**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**імені ІГОРЯ СІКОРСЬКОГО»**

**ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ**

Кафедра інформаційної безпеки

«На правах рукопису» «До захисту допущено»

УДК 004.056 В.о. завідувача кафедри

\_\_\_\_\_\_\_\_\_\_ М.В.Грайворонський

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_2019 р.

**Магістерська дисертація**

**на здобуття ступеня магістра**

зі спеціальності: 125 Кібербезпека

на тему: Метод підвищення ефективності механізму автентифікації та авторизації протоколу RADIUS

Виконала: студентка 2-го курсу, групи ФБ-81мп

Голобородько Василиса Валентинівна

Науковий керівник: доц.каф. ІБ, к.т.н. Барановський О.М.

Рецензент доц.каф. АУТС, к.т.н. Букасов М.М.

Засвідчую, що у цій магістерській дисертації 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 – другий (магістерський) за освітньо-професійною програмою

Спеціальність (освітньо-професійна програма) – 125 Кібербезпека («Системи і технології та математичні методи кібербезпеки)

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

\_\_\_\_\_\_\_\_\_\_ М.В.Грайворонський

(підпис)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2019 р.

**ЗАВДАННЯ**

**на магістерську дисертацію студенту**

Голобородько Василисі Валентинівні

1. Тема дисертації: Метод підвищення ефективності механізму автентифікації та авторизації протоколу RADIUS

науковий керівник дисертації: к.т.н., доц. Барановський Олексій Миколайович,

затверджені наказом по університету від «\_\_\_» \_\_\_\_\_\_\_\_ 2019 р. № \_\_\_\_\_\_\_

2. Термін подання студентом дисертації 10.12.2019 р.

3. Об’єкт дослідження: безпека автентифікації та авторизації протоколу RADIUS

4. Вихідні дані: безпечний аналог протоколу RADIUS

5. Перелік завдань, які потрібно розробити:

А. Огляд літературних джерел щодо вразливостей та можливих атак проти протоколу RADIUS

Б. Ознайомлення з існуючими практичними реалізаціями протоколу RADIUS

В. Розробка практичної реалізації запропонованих рішень покращення протоколу RADIUS

6. Орієнтовний перелік ілюстративного матеріалу: 17 ілюстрацій

7. Орієнтовний перелік публікацій

8. Дата видачі завдання: 02.09.2019

Календарний план

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів виконання  магістерської дисертації | Термін виконання етапів магістерської дисертації | Примітка |
| 1 | Опрацювання літературних джерел | 30.09.2019 | Виконано |
| 2 | Отримання технічного завдання | 01.10.2019 | Виконано |
| 3 | Ознайомлення з існуючими варіантами реалізації протоколу RADIUS | 15.10.2019 | Виконано |
| 4 | Виконання технічного завдання | 15.11.2019 | Виконано |
| 5 | Доопрацювання та оформлення магістерської дисертації | 03.12.2019 | Виконано |
| 6 | Підготовка до захисту | 18.12.2019 | Виконано |

Студент \_\_\_\_\_\_\_\_\_\_\_\_ Голобродько В.В.

(підпис)

Науковий керівник дисертації \_\_\_\_\_\_\_\_\_\_\_\_ Барановський О.М.

(підпис) (ініціали, прізвище)

**РЕФЕРАТ**

Робота обсягом 85 сторінок включає 17 ілюстрацій, 24 таблиці і 14 джерел літератури.

Метою даної роботи є розробка методів покращення ефективності протоколу RADIUS для їх подальшого використання у практичній реалізації даного протоколу.

Об’єктом дослідження є існуючи механізми безпеки в протоколі RADIUS та протоколах автентифікації, які він використовує.

Предметом дослідження є захист користувацьких даних при процесах автентифікації, авторизації у протоколі RADIUS та протоколах автентифікації, які використовуються в ньому.

В процесі виконання даної дипломної роботи були детально вивчені існуючи протоколи автентифікації, авторизації та аудиту, механізми захисту даних, які в них використовуються. Запропоновані методи підвищення ефективності протоколу RADIUS та протоколу EAP-MD5.

Результати даного дослідження можуть бути використані для імплементації протоколу RADIUS в інші системи та мережі.

Radius Protocol, MD5, EAP-MD5, Authentication, Authorization, Accounting, AAA.

**ABSTRACT**

Work consists of 85 pages that include 17 illustrations, 24 tables and 14 sources of literature.

The purpose of this work is developing the methods to improve the efficiency of RADIUS protocol to use in RADIUS implementation.

The object of the study is the security mechanisms of RADIUS protocol and authentication protocols that it uses.

The subject of the study is protecting of user data during authentication, authorization in RADIUS protocol.

During the work the existed AAA protocols and their protection mechanisms that they use were studied. The methods of improving efficiency of RADIUS protocol and EAP-MD5 protocol were developed.

Work results can be used in the implementation of RADIUS protocol into other systems and networks.

Radius Protocol, MD5, EAP-MD5, Authentication, Authorization, Accounting, AAA.

**ЗМІСТ**

[Перелік умовних позначень, символів, одиниць, скорочень і термінів 8](#_Toc325014104)

[Вступ 10](#_Toc325014105)

[1 Огляд існуючих рішень ААА на базі протоколу RADIUS 12](#_Toc325014106)

[1.1 Огляд механізмів автентифікації, авторизації та аудиту в сучасних мережах 12](#_Toc325014107)

[1.2 Забезпечення процесів автентифікації та авторизації в протоколі RADIUS ..18](#_Toc325014108)

[1.3 Механізм аудиту протоколу RADIUS 21](#_Toc325014109)

[1.4 Аналіз вразливостей протоколу RADIUS 22](#_Toc325014109)

[Висновки до розділу 1 30](#_Toc325014115)

[2 Практичні методи покращення методів автентифікації та авторизації протоколу RADIUS 31](#_Toc325014116)

[2.1 Встановлення вимог до протоколів ААА 31](#_Toc325014107)

[2.2 Аналіз методів покращення алгоритму хешування MD5 32](#_Toc325014107)

[2.3 Пропозиція щодо покращення алгоритму хешування MD5 35](#_Toc325014107)

[2.4 Пропозиція щодо покращення протоколу EAP-MD5 38](#_Toc325014107)

[2.5 Додаткові поради щодо практичної реалізації протоколу RADIUS 45](#_Toc325014107)

[Висновки до розділу 2](#_Toc325014117) 47

[3 Практична реалізація запропонованих методів покращення протокола RADIUS та EAP-MD5](#_Toc325014118) 49

[3.1 Дослідження швидкості роботи запропонованого алгоритму 49](#_Toc325014119)

[3.2 Опис практичної реалізації покращеного алгоритму EAP-MD5 51](#_Toc325014119)

[3.3 Аналіз запропонованого рішення за допомогою інструменту ProVerif 57](#_Toc325014119)

[Висновки до розділу 3](#_Toc325014121) 60

[4 Розроблення стартап-проекту 62](#_Toc325014122)

[4.1 Опис ідеї проекту 62](#_Toc325014123)

[4.2 Технологічний аудит ідеї проекту 65](#_Toc325014124)

[4.3 Аналіз ринкових можливостей запуску стартап-проекту 66](#_Toc325014125)

[4.4 Розроблення ринкової стратегії проекту 73](#_Toc325014125)

[4.5 Розроблення маркетингової програми стартап-проекту 7](#_Toc325014125)6

[Висновки до розділу 4 7](#_Toc325014130)9

[Висновки 79](#_Toc325014131)

[Перелік джерел посилань](#_Toc325014132) 80

[Додаток А Функція main() реалізації клієнта RADIUS 83](#_Toc325014133)

[Додаток Б Основні методи здійснення автентифікації за допомогою EAP-MD5 класу EAPAuthenticator 84](#_Toc325014133)

[Додаток В Скрипт для тесутвання алгоритму за допомогою ProVerif 85](#_Toc325014133)

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AAA (англ. Authentication Authorization Audit) – стек протоколів, що використовуються для опису процесу надання доступу та контролю за ним.

Network access server (NAS) – сервер, який дозволяє незалежним постачальникам послуг забезпечувати користувачем доступом до інтернету.

RFC (англ. Request for Comments) – документ із серії пронумерованих інформаційних документів інтернету, що містить технічні специфікації та стандарти та має широке застосування у всесвітній мережі.

VPN (англ. Virtual Private Network) – технологія, що дозволяє забезпечити одно чи декілька мережвих з’єднань поверх іншої мережі.

UDP (англ. User Datagram Protocol) – один із протоколів стеку TCP/IP, що працює без встановлення з’єднання.

LDAP (англ. Lightweight Directory Access Protocol) – мережевий протокол прикладного рівня для надсилання запитів та модифікації даних, що містяться в службі каталогів, за допомогою стеку TCP/IP.

DOS атака (англ. Denial of Service) – атака на обчислювальну систему з метою створення умов, при яких користувачі не можуть отримати доступ до ресурсів системи, що підлягає атаці.

ASCII (англ. American standard code for information interchange) – назва таблиці, в якій числа від 0 до 127 поставлені у відповідність літерам, цифрам і символам пунктуації.

SQL (англ. Structured Query Language) – декларативна мова програмування, яка використовується для створення, модифікації та управління даними у реляційній базі даних.

TCP (англ. Transmission Control Protocol) – протокол, який призначений для управління передачею даних у комп’ютерних мережах, та працює на транспортному рівні моделі OSI.

SSL (англ. Secure Sockets Layer) – криптографічний протокол з забезпеченням безпечного з’єданння між відправником та отримувачем.

# ВСТУП

За останні десятиліття кількість інтернет сервісів швидко збільшувалась, що призвело до потреби постійного вдосконалення маршрутизаторів та серверів мережевого доступу (Network Access Server, NAS) для обробки нових сервісів. Постійне зростання кількості користувачів також вимагає великої кількості NAS зі складною конфігурацією. Але провайдери та постачальники послуг розмежування доступу до ресурсів повинні не тільки запропонувати вільний порт для кожного з’єднання. Вони також повинні забезпечити захист проти викрадення користувацьких даних, перевірку рівня доступу користувача до ресурсів, а для виставлення рахунків за використані ресурсі та подальше планування використання ресурсів, їм може знадобитися такі дані, як час підключення користувача до мережі. Усі ці послуги потребують координації між різними системами.

Саме вирішення цих проблем лежить в основі використання протоколу автентифікації, авторизації та аудиту (ААА), який є по суті фреймоврком, що вирішує разом всі ці окремі задачи завдяки різним мережевим технологіям та платформам, зокрема протоколу RADIUS.

Оскільки протокол RADIUS не володіє високим рівнем безпеки, але все ж залишається одним з найрозповсюджених протоколів ААА, метою даного дослідження є розробка методів покращення механізмів протоколу RADIUS.

Об’єктом дослідження є механізми забезпечення процесів автентифікації та авторизації протоколу RADIUS.

Предметом дослідження є безпека та її забезпечення у протоколі RADIUS.

Досягнення мети дослідження реалізується за допомогою детального вивчення існуючих механізмів захисту протоколу, огляду існуючих рішень захисту в криптографічних протоколах та можливості використання цих механізмів по відношенню до протоколу, який досліджується, з урахуванням специфіки автентифікації, авторизації та аудиту.

Таким чином, завдяки розробленим в ході виконання дослідження покращеним методам автентифікації та авторизації можуть бути вирішена велика кількість проблем безпеки сучасних реалізацій протоколу RADIUS, а отже підвищення рівня безпеки як користувацьких даних, так і обмежених ресурсів систем.

Отримані результати повинні бути перевірені за допомогою існуючих інструментів перевірки безпеки криптографічних протоколів, а також на відповідність міжнародним стандартам у сфері ААА.

Один з отриманих результатів виконання магістерської роботи був поданий у якості статті «Метод покращення рівня безпеки протоколу EAP-MD5 проти атаки за словником» у науковий журнал «Theoretical and Applied Cybersecurity».

# 1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ААА НА БАЗІ ПРОТОКОЛУ RADIUS

У першому розділі магістерської дисертації буде розглянуто призначення протоколу автентифікації, авторизації та аудиту та механізми забезпечення цих операцій за допомогою протоколу RADIUS, а також проаналізовані основні вразливості та можливі атаки на зазначений протокол.

## Огляд механізмів автентифікації, авторизації та аудиту в сучасних мережах

Для початку розглянемо кожний елемент протоколу автентифікації, авторизації та аудиту окремо.

Автентифікація – процес визначення особистості користувача, який намагається отримати доступ до ресурсів/об’єкта. Цей процес визначає, чи володіє кінцевий користувач унікальною інформацією, комбінацією «ім’я користувача - пароль», секретним ключем або біометричними даними (наприклад, відбитки пальців), які слугують однозначними ідентифікаційними даними. Сервер ААА порівнює дані автентифікації, що надаються користувачем, з даними, пов’язаними з користувачем, які зберігаються в базі даних сервера. Якщо облікові дані збігаються, користувачеві надається доступ до мережі. Інший випадок призводить до помилки автентифікації та відмови користувачеві доступу до мережі.

Відповідно до RFC2989 авторизація – це визначення, чи може певне право чи послуга бути надане пред’явнику ідентифікаційних даних. Таким правом/послугою можуть бути доступ до мережі, надання IP-адреси, виклик фільтра для визначення, які програми чи протоколи підтримуються. У деяких випадках процеси авторизації та автентифікації поєднані.

Аудит – метод збору інформації про споживання ресурсів кінцевим користувачем, яка потім може бути оброблена для виставлення рахунків, проведення аудиту, планування подальшого навантаження мережі, а також для моніторингу дій зловмисників.

У сучасних системах використовуються наступні протоколи для автентифікації, авторизації та аудиту: Remote Authentication Dial-In User Service (RADIUS), Cisco’s Terminal Access Controller Access-Control System Plus (TACACS+) та DIAMETER. У даній дипломній роботі буде розглядатися лише протокол RADIUS.

Служба для віддаленої автентифікації користувачів RADIUS – протокол, що дозволяє здійснювати централізовану автентифікацію, авторизацію та облік доступу до мережі. Спочатку розроблений для покращення функцій автентифікації Point-to-Point протоколу при передачі даних між кінцевим користувачем і NAS, RADIUS тепер підтримується серверами віртуальної приватної мережі (VPN), точками бездротового доступу, доступом до цифрових абонентських ліній та іншими типами доступу до мережі. Хоча протокол був розроблений у 1991 році, він до сих пір широко використовується, зокрема у білінгових системах (автоматизована система розрахунків «Гідра», AMSD Billing System, автоматизована система розрахунків LANBilling) (рис. 1.1), для доступу співробітників до робочого місця, для доступу адміністратора до локальної мережі підприємства з непередбачуваних місць.

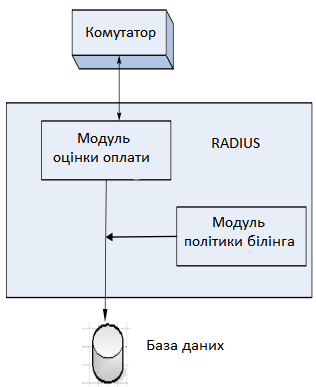


Рисунок 1.1 – Приклад архітектури білінгової системи на базі протоколу RADIUS

Специфікація RADIUS описана в наступних стандартах: RFC 2865 «Служба для віддаленої автентифікації користувачів(RADIUS)» та RFC 2866 «Аудит RADIUS».

Повідомлення у цьому протоколі передаються у формі пакетів протоколу передачі даних без встановлення з’єднання UDP. Тож усі проблеми, які пов’язані з доступністю сервера, повторною передачею даних, а також тайм-аутами, обробляються пристроями, які підтримають RADIUS, а не самим протоколом передачі. У пакеті UDP завжди знаходиться лише одно повідомлення RADIUS. Сучасна реалізація протоколу використовує порт 1812 для автентифікації та порт 1813 для аудиту. Деякі сервери можуть використовувати порт 1645 для автентифікації, а порт 1646 – для аудиту. Зазначений протокол реалізує клієнт-серверну архітектуру, де в якості клієнта постає Network Access Server (NAS), а в якості сервера – демон процес, який працює на машині з операційною системою UNIX або Windows NT. NAS виконує роль шлюзу між користувачем та сервером, відповідає за передачу інформації про користувача серверам RADIUS, потім діє в залежності від відповіді, отриманої від сервера. RADIUS сервера отримують запроси на надання з’єднання користувачеві, автентифікують користувача, а потім повертають інформацію про конфігурацію, необхідну клієнту для надання послуг, які запросив користувач. На рис. 1.2 зображена загальна схема взаємодії користувача, клієнта та сервера у протоколі RADIUS.

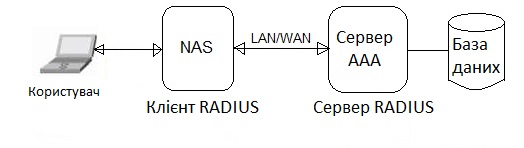


Рисунок 1.2 – Загальна схема взаємодії протоколу RADIUS

Будь-яка функціональність RADIUS серверу може бути делегована іншому RADIUS серверу. Тоді RADIUS сервер, що отримує повідомлення від клієнта, стає проксі-сервером. Наприклад Cisco часто використовує наступну модель взаємодії за протоколом RADIUS. У якості проксі-сервера, що приймає запити від клієнтів, постає Cisco Access Registrar (Cisco AR); у якості клієнта – NAS. Cisco AR передає запит на отримання інформації про користувача до серверу полегшеного протоколу доступу до каталогів LDAP (Lightweight Directory Access Protocol). І вже на підставі отриманих даних приймає рішення про автентифікацію та авторизацію користувача. Така схема роботи протоколу зображена на рис. 1.3.

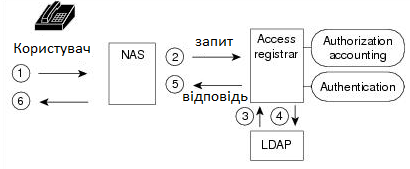


Рисунок 1.3 – Схема проксі Cisco AR з використанням LDAP серверу

Розглянемо процеси автентифікації та авторизації RADIUS протоколу більш детально.

Автентифікація та авторизація протоколу RADIUS описані у стандарті RFC 2865. Як правило, логін користувача складається із запиту від NAS до RADIUS сервера (Access-Request) та з відповіді сервера (Access-Accept або Access-Reject). Стандартами RFC 2865 та 2866 підтримуються наступні типи повідомлень:

1. Access-Request. Надсилається клієнтом RADIUS для запиту спроби автентифікації та авторизації при підключенні до мережі.
2. Access-Accept. Надсилається сервером у відповідь на Access-Request повідомлення. Це повідомлення інформує клієнта, що його спроба автентифікації та авторизації прийнята.
3. Access-Reject. Надсилається сервером у відповідь на Access-Request повідомлення. Це повідомлення інформує клієнта, що його спроба автентифікації та авторизації відхилена. Сервер відсилає це повідомлення, якщо дані користувача не дійсні або спроба з’єднання не дозволена.
4. Access-Challenge. Містить додаткову інформацію від користувача, наприклад, другий пароль, PIN або токен. Також використовується у більш складних випадках автентифікації, коли встановлюється захищений тунель між хостом користувача та сервером таким чином, що клієнт не має доступу до даних користувача.
5. Accounting-Request. Надсилається клієнтом для початку процесу аудита для з’єданння, яке буле прийняте.
6. Accounting-Response. Надсилається сервером у відповідь на Accounting-Request повідомлення. Підтверджує успішне отримання запиту на аудит.

Загальна схема пакету RADIUS зображена на рис. 1.4.

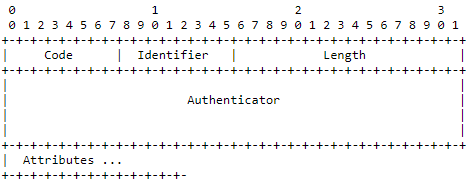


Рисунок 1.4 – Загальна схема пакету RADIUS

Кожен пакет складається з наступних полів:

1. Код. Визначає тип пакету: Access-Request, Access-Accept, Access-Reject, Access-Challenge, Accounting-Request або Accounting-Response.
2. Ідентифікатор. Містить значення, яке копіюється у відповідь сервера для того, щоб клієнт зміг зіставити свої запити з відповідями сервера. Також використовується сервером для розпізнавання клієнтів при автентифікації декількох користувачів.
3. Довжина. Слугує простим механізмом перевірки помилок. Сервер відкидає пакет, якщо його довжина коротша, ніж вказане в цьому полі значення. Якщо довжина пакет перевищує значення даного поля, то зайві октети ігноруються сервером.
4. Автентифікатор. Містить значення автентифікатора запиту чи автентифікатора відповіді. Автентифікатор запиту міститься у пакеті Access-Request. Це значення непередбачуване та унікальне. Слугує для шифрування пароля користувача при передачі від клієнта до сервера.
5. Атрибути. Залежить від типу пакету, до складу якого входить. Кількість пар атрибут/значення, що міститься у цьому полі є змінною, включаючи необхідні або необов’язкові для типу запитуваної послуги. Атрибути RADIUS описані в стандартах RFC 2865, 2866, 2867, 2868, 2869 і 3162.

Кінцевий користувач має довірчі відносини з сервером RADIUS через логін та пароль. NAS та сервер мають довірчі відносини один з одним на основі спільного секрету. Коли користувач намагається отримати доступ до мережі, NAS передає інформацію про автентифікацію між користувачем та сервером RADIUS. Цей процес називається «RADIUS conversation». Усі RADIUS conversations ініціюються NAS від імені користувача.

## Забезпечення процесів автентифікації та авторизації в протоколі RADIUS

Транзакції між клієнтом та сервером RADIUS здійснюються за допомогою використання спільного секрету, який відомий як клієнту, так і серверу, не надсилається по мережі та задається як текстовий рядок системним адміністратором. Єдине технічне обмеження спільного секрету полягає в тому, що довжина спільного секрету повинна бути більше 0, але RFC рекомендує, щоб довжина секрету була не менше 16 байтів. Ця вимога обумовлена тим, що секрет такої довжини практично неможливо зламати за допомогою атаки грубої сили.

Клієнт створює Access-Request пакет, який містить щонайменше ім’я та пароль користувача. Ідентифікатор пакета Access-Request генерується клієнтом. Зазвичай це звичайний лічильник, який інкриментується з кожним новим запитом. Access-Request пакет містить автентифікатор запиту, який є випадково обраною строкою розміром 16 байтів. Access-Request повністю незахищений, окрім атрибуту User-Password, захист якого забезпечується наступним механізмом. Клієнт та сервер мають спільний секрет. Результат конкатенації спільного секрету та автентифікатору запиту хешується за допомогою MD5 для отримання значення розміром 16 байтів, який додається по модулю з паролем користувача. Якщо пароль користувача перевищує 16 байтів, проводяться додаткові обчислення MD5, де замість автентифікатора запиту використовується зашифрований на минулому етапі блок пароля користувача.

Сервер отримує пакет Access-request та здійснює перевірку, чи має сервер спільний секрет для конкретного клієнта. Це відбувається за допомогою звернення сервера до бази даних (альтернативою бази даних можуть слугувати текстові файли, LDAP сервера і т.д.). Якщо такого розподіленого секрету для даного клієнту, запит просто відкидається. Якщо поле Request-Authenticator відсутнє у повідомленні клієнта або його не вдалося перевірити сервер також відкидає запит.

На наступному етапі сервер використовує свою базу даних автентифікації для перевірки імені користувача та пароля. База даних RADIUS зберігає інформацію про користувача (ім’я, пароль) та дані сеансу (загальний час сесії та статистику про трафік, що надходить до/від користувача). Якщо пароль дійсний, або ім’я користувача відсутнє в базі даних сервера, сервер створює пакет Access-Accept для надсилання назад клієнту. Якщо пароль недійсний, сервер створює пакет Access-Reject для відправки назад клієнту. Це повідомлення може супроводжуватися текстовим повідомленням з причиною відмови. І пакет Access-Accept, і пакет Access-Reject використовують один й той самий ідентифікатор з пакету Access-Request, надісланого клієнтом. Також сервер розміщує автентифікатор відповіді (Response Authenticator) у полі Authenticator. Автентифікатор відповіді – це результат хеш-функції MD5, вхідними даними якої є пакет відповіді сервера з відповідним автентифікатором запиту та спільного секрету:

*Response Authenticator = MD5(Code + Id + Length + RequestAuthenticator + Attributes + SharedSecret),* (1.1)

де *RequestAuthenticator* – автентифікатор запросу;

*SharedSecret* – спільний секрет клієнта та сервера.

Коли клієнт отримує пакет з відповіддю від сервера, він намагається зіставити його з своїм запитом, використовуючи поле ідентифікатора. Якщо клієнт не відсилав жодного запиту, але отримав відповідь, то така відповідь відкидається. В іншому випадку клієнт перевіряє автентифікатор відповіді, виконуючи ті ж самі розрахунки, що й у формулі (1.1), а потім порівнює результат з полем Authenticator з відповіді від сервера. Якщо вони не збігаються, то пакет з відповіддю від сервера відкидається. Якщо клієнт отримав пакет Access-Accept, ім’я та пароль користувача вважаються правильними, а користувач автентифікованим. Якщо клієнт отримав підтверджене повідомлення Access-Reject, ім’я користувача та пароль вважаються неправильними, а користувач не проходить автентифікацію.

У RADIUS процеси авторизації та автентифікації поєднані, тож при проходженні автентифікації користувач автоматично стає авторизованим у системі.

## Механізм аудиту протоколу RADIUS

Процес аудиту у протоколі RADIUS здійснюється незалежно від автентифікації та авторизації. Аудит протоколу описан в стандарті RFC2866. Аудит дозволяє передавати дані на початку та наприкінці сесії, вказуючи кількість ресурсів, використаних підчас сесії. Такими ресурсами можуть бути: час, пакети, байти і так далі.

Основна мета процесу аудиту полягає в тому, щоб виставити рахунок користувачеві за використаний доступ до мережі, для статичних цілей та для загального моніторингу мережі.

Коли NAS надає доступ до мережі користувачеві, він надсилає запис Accounting Start серверу, щоб проінформувати сервер про початок користування ресурсами мережі користувачем. Accounting Start – це Accounting Request пакет, що містить атрибут Acct-Status-Type у значенні «start». Цей пакет зазвичай містить інформацію, що ідентифікує користувача, його мережеву адресу, точку доступу та унікальний ідентифікатор сесії.

Періодично під час сесії NAS може надсилати записи проміжного оновлення (Interim Update records) серверу для того, щоб оновити статус поточної сесії. У тілі таких пакетів атрибут Acct-Status-Type встановлений у значення «interim-update». Проміжні пакети зазвичай передають інформацію про поточну тривалість сеансу та використання даних.

Нарешті, коли сесія завершена, та доступ користувача до мережі припинений, NAS надсилає серверу запис Accounting Stop, надаючи інформацію про час сесії, передані пакети, передані дані, причину відключення та іншу інформацію, що стосується доступу користувача до мережі. Атрибут Acct-Status-Type у такому записі встановлений у значення «stop». Зазвичай клієнт буде продовжувати надсилати пакети Accounting-Request до тих пір, поки не отримує пакет Accounting-Response від сервера.

Обчислення вмісту пакета Access-Request відрізняються від обчислень вмісту пакета Accounting-Request. На відміну від Request Authenticator у пакеті Access-Request, значення Request Authenticator у пакеті Authenticator-Request не є випадковим та обчислюється відповідно до стандарту RFC 2866 згідно формули (1.2).

*Response Authenticator = MD5(Code + Id + 16 нульових байтів + Attributes + Secret),* (1.2)

де *Secret* – спільний секрет сервера та клієнта.

Таким чином, сервер може перевірити Accounting-Request та відкинути пакет, якщо при перерахунку цього ж самого значення на стороні сервера отриманий результат не збігається з тим, що надійшов від клієнта. У більшості випадків до цього призводить неправильний спільний секрет.

## Аналіз вразливостей протоколу RADIUS

RADIUS сервер підтримує декілька протоколів автентифікації. Найбільш часто використовуються протокол автентифікації пароля (Password Authentication Protocol, PAP), протокол автентифікації з попереднім узгодженням (Challenge Handshake Authentication Protocol, CHAP), а також MS-CHAP (CHAP від Microsoft) першої або другої версії (MS-CHAPv2), протокол контролю доступу клієнт-сервер 802.1X (EAP, EAP-TLS, PEAP, EAP-TTLS, EAP-MD5, EAP-GTC, LEAP), HTTP дайджест автентифікація. Крім того можливе використання RADIUS з протоколом передачі «точка-точка» (Point-to-Point Protocol). При спільному використанні RADIUS сервера з усіма вищезазначеними протоколами результати автентифікації користувача та сервера доступу передаються на сервер RADIUS, який і приймає рішення щодо авторизації користувача.

В основі безпеки даного алгоритму лежить хеш MD5. MD5 – 128-бітний алгоритм хешування, який був розроблений у 1991 році. Призначений для створення дайджестів повідомлень довільної довжини. Прийшов на зміну MD4, що був недосконалим. У часи розробки протоколу хеш MD5 вважався безпечним, але з 2011 року відповідно до RFC 6151 алгоритм вважається ненадійним. На даний момент існує декілька видів злому алгоритму хешування MD5:

1. Метод грубої сили (brute force). Це підхід, який передбачає перебір усіх можливих вхідних значень до тих пір, поки не буде отриманий потрібний результат хешування.
2. Використання райдужних таблиць. Райдужні таблиця – це попередньо обчислена таблиця для інвертування хуш-функцій, як правило, для пошуку хешей паролей.
3. Колізія хеш-функцій. Під колізією розуміють отримання однакового результату обчислення хеш-функції при різних вхідних даних.

Також протокол RADIUS має вразливості, які обумовлені як самим протоколом, так і поганою реалізацію клієнта.

Одною з головних проблем протоколу, яка призводить до можливості багатьох атак на зазначений протокол, є те, що не існує перевірки сервером джерела вхідного повідомлення Access-Request. Сервер перевіряє запит на доступ приходить з IP-адреси для налаштованого клієнта RADIUS, але IP-адреси можуть бути легко підроблені.

Вирішенням цієї проблеми є використання атрибуту Message-Authenticator в пакетах Access-Request. Цей атрибут визначений у стандарті RFC 3579. Він використовується для валідації пакету Access-Request на стороні сервера та включає в себе хеш-код автентифікації повідомлень (HMAC, hash-based message authentication code), який слугує для забезпечення механізму перевірки цілісності інформації та дозволяє гарантувати, що дані, які передаються від клієнта до сервера не були змінені сторонніми особами, тобто робить неможливою атаку «людина посередені».

Клієнт перед тим, як надіслати пакет Access-Request обчислює HMAC MD5 над даними пакета, який він збирається надіслати серверу, та додає значення отриманого хеша у якості атрибута Message-Authenticator. Сервер на своїй стороні перевіряє валідність надісланого поля Message-Authenticator, здійснюючи ті ж самі дії. Якщо пакет не пройшов перевірку на стороні сервера, сервер повинен відкинути такий пакет. Формула обчислення атрибуту Message-Authenticator виглядає наступним чином:

*Message-Authenticator = HMAC-MD5 (Type, Identifier, Length, Request Authenticator, Attributes)* (1.3)

Цей атрибут повинен використовуватися для кожного пакета, що включає в себе повідомлення розширеного протоколу автентифікації. Для інших протоколів атрибут є опціональним.

Треба зазначити, що перший в сесії RADIUS пакет відповідно до стандарту 802.1х ніколи не містить EAP повідомлення. У цьому випадку поле Message-Authenticator не генерується та неможливо зробити перевірку дійсності клієнтського пакету на стороні сервера. Але клієнт все ще може здійснити перевірку відповіді від сервера за допомогою поля Request Authenticator.

Відомі наступні атаки на зазначеий протокол:

1. Атака отриманная розподіленого секрету за допомогою атрибуту логін-пароль.

Атакуючий може отримати інформацію про спільний секрет, якщо він може слідкувати за трафіком мережі. Це відбувається наступним чином: атакуючий намагається пройти автентифікацію, надсилаючи будь-який пароль клієнту. Далі атакуючий перехоплює Access-Request пакет, який клієнт надсилає серверу для перевірки даних користувача, та виконує операцію XOR, операндами якої є пароль, з яким він здійснив спробу автентифікації, та зашифровані атрибути логін-пароль з перехопленого пакету Access-Request. Таким чином він отримує значення MD5(спільний секрет + автентифікатор запиту) та автентифікатора запиту і може обчислити значення спільного секрету за допомогою формули (1.4).

*Attributes = (pass XOR MD5(SharedSecret + Request Authenticator)) XOR pass = MD5(SharedSecret + Request Authenticator),* (1.4)

де *pass –* пароль користувача;

*SharedSecret* – спільний секрет сервера та клієнта;

*Request Authenticator –* аутентифікатор запиту.

1. Атака на отримання пароля за допомогою пари атрибутів логін-пароль.

Спочатку зловмисник намагається пройти автентифікацію, використовуючи валідне ім’я користувача та будь-який пароль. Наступним кроком атакуючий перехоплює пакет Access-Request, що надходить від клієнта до сервера, та визначає результат MD5(спільний секрет + автентифікатор запиту), як в попередній атаці. Далі зловмисник може відсилати на сервер від імені клієнта пакети Access-Request, використовуючи автентифікатор запиту та MD5(спільний секрет + автентифікатор запиту) та перебирати паролі, при цьому змінюючи значення атрибуту логін-пароль, до тих пір, поки не буде отриманий пакет Access-Accept від сервера, тобто знайдений відповідний логіну пароль. Ця атака можлива за умови відсутності обмежень на кількість запитів на стороні сервера від клієнта. Також для цього виду атаки необхідно, щоб довжина паролю становила менше, ніж 16 символів, оскільки механізм потокового шифру пари логін-пароль використовує зашифровані на минулому етапі дані, якщо ті перевищують 16 байтів виводу. Будь-яка потужна автентифікація, наприклад двухфакторна автентифікація, унеможливлює цю атаку.

Рівень безпеки протоколу RADIUS залежить від генерації Request Authenticator. Request Authenticator повинен бути унікальним та непередбачуваним. Багато реалізацій протоколу використовують ненадійні генератори псевдовипадкових чисел для отримання Request Authenticator. Для реалізації наступних двох атак зловмиснику необхідно, щоб клієнт згенерував певне значення Request Authenticator. Це, як правило, не є серйозною проблемою, оскільки це поле не буле розроблено як функція безпеки. Протокол не описує процес генерації Request Authenticator, але найпоширенішим варіантом його генерування є збільшення одного байта для кожного нового запиту та використання такого лічильника у якості значення цього поля. Таким чином зловмисник може надіслати додаткові запити, змушуючи таким чином інкрементувати лічильник набагато частіше, ніж у звичному режимі роботи.

1. Пасивна компрометація пари атрибутів логін-пароль завдяки повторюваному значенню Request Authenticator.

Якщо зловмисник має можливість слідкувати за трафіком між клієнтом та сервером RADIUS, він може згенерувати словник, де кожному значенню автентифікатора запиту ставиться у відповідність захищена пара атрибутів логін-пароль. Якщо зловмисник помітить повторювання автентифікатора запиту для двох запитів, значення спільного секрету не буде мати значення, тому що завдяки операції XOR над двома відомими парами логін-пароль можна отримати перші 16 байтів незахищених паролів, до яких використана операція XOR:

*Attributes1 = pass1* XOR *MD5(SharedSecret* + *RequestAuthenticator)*

*Attributes2 = pass2* XOR *MD5(SharedSecret* + *RequestAuthenticator)*

*Attributes1* XOR *Attributes2 = pass1* XOR *pass2* XOR *MD5(SharedSecret* + *RequestAuthenticator)* XOR *MD5(SharedSecret* + *RequestAuthenticator) = pass1* XOR *pass2,*  (1.5)

де *pass1, pass2* – паролі користувачів з двох запитів відповідно;

*SharedSecret –* спільний секрет сервера та клієнта;

*RequestAuthenticator –* аутентифікатор запиту.

Успішність цієї атаки залежить від вимог до вибору користувачів паролів. Якщо всі користувачі обрали випадкові паролі однакової довжини, зловмисник не зможе нічого отримати з вилучених даних. Це досі малоймовірне явище. На практиці зазвичай зустрічаються випадки, коли користувачі вибирають паролі різної довжини (як правило, менше 16 символів) та різного ступеня випадковості.

Найпростішим випадком отримання паролів зі знайденого результату операції XOR є випадок, коли паролі значно відрізняються за довжиною. У цьому випадку один з паролів має більше символів, тому символи цього пароля, які виходять за рамки довжини іншого пароля, будуть незмінні (тобто ці символи будуть результатом операції XOR з 0). Це призводить до того, що символи, що не перекриваються довшим паролем, піддаються дії зловмисника без аналізу.

Більше складним випадок, коли зловмисник припускає, що користувачі обрали паролі з низькою ентропією. У цій ситуації можна здійснити атаку за словником, керуючись статистичним аналізом тих біт, що перекриваються у двох паролів.

Навіть паролі довші за 16 символів підпадають під цю атаку, оскільки зловмисник все одно отримує інформацію про перші 16 символів пароля.

1. Активна компрометація пари атрибутів логін-пароль завдяки повторюваному значенню Request Authenticator.

Ця атака зводиться к тому, що зловмисник спробує багато разів пройти автентифікацію, використовуючи різні значення паролів, кожного разу перехоплюючи згенерований клієнтом пакет Access-Request, отримуючи автентифікатор запиту та відповідну йому захищену пару логін-пароль. Наступним кроком атакуючий отримає значення MD5(спільний секрет + автентифікатор запиту), здійснюючи операцію XOR над паролем, який використовувався при спробах автентифікації та захищеним атрибутом логін-пароль, як показано у формулі (1.4). Зловмисник генерує словник значень, де кожному автентифікатору запита ставиться у відповідність значення MD5(спільний секрет *+* автентифікатор запиту).

Коли зловмисник перехоплює валідний пакет Access-Request, який містить значення автентифікатора запиту, що присутнє у його словнику, він може здійснити операцію XOR над захищеною парою логін-пароль та значенням MD5(спільний секрет +автентифікатор запиту), що знаходиться у його словнику, таким чином отримуючи значення паролю:

*MD5(SharedSecret + RequestAuthenticator) XOR Attributes = MD5(SharedSecret + RequestAuthenticator) XOR pass XOR MD5(SharedSecret + RequestAuthenticator) = pass,* (1.6)

де *pass –* пароль користувача;

*SharedSecret* – спільний секрет сервера та клієнта;

*Request Authenticator –* аутентифікатор запиту.

1. Атака на повторення відповідей від сервера за допомогою повторюваного Request Authenticator.

Зловмисник може скласти словник, де кожному Request Authenticator та ідентифікатору ставиться у відповідність пакет з відповіддю від сервера. Коли зловмисник бачить запит, у якому знаходиться Request Authenticator та пов’язаний з ним ідентифікатор, який присутній у словнику, він може маскуватися як сервер та відсилати відповідь клієнту, яку він спостерігав раніше при складанні словника.

Більш того, якщо серед відповідей сервера у словнику міститься хоч один пакет Access-Accept, зловмисник може здійснити спробу проходження автентифікації, змусивши клієнта створити пакет Access-Request з тим самим Request Authenticator та ідентифікатором, що у словнику відповідають пакету Access-Accept, а потім підмінити відповідь від сервера на ту, що знаходиться у словнику, таким чином успішно підтвердити автентифікацію клієнта, насправді не знаючи пароля.

1. DOS атака, що можлива завдяки передбачуваному значенню Request Authenticator.

Завдяки тому, що зловмисник може передбачати майбутні значення Request Authenticator, він може діяти як клієнт та створити словник майбутніх значень Request Authenticator та пов’язаних з ними відповідей сервера. Наступним кроком зловмисник від імені сервера на кожен клієнтський запит Access-Reject, створюючи таким чином відмову в обслуговуванні.

1. Оскільки замість атрибута User-Name у повідомленнях RADIUS можуть використовуватися атрибути NAS-IP-Address або NAS-Identifier, які містять відповідно IP адресу NAS або ідентифікатор NAS, то зловмисник, який має доступ до трафіку RADIUS, може визначити географічне розташування учасників протоколу у режимі реального часу.
2. Атака переговорів (Negotiation attack).

У ході цієї атаки NAS, RADIUS проксі-сервер або RADIUS сервер атакуючого намагаються змусити хост, який забезпечує процес автентифікації вибрати менш безпечний метод автентифікації. Наприклад, сеанс автентифікацї, що використовує EAP, замість цього буде використовувати CHAP або PAP; з’єднання, яке було встановлено за допомогою EAP-TLS, під час цієї атаки може бути здійснено за допомогою EAP-MD5-Challenge і т.д.

Також одним з слабких місць стандарту протоколу є те, що він дозволяє використовувати один той самий спільний секрет багатьма клієнтами, завдяки чому один вразливий клієнт може скомпрометувати кілька машин. Усі клієнти RADIUS, що мають однаковий спільний секрет, можуть розглядатися як єдиний клієнт RADIUS для усіх вищезазначених атак, оскільки жоден механізм захисту протоколу не використовує у своїй роботі адресу клієнта, або іншу унікальну інформацію.

Також обмеження, які існують для структури спільних секретів, можуть впливати на стан безпеки. Більшість реалізацій клієнтів та серверів дозволяють вводити спільні секрети лише як рядки ASCII. Є лише 94 різні символи ASCII, які можна ввести зі стандартної клавіатури в США (з 256 можливих). Багато реалізацій протоколу також обмежують загальну довжину спільного секрету до 16 символів або менше. Ці обидва обмеження штучно зменшують розмір значень, який повинен шукати зловмисник, щоб отримати спільний секрет.

**Висновки до розділу 1**

В першому розділі магістерської дисертації були розглянуті основні механізми реалізації ААА на базі протоколу RADIUS, проаналізовані вразливості та проблеми реалізації даного протоколу. В результаті можна виділити наступні головні проблеми безпеки даного протоколу:

1. Погана реалізація механізму захисту атрибуту логін-пароль через використання алгоритму хешування MD5 для шифрування пари логін-пароль;
2. Погана реалізація використання Request Authenticator, головними недоліками якої є недостатня ступінь випадковості Request Authenticator та його не унікальність;
3. Перевірка пакету Access-Request на стороні сервера здійснюється тільки при використанні повідомлень розширеного протоколу автентифікації у пакетах протоколу;
4. Вимоги на створення спільних секретів, через які системні адміністратори задають спільні секрети з недостатньою інформаційною ентропією.

Таким чином для покращення механізмів безпеки протоколу RADIUS необхідно замінити алгоритм хешування з MD5 на більш стійкий або додати додаткові операції у процес захисту даних користувача, а також визначити політику практичної реалізації протоколу, яка регламентувала б вимоги на формат та генерацію Request Authenticator та спільного секрету.

# 2 ПРАКТИЧНІ МЕТОДИ ПОКРАЩЕННЯ МЕХАНІЗМІВ АВТЕНТИФІКАЦІЇ ТА АВТОРИЗАЦІЇ ПРОТОКОЛУ RADIUS

У другому розділі будуть запропоновані основні вимоги до протоколів ААА та методи для запобігання атакам, які обумовлені вразливостями протоколу RADIUS, описаними у першому розділі даної магістерської дисертації.

## 2.1 Встановлення обмежень на практичну реалізацію протоколів ААА

Насамперед при розробці будь-яких модулів покращення роботи протоколу ААА чи введенні додаткових функцій з цією ж самою метою повинні враховуватися вимоги стандарту RFC2989 «Критерії оцінки протоколів ААА для доступу до мережі».

Для будь-яких протоколів ААА повинні бути визначені основні вимоги до їх реалізації. Можуть бути запропоновані наступні вимоги:

1. Масштабованість. Будь-яка реалізація протоколу повинна підтримувати можливість підтримки мільйонів користувачів і десятки тисяч одночасних запитів.
2. Архітектура реалізації протоколу повинна бути здатна підтримувати десятки тисяч пристроїв.
3. Протокол повинен забезпечувати механізм аутентифікації, захисту цілісності та конфіденційності на рівні передачі даних. Захист повинен встановлюватися між двома пристроями передачі даних, а не тільки між кінцевими пристроями ланцюга процесу автентифікації.
4. У разі невдалого з’єднання з сервером протокол повинен передбачити механізм зміни звернення до іншого сервера.
5. Повинен бути визначений механізм відновлення стану роботи після втрати даних через несправності, такі як перезавантаження серверів NAS або ААА та втрата зв’язку з NAS/AAA.
6. Повинен бути визначений механізм відправки запиту сервером до NAS, що вимагає припинення поточного сеансу авторизації.
7. Повинна забезпечуватися конфіденційність на рівні об’єкта, тобто дані, які передаються, не можуть бути змінені проміжними вузлами та повинні бути розшифровані лише тими суб’єктами ААА, для яких вони призначені.
8. Контроль за механізмом повторної передачі даних повинен бути здійснений за допомогою конкретної практичної реалізації, а не за допомогою протоколу.
9. Протокол повинен мати можливість розширювання, тобто можливість визначення додаткових атрибутів.
10. Протокол повинен забезпечувати можливість повторної автентифікації користувача за запитом клієнта.
11. Аудит повинен бути здійснений у режимі реального часу.
12. Формат даних аудиту повинен бути компактним. Дана вимога висувається для того, щоб забезпечити можливість буферизації або зберігання декількох пакетів аудиту для їх сумісної передачі згодом.
13. Повинні підтримуватися наступні протоколи аутентифікації: CHAP, EAP, PAP, а також передача паролей у відкритому вигляді для систем з одноразовим паролем.

**2.2 Аналіз методів покращення алгоритму хешування MD5**

Одною з головних проблем безпеки роботи протоколу RADIUS є використання алгоритму хешування MD5 для захисту пароля користувача при передачу між клієнтом та сервером.

Звичайно одним з найкращих варіантів у цьому випадку є заміна ненадійної хеш-функції більш стійкими варіантами, наприклад SHA256 або SHA512. Але використання цих алгоритмів призводить до результуючої строки довжиною більшою ніж в початковому варіанті алгоритму з використанням алгоритму MD5. У випадках, коли від результату обчислення хеш-функції залежать сторонні сервіси або інші поля у базі даних, це може призвести до суттєвих змін у всій системі, які можуть дорого обійтися власнику. В таких ситуаціях імплементація додаткових механізмів захисту у вже існуючі рішення може вирішити цю проблему.

Як вже зазначалось у першому розділі даної роботи, MD5 схильний до трьох видів атак: атаки за словником, атаки за допомогою райдужних таблиць та атак на основі колізій хеш-функцій.

Для забезпечення більш надійного захисту пароля при хешуванні MD5 можуть бути використані наступні методи:

1. Використання солі при хешуванні.

Сіль – це вторинна інформація, яка складається з рядка символів, який додається до основних даних та хешується. Цей процес робить дані більш стійкими до використання до них райдужних таблиць, оскільки солоні захешовані дані матимуть більшу ентропію, а отже набагато менше шансів, що ці данні існують у райдужних таблицях.

Існує декілька підходів до використання солі при хешуванні. Перший передбачає використання одного й того ж самого значення для всіх хешів. Такий підхід доцільно використовувати при різних значеннях даних, до яких застосовується функція хешування. Але оскільки в даному випадку аргументами функції хешування постають спільний секрет, який є загальним для кожної пари клієнт-сервер, та Request Authenticator, який може повторюватися, то ймовірність того, що результат хешування буде однаковим, велика.

Інший підхід передбачає використання для кожних унікальних даних свого значення солі, яке буде зберігатися разом з відповідними даними у базі даних. Але оскільки доступ до бази даних має лиш сервер, а процес хешування даних повинен відбуватися ще на стороні клієнта, який нічого не знає про сіль, то такий підхід призведе до ще одного запиту до сервера, у результаті якого сервер поверне клієнту сіль саме для цих даних користувача. Окрім додаткового звернення до сервера, цей варіант вимагатиме ще одного звернення до бази даних та створення додаткового атрибуту в пакеті даних. А також, хоч цей метод і допоможе протидіяти використанню райдужних таблиць, але дані все ще будуть так само вразливі до атаки за словником, оскільки сіль буде передаватися у відкритому вигляді мережою.

Ще один підхід передбачає використання у якості солі випадково згенерованого числа для поточного сеансу автентифікації на основі даних, які відомі як клієнту, так і серверу для уникнення передачі солі по відкритому каналу зв’язку. Таким чином сіль може бути обчислення як клієнтом, так і сервером. У якості спільних даних може виступати спільний секрет клієнта та сервера.

1. Ітеративне хешування.

У цьому випадку процес хешування буде повторюватися декілька разів. Оскільки md5 швидкий алгоритм хешування, то для уповільнення процесу атаки на хеш потрібне використання щонайменше 1000 ітерацій хешування.

1. Попереднє перетворення даних, що подаються на вхід алгоритму хешування.

Цей підхід може бути використаний наступним чином: перед використанням алгоритму MD5 до результату конкатенації спільного секрету з автентифікатором запиту, це значення може бути пропущене через будь-який симетричний шифр, де у якості ключа може виступати спільний секрет клієнта та сервера. Оскільки спільний секрет та автентифікатор запиту не є осмисленими текстами, то криптоаналіз отриманих після симетричного шифрування даних можливий тільки за умови повного перебору.

1. Метод ланцюжка.

Використовується разом з ітеративним хешуванням. Полягає в тому, що результат хешування попередньої ітерації використовується у якості вхідного значення для наступної ітерації.

**2.3 Пропозиція щодо покращення алгоритму хешування MD5**

Враховуючи усі вище зазначені методи покращення обчислення хеш-функції md5, можна отримати наступний алгоритм хешування, не прибігаючи до реалізації більш складних та повільних методів хешування:

1. Вхідне значення для хешування (у випадку пакета типу Access-Request це результат конкатенації автентифікатора запиту та спільного секрету) трансформується у значення, яке отримується за допомогою стовпчастого шифру перестановки, де у якості ключа постає спільний секрет.

Припустимо, що спільний секрет дорівнює строці «sharedSecret», а автентифікатор запиту – «reqAuthenticator». Отримаємо матрицю, яку можна представити у вигляді таблиці 2.1, де пусті осередки таблиці заповнюються символами «\*» та «@».

Таблиця 2.1 - Матриця символів спільного секрету та автентифікатора запиту

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 7 | 1 | 8 | 4 | 3 | 11 | 5 | 2 | 9 | 6 | 12 |
| s | h | a | r | e | d | S | e | c | r | e | t |
| r | e | q | A | u | t | h | e | n | t | i | c |
| a | t | o | r | \* | @ | \* | @ | \* | @ | \* | @ |

Застосовуючи стовпчастий шифр перестановки маємо наступний результат:

aqocn\*dt@eu\*ee@ei\*hetrArrt@sraSh\*tc@

1. Наступним кроком обчислюємо значення солі, виконуючи посимвольно операцію XOR, операндами якої є спільний секрет та строка, отримана з шифру перестановки. Отримаємо наступний результат:

*salt = aqocn\*dt@eu\*ee@ei\*hetrArrt@sraSh\*tc@ XOR sharedSecretsharedSecretsharedSecret = N7#^!N;$!I4*

1. Хешування. Хешування відбувається в декілька ітерацій. Вихідне значення попереднього етапа хешування є вхідним значенням для поточної операції хешування. Обчислення значення хешу має наступний вигляд:

*HV0 = MD5(ComplexPass, salt)*

*HV1 = MD5(ComplexPass, HV0, salt)*

*.*

*.*

*.*

*HVn = MD5(ComplexPass, HVn-1, salt)*

*HV = HV0 + HV1 + … + HVn,* (2.1)

де *ComplexPass* – значення отримане в результаті стовпчастого шифру перестановки на першому етапі алгоритму;

*salt –* значення отримане в результаті операції XORта спільного секрету на другому етапі алгоритму.

За допомогою зазначеного вище алгоритму хешування вирішується проблема атаки, при якій зловмисник отримує значення спільного секрету, ініціюючи процес автентифікації з будь-яким паролем, перехоплює пакет Access-Request та отримує значення хеш-функції. Хоча атакуючий і має значеня хеша, отримати значення спільного секрету не вдасться, оскільки до результату цього алгоритму марно використовувати райдужні таблиці. Також отримання однакових вихідних значень хеш-функцій можливе лише за умови, якщо користувачами були обрані однакові паролі при цьому клієнтом був згенерований однаковий автентифікатор запиту.

Допустимо, що усі паролі обрані користувачами однаково ймовірні, тобто частота зустрічі їх у виборці однакова. Тоді ймовірність того, що два користувача оберуть однаковий пароль з усіх можливих дорівнює:

(2.2)

де *n –* середня довжина пароля користувача;

*k –* кількість ASCII символів, які може вводити користувач у якості символів пароля.

Ймовірність вибору клієнтом однакового значення автентифікатора запиту залежить від вибору алгоритму для отримання автентифікатора запиту. Отже отримаємо ймовірність отримання однакового значення хеш-функції у запропонованому вище алгоритмі:

(2.3)

де *n –* середня довжина пароля користувача;

*k –* кількість ASCII символів, які може вводити користувач у якості символів пароля;

*preqAuth –* ймовірність обрання сервером однакового значення аутентифікатора запиту.

Отримана ймовірність дуже мала для того, щоб атакуючий міг перехопити два однакових результати хешування, а отже отримати будь-які значення на основі отриманих статистичних даних.

Вищезазначений метод вирішує проблему застосування райдужних таблиць до отриманого значення хеш-функції, а також колізій завдяки використанню значення солі. Але все ще можливе використання атаки за словником, при якому атакуючий вимушений буде перебрати усі можливі значення спільного секрету, тобто перебрати 95(довжина спільного секрету) значення (95 – ASCII символи, які можна ввести з клавіатури, тобто задати як символи для спільного секрету клієнта та сервера) та обчислити наступну кількість хешів:

*hashesNumber = iterations\*(95^shSLength),* (2.4)

де *iterations –* кількість ітерацій хешування;

*shSLength –* довжина спільного секрету клієнта та сервера.

Якщо обрати таку кількість ітерацій хешування, що уповільнить пошук зловмисником правильного значення спільного секрету, це також уповільнить роботу сервера. При великій кількості запитів до сервера це може призвести до повільної обробки кожного запиту, а отже сервер стане більш вразливим до DOS атак.

Але проблема передбачуваних та повторюваних значень автентифікатора запиту, при якій можна скласти словник, який містить значення автентифікатора запиту та відповідний йому хеш, залишається. Для вирішення цієї проблеми у процесі хешування пароля можна використовувати не значення автентифікатора запиту, а унікальне значення згенероване у рамках однієї сесії сервером та надіслане клієнту, так званий параметр challenge.

Однак використання параметру challenge не захищає від атаки повного перебору, оскільки значення challenge передається у відкритому вигляді мережою. Це значить, що атакуючий має додаткове вхідне значення хеш-функції.

**2.4 Пропозиція щодо покращення протоколу EAP-MD5**

Атаці за словником також підлягає протокол EAP-MD5, що використовується у якості протоколу автентифікації протоколу RADIUS. Протокол EAP-MD5 є одним із протоколів стандарту IEEE802.1x, що містить у своїй архітектурі апліканта, автентифікатора та сервер. Незважаючи на те, що на даний момент існують більше безпечні версії протоколу EAP, наприклад EAP-TLS, EAP-TTLS, EAP-FAST, EAP-MD5 досі залишається широкорозповсюдженим протоколом завдяки високій швидкості роботи та відносно легкій конфігурації. Також EAP-MD5 працює з інструментами доступу даних, які оперують з паролями, які зберігаються у відкритому вигляді такими, як LDAP, SQL, FILE, DBFILE і т.д. У цьому випадку легше оптимізувати процеси захисту даних користувача, ніж повністю змінювати структуру бази даних для того, щоб мати можливість працювати з більш сучасними протоколами. Таким чином підвищення рівня безпеки EAP-MD5 залишається актуальною задачею у сучасних реалізаціях протоколу RADIUS.

Маючи два з трьох вхідних параметра хеш-функції, наступним кроком атакуючий може здійснити атаку грубої сили, використовуючи словник значень паролів для того, щоб згенерувати хеш від значення ідентифікатора та випадкового параметру challenge отриманих при перехваті трафіку мережі та паролів зі словника по черзі до тих пір, поки не буде знайдений потрібний пароль.

Фактично атакуючий використовує відповідь від сервера для того, щоб перевірити свої здогадки про пароль. Причина простоти процесу злому полягає в тому, що противник має два з трьох вхідних параметрів: ідентифікатор запиту та параметр challenge. То ж якщо один з відкритих параметрів може бути переданий не у відкритому вигляді, зловмиснику буде важче знайти пароль користувача. Так як параметр challenge має більший розмір, то більш доцільно саме цей вхідний параметр хеш-функції передавати мережою у захешованому вигляді.

Для реалізації цього рішення може бути запропонований наступний алгоритм роботи протоколу EAP-MD5:

1. Аплікант надсилає початковий запит.
2. Автентифікатор надсилає ідентифікатор розміром 1 байт аплікату.
3. Аплікант надсилає ім’я та ідентифікатор автентифікатору.
4. Автентифікатор пересилає отриману інформацію апліканта на сервер.
5. Сервер перевіряє наявність користувача та спільний секрет для автентифікатора, від якого прийшов запит, у базі. Якщо такі дані існують, він генерує випадкову строку розміром 16 байт та обчислює значення запиту до сервера у відповідності до (2.5).

*Request = MD5 (ID + challenge) XOR password*, (2.5)

де *challenge* – випадково згенерована сервером строка;

*password* – пароль користувача.

1. Автентифікатор надсилає ту ж саму інформацію апліканту.
2. Отримавши пакет з зашифрованим паролем, аплікант отримує значення зашифрованого параметру challenge у відповідності з (2.6).

*Request XOR password = MD5 (sharedSecret + challenge) XOR password XOR password = MD5 (sharedSecret + challenge),* (2.6)

де *Request –* зашифроване значення паролю, отримане від сервера;

*password* – пароль користувача;

*sharedSecret* – спільний пароль сервера та користувача.

Наступним кроком він генерує значення поточного часу та обчислює

відповідь Response згідно з (2.7).

*Response = MD5 (hashedChallenge + timestamp),* (2.7)

де *hashedChallenge -* отриманий на минулому етапі роботи алгоритму результат хешування сервером параметра challenge;

*timestamp –* поточний час у мілісекундах.

1. Автентифікатор надсилає інформацію отриману від апліканта на сервер.
2. При отриманні відповіді від клієнта сервер насамперед перевіряє значення відмітки часу. Якщо пройшло багато часу з моменту генерації відповіді клієнта, сервера відповідає пакетом Access-Reject. В іншому випадку сервер обчислює те ж саме значення хеш-функції, що й клієнт. Якщо обчислене значення збігається з тим, що надійшло від клієнта, сервер відповідає пакетом Access-Accept.

Для того, щоб сервер не виконував обчислення хеш-функції для отримання значення *hashedChallenge* ще один раз, на кроці 5 сервер може формувати словник, де кожному ідентифікатору клієнта ставиться у відповідність обчислене значення *MD5 (ID + challenge).* У цьому випадку при отриманні сервером запиту від клієнта з відміткою часу, йому необхідно лише знайти відповідний запис у створеному словнику за ідентифікатором клієнта та обчислити значення хеш-функції з вхідними параметрами *hashedChallenge* та відмітки часу, що прийшла від клієнта.

На рис. 2.1 зображена схема взаємодії апліканта, автентифікатора та сервера згідно описаного протоколу.

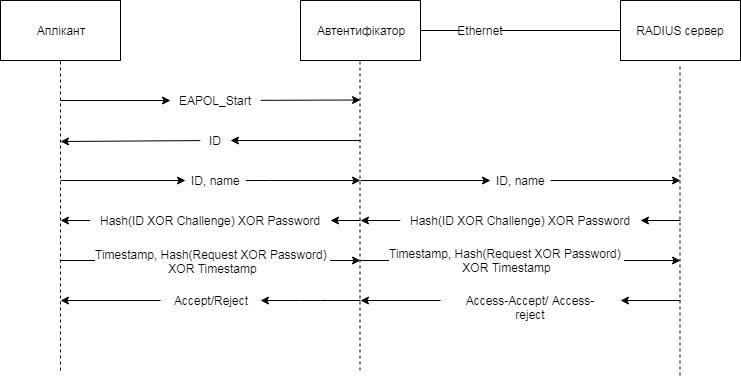


Рисунок 2.1 – Схема взаємодії апліканта, аутентифікатора та сервера згідно запропонованому алгоритму аутентифікації

Таким чином згенероване сервером значення challenge не передається каналом зв’язку, атакуючий не має два з трьох вхідних параметрів хеш-функції, а значить атака перебору паролів для успішного проходження автентифікцації є неефективною.

У цьому випадку виникає проблема генерації достатньо випадкового параметру challenge, який важко передбачити. Для генерації криптографічно стійких випадкових даних, наприклад, можна використовувати значення поточного часу, як це відбувається у UNIX-подібних операційних системах для генерації солі для використання у функції хешування паролей користувачем crypt() для зберігання у файлі /etc/passwd. Або криптографічно безпечний генератор псевдовипадкових чисел з джерел, алгоритми яких явно не можливо спостерігати, наприклад, API генератора випадкових чисел операційної системи сервера. У цьому випадку випадковість отриманих даних забезпечується даними з апаратних джерел (дисперсія шуму вентилятора або жорсткого диску), рухів мишки або генераторів випадкових чисел. Наприклад, ядро Linux (більшість RADIUS серверів працюють на хості з цією операційною системою) генерує ентропію на підставі часу рухів мишки або апаратного генератора випадкових чисел процесора, та робить отримані випадкові дані доступні другим системним процесам за допомогою файлів /dev/random та /dev/urandom, які надають інтерфейс до системного генератора випадкових/псевдовипадкових чисел. При цьому /dev/random – інтерфейс до даних генератора випадкових чисел, який виводить лише випадкові байти, які повністю складаються з бітів шума «хаотичного» пула операційної системи. Якщо пул пустий, інтерфейс не видасть нічого, а програма буде чекати отримання наступного випадково згенерованого байта з пула. На відміну від /dev/random, /dev/urandom поверне стільки байтів, скільки було запитано. Якщо в пулі недостатньо бітів, теоретично можна знайти вразливість алгоритму, який використовує випадкові дані цього пристрою. Але практично нема жодних даних про реалізацію такої атаки.

У Windows задля отримання випадкових даних використовується CryptoAPI, механізм збору ентропії системи схожий на механізм у Linux системі. Але його алгоритм генерації даних є закритим.

Серед інших вимог, які висуваються до процесу генерації параметру challenge потрібно виділити наступні: для кожної окремої сесії генерація параметру challenge відбувається заново. А також згенеровані для різних сесій параметри не повинні залежати один від одного. Механізми, що використовуються для генерації параметра, повинні гарантувати, що розкриття параметра challenge не призведе до розкриття зловмисником будь-якого іншого параметра для іншої сесії.

Запропонований метод покращення роботи протоколу зберігає швидкість реалізації протоколу завдяки використанню тієї ж самої хеш-функції, але вводить додаткове випадкове значення, яке не передається каналом зв’язку у відкритому вигляді. Такий алгоритм роботи протоколу унеможливлює отримання зловмисником спільного секрету сервера та клієнта, а також створення словника, де кожному значенню аутентифікатора запиту зіставляється значення обчисленої хеш-функції, яке передається каналом зв’язку.

Оскільки значення challenge, що надсилається сервером на першому етапі алгоритму, є випадковим числом та ніколи не передається по мережі у відкритому вигляді, то для отримання паролю користувача, окрім атаки за словником, необхідно також перебрати увесь простір можливих значень параметру challenge.

Оскільки довжина значення параметру challenge дорівнює 16 байтам = 128 бітам, простір можливих значень challenge дорівнює 2^128. Тобто якщо складність атаки на оригінальний протокол дорівнює 2х (x залежить від довжини пароля та символів, з яких він складається), то складність атаки проти запропонованого протоколу при таких же умовах складає 2(128+x). Якщо усі значення параметру challenge та користувацького пароля рівно ймовірні, тоді потрібні значення challenge та пароля у середньому будуть отримані після 2(64+x/2) спроб. У реальному житті паролі користувача не є рівномірно розподіленою величиною. Але навіть враховуючи частоту використання різних значень паролів, величина спроб для отримання потрібного значення пароля є досить великою для того, щоб обчислити її за прийнятний проміжок часу.

Запропонований метод покращення протоколу EAP-MD5 відповідає вимозі стандарту RFC4962 про забезпечення механізмів захисту від атаки відтворення.

У таблиці 2.1 наведено порівняння оригінального протоколу EAP-MD5 та запропонованого покращеного алгоритму.

Таблиця 2.1 - Порівняльний аналіз оригінального та запропонованого алгоритмів

|  |  |  |  |
| --- | --- | --- | --- |
|  | Швидкість роботи | Стійкість проти атаки відтворення | Складність атаки |
| Оригінальний протокол | Висока | Не стійкий | 2х |
| Запропонований протокол | Висока | Стійкий | 2128+х |

Однією з головних переваг запропонованого протоколу є повторне використання хеш-функції, що збільшує складність проведення атаки. А використання та перевірка відмітки часу запобігає атаці відтворення.

Запропоновані методи покращення ефективності автентифікації та авторизації RADIUS відповідають вимогам стандарту RFC6421 «Вимоги до крипто-динамічності RADIUS», що встановлює вимоги до здатності протоколу пристосовуватися до змін криптографії та вимог безпеки. Наприклад, ці вимоги можуть бути застосовані до нових модульних механізмів, які дозволяють оновити криптографічні алгоритми без істотних порушень в існуючих полях пакетів RADIUS, як це і відбувається у вищезазначених методах. Зокрема запропоновані методи відповідають таким вимогам цього стандарту: запропоновані методи не повинні вимагати нових форматів атрибутів пакету; запропоновані методи повинні підтримувати усі типи пакетів RADIUS.

**2.5 Додаткові поради щодо практичної реалізації протоколу RADIUS**

Серед додаткових порад щодо покращення рівня безпеки протоколу можна виділити наступні:

1. Використання захищеного з’єднання для транспортування пакетів RADIUS.

У якості таких з’єднань можуть виступати SSL або IPsec. Або використання готових рішень для реалізації таких з’днань. Наприклад, протоколу RadSec, який передає пакети RADIUS за допомогою TCP та TLS.

1. Встановлення вимоги до формату спільного секрету про унікальне значення спільного секрету для кожного окремого RADIUS клієнта.
2. Встановлення вимоги до довжини спільного секрету сервера та клієнта.

Якщо вибір спільного секрету обмежений символами, які можна ввести з клавіатури, то послідовність символів повинна складатись як мінімум з 22 знаків і при цьому містити малі, великі літери, цифри та спеціальні символи. Це обумовлено тим, що для досягнення значення ентропії, яка дорівнює 128 біт, кожний символ повинен мати ентропію 8 біт. При виборі символів лише з клавіатури ентропія 8-бітного символу зменшується до 5.8 біт. Для того, щоб отримати необхідний рівень ентропії, необхідно 22 символи.

1. Використання криптографічно стійких генераторів псевдовипадкових чисел у процесі генерації автентифікатора запиту для забезпечення його непередбачуваності.

Можуть бути використані методи, які були зазначені у якості методів генерації випадкового параметра challenge.

1. Формування пакетів RADIUS таким чином, щоб не всі сервера/проксі-сервера на шляху пакету автентифікації користувача мали доступ до даних цього пакета.

Це питання може бути вирішено за допомогою зберігання клієнтом окремого спільного секрету для кожного сервера, який бере участь у передачі мережею пакета RADIUS. Тоді дані необхідні серверу можуть бути захешовані з використанням спільного секрету клієнта з цим сервером. Але такий підхід передбачає знання клієнтом усього маршруту пакета та зберігання великої кількості спільних секретів, а отже погану масштабованість такої системи.

Проблема несанкціонованої зміни атрибутів пакета проксі серверами на протязі усього маршруту транспортування RADIUS пакета може бути частково вирішена за допомогою встановлення політики, що описує атрибути, які можуть бути модифіковані, а які – ні. Для атрибутів, які не можуть бути модифіковані, повинні бути визначені механізми перевірки цілісності на стороні клієнта та сервера.

1. Для захисту проти атаки переговорів, коли атакуючий змушує використовувати більш безпечний протокол автентифікації мусить бути визначена політика взаємодії сервера автентифікації з іншими хостами мережі, згідно з якою сервер автентифікації може здійснювати більш потужний метод автентифікації для одних сервісів, та менш потужний - для інших.

Згідно цієї політики, якщо сервер автентифікації отримує від клієнта пакет з типом протоколу автентифікації з рівнем безпеки нижчим, ніж від нього очікує сервер, то такий пакет повинен бути проігнорований.

1. Задля запобігання підробки ідентифікаційних даних про джерело надходження повідомлення, які міститься у полях NAS-IP-adress, NAS-IPv6-Address, Called-Station-Id, повинні бути визначені механізми захисту цих полів у пакеті RADIUS.
2. Задля запобігання підробки даних аудиту проксі-серверами для отримання додаткової плати клієнт та сервер повинні мати доступ до логів автентифікації та забезпечити процес перевірки даних аудиту, отриманих від проксі-серверів, на відповідність з даними з логів.

**Висновки до розділу 2**

В другому розділі магістерьскої дисертації були запропонован метод покращення механізму захисту користувацьких даних у протоколі RADIUS, а також метод підвищення рівня безпеки протоколу EAP-MD5 при використанні його у якості протоколу автентифікації на базі протоколу RADIUS, а також були запропоновані поради для підвищення рівня безпеки протоколу при його практичній реалізації.

**3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ЗАПРОПОНОВАНИХ МЕТОДІВ ПОКРАЩЕННЯ ПРОТОКОЛУ RADIUS ТА EAP-MD5**

У третьому розділі магістерської дисертації буде проаналізоване запропоноване покращення протоколу RADIUS, описана практична реалізація модифікованого алгоритму EAP-MD5, механізми, які використовувалися при написанні практичної реалізації, а також представлений аналіз запропонованого алгоритму за допомогою інструменту ProVerif.

**3.1 Дослідження швидкості роботи запропонованого алгоритму**

У другому розділі був запропонований метод щодо покращення процесу хешування паролю користувача завдяки використанню алгоритму симетричного шифрування та збільшенню ітерацій хешування. Дослідимо швидкість роботи запропонованого алгоритму, змінюючи кількість ітерацій процесу хешування.

Для обчислення результату хешування використовується клас Java MessageDigest, який представляє криптографічну хеш-функцію, яка обчислює хеш з двоїчних даних. У якості вхідних даних цієї функції будемо використовувати значення солі довжиною 16 байтів та значення зашифрованого симетричним шифром паролю довжиною теж 16 байтів. Тестування проводилось на процесорі Intel® Pentium® CPU N3540 2159МГц у рамках одного потоку. Отриманий результат зображений на рис. 3.1.

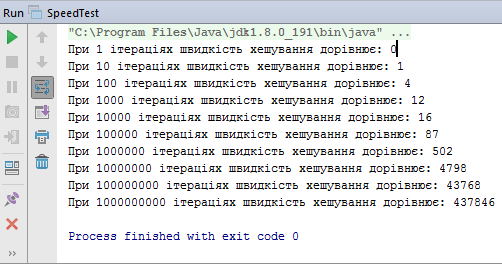


Рисунок 3.1 – Швидкість виконання хешування в мілісекундах при різній кількості ітерацій

Збільшення часу роботи програми стає помітним вже при 10000000 ітераціях, а при 1000000000 час обчислення значення хешу неприпустимо уповільнює загальну роботу програми. На рис. 3.2 зображен графік залежності швидкості роботи програми від кількості ітерацій хешування.

Рисунок 3.2 – Графік залежності швидкості виконання програми від кількості ітерацій хешування

Отже оптимальне значення кількості ітерацій для того, щоб це не призвело до помітних затримок у роботі сервера, але уповільнило використання зловмисником початкового значення пароля за словником, приблизно дорівнює 106 ітерацій.

**3.2 Опис практичної реалізації покращеного алгоритму EAP-MD5**

Для реалізації запропонованого у другому розділі методу покращення безпеки протоколу EAP-MD5, була обрана мова програмування Java. У якості апліканта та аутентифікатора виступає один процес, а у якості RADIUS сервера інший. Обидва процеси були запущені на одному хості. Тоді схема взаємодії головних учасників протоколу буде мати вигляд, який можна зобразити за допомогою рисунку 3.3.

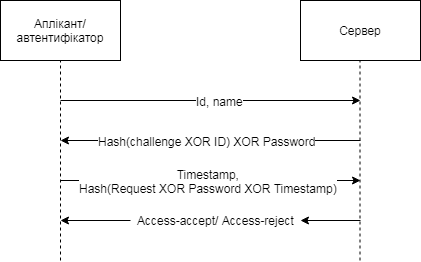


Рисунок 3.3 – Схема взаємодії учасників протоколу EAP-MD5 у практичній реалізації запропонованого алгоритму

Для реалізації прийняття рішення сервером про аутентифікацію на останньому кроку алгоритму була складена блок-схема, яка зображена на рисунку 3.4.

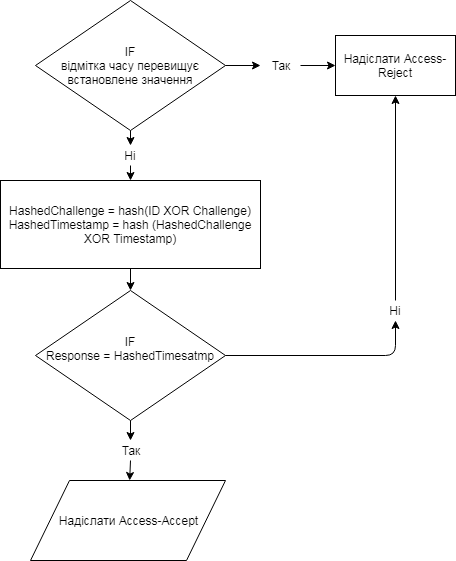


Рисунок 3.4 – Блок-схема прийняття рішення про аутентифікацію сервером RADIUS у протоколі EAP-MD5

Сервер RADIUS представлен у вигляді двох паралельно працюючих потоків: для аутентифікації на порту 1812 та для аудиту на порту 1813. Результат запуску сервера зображен на рис. 3.5.

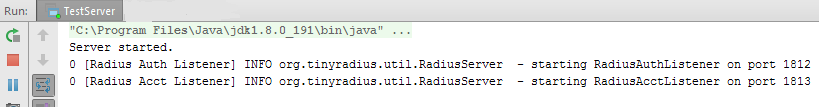


Рисунок 3.5 – Результат запуску сервера RADIUS

Клієнт RADIUS створю пакет типу Access-Request та надсилає його серверу за допомогою метода authenticate() (рис. 3.6).

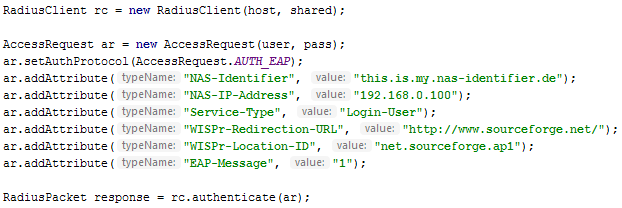


Рисунок 3.6 – Створення пакету, яким розпочинається RADIUS conversation

Отримавши запрос від клієнта на аутентифікацію користувача, сервер перевіряє, пакет якого протоколу аутентифікації він отримав (PAP, CHAP, EAP) та діє у відповідності з механізмом аутентифікації-авторизації відповідного протоколу. У даному випадку сервер обробляє пакет протоколу EAP-MD5 (рис. 3.7).



Рисунок 3.7 – Обробка сервером початкового пакету клієнта

Для формування відповіді клієнту сервер генерує значення challenge.

Для генерації значення challenge був обраний java.security.SecureRandom класс. Цей клас надає криптографічно стійкий генератор випадкових чисел. Згенероване цим класом випадкове число відповідає тестам генератора статистично випадкових чисел, визначеним у стандарті FIPS 140-2 «Вимоги безпеки до криптографічних модулів». На відміну від класу java.util.Random, який використовує лінійний конгруентний генератор для генерації випадкових чисел, SecureRandom клас використовує у своїй роботі криптографічно стійкий генератор псевдовипадкових чисел. Кожен екземпляр цього класу створюється з початковим значенням зерна, яке є основою для генерації випадкового числа та змінюється кожен раз при генерації нового значення.

Наступним кроком відбувається хешування значення параметра challenge та Id клієнта та операція XOR, операндами якої виступають захешоване значення параметру challenge та пароль користувача, отриманий сервером з бази даних за допомогою імені користувача. Усі вищезазначені методі знаходяться у класі EAPAuthenticator, який відповідає за надання функціоналу для здійснення аутентифікації за допомогою EAP (рис.3.8).



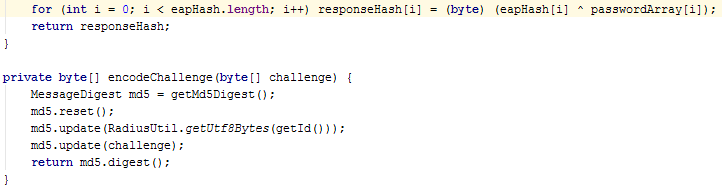


Рисунок 3.8 – Методи для перевірки клієнтських пакетів на сервері

Отримавши пакет з зашифрованим паролем користувач генерує фінальний запит за допомогою метода generateTimestampPacket(), а поточну відмітку часу розміщує у полі «Event-Timestamp».

Отримавши пакет від клієнта сервер в першу чергу за допомогою методу verifyEapRequest() перевіряє, скільки часу пройшло з моменту відправки пакету клієнтом (рис. 3.9). Це значення не повинно перевищувати встановлене на сервері значення. У практичній реалізації, опис якої приводиться, це значення дорівнює 300 мілісекундам. У реальному житті для встановлення цього значення повинна враховуватись архітектура мережі та можливі затримки при передачі даних мережою.

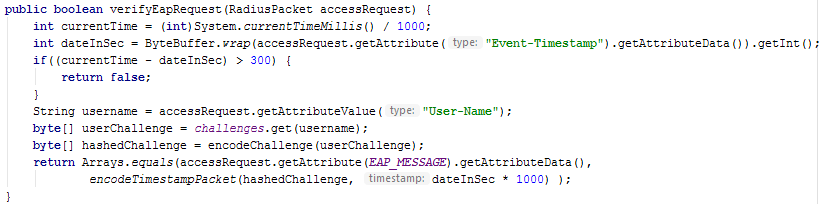
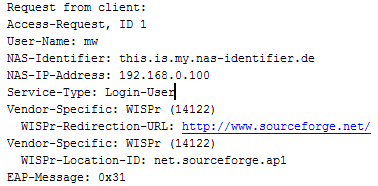
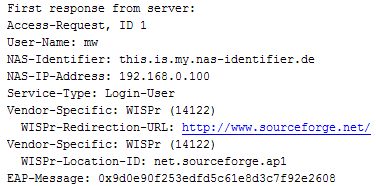


Рисунок 3.9 – Обробка сервером клієнтського пакету з відміткою часу

Для збереження на сервері пар значень «id клієнта – захешований параметр challenge» був використаний класс HashMap. Цей клас використовує хеш-таблицю для зберігання значення. Будь-яке значення у хеш-таблиці зберігається за індексом, який обчислюється знаходженням хеша від заданого ключа. Таким чином забезпечується висока швидкість пошуку потрібного значення серед усіх користувачів, яких на даний час обслуговує сервер.

Результат RADIUS conversation між клієнтом та сервером відповідно до запропонованого алгоритму EAP-MD5 зображений на рисунку 3.10.





C:\Users\Васька-супер)\Desktop\код.png

Рисунок 3.10 – Результат RADIUS conversation між клієнтом та сервером

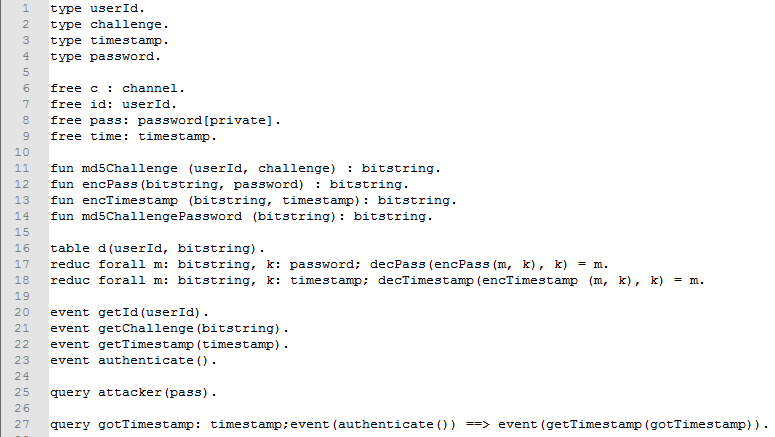
**3.3 Аналіз запропонованого рішення**

Для аналізу запропонованого протоколу був обраний інструмент ProVerif. ProVerif – це інструмент для автоматичного аналізу безпеки криптографічних протоколів, які можуть бути описані за допомогою моделі загроз Долева-Яо. ProVerif підтримує, але не обмежується лише ними, наступні криптографічні примітиви:

1. Симетричне та асиметричне шифрування;
2. Цифрові підписи;
3. Хеш функції;
4. Протокол bit commitment;
5. Неінтерактивний протокол доказу з нульовим розголошенням (Non-interactive zero-knowledge proof).

За допомогою ProVerif можна перевірити властивості доступності, припущення про відповідність деяким ознакам та еквівалентність твердженням, таким чином надаючи змогу проаналізувати властивості секретності та процесу автентифікації в протоколах, що аналізуються. Також можуть бути враховані властивості конфіденційності, процесу передачі даних та верифікації. При аналізі протоколу припускається, що при роботі протоколу може бути використана необмежена кількість сеансів та необмежений простір повідомлень. За допомогою ProVerif можна відновити можливу атаку на протокол.

Для аналізу криптографічного протоколу за допомогою ProVerif необхідно скласти модель криптографічного протоколу, описану за допомогою синтаксису π-обчислень, які описують паралельні обчислення у мережі, конфігурація якої з часом може змінюватись, та властивості безпеки, які необхідно перевірити. Модель протоколу зображена на рис. 3.11.



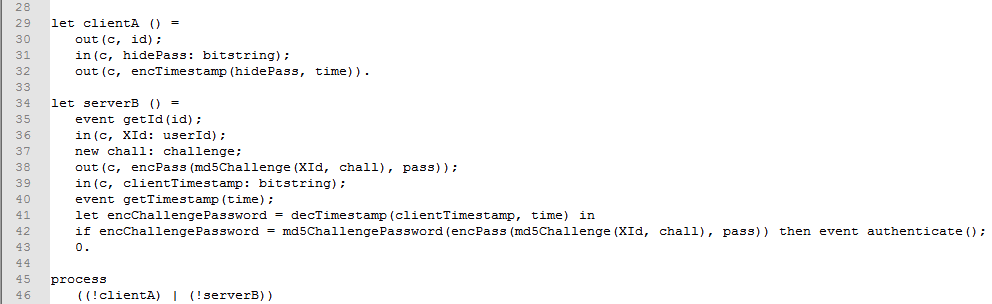


Рисунок 3.11 – Модель протоколу відповідно до синтаксису ProVerif

Спочатку описуються типи даних, які використовуються у протоколі. У даному випадку маємо наступні типи даних: клієнтський id clientId; challenge для опису випадкового значення, яке генерується сервером; timestamp для опису поточної відмітки часу та password для опису пароля користувача.

Наступним кроком описуються ті змінні та функції, які використовуються у процесі аутентифікації. Для опису процесу комунікації необхідно визначити канал передачі – змінна с типу channel.

Маємо чотири функції: md5Challenge() для хешування параметру challenge; encPass() для операції XOR над паролем користувача та результатом обчислення md5Challenge(); encTimestamp() для операції XOR над зашифрованим паролем та відміткою часу та md5ChallengePassword, яка виконує хешування над захешованим параметром challenge та паролем користувача у якості першого оператора та відмітки часу як другого оператора. Так як операція XOR є обратимою функцією, потрібно визначити обратні функції для методів encPass() та encTimestamp() за допомогою оператора reduc.

За допомогою оператору query робляться припущення про можливість отримання зловмисником пароля та порядок виконання дій у протоколі, на основі яких ProVerif проводить аналіз даного протоколу. У даному випадку робляться два припущення: про те, що зловмисник не може отримати пароль:

*query attacker(pass).*

Та про те, що подія, яка позначає аутентифікцаію клієнта, відбудеться тільки після того, як будут виконані усі попередні кроки:

*if encChallengePassword = md5ChallengePassword(encPass(md5Challenge(XId, chall), pass)) then event authenticate();*

Далі описуються дії, які виконують clientA та serverB.

При запуску вищезазначеного скрипта маємо результат, який зображений на рисунку 3.12.

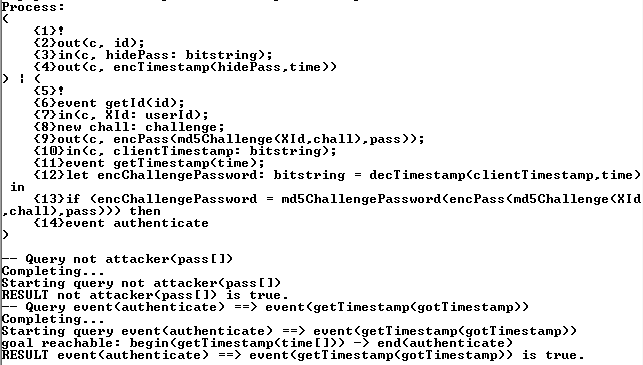


Рисунок 3.12 – Результат роботи ProVerif

Усі припущення справдилися, про що свідчить наступні дві строчки:

C:\Users\Васька-супер)\Desktop\код.png

C:\Users\Васька-супер)\Desktop\код.png

Отримані результати означають, що зловмисник не має доступу до змінної pass, яка містить пароль користувача, та подія authenticate(), що позначає аутентифікацію користувача, відбудеться тільки після того, як будуть пройдені минулі кроки алгоритму.

**Висновки до розділу 3:**

В третьому розділі була проаналізована швидкість роботи запропонованого покращення процесу хешування за допомогою алгоритму MD5 для того, щоб знайти оптимальне значення кількості ітерацій для уповільнення повного перебору значень зловмисником, але без втрати швидкості роботи основного алгоритму. Таке значення приблизно дорівнює 106 ітерацій.

Також була описана детальна схема та практична реалізація запропонованого покращеного алгоритму EAP-MD5, запропонованого у минулому розділі магістерської роботи. А також їх аналіз за допомогою існуючого рішення для аналізу криптографічних протоколів. В результаті використання інструменту тестування ProVerif було доведено, що атакуючий не зможе отримати пароль користувача, а автентифікація буде здійснена тільки після пройдешня усіх кроків алгоритму.

# 4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

У четвертому розділі описується реалізація першого етапу розроблення стартап-проекту, а саме висвітлення маркетингових аспектів створення стартапу по реалізації оновленої версії протоколу RADIUS: відбір ідей, створення концепції продукту, визначення перспектив ринкової реалізації проекту та розроблення маркетингової стратегії для просування майбутнього продукту на ринку.

* 1. **Опис ідеї проекту**

Незважаючи на те, що на зміну протоколу RADIUS прийшов протокол DIAMETER, RADIUS у якості протоколу ААА до сих пір підтримується багатьма пристроями, завдяки більш легкій конфігурації порівняно з DIAMETER, а також неповній зворотній сумісності, завдяки якій після розробки та впровадження DIAMETER, не всі пристрої підтримуються використання DIAMETER для ААА. Також швидкість роботи RADIUS у рази більша, ніж швидкість DIAMETER, а також процес аутентифікації RADIUS потребує набагато менше трафіку порівняно з DIAMETER, що робить використання RADIUS більш прийнятним у ситуаціях, коли швидкість роботи протоколу має значення, а отже пошук методів покращення ефективності RADIUS і у наші дні залишається важливим.

Таблиця 4.1 - Опис ідеї стартап-проекту

|  |  |  |
| --- | --- | --- |
| Зміст ідеї | Напрямки застосування | Вигоди для користувача |
| Створення більш ефективної реалізації протоколу RADIUS. | Використання покращеної версії протоколу у сучасних білінгових системах. | Використання вдосконаленої, більш безпечної версії протоколу. |
| Використання провайдерами у мережах малого та середнього розміру. | Мінімізація фінансових витрат у порівнянні з переходом на DIAMETER. |

Після аналізу покращених версій протоколу конкурентів потрібно виділити сильні, слабкі та нейтральні характеристики ідеї проекту, які представлені у таблиці 4.2.

Таблиця 4.2 - Визначення сильних слабких та нейтральних характеристик ідеї проекту

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | | 4 | 5 |
| № п/п | Техніко-економічні характеристики ідеї | (потенційні) товари/концепції конкурентів | | W (слабка сторона) | S (сильна сторона) |
| Запропонований проект | DIAMETER |
| 1 | економічні | Витрати на розробку рішення, закупку ліцензій, процес стандартизації | Витрати на розробку рішення, закупку ліцензій, закупку оновленої техніки, що підтримує дану технологію | Недостатній рівень витрат на маркетинг. | Значно дешевше застосування протоколу завдяки відсутності необхідності закупівлі нової апаратури. |
| 2 | технічні | Використання продукту у якості протоколу для здійснення ААА. | Підтримка мереж лише малого чи середнього розміру. | Простота налаштування. |  |

Продовження таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | | 4 | 5 |
| 3 | надійності | Відповідність стандартам, які регламентують роботу протоколів ААА для доступу до мережі. | | Нездатність розширюватись під запити мереж великого розміру. Ненадійність траспортування пакетів даних. | Незалежність від криптографічних протоколів. |
| 4 | технологічні | Здатність підтримки пристроями рішень продукту | | Нездатність підтримувати сучасними пристроями запропонованих рішень через застарілі технології, які використовуються у продукті. | Відсутність необхідності закупівлі нової апаратури для реалізації продукту. |
| 5 | ергономічні | Рівень залучення працівників до процесу налаштування роботи продукту | | Погана масшатбованість. | Простота налаштування. Висока швидкість роботи протоколу. |
| 6 | естетичні | Наявність сучасного інтерфейсу для взаємодії користувача та персоналу обслуговування (наприклад, системний адміністратор) з системою. | | Недостатній рівень конфігурабельності системи. | Більш сучасний інтерфейс взаємодії користувача/персоналу з системою |
| 7 | транспортабельності | Має максимальну простоту налаштування для використання. | Налаштування вимагає великої кількості дій для інтеграції системи. | Недостатній рівень контролю за розповсюдженням продукту без ліцензії. | Простота налаштування та інтеграції продукту. |

Кінець таблиці 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | | 4 | 5 |
| 8 | екологічності | Велика кількість необхідного обладнання | Менша кількість обладнання, що використовується, але можливе використання більш застарілих апаратних рішень. | Підтримка застарілого обладнання, яке потребує використання більшої кількості ресурсів. | Використання меншої кількості обладнання, як наслідок менше використання ресурсів живлення. |

* 1. **Технологічний аудит ідеї проекту**

Результати аналізу технології, за допомогою якої можна реалізувати створення покращеної версії протоколу RADIUS, представлені у таблиці 4.3.

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Ідея проекту | Технології її реалізації | Наявність технологій | Доступність технологій |
| 1 | Впровадження методів для покращення ефективності протоколу RADIUS, описаних у другому розділі | Покращення рівня безпеки існуючих засобів аутентифікації та авторизації RADIUS | Наявна | Доступна |
| 2 | Впровадження сервісу для аналіза логів аудиту | Впровадження додаткового сервісу по аналізу логів аудиту RADIUS | Необхідно доробити | Доступна |
| 3 | Використання більш сучасного інтерфейсу налаштування зазначеного протоколу | Використання оновленого інтерфейсу доступу до налаштування ААА з прилученням до роботи веб-дизайнерів | Необхідно доробити | Доступна |
| Обрана технологія реалізації ідеї проекту: так як для реалізації ідеї проекту, всі технології є наявними та доступними, тому обираються всі вище описані технології. | | | | |

* 1. **Аналіз ринкових можливостей запуску стартап-проекту**

У даному розділі визначаються ринкові можливості, які можна використати під час ринкового впровадження запропонованого проекту, та ринкові загрози, які можуть перешкодити реалізації проекту. По-перше, для цього необхідно визначити попередню характеристику потенційного ринку стартап-проекту, яка представлена у таблиці 4.4.

Таблиця 4.4 - Попередня характеристика потенційного ринку стартап-проекту

|  |  |  |
| --- | --- | --- |
| № п/п | Показники стану ринку протоколів ААА | Характеристика |
| 1 | Кількість головних гравців, од | <200 |
| 2 | Загальний обсяг продаж, грн/ум.од | > 1 млн ум.од |
| 3 | Динаміка ринку (якісна оцінка) | Зростає |
| 4 | Наявність обмежень для входу | Обмеження на забезпечення певного рівня безпеки для протоколу, які використовуються у продукті. |
| 5 | Специфічні вимоги до стандартизації та сертифікації | Необхідність відповідності до прийнятих стандартів у галузі ААА. |
| 6 | Середня норма рентабельності в галузі, % | Приблизно 150% |

За попередніми оцінками ринок розробки рішень для забезпечення роботи ААА є привабливим для розробки та впровадження нових рішень та не вимагає великого розміру стартового капіталу, але висуває ряд вимог та обмежень до відповідності рішень до існуючих специфікацій у даній сфері.

Для подальшого просування продукту на ринку необхідно визначити потенційні групи споживачів продукту та їх характеристики. Ці дані представлені у таблиці 4.5.

Таблиця 4.5 – Таблиця потенційних клієнтів стартап-проекту

|  |  |  |  |
| --- | --- | --- | --- |
| Потреба, що формує ринок | Цільова аудиторія (цільові сегменти ринку) | Відмінності у поведінці різних потенційних цільових груп клієнтів | Вимоги споживачів до товару |
| Безпечний протокол ААА, який не вимагає складної конфігурації та може обслуговувати велику кількість користувачів | Невеликі інтернет- провайдери | Спостереження за даними аудиту для контролю рахунків, що виставляються | Надійний механізм забезпечення процесу аудиту |
| Постачальники послуг білінгових систем | Високий рівень безпеки даних, що передаються по мережі |
| Компанії, які потребують розмежування доступу до внутрішніх корпоративних ресурсів | Розмежування доступу до внутрішніх корпоративних ресурсів | Надійність механізмів, що забезпечують безпеку аутентифікації та авторизації |

Був проведений аналіз ринкового середовища. Результат проведеного аналізу може бути зображений у вигляді таблиці 4.6 та таблиці 4.7.

Таблиця 4.6 – Фактори загроз

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| № п/п | Фактор | Зміст загрози | Можлива реакція компанії |
| 1 | Конкуренція | Поява у конкурентів більш надійних механізмів захисту | Залучення фахівців для розробки більш надійного рішення.  Пропозиція покупки великою компанією |
| 2 | Збільшення розміру податків | Відтік коштів із сфери виробництва до бюджету | Пошук шляхів мінімізації податків |

Кінець таблиці 4.6

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 3 | Збільшення рівню інфляції | Знецінювання коштів, ріст різниці в курсах валют | Отримання довгострокового кредиту |

Таблиця 4.7 – Фактори можливостей

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Фактор | Зміст можливості | Можлива реакція компанії |
| 1 | Зниження довіри до можливих конкурентів | Втрата можливими конкурентами своїх позицій на ринку | Розширення послуг, які надає компанія. Розширення маркетингової компанії. |
| 2 | Поява нових технологій та більш досконалого обладнання | Покращення ефективності роботи протоколу за рахунок використання нових технологій | Підвищення вартості послуг. Розширення маркетингової компанії. Залучення нових клієнтів на інвестицій. |
| 3 | Стабілізація політичного та економічного становища в державі | Збільшення купівельної спроможності користувачів. Підвищення попиту на продукт. | Підвищення вартості послуг. Розширення маркетингової компанії. Залучення нових клієнтів на інвестицій. |
| 4 | Збільшення кількості інтернет- сервісів | Збільшення рівня попиту на послуги компанії. | Підвищення вартості послуг. Розширення маркетингової компанії. Розширення спектру послуг. |

Аналіз загальних рис конкуренції на ринку представлений у таблиці 4.8.

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

|  |  |  |
| --- | --- | --- |
| Особливості конкурентного середовища | В чому проявляється дана характректиристика | Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентноспроможною) |
| Тип конкуренції: чиста | Невелика кількість компаній, що впроваджує продукт, який відповідає стандартам. | Об’єднання з іншими компаніями. |
| За рівнем конкурентної боротьби: національний | Споживачами продукту є представники IT-сектору в незалежно від географічного розташування. | Підвищення якості продукту. Створення продукту, що відповідає міжнародним стандартам. |
| За галузевою ознакою: міжгалузева | Використання продукту користувачами з різних сфер, оскільки необхідність розмежування доступу існує в багатьох сферах діяльності. | Розширення спектру послуг. Більш спеціалізоване надання послуг для конкретної галузі. |
| Конкуренція за видами товарів: товарно-видова | Конкуренція між модифікованими реалізаціями протоколу RADIUS. | Підвищення якості послуг. Надання додаткових функцій для конкретної галузі. |
| За характером конкурентних переваг: нецінова | Хоча послуги, які надаються, схожі за своїм функціоналом, але кожна використовується для окремих випадків або стеку технологій. | Покращення якості послуг. Розширення спектру послуг. Надання послуг в унікальних галузях |
| За інтенсивністю: марочна | Наявність відомих брендів на ринку (Cisco). | Розширення маркетингової компанії. |

Був проведений більш детальний аналіз умов конкуренції в галузі за моделлю п’яти сил М. Портера, результати якого представлені у виді таблиці 4.9.

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Складові аналізу | Прямі конкуренти в галузі | Потенційні конкуренти | Постачальники | Клієнти | Товари-замінники |
| Cisco  JumpCloud  Free Radius | Диференціація продукту та високий рівень довіри до існуючих продуктів. Потреба в початковому капіталі. | Можливість підвищення ціни на продукт, обмеження обсягів послуг, які надаються. | Можливість використання продуктів-замінників. Велика кількісні клієнтів. Доступність інформації з приводу послуг, які надаються. | Інтегрованість товарів-замінників з іншими системами. Краще співвідношення «ціна-якість» |
| Висновки | Високий рівень інтенсивності конкурентної боротьби з боку прямих конкурентів. | Є можливості входу в ринок.  Потенційні конкуренти, які вже вийшли на ринок, наявні. | Постачальники у меншій мері диктують умови роботи на ринку, оскільки технології розвиваються в результаті потреб клієнтів. | Саме клієнти у великій мірі диктують умови роботи на ринку, а саме: стек технологій, що використовуються у продукті, ціну та якість. | Для можливості конкурувати з продуктами-замніниками необхідно використовувати лише сучасні технології. |

Розроблений стартап-проект може конкурувати на даному ринку. Для того, щоб бути конкурентоспроможним на даному ринку, необхідно, щоб стек технологій, який використовується у продукті, відповідав сучасним потребам клієнтів та міжнародним стандартам у галузі ААА, а також міг динамічно пристосовуватися до майбутніх потреб споживачів у даній галузі.

На основі провденего аналізу конкуренції, а також враховуючи характеристики проекту (таблиця 4.2) та факторів маркетингового середовища (таблиця 4.6 та 4.7), були визначені фактори конкурентоспроможності. Результат проведеного аналізу представлений у таблиці 4.10.

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

|  |  |  |
| --- | --- | --- |
| № п/п | Фактор конкурентоспроможності | Обґрунтування |
| 1 | Гнучкість цін на послуги | Оскільки продукт тільки розвивається, є можливість гнучкого впровадження цін на послуги |
| 2 | Мотивація персоналу на підвищення якості продукції | Продукт не залежить від апаратної платформи (на відміну, наприклад, від Cisco), тож може більш динамічно покрити більше сценаріїв використання. |
| 3 | Можливість злиття компанії з іншими компаніями | Оскільки продукт є стартап-проектом, може відбутися злиття компанії з іншими компаніями. Як результат, більший спектр послуг, що надається користувачам. |
| 4 | Наявність великої бази споживачів | Компанії-конкуренти знаходяться більший час на ринку, а отже мають більше клієнтів та шляхів реалізації послуг. Але запропонований проект має перевагу в наявності тих клієнтів, що використовують застарілі технології, але хочуть підвищити їх рівень безпеки. |

За визначеними факторами конкурентоспроможності (таблиця 4.10) був провдений аналіз сильних та слабких сторін стартап-проекту, результати якого представлені у таблиці 4.11.

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін запропонованої модифікації RADIUS

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № п/п | Фактор конкурентоспроможності | Бали  1-20 | Рейтинг товарів-конкурентів у порівнянні з запропонованою модифікацією RADIUS | | | | | | |
| -3 | -2 | -1 | 0 | 1 | 2 | 3 |
| 1 | Гнучкість цін на послуги | 15 |  | + |  |  |  |  |  |
| 2 | Мотивація персоналу на підвищення якості продукції | 12 |  |  | + |  |  |  |  |
| 3 | Можливість злиття компанії з іншими компаніями | 17 | + |  |  |  |  |  |  |
| 4 | Наявність великої бази споживачів | 8 |  |  |  |  |  |  | + |

Таблиця 4.12 – SWOT-аналіз стартап-проекту

|  |  |
| --- | --- |
| Сильні сторони:   * Легка конфігурація та швидке впровадження технології * Низька ціна * Можливість комбінування рішення з різними апаратними та програмними продуктами | Слабкі сторони:   * Відсутність репутації та маркетингової компанії * Часткове використання застарілих тежнологій |
| Можливості:   * Злиття компанії з іншими для розширення спектру послуг * Швидке впровадження нових технологій | Загрози:   * Використання товарів-замінників * Використання більш сучасних технологій * Невисокий рівень попиту |

На основі SWOT-аналізу були розроблені альтернативи ринкової поведінки для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів. Результати аналізу альтернатив ринкового впровадження стартап-проекту представлені у таблиці 4.13.

Таблиця 4.13 – Альтернативи ринкового провадження стартап-проекту

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Альтернатива ринкової поведінки | Ймовірність отримання ресурсів | Строки реалзіації |
| 1 | Концентрація на конкретних споживачах | Середня | 6 місяців |
| 2 | Розробка сумісного проекту з іншими постачальниками послуг | Висока | 1-1.5 роки |

З вище перерахованих альтернатив була обрана «Розробка сумісного проекту з іншими постачальниками послуг». Хоча ця альтернатива вимагає більше часу для впровадження, але ймовірність отримання ресурсів більш велика в порівнянні з іншим варіантом.

* 1. **Розроблення ринкової стратегії проекту**

Для розроблення ринкової стратегії насамперед необхідно визначити стратегії охоплення ринку за допомогою опису цільових груп потенційних споживачів (таблиця 4.14).

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| № п/п | Опис профілю цільової групи потенційних клієнтів | Готовність споживачів сприйняти продукт | Орієнтовний попит в межах сегменту | Інтенсивність конкуренції в сегменті | Простота входу в сегмент |
| 1 | Інтернет провайдери невеликих за розміром мереж | Потребують безпечного процесу аудиту за використанням доступу, який надається | Середній рівень попиту | Середній рівень конкуренції | Складність полягає в невисокому рівні довіри |

Кінець таблиці 4.14

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 2 | Компанії з розподіленою внутрішньою мережою | Потребують розмежування доступу до своїх внутрішніх ресурсів | Високий рівень попиту | Середній рівень конкуренції | Складність входу полягає в проведенні відповідної маркетингової компанії |
| 3 | Білінгові системи | Потребують безпечного процесу аутентифікації та авторизації | Середній рівень попиту | Високий рівень конкуренції | Складність входу полягає в необхідності відповідності великій кількості стандартів у даній сфері |
| Обрані цільові групи: компанії з розподіленою внутрішньою мережою, інтернет-провайдери | | | | | |

У результаті аналізу потенційних груп споживачів були обрані дві цільові групи, а у якості стратегії охоплення ринку була обрана стратегія диференційованого маркетингу.

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (таблиця 4.15).

Таблиця 4.15 – Визначення базової стратегії розвитку

|  |  |  |  |
| --- | --- | --- | --- |
| Обрана альтернатива розвитку проекту | Стратегія охоплення ринку | Ключові конкурентоспроможні позиції відповідно до обраної альтернативи | Базова стратегія розвитку |
| Розробка сумісного проекту з іншими постачальниками послуг | Диференційованого маркетингу | Широкий спектр послуг з використанням різних технологій | Спеціалізації |

Наступним кроком необхідно обрати стратегію конкурентної поведінки (таблиця 4.16).

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

|  |  |  |  |
| --- | --- | --- | --- |
| Чи є проект «першопрохідцем» на ринку? | Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів? | Чи буде компанія копіювати основні характеристики товару конкурента, і які? | Стратегія конкурентної поведінки |
| Ні | Шукати нових користувачів | Так, підтримку специфічних апаратних рішень | Стратегія зайняття конкурентної ніші |

На основі вимог споживачів з обраних сегментів до стартап-компанії та до продукту, а також відповідно до стратегії спеціалізації та стратегії зайняття конкурентної ніші була розроблена стратегія позиціонування, опис якої представлений у таблиці 4.17.

Таблиця 4.17 – Визначення стратегії позиціонування

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Вимоги до товару цільової аудиторії | Базова стратегія розвитку | Ключові конкурентоспроможні позиції стартап-проекту | Вибір асоціацій, які мають сформувати комплексну позицію проекту |
| 1 | Високий рівень безпеки | Спеціалзації | Легка конфігурація та швидке впровадження технології  Високий рівень безпеки  Можливість комбінування рішення з різними апаратними та програмними продуктами  Сучасний та зручний інтерфейс доступу до даних аудиту | Безпечний процес здійснення аутентифікації та авторизації.  Зручний аудит та аналіз даних. |
| 2 | Зручний інтерфейс використання та перегляду даних аудиту |
| 3 | Швидкий процес конфігурації |
| 4 | Висока швидкість |
| 5 | Відсутність сторонніх сервісів, які беруть участи у ААА |

* 1. **Розроблення маркетингової програми стартап-проекту**

Першим кроком розроблення маркетингової програми стартап-проекту є формування маркетингової концепції продукту, який отримає споживач. Для цього були підсумовані результати попереднього аналізу конкурентоспроможності товару. Результати підсумку представлені у таблиці 4.18.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Потреба | Вигода, яку пропонує товар | Ключові переваги перед конкурентами |
| 1 | Безпечне розмежування доступу до ресурсів | Безпечні механізми здійснення авторизації, аутентифікації та аудиту | Незалежність від апаратної частини. |
| 2 | Спостереження за даними аудиту | Проведення аудиту з перевірками на базі отриманих логів | Високий рівень безпеки проведення аудиту |
| 3 | Аналіз даних аудиту | Механізм аналізу даних аудиту за заданими характеристиками | Наявність механізму аналізу даних аудиту за заданими характеристиками |

Була розроблена трирівнева маркетингова модель товару, опис якої приведений у таблиці 4.19.

Таблиця 4.19 – Опис трьох рівнів моделі товару

|  |  |
| --- | --- |
| 1 | 2 |
| Рівні товару | Сутність та складові |
| І. Товар за задумом | Безпечний механізм забезпечення аутентифікації, авторизації та аудиту за допомогою протоколу RADIUS |

Кінець таблиці 4.19

|  |  |  |
| --- | --- | --- |
| 1 | 2 | |
| ІІ. Товар у реальному виконанні | Властивості/характеристики | Вр/Тх/Тл/Е/Ор |
| Швидкість роботи | Тх/Тл |
| Захищеність користувацьких даних | Тх/Тл |
| Масштабованість | Тх/Тл |
| Складність налаштування | Тх/Тл |
| Залежність від апаратного середовища | Тх/Тл |
| Реклама | Вр |
| Якість:  Відповідність стандартам RFC 2865, 2866, 3579, 2989 | |
| Марка: Оновлена версія протоколу RADIUS, EAP-MD5 | |
| ІІІ. Товар із підкріпленням | До продажу: пропозиція використання тестового варіанту у тестовий період часу | |
| Після продажу: розробка та проведення маркетингової компанії, надання тестової версії разом з іншими сервісами, які необхідні при роботі у розподіленій мережі | |
| За рахунок чого потенційний товар буде захищено від копіювання:  Захист інтелектуальної власності | | |

Далі необхідно визначити цінові межі, якими необхідно керуватись при встановленні ціни на потенційний товар. Для цього необхідно провести аналіз ціни на товари-аналоги та аналіз рівня доходів цільової групи споживачів. Результати цього аналізу представлені у таблиці 4.20.

Таблиця 4.20 – Визначення меж встановлення ціни

|  |  |  |  |
| --- | --- | --- | --- |
| Рівень цін на товари-замінники | Рівень цін на товари-аналоги | Рівень доходів цільової групи споживачів | Верхня та нижня межі встановлення ціни на послугу |
| 1500 ум.од. | 2000 ум.од. | Середній | 1000 ум.од. – 2000 ум.од. |

Далі необхідно визначити оптимальну систему, в межах якого приймається рішення. Формування системи збуту описане в таблиці 4.21.

Таблиця 4.21 – Формування системи збуту

|  |  |  |  |
| --- | --- | --- | --- |
| Специфіка закупівельної поведінки цільових клієнтів | Функції збуту, які має виконувати постачальник товару | Глибина каналу збуту | Оптимальна система збуту |
| Рішення про закупівлю продукту приймається як правило при побудові нової внутрішньої мережі або при розширенні існуючої | Проведення маркетингової компанії.  Аналіз потреб споживачів.  Пошук потенційних партнерів та споживачів | Однорівневий канал збуту | Залучення сторонніх посередників |

Останнім кроком розробки маркетингової програми необхідно розробити концепцію маркетингових комунікацій, яка описана у таблиці 4.22.

Таблиця 4.22 – Формування системи збуту

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Специфіка поведінки цільових клієнтів | Канали комунікації, якими користуються цільові клієнти | Ключові позиції, обрані для позиціонування | Завдання рекламного повідомлення | Концепція рекламного звернення |
| Користування довіреними та перевіреними застосунками, що динамічно адаптуються під потреби користувача. | Реклама  Брендінг  Стимулювання збуту | Безпечне та сучасне розмежування доступу | Покращити рівень довіри споживача до продукту | Акцентування уваги на сучасному та безпечному застосунку |

**Висновки до розділу 4:**

В результаті проведеного аналізу першого етапу розроблення стартап проекту було визначено, що попит на даний проект наявний, а проект може динамічно підлаштовуватися під потреби ринку, а отже існує можливість ринкової комерціалізації проекту.

У запропонованого стартап проекту є доволі широкий обсяг клієнтів. Незважаючи на високі бар’єри входження та значну конкуренцію у даній галузі, проект має добрі перспективи впровадження завдяки широкому обсягу послуг та різноманітності клієнтів, а також здатності адаптуватися під специфічні потреби клієнтів.

У якості альтернативи розробки стартап-проекту доцільно обрати варіант з використанням запропонованого рішення у якості складової більш крупного проекту або варіант розвитку проекту як аналізатора даних аудиту протоколу RADIUS.

Враховуючи всі вище зазначені факти подальша імплементація проекту є доцільною.

**ВИСНОВКИ**

В ході виконання магістерської роботи були ретельно досліджені існуючи проблеми безпеки в сучасних реалізаціях протоколів автентифікації, авторизації та аудиту та розглянуті можливі методи покращення рівня безпеки для запобігання відомим атакам на протокол RADIUS. Розроблена програмна реалізація запропонованого покращеного алгоритму EAP-MD5 мовою програмування Java, що робить дану програмну реалізацію платформо незалежною.

Також були запропоновані загальні поради та встановлені вимоги, які повинні виконуватися при розробці та імплементації будь-яких реалізацій протоколу RADIUS для його безпечного використання.

Був проведений аналіз стійкості отриманої практичної реалізації протоколу за допомогою інструменту ProVerif.

А також був проведений аналіз можливості подальшої комерціалізації розробленого рішення.

**ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ**

1 RFC 6151. Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms. [Електронний ресурс] – Режим доступу: <https://tools.ietf.org/html/rfc6151>

2 RFC 2865. Remote Authentication Dial In User Service (RADIUS). [Електронний ресурс] – Режим доступу: <https://tools.ietf.org/html/rfc2865>

3 RFC 2903. Generic AAA Architecture. [Електронний ресурс] – <https://tools.ietf.org/html/rfc2903>

4 RFC 2866. RADIUS Accounting. [Електронний ресурс] – <https://tools.ietf.org/html/rfc2866>

5 RFC 2989. Criteria for Evaluating AAA Protocols for Network Access. [Електронний ресурс] – <https://tools.ietf.org/html/rfc2989>

6 Design in the Authentication and Billing System Based on Radius and 802.1x Protocol / Chaoyi Chen, Jianyong Zhang, Junli Liu., 2015. – 6c. - (International Symposium on Computers & Informatic) – [Текст]

7 RFC 3579. RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP). [Електронний ресурс] – <https://tools.ietf.org/html/rfc3579>

8 RFC 4086. Randomness Requirements for Security. [Електронний ресурс] – <https://tools.ietf.org/html/rfc4086>

9 RFC 4962. Guidance for Authenticaton, Authorization, and Accounting (AAA) Key Management. [Електронний ресурс] – <https://tools.ietf.org/html/rfc4962>

10 An Analysis of the RADIUS Authentication Protocol. [Електронний ресурс] – <https://www.untruth.org/~josh/security/radius/radius-auth.html>

11 Security Analysis of MD5 Algorithm in Password Storage / Mary Cindy Ah Kioon, ZhaoShun Wang, Shubra Deb Das., 2013. – 4 c. – (Atlantis Press) – [Текст]

12 RADIUS Configuration Guide, Cisco IOS Release. [Електроний ресурс] - <https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/sec_usr_rad/configuration/15-mt/sec-usr-rad-15-mt-book/sec-cfg-radius.html?referring_site=RE&pos=2&page=https://www.cisco.com/c/en/us/support/docs/security-vpn/remote-authentication-dial-user-service-radius/12433-32.html>

13 How to Break MD5 and Other Hash Functions / Xiaoyun Wang, Hongbo Yu., - 17 c. – [Текст]

14 Comparison of the RADIUS and Diameter Protocols / Mladen Stanke, Mile Sikic., 2008. – 7c. – [Текст]

**ДОДАТОК А**

**Функція main() реалізації клієнта RADIUS**

**public static void** main(String[] args)  
 **throws** Exception {  
 BasicConfigurator.*configure*();  
 **if** (args.**length** != 4) {  
 System.***out***.println(**"Usage: TestClient hostName sharedSecret userName password"**);  
 System.*exit*(1);  
 }  
  
 String host = args[0];  
 String shared = args[1];  
 String user = args[2];  
 String pass = args[3];  
  
 RadiusClient rc = **new** RadiusClient(host, shared);  
  
 AccessRequest ar = **new** AccessRequest(user, pass);  
 ar.setAuthProtocol(AccessRequest.***AUTH\_EAP***);  
 ar.addAttribute(**"NAS-Identifier"**, **"this.is.my.nas-identifier.de"**);  
 ar.addAttribute(**"NAS-IP-Address"**, **"192.168.0.100"**);  
 ar.addAttribute(**"Service-Type"**, **"Login-User"**);  
 ar.addAttribute(**"WISPr-Redirection-URL"**, **"http://www.sourceforge.net/"**);  
 ar.addAttribute(**"WISPr-Location-ID"**, **"net.sourceforge.ap1"**);  
 ar.addAttribute(**"EAP-Message"**, **"1"**);  
  
 System.***out***.println(**"Request from client:\n"** + ar);  
  
 RadiusPacket response = rc.authenticate(ar);  
 System.***out***.println(**"First response from server:\n"** + response);  
  
 AccessRequest requestWithTimestamp = **new** AccessRequest(user, pass);  
 RadiusPacket resultPacket = rc.authenticateEapRequest(*generateTimestampPacket*(response, requestWithTimestamp));  
 System.***out***.println(**"Second response from server:\n"** + resultPacket);  
  
 rc.close();  
}

**ДОДАТОК Б**

**Основні методи здійснення аутентифікації за допомогою EAP-MD5 класу EAPAuthenticator**

**public** RadiusPacket processRequest(RadiusPacket p, String password) {  
 **byte**[] challenge = createChallenge(p.getAttributeValue(**"User-Name"**));  
 **byte**[] encodedChallenge = encodePassword(challenge, password);  
 p.addAttribute(**"EAP-Message"**, encodedChallenge);  
 **return** p;  
}

**private byte**[] createChallenge(String username) {  
 **byte**[] challenge = **new byte**[16];  
 *random*.nextBytes(challenge);  
 *challenges*.put(username, challenge);  
 **return** challenge;  
}

**private byte**[] encodePassword(**byte**[] challenge, String password) {  
 **byte**[] eapHash = encodeChallenge(challenge);  
 **byte**[] passwordArray = **new byte**[eapHash.**length**];  
 System.*arraycopy*(  
 RadiusUtil.*getUtf8Bytes*(password),  
 0,  
 passwordArray,  
 0,  
 RadiusUtil.*getUtf8Bytes*(password).**length**);  
 **byte**[] responseHash = **new byte**[eapHash.**length**];  
 **for** (**int** i = 0; i < eapHash.**length**; i++) responseHash[i] = (**byte**) (eapHash[i] ^ passwordArray[i]);  
 **return** responseHash;  
}

**private byte**[] encodeChallenge(**byte**[] challenge) {  
 MessageDigest md5 = getMd5Digest();  
 md5.reset();  
 md5.update(RadiusUtil.*getUtf8Bytes*(getId()));  
 md5.update(challenge);  
 **return** md5.digest();  
}

**public static byte**[] encodeTimestampPacket(**byte**[] hashedChallenge, **int** timestamp) {  
 **byte**[] timeArray = ByteBuffer.*allocate*(4).putInt(timestamp).array();  
  
 EAPAuthenticator eapAuthenticator = **new** EAPAuthenticator();  
 MessageDigest md5 = eapAuthenticator.getMd5Digest();  
 md5.reset();  
 md5.update(hashedChallenge);  
 md5.update(timeArray);  
 **return** md5.digest();  
}

**ДОДАТОК В**

**Скрипт для тестування алгоритму за допомогою ProVerif**

type userId.

type challenge.

type timestamp.

type password.

free c : channel.

free id: userId.

free pass: password[private].

free time: timestamp.

fun md5Challenge (userId, challenge) : bitstring.

fun encPass(bitstring, password) : bitstring.

fun encTimestamp (bitstring, timestamp): bitstring.

fun md5ChallengePassword (bitstring): bitstring.

table d(userId, bitstring).

reduc forall m: bitstring, k: password; decPass(encPass(m, k), k) = m.

reduc forall m: bitstring, k: timestamp; decTimestamp(encTimestamp (m, k), k) = m.

event getId(userId).

event getChallenge(bitstring).

event getTimestamp(timestamp).

event authenticate().

query attacker(pass).

query gotTimestamp: timestamp;event(authenticate()) ==> event(getTimestamp(gotTimestamp)).

let clientA () =

out(c, id);

in(c, hidePass: bitstring);

out(c, encTimestamp(hidePass, time)).

let serverB () =

event getId(id);

in(c, XId: userId);

new chall: challenge;

out(c, encPass(md5Challenge(XId, chall), pass));

in(c, clientTimestamp: bitstring);

event getTimestamp(time);

let encChallengePassword = decTimestamp(clientTimestamp, time) in

if encChallengePassword = md5ChallengePassword(encPass(md5Challenge(XId, chall), pass)) then event authenticate();

0.

process

((!clientA) | (!serverB))