НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«До захисту допущено»
В.о. завідувача кафедри

_____ М.В.Грайворонський
(підпис)

“_____”_____2019 р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.170101 «Безпека інформаційних і комунікаційних систем»

на тему: Аналіз захищенності SDN мереж та протоколу OpenFlow _____

_____

_____

Виконав (-ла): студент (-ка) __4__ курсу, групи ____ФБ-51_____
(шифр групи)

Тімофєєв Руслан Сергійович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доц., к.ф.-м.н., доц. Грайворонський М.В. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент доц., к.т.н., доц. Волокита А.М._____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.170101 «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ М.В.Грайворонський
(підпис)
«___»_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту

Тімофєєва Руслана Сергійовича _____
(прізвище, ім'я, по батькові)

1. Тема роботи _____
Аналіз захищенності SDN мереж та протоколу OpenFlow_____
_____,
науковий керівник роботи _____
__доц., к.ф.-м.н., доц. Грайворонський М.В._____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «_27.05___ » 2019 р. № _1414-С_

2. Термін подання студентом роботи 10 червня 2019 р.

3. Вихідні дані до роботи _____
_____
_____

4. Зміст роботи _____
_____
_____
_____
_____

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) _____
_____
_____
_____

6. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка

Студент _____ ___Тімофєєв Р.С._____
(підпис) (ініціали, прізвище)

Науковий керівник роботи _____ __Грайворонський М.В.__
(підпис) (ініціали, прізвище)

РЕФЕРАТ

бакалаврської роботи на Тімофєєва Руслана Сергійовича на тему «Аналіз захищенності SDN мереж та протоколу OpenFlow».

Метою даної роботи є дослідження основних принципів роботи архітектури SDN та протоколу OpenFlow, визначення вразливих місць, побудова моделі загроз. Традиційні ієрархічні мережі більше не здатні задовольняти потреби сучасного світу великих даних, мобільних та IoT пристроїв і хмарних технологій. Для вирішення проблем гнучкості та адаптивності мереж була розроблена архітектура SDN. У роботі розглядаються принципи архітектури SDN, механізми роботи протоколу OpenFlow, описуються їх вразливості, надаються рекомендації по протидії атакам. Також, наводиться приклаж застосунку для автоматичної протидії атакам DoS.

Загальний обсяг роботи __ сторінки, __ ілюстрацій, __ таблиць та __ бібліографічних найменувань.

Ключові слова: SDN, OpenFlow, mininet, безпека.

ABSTRACT

to the bachelor thesis by Timofieiev Ruslan Serhiyovych on «___».

The purpose of the thesis is to study the main principles of the SDN architecture and the OpenFlow protocol, to identify vulnerabilities and to create a threat model for SDN networks. Today, traditional hierarchical networks can no longer meet the needs of Big Data, mobile and IoT devices and cloud techologies. An SDN architecture was developed to address the issues of flexibility and adaptability of networks. The paper considers the principles of SDN architecture, the mechanisms of the OpenFlow protocol, describes their vulnerabilities and provides recommendations for counteracting attacks. Also, the attachment of an application for automatic counteraction to DoS attacks is provided.

The total amount of work: __ pages, __ illustrations, __ tables and __ bibliographic titles.

Keywords: SDN, OpenFlow, mininet, security.

Зміст

Перелік умовних позначень, символів, одиниць, скорочень і термінів 7

Вступ 8

1 Огляд архітектури SDN і протоколу OpenFlow 11

1.1 Рівні архітектури 11

1.2 Протоколи південного інтерфейсу 12

1.3 Протоколи північного інтерфейсу 13

1.4 Протокол OpenFlow 14

Висновки до розділу 1 20

2 Безпека SDN мереж та протоколу OpenFlow 21

2.1 Переваги архітектури SDN з точки зору безпеки 21

2.2 Вразливості архітектури SDN 25

2.3 Вразливості протоколу OpenFlow 29

2.3 Модель загроз мереж SDN 38

Висновки до розділу 2 39

3 Створення моделі SDN мережі 40

3.1 Моделювання DoS атаки 41

3.2 Автоматичне запобігання DoS атак 44

Висновки до розділу 3 46

Висновки 47

Додаток 1 53

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SDN – Software Defined Networking;

OFDP – OpenFlow Discovery Protocol;

DoS – Denial of Service;

DDoS – Distributed Denial of Service;

Хост – кінцевий пристрій у мережі: сервер, персональний комп’ютер, тощо.

ВСТУП

Із збільшенням розмірів і складності мережі, збільшуються і витрати на адміністрування та керування. Існуючі мережі дуже неоднорідні і включають багато різних пристроїв, від маленьких датчиків і приладів до мережевих пристроїв, таких як маршрутизатори, сервери і периферійні пристрої. Крім того, ці пристрої використовують різні мережеві технологій, такі як дротові, бездротові, а також мобільні мережі. У такому складному та неоднорідному середовищі керування мережевими пристроями (такими як комутатори і маршрутизатори), переміщення користувачів і пристроїв, часті зміни в мережах (через відмову пристроїв і мережевих зв'язків), а також різке зростання кількості кібератак створює серйозні проблеми. Програмно визначені мережі (SDN) [1] пропонують багатообіцяючий підхід до вирішення деяких з проблем. SDN – технологія яка швидко розвивається і готова змінити комп'ютерні мережі так само як і хмарні обчислення змінили «обчислювальний» світ. Вона змінює структуру сучасних мереж, віддаляючись від поточних протоколів керування, що домінують в TCP/IP стеку Інтернету, до чогось більш гнучкого і програмованого. Це потенційно змінює спосіб створення мереж в майбутньому, шляхом використання пристроїв, які керуються зовнішнім програмним забезпеченням, на відміну від сучасного пропрієтарного мережевого обладнання, яке постачається з вбудованими виробником протоколами. SDN відкриває нові шляхи дослідження мережевих можливостей, які були надзвичайно складними до цього, тим самим допомагаючи зробити майбутні мережі більш керованими і практичними. Відділення рівня керування від рівня даних в SDN призводить до того, що мережеві комутатори стають простішими пристроями переадресації з більш витонченою логікою керування, реалізованою в програмному забезпеченні в логічно централізованому контролері. Це відділення в SDN дозволяє проектувати нові та інноваційні мережеві функції та протоколи. По-перше, модифікація конфігурації мережевих пристроїв, з допомогою програмного забазпечення, стає набагато простішою і з меншою ймовірністю

призводить до помилок, ніж через низькорівневі конфігурації пристроїв. По-друге, програма управління мережею може автоматично реагувати на шкідливі зміни в мережі і таким чином підтримувати актуальну політику високого рівня. По-третє, централізація логіки керування в контролері якому відомий стан мережі, допомогає спростити процес розробки складних мережевих функцій. SDN пропонує такі переваги для вирішення складності в сучасних мережах, однак, наразі критичною проблемою SDN є безпека[2]. Забезпечення захищенності мереж стає все складнішим завданням для підприємств, особливо з політикою власних пристроїв (BYOD), збільшенням використання хмарних технологій та Інтернету речей (IoT). Основні причини проблем безпеки, ймовірно, полягають у основних перевагах SDN, а саме: у програмованості мереж і централізації логіки управління. Ці можливості вводять нові загрози безпеці та атаки, які не існували в традиційних мережах. Закритий (пропрієтарний) характер мережевих комутаторів разом з великою кількістю різних постачальників програмного забезпечення та вбудовані функції керування, які раніше пропонувалися, були природними шарами захисту в традиційних мережах. Тобто, атаки на мережевий пристрій від конкретного постачальника доволі часто не працюють проти іншого мережевого пристрою від іншого постачальника. У певному сенсі різноманітність мережевих пристроїв і протоколів забезпечують певний рівень безпеки, як і секретність пов'язана з пропрієтарними системами.

З іншого боку, SDN із стандартизованими інтерфейсами та протоколами (наприклад, OpenFlow [1] – найпоширеніший протокол між контролером і комутаторами) може являти собою ціль для нападників; крім того, будь-які логічні помилки безпеки в реалізаціях протоколів SDN і програмного забезпечення можуть збільшити ризики. Також, централізація функціональності контролера SDN представляє іншу точку інтересу для потенційних зловмисників.

В цій роботі розглядаються принципи побудови мереж з архітектурою SDN, основні вектори атаки на SDN, відомі вразливості протоколу OpenFlow, а також надаються рекомендації щодо зменшення ризиків у середовищі SDN.

1 ОГЛЯД АРХІТЕКТУРИ SDN І ПРОТОКОЛУ OPENFLOW

1.1 Рівні архітектури

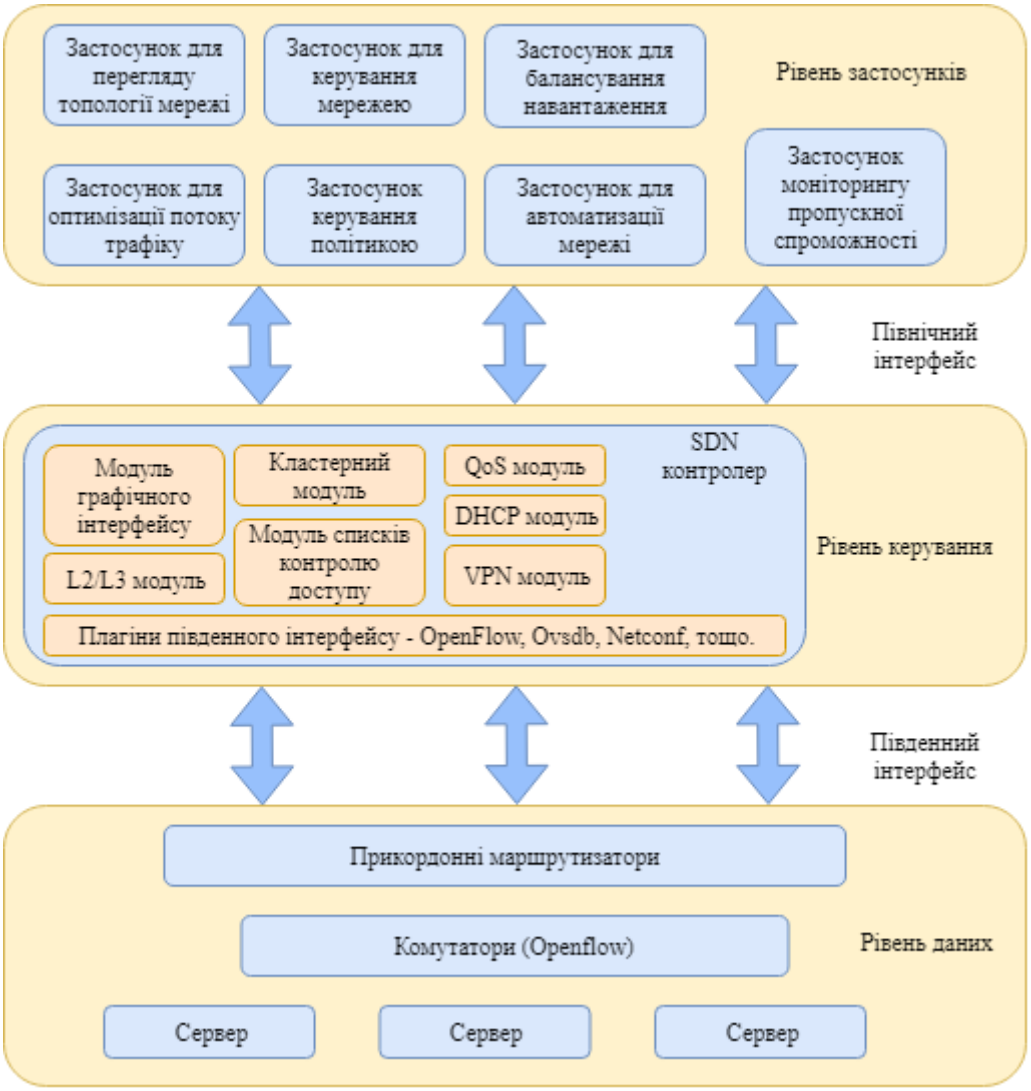

Рисунок 1.1 – Архітектура SDN

Відповідно до визначення Open Networking Foundation (ONF) SDN складається з трьох рівнів (рис. 1)[1]:

1. Рівень застосунків (англ. Applicaion Layer)
2. Рівень керування (англ. Control Layer)
3. Рівень даних (англ. Infrastructure layer)

Рівень даних складається з різноманітного мережевого обладнання, яке формує базову мережу для пересилання мережевого трафіку. Обов'язками рівня

даних в основному є пересиланням даних, а також моніторинг місцевої інформації та збір статистики.

Рівень керування відповідає за програмне управління рівнем даних. З цією метою він використовує надану інформацію по рівню даних і визначає роботу мережі і маршрутизацію. Він містить один або більше програмних контролера, які зв'язуються з елементами мережі пересилання через стандартизовані інтерфейси, які називаються південними інтерфейсами. Одним з найпоширеніших протоколів для південних інтерфейсів є OpenFlow.

Рівень застосунків містить мережеві програми, які можуть надавати нові функції для керування мережею, такі як налаштування схеми переадресації або допоможні фукнції для конфігурації мережі. Рівень застосунків може отримувати абстрактний і глобальний вигляд мережі від контролерів і використовувати цю інформацію для надання відповідних вказівок рівню керування. Інтерфейс між рівнем застосунків і рівнем керування називається північним інтерфейсом (англ. NBI). На сьогодні не існує стандартизованого API для північних інтерфейсів, тому, програмне забезпечення контролеру надає своїм власним API додатки.

1.2 Протоколи південного інтерфейсу

Найбільш поширеним протоколом південного інтерфейсу є OpenFlow, який стандартизований ONF. OpenFlow - це протокол, що описує взаємодію одного або декількох керуючих серверів з комутаторами, сумісними з OpenFlow. Контролер OpenFlow встановлює записи таблиці потоків у комутаторах після чого комутатори комутують трафік відповідно до цих записів. Таким чином, конфігурація комутаторів OpenFlow залежить від контролера. Потік класифікується за полями відповідності, подібними до списків керування доступом (ACL) і можуть містити маски.

Іншим варіантом протоколу південного інтерфейсу є Forwarding and Control Element Separation (ForCES) [4,5], який обговорюється і стандартизований робочою групою Internet Engineering (IETF) починаючи з 2004 року. ForCES, окрім протоколу, також являє собою фреймворк, який

відокремлює площину управління і площину даних і вважається більш гнучким і потужнішим, ніж OpenFlow [6,7]. Пристрої передачі трафіку моделюються з використанням логічних функціональних блоків (англ. Logical Funtional Blocks), які можуть бути об’єднані для формування складних механізмів пересилання. Кожен LFB надає заданий перелік функціональних можливостей, наприклад IP-маршрутизацію.

Архітектура SoftRouter [8] також визначає окремі функціональні можливості керування і рівня даних. В [8] автори представляють архітектуру SoftRouter і виділяють його переваги щодо Border Gateway Protocol (BGP).

ForCES і SoftRouter схожі на OpenFlow і можуть виконувати роль південного інтерфейс. IETF не зупиняє обговорення інших мережевих технологій, а також можливих південних інтерфейсів. Наприклад, Path Computation Element (PCE) [9] і Locator/ID Separation Protocol (LISP) [10] є кандидатами на південні інтерфейси.

1.3 Протоколи північного інтерфейсу

Протокол OpenFlow надає інтерфейс, який дозволяє програмному забезпеченню програмно налаштовувати комутатори в мережі. В основному, контролер може змінити таблиці потоків для перенаправлення трафіку. Контролери часто надають аналогічний інтерфейс застосункам, який називається північним інтерфейсом. Північний інтерфейс не є стандартизованим і дозволяє точно налаштовувати комутатори. Програмам не потрібно знати деталі південного інтерфейсу, так, застосунок не потребує повної інформації про топологію мережі, тощо.

Для того щоб застосунок міг налаштовувати комутатори, йому потрібно повідомити контролер про бажаний шлях передачі трафіку, але у цей же час контролеру необхідно створити відповідні команди для модифікації таблиць потоків комутаторів. Тому, для полегшення і автоматизації налаштування мережі, використовуються мережеві мови програмування.

Вимоги мови для SDN обговорюються в [11]. Автори зосереджуються на трьох важливих аспектах:

1. мова мережевого програмування повинна забезпечувати засоби для запиту стану мережі. У середовищі виконання мови збирається стан мережі і статистика, яка потім надається до застосунку;
2. мова повинна вміти виражати мережеву політику, що визначає метод передачі трафіку. Повинна бути можливість комбінувати політики різних мережевих застосунків. Мережеві застосунки можуть створювати конфліктні політики мережі, тому, мережева мова програмування повинна бути достатньо потужною, щоб визначити та вирішити такі конфлікти;
3. переналаштування мережі - складне завдання, особливо з різними мережевими політиками. Середовище виконання повинно викликати процес оновлення пристроїв, щоб гарантувати збереження контролю доступу, уникнути петель, тощо.

Найпопулярнішими мовами програмування SDN, що відповідають представленим вимогам, є Frenetic [12], його наступник - Pyretic [13], і Procera [14]. Ці мови забезпечують декларативний синтаксис і базуються на на функціональному реактивному програмуванні. Через характер функціонального реактивного програмування, ці мови надають композитний інтерфейс для мережевих політик. Існують і інші варіанти. The European FP7 науково-дослідний проект, NetIDE, може використовуватись як північний інтерфейс SDN [15].

1.4 Протокол OpenFlow

1.4.1 Загальний опис

Специфікація визначає OpenFlow як протокол для зв'язку з мережевими пристроями і керуванням роботою рівня даних мережі. Рівень даних керується шляхом надання мережевим пристроям(комутаторам) інструкцій, які називаються потоками.

Протокол OpenFlow є найбільш часто використовуваним протоколом для південного інтерфейсу SDN, який відокремлює рівень даних від рівня керування. [16] висвітлює переваги гнучко налаштовуваного рівня даних. OpenFlow був спочатку запропонований Університетом Стенфорду, і зараз він стандартизований ONF [3].

Архітектура OpenFlow складається з трьох основних понять:

1. мережа будується з OpenFlow-сумісних комутатори, які складають рівень даних;
2. рівень керування складається з одного або більше контролерів OpenFlow;
3. безпечний канал керування з'єднує комутатори з рівнем керування.

OpenFlow-сумісний комутатор є базовим пристроєм, що пересилає пакети відповідно до його таблиці потоків. Записи таблиці потоків складаються з полей заголовків, лічильників та інструкцій. Тобто, кожен потік визначає умови для відповідності вхідних пакетів і інструкції які слід застосовувати до відповідних пакетів. Із розвитком специфікації, доступні операції та критерії відповідності стали більш складними (наприклад, контроль за якістю обслуговування з допомогою черг та відповідність вхідних пакетів до кількох таблиць потоків), але основна ідея відповідності правилу і виконання певної дії залишилося незмінним.

Поля заголовків у записі таблиці описують, до яких пакетів цей запис застосовується. Поля заголовків можуть співпадати з різними протоколами в залежності від специфікації OpenFlow, наприклад, Ethernet, IPv4, IPv6 або MPLS.

Лічильники зарезервовані для збору статистики про передачу. Вони зберігають кількість прийнятих пакетів і байтів, а також тривалість передачі.

Інструкції вказують, як обробляються пакети. Загальні дії - "forward", "drop", "modify field", тощо.

Контролер відповідає за заповнення та маніпулювання таблицями коммутації комутаторів. За допомогою вставки, модифікації та видалення записів потоку контролер може змінювати поведінку комутаторів щодо

пересилання. Специфікація OpenFlow визначає протокол, який дозволяє контролеру надавати інструкції комутаторам. Для цього контролер використовує безпечний канал керування.

1.4.2 Зв'язок між контролером та комутаторами

Зв'язок між контролером і комутаторами відбувається по TCP-з'єднанню, яке може бути додатково захищене з допомогою TLS з взаємно аутентифікованими сертифікатами, підписаними приватним ключем (наприклад, кореневий сертифікат). В офіційних списках розсилки відбуваються дискусії [19], щодо включення до стандарту протоколу автоматичного виявлення контролеру, але остання версія (v1.5.1) досі вказує тільки ручну конфігурацію [20].

У протоколі можна виділити три типи комунікації: контролер-комутатор, асинхронна комунікація і симетрична комунікація. Зв'язок контролер-комутатор відповідає за виявлення функцій, конфігурацію, програмування комутатора та пошуку інформації. Асинхронна комунікація ініціюється сумісним з OpenFlow комутатором без жодного підтвердження від контролера. Данний тип комунікації використовується для повідомлення контролера про надходження пакетів, зміни стану на комутаторі і помилки. Нарешті, симетрична комунікація ініціюється однією з сторін, тобто перемикач або контролер вільні ініціювати спілкування без підтвердження з іншого боку. Прикладами симетричної зв'язку є повідомлення hello або echo, які можуть бути використані для визначення того, чи є канал управління ще доступним.

1.4.3 Механізм прийняття рішень

Базовий механізм прийняття рішень щодо пакету ілюструється на рисунку 2.

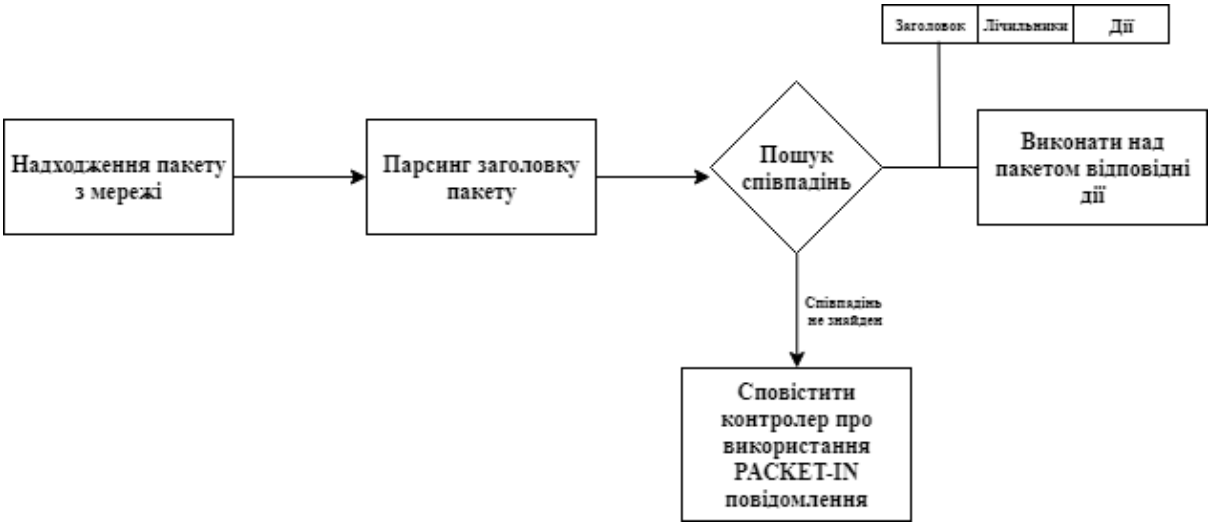

Рисунок 1.2 Механізм прийняття рішень OpenFlow комутатора

Коли комутатор приймає пакет, він аналізує заголовок пакету для пошуку відповідного запису у таблиці потоків. Якщо в таблиці знайдено запис, у якому поле-заголовок відповідає заголовку пакета, запис враховується. Якщо знайдено кілька таких записів, пакети узгоджуються на основі пріоритетів, тобто, обирається запис з найвищим пріоритетом. Потім, комутатор оновлює лічильники у записі. Нарешті, комутатор виконує дії над пакетом, вказані записом таблиці, наприклад пересилає пакет на визначений порт. В іншому випадку, якщо не існує записів у таблиці що відповідають заголовку пакета, комутатор повідомляє контролеру про пакет, який буферизується, якщо комутатор здатний до буферизації. Він інкапсулює або небуферизований пакет, або перші байти буферизованого пакету з використанням повідомлення PACKET-IN і передає його контролеру; зазвичай інкапсулюється заголовок пакета і 128 байтів за замовчуванням. Контролер, який отримує повідомлення PACKET-IN визначає правильну дію для пакета і встановлює один або більше відповідних записів таблиці потоків у запитуючому комутаторі. Буферизовані пакети потім пересилаються відповідно до правил.

1.4.3 Схеми задання правил та побудови мережі

У мережах OpenFlow існують два підходи до створення правил: активний — правила встановлюються на комутаторах, перш ніж вони знадобляться; і

реактивний — правила вставлюються на комутаторах у відповідь на пакети, які контролер отримує через повідомлення PACKET-IN.

При суворому дотриманні специфікації, правила потоку для комутатора можуть бути встановлені тільки контролером OpenFlow через з'єднання TCP, що було ініційоване комутатором. Однак деякі комутатори підтримують додатковий "Режим слухача", в якому вони приймають з'єднання до спеціально налаштованих TCP портів від будь-якого хоста у мережі. Такі зовнішні з’єднання також можуть бути використані для встановлення правил потоків на комутаторі та зчитування інформації з лічильників. Даний режим роботи дозволяє легко налагоджувати й перевіряти стан правил без додаткового навантаження або ускладнення контролерів.

Існує два загальні методи обробки трафіку OpenFlow між комутаторами і контролером: in-band та out-of-band(рис. 1.3).

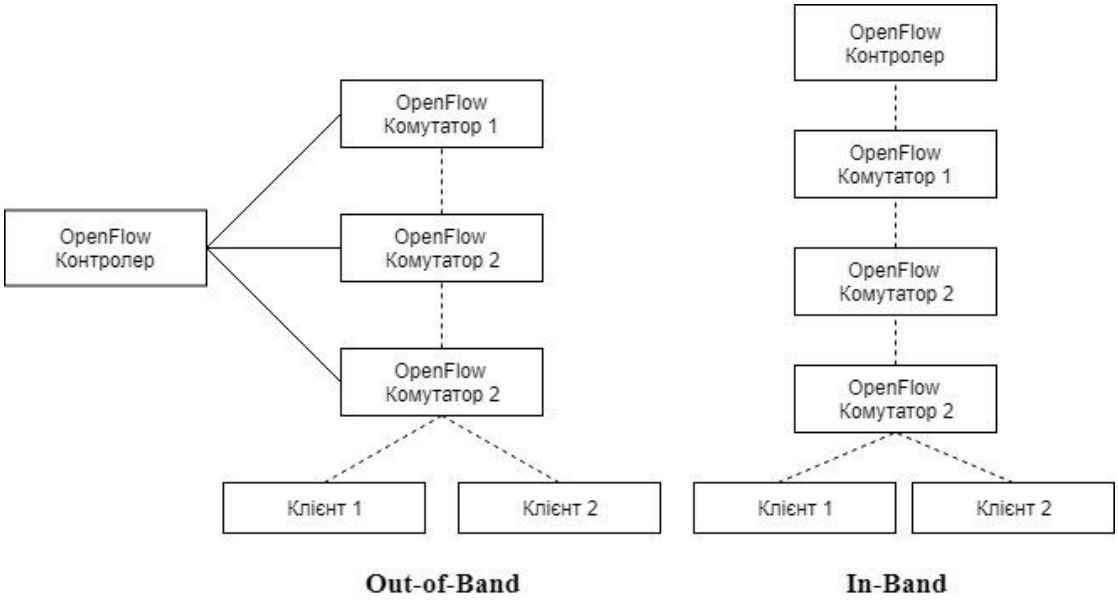

Рисунок 1.3 Методи обробки трафіку між комутаторами та контролером(пунктир означає канали OpenFlow)

У випадку використання схеми out-of-band, комутатори мають мережеві шляхи для зв'язку з контролером на які робота протоколу OpenFlow не впливає. Однак, це вимагає додаткового налаштування VLAN або окремих фізичних інтерфейсів, до яких не застосовуються правила OpenFlow. А у випадку

використання схеми in-band, комутатори використовують мережу OpenFlow для передачі даних і для зв'язку з контролером.

1.4.4 Протокол визначення топології OpenFlow

Одна з найбільш фундаментальних функцій яку OpenFlow контролер повинен виконувати – це надання точного вигляду мережі у майже реальному часі. Ця функція відома як Topology Discovery(англ. виявлення топології). У мережах SDN для визначення топології, усі OpenFlow контролери використовують один і той же протокол – OFDP (англ. OpenFlow Discovery Protocol).

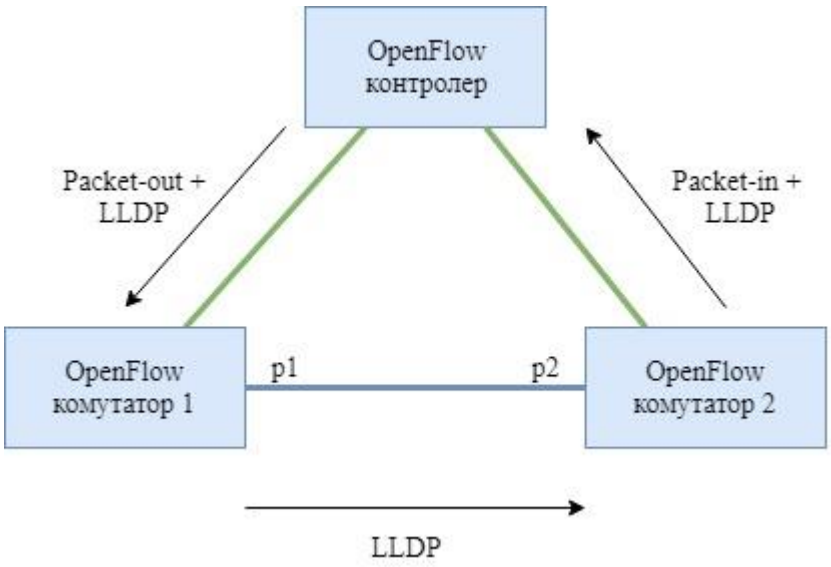

Рисунок 1.4 – Принцип роботи протоколу OFDP

Рисунок 1.4 показує як працює OFDP. Для того щоб виявити однонаправлений канал «Комутатор 1 -> Комутатор 2» контролер інкапсулює LLDP пакет у повідомлення «Packet-out» і відправляє його до Комутатора 1. Цей пакет містить інструкції для Комутатора 1 відправили LLDP пакет через інтерфейс p1. Отримавши LLDP повідомлення на інтерфейсі p2 Комутатор 2 інкапсулює його у повідолення «Packet-in» і відправляє його до контролера. Коли контролер отримає LLDP пакет, він може зробити висновок, що між Комутатором 1 та Комутатором 2 існує однонаправлений канал. Аналогічну

процедуру OpenFlow контролер виконує для каналу Комутатор 2 -> Комутатор 1 та для всіх інших каналів.

Для підтримки актуальної бази даних про канали, комутатор періодично оновлює її, повторюючи процедуру описану на Рисунку 1.4.

Висновки до розділу 1

У даному розділі був проведений оглядовий аналіз науково-технічної літератури, но основі якого були наведені основні принципи архітектури SDN, принципи роботи складових частих мережі, таких як контролер та комутатори, підходи до побудови мереж і задання правил. Також був описаний механізм роботи протоколу OpenFlow та його основних частин.

На основі даного огляду літератури можна сказати, що захищеність SDN мереж та протоколу OpenFlow потребує більш детального дослідження та аналізу.

2 БЕЗПЕКА SDN МЕРЕЖ ТА ПРОТОКОЛУ OPENFLOW

2.1 Переваги архітектури SDN з точки зору безпеки

Серед переваг SDN у порівнянні з традиційними мережами можна виділити декілька факторів які позитивно впливають на захищенність мережі[17]. У таблиці 2.1 наведені основні з цих параметрів.

Таблиця 2.1 – позитивні характеристики архітектури SDN

Характеристика	Наслідок з	Застосування
Повний вигляд мережі	• Централізація • Збір статистики трафіку	• Система виявлення вторгнень по всій мережі • Виявлення зловмисної роботи комутатору • Розслідування інцидентів у мережі
Механізм самовідновлення	• Виконання правил за умов • Збір статистики трафіку	• Реактивне відкидання пакетів • Реактивне пересилання пакетів
Контроль стану мережі	• Схема побудована на потоках	• Контроль доступу

Далі кожна з наведених харектиристик буде описана більш детально.

1. Той факт, що контролер в SDN архітектурі у будь-який час має доступ до глобального перегляду мережі є, можливо, найбільшою перевагою безпеки SDN на безпекою традиційних мереж. Це стало можливим завдяки централізації керування мережею, а також завдяки тому, що кожен елемент мережі збирає та відсилає контролеру статистику використання мережі. На відміну від SDN

традиційі мережі, потребують від елементів мережі обміну великими об’ємами інформації, а також часу на сходження для того щоб частково визначити стани лише певної частини мережі. Окрім цього зазвичай тільки певні пристрої у традиційних мережах ведуть журнали подій та використання трафіку.

Здатність у будь-який момент часу отримати повний вигляд мережі надає наступні можливості з посилення захищеності мережі:

1) Виявлення вторгнень у всій мережі. Маючи повний вигляд мережі SDN контролер може мати у собі модуль Системи виявлення вторгнень(англ. Intrusion Detection System) яка працює у масштабах повної мережі. Дана система аналізує статистику трафіку яку надаються елементи мережі, а також журнали подій для виявлення зловмисного трафіку. У традиційних мережах, у свою чергу, система IDS зазвичай встановлюється у певному елементі мережі, і, відповідно, захищає тільки цей елемент мережі, не маючи інформаціїї про стан усієї іншої частини мережі. Після отримання статистики від SDN комутаторів, IDS може працювати за двома різними принцпами:
    i) Виявлення шкідливої активності: IDS створює сигнатури для відомих атак. Стан мережі і поведінка елементів мережі постійно переглядається, а вторгнення рапортується у випадку якщо поточний стан мережі підпадає під відому сигнатуру.
    ii) Виявлення аномалій: у момент часу коли наперед відомо що, ніякої зловмисної активності у мережі не відбувається, трафік перехоплюється і будується його профіль, в основному на загальних характеристиках пакетів від довірених елементів мережі, на яких працюють довірені застосунки. Вторгнення рапортується коли поведінка елементів мережі значно відрізняється від профілю.

    Обидва названих механізма мають свої переваги та недоліки. Так, «виявлення аномалій» має здатність до виявлення нових невідомих атак і потребує меншого знання про зловмисну поведінку. Однак, даний метод рапортує більше помилкових повідомлень ніж «виявлення шкідливої активності», так як трафік, що відрізняється від звичного трафіку у мережі не обов’язково є

зловмисним. Розробники не зупиняються у пошуку можливостей з покращення IDS фреймворків для SDN і зараз зосередженні у зменшенні навантаження на процесор, збільшенні точності виявлення і можливості фреймворків автоматично вивчати нову довірену або зловмисну активність.

2) Виявлення зловмисної активності комутаторів. Доступ до повного вигляду мережі не тільки надає змогу більш ефективно виявляти втручання за зловмисним трафіком, але також допомагає у виявленні шкідливої поведінки мережевих комутаторів. Наприклад, випадок, коли мережевий пристрій відкидає деякі або всі вхідні пакети, створюючи чорну диру(англ. Black hole). Визначення відповідального за таку поведінку пристрою маршрутизації – доволі складна задача у традиційних мережах. Техніка опитування, яка зазвичай використовується у таких мережах, наприклад Traceroute, може бути не ефективною, так як зловмисний комутатор може приховувати свою активність, відкидаючи тільки певний тип трафіку. У мережах SDN, дана задача значно полегшується, так як усі комутатори періодично звітують контролеру кількість отриманих, пересланих та відкинутих пакетів. Аналізуючи ці звіти можна якщо, не виявити зловмисний комутатор то хоча б значно звузити круг «підозрюваних». Це можливо, навіть якщо зловмисний комутатор у свому звіті підмінює значення пересланих і відкинутих пакетів, так як сусідні комутатори будуть вказувати справжню інформацію про отримані пакети. Однак, відрізнити відкинуті пакети внаслідок помилок каналу і внаслідок зловмисної діяльності все ще залишається складною задачею.
3) Розлідування інцидентів у мережі. Факт, що на рівні котролю періодично зберігаються журнали глобального стану мережі полегшує проведення розлідувань. Є можливість переглянути трафік що проходив по мережі у минулому, щоб зрозуміти, як не виявлені атаки виконувалися, що дуже корисно для покращення захисних механізмів проти майбутніх випадків таких атак. Окрім цього, це допомагає ідентифікувати та ізолювати

скомпрометовані хости, а також відслідити особу/організацію, що стоїть за атаками.

2. Ще одна характеристика, що відрізняє SDN мережу від традиційної мережі є те, що вона забезпечена механізмами самовідновлення. Умовні правила є прикладом таких механізмів які були введені в парадигму SDN в [18]. Ці правила встановлюються на комутаторі рівнем керування і активуються після виконання певної умови. Умови зазвичай пов'язані з зібраної статистикою комутатора, наприклад, з кількістю пакети, що належать певному потоку, отриманому протягом певного періоду, перевищує попередньо визначений поріг. Активоване правило вказує, як комутатор повинен діяти у випадку, коли вказана подія виконана. Таке реагування на події забезпечує автоматичну стійкість проти нападів. Наприклад, дія яку необхідно виконати комутатору зазначена за правилом може бути відкидання пакетів, визначених правилом, або пересилання цих пакетів через інші шляхи для зменшення навантаження на певні частини мережі. Це забезпечує стійкість проти атаки Denial of Service (DoS), орієнтованої на мережеві хости або мережеві канали. Також існує підхід коли дією є зміна адреси призначення певних пакетів так, щоб вони доставлялися на «пастку»(англ. honeypot), яка являє собою ізольований і контрольований хост, який використовується в якості пастки для збору додаткової інформації про шкідливі дії. Це перенаправлення виконується прихованим способом, щоб джерело даних шкідливих пакетів не помітило зміни. Даний підхід показує себе дуже вдало у задачах виявлення ботнетів, які являють собою набір пов’язаних між собою хостів, контроль над якими має зловмисник і використовує їх як платформу для запуску розподілених атак проти інших частин мережі.

3. Мережі SDN мають більше можливостей керування у порівнянні з традиційними мережами. Це пов'язано з тим що SDN мережі побудовані навколо механізму прийняття рішень який покладається не тільки на адресу призначення, як у традиційних мережах. Рішення щодо перенаправлення пакетів базується на декількох атрибутах з заголовку пакетів на комутатор.

Таким чином, контролер SDN має більший контроль над трафіком у мережі, вказуючи, які типи пакетів слід переносити в межах мережі на основі типу корисного навантаження, адреси джерела або будь-якого іншого значення поля заголовка. Правила, встановлені контролером, можуть, наприклад, дозволяти проходити через мережу лише пакетам TCP, що походять з певного хоста. Це допомагає у обмеженні шкідливого трафіку від надходження до або від від будь-якого комутатора мережі SDN. У традиційних мережах, де мережеві пристрої передають пакети сліпо, залишаючи рішення про доступ до самого кінця, де зазвичай розташований брендмауер, який повинен сканувати усі пакети що надходять для перевірки корисного навантаження, що зазвичай спричиняє затримки у передачі трафіку.

2.2 Вразливості архітектури SDN

2.2.1 Рівень даних

DoS спрямована на комутатор: враховуючи, що сьогоднішні комутатори мають обмежений об’єм сховища, а також, що правила, які надходять від контролера, повинні охоплювати усі можливі типи трафіку, стає очевидним, що неможливо зберегти всі ці правила в комутаторі. Замість цього застосовується механізм реактивного кешування у поточних реалізаціях SDN, де кожен раз коли комутатор не знаходить відповідного правила для певного типу трафіка вхідних пакети, пакет зберігається тимчасово у буфері комутатора, і надсилається PACKET-IN повідомлення до контролера з інформацією про отриманий пакет і запитом на відповідне правило. Після того, як відповідне правило отримано комутатором, пакет обробляється на основі цього правило, яке потім кешується в таблиці потоків комутатора для того щоб наступні пакети цього потоку оброблялись безпосередньо.

Використання даного реактивного механізму кешування призводить до того, що комутатори є вразливими до атаки DoS, коли зловмисник перевантажує комутатор пакетами з великим корисним навантаження, що належать до різних потоків. Правила деяких з цих потоків можуть бути відсутніми в локальній таблиці потоків комутатора, що вимагає відправлення PACKET-IN повідомлень

до площини керування, тобто, до контролера. У результаті комутатор зберігає деякі з цих великих пакетів у буфері, чекаючи відповідей на PACKET-IN повідомлення від контролера. Даний буфер може бути швидко заповнений, особливо, якщо отримані пакети містять велике корисне навантаження, що призведе до того, що звичайні пакети, що належать до нових потоків будуть відкинуті, так як відсутнє місце у буфері для їх тимчасового зберігання.

Було запропоновано багато варіантів протидії даній атаці. Одни з них є проактивне кешування, коли комутатор не чекає на надходження певного пакету для того щоб запросити відповідне правило у контролера, а зберігає у кеші якнайбільшу кількість правил для різних потоків. Дана техніка стає дуже ефективною у поєднанні з використанням комбінованих правил потоків, коли правила та дії описуються для декількох можливих полів заголовків, а не по одному правилу на один можлививй заголовок, що призводить до зменшення необхідної кількості правил. Також, очевидно, що використання більш широких каналів між комутаторами та контролером призводить до зменшення затримки для передачі буферизованих повідомлень, що ускладнює реалізацію даної атаки.

Шифрування пакетів та обхід правил доступу: використання схеми пересилання пакетів на основі потоків дозволяє мережі SDN визначати як пакети з різним корисним навантаженням повинні оброблятися комутаторами. Цей механізм може бути використаний для контролю доступу, тобто, контролер може вказувати, яким типам пакетів дозволяється проходити через мережу. Пакети з корисними навантаженнями, що не підпадає під визначені правила повинні бути відкинуті або відправлені контролеру для подальшої перевірки.

Хоча схема, що базується на потоках, надає нові можливості керування мережею, не зрозуміло, яким чином ця схема має обробляти зашифровані пакети, у яких не всі поля заголовків пакетів видимі мережевим комутаторам. Насправді, пакети (і їх заголовки, і корисне навантаження) можуть бути приховані повністю від мережевих комутаторів за допомогою зашифрованих тунелів, в яких необхідні пакети інкапсулюються в інших пакетах. Після того, як ці інкапсульовані пакети будуть отримані на іншому кінці тунелю, внутрішній

пакет розшифровується, декапсулюється і маршрутизується на основі політики нової мережі, якій належить цей кінець тунелю. Використання зашифрованих пакетів і тунельних з'єднаннь дозволяє зловмисникам «перетинати» межі мережі у якій вони знаходяться і обходити політики контролю доступу.

Проти даного типу атак, дуже ефективно себе показують системи роспізнавання зашифрованого трафіку по розміру пакетів, затримках між пакетами та іншим характеристиками, що допомагають у роспізнаванні трафіку.

2.2.2 Рівень керування

Атака DDoS: Рівень керування вразливий до DDoS атак, коли декілька скомпрометованих хостів, що розподілені в мережі, починають синхронно генерувати велику кількість пакетів і надсилати їх через комутатори. Оскільки не всі сгенеровані пакети будуть підпадати під правила, що знаходяться у локальних таблицях потоків комутаторів, буде створюватись велика кількість запитів PACKET-IN які надсилатимуться до контролера, що призведе до перевантаження обчислювальних можливостей контролера, у результаті чого обробка звичайних, легітимних запитів буде відкладена або, навіть, відхилена.

Одним з рішень є використання реплікації контролеру – випадок коли декілька контролерів керують однією мережею. Однак, у цьому випадку, рівень даних повинен продовжувати роботу ніби при одному контролері. Для цього використовується логічна централізація, яка вимагає наявності захищених каналів для підтримки постійних правил потоків, а також для налаштування розподілення навантаження. Коли реплікація налаштована, комутатори можуть бути під’єднані до декількох контролерів, серед яких для кожного комутатора обирається керуючий контролер. Керуючий контролер для комутатора обирається на основі кількості запитів сгенерованих комутатором, а також на основі затримки між контролером та комутатором. Внаслідок цього, мережа стає стійкою до DDoS атак спрямованих на контролер. Однак, це значно ускладнює задачу проектування мережі, так як розташування контролерів напряму впливає на стійкість мережі та на затримки у мережі.

Компрометований контролер: Один з найгірших сценаріїв це той, коли зловмисник якимось чином (наприклад, шляхом експлуатації ПЗ комп'ютеру на якому запущений контролер), отримує доступ до контролера, отримуючи також повний контроль над комутаторами, що керуються даним контролером. Компрометований контролер може відправити цим комутаторам команду відкидати увесь вхідний трафік, що призведе до повної недієздатності мережі або використати їх як платформу для запуску сконцентрованої атаки на інші цілі, перенаправляючи усі вхідні пакети обраній жертві, щоб вичерпати її ресурси.

Одним з очевидних рішень є використання реплікації контролерів. Однак, якщо у мережі усі контролери побудовані на одній й тій самій платформі, то, відповідно, у всіх контролерів одні й ті самі вразливості, що дає змогу зловмисник скомпрометувати усі контролери як тільки він зможе скомпрометувати один з них. Тому проти атак компрометування контролеру необхідно будувати мережі у яких використовуються контролеру побудовані на різних платформах.

Зловмисний контролер: Інший можливий сценарій це створення зловмисником контролеру і підміна легітимного контролеру зловмисним. Це досягається шляхом експлуатації вразливостей протоколу керування комутаторами(наприклад, Openflow).

Проти даного типу атак у [21] рекомендується використовувати систему рейтингу контролерів у мережі.

Атака повторного відтворення: Якщо у мережі використовуються захищені канали зв'язку мід контролером та комутаторами, зловмисник може перехоплювати зашифровані пакети керування, зберігати їх, а у майбутньому повторно використовувати їх для того щоб повернути комутатори у попередні стани.

Для захисту від даного типу атак необхідно використовувати часові мітки у зашифрованих пакетах керування, що надсилаються від контролера до комутаторів.

2.2.3 Рівень застосунків

У мережах SDN застосовується велика кількість застосунків для різних цілей. Більшість з них – це проекти з відкритим кодом, який розміщається у відкритих репозиторіях(наприклад, github репозиторії). Це дає зловмиснику можливість досконально вивчити принципи роботи цих застосунків, знайти можливі вразливості у їх коді або, навіть, завантажити зловмисний код у зовнішній репозиторій з цілью використати це у майбутньому.

2.3 Вразливості протоколу OpenFlow

2.3.1 Недостатня підтримка TLS

Оригінальна специфікація OpenFlow (v1.0) вимагала обав’язково захищати канал управління між контролерами і комутаторами за допомогою TLS [22]. Однак наступні специфікації видалили вимогу і зробили її необов’язковою функцією. Впровадження TLS має вищий «технічний бар'єр» для операторів зв'язку через більш складні дії необхідні для його правильного налаштування, які включають наступне: генерація site-сертифікату, генерація сертифікатів контролерів, генерація сертифікатів комутаторів, підпис сертифікатів закритим site-ключем та встановлення правильних ключів та сертифікатів на всі пристрої. Порівнюючи з цим, налаштування комунікації у вигляді відкритого тексту вимагає лише одного аргументу - адреси контролера. Цей факт може схилити мережевих адміністраторів повністю пропустити налаштування TLS, чим потім можуть скористатися зловмсиники.

Через швидкий розвиток принципів OpenFlow, багато виробників як комутаторів, так і контролерів могли не повністю дотримуватись специфікації і пропустили впровадження підтримки TLS. Таблиця 2.2 і 2.3 показують які виробники багатьох популярних комутаторів і контролерів OpenFlow впровадили підтримку TLS.

Таблиця 2.2 – Підтримка TLS комутаторами

Виробник комутаторів	Підтримка TLS
HP	Ні[23]
Brocade	Ні
Dell	Ні
NEC	Частково(тільки на певній моделі)
Indigo	Ні
Pica8	Ні
OpenWRT	Так[24]
Open vSwitch	Так[25]

Таблиця 2.3 – Підтримка TLS контролерами

OpenFlow Контролер	Підтримка TLS
NOX	Тільки контролер(комутатори не автентифікуються)
POX	Так[25]
Beacon	Ні[26]
Floodlight	Ні[35]
MuL	Ні[27]
FlowVisor	Так[28]
Big Network Controller	Ні
Open vSwitch Controller	Так[25]

Існує помітна відсутність підтримки з боку обох сторін, що, таким чином, звільняє обидві сторони від значних зусиль, спрямованих на його реалізацію.

Наприклад, обговорюючи підтримку TLS, один з розробників контролера Floodlight написав: «Це було б досить тривіально додати його, якщо існує достатній інтерес», таким чином, відсутність попиту на цю функцію пояснюється обмеженою підтримкою TLS комутаторами[29].

Відсутність підтримки TLS і відсутність мотивації впровадити дану функцію створює зловмисникам широкий простір для проникнення в OpenFlow мережі і надає змогу залишатись, в значній мірі, непоміченими після таких дій. Цей факт може не відноситись до деяких фізично захищених мереж, таких як центри обробки даних, де неавторизованим особам неймовірно важко отримати доступ до комутаторів. Однак для, наприклад, кампусів або віддалених офісів, де комутатори розміщуються у менш захищених та контрольованих місцях (наприклад, шафи, зовнішні розподільні коробки, тощо) це створює більш значні і ймовірні загрози.

Як тільки зловмисник зможе розмістити пристрій між контролером і комутатором, для перехоплення трафіку OpenFlow, він отримає можливість встановити додаткові правила на комутаторі для перехоплення або модифікації конфіденційного трафіку, отримати доступ до захищенної мережі, перешкоджати роботі іншим пристроями на рівні доступу і навіть керувати всіма комутаторами. Хоча цей вид атак і можливий в традиційних мережах, але для зловмисника налаштувати пристрої і правильно підробити відповіді до програмного забезпечення керування мережею – дуже складна задача, через змінну природу SNMP OID та різні формати повідомлень керування у різних постачальників. OpenFlow значно зменшує цей технічний бар'єр шляхом стандартизації збору статистики та керування потоками між усіма постачальниками.

Ризики, викликані успішною атакою man-in-the-middle у in-band схемі мережі OpenFlow, набагато більші за такі ж ризики у звичайній, традиційній мережі, через здатність зловмисника одразу ж переналаштувати всі

підконтрольні комутатори. У традиційних мережах зловмисник повинен чекати, поки оператор не авторизується в систему керування кожного окремого комутатора з використанням небезпечного протоколу (наприклад, Telnet або SNMPv2) для успішного перехоплення даних для автентифікації. Однак через постійний зв'язок і відсутність аутентифікації в OpenFlow TCP каналі, зловмисник може негайно скористатися повним контролем над будь-яким комутатором і налаштувати перехоплення трафіку, факт якого дуже важко виявити.

Наприклад, на рис.2.1 зловмисник може звичайним чином передавати трафік OpenFlow між контролером і сусіднім комутатором, але також встановити додаткове правило на комутаторі 3 для дублювання будь-якого трафіку, який несе інформацію про бази даних між двома клієнтами і відправляти його на віддалений зловмисний хост.

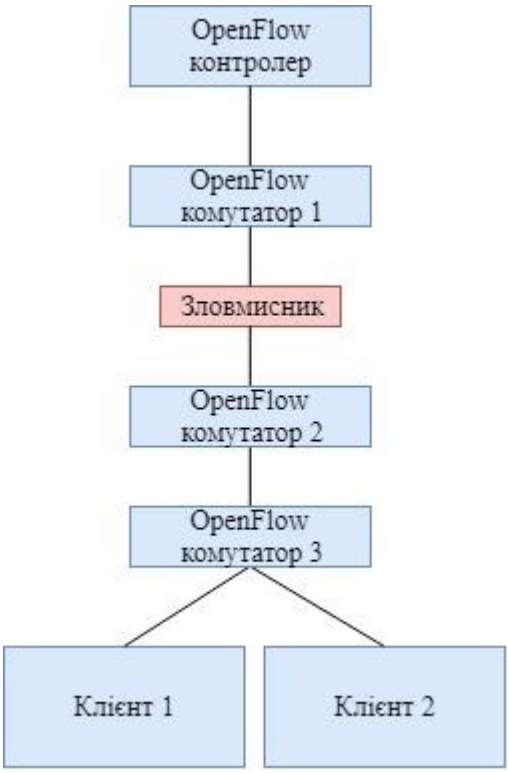

Рисунок 2.1 – MitM атака у схемі in-band

У традиційній мережі, дублювання специфічного трафіку між двома клієнтами, що з’єднані з комутатором і пересилання цього трафіку до віддаленого хоста, вимагало б складну процедуру переналаштування кожного комутатора після очікування, щоб захопити дані для автентифікації і можливо навіть потребувало б функцій, які багато комутаторів не підтримують (наприклад, інкапсуляцію GRE, або перевірка TCP заголовків).

Хоча цей тип атаки технічно досягався і раніше, OpenFlow різко знизив рівень складності до точки, де її виконання може бути легко автоматизоване і упаковане у вірус. OpenFlow усунув і проблему неоднорідності стандартів конфігурацій мережевого обладнання, і потребу в очікуванні для перехоплення даних для автентифікації.

Ризик атаки даного виду значно збільшується, якщо комутатор залишився з увімкненим «Режимом слухача». Це усуває вимогу для зловмисника проводити початкову MitM атаку. Просто знайшовши комутатор у режимі прослуховування, шляхом мережевого сканування(наприклад, використавши програму nmap[30]), зловмисник може зібрати статистику по всім потокам для додаткової розвідки, а після цього встановити правила для перехоплення трафіку, що проходить через комутатор і використовувати його для атак у майбутньому.

2.3.2 Вразливість протоколу OFDP

У усіх найбільш використовуваних OpenFlow контролерах (OpenDaylight, Floodlight, NOX, POX, Beacon, Ryu та Cisco Open SDN Controller) OFDP використовує незашифровані повідомлення LLDP для виявлення каналів між комутаторами, що робить його вразливим до наступних атак:

- **Switch spoofing**. Як показано на рисунку 2.2, кожен LLDP пакет містить поля version, flags, TTL і TLVs(Type-length-value) для розголошення інформації. Обов’язковими TLV у OFDP є *ChassisSubtype* та

PortSubtype. *ChssisSubtype* вказує ідентифікатор комутатора якому відсилається пакет, а *PortSubtype* – інтерфейс комутатора з котрого відсилається пакет.

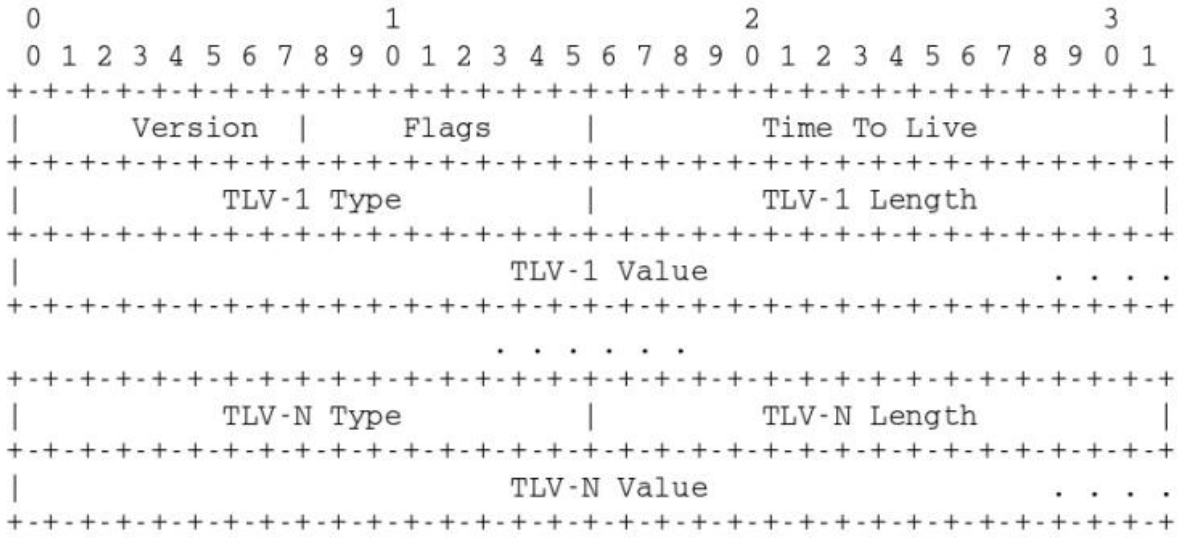

Рисунок 2.2 – Формат пакету LLDP

Проблема в тому, що всі вище названі контролери встановлюють значення *ChassisSubtype* MAC-адресу локального інтерфейсу комутатора, що полегшує противнику задачу підмінити цей комутатор, оскільки контролери використовують MAC-адресу як унікальний ідентифікатор комутатора. Перехоплюючи незашифровані LLDP пакети, що містять MAC-адреси, зловмисний комутатор може підміняти інші комутатори для фальсифікації топології мережі як її бачить контролер. У прикладі, показаному на рисунку 2.3, Комутатор 4 перехоплює пакети LLDP з Комутатора 1, що містять MAC-адресу локального інтерфейсу Комутатора 1. Після цього Комутатор 4 може використовувати його як свою власну MAC-адресу і перепідключившись до контролера як Комутатор 1 переплутати граф топології контролера (наприклад, контролер «бачить» неіснуючі канали Комутатор 1 → Комутатор 3 і Комутатор 3 → Комутатор 1).

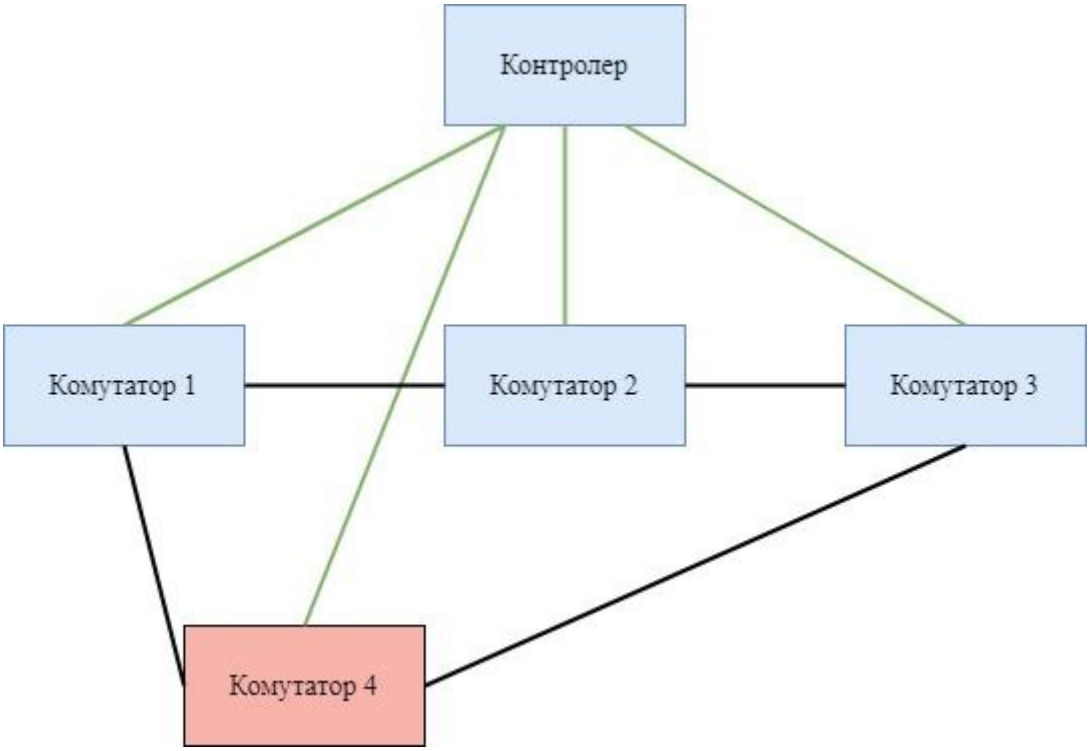

Рисунок 2.3 – Атака Switch Spoofing

- **Link fabrication.** На рисунку 2.4, зловмисник отримав контроль над двома комп’ютерами К1 та К2, які під’єднані до Комутатора 1 та Комутатора 3 відповідно. К1 відсилає LLDP пакети, отримані від Комутатор 1, до К2, а К2 дублює їх і відсилає до Комутатора 3. Отримавши LLDP пакет від Комутатора 3, контролер створює канал між Комуатором 1 та Комутатором 3. Фальшивий канал змушує контролер робити невірні рішення стосовно маршрутизації пакетів. Якщо зловмисник має контроль тільки над К1, але йому відомий Datapath ID(DPID) Комутатора 3 він все одно може зфабрикувати однонаправлений канал Комутатор 3 -> Комутатор 1, шляхом відправки підроблених пакетів LLDP до Комутатора 1.

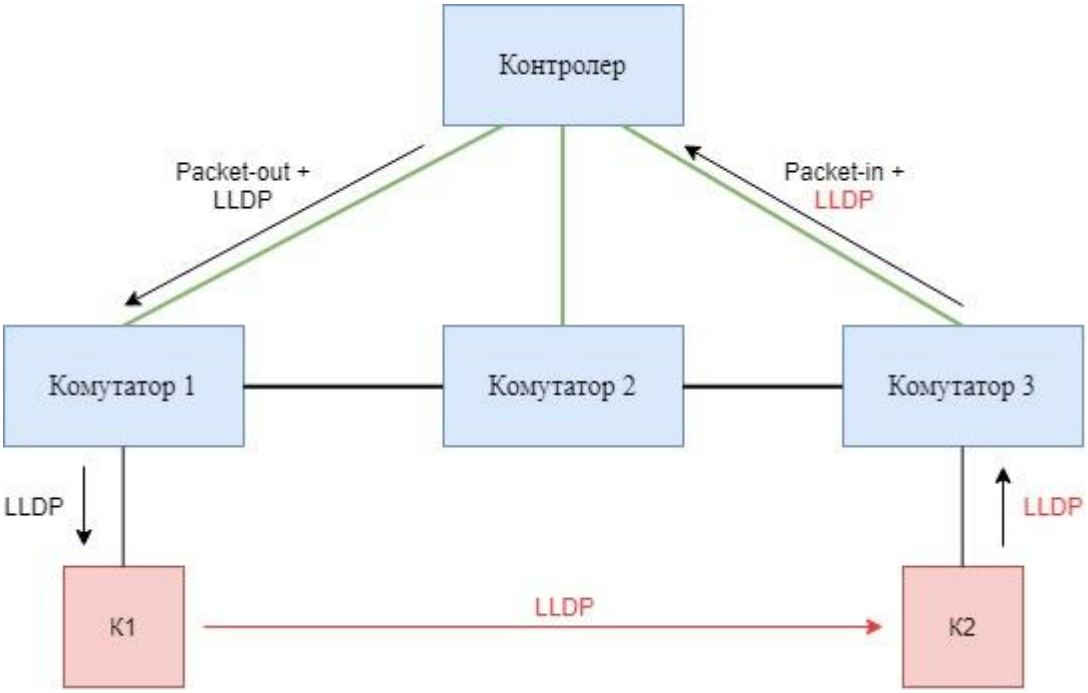

Рисунок 2.4 – Атака Link Fabrication

Інший варіант фабрикації каналів – це LLDP ін’єкція. Прослуховуючи трафік, зловмисник отримує усю LLDP інформацію яку розсилає контролер. Після цього він може розсилати такі самі LLDP пакети у мережу створюючи фіктивні канали між комутаторами або між комутаторами та компрометованим комп’ютером.

- **Controller Fingerprinting.** У [31] описується експлуатація вразливостей протоколу OFDP для створення зловмисного контролеру. Вміст пакетів LLDP залежить від того який контролер керує мережею. Зловмисник (К1 на рисунку 2.4) шукає співпадіння між вмістом LLDP пакетів, які він отримує від Контролеру 1, та записами бази даних сигнатур контролерів, для того щоб визначити який контролер керує мережею. Така інформація може бути дуже важливою у майбутньому для виконання більш серйозних атак на контролер. На рисунках 2.5 та 2.6 відображені перехоплені пакети LLDP від контролерів POX та Floodlight відповідно.

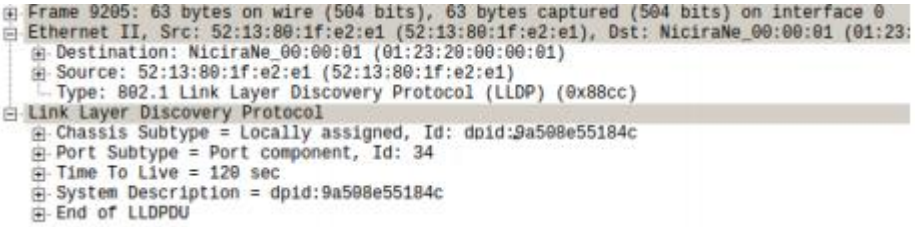

Рисунок 2.5 – Пакет LLDP відправлений POX контролером

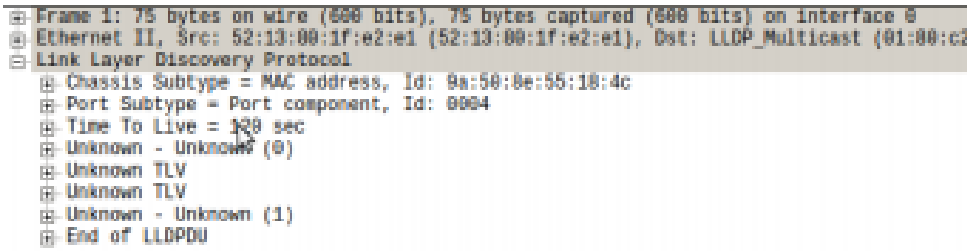

Рисунок 2.6 – Пакет LLDP відправлений Floodlight контролером

- **LLDP Flood.** Також можливий сценарій – це відправка скомпрометованим комп’ютером у мережі великої кількості LLDP пакетів з метою виснажити ресурси контролера. У цьому випадку базові контрдії, такі як блокування інтерфейсів комутаторів або філтрація трафіку, можуть виявитись неефективними, особливо, у мережах, що мають тенденцію швидко змінюватись(наприклад, хмарні мережі), тому що легітимні LLDP пакети можуть не доходити до контролера.

Усі вище наведені вразливості є наслідком недосконалості протоколу OFDP. Тому у [32] пропонується новий стандарт протоколу – sOFDP(secure OpenFlow Discovery Protocol) який призначений вирішити усі вище названі проблеми.

2.3 Модель загроз мереж SDN

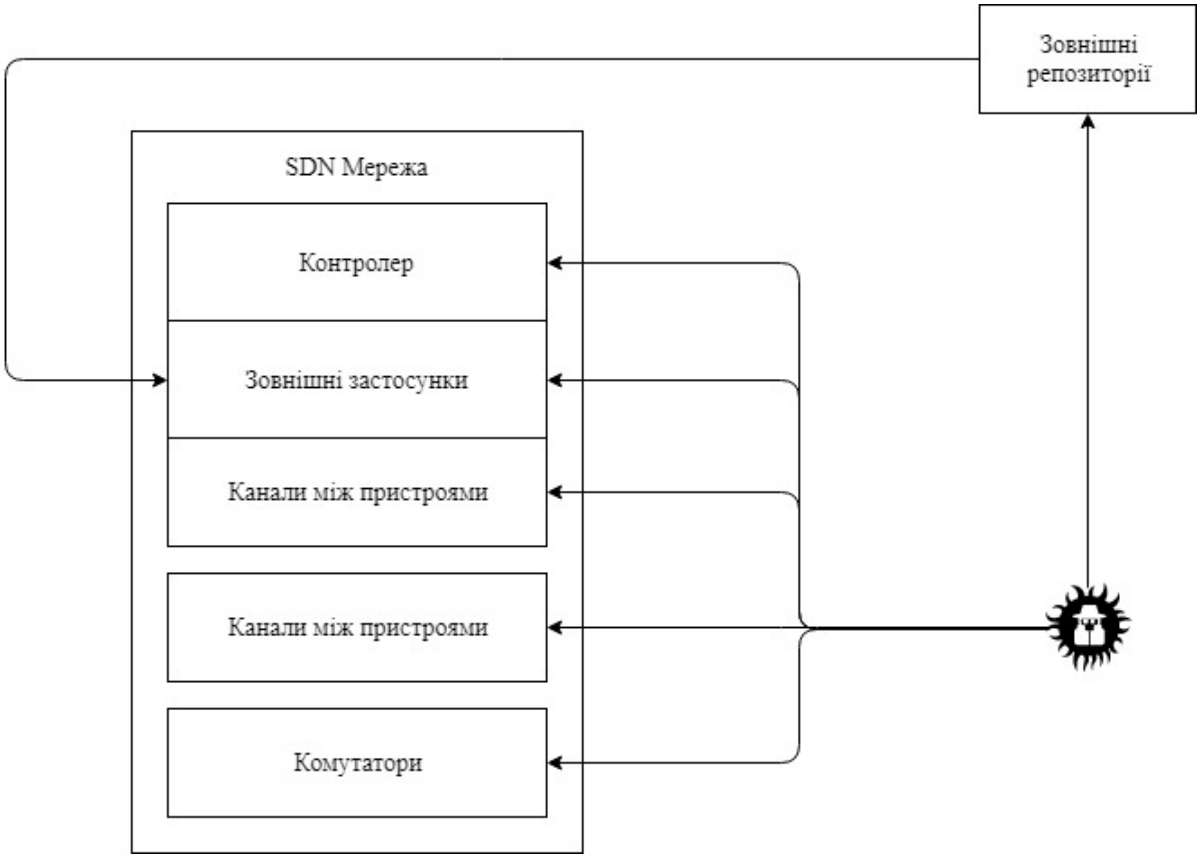

Рисунок 2.7 – Модель загроз SDN архітектури

Детальний опис даної моделі (рис. 2.7) знаходиться у таблиці 2.3. У наведеній таблиці описуються загрози, об’єкти до яких вони реалізуються, мета атаки та методи протидії наведеним загрозам.

Таблиця 2.4 – Опис загроз

№	Об’єкт	Мета	Методи протидії
1	2	3	4
1	SDN Контролер	Виведення мережі з ладу або отримання повного контролю над мережею.	Використання резервування контролеру. Ефективне розміщення контролеру.

1	2	3	4
2	Зовнішні застосунки	Розміщення вразливого коду з метою компрометації мережі	Використання систем сканування на вразливості. Перевірка залежностей.
3	OpenFlow протокол	Несанкціонований доступ до приватної інформації, DoS. Визначення типу контролеру.	Використання захищених каналів зв’язку.
4	Комутатори	DoS	Проактивне кешування правил. Комбіновані правила.
5	Канали між пристроями	Порушення роботи мережі	Використання захищених каналів зв’зку

Висновки до розділу 2

В даному розділі були проаналізовані та наведені переваги SDN мереж у порівнянні з традиційними мережами, з точки зору безпеки, та вразливості мереж SDN й протоколу OpenFlow. Також були описані методи протидії певним атакам і представлені пропозиції з покращення захищенності протоколу OpenFlow. Базуючись на наведених вище вразливостях та методах протидії їм, була розроблена та побудована модель загроз мереж SDN, які використовують протокол OpenFlow.

3 СТВОРЕННЯ МОДЕЛІ SDN МЕРЕЖІ

Для моделювання та тестування SDN мережі були використані:

- Mininet[33] – емулятор мережі, який використовує віртуалізацію процесів для емуляції роботи комутаторів, хостів та каналів. За домогою даного емулятора можно створювати віртуальні мережі з будь-якою топологією для тестування, отримання навичок тощо.
- sFlow-rt[34] – застосунок для перегляду та аналізу метрик з мережевого обладнання, який також надає можливості створення певних лічильників та правил для керування мережею за допомогою API контролера.
- Floodlight[35] – SDN контролер, розроблений на мові програмування Java відкритою спільнотою.
- Hping3[36] – відкритий генератор пакетів, який найчастіше використовується в якості засобу тестування захищеності мережі.
- Open vSwitch[37] – програмний многорівневий комутатор, створений для роботи в гіпервізорах та на комп’ютерах з віртуальними машинами, який підтримує роботу з OpenFlow протоколом.

3.1 Моделювання DoS атаки

За допомогою mininet була створена SDN мережа, яка складається з 2 комутаторів, до кожного з яких підключено 3 комп’ютера(рис. 3.1).

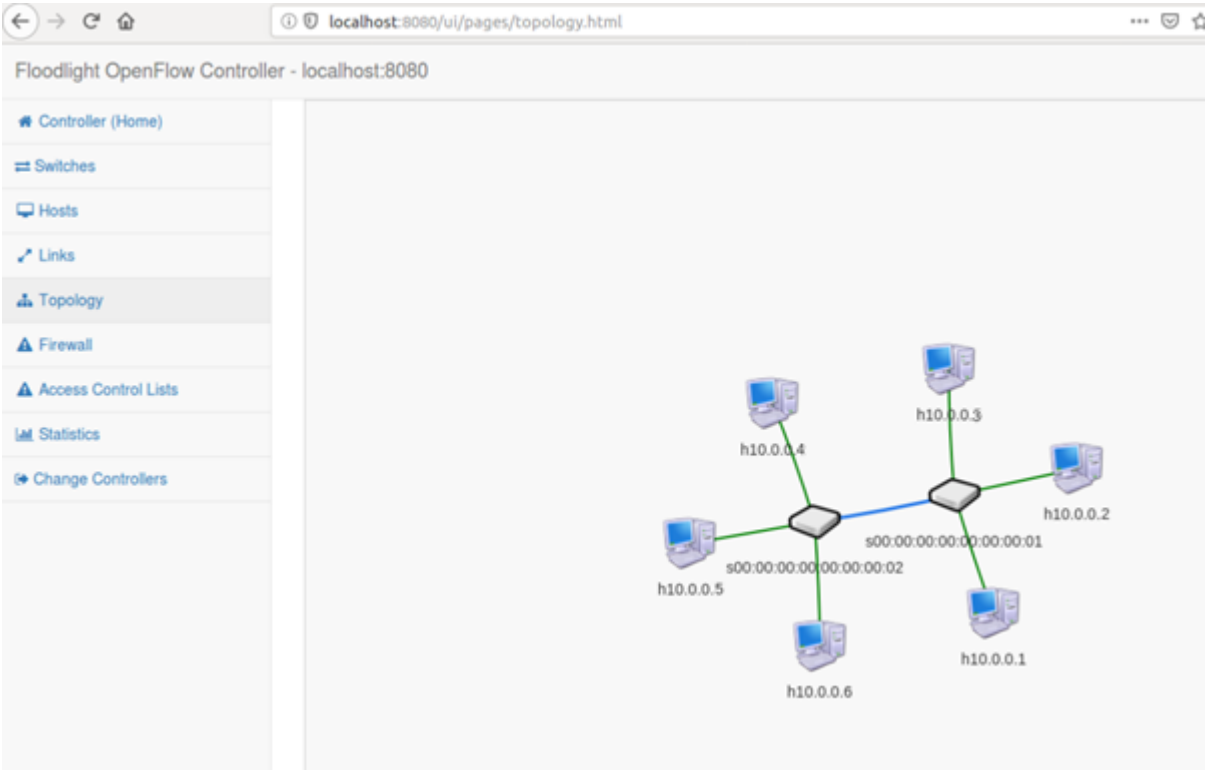

Рисунок 3.1 – топологія мережі

Для симуляції звичайного трафіку, на хості 10.0.0.2 був запущений примітивний HTTP сервер засобами python(рис. 3.2).

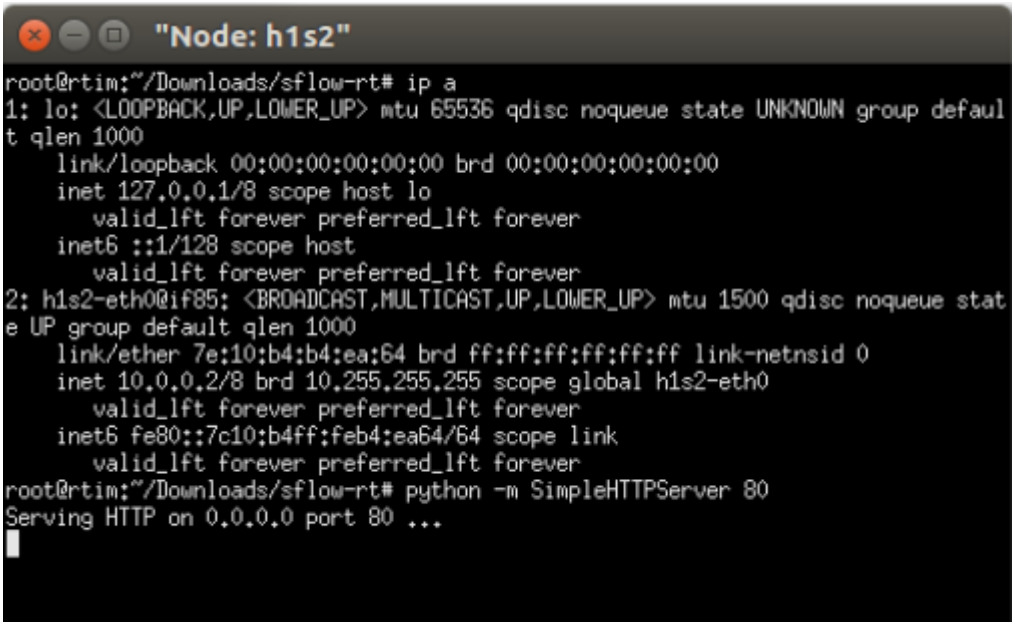

Рисунок 3.2 – запущений http сервер

Відповідно на хості 10.0.0.3 був запущений генератор трафіку з інтервалом у 3 секунди(рис. 3.3).

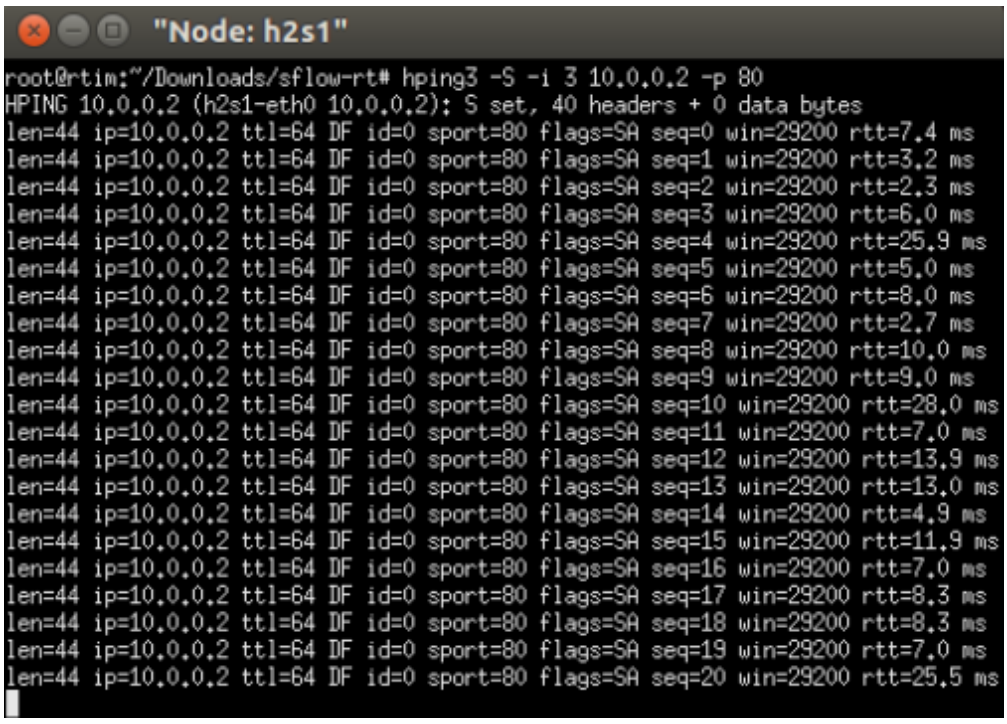

Рисунок 3.3 – симуляція звичайного трафіку

Перед початком атаки використовуючи sflow-rt можна побачити графіки використання мережі у звичайному стані (рис. 3.4).

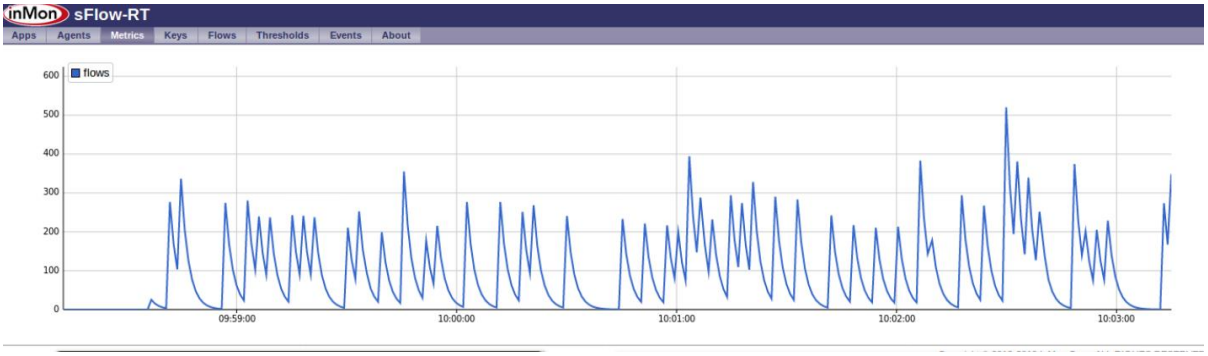

Рисунок 3.4 – графіки звичайного трафіку

Далі, за сценарієм зловмисник на хості 10.0.0.1 запускає генерацію зловмисного трафіку, використовуючи утиліту hping3 у режимі TCP-SYN flood. В даному режимі утиліта намагається якнайшвидше відсилати кожен TCP-SYN пакет не витрачаючи час на вивидення відповіді(рис. 3.5).

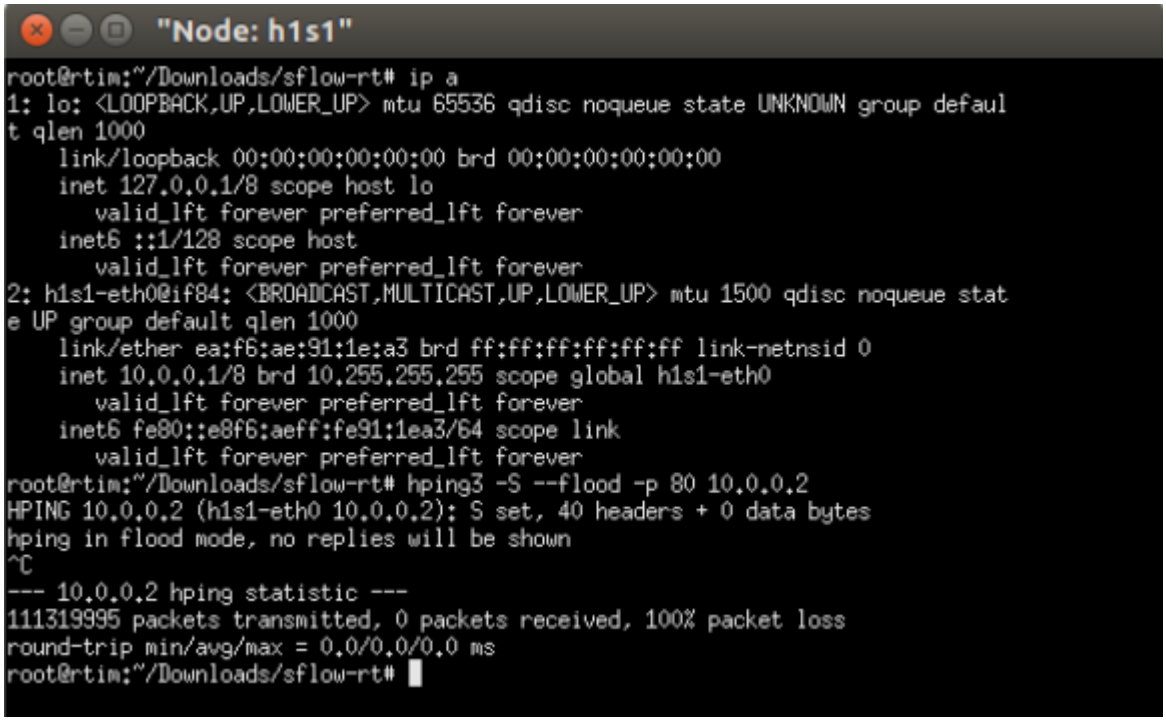

Рисунок 3.5 – генерація трафіку у режимі TCP-SYN flood

Як тільки генерація зловмисних пакетів починається, використання мережі значно збільшується(рис. 3.6). Окрім цього, звичайний незловмисний трафік не може бути оброблений жертвою атаки(рис. 3.7), оскільки усі її ресурси направлені на обробку великої кількості зловмисного трафіку.

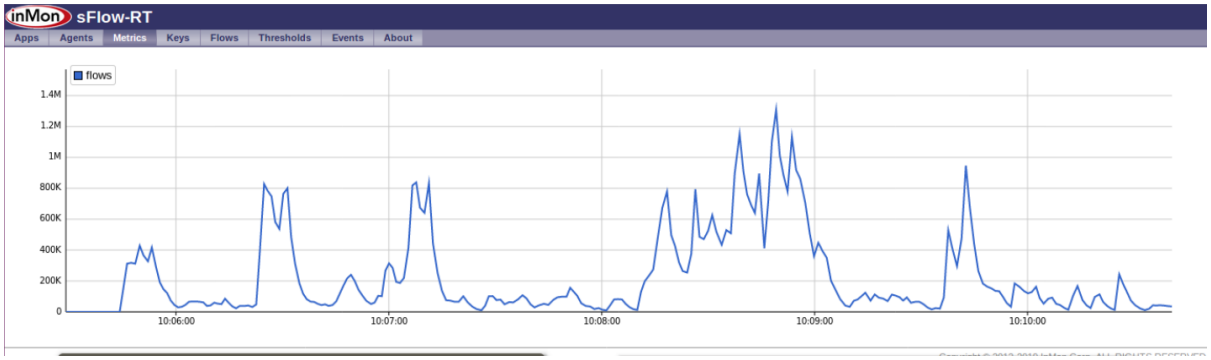

Рисунок 3.6 – використання мережі під час атаки

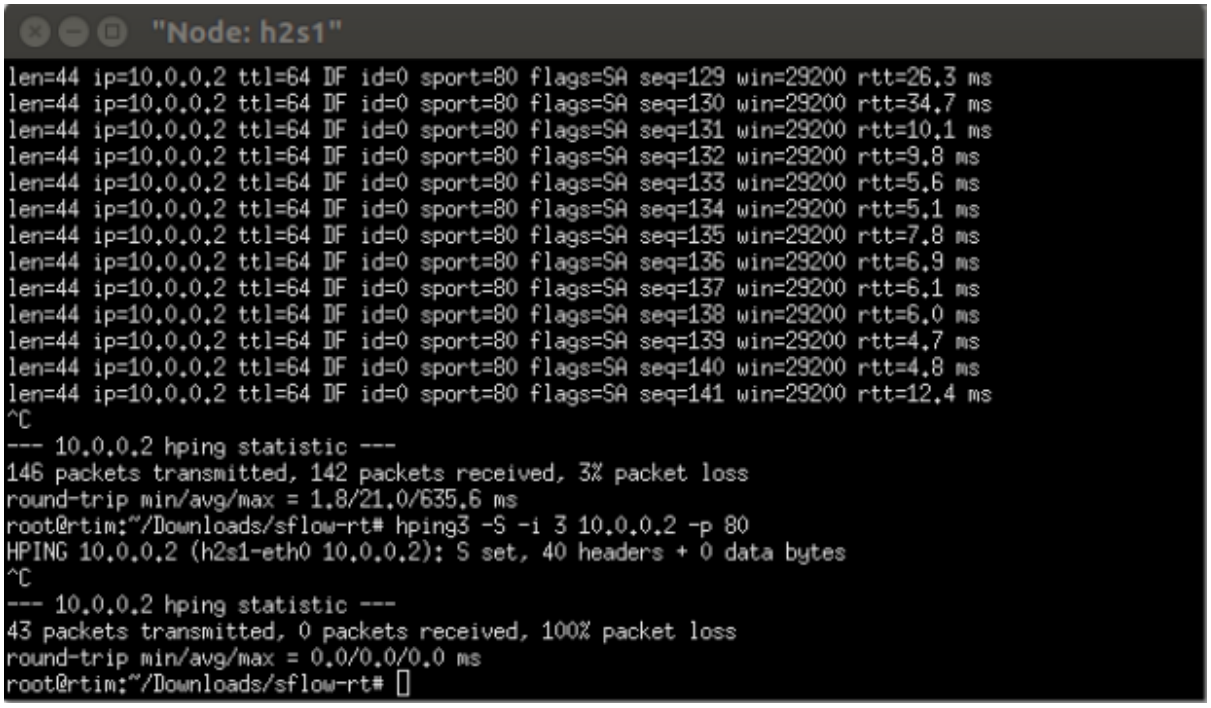

Рисунок 3.6 – жертва атаки не здатна обробляти «звичайний» трафік

3.2 Автоматичне запобігання DoS атак

Для автоматичного припинення DoS на мові програмування Javascript був розроблений скрипт для застосунку Sflow-rt, який створює поріг по кількості відправлених пакетів за вказаний період. У випадку тривалого перевищення створеного порогу, скрипт визначає IP адреси джерела та призначення, і за допомогою REST API контролеру активує правило браундмаєру яке блокує трафік між хостами з отриманими IP адресами. Після визначеного таймауту дане правило деактивується. Іллюстрація роботи скрипта наведена на рисунку 3.7.

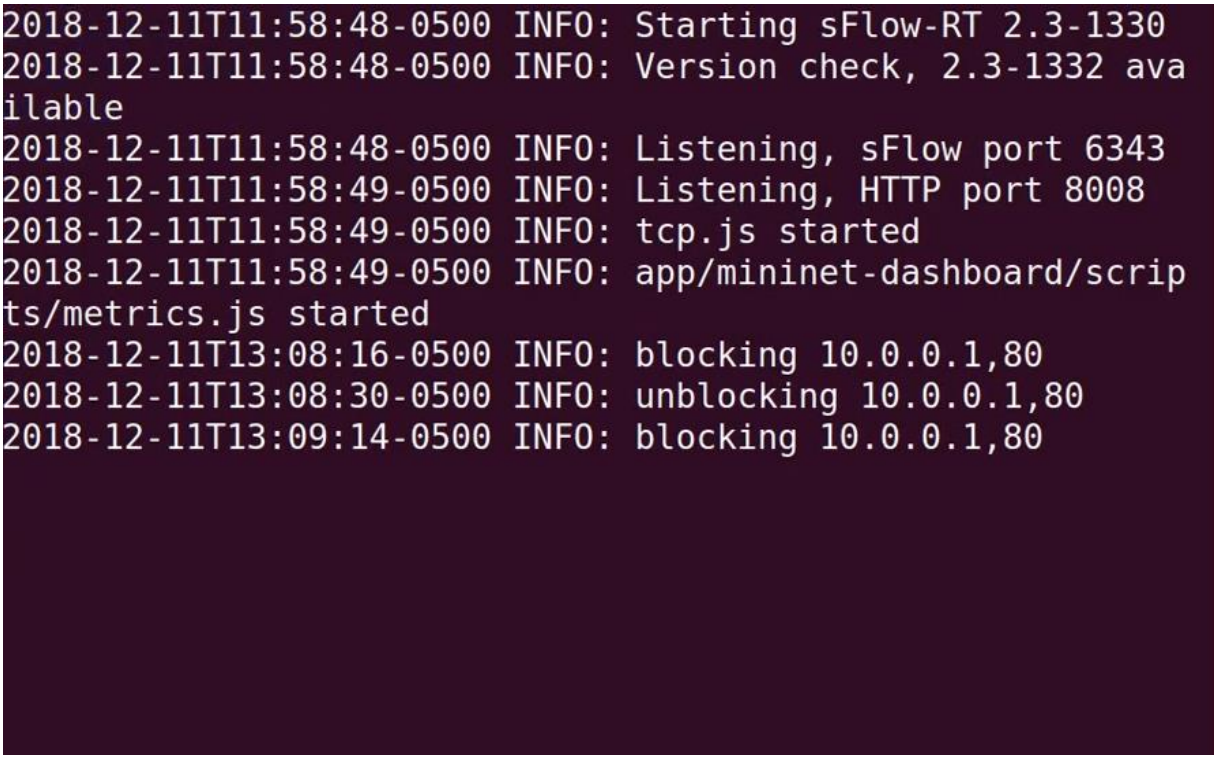

Рисунок 3.7 – блокування трафіку від зловмисного хоста

3.3 Моделювання атаки ARP Spoofing

Зловмисник отримавши контроль над певним хостом може виконати arp-spoofing атаку для:

a) MITM

b) DoS

c) Порушення топології

За сценарієм зловмисник отримав контроль над хостом 10.0.0.1. Після цього він може використати пакет утиліт для аналізу та перехоплення мережевого трафіку dsniff (рис. 3.8).

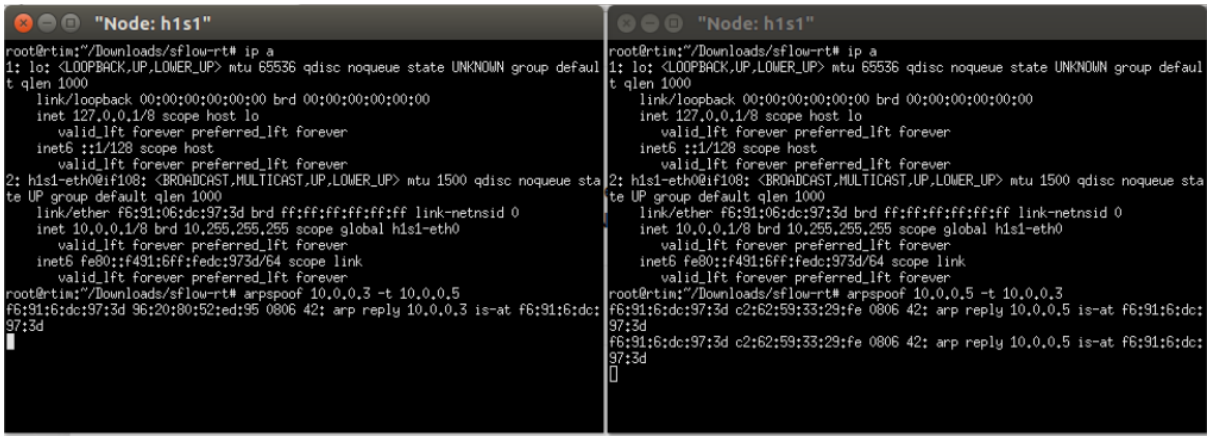

Рисунок 3.8 – атака arp-spoofing

Після цього увесь трафік від 10.0.0.3 до 10.0.0.5 буде проходити через хост 10.0.0.1. Окрім цього, вигляд мережі як її бачить контролер був змінений, що показує постійно змінюючася топологія(рис. 3.9).

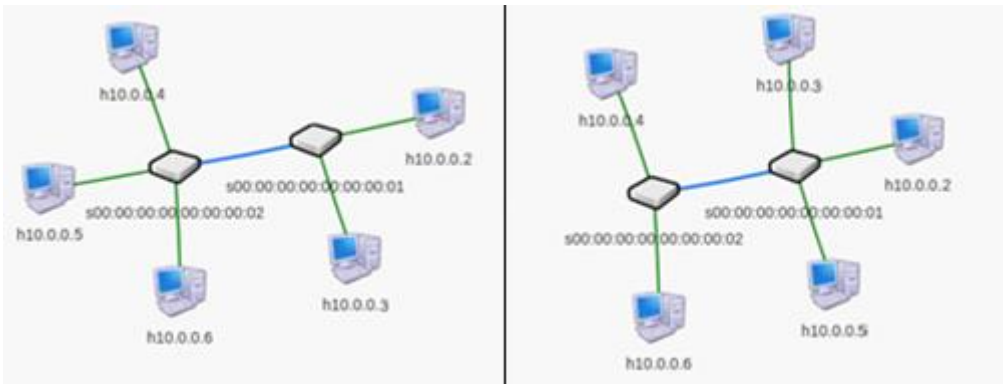

Рисунок 3.9 – Непостійність вигляду мережі для контролера

Висновки до розділу 3

У даному розділі була змодельована мережа з архітектурою SDN за допомогою емулятору mininet, та Openflow контролер на платформі Floodlight. Також були змодельовані DoS та ARP Spoofing атаки на попередньо змодельовану мережу і розроблений javascript скрипт для автоматичного запобігання DoS атаки. Було показано, що у випадку мереж SDN така тривіальна атака як ARP Spoofing створює більшу загрозу, ніж у традиційних мережах, так як окрім перехоплення трафіку зловмисним хостом, також порушується топологія мережі як її бачить контролер.

ВИСНОВКИ

Дипломний проект було виконано з метою оцінки захищеності мереж SDN та протоколу OpenFlow. Були розглянуті та описані вразливі місця, можливі вектори атак, надані рекомендації по протидіїї цим атакам, та побудована модель загроз для SDN мереж. Також були проілюстровані DoS та ARP Spoofing атаки і робота механізми автоматичної протидії DoS атакам.

SDN мережі сьогодні широко використовуються у хмарних технологіях та центрах обробки даних, так як вони значно спрощують процес створення та керування мережею та мають деякі переваги над традиційними ієрархічними мережами з точки зору безпеки. Так SDN мережі надають змогу бачити повний вигляд мережі у реальному часі, мають можливість автоматично відновлюватись після атак та притидіяти їм і надають більший контроль над мережею та доступом до неї.

Однак, централізація керування та програмованість мережі створюють нові задачі по забезпеченню захищеності мережі. Наприклад, очевидним є, те, що централізований контролер, який керує усією мережею є приорітетною ціллю для атакуючого, або здатність віддалено керувати усією логікою пересилання пакетів у мережі за допомогою API вимагає більш серйозного підходу до контролю доступу. Контролери у SDN мережах використовують зовнішні відкриті застосунки, які можуть мати свої власні вразливості, які доволі просто виявити звичайним сканером, що потенційно може стати початковою точкою для атаки.

Також, протокол OpenFlow, який є новим протоколом, що розвивається, може не повністю відповідати вимогам специфікації (наприклад, підтримка TLS), або використовувати механізми, які є недосконалими як за ефективністю, так і за безпекою(наприклад, OFDP), що створює додаткові ризики для мережі.

Варто зазначити, що давно відомі атаки, як, наприклад, ARP Spoofing можуть мати додаткові наслідки, якщо вони виконані у мережі SDN. А саме, порушувати логічну топологію мережі, як її бачить контролер, що може

призвести до того, що деякі пакети будуть невірно пересилатися у мережі, або взагалі відкидатися.

СПИСОК ДЖЕРЕЛ ПОСИЛАНЬ

[1] Software-defined networking: The new norm for networks [Електронний ресурс] // O. N. Foundation – Режим доступу до ресурсу: https://www.opennetworking.org/images/stories/downloads/sdnresources/white-papers/wp-sdn-newnorm.pdf.

[2] Kreutz D. Towards secure and dependable software-defined networks / D. Kreutz. // Proc of 2nd ACM SIGCOMM workshop. – 2013. – С. 55–60.

[3] Open Networking Foundation [Електронний ресурс] – Режим доступу до ресурсу: https://www.opennetworking.org/.

[4] Forwarding and Control Element Separation (ForCES) Protocol Specification. RFC 5810 (Proposed Standart) [Електронний ресурс] / [A. Doria, J. H. Salim, R. Haas та ін.] // IETF. – 2010. – Режим доступу до ресурсу: https://datatracker.ietf.org/doc/rfc5810/.

[5] Forwarding and Control Element Separation (ForCES) Framework. RFC 3746 (Informational) [Електронний ресурс] / L.Yang, R. Dantu, T. Anderson, R. Gopal // IETF. – 2004. – Режим доступу до ресурсу: https://datatracker.ietf.org/doc/rfc3746/.

[6] Hares S. Analysis of Comparisons between OpenFlow and ForCES. Internet Draft (Informational) [Електронний ресурс] / S. Hares // IETF. – 2012. – Режим доступу до ресурсу: https://datatracker.ietf.org/doc/ draft-hares-forces-vs-openflow/.

[7] Software-Defined Networking: Experimenting with the Control to Forwarding Plane Interface. / [E. Haleplidis, S. Denazis, O. Koufopavlou та ін.]. // European Workshop on Software Defined Networks (EWSDN), Darmstadt, Germany. – 2012. – С. 91–96.

[8]. The SoftRouter Architecture / [T. V. Lakshman, T. Nandagopal, R. Ramjee та ін.]. // In Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets). – 2004.

[9] Zheng H. Path Computation Element to Support Software-Defined Transport Networks Control. Internet Draft (Informational) [Електронний ресурс] / H. Zheng,

X. Zhang // IETF. – 2014. – Режим доступу до ресурсу: https://datatracker.ietf.org/doc/draft-zheng-pce-for-sdn-transport/.

[10] Software Defined Networking Extensions for the Locator/ID Separation Protocol. Internet Draft (Experimental) [Електронний ресурс] / [A. Rodriguez-Natal, S. Barkai, V. Ermagan та ін.] // IETF. – 2014. – Режим доступу до ресурсу: http://wiki.tools.ietf.org/id/ draft-rodrigueznatal-lisp-sdn-00.txt.

[11] Languages for Software-Defined Networks / [J. Rexford, M. Freedman, N. Foster та ін.]. // IEEE Community Mag.. – 2013. – С. 51, 128–134.

[12] Frenetic: A Network Programming Language / [N. Foster, R. Harrison, M. Freedman та ін.]. // In Proceedings of the ACM SIGPLAN International Conference on Functional Programming. – 2011.

[13] Composing Software-Defined Networks. / [C. Monsanto, J. Reich, N. Foster та ін.]. // In Proceedings of the USENIX Syposium on Networked Systems Design & Implementation (NSDI). – 2013. – С. 1–14.

[14] Procera: A Language for High-Level Reactive Network Control. / A. Voellmy, H. Kim, N. Feamster. // In Proceedings of the ACM Workshop on Hot Topics in Software Defined Networks (HotSDN). – 2012. – С. 43–48.

[15] NetIDE: First Steps towards an Integrated Development Environment for Portable Network Apps. / [F. M. Facca, E. Salvadori, H. Karl та ін.]. // In Proceedings of the European Workshop on Software Defined Networks (EWSDN). – 2013. – С. 105–110.

[16] OpenFlow: Enabling Innovation in Campus Networks. / [N. McKeown, T. Anderson, H. Balakrishnan та ін.]. // ACM SIGCOMM Comput. Commun.. – 2008. – С. 38,69–74.

[17] Software-Defined Networking Security: Pros and Cons [Електронний ресурс] / D.Mehiar, H. Bechir, G. Mohsen, R. Ammar. – 2015. – Режим доступу до ресурсу: https://www.researchgate.net/publication/274315299_Software-Defined_Networking_Security_Pros_and_Cons.

[18] AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks / S.Shin, P. Yegneswaran, P. Porras, P. Gu. // ACM SIGSAC conference on Computer & communications security. – 2013. – С. 413–424.

[19] Openflow 0.9 features list [Електронний ресурс]. – 2009. – Режим доступу до ресурсу: https://mailman.stanford.edu/pipermail/ openflow-spec/2009-May/000171.html.

[20] OpenFlow Switch Specification [Електронний ресурс] // Open Networking Foundation. – 2015. – Режим доступу до ресурсу: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

[21] Bilal K. M. A Centralized Reputation Management Scheme for Isolating Malicious Controller(s) in Distributed Software-Defined Networks / K. M. Bilal, H. Sufian, M. S. Ghulam. // (IJACSA) International Journal of Advanced Computer Science and Applications. – 2016. – №12.

[22] Openflow switch specification: Version 1.0.0. [Електронний ресурс] // Open Networking Foundation. – 2009. – Режим доступу до ресурсу: http://www. openflow.org/documents/openflow-spec-v1.0.0.pdf.

[23] Hp switch software - openflow supplement. [Електронний ресурс] – Режим доступу до ресурсу: http://.www2.hp.com/bc/docs/support/ SupportManual/c03170243/c03170243.pdf.

[24] Pantou : Openflow 1.0 for openwrt. [Електронний ресурс] – Режим доступу до ресурсу: http://www.openflow. org/wk/index.php/OpenFlow_1.0_for_OpenWRT.

[25] Configuring open vswitch for ssl. [Електронний ресурс] – Режим доступу до ресурсу: http://openvswitch.org/cgi-bin/gitweb.cgi?p= openvswitch;a=blob_plain;f=INSTALL.SSL;hb=HEAD.

[26] Ssl - stanford openflow forums [Електронний ресурс] – Режим доступу до ресурсу: https: //openflow.stanford.edu/forums/topic/210-ssl/page_ _p__737__hl__certificate__fromsearch__1#entry737.

[27] Mul - openflow controller. [Електронний ресурс] – Режим доступу до ресурсу: http://sourceforge.net/projects/mul/.

[28] Opennetworkinglab/flowvisor - github. [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/OPENNETWORKINGLAB/flowvisor.

[29] Floodlight-developers mailing list: Tls support. [Електронний ресурс] – Режим доступу до ресурсу: https://groups.google.com/a/openflowhub.org/forum/.

[30] Nmap Security Scanner [Електронний ресурс] – Режим доступу до ресурсу: https://nmap.org/.

[31] Fingerprinting OpenFlow controllers: The first step to attack an SDN control plane [Електронний ресурс] / [A. Abdelhadi, B. Othmen, T. Nguyen та ін.]. – 2017. – Режим доступу до ресурсу: https://arxiv.org/pdf/1611.02370.pdf.

[32] Limitations of OpenFlow Topology Discovery Protocol [Електронний ресурс] / A.Abdelhadi, T. Nguyen, B. Raouf, P. Guy. – 2017. – Режим доступу до ресурсу: https://arxiv.org/pdf/1705.00706.pdf.

[33] Mininet An Instant Virtual Network on your Laptop (or other PC) [Електронний ресурс] – Режим доступу до ресурсу: http://mininet.org/.

[34] sFlow-RT [Електронний ресурс] – Режим доступу до ресурсу: https://sflow-rt.com/.

[35] Floodlight - an Open SDN Controller [Електронний ресурс] – Режим доступу до ресурсу: http://www.projectfloodlight.org/floodlight/.

[36] hping3 - Linux man page [Електронний ресурс] – Режим доступу до ресурсу: https://linux.die.net/man/8/hping3.

[37] Open v Switch [Електронний ресурс] – Режим доступу до ресурсу: https://www.openvswitch.org/.

ДОДАТОК 1

```
var floodlight = 'localhost';
var controls = {};

setFlow('tcp_flood',
   { keys: 'ipdestination,tcpsourceport', value: 'frames', log: true });

setThreshold('tcp_requests',
   { metric: 'tcp_flood', value: 1000, byFlow: true, timeout: 1 });

setEventHandler(function (evt) {
   var link = topologyInterfaceToLink(evt.agent, evt.dataSource);
   if (link) return;

   var port = topologyInterfaceToPort(evt.agent, evt.dataSource);
   if (!port) return;


   if (!port.dpid || !port.ofport) return;


   if (controls[evt.flowKey]) return;

   var [ipdestination,tcpsourceport] = evt.flowKey.split(',');
   var msg = {
      flows: [
         {
            priority: 4000,
            timeout: 0,
            isPermanent: true,
            deviceId: 'of:' + port.dpid,
            treatment: [],
            selector: {
```

```
                criteria: [
                    { type: 'IN_PORT', port: port.ofport },
                    { type: 'ETH_TYPE', ethType: '0x800' },
                    { type: 'IPV4_DST', ip: ipdestination + '/32' },
                    { type: 'IP_PROTO', protocol: '6' },
                    { type: 'TCP_SRC', tcpPort: tcpsourceport }
                ]
            }
        }
    ]
};

var resp = http2({
    url: 'http://' + localhost + ':8181/onos/v1/flows?appId=policy',
    headers: { 'Content-Type': 'application/json', 'Accept': 'application/json' },
    operation: 'post',
    user: user,
    password: password,
    body: JSON.stringify(msg)
});

var { deviceId, flowId } = JSON.parse(resp.body).flows[0];
controls[evt.flowKey] = {
    time: Date.now(),
    threshold: evt.thresholdID,
    agent: evt.agent,
    metric: evt.dataSource + '.' + evt.metric,
    deviceId: deviceId,
    flowId: flowId
};

logInfo("blocking " + evt.flowKey);
}, ['tcp_requests']);
```

```
setIntervalHandler(function () {
   var now = Date.now();
   for (var key in controls) {
      let rec = controls[key];

      if (now - rec.time < 10000) continue;
      if (thresholdTriggered(rec.threshold, rec.agent, rec.metric, key)) continue;

      var resp = http2({
         url: 'http://' + localhost + ':/onos/v1/flows/'
            + encodeURIComponent(rec.deviceId) + '/' + encodeURIComponent(rec.flowId),
         headers: { 'Accept': 'application/json' },
         operation: 'delete',
         user: user,
         password: password
      });

      delete controls[key];

      logInfo("unblocking " + key);
   }
});
```