық шкафтың қорғаныстық жерге тұйықталуы және серверлік бөлмеде өрт кезінде эвакуациялаудың рұқсат етілген ұзақтығы есептелген. Сондай-ақ ғылыми-зерттеу жұмысын орындаудың еңбек сыйымдылығы анықталды.

**Аннотация**

В дипломной работе будет рассмотрена технология DPI, ее принцип работы в сети операторов связи и функциональные возможности. Для анализа сетевого трафика, разработана программа, написанная на высокоуровневом языке Python, которая будет работать на сервере. В разделе «Безопасность жизнедеятельности» рассчитано защитное заземление телекоммуникационного шкафа и допустимая продолжительность эвакуации при пожаре в серверной комнате. Также определена трудоемкость выполнения научно-исследовательской работы.

**Abstract**

In this paper, we consider the DPI technology, its functionality and principles of operation in the network of Telecom operators. A program has been developed for analyzing network traffic which will work on a server. The program is written on the high-level Python programming language. The protective grounding of the telecommunication racks and the permissible duration of evacuation in the event of a fire in the server room are calculated in the section «The Safety of Life». The complexity of performing research work is also determined.

## Мазмұны

## Кіріспе

Бір желі бойынша мультимедиалық қосымшалардың трафигін және өткізу қабілеттілігіне қойылатын әр түрлі талаптары бар деректер трафигін тарату мақсатында, оларға қолданылатын талаптарға байланысты желілік трафиктің түрлерін саралау және өңдеу мүмкіндігін қамтамасыз ететін үрдістер қажет.

Бұл дипломдық жобаның мақсаты телекоммуникациялық желілердегі жоғары жылдамдықты дестелік трафикті талдауға арналған бағдарламаны жобалау болып табылады. Телекоммуникациялық жүйелерде деректердің дестелер арқылы алмасу трафигін тиімді тарату маңызды жобалаудың бірі болып саналады. Осындай жобаларды есептеу үшін DPI технологиясы негізінде әртүрлі құрылғылар мен бағдарламалар қолданылады. Бұл технология дестелерді өзгертуге, сүзуге немесе қайта бағыттауға қатысты осындай әрекеттерді жасайды. Бұл тәсілде анализатор әр дестенің мазмұнын толығымен қарастырады. Бұрынғы технологиялардан маңызды айырмашылықтардың бірі – DPI негізіндегі жүйелер дестелердің мазмұнына ғана емес, сонымен қатар белгілі бір желілік бағдарламалар мен хаттамаларға тән басқа сипаттамаларға қатысты шешім қабылдай алады. Ол үшін статистикалық талдау қолданылуы мүмкін. Алдыңғы тәсілдермен салыстырғанда, технологиялық қосымшалардың тізімі едәуір кеңейтілді – классификация, жолақтарды шектеу, таңбалау және т.б.

DPI технологиясы микросхемалардың (процессорлардың) есептеу мүмкіндіктерінің жылдам өсуіне, желінің деректерін неғұрлым толық және нақты талдау үшін олардың жылдамдығы мен мүмкіндіктерінің артуына байланысты дамыды.

Бұл жұмыстың бағдарламасы жоғары деңгейлі Python тілінде жазылған. Сондай бағдарламалау тілін қолдану құрылғылардың көпшілігінде есептеулерді жүргізуге мүмкіндік береді. Жобаланған бағдарлама сервер құрылғысының бағдарламалық жасақтамасының ішіне орнатылатын болады. Мұндай технологияларды ірі байланыс операторларынан бастап, корпоративтік желі администраторларына дейін пайдаланады.

Қойылған мақсатқа жету үшін желілік трафикті терең талдау жүйесінің белгіленуін, жұмыс істеу принципін және қолдану аясын түсініп, осы салада қолданылатын шешімдерді қарастыру қажет. Сондай-ақ, бұл жүйелердің құрылымын және ерекшеліктерін талдап, желінің нақты жұмыс жағдайына жақын тестілеуін жүргізу керек.

Бағдарламалық-аппараттық деңгейде желілік дестелік трафикті алу және талдау кезінде болып жатқан үрдістерді түсіну, болашақ желілік технологияларды дамыту бағыты туралы түсінік береді.

## 1 Deep Packet Inspection технологиясы

### 1.1 Телекоммуникациялық желілердегі трафикті талдау

Бүкіл әлем бойынша телекоммуникация саласы ортақ IP-инфрақұрылымға қолданыстағы және жаңа желілік қызметтерді дамыту үрдісінде тұр. Ал жаһандық IP-желілер үлкен мүмкіндіктер тудырғанымен, олар да жаңа мәселелердің туындауына алып келді. Ғаламтор қызметтерін жеткізушілер үшін осындай маңызды мәселелердің бірі – өз желісіндегі ағынды бақылау және талдай білу.

Ағынды талдау міндеттері телекоммуникациялық желілерді дамытудың ерте сатысында пайда болды. Олар, атап айтқанда, әртүрлі тарифтік жоспарлары бар жергілікті, қалааралық және халықаралық қосылыстардың бөлінуімен, содан кейін байланыс желілеріндегі алаяқтықпен күрес мәселелерінің кең ауқымымен байланысты болды.

Мысалы, ғаламтор белсенділіктің үлкен және өсіп келе жатқан үлесі P2P-ағынға тиесілі. Әдетте ол қызмет көрсетушілерге табыс әкелмейді, бірақ желі ресурстарының аз бөлігін алады. Нәтижесінде бақыланбайтын P2P-ағын шығындарды арттырады және желі инфрақұрылымын құруға қосымша күш салуды талап етеді. Сонымен қатар, қызмет көрсетушілер қандай да бір желілік бағдарламалардың бақыланбауы қызмет көрсету деңгейі туралы SLA келісімнің бұзылуына әкеліп, кіріс әкелетін қызметтердің (мысалы, VoIP) бұзылуына әкеліп соқтыратын шығындарға ұшырауы мүмкін. SLA бұзылуы сондай - ақ DDoS таратылған шабуылдарды тудыруы мүмкін [1].

IP-желісін қолданушының шындыққа сәйкес келмейтін ағын түрін көрсете отырып DiffServ қолдану мүмкіндігін немесе бөгде көліктік порттары мен хаттамаларының бағдарламаларын пайдалану мүмкідігін атап өтейін. Мұндай пайдаланушы трафигі дифференциалды қызмет көрсету жүйесінің сапасын бұзады немесе провайдер желісінде қауіпсіздік жүйесін аттап өтуге мүмкіндік береді [2].

Осы және басқа да ұқсас мәселелерді шешу стандартты коммутаторлардың, маршрутизаторлардың және желі аралық экрандардың мүмкіндіктерінен асып түседі. Сондықтан мұндай құрылғылар, мысалы, HTTP хаттаманың үстінен берілетін бағдарламаларды ажырата алмайды, онда Web-беттен басқа дауыс, бейне, жедел хабарлар және сол P2P-ағын берілуі мүмкін.

Барлық осы, сондай - ақ басқа да жағдайларда операторға DPI дестелерді терең зерттеу технологиясы көмектесе алады. DPI термині дестелердің мазмұнын тексеруге және осы мазмұн негізінде белгілі бір әрекеттерді орындауға мүмкіндік беретін құрылғылар мен технологияларға жатады.

Егер пошта ұқсастығын пайдалансаңыз, онда десте - пошта хаттарының аналогы, конверттегі мекенжай дестенің тақырыбына ұқсас, пайдалы жүктеменің аналогы ішіндегі ақпарат. DPI-пошта хат-хабарларын өңдеу жөніндегі шешімдерді қабылдау аналогы тек мекенжай негізінде ғана емес, сонымен қатар хат мазмұнын ескере отырып.

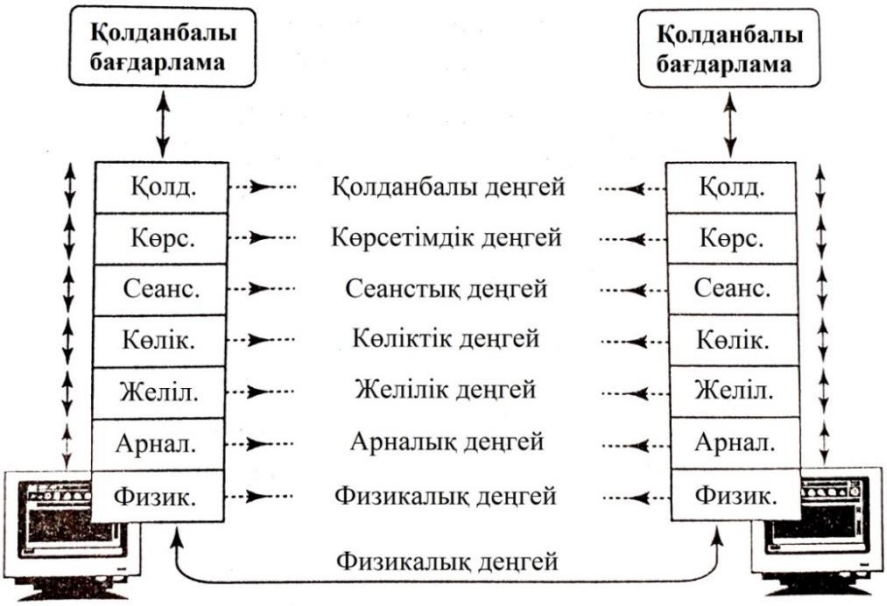
Кейде жалпы термині – DPP қолданылады, ол түрлендіру, сүзу немесе қайта бағыттау сияқты дестелерге қатысты әрекеттерді білдіреді.

### 1.2 ISO OSI өзара әрекеттесу моделі

IP-желілердің трафигін тексерудің бірнеше технологиясы бар, олар талдау тереңдігімен ерекшеленеді.

Бұл технологияларды санамас бұрын ISO OSI моделінің деңгейлерін қарастырайық.

ISO OSI моделі аясында екі компьютер арасындағы өзара әрекеттесу сұлбасы 1.1 суретте көрсетілген.




1.1 сурет – Жетідеңгейлі ISO OSI моделі

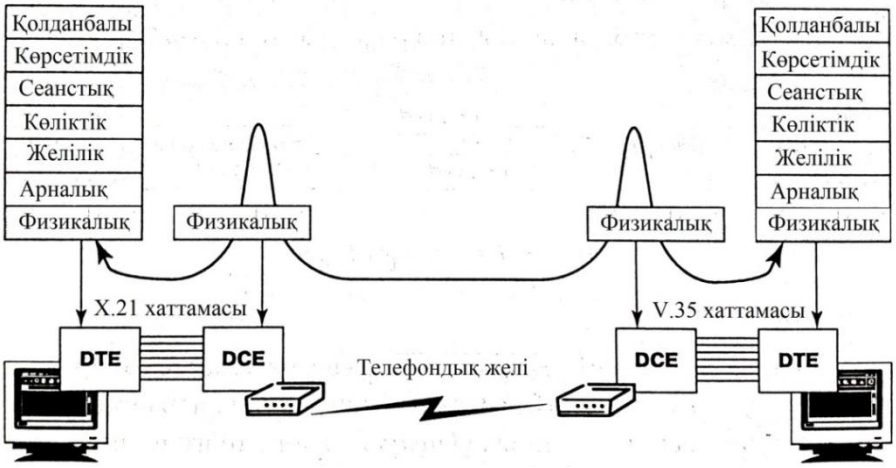
Ресми тілде физикалық қабат DTE және DCE арасындағы физикалық тізбекті іске қосуға, байланыстыруға және өшіруге жауап береді. Сонымен қатар, ол DCE арасындағы байланыс үшін жауап береді (1.2 сурет). DTE арқылы біз, мысалы, компьютерді немесе роутерді, ал DCE бойынша, әдетте, модем немесе мультиплексорды айтамыз.

Мысалы, физикалық деңгейде, шын мәнінде, ашық жүйелер арасында ақпарат беру жүзеге асырылады. Ақпарат екілік разряд болып табылатын және базалық ақпараттық бірліктер болып табылатын биттерде беріледі.

Физикалық деңгейдің хаттамалары ақпаратты беру жүзеге асырылатын сигналдар мен ортаның физикалық сипаттамаларын ескереді (сымның, кабельдің, ажыратқыштардың және т.б.). Ақпаратты беру үрдісінің өзі өзгертіліп жіберілетін деректерді көрсететін сигналды беру болып табылады. Сигнал әртүрлі болуы мүмкін. Кабель желілері кернеулердің әртүрлі мәндері нөлдер мен бірліктерді (бит мәндерін) білдіреді, деректерді дыбыстық беру кезінде осы мақсаттар үшін дыбыстық модуляция (жиіліктердің әртүрлі мәндерін) және т.б. қолданылады.

Қазіргі заманғы технологиялық жетістіктер ақпаратты тарату үшін әртүрлі тасымалдаушыларды қолдануға мүмкіндік береді: барлық мүмкін болатын электр кабельдері, оптоталшық, дециметрлік толқындар және тіпті қарапайым жарық. Олардың әрқайсысын пайдалану физикалық деңгейдегі хаттамалардың жеке жиынтығы арқылы іске асырылуы мүмкін.

Физикалық деңгейден жоғары барлық басқа деңгейлер өзара әрекеттесетін ортадан тәуелсіз. Мысалы, егер телефондық байланыс желісін оптикалық талшыққа ауыстырса, онда физикалық деңгейдегі хаттамалар өзгереді, ал қалған барлық деңгейлер өзгеріссіз болып қалады.




1.2 сурет – DTE және DCE байланыс схемасы

Физикалық деңгейде физикалық тізбек құрылады. Физикалық тізбекте екі компьютер арасында басқа құрылғылар жиі орналастырылады, мысалы, сандық деректерді телефон желісі бойынша берілетін аналогтық сигналға модуляциялайтын модемдер және т.б.

Физикалық деңгейдегі хаттамалар мыналарды көрсетеді:

- электр сигналдары (мысалы, + IV);
- қосқыштар түрлері (мысалы, V.35);
- тасушының түрі: бұралған қос қалайы сым, коаксиалды кабель, оптикалық талшық және т.б;
- жіберу жылдамдығы (мысалы, 128 Кб / с);
- модуляция (мысалы, ФА, ЖМ және т.б.);
- кодтау (мысалы, RZ, NRZ және т. б.);
- синхрондау: синхронды / асинхронды байланыс, уақыт көзі және т.б.

Егер физикалық деңгей физикалық орта бойынша деректерді беруді іске асырса, онда арналық деңгей осы таратуды қамтамасыз етуге жауап береді. Ол бір жергілікті желі шеңберінде тораптардың байланысы үшін жауапты. Арналық деңгей деректер кадрларын тізбекті желіге орналастыруды қамтамасыз етеді. Ол компьютерлерді ғаламдық желіде байланыстырады, ал жергілікті желілерде желі ішінде деректермен алмасуды қамтамасыз етеді.

| Тақырып (*Header*) | Деректер (*Payload*) | Соңғы өріс (*Trailer*) |
|---|---|---|

1.3 сурет – Арналық фрейм

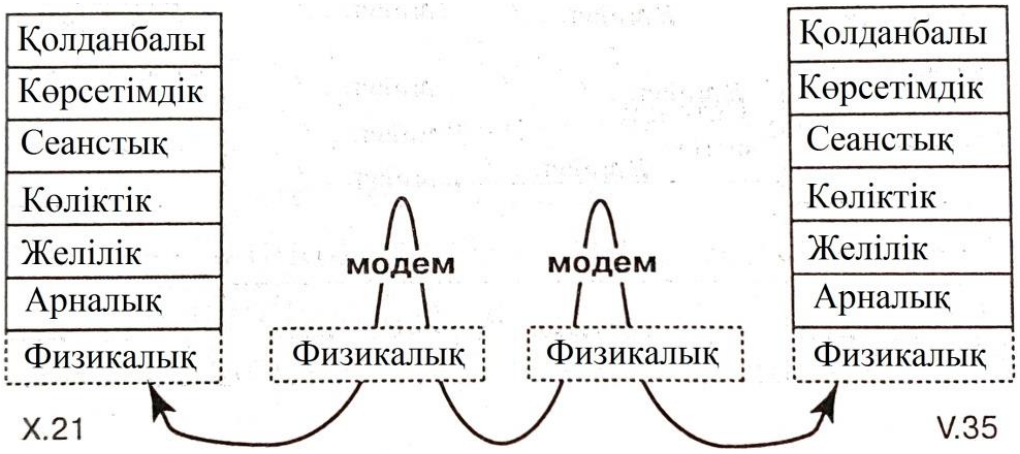
Арна деңгейінде деректерді берудің негізгі бірлігі деректер фреймі болып табылады (1.3 сурет). Деректер шеңбері тақырыптардан (Header), жіберілген мәліметтерден (Payload) және соңғы өрістен (Trailer) тұрады. Деректер фреймі тақырыбында алушының арналық мекенжайын, жіберушінің арналық мекенжайын және басқа басқарушы ақпаратты қамтиды.

Соңғы өрісте, әдетте, берілетін деректерден бақылау коды бар. Бақылау кодының көмегімен деректерді тарату барысында деректердің бүлінбеуін анықтауға болады. Берілген деректерде желілік деңгейінің дестесі бар.

Егер 1.2 суретті қарастырсақ, арналық деңгей DTE мен DCE арасындағы байланыс мәселелерімен айналыспайды (арналық деңгей DCE «көрмейді»). Ол DTE арасында фреймдердің алмасуымен айналысады. Яғни, DCE мәселесі физикалық деңгейде.

1.4 суретте көрсетілгендей, әрбір байланыс торабы үшін Физикалық деңгейде әртүрлі хаттамалар қолданылуы мүмкін. Бұл жағдайда бір байланыс торабы X.21 хаттамасын, ал екіншісі – V.35 хаттамасын қолданады. Бұл тек тізбекті арналарға ғана емес, сондай-ақ жергілікті желілерге да қатысты.

Жергілікті желілерде қосылыстың екі түйіні арасында коммутатор орналастырылған кезде, бір арналы хаттаманың арналық кадрларын басқа арна хаттамасының кадрларына айналдырған кезде қиынырақ жағдай туындауы мүмкін (мысалы, FDDI-дан Ethernet-ке), нәтижесінде, әрине, физикалық деңгейде әртүрлі хаттамалар қолданылатын болады.




1.4 сурет – Арналық деңгейдегі байланыс

Арна интерфейсі, мысалы, Ethernet-картасы немесе сериялық желінің порты болуы мүмкін. Арна интерфейсінде LAN ішінде ерекше болатын арна мекенжайы бар.

Желілік деңгейдің болуы желіаралық өзара байланысын іске асыру қажеттілігімен байланысты. Тек ең қарапайым желілер бір жергілікті желіден тұрады. Әдетте желілер кең құрылымды болады және бірнеше біріккен жергілікті желілерден тұрады [3].

Неге бір үлкен жергілікті желіні жасауға болмайды? Өйткені, жергілікті желіде деректерді таратудың, арналық деңгеймен жүзеге асырылғанда, кең тарату сипаты бар, яғни деректер барлық компьютерлерге бірден беріледі. Қауіпсіздік мәселелерінен басқа, техникалық аспект – жеке алынған учаскелерде ағынды (берілетін деректер көлемін) едәуір ұлғайту, бұл өз кезегінде желі жұмысының баяулауын туындатады. Ғаламтор сияқты ғаламдық желінің болуы, мұндай мәселені қою кезінде мүлдем мүмкін емес еді.

Желі деңгейі арналық деңгейге қарағанда, желіге қосылған әрбір компьютер үшін жеке логикалық мекенжайлармен жұмыс істейді.

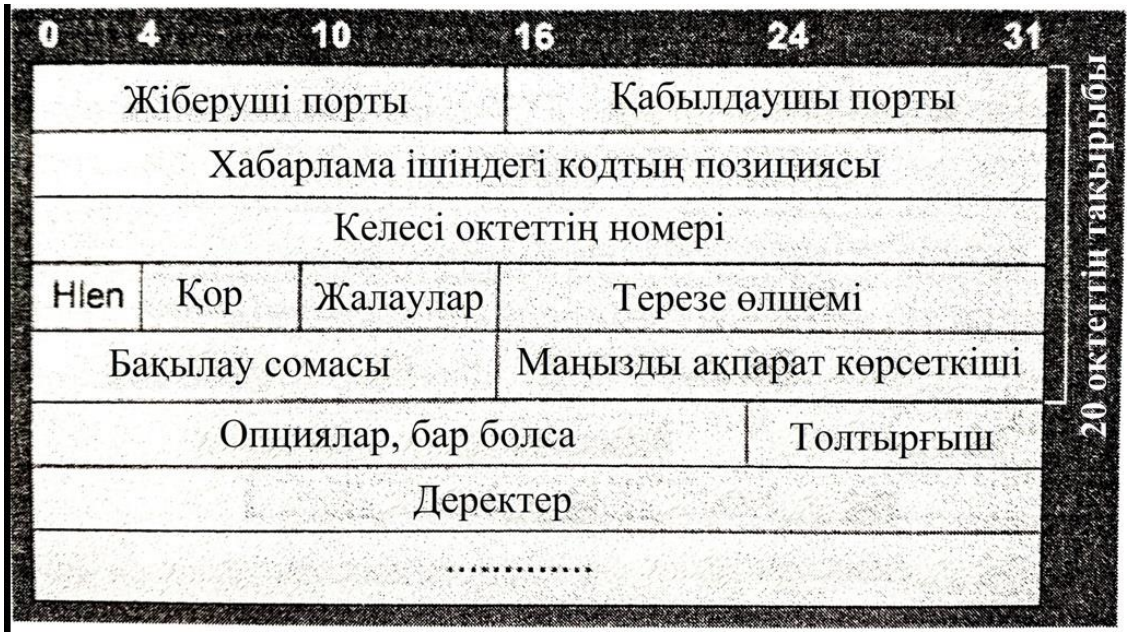
Негізгі желілік хаттама ретінде стекте IP хаттамасы қолданылады, ол бастапқыда жергілікті және жаһандық байланыстарға біріктірілген көптеген жергілікті желілерден тұратын құрамдас желілерде дестелерді тарату хаттамасы ретінде жобаланған. Сондықтан IP хаттамасы күрделі топологиясы бар желілерде жақсы жұмыс істейді, қосалқы жүйелерді тиімді пайдаланып және төмен жылдамдықты байланыс желілерінің өткізу қабілетін үнемді жұмсай отырып жұмыс жасайды. IP хаттамасы дейтаграммалық хаттама болып табылады, яғни ол дестелерді тағайындалған торабқа дейін жеткізуге кепілдік бермейді.

Желіаралық өзара жұмыс жасайтын деңгейіне маршруттау кестелерін құрастырумен және түрлендірумен де байланысты барлық хаттамалар жатады. RIP және OSPF маршруттық ақпаратты жинау хаттамалары, сондай-ақ ICMP желіаралық хабарламаларды басқару хаттамасы сияқты. Соңғы хаттама желілік маршрутизаторлар мен дестелік көздер торабы арасында қате туралы ақпарат алмасуға арналған. ICMP арнайы дестелердің көмегімен дестені жеткізу мүмкін еместігі туралы, фрагменттерден дестені жинау ұзақтығы туралы, параметрлердің ауытқушылық шамасы туралы, жіберу маршрутының және қызмет көрсету түрінің өзгеруі туралы, жүйенің күйі және т.б. хабарланады.

Сонымен, көліктік деңгейінің негізгі хаттамасы – TCP беруді басқару хаттамасы, виртуалды қосылыстар арқылы қашықтықтағы қолданбалы үрдістер арасындағы сенімді хабарлама алмасуды қамтамасыз етеді. TCP – бұл жоғары деңгейдегі хаттама, ол мәліметтер ағынын TCP сегменттері деп аталатын тізбектерге бөледі және оларды IP көмегімен таратады [4].

Осылайша, TCP көліктік қабатының хаттамасының мақсаты Internet сияқты дестелік желілерде жұмыс жасайтын жіберуші мен қабылдағыш арасында ақпараттардың сенімді, жүйелі түрде берілуін қамтамасыз ету болып табылады. Пайдаланушы деректерді байт ағыны ретінде алатынына қарамастан, TCP оларды дестелер түрінде жібереді. Олардың әрқайсысында

тіркелген TCP тақырыптары болады. TCP-сегментінің пішімі 1.5 суретте көрсетілген.




1.5 сурет – TCP-сегментінің пішімі

«Hlen» өрісі – сегмент тақырыбының ұзындығын анықтайды, өйткені тақырыптағы айнымалы ұзындық опцияларының өрістері болуы мүмкін. Бұдан әрі болашақта пайдалануға арналған «Қор» өрісі бар, қазіргі уақытта нөлденуі тиіс. «Терезе өлшемі» өрісі, келесі болып келетін «Октет номері» өрісінде көрсетілген байттан кейін, алушы қанша октеттерді қабылдауға дайын екенін хабарлайды (ACK=1 жалауы). Терезенің енінің мәні сессия кезінде өзгеруі мүмкін. Бұл өрістің нөлге тең мәні де рұқсат етіледі және байттардың «Октет номері» өрісінде көрсетілгенге дейін алынғанын көрсетеді, бірақ қабылдаушы деректерді уақытша қабылдай алмайды. Жаңа ақпаратты жіберуге рұқсат келесі, бірақ «Терезе өлшемі» өрісінің нөлдік емес мәнімен келетін «Октет номері» өрістің мәні бар сегментті жіберу арқылы берілуі мүмкін. «Бақылау сомасы» өрісі хабарламаның тұтастығын қамтамасыз етуге арналған. Бақылау қосындысы 1 модуль бойынша орындалады.

TCP хаттамасының бастапқы нұсқасының ерекшелігінде оның үш негізгі сипаттамасы қарастырылады: көптеген қосылыстар, деректерді беру сенімділігі, алушы тарапынан ағындарды басқару.

TCP хаттамасының көмегімен барлық деректерді жіберу әрбір қосылысты орнату мүмкіндігін растауға мүмкіндік беретін шағын дестелермен алмасудан тұратын үш жақты квитирлеу үрдісінің көмегімен орнатылатын байланыс арқылы жүзеге асырылады. TCP-ге қосылу опциялары да осы үрдісте келісіледі. Бір пайдаланушының бірнеше қосылым жасай алатындығы және бірнеше деректер ағынына ие болғаны жөн. Ол үшін әр TCP

қосылымы желілік адрестермен және қосымша идентификатормен (портпен) анықталады. Деректер көзінің порт нөмірі мен тағайындалған порт нөмірі әр TCP дестесінің тақырыбында болғандықтан, әр сегмент тиісті қосылыммен байланыстырылуы мүмкін. Сонымен қатар, байланыс дегеніміз – абстрактілі түсінік, ол тек желі қолданушылары сақтайтын жазбаларда болады. Егер сегмент дұрыс қосылуға байланысты болмаса, TCP жіберілген қате туралы жіберушіге хабарлауға тырысады.

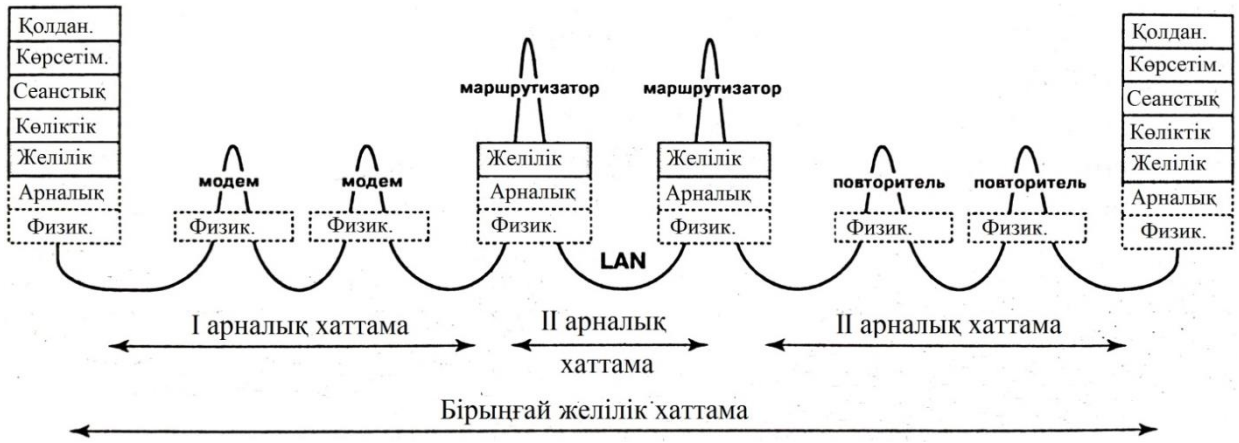
Егер қосылуға қатысушылардың бірінде жіберетін деректер қалмаса, ол байланысты жаба алады, ол үшін басқа тарапқа тиісті хабарлама жіберіледі. Дегенмен, байланыс толығымен жабылмаған, сондықтан екі жақ оны жабғысы келетінін білдірмейінше жадтан өшірілмейді.

Деректерді берудің сенімділігі екі параметрмен қамтамасыз етіледі. Біріншісі – мәліметтердің сол ретпен келген-келмегенін тексеру. Екіншісі – жоқ немесе артық деректердің бар екенін тексеру.

Бірінші жағдайда TCP жіберетін әрбір сегментте өзінің 32 биттік реттік нөмірі бар. Алушы деректер байты ағымында болмаған кезде TCP пайдаланушыға деректерді бере алмайды. Әрбір жіберілген TCP сегменті атауында байланыс орнату кезінде жіберуші таңдаған сегменттегі бірінші деректер байтының реттік нөмірі болады. Бұл механизм жіберілген деректердің дұрыс тәртібін анықтауға жеткілікті, алайда жекелеген дестелерді жоюға немесе кешіктіруге болатын дестелік желілерде жұмыс істеу кезінде мәліметтерді беруде қиындықтар туындайды.

Барлық жіберілген деректердің алушыға жеткізілгенін тексеру алушыдан жіберушіге оралған TCP дестелерінің тақырыптарындағы қабылдауды растау нөмірімен қамтамасыз етіледі. Бұл сан жіберілген мәліметтердің TCP тақырыбындағы реттік нөмірлерге сәйкес келеді. Алушы барлық деректерді қатесіз қабылдағанын білдіреді, ал жіберуші жіберген барлық сегменттердің қабылданғаны туралы растауды күтеді. Егер ол оны алмаса, ол деректерді қайта жіберуге мәжбүр болады.

WAN компьютерлер арасындағы желілерде, әдетте, бір немесе бірнеше маршрутизаторлар бар. Көрші маршрутизаторлар арасында арналық деңгейде тікелей байланыс ұйымдастырылған. Маршрутизатор желілік дестені деректер фреймінен (бір арналық хаттамасынан) алып, оны басқа арнаға жібермес бұрын, басқа деректер фреймінің (әдетте басқа арналық хаттамамның) қабықшасына орналастырады. Желілік деңгей физикалық және арналық деңгейде жұмыс істейтін құрылғыларды (модемдерді, қайталағыштарды, коммутаторларды және т.б.) «көрмейді» (1.6 сурет).




1.6 сурет – Желілік деңгейдегі байланыс

Желілік қабат үшін қосылымның екі түйіні арасындағы жолда қандай арна хаттамалары пайдаланылғандығы маңызды емес.

Желілік интерфейс, мысалы, Ethernet картасы немесе сериялық порт болуы мүмкін. Желілік интерфейс WAN ішінде бірмәнді мекенжайға ие.

Көліктік деңгей желілік деңгейдің кейбір функцияларына ұқсас функцияларға ие. Оның болуы желілік деңгейдегі хаттамалар логикалық мекенжайлар бойынша ақпаратты қайта құйып, қабылдай алады, бірақ олар екі компьютер арасында деректерді берудің сенімділігіне кепілдік бермейтінімен түсіндіріледі. Бұл кемшілікті жою үшін ерекше көліктік деңгейін құрайтын хаттамалар қолданылады.

Сонымен бірге, көлік деңгейінің сенімділігі ақпаратты беру кезінде қателіктер болмайды дегенді білдірмейді. Тасымалдау қабатының сенімділігі бұл қателіктердің анықталуына және түзетілуіне кепілдік береді (қайта жіберу арқылы) (1.7 сурет).

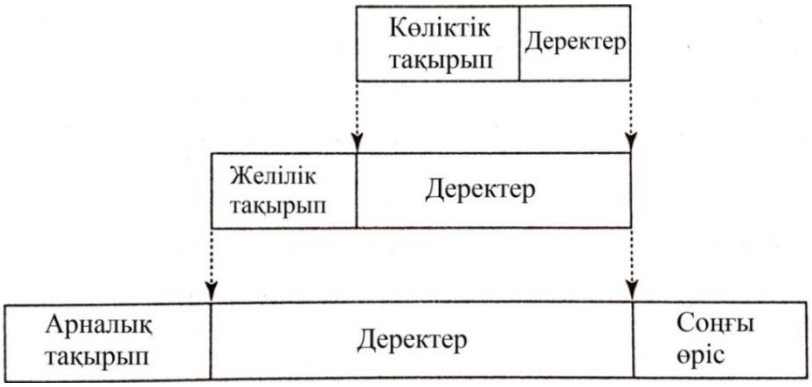

1.7 сурет – Көліктік деңгейдегі қосылыс

Желілік деңгей қашықтағы компьютерлер арасындағы байланысты қамтамасыз етеді, сондықтан көліктік деңгейінің жолында модемдер, қайталағыштар, көпірлер немесе маршрутизаторлар жоқ сияқты көрінеді. Көлік қабаты толығымен төменгі деңгейдегі қызметтердің жұмысына сүйенеді. Ол компьютерлер арасында байланыс орнатылды деп есептейді,

сондықтан еш қиындықсыз ол қашықтағы компьютерлердегі бағдарламалар арасында қосылу тапсырмасын орындайды.

Екі компьютер арасында бір уақытта бірнеше көлік қосылымы орнатылуы мүмкін – бір байланыс, мысалы, виртуалды терминал үшін, ал екіншісі – электрондық пошта үшін. Желілік деңгей тұрғысынан, көліктік дестелер компьютердің (немесе оның желілік интерфейсінің) мекенжайына ие. Көлік деңгейі тұрғысынан олар жеке бағдарламаға арналған. Бағдарламаларда бір компьютердің ішінде бірыңғай мекенжайлар болады, сондықтан көлік мекенжайы әдетте желілік және көліктік адрестерді қосу арқылы құрылады.

Деректерді беру бірлігі – тақырыптар мен мәліметтерден тұратын көлік дестесі. Тасымалдау дестесі желілік дестенің деректер бөлігінде беріледі (1.8 сурет).




1.8 сурет – Арналық фреймдерге салынатын, көліктік дестелерін желілік дестелерге орналастыру

Сеанстық деңгей өз міндетіне компьютерлер (бағдарламалар) арасында сеанстарды орнату және үзу, сондай-ақ олардың арасындағы диалогты басқару кіреді. Сеанс - бұл екі компьютер арасындағы логикалық байланыс.

Әр сеанс үш кезеңнен тұрады:

– байланыс орнату. Бұл кезеңде компьютерлер бір-бірімен байланыс жасайды: бірдей хаттамалар мен байланыс параметрлерін пайдалануға келіседі;

– ақпаратты беру. Бұл кезеңде компьютерлер деректер алмасуды жүзеге асырады;

– байланысты жою.

Сеанс деңгейі бағдарламалар арасында деректер алмасуды қамтамасыз етеді, яғни checkpoint, транзакцияларды синхрондау (commit), файлдарды дұрыс жабу және т.б. Мысалы, желілік дискіні ортақ пайдалану сияқты қосылыстың көрнекі үлгісі бола алады.

Дискіні белгілі уақытқа ортақ пайдалануға болады, бірақ ол жиі пайдаланылмайды. Мысалы, желі дискісіндегі белгілі бір файлмен жұмыс істеу қажет болған кезде, байланыс ашылған сәттен бастап ол жабылғанға дейін көліктік деңгейде байланыс орнатылады. Дегенмен, сеанстық деңгейдегі

қосылымның өзі дискіні ортақ пайдалану мерзімі ішінде ғана жүзеге асырылады.

Деректер берудің негізгі бірлігі сеанс дестесі болып табылады. Ол кейін көліктік дестесіне салынады. Сеанстық деңгейінен жоғары мұндай болмау керек.

Сеанс деңгейіндегі ақпараттарды мәліметтер ішіне орналастыруға болады. Одан да қызықты жағдай тұтас дестенің мазмұнын өзгерте отырып, деректерді шифрлайтын көрсетімдік қабатымен байланысты.

Көрсетімдік деңгей жоғары тұрған қолданбалы деңгейге деректерді дұрыс ұсыну үшін қызмет етеді. Оның көмегімен өзара әрекеттесетін екі хаттамалық стек (немесе екі өзара әрекеттесетін ашық жүйелер) бір-біріне берілетін ақпараттың форматымен немесе басқаша айтқанда синтаксисі жайында келіседі.

Мысалы, егер екі компьютер әртүрлі символдарды кодтаумен байланысса, онда көрсетімдік деңгей жіберілген ақпаратты бөтен кодтаудан өз кодтауына түрлендіруге жауап бере алады.

Сонымен қатар, көрсетімдік деңгейі деректерді қорғауға жауап береді. Бұл ретте қорғау деп деректерді шифрлеу, олардың тұтастығын қамтамасыз ету, цифрлық қолтаңбаны орналастыру және т.б. түсініледі.

Қолданбалы қабат ашық жүйенің өзара әрекеттесу моделіндегі ең жоғары деңгей болып табылады. Дәл осы деңгейде пайдаланушылардың бағдарламалары желіде жұмыс істеуі үшін қажет қызметтердің жұмысы жүзеге асырылады.

Бұл қызметтер мыналарды қамтуы мүмкін:

- электрондық поштамен алмасу. Сонымен қатар, электрондық поштамен жұмыс істейтін хаттама көптеген түрлі қосымшаларды қолдайды (1.9 сурет);
- файлдарға қашықтан кіру, файлдарды жіберу;
- қашықтағы компьютерде әртүрлі тапсырмаларды орындау және т.б.

| | |
|---|---|
| Қолданбалы | X.400, FTAM, CMIP |
| Көрсетімдік | X.226, X.216, ASN.1 |
| Сеанстық | X.225, X.215 |
| Көліктік | TP 0-4, TP дискрет. |
| Желілік | X.25, X.75, ISDN |
| Арналық | HDLC, LAPB, ISDN |
| Физикалық | V.24, V.35, X.21, ISDN |

1.9 сурет – ISO OSI моделінің кейбір хаттамалары

Қолданбалы деңгей бағдарламалар арқылы берілетін деректердің қандай форматта қабылдау/беру тәсілдерін анықтайды. Мысалы, «Виртуалды

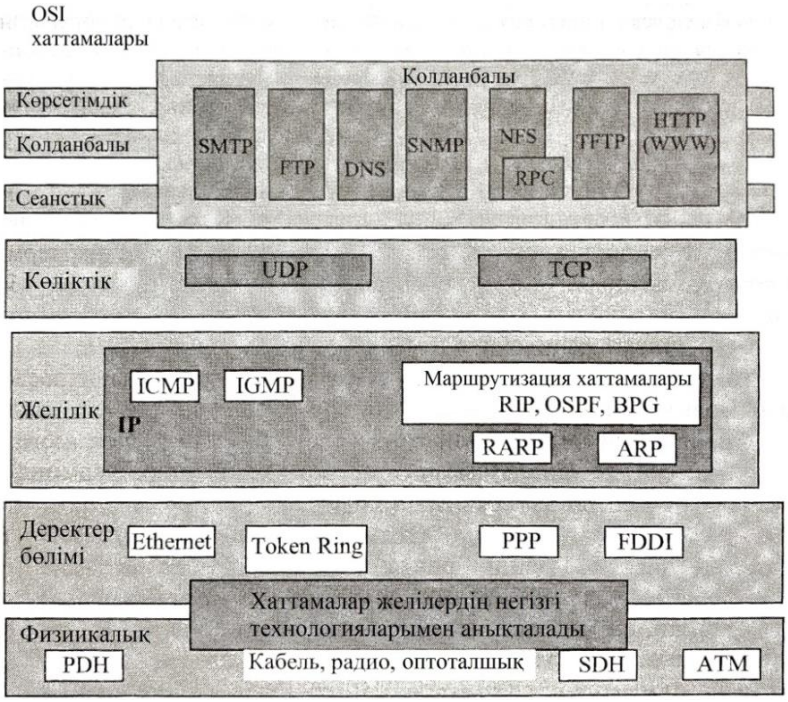
терминал» хаттамасы деректерді қалай форматтау керектігін және қосылудың екі түйіні арасындағы диалогты сипаттайды.

### 1.3 TCP/IP хаттамалар стегі

ISO OSI моделінен басқа, TCP/IP хаттамалар стегі бар. Бұл модель OSI моделіне дейін әзірленді. TCP / IP хаттамалары физикалық және арналық деңгейлердегі үрдістерді қамтамасыз етуге арналмаған. Ғаламторда физикалық және арналық деңгейлер үшін ISO OSI нормаларына сәйкес хаттамалар жиі қолданылады.

ISO OSI мен TCP / IP арасындағы байланыс қандай? Әр топтың өз деңгейлері мен осы деңгейлердегі хаттамалардың өзіндік анықтамалары болады. Осыған байланысты ISO OSI және TCP/IP хаттамалары салыстырылмайды. Алайда, іс жүзінде IP-дестелерін беру үшін, немесе, керісінше, Ғаламтор арқылы ISO OSI бойынша қызметтерді жүзеге асыру үшін ISO OSI сәйкес келетін байланыс құралдарын пайдалану қажет.

Ғаламтор хаттамаларының стегі бес деңгейден тұрады: физикалық, арналық (деректерді беру буыны), желілік, көліктік және қолданбалы. Бірінші төрт деңгей физикалық стандарттарды, желілік интерфейсті, желіаралық өзара қатынасты және OSI моделінің бірінші төрт деңгейіне сәйкес келетін көліктік функцияларды қамтамасыз етеді. OSI моделіндегі ең жоғарғы үш деңгей Ғаламтор хаттамаларының стегінде қолданбалы деңгей деп аталатын бір деңгеймен ұсынылған (1.10 сурет).




1.10 сурет – OSI-мен салыстырғанда Ғаламтор хаттамаларының стегі

Ғаламтордың негізгі хаттамаларының жиынтығы иерархиялық, диалогтық модульдерден тұрады, олардың әрқайсысы берілген

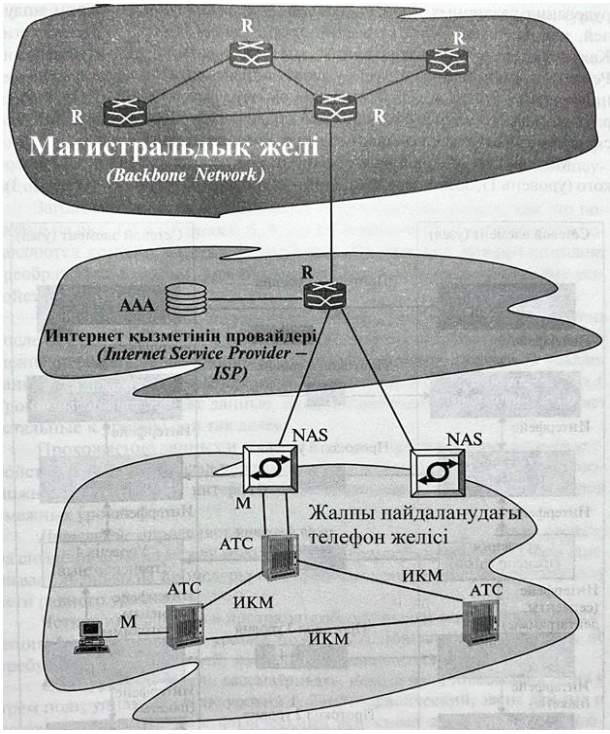
функционалдық мүмкіндіктерді қамтамасыз етеді, бірақ бұл модульдер міндетті түрде өзара тәуелді емес. TCP/IP хаттамалар жиынтығының деңгейлері жүйенің қажеттіліктеріне байланысты араласуы және келісілуі мүмкін салыстырмалы тәуелсіз хаттамалардан тұрады. Иерархиялық термині деңгейдің әрбір жоғарғы хаттамасы тиісінше төменгі деңгейдің бір немесе одан да көп хаттамаларымен қолдау көрсететінін білдіреді.

Көлік деңгейінде стек екі хаттаманы анықтайды: берілісті басқару хаттамасы (TCP) және пайдаланушы дейтаграммасының хаттамасы (UDP). Желілік деңгейде желіаралық өзара қатынасқа арналған негізгі хаттама IP болып табылады. Дегениен бұл деңгейде төменде айтылған кейбір басқа да хаттамалар пайдаланылады.

TCP/IP хаттамаларының стегі физикалық және арналық деңгейлерінде бірде-бір хаттамаға артықшылық бермейді. Ол осы деңгейде желілік технологиялармен анықталатын кабель, оптоталшықты кабель және радиоарналар бойынша барлық стандартты және жеке тарату хаттамаларын (PDH, SDH, ATM және т.б) қолдайды.

Коммутациялық құрылғылар арналық деңгейде әртүрлі технологияларды қолданады: Ethernet, Token Ring, FDDI және т.б

Ғаламтор кез келген ақпарат түрін дерек көзінен алушыға тасымалдауға арналған. Ақпаратты тасымалдауға желінің әртүрлі элементтері – шеткі құрылғылар, коммутациялық құрылғылар мен серверлер қатысады. Коммутациялық құрылғылардың көмегімен түйін топтары жергілікті желіге біріктіріледі, жергілікті желілер бір-бірімен шлюздермен (маршрутизаторлармен) қосылады (1.11 сурет).




1.11 сурет – ЖПТЖ абонентінің Ғаламторға қосылу сұлбасы

Желі тұрғысынан түйіндер ақпарат көздері мен алушылар болып табылады. Төменгі төрт деңгей берілетін ақпараттың түріне байланысты тәуелсіз. Төртінші деңгеймен байланысатын әрбір бағдарлама өзінің ерекше порт нөмірімен анықталады. Порттардың нөмірлері 0-ден 65535-ке дейін диапазонды алады. Бұл диапазонда 0 – 1023 порттар нөмірлері жалпы желілік қосымшалар (well-known ports) ретінде бөлінген, 1024 – 49151 порттарды бағдарламалық жасақтаманы әзірлеушілер пайдаланады, 49152 – 65535 порттар нөмірлері байланыс сеансындағы пайдаланушы бағдарламаларына динамикалық түрде бекітіледі.

TCP/IP көліктік деңгейде екі хаттаманы анықтайды: TCP және UDP.

UDP және TCP – үрдістің (жұмыс істейтін бағдарламаның) басқа үрдіске хабарламаны жеткізуге жауап беретін деңгейдің көліктік хаттамалары.

UDP пайдаланушы дейтаграммалардың хаттамасы – екі стандартты TCP/IP тасымалдау хаттамаларының ішіндегі ең қарапайымы. Ол әр түрлі жұмыс станцияларының қолданбалы деңгейлері арасында порт мекен-жайы бойынша беру функцияларын орындайды, бақылау сомасы бойынша қателерді бақылайды және ақпаратты жоғарғы деңгейлерге береді.

TCP хаттамасы қосымшаларға көліктік деңгейдің толық қызметін қамтамасыз етеді. TCP – логикалық байланысты орната отырып, толық дуплексті байланыс режиміне бағытталған ағынға арналған сенімді көліктік хаттамасы. Бұл үшін әрбір берілетін дестеге реттік нөмір беріледі және оның дұрыс қабылданғанын қабылдаушы тарапынан расталуы қажет. Бұл контексте «ағын» термині деректерді беру басталмас бұрын қосылымның екі ұшы арасында орнатылуы тиіс деп есептелген деректерді беруді білдіреді. TCP хаттамасының коды 6 (он алтылық кодта – 0x06) және ақпаратты кепілді тасымалдау үшін пайдаланылады.

Әр берілім соңында TCP деректер ағынын сегменттер деп аталатын кішігірім бөліктерге бөледі. Әрбір сегмент қабылданған соң ақпаратты қайта реттеуге қажетті реттік нөмірді және алынған сегменттер үшін растау нөмірін қамтиды. Сегменттер желі арқылы IP дейтаграммаларында тасымалданады. Алмасу соңында TCP әрбір дейтаграмманы келіп түскеніне қарай жинайды және реттік нөмірлерге негізделе отырып, қайта реттейді [5].

TCP/IP желілік деңгейде (немесе, дәлірек, желіаралық деңгейде) желіаралық өзара байланыс (IP) хаттамасын қолдайды. IP, өз кезегінде, төрт қолдау хаттамасынан тұрады: мекенжайды анықтайтын ARP хаттамасы, орналасқан жері бойынша желілік мекенжайды анықтау RARP хаттамасы, Internet басқару хабарламаларының ICMP хаттамасы және топтарды басқару желіаралық IGMP хаттамасы. Осы деңгейде маршрутизация хаттамалары қолданылады: маршруттық ақпаратпен алмасу RIP хаттамасы, динамикалық бағыттау OSPF хаттамасы, шекаралық маршруттау BGP хаттамасы.

Желіаралық өзара әрекеттесу IP хаттамасы – TCP/IP хаттамалары пайдаланатын тасымалдау механизмі. Бұл байланыссыз, бірақ «максималды күшпен» (best-effort) деректер кестесін жеткізудің сенімсіз қызметі.

«Максималды күшпен» термині ақпаратты оның тағайындалған пунктіне беру үшін барлық мүмкін болатын күшпен жасалатынын білдіреді, бірақ IP қателерді тексеруді немесе оларды қадағалауды қамтамасыз етпейді. IP қажетті сервис деңгейіне кепілсіз деңгейлердің сенімсіздігін болжайды.

IP деректерді дейтаграммалар деп аталатын дестелер тасымалдайды, олардың әрқайсысы бөлек тасымалданады. Деректер әртүрлі маршруттар бойынша жүре алады және бастапқы күйде келмеуі мүмкін. IP маршруттардың көшірмесін сақтамайды және олар тағайындалған пунктке жеткенде дейтаграммаларды қайта реттей алмайды.

Шектеулі IP функционалдығын кемшілік деп санауға болмайды. IP пайдаланушының ерекшелігі жоқ «таза» беру функцияларын қамтамасыз етеді және басқа деңгейлерде осы қосымшаны жүзеге асыру үшін қажетті құралдар қосылады және осылайша максималды тиімділікке қол жеткізіледі.

Мекенжайды анықтау ARP хаттамасы IP-мекенжайын физикалық мекенжаймен байланыстыру үшін қолданылады. LAN желісі сияқты жергілікті физикалық желіде байланыс желісіндегі әрбір құрылғы, әдетте желілік интерфейс картасында тіркелген, физикалық мекенжай немесе станция мекенжайы бойынша анықталады. ARP Ғаламтор желісінде түйіннің мекенжайы белгілі болған кезде оның физикалық мекенжайын табу үшін қолданылады.

Мекенжайларды анықтаудың кері RARP хаттамасы (желінің мекенжайын орналасқан жері бойынша анықтауға арналған хаттама), хост өзінің физикалық мекенжайын білсе ғана Ғаламтордан өз мекенжайын табуға мүмкіндік береді. Бұл компьютер бірінші рет желіге қосылған кезде немесе компьютер дискісіз жүктелгенде қолданылады.

Ғаламтордың басқару хабарламаларының ICMP хаттамасы – дейтаграммалық мәселелер туралы хабарламаны кері таратқышқа жіберу үшін хостармен және шлюздермен пайдаланылатын механизм.

Топтарды басқаратын желіаралық IGMP хаттамасы – хабарламаны алушылар тобына бір уақытта жіберуді қамтамасыз етеді.

Шекаралық маршруттау BGP хаттамасы – векторды қолдануға негізделген автономды жүйелер арасындағы маршруттау хаттамасы.

Маршруттық ақпаратпен алмасу RIP хаттамасы – қашықтық векторының алгоритмін пайдалануға негізделген маршруттау хаттамасы.

Динамикалық бағыттау OSPF хаттамасы – байланыс желілерінің жағдайын талдауға негізделген желі ішіндегі маршрутизация хаттамасы.

Ғаламтор хаттамаларының стегінде қолданбалы деңгей OSI моделінде сеанстық, көрсетімдік және қолданбалы деңгейді біріктіруге баламалы. 1.10 суретте келесі қолданбалы деңгейдегі хаттамалар көрсетілген:

SMTP – қарапайым пошта хаттамасы. Ол Ғаламтор арқылы электрондық пошта хабарларын таратуды қолдайды. Хаттама қарапайым деп аталады, себебі тез арада жеткізуге дайын ақпаратты пайдаланушыларға беруді қамтамасыз етеді. Беру 7 биттік сөз режимінде жүзеге асырылады. Ол

көптеген бағдарламалардан 8-биттік сөздермен және 7-биттік сөздермен форматқа көшу бағдарламаларының болуын талап етеді.

Жүйе қолдайды:

- бір немесе одан көп алушыларға бір хабарлама жіберу;
- мәтін, дауыстық хабарлар, бейне немесе графикалық материалдар кіретін хабарламаларды жіберу.

Файлдарды жіберу FTP хаттамасы файлдарды бір компьютерден екінші компьютерге тасымалдау үшін қолданылады. Қашықтағы компьютерде каталогтарды көруге, файлдарды көшіруге, жоюға және жіберуге мүмкіндік береді. FTP басқа хаттамалардан ерекшеленеді, ол хосттар арасында екі байланыс орнатады. Біреуі ақпаратты беру үшін, ал екіншісі берілісті басқару үшін қолданылады.

DNS – домендік атаулар қызметі. Ол Ғаламтор желісінің барлық пайдаланушылары мен тораптарына бірегей атаулар беруді жүзеге асырады және олардың желілік мекенжайларымен логикалық байланыс орнатады. Домен аты бірнеше деңгейдегі иерархиялық құрылым болып табылады. Жоғарғы деңгейдегі домендердің типтік атаулары келесідей бекітілген:

- .com – коммерциялық ұйымдар;
- .gov – үкіметтік мекемелер;
- .org – коммерциялық емес ұйымдар;
- .net – желіні қолдау орталықтары;
- .int – халықаралық ұйымдар;
- .mil – әскери құрылымдар.

SNMP – желіні басқарудың қарапайым хаттамасы. Ол Ғаламторды бақылау және қолдау бойынша негізгі іс-шаралар жиынтығын ұсынады.

Хаттама әртүрлі өндірушілер жасаған және әртүрлі физикалық желілерде орнатылған құрылғыларды басқара алатын етіп жасалған. Басқаша айтқанда, SNMP басқару міндеттерін басқарылатын құрылғылардың физикалық сипаттамаларын және негізгі желілік технологияны ескеруден босатады.

NFS желілік файлдық жүйе. Бұл көптеген хаттамалардың бірі (мысалы, суретте тағы бір RPC хаттамасы) ол басқа компьютерде басқару және периферия процедуралары бар файлдарды пайдалануға мүмкіндік береді.

TFTP файлдардың тарату тривиальды (қарапайым) хаттамасы. Қарапайым жағдайларда жұмыс станцияларын бастапқы жүктеу немесе сыртқы жады жоқ маршрутизаторларды жүктеу кезінде пайдаланылады.

Гипермәтінді беру HTTP хаттамасы – гипермәтінді құжаттарды сипаттау тілінде ұсынылған құжаттармен алмасу кезінде Ғаламторда қолданылатын көліктік хаттама.

HTML – WWW желісінде қолданылатын негізгі тілдердің бірі.

WWW – ғаламдық гипермәтіндік ақпараттық жүйе. Ол әлемнің көптеген елдерінде сақталған және Ғаламтордағы байланыс арналары арқылы байланысқан тораптар арқылы қол жетімді көптеген құжаттарды біріктіреді.

## 1.4 Дестелік трафикті талдау тереңдігінің деңгейлері

Дестелік ағынды зерттеу технологияларына қайта оралғанда, бұл технологиялар желілік модельдер деңгейлерін талдау тереңдігімен ерекшеленетінін атап өткен жөн.

SPI ағынды «терең емес» талдау технологиясы OSI моделінің арналы және желілік деңгейінде ғана жұмыс істейді және дестенің пайдалы жүктемесінің мазмұнын талдамайды. SPI тек маршруттауды оңтайландыру, желіні теріс пайдалану әрекеттерін анықтау және статистикалық талдау үшін дестелердің тақырыптарын тексеруді жүзеге асырады. Егер бума тақырыбының ақпараты «қара тізімде» болса, бума өшіріледі. SPI құралдары желіні бір-бірінен бөле алатын, таңдалған деректер беру хаттамасына байланысты ағынды рұқсат ететін немесе тыйым салатын желіаралық экрандар үшін тиімді. SPI оқиғалар журналын жүргізеді, NAT-адрестерді таратуды қамтамасыз етеді, vpn-да соңғы нүкте функциясын орындайды, ал қажет болған жағдайда барлық ағынды бұғаттайды. SPI технологиясы негізінде операциялық жүйелердің брандмауэрлерінің көпшілігі жұмыс істейді. SPI OSI моделінің төменгі деңгейлерінде ғана жұмыс істейтіндіктен, ол басқа ресурстарға қарағанда аз талап етеді, соның есебінен жоғары жылдамдықпен ағынның үлкен көлемін өңдей алады [6].

IP-байланысты ұйымдастырудың әртүрлілігі мынада, қолданбалы деңгейінде деректерге қол жеткізу үшін дестелердің тақырыптарын 4-ші деңгейге дейін тарату жеткіліксіз. Мысалы, HTTP-ағыны Ethernet/IP/TCP/HTTP стегін пайдалана отырып берілуі мүмкін және 3G желісінде сол ағын Ethernet/IP/UDP/GTP/IP/TCP/HTTP стегін пайдаланады, мұнда GTP – GPRS туннелдеу хаттамасы.

MPI ағынды «орташа» талдау технологиясы қосымшадағы, бірақ прокси-шлюзбен орнатылатын сессиялар мен байланыс сеанстарын талдауға негізделеді. Желінің барлық трафигінің желілік саясатын орындауды қамтамасыз ететін прокси-сервер арқылы өтеді. MPI брандмауэрде қолданылады, бірақ ол нашар масштабталған болып саналады. Өйткені, әр бағдарламаға қолданбалы деңгейдің прокси-шлюзі қажет. Ал мұндай әрбір дестені тексеру үрдісі ағынның жылдамдығын азайтады.

LPI трафигін "жеңіл" талдау технологиясы көлік қабаты үстіндегі жүктеме өрісін MPI сияқты емес, SPI сияқты алғашқы байттарды бағалау үшін OSI моделінің көрсетімдік деңгейін талдап, қолданбалы қабаттың аспектілерін анықтауға мүмкіндік береді. DPI-ға бағытталған маңызды қадам болып табылады.

Бірақ қолданбалы деңгейге дейін «жеткен» болсақ та, ағынды TCP/UDP порт нөмірімен бірегей анықтау әрқашан мүмкін бола бермейді. Барлық бағдарламаларда IANA-да тіркелген порттар бола бермейді (мысалы, Skype). Осы мақсатта DPI платформаларында қолданылатын негізгі әдістердің бірі – хаттама мен қосымшаның сигнатурасын тексеру. Сигнатура –байланысқан бағдарламаны/хаттаманы сәйкестендіру үшін таңдалған мәліметтер шаблоны.

Әрбір DPI платформасында сигнатура кітапханасын сақтайды. Ол жаңа нұсқалар немесе бағдарламалар пайда болған кезде жаңартылып отырады [1].

Сигнатурлық әдістен басқа желілік транзакцияларды талдау да пайдаланылады, ол да әрбір қосымшалар мен хаттамаларға тән сипаттамаларға ие болуы мүмкін (пайдалы жүктеме мөлшері, сұранымға жауап ретінде дестелердің саны мен өлшемдері, дестенің ішіндегі тіркелген жолдар немесе байттар позициясы және т.б.). DPI-да деректер ағынының статистикалық және мінездік сипатына негізделген әдістер және басқа да эвристикалық әдістер бар.

DPI жүйелерін пайдаланудың ең танымал мысалдарының бірі – URL мекенжайы бойынша тыйым салынған ресурсқа ағынды нүктелік бұғаттау есебінен және хостинг компаниясының серверін толық бұғаттаусыз сайт мазмұнына қол жеткізуді бақылау. URL "қара"/"ақ" тізімдермен салыстыру үшін, жасырын контентті орталықтандырылған бақылау үшін пайдаланылуы мүмкін [6].

DPI-дің тағы бір қолданылуы тариф саясатымен байланысты. Бұл саясат уақыттың, тасымалдың көлеміне және оның мінез-құлқына байланысты ережелердің динамикалық өзгеруіне негізделген, оларды тәулік уақытына немесе желінің толып кетуіне байланысты тарифтерді енгізу үшін қолдануға болады.

Тарифтеу келесі критерийлер бойынша жүзеге асырылуы мүмкін: time-нүктелер – белгілі бір уақытта арзан; geo-нүктелер – белгілі бір жерде арзан/тегін; empty-нүктелер – желі тоқтап тұрған кезде арзан [7]. DPI құралдары өте икемді тарифтік торды әзірлеуге және деректерді беру жылдамдығын алдын ала шектеуді жүзеге асыруға мүмкіндік береді.

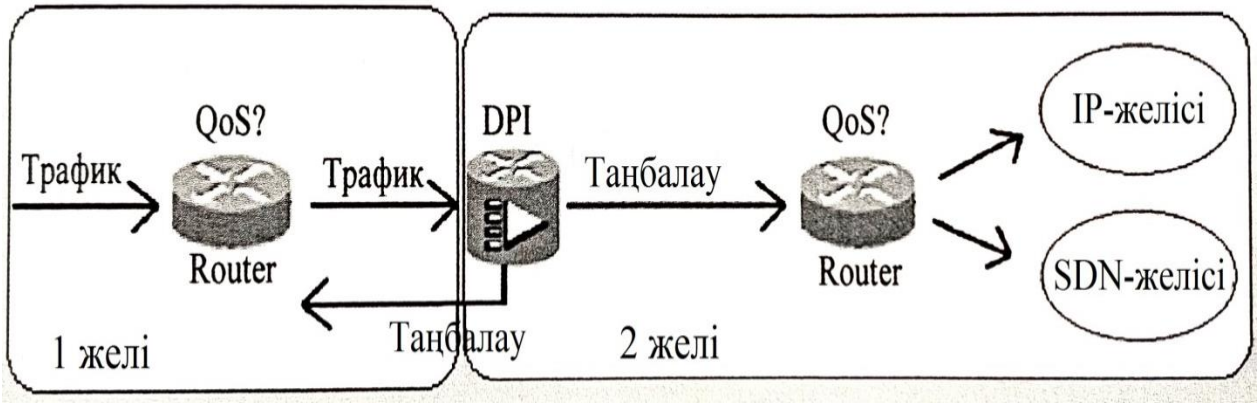
Жүйені қолданудың жоғарыда аталған барлық мысалдары айтарлықтай есептеу ресурстарын қажет ететіні түсінікті және DPI платформасына қойылатын басты талаптардың бірі – дестелерді деректер арнасының жылдамдығында талдау болып табылады. DPI өнімдеріне қойылатын тағы бір міндетті талап – икемділік, яғни ағынды өңдеудің жаңа мүмкіншіліктері мен сценарийлерін қосу мүмкіндігі.

DPI-өнімдердің бірінші буыны аз талаптарды шешу үшін бейімделген және пайда болған проблемаларды шешу немесе жаңа қызметтерді енгізу үшін айтарлықтай икемді болып табылған жоқ. Бұл желіге жаңа жабдықты қосу қажеттігіне алып келді. Жаңа аппараттық құралдарды қосу өте қымбат болуы мүмкін. Сонымен қатар, арнайы DPI-өнімдерді қолдану жаңа талаптарға тез және уақтылы жауап беруге мүмкіндік бермейді [1].

DPI өнімдерінің екінші буыны, бағдарламаланатын DPI құрылғылары жаңа жабдықты қосуды және желіні қайта құру қажеттіліктерін тудырмайды. DPI жаңа буыны көп ядролы желілік процессорларға негізделген. Бұл көптеген DPI қосымшаларын арна жылдамдығымен басқаруға және тек бағдарламалық жасақтаманың жаңартуларын қолдана отырып жаңа мүмкіндіктер қосуға мүмкіндік береді.

DPI-дың негізгі міндеті – OSI моделінің 7-ші деңгейіне дейін дестелік ағындарды терең талдау. Траффикті өткізіп жіберу, шектеу, бұғаттау функцияларына келетін болсақ, оларды қарапайым құрылғыларда және желінің әртүрлі нүктелерінде орындауға мүмкіндік пайда болды. Бұл DPI-да талданған трафикті белгілеу арқылы дестедегі талдау мәліметтерін желідегі басқа құрылғылармен бөлісу идеясына әкелді.

Ағын туралы метадеректердің мұндай таңбалануы мен берілуін белгілі бір ағынға немесе белгілі бір класстағы ағындар тобына жататын дестелерді, сол немесе басқа тасымалдаушының желілік құрылғыларымен кейіннен өңдеу үшін де қолдануға болады. 1.12 суретте көрсетілгендей, ағын туралы метадеректер ақпарат көзіне берілген кезде, оны DPI жүйесіне келместен бұрын ағынды өңдеуді ұйымдастыруға болады. Осылайша, сараланған қызметті DPI жүйесінің өзінде де, ағын бағытында орналасқан желілік құрылғылар арқылы да жасауға болады.



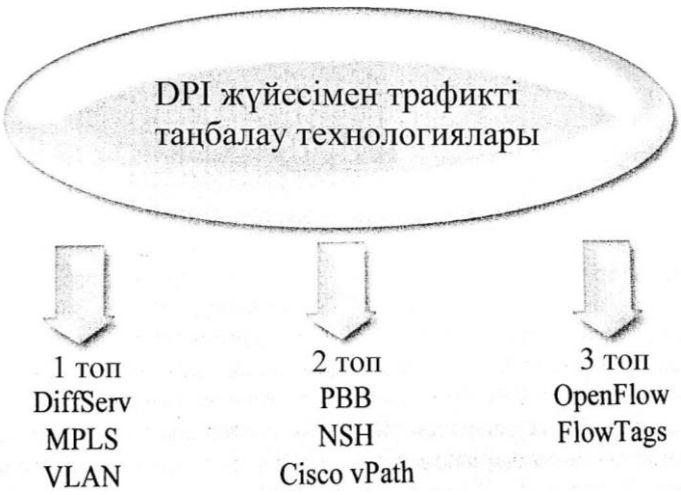

1.12 сурет – Таңбалы желі үлгісі

DPI-дегі ағынды белгілеу үшін бүгінгі таңда келесі технологияларды қолдануға болады: DiffServ, MPLS, VLAN, PBB, NSH, Cisco vPath, OpenFlow, FlowTags (1.13 сурет).

Таңбалауды бірінші топтың классикалық технологиялары жүргізе алады, алайда олар ағын туралы ақпараттың жеткіліксіз көлемін жібереді [6].

Екінші топ технологияларының артықшылығы – ағын туралы тегті немесе метадеректерді жазуға арналған өрістердің үлкен мөлшері және ағын туралы ақпараттың өзі, мысалы, алушы мен жіберушінің мекен-жайы, технологияда болуы.

Барлық үш технология тобы қазіргі заманғы SDN бағдарламалық-анықталатын желілерімен жұмыс жасайды. Өз кезегінде, алғашқы екі топқа қарағанда, үшінші топ тек SDN-ге бағытталған және кәдімгі IP-желілерде қолданыла алмайды.




1.13 сурет – Таңбалау технологияларының топтары

DPI-да ағынды таңбалау құралдарымен желіде қандай да бір контент көздерін қадағалауға және бұғаттауға болады. Сандық мазмұнды таңбалау заңсыз сандық өнімнің таралуын орнатуға мүмкіндік береді. Сандық ақпаратты таңбалау технологияларын авторлық құқық иелері мен таратушы компаниялар кеңінен қолданады.

BT мінездік таргетинг технологиясы абонент туралы мәліметтерді оның профиліне жинайды. Содан кейін абонент профилінің жиынтығын оператордың/провайдердің өзі мақсатталған жарнама үшін пайдаланылады немесе мамандандырылған жарнамалық агенттіктерге сатады. Провайдерде орнатылған DPI абоненттік ағынды талдайды және мінездік таргетингі үшін қажетті деректерді алады [6].

Әдетте, деректер Ғаламтор-сайттардың өзара әрекеттесуін талдау арқылы, басқа сайттарға сілтемелердегі немесе javascript-нұсқауларындағы айнымалыларды қолдану арқылы жиналады. Сондай-ақ веб-браузердің (cookies) деректері сұралады және кірген беттер туралы ақпарат жиналады.

DPI HTTP-сұраныстарды қайта адрестеуді және абонент трафигін қайта бағыттауды жүзеге асыра алады. Сондай-ақ DPI трафикті таңбалауды қолдана отырып қосымша қызметтер сервері (контентті оңтайландыру) арқылы ілмектер ұйымдастырады және HTTP-трафигін жарнамалық порталдарға бағыттай алады. Сонымен қатар, тыйым салынған URL мекенжайына қол жеткізу әрекеті кезінде немесе ақылы қол жеткізу кезінде құралдар болмаған (таусылған) кезде, сондай-ақ заңсыз контент алуға әрекет жасаған кезде беттердің мазмұнын ауыстыру және/немесе ұқсас заңды контентті таратудың серіктестік сайттарына бағыттау мақсатында қайта бағыттау жүзеге асырылуы мүмкін.

### 1.5 DPI технологиясының құрылымы

DPI қолдану айтарлықтай аппараттық ресурстарды қажет етеді, сонымен қатар қосымша кідірістерге әкеледі. Алайда, QoS үшін дестелерді терең қосымша кідірістер, джиттер және дестелердің жоғалуы салдарынан қызмет

көрсететін трафиктің QoS азаюына әкелуі мүмкін. Кідіріс шамасы жаңа ағынды талдау алгоритмдерінің күрделілігіне байланысты, яғни ағын алдымен талданады, содан кейін жүйенің белгілі бір саясатына сәйкес беріледі. Егер DPI ағынды сақтап, оның көшірмесін талдай отырып және оның талдаудың аяқталуын күтпей-ақ өткізсе, мұндай кідірістер болдырмауға болады.

Желідегі DPI-дың орнын екі түрге бөлуге болады: бөлінген және жергілікті байланыс.

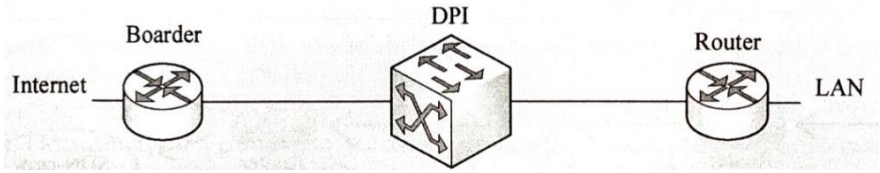
Бөлінген желі ағыны туралы деректерді жинау үшін сынамалардан (probes) және сол сынамалардан деректерді алатын анализаторлар жиынтығынан (collectors) тұрады.

Жергілікті жүйелер нақты деректер беру арнасына қосылады. Жергілікті жүйелер соңғы пайдаланушының жағында да, шлюзде де болуы мүмкін. Соңғы пайдаланушы жағындағы жергілікті жүйелер HIPS сияқты пайдаланушының желілік картасы деңгейінде қосылады. Шлюз жағындағы жергілікті жүйелер кейбір жергілікті ішкі желі LAN үшін ғаламдық желіге WAN қол жеткізуге болатын жалғыз нүктеде қосылады. Бірнеше шлюздердің біреуіне DPI-ді орнатқан кезде, мысалы, қосылыстың кіріс ағыны бір шлюзден өтетін болса, ал шығыс ағыны екінші трафик арқылы өтетін болса, «бір бағыттағы ағындар» жағдайын байқауға болады. Мұндай жағдай «желілік асимметрия» (Network Asymmetry) деп аталады.

DPI, әдетте, оператордың/провайдерлердің шекарасында орнатылады. Сондықтан бұл жүйе арқылы өтетін немесе кіретін барлық ағынды осы жүйеден бақылауға және басқаруға мүмкіндік береді.

1.14 суретте оператор желісінің шетіндегі DPI платформасының негізгі қосындыларының бірі көрсетілген. Қосылымды үш түрде жүзеге асыруға болады: көшіру, «ажырау» және Bypass.

Белгілі бір мәселелерді шешу үшін ағынның терең талдау жүйесі желі шекарасында емес, пайдаланушыларға жақын желілерде, BRAS/CMTS/GGSN деңгейіне және т.б. дейін орнатылады. Бұл операторларға сыртқы арналарды басқарумен қатар, ішкі желілерді бақылау мәселесін шешу үшін пайдалы болуы мүмкін.

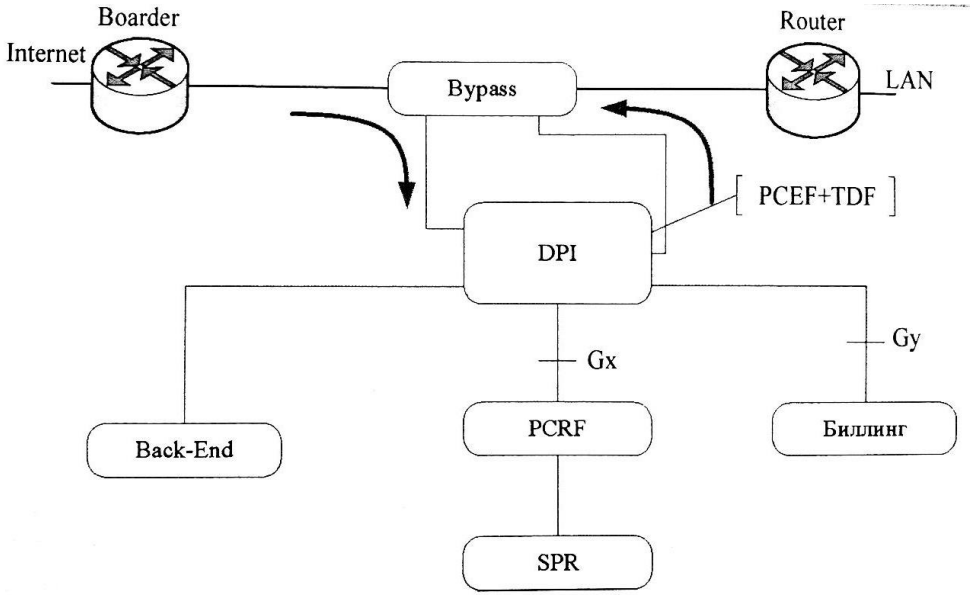



1.14 сурет – DPI жүйесін типті қосу

Жұмыс режимінде ағын DPI-да "оралады" (1.15 сурет), бірақ желіде артық жүктеме немесе апат жағдайы пайда болса DPI қорғаныс режиміне ауысады. Dpi жүйесімен ағынды өңдегеннен кейін ол әрі қарай оператор желісі бойынша беріледі. DPI жүйесі Bypass серверінен, аппараттық сүзгіден,

Front-End, IMS архитектурасының PCRF және Back-End құралады [6]. 1.15 суретте аппараттық сүзгі және FrontEnd біріктірілген. Олар DPI ретінде көрсетілген.

1.15 суретте көрсетілгендей әрбір DPI сервері, өз міндеттерін орындайды және басқаларымен белсенді әрекеттеседі.




1.15 сурет – Bypass серверінің жұмыс істеу тәртібінің үлгісі

Bypass DPI жүйесі істен шыққан жағдайда қажет. Ол арқылы жүйенің аппараттық фильтріне түсетін трафикті, желіге талдау жүргізбей, кідіріссіз жібереді. Аралас режим талдауға жіберу үшін аппараттық сүзгіге трафикті көшіруді пайдалануға мүмкіндік береді.

Front-End - бұл жүйенің негізгі элементі, өйткені ол трафик ағынын тікелей талдайды. Сонымен қатар, трафикті қабылдау және беру үшін желілік карталар қолданылады. Трафик ағынын біліп, Front-End PCRF (Policy and Charging Rules Function) серверінен осы трафикпен не істеу қажеті туралы шешімді сұрайды. Содан кейін, осы шешім негізінде ол Back-End серверінен егжей-тегжейлі сүзгілеу нұсқауларын алады. Содан кейін ол ағынды және орындау нұсқауларын аппараттық сүзгіге береді. Егер ағынды бөлу сатысында мұндай ағынның бар екендігі белгілі болса, ол бірден аппараттық сүзгіге жіберіледі. Аппараттық сүзгі трафикті өңдеудің негізгі функцияларын орындайды: саясатты қолдану (блоктау немесе өткізіп жіберу), статистика жинау және өткізу қабілеттілігін басқару.

Back-End – жиналған статистика, қолтаңбалар (талдау критерийлері), көшіру және қайта бағыттау ережелері, AAA негізінде Radius немесе Diameter хаттамалары бойынша саясаттың орындалуы туралы ақпарат қоймасы [8].

PCRF – нақты уақыттағы саясатты орындау және басқару үшін шешім қабылдау сервері. Әр пайдаланушы үшін саясат туралы ақпаратты қамтиды.

Front-End абоненттің немесе бағдарламаның ID жібереді, ал PCRF қажетті саясаттың нөмірін қайтарады.

DPI жүйесімен дестелерді өңдеу операциясының кезектілігін келесі кезеңдерге бөлуге болады. Келіп түсетін десте деректердің ағымына тиесілі екендігіне талданады. Егер мұндай ағын болса, дестеге түсіру функциясы беріледі. Егер десте жаңа ағынға тиесілі болса, оны OSI желілік моделінің жетінші деңгейінде берілетін деректерді ескере отырып жіктеу қажет. Бұл міндет ағындарды өңдеу үрдісінде ең көп ресурсты қажетсінеді. Бастапқыда DPI жүйесі дестенің L2-L4 тақырыптарын және туннельдер тақырыптарын, болған жағдайда, талдайды. Бұдан әрі жүйе қолданбалы деңгейдегі мазмұнды қазіргі заманғы жүйелерде көлемі мың данадан асатын қосымшалар сигнатурасының базасымен салыстыруды орындайды. Сигнатура қарапайым жағдайда белгілі бір белгіленген символдар тізбегі болып табылады. Ол дестенің ішінде белгілі бір ығысумен орналасуға тиіс. Күрделі жағдайларда бекітілген ығысулар берілмеген болуы мүмкін. Сондықтан оларды көрсетілген тізбектілік дестенің барлық ұзындығы бойынша іздеуге тура келеді. Ең күрделі сигнатурасы быр ағындар статистикалық сипаттамасын қамтиды немесе бір қосымшаның байланысқан ағындарының параметрлерін талдау жолымен анықталады (мысалы, SIP және RTP).

Ағын жіктелгеннен кейін, сол ағын дестелерін сигнатура базасымен сәйкестендіру көп жағдайда қажет емес. Осыдан кейін ағын түсіру функциясына беріледі. Түсіру режимінде қолданбалы деңгейдегі хабарламаларды талдау жүргізілмегеніне қарамастан, ресурсты қажетсінетін өңдеу әлі де орын алады. Себебі, трафикті есептеу оның қандай да бір қосымшаға тиесілілігін ескере отырып жүргізіледі. Яғни, егер абонентте бес жіктелген ағын болса, олардың екеуі HTTP және үшеуі Bittorrent болса, әртүрлі бағдарламалар мен хаттамалар үшін берілген және қабылданған дестелерді (және байттарды) есептеу бөлек жүргізіледі. Есептеу нәтижесінде алынған ақпарат Diameter Gy хаттамасы бойынша тарифке жүгініп, PCRF жүйесін Diameter Gx хаттамасы бойынша абоненттің ағымдағы тұтыну деңгейі туралы ақпараттандыру үшін пайдаланылады. Сондай-ақ, ақпарат CDR-файлдар түрінде offline-биллинг жүйесіне берілуі және есептерді құру жүйесінде көрсетілуі мүмкін [9].

PCRF жүйесі болмаған кезде абонент трафигін есептеуіштер жергілікті DPI жүйесінде есепке алынады. Егер абонент белгілі бір уақыт ішінде шектеулі тарифтік жоспарды қолданса, онда есептеуіштердің деректері трафикті тұтынуды реттеу үшін пайдаланылады.

Соңында, деректер ағындары жылдамдықты бақылаудың ішкі жүйесіне жіберіледі. Абоненттің трафигіне (тарифтік жоспармен анықталған) қолданылатын саясатқа байланысты, жүйе белгілі бір уақытта, мысалы, үлкен жүктемеде, ағындардың өткізу қабілетін шектеуі мүмкін. Бұл операция ресурстарды көп қажет етеді. Өйткені жүйеде дестенің ұзақтығына қатысты жоғарыда көрсетілген шектеулерді ескере отырып, абоненттік деректердің әр

классификацияланған ағынының әр жеке десте үшін саясат анықтамасын қажет етеді.

Осылайша, тіпті трафикті талдаудың типтік жүйесі функцияларының осындай жоғары деңгейлі сипаттамасы әрбір жеке IP-дестені өңдеу үрдісінде орындалатын есептеу операцияларының саны туралы қорытынды жасауға мүмкіндік береді. DPI жүйесінің қажетті жиынтық өнімділігін бір серверге секундына бірнеше миллионға жеткен дестелердің түсу жылдамдығын назарға ала отырып, бағалауға болады.

Ақпараттың мұндай көлемін дәстүрлі тәсілдермен өңдеу, әрине, тиімділігі жоғары жүйелерді жобалауға және құруға мүмкіндік бермейді. Өйткені олардың өткізу қабілеттілігі қазіргі заманғы телекоммуникация операторларының деректерді беру арналарының өткізу қабілеттілігіне қарағанда екі есе төмен болар еді. Сондықтан осындай жүйелерді әзірлеушілер трафикпен жұмыс істеудің жаңа әдістемелерін қолдануға мәжбүр, олардың ішінен ең тиімдісі ретінде бағдарламалық және аппараттық босатуды келтіруге болады.

Бағдарламалық босату қазіргі трафикті өңдеудің көптеген жүйелерінде қолданылады. Оның негізгі мағынасы – жіктелген ағындарды жылдам өңдеудің арнайы функцияларына беру болып табылады. Бұл сервердің есептеу ресурстары төмен болған кезде дестелермен көптеген қарапайым әрекеттерді орындауға мүмкіндік береді. Осылайша, трафикті талдау жүйесінің процессорлары жүйенің интерфейстері арасында дестелерді жіберу, сондай-ақ саясатты қолдану (жіберу, тастау) өнімділігінің айтарлықтай бөлігін үнемдейді.

Трафикті өңдеу жүйесінің қуатын едәуір арттыруға болады. Бұл жүйенің аппараттық жүктемесін қолдану арқылы жүзеге асырылады. Бағдарламаланатын логикалық интегралды схемаларды қолдануды қажет ететіндіктен, бұл әдісті DPI платформаларын өндірушілердің бір бөлігі ғана қолданады. Аппараттық босату бағдарламалық босатудың барлық функцияларын БЛИС-қа көшіруге мүмкіндік береді, яғни бұл процессорлардағы жүктемені төмендету үшін дестелерді қайта жіберуге және санауға мүмкіндік береді. Сонымен қатар, платформаның аппараттық бөлігіне ағын жылдамдығын шектеу функциялары жүктеледі. Сондай-ақ, онда жаңа деректер ағындары БЛИС-та өңделетін трафикті алдын ала өңдеу де жүреді. Нәтижесінде сервердің есептеу ресурстарының көп бөлігі ағындарды жіктеуге және абоненттерге қызмет көрсету саясатын қолдануға бағытталады. Бұл трафикті өңдеу жүйесінің өнімділігін абоненттік қатынаудың тұрақты эволюцияланатын технологияларымен бір деңгейде болуына мүмкіндік береді.

Трафикпен алмасу қызметіне деген сұраныстың өсуі байланыс операторларын деректерді беру арналарының қуатын арттыруға, демек, желі ядросы құрылғыларының қуатын арттыруға ынталандырады. Дербес құрылғылар экрандарының диагональдарын ұлғайту, Smart TV, M2M-технологиялар және басқа да дамып келе жатқан IT-индустриясының

бағыттары қарапайым абонентке қажетті өткізу қабілетінің талаптарын үздіксіз арттырады. Ағындарды жоғары жылдамдықпен өңдеудің заманауи технологиялар, соның ішінде жоғарыда келтірілгендер, абоненттердің біріктірілген ағымының жылдамдығына сәйкес келетін трафикті өңдеу жүйесін құруға мүмкіндік береді.

Бұл дипломдық жобаның мақсаты телекоммуникациялық желілердегі жоғарыжылдамдықты дестелік ағынды талдауға арналған бағдарламаны жобалау болып табылады. Бұл жұмыстың бағдарламасы жоғары деңгейлі Python тілінде жазылатын болады. Бағдарламаның жұмыс істеу алгоритмі және және мүмкіндіктері келесі бөлімдерде қарастырылады. Python бағдарламалау тілін қолдану құрылғылардың көпшілігінде есептеулерді жүргізуге мүмкіндік береді.

Қойылған мақсатқа жету үшін желілік трафикті терең талдау жүйесінің белгіленуі, жұмыс істеу принципі және осы салада қолданылатын шешімдер келесі бөлімде қарастырылады. Сондай-ақ, бұл жүйелердің құрылымын және ерекшеліктерін талдап, желінің нақты жұмыс жағдайына жақын тестілеуін жүргізу керек.

Бұл бөлімде желілік трафикті жинау мен терең талдаудың қазіргі заманғы технологиялары қарастырылды. TCP/IP және ISO OSI хаттамалық жүйелері туралы қысқаша айтылды. Талдау тереңдігімен ерекшеленетін IP-желісінің трафигін тексерудің бірнеше технологиялары талданды. Провайдер желісінде DPI технологиясын қолдану мысалдары қарастырылды. Олар: тарифтеу, нүктелік бұғаттау, мақсатталған жарнама. Желідегі DPI жүйесінің орналасуы сипатталды.

## 2 Python арқылы желілік дестелерді алу

### 2.1 Желілік белсенділікті бақылау

Корпоративтік желілер бүгінде күрделі, оларды зерттеу қиын, олар арнайы құралдарды талап етеді және тосын оқиғаларға дұрыс әрекет ету үшін ерекше және сараптамалық дағдыларды талап етеді. Инфрақұрылымның маңызды оқиғалары немесе басқа реттелетін өнеркәсіптік жағдайлар туралы айтсақ, құралдар жиынтығының мамандануы шынымен де қорқынышты болуы мүмкін.

Оқиға орнына әрекет ету топтары мен сот-медициналық сарапшылардың алғашқы мәселелерінің бірі: «Сіздің желіңіз қандай және ол қалай конфигурацияланған?». Бұл IT-тобына оңай жауап бере алатын қарапайым сұрақ сияқты көрінуі мүмкін. Алайда, Heartbleed, Operation Shady Rat сияқты оқиғаларға және ірі бөлшек сауда орындарындағы ақауларға жауап беру кезінде техникалық ақпарат және желілік карта туралы мәліметтер маңызды болуы мүмкін.

Нақты сұрақтарға мыналар кіруі мүмкін:

– сіз қандай Интернет-хаттама мекенжайларын және ішкі желілерді пайдаланасыз;

– қандай серверлер мен соңғы нүктелер жұмыс істейді;

– серверлер сыртқы сайтта немесе бұлтта орналасқан жергілікті болып табылады ма;

– қандай операциялық жүйелер қолданылады? Қандай нұсқалар және олар қаншалықты маңызды;

– әр серверде және хостта қандай қызметтер (ашық порттар) бар;

– қандай қосымшалар мен деректер қорлары қолданылады;

– сіздің желіңіз қалай конфигурацияланған, қорғалған және оқшауланған;

– серверлер, хосттар және Ғаламтор пайдаланушылары арасында қандай қосылыстарға рұқсат етіледі;

– соңғы уақытта қандай қосылыстар болды;

– соңғы нүктелерден қосылыс немесе оларға дейінгі іс-әрекеттер ауытқушылық болып табылды ма;

– бұл қосылыстар қайда жасалған? Егер байланыс ішкі желіден тыс хосттарды қамтыса, онда бұл байланыстар қай жерде орналасқан? Оларды дәл анықтап, тексеруге бола ма.

Егер кейбір немесе барлық осы сұрақтарға жауап беруге болатын болса, келесі сұрақтар мынадай болады:

– сіз оны қайдан білесіз;

– сіз сенімдісіз бе.

Әдетте, бұл жауаптар Ақпараттық технологиялар жөніндегі бас қызметкерден немесе желіге жауапты IT-қызметкерлерден, сондай-ақ ақпараттық қауіпсіздік жөніндегі бас қызметкерден және кибер-қауіпсіздік жөніндегі тиісті қызметкерлерден келіп түседі. Осы топтардың әрқайсысы кибер-активтерді олардың бақылауымен басқаруға көмектесу үшін түрлі құралдарды пайдаланады. Бұл құралдар электрондық кестелердің қарапайым жиынтығынан бастап күрделі түгендеу және активтерді басқару жүйелеріне дейін өзгеруі мүмкін немесе ең болмағанда қызметкерлердің құлағында сақталуы мүмкін. Бұл барлық ақпарат, оның көзіне қарамастан, оқиғаға әрекет ету топтары үшін маңызды және құнды болып табылады. Оларға, әрине, не болып жатқанын немесе не болғанын, оны кім жасайтындығын, залалды қалай азайтуға және жоюға болатындығын, алдағы оқиғалардан жақсы қорғауды анықтау қиын. Алайда, жинау әдісіне қарамастан барлық деректер маңызды болып табылады.

Python Passive Network Mapping: P2NMAP – ашық бастапқы коды бар, жалған желілік белсенділікті анықтауға арналған шешім. Бұл «сіздің желіңіз неден тұрады және ауытқулар пайда болды ма» деген сұрақтарға жауап беруге мүмкіндік береді. Әдеттегідей, желілік карталау – IT-мамандар мен кибер-топтар желілік активтерді сәйкестендіру үшін құралдарды қолданатын белсенді үрдіс.

Nmap – компьютерлік желідегі хосттар мен қызметтерді анықтау үшін қолданылатын қауіпсіздік сканері, көрсетілген IP-адрестердің диапазондарына

өңделмеген IP-дестелерді жіберу арқылы жұмыс істейді. Оның көмегімен келесі мәліметтерді білуге болады:

– диапазон ауқымында қандай хост бар екенін;

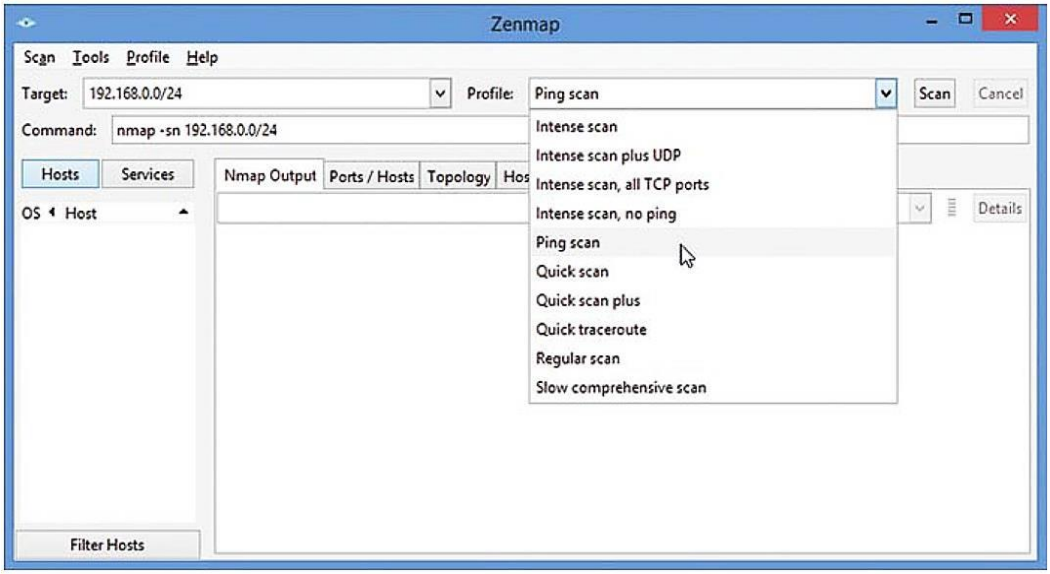
– табылған хосттардың әрқайсысында қандай қызметтер жұмыс істейтіндігн;

– осы хосттардың қандай операциялық жүйе қолданғанын.

Сондай-ақ, ол осы белсенді зерттеу әдісі арқылы сынауға және өлшеуге болатын басқа да көптеген сипаттамаларды анықтайды.

Қысқаша кіріспе ретінде Zenmap графикалық пайдаланушы GUI интерфейсін қолдана отырып, Windows үшін Nmap-тің қазіргі уақыттағы орындалуын қарастырайық.

2.1 суретте Zenmap 6.47 GUI нұсқасымен жұмыс жасайтын негізгі Nmap терезесі көрсетілген. Zenmap - стандартты Nmap командалық жолымен өзара әрекеттесетін, содан кейін нәтижелерді неғұрлым ыңғайлы және интерактивті форматта көрсететін көп платформалы графикалық интерфейс.




2.1 сурет – Zenmap графикалық интерфейсімен жұмыс істейтін Nmap-тың негізгі терезесі

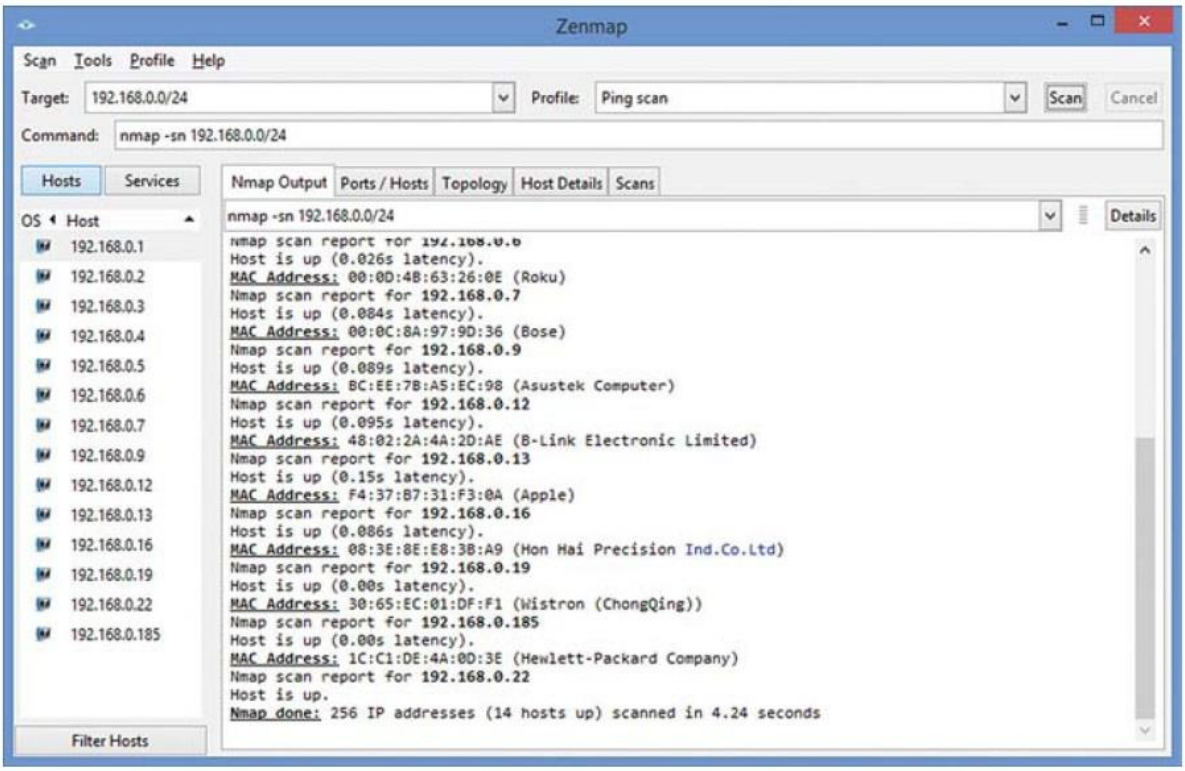
2.2 суретте көрініп тұрғандай, мен 192.168.0.0/24 мақсатты таңдауымен қарапайым pingScan-ды жүргіздім. Zenmap жасалған таңдау негізінде орындалатын нақты Nmap пәрменін көрсетеді.


2.2 сурет – 192.168.0.0/24 мақсатты таңдауымен қарапайым pingScan пәрмені

nmap –sn 192.168.0.0/24 пәрменінің мәні келесі:

– nmap – Nmap пәрмені;
– sn – қарапайым пинг-сканды көрсетеді;
– 192.168.0.0/24 IP-мекенжайы және диапазон.

Бұл жылдам сканерлеу нәтижелерін 2.3 суретте көруге болады. Менің жергілікті желідегі компьютерлер мен басқа да құрылғылар тізімін қараған кезде, қызықты жауаптар таба аласыз.




2.3 сурет – Сканерлеу нәтижесі

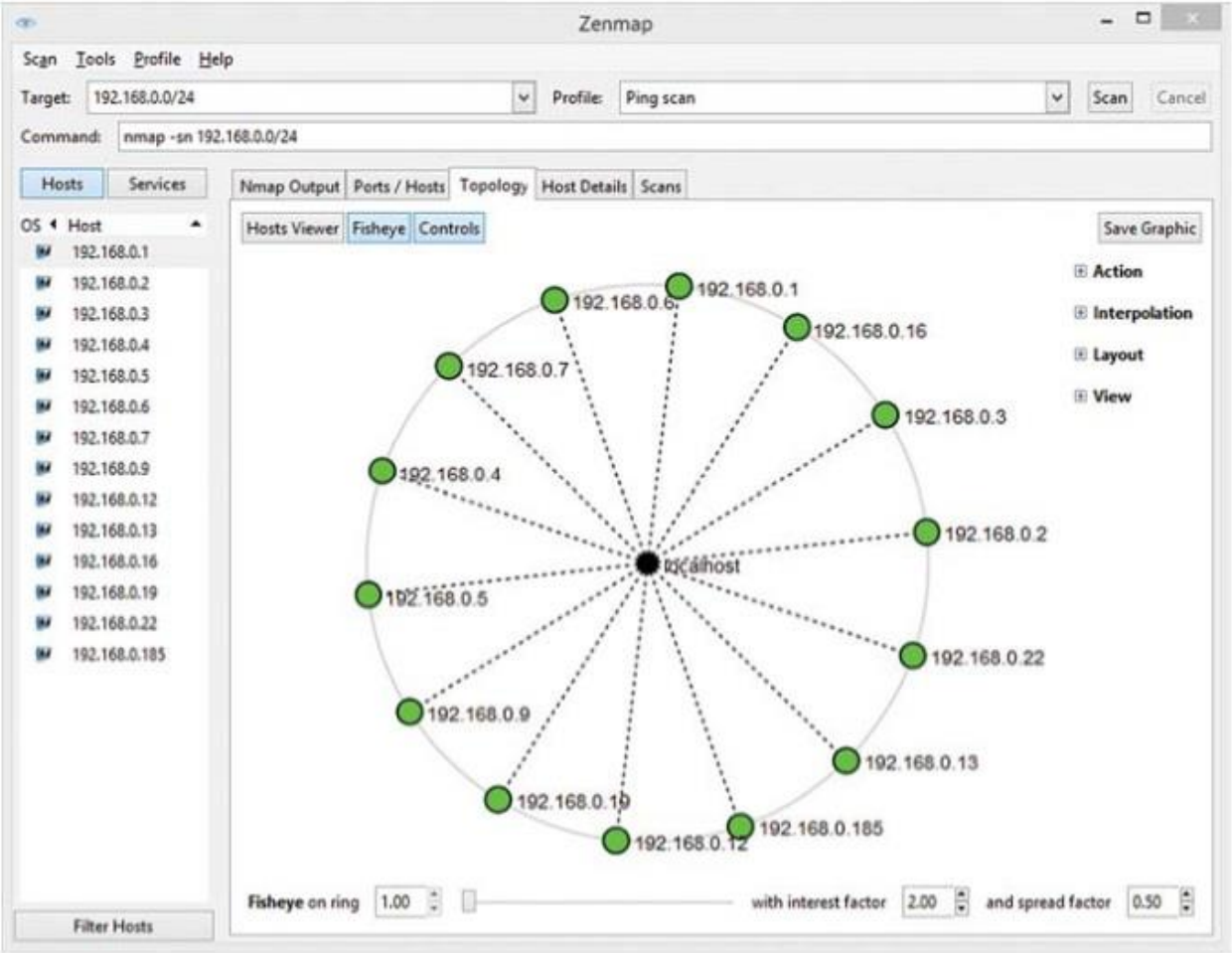
Жоғарыда көрсетілген суреттен келесілерді түсінуге болады:

– 192.168.0.7 IP-мекенжайы Ғаламтордан мазмұнды тарату үшін қолданылатын Roku өрісі ретінде анықталды;
– 192.168.0.7 IP-мекенжайы Radio ретінде анықталды;
– 192.168.0.13 IP-мекенжайы видеокамера ретінде анықталды;
– Apple құрылғысы ретінде анықталған 192.168.0.16 IP-мекенжайы;
– IP-мекенжайы 192.168.0.19 - DirecTV қабылдағышы;
– IP-мекенжайы 192.168.0.185 - Интернет-радио.

Бұл құрылғылардың өндірушісін идентификациялау OUI (ұйымдастыру бірегей идентификаторы) MAC-адресінің бөлігіне негізделген.

Бұл жергілікті желідегі белсенді құрылғыларды қарап шығуға мүмкіндік береді. Әрине, бұл сканерлеуге жауап беретін құрылғылар. Алайда, анықталмаған принтерлер мен басқа мобильді құрылғылар туралы не айтуға болады? Біз бұл мәселені жұмыс барысында талқылаймыз.

Егер сіз көрнекі болсаңыз, 2.4 суретте желінің IP-мекенжайларының графикалық көрінісі берілген. Бұл пайдаланушыларға нақты құрылғыларды талдауға және қосымша ақпарат алуға мүмкіндік береді.




2.4 сурет – Сәйкестендірілген желілік IP-мекенжайлардың графикалық көрінісі

Сонымен пинг деген не? Ping – су астындағы объектілердің орнын анықтау үшін пайдаланылатын дәстүрлі SONAR немесе "пинг" кибер-баламасы. Кибер-пинг іс жүзінде арнайы желілік хаттаманы, атап айтқанда Интернеттің басқару хабарламаларының ICMP хаттамасын пайдалануға жатады. Негізінен желілік құрылғылар белгілі бір қызметтердің қол жетімді немесе қол жетімді еместігін көрсететін қате туралы хабарлама жіберу немесе байланыс орнатып, белгілі бір күйді сұрау үшін қолданылады.

Хостты табу үшін ICMP Echo Request хабары белгілі бір IP-мекенжайларына сұраныс жіберу үшін қолданылады, содан кейін тиісті Echo Reply Type хабарламасын күту қажет болады. Әдеттегідей, егер сіз пинг жіберген хосттан жауап ала алмасаңыз, басқа қызметтер қол жетімді болмауы мүмкін. Көптеген жағдайларда қосылысқа қатысты ақаулықтарды жою

кезінде, пинг белгілі бір IP-мекенжайына қосылуды тексеру үшін қолданылады.

ICMP 2.5 суретте көрсетілгендей, TCP / IP хаттамалардың бір бөлігі болып табылады, ал ICMP хабарламалары 2.6 суретте көрсетілгендей IP-дейтаграммаларымен жіберіледі.


2.5 сурет – ICMP хаттамсының TCP / IP моделіндегі орналасу деңгейі

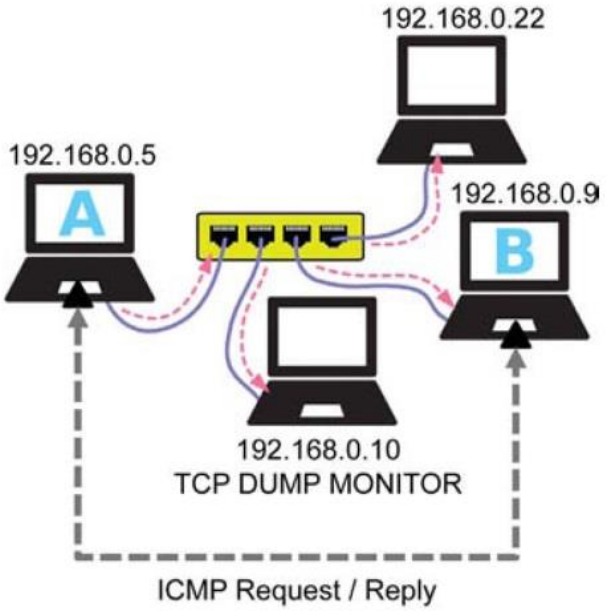

2.6 сурет – ICMP құрылымы

Хабарламалар мен кодтардың көптеген түрлері бар, олар 2.7 суретте көрсетілген. Хост табылған кезде, Echo Request Type 8, Code 0 және Echo Reply Type 0, Code 0, біздің негізгі қолданылуын білдіреді. Дегенмен, ICMP-дің желілік құрылғылар пайдаланылатын басқа да түрлері мен кодтары бар.

| Түрі | Код | Сипаттама | Сурақ | Қате |
|---|---|---|---|---|
| 0 | 0 | Echo Reply (Ping Reply) | ☑ | |
| 3 | | Destination Unreachable | | ☑ |
| | 0 | Network Unreachable | | ☑ |
| | 1 | Host Unreachable | | ☑ |
| | 2 | Protocol Unreachable | | ☑ |
| | 3 | Port Unreachable | | ☑ |
| | 4 | Fragmentation Error | | ☑ |
| | 5 | Source Route Failure | | ☑ |
| | 6 | Destination Route Failure or Unknown | | ☑ |
| | 7 | Destination Host Unknown | | ☑ |
| | 8 | Obsolete | | ☑ |
| | 9 | Destination Network Blocked | | ☑ |
| | 10 | Destination Host Blocked | | ☑ |
| | 11 | Network Unreachable | | ☑ |
| | 12 | Host Unreachable | | ☑ |
| | 13 | Communication Filtered | | ☑ |
| | 14 | Host Precedence Violation | | ☑ |
| | 15 | Precedence Cutoff | | ☑ |
| 4 | 0 | Source Quench | | ☑ |
| 5 | | Redirect | | ☑ |
| | 0 | Network Redirect | | ☑ |
| | 1 | Host Redirect | | ☑ |
| | 2 | Type of Service Redirect based on Network | | ☑ |
| | 3 | Type of Service Redirect based on Host | | ☑ |
| 8 | 0 | Echo Request Ping | ☑ | |
| 9 | 0 | Router Advertisement | ☑ | |
| 10 | 0 | Router Solicitation | ☑ | |
| 11 | | Time Errors | | |
| | 0 | Time to Live == 0 during transit | | ☑ |
| | 1 | Time to Live == 0 during reassembly | | ☑ |
| 12 | | Parameter Error | | |
| | 0 | IP Header Error | | ☑ |
| | 1 | Option Field Missing | | ☑ |
| 13 | 0 | Timestamp Request | ☑ | |
| 14 | 1 | Timestamp Reply | ☑ | |
| 15 | 0 | Obsolete | | |
| 16 | 0 | Obsolete | | |
| 17 | 0 | Address Mask Request | ☑ | |
| 18 | 0 | Address Mask Reply | ☑ | |

2.7 сурет – ICMP түрлері

Мысал үшін мен 2.8 суретте көрсетілгендей, 4 компьютерден тұратын қарапайым желіні баптадым.




2.8 сурет – Пинг тесті жүргізілетін желі құрылымы

Бұл мысалда, ping пәрменін пайдалана отырып, мен ICMP сұраныс дестелерін 192.168.0.5-тен 192.168.0.9-ға жібердім. 192.168.0.9 IP-мекенжайы сәйкес хабарламамен жауап берді (2.9 сурет).

```
chet@PythonForensics:~$ ping 192.168.0.9
PING 192.168.0.9 (192.168.0.9) 56(84) bytes of data.
64 bytes from 192.168.0.9: icmp_req=1 ttl=64 time=279 ms
64 bytes from 192.168.0.9: icmp_req=2 ttl=64 time=201 ms
64 bytes from 192.168.0.9: icmp_req=3 ttl=64 time=122 ms
64 bytes from 192.168.0.9: icmp_req=4 ttl=64 time=40.6 ms
64 bytes from 192.168.0.9: icmp_req=5 ttl=64 time=269 ms
64 bytes from 192.168.0.9: icmp_req=6 ttl=64 time=197 ms
```

2.9 сурет – Пинг пәрменін тестілеу нәтижесі

Дестелік кідірістің синхрондалғанын және 40,6-дан 279 мс-ге дейін өзгергенін байқай аласыз. Бұл сізге ерекше болып көрінуі мүмкін. Мен пингтің жауабын көрсету үшін осы нақты IP-мекенжайды (2.3 суретте көрсетілген) таңдадым. Көріп отырғаныңыздай, осы құрылғының жауаптары әдеттегі жұмыс үстеліндегі компьютермен салыстырғанда біршама тұрақсыз. Сонымен қатар, ICMP сұраныстардың әрқайсысында icmp req = 1, icmp req = 2 және icmp req = 6 деп белгіленген, әр түрлі реттік нөмір бар екенін байқай аласыз. Бұл пинг пәрмені 1-ден бастап өсетін мәнді қолданылатына байланысты, өйткені IP-дестелер, олардың анықтамасы бойынша, сенімсіз болып табылады және бұл дестелердің жоғалуы, кездейсоқ келуі немесе кешіктірілуі мүмкін. Соңында, сіз pingrequest сұранысында TTL 64 мәні бар екенін байқадыңыз, онда ttl дестенің өмір сүру уақытын білдіреді және 1 десте маршрутизатор арқылы өткен сайын оның саны 1-ге азаяды. Сондықтан 64-те орнатылған ttl мәні дестені 64-ке тең қарқынмен жіберуге мүмкіндік береді, ал бұл бір типті дестенің қайта-қайта жіберілуіне жол бермейді.

Мен сондай-ақ 192.168.0.10 мекенжайын tcpdump іске қосылған Linux хост ретінде баптадым. Tcpdump – TCP / IP деректерін жинайтын және жазатын желі мониторинг бағдарламасы. Tcpdump ең алдымен дестелерді алуға арналған, алайда бағдарламада сүзгілеу мен статистикалық есептеулерді орындауға және пайдаланушыларға олардың желісінің жұмысқа қабілеттілігін анықтауға көмектесетін ақпаратты ұсынуға көмектесетін көптеген опциялар бар.

Tcpdump сеансын орындау үшін мен келесі пәрмен жолын қолдандым:

sudo tcpdump –vv icmp

Көрсетілген sudo (su «do») пәрмені пайдаланушының өз есептік жазбасына қатысты тиісті артықшылықтары болған жағдайда кейбір (немесе барлық) әрекеттерді root пайдаланушы түрінде орындауға мүмкіндік береді. Tcpdump – root пайдаланушының атынан орындалатын үрдіс. VV – толық

қорытындыларды көрсететін tcpdump параметрі және соңындағы ICMP – tcpdump индикаторы, ол тек icmp дестелерін басып шығаруды көрсетеді. Төменде tcpdump пәрменімен алынған дестенің қысқартылған нәтижелері келтірілген (2.10 және 2.11 суреттер).

```
11:10:03.205298 IP (tos 0x0, ttl 64, id 18014, offset 0, flags
[DF], proto ICMP (1), length 84)PythonForensics.local >
192.168.0.9: ICMP echo request, id 4209, seq 1, length 64
11:10:03.484480 IP (tos 0x0, ttl 64, id 24829, offset 0, flags
[none], proto ICMP (1), length 84)192.168.0.9 >
PythonForensics.local: ICMP echo reply, id 4209, seq 1, length
64
```

2.10 сурет – Алынған дестенің tcpdump бағдарламасындағы бірінші сұранысы және жауабы

```
11:10:04.206413 IP (tos 0x0, ttl 64, id 18015, offset 0, flags
[DF], proto ICMP (1), length 84)PythonForensics.local >
192.168.0.9: ICMP echo request, id 4209, seq 2, length 64
11:10:04.407831 IP (tos 0x0, ttl 64, id 24830, offset 0, flags
[none], proto ICMP (1), length 84)192.168.0.9 >
PythonForensics.local: ICMP echo reply, id 4209, seq 2, length
64
```

2.11 сурет – Алынған дестенің tcpdump бағдарламасындағы екінші сұранысы және жауабы

Енді біз Nmap-пен танысып, пингтің негізгі сканерлеу туралы негізгі түсінік алған кезде, біз келесі P2NMAP бағдарламасын қарастырамыз.

Қарапайым тілмен айтқанда, P2NMAP – дестені желіге жібермей, тек Python бағдарламалау тілін қолдану арқылы желілерді бейнелеу әдісі. Сонымен қатар, ол желіге жүктеме түсірмей сканерлеуді жүргізуге мүмкіндік береді.

Бұл әдістің бірнеше артықшылықтары мен кемшіліктері бар. 2.1 кестеде осы кейбір артықшылықтар мен кемшіліктер көрсетілген.

2.1 кесте – P2NMAP артықшылықтары мен кемшіліктері

| Артықшылықтары | Кемшіліктері |
|---|---|
| 1. P2NMAP желіге жүктеме түсірмей сканерлеуді жүргізуге мүмкіндік береді. Бұл, әсіресе сканерлеу технологиялары кедергі болатын негізгі инфрақұрылымдық ортада өте маңызды болуы мүмкін. | 1. P2NMAP сканерлеудің ең белсенді әдістерімен қамтамасыз ететін суретке түсіру әдісімен салыстырғанда толық қозғалыстағы бейнені ұсынады.<br>2. P2NMAP кеңейтілетін ортаны |

*2.1 кестенің жалғасы*

| Артықшылықтары | Кемшіліктері |
|---|---|
| 2. Желінің толық картасын құрастыруға көп уақыт кетуі мүмкін, бірақ ол қоршаған ортаны неғұрлым толық бейнелейді.<br>3. Нақты операциялық жүйелер мен жабдықтардың түрлері сияқты бөлшектерді анықтау қиын.<br>4. Белгілі бір саясатқа ықтимал қауіпті немесе сәйкес келмейтін әркеттерді анықтайды. | ұсынады, мұнда пайдаланушылар ең танымал және үйренуге болатын бағдарламалау орталарының бірін қолдана отырып, жаңа мүмкіндіктер қосып, кеңейте алады. |

Барлық осал жүйелерді толық сәйкестендіру үшін осалдықтардың қазіргі заманғы сканерлері әрбір IP-мекенжайын және осы жүйелердің әрқайсысында жұмыс істейтін әрбір мүмкін болатын портты тексерулері тиіс. OpenSSL жалпы порттарын қарап шығып, осалдықты тексеру жеткіліксіз. Мыңдаған қосымшалар мен сервистер OpenSSL пайдаланады, ал көпшілігі 443 сияқты стандартты порттарды пайдаланбайды.

Бұл қосымшалар мен сервистерді сканерлеу үрдісінің дұрыс жүргізілуі келесі шарттарға тәуелді:

– барлық жүйелер қосылған болу керек;

– сканерлерге желіні қарау мүмкіндігі болу керек және олар брандмауэрмен немесе антивируспен бұғатталмау қажет;

– сканерлеу поцесі желі жұмысына кедергі болмайды;

– осал қызметтер жұмыс істеп тұру керек;

– осал қызметтер зерттеуге тиісті түрде жауап беру керек.

Сонымен қатар, егер бұл жүйелер күрделі инфрақұрылымдық ортада жұмыс істесе, байланыс операторлары әрбір IP-мекенжай мен әрбір порттың толық сканерленуіне рұқсат беру қажет. Оның орнына, зияндылықтың нөлдік тәуекелімен және Heartbleed әсеріне ұшыраған жүйелердің толық спектрін анықтаудың ықтималдығымен, осы ортаны пассивті түрде бақылау тиімді әдіс болып табылады.

Осыған байланысты Active Network Mapping бірнеше ерекше әсерлерге ие. Олар:

– желінің белсенді әрекеті хакерлік енуге еліктейді және шабуылдың алдын-алу жүйелеріне қарсы шаралар қабылдауға себеп болуы мүмкін;

– хост негізіндегі сенсорлар желідегі белсенді әрекетті шабуылға теңеп, байланысты ажырауы мүмкін;

– белсенді сканерлеу операциялары желіге, серверлерге, маршрутизаторларға және желілік құрылғыларға айтарлықтай жүктеме жасайды;

– сканерлерді конфигурациялаудағы қателіктер (мысалы, дұрыс емес IP ауқымдарын сканерлеу) байқаусызда көрші желілерге әсер етуі мүмкін. Егер сканерлеу осы желілердің зақымдалуына немесе жұмысының үзілуіне әкелсе, сканер операторлары жауапкершілікке тартылуы мүмкін.

Қазіргі заманғы орталар желелерді қорғауға арналған үлкен инфрақұрылымға және күрделі қауіпсіздік технологияларына ие.

Бүгінгі таңда қорғаныс технологиялары кешені дәстүрлі брандмауэрді, қосымшалардың брандмауэрлерін, DMZ, виртуалды жеке желілерді, антивирусты, түзетуді басқару инфрақұрылымды, мазмұн сүзгілерін, хост пен желіде деректерді қайта бағыттаудан қорғауды, артықшылықтар мен оқиғалар мен инциденттерді басқаруға арналған арнайы шешімдерді қамтиды. Өкінішке орай, бұл жүйелер мен технологиялар өздері қорғайтын ортада болатын жаңа қауіптерден немесе жасырын осалдықтардан қорғану үшін жеткіліксіз болып табылады. Кейбір жағдайларда осалдықтар қауіпсіздік шешімдерінде де болуы мүмкін.

Ақпарат қорын сақтау және қолдану біздің түсінігіміз бойынша, тек серверлер мен жұмыс станцияларда ғана орындалуы мүмкін деп қабылдаймыз. Алайды, біздің инфрақұрылымымыздың дамуына және өзгеруіне байланысты, мұндай үрдістерді көптеген басқа да құрылғыларда орындауға болады. Бізге бүгін қажет құрылғылар мен жүйелердің қысқаша тізімін қарастырайық (мен компьютерлерді және серверлерді әдейі алып тастадым):

– Android телефоны және планшет;
– iOS телефоны және планшет;
– Windows телефоны және планшет;
– Blackberry телефоны және планшет;
– принтерлер и көпфункциялы құрал-жабдықтар (басып шығару, сканерлеу, факс);
– VoIP;
– қауіпсіздік камералары;
– Ғаламтор-радио;
– NFC;
– конференц-зал байланысы;
– алып жүретін технологиялар (фитнес, бақылау).

Ұялы, сымсыз, Bluetooth, алып жүруге болатын және NFC құрылғыларының маңызды аспектісі – олардың желіде уақытша ғана белсенділік іздерін қалдыру беймділігінде. Бұл белсенді желіні сканерлеудің дәстүрлі әдістерінің тиімсіз болуы мүмкін дегенді білдіреді.

Осы мәліметтерге сүйене отырып, біздің желіге қосылу керек құрылғылар туралы нақты түсінікке ие болудың маңызды артықшылықтары бар екенін көре аласыз. Мұны үй өрісінің артықшылығы деп ойлаңыз, сіздің желіңізде не жұмыс істейтінін түсініп, онда болмауы керек құрылғыларды анықтау оңайырақ болады.

Бұл бөлімде көрсетілгендей, Nmap пайдаланып желідегі құрылғыларды белсенді сәйкестендіру анық күдіктілер туралы ақпаратты тез алуға болады. Мұнда біз уақытша немесе әдейі жасырылған құрылғыларды іздейміз. Пассивті сканерлеу құрылғылар белсенді қатысқанда өз қатысуын анықтағанға дейін күтуге тура келетінімен ерекшеленеді.

Тағы бір рет, біз пассивті дестелерді алудың кейбір әдістерін көрсету үшін tcpdump-қа жүгінеміз. Мен Wireshark немесе басқа да құралдар жиынтығын пайдаланып, бұл үрдісті жүргізе алатынымды түсінемін. Алайда, бұл тәсілдерді пайдалану үшін менде рұқсат болу қажет және бұл үшін күрделі қауіпсіздік құралдарын қолдануға тура келеді.Осылайша, менің жұмысымның басты мақсаты жоғары деңгейдегі операцияларды орындау үшін ашық бастапқы коды бар қарапайым белгілі технологияларды пайдалану болады. Осылайша, біз тек қажет үрдістерге ғана root артықшылықтарын беріп, ал басқаларды шектей аламыз. Сол сияқты, біздің талдау құралдары (біз қажетті дестелер үлгілерін жинағаннан кейін) пайдаланушы деңгейінде жұмыс істей алады.

Келесі қарапайым сұраққа жауап берейік. Менің желімдегі қандай компьютерлер қашықтағы веб-серверлерді пайдаланады? Мысалыға мен 80 порттың тағайындалған мекенжайы бойынша трафикті алғым келеді. Осыны көрсету үшін мен келесі Linux / Unix командаларын қолдана отырып, tcpdump көмегімен үй желісінен трафикті алдым.

Алдымен мен eth0 адаптерін ретсіз режимге ауыстырдым. Ол үшін келесі команданы енгіземін:

```
sudo ifconfig eth0 promisc
```

Жоғарыда көрсетілген пәрмен келесіні білдіреді:

- sudo – үрдісті толық рұқсат негізінде орындау;
- ifconfig – Linux пәрмені;
- eth0 – Ethernet адаптері;
- promisc – eth0 адаптерін ретсіз режимге ауыстыру.

Үрдіс аяқталғаннан кейін ifconfig қолдану арқылы нәтижелерді тексере аламыз. Көріп отырғаныңыздай, eth0 адаптері енді аралас көп адресті режимде жұмыс істейді (2.12 сурет).

Содан кейін tcpdump пәрменін 80 бастапқы порттан келетін кез келген дестелерді жинау үшін қолданамын:

```
sudo tcpdump –i eth0 –n src port 80
```

Жоғарыда көрсетілген пәрмен келесіні білдіреді:

- sudo – үрдісті толық рұқсат негізінде орындау;
- tcpdump – рұқсатпен іске қосқымыз келетін үрдіс;
- -i eth0 – трафикті алу үшін eth0 адаптерін қолдану;

– -n – IP-мекенжайды атаумен сейкестендірмеу;
– src port 80 – тек 80 порт арқылы келетін дестелерді алу.

```
$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:1e:8c:b7:6d:64
          inet6 addr: fe80::21e:8cff:feb7:6d64/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST
          MTU:1500  Metric:1
          RX packets:43842 errors:0 dropped:108
          overruns:0 frame:0
          TX packets:33 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4981889 (4.9 MB)  TX bytes:5723 (5.7 KB)
```

2.12 сурет – eth0 қосылыс күйі туралы ақпарат алу

Нәтижесінде пәрмен келесі деректерді қайтарады (2.13 сурет):

```
16:37:06.559388 IP 50.62.120.26.80 > 192.168.0.22.48637: Flags [.], seq
1:1461, ack 505, win 31, length 1460
16:37:06.560713 IP 50.62.120.26.80 > 192.168.0.22.48637: Flags [.], seq
1461:2921, ack 505, win 31, length 1460
<…SNIPPED…>
16:37:53.787370 IP 108.160.165.54.80 > 192.168.0.22.48532: Flags [P.], seq
380428656:380428835, ack 1424003206, win 31624, length 179
16:37:53.889243 IP 108.160.165.54.80 > 192.168.0.22.48532: Flags [.], ack
360, win 32696, length 0
16:37:59.812185 IP 173.194.37.84.80 > 192.168.0.22.48644: Flags [.], ack
1424, win 361, options [nop,nop,sack 1 {1423:1424}], length 0
<…SNIPPED…>
16:38:09.668759 IP 23.52.91.27.80 > 192.168.0.22.48660: Flags [.], ack 445,
win 490, length 0
16:38:09.670682 IP 23.52.91.27.80 > 192.168.0.22.48660: Flags [.], seq
1:1461, ack 445, win 490, length 1460
16:38:09.670758 IP 23.52.91.27.80 > 192.168.0.22.48660: Flags [P.], seq
1461:2244, ack 445, win 490, length 783
<…SNIPPED…>
```

2.13 сурет – Жеке желіні сканерлеу нәтижелері

Бұл желіні салыстыру тұрғысынан келесі бірегей мәндерге әкеледі (2.2 кесте).

2.2 кесте – Анықталған бірегей мәндер

| Сервер IP | Клиент IP | Бастапқы порт | Тағайындалған порт |
|---|---|---|---|
| 50.62.120.26 | 192.168.0.22 | 80 | 48637 |
| 108.160.165.54 | 192.168.0.22 | 80 | 48532 |
| 23.52.91.27 | 192.168.0.22 | 80 | 48660 |

## 2.2 Python тілінде жазылған дестелік трафикті алатын негізгі бағдарламаны әзірлеу

Пассивті желілік бейнені және оны Python-да оны қалай жүзеге асыру керек? Мен 443 порт арқылы сервер / клиенттің өзара әрекеттесуінің бірегей тізімін автоматты түрде құру мүмкіндігін қосамын.

Бағдарлама екі негізгі бөлімнен тұрады:

– негізгі бағдарлама;
– PacketExtractor () функциясы.

Негізгі бөлімде келесі төмендегі үрдістер орындалады:

– желілік интерфейсті аралас режимге баптайды;
– өңделмеген сокетті ашады;
– өңделмеген сокеттен дестелерді оқиды;
– дестені декодтау үшін PacketExtractor () функциясын қолданады;
– порт талаптарына сәйкес келетін дестелер тізімін жаңартады;
– дестелердің сәйкес саны жиналғаннан кейін бірегей тізімді құрайды.

Ал, PacketExtractor () функциясында келесі үрдістер орындалады:

– IP тақырыпты алады;
– TCP тақырыпты алады;
– бастапқы және тағайындалған IP-мекенжайларын алады;
– бастапқы және тағайындалған порт номерлерін алады;
– сервер мен клиентке қатысты негізделген болжам жасайды;
– ServerIP, ClientIP, ServerPort бар тізімді қайтарады.

А қосымшасында жоғарыда көрсетілген сценарийге сәйкес Python бағдарламалау тілінде жазылған негізгі бағдарлама көрсетілген.

2.14 суретте 443 порттағы желілік трафикті алу бағдарламасының нәтижесі көрсетілген.

Осы мысалдан көріп отырғаныңыздай, Python тілінде желілік трафикті алып, желідегі қарапайым әрекеттерін суреттеу қиынға түспейді. Бұл бағдарлама келесіде алынған трафик деректерін өңдеп, нақты хосттар мен қызметтерді анықтай алатын болады.

```
eth0 configured in promiscuous mode

Raw Socket Open

Server              Client         Port
Unique Packets
('173.194.37.62', '192.168.0.13', 443)
('199.16.156.241', '192.168.0.13',443)
('199.16.156.52', '192.168.0.13', 443)
('23.235.39.223', '192.168.0.13', 443)
('23.253.135.79', '192.168.0.13', 443)
('23.78.213.231', '192.168.0.13', 443)
('54.192.160.200','192.168.0.13', 443)
('64.233.185.132','192.168.0.13', 443)
('64.233.185.95', '192.168.0.13', 443)
('66.153.250.212','192.168.0.13', 443)
('66.153.250.240','192.168.0.13', 443)
('66.153.250.241','192.168.0.13', 443)
('69.172.216.111','192.168.0.13', 443)
('74.125.137.132','192.168.0.13', 443)
('74.125.196.154','192.168.0.13', 443)
('74.125.196.99', '192.168.0.13', 443)
('93.184.216.146','192.168.0.13', 443)

Unique Fingerprints

Server              TOS  TTL   DF  Window Size
('23.235.39.223',    0,  53,    1,   14480)
('23.253.135.79',    0,  50,    1,   14480)
('64.233.185.132',   0,  42,    0,   42540)
('64.233.185.95',    0,  42,    0,   42540)
('66.153.250.240',   0,  57,    0,   28960)
('66.153.250.241',   0,  57,    0,   28960)
('69.172.216.111',   0,  47,    1,   14480)
('74.125.137.132',   0,  46,    0,   42540)
```

2.14 сурет – 443 порттағы желілік трафикті алудың нәтижесі

Осы бағдарламаға қатысты арнайы ескертулер:

а) бұл бағдарлама тек Linux жүйелері үшін арналған;

б) бағдарламаның орындалуы пайдаланушы жағынан толық рұқсатпен жүргізілуі қажет. Ол үшін келесі пәрмен жазылады:

sudo python capture443.py

в) Tcpdump немесе Wireshark-қа қарағанда бағдарламаның артықшылығы:

1) пайдаланушының әрекетін толық бақылау;
2) операцияның қарапайымдылығы;
3) белгілі бір нәтижеге талап қою мүмкіндігі.

Бастапқыда мен ОЖ-дің сандық ізі тұжырымдамасын енгізгім келді, өйткені желілік салыстыруға байланысты көптеген пікірталастар нақты IP-мекенжайына жұмыс істейтін операциялық жүйені сәйкестендіруге тырысады. Бұл үрдіс пассивті әдістерді қолдану арқылы күрделене түсуі мүмкін, дегенмен қолданылатын ОЖ анықтайтын бағдарламаны жасауға болады. Келесі бөлімде біз трафикті басып алуға және талдауға, IP - мекенжайдың дипазонын толтыруға, порттардың / қызметтердің белсенділігін бақылауға және сәйкестендіруге, сондай-ақ пайдаланушылар туралы нақты ақпарат беруге мүмкіндік беретін сценарийлерді әзірлеуге баса назар аударамыз.

Әрбір IP-мекенжайда орындалатын ОЖ туралы ақпаратты пассивті түрде жинау үшін бірнеше белгілі атрибуттар бар. Олар 2.3 кестеде көрсетілген:

2.3 кесте – ОЖ іздерін алудың жалпы әдістері

| IP тақырып | Анықталуы | TCP тақырып | Анықталуы |
|---|---|---|---|
| TTL | Time to Live (IP хаттамасында деректер дестенің өмір сүру уақыты) | Терезе | Терезе өлшемі |
| TOS | Type of Service (Қызмет көрсету уақыты) | | |
| DF | Don’t Fragment Flag (Бөлінбейтін жалау) | | |

Бұл мәндер SYN жалаушасы нақты TCP дестесіне орнатылған болса ғана пайдалы болатындығын ескеріңіз. Байқасаңыз capture443.py бағдарламасында мен IP тақырыбынан TTL, TOS, DF шығарып, TCP тақырыбынан терезе өлшемін алдым (2.15 сурет). Сонымен қатар мен сандық іздердің бақыланған мәндерінің бірегей тізімін жасаймын. Содан кейін бұл бағдарламаны толық "бақыланатын" ОЖ-нің іздерін жасау үшін осы тақырып өрістерін жазатындай етіп пайдалануға болады.


2.15 сурет – Сандық ізінің негізгі өрістері ерекшеленген TCP / IP тақырыбы

Көптеген әдебиеттерден алынған бақылауларды негізге [10] ала отырып, 2.4 кестеде ОЖ сандық іздерін алу үрдісін түсіндіретін бақыланатын мәндердің тізімі келтірілген. Бұл іздерді алу үрдісі пассивті және белсенді карталау үшін бірдей. Тек қана жалғыз айырмашылығы бар. Бұл пассивті сканерлеу кезінде бақылау жасанды емес, қалыпты желілік трафикте жүргізілу керек.

IP-мекенжай негізінде ОЖ анықтау ең көп таралған құрылғылардың толық тізімін пайдалану арқылы мүмкін. TCP / IP тақырыбының осы өрістерін жасыру мүмкін екенін атап өту маңызды. Осылайша, бірнеше әдістерді пайдалану маңызды:

2.4 кесте – ОЖ бақыланатын мәндерін таңдау

| Қарастырылатын ОЖ | TTL | Терезе өлшемі |
|---|---|---|
| | Бастапқы мән | Типтік орнату |
| Linux | 64 | 5840 |
| OpenBSD | 64 | 16,384 |
| Solaris | 255 | 8,760 |
| AIX | 64 | 16,384 |
| Windows XP | 128 | 65,535 |
| Windows 2k | 128 | 16,384 |
| Windows 7 | 128 | 8,192 |
| Mac OS X | 64 | 65,535 |

Тағы бір кең таралған әдіс – ашық порт үлгілерін тексеру. Бұл әсіресе басқарылатын ортада жұмыс істейтін хосттар желісінде трафикті пассивті жинау кезінде пайдалы. 2.5 кестеде IP негізінде жұмыс істейтін операциялық жүйеге кілттерді ұсына алатын ортақ порттардың кейбірі ғана көрсетілген.

Біз келесі тарауда дедуктивті және индуктивті дәлелдемелерді қолдана отырып, ОЖ сандық іздерін талдауды қарастырамыз.

2.5 кесте – Ашық порттардан үлгілерді таңдау

| Порт номері | Ең көп қолданылатын түрі | ОЖ саусақ ізі туралы болжам |
|---|---|---|
| 455 | Microsoft Active Directory | Windows |
| 987 | Microsoft Sharepoint Service | Windows |
| 1270 | Microsoft System Center Operations Manager (SCOM) | Windows |
| 331 | Apple OS Server Admin | Mac OS X |
| 660 | Mac OS Server Admin | Mac OS X |
| 11111 | Remote Configuration Interface | RedHat Linux |

Мен осы тарауда келтірген қарапайым capture443.py бағдарламасына сүйене отырып, алынған нәтижелерге мынадай қорытынды жасауға болады. 192.168.0.13 жергілікті клиент келесі серверлерге қауіпсіз қосылуын орнатты:

– 199.16.156.201;
– 23.73.162.234;
– 66.153.250.229;
– 66.153.250.234;
– 66.153.250.238;
– 66.153.250.241;
– 74.125.137.132;
– 74.125.137.154;
– 74.125.196.99;
– 74.125.230.127.

Бұл қорытынды келесі деректер негізінде жасалды:

– RFC 1918 сәйкес 192.168.0.13 мекенжайы 192.168.0-192.168.255.255 (сондай-ақ 192.168.0/16 ретінде белгіленуі мүмкін) диапазонында С класының кез келген мекенжайы жеке және маршрутталмаған ретінде қарастырылуы тиіс. Бұл, егер мен С класы бірдей физикалық желіге қосылмаған болсам, осы ауқымдағы кез-келген С класының мекенжайына тікелей кіре алмайтынымды білдіреді.

– басқа IP-мекенжайлардың әрқайсысы географиялық орналасуы мүмкін. Мысалы, 199.16.156.201 мекенжайы Калифорния штатының Маунтин-Вьюде орналасқан. 66.153.25.x IP-мекенжайлары Оңтүстік Каролинада орналасқан. Осы IP-мекенжайлардың әрқайсысы қауіпсіз TLS немесе SSL қосылымымен жұмыс жасайтын әдепкі http хаттамасы болып табылатын 443 сервистік порт арқылы клиентпен байланысады.

Сонымен қатар, мен 192.168.0.13 клиенті басқа анықталған серверлерге қосылды деп қорытындылай аламын. Менің бұл тұжырымым дұрыс, себебі 74.125.137.x IP-мекенжайлары Google-ге [11] тиесілі және сіз бұл жерден 192.168.0.13 клиенті Google-дің көмегімен іздеуді жасағанын көре аласыз.

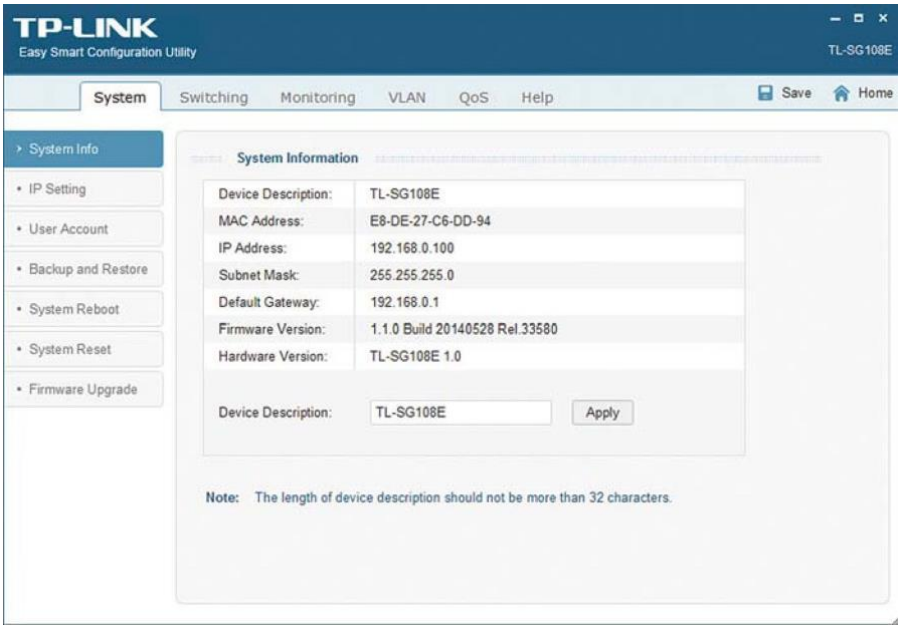
Пассивті желілік карталау орындау үшін, біз бүкіл үрдіс бойы дедуктивті және индуктивті әдістерді пайдаланамыз. Жасаған дәлелдеріміздің, бақылауларымыздың сапасы біздің нәтижелеріміздің қаншалықты дәл болатындығын анықтайды. Осыған сүйене отырып, уақыт өте келе дәлелдер мен бақылаулардың тұжырымдамаларын жетілдіру үшін оларды дұрыс құру маңызды болып табылады.

Осы сәтте желілік трафикті ұсынылған әдістерді пайдаланып алу үшін қандай орта қажет деген сұрақ туындауы мүмкін. Қазіргі заманғы желілік инфрақұрылымдардың көпшілігінде коммутаторлар коммутацияланатын порттардың SPAN анализаторы немесе порттардың қашықтағы коммутаторларының RSPAN анализаторы арқылы порттардың айналуын қолдайды. Өз тәжірибиелерім үшін мен 8-порттық гигабиттік Easy Smart

Switch TL-SG108E TP-LINK коммутаторын пайдаланамын. Бұл коммутатор 2.16 суретте көрсетілген.


2.16 сурет – 8-порттық гигабиттік Easy Smart Switch TL-SG108E TP-LINK коммутаторы

Коммутатордың қарапайымдылығы коммутатормен бірге келетін, 2.17 суретте көрсетілген, "Easy Smart Configuration Utility" бағдарламалық қосымшасына негізделген. Баптау құралы TL-SG108E коммутаторындағы барлық функцияларды қолданыға мүмкіндік береді.

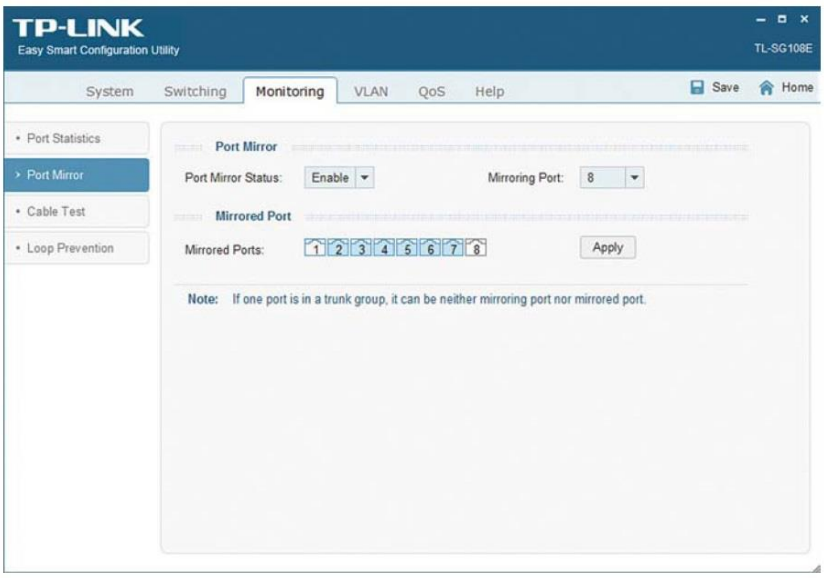



2.17 сурет – Easy Smart Configuration Utility жүйелік ақпарат терезесі

Біздің мақсатымыз үшін ең маңызды функция желілік трафикті пассивті түрде алуға болатын бақылау портын құру болып табылады. 2.18 суретте портты бақылауға арналған конфигурациялық терезесі көрсетілген. Бұл мысалда мен 8-портта 2-7 порттарға келетін трафикті бақылауға болатындай етіп баптадым. Бұл 2-7 портқа кіретін немесе шығатын барлық трафикті 8-порт арқылы бақылауға болатындығын білдіреді.

Е с к е р т у – Мен 1-ші портты таңдаудан тыс қалдырдым. Содан кейін мен бақылау құралын (бұл жағдайда Linux ОЖ орнатылған компьютерімді) коммутатордың 8 портына қосамын. Желілік трафикті алу және тәжірибе

жасау үшін бұл бөлімде жасалған tcpdump немесе Python бағдарламасын қолдана бастаймын.




2.18 сурет – Easy Smart Configuration Utility көмегімен портты бақылау конфигурациясын орнату

### 2.3 Алынған деректерді сақтау

Біздің алдымызда тұрған келесі мәселе – жинақталған дестелерді сақтау, соның ішінде қандай ақпаратты сақтау керек екенін анықтау. Python осы мақсат үшін көптеген ішкі деректер құрылымдарын ұсынады. Мен Python тізіміндегі нысандарды түсірілген деректерді сақтау үшін қолдандым (2.19 сурет):

```
ipObservations = []
osObservations = []
```

2.19 сурет – Жинақталған деректерді сақтау үшін Python тізім нысандары

ipObservations тізіміндегі әрбір жазбада келесі нысандар бар:

– serverIP – таңдалған сервердің IP-мекенжайы;
– clientIP – таңдалған клиенттің IP-мекенжайы;
– serverPort – таңдалған сервер порты.

Әрбір osObservations жазбасы дестеге SYN жалауы орнатылған болса, TCP / IP тақырыбынан алынған деректерді қамтитын тізімді жасайды. Бұл деректер кейінірек ОЖ сандық іздерін анықтау үшін көмекші құрал ретінде пайдаланылады. OsObservations әрбір тізімінде мына жазбалар бар:

– serverIP – таңдалған сервердің IP-мекенжайы;
– TOS – қызмет көрсету өрісінің түрі;

– DF – битті бөлшектемеңіз жазбасы; егер орнатылған болса, десте бөлшектенбейді;

– windowSize – серверді өңдеуге болатын ең үлкен TCP қабылдау терезесі;

– timeToLive – дестесінің қызмет ету мерзімі үшін желінің ауысу шегін орнатады. TTL маршрутизатордан өткен сайын бір-бірден азаяды. Оның мәні нөлге жеткен кезде, бұл үрдістің шексіз қайталанбауы үшін десте алып тасталынады.

Бұл тәсілдің жақсы жағы – қарапайымдылғы. Дегенмен Python тізімінде көшірмелер бар және жиналатын дестелер санына байланысты біз дестелер туралы сақталған ақпараттың көлемін азайтуды қалаймыз. Сонымен қатар, талдауға пайдалы болатын қосымша ақпаратты енгізу мүмкіндігі бар. Дәлірек айтсақ, дестелердің келіп түсу уақыт.

### 2.4 Python сөздіктерін қолдану

Келесі сұрақ, біз жинайтын деректерді толық қанықтырмай қалай сақтауға болады? Әр дестенің уақыты әр түрлі болғандықтан, егер біз әр дестенің нақты уақыт мәнін сақтауға шешім қабылдасақ, біз serverIP, ClientIP, serverPortpacket көшірмелерін алып тастай алмаймыз. Осылайша, мен қайталанған дестелерді есепке алмай, әрбір десте туралы маңызды ақпаратты сақтау әдісін ойлап таптым. Сонымен қатар, бұл тәсіл Python сөздіктерін сақтаудың негізгі үрдісі ретінде іске асыруға мүмкіндік береді.

Python сөздіктері тілге енгізілген және сондықтан бізге өте пайдалы. Әдетте, Python сөздіктері Key / Value жұптары болып табылады. Мұндағы кілт пен мән тізімдер немесе кортеждер сияқты күрделі түрде болуы мүмкін.

Кортеж деген не? Кортеж – өзгермейтін Python нысандарының тізбегі. Кортеждер – тізімдерге өте ұқсас тізбектер, бірақ оларды өзгерту мүмкін емес. Кортеждердің үлкен артықшылығы бар. Олар сөздікте кілт ретінде пайдаланылуы мүмкін.

Сондықтан ipObservations тізімін ауыстыру үшін сөздік жұп кілт / мәнді құрайық (2.20 сурет).

```
Key  = tuple (serverIP, clientIP, serverPort)
Value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ]
```

2.20 сурет – Сөздік жұп кілт / мән

Әрбір мән жазбасы - бұл комбинацияның бір сағатта пайда болу саны. Тәуліктің әр сағатын білдіретін 24 мән бар. Мұнда сөздік құрудың әдісі көрсетілген кодтың үзіндісі берілген (2.21 сурет). Біз топтамалық деректерді динамикалық түрде шығарып, сөздік пен кілт / мән жұптарын құратынымыз анық.

```
1   ipObserved = ()        # Бос сөздікті жасау
2   # Бірнеше жалған бақыланатын жұпты (кілт / мән) жасау
3   ipObserved["192.168.0.2", "129.168.0.39", 80] = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
4   ipObserved["192.168.0.2", "129.168.0.40", 80] = [0,0,0,0,0,0,5,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,9,0]
5   ipObserved["192.168.0.2", "129.168.0.41", 80] = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
6   # Белгілі кілттен мәнді алу
7   value = ipObserved["192.168.0.2", "129.168.0.40", 80]
8   # мәнді басып шығару
9   print(value)
10  # Скрипт шығысы
11  Value = [0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 9, 0]
```

2.21 сурет – Дестелік деректерді алу және оны сөздікке жазу коды

Мұндай тәсіл тек қосылыстардың бірегей бақылауларын жазып, қойманың өлшемін минимумға дейін сақтауға мүмкіндік береді (мысалы, сервер-клиент бірегей қосылыстары). Сонымен қатар, мен келесі тарауда күн сағаттарына негізделген, қызмет көрсету түрі мен serverIP, clientIP қосылыс гистограммаларын жасауға қабілетті болатын функияларды бағдарламаға қосамын.

Бұл тәсілді бірнеше рет қолдану үшін, мен IPObservationDictionary (Б қосымшасы) өңдеу класын құрдым. Алдымен бұл класс қарапайым болады, ал біз Python Capture үрдісімен жиналған деректерді өңдегенде келесі тарауда кеңейтіледі.

Кодты орындалу мысалы мен нәтижесі сөздік пен кластың дұрыс жұмыс істейтіндігін тексереді. Біз кластың әр әдісін тексердік:

– init – бос сөздікті құрады;

– AddOb – сөздікке бақылауды қосады. Егер кілт болмаса, ол жаңа жазба жасайды. Кілт бар болса, ол гистограмма үшін қажетті сағатқа (уақыт аралығына) бақылауды қосады;

– GetOb – кілттің негізінде алу және бақылау әрекеттері. Егер кілт жоқ болса, ол «жоқ» сөзін қайтарады;

– SaveOb – ағымдағы Dictionary нысанын біздің таңдауымыз бойынша файлға сақтайды;

– LoadOb – бұрын сақталған Dictionary нысанын жүктейді.

IPObservationDictionary класына өте ұқсас. Бұл класс операциялық жүйенің бақылау деректерін сақтау операцияларын өңдейді (В қосымшасы). Бұл бақылаулар мыналарды қамтиды: ServerIP, TOS, timeToLive, DF және windowSize. Олардың барлығы осы тарауда бұрын анықталған.

Енді біз осы бөлімде әзірленген қарапайым желілік трафикті алу бағдарламасына келесі мүмкіндіктерді қосу қажет:

– командалық жолда көрсетілген TCP немесе UDP дестелерін басып алуға рұқсат ету;

– жаңа құрылған IPObservationsClass басып алу дестелерін сақтауға рұқсат ету;

– жаңа құрылған OSObservationsClass Операциялық жүйе бақылауларын сақтауға рұқсат ету;

– PrintOb әдісін IPObservation класына да, OSObservation класына да қосу. Бұл бақылау мазмұнын басып шығаруға мүмкіндік береді;

– пайдаланушыға басып алу уақытын көрсетуге рұқсат ету;

– басып алу нәтижелерін келесі талдау үшін файлға сақтау.

Мен трафикті алу уақытын қоспағанда, осы жеке қадамдарға мен басты ерекшеліктерге назар аудардым. Осы мақсатқа жету үшін мен сигнал беру ұғымын енгіземін және уақыт өткеннен кейін ерекшелікті енгіземін. Содан кейін мен негізгі бағдарлама циклына ерекше жағдайларды өңдеу әрекетін қосамын. Бұл бірнеше қашықтықтан қадамдарды қажет етеді. Олар:

– мен myTimeout класын құрамын, ол өңдеуші басталған кезде сценарийде ерекше жағдайды тудырады (2.22 сурет);

– мен белгіленген уақыттан кейін күту уақыты болатын сигнал өңдегішін жасаймын;

– трафикті алу ұзақтығына қарай шектеулі уақытты белгілеуім керек;

– соңында, try / except блогында белгілі бір үзілістер пайда болады және олар қайталанатын циклды үзеді.

Е с к е р т у – Трафикті алу ұзақтығы секундпен берілген.

```
1   # Трафикті алу ұзақтығын өңдеу үшін күту уақыты класын жасау
2
3   class myTimeout(Exception):
4       pass
5
6   # Трафикті алу ұзақтығы жеткен кезде үзілісті
7   # тудыратын сигнал өңдегішін жасау
8
9   def handler(signum,frame):
10      print('timeout received', signum)
11      raise myTimeout()
12
13  # Сигнал өңдеушісін пайдаланушы көрсеткен уақытқа орнатыңыз
14
15  signal.signal(signal.SIGALRM, handler)
16  signal.alarm(60)   # Бір циклдың уақыты 60 секунд
17
18  try:
19      # Мәңгілік циклді құру
20      # try / except бөлімдердің ішінде
21
22      while True:
23          # жұмыс істеу (мысал)
24          a = 1+1
25
26  # үзіліс пайда болса, мәңгілік цикл үзіледі
27  # және while оны қайта жалғастырға мүмкіндік береді
28
29  except myTimeout:
30              pass
```

2.22 сурет – myTimeout класының коды

Г қосымшасында көрсетілген, P2NMAP трафикті алудың түсініктеме бағдарламасы жоғарыда анықталған барлық мүмкіндіктерді қамтиды. Мен сондай-ақ дестелерді алу бағдарламасының шығыс нәтижелерін Д қосымшасында көрсеттім.

Мен келесі тарауда осы бағдарлама нәтижелерінің негізінде нақты желі картасын жасайтын боламын.

Windows жүйесінде P2NMAP-Capture.py скриптің қолдану үшін командалық жолға келесілерді енгізу қажет (командалық жол толық рұқсатпен іске қосылуы тиіс екенін ескеріңіз):

python P2MAP-Capture.py –v –m 2 –p ./

Linux жүйелері үшін төмендегіні енгізу керек:

sudo python P2MAP-Capture.py –v –m 2 –p ./

Бұл бөлімде біз Nmap-ты және қарапайым желіні сканерлеу және карталаудың негізгі әдісін қарастырдық. Біз ICMP хаттамасын тексеріп, желідегі IP-мекенжайларды анықтау үшін ICMP Requests және Reply хабарламаларымен алмасуын ping үрдісі арқылы көрсеттік. Осы үрдіс арқылы желіге қандай құрылғылардың  қосылғанын анықтадық. Біз сондай-ақ құрылғылардың осы кең спектрін қарастырып, олармен байланысты тәуекелдерді талқыладық. Бұдан әрі біз P2NMAP дегеніміз не және осы тәсілдің кейбір артықшылықтарына және кемшіліктеріне қысқаша шолу жасадық. Сонымен қатар біз желілік трафикті пассивті түрде алу үрдісін неғұрлым қауіпсіз және мұқият әдіс екенін анықтадық.

Содан кейін біз Linux жүйесі және tcpdump бағдарламасы арқылы желіні басып алуды аралас режимге баптадық. Мұның негізінде жұмыс істейтін, Python тілінде жазылған бағдарламаны әзірледік, бірақ ол 443 порт арқылы келетін TLS / SSL бойынша http хаттамасымен байланысты желілік белсенділікті қарастырады. Сонымен қатар, бағдарлама әдеттегі IP-мекенжайын және тағайындалған IP- серверлік сипаттамасына аударады және негізделген болжам жасайды. Бұл бізге 443 портында орын алатын клиент-сервердің өзара әрекеттесуінің бірегей тізімін автоматты түрде құруға мүмкіндік берді. Содан кейін біз TCP / UDP порттарын салыстыруды зерттеп, белгілі, тіркелген және уақытша порттардың диапазонын анықтадық.

Содан кейін біз дестелерді алу ортасын реттеу үшін қажетті эмпирикалық ережелерді, соның ішінде коммутаторды таңдау мен баптау, сондай-ақ жүйенің аппараттық қамтамасыз етілуіне қатысты түсініктерді қарастырдық. Содан кейін біз желілік дестелерден жинау үшін талап етілетін «кейбір» ақпаратты зерделедік. Нәтижесінде желіні пассивті бейнелеуге және операциялық жүйені сәйкестендіруге көмектеседі. Содан кейін біз дестелердің нәтижелерін динамикалық сақтай алатын Python деректерінің бірнеше түрлерін қарастырдық және деректерді сақтау түрі ретінде Python сөздігі таңдалды. Қайталанатын бақылауларды жою мақсатында, осы сөздіктерді құруға ерекше назар аударылды. Сонымен қатар, біз сервердің IP-мекенжайының, сервер портының және клиенттің IP-мекенжайының әрбір бірегей комбинациясы үшін дестелердің ұқсас пайда болуының незізгі гистограммасын шығару әдісін әзірледік.

Осы кезеңге дейін біз екі класты құрдық. Олар IPObservationsDictionary және OSObservationDictionary. Аталған кластар сөздікті құру, қосу, оқу, жүктеу, сақтау және басып шығарумен айналысады. Содан кейін біз уақытты ескере отырып, дестелік алуды өңдеудің сигнал беру тұжырымдамасын енгіздік. Ақырында, біз дестелерді алу және сақтау үшін осы мүмкіндіктердің барлығын бір бағдарламаға біріктірдік.

## 3 Алынған дестелік трафикті талдау

### 3.1 Талдау тәсілдерін әзірлеу

Енді бізде Windows платформаларында да, Linux-та да жақсы жұмыс істейтін Python дестелерін басып алу құралы «P2NMAP-Capture.py» бар кезде, біз жиналған деректерге байланысты кейбір пайдалы талдауларды орындай аламыз.

2 бөлімде байқасаңыз, бағдарлама екі шығыс файлын жасаған (3.1 сурет):

| Name | Size | Type | Modified |
|---|---|---|---|
| __pycache__ | 8 items | Folder | Сәу 2 |
| venv | 5 items | Folder | Мам 28 |
| 20200524-185904.ipDict | 8,6 kB | Text | Мам 24 |
| 20200524-185904.osDict | 8,4 kB | Text | Мам 24 |
| 20200524-190223.ipDict | 12,1 kB | Text | Мам 24 |
| 20200524-190223.osDict | 8,2 kB | Text | Мам 24 |
| 20200524-190749.ipDict | 12,8 kB | Text | Мам 24 |
| 20200524-190749.osDict | 12,7 kB | Text | Мам 24 |
| 20200524-191400.ipDict | 9,7 kB | Text | Мам 24 |
| 20200524-191400.osDict | 4,7 kB | Text | Мам 24 |

3.1 сурет – P2NMAP-Capture.py іске қосқаннан кейін жасалған екі шығыс файл

20200524-185904.ipDict Интернет хаттаманың бақылау сөздігін, ал 20200524-185904.osDict операциялық жүйенің бақылау сөздігін қамтиды. Бұл тарауда мен .ipDict бақылауларын талдауға тоқталамын.

P2NMAP тәсілінің негізгі аспектісі желілік трафиктің пассивті мониторингі және нәтижелерді жазу болып табылады, ол тіпті дестені желіге орналастырмайды. Екінші аспект, егер күн болмаса, кем дегенде бірнеше сағат ішінде өлшенетін уақыт кезеңі үшін деректерді жинау болып табылады. Бұл тәсіл желілік құрылғыларды зерттейтін белсенді бейнелеу әдістерімен

тікелей қарама-қайшы келеді және екі әдістің де артықшылықтары мен кемшіліктері бар.

Пассивті тәсілдің негізгі артықшылықтарының бірі бірнеше күн немесе тіпті апта бойы желілік құрылғылардың белсенділігн бақылау және серверлердің, сондай-ақ осы кезеңдегі клиенттердің әрекеттерін көрсету мүмкіндігі болып табылады.

Техникалық тәсілдің тағы бір маңызды аспектісі ipObservationDictionary класын әзірлеу болып табылады. Осы мақсат үшін класты пайдалану бізге класты талдау әдістерін әзірлеу үшін бастапқы нүкте ретінде қайта пайдалануға мүмкіндік береді. Мысалы, класс IP-сөздіктің файлдарын сақтау және жүктеуге арналған әдістерді, сондай-ақ қазіргі уақытта жүктелген сөздікте сақталатын Ғаламтор хаттамасының бақылауларын басып шығаруға арналған әдістерді қамтиды. Кластың және нысандардың даналарының нәтижесінде алынатын мүмкіндіктерін кеңейте отырып, біз қазір және болашақта талдау мүмкіндіктерін кеңейту үшін қарапайым әдісті ұсына аламыз.

Бастапқыда дестелік трафикті алу кезеңінде ipObservationDictionary класына қосылуы тиіс әдістердің жинағына мыналар кіреді:

– бақылау файлын жүктеу;

– барлық жазылған бақылауларды басып шығару;

– пайдаланушы порттарымен бірге бірегей идентификацияланған серверлер тізімін басып шығару;

– сәйкестендірілген клиенттердің бірегей тізімін басып шығару;

– порт туралы толық ақпараты бар бірегей қосылыстар (клиент / сервер) тізімін басып шығару;

– әр бірегей сервер / байланыс клиенті үшін 24 сағаттық гистограмманы басып шығару.

Бұл талдау элементтерінің негізгі жиынтығынан басқа, мен талдаушыға қосымша ақпарат беру үшін үш арнайы іздеуді ұсындым. Олар келесілерді қамтиды:

– порт нөмірін порт атына түрлендіру;

– IP-мекенжайы негізінде хост атын іздеу;

– IP-мекенжайы негізінде елді анықтау.

Е с к е р т у – Бұл әдіс Ғаламторға қосылыстың орнатылуын талап етеді

Бұл мүмкіндіктерге қол жеткізу үшін мен белгілі бір талдау операцияларын орындау үшін қарапайым менюмен басқарылатын P2NMAP-Analyze.py бағдарламасын жасадым. 3.2 суретте P2NMAP-Analyze.py. бағдарламасының мәзірі бейнеленген.

```
========== P2NMAP Analysis Menu ==========

Current Observation File:  test.ipdict

[L]    Load Observation File for Analysis
[O]    Direct Output to File   (Current = Stdout)
[H]    Turn On Host Lookup     (Current = Host Lookup Off)
[C]    Turn On Country Lookup  (Current = Country Lookup Off)
==========================================================
[1]    Print Observations      (ALL)
[2]    Print Servers           (Unique)
[3]    Print Clients           (Unique)
[4]    Print Connections       (Unique by Server)
[5]    Print Histogram

[X]    Exit P2NMAP Analysis

Enter Selection:
```

3.2 сурет – Белгілі бір талдау операцияларын орындау мәзірі

Осы тарауда мен мәзірдің әрбір операциясы үшін жұмысты, іске асыруды және негіздеуді талқылайтын боламын.

### 3.2 Желілік трафиктің талдау нұсқаларын баптау

Талдау әдістерін бастамас бұрын, интерфейсте бірнеше көрсеткіштерді орнату қажет. Олар келесілерді қамтиды:

– бақылау файлын жүктеу;
– бағдарламаның шығу бағыты;
– хосттың іздеу опциясын көрсету;
– елді іздеу опциясын көрсету.

Бақылау файлын жүктеу өте қарапайым. Басып алу үрдісі кезінде мен Python Pickle модулін пайдаланып ipDict файлын сақтадым. Python Pickle стандартты кітапханасының модулі таңдау кез келген Python нысанын тізім, сөздік немесе кез келген басқа объект, таңбалар ағыны сияқты түрлендіретін нысанды таңдауға және таңдамауға мүмкіндігін береді. Таңбалар ағыны басқа Python бағдарламасында нысанды қалпына келтіру үшін қажет болатын барлық ақпаратты қамтиды. Мен дәл осылай жасағым келді, өйткені мен P2NMAP-ті қолданып трафикті алу және талдау мүмкіндіктерін бөлдім.

Осы мақсатқа жету үшін қажетті екі әдіс төменде көрсетілген (3.3 сурет):

```
pickle.dump(self.Dictionary, fp)              # Нысанды файлға жазу

self.Dictionary = pickle.loads(fp.read())     # Нысанды файлдан оқу
```

3.3 сурет – Файлға жазу және шығару үшін Pickle модулін пайдалану

3.3 сурет, self.Dictionary - сақталынатын немесе жүктелінетін Dictionary нысаны. fp нысаны шығыс немесе кіріс файлының көрсеткіші болып табылады.

Мен төменде көрсетілгендей IPObservationDictionary класына келесі әдісті қостым (3.4 сурет):

```
# Көрсетілген файлдан
# бақылау сөздігіне жүктеу

def LoadOb(self, fileName):
try:
    with open(fileName, 'rb') as fp:
        self.Dictionary = pickle.loads(fp.read())
        self.observationFileName = fileName
        self.observationsLoaded  = True
except:
    print("Loading Observations - Failed")
    self.observationsLoaded = False
    self.observationFileName = ""
```

3.4 сурет – IPObservationDictionary класының коды

Егер әдіс сәтті болса, онда ол нысанның келесі атрибуттарын орнатады:

– self.observationsLoaded True мағынасында;
– self.observationFileName жүктелген файл атауына.

Бұл екі атрибут IPObservationDictionary класында басқа әдістермен қолданылады.

Алайда, егер жүктеу сәтсіз болса, self.observationsLoaded жалған мәнге ие болады, ал self.observationFileName бос қалады. Сонымен қатар, қате туралы хабарлама пайдаланушыға көрсетіледі.

Қолданыстағы бақылау файлы сәтті жүктелгенге дейін бағдарлама жұмысы кезінде пайдаланушыға ешқандай басқа операцияларды іске асыра алмайды.

3.5 суретте «>» қайта бағыттау таңбасын пайдалану мәселесі көрсетілген.

```
~$ python P2NMAP-Analysis.py > results.txt
```

3.5 суретте – Файлға жазу үшін шығыс деректерін қайта бағыттау пәрмені

Осы пәрменге сәйкес хабарламалар нәтиже файлына, оның ішінде шақыруларға, ақпараттық және ескерту хабарламаларына жіберіледі. Мұны Python-да келесі әдісті қолдана отырып шешуге болады.

Мен OUT деп аталатын айнымалыны құрып, оны төменде көрсетілгендей жалпы әдіс нәтижесіне тең етіп орнатамын (3.6 сурет). Содан

кейін, әр басып шығару хабарламасында мен «print >> OUT» деп жазамын, ал қалғаны күрделілігіне қарамай тікелей шығару файлына жазылады. Бұл шығыс файлы «стандартты шығыс» көмегімен экранда көрсетілетін шығыс сияқты көрінуге мүмкіндік береді.

```
OUT = open("results.txt", 'w+')

print >> OUT, ("This is a message")
```

3.6 сурет – OUT айнымалысы

Егер OUT айнымалысы Ғаламдық болса, онда ол пайдаланушыға айнымалыны өзгертуге мүмкіндік береді. Шығыс деректері тиісті шығыс сигналына жіберіледі. Бұл жағдайда «стандартты шығыс» файлына немесе results.txt файлына жіберіледі (3.7 сурет).

```
OUT = sys.stdout

print >> OUT, ("This is a message")
```

3.7 сурет – OUT айналымы

Мұны модульде іске асыру үшін, мен пайдаланушыға «стандартты шығыс» пен файл арасындағы шығыс бағытын өзгертуге мүмкіндік беретін қосқышты жасаймын. Осылайша, талдаушы экрандағы шығыс деректерін көре алады, содан кейін олар нәтижелермен қанағаттанған соң, олар шығыс деректерін файлға ауыстыра және жібере алады. Пайдаланушы мәзірден "O" шығару нұсқасын таңдайтын кезде ауыстырып қосуды орындайтын кодтың фрагменті 3.8 суретте көрсетілген.

Мен close OUT.close () әдісін файлды кері STDOUT-ке ауыстырғанда орындайтынымды көре аласыз. Бұл файлдың жабылуына және барлық деректер файлға жазылуына әкеледі. Сонымен қатар, мен шығыс файлын "w +" пайдаланып ашамын, бұл деректер results файлына қосылады дегенді білдіреді.txt.

```
elif menuSelection == 'O':
    if PRINT_STDOUT:
        PRINT_STDOUT = False
        OUT = open("results.txt", 'w+')
    else:
        PRINT_STDOUT = True
        OUT.close()
        OUT = sys.stdout
```

3.8 сурет – Мәзірден «O» шығыс нұсқасын таңдау коды

Хост іздеу параметрін көрсету

Пассивті желіні сканерлеудің маңызды аспектілерінің бірі IP-мекенжайларын хостардың аттарымен салыстыру болып табылады. Бұл хост атына желілік адрестерді ауыстыру арқылы жасалады. Мен мұны өз Python кодымды және Python стандартты кітапханаларын пайдалана отырып жасаймын. Бірақ ол үшін Ғаламторға қосылу қажет. Мен пайдаланушыға Host Lookup әдісін қосу немесе өшіру мүмкіндігін беру үшін мәзір жүйесінде тағы бір рет қосқышты жасаймын. Әдетте Host Lookup өшірілген болады (3.9 сурет).

```
elif menuSelection == 'H':
    if HOST_LOOKUP:
        HOST_LOOKUP = False
    else:
        HOST_LOOKUP = True
```

3.9 сурет – Мәзірден «H» шығыс нұсқасын таңдау коды

Host_lookup айнымалысы содан кейін талдау әдістерінің әрқайсысымен бағаланады. Егер HOST_LOOKUP True мәніне ие болса, онда талдау әдістері IP-мекенжайды хост атына түрлендіреді. Бұл әдістің коды, сокет деп аталатын, Python стандартты кітапханалық модульді пайдаланады. Бұл кодта сокет тек қана бір рет пайдаланылады (3.10 сурет):

```
try:
    # егер шақырушы тарапынан хосттың атын іздеу сұранысы түссе,
    # іздеуді жасау керек, немесе бос атауды орнатамыз
    if HOST_LOOKUP:
        hostName = socket.gethostbyaddr(serverIP)
    else:
        hostName = ["", "", ""]
except:
    hostName = ""
    continue
```

3.10 сурет – HOST_LOOKUP айнымалысын өңдеу әдістері

Е с к е р т у – socket.gethostbyaddr() үштікті қайтарады. Python стандартты кітапханасы бойынша анықтамалыққа сәйкес: «үштік хост атынан, жалған атаудан және ipad-drlist тұрады. Мұндағы, хост аты – ip_address осы мекенжайына жауап беретін негізгі хост аты, жалған атау - (мүмкін бос) сол мекенжайға арналған хосттың альтернативті атауларының тізімі, ipaddrlist – бұл бір хостадағы бір интерфейс үшін IPv4 / v6 адрестерінің тізімі болып табылады [12].

Біздің бағдарламамыз үшін бізге тек бірінші үштік элементі, хост аты қажет. Егер шақыру кезінде ерекшеліктер пайда болса (басқа сөзбен айтқанда, хост аты нақты IP-мекенжаймен байланысы орнатылмасы), мен бұл элементтерді, кодта жұмыс істегенде бос орын ретінде басылу үшін бос орынмен толтырамын.

Сервердің және клиенттің IP-мекенжайларын зерттеу кезінде туындайтын сұрақтардың бірі: «IP-клиент қайда орналасқан?». Кейбір жағдайларда сервердің немесе клиенттердің өз орындарын жасыруға тырысатындығына қарамастан, көп жағдайда IP-мекенжайын жалпы географиялық аймақтармен салыстыру мүмкіндігі бар.

Осы нақты іздеуді орындау үшін мен үшінші тарап Python кітапханасын және мәліметтер қорын қолданамын. Қолданылатын кітапхана – pygeoip.

Pygeoiplibrary қызметтерін Python ортасында орнату үшін сіз pip-ті пайдалана аласыз. Pip – ең танымал Python дестелерін басқару жүйесі және Python-да жазылған үшінші тарап дестелерін орнату және басқару үшін қолданылады. pygeoip кітапханасы командалық жолынан орнатылады (3.11 және 3.12 суреттер). Pip басқару жүйесі алдын ала орнатылуы керек екенін ескеріңіз.

```
~$ pip install pygeoip
```

3.11 сурет – pygeoip кітапханасын Linux жүйесіне орнату пәрмені

```
C:\> pip install pygeoip_
```

3.12 сурет – pygeoip кітапханасын Windows жүйесіне орнату пәрмені

Pygeoip-ді орнатқаннан кейін, сіз MAXMIND жасаушы веб-сайтынан соңғы дерекқорды келесі мекенжайдан жүктеп алуыңыз керек: http://dev.maxmind.com/geoip/legacy/geolite/.

Осы тараудағы мысалдар үшін мен 3.13 суретте көрсетілгендей, елге арналған GoLite / Gzip-тің екілік нұсқасын қостым. Содан кейін мен ыңғайлық үшін geo.dat файлын негізгі каталогыма орналастырдым.

Е с к е р т у – Мен атауды geo.dat деп өзгерттім, өйткені дестеден шығару үрдісі GeoIP.dat файлын жасайды, сондықтан жаңартуларды жүктеген кезде мен жаңаларын және ескілерін қадағалай аламын.

Downloads

| | Download links | | | | |
|---|---|---|---|---|---|
| Database | Binary / gzip | Binary / xz | CSV / gzip | CSV / zip | CSV / xz |
| GeoLite Country | Download | Gzip only | Zip only | Download | Zip only |

3.13 сурет – MAXMIND GeoLite қорынан ел деректерінің екілік / Gzip нұсқасы

MAXMIND веб-сайтындағы нұсқауларды басшылыққа ала отырып, мен қажет болған жағдайда, төменде көрсетілгендей, үшінші тарап кітапханасын бағдарламаға енгізу кезінде осы бекітуді орнаттым (3.14 сурет).

```
# Бөгде кітапханалар
import pygeoip              # Geo IP Lookup
                            # командалық жолдан geoip орнату үшін: pip install pygeoip
                            # Бұл кітапхана Maxmind жасаған GeoLite деректерін қамтиды
                            # <a href="http://www.maxmind.com">http://www.maxmind.com</a>
```

3.14 сурет – Python-да pygeoip кітапханасын пайдалану

Енді pygeoip кітапханасы және оған байланысты geo.dat деректері орнатылған кезде, мен IP-мекенжайды белгілі бір елмен байланыстыру үшін оларды пайдалана аламын. Мен ел атауын шақыру және көрсету үшін қарапайым функцияны құрдым (3.15 сурет). Егер бірде-бір ел IP-мекенжайына сәйкес болмаса, онда бос жол қайтарылады.

```
#
# Елді іздеу
#

def GetCountry(ipAddr):

    # геолокация деректерін келесі сайттан жүктеу: http://dev.maxmind.com/geoip/legacy/geolite/

    gi = pygeoip.GeoIP('geo.dat')
    return gi.country_name_by_addr(ipAddr)

# GetCountry функциясын аяқтау
```

3.15 сурет – Ел атауын шақыру және көрсету үшін қарапайым функцияның коды

Хостты іздеу әдісі сияқты, мен COUNTRY_LOOKUP айнымалысы үшін ағымдағы күйіне байланысты True немесе False мәніне негізделген қосқышты жасаймын.

3.16 суретте көрсетілгендей, бұл әдіс пайдаланушының «C» параметрін енгізуімен жүзеге асырылады.

```
elif menuSelection == 'C':
    if COUNTRY_LOOKUP:
        COUNTRY_LOOKUP = False
    else:
        COUNTRY_LOOKUP = True
```

3.16 сурет – Мәзірден "C" шығыс нұсқасын таңдау коды

Содан кейін елдің атауын қосу орынды болатын кез келген код орнында COUNTRY_LOOKUP айнымалысы сұралады және тиісті түрде пайдаланылады (3.17 сурет).

```
# егер ел бойынша іздеу таңдалса,
# іздеуді орындау керек, әйтпесе ел өрісін бос қалдыру қажет

if COUNTRY_LOOKUP:
    countryName = GetCountry(serverIP)
else:
    countryName = ""
```

3.17 сурет – Ел атауын көру үшін COUNTRY_LOOKUP айнымалысының коды

Енді бастапқы орнату аяқталғаннан кейін біз бөлек талдау операцияларын жасай аламыз. Оларға келесілер жатады:

– жүктелген бақылау файлындағы барлық бақылауларды басып шығару;

– бақыланатын серверлер тізімін басып шығару;

– бақыланатын клиенттер тізімін басып шығару;

– бақыланатын серверді клиенттік байланыстарға басып шығару;

– бақылау гистограммасын басып шығару.

Барлық бақылауларды көрсету үшін әрбір сөздік жазбасын және мазмұнды басып шығаруды талап етеді. Бұл сервердің IP-мекенжайын, клиенттің IP-мекенжайын, сервер портының нөмірін, порттың түрін (TCP немесе UDP), сондай-ақ әрбір сағаттық кезең ішінде туындайтын осы бірегей комбинацияны бақылау санын қамтиды. Бұл операцияның орындалу әдісі 3.18 суретте көрсетілген.

```
# Сөздіктің мазмұнын басып шығару

def PrintOb(self):
    print >> OUT, ("\nIP Observations")
    print >> OUT, ("Unique Combinations:    ", str(len(self.Dictionary)))
    print >> OUT

    # Тақырыпты басып шығару

    print >> OUT, ('                                                    ')
    print >> OUT, ("|------------------------------------ Hourly Observations -------------------------------------")
    print >> OUT, ('%16s' % "Server"),
    print >> OUT, ('%16s' % "Client"),
    print >> OUT, ('%7s'  % "Port"),
    print >> OUT, ('%5s'  % "Type"),

    for i in range(0, 24):
        print >> OUT, ('  '),
        print >> OUT, ('%02d' % i),
    print >> OUT

   # Мазмұнды басып шығару
   for keys,values in self.Dictionary.items():

       print >> OUT, ('%16s' % keys[SERVER]),
       print >> OUT, ('%16s' % keys[CLIENT]),
       print >> OUT, ('%7s'  % str(keys[PORT])),
       print >> OUT, ('%5s'  % keys[TYPE]),

       for i in range(0,24):
           print >> OUT, ('%4s' % str(values[i])),
       print >> OUT
```

3.18 сурет – Сөздіктің мазмұнын басып шығару коды

Осы кодтың орындалуы Е қосымшасында көрсетілген келесі (қысқартылған) нәтижеге әкеледі.

Келесі талдау функциясы сөздікті таңдап, бақылаудағы серверлердің сұрыпталған тізімін ұсынады. Әрбір сервер үшін бұл сервермен қолдау көрсетілетін қызмет порттарының тізімі де көрсетіледі. Сонымен қатар, геолокация (яғни ел), хост аты және порттың сипаттамасы сияқты деректер пайдаланушы көрсеткен параметрлер негізінде енгізіледі. Бұл деректерді бақылау сөздігінен алу үшін әзірленген әдіс 3.19 суретте көрсетілген.

```
1   #
2   # PrintUniqueServer тізімі
3   #
4   # әрбір сервердің стандартты IP мекенжайын басып шығару әдісі
5   # Опцияларға кіреді: lookupHost және lookupCountry
6   # Егер таңдалған болса, олар тиісті іздеулерді орындап,
7   # алынған деректерді хабарлайтын болады
8   #
9   def PrintServers(self):
10
11      print >> OUT, ("\nUnique Server List\n")
12      print >> OUT, ('-----------------------------------------------------------------------------------------')
13      # Сөздіктен серверлер IP мекенжайларының
14      # "терілуін" құру
15
16      self.servers = set()
17      for keys,values in self.Dictionary.items():
18          self.servers.add(keys[SERVER])
19
20      # Тізімге түрлендіру және сұрыптау
21      # Бұл әдіс бірегей сұрыпталған тізімді қамтамасыз етеді
22
23      serverList = list(self.servers)
24      serverList.sort()
25
26      # Сұрыпталған тізімде сервердің әрбір IP мекенжайын өңдеу
27
28      for serverIP in serverList:
29
30          # егер ел бойынша іздеу таңдалса,
31          # іздеуді орындау керек, әйтпесе ел өрісін бос қалдыру қажет
32          if COUNTRY_LOOKUP:
33              countryName = GetCountry(serverIP)
34          else:
35              countryName = ""
36
37          # Желілік қате жағдайында Try / Except Loop орнату
38
39          try:
40              # егер шақырушы тарапынан хост атын іздеуді сұранысы
41              # түссе, онда іздеуді орындау қажет, әйтпесе өрісті бос қалдыру керек
42              if HOST_LOOKUP:
43                  hostName = socket.gethostbyaddr(serverIP)
44              else:
45                  hostName = ["", "", ""]
46          except:
47              hostName = ""
48              pass
49          # Қалыптасқан нәтижелерді басып шығару
50          print >> OUT,(' $15s ' % serverIP),
51          print >> OUT,(' $15s ' % countryName),
52          print >> OUT,(' $60s ' % hostName[HOST_NAME])
53
54          self.ports = set()
55
56          for keys,values in self.Dictionary.items():
57              if keys[SERVER] == serverIP:
58                  self.ports.add( (keys[PORT], keys[TYPE]) )
59
60          portList = list(self.ports)
61          portList.sort()
62
63          for port in portList:
64              print >> OUT,(' %27s ' % str(port[0])),
65              print >> OUT,(' %5s ' % port[1]),
66              print >> OUT,('%40s' % self.portDB.Lookup(port[0],port[1]))
67          print >> OUT, ('-----------------------------------------------------------------------------------------')
68
69      print >> OUT, ("\n\n")
70
71  # PrintUniqueServer тізімінің соңы
```

3.19 сурет – Бақыланатын серверлер тізімін ұсыну коды

Осы кодтың орындалуы келесі (қысқартылған) нәтижеге әкеледі (3.20 сурет):

```
Unique Server List

-----------------------------------------------------------------------------
         0.0.0.0
                          68     UDP                  Bootstrap Protocol Client
-----------------------------------------------------------------------------
     103.31.6.36         Australia
                          80     TCP                       World Wide Web HTTP
-----------------------------------------------------------------------------
 104.130.251.189     United States
                          80     TCP                       World Wide Web HTTP
-----------------------------------------------------------------------------
  104.130.53.116     United States
                          80     TCP                       World Wide Web HTTP
…
… Abbreviated
…
-----------------------------------------------------------------------------
     192.168.0.1
                          67     UDP                  Bootstrap Protocol Server
                        1027     UDP                                    Unknown
                        1900     UDP                                  UPnP SSDP
-----------------------------------------------------------------------------
    192.168.0.10
                          68     UDP                  Bootstrap Protocol Client
                         137     UDP                      NETBIOS Name Service
                        5353     UDP                                    Unknown
                        8612     UDP                                    Unknown
                       17500     UDP                                    Unknown
-----------------------------------------------------------------------------
   192.168.0.100
                       29808     UDP                                    Unknown
-----------------------------------------------------------------------------
    192.168.0.11
                        5353     UDP                                    Unknown
-----------------------------------------------------------------------------
    192.168.0.12
                          68     UDP                  Bootstrap Protocol Client
                         137     UDP                      NETBIOS Name Service
                         161     UDP                                      SNMP
                         427     UDP                          Server Location
                        3910     TCP                                    Unknown
                        8612     UDP                                    Unknown
                        9100     TCP                            HP JetDirect
                       43041     UDP                                    Unknown
                       43528     UDP                                    Unknown
-----------------------------------------------------------------------------
    192.168.0.13
                        5353     UDP                                    Unknown
-----------------------------------------------------------------------------
    192.168.0.14
                          68     UDP                  Bootstrap Protocol Client
                        5353     UDP                                    Unknown
-----------------------------------------------------------------------------
    192.168.0.15
                        8612     UDP                                    Unknown
-----------------------------------------------------------------------------
    192.168.0.19
                          68     UDP                  Bootstrap Protocol Client
                         123     UDP                      Network Time Protocol
                         137     UDP                      NETBIOS Name Service
                         138     UDP                  NETBIOS Datagram Service
                        5353     UDP                                    Unknown
                        8612     UDP                                    Unknown
                       16403     UDP                                    Unknown
                       17500     UDP                                    Unknown
-----------------------------------------------------------------------------
   192.168.0.255
                         137     UDP                      NETBIOS Name Service
                        1947     UDP                                   hlserver
                        8612     UDP                                    Unknown
-----------------------------------------------------------------------------
```

3.20 сурет – PrintServers функциясының орындалу нәтижесі

Бақыланатын клиенттердің тізімін алу және басып шығару бақыланатын серверлермен тәсілімен бірдей. Алынған деректер геолокация (яғни ел), хост аты және порттың сипаттамасы сияқты ақпаратты пайдаланушы көрсеткен параметрлер негізінде көрсетіледі. Төменде осы мәліметтерді бақылау сөздігінен алу әдісі келтірілген (3.21 сурет). Сіз: неге клиенттің порты көрсетілмеген? Деген сұрақты қоя аласыз.

Клиент портының алынып тасталуы (бұл әдеттегідей және бізге қажет емес) біздің сөздіктің көлемін айтарлықтай азайтады. Егер біз уақытша порттарды сөздік кілтіне енгізсек, серверге әр қосылыс бірегей болатын еді.

```python
#
# Клиенттердің бірегей тізімін басып шығару
#
# әр IP клиентін басып шығару тәсілі.
# Опциялар: lookupHost және lookupCountry
# Егер таңдалған болса, олар тиісті іздеулерді орындап,
# алынған деректерді хабарлайтын болады

    def PrintClients(self):

        print >> OUT,("\nUnique Client List\n")

        self.clients = set()
        for keys,values in self.Dictionary.items():
            self.clients.add(keys[1])

        clientList = list(self.clients)
        clientList.sort()

        # Сұрыпталған тізімде сервердің әрбір IP мекенжайын өңдеу

    for clientIP in clientList:

        # егер ел бойынша іздеу таңдалса,
        # іздеуді орындау қажет, әйтпесе ел өрісін бос қалдыру керек
        if COUNTRY_LOOKUP:
            countryName = GetCountry(clientIP)
        else:
            countryName = ""

        # Желілік қате жағдайында Try / Except Loop орнату.

        try:
            # егер шақырушы тарапынан хост атын іздеуді сұранысы
            # түссе, іздеуді орындау қажет, әйтпесе өрісті бос қалдыру керек
            if HOST_LOOKUP:
                hostName = socket.gethostbyaddr(clientIP)
            else:
                hostName = ["", "", ""]
        except:
            hostName = ["", "", ""]
        pass

    # Қалыптасқан нәтижелерді басып шығару

    print >> OUT,(' %15s ' % clientIP),
    print >> OUT,(' %15s ' % countryName),
    print >> OUT,(' %60s ' % hostName[HOST_NAME])

print >> OUT,("\nEnd Print Client List\n")

    # PrintUniqueClient тізімінің соңы
```

3.21 сурет – Бақыланатын клиенттер тізімін ұсыну коды

Осы кодтың орындалуы келесі (қысқартылған) нәтижеге әкеледі (3.22 сурет):

```
Unique Client List

   118.98.104.21         Indonesia                    21.subnet118-98-104.astinet.telkom.net.id
       127.0.0.1                                                          Lenovo-UpStairs
  141.212.122.34     United States                          researchscan289.eecs.umich.edu
  141.212.122.39     United States                          researchscan294.eecs.umich.edu
    169.229.3.91     United States                         researchscan1.EECS.Berkeley.EDU
…
… Abbreviated
…
   37.247.36.119       Netherlands
    41.218.92.89           Namibia                   westair-schneider-int.cust.na.afrisp.net
      46.4.7.155           Germany                                        plesksolutions.com
    50.17.79.135     United States                   ec2-50-17-79-135.compute-1.amazonaws.com

========== P2NMAP Analysis Menu ==========
```

3.22 сурет – PrintClients функциясының орындалу нәтижесі

Бақылау нәтижелерін көрудің тағы бір жолы - әрбір серверді және осы сервермен жасалған барлық клиенттік байланыстарды көрсету. Бұл сервер /

клиент қосылыстарының толық тізімін қамтамасыз етеді. Бұл әдіс сәл күрделі болып табылады, өйткені сөздік алдымен бақыланатын серверлердің тізімін жасау керек, содан кейін осы серверге кез келген порт арқылы қосылған клиенттер тізімін жасау керек. Бұл бөлшектерді бақылау сөздігінен алу үшін әзірленген әдіс төменде көрсетілген (3.23 сурет) .

```
1   #
2   # Серверлердің толық тізімін басып шығару
3   #
4   # Стандартты түрде басып шығару әдісі
5   # Бірегей сервер / клиент өзара әрекеті
6   #
7
8   def PrintServerDetails(self):
9
10      # Сөздіктен серверлер IP мекенжайларының
11      # "терілуін" құру
12
13      self.servers = set()
14
15      for keys,values in self.Dictionary.items():
16          self.servers.add(keys[SERVER])
17          # Тізімге түрлендіру және сұрыптау
18          # Бұл әдіс бірегей сұрыпталған тізімді қамтамасыз етеді
19
20      # Енді бірегей серверлердің сұрыпталған тізімін жасау керек
21      serverList = list(self.servers)
22      serverList.sort()
23
24      # Енді серверге сәйкес келетін барлық қосылымдарды табу
25      # және қосылым туралы ақпарат беру үшін
26      # серверлер тізімі бойынша іздеуді орындау қажет.
27
28      for serverIP in serverList:
29
30          # егер ел бойынша іздеу таңдалса,
31          # іздеуді орындау қажет, әйтпесе ел өрісін бос қалдыру керек
32          if COUNTRY_LOOKUP:
33              countryName = GetCountry(serverIP)
34          else:
35              countryName = ""
36
37          # Желілік қате жағдайында Try / Except Loop орнату.
38
39          try:
40              # егер шақырушы тарапынан хост атын іздеуді сұранысы
41              # түссе, іздеуді орындау қажет, әйтпесе өрісті бос қалдыру керек
42              if HOST_LOOKUP:
43                  hostName = socket.gethostbyaddr(serverIP)
44              else:
45                  hostName = ["", "", ""]
46          except:
47              hostName = ""
48              continue
49
50  # Қалыптасқан нәтижелерді басып шығару
51  print >> OUT,("\n==========================================================================================")
52  print >> OUT,("Server: "),
53  print >> OUT,(' %15s ' % serverIP),
54  print >> OUT,(' %15s ' % countryName),
55  print >> OUT,(' %60s ' % hostName[HOST_NAME])
56  print >> OUT,("==========================================================================================")
57  print >> OUT,('%16s' % "Client"),
58  print >> OUT,('%7s'  % "Port"),
59  print >> OUT,('%40s' % "Port Description"),
60  print >> OUT,('%5s'  % "Type"),
61  print >> OUT
62
63  for keys,values in self.Dictionary.items():
64
65      # Егер сервер сәйкестендірілсе
66      # мәліметтерді басып шығару керек:
67
68      if keys[SERVER] == serverIP:
69          print >> OUT, ('%16s' % keys[CLIENT]),
70          print >> OUT, ('%7s'  % str(keys[PORT])),
71          print >> OUT, ('%40s' % self.portOB.Lookup(keys[PORT],keys[TYPE])),
72          print >> OUT, ('%5s'  % keys[TYPE])
73
74  # PrintUniqueServer тізімінің соңы
```

3.23 сурет – Бақыланатын серверлер мен оларға қосылған клиенттер тізімін ұсыну коды

Осы кодтың орындалуы келесі (қысқартылған) нәтижеге әкеледі (3.24 сурет):

```
Unique Server Client Connection List
-----------------------------------------------------------------------------

==============================================================
Server:           0.0.0.0
==============================================================
          Client    Port                          Port Description  Type
 255.255.255.255      68                Bootstrap Protocol Client   UDP

==============================================================
Server:       103.31.6.36         Australia
==============================================================
          Client    Port                          Port Description  Type
    192.168.0.10      80                       World Wide Web HTTP   TCP

==============================================================
Server:   104.130.251.189     United States
==============================================================
          Client    Port                          Port Description  Type
    192.168.0.10      80                       World Wide Web HTTP   TCP

==============================================================
Server:    104.130.53.116     United States
==============================================================
          Client    Port                          Port Description  Type
    192.168.0.10      80                       World Wide Web HTTP   TCP
…
… Abbreviated Output
…
==============================================================
Server:       96.6.113.90     United States

==============================================================
          Client    Port                          Port Description  Type
    192.168.0.10      80                       World Wide Web HTTP   TCP

==============================================================
Server:     98.137.170.33     United States
==============================================================
          Client    Port                          Port Description  Type
    192.168.0.10      80                       World Wide Web HTTP   TCP

==============================================================
Server:    98.139.225.168     United States
==============================================================
          Client    Port                          Port Description  Type
    192.168.0.10     443                HTTP protocol over TLS/SSL   TCP
    192.168.0.10      80                       World Wide Web HTTP   TCP

==============================================================
Server:     98.139.225.35     United States
==============================================================
          Client    Port                          Port Description  Type
    192.168.0.15      80                       World Wide Web HTTP   TCP
```

3.24 сурет – PrintServerDetails функциясының орындалу нәтижесі

Соңғы әдіс сервер мен клиенттің әрбір өзара қосылысы үшін гистограмманы құрайды. Құрылған гистограмма 24 сағатқа есептелген. Егер P2NMAP-Capture бағдарламасы бірнеше күн бойы іске қосылса, әрбір сағат үшін белсенділік шоғырланған болады. Бұл күн ішінде немесе бірнеше рет немесе бір рет пайда болатын әрекеттерді жылдам байқауға мүмкіндік береді. Бұл белсенділік жүрек соғысына немесе маяк сигналына ұқсас болуы мүмкін. Төменде осы мәліметтерді бақылау сөздігінен алу әдісі келтірілген (3.25 сурет).

```python
#
# Алынған трафиктің гистограммасын шығару
#
# Әрбір жазу үшін гистограмманы
# басып шығару әдісі
#
#

def PrintHistogram(self):

    # Сөздіктен серверлер IP мекенжайларының
    # "терілуін" құру

    print >> OUT, ("\nHourly Histogram\n")

    self.servers = set()

    for keys,values in self.Dictionary.items():
        self.servers.add(keys[SERVER])
        # Тізімге түрлендіру және сұрыптау
        # Бұл әдіс бірегей сұрыпталған тізімді қамтамасыз етеді

    # Енді бірегей серверлердің сұрыпталған тізімін жасау керек
    serverList = list(self.servers)
    serverList.sort()

    # Енді серверге сәйкес келетін барлық қосылымдарды табу
    # және қосылым туралы ақпарат беру үшін
    # серверлер тізімі бойынша іздеуді орындау қажет.

    for serverIP in serverList:

        # егер ел бойынша іздеу таңдалса,
        # іздеуді орындау қажет, әйтпесе ел өрісін бос қалдыру керек

        if COUNTRY_LOOKUP:
            countryName = GetCountry(serverIP)
        else:
            countryName = ""

        # Желілік қате жағдайында Try / Except Loop орнату.

        try:
            # егер шақырушы тарапынан хост атын іздеуді сұранысы
            # түссе, іздеуді орындау қажет, әйтпесе өрісті бос қалдыру керек
            if HOST_LOOKUP:
                hostName = socket.gethostbyaddr(serverIP)
            else:
                hostName = ["", "", ""]
        except:
            hostName = ["", "", ""]
            continue

        # Қалыптасқан нәтижелерді басып шығару
        print >> OUT, ("\n==============================================================================")
        print >> OUT, ("Server: "),
        print >> OUT, (' %15s ' % serverIP),
        print >> OUT, (' %15s ' % countryName),
        print >> OUT, (' %60s ' % hostName[HOST_NAME])
        print >> OUT, ("==============================================================================")

        for keys,values in self.Dictionary.items():

            # Егер сервер сәйкестендірілсе,
            # гистограмманы басып шығару керек

            if keys[SERVER] == serverIP:
                if keys[SERVER] == serverIP:
                    print >> OUT, ('%16s' % "Client"),
                    print >> OUT, ('%7s'  % "Port"),
                    print >> OUT, ('%40s' % "Port Description"),
                    print >> OUT, ('%5s'  % "Type"),
                    print >> OUT
                    print >> OUT, ('%16s' % keys[CLIENT]),
                    print >> OUT, ('%7s'  % str(keys[PORT])),
                    print >> OUT, ('%40s' % self.portOB.Lookup(keys[PORT], keys[TYPE])),
                    print >> OUT, ('%5s'  % keys[TYPE])
                    print >> OUT
                    print >> OUT, ("HR")
                    self.Histogram(values)
                    print >> OUT
                    print >> OUT,("End Print Histogram\n")

            # Histogram шығысының соңы
```

3.25 сурет – Алынған трафикке гистограмманы шығару коды

Осы кодтың орындалуы Ж қосымшасында көрсетілген келесі (қысқартылған) нәтижеге әкеледі.

P2NMAP-Analysis.py бағдарламасының толық аяқталған коды И қосымшасында көрсетілген. Бүкіл бағдарлама бір Python файлы болып табылады және орындалу үшін аргументтерді қажет етпейді. Дегенмен, бағдарламаны іске асыру үшін бірнеше талаптар орындалу қажет:

а) geo.dat файлы бастапқы каталогқа қосылу қажет;

б) pygeoip кітапханасы төмендегі пәрмендерді пайдалану арқылы орнатылуы тиіс. Linux жүйесі үшін:

```
pip install pygeoip
```

Windows жүйесі үшін:

```
C: \> pip install pygeoip
```

Енді бағдарлама P2NMAP-Analysis.py іске қосуға дайын. Бағдарламаны орындау үшін төмендегі пәрменді енгізу қажет. Linux жүйесі үшін:

```
python P2NMAP-Analysis.py
```

Ал Windows жүйесі үшін:

```
C: \> python P2NMAP-Analysis.py
```

Бұл бағдарламаның бас мәзірін көрсетеді. Енді сізге әртүрлі жұмыс режимдерімен және талдау функцияларымен тәжірибе жасауға болады.

Бұл тарауда біз P2NMAP-Capture бағдарламасын пайдаланып жасалған .ipdict файлын қолдандық. Бұл файлда IP-бақылау сөздігінің толық тізімі бар. Осы бақылау сөздігін пайдаланып, ipObservationsDictionary класында бірнеше негізгі әдістерді құрдық. Бұл әдістер келесі үрдістерді орындайды: IP-бақылауларының толық сөздігін басып шығару, серверлер мен клиенттердің толық тізімін құру, сервер мен клиент арасындағы қосылыстардың тізімін құру және бақылау деректерінің толық гистограммасын шығару. Сонымен қатар, біз бақыланатын мәліметтерден негізгі ақпаратты алдық, оның ішінде IP-мекен-жайына негізделген хост атауы, пайдаланылған сервер порттарына негізделген порттың толық сипаттамалары және көптеген бақыланатын серверлер мен клиенттердің географиялық орны. Соңында біз жасалған талдау әдістерімен тәжірибе жасауға болатын қарапайым мәзірге негізделген интерфейсті жасадық.

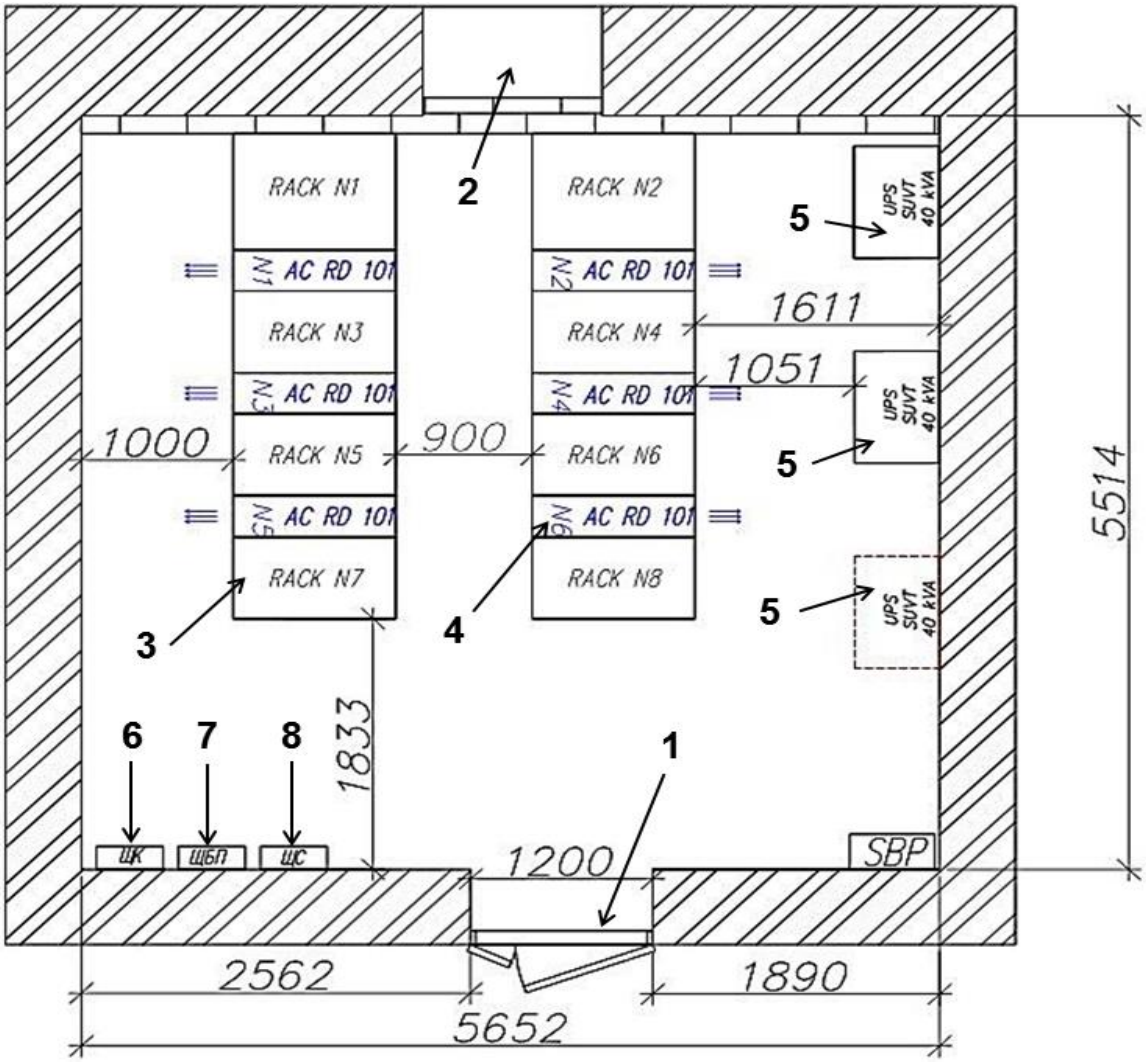
## 4 Өмір тіршілігі қауіпсіздігі

### 4.1 Кәсіпорындағы еңбек жағдайларын талдау

Бұл жұмыстың бағдарламасы жоғары деңгейлі Python тілінде жазылған. Жобаланған бағдарлама сервер құрылғысының бағдарламалық жасақтамасының ішіне орнатылады. Көптеген ұйымдар арнайы серверлік жабдықтарды мамандандырылған бөлмелерге орналастыратындығына байланысты, бұл бөлім кәсіпорынның деректерді өңдеу орталығында қауіпсіздік шараларын сақтаудың негізгі қағидаларын қарастырады.

Әрі қарай, мен тіршілік қауіпсіздігінің барлық негізгі қағидалары сақталатын серверлік бөлмесін мысалға келтіремін.

Серверлік бөлме кәсіпорынның жертөле қабатында орналасқан және оның ауданы 31,7 м². Жобаланатын ДӨО-ның ішінде орнатылатын жабдықтардың орналастыру жоспары және нақты өлшемдері (4.1 суретте) көрсетілген.




4.1 сурет – Серверлік бөлменің жоспары

Шартты белгілер:

1 – есік; 2 – терезе; 3 – телекоммуникациялық шкаф; 4 – салқындату жүйесі; 5 – үздіксіз қоректендіру көзі; 6 – пәтер қалқаны; 7 – кепілді қоректендіру қалқаны; 8 – күштік қалқаны.

Негізгі еденнен төбеге дейінгі биіктік 2385 мм, ал фальшполдың биіктігі 200 мм. Бөлме төбесінің жалпы биіктігі 2,6 м. Бөлмеде бір терезе ойығы бар, ол кірпішпен бітелген. Орнатылған серверлік және желілік құрылғылардың жиынтық қуаты 40 кВт құрайды.

ДӨО 8 телекоммуникациялық шкафтармен, серверлік және коммуникациялық жабдықтармен, қоректендіруді тарату блоктарымен, мониторинг жүйесімен, салқындату жүйесімен, тарату қалқандарымен және дизельді генераторлық қондырғымен жабдықталған.

Серверлік бөлмелердің технологиялық және санитарлық-техникалық қабылдағыштары, ЭҚҚ (1 тарау, 2 параграф, 1.2.17 п.) [12] классификациясы бойынша, бірінші санаттағы арнайы топқа жатады. Олардың электрмен жабдықтаудың үзілісі күрделі технологиялық үрдістің бұзылуына әкеп соғады.

ДӨО-ның электрмен жабдықтаудың негізгі көзі ретінде 10/0,4 кВ трансформаторлық қосалқы станция болып табылады.

ДӨО электр қабылдағыштарын қосуға арналған енгізуші-тарату құрылғылары бөлмеде орнатылған тарату қалқандары (ПҚ, КҚК, КҚ) болып табылады.

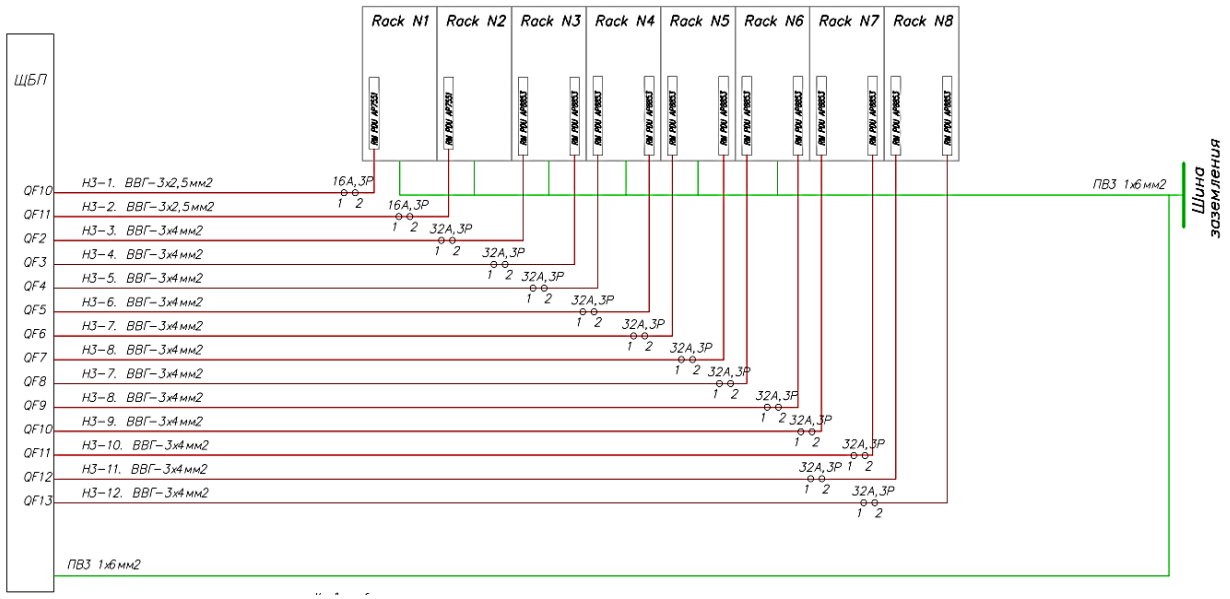
0,4 кВ қоректендіргіш, тарату және жерлендіру желісі торлы кәбілдік лотоктарда төселетін мысты талшығы бар кәбілдермен және сымдармен орындалған.

ЭҚҚ-на (7 тарау, 1 параграф, 7.1.150 п.) [12] сәйкес, оқшаулаудың бұзылуы салдарынан, кернеуде болуы мүмкін электр қондырғыларының, электр жабдықтарының корпустары мен телекоммуникациялық шкафтардың барлық ток өткізбейтін бөліктері, қызмет көрсетуші персоналдың қауіпсіздігін қамтамасыз ету үшін негізгі жерлендіру жүйесіне қосылған.

ДӨО-ның технологиялық электр қондырғылары үшін TN-S жерлендіру жүйесі қабылданған, онда нөлдік қорғаныс және нөлдік жұмыс өткізгіштері өзара бөлінген.

Потенциалдарды теңестірудің негізгі жүйесі ғимараттың қолданыстағы жерлендіру контурымен қосылған. Сонымен қатар, серверлік бөлмеде негізгі жерлендіру шинасына қосылған қосымша потенциалдарды теңестіру жүйесі ұйымдастырылған. Ол да ғимараттың жерлендіру контурына жалғанған. Қосымша потенциалдарды теңестіру жүйесі барлық қол жетімді ашық өткізгіш бөліктерді қосады.

ДӨО технологиялық электр қондырғыларын қоректендіру және жерлендіру жүйесінің схемасы (4.2 суретте) көрсетілген.




4.2 сурет – Телекоммуникациялық шкафтарды қоректендіру және жерлендіру схемасы

Деректер орталығы ұзақ уақыт бойы жоғары жүктеме жағдайында үздіксіз жұмыс істеуге арналған. Бұл бөлмеде орналастырылған жабдықтардың тығыздығы өте жоғары. Телекоммуникациялық құрылғылар жұмыс кезінде қызатындықтан, олар өрттің ықтимал көзі болып табылады. Жабдықтардың корпусында және кабельдік сымдардың оқшаулауында әр түрлі пластмассаның болуы өрт қауіптілігін одан әрі арттыруға ықпал етеді. Нәтижесінде, тіпті кішігірім техникалық ақау ауыр зардаптары бар ірі жануға әкеп соқтыруы мүмкін.

Өрт қауіпсіздігі шараларын жүзеге асыру кезінде өндірістің өрт қауіптілігін бағалау үлкен маңызға ие. Жұмыстағы өрттен туындаған жағдайлар ғимаратта қандай заттар қолданылатынына немесе сақталуына байланысты [13]. «Өнеркәсіптік кәсіпорындарының өндірістік ғимараттарын» [14] жобалау нормаларына сәйкес, өрт қауіптілігі бойынша ДӨО-ның бөлмесі Д категориясына жатады.

Төтенше жағдайлардан қорғаудың негізгі әдістерінің бірі – қызметкерлер мен объектілерді қауіпті аймақтарынан уақытылы эвакуациялау және орналастыру болып саналады.

Ғимараттар мен құрылыстарды жобалау кезінде маңызды міндеттердің бірі ықтимал төтенше жағдай кезінде адам қозғалысы үшін барынша қолайлы жағдайлар жасау және оның қауіпсіздігін қамтамасыз ету болып табылады. Мәжбүрлі қозғалыс қауіптің себебінен (өрт, авария және т.б.) бөлмеден немесе ғимараттан кету қажеттілігімен байланысты пайда болады.

Өрт шыққан кезде әртүрлі факторлар мәжбүрлі эвакуация кезіндегі адамның денсаулығына немесе өміріне қауіп төндіреді. Сондықтан ғимаратты жобалау кезінде эвакуациялау үрдісі қажетті уақытта аяқталуы мүмкін болатын шаралар қабылданады.

Қауіп-қатерге байланысты адамдардың қозғалу үрдісі шығу жағына қарай бір бағытта инстинктивті түрде басталады. Бұл адамдар ағынының белгілі бір тығыздығымен тез толтырылуына әкеледі. Қозғалыс жылдамдығы ағынының тығыздығының ұлғаюымен төмендейді.

Егер адамдарды бөлмелерден немесе ғимараттардан эвакуациялау ұзақтығы жалпы өрт ұзақтығынан аз болған жағдайда мәжбүрлі эвакуациялаудың қауіпсіздігі жүзеге асырылады [16].

## 4.2 Есептеу бөлімі

### 4.2.1 Жасанды қорғаныстық жерлендіруді есептеу

Есептеу жұмысы әдістемелік нұсқаулық [17] бойынша жасалды. Диаметрі 0,03 м, ұзындығы 3 м болатын және ені 0,02 м болат жолақпен байланысқан құбырларға жалғанған телекоммуникациялық шкафтардың жасанды қорғаныстық жерлендіруін есептеу қажет. Қосқыш контурлық жолақтың есептік тереңдігі 0,6 м.

Жасанды жерлендіруді есептеу кезінде алдымен бір тік электродтың электрлік кедергісі ($R_{\text{Т}}$) мынадай формула бойынша анықталады:

$$R_{\text{Т}} = \frac{0{,}16\cdot\rho}{l}\cdot\left[\ln\frac{2\cdot l}{d} + 0{,}5\cdot\ln\frac{4\cdot(h_0+0{,}5\cdot l)+l}{4\cdot(h_0+0{,}5\cdot l)-l}\right], \quad (4.1)$$

мұндағы, $\rho$ – топырақтың меншікті кедергісі, Ом · м;

$l, d$ –құбырлардың ұзындығы және диаметрі (м);

$h_0$ – жолақтың орналасу тереңдігі, м.

Саздақ типті топырақтың есептік үлестік кедергісі ретінде 100 Ом · м мәні қабылданады. Жеке тік электродтың электрлік кедергісін есептейік:

$$R_{\text{Т}} = \frac{0{,}16\cdot 100}{3}\cdot\left[\ln\frac{2\cdot 3}{0{,}03} + 0{,}5\cdot\ln\frac{4\cdot(0{,}6+0{,}5\cdot 3)+3}{4\cdot(0{,}6+0{,}5\cdot 3)-3}\right] = 30{,}25 \text{ Ом}.$$

Контурлық жерлендіру құрылғысында тік электродтарды қосатын көлденең электродтың жиынтық ұзындығы ($l_{\text{К}}$), келесі формуламен есептеледі:

$$l_{\text{К}} = a\cdot(n-1), \quad (4.2)$$

мұндағы, $n$ – тік электродтардың саны, $n = 20$;

$a$ – электродтар арасындағы арақашықтық, м.

Тік электродтар арасындағы қашықтық құбырлы электродтың ұзындығына тең, яғни $l = a = 3$ м болып қабылданады.

Көлденең электродтың жиынтық ұзындығын анықтаймыз:

$$l_{\text{К}} = 3\cdot(20-1) = 57 \text{ м}.$$

Осыдан кейін осы электродтың электр кедергісі есептеледі ($R_К$):

$$R_К = \frac{0,16 \cdot \rho}{l_К} \cdot \ln \frac{{l_К}^2}{b \cdot h_0}, \quad (4.3)$$

мұндағы, $b$ – жолақтың ені, м.

Көлденең электродтың электр кедергісін есептейміз:

$$R_К = \frac{0,16 \cdot 100}{57} \cdot \ln \frac{57^2}{0,02 \cdot 0,6} = \frac{16}{57} \cdot \ln 270750 = 3,51 \text{ Ом} .$$

Жерге тұйықтау құрылғысының ток қашықтығына есептік электр кедергісі есептеледі ($R$):

$$R = \frac{R_Т \cdot R_К}{R_Т \cdot \eta_К + R_К \cdot \eta_Т \cdot n}, \quad (4.4)$$

мұндағы, $\eta_Т$, $\eta_К$ – өзекшелерді мен жолақтарды экрандау коэффициенттері.

Содан кейін есептік кедергі ($R$) рұқсат етілген жерлендіру кедергісімен $R_{РҰҚ}$ салыстырылады. Рұқсат етілген кедергінің ($R_{РҰҚ}$) шамасы 4 Ом.

$\eta_Т$ және $\eta_К$ мәндері (4.1 кесте) бойынша берілген шарттар үшін анықталады.

4.1 кесте – $a = l$ кезіндегі электродтар санына $\eta_В$ және $\eta_Г$ шамаларының тәуелділігі

| | |
|---|---|
| Тік электродтардың саны n | 20 |
| $\eta_Т$ мәні | 0,27 |
| $\eta_К$ мәні | 0,47 |

Жерге тұйықтау құрылғысының есептік электр кедергісін ток қашықтығына анықтаймыз:

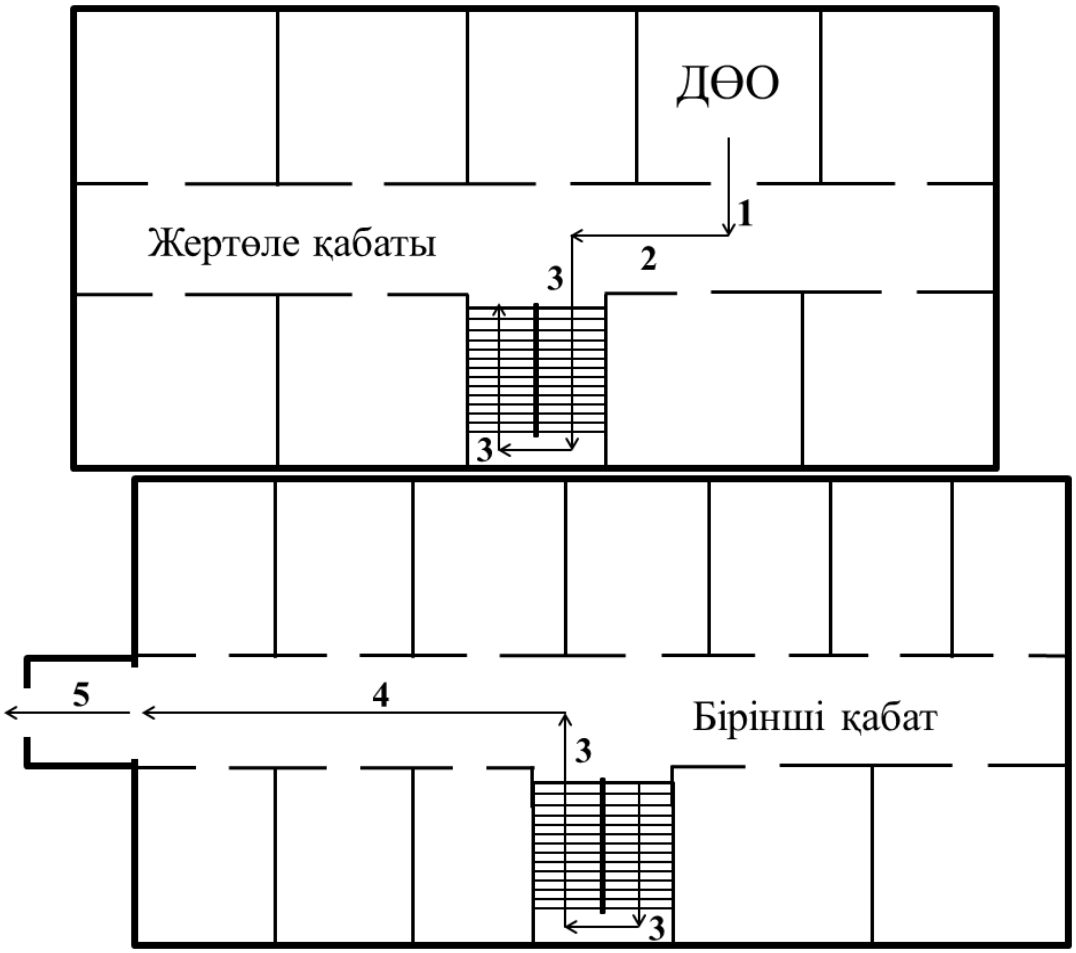
$$R = \frac{30,25 \cdot 3,51}{30,25 \cdot 0,27 + 3,51 \cdot 0,47 \cdot 20} = \frac{7865}{3049} = 2,57 \text{ Ом} .$$

Есептелген жерлендіру құрылғысының кедергісі жерге тұйықтау құрылғыларына арналған ЭҚҚ (2 бөлім, 17 параграф, 2.17.292 п.) [12] талаптарын қанағаттандырады және 4 Ом аспайды.

4.2.2 Эвакуация уақытын есептеу. Есептеу жұмысы әдістемелік нұсқаулық [16] бойынша жасалды. Ғимаратта өрт болған жағдайда кәсіпорын қызметкерлерін сервер бөлмесінен эвакуациялау уақытын анықтау қажет. Панель түріндегі әкімшілік ғимарат автоматты өрт дабылы және ескерту жүйесімен жабдықталған. Ғимарат бір қабатты жертөлесі бар, өлшемі

18х36 м, оның дәліздерінің ені 3 м. Көлемі 59 м$^3$ болатын серверлік бөлмесі жертөле қабатында орналасқан. Кестелік мәліметтерге сәйкес (А, В қосымшасы) [16], ДӨО бөлмесі D категорияға және II отқа төзімділік дәрежесіне жатады. Баспалдақтардың ені 1,8 м және ұзындығы 12 м құрайды Бөлмеде 5 адам жұмыс істейді. Барлығы 46 адам жертөле қабатында жұмыс атқарады. Бірінші қабатта 76 адам жұмыс істейді. ДӨО бөлмесінен эвакуациялау схемасы (4.3 суретте) көрсетілген.

Адамдарды ғимараттың сыртына эвакуациялаудың ұзақтығы эвакуация жолдарының ұзындығымен және есіктер мен баспалдақтардың өткізу қабілетімен анықталады. Есеп эвакуация жолдарының бойында ағындардың тығыздығы бірқалыпты және максималды мәндерге жететін жағдайлар үшін жүргізіледі.




4.3 сурет – Деректерді өңдеу орталығынан эвакуациялау схемасы

Адамдарды эвакуациялаудың есептік уақытын ($t_р$) жолдың жекелеген учаскелері бойынша адам ағынының қозғалыс уақытының сомасы ретінде айқындаған жөн:

$$t_р = t_{н.э.} + t_1 + t_2 + t_3 + \ldots + t_n, \quad (4.5)$$

мұндағы, $t_{н.э.}$ – эвакуацияны бастаудың кідірту уақыты;

$t_1$ – бірінші учаскедегі адам ағынының қозғалыс уақыты, мин;

$t_2,\ t_3, \ldots t_n$ – бірінші кезеңнен кейінгі келесі учаскелердегі адам ағынының қозғалыс уақыты, мин.

Өрт пайда болған кезде адамдарға жоғары температура, ауадағы оттегі концентрациясының төмендеуі және ғимараттардың түтінінен көрінудің жоғалуы қауіп төндіреді.

Өрт кезінде критикалық температура мен оттегінің концентрациясына жететін уақыт өрттің шектік ұзақтығы деп аталады.

Өндірістік ғимараттардағы температура бойынша өрттің шекті ұзақтығы мынадай формула бойынша анықталады:

$$\tau_{ш.ұ.} = \sqrt[3]{\frac{W_{бөл} \cdot c \cdot (t_ш - t_б)}{(1-\varphi) \cdot \pi \cdot Q \cdot n \cdot V^2}}, \qquad (4.6)$$

мұндағы, $W_{бөл}$ – қарастырылатын ғимараттағы немесе бөлмедегі ауа көлемі, м$^3$;

$c$ – газдың изобарлы меншікті жылу сыйымдылығы, кДж/кг-град;

$t_ш$ – адам үшін шектік температура, 70°С тең;

$t_б$ – бастапқы ауа температурасы, 20°С тең;

$\varphi$ – құрылыстарды және айналадағы заттарды жылытуға байланысты жылу шығынын сипаттайтын коэффициент, орташа есеппен 0,5-ке тең болады;

$Q$ – заттардың жану жылуы, кДж/кг;

$n$ – жану жылдамдығы, кг/м2-мин;

$V$ – жанатын заттардың беті бойынша оттың таралуының сызықтық жылдамдығы, м/мин.

Қарастырылып отырған бөлмедегі ауа көлемі бөлменің геометриялық көлемі мен жабдықтың немесе ішіндегі заттардың көлемі арасындағы айырмашылыққа сәйкес келеді. Егер бос көлемді есептеу мүмкін болмаса, оны геометриялық көлемнің 80% -на тең деп қабылдауға рұқсат етіледі.

760 мм. с. б. атмосфералық қысым кезіндегі құрғақ ауаның меншікті жылу сыйымдылығы, [18] кесте мәліметтеріне сәйкес, 0-ден 60°С -қа дейінгі температурада 1005 кДж/кг-град және 60-тан 120°С -қа дейінгі температурада 1009 кДж/кг-град құрайды.

Есептеу үшін ағаштың жанудың орташа жылдамдығы $n$ және жану жылуы $Q$ қолданылады. Кестелік мәліметтерге сәйкес (В қосымшасы) [16], бұл мәндер 14 кг/м2-мин және 13800 кДж/кг құрайды. Ағаш бетіндегі от таралуының сызықтық жылдамдығы, кестелік деректерге сәйкес (Г қосымшасы) [16], 0,36 м/мин тең.

Бөлмедегі жабдықтарды есепке ала отырып, температура бойынша өрттің шекті ұзақтығын есептейік:

$$\tau_{\text{ш.ұ.}} = \sqrt[3]{\frac{59{,}4 \cdot 1009 \cdot (70-20)}{(1-0{,}5) \cdot 3{,}14 \cdot 13800 \cdot 14 \cdot 0{,}36^2}} = 4{,}24 \text{ мин .}$$

Бөлме ауасындағы оттегі мөлшерінің азайына байланысты өрттің шектік ұзақтығы келесі формула бойынша анықталады:

$$\tau_{\text{ш.ұ.}}^{O_2} = \sqrt[3]{\frac{(0{,}01)^{-1} \cdot W_{\text{бөл}}}{\pi \cdot n \cdot W_{O_2} \cdot V^2}}, \qquad (4.7)$$

мұндағы, $W_{O_2}$ – 1 кг жаңғыш заттардың жануына оттегінің шығыны, м/кг, теориялық есепке сәйкес 4,76 м$^3$ тең [18].

$W_{O_2}$ біле отырып, өрттің шектік ұзақтығын есептейміз:

$$\tau_{\text{ш.ұ.}}^{O_2} = \sqrt[3]{\frac{(0{,}01)^{-1} \cdot 59{,}4}{3{,}14 \cdot 14 \cdot 4{,}76 \cdot 0{,}36^2}} = 6 \text{ мин .}$$

Есептеу нәтижесінде алынған өрттің шекті ұзақтығы мәндерінің ең азы таңдалады. Эвакуацияның рұқсат етілген ұзақтығы келесі формула арқылы анықталады:

$$\tau_{\text{РҰҚ}} = m \cdot \tau_{\text{ш.ұ.}}, \qquad (4.8)$$

мұндағы, $\tau_{\text{ш.ұ.}}$ – сәйкесінше эвакуацияның рұқсат етілген ұзақтығы және эвакуация кезінде өрттің шектік ұзақтығы, мин;

$m$ – өндірістік орындағы жанғыш заттардың қасиеттеріне байланысты қауіпсіздік коэффициенті.

$m$ коэффициентінің мәнін қарастырылып отырған ғимараттың өртке қарсы қорғаныс құралдарының сенімділік дәрежесіне байланысты орнату ұсынылады. Өндірістік ғимараттарда автоматты сөндіру және өрт туралы хабарлау құралдары болған кезде қауіпсіздік коэффициенті 2-ге тең [16].

Температура бойынша өрттің минималды ұзақтығы – 4,24 минут. Сонда осы бөлме үшін эвакуацияның рұқсат етілген ұзақтығы келесіге тең:

$$\tau_{\text{РҰҚ}} = 2 \cdot 4{,}24 = 8{,}48 \text{ мин .}$$

Кестелік мәліметтерге сәйкес (Д қосымшасы) [16], ғимаратта автоматты өрт дабылы мен ескерту жүйесі болғандықтан, эвакуацияны бастаудың кешігу уақыты 4,1 минут болып алынады.

Есептеу кезінде адам ағыны қозғалысының барлық жолы ұзындығы ($l$) және ені ($b$) учаскелерге (өткел, дәліз, есік ойығы, баспалдақ маршы, тамбур) бөлінеді. Бастапқы учаскелер жұмыс орындары мен жабдықтар арасындағы өткелдер болып табылады.

Бірінші учаске бойынша адамдардың қозғалыс уақытын анықтау үшін, 5,652х5,514 м бөлме габариттік өлшемдерін ескере отырып, онда адам ағынының қозғалыс тығыздығы мынадай формула бойынша анықталады:

$$D_1 = \frac{N_1 \cdot f}{L_1 \cdot B_1}, \quad (4.9)$$

мұндағы, $N_1$ – бірінші учаскедегі адам саны;

$f$ – адамның көлденең проекциясының орташа ауданы, $м^2/$ адам.;

$L_1$ и $B_1$– жолдың есептелген учаскесінің ұзындығы мен ені, м.

Есептеу үшын үй киіміндегі ересек адам проекциясының ауданы қолданылады. Кестелік мәліметтерге сәйкес (Е.1 қосымшасы) [16], бұл көлем 0,1 м$^2$/адам тең. Бөлмеде 5 адам бар.

Бірінші учаске бойынша адамдардың қозғалыс уақытын анықтайық:

$$D_1 = \frac{5 \cdot 0,1}{5,652 \cdot 5,514} = 0,01\ м^2/м^2\ .$$

Кестелік мәліметтерге сәйкес (Е.2 қосымшасы) [16], қозғалыс жылдамдығы $V$ 100 м/мин және қозғалыс қарқыны ($q$) 1 м/мин болып алынады.

Жолдың бірінші учаскесі бойынша адам ағынының қозғалыс уақыты ($t_n$), мин, мына формула бойынша есептеледі:

$$t_1 = \frac{L_1}{V_1}, \quad (4.10)$$

мұндағы, $L_1$ – бірінші учаскенің ұзындығы, м;

$V_1$ – бірінші учаскедегі көлденең жол бойынша адам ағынының қозғалыс жылдамдығының мәні салыстырмалы тығыздыққа байланысты анықталады D, м2/м2.

Қозғалыс жылдамдығын біле отырып, бірінші учаскеде қозғалыс уақытты есептейміз:

$$t_1 = \frac{5,514}{100} = 0,55\ мин\ .$$

Бұдан әрі бірінші жолдан кейінгі жол учаскелеріндегі адам ағынының ($V$) қозғалысының жылдамдығын, оның ішінде есік ойықтары үшін есептеу қажет. Есік ойығының ұзындығы нөлге тең болып қабылданады. Қалыпты жағдайда ойықтағы қозғалыстың ең үлкен ықтимал қарқындылығы $q_{max} = 19.6\ м/мин$ құрайды [16]. 1,6 м-ден кем есік ойығындағы қозғалыс қарқындылығы мына формула бойынша анықталады:

$$q_{\text{Е}} = 2{,}5 + 3{,}75 \cdot b, \qquad (4.11)$$

мұндағы, $b$ – есік ойығының ені.

Ені 1,1 м болатын есік ойығындағы қозғалыс қарқындылығы келесіге тең:

$$q_{\text{Е}} = 2{,}5 + 3{,}75 \cdot 1{,}1 = \ 6{,}62 \text{ м/мин}.$$

$q_{\text{Е}} < q_{max}$, сондықтан ойық арқылы қозғалыс кедергісіз өтеді.

Содан кейін ойық арқылы қозғалыс уақытын есептейміз. Ол ағындағы адамдар санын ойықтың өткізу қабілетіне жеке бөлу ретінде анықталады:

$$t_{\text{Е}_1} = \frac{N_1 \cdot f_1}{q_{\text{Е}} \cdot b} = \frac{5 \cdot 0{,}1}{6{,}62 \cdot 1{,}1} = 0{,}06 \text{ мин}.$$

Жертөледе 46 адам жұмыс істейтіндіктен, бұл қабаттағы адам ағымының тығыздығы:

$$D_2 = \frac{N_2 \cdot f}{L_2 \cdot B_2} = \frac{46 \cdot 0{,}1}{32{,}4 \cdot 3} = 0{,}05 \text{ м}^2/\text{м}^2.$$

Кестелік мәліметтерге сәйкес (Е.2 қосымшасы) [16], қозғалыс жылдамдығы ($V$) 100 м/мин және қозғалыс қарқыны ($q$) 5 м/мин болып алынады. Жолдың екінші учаскесі бойынша (дәлізден баспалдаққа) адам ағынының қозғалыс уақыты келесіге тең:

$$t_2 = \frac{L_2}{V_2} = \frac{32{,}4}{100} = 0{,}32 \text{ мин}.$$

Баспалдақ бойынша қозғалыс жылдамдығын анықтау үшін үшінші учаскедегі қозғалыс қарқындылығы мынадай формула бойынша есептеледі:

$$q_i = \frac{q_{i-1} \cdot b_{i-1}}{b_i}, \qquad (4.12)$$

мұндағы, $b_i$, $b_{i-1}$ – і – түрлі қарастырылып отырған және оның алдындағы жол учаскесінің ені, м;

$q_i$, $q_{i-1}$ – і – түрлі қарастырылатын және алдыңғы жол учаскелері бойынша адам ағыны қозғалысының қарқындылығы мәндері, м/мин.

Қозғалыс қарқындылығы 5 м/мин кезінде ($q_i$) есептейік:

$$q_i = \frac{5 \cdot 3}{1{,}5} = 10 \text{ м/мин}.$$

Кестелік мәндерге сәйкес (Е.2 қосымшасы) [16], баспалдақта адам ағынының жылдамдығы 32 м/мин дейін төмендегені анықталды. Баспалдақпен қозғалыс уақыты (3-ші учаскесі) келесіге тең:

$$t_3 = \frac{L_3}{V_3} = \frac{12}{32} = 0{,}37 \text{ мин} .$$

Бірінші қабатқа өту кезінде сол қабатта қозғалатын адамдардың ағынымен араласу болады. Бірінші қабат үшін адам ағынының тығыздығы:

$$D_4 = \frac{N_4 \cdot f}{L_4 \cdot B_4} = \frac{76 \cdot 0{,}1}{32{,}4 \cdot 3} = 0{,}078 \text{ м}^2/\text{м}^2 .$$

Бұл жағдайда қозғалыс қарқындылығы шамамен 8 м/мин құрайды.

4-ші учаскеге өту кезінде адамдар ағынының қосылуы жүреді, сондықтан қозғалыс қарқындылығы келесі формула арқылы анықталады:

$$q_i = \frac{\sum q_{i-1} \cdot b_{i-1}}{b_i}, \tag{4.13}$$

мұндағы, $q_{i-1}$ – учаскенің басында қосылатын адамдар ағындарының қозғалыс қарқындылығы, м/мин;

$b_{i-1}$ – қосылу учаскелерінің ені, м;

$b_i$ – қарастырылатын жол учаскесінің ені, м.

4-ші учаскеге өту кезінде адамдар ағынының қозғалыс қарқындылығын есептейік:

$$q_i = \frac{(10 \cdot 1{,}5) + (8 \cdot 3)}{3} = 13 \text{ м/мин} .$$

Кестелік мәліметтерге сәйкес (Е.2 қосымшасы) [16], қозғалыс жылдамдығы ($V$) 47 м/мин болып алынады. Осыған сәйкес бірінші қабаттың дәлізі арқылы қозғалыс уақыты келесіге тең:

$$t_4 = \frac{L_4}{V_4} = \frac{32{,}4}{47} = 0{,}68 \text{ мин} .$$

Көшеге шығатын тамбурдың ұзындығы 3 м және ені 3 м. Бұл учаскеде адам ағынының ең жоғары тығыздығы қалыптасады. Сондықтан, кестелік мәліметтерге сәйкес (Е.2 қосымшасы) [16], жылдамдық 15 м/мин дейін төмендейді, ал тамбур бойынша қозғалыс уақыты келесіге тең болады:

$$t_5 = \frac{L_5}{V_5} = \frac{3}{15} = 0{,}2 \text{ мин} .$$

Адам ағынының ең үлкен тығыздығы кезінде ені 1,6 м болатын сыртқы есік ойығы арқылы қозғалыс қарқындылығы 8,5 м/мин құрайды. Ол арқылы қозғалыс уақыты мынаған тең:

$$t_{Д_2} = \frac{N_0 \cdot f_1}{q_{Д_2} \cdot b_{Д_2}} = \frac{122 \cdot 0,1}{8,5 \cdot 1,6} = 0,89 \text{ мин} .$$

Адамдарды эвакуациялаудың есептік уақыты (4.5) формула бойынша есептеледі:

$$t_р = t_{н.э.} + t_1 + t_{Д_1} + t_2 + t_3 + t_4 + t_5 + t_{Д_2} = 4,1 + 0,55 + 0,06 + 0,32 + \\ +0,37 + 0,68 + 0,2 + 0,89 = 7,17 \text{ мин} .$$

Алынған мәліметтерге байланысты эвакуацияның есептік уақыты эвакуациялауға рұқсат етілген ұзақтығынан аз. Осылайша қауіпсіз эвакуациялау шарты орындалады.

Бұл бөлімде серверлік бөлмедегі техникалық қызметкерлердің еңбек жағдайына талдау жүргізілді. Электр желісінің фазалық сымының аспап корпусына тұйықталу ықтималдығы болғандықтан, жасанды қорғаныстық жерлендіру құрылғысының есебі жүргізілді. Есептелген жерлендіру құрылғысының кедергісі ЭҚҚ талаптарын қанағаттандырады және 4 Ом аспайды. Сондай-ақ, серверлік бөлмеден эвакуациялау уақыты есептелді. Барлық учаскелердегі мәжбүрлі эвакуацияның жиынтық ұзақтығы қажетті эвакуация уақытынан аз. Қауіпсіз эвакуациялау шарты орындалады.

## 5 Экономикалық бөлім

### 5.1 Ғылыми – зерттеу элементтері бар тақырыптар жұмысының сипаттамасы

Бұл бөлімде ғылыми – зерттеу жұмысының орындау күрделілігі, өзіндік құнының шығыны, яғни заттар шығыны, негізгі іскерлердің еңбекақысы, еңбекақы қорынан әлеуметтік салыққа аударма, амортизациялық аударма, қосымша және үстеме шығындар анықталды.

### 5.2 ҒЗЖ көлемін және күрделілігін анықтау

ҒЗЖ күрделілігін анықтау үшін, ең алдымен, барлық орындалатын жұмыстардың негізгі кезеңдері мен түрлерлерінің тізімі жасалынады. Бұл кезеңдер логикалық тұрғыда дұрыс реттелген болу керек. Әр ҒЗЖ кезеңінің орындалуына белгіленген уақыт қойылу қажет. Дұрыс ұйымдастырылған жоспар ҒЗЖ жүзеге асырылуының уақытын қысқартады [19].

Барлық орындалатын жұмыстардың негізгі кезеңдері мен еңбек сыйымдылығының тізімі 5.1 кестеде көрсетілген.

5.1 кесте – Жұмыстарды негізгі кезеңдер және еңбек сыйымдылықтар бойынша бөлу

| ҒЗЖ орындалу кезеңі | Орындалатын жұмыс түрі | ҒЗЖ еңбек сыйымдылығы, адам×сағ |
|---|---|---|
| Ақпарат жинау және жалпы талдау | Технология бойынша ақпараттарды жинау және талдау жүргізу.<br>Қолданымдағы бағдарламаларды және модельдерді қарастыру.<br>Мәселені айқындау.<br>Жаңа бағдарлама үшін талаптарды қою. | 310 |
| Кәсіпорынның телекоммуникациялық желісінің құрылым ерекшеліктерін зерттеу | Кәсіпорын желісінің құрылымын зерттеу.<br>Осы желімен бағдарламаның жүзеге асырылуын талдау. | 230 |
| Бағдарламалық жасақтаманың функцияларын жоспарлау | Бағдарламалау әдісін таңдау және шешім алгоритмін әзірлеу.<br>Қолданыстағы технологияларды жетілдіру. | 550 |
| Бағдарлама кодын жазу | Бағдарлама жоспарын құру.<br>Скрипт жазу.<br>Барлық синтаксистік қателерді жою. | 630 |
| Тестілеу | Тестілеу жоспарын құру.<br>Бағдарламаны тестілеу режимінде іске қосу.<br>Қателерді табу және түзету.<br>Қайта іске қосу. Бағдарлама құжаттамасын жазу. | 600 |
| Желілік біріктіру (интеграция) | Тапсырыс беруші желісіне бағдарламаны енгізу бойынша жоспар құру.<br>Серверлік платформаның құрылғыларын және функцияларын орнату және баптау.<br>Кәсіпорын серверлеріне бағдарламалық | 570 |

*5.1 кестенің жалғасы*

| ҒЗЖ орындалу кезеңі | Орындалатын жұмыс түрі | ҒЗЖ еңбек сыйымдылығы, адам×сағ |
|---|---|---|
| | жасақтаманы орнату.<br>Барлық құрылған жүйенің жұмысын тексеру. | |
| Бағдарламаны баптау | Тестілеу нәтижелері бойынша бағдарламаны соңғы рет баптау | 150 |
| Пайдалану және қызмет көрсету | Жаңа бағдарламамен жұмыс істеуге тапсырыс беруші тарапынан мамандар тобын даярлау.<br>Тапсырыс берушінің жаңа бағдарламаны пайдалану тәжірибесіне сәйкес өзгерістер енгізу.<br>Қызмет көрсету жоспарын әзірлеу. | 400 |
| ҒЗЖ еңбек сыйымдылығының барлығы | | 3440 |

### 5.3 ҒЗЖ кеткен шығындарды есептеу

Ғылыми – зерттеу жұмысының өзіндік құнының шығын статьяларына кіретіндер [19]:

– заттар шығыны.
– негізгі іскерлердің енбекақысы.
– енбекақы қорынан әлеуметтік салыққа аударма.
– амортизациялық аударма.
– қосымша шығындар.
– үстеме шығындар.

Барлық шығын статьяларын қосып, ҒЗЖ жұмсалған жалпы шығынның ($Ш_ж$) сомасын табамыз. Бұл келесі формула арқылы анықталады [21]:

$$Ш_ж = Ш_М + Ш_Э + Ш_Е + Ш_{ЖӘ} + Ш_{Ам} + Ш_{Қж} + Ш_{Үж}, \quad (5.1)$$

мұндағы, $Ш_М$ – материалдық ресурстарға жұмсалған жалпы шығын;
$Ш_Э$ – электр қуатына жұмсалатын жалпы шығын;
$Ш_Е$ – еңбекақы төлеуге жұмсалған жалпы шығын;
$Ш_{ЖӘ}$ – әлеуметтік қамтамасыз етуге жұмсалған жалпы шығын;
$Ш_{Ам}$ – амортизациялық аударымдардың жалпы шығыны;

$Ш_{Қж}$ – жалпы қосымша шығындар;

$Ш_{Үж}$ – жалпы үстеме шығындар.

«Заттар шығыны» статьясына ҒЗЖ орындауға қажетті негізгі және қосалқы материалдарға, энергияға жұмсалған шығындар кіреді [19].

Материалдық ресурстардың шығынын есептеу К қосымшасындағы К.1 кестеде көрсетілген нысандар бойынша жүргізіледі.

Материалдық ресурстар шығындарының ($Ш_М$) жалпы сомасы келесі формула арқылы табылады [20]:

$$Ш_М = \sum_{i=1}^{n} Б_i(1 + k_{ТОБ}), \quad (5.2)$$

мұндағы, $Б_i$ – натуральді түрдегі, i-түрлі материалдық ресурстың бағасы;

$k_{ТОБ}$ – техникалық құрылғылар кешенін тасымалдау, орнату және баптау коэффициенті (техникалық құрылғылардың бағасынан 10%) [20];

i - материалдық ресурстың түрі;

n - i-түрлі материалдық ресурстың саны.

Жабдықтың құнын анықтау кезінде тасымалдау, орнату және баптау шығындарын ескеру қажет. Бұл шығындар жабдықты сатып алу бағасынан 10-25 % мөлшерінде қабылдануы мүмкін.

Онда материалдық ресурстар шығындарының ($Ш_М$) жалпы сомасы келесідей анықталады:

$$Ш_М = (1729100 + 1675100 + 19450 + (7300 \cdot 2) + 40161 + 13769 + (9704 \cdot 2) + 36900 + 120386 + 7348 + 20685 + 6259 + 7016 + 12839 + 8689 + 5990 + 4981 + (309 \cdot 10) + (913 \cdot 10) + (1750 \cdot 5) + 28900 + 341466 + 162000 + 11100 + 1269950 + 49950 + 61750 + 68000 + 8592 + 18550) \cdot (1 + 0{,}1) = 6362300 \text{ тг}.$$

ҒЗЖ орындау үшін электржабдықтар қолданғандықтан, электрқуатына жұмсалатын шығындарды есептеу 5.2 кестеде көрсетілген нысан бойынша жүргізіледі [19].

5.2 кесте – Электр қуатына жұмсалатын шығын

| Жабдықтың атауы | Төлқұжат қуаты, кВт | Қуатты пайдалану коэффициенті | ҒЗЖ орындауға арналған жабдықтың жұмыс уақыты, сағ. | Электр қуаттың бағасы, $\frac{тг}{кВт \cdot сағ}$ | Сомасы, тг |
|---|---|---|---|---|---|
| Сервер Dell PowerEdge | 0,75 | 0,9 | 2350 | 19,17 | 30409 |

*5.2 кестенің жалғасы*

| Жабдықтың атауы | Төлқұжат қуаты, кВт | Қуатты пайдалану коэффициенті | ҒЗЖ орындауға арналған жабдықтың жұмыс уақыты, сағ. | Электр қуаттың бағасы, $\frac{тг}{кВт \cdot сағ}$ | Сомасы, тг |
|---|---|---|---|---|---|
| R730 210-ACXU-350 | | | | | |
| Сервер Dell PowerEdge R630 210-ACXS-227 | 0,75 | 0,9 | 2350 | 19,17 | 30409 |
| Моноблок Acer AspireC22-865 | 0,135 | 0,9 | 2350 | 19,17 | 5474 |
| Wi-Fi Модем TP-Link TD-W8961N-RU | 0,012 | 0,7 | 2350 | 19,17 | 379 |
| Коммутатор D-Link DES-1026G/E1A | 0,01568 | 0,8 | 2350 | 19,17 | 566 |
| Бақылау құрылғы UniPing server solution v3/SMS | 0,005 | 0,8 | 2350 | 19,17 | 181 |
| IP видеокамера HIKVISION DS-2CD2041GO-I | 0,0075 | 0,7 | 2350 | 19,17 | 237 |
| UPS CyberPower OLS3000ERT 2U | 0,405 | 0,9 | 2350 | 19,17 | 16421 |
| Электр қуатына жұмсалатын шығын барлығы | | | | | 84076 |

Электр қуатына жұмсалатын жалпы шығын ($Ш_Э$) келесі формула арқылы есептелінеді [19]:

$$Ш_Э = \sum_{i=1}^{n} Т_i \cdot K_i \cdot T_i \cdot Б, \quad (5.3)$$

мұндағы, $Т_i$ – і – түрлі электржабдықтың төлқұжат қуаты, кВт;

$K_i$ - і – түрлі электржабдықтың қуатты пайдалану коэффициенті (0,7-ден 0,9-ға дейін алынады);

$T_i$ – ҒЗЖ орындауға арналған і – түрлі электржабдықтың жұмыс уақыты, сағ;

Б – электр қуатының бағасы, тг/кВт×сағ.;

і – электржабдықтың түрі;

n – і – түрлі электржабдықтың саны.

Электр қуатына жұмсалатын жалпы шығын ($Ш_Э$) сомасы келесіге тең болады:

$$Ш_Э = 30409 + 30409 + 5474 + 379 + 566 + 181 + 237 + 16421 = 84076 \text{ тг}.$$

«Еңбекақы төлеу шығындары» статьясына ҒЗЖ орындаумен айналысатын барлық қызметкерлердің еңбекақысы бойынша шығындар енгізіледі [19].

Еңбекақы төлеу шығындары 5.3 кестеде келтірілген нысан бойынша есептеледі.

5.3 кесте – Еңбекақы төлеу шығындары

| Қызметкердің санаты | Жұмыс мазмұнының атауы | ҒЗЖ еңбек сыйымдылығы, адам×сағ. | Сағаттық мөлшерлеме, тг/сағ. | Сомасы, тг |
|---|---|---|---|---|
| Жетекші инженер | Жоба бойынша басшылық ету | 528 | 1489 | 786192 |
| Байланыс инженері | Жабдықтарды таңдау және байланыс желісін жобалау | 294 | 1191 | 350154 |
| Бағдарламашы | Бағдарлама кодын жазу | 1107 | 1786 | 1977102 |
| Бағдарламашы | Бағдарлама кодын жазу | 1107 | 1786 | 1977102 |
| Көмекші инженер | Жабдықтарды орнату және баптау | 404 | 893 | 360772 |
| Еңбекақы төлеу шығындарына барлығы | | | | 5451322 |

Жалпы еңбекақы төлеу шығындары ($Ш_Е$) келесі формула арқылы анықталады [19]:

$$Ш_Е = \sum_{i=1}^{n} СМ_i \cdot T_i, \quad (5.4)$$

мұндағы, $СМ_i$ – i – түрлі қызметкердің сағаттық мөлшерлемесі, тг;

$T_i$ - i – түрлі қызметкердің ҒЗЖ кезінде еңбек сыйымдылығы, адам×сағ.;

i – қызметкердің санаты;

n – ҒЗЖ орындаумен айналысатын қызметкерлердің саны.

Қызметкердің сағаттық мөлшерлемесін келесі формула арқылы табуға болады:

$$СМ_i = \frac{Е_i}{ЖУҚ_i}, \quad (5.5)$$

мұндағы, $Е_i$ - i – түрлі қызметкердің бір айлық еңбекақысы, тг

$ЖУҚ_i$ – i – түрлі қызметкердің жұмыс уақытының айлық қоры, сағ.

Қызметкерлердің 1 күндік жұмыс уақыты 8 сағат. Бір айда орташа есеппен алғанда 21 жұмыс күні бар. Онда i – түрлі қызметкердің жұмыс уақытының айлық қоры келесіге тең болады:

$$ЖУҚ_i = 21 \cdot 8 = 168 \text{ сағ}.$$

Жетекші инженердің бір айлық еңбекақысы $Е_{ЖИ} = 250000$ тг болғанда, оның сағаттық мөлшерлемесі келесіге тең:

$$СМ_{ЖИ} = \frac{Е_{ЖИ}}{ЖУҚ_{ЖИ}} = \frac{250000 \text{ тг}}{168 \text{ сағ}} \approx 1489 \text{ тг/сағ}.$$

Байланыс инженердің бір айлық еңбекақысы $Е_{БИ} = 200000$ тг болғанда, оның сағаттық мөлшерлемесі келесіге тең:

$$СМ_{БИ} = \frac{Е_{БИ}}{ЖУҚ_{БИ}} = \frac{200000 \text{ тг}}{168 \text{ сағ}} \approx 1191 \text{ тг/сағ}.$$

Бағдарламашының бір айлық еңбекақысы $Е_Б = 300000$ тг болғанда, оның сағаттық мөлшерлемесі келесіге тең:

$$СМ_Б = \frac{Е_Б}{ЖУҚ_Б} = \frac{300000 \text{ тг}}{168 \text{ сағ}} \approx 1786 \text{ тг/сағ}.$$

Көмекші инженердің бір айлық еңбекақысы $Е_{КИ} = 150000$ тг болғанда, оның сағаттық мөлшерлемесі келесіге тең:

$$\text{СМ}_{\text{КИ}} = \frac{\text{Е}_{\text{КИ}}}{\text{ЖУҚ}_{\text{КИ}}} = \frac{150000 \text{ тг}}{168 \text{ сағ}} \approx 893 \text{ тг/сағ}.$$

Еңбекақы төлеу шығындарының ($\text{Ш}_{\text{Е}}$) жалпы сомасы келесіге тең болады:

$$\text{Ш}_{\text{Е}} = 786192 + 350154 + 1977102 + 1977102 + 360772 = 5451322 \text{ тг.}$$

«Еңбекақы қорынан әлеуметтік салыққа аударма» статьясына ҒЗЖ орындаумен айналысатын барлық қызметкерлердің еңбекақыларынан заңнамада белгіленген мөлшерде сәйкес міндетті зейнетақы жарналары, әлеуметтік салық және міндетті әлеуметтік медициналық сақтандыру төлемі алынады. Бұл шығын келесі формула арқылы есептелінеді [22]:

$$\text{Ш}_{\text{ЖӘ}} = (\text{Ш}_{\text{Ж}} - \text{А}_{\text{МЗЖ}} - \text{А}_{\text{МӘМС}}) \cdot \text{С}_{\text{Ә}} - \text{А}_{\text{Ә}}, \quad (5.6)$$

мұндағы, $\text{Ш}_{\text{ЖӘ}}$ – әлеуметтік қамтамасыз етуге кеткен жалпы шығындар;

$\text{Ш}_{\text{Е}}$ – жалпы еңбекақы шығыны;

$\text{А}_{\text{МЗЖ}}$ – міндетті зейнетақы жарналары (жалақыдан 10% [23]);

$\text{А}_{\text{МӘМС}}$ – міндетті әлеуметтік медициналық сақтандыру (жалақыдан 2% [24]);

$\text{С}_{\text{Ә}}$ – әлеуметтік салық төлемі ($\text{С}_{\text{Ә}} = 9{,}5\%$ [25]);

$\text{А}_{\text{Ә}}$ – әлеуметтік аударым (МЗЖ-сыз жалақыдан 3,5% алынатын өлшем [26]).

Онда әлеуметтік қамтамасыз етуге кеткен жалпы шығындар келесі тең болады:

$$\text{Ш}_{\text{ЖӘ}} = \big(5451322 - (5451322 \cdot 0{,}1) - (5451322 \cdot 0{,}02)\big) \cdot 0{,}095 - (4906189{,}8 \cdot 0{,}035) = 284013{,}87 \approx 284014 \text{ тг}.$$

«Амортизациялық аударма» статьясына ҒЗЖ орындау кезінде пайдаланылатын жабдықтар мен аспаптардың құнынан амортизациялық аударымдар сомасы енгізіледі [19]. Амортизациялық аударымдар Л қосымшасындағы Л.1 – кестеде келтірілген нысандар бойынша есептеледі.

Амортизациялық аударымдардың жалпы шығыны ($\text{Ш}_{\text{Ам}}$) келесі формула арқылы табылады [19]:

$$\text{Ш}_{\text{Ам}} = \sum_{i=1}^{n} \frac{\text{Б}_i \cdot H_{Ami} \cdot \text{Т}_{\text{ҒЗЖ}_i}}{100 \cdot \text{Т}_{\text{Эф}_i}}, \quad (5.7)$$

мұндағы, $\text{Б}_i$ – і – түрлі жабдықтың бағасы, тг;

$H_{Ami}$ – і – түрлі жабдықтың бір жылдық амортизация нормасы, %;

$Т_{ҒЗЖ_i}$ – і – түрлі жабдықтың ҒЗЖ кезінде жұмыс істеу уақыты, сағ;

$Т_{Эф_i}$ – бір жылдағы і – түрлі жабдықтың жұмыс уақытының тиімді қоры, сағ,/жыл;

і – жабдықтың түрі;

n – жабдықтардың саны.

Жабдық амортизациясының жылдық нормалары ҚР Салық кодексі бойынша немесе жабдықты пайдаланудың ықтимал мерзімін ескере отырып анықталады. Жабдықты пайдаланудың ықтимал мерзімі 3 жылдан 10 жылға дейін қабылдануы мүмкін. Мерзімдері орташа мәнмен алынғанда [19]:

$$H_{Aмi} = \frac{100}{T_{Ni}}, \quad (5.8)$$

мұндағы, $T_{Ni}$ – і – түрлі жабдықтың пайдаланудың ықтимал мерзімі, жыл.

Онда жабдықтардың бір жылдық амортизация нормалары келесідей есептеледі:

$$H_{Aмc\text{ЕРВ.}} = \frac{100}{T_{Nc}} = \frac{100}{5} = 20,$$

$$H_{A\text{мПП}} = \frac{100}{T_{N\text{пп}}} = \frac{100}{10} = 10,$$

$$H_{A\text{мЖЕЛД.}} = \frac{100}{T_{N\text{ж}}} = \frac{100}{3} = 33{,}3,$$

$$H_{A\text{мЖФ}} = \frac{100}{T_{N\text{ж}}} = \frac{100}{10} = 10,$$

$$H_{A\text{мКОММ.}} = \frac{100}{T_{N\text{к}}} = \frac{100}{5} = 20,$$

$$H_{A\text{мБҚ}} = \frac{100}{T_{N\text{бқ}}} = \frac{100}{5} = 20,$$

$$H_{A\text{мХАБ.}} = \frac{100}{T_{N\text{х}}} = \frac{100}{7} = 14{,}29,$$

$$H_{A\text{мПК}} = \frac{100}{T_{N\text{пк}}} = \frac{100}{10} = 10,$$

$$H_{A\text{мКАМ.}} = \frac{100}{T_{N\text{к}}} = \frac{100}{6} = 16{,}7,$$

$$H_{AмUPS} = \frac{100}{T_{NU}} = \frac{100}{10} = 10,$$

$$H_{A\text{мМОД.}} = \frac{100}{T_{NU}} = \frac{100}{5} = 20,$$

$$H_{A\text{мМОН.}} = \frac{100}{T_{NU}} = \frac{100}{5} = 20,$$

$$H_{A\text{мПТ}} = \frac{100}{T_{NU}} = \frac{100}{10} = 10.$$

Амортизациялық аударымдардың жалпы шығынының ($Ш_{Ам}$) сомасы келесіге тең болады:

$$Ш_{Ам} = 92772 + 89874 + 522 + 3588 + 261 + 1980 + 6459 + 282 + 793 + 240 + 269 + 492 + 333 + 230 + 191 + 9 + 25 + 47 + 1295 + 9161 + 596 + 13628 + 268 = 223315 \text{ тг}.$$

Нақты бағдарламаға арналған "қосымша шығындар" статьясы бойынша шығыстар арнайы ғылыми-техникалық ақпарат пен әдебиетті сатып алуға және дайындауға арналған шығындарды қамтиды. Ұйымға тұтастай әзірленетін норматив бойынша негізгі жалақыға сәйкес пайызбен айқындалады [20]:

$$Ш_{Қi} = \frac{Ш_{Е} \cdot Н_{ҚШ}}{100\%}, \qquad (5.9)$$

мұндағы, $Ш_{Е}$ – еңбекақы төлеу шығындарының ($Ш_{Е}$) жалпы сомасы, тг;
$Н_{ҚШ}$ – ұйымға тұтастай әзірленетін норматив бойынша қосымша шығындар (%), дипломдық жобалауда ол 20%-ға тең.

Онда қосымша шығындар келесідей есептеледі:

$$Ш_{Қж} = \frac{5451322 \cdot 20\%}{100\%} = 1090265 \text{ тг.}$$

"Үстеме шығындар" ($Р_{Нi}$) статья бойынша шығындар басқару аппаратын, қосалқы шаруашылықтарды және эксперименттік өндірістерді ұстау қажеттілігімен байланысты қойылады. Сондай-ақ бұл шығындар жалпы шаруашылық мұқтаждықтарға жұмсалады және олар негізгі жалақыға пайыздық қатынаста норматив ($Н_{НР}$) бойынша нақты бағдарламаға белгіленеді [20]:

$$Ш_{Yi} = \frac{Ш_{Е} \cdot Н_{YШ}}{100\%}, \qquad (5.10)$$

мұндағы, $Ш_{Е}$ – еңбекақы төлеу шығындарының ($Ш_{Е}$) жалпы сомасы, тг;
$Ш_{Yi}$ – нақты бағдарлама бойынша үстеме шығындар, тг;

$Н_{ҮШ}$ – ұйымға тұтастай әзірленетін норматив бойынша үстеме шығындар (%), дипломдық жобалауда ол 70%-ға тең.

Үстеме шығындар келесіге тең болады:

$$Ш_{Үж} = \frac{5451322 \cdot 70\%}{100\%} = 3815926 \text{ тг.}$$

Статьялар бойынша табылған деректер негізінде 5.4 кестеде келтірілген нысан бойынша ҒЗЖ орындау үшін шығындар сметасы жасалады.

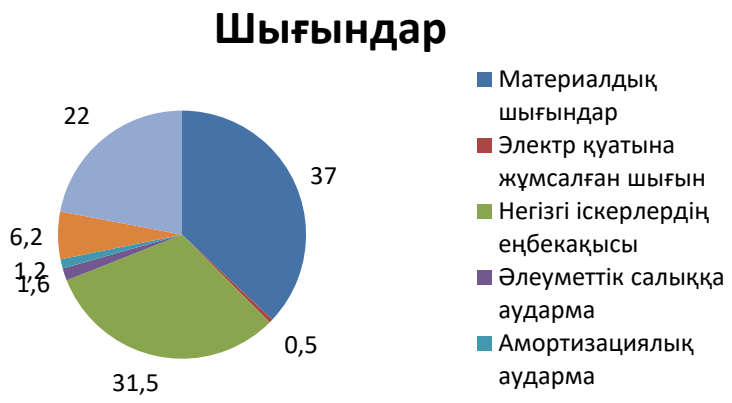
ҒЗЖ жұмсалған жалпы шығынның ($Ш_ж$) сомасы келесіге тең болады:

$$Ш_ж = 6362300 + 84076 + 5451322 + 284014 + 223315 + 1090265 + 3815926 = 17311218\ .$$

5.4 кесте – ҒЗЖ орындау үшін шығындар сметасы

| Шығын статьялары | Сомасы, тг | Пайзбен жалпы сомадан |
|---|---|---|
| 1. Заттар шығыны, оның ішінде:<br>- материалдық шығындар<br>- электр қуатына жұмсалған шығын | <br>6362300<br>84076 | <br>37<br>0,5 |
| 2. Негізгі іскерлердің еңбекақысы | 5451322 | 31,5 |
| 3. Әлеуметтік салыққа аударма | 284014 | 1,6 |
| 4. Амортизациялық аударма | 223315 | 1,2 |
| 5. Қосымша шығындар | 1090265 | 6,2 |
| 6. Үстеме шығындар | 3815926 | 22 |
| Смета бойынша барлығы | 17311218 | 100 |

5.1 суретте ғылыми – зерттеу жұмысына жұмсалатын жалпы шығындары диаграмма түрінде көрсетілген.




5.1 сурет – ҒЗЖ шығындардың құрылым диаграммасы

### 5.4 ҒЗЖ ықтимал (келісімді) бағасын анықтау

ҒЗЖ ықтимал (келісімді) бағасының шамасы тапсырыс берушінің (тұтынушының) және орындаушының экономикалық мүдделеріне жауап беретін деңгейде орындалып, оның тиімділігі, сапасы мен мерзімдері ескеріле отырып белгіленуі тиіс [19].

Қолданбалы ҒЗЖ келісімді бағасы ($Б_К$) келесі формула бойынша есептеледі [19]:

$$Б_К = Ш_Ж \cdot \left(1 + \frac{P}{100}\right), \quad (5.11)$$

мұндағы, $Ш_Ж$ – ҒЗЖ жұмсалған жалпы шығындар (5.4 кестеден), тг;

Р – ҒЗЖ тиімділігінің орташа деңгейі, 20%-ға тең (20-30% шамасында алынады).

Онда қолданбалы ҒЗЖ келісімді бағасы ($Б_К$) келесіге тең болады:

$$Б_К = 17311218 \cdot \left(1 + \frac{20\%}{100\%}\right) = 20773462 \text{ тг}.$$

Енді қосылған құн салығын есепке ала отырып іске асыру бағасын ($Б_I$) анықтаймыз. Бұл келесі формула негізінде анықталады:

$$Б_I = Б_К + Б_К \cdot ҚҚС, \quad (5.12)$$

ҚҚС мөлшерлемесі ҚР Салық кодексімен белгіленеді.

2020 жылға ҚҚС 12% мөлшерде болып белгіленді [27].

Онда ҚҚС есепке ала отырып іске асыру бағасы ($Б_I$) келесіге тең болады:

$$Б_I = 20773462 + 20773462 \cdot 0{,}12 = 23266278 \text{ тг}.$$

ҒЗЖ болжамды пайдасы мынадай формула бойынша есептеледі [20]:

$$П_{ҒЗЖ} = Б_I \cdot \frac{P}{100}, \quad (5.13)$$

мұндағы, $Б_I$ – ҚҚС есепке ала отырып ҒЗЖ іске асыру бағасы, тг;

Р – ҒЗЖ тиімділігінің орташа деңгейі, 22%-ға тең (20-30% шамасында алынады) [19].

Онда ҒЗЖ болжамды пайдасы келесіге тең болады:

$$П_{ҒЗЖ} = 23266278 \cdot \frac{22}{100} = 5118581 \text{ тг}.$$

Экономикалық бөлім бойынша қорытынды жасай отырып, ғылыми-зерттеулік жұмысына жұмсалатын шығын сомасы 17311218 теңгені құрайды. 5.4 кестенің нәтижелерін негізге ала отырып, шығыстардың негізгі бөлігі материалдық шығындарға 37% және негізгі іскерлердің еңбекақыларына 31,5% жұмсалғанына көз жеткізуге болады. Сонымен қатар үстеме шығындарына, әгни өндірістік жабдықтарға қызмет көрсету және жөндеу үшін пайдаланылған материалдардың, қосалқы бөлшектердің құны, бөлмені жалдау ақысы мен т.б. қордың 22% бөлінгенін көреміз. ҚҚС есепке ала отырып ҒЗЖ іске асыру құны 23266278 теңгені құрайды.

Бұл дипломдық жоба экономикалық пайда алудан гөрі ғылыми қызығушылықты тудырғандықтан, онда есептеулер шамамен жүргізілген.

## Қорытынды

Бұл дипломдық жобада желілік трафикті алу мен терең талдаудың қазіргі заманғы технологиялары қарастырылды. TCP/IP және ISO OSI хаттамалық жүйелері туралы қысқаша айтылды. Талдау тереңдігімен ерекшеленетін IP-желісінің трафигін тексерудің бірнеше технологиялары талданды. Nmap-ты және қарапайым желіні сканерлеу және карталаудың негізгі әдістері қарастырылды. Содан кейін біз Linux жүйесі және tcpdump бағдарламасы көмегімен желілік дестелік трафигін алып, зерттедік. Қарастырылған әдістердің негізінде, Python тілінде жазылған, желілік құрылғыға келіп түсетін трафикті алатын бағдарламаны әзірледік.

Жасалынған бағдарламаны бүкіл жұмыс барысында жаңа функциялармен толықтыра отырып, біз желілік трафикті талдаудың бірнеше әдістерін құрдық. Бұл әдістер келесі үрдістерді орындайды: IP-бақылауларының толық сөздігін басып шығару, серверлер мен клиенттердің толық тізімін құру, сервер мен клиент арасындағы қосылыстардың тізімін құру және бақылау деректерінің толық гистограммасын шығару. Соңында біз әзірленген талдау әдістерімен тәжірибе жасауға болатын қарапайым мәзірге негізделген интерфейсті құрдық.

Бұл ғылыми-зерттеу жұмысы Python тілінде жазылған жоғарыжылдамдықты желілік трафикті талдайтын бағдарламалардың одан әрі дамытуына мүмкіндік береді. Өйткені ол ашық бастапқы коды бар бағдарлама болып саналады.

Тіршілік қауіпсіздігі бөлімінде серверлік бөлмедегі техникалық қызметкерлердің еңбек жағдайына талдау жүргізілді. Электр желісінің фазалық сымының аспап корпусына тұйықталу ықтималдығы болғандықтан, жасанды қорғаныстық жерлендіру құрылғысының есебі жүргізілді. Есептелген жерлендіру құрылғысының кедергісі ЭҚҚ талаптарын қанағаттандырады және 4 Ом аспайды. Сондай-ақ, серверлік бөлмеден эвакуациялау уақыты есептелді. Барлық учаскелердегі мәжбүрлі эвакуацияның жиынтық ұзақтығы қажетті эвакуация уақытынан аз. Қауіпсіз эвакуациялау шарты орындалады.

Экономикалық бөлімде біз, ғылыми-зерттеулік жұмысына жұмсалатын шығын сомасы 17311218 теңгені құрайтынын анықтадық. Есептелген нәтижелерді негізге ала отырып, шығыстардың негізгі бөлігі материалдық шығындарға 37% және негізгі іскерлердің еңбекақыларына 31,5% жұмсалғанына көз жеткізуге болады. Сонымен қатар үстеме шығындарына, әғни өндірістік жабдықтарға қызмет көрсету және жөндеу үшін пайдаланылған материалдардың, қосалқы бөлшектердің құны, бөлмені жалдау ақысы мен т.б. қордың 22% бөлінгенін көреміз. ҚҚС есепке ала отырып ҒЗЖ іске асыру құны 23266278 теңгені құрайды.

**Қысқартулар тізбесі**

DPI – Deep Packet Inspection
P2P – Peer-to-Peer
SLA – Service Level Agreement
DDoS – Distributed Denial of Service
DiffServ – Differentiated services
HTTP – HyperText Transfer Protocol
DPP – Deep Packet Processing
DTE – Data Terminal Equipment
DCE – Data Circuit-terminating Equipment
ФА – Фазалық модуляция
ЖМ – Жиіліктік модуляция
RZ – Return to Zero
NRZ – Non Return to Zero
FDDI – Fiber Distributed Data Interface
LAN – Local Area Network
RIP – Routing Internet Protocol
OSPF – Open Shortest Path First
ICMP – Internet Control Message Protocol
ACK – Acknowledgment flag
TCP – Transmission Control Protocol
WAN – Wide Area Network
PDH – Plesiochronous Digital Hierarchy
SDH – Synchronous Digital Hierarchy
ATM – Asynchronous Transfer Mode
UDP – User Datagram Protocol
ARP – Address Resolution Protocol
RARP – Reserve Address Resolution Protocol
ICMP – Internet Control Message Protocol
IGMP – Internet Group Message Protocol
RIP – Routing Information Protocol
OSPF – Open Shortest Path First
BGP – Border Gateway Protocol
SMTP – Simple Mail Protocol
FTP – File Transfer Protocol
DNS – Domain Name System
SNMP – Simple Network Management Protocol
NFS – Network File System
RPC – Remote Procedure Call
TFTP – Trivial File Transfer Protocol
HTML – Hypertext Markup Language
WWW – World Wide Web
SPI – Shallow Packet Inspection

NAT – Network Address Translation
GTP – GPRS Tunnelling Protocol
MPI – Medium Packet Inspection
LPI – Lightweight Payload Inspection
IANA – Internet Assigned Numbers Authority
URL – Uniform Resource Locator
MPLS – Multiprotocol label switching
VLAN – Virtual Local Area Network
NSH – Network Service Header
SDN – Software Defined Network
BT – Behavioral Targeting
QoS – Quality of Service
HIPS – Host-Based Intrusion Prevention System
BRAS – Broadband Remote Access Server
CMTS – Cable Modem Termination System
GGSN – GPRS Gateway Support Node
IMS – IP Multimedia Subsystem
PCRF – Policy and Charging Rules Function
AAA – Authentication, Authorization, Accounting
БЛИС – Бағдарламаланатын логикалық интегралды схемалары
M2M – Machine to Machine
P2NMAP – Python Passive Network Mapping
GUI – Graphical User Interface
SONAR – SOund Navigation And Ranging)
TTL – Time to Live
OpenSSL – Open Secure Sockets Layer
DMZ – Demilitarized Zone
SPAN – Switched Port Analyzer
RSPAN – Remote Switched Port Analyzer
ДӨО – Деректерді өңдеу орталығы
ЭҚҚ – Электр қондырғыларының қағидалары
ПҚ – Пәтерлік қалқан
КҚК – Кепілді қоректендіру қалқаны
КҚ – Күштік қалқан
ҒЗЖ – Ғылыми – зерттеу элементтері бар тақырыптар жұмысы
МЗЖ – Міндетті зейнетақы жарналары
МӘМС – Міндетті әлеуметтік медициналық сақтандыру
ҚҚС – Қосылған құн салығы

## Әдебиеттер тізімі


1. Горнак А. Технология DPI для широкополосного доступа // Технологии и средства связи. 2010. № 4, сентябрь. с. 66 – 67

2. Гольдштейн А.Б., Гольдштейн Б.С. Технология и протоколы MPLS. – СПб.: БХВ-Петербург. 2005. 304 с.

3. TCP/IP и DNS в теории и на практике. Полное руководство / Пер. с чеш. Рус. изд. под ред. М.В. Финкова и А.В. Анисимова. Серия «Полное руководство». – СПб.: Наука и Техника, 2006. – 608 с., ил

4. Березовский В.А., Дулькейт И.В., Савицкий О.К. Современная декаметровая радиосвязь: оборудование, системы и комплексы / Под ред. В.А. Березовского. – М.: Радиотехника, 2011. – 444 с.: ил

5. Берлин А.Н. Основные протоколы Интернет: Учебное пособие / А.Н. Берлин – М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2012. – 504 с.: ил., табл.

6. Гольдштейн Б.С. Глубокая инспекция пакетов DPI: проблемы и подходы // Вестник связи. 2018. № 9, сентябрь. с. 5 – 10

7. Сенченко Ю.Л. Некоторые аспекты высокоскоростной обработки трафика// Технологии и средства связи. 2013 № 1, март. с. 52 – 53.

8. Гольдштейн Б.С., Елагин В.С., Сенченко Ю.Л. Протоколы AAA: RADIUS и Diameter./ Серия «Телекоммуникаионные протоколы» . Книга 9. – СПб.: БХВ-Петербург. 2011. 352 с.

9. Сенченко Ю. Некоторые аспекты высокоскоростной обработки трафика // Технологии и средства связи. 2013. № 1, январь. с. 8 – 9

10. Chet Hosmer. Python Passive Network Mapping P2NMAP 1st Edition. USA: Syngpress Imprint, 2015, ISBN: 978-0-12-802721-9 – 162 p.

11. IP адреса поисковых систем // [Программирование Asp.Net] URL: https://www.aspnet.com.ua/BlogAll/ip-adresa-poiskovyh-sistem-google-yandex-i-drugie.aspx/322 (қолдану уақыты 22.05.2020)

12. socket.gethostbyaddr() // [Socket – Low-level networking interface] URL: https://docs.python.org/2/library/socket.html (қолдану уақыты 06.04.2020)

13. «Правила устройства электроустановок Республики Казахстан» (ПУЭ РК – 2015) / Утверждены приказом Министра энергетики Республики Казахстан от 20 марта 2015 года № 230

14. Учебное пособие (для студентов высших учебных заведений специальности «Безопасность жизнедеятельности и защита окружающей среды»)/ Н.Г. Приходько, Ф.Р. Жандаулетова, А.Ж. Амренова. – Алматы: АУЭС, 2015. – 110 с.: табл. 10, ил.18, библиогр. - 21 назв.

15. СНиП II-90-81. Производственные здания промышленных преприятий / Госстрой СССР. – М.: Стройиздат, 1982. – 14 с.

16. Н.Г. Приходько, Ф.Р. Жандаулетова. Основы пожарной безопасности. Методические указания к выполнению курсовой работы для студентов специальности 5В073100 – Безопасность жизнедеятельности и защита окружающей среды. - Алматы: АУЭС, 2013 - 31 с.

17. Ж.С. Абдимуратов. Охрана труда. Методические указания к выполнению расчетно-графических работ для студентов - бакалавров специальности 5В071800 - «Электроэнергетика» - Алматы: АУЭС, 2013 - 22с.

18. Шрайбер, Г. Огнетушащие средства. Физико-химические процессы при горении и тушении. Пер. с нем. – М.: Стройиздат, 1985. – 240 с.

19. Бекишева А.И. Методические указания к выполнению экономической части дипломной работы для бакалавров специальности 5В0703 - Информационные системы – Алматы: АУЭС; 2013. –24 с.

20. З.Д. Еркешева, Г.Ш. Боканова. Методические указания к выполнению экономической части дипломных работ для студентов специальности 5В070400 – Вычислительная техника и программное обеспечение. – Алматы: АУЭС, 2013 – 40 с.

21. проф. Базылов Қ.Б., проф. Алибаева С.А., аға лаб. Нурмагамбетова С. С. Бітіруші жұмысының экономикалық бөлімі үшін әдістемелік нұсқаулар. 050719 – Радиотехника, электроника және телекоммуникация мамандығының барлық оқу түрінің студенттеріне арналған.

22. Жалақыдан «таза» табысты қалай есептеуге болады // [egov – электрондық үкіметі] URL: https://egov.kz/cms/kk/articles/employment_relations/calculate_net_income (қолдану уақыты 02.04.2020)

23. «ҚР зейнетақымен қамсыздандыру туралы» Қазақстан Республикасының 2013 жылғы 21 маусымдағы № 105-V Заңы. // [Әділет - Қазақстан Республикасынормативтік құқықтық актілеріниңақпараттық-құқықтық жүйесі] URL: http://adilet.zan.kz/kaz/docs/Z1300000105#z132 (қолдану уақыты 02.04.2020)

24. Міндетті әлеуметтік медициналық сақтандыру // [egov – электрондық үкіметі] URL: https://egov.kz/cms/kk/articles/health_care/osms (қолдану уақыты 02.04.2020)

25. Ст. 485 Налоговый Кодекс РК 25 декабря 2017 года № 120-VI ЗРК « О ставках налога» // [Кодексы KZ – комплекс правовой информации (законодательство) Республики Казахстан] URL: https://kodeksy-kz.com/ka/nalogovyj_kodeks/485.htm (қолдану уақыты 02.04.2020)

26. Закон РК от 26 декабря 2019 г. № 286-VI «Об обязательном социальном страховании» // [Zakon.kz – Законадательство Республики Казахстан] URL: https://www.zakon.kz/5001725-ob-ischislenii-sotsialnyh-otchisleniy-s.html (қолдану уақыты 02.04.2020)

27. 1 статьи 422 Налогового кодекса РК от 25 декабря 2017 года «О ставках НДС» // [Учёт.kz] URL: https://uchet.kz/stavki/NDS

# А қосымшасы

## capture443.py бағдарламасы

```python
# Бір портта белсенділікті салыстыратын Python Script
# Linux-та жұмыс істейді

# Стандартты кітапханалық модульдерді енгізу

import socket                # өңделмеген сокеттер үшін пайдаланылатын желілік интерфейс кітапханасы
import os                    # операциялық жүйенің функциялары, яғни файлдық енгізу / шығару
import sys                   # жүйелік деңгейдегі функциялар, яғни exit ()
from struct import *         # Жолдарды екілік деректер ретінде өңдеу

# Тұрақтылар

PROTOCOL_TCP = 6             # IP деңгейі үшін TCP хаттамасы

# PacketExtractor
#
# Мақсаты: IP және TCP тақырыбында өрістерді алу
#
# Кіріс:   пакет:     socket.recvfrom() әдісінің буфері
# Шығыс:   тізім:     serverIP, clientIP, serverPort
#

def packetextractor(packet):

    # Ip тақырыбы үшін алғашқы 20 таңбаны жойыңыз
    stripPacket = packet[0:20]

    # енді оларды ашу
    ipHeaderTuple = unpack('!BBHHHBBH4s4s', stripPacket)

    # unpack кортежді қайтарады, мысалды сипаттау үшін мен
    # әрбір мәнді жеке аламын
                                          # Өріс мазмұны
    verLen        = ipHeaderTuple[0]      # Өріс 0: Нұсқасы мен ұзындығы
    TOS           = ipHeaderTuple[1]      # Өріс 1: Қызмет көрсету түрі
    packetLength  = ipHeaderTuple[2]      # Өріс 2: Пакеттің ұзындығы
    packetID      = ipHeaderTuple[3]      # Өріс 3: Сәйкестендіру
    flagFrag      = ipHeaderTuple[4]      # Өріс 4: Жалаулар / фрагменттің жылжуы
    RES           = (flagFrag >> 15) & 0x01 # Резервтелген
    DF            = (flagFrag >> 14) & 0x01 # Бөлінбейді
    MF            = (flagFrag >> 13) & 0x01 # Көп бөліктер
    timeToLive    = ipHeaderTuple[5]      # Өріс 5: Time to Live (TTL)
    protocol      = ipHeaderTuple[6]      # Өріс 6: Хаттама нөмері
    checkSum      = ipHeaderTuple[7]      # Өріс 7: Тақырыптың бақылау сомасы
    sourceIP      = ipHeaderTuple[8]      # Өріс 8: Бастапқы IP
    destIP        = ipHeaderTuple[9]      # Өріс 9: Тағайындалған IP

    # Алынған мәндерді есептеу / түрлендіру

    version       = verLen >> 4        # Upper Nibble - нұсқа нөмірі
    length        = verLen & 0x0F      # Lower Nibble өлшемін ұсынады
    ipHdrLength   = length * 4         # Байтта тақырып ұзындығын есептеу

    # жіберуші мен алушының мекенжайын белгіленген нүктелік жолдарға түрлендіру

    sourceAddress      = socket.inet_ntoa(sourceIP);
    destinationAddress = socket.inet_ntoa(destIP);

    if protocol == PROTOCOL_TCP:

        stripTCPHeader = packet[ipHdrLength:ipHdrLength+20]

        # unpack кортежді қайтарады, мысалды сипаттау үшін мен
        # мен unpack() функциясы арқылы әрбір жеке мәнді шығарамын

        tcpHeaderBuffer = unpack('!HHLLBBHHH', stripTCPHeader)

        sourcePort            = tcpHeaderBuffer[0]
        destinationPort       = tcpHeaderBuffer[1]
        sequenceNumber        = tcpHeaderBuffer[2]
        acknowledgement       = tcpHeaderBuffer[3]
        dataOffsetandReserve  = tcpHeaderBuffer[4]
        tcpHeaderLength       = (dataOffsetandReserve >> 4) * 4
        flags                 = tcpHeaderBuffer[5]
        FIN                   = flags & 0x01
        SYN                   = (flags >> 1) & 0x01
        RST                   = (flags >> 2) & 0x01
        PSH                   = (flags >> 3) & 0x01
        ACK                   = (flags >> 4) & 0x01
        URG                   = (flags >> 5) & 0x01
        ECE                   = (flags >> 6) & 0x01
        CWR                   = (flags >> 7) & 0x01
        windowSize            = tcpHeaderBuffer[6]
        tcpChecksum           = tcpHeaderBuffer[7]
        urgentPointer         = tcpHeaderBuffer[8]
```

```
86
87          if sourcePort < 1024:
88              serverIP   = sourceAddress
89              clientIP   = destinationAddress
90              serverPort = sourcePort
91          elif destinationPort < 1024:
92              serverIP   = destinationAddress
93              clientIP   = sourceAddress
94              serverPort = destinationPort
95          else:
96              serverIP   = "Filter"
97              clientIP   = "Filter"
98              serverPort = "Filter"
99
100         return([serverIP, clientIP, serverPort], [SYN, serverIP, TOS, timeToLive, DF, windowSize])
101     else:
102         return(["Filter", "Filter", "Filter"], [NULL, Null, Null, Null])
103
104 # ------------ НЕГІЗГІ СКРИПТ ОСЫ ЖЕРДЕН БАСТАЛАДЫ -----------------
105
106 if __name__ == '__main__':
107
108     # Ескерту скрипті root режимінде іске қосылуы керек
109     # яғни sudo python ..
110
111     # Желілік картадағы Promiscious (тәртіпсіз) режимді қосу
112     # Жүйелік шақыруды жасау
113     # Ескерту: Linux негізінде
114
115     ret = os.system("ifconfig eth0 promisc")
116
117     # Егер бәрі сәтті өтсе, жалғастырыңыз
118     if ret == 0:
119
120         print("eth0 configured in promiscous mode")
121
122         # Python сокеті модулін пайдаланып жаңа сокет жасаңыз
123         # AF_INET     : Address Family Internet (Мекенжайлық Отбасылық Интернет)
124         # SOCK_RAW    : Желілік деңгейде өңделмеген хаттама
125         # IPPROTO_TCP : Сокеттің көлік деңгейі TCP екенін көрсетеді
126
127         # Сокетті ашу әрекеті
128         try:
129             mySocket = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP)
130
131             print("Raw Socket Open")                                # егер нәтиже сәтті жарияланса
132         except:
133             print("Raw Socket Open Failed")                         # сокет жұмыс істемесе
134             sys.exit()
135
136         # пакетті алу нәтижесін сақтау үшін тізімді жасау
137         # Біз бұл мысал үшін тек қана Server IP, Client IP,
138         # Server Port сақтаймыз. Назар аударыңыз, біз клиенттен серверді ажырату үшін
139         # негізделген болжамдар жасаймыз
140
141         ipObservations = []
142         osObservations = []
143
144         # Ең көп 500 бақылауды ғана алу
145         maxObservations = 500
146
147         # Порт сүзгісі 443 портқа орнатылған
148         # TCP порты 443 TLS / SSL арқылы HTTP протоколы ретінде анықталады
149
150         portValue = 443
151
152         try:
153
154             while maxObservations > 0:
155
156                 # алу әрекеті (бұл шақыру синхронды болып табылады)
157                 recvBuffer, addr = mySocket.recvfrom(225)
158
159                 # алынған пакетті декодтау
160                 # жоғарыдағы жергілікті пакетті шығарып алу функциясын шақырыңыз
161
162                 content, fingerPrint = packetextractor(recvBuffer)
163
164                 if content[0] != "Filter":
165                     # нәтижелерді біздің тізімге қосу
166                     # егер бұл біздің портқа сәйкес келсе
167                     if content[2] == portValue:
168                         ipObservations.append(content)
169                         maxObservations = maxObservations - 1
170                         # егер SYN жалауы орнатылса, онда
171                         # osObservations ішіне саусақ ізі туралы деректерді жазыңыз
172                         if fingerPrint[0] == 1:
173                             osObservations.append([fingerPrint[1], \
174                                                    fingerPrint[2], \
175                                                    fingerPrint[3], \
176                                                    fingerPrint[4], \
177                                                    fingerPrint[5]])
178                         else:
```

```
179                                 # Біздің порт емес
180                                 continue
181                             else:
182                                 # Жарамайтын пакет
183                                 continue
184         except:
185             print("socket failure")
186             exit()
187 
188         # Алу процесі аяқталды
189         # Linux жүйелік шақыруды пайдаланып
190         # Promiscuous режимін өшіру
191         ret = os.system("ifconfig eth0 -promisc")
192 
193         # Сокетті жабамыз
194         mySocket.close()
195 
196         # Бірегей сұрыпталған тізім жасау
197         # Кейін біз кез келген қайталанатын жазбаларды болдырмау үшін
198         # тізімді жиынтыққа түрлендіреміз
199         # содан кейін сұрыптауға арналған тізімге қайта түрлендіреміз
200 
201         uniqueSrc = set(map(tuple, osObservations))
202         finalList = list(uniqueSrc)
203         finalList.sort()
204 
205         uniqueFingerprints = set(map(tuple, osObservations))
206         finalFingerPrintLIst = list(uniqueFingerprints)
207         finalFingerPrintLIst.sort()
208 
209         # Бірегей комбинацияларды басып шығарыңыз
210         print("Unique Packets")
211         for packet in finalList:
212             print(packet)
213         print("Unique Fingerprints")
214         for osFinger in finalFingerPrintLIst:
215             print(osFinger)
216     else:
217         print('Promiscious Mode not Set')
```

# Б қосымшасы

## IPObservationDictionary класы

```python
# ipObservation класын анықтау
# және тестілеу коды

import datetime         # Python стандартты кітапханасының күн мен уақыт әдістері
import pickle           # Python стандартты кітапханасының таңдау әдістері

#
# Класс: IPObservationDictionary
#
# Сипаттамасы: IP бақылаумен байланысты барлық
#              әдістер мен қасиеттерді өңдейді
#
#

class IPObservationDictionary:

    # Құрылым

    def __init__(self):

        # Нысан атрибуттары

        self.Dictionary = {}                # IP бақылауларды сақтауға арналған сөздік

    # Бақылауды қосу әдісі

    def AddOb(self, key):

        # Ағымдағы сағатты алу

        now = datetime.datetime.now()
        hour = now.hour

        # Сөздікте кілт бар ма екенін тексеріңіз

        if key in self.Dictionary:

            # Егер иә болса, ағымдағы мәнін алу
            curValue = self.Dictionary[key]

            # Ағымдағы сағат үшін есептелуді ұлғайту
            curValue[hour-1] = curValue[hour-1] + 1

            # Осы кілтпен байланысты мәнді жаңарту
            self.Dictionary[key] = curValue

        else:
            # егер кілт әлі жоқ болса,
            # онда жаңасын жасау керек

            curValue = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

            # Ағымдағы сағат үшін есептелуді ұлғайту
            curValue[hour-1] = curValue[hour-1] + 1

            self.Dictionary[key] = curValue

    # Бақылауды алу әдіс
    # Егер бақылау табылмаса, None-ді қайтару керек

    def GetOb(self, key):

        if key in self.Dictionary:
            curValue = self.Dictionary[key]
            return curValue
        else:
            return None

    # Ағымдағы бақылау сөздігін
    # көрсетілген файлда сақтау

    def SaveOb(self, fileName):

        with open(fileName, 'wb') as fp:
            pickle.dump(self.Dictionary, fp)

    # Көрсетілген файлдан бақылау
    # сөздігін жүктеу және оған жазу

    def LoadOb(self, fileName):

        with open(fileName, 'rb') as fp:
            self.Dictionary = pickle.loads(fp.read())

    # Құрылымды жою / Нысанды жою

    def __del__(self):
        print("Closed")
```

*Б қосымшасының жалғасы*

```
90  # IPObservationDictionary класының соңы ============================
91
92
93  ipOB = IPObservationDictionary()
94
95  ipOB.AddOb( ("192.168.0.2", "129.168.0.39", 80) )
96  ipOB.AddOb( ("192.168.0.2", "129.168.0.41", 80) )
97  ipOB.AddOb( ("192.168.0.2", "129.168.0.41", 80) )
98  ipOB.AddOb( ("192.168.0.2", "129.168.0.41", 80) )
99  ipOB.AddOb( ("192.168.0.2", "129.168.0.41", 80) )
100 ipOB.AddOb( ("192.168.0.2", "129.168.0.39", 80) )
101
102
103 print("Print out observed values\n")
104
105 theValue = ipOB.GetOb( ("192.168.0.2", "129.168.0.41", 80) )
106 print(theValue)
107
108 theValue = ipOB.GetOb( ("192.168.0.2", "129.168.0.39", 80) )
109 print(theValue)
110
111 theValue = ipOB.GetOb( ("192.168.0.2", "129.168.0.47", 80) )
112 print(theValue)
113
114 # Енді бақылауларды файлға сақтау
115 ipOB.SaveOb("SavedObservation.dict")
116
117 # Енді бақылауларды файлға жүктеу
118 ipOB.LoadOb("SavedObservation.dict")
119
120 # Сол нәтижелерді қайта тексеру
121
122 print("Print out observed values after re-loading\n")
123
124 theValue = ipOB.GetOb( ("192.168.0.2", "129.168.0.41", 80) )
125 print(theValue)
126
127 theValue = ipOB.GetOb( ("192.168.0.2", "129.168.0.39", 80) )
128 print(theValue)
129
130 theValue = ipOB.GetOb( ("192.168.0.2", "129.168.0.47", 80) )
131 print(theValue)
```

# В қосымшасы

## OSObservationDictionary класы

```python
# OSObservation класын анықтау
# және тестілеу коды

import datetime         # Python стандартты кітапханасының күн мен уақыт әдістері
import pickle           # Python стандартты кітапханасының таңдау әдістері

#
# Класс: OSObservationDictionary
#
# Сипаттамасы: IP бақылаумен байланысты барлық
#              әдістер мен қасиеттерді өңдейді
#
#

class OSObservationDictionary:

    # Құрылым

    def __init__(self):

        # Нысан атрибуттары

        self.Dictionary = {}                # IP бақылауларды сақтауға арналған сөздік

    # Бақылауды қосу әдісі

    def AddOb(self, key):

        # Ағымдағы сағатты алу

        now = datetime.datetime.now()
        hour = now.hour

        # Сөздікте кілт бар ма екенін тексеріңіз

        if key in self.Dictionary:

            # Егер иә болса, ағымдағы мәнін алу
            curValue = self.Dictionary[key]

            # Ағымдағы сағат үшін есептелуді ұлғайту
            curValue[hour-1] = curValue[hour-1] + 1

            # Осы кілтпен байланысты мәнді жаңарту
            self.Dictionary[key] = curValue

        else:
            # егер кілт әлі жоқ болса,
            # онда жаңасын жасау керек

            curValue = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

            # Ағымдағы сағат үшін есептелуді ұлғайту
            curValue[hour-1] = curValue[hour-1] + 1

            self.Dictionary[key] = curValue

    # Бақылауды алу әдіс
    # Егер бақылау табылмаса, None-ді қайтару керек

    def GetOb(self, key):

        if key in self.Dictionary:
            curValue = self.Dictionary[key]
            return curValue
        else:
            return None

    # Ағымдағы бақылау сөздігін
    # көрсетілген файлда сақтау

    def SaveOb(self, fileName):

        with open(fileName, 'wb') as fp:
            pickle.dump(self.Dictionary, fp)

    # Көрсетілген файлдан бақылау
    # сөздігін жүктеу және оған жазу

    def LoadOb(self, fileName):

        with open(fileName, 'rb') as fp:
            self.Dictionary = pickle.loads(fp.read())

    # Құрылымды жою / Нысанды жою

    def __del__(self):
        print("Closed")
```

```
89
90  # OSObservationDictionary класының соңы ============================
91
92
93  osOB = OSObservationDictionary()
94
95  osOB.AddOb( ('23.235.39.223',  0,  53,   1,   14480) )
96  osOB.AddOb( ('23.253.39.223',  0,  53,   1,   14480) )
97  osOB.AddOb( ('64.233.185.95',  0,  42,   0,   42540) )
98  osOB.AddOb( ('64.233.185.95',  0,  42,   0,   42540) )
99  osOB.AddOb( ('66.153.250.240', 0,  57,   0,   28960) )
100 osOB.AddOb( ('66.153.250.240', 0,  57,   0,   28960) )
101
102
103 print("Print out observed OS values\n")
104
105 theValue = osOB.GetOb( ('23.235.39.223',   0,  53,   1,   14480) )
106 print(theValue)
107
108 theValue = osOB.GetOb( ('66.153.250.240',  0,  57,   0,   28960) )
109 print(theValue)
110
111 theValue = osOB.GetOb( ('66.153.250.240',  0,  59,   1,   28960) )
112 print(theValue)
113
114 # Енді бақылауларды файлға сақтау
115 osOB.SaveOb("SavedOSObservation.dict")
116
117 # Енді бақылауларды файлға жүктеу
118 osOB.LoadOb("SavedOSObservation.dict")
119
120 # Сол нәтижелерді қайта тексеру
121
122 print("Print out observed values after re-loading\n")
123
124 theValue = osOB.GetOb( ('23.235.39.223',   0,  53,  1,   14480) )
125 print(theValue)
126
127 theValue = osOB.GetOb( ('66.153.250.240',  0,  57,  0,   28960) )
128 print(theValue)
129
130 theValue = osOB.GetOb( ('66.153.250.240',  0,  59,  0,   28960) )
131 print(theValue)
```

# Г қосымшасы

## P2NMAP-Capture.py бағдарламасының коды

```python
# Python пакеттерін алу скрипті
# IP жазу және ОЖ-ні бақылауға ареалған Python скрипті
# Linux және Windows операциялық жүйелеріне арналған

# Стандартты кітапханалық модульдерді енгізу

import argparse             # Python стандартты кітапханасы - командалық жол параметрлері мен
                            # аргументтері үшін синтаксистік талдау
import socket               # өңделмеген сокеттер үшін пайдаланылатын желілік интерфейс кітапханасы
import signal               # тоқтату сигналдарын генерациялау, яғни күту уақыты
import os                   # операциялық жүйенің функциялары, яғни файлдық енгізу / шығару
from struct import *        # Жолдарды екілік деректер ретінде өңдеу
import datetime             # Python стандартты кітапханасының күн мен уақыт әдістері
import time                 # Python стандартты кітапханасының уақыт әдістері
import pickle               # Python стандартты кітапханасының таңдау әдістері
import platform             # Python стандартты кітапханасының платформасы
import sys                  # Python стандартты кітапханасының жүйелік модулі


# Тұрақтылар

PROTOCOL_TCP = 6            # IP деңгейі үшін TCP хаттамасы
PROTOCOL_UDP = 17           # IP деңгейі үшін UDP хаттамасы

#
# Атауы: ValDirWrite
#
# Сипаттамасы: Каталогқа баратын жолдың бар екенін тексеретін функция.
#              Тек дәлелдерді тексеру үшін қолданылады
#
# Кіріс: каталогқа баратын жол
#
# Әрекеттер:
#              егер сәйкес болса, каталог жолы қайтарылады
#
#              егер ол сәйкес болса, онда ArgumentTypeError argparse
#              арқылы шақырып, пайдаланушыға қайтарылады
#

def ValDirWrite(theDir):

    # Каталогтың жолын тексеру
    if not os.path.isdir(theDir):
        raise argparse.ArgumentTypeError('Directory does not exist')

    # Жазу үшін жолды тексеру
    if os.access(theDir, os.W_OK):
        return theDir
    else:
        raise argparse.ArgumentTypeError('Directory is not writable')

# End ValDirWrite ==================================


# Трафикті алу ұзақтығын өңдеу үшін күту уақыты класын жасау

class myTimeout(Exception):
    pass

# Трафикті алу ұзақтығы жеткен кезде үзілісті
# тудыратын сигнал өңдегішін жасау

def handler(signum, frame):
    if VERBOSE:
        print('Capture Complete', signum)
        print()

    raise myTimeout()

#
# Класс: IPObservationDictionary
#
# Сипаттамасы: IP бақылаумен байланысты барлық
#              әдістер мен қасиеттерді өңдейді
#
#

class IPObservationDictionary:

    # Құрылым

    def __init__(self):

        # Нысан атрибуттары

        self.Dictionary = {}              # IP IP бақылауларды сақтауға арналған сөздік

```

```
88      # Бақылауды қосу әдісі
89
90      def AddOb(self, key):
91
92          # Ағымдағы сағатты алу
93
94          now = datetime.datetime.now()
95          hour = now.hour
96
97          # Сөздікте кілт бар ма екенін тексеріңіз
98
99          if key in self.Dictionary:
100
101             # Егер иә болса, ағымдағы мәнін алу
102             curValue = self.Dictionary[key]
103
104             # Ағымдағы сағат үшін есептелуді ұлғайту
105             curValue[hour-1] = curValue[hour-1] + 1
106
107             # Осы кілтпен байланысты мәнді жаңарту
108             self.Dictionary[key] = curValue
109
110         else:
111             # егер кілт әлі жоқ болса,
112             # онда жаңасын жасау керек
113
114             curValue = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
115
116             # Ағымдағы сағат үшін есептелуді ұлғайту
117             curValue[hour-1] = curValue[hour-1] + 1
118
119             self.Dictionary[key] = curValue
120
121     # Бақылауды алу әдіс
122     # Егер бақылау табылмаса, None-ді қайтару керек
123
124     def GetOb(self, key):
125
126         if key in self.Dictionary:
127             curValue = self.Dictionary[key]
128             return curValue
129         else:
130             return None
131
132     # Сөздіктің мазмұнын басып шығару
133
134     # Сөздіктің мазмұнын басып шығару
135
136     def PrintOb(self):
137         print("\nIP Observations")
138         print("Unique Combinations:    ", str(len(self.Dictionary)))
139         print()
140
141         # Басып шығару тақырыбы
142
143         print('                                                ',)
144         print("|-------------------------------------------  Hourly Observations  ---------------------------------------
145         print('%16s' % "Server",)
146         print('%16s' % "Client",)
147         print('%7s'  % "Port",)
148         print('%5s'  % "Type",)
149
150         for i in range(0, 24):
151             print( ' ' )
152             print('%02d' % i,)
153         print()
154
155         # Мазмұнды басып шығару
156         for keys, values in self.Dictionary.items():
157
158             print('%16s' % keys[0],)
159             print('%16s' % keys[1],)
160             print('%7s'  % str(keys[2]),)
161             print('%5s'  % keys[3],)
162
163             for i in range(0, 24):
164                 print('%4s' % str(values[i]),)
165             print()
166
167     # Ағымдағы бақылау сөздігін
168     # көрсетілген файлда сақтау
169
170     def SaveOb(self, fileName):
171
172         with open(fileName, 'wb') as fp:
173             pickle.dump(self.Dictionary, fp)
174
175     # Көрсетілген файлдан бақылау
176     # сөздігін жүктеу және оған жазу
177
178     def LoadOb(self, fileName):
179
180         with open(fileName, 'rb') as fp:
```

```
181             self.Dictionary = pickle.loads(fp.read())
182
183         # Құрылымды жою / Нысанды жою
184
185         def __del__(self):
186             if VERBOSE:
187                 print("Closed")
188
189     # IPObservationDictionary класының соңы ============================
190
191     #
192     # Класс: OSObservationDictionary
193     #
194     # Сипаттамасы: IP бақылаумен байланысты барлық
195     #              әдістер мен қасиеттерді өңдейді
196     #
197     #
198
199     class OSObservationDictionary:
200
201         # Құрылым
202
203         def __init__(self):
204
205             # Нысан атрибуттары
206
207             self.Dictionary = {}               # IP бақылауларды сақтауға арналған сөздік
208
209         # Бақылауды қосу әдісі
210
211         def AddOb(self, key):
212
213             # Ағымдағы сағатты алу
214
215             now = datetime.datetime.now()
216             hour = now.hour
217
218             # Сөздікте кілт бар ма екенін тексеріңіз
219
220             if key in self.Dictionary:
221
222                 # Егер иә болса, ағымдағы мәнін алу
223                 curValue = self.Dictionary[key]
224
225                 # Ағымдағы сағат үшін есептелуді ұлғайту
226                 curValue[hour-1] = curValue[hour-1] + 1
227
228                 # Осы кілтпен байланысты мәнді жаңарту
229                 self.Dictionary[key] = curValue
230
231             else:
232                 # егер кілт әлі жоқ болса,
233                 # онда жаңасын жасау керек
234
235                 curValue = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
236
237                 # Ағымдағы сағат үшін есептелуді ұлғайту
238                 curValue[hour-1] = curValue[hour-1] + 1
239
240                 self.Dictionary[key] = curValue
241
242         # Бақылауды алу әдіс
243         # Егер бақылау табылмаса, None-ді қайтару керек
244
245         def GetOb(self, key):
246
247             if key in self.Dictionary:
248                 curValue = self.Dictionary[key]
249                 return curValue
250             else:
251                 return None
252
253         # Сөздіктің мазмұнын басып шығару
254
255         def PrintOb(self):
256
257             print("\nOS Observations")
258             print("Unique Combinations:    ", str(len(self.Dictionary)))
259             print()
260
261             # Басып шығару тақырыбы
262             print('                                                    ')
263             print("|-----------------------------------------  Hourly Observations  ----------------------------------------
264
265             print('%16s' % "Server",)
266             print('%4s'  % "TOS",)
267             print('%4s'  % "TTL",)
268             print('%6s'  % "DF",)
269             print('%7s'  % "Window",)
270
271             for i in range(0, 24):
272                 print(  ' ' )
273                 print('%02d' % i,)
```

```
274         print()
275
276         # Мазмұнды басып шығару
277         for keys, values in self.Dictionary.items():
278             print('%16s' % keys[0],)
279             print('%4s'  % str(keys[1]),)
280             print('%4s'  % str(keys[2]),)
281             print('%6s'  % str(keys[3]),)
282             print('%7s'  % str(keys[4]),)
283
284             for i in range(0, 24):
285                 print('%4s' % str(values[i]),)
286             print()
287
288     # Ағымдағы бақылау сөздігін
289     # көрсетілген файлда сақтау
290
291     def SaveOb(self, fileName):
292
293         with open(fileName, 'wb') as fp:
294             pickle.dump(self.Dictionary, fp)
295
296     # Көрсетілген файлдан бақылау
297     # сөздігін жүктеу және оған жазу
298
299     def LoadOb(self, fileName):
300
301         with open(fileName, 'rb') as fp:
302             self.Dictionary = pickle.loads(fp.read())
303
304     # Құрылымды жою / Нысанды жою
305
306     def __del__(self):
307         if VERBOSE:
308             print("Closed")
309
310 # OSObservationDictionary класының соңы ============================
311
312
313 # PacketExtractor
314 #
315 # Мақсаты: IP және TCP тақырыбында өрістерді алу
316 #
317 # Кіріс:   пакет:     socket.recvfrom() әдісінің буфері
318 # Шығыс:   тізім:     serverIP, clientIP, serverPort
319 #
320
321 def PacketExtractor(packet):
322
323     if PLATFORM == "LINUX":
324
325         ETH_LEN  = 14       # ETHERNET ТАҚЫРЫБЫНЫҢ ҰЗЫНДЫҒЫ
326         IP_LEN   = 20       # IP       ТАҚЫРЫБЫНЫҢ ҰЗЫНДЫҒЫ
327         UDP_LEN  = 8        # UDP      ТАҚЫРЫБЫНЫҢ ҰЗЫНДЫҒЫ
328
329     elif PLATFORM == "WINDOWS":
330
331         ETH_LEN  = 0        # ETHERNET ТАҚЫРЫБЫНЫҢ ҰЗЫНДЫҒЫ
332         IP_LEN   = 20       # IP       ТАҚЫРЫБЫНЫҢ ҰЗЫНДЫҒЫ
333         UDP_LEN  = 8        # UDP      ТАҚЫРЫБЫНЫҢ ҰЗЫНДЫҒЫ
334
335     else:
336         print("Platform not supported")
337         quit()
338
339     ethernetHeader = packet[0:IP_LEN]
340
341     # Ip тақырыбы үшін алғашқы таңбаларды жою
342     ipHeader = packet[ETH_LEN:ETH_LEN+IP_LEN]
343
344     # енді оларды ашу
345     ipHeaderTuple = unpack('!BBHHHBBH4s4s', ipHeader)
346
347     # unpack кортежді қайтарады, мысалды сипаттау үшін мен
348     # әрбір мәнді жеке аламын
349                                         # өріс мазмұны
350     verLen        = ipHeaderTuple[0]    # өріс 0: Нұсқасы мен ұзындығы
351     TOS           = ipHeaderTuple[1]    # өріс 1: Қызмет көрсету түрі
352     packetLength  = ipHeaderTuple[2]    # өріс 2: Пакеттің ұзындығы
353     packetID      = ipHeaderTuple[3]    # өріс 3: Сәйкестендіру
354     flagFrag      = ipHeaderTuple[4]    # өріс 4: Жалаулар / фрагменттің жылжуы
355     RES           = (flagFrag >> 15) & 0x01  # Резервтелген
356     DF            = (flagFrag >> 14) & 0x01  # Бөлінбейді
357     MF            = (flagFrag >> 13) & 0x01  # Көп бөліктер
358     timeToLive    = ipHeaderTuple[5]    # өріс 5: Time to Live (TTL)
359     protocol      = ipHeaderTuple[6]    # өріс 6: Хаттама нөмері
360     checkSum      = ipHeaderTuple[7]    # өріс 7: Тақырыптың бақылау сомасы
361     sourceIP      = ipHeaderTuple[8]    # өріс 8: Бастапқы IP
362     destIP        = ipHeaderTuple[9]    # өріс 9: Тағайындалған IP
363
364     # Алынған мәндерді есептеу / түрлендіру
```

```
365
366         version        = verLen >> 4         # Upper Nibble - нұсқа нөмірі
367         length         = verLen & 0x0F       # Lower Nibble өлшемін ұсынады
368         ipHdrLength    = length * 4          # Байтта тақырып ұзындығын есептеу
369
370         # жіберуші мен алушының мекенжайын белгіленген нүктелік жолдарға түрлендіру
371
372         sourceAddress       = socket.inet_ntoa(sourceIP);
373         destinationAddress = socket.inet_ntoa(destIP);
374
375         if protocol == PROTOCOL_TCP:
376
377             stripTCPHeader = packet[ETH_LEN+ipHdrLength:ipHdrLength+ETH_LEN+IP_LEN]
378
379             # unpack кортежді қайтарады, мысалды сипаттау үшін мен
380             # unpack() функциясы арқылы әрбір жеке мәнді шығарамын
381
382             tcpHeaderBuffer = unpack('!HHLLBBHHH', stripTCPHeader)
383
384             sourcePort              = tcpHeaderBuffer[0]
385             destinationPort         = tcpHeaderBuffer[1]
386             sequenceNumber          = tcpHeaderBuffer[2]
387             acknowledgement         = tcpHeaderBuffer[3]
388             dataOffsetandReserve    = tcpHeaderBuffer[4]
389             tcpHeaderLength         = (dataOffsetandReserve >> 4) * 4
390             flags                   = tcpHeaderBuffer[5]
391             FIN                     = flags & 0x01
392             SYN                     = (flags >> 1) & 0x01
393             RST                     = (flags >> 2) & 0x01
394             PSH                     = (flags >> 3) & 0x01
395             ACK                     = (flags >> 4) & 0x01
396             URG                     = (flags >> 5) & 0x01
397             ECE                     = (flags >> 6) & 0x01
398             CWR                     = (flags >> 7) & 0x01
399             windowSize              = tcpHeaderBuffer[6]
400             tcpChecksum             = tcpHeaderBuffer[7]
401             urgentPointer           = tcpHeaderBuffer[8]
402
403             if sourcePort <= 1024:              # IP серверді сервер деп болжаймыз
404                 serverIP   = sourceAddress
405                 clientIP   = destinationAddress
406                 serverPort = sourcePort
407                 status = True
408             elif destinationPort <= 1024:       # IP адресті тағайындау сервері деп болжаймыз
409                 serverIP   = destinationAddress
410                 clientIP   = sourceAddress
411                 serverPort = destinationPort
412                 status = True
413             elif sourcePort <= destinationPort: # IP серверді сервер деп болжаймыз
414                 serverIP   = sourceAddress
415                 clientIP   = destinationAddress
416                 serverPort = sourcePort
417                 status = True
418             elif sourcePort > destinationPort:  # IP адресті тағайындау сервері деп болжаймыз
419                 serverIP   = destinationAddress
420                 clientIP   = sourceAddress
421                 serverPort = destinationPort
422                 status = True
423             else:                               # Ешқашан мұнда түспеуі керек
424                 serverIP   = "Filter"
425                 clientIP   = "Filter"
426                 serverPort = "Filter"
427                 status = False
428
429             return( status, (serverIP, clientIP, serverPort, "TCP"), [SYN, serverIP, TOS, timeToLive, DF, windowSize] )
430
431         elif protocol == PROTOCOL_UDP:
432
433             stripUDPHeader = packet[ETH_LEN+ipHdrLength:ETH_LEN+ipHdrLength+UDP_LEN]
434
435             # unpack кортежді қайтарады, мысалды сипаттау үшін мен
436             # unpack() функциясы арқылы әрбір жеке мәнді шығарамын
437
438             udpHeaderBuffer = unpack('!HHHH', stripUDPHeader)
439
440             sourcePort              = udpHeaderBuffer[0]
441             destinationPort         = udpHeaderBuffer[1]
442             udpLength               = udpHeaderBuffer[2]
443             udpChecksum             = udpHeaderBuffer[3]
444
445             if sourcePort <= 1024:              # IP серверді сервер деп болжаймыз
446                 serverIP   = sourceAddress
447                 clientIP   = destinationAddress
448                 serverPort = sourcePort
449                 status = True
450             elif destinationPort <= 1024:       # IP адресті тағайындау сервері деп болжаймыз
451                 serverIP   = destinationAddress
452                 clientIP   = sourceAddress
453                 serverPort = destinationPort
454                 status = True
455             elif sourcePort <= destinationPort: # IP серверді сервер деп болжаймыз
456                 serverIP   = sourceAddress
457                 clientIP   = destinationAddress
```

```
458                 serverPort = sourcePort
459                 status = True
460             elif sourcePort > destinationPort:  # IP адресті тағайындау сервері деп болжаймыз
461                 serverIP   = destinationAddress
462                 clientIP   = sourceAddress
463                 serverPort = destinationPort
464                 status = True
465             else:                               # Ешқашан мұнда түспеуі керек
466                 serverIP   = "FILTER"
467                 clientIP   = "FILTER"
468                 serverPort = "FILTER"
469                 status = False
470
471             return( status, (serverIP, clientIP, serverPort, "UDP"), ["FILTER","FILTER","FILTER","FILTER","FILTER","FILTER"]
472         else:
473
474             return( False, ("Filter", "Filter", "Filter", "Filter"), ["FILTER","FILTER","FILTER","FILTER","FILTER","FILTER"]
475
476     #
477     # Spinner класы
478     #
479     # Процесті көрсету үшін айналмалы таңбаны экранда көрсету үшін қолданылады
480     #
481     #
482     class Spinner:
483
484         # Құрылым
485
486         def __init__(self):
487
488             self.symbols = [' |', ' /', ' -', ' \\', ' |', ' \\', ' -', 'END']
489             self.curSymbol = 0
490
491             sys.stdout.write("\b\b\b%s " % self.symbols[self.curSymbol])
492             sys.stdout.flush()
493
494         def Spin(self):
495
496             if self.symbols[self.curSymbol] == 'END':
497                 self.curSymbol = 0
498
499             sys.stdout.write("\b\b\b%s " % self.symbols[self.curSymbol])
500             sys.stdout.flush()
501             self.curSymbol += 1
502
503
504     # End Spinner Class
505
506
507     # НЕГІЗГІ БАҒДАРЛАМА ОСЫ ЖЕРДЕН БАСТАЛАДЫ
508     #==============================================
509
510     if __name__ == '__main__':
511
512         # Parser Object аргументін орнату
513
514         parser = argparse.ArgumentParser('P2NMAP-Capture')
515
516         parser.add_argument('-v', '--verbose', help="Display packet details", action='store_true')
517         parser.add_argument('-m', '--minutes', help='Capture Duration in minutes', type=int)
518         parser.add_argument('-p', '--outPath', type= ValDirWrite, required=True, help="Output Directory")
519
520         theArgs = parser.parse_args()
521
522         VERBOSE = theArgs.verbose
523
524     # Трафикті алу ұзақтығын орнату
525     captureDuration = theArgs.minutes * 60
526
527     try:
528         # Ескерту скрипті root режимінде іске қосылуы керек
529         # яғни sudo python ..
530
531         if platform.system() == "Linux":
532
533             PLATFORM = "LINUX"
534
535             # Желілік картадағы Promiscious (тартпісіз) режимді қосу
536             # Жүйелік шақыруды жасау
537             # Ескерту: Linux негізінде
538
539             ret = os.system("ifconfig eth0 promisc")
540             if ret != 0:
541                 print('Promiscuous Mode not Set')
542                 quit()
543
544             # Python сокеті модулін пайдаланып жаңа сокет жасау
545             # PF_PACKET    : Хаттамалардың отбасы деңгейін анықтайды
546             # SOCK_RAW     : Желілік деңгейде өңделмеген хаттама
547             # socket.htons(0x0800) : Барлық тақырыптар мен пакеттерді анықтайды
548             #                      : Ethernet және IP, соның ішінде TCP / UDP және т.б.
549
```

```
550         # Өңделмеген пакеттерді алу үшін сокетті ашу
551
552         rawSocket = socket.socket(socket.PF_PACKET, socket.SOCK_RAW, socket.htons(0x0800))
553
554         # Сигнал өңдеушісін пайдаланушы көрсеткен уақытқа орнатыңыз
555
556         signal.signal(signal.SIGALRM, handler)
557         signal.alarm(captureDuration)  # Бір циклдың уақыты
558
559     elif platform.system() == "Windows":
560
561         PLATFORM = "WINDOWS"
562
563         # Windows операциялық жүйесі үшін орнату әр түрлі болады
564
565         # Сәйкестендіру үшін біздің IP мекенжайымызды алу
566         hostname = socket.gethostname()
567         host = socket.gethostbyname(hostname)
568
569         # rawSocket құру
570         rawSocket = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_IP)
571         # Сокеттің опцияларын орнату
572         rawSocket.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)
573         # Хостқа сәкестендіру
574         rawSocket.bind((host, 0))
575         # Барлық пакеттерді қабылдау үшін сокетті орнату
576         rawSocket.ioctl(socket.SIO_RCVALL, socket.RCVALL_ON)
577         startTime = time.time()
578         endTime = startTime + captureDuration
579     else:
580         print("Platform not supported")
581         quit()
582 except:
583     print("Socket Error")
584     quit()
585
586 if VERBOSE:
587     print("Network      : Promiscuous Mode")
588     print("Sniffer      : Ready: \n")
589
590     # Процесті көрсету үшін Spinner нысанын жасау
591     obSPIN = Spinner()
592
593 # IP және ОЖ бақылау сөздіктерін жасау
594
595 ipOB = IPObservationDictionary()
596 osOB = OSObservationDictionary()
597
598 # Тек үзілістің мәнімен
599 # үзілетін мәңгілік циклді жасау
600
601 packetsCaptured = 0
602 try:
603     while True:
604
605         # Трафикті алу процесі (синхронды шақыру, бұл кезде ол күтеді)
606         receivedPacket=rawSocket.recv(65535)
607
608         packetsCaptured += 1            # Алынған пакеттерді санау
609
610         if VERBOSE:
611             # Экранды жаңарту
612             obSPIN.Spin()
613
614         # алынған пакеттерді декодтау
615         # жоғарыдағы пакеттерді шығару функциясын шақыру
616
617         status, osContent, fingerPrint = PacketExtractor(receivedPacket)
618
619         # Егер жауап True болып қайтарылса,
620         # нәтижелерді өңдей аламыз
621
622         if status:
623
624             # Мазмұнды ipObservations-қа қосу
625
626             ipOB.AddOb(osContent)
627
628             if fingerPrint[0] == 1:
629                 osContent = tuple(fingerPrint[1:])
630                 osOB.AddOb(osContent)
631
632         else:
633             # Сейкес емес пакет
634             continue
635
636         if PLATFORM == "WINDOWS":
637             if time.time() > endTime:
638                 raise myTimeout
```

```
639
640     except myTimeout:
641         pass
642
643     # Трафикті алу процесі аяқталды
644
645     if VERBOSE:
646
647         print("\nTotal Packets Captured: ", str(packetsCaptured))
648         print()
649
650         ipOB.PrintOb()
651         osOB.PrintOb()
652
653         print("\nSaving Observations exit: .ipDict and .osDict")
654
655     ipOutFile = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")+".ipDict"
656     osOutFile = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")+".osDict"
657
658     ipOutput = os.path.join(theArgs.outPath, ipOutFile)
659     osOutput = os.path.join(theArgs.outPath, osOutFile)
660
661     ipOB.SaveOb(ipOutput)
662     osOB.SaveOb(osOutput)
663
664     if PLATFORM == "LINUX":
665         # Алу процесі аяқталды
666         # Linux жүйелік шақыруды пайдаланып
667         # Promiscuous режимін өшіру
668
669         ret = os.system("ifconfig eth0 -promisc")
670
671     elif PLATFORM == "WINDOWS":
672         rawSocket.ioctl(socket.SIO_RCVALL, socket.RCVALL_OFF)
673
674     else:
675         print("Platform not supported")
676         quit()
677
678     # Өңделмеген сокетті жабу
679     rawSocket.close()
```

## Д қосымшасы

## P2NMAP-Capture.py бағдарламасының нәтижесі

```
Network       : Promiscuous Mode
Sniffer       : Ready:


Total Packets Captured:  2195


IP Observations
Unique Combinations:      33
```

| Server | Client | Port | Type | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Hourly Observations | | | | | | | | | | | | | | | | | | | | | | | |
| 192.168.0.255 | 192.168.0.6 | 8612 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 91.189.92.152 | 192.168.0.19 | 80 | TCP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 127.0.0.1 | 127.0.0.1 | 53 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 192.168.0.11 | 255.255.255.255 | 17500 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 192.168.0.11 | 239.255.255.250 | 1900 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 91.189.92.191 | 192.168.0.19 | 80 | TCP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 255.255.255.255 | 192.168.0.14 | 8611 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.0.0.0 | 255.255.255.255 | 68 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 192.168.0.11 | 192.168.0.255 | 137 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 239.255.255.250 | 192.168.0.12 | 1900 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 209.18.47.61 | 192.168.0.19 | 53 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 192.168.0.5 | 224.0.0.251 | 5353 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 224.0.0.1 | 192.168.0.6 | 8612 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 255.255.255.255 | 192.168.0.14 | 8612 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 239.255.255.250 | 192.168.0.13 | 1900 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 56 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 169.254.7.79 | 224.0.144.1 | 52613 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 192.168.0.14 | 255.255.255.255 | 17500 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 192.168.0.6 | 192.168.0.255 | 17500 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 192.168.0.14 | 192.168.0.255 | 138 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 239.255.255.250 | 192.168.0.4 | 1900 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 209.18.47.62 | 192.168.0.19 | 53 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 91.189.91.24 | 192.168.0.19 | 80 | TCP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1340 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 192.168.0.11 | 192.168.0.255 | 17500 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 91.189.91.14 | 192.168.0.19 | 80 | TCP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 596 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 255.255.255.255 | 192.168.0.11 | 1947 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 224.0.0.252 | 192.168.0.11 | 5355 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 239.255.255.250 | 169.254.7.79 | 1900 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 192.168.0.6 | 255.255.255.255 | 17500 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 239.255.255.250 | 192.168.0.18 | 1900 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 239.255.255.250 | 192.168.0.14 | 1900 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 127.0.0.1 | 127.0.0.1 | 41637 | TCP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 192.168.0.14 | 192.168.0.255 | 17500 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 192.168.0.255 | 192.168.0.11 | 1947 | UDP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

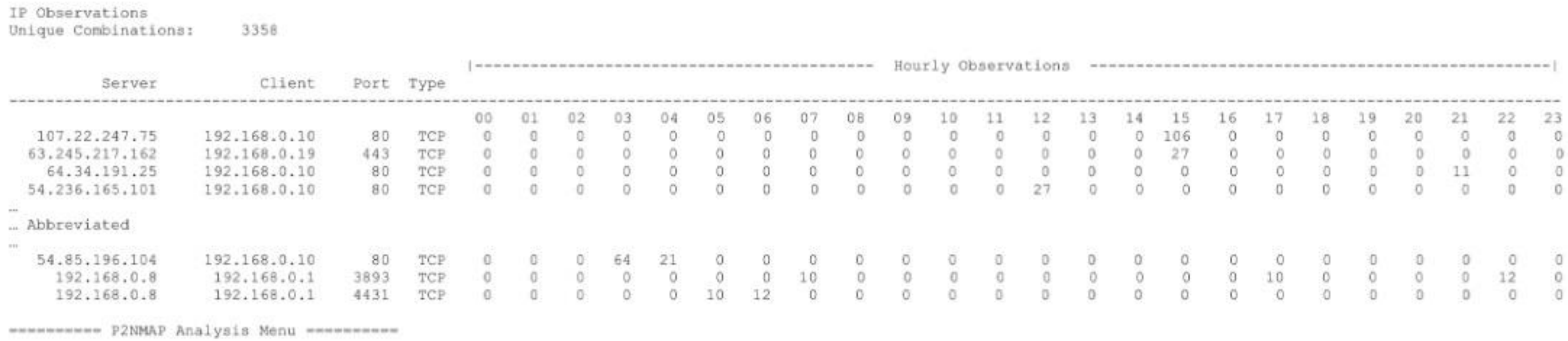
```
OS Observations
Unique Combinations:      4
```

| Server | TOS | TTL | DF | Window | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Hourly Observations | | | | | | | | | | | | | | | | | | | | | | | |
| 91.189.91.14 | 40 | 50 | 1 | 14480 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 91.189.92.152 | 0 | 48 | 1 | 14480 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 91.189.92.191 | 0 | 49 | 1 | 5792 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 91.189.91.24 | 40 | 47 | 1 | 28960 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
Saving Observations ext: .ipDict and .osDict
Closed
Closed
```

# Е қосымшасы

## PrintOb функциясының орындалу нәтижесі



```
IP Observations
Unique Combinations:     3358

                                         |------------------------------------------  Hourly Observations  ------------------------------------------------|
        Server           Client     Port  Type
---------------------------------------------------------------------------------------------------------------------------------------------------------------------
                                              00   01   02   03   04   05   06   07   08   09   10   11   12   13   14   15   16   17   18   19   20   21   22   23
  107.22.247.75     192.168.0.10     80   TCP   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0  106    0    0    0    0    0    0    0    0
 63.245.217.162     192.168.0.19    443   TCP   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0   27    0    0    0    0    0    0    0    0
   64.34.191.25     192.168.0.10     80   TCP   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0   11    0    0
 54.236.165.101     192.168.0.10     80   TCP   0    0    0    0    0    0    0    0    0    0    0    0   27    0    0    0    0    0    0    0    0    0    0    0
…
… Abbreviated
…
  54.85.196.104     192.168.0.10     80   TCP   0    0    0   64   21    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    192.168.0.8      192.168.0.1   3893   TCP   0    0    0    0    0    0    0   10    0    0    0    0    0    0    0    0    0   10    0    0    0    0   12    0
    192.168.0.8      192.168.0.1   4431   TCP   0    0    0    0    0   10   12    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0

========== P2NMAP Analysis Menu ==========
```

## Ж қосымшасы

## Сервер / клиент байланыс гистограммасы

```
Hourly Histogram
==============================================================
Server:            0.0.0.0
==============================================================
          Client     Port                          Port Description  Type
 255.255.255.255       68                 Bootstrap Protocol Client   UDP

HR
00: * (3)
01: * (2)
02: * (3)
03: * (2)
04: * (2)
05: * (3)
06: * (2)
07: * (2)
08: * (1)
09: ************************************************************************************************************ (518)
10: ************ (65)
11: * (5)
12: * (2)
13: * (8)
14: ** (12)
15: * (2)
16: * (2)
17: ** (11)
18: * (2)
19: * (2)
20: * (2)
21: * (3)
22: * (2)
23: * (5)
==============================================================
Server:        103.31.6.36          Australia
==============================================================
          Client     Port                          Port Description  Type
    192.168.0.10       80                        World Wide Web HTTP   TCP

HR
00:
01:
02:
03:
04:
05:
06:
07:
08:
09:
10:
11:
12:
13:
14: ************************************************************************************************************ (39)
15:
16:
17:
18:
19:
20:
21:
22:
23:
==============================================================
Server:   104.130.251.189     United States
==============================================================
          Client     Port                          Port Description  Type
    192.168.0.10       80                        World Wide Web HTTP   TCP

HR
00:
01:
02:
03:
04:
05:
06:
07:
08:
09:
10:
```

```
11:
12:
13:
14:
15: ************************************************************************************************** (17)
16:
17:
18:
19:
20:
21:
22:
23:
…
… Abbreviated
…
============================================================
Server:    23.218.114.120     United States
============================================================
          Client    Port                          Port Description  Type
    192.168.0.10      80                        World Wide Web HTTP   TCP

HR
00:
01:
02:
03:
04: ************************************** (16)
05: ***************************************** (17)
06: ************************************** (15)
07: ***************************************** (17)
08: ******************************************** (19)
09:
10: ************************************************************** (28)
11: ******************************************************** (24)
12:
13: ************************************************************************************************** (44)
14: ********************************************************************** (32)
15: ************************************** (15)
16: ****************************** (14)
17: ****************************** (14)
18: ******************************************************** (24)
19: ********************************************************************** (30)
20: ********************************************** (21)
21: ******************************************** (19)
22:
23: ************************************** (17)
============================================================
Server:    23.218.114.211     United States
============================================================
          Client    Port                          Port Description  Type
    192.168.0.10     443                 HTTP protocol over TLS/SSL   TCP

HR
00:
01:
02:
03:
04: ************************************************************************************************ (66)
05: ******************************************************** (39)
06: ************************************************************** (42)
07:
08:
09:
10:
11:
12:
13:
14:
15:
16:
17:
18:
19:
20:
21: ************************************************************************************************** (70)
22:
23:
```

*Ж қосымшасының жалғасы*

```
==========================================================
Server:     23.218.121.55     United States
==========================================================
          Client     Port                        Port Description  Type
    192.168.0.10      80                      World Wide Web HTTP   TCP

HR
00:
01:
02:
03: ****************************************************** (107)
04: ********************************** (68)
05:
06:
07:
08:
09:
10:
11:
12:
13:
14:
15: ********************************** (66)
16:
17:
18:
19:
20:
21: ************************************************************************************************************ (192)
22:
23:
```

# И қосымшасы

## P2NMAP-Anaysis.py бағдарламасының коды

```
#
# P2NMAP-Analyze.py скрипті
#
# Алдында басып алынған .ip dect файлдарды талдау
#

import argparse             # Python стандартты кітапханасы - командалық жол параметрлері мен
                            # аргументтері үшін синтаксистік талдау
import os                   # операциялық жүйенің функциялары, яғни файлдық енгізу / шығару
import datetime             # Python стандартты кітапханасының күн мен уақыт әдістері
import pickle               # Python стандартты кітапханасының таңдау әдістері
import socket               # өңделмеген сокеттер үшін пайдаланылатын желілік интерфейс кітапханасы
import sys                  # Python стандартты кітапханасының жүйелік модулі

# Бөгде кітапханалар
import pygeoip              # Geo IP Lookup
                            # командалық жолдан geoip орнату үшін: pip install pygeoip
                            # Бұл кітапхана Maxmind жасаған GeoLite деректерін қамтиды
                            # <a href="http://www.maxmind.com">http://www.maxmind.com</a>

# import matplotlib.pyplot as plt  # Сыртқы баспа кітапханасын енгізу

# Жалған константаларды анықтау

SERVER     = 0  # Сервер кілтінің индексі
CLIENT     = 1  # Клиент кілтінің индексі
PORT       = 2  # Порт кілтінің индексі
TYPE       = 3  # Келіп түскен трафик түрінің кілт индексі ("TCP" or "UDP")
HOST_NAME  = 0  # Хост аты индексі gethostbyaddr-дан келеді

# Мәзірді таңдау арқылы орнатылғанына назар аударыңыз

HOST_LOOKUP    = False  # gethostbyaddr() хост атын алады
COUNTRY_LOOKUP = False  # Ел атауы IP-ға байланысты болады
PRINT_STDOUT   = True   # Егер True болса, барлық шығыс деректері мен мәзір тармақтары
                        # Егер False болса, барлық қорытынды мәзірден басқа файлға жіберіледі

OUT            = sys.stdout        # Негізгі шығыс

OSOB_LOADED    = False             # Бақыланатын ОЖ жүктелген жалауы

#
# Елді іздеу
#

def GetCountry(ipAddr):

    # геолокация деректерін келесі сайттан жүктеу: http://dev.maxmind.com/geoip/legacy/geolite/
    gi = pygeoip.GeoIP('geo.dat')
    return gi.country_name_by_addr(ipAddr)

# GetCountry функциясын аяқтау

#
# Атау: ValFileRead
#
# Сипаттама: Файлдың бар-жоғын және оған қолжетімділігін тексеретін функция
#            Аргументтерді тексеру үшін ғана қолданылады
#
# Кіріс: файлға баратын жол
#
# Әрекеттер:
#              егер сәйкес болса, каталог жолы қайтарылады
#
#              егер ол сәйкес болса, онда ArgumentTypeError argparse
#              арқылы шақырып, пайдаланушыға қайтарылады
#

def ValFileRead(theFile):

    # Файл жолын тексеру
    if not os.path.exists(theFile):
        raise argparse.ArgumentTypeError('File does not exist')

    # Жолдың оқылуын растау
    if os.access(theFile, os.R_OK):
        return theFile
    else:
        raise argparse.ArgumentTypeError('File is not readable')

# ValFileRead соңы ==================================

#
# Портты іздеу класы
#
class PortsClass:

    # Құрылым

```

```
90      def __init__(self, portTextFile):
91
92          # Нысана атрибуттары
93          self.portDictionary = {}
94
95          # PortList мәтіндік файлын ашу
96          with open(portTextFile, 'r') as infile:
97
98              # EachLine өңдеу
99              for nextLine in infile:
100
101                 lineList = nextLine.split()
102                 # Бізде дұрыс енгізу жолы бар екеніне көз жеткізіңіз
103
104                 if len(lineList) >= 3:
105                     # Жолды бөлікке бөлу
106
107                     # lineList[0] == PortType (TCP or UDP)
108                     # lineList[1] == PortNumber
109
110                     # Түрі мен порттан кейін қанша бөлшектер бар екенін анықтаңыз
111
112                     # portDescList = lineList[2:]
113                     portDesc = ' '.join(lineList[2:])
114
115                     # Енді сөздіктің енгізілуін жасау
116                     # key = Port,Type
117
118                     # Value = Description
119
120                     self.portDictionary[(lineList[1], lineList[0])] = portDesc
121                 else:
122                     # Бұл жолды тастап кету
123                     continue
124
125     def Lookup(self, portNumber, portType):
126
127         try:
128             portDesc = self.portDictionary[str(portNumber),portType]
129         except:
130             portDesc = "Unknown"
131
132         return portDesc
133
134 # PortsClass анықталуын аяқтау
135
136
137 #
138 # Класс: IPObservationDictionary
139 #
140 # Сипаттамасы: IP бақылаумен байланысты барлық
141 #              әдістер мен қасиеттерді өңдейді
142 #
143 #
144
145 class IPObservationDictionary:
146
147     # Құрылым
148
149     def __init__(self):
150
151         # Нысан атрибуттары
152
153         # IP IP бақылауларды сақтауға арналған сөздік
154         self.Dictionary = {}
155         self.observationsLoaded  = False
156         self.observationFileName = ""
157
158         # PortsClass нысананы жасау
159         # Порттардың сипаттамасын іздеу үшін
160         # пайдалануға болатын нысананы жасау
161         #
162
163         self.portOB = PortsClass("PortList.txt")
164
165     # Бақылауды қосу әдісі
166
167     def AddOb(self, key):
168
169         # Ағымдағы сағатты алу
```

```
170
171         now = datetime.datetime.now()
172         hour = now.hour
173
174         # Сөздікте кілт бар ма екенін тексеріңіз
175
176         if key in self.Dictionary:
177
178             # Егер иә болса, ағымдағы мәнін алу
179             curValue = self.Dictionary[key]
180
181             # Ағымдағы сағат үшін есептелуді ұлғайту
182             curValue[hour-1] = curValue[hour-1] + 1
183
184             # Осы кілтпен байланысты мәнді жаңарту
185             self.Dictionary[key] = curValue
186
187         else:
188             # егер кілт әлі жоқ болса,
189             # онда жаңасын жасау керек
190
191             curValue = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
192
193             # Ағымдағы сағат үшін есептелуді ұлғайту
194             curValue[hour-1] = curValue[hour-1] + 1
195
196             self.Dictionary[key] = curValue
197
198     # Бақылауды алу әдіс
199     # Егер бақылау табылмаса, None-ді қайтару керек
200
201     def GetOb(self, key):
202
203         if key in self.Dictionary:
204             curValue = self.Dictionary[key]
205             return curValue
206         else:
207             return None
208
209     # Сөздіктің мазмұнын басып шығару
210
211     def PrintOb(self):
212         print >> OUT, "\nIP Observations"
213         print >> OUT, "Unique Combinations:    ", str(len(self.Dictionary))
214         print >> OUT
215
216         # Тақырыбты басып шығару
217
218         print >> OUT,'                                                  ',
219         print >> OUT,"|-----------------------------------------  Hourly Observations  ------------------------------
220         print >> OUT,'%16s' % "Server",
221         print >> OUT,'%16s' % "Client",
222         print >> OUT,'%7s'  % "Port",
223         print >> OUT,'%5s'  % "Type"
224         print >> OUT,'-----------------------------------------------------------------------------------------------'
225         print >> OUT,'                                                  ',
226         for i in range(0, 24):
227             print >> OUT,' ',
228             print >> OUT,'%02d' % i,
229         print >> OUT
230
231         # Мазмұнды басып шығару
232         for keys, values in self.Dictionary.items():
233
234             print >> OUT,'%16s' % keys[SERVER],
235             print >> OUT,'%16s' % keys[CLIENT],
236             print >> OUT,'%7s'  % str(keys[PORT]),
237             print >> OUT,'%5s'  % keys[TYPE],
238
239             for i in range(0, 24):
240                 print >> OUT,'%4s' % str(values[i]),
241
242             print >> OUT
243
244         print >> OUT, "\nEnd Print Observations\n"
245
246     def Histogram(self, observations):
247         """
248         Histogram data to stout
249         """
250         largest = max(observations)
251         scale = 100. / largest
252         for hr, datum in enumerate(observations):
253             bar = "*" * int(datum * scale)
```

```
254             if bar == "" and datum > 0:
255                 bar = "*"
256                 print >> OUT, "%02d: %s (%d)" % (hr, bar, datum)
257             elif datum != 0:
258                 print >> OUT, "%02d: %s (%d)" % (hr, bar, datum)
259             else:
260                 print >> OUT, "%02d:" % hr
261         print >> OUT, "\n"
262
263     #
264     # PrintUniqueServer тізімі
265     #
266     # әрбір сервердің стандартты IP мекенжайын басып шығару әдісі
267     # Опцияларға кіреді: lookupHost және lookupCountry
268     # Егер таңдалған болса, олар тиісті іздеулерді орындап,
269     # алынған деректерді хабарлайтын болады
270     #
271
272     def PrintServers(self):
273
274         print >> OUT, "\nUnique Server List\n"
275         print >> OUT, '-----------------------------------------------------------------------------------------------
276         # Сөздіктен серверлер IP мекенжайларының
277         # "терілуін" құру
278
279         self.servers = set()
280         for keys,values in self.Dictionary.items():
281             self.servers.add(keys[SERVER])
282
283         # Тізімге түрлендіру және сұрыптау
284         # Бұл әдіс бірегей сұрыпталған тізімді қамтамасыз етеді
285
286         serverList = list(self.servers)
287         serverList.sort()
288
289         # Сұрыпталған тізімде сервердің әрбір IP мекенжайын өңдеу
290
291         for serverIP in serverList:
292
293             # егер ел бойынша іздеу таңдалса,
294             # іздеуді орындау керек, әйтпесе ел өрісін бос қалдыру қажет
295             if COUNTRY_LOOKUP:
296                 countryName = GetCountry(serverIP)
297             else:
298                 countryName = ""
299
300             # Желілік қате жағдайында Try / Except Loop орнату
301
302             try:
303                 # егер шақырушы тарапынан хост атын іздеуді сұранысы
304                 # түссе, онда іздеуді орындау қажет, әйтпесе өрісті бос қалдыру керек
305                 if HOST_LOOKUP:
306                     hostName = socket.gethostbyaddr(serverIP)
307                 else:
308                     hostName = ["", "", ""]
309             except:
310                 hostName = ""
311                 pass
312
313             # Қалыптасқан нәтижелерді басып шығару
314             print >> OUT,' $15s ' % serverIP,
315             print >> OUT,' $15s ' % countryName,
316             print >> OUT,' $60s ' % hostName[HOST_NAME]
317
318             self.ports = set()
319
320             for keys,values in self.Dictionary.items():
321                 if keys[SERVER] == serverIP:
322                     self.ports.add( (keys[PORT], keys[TYPE]) )
323
324             portList = list(self.ports)
325             portList.sort()
326
327             for port in portList:
328                 print >> OUT,' %27s ' % str(port[0]),
329                 print >> OUT,' %5s ' % port[1],
330                 print >> OUT,'%40s' % self.portOB.Lookup(port[0],port[1])
331             print >> OUT, '-----------------------------------------------------------------------------------------------
332
333         print >> OUT
334         print >> OUT, "End Print Servers\n"
335         print >> OUT, "\n\n"
336
337     # PrintUniqueServer тізімінің соңы
338
339     #
340     # Серверлердің толық тізімін басып шығару
341     #
```

```
342     # Стандартты түрде басып шығару әдісі
343     # Бірегей сервер / клиент өзара әрекеті
344     #
345
346     def PrintServerDetails(self):
347
348         # Сөздіктен серверлер IP мекенжайларының
349         # "терілуін" құру
350         print >> OUT, "\nUnique Server Client Connection List\n"
351         print >> OUT, '-----------------------------------------------------------------------------------------------'
352
353
354         self.servers = set()
355
356         for keys,values in self.Dictionary.items():
357             self.servers.add(keys[SERVER])
358             # Тізімге түрлендіру және сұрыптау
359             # Бұл әдіс бірегей сұрыпталған тізімді қамтамасыз етеді
360
361         # Енді бірегей серверлердің сұрыпталған тізімін жасау керек
362         serverList = list(self.servers)
363         serverList.sort()
364
365         # Енді серверге сәйкес келетін барлық қосылымдарды табу
366         # және қосылым туралы ақпарат беру үшін
367         # серверлер тізімі бойынша іздеуді орындау қажет.
368
369         for serverIP in serverList:
370
371             # егер ел бойынша іздеу таңдалса,
372             # іздеуді орындау қажет, әйтпесе ел өрісін бос қалдыру керек
373             if COUNTRY_LOOKUP:
374                 countryName = GetCountry(serverIP)
375             else:
376                 countryName = ""
377
378             # Желілік қате жағдайында Try / Except Loop орнату.
379             try:
380                 # егер шақырушы тарапынан хост атын іздеуді сұранысы
381                 # түссе, іздеуді орындау қажет, әйтпесе өрісті бос қалдыру керек
382                 if HOST_LOOKUP:
383                     hostName = socket.gethostbyaddr(serverIP)
384                 else:
385                     hostName = ["", "", ""]
386             except:
387                 hostName = ""
388                 continue
389
390             # Қалыптасқан нәтижелерді басып шығару
391             print >> OUT,"\n=============================================================================================
392             print >> OUT,"Server: ",
393             print >> OUT,' %15s ' % serverIP,
394             print >> OUT,' %15s ' % countryName,
395             print >> OUT,' %60s ' % hostName[HOST_NAME]
396             print >> OUT,"=============================================================================================
397             print >> OUT,'%16s' % "Client",
398             print >> OUT,'%7s'  % "Port",
399             print >> OUT,'%40s' % "Port Description",
400             print >> OUT,'%5s'  % "Type",
401             print >> OUT
402
403             for keys,values in self.Dictionary.items():
404
405                 # Егер сервер сәйкестендірілсе
406                 # мәліметтерді басып шығару керек:
407
408                 if keys[SERVER] == serverIP:
409                     print >> OUT,'%16s' % keys[CLIENT],
410                     print >> OUT,'%7s'  % str(keys[PORT]),
411                     print >> OUT,'%40s' % self.portOB.Lookup(keys[PORT],keys[TYPE]),
412                     print >> OUT,'%5s'  % keys[TYPE]
413         print >> OUT
414         print >> OUT, "End Print Server Details\n"
415
416     # PrintUniqueServer тізімінің соңы
417
418     #
419     # Алынған трафиктің гистограммасын шығару
420     #
421     # әрбір жазу үшін гистограмманы
422     # басып шығару әдісі
423     #
424
425     def PrintHistogram(self):
426
427         # Сөздіктен серверлер IP мекенжайларының
428         # "терілуін" құру
429
430         print >> OUT,"\nHourly Histogram\n"
431
432         self.servers = set()
```

```
433
434         for keys, values in self.Dictionary.items():
435             self.servers.add(keys[SERVER])
436             # Тізімге түрлендіру және сұрыптау
437             # Бұл әдіс бірегей сұрыпталған тізімді қамтамасыз етеді
438
439         # Енді бірегей серверлердің сұрыпталған тізімін жасау керек
440         serverList = list(self.servers)
441         serverList.sort()
442
443         # Енді серверге сәйкес келетін барлық қосылымдарды табу
444         # және қосылым туралы ақпарат беру үшін
445         # серверлер тізімі бойынша іздеуді орындау қажет.
446
447         for serverIP in serverList:
448
449             # егер ел бойынша іздеу таңдалса,
450             # іздеуді орындау қажет, әйтпесе ел өрісін бос қалдыру керек
451
452             if COUNTRY_LOOKUP:
453                 countryName = GetCountry(serverIP)
454             else:
455                 countryName = ""
456
457             # Желілік қате жағдайында Try / Except Loop орнату.
458
459             try:
460                 # егер шақырушы тарапынан хост атын іздеуді сұранысы
461                 # түссе, іздеуді орындау қажет, әйтпесе өрісті бос қалдыру керек
462                 if HOST_LOOKUP:
463                     hostName = socket.gethostbyaddr(serverIP)
464                 else:
465                     hostName = ["", "", ""]
466             except:
467                 hostName = ["", "", ""]
468                 continue
469
470             # Қалыптасқан нәтижелерді басып шығару
471             print >> OUT,"\n==============================================================================="
472             print >> OUT,"Server: ",
473             print >> OUT,' %15s ' % serverIP,
474             print >> OUT,' %15s ' % countryName,
475             print >> OUT,' %60s ' % hostName[HOST_NAME]
476             print >> OUT,"==============================================================================="
477
478             for keys, values in self.Dictionary.items():
479
480                 # Егер сервер сәйкестендірілсе,
481                 # гистограмманы басып шығару керек
482
483                 if keys[SERVER] == serverIP:
484                     print >> OUT,'%16s' % "Client",
485                     print >> OUT,'%7s' % "Port",
486                     print >> OUT,'%40s' % "Port Description",
487                     print >> OUT,'%5s' % "Type",
488                     print >> OUT
489                     print >> OUT,'%16s' % keys[CLIENT],
490                     print >> OUT,'%7s' % str(keys[PORT]),
491                     print >> OUT,'%40s' % self.portOB.Lookup(keys[PORT], keys[TYPE]),
492                     print >> OUT,'%5s' % keys[TYPE]
493                     print >> OUT
494                     print >> OUT,"HR "
495                     self.Histogram(values)
496         print >> OUT
497         print >> OUT, "End Print Histogram\n"
498
499     # Histogram шығысының соңы
500
501     #
502     # Клиенттердің бірегей тізімін басып шығару
503     #
504     # әр IP клиентін басып шығару тәсілі.
505     # Опциялар: lookupHost және lookupCountry
506     # Егер таңдалған болса, олар тиісті іздеулерді орындап,
507     # алынған деректерді хабарлайтын болады
508
509     def PrintClients(self):
510
511         print >> OUT,"\nUnique Client List\n"
512
513         self.clients = set()
514         for keys, values in self.Dictionary.items():
515             self.clients.add(keys[1])
516
517         clientList = list(self.clients)
518         clientList.sort()
519
520         # Сұрыпталған тізімде сервердің әрбір IP мекенжайын өңдеу
521
522         for clientIP in clientList:
```

```
523
524             # егер ел бойынша іздеу таңдалса,
525             # іздеуді орындау қажет, әйтпесе ел өрісін бос қалдыру керек
526             if COUNTRY_LOOKUP:
527                 countryName = GetCountry(clientIP)
528             else:
529                 countryName = ""
530
531             # Желілік қате жағдайында Try / Except Loop орнату.
532             try:
533                 # егер шақырушы тарапынан хост атын іздеуді сұранысы
534                 # түссе, іздеуді орындау қажет, әйтпесе өрісті бос қалдыру керек
535                 if HOST_LOOKUP:
536                     hostName = socket.gethostbyaddr(clientIP)
537                 else:
538                     hostName = ["", "", ""]
539             except:
540                 hostName = ["", "", ""]
541             pass
542
543         # Қалыптасқан нәтижелерді басып шығару
544
545         print >> OUT,' %15s ' % clientIP,
546         print >> OUT,' %15s ' % countryName,
547         print >> OUT,' %60s ' % hostName[HOST_NAME]
548
549     print >> OUT, "\nEnd Print Client List\n"
550
551     # PrintUniqueClient тізімінің соңы
552
553
554     # Ағымдағы бақылау сөздігін
555     # көрсетілген файлда сақтау
556
557     def SaveOb(self, fileName):
558
559         with open(fileName, 'wb') as fp:
560             pickle.dump(self.Dictionary, fp)
561
562     # Көрсетілген файлдан бақылау
563     # сөздігін жүктеу және оған жазу
564
565     def LoadOb(self, fileName):
566         try:
567             with open(fileName, 'rb') as fp:
568                 self.Dictionary = pickle.loads(fp.read())
569                 self.observationFileName = fileName
570                 self.observationsLoaded  = True
571
572         except:
573             print("Loading Observation - Failed")
574             self.observationsLoaded  = False
575             self.observationFileName = fileName
576
577     def PrintIPAnalysisMenu(self):
578
579         print("========== P2NMAP Analysis Menu ==========\n")
580
581         if self.observationsLoaded:
582             print("Current Observation File: ", self.observationFileName)
583             print()
584
585         print("[L]    Load Observation File for Analysis")
586
587         if self.observationsLoaded:
588             if PRINT_STDOUT:
589                 print("[O]    Direct Output to File    (Current = Stdout)")
590             else:
591                 print("[O]    Direct Output to Stdout  (Current = results.txt)")
592
593             if HOST_LOOKUP:
594                 print("[H]    Turn Off Host Lookup     (Current = Host Lookup On)")
595             else:
596                 print("[H]    Turn On Host Lookup      (Current = Host Lookup Off)")
597
598             if COUNTRY_LOOKUP:
599                 print("[C]    Turn Off Country Lookup  (Current = Country Lookup On)")
600             else:
601                 print("[C]    Turn On Country Lookup   (Current = Country Lookup Off)")
602
603             print("==============================================================")
604             print("[1]     Print Observations      (ALL)")
605             print("[2]     Print Servers           (Unique)")
606             print("[3]     Print Clients           (Unique)")
607             print("[4]     Print Connections       (Unique by Server)")
608             print("[5]     Print Histogram")
609             print()
610         print("[X]    Exit P2NMAP Analysis")
611         print()
```

```
612
613         # Құрылымды / Нысананы жою
614
615         def __del__(self):
616             if VERBOSE:
617                 print >> OUT,"Closed"
618
619     # IPObservationClass соңы ==========================================
620
621
622     # НЕГІЗГІ БАҒДАРЛАМА ОСЫ ЖЕРДЕН БАСТАЛАДЫ
623     #==============================================
624
625     if __name__ == '__main__':
626
627         # VERBOSE-ті True болып орнату
628         VERBOSE = True
629
630         # Ip бақылау нысанын жасау
631
632         ipOB = IPObservationDictionary()
633
634         while True:
635
636             ipOB.PrintIPAnalysisMenu()
637
638             menuSelection = raw_input("Enter Selection: ").upper()
639             if menuSelection == 'L':
640                 fileName = raw_input("Enter IP Capture File: ")
641                 ipOB.LoadOb(fileName)
642                 print()
643
644             elif menuSelection == 'O':
645                 if PRINT_STDOUT:
646                     PRINT_STDOUT = False
647                     OUT = open("results.txt", 'w+')
648                 else:
649                     PRINT_STDOUT = True
650                     OUT.close()
651                     OUT = sys.stdout
652             elif menuSelection == 'H':
653                 if HOST_LOOKUP:
654                     HOST_LOOKUP = False
655                 else:
656                     HOST_LOOKUP = True
657
658             elif menuSelection == 'C':
659                 if COUNTRY_LOOKUP:
660                     COUNTRY_LOOKUP = False
661                 else:
662                     COUNTRY_LOOKUP = True
663             elif menuSelection == '1':
664                 ipOB.PrintOb()
665             elif menuSelection == '2':
666                 ipOB.PrintServers()
667             elif menuSelection == '3':
668                 ipOB.PrintClients()
669             elif menuSelection == '4':
670                 ipOB.PrintServerDetails()
671             elif menuSelection == '5':
672                 ipOB.PrintHistogram()
673             elif menuSelection == 'X':
674                 break
675             else:
676                 print("Entry not recognized")
677                 continue
678
679             OUT.flush()
680
681     print >> OUT, "End P2NMAP"
```

## К қосымшасы

## Материалдық ресурстардың шығындары

К.1 кесте – Материалдық ресурстардың шығындары

| Материалдық ресурстың атауы | Саны | Бірлік үшін бағасы, тг | Сомасы, тг |
|---|---|---|---|
| Сервер Dell PowerEdge R730 210-ACXU-350 | 1 | 1729100 | 1729100 |
| Сервер Dell PowerEdge R630 210-ACXS-227 | 1 | 1675100 | 1675100 |
| Патч-панель SHIP P197-24M (1U, FTP) | 1 | 19450 | 19450 |
| Кабель ұйымдастырушысы Toten SA.2001.0000 | 2 | 7300 | 14600 |
| Желдеткіш блогы Toten SA.3004.0301 | 1 | 40161 | 40161 |
| Орнату рельсі CyberPower 4POSTRAILKIT1832 | 1 | 13796 | 13769 |
| Желілік фильтр SHIP 700508102 | 2 | 9704 | 19408 |
| Коммутатор D-Link DES-1026G/E1A | 1 | 36900 | 36900 |
| Бақылау құрылғы UniPing server solution v3/SMS | 1 | 120386 | 120386 |
| Температура хабаршысы (датчик) NetPing 1-wire, THS | 1 | 7348 | 7348 |
| Ылғалдылық хабаршысы (датчик) NetPing 1-wire, HS | 1 | 20685 | 20685 |
| Түтін хабаршысы (датчик) NetPing ИП212-141 | 1 | 6259 | 6259 |
| Электр қуат хабаршысы (датчик) NetPing 995S1 | 1 | 7016 | 7016 |
| Электр қуат хабаршысы (датчик) NetPing 1-wire 910S20 | 1 | 12839 | 12839 |
| Қозғалыс хабаршысы (датчик) NetPing SWAN-QUAD ИК | 1 | 8689 | 8689 |
| Соққы хабаршысы (датчик) NetPing PI-99D | 1 | 5990 | 5990 |

*К қосымшасының жалғасы*

*К.1 кестенің жалғасы*

| Материалдық ресурстың атауы | Саны | Бірлік үшін бағасы, тг | Сомасы, тг |
|---|---|---|---|
| Сирена NetPing AC-10 | 1 | 4981 | 4981 |
| Патч-корд Cablexpert PP12-2M/B | 10 | 309 | 3090 |
| Патч-корд Cablexpert PP12-10M | 10 | 913 | 9130 |
| Патч-корд Cablexpert PP12-20M | 5 | 1750 | 8750 |
| IP видеокамера HIKVISION DS-2CD2041GO-I | 1 | 28900 | 28900 |
| UPS CyberPower OLS3000ERT2U | 1 | 341466 | 341466 |
| Серверлік шкаф Toten A2.6842.8101 | 1 | 162000 | 162000 |
| Wi-Fi Модем TP-Link TD-W8961N-RU | 1 | 11100 | 11100 |
| Моноблок Acer AspireC22-865 | 5 | 253990 | 1269950 |
| Пернетақта+тышқан Logitech Wireless Combo MK220 | 5 | 9990 | 49950 |
| Үстел ZETA Октавия 1400x600x750 | 5 | 12350 | 61750 |
| Орындық ZETA Сальса | 5 | 13600 | 68000 |
| Мұрағат сөресі 2000x1500x600 | 1 | 8592 | 8592 |
| Шкаф ZETA КУЛ ШО-2 | 1 | 18550 | 18550 |
| Материалдық ресурстар шығындарының барлығы | | | 5783909 |

## Л қосымшасы

## Негізгі қорлардың амортизациясы

Л.1 кесте – Негізгі қорлардың амортизациясы

| Жабдықтың атауы | Жабдықтың бағасы, тг | Бір жылдық амортизация нормасы, % | Жабдықтың жұмыс уақытының тиімді қоры, сағ/жыл | Жабдықтың ҒЗЖ кезінде жұмыс істеу уақыты, сағ | Сомасы, тг |
|---|---|---|---|---|---|
| Сервер Dell PowerEdge R730 210-ACXU-350 | 1729100 | 20 | 8760 | 2350 | 92772 |
| Сервер Dell PowerEdge R630 210-ACXS-227 | 1675100 | 20 | 8760 | 2350 | 89874 |
| Патч-панель SHIP P197-24M (1U, FTP) | 19450 | 10 | 8760 | 2350 | 522 |
| Желдеткіш блогы Toten SA.3004.0301 | 40161 | 33,3 | 8760 | 2350 | 3588 |
| Желілік фильтр SHIP 700508102 | 9704 | 10 | 8760 | 2350 | 261 |
| Коммутатор D-Link DES-1026G/E1A | 36900 | 20 | 8760 | 2350 | 1980 |
| Бақылау құрылғы UniPing server solution v3/SMS | 120386 | 20 | 8760 | 2350 | 6459 |
| Температура хабаршысы (датчик) NetPing 1-wire, THS | 7348 | 14,29 | 8760 | 2350 | 282 |

*Л қосымшасының жалғасы*

*Л.1 кестенің жалғасы*

| Жабдықтың атауы | Жабдықтың бағасы, тг | Бір жылдық амортизация нормасы, % | Жабдықтың жұмыс уақытының тиімді қоры, сағ/жыл | Жабдықтың ҒЗЖ кезінде жұмыс істеу уақыты, сағ | Сомасы, тг |
|---|---|---|---|---|---|
| Ылғалдылық хабаршысы (датчик) NetPing 1-wire, HS | 20685 | 14,29 | 8760 | 2350 | 793 |
| Түтін хабаршысы (датчик) NetPing ИП212-141 | 6259 | 14,29 | 8760 | 2350 | 240 |
| Электр қуат хабаршысы (датчик) NetPing 995S1 | 7016 | 14,29 | 8760 | 2350 | 269 |
| Электр қуат хабаршысы (датчик) NetPing 1-wire 910S20 | 12839 | 14,29 | 8760 | 2350 | 492 |
| Қозғалыс хабаршысы (датчик) NetPing SWAN-QUAD ИК | 8689 | 14,29 | 8760 | 2350 | 333 |
| Соққы хабаршысы NetPing PI-99D | 5990 | 14,29 | 8760 | 2350 | 230 |
| Сирена NetPing AC-10 | 4981 | 14,29 | 8760 | 2350 | 191 |

*Л қосымшасының жалғасы*

*Л.1 кестенің жалғасы*

| Жабдықтың атауы | Жабдықтың бағасы, тг | Бір жылдық амортизация нормасы, % | Жабдықтың жұмыс уақытының тиімді қоры, сағ/жыл | Жабдықтың ҒЗЖ кезінде жұмыс істеу уақыты, сағ | Сомасы, тг |
|---|---|---|---|---|---|
| Патч-корд Cablexpert PP12-2M/B | 309 | 10 | 8760 | 2350 | 9 |
| Патч-корд Cablexpert PP12-10M | 913 | 10 | 8760 | 2350 | 25 |
| Патч-корд Cablexpert PP12-20M | 1750 | 10 | 8760 | 2350 | 47 |
| IP видеокамера HIKVISION DS-2CD2041GO-I | 28900 | 16,7 | 8760 | 2350 | 1295 |
| UPS CyberPower OLS3000ERT 2U | 341466 | 10 | 8760 | 2350 | 9161 |
| Wi-Fi Модем TP-Link TD-W8961N-RU | 11100 | 20 | 8760 | 2350 | 596 |
| Моноблок Acer AspireC22-865 | 253990 | 20 | 8760 | 2350 | 13628 |
| Пернетақта+тышқан Logitech Wireless Combo MK220 | 9990 | 10 | 8760 | 2350 | 268 |
| Негізгі қорлар амортизациясының барлығы | | | | | 223315 |