МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Фучило (Бабій) Юлія Степанівна

Система фільтрацію контенту на базі cisco і FreeBSD / Content filtering system based on cisco and FreeBSD

спеціальність: 123 – Комп'ютерна інженерія
освітньо-професійна програма – Комп'ютерна інженерія

Кваліфікаційна робота

Виконала: студентка групи КІ-42
Фучило (Бабій) Юлія Степанівна
_____

Науковий керівник
І.В. Ігнатєв

Кваліфікаційну роботу
Допущено до захисту
«___» _____ 20 ___ р.

Завідувач кафедри
_____ О.М. Березький

ТЕРНОПІЛЬ - 2021

РЕЗЮМЕ

Кваліфікаційна робота містить 83 сторінок пояснюючої записки, 18 рисунків, 8 таблиць, 3 додатки.

Метою кваліфікаційної роботи є розробка програмних налаштувань для системи фільтрації контенту на основі Cisco та FreeBSD.

Методи досліджень базуються на теорії алгоритмів (для аналізу розроблених методів та алгоритмів), алгоритмах теорії графів (для обробки млок-схем у вигляді зв'язних графів), технологій об'єктно-орієнтованого програмування (для програмної реалізації спроектованої структури програмного додатку).

В кваліфікаційній роботі на основі аналізу існуючих алгоритмів обробки графів та вимог до створення конструкторської документації розроблений програмний додаток створення, редагування та візуалізації блок-схем алгоритмів довільної складності.

В кваліфікаційній роботі на основі аналізу функцій Cisco Catalyst проведено вибір необхідного мережевого обладнання для побудови мережі. Проведено аналіз функцій передньої панелі маршрутизатора для виконання необхідних налаштувань та встановлення обладнання. Проаналізовано аналіз налаштування портів комутатора Catalyst.

Проведено налаштування фаєрволу засобами FreeBSD, а саме налаштування маршрутизації пакетів, використання макросів та списків. Проаналізовано розширені можливості пошуку та фільтрації пакетів для налаштування контролю доступу. Виконано налаштування якорів та робота з декількома провайдерами.

Проведено тестування розроблених програмних налаштувань фаєрволу FreeBSD, що підтвердило ефективність розроблених налаштувань.

Ключові слова: CISCO CATALYST, FREEBSD, ЯКОРЯ, ФІЛЬТРАЦІЯ ПАКЕТІВ.

RESUME

Qualification work contains 83 pages of explanatory note, 18 figures, 8 tables, 3 appendices.

The purpose of the qualification work is to develop software settings for the content filtering system based on Cisco and FreeBSD.

Research methods are based on algorithm theory (for analysis of developed methods and algorithms), graph theory algorithms (for processing block diagrams in the form of connected graphs), object-oriented programming technologies (for software implementation of the designed structure of the software application).

In the qualification work on the basis of the analysis of existing algorithms of processing of graphs and requirements to creation of the design documentation the software application of creation, editing and visualization of block diagrams of algorithms of arbitrary complexity is developed.

In the qualification work, based on the analysis of Cisco Catalyst functions, the necessary network equipment for network construction was selected. An analysis of the functions of the front panel of the router to perform the necessary settings and installation of equipment. The analysis of Catalyst switch port settings is analyzed.

The firewall was configured using FreeBSD tools, namely the configuration of packet routing, use of macros and lists. Advanced packet search and filtering capabilities for access control setup are analyzed. Anchors have been set up and work with several providers.

The developed software settings of the FreeBSD firewall were tested, which confirmed the effectiveness of the developed settings.

Keywords: CISCO CATALYST, FREEBSD, ANCHORS, PACKAGE FILTRATION.

ЗМІСТ

					КР.КІ. 07157/19.00.00.000 ПЗ		
Змн.	Лист	№ докум.	Підпис	Дата			
Розробив		Фучило Ю.С.			СИСТЕМА ФІЛЬТРАЦІЮ КОНТЕНТУ НА БАЗІ CISCO І FREEBSD	Літ.	Арк. 8 / Акрушів 83
Перевір.		Ігнатєв І.В.					
Консульт.		Савка Н.Я.				ЗУНУ,ФКІТ, КІ-42	
Н. Контр.		Мельник Г.М.					
Затвердив		Березький О.М.					

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

HTTP	–	Протокол передачі гіпертексту.
WCCP		Web Cache Communication Protocol.
SFP	–	Small Form-factor Pluggable.
ACL	–	Список контролю доступу.
PF	–	Personal Firewall

ВСТУП

При побудові захищених корпоративних мереж, як і використанні інтернет речей, важливою задачею є фільтрування вхідного та вихідного трафіку. Одним з складних видів такої фільтрації є фільтрація контенту, що проходить мережевими каналами. Існує безліч систем фільтрації контенту проте вони усі мають складені та персоналізовані налаштування під конкретну установу.

У кваліфікаційній роботі докладно розглянуто один з варіантів фільтрації трафіку за реєстром за допомогою SQUID на базі FreeBSD, зістикований з бекбону CISCO по протоколу WCCP2.

Після детального аналізу теми фільтрації, продуктивності (якщо пустити в фільтр весь http-трафік, то фільтрація повинна здійснюватися на швидкостях понад гігабіта в секунду), була обрана наступна схема:

Трафік HTTP і HTTPS, що виходить з хмари нашої мережі (в прикладі VLAN90) через бордер CISCO, будемо порівнювати з ACL, що містить IP-адреси, витягнуті зі списку. У разі збігу, ми відправляємо трафік по протоколу WCCP (Web Cache Communication Protocol) на сервер (а) SQUID. Дана схема дозволить уникнути аналізу всього трафіку і аналізувати тільки той, який йде на заявлені IP, що не може не позначитися позитивно на загальній продуктивності системи.

З мінусів цієї зв'язки слідує, що припустимо якщо внесений до реєстру ресурс з IP = 3.3.3.3, доменом = bad.xxx і URL = http://www.bad.xxx/1.jpg переїжджає на інший IP, то фільтр працювати не буде. Але з іншого боку актуалізація списків блокування, це не наша проблема. Даний підхід дозволить нам з одного боку, не витрачається на дорогі системи аналізу трафіку, що проходить, з іншого боку виключити "простий" підхід блокування всього ресурсу по IP, а блокувати тільки зазначений в реєстрі контент. Зв'язку з WCCP краще піднімати на приватних адресах, що б не писати зайві ACL в SQUID.

Для реалізації завдання обрано сервер на базі FreeBSD (можна реалізовувати налаштування на будь-якій * NIX машині).

Сервер як правило забезпечує аналіз і фільтрацію контента. Так як системний OpenSSL в FreeBSD не підтримує шифрування ДСТУ, потрібно встановити нову версію операційної системи.

Метою роботи є розробка узагальненої схеми системи фільтрації мережевого трафіку за контентом[1-2].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз функцій Cisco Catalyst

Комутатор Catalyst 2960 (також комутатор) - це комутатор Ethernet, до якого можна підключати робочі станції, точки бездротового доступу Cisco, IP-телефони Cisco і інші мережеві пристрої, в тому числі сервери, маршрутизатори та інші комутатори. Проаналізуємо функціональний опис комутаторів Catalyst 2960.

Комутатори Catalyst 2960 на 24 і 48 портів можуть використовуватися в якості магістральних комутаторів для агрегування Ethernet-трафіку 10BASE-T, 100BASE-TX і 1000BASE-T, одержуваного від інших мережевих пристроїв. Компактні 8-портові комутатори Catalyst 2960 володіють такими ж можливостями підключення Ethernet, але можуть також використовуватися поза традиційним комутаційним середовищем, наприклад в офісах і аудиторіях. Наведемо приклади розгортання програмного забезпечення комутатора. Таблиця 1.1 містить опис характеристик для різних моделей [3].

Таблиця 1.1 Описи моделей комутатора Catalyst 2960

Модель коммутатора	Підтримуваний образ ПЗ	Опис
Catalyst 2960-8TC-S	LAN-Lite	8 портів Ethernet 10 / 100BASE-TX і 1 порт подвійного призначення (1 порт для мідного кабелю 10/100 / 1000BASE-T і 1 роз'єм для плагіна малого формфактору (SFP), без вентилятора або порту RPS)
Catalyst 2960-24-S	LAN-Lite	24 порта Ethernet 10/100BASE-TX (без порта RPS або роз'єму для модуля SFP)
Catalyst 2960-Plus 24TC-S	LAN-Lite	24 порта Ethernet 10/100BASE-TX і 2 порти подвійного призначення (без порту RPS)
Catalyst 2960-24TC-S	LAN-Lite	24 порти Ethernet 10 / 100BASE-TX і 2 порти подвійного призначення (без порту RPS)

Комутатори Catalyst 2960-8TC-S, 2960-8TC-L, 2960G-8TC-L і 2960PD-8TT-L мають менший розмір, ніж інші комутатори Catalyst 2960. Вони можуть кріпитися на магнітах і мають роз'єми для замка безпеки, але не обладнані вентиляторами. Наступні комутатори PoE відповідають вимогам попереднього стандарту PoE Cisco і стандарту IEEE 802.3af:

- Catalyst 2960-24LC-S.
- Catalyst 2960-Plus 24LC-S.
- Catalyst 2960-Plus 24LC-L.
- Catalyst 2960-24LT-L.
- Catalyst 2960-24PC-L.
- Catalyst 2960-Plus 24PC-L.
- Catalyst 2960-24PC-S.
- Catalyst 2960-Plus 24PC-S.
- Catalyst 2960-48PST-L.
- Catalyst 2960-Plus 48PST-L.
- Catalyst 2960-48PST-S.
- Catalyst 2960-Plus 48PST-S.

Комутатори підтримують наступні модулі SFP:

- 1000BASE-CWDM.
- 1000BASE-BX.
- 1000BASE-LX / LH.
- 1000BASE-SX.
- 1000BASE-T.
- 1000BASE-ZX.
- 100BASE-BX.
- 100BASE-FX.
- 100BASE-LX.

Комутатори Catalyst 2960-24PC-L, 2960-Plus 24PC-L, 2960-24PC-S, 2960-Plus 24PC-S, 2960-24LC-S, 2960-Plus 24LC-S, 2960-Plus 24LC-L, 2960- 24TC-L, 2960-Plus 24TC-L, 2960-48TC-L, 2960-Plus 48TC-L, 2960-48PST-L, 2960-Plus

48PST-L, 2960-48PST-S, 2960-Plus 48PST-S, 2960G-24TC-L і 2960G-48TC-L підтримують всі модулі SFP [4].

Комутатори Catalyst 2960-8TC-S, 2960-24TC-S, 2960-Plus 24TC-S, 2960-48TC-S і2960-Plus 48TC-S підтримують тільки модулі 1000BASE-LX / LH, 1000BASE-SX і SFP 100BASE-FX.

Комутатори Catalyst 2960-8TC-L, 2960G-8TC-L і 2960-8TC-S не підтримують модулі SFP 1000BASE-T і GLC-GE-100FX.

Для отримання конкретної інформації про те, які модулі SFP підтримуються тими чи іншими комутаторами, необхідно розглянути таблицю сумісності модулів приймача Cisco Gigabit Ethernet на веб-сайті Cisco.com.

При установці в комутатори Catalyst 2960 модулі SFP 1000BASE-T працюють зі швидкістю 10, 100 або 1000 Мбіт / с в повнодуплексному режимі і 10 або 100 Мбіт / с в напівдуплексному режимі. Порти 10/100 і 10/100/1000 виконують автоузгодження швидкості і підтримують повнодуплексний або напівдуплексний режим.

Деякі комутатори Catalyst 2960 мають роз'єм RPS для опціональних резервних систем живлення Cisco RPS 2300 або Cisco RPS 675, які працюють на змінному струмі і забезпечують комутатор резервним живленням постійного струму.

Наступні моделі комутаторів не мають вбудованого резервного джерела живлення:

- Catalyst 2960-8TC-L.
- Catalyst 2960G-8TC-L.
- Catalyst 2960-8TC-S.
- Catalyst 2960PD-8TT-L.
- Catalyst 2960-24-S.
- Catalyst 2960-24TC-S.
- Catalyst 2960-Plus 24TC-S.
- Catalyst 2960-48TT-S.
- Catalyst 2960-48TC-S.
- Catalyst 2960-Plus 48TC-S.

Комутатори Catalyst 2960-24-S, 2960-Plus 24TC-S, 2960-24TC-S, 2960-Plus 48TC-S, 2960-48TC-S і 2960-48TT-S.

Порти 10/100 комутатора Catalyst 2960-24-S пронумеровані в такий спосіб: перший член пари (порт 1) розташований над другим членом (порт 2), порт 3 розташований над портом 4 і т. д. (Рисунок 1.1).

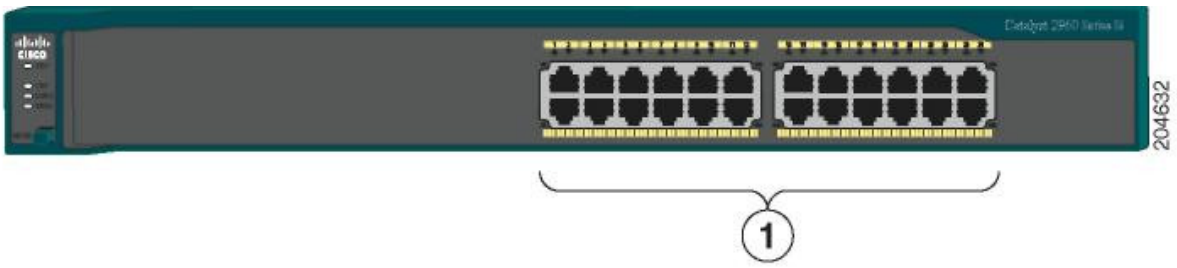

Рисунок 1.1 - Передня панель комутатора Catalyst 2960-24-S

Де – 1: порти зі швидкостями 10/100.

Порти 10/100 на комутаторах Catalyst 2960-Plus 24TC-S, 2960-24TC-S, 2960-Plus 48TC-S і 2960-48TC-S пронумеровані так само, як і на комутаторах серії 2960-24-S. Ці комутатори мають порти подвійного призначення. Порти 10/100/1000 1 і 2 можуть використовуватися для підключення модуля SFP або роз'єму RJ-45, але не допускають їх одночасного підключення. Щоб налаштувати тип роз'єму для цих портів, необхідно використати програмне забезпечення (Рисунок 1.2 і 1.3).

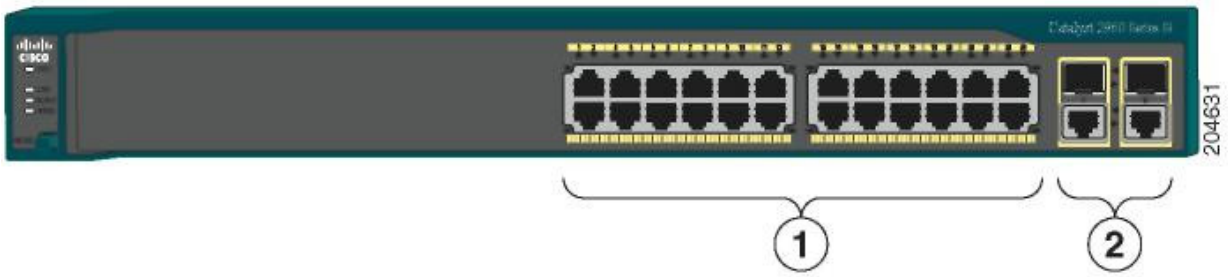

Рисунок 1.2 - Передня панель комутатора Catalyst 2960-Plus 24TC-S 2960-24TC-S

де – 1: порти 10/100 2 Порти подвійного призначення

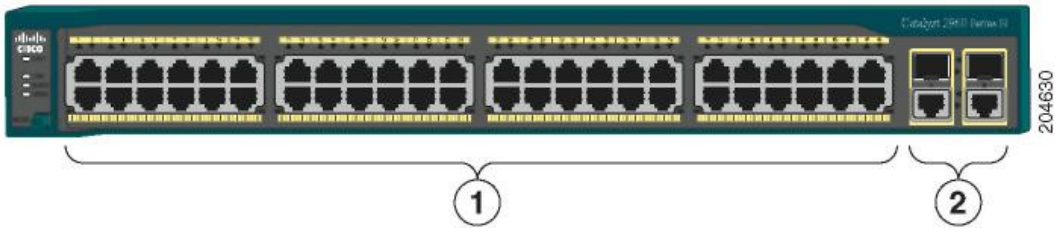

Рисунок 1.3 - Передня панель комутатора Catalyst 2960-Plus 48TC-S і 2960-48TC-S

Де -1: порти 10/100 2 порти подвійного призначення.

Порти 10/100 комутатора Catalyst 2960-48TT-S пронумеровані в такий спосіб: перший член пари (порт 1) розташований над другим членом (порт 2), порт 3 розташований над портом 4 і т. д. Комутатор оснащений 2 портами каскадирования 10/100 / 1000 з нумерацією 1 і 2. Як показано на рисунку 1.4.

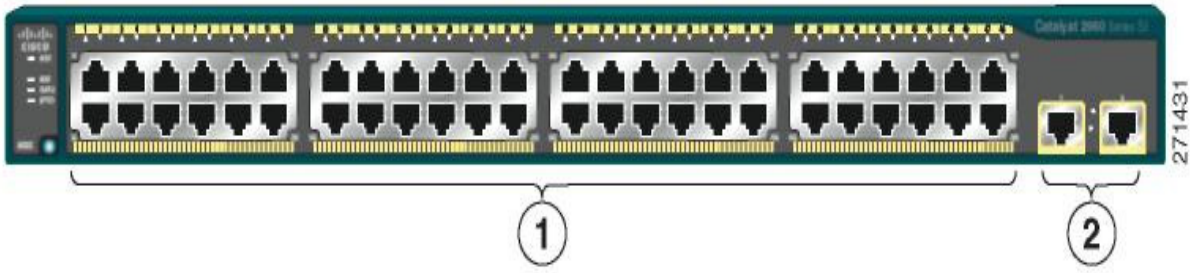

Рисунок 1.4 - Передня панель комутатора Catalyst 2960-48TT-S

Тут 1: порти 10/100, 2: порти 10/100/1000. Комутатори Catalyst 2960-Plus 24PC-L, 2960-24PC-L, 2960-Plus 24PC-S, 2960-24PC-S, 2960-Plus 24LC-L, 2960-Plus 24LC-S, 2960-24LC-S, 2960- Plus 24TC-L, 2960-24TC-L, 2960-Plus 48TC-L, 2960-48TC-L, 2960-24LT-L, 2960-24TT-L, 2960-48TT-L, 2960-Plus 48PST-L, 2960 -48PST-L 2960-Plus 48PST-S, і 2960-48PST-S

Порти 10/100 на комутаторах згруповані в пари. Перший член пари (порт 1) розташований над другим членом (порт 2), порт 3 над портом 4 і т. п.

Фіксовані порти 10/100 на комутаторах Catalyst 2960-24PC-L і 2960-24PC-S є портами PoE (Рисунок 1.5, 1.6).

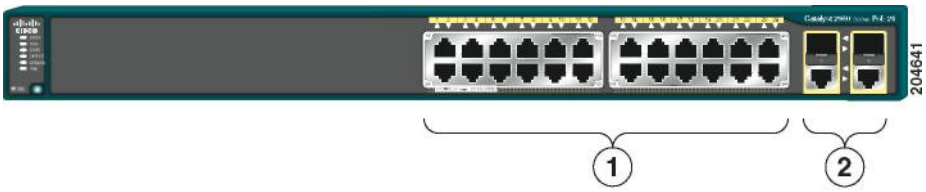

Рисунок 1.5 - Передня панель комутатора Catalyst 2960-Plus 24PC-L і 2960-24PC-L

Позначка 1: порти PoE 10/100, 2: порти подвійного призначення.

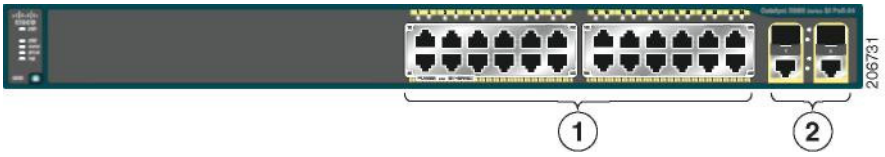

Рисунок 1.6 - Передня панель комутатора Catalyst 2960-Plus 24PC-S і 2960-24PC-S

Позначка 1: порти PoE 10/100, 2: порти подвійного призначення. Порти з 1 по 8 на комутаторі Catalyst 2960-24LC-S є портами PoE (Рисунок 1.7).

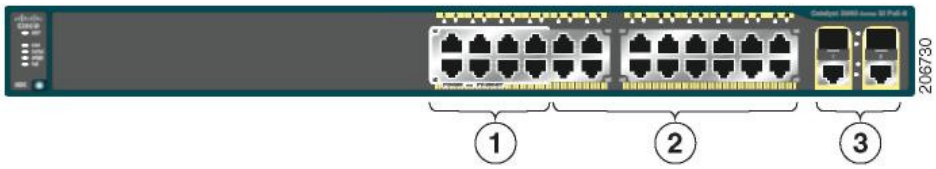

Рисунок 1.7 - Передня панель комутатора Catalyst 2960-Plus 24LC-L і 2960-24LC-S

Позначка 1: порти PoE 10/100, 2: порти 10/100, 3: порти подвійного призначення.

Комутатори Catalyst 2960-24TC-L і Catalyst 2960-48TC-L обладнані портами подвійного призначення, т. Е. Порти 10/100/1000 1 і 2 можуть використовуватися для підключення модуля SFP або роз'єму RJ-45, але не і того й іншого одночасно [5]. Щоб налаштувати тип роз'єму для цих портів, використовуйте програмне забезпечення (Рисунок 1.8 і 1.9).

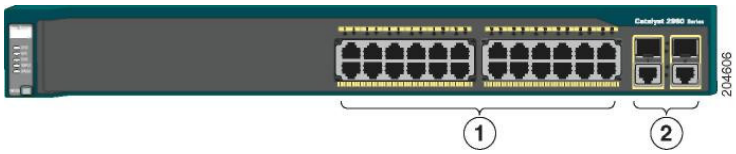

Рисунок 1.8 - Передня панель комутатора Catalyst 2960-Plus 24TC-L і 2960-24TC-L

Позначка 1: порти 10/100, 2: порти подвійного призначення.

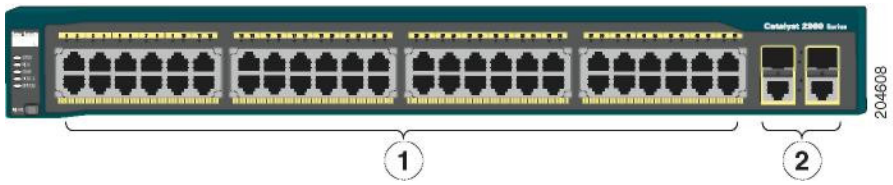

Рисунок 1.9 - Передня панель комутатора Catalyst 2960-Plus 48TC-L і 2960-48TC-L

Позначка 1: порти 10/100, 2: порти подвійного призначення. Комутатори Catalyst 2960-24LT-L, Catalyst 2960-24TT-L і Catalyst 2960-48TT-L обладнані 2 портами каскадирования 10/100/1000 з нумерацією 1 і 2. Порти з 1 по 8 на комутаторі Catalyst 2960-24LT-L є портами PoE (Рисунок 1.10, 1.11 і 1.12).

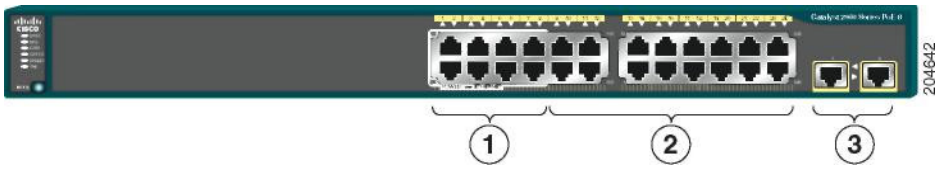

Рисунок 1.10 - Передня панель комутатора Catalyst 2960-24LT-L

Позначка 1: порти PoE 10/100, 2: Порти 10/100, 3: Порти каскадування 10/100/1000.

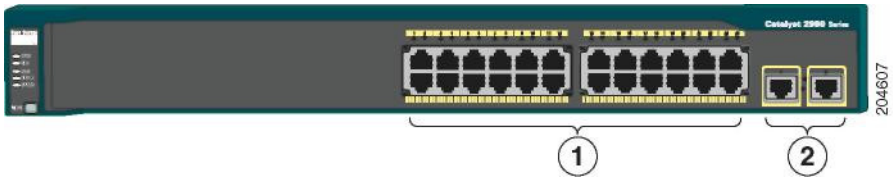

Рисунок 1.11 - Передня панель комутатора Catalyst 2960-24TT-L

Позначка 1: порти 10/100, 2: Порти каскадирования 10/100/1000.

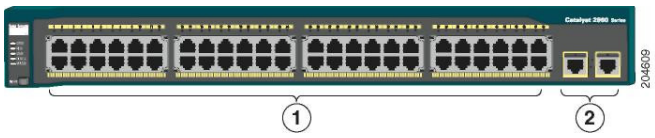

Рисунок 1.12 - Передня панель комутатора Catalyst 2960-48TT-L.

Позначка 1: порти 10/100, 2: порти каскадування 10/100/1000.

Комутатори 2960-48PST-L і 2960-48PST-S Catalyst оснащені 2 роз'ємами для модулів SFP (з нумерацією 1 і 2) і 2 портами каскадування 10/100/1000 (з нумерацією 3 і 4). Порти з 1 по 48 на комутаторі є портами PoE(Рисунок 1.13 і 1.14).

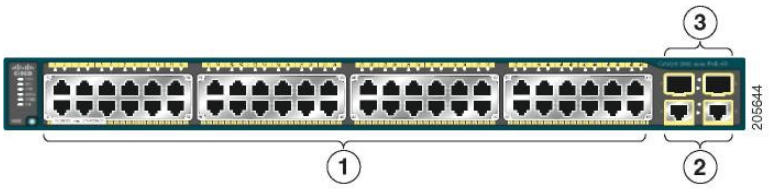

Рисунок 1.13 - Передня панель комутатора Catalyst 2960-Plus 48PST-L і 2960-48PST-L

Позначка 1: порти PoE 10/100 3 Роз'єми SFP, 2: Порти каскадування 10/100/1000

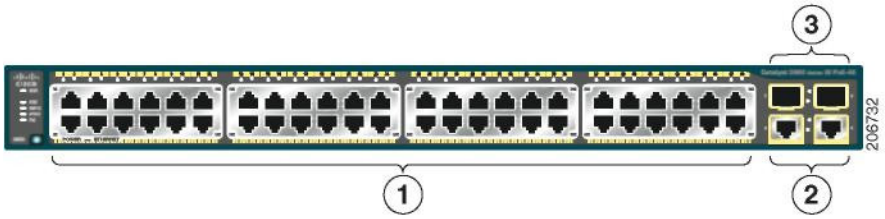

Рисунок 1.14 - Передня панель комутатора Catalyst 2960-Plus 48PST-S і 2960-48PST-S

Позначка 1: порти PoE 10/100, 3: роз'єми SFP, 2: порти каскадування 10/100/1000.

Порти 10/100/1000 на комутаторах Catalyst 2960G-24TC-L і Catalyst 2960G-48TC-L згруповані в пари. Перший член пари (порт 1) розташований над другим членом (порт 2), порт 3 над портом 4 і т. Д. Роз'єми модуля SFP мають нумерацію з 21 до 24 на комутаторі Catalyst 2960G-24TC-L і з 45 до 48 на комутаторі Catalyst 2960G-48TC-L. Див. Рисунок 1-15 і Рисунок 1-16.

Комутатори Catalyst 2960G-24TC-L і Catalyst 2960G-48TC-L мають порти подвійного призначення, т. Е. Порти з 21 по 24 або з 45 по 48 можуть використовуватися для підключення модуля SFP або роз'єму RJ-45, але не допускають їх одночасного використання. Щоб налаштувати тип роз'єму для цих портів, використовуйте програмне забезпечення. Для отримання додаткової інформації про порт подвійного призначення [6].

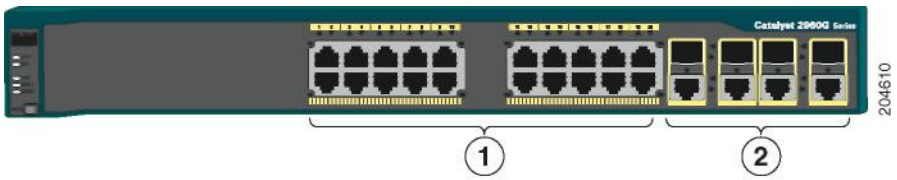

Рисунок 1.15 - Передня панель комутатора Catalyst 2960G-24TC-L.

Позначка 1: порти 10/100/1000, 2: порти подвійного призначення.

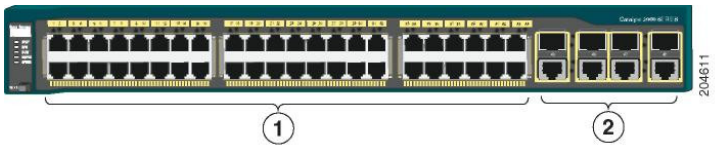

Рисунок 1.16 - Передня панель комутатора Catalyst 2960G-48TC-L

Позначка 1: порти 10/100/1000, 2: порти подвійного призначення.

На передній панелі комутатора Catalyst 2960PD-8TT-L (Малюнок 1-17) розташовані консольний порт, 8 портів 10/100 і порт каскадирования 10/100/1000, який може отримувати живлення від верхнього комутатора PoE. Комутатор також може отримувати живлення від додаткового адаптера змінного струму, підключеного через задню панель.

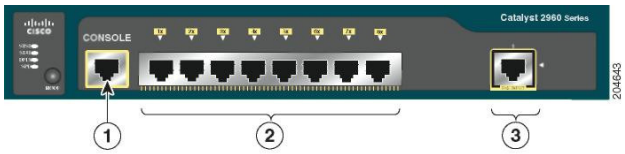

Рисунок 1.17 - Передня панель комутатора Catalyst 2960PD-8TT-L

Позначка 1: консольний порт, 2: порти 10/100, 3: порт живлення 10/100/1000.

Консольні порти комутаторів Catalyst 2960-8TC-S, Catalyst 2960-8TC-L і Catalyst 2960G-8TC-L (Рисунок 1.18 по 1.20) розташовані на передніх панелях. Комутаторитакож мають порт подвійного призначення, які можуть бути використані для підключення роз'єму RJ-45 або модуля SFP, але не допускає їх одночасного використання [7]. Щоб налаштувати тип роз'єму для цих портів, використовуйте програмне забезпечення.

Порти 10/100 можуть бути налаштовані для роботи зі швидкістю 10 або 100 Мбіт / с в повнодуплексному або напівдуплексному режимі. Також ці порти можуть використовуватися для підвищення швидкості і дуплексного автоузгодження. За умовчанням встановлений режим автоузгодження. При використанні порту для виконання автоузгодження він запитує відомості про швидкість і настройках дуплексного режиму підключеного пристрою і відправляє відомості про свої можливості. Якщо підключений пристрій також підтримує автоузгодження, порт комутатора вибирає оптимальні параметри підключення (т. п. Максимальну швидкість передачі даних по лінії, доступну для обох пристроїв, і повнодуплексну передачу, якщо підключений пристрій її підтримує) і встановлює відповідні налаштування. Підключений пристрій повинен розміщуватися на відстані не більше 100 м (328 футів).

Трафік 100BASE-TX вимагає використання кабелю категорії 5 або вище. Для трафіку 10BASE-T можуть використовуватися кабелі категорії 3 або 4.

Для підключення комутатора до робочих станцій, серверів, маршрутизаторів і IP-телефонами Cisco обов'язково повинен

використовуватися прямий кабель. Для підключення комутатора до комутаторів або концентраторів використовуйте перехресний кабель.

1.2 Аналіз функцій передньої панелі

Для підключення функції Auto-MDIX можна використовувати команду настройки інтерфейсу mdix auto в інтерфейсі командного рядка (CLI). При використанні функції Auto-MDIX комутатор визначає необхідний тип кабелю для мідних підключень Ethernet і відповідним чином налаштовує інтерфейси. Таким чином, можна використовувати або перехресний, або прямий кабель для підключень до мідного порту 10/100/1000 або 1000BASE-T SFP модуля на комутаторі незалежно від типу пристрою на іншому кінці з'єднання.

Порти 10/100/1000 можуть бути налаштовані для роботи зі швидкістю 10, 100 або 1000 Мбіт / с в повнодуплексному або напівдуплексному режимі. Також ці порти можуть використовуватися для підвищення швидкості і дуплексного автоузгодження. При використанні порту для виконання автоузгодження він запитує відомості про швидкість і настройках дуплексного режиму підключеного пристрою і відправляє відомості про свої можливості. Якщо підключений пристрій також підтримує автоузгодження, порт комутатора вибирає оптимальні параметри підключення. Максимальну швидкість передачі даних по лінії, доступну для обох пристроїв, і полнодуплексну передачу, якщо підключений пристрій її підтримує і встановлює відповідні налаштування. Підключений пристрій повинен розміщуватися на відстані не більше 100 м (328 футів).

Трафік 100BASE-TX і 1000BASE-T вимагає використання кабелю категорії 5 або вище. Для трафіку 10BASE-T можуть використовуватися кабелі категорії 3 або 4 [8].

Для підключення комутатора до робочих станцій, серверів, маршрутизаторів і IP-телефонами Cisco обов'язково повинен

використовуватися прямий кабель. Для підключення комутатора до комутаторів або концентраторів використовуйте перехресний кабель. При використанні прямого або перехресного кабелю для підключення 1000BASE-T для забезпечення належного функціонування слід використовувати кручені чотирипарні кабелі категорії 5 або вище.

За допомогою команди налаштування інтерфейсу mdix auto в інтерфейсі командного рядка (CLI) можна включити функцію автоматичного перемикання залежить від середовища інтерфейсу з перехрещуванням (auto-MDIX). При використанні функції Auto-MDIX комутатор визначає необхідний тип кабелю для мідних підключень Ethernet і відповідним чином налаштовує інтерфейси. Таким чином, можна використовувати або перехресний, або прямий кабель для підключень до мідного порту 10/100/1000 або 1000BASE-T SFP модуля на комутаторі незалежно від типу пристрою на іншому кінці з'єднання [9].

1.3 Порти комутатора Catalyst

Небезпечні напруги можуть бути присутніми в ланцюгах передачі живлення по кабелю Ethernet (PoE), якщо з'єднання є неізольованими металевими контактами, проводами або клемами.

Порти 10/100 на комутаторах Catalyst 2960-Plus 24PC-L, 2960-24PC-L, 2960-Plus 48PST-L, 2960-48PST-L, 2960-Plus 48PST-S, 2960-48PST-S 2960-Plus 24PC -S, і 2960-24PC-S, комутатори і порти з нумерацією від 1 до 8 портів 10/100 на комутаторах Catalyst 2960-24LT-L, 2960-Plus 24LC-L, 2960-Plus 24LC-S, і 2960-24LC -S забезпечують підтримку PoE для пристроїв, що відповідають вимогам стандарту IEEE 802.3af. Попередній стандарт PoE компанії Cisco також підтримується для IP-телефонів Cisco і точок доступу Cisco Aironet.

Кожен з портів PoE на комутаторах Catalyst 2960 забезпечує живлення потужністю до 15,4 Вт по кабелю Ethernet.

Комутатори Catalyst 2960-Plus 24PC-L, 2960-24PC-L, 2960-Plus 48PST-L, 2960-48PST-L, 2960-Plus 48PST-S, 2960-48PST-S 2960-Plus 24PC-S, і 2960-24PC-S забезпечують максимальну потужність живлення PoE приблизно всмоктування 370 Вт.

Комутатори Catalyst 2960-24LT-L, 2960-Plus 24LC-L, 2960-Plus 24LC-S, і 2960-24LC-S забезпечують максимальну потужність жисвлення PoE приблизно 124 Вт.

Для кожного порту PoE Catalyst 2960 можна вказати, чи буде він автоматично подавати живлення при підключенні IP-телефону або точки доступу. Диспетчер пристроїв Network Assistant і інтерфейс командного рядка визначають параметри PoE для кожного порту PoE 10/100.

Автоматично: при виборі автоматичної настройки порт забезпечує харчування тільки при підключенні допустимого пристрою, що відповідає стандарту IEEE 802.3af,

IP-телефону або точки доступу Cisco, що відповідають вимогам попереднього стандарту Cisco. Автоматична настройка визначено як стандартну.

Ніколи: при виборі цієї настройки порт не подає живлення навіть при підключенні IP-телефону Cisco або точки доступу [11].

IP-телефон Cisco або точку доступу Cisco Aironet також можна підключити до порту 10/100 PoE комутатора Catalyst 2960 і джерела змінного струму для забезпечення резервного живлення. При підключенні до джерела змінного струму пристрій може почати використовувати його в якості основного джерела живлення. В цьому випадку порт PoE стає резервним джерелом живлення.

При виході з ладу основного джерела живлення для підключеного пристрою основним стає додаткове джерело. Під час перемикання джерел живлення IP-телефон може виконати перезавантаження або перепідключення до комутатора.

Багато застарілих пристроїв, включаючи старі моделі IP-телефонів Cisco і точок доступу, в повному обсязі відповідають вимогам IEEE 802.3af, можуть не

підтримувати PoE при підключенні до комутаторів за допомогою перехресного кабелю [10].

Комутатори Catalyst 2960 (за винятком перерахованих) використовують модулі SFP Gigabit Ethernet для гігабітних каскадних підключень і модулі SFP 100-Megabit для підключень 100-Megabit за допомогою оптоволоконного кабелю. Наступні комутатори Catalyst 2960 не обладнані роз'ємами для модулів SFP:

- Catalyst 2960PD-8TT-L.
- Catalyst 2960-24LT-L.
- Catalyst 2960-24-S.
- Catalyst 2960-24TT-L.
- Catalyst 2960-48TT-L.
- Catalyst 2960-48TT-S.

Модулі приймачів мають можливість заміни на місці, забезпечуючи інтерфейси каскадирования при підключенні модуля SFP. Модулі SFP можна використовувати для гігабітних каскадних підключень з іншими комутаторами. Для підключення до оптоволоконних модулів SFP використовуються оптоволоконні кабелі LC з роз'ємами LC. Модулі SFP для мідного кабелю використовують кабелі категорії 5 або вище з роз'ємами RJ-45 [12].

Для отримання додаткової інформації про модулях SFP см. Документацію для вашого модуля SFP або примітки до випуску програмного забезпечення комутатора. Для отримання додаткової інформації про вимоги до кабелів див. Додаток В, «Технічні характеристики роз'ємів і кабелів».

Порт подвійного призначення можна налаштувати в якості порту 10/100/1000 або порту модуля SFP. Кожен порт являє собою єдиний інтерфейс з двома входами: роз'ємом RJ-45 і роз'ємом для модуля SFP. Два входу не є резервними інтерфейсами по відношенню один до одного. Комутатор активує тільки один роз'єм одночасно.

За замовчуванням комутатор динамічно вибирає тип інтерфейсу, який використовували при першому підключенні. Кожен порт каскадування має 2

індикатора: один відображає стан порту RJ-45, інший - стан порту модуля SFP. Індикатор порту горить тільки у активного роз'єму [13].

Постановка завдання:

– Дозволити доступ з довірених IP на будь-який порт.

– Створити ban-list із завантаженням з файлу.

– Дозволити доступ з crm і api нашої компанії на порт 8080 і 8443.

– Дозволити вихідний доступ по порам 80 і 443 до CRM і API.

– Створити захист від брут-форсу на порту 22 з внесенням хостів в бан-лист на годину.

– Створити захист від DDOS і DOS на портах 80 та 443, з внесенням хостів в інший бан-лист.

2 ЗАСОБИ МАРШРУТИЗАЦІЇ

2.1 Маршрутизація засобами фаєрволу

Розглянемо такі засоби як маршрутизація засобами фаєрволу (Policy Based Routing) . Однак, є відмінності в синтаксисі команд для FreeBSD і OpenBSD.

PF - Packet Filter - це міжмережевий екран, спочатку створений в рамках проекту OpenBSD. У 2003 році був портований під FreeBSD. У 2004 році був інтегрований в основну систему. Основні можливості:

- фільтрація на основі адрес, портів, протоколів, інтерфейсів;
- NAT - Source NAT, підміна адреси відправника. Destination nat, підміна адреси одержувача, кидок порту;
- Scrub - нормалізація мережевого трафіку. Допомагає від деяких видів dos атак, заснованих на формуванні спеціально підготовлених пакетів;
- SYN-proxy - Захист від SYN-flood атак;
- балансування з'єднань;
- відмовостійкість - pfcync дозволяє синхронізувати стан файрволов на нескольно хостах, що, в поєднанні з протоколом CARP, дозволяє створити відмовостійкий файрвол, який продовжить обробляти з'єднання після падіння активної Ноди [14];
- прозорий файрвол (Без власного IP-адреси, включаючи фільтр 2 рівня).
- макроси - аналог змінних;
- таблиці - динамічно змінні без перезавантаження конфігурації списки IP адрес;
- мітки - дозволяє мітити пакети, якщо простий фільтрації недостатньо;
- якоря (anchors) - набори правил, схожі на таблиці IPTables в Linux;
- збір статистики та висновок графіків за допомогою утиліти pfsta;
- автоматична оптимізація правил при завантаженні.

Основна відмінність від того ж IPTables - незвична схема роботи. Обробка пакета не закінчується після першого збігу з правилом. Тобто, якщо ви

поставили першим правилом «заборонити все» , то пакет не буде відкинутий, а послід як заборонений, піде далі за правилами, і, якщо його не дозволить ніяке правило, то буде відкинутий. Це важливо розуміти і використовувати. Однак, якщо потрібно, ці дії можна скасувати параметром quick в правилі.

Для включення PF досить в файлі «/etc/rc.conf» вказати опції:

```
pf_enable="YES" # включает pf и загружает модуль
pf_flags="" # дополнительные флаги pfctl
pf_rules="/etc/pf.conf"  # файл конфигурации
pflog_enable="YES" # запуск pflog
pflog_flags="" # флаги pflog
pflog_logfile="/var/log/pflog" # файл лога
```

Основні команди управління файрволом:

```
pfctl - # Включити файрвол.
pfctl -d # Вимкнути файрвол.
pfctl -nf # Перевірити синтаксис файлу.
pfctl -f # Перечитать правила з файлу.
pfctl -Rf # Перечитать правила фільтрації з файлу.
pfctl -Nf # Перечитать правила NAT з файлу.
pfctl -sa # Перегляд всіх станів.
pfctl -s # Перегляд правил фільтрації.
pfctl -sn # Перегляд правил NAT.
pfctl -s Anchors -v # Перегляд дерева якорів.
pfctl -ss # Перегляд поточних з'єднань.
```

Структура файлу конфігурації і базові настройки. Файл конфігурації складається з розділів:

- Правила нормалізації трафіку (scrub).
- Черги, пріоритезація і контроль швидкості.
- NAT трансляції адрес.
- фільтрація пакетів.

Правила в загальному випадку мають наступний синтаксис:

```
    action [direction] [log] [quick] [on interface] [af] [proto
protocol]
      [from src_addr [port src_port]] [to dst_addr [port dst_port]]
      [flags tcp_flags] [state]
```

Команда передбачає наступні опції:

- action - що слід зробити з пакетом;

- direction - in out, напрямок;

- log - чи потрапить пакет в pflog;

- quick - якщо пакет потрапив під це правило, то подальшої обробки не буде.

Це правило буде останнім для пакета з наступними параметрами:

- interface - назва мережевого інтерфейсу;

- af - address family, inet або inet6, IPv4 або IPv6 відповідно;

- protocol - протокол 4 рівня, наприклад: tcp, udp, icmp;

- scr_addr, dst_addr - адреси джерела і призначення;

- src_port, dst_port – порти;

- tcp_flags - прапори tcp;

- state - опції збереження стану.

Наприклад, keep state означатиме, що з'єднання збережеться в таблиці станів, і відповідні пакети можуть проходити. Поведінка за замовчуванням.

Візьмемо найпростіший веб-сервер в вакуумі. Необхідно відкрити вхідні з'єднання по портам tcp 22, 80, 443 (ssh, http, https). Також потрібно відкрити вихідні з'єднання по портам tcp 22, 80, 443 (ssh, http, https) і udp 53, 123 (dns і ntp). Все інше заборонити [15-17].

```
# ee pf.conf
#macros section
permit_tcp_ports="22,80,443"
permit_udp_ports="53,123"
#table section
# наданий час пуста
#options section
set block-policy return
```

Надалі необхідно розірвати зєднання а не просто дропаєм пакети.

```
set skip on lo0
# пропускаємо провірку на локальній петлі, там фильтрація не потрібна
#scrub section
scrub in all
```

```
# нормалізація всього вхідного трафіка
#Queueing section
# в данний момент пуста
#nat section
# пуста, в нас нема чого транслювати
#filtering section
block all
```

Після виконання правил необхідно закриття всього трафіку по замовчуванню.

```
pass in proto tcp to port { $permit_tcp_ports }
# дозволяєм вхідні підключення
pass out proto tcp to port { $permit_tcp_ports }
# дозволяєм вихідні підключення tcp
pass out proto udp to port { $permit_udp_ports }
# дозволяєм вихідні підключення udp
pass out inet proto icmp
 # дозволяєм вихідний icmp
```

Потім вводимо команду перевірки синтаксису:

```
pfctl -nf pf.conf
```

Якщо повідомлень про помилки немає, вводимо команду застосування правил:

```
pfctl -f pf.conf
```

Для перевірки подивимося правила фільтрації:

```
# pfctl -sr
scrub in all fragment reassemble
block return all
pass out proto tcp from any to any port = ssh flags S/SA keep
state
pass out proto tcp from any to any port = http flags S/SA keep
state
```

Як видно, макроси розгорнулися в окремі правила по кожному порту, порядок змінений автоматично. В іншому випадку все так, як потрібно.

```
pass out proto tcp from any to any port = https flags S/SA keep
state
pass out proto udp from any to any port = domain keep state
pass out proto udp from any to any port = ntp keep state
pass out inet proto icmp all keep state
pass in proto tcp from any to any port = ssh flags S/SA keep state
pass in proto tcp from any to any port = http flags S/SA keep
state
pass in proto tcp from any to any port = https flags S/SA keep
state
```

Створили конфігурацію, яка, однак цілком функціональна, включає макроси, нормалізацію пакетів і фільтрацію вхідних і вихідних пакетів. У наступній статті розберемо більш детально правила фільтрації, управління власністю та прапори [18].

Розроблене рішення - це шлюз безпеки, побудований на базі FreeBSD. Дана система була обрана з огляду на свою стабільності, швидкості мережевого стека, вбудованої в ядро підтримки ZFS і відсутності проблем з ліцензіями. Крім того, робота з FreeBSD досить приємна, хоч і не в усьому проста.

Дослідимо роботу з мітками і виконаємо більш просунуту фільтрацію трафіку за різними умовами.

Макроси в різних місцях конфігураційного файлу. Списки - набір параметрів. PF розкриє його в окремі правила. Правила, що обмежують кількість підключень з однієї IP, їх частоту. Таблиці - це список IP адрес з якими може порівнювати правило. Можливі варіанти динамічного заповнення і перевірку стану наших таблиць.

2.2 Налаштування макросів та списків

З макросами все відносно просто. Це вставка рядка в потрібне місце конфігураційного файлу. У макросі можна тримати ім'я інтерфейсу, IP адресу (або адреси), опції tcp, і так далі.

```
# macros section
If="re1"
# інтерфейс нашого сервера
IfIp="10.64.5.110"
# IP нашого сервера
permit_tcp_ports="22,53"
permit_udp_ports="53,123"
web_ports = "http,https"
```

Після цього можна використовувати їх в правилах подібного виду:

```
pass in on $IfIp inet proto tcp from any to $IfIp port {$permit_tcp_ports}
```

Після завантаження правила будуть виглядати так:

```
pfctl -sr
...
pass in on re1 inet proto tcp from any to 10.64.5.110 port = ssh flags S/SA keep state
pass in on re1 inet proto tcp from any to 10.64.5.110 port = domain flags S/SA keep state
    …
```

Як бачимо, був підставлений інтерфейс і порти з макросу. Потім, для кожного порту було створено окреме правило відповідно до списку. Були встановлені значення за замовчуванням. flags це tcp прапори, встановлені у пакет для відповідності правилу. S / SA - це пакети з встановленим прапором SYN. Тобто, якщо спростити, перший пакет сесії.

Keep state - означає збереження з'єднання в таблиці станів або сесій. Це означає, що тільки перший пакет із з'єднання пробіжить по правилам фаєрволу. Всі інші пакети цього з'єднання фаєрвол пропустить відповідно таблиці станів (state table), не проводячи перевірок [19].

Важливо для оптимізації конфіга використовувати опцію quick. Вона вказує файрвол припинити подальшу обробку цього пакета, і відразу застосувати поточну дію.

```
pass in quick on em0 from { 10.64.1.0/16} to 10.64.1.1
```

Таблиці - це одне з найпотужніших засобів PF. Після створення в конфіги, з ними можна творити все, що завгодно. Завантажувати з файлу, чистити, очищати від записів, старіше заданого терміну, ну і, звичайно, додавати і видаляти записи. І все це на льоту без перезавантаження конфігурації файрвола [20].

Таблиці можуть бути декількох видів:

- const - цей прапор вказує на незмінність таблиці. Користувач не може додавати і видаляти IP "на льоту".
- counters - включає ведення статистики по кожному IP в таблиці.
- persist - вказує ядру зберігати таблицю, навіть якщо в ній немає записів.

У цій статті будуть показані приклади таблиць без прапорів і з прапором persist. Решта прапори інтуїтивно зрозумілі і використовуються значно рідше.

Проста таблиця:

```
# для адміністрування
table <admins> { 172.16.11.20, 172.17.11.44 }
```

Для використання в правилі файрволу:

```
pass in quick on $If from <admins> to $IfIp
```

Це правило дозволить все, що приходить від IP адрес, перерахованих в таблиці, на мережевий інтерфейс і IP сервера.

Таблиця із завантаженням з файлу:

```
# хости які заблоковані
table <block_source> file "/etc/pf.blocklist.conf"
```

Використовуємо в правилі:

```
block in quick on $If from <block_source> to any
```

Правило заблокує будь-який трафік від <simple-ban> на інтерфейсі $ If

Сама таблиця буде мати такий вигляд:

```
# cat /etc/pf.blocklist.conf
1.1.1.1
110.12.47.0/24
!111.12.46.17
```

Можна вказати знак заперечення для виключення з таблиці. В даному випадку буде відповідність IP 1.1.1.1, і мережі 110.12.47.0/24, крім IP 111.12.46.17. Наповненням цього файлу можна займатися вручну. Потім завантажити таблицю з файлу знову [23-25]:

```
pfctl -t block_source -T replace -f /etc/pf.blocklist.conf
3 addresses added.
```

У таблицю можна додавати як IP адреси, так і доменні імена. Доменні імена вирішаться при завантаженні. Для поновлення адрес можна поставити завантаження таблиці в cron кожну годину, наприклад [21-22]. Однак, якщо хоча б одна адреса дозволити не вдасться, оновлення таблиці не вдасться:

```
# cat  /etc/pf.blocklist.conf
ki.wunu.edu.ua
5.6.5.4
wunu.edu.ua
```

Після завантаження можна подивитися, що міститься в таблиці:

```
# pfctl -t blocklist -T show
   1.2.3.4
   87.251.250.242
   94.101.181.201
   94.100.180.202
   217.69.139.203
   217.69.135.205
   2a01:1148:db01:0:b0b0::1
   2a03:6b9::2:242
```

Можна зручним чином додати / видалити записи:

```
# pfctl -t blocklist -T add 123.12.34.5
# pfctl -t blocklist -T delete 123.12.34.5
```

Звичайно, ці записи не потраплять магічно в файл, і після перезавантаження таблиці їх не буде.

Важливо пам'ятати про можливість перевірки відповідності IP таблиці. Можна перевірити і доменне ім'я.

```
# pfctl -t blocklist -T test 1.2.3.4
1/1 addresses match.
# pfctl -t blocklist -T test 1.1.1.1
0/1 addresses match.
# pfctl -t blocklist -T test ya.ru
2/2 addresses match.
# pfctl -t blocklist -T test 1.1.1.1 1.2.3.4
1/2 addresses match.
```

Завдання в cron для поновлення:

```
# cat /etc/crontab
...
0  *  *  *  *  root  /sbin/pfctl  -t  block_source  -T  replace  -f
/etc/pf.blocklist.conf
...
```

Тепер додамо можливість ходити на корпоративні ресурси по портам http (s) і назад по портам 8080 і 8443. Створюємо макроси зі списком портів, можна використовувати назви протоколів:

```
permit_corp_ports ="http,https"
# порти для доступу до корпоративних ресурсів
permit_add_tcp_ports = "8081,8444"
# додаткові порти
```

Таблицю, що завантажується з файлу, зі списком хостів:

```
table <corp_res> file "/etc/pf.corpres.conf"
# корпоративні ресурси, доступ по http(s)
```

Для прикладу, в таблиці буде один хост:

```
# cat /etc/pf.corpres.conf
ki.wunu.edu.ua
```

І тепер можна створити самі правила:

```
pass in quick on $If proto tcp from <corp_res> to $IfIp ports {
$permit_add_tcp_ports }
pass out quick on $If proto tcp from $IfIp to <corp_res> port {
$permit_corp_ports }
```

Створені правила дозволять ефективно використовувати макроси в файлі налаштувань фаєрволу.

2.3 Розширені можливості пошуку фільтрації

Тепер уеобхідно захистити наш сервер від перебору паролів ssh. Ще одна можливість PF - лімітувати кількість з'єднань, що відповідають правилу, за різними параметрами. Таблиця параметрів:

```
table <block_ssh_daemon> persist
# Брутфорсери ssh. якщо таблиця пуста, PF видалить її.
```

Для відміни такої поведінки використовується ключеве слово persist.

persist - тримати в пам'яті, навіть якщо порожня. Якщо не вказати таку поведінку, PF видалить порожню таблицю [26].

І правила:

```
block in quick on $If proto tcp from <block_ssh_daemon> to any
port 22
pass in on $If proto tcp to $IfIp port { 23 } \
keep state (max-src-conn 10, max-src-conn-rate 3/10, \
overload <block_ssh_daemon> flush)
```

В даному випадку max-src-conn 10 - тільки 10 одночасних з'єднань з однієї адреси, max-src-conn-rate 3/10 - тільки 3 нових з'єднання за 10 секунд, overload <block_ssh_daemon> - додавати в таблицю адреси джерела порушників, flush - скидати все з'єднання від порушника, створені цим правилом.

За такими ж правилами надамо дозвіл користувачеві відвідувати веб-сервер. Таблиця:

```
table <block_web> persist # заблокований доступ до веб
```

І тепер необхідно додати наступні правила:

```
# дозволяємо вхідні зєднання
block in quick on $If from <block_web>
pass in on $If proto tcp to $IfIp port { $permit_tcp_ports } \
synproxy state (max-src-conn 100, max-src-conn-rate 20/1, \
overload <block_web> flush global)
```

Параметри цього правила треба дуже акуратно підбирати під ваш сервіс. Synproxy - включає проксінг syn запитів, захист від synflood атак. А flush global означає скидання всіх з'єднань з джерела-порушника, навіть якщо вони не належать до цього правила [27].

Старіння записів в таблицях заблокованих забезпечуються командами такого виду:

```
pfctl -t block_ssh_daemon -T expire 3600
```

Ця команда видалить всі записи, старше години з відповідної таблиці. Команди поставимо в крон:

```
...
0 * * * * root /sbin/pfctl -t block_ssh_daemon -T expire 3600
0 * * * * root /sbin/pfctl -t block_web -T expire 3600
...
```

Для перевірки правильності налаштування виконуємо команду:

```
# pfctl -nf pf.conf
```

Завантажуємо конфігураційний файл для служби:

```
# pfctl -f pf.conf
```

Завантаживши конфігурацію, служба готова до роботи після її перезапуску.

3 ЗАСОБИ ТРАНСЛЯЦІ ПАКЕТІВ

3.1 Види трансляцій PF

PF підтримує три види трансляцій.

Anchors - це окремі набори правил PF, якими, подібно таблиць, можна управляти динамічно.

Tag - це спосіб позначити пакет, щоб потім обробити в іншому правилі. Зручно, наприклад, мітити пакети, що проходять через кидок, щоб потім їх дозволити скопом в одному правилі.

У мережі 172.17.3.0/24 знаходиться бухгалтерія. Головний бухгалтер, має особливі права 172.17.3.2.

У другій мережі - 172.17.3.0/24 знаходиться сервер (172.17.3.1), який треба транслювати в інтернет 1 в 1. Комп'ютер системного адміністратора, особливі права плюс треба зробити кидок порту для торрентів (59715 TCP і UDP).

Крім того, потрібно заборонити ходити на сайти заблоковані законодавством, для прикладу візьмемо кілька доменів на зразок vk.com і odnoklasniki.ru, для всіх, крім обраних. І зробити для штрафників забороне входу в інтернет, крім кількох сайтів, нехай будуть google.com і bing.com, необхідних для роботи.

Всім потрібно ходити тільки до сервера в Інтернет, 172.17.1.2. Провайдер нам дає мережу 172.16.1.1/24 зі шлюзом 172.17.1.254 і додатковий IP для внутрішнього сервера 172.16.1.5.

Виконаємо налаштування схеми мережі. Для початку підготуємо роутер. Нехай мережеві інтерфейси і маршрут за замовчуванням вже налаштований.

По-перше потрібно додати можливість маршрутизації. Для цього в /etc/rc.conf додаємо рядок:

```
gateway_enable="yes"
```

Потім активуємо маршрутизацію:

```
# sysctl net.inet.ip.forwarding=1
```

Або перезавантажуємо машину.

Все, маршрутизація вже працює. Додаємо кілька макросів, таблиць, опції, черги , трансляції. Час додати трохи трансляцій. Правила будуються за загальним синтаксису. Однак додалося виключення дії, опції якої:

- binat - двунаправленна трансляція, статична трансляція, або трансляція один в один в обидва напрямки;

- nat - source nat, підміна джерела. Те, що називають NAT майже всюди;

- rdr - кидок портів, трансляція адреси призначення, destination nat, dnat.

Схема проектованої мережі приведена на рисунку 3.1.

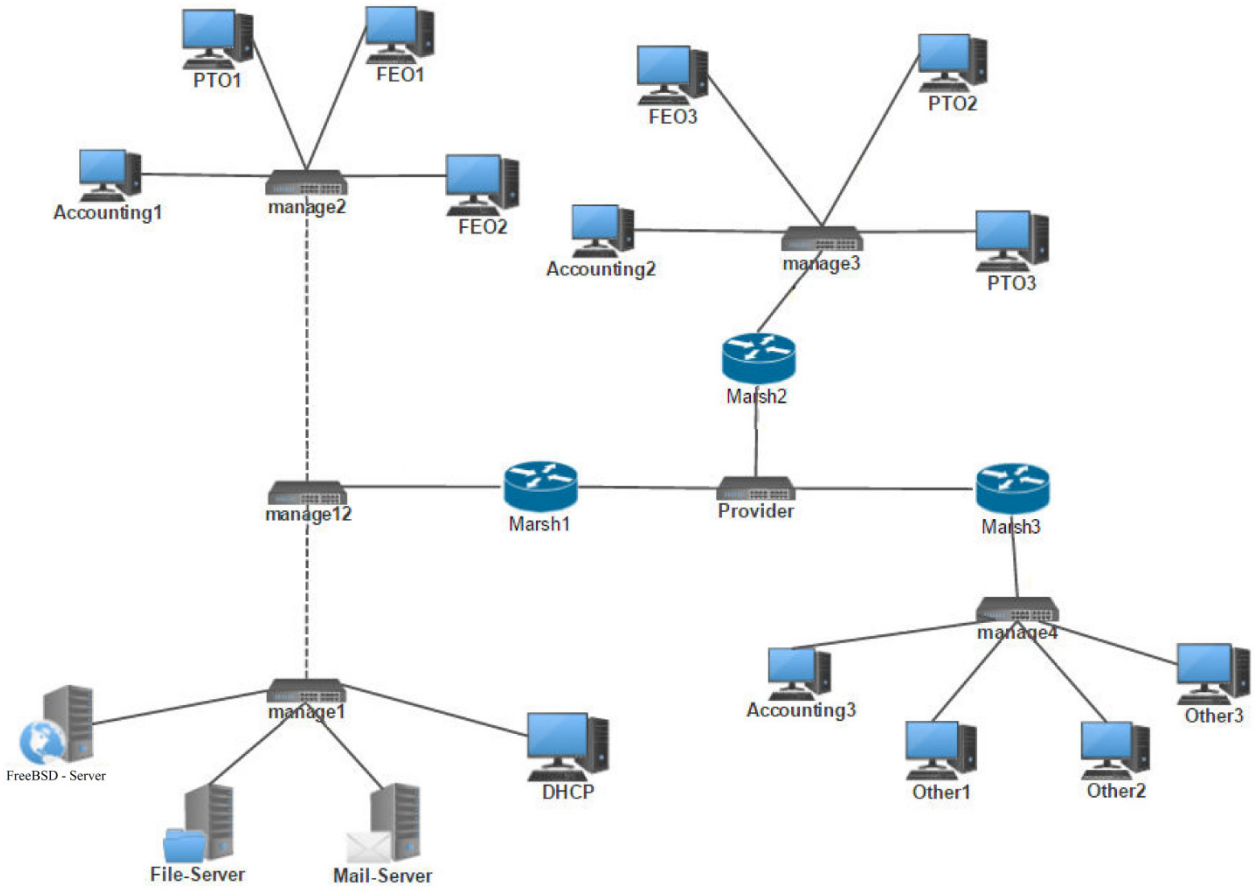

Рисунок 3.1 – Структурна схема проектованої мережі

Проектування мережі передбачає декілька етапів. Схема мережі рівня L-1 приведена на рисунку 3.1.

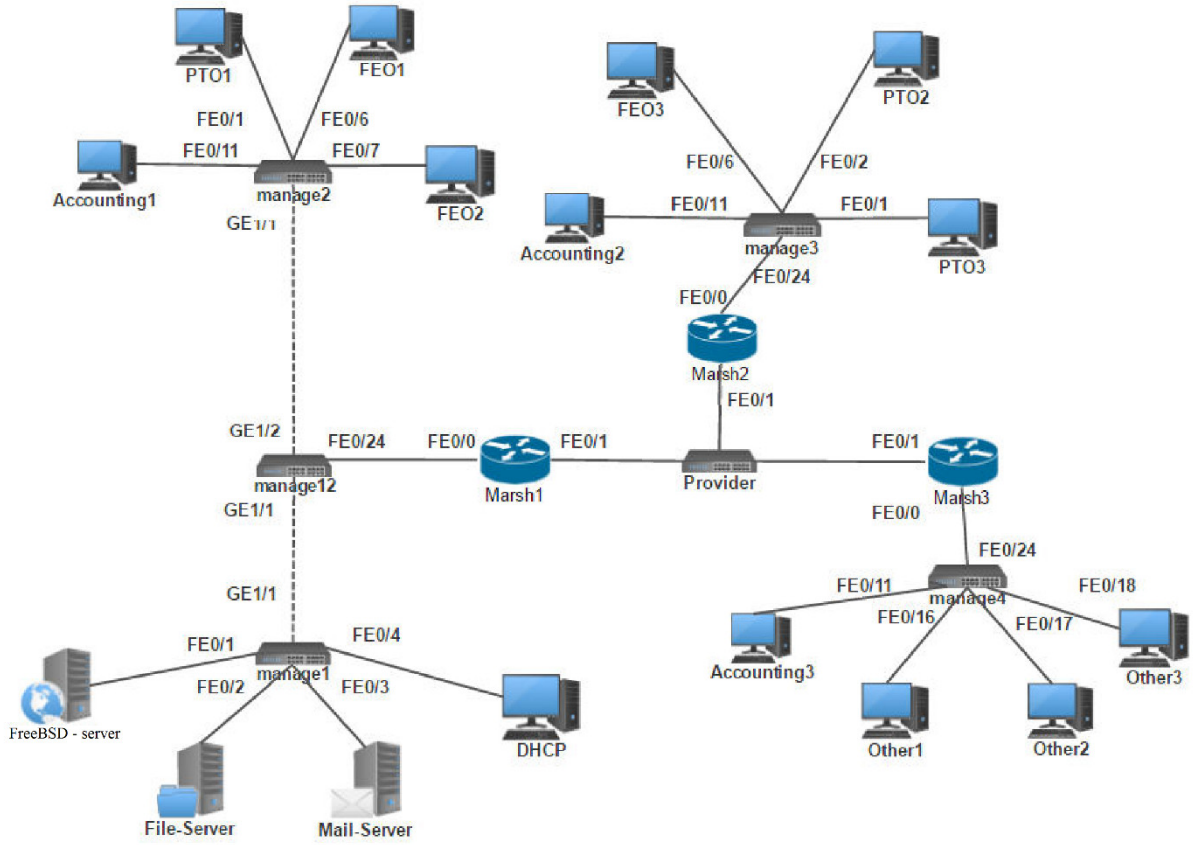

Рисунок 3.1 –Схема проектованої мережі L-1

Почнемо налаштування фаєрвола зі статичної трансляції. Синтаксис команди (з офіційної документації):

```
[ "no" ] "binat" [ "pass" [ "log" [ "(" logopts ")" ] ] ]
             [ "on" interface-name ] [ af ]
             [ "proto" ( proto-name | proto-number ) ]
             "from" address [ "/" mask-bits ] "to" ipspec
             [ "tag" string ] [ "tagged" string ]
             [ "->" address [ "/" mask-bits ] ]
```

Всі параметри приблизно ті ж, що і з правилами фільтрації, крім no на початку. Це як раз той самий виняток. Якщо поставити його над правилом трансляції, що потрапляє під нього трафік трансльований не буде. Виглядає це так:

```
no binat on $ExtIf from $IntSrv to 172.17.34.0/24
binat on $ExtIf from $IntSrv to any -> $ExtIfIp2
```

Це означає, що весь трафік, крім виключення, від хоста IntSrv (172.17.3.1) буде при виході в інтернет ре трансльований на адресу ExtIfIp2 (172.16.1.5), і навпаки, все, що прийшло з інтернету на цей зовнішній адресу, полетить на наш сервер. Зручно, коли треба прокинути купу портів, і в деяких інших випадках. Наприклад для виділеної АТС. У конфігурацію піде таке правило:

```
binat on $ExtIf from $IntSrv to any -> $ExtIfIp2
```

Тепер - NAT з локальної мережі в Інтернет. Синтаксис правила:

```
[ "no" ] "nat" [ "pass" [ "log" [ "(" logopts ")" ] ] ]
               [ "on" ifspec ] [ af ]
               [ protospec ] hosts [ "tag" string ] [ "tagged"
string ]
               [ "->" ( redirhost | "{" redirhost-list "}" )
               [ portspec ] [ pooltype ] [ "static-port" ] ]
```

І рядок в конфігураційного файлу:

```
nat on $ExtIf from { $IntIf1Net, $IntIf2Net } to any -> $ExtIfIp1
```

Це правило говорить про те, що все, що йде в інтернет потрібно закрити першим IP зовнішнього інтерфейсу. Можна поставити виключення, подібно binat . Крім звичайної трансляції в один статичний IP, можна, наприклад, щоб PF сам знаходив цей IP з інтерфейсу. Корисно, якщо провайдер дає динамічний IP по PPPoE або DHCP:

```
nat on $ExtIf from { $IntIf1Net, $IntIf2Net } to any -> ($ExtIf)
```

Якщо на інтерфейсі кілька IP, або мереж то її можна вказати як безпосередньо, так і використовуючи такий синтаксис, буде вказана вся мережа зовнішнього інтерфейсу (підтримується тільки round-robin):

```
nat on $ExtIf from { $IntIf1Net, $IntIf2Net } to any ->
($ExtIf:network)
```

За замовчуванням PF буде розподіляти відповідність з'єднань і зовнішній IP по стратегії round-robin . Цю поведінку можна змінити, використовуючи явну вказівку мережі. Доступні ще стратегії source-hash , коли вибирається для кожної адреси IP з отриманого пулу, грунтуючись на внутрішньому, або випадковому виборі random. Можна включити прилипання, sticky-address , яке гарантуватиметься, на тому що один внутрішній IP завжди буде закриватися одним і тим же зовнішнім, поки є живі з'єднання. Якщо живих з'єднань немає, то адреса відлипає:

```
nat on $ExtIf from { $IntIf1Net, $IntIf2Net } to any ->
172.16.1.0/27 \
source-hash sticky-address
```

При прокиданні портів відмінностей мало.

```
[ "no" ] "nat" [ "pass" [ "log" [ "(" logopts ")" ] ] ]
        [ "on" ifspec ] [ af ]
        [ protospec ] hosts [ "tag" string ] [ "tagged" string ]
        [ "->" ( redirhost | "{" redirhost-list "}" )
        [ portspec ] [ pooltype ] [ "static-port" ] ]
```

Правило в конфігураційний файл:

```
rdr on $ExtIf inet proto {tcp, udp} from any to $ExtIfIp1 \
    port { $torrent_port } -> 172.17.3.15 port $torrent_port
```

Правило заверне вхідні пакети на IP адміністратора. З додаткових можливостей - можна організувати балансувальник:

```
rdr on $ExtIf inet proto {tcp, udp} from any to $ExtIfIp1 \
    port { $torrent_port } -> { 172.17.3.15, 172.17.3.15 } \
    port $torrent_port round-robin
```

Ну і, як і всюди, вище можна поставити виняток. В один час у пакета може бути тільки один тег. Правило може перевизначити тег пакету, але не видалити його

Правила відбору можна інвертувати знаком заперечення перед ключовим словом :! tagged PASS

Тег (tag) - це спосіб позначити пакет, який пройшов за правилом, для подальшої обробки. Для розуміння, в чому чарівництво моменту, потрібно трохи розібратися з тим, як пакет проходить по PF.

У разі звичайного хоста все просто. Є інтерфейси, є вхідний трафік, є вихідний, який виробляє сам хост. Однак в разі маршрутизатора пакет спершу входить на інтерфейс, проходить за відповідними правилами in, на кшталт:

```
pass in quick on $IntIf1 from $IntIf1Net to $IntIf2Net
```

потім, перед виходом з іншого інтерфейсу, проходить за правилами out:

```
pass out quick on $IntIf2 from $IntIf1Net to $IntIf2Net
```

і, якщо нам потрібно просто дозволити ходити між двома своїми локальними мережами, доводиться робити чотири правила:

```
pass in quick on $IntIf1 from $IntIf1Net to $IntIf2Net
pass out quick on $IntIf2 from $IntIf1Net to $IntIf2Net
pass in quick on $IntIf2 from $IntIf2Net to $IntIf1Net
pass out quick on $IntIf1 from $IntIf2Net to $IntIf1Net
```

Є два варіанти уникнути цього. Перший - не вказувати інтерфейси і напрямки. Спорудити щось на кшталт:

```
pass quick from $IntIf1Net to $IntIf2Net
```

Проте це зменшує безпеку файрвола. Рішення подібної проблеми дуже просте:

```
pass in quick on $IntIf1 from $IntIf1Net to $IntIf2Net tag PASS
pass in quick on $IntIf2 from $IntIf2Net to $IntIf1Net tag PASS
pass out tagged PASS
```

Тег, зберігаючись на всьому шляху проходження пакету, в кінці кінців потрапить на правило pass out, і трафік пройде.

Такий прийом дозволяє скоротити кількість правил і вибудувати більш логічну і просту їх структуру. Точно так само можна, наприклад, блокувати трафік на частини інтерфейсів.

В результаті на вихідному інтерфейсі не завжди можна однозначно зрозуміти, звідки прийшов пакет. Наприклад правило, яке дозволяє торрент адміністратора, на зовнішньому інтерфейсі, буде використовувати локальний IP:

```
pass in on $ExtIf proto { tcp, udp } from any to 172.17.3.15 \
port { $torrent_port } tag PASS
```

А ще правила фільтрації PF, в разі, якщо на інтерфейсі є трансляція, побачать вже ретрансльовані пакети.

3.2 Використанння якорів

Якщо зібрати всі налаштування в конфіг, то на даний момент, вийде цілком робочий конфігураційний файл PF, який зробить все, що потрібно. Однак можна налаштування можна покращити використавши якоря. Так само відомі як anchors. Це окремі набори правил PF. Основні властивості:

Якоря можуть містити інші якоря, шикуючись в деревоподібну структуру. Якір можна завантажувати з файлу. Якорями можна управляти динамічно. Видаляти, додавати правила, etc. Макроси необхідно визначити в тому ж файлі, що і якір. Є кілька способів визначення якорів:

– nat-anchor NAME виконає правила nat;

– rdr-anchor NAME виконає правила rdr;

– binat-anchor NAME виконає правила binat;

– anchor NAME виконає правила фільтрації;

– load anchor NAME from FILE команда завантаження з файлу.

Під час визначення anchor так само можна вказати, який трафік буде відправлений в нього.

Просте визначення anchor із завантаженням з файлу:

```
anchor localnet1
load anchor localnet1 from "/etc/pf-anchor.localnet1.conf"
```

Можна обійтися і без завантаження з файлу, просто визначити anchor першим рядком, потім довантажувати в нього правила з командного рядка.

```
# echo "pass in quick on $IntIf1 from $IntIf1Net to $IntIf2Net tag
PASS" | \
pfctl -a localnet1 -f -
```

Переглянути правила, додані в якір:

```
# pfctl -a "localnet1" -sr
```

Висновок буде схожий на pfctl -sr, тому що кореневої конфігураційний файл - це просто кореневої якір.

Перезавантажити якір з файлу:

```
# pfctl -a "localnet1" -f /etc/pf-anchor.localnet1.conf
```

Видалити всі правила з якоря:

```
# pfctl -a "localnet1" -F rules
```

Anchor в файлі конфігурації викликається за місцем визначення. Як і звичайне правило, можна зробити anchor з умовами:

```
anchor localnet1 in on $IntIf1 from any to any
```

Можна визначити якір фільтрації разом з умовами прямо за місцем, використовуючи фігурні дужки:

```
anchor "localnet1" in on $IntIf1 {
    pass in quick on $IntIf1 from $IntIf1Net to $IntIf2Net tag
PASS
    pass in quick on $IntIf1 proto tcp from ($IntIf1:network) to
any \
port { $permit_tcp_ports } tag PASS
}
```

Якщо всередині якоря в правилі вказана опція quick, то обробка припиняється для всіх правил файрволла, в тому числі і після нього. Можна так само вказати опцію quick для визначення anchor, тоді обробка зупиниться на останньому правилі цього набору. Мітки так само збережуться.

Усередині якоря можна так само визначати таблиці. Вони будуть видні в цьому і всіх вкладених якорях.

Тепер можна перенести всі правила, які стосуються входить трафіку на інтерфейсі $ IntIf1 в anchor, що завантажується з файлу. Не забуваємо макроси.

```
###### macros section ######
IntIf1="em0" # локальний 0
IntIf1Ip="172.17.3.254" # IP локального
IntIf1Net="172.17.3.0/24"
IntIf2Net="172.17.3.0/24"
ExtSrv="172.17.1.2"
permit_tcp_ports="22,53"
permit_udp_ports="53,123"
web_ports = "http,https"
###### filter section ######
# дозволяємо між локалками все
pass in quick on $IntIf1 from $IntIf1Net to $IntIf2Net tag PASS
```

Необхідно надати доступ привілейованим користувачам.

```
pass in on $IntIf1 from <privileged_If1> to any tag PASS
# дозволяємо icmp всім
pass in quick on $IntIf1 proto tcp from ($IntIf1:network) to any
port { $permit_tcp_ports } tag PASS
# дозволяємо всі корпоративні ресурси
pass in quick on $IntIf1 proto tcp from ($IntIf1:network) to
$ExtSrv port { $web_ports } tag PASS
# правила для сайтів
pass in quick on $IntIf1 from ($IntIf1:network) to <sites_for_all>
tag PASS
block in quick from <bad_users> to any
block in quick from { $IntIf1Net } to <bad_sites>
pass in quick from { $IntIf1Net } to any tag PASS
```

Тепер можна видалити ці правила з конфігурації, залишивши тільки anchor, а решта, що відносяться до іншого інтерфейсу, визначити в якорі за місцем.

Таким чином можна оптимізувати за швидкістю, упорядкувати, або просто зробити конфігурацію PF коротше і більш читабельною. Крім того, правильно створеної конфігурацією досить легко управляти динамічно, без перезавантаження PF.

Policy Based Routing (PBR), Source Based Routing (SBR), умовна маршрутизація, маршрутизація на основі політик, всі ці поняття, по суті, рівнозначні і описують одну можливість. Вибирати маршрути, або таблиці маршрутизації для трафіку на основі правил фаєрвола.

Можна, наприклад, відправити різні локальні мережі через різних провайдерів, або влаштувати розподіл трафіку між VPN каналами. Розглянемо обидва ці варіанти.

Для зменшення файлів конфігурації і більшої наочності, правила фільтрації будуть максимально спрощені.

Потрібно відправити трафік від однієї локальної мережі через одного провайдера.

Другу локальну мережу - через другого провайдера.

Так само потрібно прокинути веб-порти сервера через обох провайдерів так, щоб були доступні обидва IP адреси.

У цю схему додамо другий офіс, у якого так само 2 провайдера. Між офісами є 2 тунелі.

Потрібно пустити трафік до другого офісу через VPN1, крім сервера, який повинен працювати через VPN2

Вирішимо ці завдання засобами виключно PF і окремо - засобами множинних таблиць маршрутизації.

Далі - трансляції. Так як у нас 2 провайдера, nat потрібен для кожного.

```
#NAT для клієнтів
nat pass on $ExtIf1_ from { $IntIf1Net, $IntIf2Net } to any ->
$ExtIf1_Ip1_
nat pass on $ExtIf2_ from { $IntIf1Net, $IntIf2Net } to any ->
$ExtIf2_Ip1
```

Кидок портів - http (s) і порт 2 022 з кожного провайдера на порт 22 внутрішнього сервера.

```
#прокид  на сервер
rdr on $ExtIf1_ proto tcp from any to $ExtIf1_Ip1_ port { http,
https} tag DSTNAT -> $IntSRV
rdr on $ExtIf2_ proto tcp from any to $ExtIf2_Ip1 port { http,
https} tag DSTNAT -> $IntSRV
rdr on $ExtIf1_ proto tcp from any to $ExtIf1_Ip1_ port { 2022 }
tag DSTNAT -> $IntSRV port ssh
rdr on $ExtIf2_ proto tcp from any to $ExtIf2_Ip1 port { 2022 }
tag DSTNAT -> $IntSRV port ssh
```

Конфігурації PF офісів будуть практично однакові. Відмінності стосуються макросів і кількості локальних мереж.

3.3 Робота з декількома провайдерами

Основна проблема роботи через двох провайдерів в тому, що в нашій таблиці маршрутизації є тільки один шлюз за замовчуванням. Під FreeBSD немає метрик маршрутів, так що дефолтний маршрут дійсно може бути тільки

один. Відповідно, наш роутер спробує відповісти саме з того інтерфейсу, куди вказує цей маршрут. Тобто, отримали пакет на re1 і відправили з re1 відповідь. Так маршрутизація працювати не буде.

Вирішується питання дуже легко. У правила pass в PF є опція reply-to (Interface [Gateway]), яка вказує, куди відправити у відповідь пакет. Всі опції маршрутизації краще вказувати на вхідних правилах.

Приходить перший пакет, проходить за правилом. PF створює стейт, з'єднання, за яким пакети підуть як в одну, так і в іншу сторону. Пакет відправляється за адресою. На нього приходить відповідь. І ось ця відповідь маршрутизируется туди, куди вказано в опції reply-to. Дозволяють правила з Інтернет.

```
pass in quick on $ExtIf1_ proto tcp to { $ExtIf1_Ip1_ } port 22
```

І для другого інтерфейсу:

```
pass in quick on $ExtIf2_ reply-to ( $ExtIf2_ $ExtIf2_Gw ) proto
tcp to { $ExtIf2_Ip1 } port 22
```

Цього достатньо, щоб отримані на 22 порту з другого провайдера з'єднання працювали. Додамо icmp

```
pass in quick on $ExtIf1_ proto icmp
pass in quick on $ExtIf2_ reply-to ($ExtIf2_ $ExtIf2_Gw ) proto
icmp
```

І дозволу для проброшенних портів, тег дозволить обійтися мінімальною кількістю правил для будь-якої кількості проброшенних портів:

```
pass in quick on $ExtIf1_ tagged DSTNAT
pass in quick on $ExtIf2_ reply-to ($ExtIf2_ $ExtIf2_Gw ) tagged
DSTNAT
```

Тепер дозволу ходити до нас з другого філії:

```
pass in quick on $ExtIf1_ from $RemoteExtIp1 tag PASS
pass in quick on $ExtIf2_ reply-to ($ExtIf2_ $ExtIf2_Gw ) from
$RemoteExtIp2 tag PASS
```

Цього достатньо, щоб входять сесії будувалися симетрично. Відповідний пакет буде завжди відправлений через той же інтерфейс, через який отримано перший пакет сесії.

4 ТЕХНІКО-ЕКОНОМІЧНИЙ РОЗДІЛ

Метою техніко – економічного розділу кваліфікаційної роботи є здійснення економічних розрахунків, спрямованих на визначення економічної ефективності програмного додатку створення та візуалізації алгоритмів у вигляді блок-схем на основі використання механік теорії графів та прийняття рішення про його подальший розвиток і впровадження або ж недоцільність проведення відповідної розробки. Для проведення даного дослідження необхідно провести ряд розразунків.

4.1 Розрахунок витрат на розробку програмного додатку

Витрати на розробку і впровадження програмного додатку (K) на основі запропонованих алгоритмів та розробленій структурі, що враховують результати аналізу програм-аналогів включають:

$$K = K_1 + K_2,$$

де K_1 – витрати на розробку апаратного та програмного забезпечення грн.;

K_2 – витрати на відлагодження і дослідну експлуатацію програми рішення задачі на комп’ютері, грн.

Витрати на розробку апаратних та програмних засобів включають:

- витрати на оплату праці розробників ($B_{ОП}$);
- витрати на відрахування у спеціальні державні фонди ($B_ф$);
- витрати на матеріали та комплектуючі ($П_в$);
- накладні витрати (H);
- інші витрати ($І_в$)
- витрати на використання комп’ютерної техніки($B_{КТ}$) .

Розрахунок витрат на оплату праці.

Витрати на оплату праці включають заробітну плату (ЗП) всіх категорій працівників, безпосередньо зайнятих на всіх етапах проектування. Розмір ЗП обчислюється на основі трудоємності відповідних робіт у людино-днях та середньої ЗП відповідних категорій працівників.

У розробці проектного рішення задіяні наступні спеціалісти - розробники, а саме: керівник проекту; студент-дипломант; консультант техніко-економічного розділу (таблиця 4.1).

Таблиця 4.1 – Вихідні дані для розрахунку витрат на оплату праці

№п/п	Посада виконавців	Місячний оклад, грн.
1	Керівник ДП, ст. викладач	7449
2	Консультант техніко-економічного розділу, ст. викладач	7449
3	Студент	1400

Витрати на оплату праці розробників проекту визначаються за наступною формулою (4.1):

$$B_{ОП} = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} \cdot t_{ij} \cdot C_{ij} \ , \qquad (4.1)$$

де n_{ij} – чисельність розробників i-ої спеціальності j-го тарифного розряду;

t_{ij} – затрачений час на розробку проекту співробітником i-ої спеціальності j-го тарифного розряду, год;

C_{ij} – годинна ставка працівника i-ої спеціальності j-го тарифного розряду.

Середньо годинна ставка працівника може бути розрахована за такою формулою (4.2):

$$C_{ij} = \frac{C_{ij}^{0}(1+h)}{РЧ_{i}}, \quad (4.2)$$

де C_{ij} – основна місячна заробітна плата розробника *i*-ої спеціальності *j*-го тарифного розряду, грн.;

h – коефіцієнт, що визначає розмір додаткової заробітної плати (при умові наявності доплат);

$РЧ_i$ - місячний фонд робочого часу працівника *i*-ої спеціальності *j*-го тарифного розряду, год. (приймаємо 168 год.).

Коефіцієнт h, який визначає розмір додаткової заробітної плати, для керівника та консультанта техніко-економічного розділу дорівнює 0,47.

Результати розрахунку записують до таблиці 4.2.

Таблиця 4.2 – Розрахунок витрат на оплату праці

№ п/п	Посада виконавців	Час розробки, год	Погодинна заробітна плата, грн/год.	Витрати на розробку, грн
1	Керівник ДП, старший викладач	16	64,3	1028,8
2	Консультант техніко-економічного розділу, доцент	2	59,9	119,8
3	Студент	144	8,33	1199,52
Разом				2348,12

Відрахування на соціальні заходи. Величну відрахувань у спеціальні державні фонди визначають у відсотковому співвідношенні від суми основної та додаткової заробітних плат. Згідно діючого нормативного законодавства сума відрахувань у спеціальні державні фонди складає 20,5% від суми заробітної плати:

$$B_{Ф} = \frac{20,5}{100} \cdot 2348,12 = 481,36 \text{ грн.}$$

Розрахунок витрат на матеріали та комплектуючі.

Загальна сума витрат на матеріальні ресурси (B_M) визначається за формулою (4.3):

$$B_M = \sum_{i=1}^{n} K_i \cdot Ц_i, \qquad (4.3)$$

де K_i – витрата i-го типу матеріалу, натуральні одиниці вимірювання;

$Ц_i$ – ціна за одиницю i-го типу матеріалу, грн.;

i – тип матеріального ресурсу;

n – кількість типів матеріальних ресурсів.

Таблиця 4.3 – Зведені розрахунки матеріальних витрат

№ п/п	Найменування матеріальних ресурсів	Од. виміру	Факт. витрачено матеріалів	Ціна за одиницю, грн.	Сума, грн	Транспортні витрати (10% від суми)	Загальна сума, грн
	Допоміжна література	шт	1	1000	1000	100	1100
	Папір (формат А4)	уп	2	100	200	20	220
	Ручка кулькова	шт	2	10	20	2	22
	Олівець простий	шт	2	10	20	2	22
	Диски CD-R	шт	2	20	40	4	44
	Зошит, 96 арк	шт	1	50	50	5	55
	Тонер для принтера	уп	1	90	90	9	99
	Канцелярскі маркери (синій, зелений)	шт	2	20	40	4	44
	Р а з о м						1606,00

Витрати на використання комп’ютерної техніки.

Витрати на використання комп’ютерної техніки($B_{КТ}$) включають витрати на амортизацію комп’ютерної техніки, витрати на користування програмним забезпеченням, витрати на електроенергію, що споживається комп’ютером. За даними обчислювального центру ЗУНУ для комп’ютера типу IBM PC/ATX вартість години роботи становить 12 грн. Середній щоденний час роботи на комп’ютері – 2 години. Розрахунок витрат на використання комп’ютерної техніки приведений в таблиці 4.4.

Таблиця 4.4 – Розрахунок витрат на використання комп’ютерної техніки

№ п/п	Назва етапів робіт, при виконанні яких використовується комп’ютер	Час використання комп’ютера, год.	Витрати на використання комп’ютера грн.
1	Проведення досліджень та оформлення їх результатів у вигяді звіту	60	7200
2	Оформлення техніко-економічного розділу	8	96
3	Оформлення ДП	12	144
Разом		80	960

Накладні витрати.

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими відсотками до витрат на оплату праці. Середньостатистичний відсоток накладних витрат приймемо 150% від заробітної плати:

$$H = 1{,}5 \cdot 2348{,}12 = 3522{,}18 \text{ (грн)}.$$

Інші витрати.

Інші витрати є витратами, які не враховані в попередніх статтях. Вони становлять 10% від заробітної плати:

$$I_B = 2348{,}12 \cdot 0{,}1 = 234{,}81 \text{ (грн).}$$

Витрати на розробку програмного забезпечення складають:

$$K_1 = B_{ОП} + B_Ф + B_M + H + I_B + B_{КТ},$$

$$K_1 = 2348{,}12+481{,}36+1606{,}00+3522{,}18+234{,}81+960{,}00=8849{,}47 \text{ (грн)}.$$

Витрати на відлагодження і дослідну експлуатацію програмного продукту визначаємо за формулою (4.4):

$$K_2 = S_{м.г.} \cdot t_{від} \qquad (4.4)$$

де $S_{м.г.}$ – вартість однієї машино-години роботи ПК, грн./год;

$t_{від}$ – комп'ютерний час, витрачений на відлагодження і дослідну експлуатацію створеного програмного продукту, год.

Загальна кількість днів роботи на комп'ютері дорівнює 30 днів. Середній щоденний час роботи на комп'ютері – 2 години. Вартість години роботи комп'ютера дорівнює 12 грн., тому $K_2 = 12 \cdot 60 = 720$ грн.

4.2 Визначення експлуатаційних витрат

Для оцінки економічної ефективності розроблювальної програмної системи слід порівняти її з аналогом, тобто існуючим програмним забезпеченням ідентичного функціонального призначення.

Експлуатаційні одноразові витрати по програмному забезпеченню і аналогу включають вартість підготовки даних і вартість роботи комп'ютера (за час дії програми):

$$E_{\text{П}} = E_{1\text{П}} + E_{2\text{П}},$$

де $E_{П}$ – одноразові експлуатаційні витрати на ПЗ (аналог), грн.;

$E_{1П}$ – вартість підготовки даних для експлуатації ПЗ (аналогу), грн.;

$E_{2П}$ – вартість роботи комп'ютера для виконання проектного рішення (аналогу), грн.

Річні експлуатаційні витрати $B_{ЕП}$ визначаються за формулою:

$$B_{E\text{П}} = E_{\text{П}} * N_{\text{П}},$$

де $N_{П}$ – періодичність експлуатації ПЗ (аналогу), раз/рік.

Вартість підготовки даних для роботи на комп'ютері визначається за формулою:

$$E_{1\text{П}} = \sum_{l=1}^{n} n_i t_i c_i ,$$

де i – категорії працівників, які приймають участь у підготовці відповідних даних (i=1,2,...n);

n_i – кількість працівників i-ої категорії, осіб.;

t_i – трудомісткість роботи співробітників i-ої категорії по підготовці даних, год.;

c_i – середнього годинна ставка працівника i-ої категорії з врахуванням додаткової заробітної плати, що знаходиться із співвідношення:

$$c_i = \frac{c_i^0 (1+b)}{m},$$

де c_i^0 – основна місячна заробітна плата працівника *i*-ої категорії, грн.;

b – коефіцієнт, який враховує додаткову заробітну плату (приймемо 0,57);

m – кількість робочих годин у місяці, год.

Для роботи з даними як для проектного рішення так і аналогу потрібен один працівник, основна місячна заробітна плата якого складає: *c* = 6061 грн. Тоді:

$$c_1 = \frac{6061(1+0{,}57)}{22*8} = 56{,}64 \text{ грн/год}$$

Трудомісткість підготовки даних для проектного рішення складає 1 год., для аналога 1,5 год.

Таблиця 4.5 – Розрахунок витрат на підготовку даних та реалізацію проектного рішення на комп'ютері

№	Час роботи співробітників, год.	Середньогодинна заробітна плата, грн./год.	Витрати , грн.
	Проектне рішення		
1	1	56,64	56,64
	Аналог		
1	1,5	56,64	84,95

Витрати на експлуатацію комп'ютера визначається за формулою:

$$E_{2П} = t * S_{МГ}$$

Де *t* – витрати машинного часу для реалізації рішення (аналогу), год.;

$S_{МГ}$ – вартість однієї години роботи комп'ютера, грн./год.

$E_{2П}$ = 1 * 12 = 12 грн.; $E_{2А}$=1,5 * 12 = 18 грн.

$E_{П}$=56,64+12=68,64 грн. ; $E_{А}$=84,95+12 = 96,95 грн.

$B_{ЕП}$= 68,64 * 252=17297,28 грн. ; $B_{ЕА}$=96,95 * 252=24431,4 грн.

Обчислення накладних витрат.

Накладні витрати пов'язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці.

В залежності від організаційно-правової форми діяльності господарюючого суб'єкта, накладні витрати можуть становити 60–100 % від суми основної та додаткової заробітної плати працівників.

$$H_B = 0,7 * B_{ОП} = 0,7 * (c_1 * 168), \quad (4.7)$$

де H_B – накладні витрати.

$$H_B=0,7*9515,52=6660,86 \text{ грн.}$$

Складання кошторису витрат та визначення собівартості. Результати проведених розрахунків зведемо у таблицю 4.6.

Таблиця 4.6 – Кошторис витрат ($B_{КС}$)

№ п/п	Найменування витрат	Сума витрат, грн.
1	Витрати на оплату праці ($B_{ОП}$)	2348,12
2	Відрахування у спеціальні державні фонди ($B_Ф$)	481,36
3	Витрати на матеріали та комплектуючі ($B_М$)	1606,00
4	Накладні витрати на розробку (H)	3522,18
5	Інші витрати (I_B)	234,81
6	Витрати на відлагодження і дослідну експлуатацію програмного продукту (K_2)	720
7	Накладні витрати експлуатацію (H_B)	6660,86
8	Річні експлуатаційні витрати ($B_{ЕА}$)	24431,4
Разом		40004,73

Розрахунок ціни проекту.

Договірна ціна ($Ц_Д$) для проектних рішень розраховується за формулою (4.8):

$$Ц_Д = B_{КС} \cdot \left(1 + \frac{p}{100}\right), \quad (4.8)$$

де $B_{КС}$ – кошторисна вартість, грн.;

p – середній рівень рентабельності, % (приймаємо 20% за погодженням з керівником).

$$Ц_Д = 40004{,}73 \cdot (1+0{,}2) = 48005{,}68 \text{ грн.}$$

4.3 Визначення економічної ефективності та терміну окупності

Економічна ефективність ($E_ф$) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E_ф = \frac{П}{B_{КС}}, \quad (4.9)$$

де $П = Ц_Д - B_{КС}$ – прибуток, грн.;

$B_{КС}$ – кошторисна вартість, грн..

$$E_ф = 8000{,}95 \text{ грн.} / 40004{,}73 \text{ грн.} = 0{,}2.$$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень ($Тр$):

$$T_P = \frac{1}{E_P} \ . \qquad (4.10)$$

Тобто: $T_P = 1/0{,}2 = 5$р.

Прийнятним вважається термін окупності близький до 7 років.

Розраховані економічні показники проекту занесемо до таблиці 4.7.

Таблиця 4.7 – Економічні показники розробки

№ п/п	Показник	Значення
1.	Собівартість, грн.	40004,73
2.	Плановий прибуток, грн.	8000,95
3.	Ціна, грн.	48005,68
4.	Економічна ефективність	0,2
5.	Термін окупності, рік	5

Враховуючи основі економічні показники з таблиці 4.7, можна зробити висновок, що при економічній ефективності 0,2 та терміні окупності – 5 роки проводити роботи по впровадженню даного програмного додатку є доцільним та економічно вигідним.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи отримано такі висновки:

1. В кваліфікаційній роботі на основі аналізу функцій Cisco Catalyst проведено вибір необхідного мережевого обладнання для побудови мережі, досліджено використання портів та можливості налаштування маршрутизаторів Cisco Catalyst, що дало можливість обґрунтувати вибір маршрутизатора для побудови мережі.

2. Проведено аналіз функцій передньої панелі маршрутизаторів Cisco Catalyst для виконання необхідних налаштувань та встановлення обраного обладнання.

3. Проаналізовано аналіз налаштування портів комутатора Cisco Catalyst, оскільки в них присутній ряд портів подвійного призначення, що дало можливість оптимально оцінити функції комутаторів.

3. Проведено налаштування фаєрволу засобами FreeBSD, а саме налаштування маршрутизації пакетів, використання макросів та списків для правильного функціонування мережі.

4. Проаналізовано розширені можливості пошуку та фільтрації пакетів для налаштування контролю доступу за допомогою правил ACL, що дало можливість побудувати захищену комп’ютерну мережу .

5. Виконано налаштування якорів та досліджена робота з декількома провайдерами для налаштування безперебійного доступу до мережі інтернет.

6. Проведено тестування розроблених програмних налаштувань фаєрволу FreeBSD, що підтвердило ефективність розроблених налаштувань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фучило (Бабій) Ю.С., Кузик В.М., Раїнчук В.В. Система фільтрації контенту на базі Cisco і FreeBSD. Збірник матеріалів проблемно-наукової міжгалузевої конференції «Автоматизація та компютерно інтегровані технології» (АКІТ - 2020), Тернопіль, 2020. С.33-37.

2. Методичні рекомендації до виконання кваліфікаційної роботи з освітнього ступеня “Бакалавр” спеціальності 123 «Комп’ютерна інженерія» галузі знань 12 Інформаційні технології / О.М. Березький, Л.О.Дубчак, Г.М. Мельник, Ю.М. Батько / Під ред. О.М. Березького. Тернопіль: ЗУНУ, 2020. 60с.

3. Методичні вказівки до виконання практичних робіт з дисципліни «Техніко-економічне обґрунтування розробки комп’ютерних систем”/ Н.Я. Савка, І.Р. Паздрій / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 40 с.

4. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп’ютерна інженерія» / І.В. Гураль, Л.О. Дубчак / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.

5. Bilski P. Distributed real-time measurement system using time-triggered network approach [Текст] / P. Bilski W. Winiecki // Комп`ютинг. – 2008. – липень (№2). – P.22-29.

6. Eren E. Security assessment of IEEE 802.16 (WIMAX) - a short comprasion between IEEE 80216d and 802.16e [Текст] / E. Eren // Комп`ютинг. – 2008. – липень (№2). – P.91-99.

7. Jose Chilo. Wireless data acquisition system using bluetooth technology for infrasonic records [Текст] / Chilo. Jose, Lindblad Thomas // Комп`ютинг. – 2008. – липень (№2). – P.18-21.

8. Kotenko I. Multi-agent simulation of attacks and defense mechanisms in computer networks [Текст] / I. Kotenko // Комп`ютинг. – 2008. – липень (№2). – P.35-43.

9. LillD A. Development of cooperative esafety-system using communication and localization [Текст] / LillD A. Gutjahr A. Sikora // Комп`ютинг. – 2008. – липень (№2). – P.84-90.

10. Sergey Y. Yurish Low-cost, intelligent data acquisition for QCM and other resonator-based bio- and chemical sensors [Текст] / Y. Yurish Sergey // Комп`ютинг. – 2008. – липень (№2). – P.9-17.

11. Банчук М. В. Комп'ютеризація як інструмент реалізації політики забезпечення якісної вищої медичної освіти України [Текст] / М. В. Банчук // Інвестиції : практика та досвід. – 2011. – липень (№14). – С. 93-95.

12. Васильська М. В. Узагальнена імітаційна модель розвитку системи мобільного зв'язку [Текст] / М. В. Васильська В. М. Кичак В. А. Северілов // Вісник Вінницького політехнічного інституту. – 2011. – №3. – С. 166-172.

13. Високопродуктивні комп'ютерні системи [Електронний ресурс] : консп. лекцій. – Тернопіль : ТНЕУ, 2006. – 85 с. – Режим доступу : http://library.tneu.edu.ua/images/stories/predmety/літв/високопродуктивні%20комп'ютерні%20системи/fkit_kiosu_vks_ksm_lek.pdf.

14. Глосарій з навчальної дисципліни "Адміністрування та моніторинг комп'ютерних мережних систем" [Текст] / укл. С. В. Кавун В. В. Огурцов. – Х. : ХНЕУ, 2007. – 324 с.

15. Дибкова Л. М. Інформатика і комп'ютерна техніка [Текст] : навч. посіб. / Л. М. Дибкова. – 4-те вид., стер. – К. : Академвидав, 2012. – 464 с. – (Альма-матер).

16. Козловський А. В. Комп'ютерна техніка та інформаційні технології [Текст] : навч. посіб. / А. В. Козловський Ю. М. Паночишин Б. В. Погріщук. – 2-ге вид., стер. – К. : Знання, 2012. – 464 с. – Режим доступу : http://library.tneu.edu.ua/images/stories/zmist/2014/літк/Ком'ютерна техніка та інформаційні технології.pdf.

17. Комунікаційні системи в економіці [Електронний ресурс] : опорн. консп. лекцій. – Тернопіль : ТНЕУ, 2006. – 67 с. – Режим доступу : http://library.tneu.edu.ua/images/stories/predmety/літк/комунікаційні%20системи%20в%20економіці/fkit_kiosu_dkse_ek_lek.pdf.

18. Методичні рекомендації до виконання дипломного проекту з освітньо-кваліфікованого рівня «Бакалавр» наплямку підготовки 6.050102 «Комп'ютерна інженерія» фахового спрямування «Комп'ютерні системи та мережі»/О.М. Березький, Л.О. Дубчак, Р.Б. Трембач, Г.М. Мельник, Ю.М. Батько, С.В. Івасьєв/Під ред. О.М. Березького – Тернопіль: ТНЕУ, 2013.-65с.

19. Методичні рекомендації до написання техніко-економічного розділу дипломних проектів з освітньо-кваліфікованого рівня «Бакалавр» наплямку підготовки 6.050102 «Комп'ютерна інженерія» фахового спрямування «Комп'ютерна інженерія»/І.П. Паздрій-Тернопіль: ТНЕУ, 2014.-36 с.

20. Мамченко С. Д. Основи інформатики та обчислювальної техніки: практикум [Текст] : навч. посіб. / С. Д. Мамченко В. А. Одинець. – К. : Знання, 2007. – 292 с. – (Вища освіта XXI століття). – Режим доступу : http://library.tneu.edu.ua/images/stories/zmist/2012/літо/основи інформатики мамченко 2007.pdf.

21. Мінухін С. В. Комп'ютерні мережі. Загальні принципи функціонування комп'ютерних мереж [Текст] : навч. посіб. / С. В. Мінухін, С. В. Кавун, С. В. Знахур. – Х. : ХНЕУ, 2008. – 208 с. – Режим доступу : http://library.tneu.edu.ua/images/stories/zmist/2012/літк/компютерні мережі мінухін 2008.pdf.

22. Моделювання комп'ютерних систем [Електронний ресурс] : опорн. консп. лекцій. – Тернопіль : ТНЕУ, 2006. – 74 с. – Режим доступу : http://library.tneu.edu.ua/images/stories/predmety/літм/моделювання%20компютерних%20систем/fkit_kbit_2006_sksm_dmks_lek.pdf.

23. Наливайко Н. Я. Інформатика [Текст] : навч. посіб. / Н. Я. Наливайко. – К. : ЦУЛ, 2011. – 577 с. – Режим доступу : http://library.tneu.edu.ua/images/stories/zmist/2013/літі/Інформатика. Наливайко Н. Я..pdf.

24. Олифер В. Г. Компьютерные сети. Принципы, технологии, протоколы [Текст] : учеб. пособ. / В. Г. Олифер, Н. А. Олифер. – 4-е изд. – С-Пб. : Питер, 2012. – 944 с. – (Учебник для вузов. Стандарт третьего

поколения). – Режим доступу : http://library.tneu.edu.ua/images/stories/zmist/2012/літк/компьютерные сети олифер 2012.pdf.

25. Основи програмування та алгоритмічні мови [Електронний ресурс] : консп. лекцій. – Тернопіль : ТНЕУ, 2006. – 208 с. – Режим доступу : http://library.tneu.edu.ua/images/stories/predmety/літо/основи%20програмування%20та%20алгоритмічні%20мови/fkit_kkn_doptam_spzas_lek.pdf.

26. Паркер Стив Компьютеры [Текст] / Стив Паркер. – М. : Махаон, 1998.

27. Сейдаметова З. Нова версія стандарту Computer Science Curricula: еволюція базисного корпусу знань за чверть століття [Текст] / З. Сейдаметова, В. Темненко // Вища школа. – 2012. – № 12. – С. 54-64.

28. Теслюк В. М. Дослідження і проектування комп'ютерних систем та мереж [Електронний ресурс] : консп. лекцій / В. М. Теслюк. – Тернопіль : ТНЕУ, 2012. – 62 с. – Режим доступу : http://library.tneu.edu.ua/images/stories/predmety/літд/дослідження%20і%20проектування%20комп'ютерних%20систем%20та%20мереж/Досл%20і%20проектув%20комп%20сист%20та%20мереж.pdf.

29. Хассел Д. Администрирование Windows server 2003 [Текст] / Д. Хассел. – С-Пб. : Питер, 2006. – 576 с . – Режим доступу : http://library.tneu.edu.ua/images/stories/zmist/2012/літа/администрирование виндовс хассел 2006.pdf.

30. Иванов, Д. В. Виртуализация общества. [Текст] / Д.В. Иванов. - М.: Петербургское Востоковедение, 2002. - 224 с.

31. Ленгоун, Д. Віртуалізація настільних комп`ютерів за допомогою VMware View 5: моногр.[Текст] / Д. Ленгоун. - М.: ДМК Пресс, 2013. - 268 с.

32. Віртуальні машины [Електронний ресурс] // Информаційний сайт про високі технології. Режим доступу: http://all-ht.ru/inf/vpc/p_0_0.html

33. Виртуальные машины на платформе Microsoft Virtual PC 2007 [Електронний ресурс] // WindowsFAQ.ru: FAQ, статьи, обзоры программ,

операционных систем и серверного программного обеспечения. Режим доступу: http://www.windowsfaq.ru/content/view/566/46/

34. Ежова, Е. Н. Виртуализация как средство деформации и трансформации пространства и времени в медиа-рекламной картине мира [Текст] / Е.Н. Ежова. - Москва: РГГУ, 2010. - 303 с..

35. Диттнер, Р. Виртуализация и Microsoft Virtual Server 2005 [Текст] / Р. Диттнер , К. Мейджорз , М. тен Селдан, Т.Гротениус,Д. Рул мол., Дж. Грин. - М.: Бином-Пресс, 2008. - 432 с..

36. Мельниченко А. Обучение вместо программирования. Электронные компоненты и системы [Текст] / А. Мельниченко. – 2004. – №12. – С.36-40.

37. Введение в виртуализацию. Часть 1 [Електроний ресурс] //. Режим доступу: http://interface31.ru/tech_it/2012/07/vvedenie-v -virtualizaciyuchast -1.html

38. Как работают виртуальные машины – принцип работы [Електронний ресурс] // Режим доступу: http://winsetting.ru/kak-rabotayut-virtualnye-mashinyprincip-raboty.html

39. Баричев С. Г.Основы современной криптографии[Текст]/ С. Г. Баричев, В. В. Гончаров, Р. Е. Серов. — М.: ДИАЛОГ-МИФИ, 2011. — 175 с.

40. Панасенко С.П. Алгоритмы шифрования. Специальный справочник[Текст]/ С.П. Панасенко – СПб.: БХВ-Петербург, 2009 – 576 с

41. Столлингс, В. Криптография и защита сетей: принципы и практика[Текст]/ В. Столлингс — Вильямс, 2001 - 698 с.

42. Advanced Encryption Standart [Електронний ресурс]// Режим доступу: https://uk.wikipedia.org/wiki/Advanced_Encryption_Standard

43. Баричев С. Г. Основы современной криптографии — 3-е изд. Стандарт AES. Алгоритм Rijdael [Текст]/ С. Г. Баричев, В. В. Гончаров, Р.Е. Серов — М.: Диалог-МИФИ, 2011. — С. 30–35. — 176 с.

44. Rives, Shamir and Adelman [Електронний ресурс]// Режим доступу: https://ru.wikipedia.org/wiki/RSA

45. Daemon Tools – найкращі програми для роботи з образами дисків [Електронний ресурс] // Режим доступу: https://www.daemon-tools.cc

46. Віртуальний диск. Яку програму найкраще обрати [Електронний ресурс] // Режим доступу: http://pcpro100.info/virtualnyiy-disk

47. Методичні рекомендації до виконання дипломного проекту з освітньо-кваліфікаційного рівня “магістр” напряму підготовки 6.050102 «Комп’ютерна інженерія» фахового спрямування «Комп’ютерні системи та мережі» / О.М. Березький, Л.О.Дубчак, Р.Б. Трембач, Г.М. Мельник, Ю.М. Батько, С.В. Івасьєв / Під ред. О.М. Березького. - Тернопіль: ТНЕУ, 2016.–65 с.

48. Климова, Л. М. Delphi 7. Основы программирования. Решение типовых задач. Самоучитель [Текст] / Л.М. Климова — КУДИЦ-Образ, 2017. - 480 с.

49. Культин Н. Основы программирования в Delphi 7 [Текст] /Н. Культин. - М.: СПб: БХВ, 2011. - 608 с.

50. Санников, Е. В. Курс практического программирования в Delphi. Объектно-ориентированное программирование [Текст] / Е.В. Санников. - М.: Солон-Пресс, 2013. - 188 с.

51. Осипов, Д. Delphi. Профессиональное программирование [Текст] / Д.Осипов. - М.: Символ-плюс, 2013. - 820 с.

52. Осипов, Д. Delphi. Программирование для Windows, OS X, iOS и Android [Текст] / Д. Осипов. - М.: "БХВ-Петербург", 2014. - 464 с.