ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмний інтерфейс для протоколу Diameter мовою JAVA»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Данііл ПРОКОФ'ЄВ
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. ІСД-41
Данііл ПРОКОФ'ЄВ
Ім'я, ПРІЗВИЩЕ

Керівник: Владислав ЗАВАЦЬКИЙ
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Рецензент:
науковий ступінь, вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти Бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна програма «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ
_____ Каміла СТОРЧАК
«_____» _____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
Прокоф'єв Даніїл Андрійович
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Програмний інтерфейс для протоколу Diameter мовою JAVA.
керівник кваліфікаційної роботи Владислав ЗАВАЦЬКИЙ
(Ім'я, ПРІЗВИЩЕ)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого.2025р. №56
2. Строк подання кваліфікаційної роботи «30» травня 2025 р.
3. Вихідні дані до кваліфікаційної роботи:
   1. Офіційна документація Java.
   2. Офіційна документація Diameter.
   3. Науково-технічна література.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Огляд існуючих інтерфейсів протоколу Diameter.
2. Підбір інтерфейса для аналізу та реалізації.
3. Розобка та тестування створенного інтерфейсу.
5. Перелік графічного матеріалу: презентація
6. Дата видачі завдання «24» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз науково-технічної літератури	24.02 – 04.03.2025	
2	Вивчення теоретичних основ роботи протоколу Diameter	05.03 – 14.03.2025	
3	Дослідження можливостей реалізації нового інтерфейсу	15.03 – 18.03.2025	
4	Підбір інтерфейсу для подальшого впровадження	19.03 – 25.03.2025	
5	Практична реалізація інтерфейсу	26.03 – 20.04.2025	
6	Оцінка результатів та тестування інтерфейсу	20.04 – 24.04.2025	
7	Оформлення роботи: вступ, висновки	25.04 – 21.05.2025	
8	Розробка демонстраційних матеріалів	22.05 – 26.05.2025	

Здобувач вищої освіти _____ Данііл ПРОКОФ'ЄВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Владислав ЗАВАЦЬКИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 51 стор., 30 рис., 1 табл., 20 джерел.

Мета роботи – розробка та обґрунтування ефективного, розширюваного й зручного у використанні програмного інтерфейсу (API) мовою Java для протоколу Diameter.

Об'єкт дослідження – протокол Diameter

Предмет дослідження – інтерфейс протоколу Diameter

Короткий зміст роботи: У даній роботі здійснено теоретичне й практичне дослідження протоколів автентифікації, авторизації та обліку (AAA). Особливу увагу приділено протоколу Diameter, який є розвитком протоколу RADIUS і забезпечує розширені функціональні можливості в системах мережевої безпеки. Проведено детальний аналіз архітектури Diameter, його основних компонентів та інтерфейсів взаємодії. На основі проведеного аналізу було обґрунтовано вибір інтерфейсу IKEv2 як найбільш доцільного для реалізації у межах поставлених завдань. Реалізацію обраного інтерфейсу здійснено з урахуванням вимог до безпеки та сумісності. Проведено тестування функціонування реалізованого рішення, результати якого підтверджують працездатність і відповідність очікуваним характеристикам.

КЛЮЧОВІ СЛОВА: DIAMETER, JAVA, IKE, ІНТЕРФЕЙС, ПРОТОКОЛ, ПОВІДОМЛЕННЯ, ЗАПИТ, ВІДПОВІДЬ.

ABSTRACT

Text part of the bachelor level qualification work: 51 pages, 30 figures, 1 table, 20 sources.

The purpose of the work is to develop and substantiate an efficient, extensible and easy-to-use Java application programming interface (API) for the Diameter protocol.

Object of study - Diameter protocol

Subject of research - interface of the Diameter protocol

Summary of work: In this paper, a theoretical and practical study of authentication, authorization and accounting (AAA) protocols is carried out. Particular attention is paid to the Diameter protocol, which is a development of the RADIUS protocol and provides advanced functionality in network security systems. A detailed analysis of the Diameter architecture, its main components and interaction interfaces is carried out. Based on the analysis, the choice of the IKEv2 interface was substantiated as the most appropriate for implementation within the framework of the tasks. The implementation of the selected interface was carried out taking into account the requirements for security and compatibility. The functioning of the implemented solution has been tested, the results of which confirm its efficiency and compliance with the expected characteristics.

KEYWORDS: DIAMETER, JAVA, IKE, INTERFACE, PROTOCOL, MESSAGE, REQUEST, RESPONSE.

ЗМІСТ

ВСТУП

Актуальність теми:

На сьогоднішній день телекомунікаційні мережі та рішення безпеки даних, адже обсяги трафіку та кількість підключених пристроїв зростають експоненціально. Протокол Diameter відіграє ключову роль у сучасних IP-мережах, виконуючи завдання автентифікації, авторизації та обліку (AAA) в рамках IPsec, LTE, IMS та інших стандартів. Реалізація зрозумілого та надійного API для легкої взаємодії з протоколом Diameter робить конфігурацію, моніторинг та обробку повідомлень набагато простішими, тим самим покращуючи гнучкість та швидкість розгортання нових послуг.

Мета і завдання дослідження:

Метою даної роботи є розробка та обґрунтування ефективного, розширюваного й зручного у використанні програмного інтерфейсу (API) мовою Java для протоколу Diameter.

Завданнями дослідження є:

1. Аналіз літературних джерел;
2. Ознайомлення з інтерфейсами протоколу Diameter;
3. Порівняльний аналіз інтерфейсів;
4. Підбір інтерфейсу для реалізації;
5. Реалізація інтерфейсу;
6. Оцінка та тестування інтерфейсу.

Об’єкт дослідження:

Протокол Diameter

Предмет дослідження:

Інтерфейс протоколу Diameter.

Методи дослідження

1. *аналіз літературних та нормативних джерел;*

2. *функціональний розбір протоколу;*
3. *проектування API;*
4. *прототипування та ітеративна розробка;*
5. *юніт-тесатування.*

Наукова новизна

Наукова новизна полягає в унікальній реалізації інтерфейсу, з розвиненою системою асинхронної обробки зворотних викликів та можливістю гнучко налаштовувати підсистему через програмний інтерфейс.

Апробація результатів та публікації

Кваліфікаційна робота пройшла апробацію на всеукраїнських науково – технічних конференціях таких як: «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України та світу» та «Сучасний стан та перспективи розвитку IoT».

Практична значущість даного дослідження полягає в реалізації інтерфейсу Diameter мовою програмування JAVA.Робота має прикладне значення для розробників програмного забезпечення і дозволяє інтегрувати підтримку протоколу Diameter у Java-додатки, що забезпечує гнучкість та розширює можливості існуючих систем.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО ПРОТОКОЛ DIAMETER

1.1. Загальна характеристика протоколів AAA

Протоколи AAA використовуються для контролю доступу до мережі та управління мережевими пристроями. AAA розшифровується як автентифікація, авторизація та облік.

Рис.1.1 Протоколи AAA [1]

Автентифікація - це перший крок у моделі AAA. Вона передбачає перевірку особи користувача або пристрою, який намагається отримати доступ до мережевого ресурсу. Цей процес використовує різні методи, такі як ім'я користувача та паролі, цифрові сертифікати, біометричні дані та смарт-картки.

Після автентифікації користувача наступним кроком є авторизація. Цей процес передбачає надання або заборону доступу до певних мережевих ресурсів на основі автентифікованої ідентичності. По суті, він встановлює дозволи і привілеї, пов'язані з ідентичністю або роллю користувача. Авторизація становить

основне значення у питанні підтримки безпеки мережі, так як саме вона гарантує, що користувачі отримують доступ лише до тих ресурсів, які їм дозволено використовувати.

Останнім кроком у моделі AAA є облік. Він передбачає відстеження та реєстрацію дій користувачів у мережі. Функція обліку дозволяє мережевим адміністраторам зберігати аудиторський слід дій користувачів, що може бути корисним для усунення несправностей, аналізу тенденцій використання та виявлення незвичної активності, яка може свідчити про порушення безпеки.

1.1.1 Характеристика протокола RADIUS

RADIUS, що розшифровується як (Remote Authentication Dial-In User Service) – це широко використовуваний протокол AAA. RADIUS популярний серед інтернет-провайдерів і традиційних корпоративних локальних і глобальних мереж. Він базується на протоколі UDP і найкраще підходить для доступу до мережі.

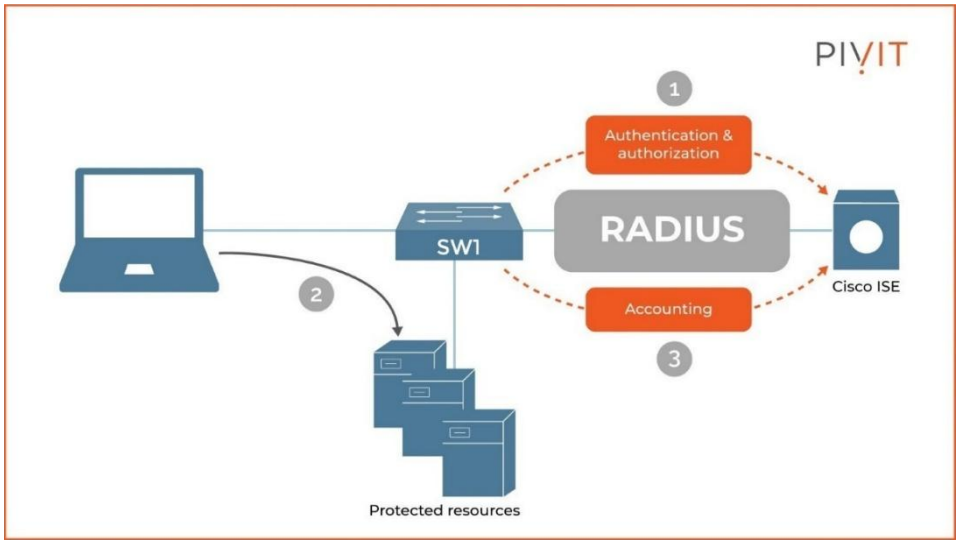


Рис.1.2 Структура протоколу RADIUS[2]

Сильні та слабкі сторони протоколу, сильною стороною RADIUS є те що він є надійним засобом для інтеграції та централізації контролю доступу до мережевих пристроїв різних виробників.

Слабкими сторонами протоколу є те що він базується на UDP, і це може впливати на його облікову функціональність, оскільки втрата пакетів без можливості повторної передачі означає безповоротну втрату облікових даних. А також процеси автентифікації та авторизації в ньому об'єднані, що ускладнює реалізацію деталізованої авторизації.

1.1.2. Характеристика протокола TACACS+

TACACS+ (Terminal Access Controller Access-Control System Plus) є одним з найбільш потужних і гнучких протоколів AAA, який пропонує переваги безпеки в порівнянні з RADIUS - наприклад, шифрування пакетів - і підтримує більш широкий діапазон розширюваності.

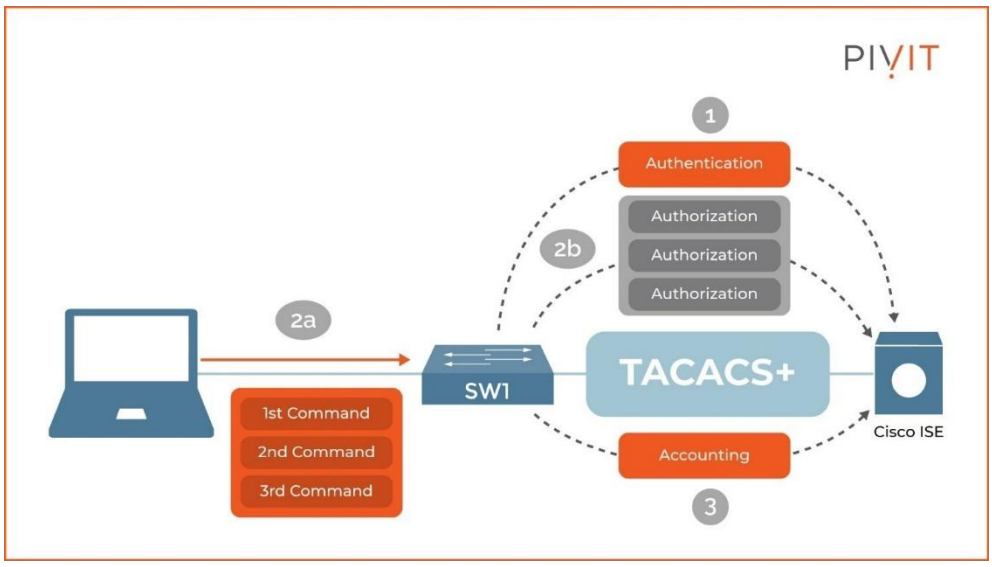


Рис. 1.3 Структура протоколу TACACS+ [2]

TACACS+ був спочатку розроблений компанією Cisco і згодом випущений у відкритий доступ. Але, будучи пропрієтарним протоколом, він не так широко підтримується за межами мережевої екосистеми Cisco.

У протоколу TACACS+ є ряд важливих особливостей, таких як:

- Цей протокол має дуже гарну підримку та сумістність в середині екосистеми Cisco .
- Для передачі пакетів інформації протокол TACACS+ задіє протокол TCP.
- Для зв’язку з протколом TCP використовується 49 порт.
- Якщо пристрій і сервер ACS використовують TACACS+, то всі пакети AAA, якими вони обмінюються, шифруються.
- У протоклі автентифікація, авторизація та облік прелставляють собою окремі елементи.
- Має перевагу над RADIUS у контролі дозволено використовувати користувачеві.

Сильними сторонами цього протоколу є те що на відміну від протоколу RADIUS протокол TACACS+ застосовує для предачі інформації протокол TCP, що забезпечує йому більш надійні та безпечні з'єднання, а також виришує питання з втратою пакетів. Через розділені процеси автентифікації, авторизації та обліку, TACACS+ дозволяє здійснювати більш детальний контроль за цими процесами.

1.1.3. Характеристика протокола Diameter

Diameter - це протокол AAA, який був розроблений для заміни протоколу RADIUS, усуваючи його обмеження, підтримуючи більш надійні заходи безпеки та масштабованість. Він був розроблений на основі і вдосконалює широко розповсюджені протоколи AAA RADIUS (Remote Authentication Dial-In User

Service) і LDAP (Lightweight Directory Access Protocol), надає більш надійні, безпечні і гнучкі транспортні механізми для мобільних мереж передачі даних. Його назва походить від того що Diameter вдвічі більший за RADIUS.

Diameter підтримує сучасні вимоги до доступу до мережі через мережі LTE та мобільні пристрої. Також він використовує TCP або SCTP для транспортування, чим забезпечує надійний зв'язок і розширені функції безпеки, такі як IPsec і TLS.

Перевагами Diameter є те що він підтимує надійну передачу даних, підтримує роумінг через проксі-ланцюжок, а також має функцію узгодження можливостей між клієнтом і серверами. Суттєвих недоліків протокол не має.

1.2. Області застосування протоколу Diameter

Diameter та інші протоколи AAA мають широку область застосування та можуть бути використані в різних сферах

У сфері корпоративних середовищ такі протоколи відіграють важливу роль у захисті доступу до внутрішніх ресурсів.

Вимагаючи автентифікації, ці системи гарантують, що тільки перевірені користувачі можуть отримати доступ до конфіденційних даних і систем компанії, після чого механізми авторизації зможуть визначити конкретні дозволи для кожного користувача, гарантуючи, що співробітники отримують доступ лише до тієї інформації, яка відповідає їхнім ролям, таким чином підтримуючи принцип найменших привілеїв.

З розвитком віддаленої роботи безпека доступу до корпоративних мереж стає все більш важливою. Протоколи AAA, особливо з VPN, дозволяють віддаленим користувачам безпечно автентифікуватися.

З боку Інтернет-провайдерів та інших постачальникі мережевих послуг протоколи AAA використовуються для автентифікації користувачів, які віддалено отримують доступ до інтернет-сервісів або корпоративних мереж.

1.3. Структура протоколу Diameter та основні компоненти

Протокол Diameter працює на основі однорангової архітектури, що дозволяє кожному вузлу, який його реалізує, функціонувати або як клієнт, або як сервер, залежно від структури мережі. Коли користувач ініціює запит на з'єднання, отримуючий вузол Diameter виступає в ролі клієнта. Цей клієнтський вузол, зібравши облікові дані користувача, такі як ім'я користувача та пароль, передає повідомлення із запитом на доступ до іншого вузла Diameter.

Потім вузол сервера Diameter, який отримує доступ, автентифікує користувача на основі наданої інформації. Якщо запит прийнято, відповідь про доступ надсилається відповідному клієнту Diameter, а якщо відхилено, надсилається повідомлення про відмову в доступі.

Протокол Diameter складається з двох логічних шарів. Перший шар це базовий протокол який реалізує функції, пов'язані з надійним транспортом, маршрутизацією та управлінням сеансами, а другий шар це програмні інтерфейси, модулі протоколу які визначають логіку роботи сервісу.

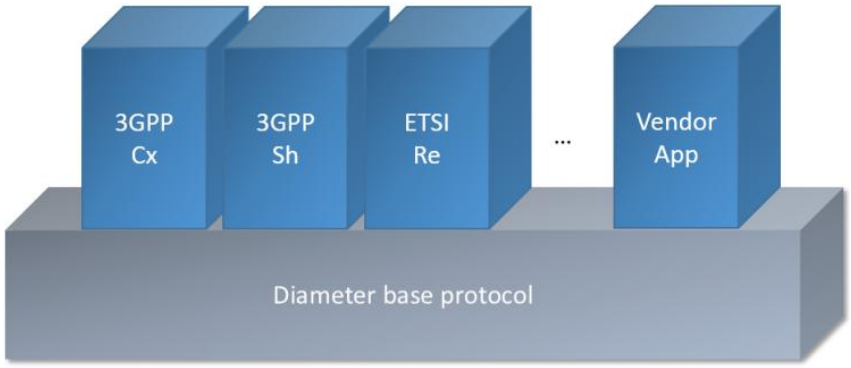


Рис.1.4 Структура протокола Diameter [3]

За допомогою такою структури протокол можна доволі легко масштабувати, шляхом додавання нових інтерфейсів, його модульність дозволяє розширювати функціональність не змінюючи базову логіку та дозволяючи паралельно працювати з різними інтерфейсами.

Модульність та Імплементація нових інтерфейсів в протоколі реалізована за допомогою статичної структури повідомлень, так як сеанс діалогу в Diameter складається з обмінів повідомленнями, де завжди один учасник спочатку надсилає запит, а інший відповідає на нього відповіддю. Повідомлення діляться на такі типи як запит та відповідь. Для цих повідомлень є спільна структура яка має назву заголовок повідомлення Diameter.

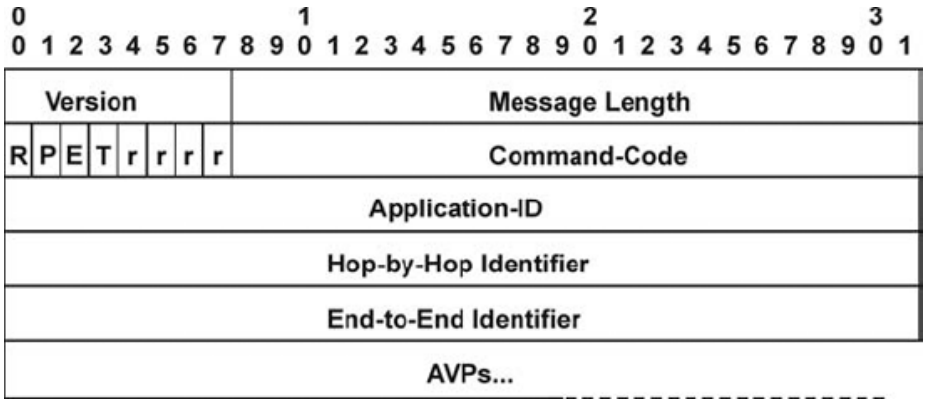


Рис. 1.5 Заголовок пакета Dimeter [4]

Заголовок повідомлення має такі поля:

- Версія (Version) - В полі Версія повинно бути встановлене значення 1, щоб вказати версію діаметра
- Довжина повідомлення (Message Length) - В полі Довжина повідомлення потрібно вказати загальну довжину повідомлення включаючи пари атрибут-значення(Attribute Value Pair)
- Прапори команд (Command Flags) - В полі Прапори команд можуть бути встановлені такі прапори:
  I. R(Request) - Якщо прапор встановлен, повідомлення є запитом, а якщо не встановленно то повідомлення є відповіддю.
  II. P(Proxiable) - Якщо прапор встановлений - повідомлення можна проксі-серверувати, ретранслювати або перенаправляти, а якщо не встановлення повинно оброблятися локально.
  III. E(Error) - Якщо прапор встановлен повідомлення міститиме помилку, цей прапор можна встановлювати тільки в відповідях
  IV. T(Potentially retransmitted message) - Якщо прапор встановлен то це повідомлення може бути дублікатом
  V. r(Reserved) - Ці біти прапорів зарезервовано для подальшого використання, вони не повинні бути в повідомленнях
- Командний код (Command Code) - Поле Код команди використовується для того, щоб передати команди, пов'язаної з повідомленням.
- Ідентифікатор програми (Application-ID) - Поле Ідентифікатор програми використовується для визначення того, до якого додатку(інтерфейсу) відноситься повідомлення.

– Ідентифікатор переходів (Hop-by-Hop Identifier) - Поле Ідентифікатор переходів використовується для того щоб було зрозуміло на який запит прийшла відповідь
– Наскрізний ідентифікатор (End-to-End Identifier) - Поле Наскрізний ідентифікатор використовується для того щоб уникати обробки дублікатів повідомлень.
– Пара Атрибут-значення (Attribute Value Pair (AVP) - Поле Пара Атрибут-значення зберігає інкапсульовану інформацію.

Повідомлення Diameter складаються із заголовка Diameter, за яким слідує певна кількість пар значень атрибутів. Diameter AVP - це основна одиниця інформації всередині повідомлення Diameter, вона містить інформаці У повідомленні Diameter має бути принаймні один AVP.

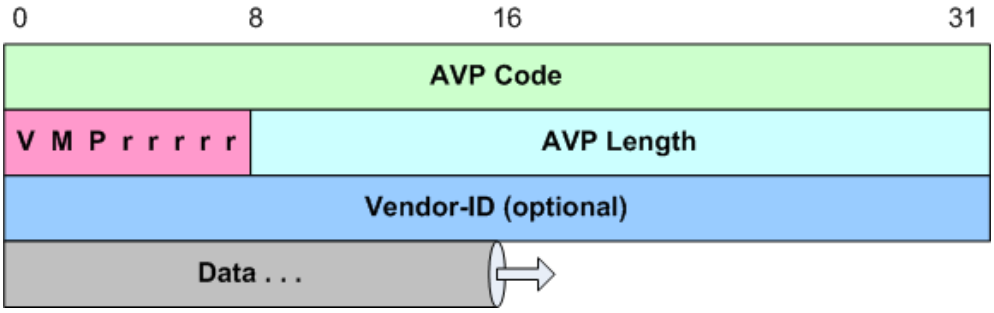


Рис. 1.6 Заголовок пакета Dimeter [5]

AVP має наступний формат:

– Код AVP (AVP Code) – Поле Avp Code містить в собі значення унікально індефікатора AVP.
– Прапори AVP (AVP Flag) - В полі Прапори команд можуть бути встановлені такі прапори:

- V (Vendor-Specific) - Прапор V вказує на те, чи є поле Vendor-Id в AVP. Якщо прапор V встановлено, поле Vendor-Id присутнє в AVP, інакше це поле відсутнє.
- M(Mandatory) - Прапор M вказує що клієнт, сервер, проксі-сервер і агент перекладу повинні підтримувати обробку цього AVP. Якщо обробка не підтримується то повідомлення повинні бути відхилені.
- P(Protected) - Прапор P зарезервовано для майбутнього використання наскрізного захисту.
- Довжина AVP (AVP Length) - Поле AVP Length містить в собі значення довжини AVP в байтах.
- Ідентефікатор постачальник (VendorI-Id) - Поле VendorI-Id це опціональне поле яке присутнє тільки коли встановлений Прапор 'V' у полі прапори AVP.
- Дані (Data) - Поле Data містить в собі корисне навантаження AVP

1.4. Транспортний протокол Diameter

У протоколі Diameter не має власної системи для транспортування повідомлень по мережі тому для цього він використовує вже існуючи рішення для передачі повідомлень такі як TCP, SCTP, TLS та DTLS.

TCP, що розшифровується як Transmission Control Protocol – є протоколом, що забезпечує транспортування повідомлень мережею. Надсилаючи інформаційні пакети він надає можливість надійної доставки повідомлень через інтернет - це і є його головною функцією.

Протокол TCP один із з основних стандартів, що визначають правила роботи в Інтернеті, а також він присутній в списку стандартів, визначених Робочою групою з розробки інтернету (IETF). Цей протокол один із найпоширеніших

протоколів, який використовується в цифровому мережевому зв'язку і забезпечує наскрізну передачу даних.

Протокол TCP структурує дані у зручному форматі для обміну між клієнтом і сервером. Він забезпечує надійність передавання даних у мережі. Перед початком обміну інформацією TCP встановлює стійке з'єднання між відправником і отримувачем, яке зберігається протягом усього сеансу зв’язку.

Щоб спростити передавання повідомлення, він ділить його на менші частини, які мають навзву пакети, це також робиться і для того щоб у випадку втрати частини повідомлення, надсилати повторно не ціле повідомлення а тільки ту частину яка була втрачена. Його пакети автоматично збираються знову в одне ціле, як тільки вони досягають місця призначення. Кожен пакет може пройти різний маршрут між комп'ютером-джерелом і комп'ютером-одержувачем, це залежить від того, чи початковий маршрут, який використовувався, в середині процессу предачі пакетів даних виявився перевантажений або недоступний.

TCP розподіляє завдання зв'язку на рівні, які забезпечують стандартизацію процесу, не вимагаючи від постачальників апаратного та програмного забезпечення самостійного управління. Пакети даних повинні пройти через чотири рівні, перш ніж вони будуть отримані пристроєм призначення, після цього протокол проводить повідомлення у зворотному порядку через всі рівні, щоб повернути повідомлення до його початкового формату.

TCP — це протокол, що працює на основі встановлення стабільного з’єднання. Він створює канал зв’язку між пристроями або програмами і підтримує його доти, доки триває обмін інформацією. У процесі передавання TCP розбиває повідомлення на окремі пакети, нумерує їх, збирає у правильному порядку та надсилає через різні мережеві пристрої — наприклад, маршрутизатори, комутатори чи шлюзи безпеки — до кінцевого отримувача. Якщо якісь пакети губляться дорогою, TCP подбає про їх повторну відправку.

Також він стежить за тим, щоб пакети доставлялися у правильному порядку і з потрібною швидкістю, не перевантажуючи канал.

Щоб почати передавати дані, TCP використовує механізм, відомий як тристороннє рукостискання. Це процес, під час якого пристрій і сервер узгоджують умови з'єднання, обмінюються сигналами підтвердження і лише тоді починають передавати інформацію. Такий підхід дозволяє здійснювати обмін даними в обох напрямках одночасно через кілька TCP-сокетів. Після завершення обміну сторони також узгоджують процедуру розриву з'єднання, щоб усе відбулося коректно й без втрат.

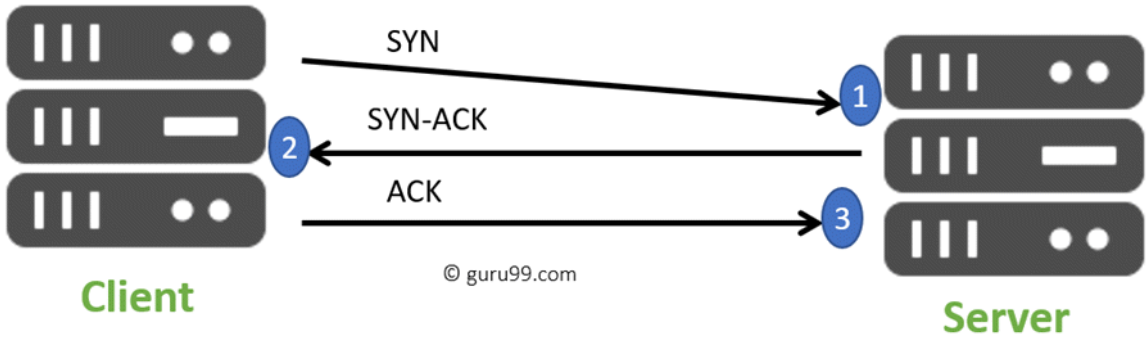


Рис. 1.7

Процес тристороннього рукостискання TCP [6]

SCTP - це новий надійний транспортний протокол, який працює поверх пакетної мережі без з'єднання, такої як IP, він працює на тому ж рівні, що і TCP.

Він встановлює з'єднання між двома кінцевими точками для передачі повідомлень користувача, з'єднання між цими кінцевими точками в протоколі SCTP має назву асоціація. Для того щоб створити асоціацію між кінцевими точками SCTP, одна кінцева точка надає іншій список своїх транспортних адрес. Ці транспортні адреси визначають адреси, які будуть надсилати та отримувати пакети SCTP. SCTP був розроблений для усунення недоліків TCP і пропонує підтверджений, безпомилковий, недубльований транспорт даних користувача.

Трафік IP-сигналізації зазвичай складається з багатьох незалежних послідовностей повідомлень між багатьма різними кінцевими точками сигналізації.

SCTP дозволяє незалежно впорядковувати сигнальні повідомлення в межах декількох потоків, щоб забезпечити послідовну доставку між пов'язаними кінцевими точками. Передача незалежних послідовностей повідомлень в окремих потоках SCTP зменшує ймовірність того, що повторна передача втраченого повідомлення вплине на своєчасну доставку інших повідомлень. На відміну від протоколу TCP у якому для встановлення з’єднання потрібно провести процесс т тристороннього рукостискання у SCTP протоколі для встановелння асоціації виконується процес чотиристороннього рукостискання.

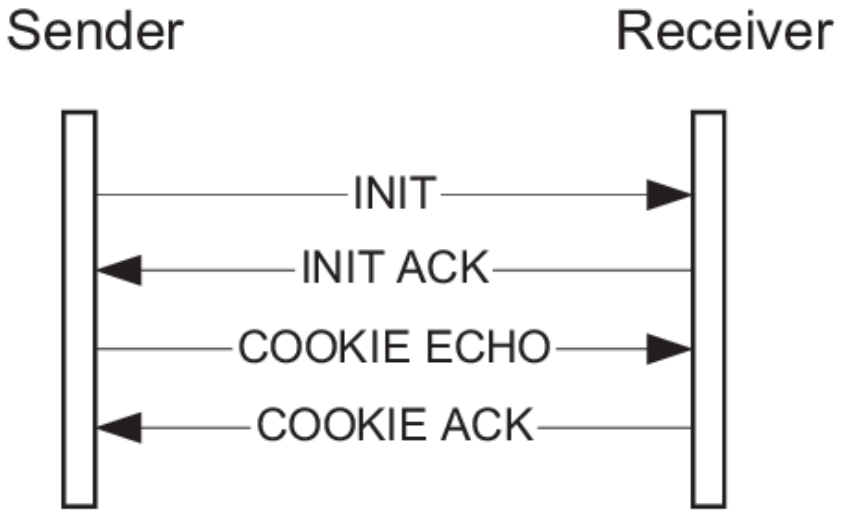


Рис. 1.8 Процес чотиристороннього рукостискання SCTP [7]

Протокол TLS (Transport Layer Security) відповідає за безпеку на транспортному рівні. Це криптографічний протокол, який забезпечує конфіденційність і цілісність даних для інтернет-комунікацій, він використовує

алгоритми шифрування, щоб зменшити ризик перехоплення зловмисниками з'єднань між онлайн-пристроями та додатками.

Протокол TLS для транспортування своїх пакетів використовує як протокол TCP, його основної цілю є встановлення безпечного з'єднання між клієнтом та сервером, для цього протоокл зашифровує сеанс, але перед шифруванням він спочатку перевіряє автентичність з'єднання.

Цей процес, відомий як «протокол рукостискання TLS», починається, коли клієнт надсилає початкове повідомлення на сервер призначення. Сервер відповідає інформацією, що підтверджує його автентичність, і після того, як обидві сторони пройшли перевірку, вони проводять обмін ключами.

Процесс рукостискання TLS - це серія повідомлень, якими обмінюються клієнт і сервер. Конкретні кроки в цьому процесі залежать від того який криптографічий алгоритм використовується а також від того, які набори шифрів підтримує кожна зі сторін. Однак, незалежно від цих факторів, всі рукостискання TLS використовують асиметричне шифрування - але не всі використовують симетричну криптографію.

Процесс рукостискання в протоколі TLS проходить так що клієнт надсилає серверу повідомлення, включаючи інформацію про те, який протокол TLS і набір шифрів він підтримує.

Сервер на це повідомлення відповідає повідомленням яке включає відповідний сертифікат TLS, який містить його відкритий ключ.

Щоб перевірити справжність сертифіката надісланого сервером та підтвердити його справжність, клієнт перевіряє цей TLS-сертифікат в центрі сертифікації, який видав цей сертифікат серверу. Ця преревірка підтверджує, що сервер є тим, за кого себе видає, і що ним керує справжній власник домену.

Після перевірки клієнт генерує попередній секретний ключ, цей ключ також відомий як сеансовий ключ. Сервер розшифровує ключ сеансу за

допомогою власного закритого ключа і після вдалого розшифрування цього ключа сервером, обмін ключами між ними завершується , після цього обидві сторони мають встановлене безпечне з'єднання.

DTLS (Datagram Transport Layer Security) - це протокол, який забезпечує конфіденційність і цілісність даних при передачі даних за допомогою дейтаграмних протоколів, які зазвичай використовуються для додатків, що вимагають зв'язку в реальному часі і низької затримки, таких як потокове мультимедіа, передача голосу по IP (VoIP). Протокол DTLS заснований на потокоорієнтованому захисті транспортного рівня (TLS), забезпечуючи аналогічний рівень безпеки. Як дейтаграмний протокол, DTLS не гарантує порядок доставки повідомлень або навіть те, що повідомлення будуть доставлені взагалі. Однак, DTLS також отримує переваги дейтаграмних протоколів, зокрема, менші накладні витрати і меншу затримку.

Протокол DTLS працює подібно до TLS протоколу, але він розроблений таким чином, щоб бути стійким до ненадійності, притаманної службам передачі дейтаграм.

Рукостискання в цьому протоколі відбуваються таким чином Спочатку протокол DTLS проводить рукостискання між клієнтом і сервером для узгодження різних параметрів безпеки, таких як алгоритм шифрування і криптографічні ключі, які будуть використовуватися під для шифрування. Це рукостискання призначене для того, щоб протистояти проблемам втрати і переупорядкування пакетів, які часто зустрічаються в дейтаграмних мережах.

Встановлення безпечного сеансу зв'язку відбувається після того як рукостискання відбулось та пройшло успішного. Всі наступні дані, що передаються в цьому сеансі, шифруються та аутентифікуються відповідно до узгоджених параметрів.

На відміну від TLS протоколу, DTLS протокол включає в себе механізм обробки втрати і переупорядкування пакетів. Оскільки такі дейтаграмні протоколи, як UDP, не гарантують доставку, DTLS включає в себе порядкові номери і повторну передачу втрачених пакетів, гарантуючи, що послідовність і повнота зашифрованих записів будуть збережені.

РОЗДІЛ 2. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ РЕАЛІЗАЦІЇ ПРОГРАМНИХ ІНТЕРФЕЙСІВ ПРОТОКОЛУ DIAMETER

2.1. Огляд існуючих інтерфейсів протоколу Diameter

Інтерфейси Diameter діють як канали зв'язку, що з'єднують різні компоненти мережі, забезпечуючи виконання таких важливих завдань, як:

- автентифікація користувачів;
- авторизація послуг;
- управління кредитуванням у режимі реального часу;
- управління сеансами.

Кожен інтерфейс відповідає певному додатку, ідентифікованому унікальним ідентифікатором, і працює відповідно до конкретних протоколів і структур повідомлень, визначених встановленими стандартами такими як: RFC6733, RFC8506, 3GPP TS 29.229 та інші.

Існують високорівневі API (програмні інтерфейси),які призначені для спрощення складності протоколу, що і дозволяє розробникам швидко створювати додатки без глибоких знань протоколу. На відміну від них, API середнього та низького рівня надають доступ до внутрішньої роботи протоколу, пропонуючи більшу гнучкість та кастомізацію в залежності від потреб програми.

Стек Diameter структурований та скомпонований відповідно до багаторівневої структури протоколу, що включає ядро Diameter Base, яке працює базуючись на надійних транспортних протоколах, таких як TCP, TLS, SCTP або IPSec. Вище над цим ядром інтегровані окремі додатки IMS Diameter як окремі модулі, що забезпечує модульну та масштабовану функціональність (рис. 2.1).

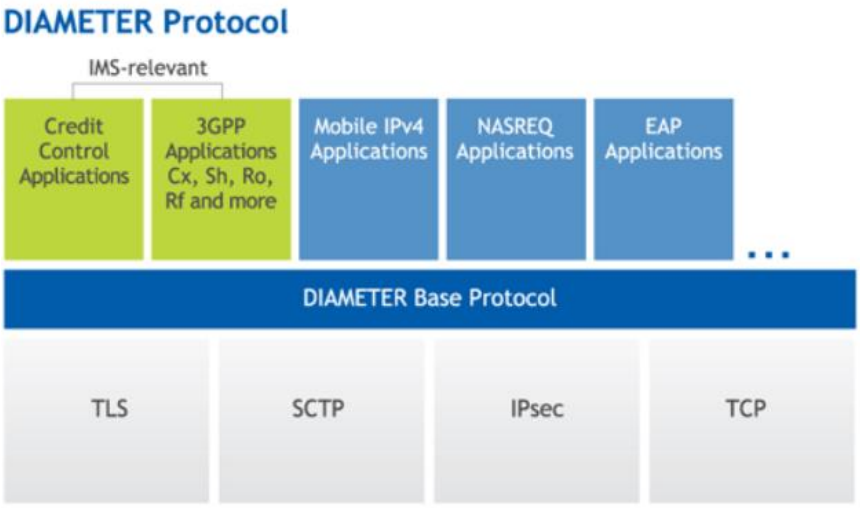


Рис. 2.1. Приклад інтергації окремих додатків IMS Diameter [6]

2.1.1. Інтерфейс Cx/Dx для IMS

Мультимедійна підсистема IP або IMS (рис. 2.2) - це заснована на стандартах архітектурна структура для надання мультимедійних послуг зв'язку, таких як передача голосу, відео та текстових повідомлень через IP-мережі. Специфікації IMS були розроблені в рамках проекту партнерства 3-го покоління (3GPP) для стандартизації впровадження цих послуг у мобільних мережах наступного покоління [7].

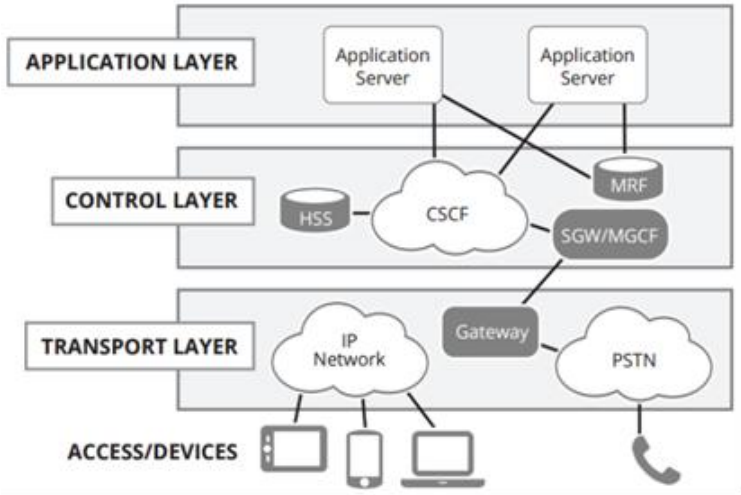


Рис. 2.2. IP Multimedia Subsystem [7]

Інтерфейси Cx і Dx відіграють вирішальну роль у структурі IMS (IP Multimedia Subsystem), яка підтримує надання мультимедійних послуг IP. Ці інтерфейси забезпечують взаємодію між функціями управління сеансами зв'язку (CSCF) і домашнім абонентським сервером (HSS) або функцією визначення місцезнаходження абонента (SLF).

Зокрема, інтерфейс Cx з'єднує обслуговуючу CSCF (S-CSCF) з HSS. Він керує кількома ключовими завданнями, такими як:

- Аутентифікація користувача: Підтвердження особи користувача.
- Авторизація користувача: Перевірка того, що користувачеві дозволено користуватися запитуваними послугами.
- Отримання даних про підписку: Доступ до профілів користувачів та інформації, пов'язаної з підпискою.

2.1.2. Інтерфейс Gx для політик доступу

Інтерфейс Gx розташований між функцією впровадження політики і тарифікації (PCEF) і функцією впровадження політики і правил тарифікації

(PCRF). Інтерфейс Gx використовується для створення і видалення правил PCC з PCRF в PCEF і передачі подій площини трафіку з PCEF в PCRF. Інтерфейс Gx може бути використаний для контролю тарифікації, управління політикою або для обох цих цілей шляхом застосування AVP, що відповідають додатку.

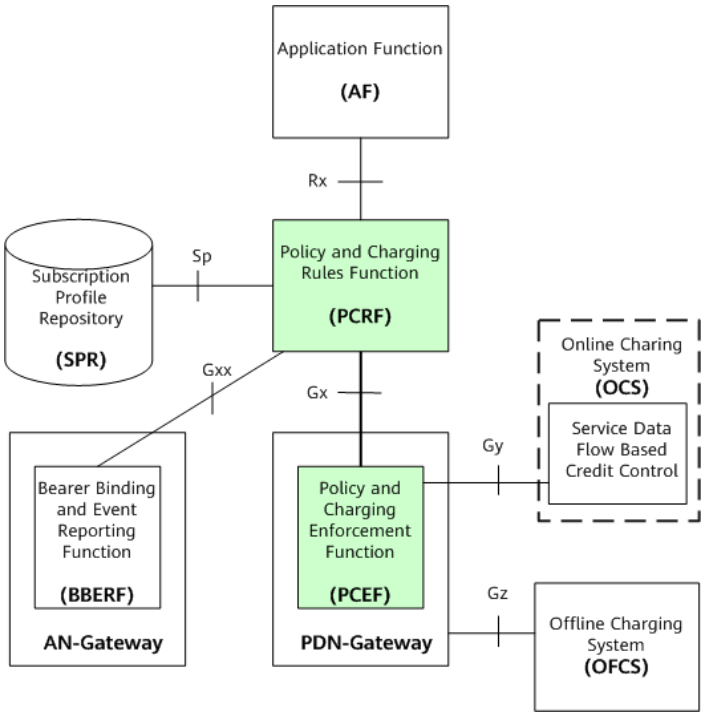


Рис. 2.3. Точка відліку Gx в архітектурі Policy and Charging Control (PCC) [8]

Інтерфейс Gx виконує кілька ключових функцій:

- він передає правила політики та контролю за нарахуванням (PCC) з PCRF до PCEF;
- він видаляє правила PCC з PCEF, коли це необхідно;
- надсилає сповіщення про події з PCEF назад до PCRF.

Правила PCC поділяються на два основні типи:

Динамічні правила PCC: Ці правила надсилаються PCRF до PCEF динамічно через інтерфейс Gx. Вони можуть бути створені, оновлені або видалені в будь-який час. Динамічні правила генеруються на основі інформації про

обслуговування користувачів та параметрів політики, налаштованих у PCRF. Кожному динамічному правилу PCC PCRF присвоює унікальне ім'я правила нарахування. Коли застосовується політика, PCRF надсилає до PCEF детальні параметри правила, такі як якість обслуговування (QoS) та час кредитного контролю (CC-Time). Пара атрибутів-значень (AVP) «ім'я правила нарахування-значення» використовується PCRF для зміни або видалення відповідного правила в PCEF.

Статичні правила PCC: Ці правила заздалегідь встановлені в PCEF. Хоча їх визначення є фіксованими, PCRF може активувати або деактивувати ці статичні правила в будь-який час. Статичні правила визначаються виключно PCEF, і PCRF контролює їх використання, надсилаючи відповідне повідомлення AVP з назвою правила нарахування. Таке повідомлення набуває чинності лише після того, як PCRF та PCEF узгодять умови під час переговорів.

2.2.3. Інтерфейс Rx для управління ресурсами

Інтерфейс Rx забезпечує зв'язок між прикладною функцією (AF) і функцією політики і правил тарифікації (PCRF). Його основна роль полягає в обміні детальною інформацією про додатки на рівні сеансу, що дозволяє PCRF застосовувати політику і правила тарифікації, адаптовані до конкретних вимог кожного сеансу обслуговування і профілю абонента.

Структура інтерфейсу Rx:

Функція додатку AF. AF - це мережевий компонент, який відповідає за управління сигналами і контроль для конкретних прикладних послуг, таких як VoIP, потокове відео або інші послуги, що базуються на даних. Через інтерфейс Rx, AF взаємодіє з PCRF, запитуючи управління політикою і розподіл ресурсів для цих послуг. Він надає життєво важливу інформацію про сеанс, таку як типи

медіа, вимоги до якості обслуговування (QoS) і тригери подій, які допомагають PCRF застосовувати відповідну політику.

Функція політики та правил тарифікації (PCRF). PCRF діє як основний орган, що приймає рішення в рамках політики та контролю за нарахуваннями (PCC). Використовуючи інформацію, надану AF, PCRF встановлює, які політики та правила тарифікації слід застосовувати до кожної сесії. Він визначає, як має забезпечуватися QoS, як розподіляються ресурси і які правила тарифікації застосовуються, а потім передає ці рішення назад до AF. Крім того, PCRF координує роботу з іншими компонентами, такими як функція забезпечення дотримання політик і тарифікації (PCEF), щоб гарантувати, що політики належним чином застосовуються протягом усього потоку даних.

2.2. Аналіз протоколу IPsec

IPsec, скорочено від Internet Protocol Security, являє собою комплексний набір протоколів і криптографічних методів, спрямованих на захист даних, що передаються через Інтернет. Розроблений IETF (Internet Engineering Task Force), IPsec забезпечує безпеку на мережевому рівні шляхом аутентифікації та шифрування IP-пакетів.

Спочатку IPsec визначив два фундаментальних протоколи:

- заголовок автентифікації (AH), що забезпечує цілісність даних і захищає від атак повторного відтворення;
- інкапсуляцію корисного навантаження безпеки (ESP), що пропонує послуги шифрування та автентифікації для захисту даних.

Крім того, структура IPsec включає в себе протокол Internet Key Exchange (IKE), який відіграє вирішальну роль у створенні спільних секретних ключів для створення асоціацій безпеки (SA). Ці SA необхідні для управління процесами

шифрування і дешифрування шляхом визначення параметрів безпеки, узгоджених між двома взаємодіючими сторонами. Узгодженням цих асоціацій безпеки зазвичай керують спеціальні мережеві пристрої, такі як маршрутизатори або брандмауери, розташовані між мережами.

Принцип роботи IPSec включає в себе 5 основних кроків (рис. 2.4):

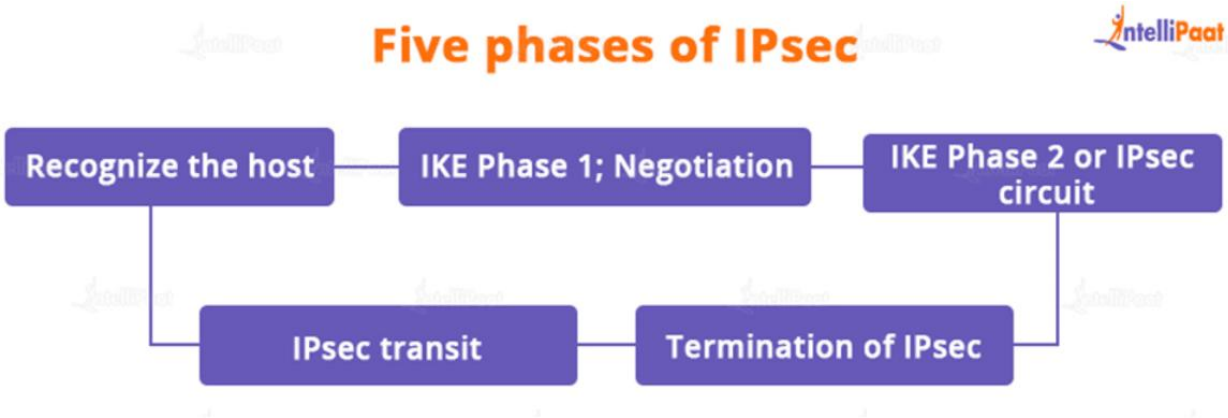


Рис. 2.4. Фази IPSec [9]

– Розпізнавання хоста:

Коли хост визначає, що пакет потребує захисту, починається обробка IPsec. Ці пакети, потрапляють під дію політик безпеки, що гарантує застосування шифрування та автентифікації перед надсиланням. Аналогічно, вхідні пакети перевіряються на належний захист, якщо вони вважаються цікавими.

– Фаза 1 IKE (узгодження):

На цій фазі хости встановлюють захищений канал і узгоджують параметри IPsec. Вони автентифікують один одного і домовляються про методи шифрування та автентифікації. Переговори можуть відбуватися в основному режимі, коли обидві сторони безпечно обмінюються пропозиціями, або в агресивному режимі, який є швидшим, але менш безпечним, що дозволяє швидше встановити з'єднання IPsec.

– Етап 2 IKE (налаштування тунелю IPsec):

Хости налаштовують тунель IPsec (рис. 2.5), узгоджуючи алгоритми та обмінюючись ключами для шифрування та дешифрування даних. Вони також обмінюються криптографічними нонсетами для аутентифікації сеансу.

– Передача даних:

Після встановлення безпечного тунелю хости обмінюються зашифрованими даними, використовуючи узгоджені асоціації безпеки IPsec (SA).

– Завершення:

Сеанс IPsec завершується після досягнення встановленого ліміту даних або таймауту. Хости повідомляють один одного, роз'єднуються і стирають ключі, які використовувалися для захисту сеансу.

Режими IPSec (рис. 2.5):

– Тунельний режим - весь оригінальний IP-пакет, включаючи заголовок і корисне навантаження, шифрується і вставляється в абсолютно новий IP-пакет. Цей режим зазвичай використовується для з'єднань між мережами.;
– Транспортний режим - шифрує лише корисне навантаження (записи) справжнього IP-пакета, залишаючи IP-заголовок недоторканим. Зазвичай використовується для наскрізного зв'язку між хостами або гаджетами.

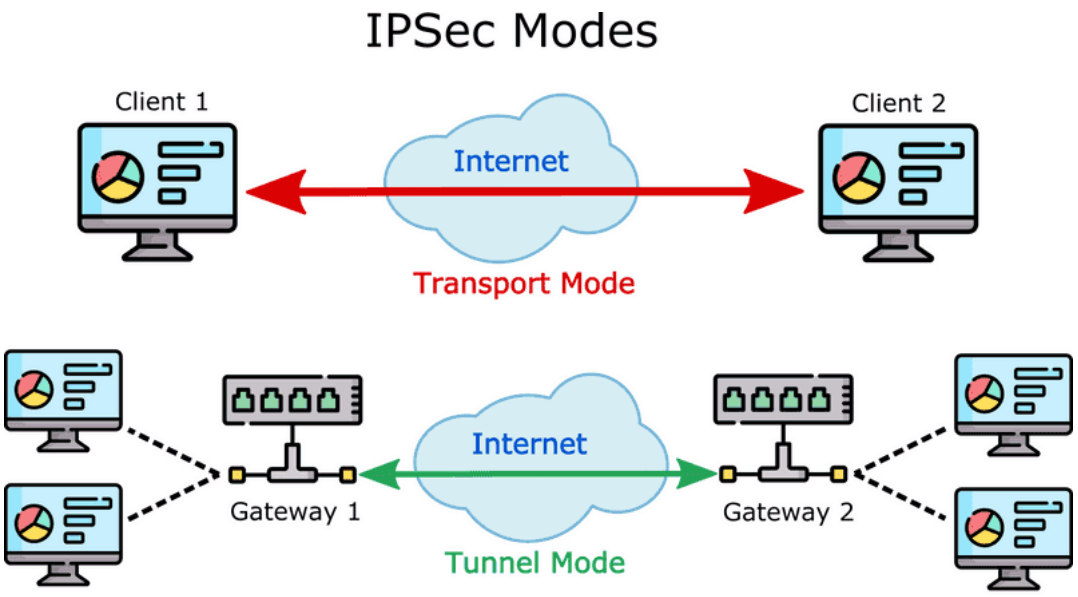


Рис. 2.5. Режими IPSec [10]

IPsec використовується для вирішення низки важливих завдань, серед яких: захист маршрутизаторів при передачі інформації через публічні мережі, такі як Інтернет, шифрування даних, що передаються додатками, для забезпечення їх конфіденційності, швидка та надійна автентифікація відправника, якщо він вже відомий системі.

Також застосовується для забезпечення безпеки мережевого трафіку шляхом створення зашифрованих каналів - так званих тунелів IPsec - які захищають всю інформацію, що передається між двома точками.

Організації використовують IPsec для запобігання атакам повторного відтворення. Ця атака, також відома як атака «зловмисника посередині», передбачає перехоплення та зміну даних шляхом перенаправлення їх через проміжний пристрій. Для захисту від неї IPsec присвоює кожному пакету унікальний порядковий номер і виконує перевірки для виявлення дублікатів або змінених пакетів.

2.3. Порівняльний аналіз IKEv1 та IKEv2

IKEv1 та IKEv2 — це дві версії протоколу Internet Key Exchange (IKE), який служить для встановлення захищеного каналу зв’язку між двома мережами.

IKEv2 є оновленою і покращеною версією IKEv1. Незважаючи на те, що обидва протоколи мають подібну функціональність, IKEv2 розроблено з метою підвищення рівня безпеки, надійності та швидкості у порівнянні з IKEv1. Порівняння інтерфейсів наведено в таблиці 2.1.

Таблиця 2.1

Особливість	IKEv1	IKEv2
Складність процесу	Використовує дві окремі фази з кількома обмінами повідомленнями; Фаза 1 включає основний і агресивний режими, обидва з яких вимагають декількох прямих і зворотних зв'язків для встановлення безпеки.	Спрощує переговори, об'єднуючи етапи, зазвичай потрібно лише два обміни повідомленнями для створення асоціації безпеки та виконання автентифікації, що скорочує накладні витрати.
Продуктивність і швидкість	Основний режим забезпечує надійний захист, але може бути повільним через численні обходи повідомлень; Агресивний режим прискорює цей процес, але жертвує деякою безпекою.	Більш впорядкований з меншою кількістю обмінів, краще керує повторними спробами та помилками, що призводить до швидшого налаштування з'єднання.
Покращення безпеки	Більш оптимізований з меншою кількістю обмінів,	Включає посилені засоби захисту, такі як

	краще керує повторними спробами та помилками, що призводить до швидшого встановлення з'єднання. Агресивний режим може спричинити витік інформації для прискорення процесу, що потенційно послаблює захист.	інтегрований обхід NAT, сильніші криптографічні стандарти та підвищену стійкість до атак.
Цілковита пряма секретність (PFS)	Підтримується, але дещо складний у налаштуванні та обслуговуванні.	Простіший у використанні з більш потужною підтримкою, що гарантує безпеку сеансових ключів, навіть якщо довгострокові ключі скомпрометовані.
Підтримка мобільності	Не підтримує роумінг або кілька IP-адрес.	Додана підтримка роумінгу та мультихомінгу через MOBIKE, що дозволяє сеансам працювати при зміні IP-адреси.
Сумісність	Широко впроваджений, але схильний до проблем із сумісністю через відмінності у виробниках.	Створено для сприяння сумісності та більш плавної інтеграції між різними системами та пристроями.
Робота з NAT-пристроями	Потребує ручного налаштування і менш	Автоматично обробляє обхід NAT, спрощуючи

	ефективний при роботі за NAT.	розгортання в середовищі NAT.

РОЗДІЛ 3. ДОСЛІДЖЕННЯ ТА РОЗРОБКА ПРОГРАМНОГО ІНТЕРФЕЙСУ ДЛЯ ПРОТОКОЛУ DIAMETER МОВОЮ JAVA

3.1. Встановлення вимог для створюваного інтерфейсу

Перед початком реалізації інтерфейсу нами було визначено основні критерії та вимоги.

- Інкапсуляція створення та парсингу повідомлень

Реалізація фабрик повідомлень (MessageFactory) для двох типів повідомлень - IKEV2SKRequest і IKEV2SKAnswer, щоб і серверна, і клієнтська частини легко формували та парсили необхідні AVP.

- Забезпечення зручного керування сесіями

Застосовуючи фабрику сесій (SessionFactory) користувачі створеного API повинні мати змогу створювати клієнтські та серверні сесії IKE SK. Інтерфейс має автоматично оброблювати стан сесії та підтримати попередньо-сконфігуровані тайм-аути, не потребуючи додаткового втручання розробника.

- Налаштування мережевого рівня

API мусить мати можливість ініціалізувати та конфігурувати пул робочих потоків (WorkerPool). Також додавати та видаляти мережеві слухачі для прослуховування різних подій в стеку та реєструвати або скасовувати Application-ID динамічно, залежно від потреб застосунку.

- Асинхронна обробка подій

Система колбеків має забезпечувати зворотний виклик при успішній відправці повідомлень або при помилках передачі. API повинно мати можливість підписуватися на події тайм-ауту і помилки щоб розробник міг логувати інциденти або додавати власну логіку відновалення з'єднання.

3.2. Проєктування архітектури програмного інтерфейсу IKEv2

Проєктування архітектури інтерфейсу Diameter IKE SK спираючись на документації rfc6738, було розпочато з аналізу та створення java-інтерфейсів, які описують AVP для цього інтерфейса. Для кожного AVP було створено java-інтерфейс який інкапсулює його структуру та типи його полів. У спецефікації також определілена обов'язкова чи не обов'язкова наявність тих чи інших AVP у групових AVP та самих повідомленнях. Обов'язкова наявність позначена фігурними дужками всередині котрих назва AVP {Назва AVP}, необов'язкова наявність позначена квадратними дужками всередині котрих назва AVP [Назва AVP], позначення зірочка перед квадратними дужками вказує на те що AVP може з'явитись нуль або більше раз *[Назва AVP], позначення *[AVP] дозволяє

додавати будь які додаткові AVP не зазначаючи їх явно, це дозволяє забезпечити гнучкість та розширюваність інтерфейсу та протоколу Diameter для майбутнього або специфічного використання без необхідності вносити зміни в сталу структуру.

IKEv2-Nonces – цей AVP групованого типу, він містить в собі такі AVP:

- {Ni} це AVP містить в собі nonce-значення ініціатора
- {Nr} це AVP містить в собі nonce-значення респондента
- *[AVP]

Цими nonce-занченнями обмінюються між собою IKEv2- peer та IKEv2-сервер під час початкового обміну IKEv2. Nonce - це випадкове або напів-випадкове число яке використовується один раз в цьому випадку nonce використовуються для генерації SK(Shared Key).

```
@DiameterAvpDefinition(code = AvpCodes.IKEV2_NONCES, vendorId = -1L, name = "IKEv2-Nonces"
public interface IKEv2Nonces extends DiameterGroupedAvp
{
    ByteBuf getNi();

    void setNi(ByteBuf value) throws MissingAvpException;

    ByteBuf getNr();

    void setNr(ByteBuf value) throws MissingAvpException;
}
```

Рис 3.1 Створення Java-інтерфейс для IKEv2Nonces AVP

IKEv2-Identity – цей AVP групованого типу і він використовується для передачі ідентифікаційної інформації під час встановлення захищеного з’єднання через IKEv2, він зберігає в собі такі AVP :

- {Initiator-Identity} - цей AVP групованого типу, він містить AVP Identification-Data та AVP ID-Type . AVP Identification-Data AVP містить в

собі ідентифікаційні дані ініціатора, AVP ID-Type містить в собі тип ідентифікатора.

- [Responder-Identity] - цей AVP групованого типу, він містить AVP Identification-Data та AVP ID-Type. AVP Identification-Data AVP містить в собі дані респондента а AVP ID-Type містить в собі тип ідентифікатора.
- *[AVP]

```
@DiameterAvpDefinition(code = AvpCodes.IKEV2_IDENTITY, vendorId = -1L, name = "IKEv2-Identity")
public interface IKEv2Identity extends DiameterGroupedAvp
{
    InitiatorIdentity getInitiatorIdentity();

    void setInitiatorIdentity(InitiatorIdentity value) throws MissingAvpException;

    ResponderIdentity getResponderIdentity();

    void setResponderIdentity(ResponderIdentity value);
}
```

Рис 3.2 Створення Java-інтерфейс для IKEv2Identity AVP

Наступним кроком стала реалізація java-інтерфейсів для повідомлень щоб забезпечити чітку та зрозумілу структуру обімну інформацією між клієнтською та серверною стороною. Інтерфейс для запиту визначає у дані які клієнт надсилає серверу. В спецефікаї визначені які AVP можуть бути пристунім в IKEv2-SK-Request повідомленні.

Повідомлення IKEv2-SK-Request, яке вказано в специфікації з кодом команди 329 і встановленим у полі «Прапори команди» бітом 'R', який вказує на те що це повідомлення є запитом та надсилається з IKEv2-сервера до домашнього сервера автентифікації, авторизації та обліку(HAAA,Home Authentication, Authorization, and Accounting server) , щоб ініціювати процес встановлення безпечного з’єднання через протокол IKEv2 з авторизацією на основі спільного ключа (Shared Key) SK. У цьому повідомлені в полі Application-Id у заголовку повідомлення Diameter має бути встановлено значення Diameter IKE SK Application-Id яке

дорівнює одинадцять і потрібно для розуміння до якого інтерфейсу належить повідомлення.

Повідомлення IKEv2-SK-Request повинні включати в собі IKEv2 Nonces AVP які в свій час містять в собі AVP Ni та Nr, їх nonce-значення будуть обміняні між сторонами під час початкового обміну IKEv2. Повідомлення IKEv2-SK-Request також може містити у собі Key-SPI AVP. Цей AVP не обов'язковий але якщо він включений до складу повідомлення IKEv2-SK-Request , то він містить у собі унікальний ідентифікатор Security Parameter Index(SPI), який HAAA повинен використовувати разом з іншими параметрами для визначення відповідного спільного ключа (SK).

IKEv2-SK-Request повідомлення повинно включати в собі IKEv2-Identity AVP, в цьому AVP повинен бути присутній Initiator-Identity AVP, в AVP Initiator-Identity обов'язково має міститись інформація про того хто ініціював з'єднання, якщо в процессі обміну буде отримана інформація про респондента, то вона також додається до повідомлення. Це потрібно для того щоб сервер знав з ким він намагається встановити захищене з'єднання.

Стркутура IKEv2-SK-Request повідомлення:

- < Session-Id > - Session-Id це унікальний ідентифікатор сессії, за допомогою якого між клієнтом та сервером пов'язуються запит та відповідь.
- { Auth-Application-Id } - Auth-Application-Id це ідентифікатор застосунку Diameter, за допомогоб нього опріділяється до якого інтерфейсу відноситься повідомлення.
- { Origin-Host } - Origin-Host містить в собі значення хоста що відправляє повідомлення.
- { Origin-Realm }- Origin-Realm містить в собі значення домен з якого було відправлено повідомлення.

- { Destination-Realm }- Destination-Realm містить в собі домен отримувача повідомлення.
- { Auth-Request-Type } - Auth-Request-Type містить в собі тип до якого відноситься запит, авторизація, автентифікація, або авторизація та автентифікація
- [Destination-Host] - Destination-Host містить в собі значення хоста що отримує повідомлення.
- [NAS-Identifier] - NAS-Identifier містить в собі ідентифікатор серверу мережевого доступу.
- [NAS-IP-Address] - NAS-IP-Address містить в собі IPv4-адресу пристрою якого надійшло повідомлення.
- [NAS-IPv6-Address] - NAS-IPv6-Address містить в собі IPv6-адресу пристрою якого надійшло повідомлення.
- [NAS-Port] - NAS-Port містить значення фізичного або віртуального порту серверу мережевого доступа до якого підключився користувач.
- [Origin-State-Id] - Origin-State-Id містить в собі значення поточного стану вузла, це значення збільшується кожного разу коли пристрій перезапускається і втрачає свій попередній стан.
- [User-Name] - User-Name містить в собі значення імені користувача
- [Key-SPI] - Key-SPI містить в собі індентифікатор парам
- { IKEv2-Identity } -IKEv2-Identity містить в собі AVP Identification-Data та AVP ID-Type.
- [Auth-Session-State] - Auth-Session-State містить в собі значення чи має сервер зберігати стан сессії чи ні
- { IKEv2-Nonces } – IKEv2-Nonces містить в собі nonce значення ініціатора та респондента.

- *[Proxy-Info] - Proxy-Info містить в собі інформацію про кожен проксі-вузол який пройшло повідомлення .
- * [Route-Record] - Route-Record містить в собі інформацію про кожен вузол який пройшло повідомлення.
- * [AVP]

В коді для отримання доступу до приватних даних використовуються так звані “гетери” та “сетери” це спеціальні методи функціонал котрих цілком походить від їх назви “гетер” від слова get - отримувати та “сетер” від слова set – встановлювати. За допомогою цих методів реалізується принцип інкапсуляції, адже при такому використані відсутній прямий доступ до значень, що забезпечує більший рівень безпеки і унеможливлює змінення цих значень без використання зазначених методів.

```
@DiameterCommandDefinition(applicationId = ApplicationIDs.IKESK, commandCode = CommandCodes.IKESK, request = true, proxyable = true, name="IKEv2-SK-Request")
public interface IKEV2SKRequest extends AuthenticationRequest
{
    public AuthRequestTypeEnum getAuthRequestType();

    void setAuthRequestType(AuthRequestTypeEnum value) throws MissingAvpException;

    String getNASIdentifier();

    void setNASIdentifier(String value);

    InetAddress getNASIPAddress();

    void setNASIPAddress(Inet4Address value);

    InetAddress getNASIPv6Address();

    void setNASIPv6Address(Inet6Address value);

    Long getNASPort();

    void setNASPort(Long value);

    Long getKeySPI();

    void setKeySPI(Long value);

    public IKEv2Identity getIKEv2Identity();

    void setIKEv2Identity(IKEv2Identity value) throws MissingAvpException;

    public AuthSessionStateEnum getAuthSessionState();

    public void setAuthSessionState(AuthSessionStateEnum value) throws MissingAvpException;

    public IKEv2Nonces getIKEv2Nonces();

    void setIKEv2Nonces(IKEv2Nonces value) throws MissingAvpException;

}
```

Рис. 3.3 Створення java-інтерфейсу для запита IKEv2-SK-Request

Також в коді реалізований виключення такі як MissingAvpExeption цей виняток спрацьовує в разі якщо обов’язкове AVP відсутнє, тоді при спробі відправити таке повідомлення відбудеться помилка з винятком, яка повідомить

про неможливість відправлення повідомлення без присутності всіх обов'язкових AVP.

Повідомлення IKEv2-SK-Answer, яке вказано в специфікації з кодом команди 329 та очищеним бітом 'R' у полі «Прапори команди», надсилається з HAAA на сервер IKEv2 у відповідь на запит IKEv2-SK-Request. У цьому повідомленні також як і у повідомленні IKEv2-SK-Request у полі Application-Id заголовка Diameter має бути встановлене значення Diameter IKE SK Application-Id яке дорівнює одинадцять і потрібно для розуміння до якого інтерфейсу належить повідомлення. Якщо процедура авторизації пройшла успішно, то повідомлення-відповідь IKEv2-SK-Answer має містити Key AVP в якому значення Key-Type AVP має бути встановлене IKEv2 SK яке дорівнює трьом і потрібно для розуміння типу отриманого ключа. А також Keying-Material AVP має містити в собі SK. У випадку якщо Key-SPI AVP було отримано в IKEv2-SK-запиті, Key-SPI AVP потрібно додати до складу Key AVP. До Key AVP також може бути доданий Key-Lifetime AVP якщо він є в составі Key AVP то ключ переданий у відповіді не може використовуватися отримувачем відповіді, якщо термін дії цього ключа на момент отримання закінчився. Після додавання Key AVP у повному його составі додається Responder-Identity AVP. Структура повідомлення-відповіді IKEv2-SK-Answer має схожість з повідомлення-запитом IKEv2-SK-Request, але має свої суттєві відмінності. В цій структурі будуть описані тільки ті AVP яке не були описані в IKEv2-SK-Request.

Структура IKEv2-SK-Answer:

- < Session-Id >
- { Auth-Application-Id }
- { Auth-Request-Type }
- { Result-Code } - Result-Code містить в собі значення результату обробки запита.

- { Origin-Host }
- { Origin-Realm }
- [User-Name]
- [Key] – Key містить в собі значення типу SK, значення самого SK а також додаткові AVP.
- [Responder-Identity] - Responder-Identity містить в собі AVP Identification-Data та AVP ID-Type.
- [Auth-Session-State]
- [Error-Message] - Error-Message містить в собі опис помилки якщо така сталась в процессі обробки запита.
- [Error-Reporting-Host] - Error-Reporting-Host містить в собі хоста що відправляє.
- * [Failed-AVP] - Failed-AVP містить в собі AVP які спричинили помилку.
- [Origin-State-Id]
- * [Redirect-Host] - Redirect-Host містить в собі хост до якого повідомлення буде переадресовано.
- [Redirect-Host-Usage] – Redirect-Host-Usage містить в собі значення того як використовувати значення з Redirect-Host.
- [Redirect-Max-Cache-Time] - Redirect-Max-Cache-Time містить в собі максимальне значення часу, скільки може зберігатись інформація про переадресування.
- * [Proxy-Info]
- * [Route-Record]
- * [AVP]

```java
/* 5.2.  IKEv2-SK-Answer (IKESKA) Command
@DiameterCommandDefinition(applicationId = ApplicationIDs.IKESK, commandCode = CommandCodes.IKESK, request = false, proxyable = true, name="IKEv2-SK-Answer")
public interface IKEV2SKAnswer extends AuthenticationAnswer
{
    public AuthRequestTypeEnum getAuthRequestType();

    void setAuthRequestType(AuthRequestTypeEnum value) throws MissingAvpException;

    public Key getKey();

    void setKey(Key value);

    public ResponderIdentity getResponderIdentity();

    void setResponderIdentity(ResponderIdentity value);

    public AuthSessionStateEnum getAuthSessionState();

    public void setAuthSessionState(AuthSessionStateEnum value);

    public List<String> getRouteRecords();

    public void setRouteRecords(List<String> value);
}
```

Рис. 3.4 Створення java-інтерфейсу для відповіді IKEv2-SK-Answer

Наступним кроком в розробці java-інтерфейсів для їх подальшої реалізації стала реалізація класів для сесійного менеджменту, він складається з таких класів як AVP Factory, Message Factory, Session Factory, Client Listener, Server Listener а також IKE SK Client Session, та IKE SK Server Session.

AVP Factory це java-інтерфейс який відповідає за централізоване створення групових AVP інтерфейса IKE SK.

AvpFactory.java

```java
package com.mobius.software.telco.protocols.diameter.app.ikesk;
/*
 * Mobius Software LTD

import com.mobius.software.telco.protocols.diameter.exceptions.MissingAvpException;


public interface AvpFactory extends com.mobius.software.telco.protocols.diameter.app.commons.AvpFactory
{
    public Key getKey(KeyTypeEnum keyType, ByteBuf keyingMaterial) throws MissingAvpException;

    public ResponderIdentity getResponderIdentity(IDTypeEnum idType, ByteBuf identificationData) throws MissingAvpException;

    public InitiatorIdentity getInitiatorIdentity(IDTypeEnum idType, ByteBuf identificationData) throws MissingAvpException;

    public IKEv2Identity getIKEv2Identity(InitiatorIdentity InitiatorIdentity) throws MissingAvpException;

    public IKEv2Nonces getIKEv2Nonces(ByteBuf ni, ByteBuf nr)throws MissingAvpException;
}
```

Рис. 3.5 Створення java-інтерфейсу для AVP Factory

Message Factory це java-інтерфейс який відповідає за централізоване створення повідомлень інтерфейса IKE SK.

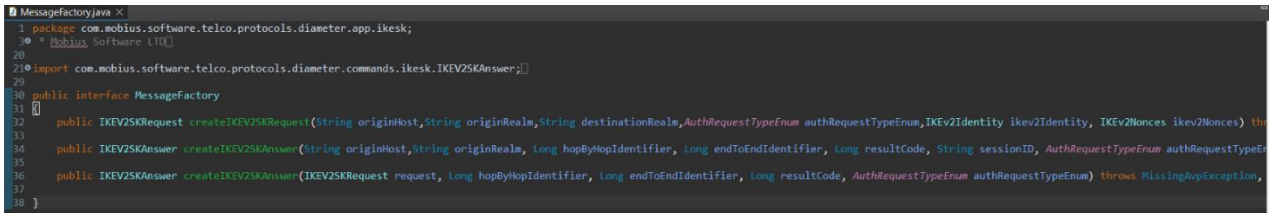


Рис. 3.6 Створення java-інтерфейсу для Message Factory

Session Factory це java-інтерфейс який відповідає за створення екземплярів ClientSessions та ServerSessions на основі повідомлення-запита інтерфейса IKE SK.

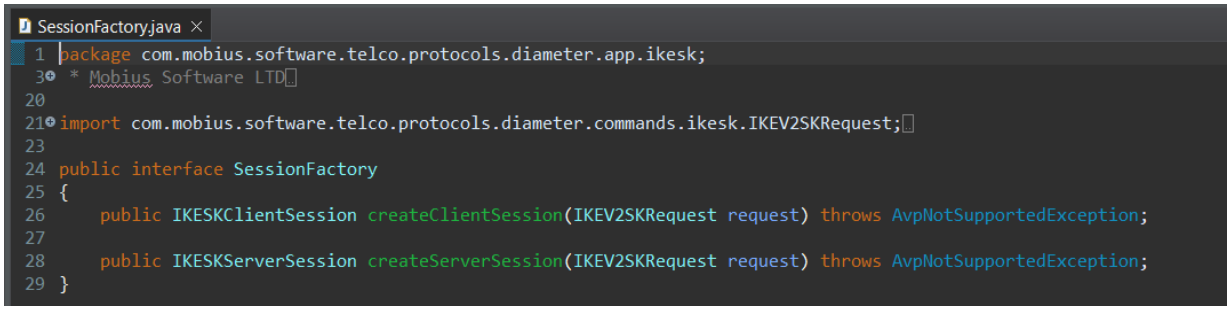


Рис. 3.7 Створення java-інтерфейсу для Session Factory

У Diameter є дві основні ролі: він може працювати як клієнт так і як сервер. Ці ролі зосереджені на обміні повідомленнями для конкретних додатків Diameter. Кожен додаток Diameter визначає, як ролі клієнта і сервера взаємодіють на рівні додатку.

Інтерфейси ClientSession і ServerSession є важливими для управління цими ролями. Кожен сеанс керує життєвим циклом взаємодії запиту-відповіді між клієнтом і сервером. Ці сеанси обробляють внутрішню логіку, необхідну для надсилання та отримання повідомлень а також керування станом сеансу протягом її життя. Для того щоб ClientSession та ServerSession могли виконувати свої ролі їм потрібно отримувати повідомлення з мережі для їх подальшої обробки, цю

функцію реалізують такі інтерфейси як ClientListener та ServerListener вони отримують запити з мережі та обробляють їх у контексті своїх сеансів.

Інтерфейс ClientSession представляє логіку на стороні клієнта, який ініціює взаємодію використовуючи SessionFactory для управління сесією, він надсилає запит до сервера. Зазвичай він очікує на відповідь від сервера і обробляє її, як тільки вона надходить.

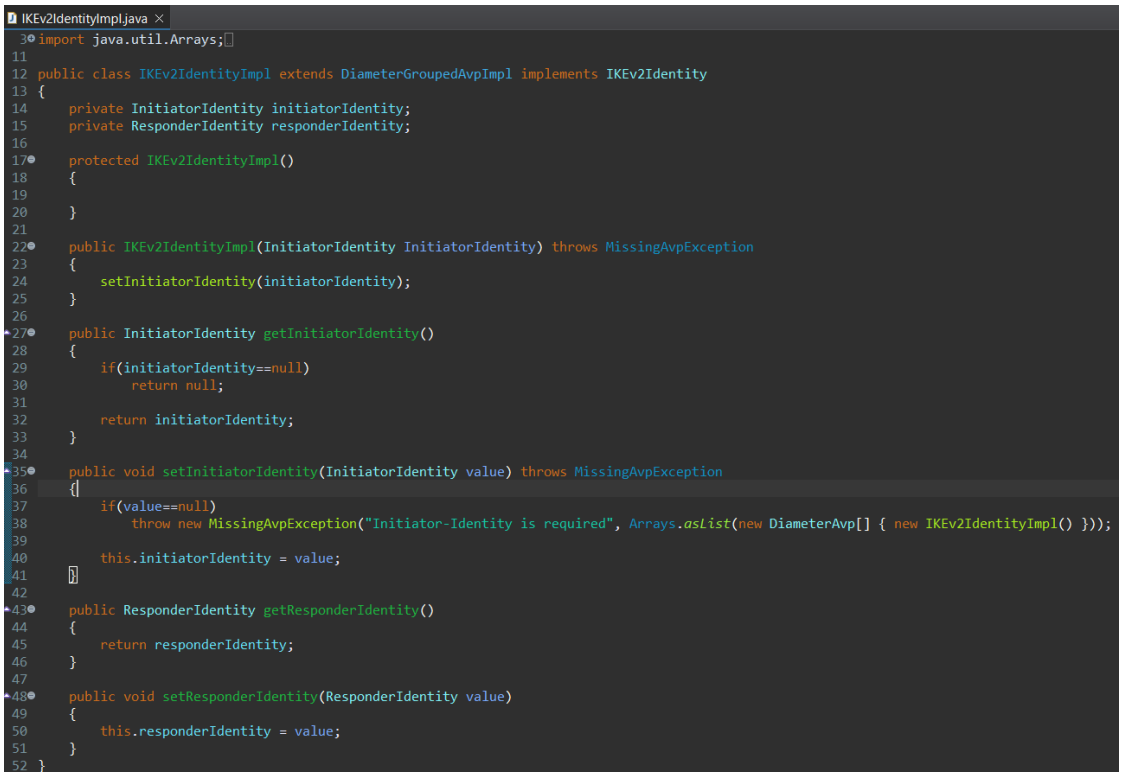
Інтерфейс ServerSession представляє логіку на стороні сервера, де сервер, використовуючи SessionFactory, створює сеанс сервера для управління взаємодією та відповідає на вхідні запити, ініційовані клієнтом. Сервер отримує запити, обробляє їх і надсилає відповідні відповіді.

3.3. Реалізація інтерфейсу IKE SK

Розробивши структуру та описавши всі методи в java-інтерфейсах можна почати реалізовувати їх функціонал, для цього ми імплементуємо раніше створені інтерфейси в нових java-класах.

Реалізація AVP згрупованого типу IKEv2IdentityImpl представляє собою наслідування класу DiameterGroupedAvpImpl та імплементацію java-інтерфейсу IKEv2Identity.

Також в класі IKEv2IdentityImpl є два конструктора один конструктор є конструктором без праметрів, він потрібен для правильної обробки AVP в середині Diameter стаку, а другий конструктор має параметр Initiator-Identitity, без цього параметра не можливо створити новий екземпляр класу так як він є обов’язковим. Також в цьому класі є “гетери” та “сетери” для доступу до його приватних полів.



```java
import java.util.Arrays;

public class IKEv2IdentityImpl extends DiameterGroupedAvpImpl implements IKEv2Identity
{
    private InitiatorIdentity initiatorIdentity;
    private ResponderIdentity responderIdentity;

    protected IKEv2IdentityImpl()
    {

    }

    public IKEv2IdentityImpl(InitiatorIdentity InitiatorIdentity) throws MissingAvpException
    {
        setInitiatorIdentity(initiatorIdentity);
    }

    public InitiatorIdentity getInitiatorIdentity()
    {
        if(initiatorIdentity==null)
            return null;

        return initiatorIdentity;
    }

    public void setInitiatorIdentity(InitiatorIdentity value) throws MissingAvpException
    {
        if(value==null)
            throw new MissingAvpException("Initiator-Identity is required", Arrays.asList(new DiameterAvp[] { new IKEv2IdentityImpl() }));

        this.initiatorIdentity = value;
    }

    public ResponderIdentity getResponderIdentity()
    {
        return responderIdentity;
    }

    public void setResponderIdentity(ResponderIdentity value)
    {
        this.responderIdentity = value;
    }
}
```

Рис. 3.8 Створення java-класу IKEv2IdentityImpl

Решта AVP інтерфейсу Diameter IKE SK реалізована по такому самому принципу що і IKEv2IdentityImpl. Тому опускаючи детальний огляд кожного з них преходимо до реалізації повідомлень.

Повідомлення IKEV2SKRequestImpl реалізовано за допомогою наслідування класу AuthenticationRequestImpl та імплементації інтерфейсу IKEV2SKRequest у цього класа є два конструктора один конструктор з параметрами, інший порожній. Основними методи цього класу це “ гетери” та “сетери” для доступу до його приватних полів, а також методи validate() та getOrderedAVPs(). Метод validate() перевіряє наявність в повідомленні обов’язкових AVP а метод getOrderedAVPs() повертає впорядкований список AVP для коректної оброки їх в середині Diameter стаку

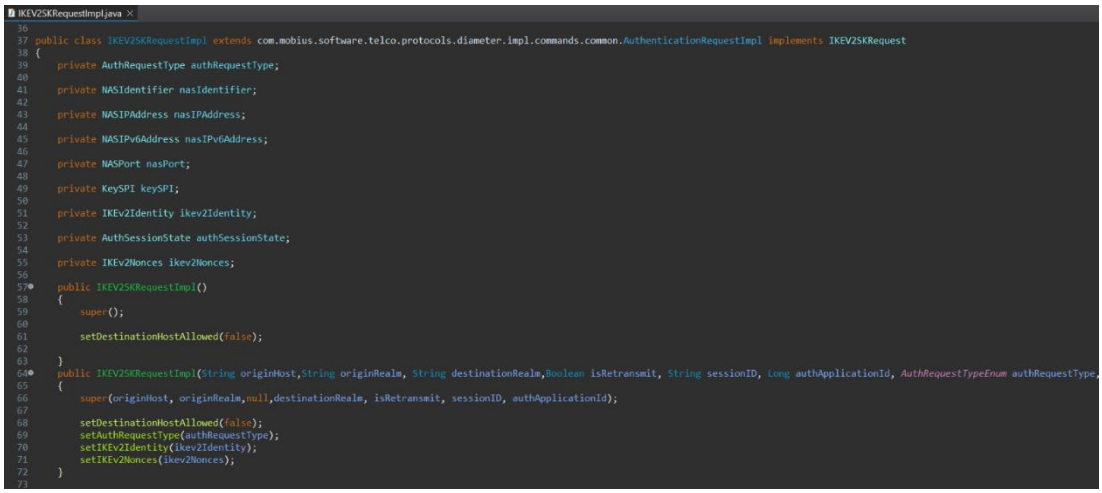


Рис. 3.9 Створення java-класу IKEV2SKRequestImpl

Реалізація класу повідомлення IKEV2SKAnswerImpl інтерфейсу Diameter IKE SK реалізована по такому самому принципу що і IKEV2SKRequestImpl. Тому опускаючи огляд класу IKEV2SKAnswerImpl переходимо до реалізації сесійного менеджменту.

Основним нововведенням в реалізації сесійного менеджменту в порівнянні з його реалізацією його архітектури у вигляді java-інтерфейсу, став клас IKESKProviderImpl, ProviderImpl для кожної програми слугує інтерфейсом між основним стеком Diameter та логікою конкретної програми. За допомогою Провайдерів виконуються такі завдання, як маршрутизація повідомлень Diameter, ініціювання сеансів та забезпечення правильної роботи машини станів.

Специфічні для додатків провайдери розширюють загальний базовий клас DiameterProviderImpl, який визначає спільну поведінку, таку як можливість керування слухачами, відновлення сеансів та обробку помилок.

Така конструкція забезпечує узгодженість між різними додатками, водночас надаючи гнучкість для реалізації специфічної поведінки для кожного додатка.

Клас IKESKProviderImpl має можливість використовувати такі класи AVP Factory, Message Factory а також Session Factory саме за допомогою цих класів він

має можливість керувати створенням та обробкою Diameter повідомлень та сессій.

IKESKProviderImpl.java

```
1  package com.mobius.software.telco.protocols.diameter.impl.app.ikesk;
3  * Mobius Software LTD
20
21 import org.apache.logging.log4j.LogManager;
35
36 public class IKESKProviderImpl extends DiameterProviderImpl<ClientListener, ServerListener, AvpFactory, MessageFactory, SessionFactory>
37 {
38     public static Logger logger=LogManager.getLogger(IKESKProviderImpl.class);
39
40     public IKESKProviderImpl(DiameterStack stack,String packageName)
41     {
42         super(stack, new AvpFactoryImpl(), new MessageFactoryImpl(stack), packageName);
43         setSessionFactory(new SessionFactoryImpl(this));
44     }
45
46     @Override
47     public DiameterSession getNewSession(DiameterRequest message)
48     {
49         try
50         {
51             if(message instanceof IKEV2SKRequest)
52                 return new IKESKServerSessionImpl(message.getSessionId(), message.getOriginHost(), message.getOriginRealm(), this);
53         }
54         catch(DiameterException ex)
55         {
56             logger.warn("An error occured while creating new session," + ex.getMessage(),ex);
57         }
58
59         return null;
60     }
61 }
```

Рис. 3.10 Створення java-класу IKESKProvider

РОЗДІЛ 4. ОЦІНКА ЕФЕКТИВНОСТІ СТВОРЕНОГО ІНТЕРФЕЙСУ

4.1. Створення JUNIT тестів для оцінки інтерфейсу

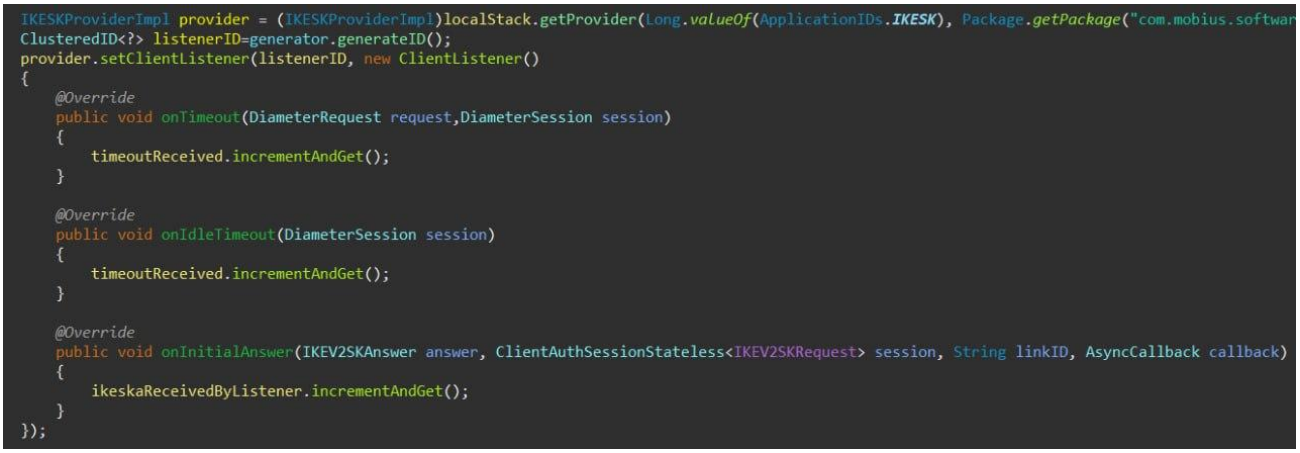
Юніт-тестування є ключовим етапом у процесі розробки та впровадження програмного забезпечення. Воно не лише підвищує якість і стабільність коду, а й сприяє своєчасному виявленню помилок, що значно знижує ризик виникнення регресій під час подальшої підтримки та розвитку продукту. Завдяки юніт-тестам розробники можуть упевнено вносити зміни, знаючи, що основна функціональність залишається непошкодженою.

Основна мета юніт-тестування полягає у перевірці, що нові або змінені компоненти не порушують роботу вже існуючих модулів. Воно дозволяє виявляти дефекти на ранніх етапах розробки, що економить час і ресурси. Крім того, юніт-тести сприяють підтримці високих стандартів якості коду, встановлених у проекті або організації.

У рамках нашої роботи було розроблено базовий клас на мові Java — NetworkTestBase, який використовується для конфігурації, запуску та завершення роботи мережевих стеків протоколу Diameter: серверного та локального (див. рис. 4.1). Цей клас також управляє пулом робочих потоків (WorkerPool) та встановлює тайм-аути для різних мережевих подій, таких як бездіяльність, очікування відповіді і перепідключення.

Об’єкти DiameterStack serverStack і localStack представляють два окремі мережеві стеки Diameter, де локальний стек виступає в ролі клієнта, що ініціює запити і очікує відповіді, а серверний — приймає ці запити і формує відповіді. Таким чином реалізується двонаправлена комунікація між клієнтом і сервером.

Для обох стеків нами було додано мережевих слухачів (network listeners). Мережевий слухач — це спеціальний механізм, що постійно прослуховує певний мережевий порт на предмет вхідних з’єднань та повідомлень. Він реагує на отримані пакети Diameter, виконуючи запрограмовані дії відповідно до типу повідомлення (рис. 4.1).



```
IKESKProviderImpl provider = (IKESKProviderImpl)localStack.getProvider(Long.valueOf(ApplicationIDs.IKESK), Package.getPackage("com.mobius.softwar
ClusteredID<?> listenerID=generator.generateID();
provider.setClientListener(listenerID, new ClientListener()
{
    @Override
    public void onTimeout(DiameterRequest request,DiameterSession session)
    {
        timeoutReceived.incrementAndGet();
    }

    @Override
    public void onIdleTimeout(DiameterSession session)
    {
        timeoutReceived.incrementAndGet();
    }

    @Override
    public void onInitialAnswer(IKEV2SKAnswer answer, ClientAuthSessionStateless<IKEV2SKRequest> session, String linkID, AsyncCallback callback)
    {
        ikeskaReceivedByListener.incrementAndGet();
    }
});
```

Рис. 4.1 Створення ClientListener, який при отриманні answer збільшує лічільник на 1

Як показано на рисунку 4.2, ми відстежуємо кількість отриманих повідомлень певних типів як у локальному стеці (IKEV2SKAnswer), так і на сервері (IKEV2SKRequest). Повідомлення типу IKEV2SKRequest — це спеціальний запит у протоколі IKEv2, призначений для створення або оновлення захищеного каналу (Security Association). Воно сигналізує про намір ініціатора (клієнта чи сервера) почати або підтримувати безпечний зв’язок. Відповідно, IKEV2SKAnswer — це відповідь на цей запит, яка повідомляє про прийняття або відхилення ініціативи та містить інформацію про параметри захищеного зв’язку.

У випадку, якщо сервер не отримує запити, лічильник ikeskaReceived не збільшуватиметься, що може свідчити про проблеми у мережі або помилки в програмному коді. Аналогічно, якщо клієнт не отримує відповіді, значення

лічильника ikeskrReceived залишатиметься незмінним — це також сигналізує про неполадки. На рисунку 4.3 наведено початкові значення цих лічильників перед початком тестування, що дає змогу оцінити правильність їхньої роботи в процесі.

```
final AtomicLong ikeskaReceived=new AtomicLong(0L);
final AtomicLong ikeskaReceivedByListener=new AtomicLong(0L);
final AtomicLong ikeskrReceived=new AtomicLong(0L);
final AtomicLong ikeskrReceivedByListener=new AtomicLong(0L);
final AtomicLong timeoutReceived=new AtomicLong(0L);
```

Рис. 4.2 Лічильники на початку тестування

І також код в кінці на рис .4.4 містить перевірки, які використовуються для підтвердження коректної роботи тестованої системи. Кожен виклик assertEquals порівнює фактичну кількість отриманих повідомлень або подій з очікуваним значенням. Зокрема, перевіряється, що сервер отримав рівно один запит типу IKEV2SKRequest, а відповідний мережевий слухач також обробив саме один такий запит. Аналогічно, клієнт отримав одну відповідь типу IKEV2SKAnswer, і його слухач обробив саме одну таку відповідь. Крім того, підтверджується, що під час тестування не виникло жодних тайм-аутів. Таким чином, ці перевірки гарантують, що обмін повідомленнями відбувся правильно, і всі основні події пройшли без помилок або затримок.

```
    assertEquals(ikeskaReceived.get() , 1L);
    assertEquals(ikeskaReceivedByListener.get() , 1L);
    assertEquals(ikeskrReceived.get() , 1L);
    assertEquals(ikeskrReceivedByListener.get() , 1L);
    assertEquals(timeoutReceived.get() , 0L);
}
```

Рис. 4.3 Лічильники в кінці тестування

4.2. Результати JUNIT тестування

Для того щоб упевнитися в результатах проведених тестів під час проведення тесту було використано застосунок WireShark для перехоплення пакетів мережевого трафіку.

Wireshark - це аналізатор мережевих пакетів. Аналізатор мережевих пакетів представляє перехоплені дані пакетів в максимально детальному вигляді.

Він може допомогти у перехопленні мережевих пакетів і відображенні їх на детальному рівні. Після того, як ці пакети розбиті, з'являється можливість використовувати їх для аналізу в реальному часі або в автономному режимі.
Цей інструмент дозволяє розглядати мережевий трафік під мікроскопом, а потім фільтрувати і деталізувати його, збільшуючи масштаб до першопричини проблем, допомагаючи в аналізі мережі і, в кінцевому підсумку, в забезпеченні мережевої безпеки.

Після того як Junit тестування завершилось вдало перехоплення пакетів мережевого трафіку було зупинено та записано у файл для детального та зручного аналізу його вмісту. За допомогою внутрішнього функціоналу застосунку WireShark мереживий трафік було відфільтровано так щоб відобразись тільки перехоплені мережеві пакети протоколу Diameter.

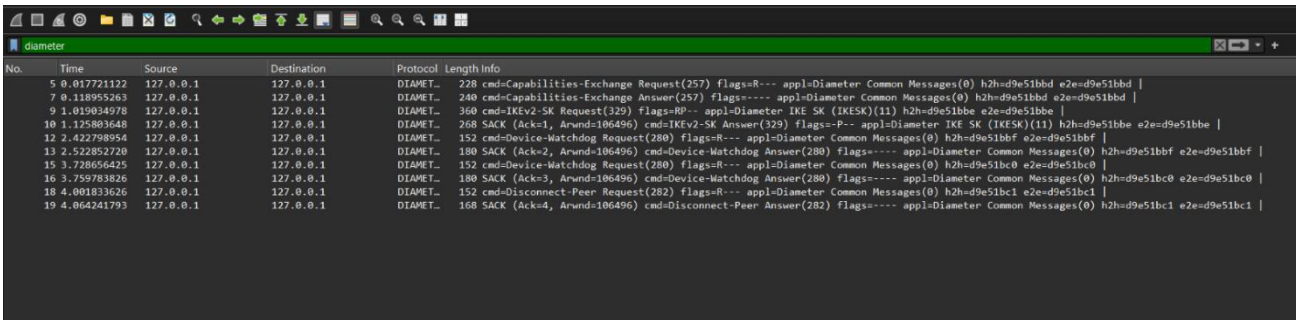


Рис. 4.4 Відфільтрований перехоплений мережевий трафік

Під час розгляду перехоплених пакетів протоколу Diameter можна побачити що с початку відбувся обмін можливостей між клієнтом та сервером, за допомогою відправлення запита Capabilities-Exchange-Request з боку клієнта та отриманні відповіді Capabilities-Exchange-Answer з боку сервера. Після вдалого обміну можливостями клієнт та сервер переходять до обміну основними повідомлення. Як можно побачити з перехопленого мережевого трафіку, клієнт відправляє IKEv2-SK-Request зі всіма AVP які були додані в тесті під час створення запита.

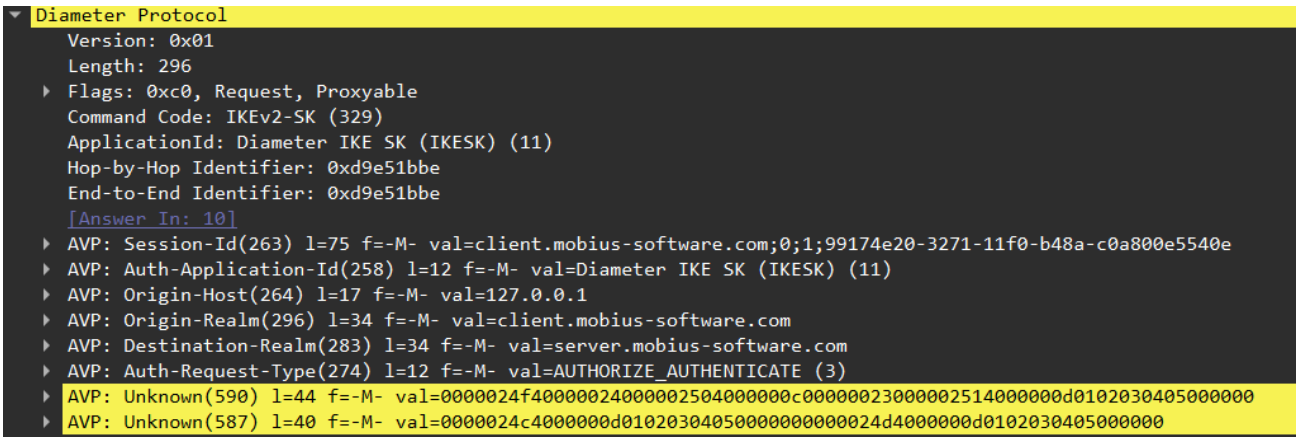


Рис. 4.5 Детальний розгляд пакету IKEv2-SK-Request

У відповідь на свій запит клієнт від сервера отримує повідомлення-відповідь IKEv2-SK-Answer роздивляючись детальніше пакет повідомлення-

відповіді можна побачити що відповідь прийшла зі значенням AVP ReusltCode DIAMETER_SUCCESS, що сповіщає клієнта про те що обробка запиту пройшла вдало.

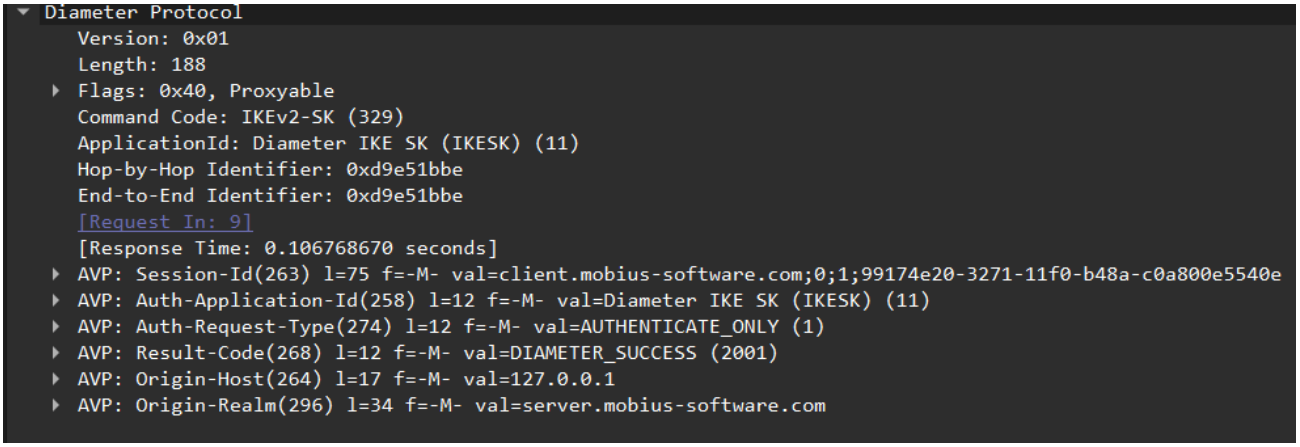


Рис. 4.6 Детальний розгляд пакету IKEv2-SK-Answer

Після того як відповідь була отримана, передача основних повідомлень між клієнтом та сервером закінчилась, але сесія все ще активна, про те що сесія ще активна можна дізнатись подивившись на пакети повідомлень Device-Watchdog-Request та Device-Watchdog-Answer, клієнт надсилає повідомлення-запит Device-Watchdog-Request для того щоб дізнатись чи активне ще з’єднання, якщо у відповідь він отримує повідомлення-відповідь Device-Watchdog-Answer то з’єднання вважається активним і очікує на продовження передачі інформації, а якщо відповідь не приходить то сесія вважається не активною.

Якщо після отримання від сервера повідомлення-відповіді Device-Watchdog-Answer обмін інформації не продовжується сесія закінчуєтьсяі через визначений час бездіяльності між клієнтом і сервером, для закінчення сесії клієнт відправляє повідомлення-запит Disconnect-Peer-Request на що очікує від сервера повідомлення відповідь Disconnect-Peer-Answer з результатом обробки повідомлення запита. Як можна побачити в прехопленому пакеті, що

повідомлення Disconnect-Peer-Answer прийшло зі значенням AVP ReusltCode DIAMETER_SUCCESS що сповіщає клієнта о том що сессія завершилась вдало.

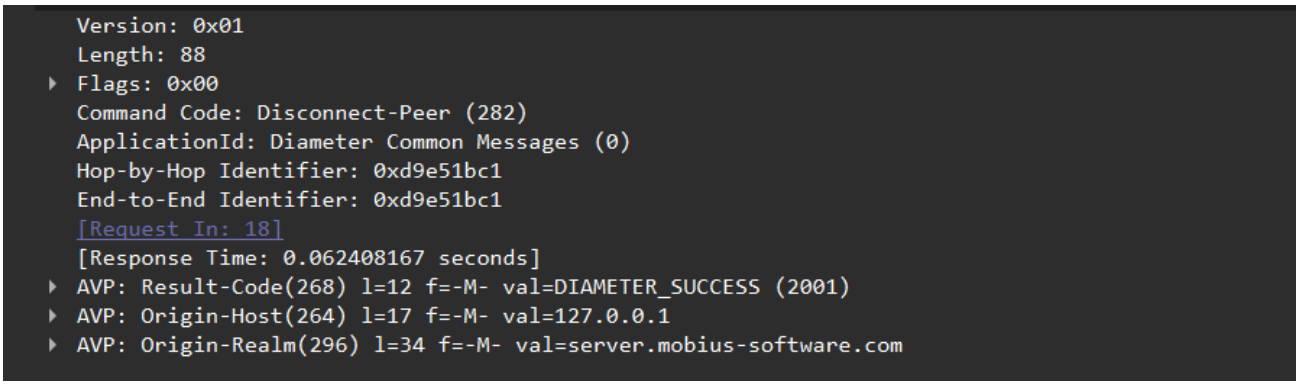


Рис. 4.7 Детальний розгляд пакету Disconnect-Peer-Answer

ВИСНОВКИ

Підбиваючи підсумки пророблені роботи нами було оброблено та вивчено специфікацію, області застосування та принципи роботи протоколу Diameter. У якості розроблюваного інтерфейсу нами було обрано інтерфейс IKEv2, що застосовується для встановлення захищеного каналу зв’язку між двома мережами.

В процесі дослідження ми реалізували повноцінний Java-інтерфейс для роботи з протоколом Diameter. Розроблено каркас, який автоматично налаштовує, стартує і зупиняє як серверний, так і клієнтський стеки, а також керує пулом потоків та мережевими слухачами.

Для спрощення роботи з повідомленнями реалізовано фабрики (MessageFactory), що ховають усі деталі формування і розбору IKEV2SKRequest та IKEV2SKAnswer, тож розробнику не потрібно маніпулювати байтовими AVP-структурами. За ініціалізацію сесій тепер відповідає SessionFactory: вона сама переходить між станами і застосовує задані тайм-аути, без зайвого коду з нашого боку.

Щоб легко реагувати на результати обміну, помилки чи тайм-аути, впроваджено асинхронні колбеки (AsyncCallback), які дозволяють відразу запускати процедури відновлення з’єднання або обробки помилок.

Автоматизовані юніт-, інтеграційні й навантажувальні тести довели: обмін повідомленнями працює правильно, регресій немає, а система зберігає високу продуктивність навіть під великим навантаженням. Отже, отриманий API забезпечує надійну і масштабовану платформу для подальшого розвитку AAA-сервісів на базі Diameter у Java-середовищі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Importance of AAA in Network Security | Blog | Adroit Information Technology Academy (AITA). *Best Cloud Computing | Cisco CCNA | CCNP | SD-WAN | Nexus 9K (ACI) | Firewall | Python Training in Kolkata*. URL: https://www.adroitacademy.com/blog/Importance-of-AAA-in-Network-Security.
2. Global P. RADIUS vs. TACACS+: Which AAA Protocol Should You Choose?. *Find the Resources You Need from PivIT Global*. URL: https://info.pivitglobal.com/resources/radius-vs.-tacacs-aaa-protocol.
3. Diameter Overview. *Real Time Communication*. URL: https://realtimecommunication.wordpress.com/2016/08/30/diameter-overview/.
4. The IMS: IP Multimedia Concepts And Services, Second Edition. *O'Reilly Online Learning*. URL: https://www.oreilly.com/library/view/the-ims-ip/9780470019061/9780470019061_message_structure.html.
5. Configuration Guide. *Moved*. URL: https://docs.oracle.com/cd/E95618_01/html/sbc_scz810_acliconfiguration/GUID-01275627-CC9A-4880-9D73-F7D708FA4378.htm.
6. TCP 3-Way Handshake (SYN, SYN-ACK,ACK). *Guru99*. URL: https://www.guru99.com/uk/tcp-3-way-handshake.html.
7. https://www.researchgate.net/figure/SCTP-four-way handshake_fig4_229047183
8. IMS Diameter Stack - Softil. *Softil*. URL: https://www.softil.com/solutions/protocol-stacks-frameworks/ims-diameter-stack/.

9. Ribbon Communications. *IP Optical Networking and Communications | Ribbon*. URL: https://ribboncommunications.com/company/get-help/glossary/ip-multimedia-subsystem-ims.
10. Definition of the Gx Interface. URL: https://info.support.huawei.com/hedex/api/pages/EDOC1100277644/AEM10221/04/resources/ne/dc_ne_gx_0007.html.
11. Shivanshu. What is IPsec (Internet Protocol Security)?. *Intellipaat*. URL: https://intellipaat.com/blog/ipsec-internet-security-protocol/.
12. IPsec Tunnel Mode vs. Transport Mode | Twingate. *Twingate: It's time to ditch your VPN*. URL: https://www.twingate.com/blog/ipsec-tunnel-mode.
13. IoT Solution and Telecommunication Service Provider | Mobius Software. *IoT Solution and Telecommunication Service Provider | Mobius Software*. URL: https://www.mobius-software.com/.
14. GitHub - mobius-software-ltd/corsac-diameter: Mobius Diameter. *GitHub*. URL: https://github.com/mobius-software-ltd/corsac-diameter?tab=readme-ov-file.
15. RFC 6733: Diameter Base Protocol. *» RFC Editor*. URL: https://www.rfc-editor.org/rfc/rfc6733.html.
16. RFC 6738: Diameter IKEv2 SK: Using Shared Keys to Support Interaction between IKEv2 Servers and Diameter Servers. *» RFC Editor*. URL: https://www.rfc-editor.org/rfc/rfc6738.html.
17. RFC 5996: Internet Key Exchange Protocol Version 2 (IKEv2). *» RFC Editor*. URL: https://www.rfc-editor.org/rfc/rfc5996.
18. Authentication, Authorization, and Accounting (AAA) Parameters. *Internet Assigned Numbers Authority*. URL: https://www.iana.org/assignments/aaa-parameters/aaa-parameters.xhtml.

19. Internet Key Exchange (IKE) Attributes. *Internet Assigned Numbers Authority*. URL: https://www.iana.org/assignments/ipsec-registry/ipsec-registry.xhtml.
20. 3GPP – The Mobile Broadband Standard. *3GPP*. URL: https://www.3gpp.org/.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)

КВАЛІФІКАЦІЙНА РОБОТА

1

На тему: «Програмний інтерфейс для протоколу Diameter мовою JAVA»

На здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Виконав:
здобувач вищої освіти гр. ІСД-41
Прокоф'єв Даніїл Андрійович

Мета, об'єкт та предмет дослідження

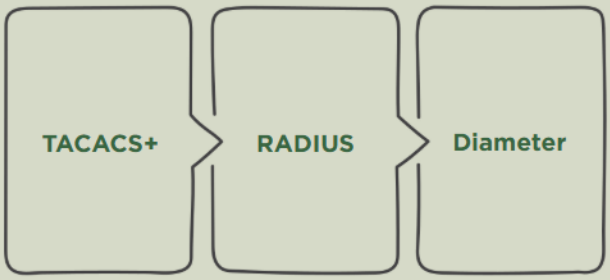
2

- Мета роботи – розробка та обґрунтування ефективного, розширюваного й зручного у використанні програмного інтерфейсу (API) мовою Java для протоколу Diameter.
- Об'єкт дослідження – протокол Diameter
- Предмет дослідження – інтерфейс протоколу Diameter

Протоколи AAA

3

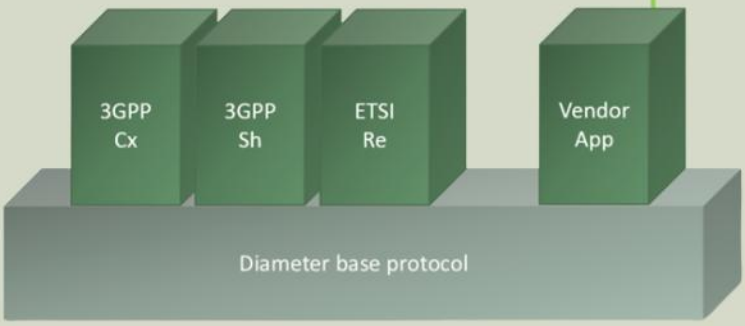
Протоколи AAA використовуються для контролю доступу до мережі та управління мережевими пристроями. AAA розшифровується як автентифікація, авторизація та облік.



Протокол Diameter

4

Протокол Diameter це новітній AAA протокол, метою якого було обійти обменження радіуса та надати можливості для подальшого масштабування



5

Інтерфейси Diameter

Інтерфейси - це модулі які дозволяють впрваджувати новий функціонал в протокол не змінюючи базовою структури та логіки

6

Протокол та інтерфейс IKE

IKE Protocol

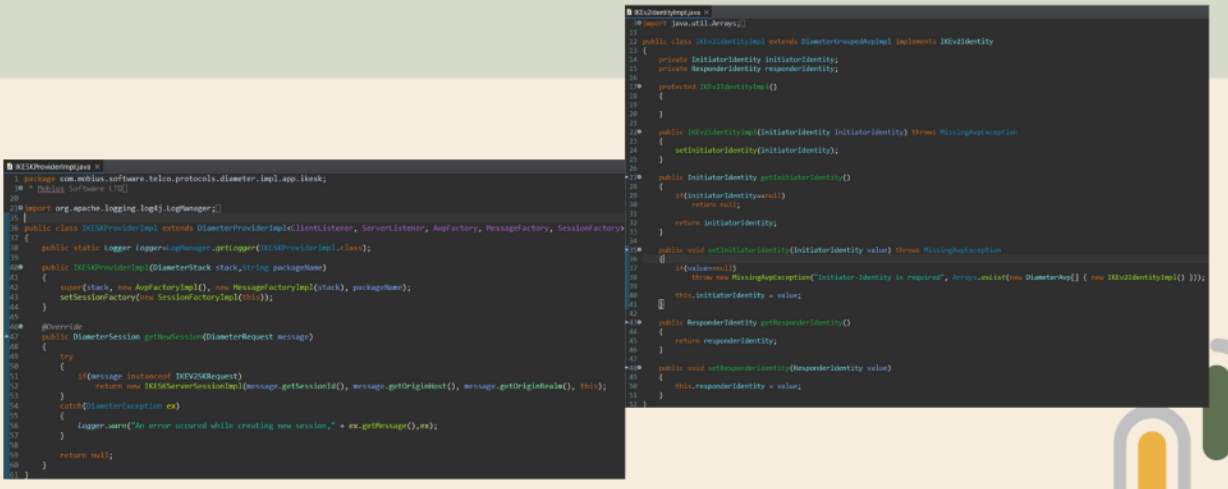
Це протокол обміну ключей в інтернеті, він використовується для створення безпечного та автентифікованого каналу зв'язку між двома сторонами через віртуальну приватну мережу

IKE Interface

Інтерфейс Diameter IKE SK це можливість використання функцій протоколу для забезпечення додатковою безпеки при обмнімні повідомленнями

Реалізація інтерфейсу

7



Тестування інтерфейсу

8

```
Diameter Protocol
    Version: 0x01
    Length: 296
  ▸ Flags: 0xc0, Request, Proxyable
    Command Code: IKEv2-SK (329)
    ApplicationId: Diameter IKE SK (IKESK) (11)
    Hop-by-Hop Identifier: 0xd9e51bbe
    End-to-End Identifier: 0xd9e51bbe
    [Answer In: 10]
  ▸ AVP: Session-Id(263) l=75 f=-M- val=client.mobius-software.com;0;1;99174e20-3271-11f0-b48a-c0a800e5540e
  ▸ AVP: Auth-Application-Id(258) l=12 f=-M- val=Diameter IKE SK (IKESK) (11)
  ▸ AVP: Origin-Host(264) l=17 f=-M- val=127.0.0.1
  ▸ AVP: Origin-Realm(296) l=34 f=-M- val=client.mobius-software.com
  ▸ AVP: Destination-Realm(283) l=34 f=-M- val=server.mobius-software.com
  ▸ AVP: Auth-Request-Type(274) l=12 f=-M- val=AUTHORIZE_AUTHENTICATE (3)
  ▸ AVP: Unknown(590) l=44 f=-M- val=0000024f4000002400000250400000c000000230000025140000000d0102030405000000
  ▸ AVP: Unknown(587) l=40 f=-M- val=0000024c4000000d0102030405000000000024d4000000d0102030405000000
```

```
Diameter Protocol
    Version: 0x01
    Length: 188
  ▸ Flags: 0x40, Proxyable
    Command Code: IKEv2-SK (329)
    ApplicationId: Diameter IKE SK (IKESK) (11)
    Hop-by-Hop Identifier: 0xd9e51bbe
    End-to-End Identifier: 0xd9e51bbe
    [Request In: 9]
    [Response Time: 0.106768670 seconds]
  ▸ AVP: Session-Id(263) l=75 f=-M- val=client.mobius-software.com;0;1;99174e20-3271-11f0-b48a-c0a800e5540e
  ▸ AVP: Auth-Application-Id(258) l=12 f=-M- val=Diameter IKE SK (IKESK) (11)
  ▸ AVP: Auth-Request-Type(274) l=12 f=-M- val=AUTHENTICATE_ONLY (1)
  ▸ AVP: Result-Code(268) l=12 f=-M- val=DIAMETER_SUCCESS (2001)
  ▸ AVP: Origin-Host(264) l=17 f=-M- val=127.0.0.1
  ▸ AVP: Origin-Realm(296) l=34 f=-M- val=server.mobius-software.com
```

9

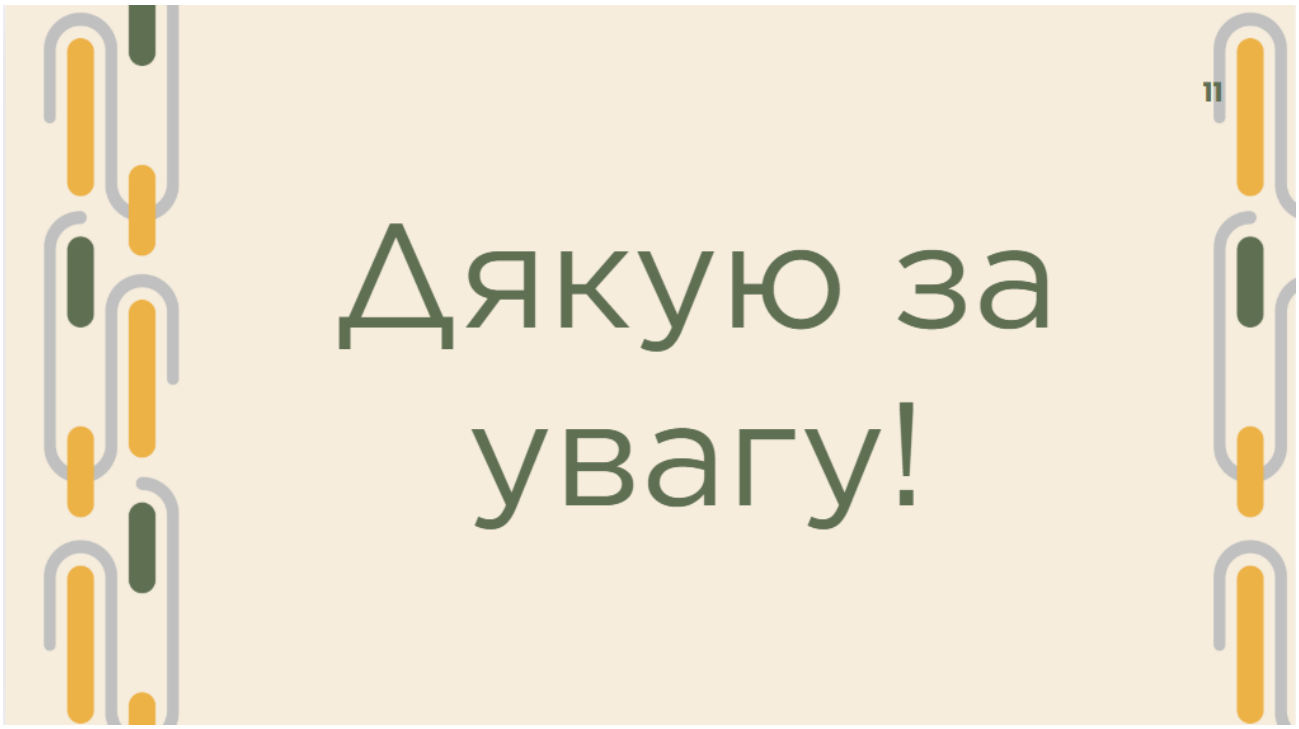
Висновки

У результаті виконаної роботи було створено інтерфейс, який успішно продемонстрував свою працездатність та відповідність заданій специфікації. Під час реалізації були враховані всі вимоги, описані в технічному завданні, а також забезпечено коректну взаємодію інтерфейсу з іншими компонентами системи. Проведене тестування підтвердило, що інтерфейс виконує покладені на нього функції стабільно та без збоїв, що свідчить про якісну реалізацію та готовність до практичного використання.

10

Апробація роботи

- Прокоф`єв Д. А. Можливості застосуваннння протоколів AAA для покращення безпеки в компьютерних мережах: матеріали VI Науково-технічної конференції «Сучасний стан та перспективи розвитку IoT». Подано до друку.
- Прокоф`єв Д. А. Застосування технологій для цифровізації життя на основі протоколу diameter: матеріали III всеукраїнської науково-технічнлї конференції «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу». Надруковано на стр. 207.


11
Дякую за увагу!