Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Програмної інженерії_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження методів захисту клієнт-серверної взаємодії від атак виду CSRF
(тема)

Виконав: студент 2 курсу, групи ПЗСм-19-1
спеціальності 121- Інженерія програмного забезпечення
(код і повна назва спеціальності)
Освітньо-професійної програми
Програмне забезпечення систем
(повна назва освітньої програми)

Жестовський С.М.
(прізвище, ініціали)
Керівник д.т.н., проф. Четвериков Г.Г.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2020 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121-Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-професійна програма

Освітня програма Програмне забезпечення систем
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 2020 р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Жестовському Сергію Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів захисту клієнт-серверної взаємодії від атак виду CSRF
затверджена наказом по університету від "___" _____2020 р. № _____
2. Термін подання студентом роботи до екзаменаційної комісії
"___" _____2020 р
3. Вихідні дані до роботи *види мережевих атак на клієнт-серверну взаємодію, методи та алгоритми захисту від атак виду CSRF, різновиди реалізації атаки виду CSRF, показники та метрики серверної частини при клієнт-серверній взаємодії, сервери на базі мікросервісної архітектури, функціональні характеристики взаємодії з використанням методів захисту, розробка алгоритмів захисту для клієнт-серверної взаємодії, середовище розробки Visual Studio Code, мова програмування Node.js.*
4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, огляд видів атак та реалізації виду атаки CSRF на клієнт-серверну взаємодію, огляд методів захисту від атак виду CSRF, визначення критеріїв оцінки ефективності методів захисту, аналіз методів захисту щодо серверів із мікросервісною архітектурою, проектування та реалізація алгоритмів захисту, методи оцінки та вимірювання критеріїв ефективності для методів захисту із застосуванням серверів із мікросервісною архітектурою, дослідження ефективності методів захисту та визначення найбільш ефективних алгоритмів, опис можливих застосувань.*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *принцип роботи атаки виду CSRF, схеми існуючих рішень, мікросервісна архітектура, схеми розроблених алгоритмів, діаграми активності, результати досліджень, результати тестування, методи підрахунку ефективності, фрагменти реалізації алгоритмів захисту, фрагменти реалізації тестових випадків.*

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Снецчастина	д.т.н., проф. Четвериков Г.Г.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка*
1.	Аналіз предметної галузі	12.03 – 06.04	виконав
2.	Огляд існуючих видів атак на клієнт-серверну взаємодію	07.04 – 27.05	виконав
3.	Аналіз існуючих методів захисту	28.05 – 13.06	виконав
4.	Розробка методів захисту від атак виду CSRF	14.06 – 04.08	виконав
5.	Підготовка пояснювальної записки	13.09 – 29.10	виконав
6.	Спецчастина	04.11.2020	виконав
7.	Підготовка презентації та доповіді	06.11 – 23.11	виконав
8.	Попередній захист	09.12.2020	виконав
9.	Нормоконтроль, рецензування	10.12.2020	виконав
10.	Занесення диплома в електронний архів		
11.	Допуск до захисту у зав. кафедри		

Дата видачі завдання "___" _____2020 р.

Студент _____
(підпис)

Керівник роботи _____ д.т.н., проф. Четвериков Г.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи: 98 с., 22 рис., 9 табл., 31 джерел.

АЛГОРИТМИ ЗАХИСТУ, ЕФЕКИВНІСТЬ, КЛІЄНТ-СЕРВЕРНА ВЗАЄМОДІЯ, МЕРЕЖЕВІ АТАКИ, МЕТОДИ ЗАХИСТУ, МІКРОСЕРВІС, CSRF.

Об’єктом дослідження є методи та алгоритми захисту клієнт-серверної взаємодії від мережевих атак на прикладі атак виду CSRF, а також методів захисту для серверів на базі мікросервісної архітектури.

Метою роботи є визначення ефективності існуючих методів та алгоритмів захисту від атак виду CSRF, а також пропонування, розробка та реалізація нових методів захисту для покращення клієнт-серверної взаємодії серверів із мікросервісною архітектурою.

Методи розробки базуються на технологіях JavaScript, Node.js, баз даних PostgreSQL та Redis, середовищі розробки Visual Studio Code.

Результатом роботи є розробка та реалізація методів захисту від атак виду CSRF для серверів переважно із мікросервісною архітектурою, аналіз ефективності методів та алгоритмів захисту від атак виду CSRF із розробленими включно.

CLIENT-SERVER INTERACTION, CSRF, EFFICIENCY, MICROSERVICE, NETWORK ATTACKS, PROTECTION ALGORITHMS, PROTECTION METHODS.

The object of research is methods and algorithms for protection of client-server interaction from network attacks based of CSRF type of attacks, as well as methods for protection of servers based on microservice architecture.

The aim of the work is to determine the effectiveness of existing methods and algorithms for protection against CSRF attacks, as well as to propose, develop and implement new protection methods to improve client-server interaction for servers with microservice architecture.

Development methods are based on JavaScript, Node.js, PostgreSQL and Redis databases, Visual Studio Code development environment.

The result is the development and implementation of methods of protection against CSRF attacks for servers mainly with microservice architecture, analysis of the effectiveness of methods and algorithms for protection against CSRF attacks include developed methods.

ЗМІСТ

ВСТУП

Ріст Інтернет технологій неприпинний. Кожен день, та швидкість, з якою прогресують мережеві технології та з’являються нові методи розробки та реалізації програмних продуктів, неупинно рухається угору. Та з їх ростом також збільшуються кількість веб-систем, розширюються вже існуючі веб-додатки. Такий бурний ріст технологій не обходять й злочинці, яких приваблюють слабко захищені або зовсім беззахисні веб-системи.

Факт слабкого захисту існуючих веб-сервісів доводить велика щоденна кількість успішних випадків атак на Інтернет-ресурси по всьому світі. Така поточна ситуація в мережі спричинена слабкою а зовсім відсутньою реалізацією та впровадженням захисних методів для веб-систем.

Однією з таких видів атак є й CSRF (Cross Site Request Forgery), атака, що реалізує міжсайтову підробку запиту, у результаті якої користувач може втратити персональні дані чи навіть кошти на електронних рахунках. Незважаючи на дуже великі шкідливі наслідки від даної атаки, її реалізація не потребує великих знать на вмінь та кожен користувач мережі Інтернет може в такий спосіб завдати шкоди для веб-сервісу чи для звичайного користувача.

Для припинення мережевих атак на клієнт-серверну взаємодію розробляється велика кількість методів захисту, і для атак виду CSRF такі захисні методи не нові. Однак існує ряд проблем для для наявних методів захисту:

– методи захисту не надають стовідсоткової гарантії захисту, існують способи обходу для окремих алгоритмів;

– не усі алгоритми однаково ефективні, як для серверів із монолітною архітектурою, так і для мікросервісної архітектури;

– певні алгоритми сприяють значному погіршенню швидкості передачі даних;

– деякі алгоритмі значною мірою збільшують навантаження на серверну частину веб-системи.

Так, з урахуванням усіх перерахованих проблем, дуже складно виділити найкращий метод захисту, що буде захищати від усіх розповсюджених видів реалізації атаки виду CSRF та не навантажувати клієнт-серверну взаємодію.

Потрібно сформулювати актуальність досліджень в даній області:

– необхідність виділити основне рішення методу захисту вид атак виду CSRF;

– необхідно проаналізувати та визначити ефективність для основних методів захисту;

– потреба в аналізі та встановленні найкращих алгоритмів для серверів з монолітною та мікросервісною архітектурами;

– потреба у розвантажені серверів з мікросервісною архітектурою з використанням основних методів захисту;

– постійний розвиток Інтернет-технологій, що сприяє збільшенню кіберзлочинності;

– відсутність універсального рішення для захисту від атак виду CSRF.

Метою роботи є дослідження існуючих методів та алгоритмів захисту від атак виду CSRF для серверів з монолітною та мікросервісною архітектурами, та запропонування окремного алгоритму для полегшення клієнт-серверної взаємодії для мікросервісної архітектури. А також оцінка ефективності для основних методів захисту з виявленням найкращого за заданих умов проведених тестувань.

Об’єктом дослідження є процеси захисту від різновидів реалізації атак виду CSRF на клієнт-серверну взаємодію.

Предметом дослідження є методи та алгоритми захисту, існуючі та запропоновані в процесі аналізу, і критерії ефективності, що становлять основу оцінки ефективності алгоритмів.

В ході даної роботи задля дослідження ефективності методів та алгоритмів захисту від атак виду CSRF, а також запропонованих алгоритмів застосовувались емпіричні методи програмної інженерії, порівняльний метод та оцінка з використанням регресивного аналізу. Було визначено поняття ефективності для методів та алгоритмів захисту, та методи досліджень з використанням тестових

випадків для встановлення оцінки критеріям для кожного окремого методу. Було розроблено алгоритми захисту на основі сторонніх сервісів для покращення ефективності роботи методів захисту для серверів з мікросервісною архітектурою.

Наукова новизна даної роботи полягає у формалізації поняття ефективності для методів та алгоритмів захисту з використанням серверів з мікросервісною архітектурою, створенні алгоритмів захисту від атак виду CSRF на базі сторонніх сервісів, встановлення найбільш ефективних алгоритмів захисту за заданими параметрами досліджень тестових випадків.

Результати даної роботи мають практичне значення для розробників серверних частин веб-систем при виборі алгоритмів та методів захисту від атак. Відштовхуючись від обраної серверної архітектури можна встановити найкращий варіант захисного методу для певного сервера.

Стислі положення результатів аналізу даного дослідження було опубліковано в матеріалах конференції МНЛ, Актуальні питання та перспективи проведення наукових досліджень від 06.11.2020.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

На сьогоднішній день в мережі Інтернет налічують більше ніж 1.74 мільярда веб-сайтів по всьому світу, більше ніж 40% від усього населення світу використовує Інтернет, у середньому користувачі мережі кожного дня витрачають до 6 з половиною годин в Інтернеті [1]. З таким високим темпом росту популярності Інтернету зростає і ймовірність зіштовхнутися з кіберзлочинністю: більше ніж 80,000 випадків кібератак скоюються кожен день, 93% витоку даних відбувається за лічені хвилини, а 83% витоку не виявляються тижнями, вартість збитків, спричинених кіберзлочинністю, коштуватиме 6 трильйонів доларів під кінець 2020 року проти трильйонів доларів лише рік тому [2].

З ростом кіберзлочинності, однак, зростає й комп’ютерна безпека, розробляються нові методи захисту клієнтів з мережевими системами, вдосконалюються існуючі алгоритми. Внаслідок розширення можливостей для користувачів Інтернету, збільшуються й ризики, адже додавання нових технологій неодмінно приводить до знаходження вразливих місць, чим й користуються злочинці. На даний момент нараховується до 23-ох основних видів комп’ютерних загроз та близько 4000 основних типів кібератак [2]. Від такої великої кількості потенційних загроз не можливо захиститися єдиним загальним способом, в такій ситуації виникає потреба захищатись від кожної загрози окремо. Але перед початком захисту потрібно виявити вразливість, яка притаманна конкретному програмному продукту.

В мережі Інтернет також існують й такі загрози, які не поділяються між різними типами програмних продуктів, а притаманна усім програмним рішенням, що знаходяться у мережі. Це пов’язано з характером загрози, яка використовує вразливості самої мережі для скоєння кібератак. Існує велика кількість загроз для звичайного веб-сайту, але найбільш розповсюджені з них також й найбільш ефективні з точки зору застосування. Такими загрозами являються та не обмежуються:

– MitM (Man-in-the-middle) – атака виду «людина посередині» реалізовується втручанням у лінію передачі даних між користувачами [3]. Таким чином зловмисник може бачити усі дані, що передаються від одного користувача до іншого;

– Phishing attack – практика надсилання електронних листів з надійних джерел з метою отримання особистої інформації або спонукання користувачів до будь-яких дій [4].

– XSS (Cross-site scripting) – атака на веб-сайт, що впроваджує вірусний код на сторінки сайту, які переглядає користувач [5]. При відкритті зараженої сторінки, вірусний код передає особисті дані користувача зловмиснику.

– CSRF (Cross Site Request Forgery) – атака, яка використовує користувацькі дані задля виконання запитів до сайтів на яких зареєстрований користувач з виконанням користувацьких дій [6-13].

– Birthday attack – атака, що направлена на хеш-алгоритми шифрування, де повідомлення, що відправляються користувачам, підписуються з використанням хеш-алгоритму. Злочинець може підробити повідомлення таким чином, що хеш-підпис не зміниться, тим самим успішно перетинаючи систему перевірки повідомлень [14].

На практиці таких або подібних їм загроз дуже багато, та для розглядання методів захисту від них потрібно обрати конкретний вид атаки.

1.1 Опис проблеми

Атака виду CSRF (міжсайтова підробка запиту) – це тип зловмисного використання веб-сайту, де несанкціоновані команди подаються від користувача, якому веб-система, що підтверджена атаці, довіряє [6-13].

Це означає, що даний вид атаки націлений на веб-систему через її довірених користувачів. Основним прикладом такої атаки слугує ситуація, де користувач

переходить на заздалегідь підготовлений злочинцем веб-сайт (надалі вірусний сайт), де знаходиться спеціальний запит до веб-сервісу, який являється ціллю для злочинця (надалі цільовий сайт). При вході жертви до вірусного сайту, виконується вірусний запит, що надсилається до цільового сайту (див. рис. 1.1).

Тут треба зауважити метод роботи браузерів в даній ситуації, який є одним із ключових місць вразливості при атаці даного виду: при відправці запиту, браузер підписує пакети, що передаються, користувацькими даними, які були видані веб-сервером (також називаються cookies).

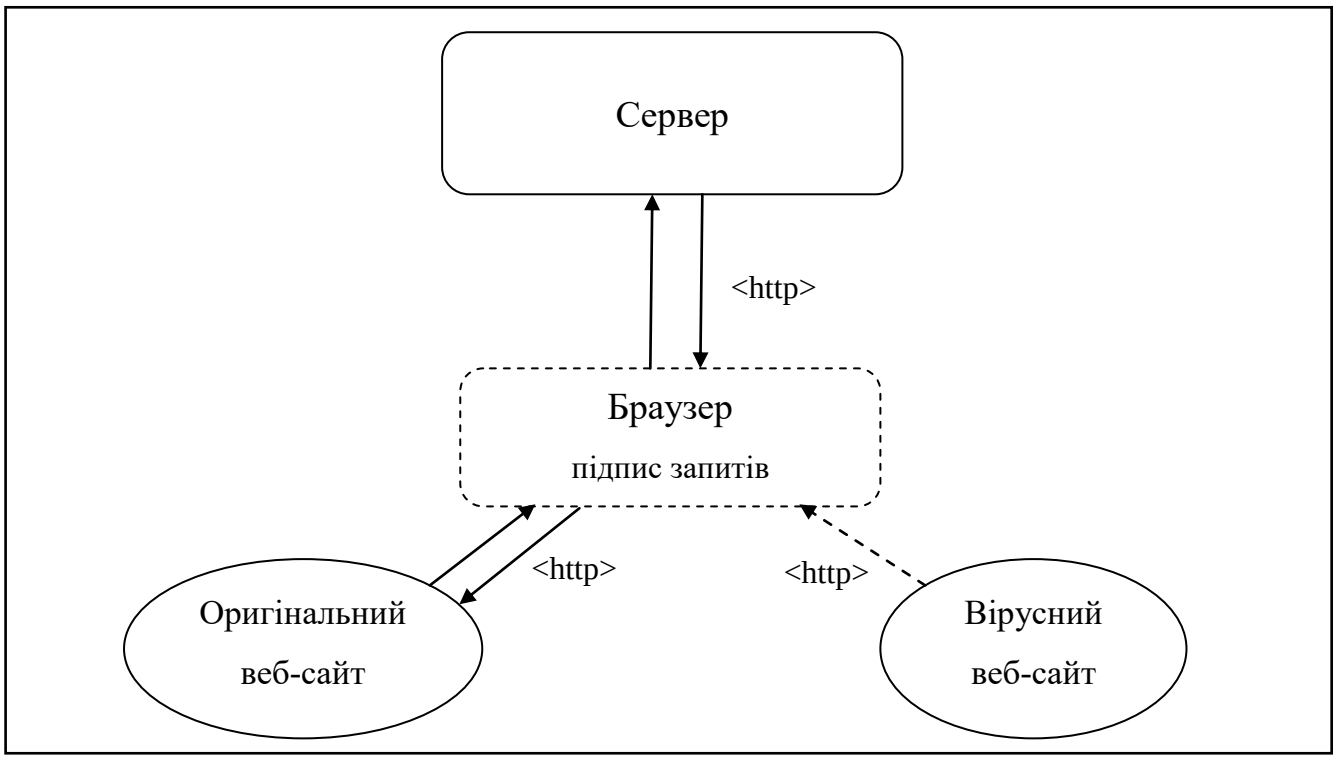

Рисунок 1.1 – схема передачі запитів до цільового сервера

Cookie-файли – це персональні користувацькі дані та додаткові серверні налаштування щодо користувача, які зберігаються у браузері. Такі дані видаються веб-сервісами та передаються браузерам, де оброблюються сторінкою веб-сайту та зберігаються у браузері. Такими даними підписуються усі пакети, що відправляються до цільового веб-сервера.

Основним недоліком такої взаємодії є підпис і відправка запитів з інших веб-сайтів, чим і користуються зловмисники. При надсиланні такого пакету

даних, серверна частина веб-сервісу вважає, що дані були надіслані користувачем, адже вони підписані клієнтськими cookie-файлами завдяки браузеру.

Зауважимо, що вірусний сайт може бути яким-завгодно та не мати наміру підробити оригінальний веб-сайт, скопіювавши зовнішній вигляд оригіналу.

При виконані запиту на цільовий сервер з вірусного сайту, сервер виконує жадану операцію від особи користувача, що може привести до серйозних наслідків, від втрати персонального кабінету чи грошей на рахунку до втрати цілого веб-сервісу. Наслідки вимірюються можливостями самого користувача, що став жертвою атаки виду CSRF.

Проте на даний вид атаки існують свої методи захисту, які направлені на з’ясування оригінальності надісланого запиту. Важливо розуміти, що такі методи захисту знаходяться лише на серверній частині та не при яких умовах не довіряються користувачу веб-сервісу.

1.2 Опис елементів захисту

На даний момент існують декілька методів захисту від даного виду атаки. Такі методи різняться підходами жаданими результатами при захисті серверної частини веб-систем та сервісів. З-поміж усіх підходів до захисту загальними рисами є елементи захисту – основні типи даних чи частини алгоритму, які являються базовою ідеєю, на яку опираються при побудові методів захисту від атак.

Слугуючи важливою частиною механізмів захисту, такі елементи не можливо не розглянути окремо. Такими основними елементами для захисту від атак виду CSRF є: веб-токени, захисні cookie-файли та проксі-сервери. Кожен з цих механізмів надає свій відповідний рівень захисту, та потребує детального перегляду перед використанням.

1.2.1 Захисні файли cookie

Загалом, cookie-файли – це дані специфічні до кожного користувача, що зберігаються на стороні клієнта. Такі дані можуть бути доступні на веб-сайті, що може бути небезпечним. Для безпечного зберігання даних у cookie-файлах потрібно налаштовувати їх з Http-only прапорцем [15], що захищає дані втручання зі сторони браузера. Такий вид взаємодії дається певною ціною – тепер веб-сайт, що колись міг опиратись на такі дані, більше не зможе їх використовувати. Помічаючи певні cookie-файли Http-only прапорцем можна забезпечити певний рівень безпеки користувацьких даних, особисто від атак виду XSS [5].

1.2.2 Захисні веб-токени

Захисні веб-токени, також авторизаційні токени чи токени доступу – тип даних у строковому форматі, що зберігає у собі інформацію про користувача, якому був надай токен, та підпису, що підтверджує оригінальність даних. Такий токен може створити для користувача та перевірити лише цільовий веб-сервер на основі секретного ключа.

Розповсюдженим прикладом такого веб-токена є JWT (JSON Web Token) – відкритий стандарт [16] для створення токенів доступу, заснований на форматі JSON [17]. Як правило, використовується для передачі даних для автентифікації в клієнт-серверних додатках.

Зазвичай такі токени зберігають корисну інформацію про користувача, що дозволяє їм бути унікальним ідентифікатором сесії – серверна частина не повинна зберігати додаткової інформації про користувацьке з’єднання, а лише перевіряти підпис на токені та використовувати підписану інформацію, як коректну. Треба

розуміти, що при такому підході токен доступу потрібно постійно оновлювати для більш безпечного з’єднання.

Для захисних токенів існують стандартні вимоги [18] задля безпечної передачі даних:

– унікальний токен для кожної операції;

– єдино разової дії;

– має розмір підпису, стійкий до підбору;

– генерований криптографічно-стійким генератором псевдовипадкових чисел;

– має обмежений час життя, після чого стає недійовим.

Захисні веб-токени можуть зберігатись у cookie-файлах чи передаватись безпосередньо на зберігання до веб-сайту. Два такі підходи супроводжуються рядом власних переваг та недоліків: з одного боку при зберіганні в безпечних cookie-файлах токену не загрожують атаки спрямовані на викрадення даних, з іншого ж боку – до токена не можливо дібратись зі сторони веб-сайту. Такий підхід обмежує веб-сайту доступ до користувача, і сайт не в змозі коректно відображати певну інформацію, що стосується певного клієнта. При зберіганні токена безпосередньо на веб-сайті (наприклад, з використанням LocalStorage [19]) у самої веб-сторінки з’являється можливість отримати інформацію про поточного користувача.

1.2.3 Захисні проксі-сервери

Сам по собі проксі-сервер – це проміжний вузол між користувацьким додатком та веб-сервером при клієнт-серверній взаємодії [20]. Такий сервер перенаправляє запити двох сторін, слугуючи точкою обміну. Це може бути корисно і при побудові системи захисту від кібератак, адже такий сервер може обробляти усі запити із застосуванням методів захисту та перевірки коректності

запиту. В такому випадку цільовий сервер може не піклуватись про деталі захисту для кожного клієнтського запиту.

Одним з прикладів таких проксі-серверів є зворотний проксі-сервер. Зворотний проксі-сервер – це тип проксі-сервера, який ретранслює запити клієнтів із зовнішньої мережі на один або кілька серверів, логічно розташованих у внутрішній мережі [21]. При цьому для клієнта це виглядає так, ніби запитувані ресурси знаходяться безпосередньо на одному сервері. На відміну від класичного проксі-сервера, який перенаправляє запити клієнтів до будь-яких серверів в Інтернеті і повертає їм результат, зворотний проксі-сервер безпосередньо взаємодіє лише з асоційованими з ним серверами і повертає відповідь тільки від них.

1.3 Опис методів та алгоритмів захисту

Для повного захисту від кібератак на веб-сервіси використовують різноманітні методи та алгоритми в основі яких лежать елементи захисту, описані вище. Для захисту клієнт-серверної взаємодії від атак виду CSRF існує декілька методів та алгоритмів, описаних нижче. Кожен окремий метод вирішує основну проблему захисту мережевого з’єднання, однак має певні недоліки, та для більш чіткого розуміння потрібно розглянуть кожен з відомих методів захисту окремо.

1.3.1 Синхронний токен

Метод синхронного токена використовує згенерований персональний токен до кожної сесії для розпізнавання користувача. Такий токен, як і сам ідентифікатор сесії, зберігається на серверній частині.

Алгоритм даного методу такий (див. рис. 1.2):

– при старті сесії на стороні сервера генерується токен;

– токен кладеться в сховище даних сесії (тобто зберігається на стороні сервера для подальшої перевірки запитів);

– у відповідь на запит, який стартував сесію, клієнту повертається токен;

– при повторних запитів клієнт зобов'язаний передати токен до сервера для перевірки;

– при отримання запиту сервер зобов'язаний перевірити на ідентичність токена, що зберігається поряд із сесією і токена, який надіслав клієнт;

– якщо обидва токена збігаються, то вважаємо, що сервер не піддається атаці виду CSRF, в іншому випадку – відхиляємо запит.

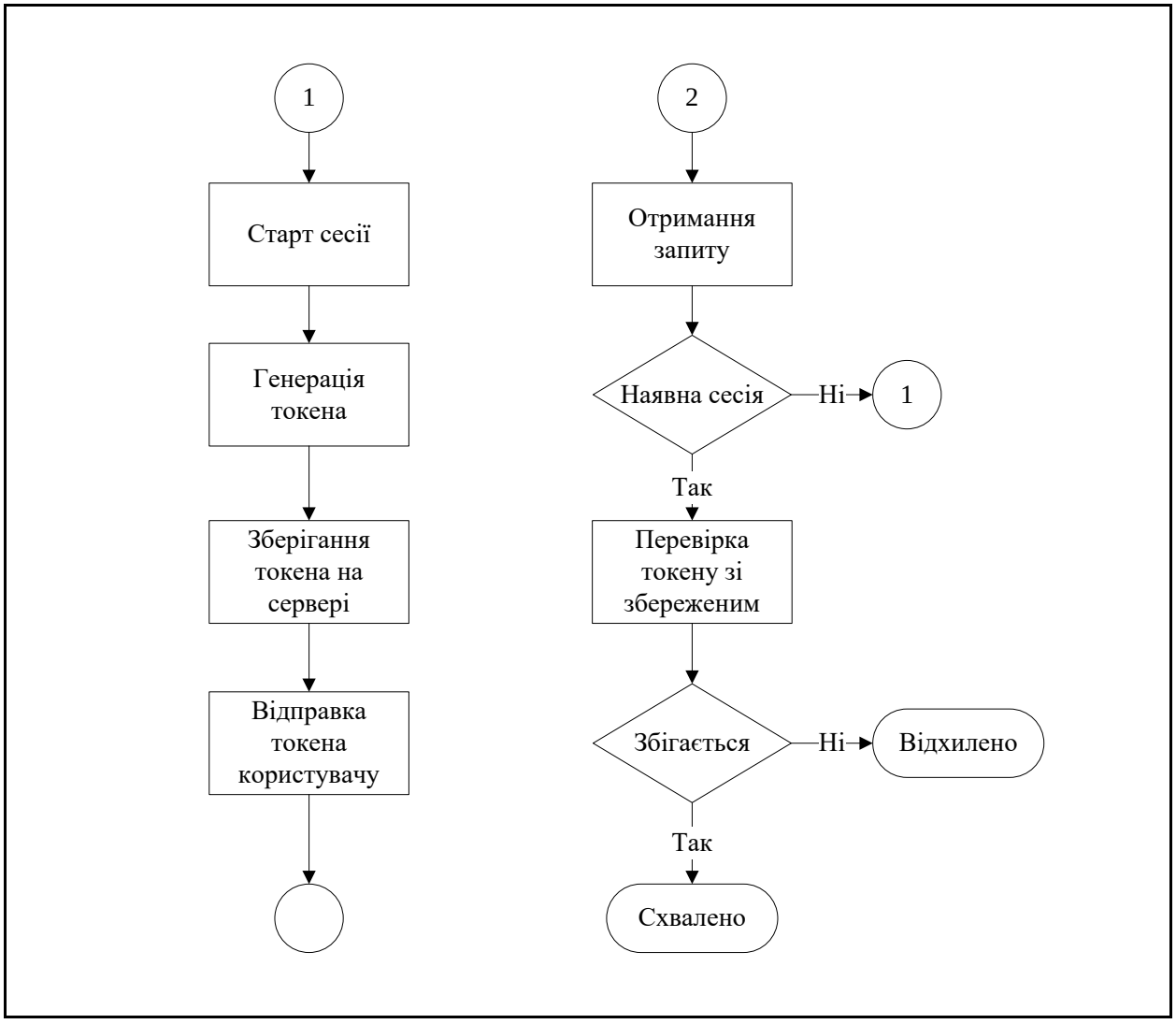

Рисунок 1.2 – Алгоритм роботи синхронного токена

Результатом такого підходу маємо: токен, який оновлюється тільки при оновлені сесії, а це відбувається, коли сесія закінчується, під час життя однієї сесії всі дії будуть перевірятися по одному токену, токен один та являється унікальним для кожного клієнта.

Недоліком такого методу є прив’язка самого токена до сесії, та якщо станеться витік токена, то зловмисник зможе виконати CSRF атаку на будь-який запит і протягом довгого терміну.

1.3.2 Cookie-файл двійної відправки

Метод cookie-файлу двійної відправки використовує двійний підпис запиту за допомогою токена для перевірки оригінальності запиту. Цей підхід не вимагає зберігання даних на стороні сервера, тим самим надаючи контроль сесії і токена до користувача. Такий підхід використовується, якщо цільовий веб-сервіс представлений у вигляді мікросервісу чи сервісу, що володіє основними рисами мікросервісної архітектури.

Ідея даного методу в тому, щоб віддати токен клієнту двома методами: в cookie-файлах і в одному з параметрів відповіді. Треба зауважити, що на cookie-файл встановлюється Http-only прапорець для коректної роботи алгоритму.

Алгоритм даного методу такий (див. рис. 1.3):

– при запиті від клієнта на стороні сервера генерується токен. У відповіді токен повертається в cookie-файлах і в одному з параметрів відповіді, найчастіше у тілі відповіді;

– у повторних запитах клієнт зобов'язаний надавати обидва отриманих раніше токена: один як cookie-файл, який зберігається з Http-only прапорцем, інший або як заголовок HTTP протоколу передачі даних, або всередині тіла даних;

– при отриманні запиту сервер зобов'язаний перевірити на ідентичність токен з cookie-файлів і токен, який явно надіслав клієнт;

– якщо обидва токена збігаються, то вважаємо, що сервер не піддається атаці виду CSRF, в іншому випадку – відхиляємо запит.

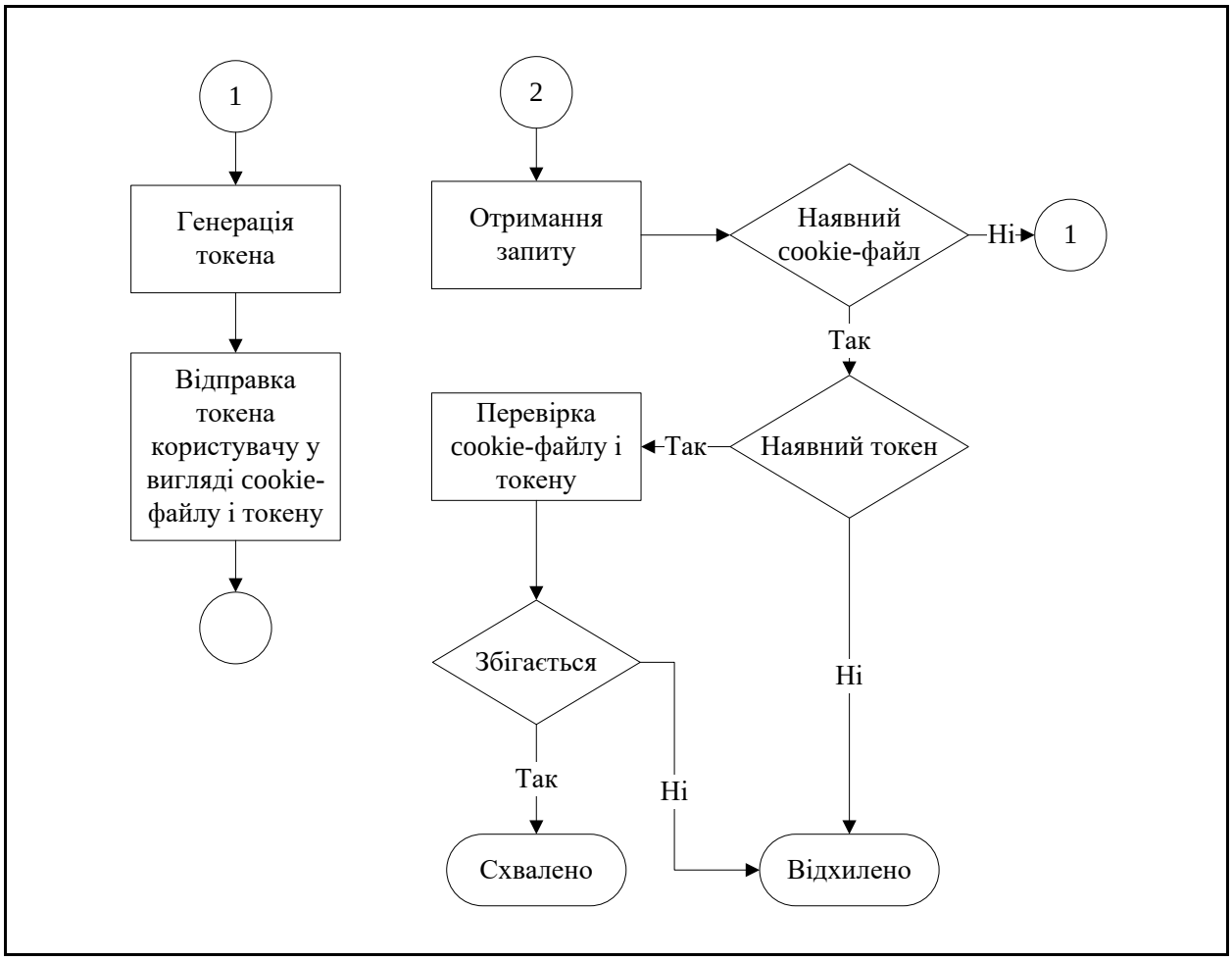

Рисунок 1.3 – Алгоритм роботи методу cookie-файлу двійної відправки

Результатом такого підходу маємо захист схожий з методом синхронного токена але без зберігання та синхронізації сесії на стороні сервера.

Необхідно враховувати, що під-домени можуть читати cookie-файли основного домену, якщо явно це не забороняти. Це може привести до спроби атаки через під-домен основного сайту. Таким чином, якщо ваш сервіс доступний на домені 3-го рівня, а зловмисник має можливість зареєструвати свій ресурс на вашому домені 2-го рівня, то встановлювати cookie-файли потрібно явно на свій домен [22].

1.3.3 Зашифрований токен

Метод зашифрованого токена цілком заснований на відкритій специфікації генерації JWT токена. Основна ідея полягає у тому, що, якщо ви зашифруєте надійним алгоритмом дані і передасте їх клієнту, то клієнт не зможе їх підробити, не знаючи ключа. Цей підхід не вимагає використання cookie-файлів, так само. Токен передається клієнту тільки в параметрах відповіді, що робить даний метод схожим з методом cookie-файл двійної відправки, де токен зберігається на стороні клієнта.

В даному підході токеном є факти, зашифровані ключем. Мінімально необхідними фактами являються: унікальний ідентифікатор клієнта чи ідентифікатор сесії, а також час створення токена. Ключ, на основі якого генерується токен, не повинен бути відомий клієнту.

Алгоритм даного методу такий (див. рис. 1.4):

– при запиті від клієнта на стороні сервера генерується токен;

– генерація токена складається в шифрування фактів, необхідних для підтвердження особи користувача;

– мінімально необхідні факти – унікальний ідентифікатор клієнта чи ідентифікатор сесії, а також час створення токена. У відповіді токен повертається в одному з параметрів відповіді;

– у повторних запитах клієнт зобов'язаний надавати отриманий раніше токен;

– при отримання запиту сервер зобов'язаний затверджувати токен, отриманий від клієнта;

– валідація токена полягає в його розшифровці і порівнянні фактів, отриманих після розшифровки, з реальними;

– якщо розшифрувати токен не вдалося, чи необхідні факти не збігаються, вважаємо, що сервер піддається атаці виду CSRF, в іншому випадку – схвалюємо запит.

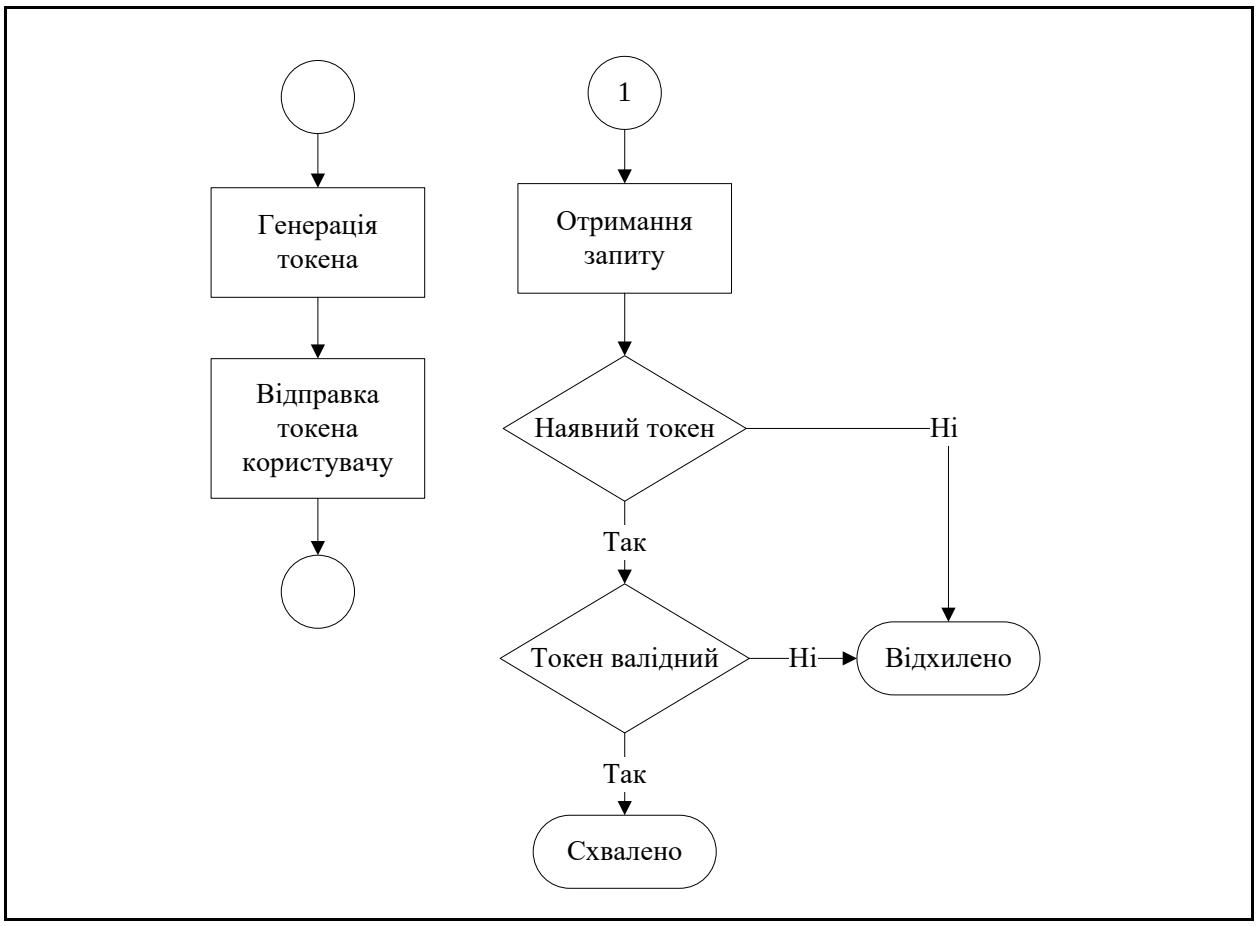

Рисунок 1.4 – Алгоритм роботи методу зашифрованого токена

Результатом такого підходу маємо захист токеном, що передається безпосередньо до клієнта та зберігається для валідації запитів. Однією з важливих особливостей є змога не зберігати токен у cookie-файлах. Також у такій реалізації немає проблем із атаками від під-доменів.

1.3.4 Заголовок авторизації

Даний метод використовує «Bearer Authentication» токен [23] для перевірки оригінальності запиту на стороні сервера. Bearer Authentication Token, або токен авторизації – це токен, який містить ідентифікаційні дані користувача. Токен розміщується у заголовку авторизації http протоколу (найчастіше у Authorization

заголовку) та зчитується на серверній стороні веб-сервісу. Алгоритм видачі такого токена ідентичний до алгоритму зашифрованого токена (див. рис. 1.4), але на відміну від нього токен розрахований безпосередньо для ідентифікації користувача, а не для перевірки запиту на оригінальність.

Даний алгоритм захищає клієнт-серверну взаємодію завдяки формі підпису запитів: клієнт може отримати токен для підпису тільки у тому випадку, якщо він доводить свою особистість (наприклад, при авторизації на веб-сайт). Зловмисник не зможе отримати такий токен і розмістити його в заголовку запиту, бо сервер не видасть такий токен авторизації для іншого, вірусного сайту.

1.3.5 Захисні налаштування cookie-файлів

Метод захисного налаштування cookie-файлів – експеримент ний вид захисту від CSRF атак, який вбудовано у самі браузери. Алгоритм захисту полягає у тому, щоб усі надані cookie-файли підписувати спеціальним SameSite прапорцем [24]. Такий прапорець дає браузерам зрозуміти, що помічені cookie-файли потрібно відправляти з цільового веб-сайту, таким чином унеможливлюючи відправлення cookie-файлів з вірусного сайту. Треба розуміти, що доступ до таких cookie-файлів можливий зі сторони цільового веб-сайту (на відміну від http-only прапорця), що може спричинити витік даних за допомогою XSS атаки.

Однак потрібно розуміти, що це нова технологія, яка покликана захистити від CSRF атак, в даний момент працює тільки в двох браузерах (Chrome, Opera). Також до того факту, що технологія нова, додається проблема особливості роботи браузерів, веб-додатків, які іноді дозволяють обходити CSRF захисту. Загалом, дана технологія ненадійна [25] та її використання на теперішній час є спірним з точки зору розробки надійної до атак системи.

1.3.6 Підписання форм-відправки

Метод підписання форм-відправки вимагає підпис кожної форми що надається користувачу. При надходженні запиту сервер перевіряє підпис форми на валідність.

Даний метод можливий лише у випадку надання самої веб-сторінки клієнту із того самого серверу, який і перевіряє валідність токену, що часто не є правдою.

Алгоритм даного методу такий (див. рис. 1.5):

– при запиті на вдачу сторінки, користувачу надається веб-сторінка з усіма наявними на ній формами. До таких форм додаються приховані поля, які вміщають у собі токен;

– при заповненні форми, виконується запит на серверну частину, прихований токен від форми передається разом з даними користувача;

– веб-сервер отримує токен з форми та перевіряє на валідність. Перевірка на валідність може виконуватись звіренням токену з тим, що знаходиться у cookie-файлах чи розшифрування токену на основі збереженого на сервері секрету.

– якщо розшифрувати токен не вдалося, чи необхідні факти не збігаються, вважаємо, що сервер піддається атаці виду CSRF, в іншому випадку – схвалюємо запит.

Таким чином маємо алгоритм, який захищає форми відправки від атак. Такий метод захисту має свої недоліки та переваги:

– даний токен можна вважати елементом захисту зі строком життя в один запит, що робить його більш безпечним;

– до кожного запиту на сторінці сформований свій токен;

– сам по собі токен не вміщує корисну інформацію, що може ідентифікувати користувача, а лише являється методом перевірки на оригінальність запиту;

– даний метод можливий лише при безпосередній видачі шаблонів сторінок клієнту, що працює лише на шаблонній архітектурі клієнт-серверної взаємодії;

– відстеження багатьох токенів може бути тяжкою задачею для цільового сервера.

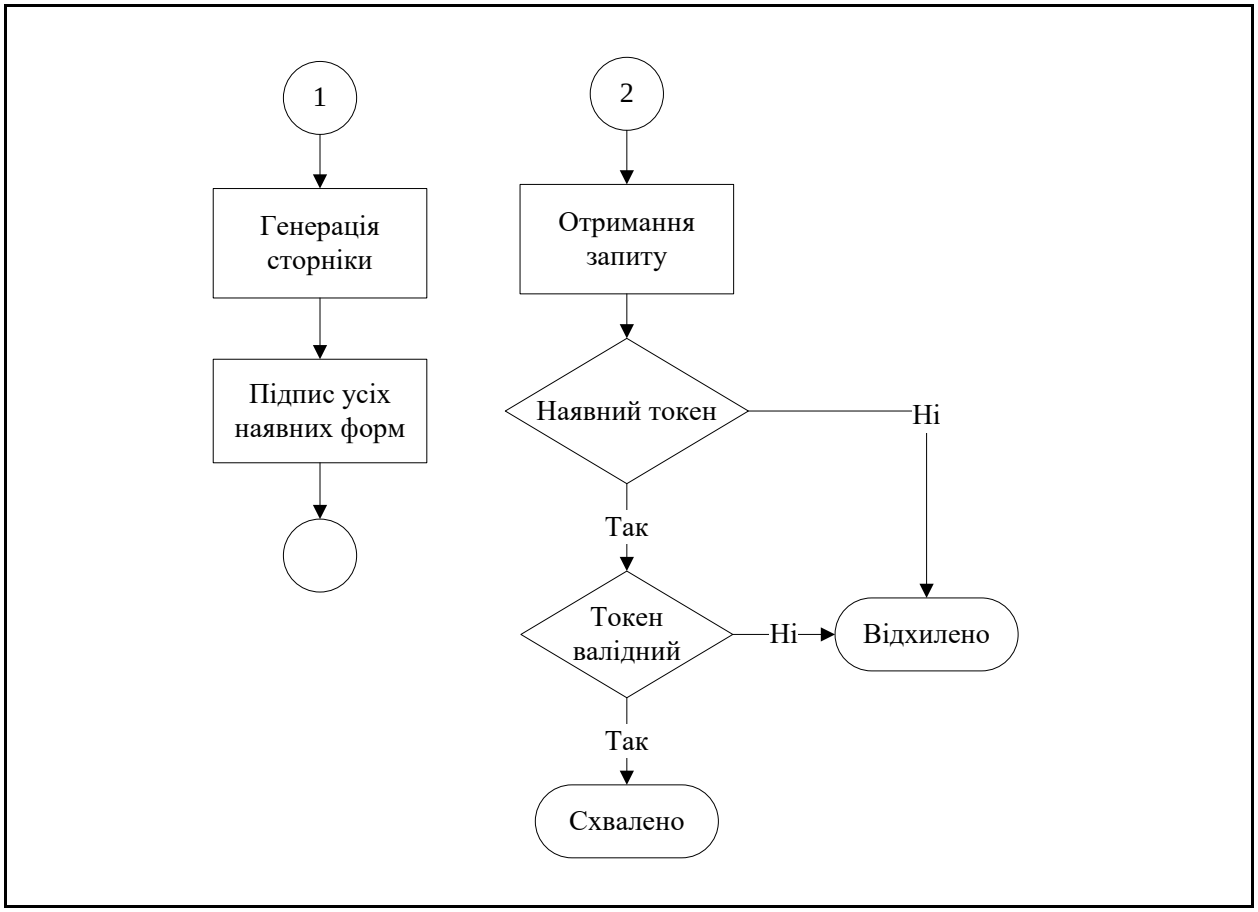

Рисунок 1.5 – Алгоритм роботи методу підписання форм-відправки

Таким чином, даний метод захисту може підходити тільки піж спеціальні сервери, що надають шаблони сторінок користувачам.

1.3.7 Таблиця доменів

Таблиця доменів – це таблиця довірених ресурсів, що мають право на з’єднання з цільовим сервером. Така таблиця розташована на сервері у місці, де

виконується контроль усіх запитів на сервер. При надходженні запиту, пакет даних вміщує в собі назву домену з якого було виконано запит. Якщо запит було виконано з оригінального сайту (тобто цільової веб-сторінки), то назва домену збігається з поточною назвою домену сервера. Завдяки звіренню кожного пакету даних з таблицею можна виконувати контроль надсилання даних і з інших, дозволених доменів.

Основним недоліком такого підходу є можливість затирання підпису пакету даних за допомогою мета-тегу домену відправки [26], то ж довіряти таким надісланим даним сервер не може.

1.4 Опис методів захисту щодо різної серверної архітектури

При виборі методів захисту від кібератак потрібно опиратися не лише на методи захисту самих алгоритмів але й на веб-сервери, що будуть перевіряти запити на оригінальність. Вибір методів захисту особливо залежить від архітектури цільового серверу. Двома основними архітекторами на серверній стороні муть бути монолітна та мікросервісну архітектури.

Монолітна архітектура сервера представлена у вигляді одного головного вузла, де виконується уся логіка: від обробки запитів, до контролю та зберігання даних. За такою архітектурою можна використовувати усі перечисленні методи захисту від атак окрім методу підписання форм відправки, який потрібно поєднувати із шаблонами сторінок, яких може і не бути.

З іншої сторони маємо мікросервісну архітектуру, де головний вузол від монолітної архітектури розбивається на менші мікросервіси, що виконують свою специфічну надану задачу. Таким чином головний вузол розвантажується та сам сервер стає легше збільшувати, підключаючи новий функціонал. Така архітектура досить популярна, але методи захисту від атак для неї ускладнюються за рахунок індивідуального захисту для кожного мікросервісу.

В такому випадку для мікросервісів стає складно використовувати єдиний токен, адже секрет для перевірки оригінальності зберігається лише на одному мікросервісі. Отже, для мікросервісу використання токенів стає важкою задачею і з перекислених методів можуть підійти лише методи: cookie-файл двійної відправки та таблиця доменів. Також можна розглянути захисні налаштування cookie-файлів, але даний метод не є стандартом і може привести до проблем взаємодії.

Основною ідеєю при захисті серверів з мікросервісною архітектурою є створення одного вузлу для валідації запитів чи створення зворотнього проксі-сервера, що обробляє запити та розподіляє їх по вузлам веб-сервера.

1.5 Пропозиція методів захисту з використанням сторонніх сервісів

При захисті від атаки виду CSRF сервер з мікросервісною архітектурою повинен опиратись на сторонній вузол, як точка перевірки запитів чи створювати захисні алгоритми на кожному мікросервісі відповідно. Таким чином архітектура мікросервісу ускладнюється, як і ускладнюється ідея незалежної роботи мікросервісних вузлів сервера.

В ході аналізу предметної галузі було встановлено лише один спосіб повного захисту серверу на основі мікросервісної архітектури: захист із застосуванням зворотнього проксі-сервера на прикладі сервісу CloudFlare. Такий підхід використовує внутрішні алгоритми сервісу для захисту від кібератак різних видів, в основі якого лежить один серверний вузол, що обробляє усі вхідні запити до веб-сервісу.

На сьогодні немає чіткого опису алгоритмів захисту від атак виду CSRF на основі стороннього сервісу, як одного з вузлів мікросервісу. Це означає, що припустимим в захисті від атак може бути мікросервісних вузол, винесений за рамки самого домену мікросервіса та знаходиться окремо. Таким чином усі вузли

в цільовому сервері мають змогу вільно спілкуватись зі стороннім сервісом, що перевіряє оригінальність запиту на стороні. Вузли цільового серверу становляться незалежними та рівноправними поміж усіма частинами серверу.

Для повноцінного функціонування методу захисту з використанням стороннього сервісу потрібно обумовити загальні положення стосовно елементів захисту – токена:

– токен захисту повинен бути криптостійким, тобто витримувати спроби втручання та підбору на час, поки токен існує;

– токен повинен бути унікальним для кожного факту клієнт-серверної взаємодії, тобто унікальним для користувача при з'єднанні з цільовим сервером;

– токен повинен бути недосяжним для зловмисника під час усього життєвого циклу токена;

– токен повинен безпосередньо слугувати єдиною можливістю встановлення оригінальності запиту.

Таким чином, при з'єднанні користувача з сервером, сторонній сервіс являється гарантом оригінальності запиту на основі деякого алгоритму, що здатен на коректну генерацію та перевірку токена.

1.5.1 Алгоритм клієнта-посередника

При стандартній роботі алгоритмів захисту від атак, токени надавалися самим серверами. Тепер таку функцію повинен виконувати сторонній сервіс, що підписує пакети передачі на замовлення клієнта (див. рис. 1.6). Роботоздатність такого алгоритму отримується за допомогою специфіки дії мережевого протоколу HTTP: пакети даних не можуть бути передані назад до веб-сторінки, що зробила запит, якщо домен цієї сторінки не співпадає з доменом цільового сервера. Тобто, при передачі даних на різні домени, сервер повинен явно надавати дозвіл на передачу.

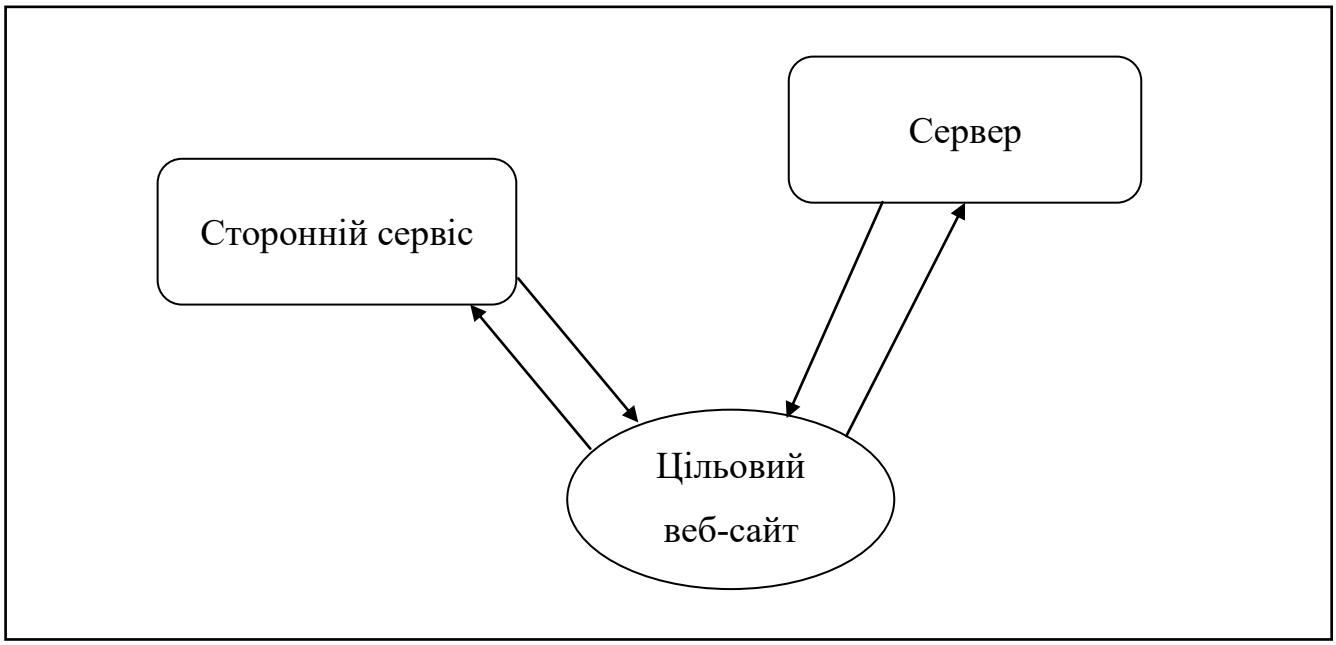

Рисунок 1.6 – Структура роботи алгоритму клієнта-посередника

При попередньому налаштуванні роботи даного алгоритму потрібно згенерувати приватну та публічну частину ключа, на основі яких і будуть проходити створення та валідація токену. Приватна частина ключа зберігається на стороні сервера, де токен буде розшифровуватись, а публічна частина надається сторонньому сервісу для генерації токенів.

Алгоритм роботи даного методу такий (див. рис. 1.7):

– при запиті клієнта до сервера, сервер надсилає клієнту певні факти, які потрібно зашифрувати, у вигляді cookie-файлу з прапорцем Http-only та у тілі відповіді. Такими даними можуть бути випадково згенерований послідовності знаків;

– при отриманні такої строки клієнт надсилає її для шифрування на сторонній сервіс. Сервіс шифрує надіслані факти за допомогою публічної частини ключа, що зберігається на стороні стороннього сервісу;

– зашифрований ключ надсилається клієнту, який надалі підписує їм усі свої запити до сервера;

– при отриманні запиту від клієнта сервер розшифровує токен за допомогою приватної частини ключа та звіряє факти з фактами, що знаходяться у cookie-файлах;

– якщо обидва токена збігаються, то вважаємо, що сервер не піддається атаці виду CSRF, в іншому випадку – відхиляємо запит.

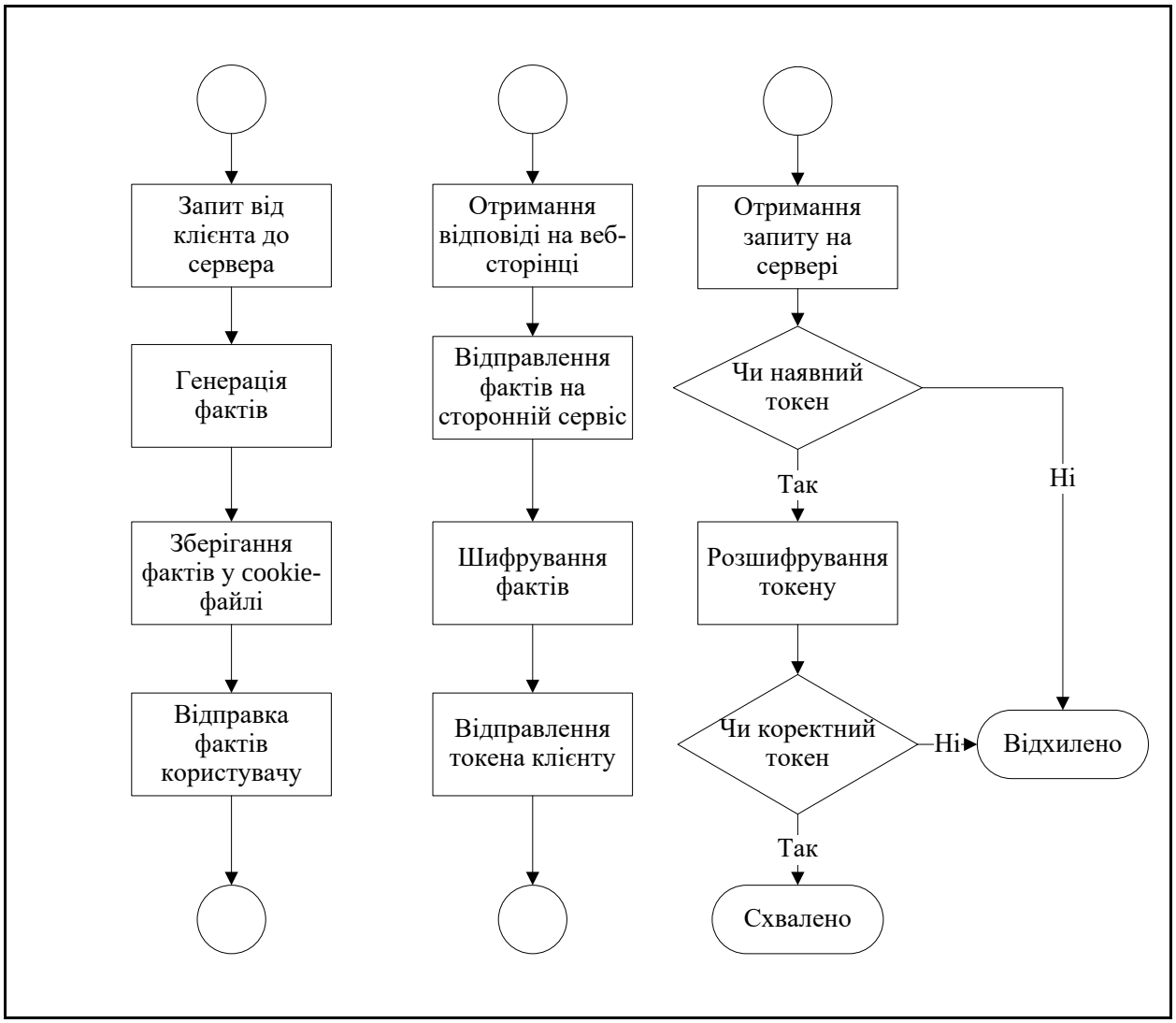

Рисунок 1.7 – Блок-схема роботи алгоритму клієнта-посередника

Основною ідеєю даного алгоритму є підтвердження оригінальності запиту при видачі токену зі сторони стороннього сервісу. Тобто, сторонній сервіс перевіряє запит перед видачею шифру за допомогою доменної таблиці. В основі видачі токену назад до клієнта лежить алгоритм взаємодії мережі зі сторонніми доменами, при якій дані, що передаються, на надаються для сторонніх непідтверджених доменів. В результаті маємо передачу шифрованого токену лише до затвердженого домену, який зберігається на стороні стороннього сервісу.

1.5.2 Алгоритм завірення передачі

Алгоритм завірення передачі базується на алгоритмі клієнта-посередника, де факти передаються для шифрування на сторонній сервіс та повертаються для підпису запитів до цільового сервера.

На відміну від свого базового алгоритму, алгоритм завірення передачі на надає токен назад до користувача, а зберігає його для подальшої верифікації безпосередньо з цільовим сервером (див. рис. 1.8)

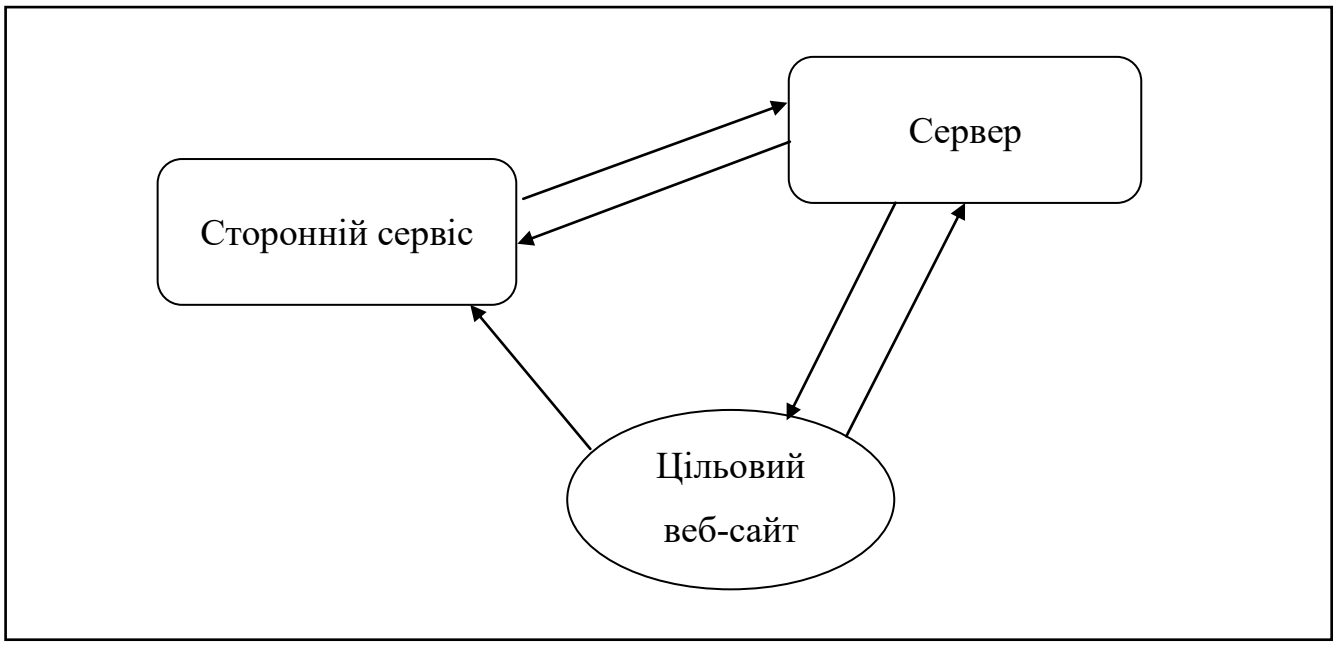

Рисунок 1.8 – Структура роботи алгоритму клієнта-посередника

Алгоритм роботи даного методу такий (див. рис. 1.9):

– при запиті клієнта до сервера, сервер надсилає клієнту певні факти, які потрібно зашифрувати, у вигляді cookie-файлу з прапорцем Http-only та у тілі відповіді. Такими даними можуть бути випадково згенерований послідовності знаків;

– при отриманні такої строки клієнт надсилає її для шифрування на сторонній сервіс. Сервіс шифрує надіслані факти за допомогою публічної частини ключа, що зберігається на стороні стороннього сервісу;

– зашифрований ключ зберігається на стороні стороннього сервісу для подальшої перевірки оригінальності запиту;

– при отриманні запиту від клієнта сервер відправляє запит до стороннього сервісу;

– сторонній сервіс, на основі попереднього запиту від клієнта, вирішує чи є запит оригінальним. Запит є оригінальним коли домен цільової сторінки збігається з цільовим сервером;

– сторонній сервіс нарає відповідь до основного серверу порівнюючи факти, що передав клієнт з фактам, що передав сервер;

– якщо факти збігаються, то вважаємо, що сервер не піддається атаці виду CSRF, в іншому випадку – відхиляємо запит.

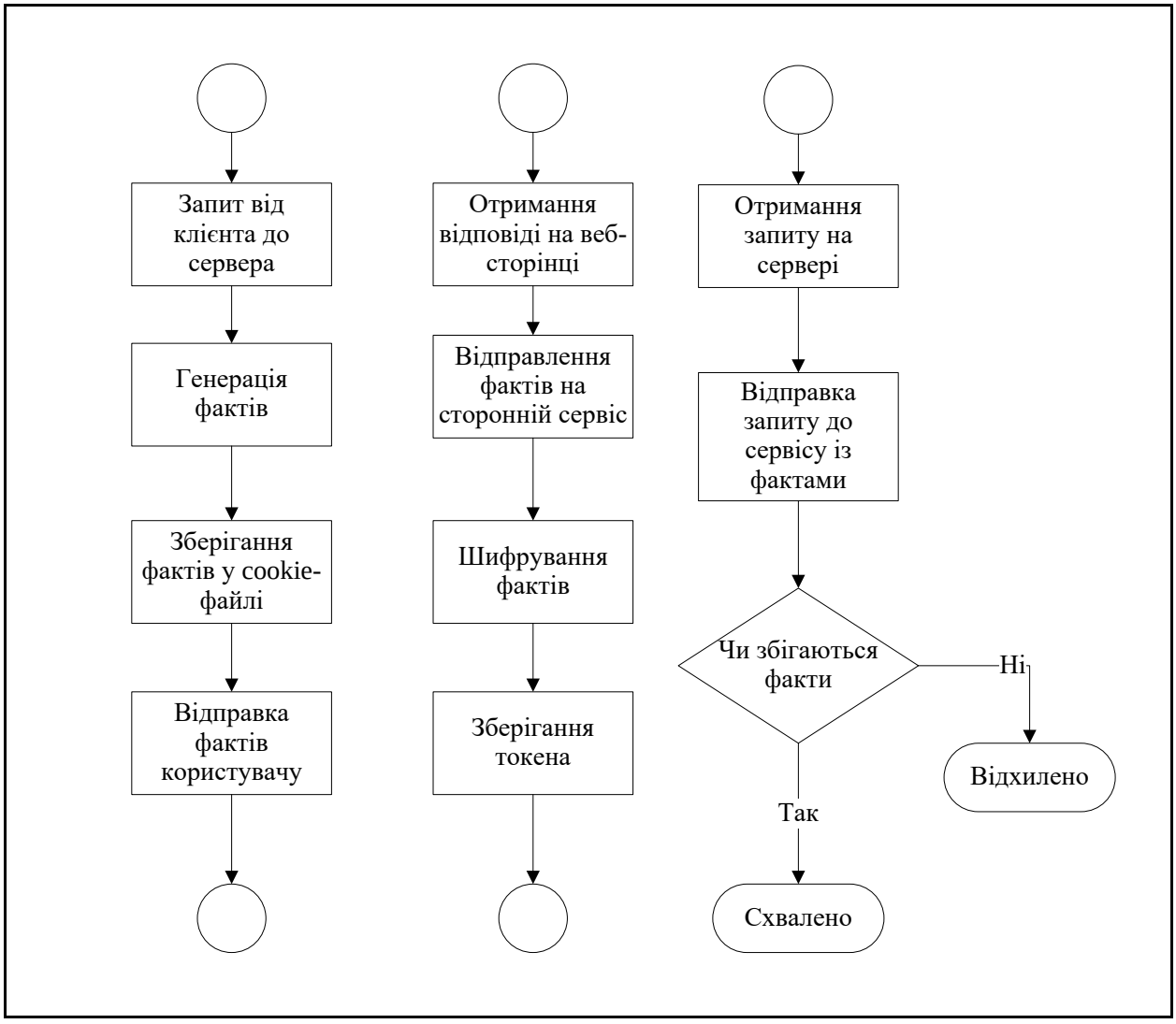

Рисунок 1.9 – Блок-схема роботи алгоритму завірення передачі

Даний алгоритм, на відміну від алгоритму клієнта-посередника, не надсилає токен назад до клієнта тим самим унеможливлює атаку виду XSS для викрадення токена. Даний токен тепер зберігається на серверах, тому зловмисник не має до нього доступу.

Таким чином, при передачі фактів від сервера до стороннього сервісу за допомогою користувача, алгоритм перевіряє оригінальність самого домену клієнта на сонові доменної таблиці. При некоректній передачі даних сторонній сервіс не пропустить факти та не зможе їх зашифрувати.

1.6 Постановка задачі

Кількість досліджень та аналізів стосовно алгоритмів захисту клієнт-серверної взаємодії від кібератак, що проводяться майже щороку, багато [1, 2, 10-13], але після аналізу наданих методів захисту вад CSRF атак, ще неможливо досконально зрозуміти:

– який з методів захисту найбільш ефективний;

– який з методів захисту краще підходить до монолітної архітектури серверної частини, а який – для мікросервісної архітектури;

– як елемент захисту впливає на швидкість обміну даними між клієнтом та сервером;

– який з методів є найбільш швидким для виявлення оригінальності запиту;

– як вибір методу впливає на навантаження серверної частини веб-сервісу;

– як використання методів ускладнює внутрішню логіку серверу та яка зростає кількість коду при імплементації алгоритму.

Тому в рамках цієї атестаційної роботи, будуть представлені результати дослідження ефективності методів захисту від атак виду CSRF, будуть представлені результати навантажень серверної частини та об’єми передачі даних в залежності від методу захисту.

Метою даної роботи є дослідження методів захисту від атак виду CSRF, формування емпіричних знань про ефективність методів та визначення факторів, пов’язаних з ефективністю клієнт-серверної взаємодії зі застосуванням методів захисту.

Для досягнення мети дипломної роботи необхідно дослідити:

– швидкість клієнт-серверної взаємодії із застосуванням методів захисту від CSRF атак;

– об’єм передачі даних залежно від використаного методу захисту;

– навантаження серверної частини при обробці запитів та створені захисних елементів для клієнта;

– безпечність використання методів захисту до серверної частини.

Дослідження буде проведено на наборі даних генерованих від імітації клієнт-серверної взаємодії. Таким чином ефективність буде перевірятись відтворенням навантаження при реальній роботі серверу, запиту будуть поступати з відповідною кількістю для отримання очікуваної відповіді.

Також в ході дослідження потрібно розробити систему на основі стороннього сервісу, що дозволить реалізувати захист серверу з мікросервісною архітектурою. Даний сторонній сервіс повинен впроваджувати метод захисту до цільового серверу на основі обраного алгоритму та слугувати ефективним способом вирішення проблеми інтеграції методів захисту до сервера з мікросервісною архітектурою. Розроблений додаток повинен мати можливість підтверджувати оригінальність клієнтських запитів до сервера на основі обраного алгоритму.

Результатом проведеного дослідження буде порівняння ефективності наданих методів захисту від атаки виду CSRF. Мірою ефективності будуть слугувати обрані критерії для встановлення ефективності алгоритму. Також, встановлення ефективності алгоритмів для серверів з певною архітектурою: монолітною чи мікросервісною.

2 АНАЛІЗ МЕТОДІВ ТА КРИТЕРІЇВ ДЛЯ ДОСЛІДЖЕННЯ

При аналізі методів захисту від CSRF атак на основі емпіричних знань потрібно визначити поняття ефективності для даних методів та обрати такі критерії, що будуть єдино значно описувати унікальні риси кожного методу та слугувати важливими факторами при обранні певного методу захисту. Критерії ефективності також повинні відповідати рисам обраної предметної області з-поміж інших методів захисту від кібератак.

Вибір самого факту ефективності методів, як основної риси, обумовлений ідеєю подальшого вибору потрібного алгоритму на базі проведених досліджень. Тобто вибір метрики ефективності цілком визначений подальшою можливістю використання наданих результатів.

Потрібно зауважити, що усі алгоритми має сенс досліджувати у системі веб-сервісу де серверна частина має мікросервісну архітектуру. Це рішення обумовлено можливими більш точними результатами стосовно роботи алгоритмів, а результати дослідження слугують більш точним показником ефективності для обох архітектур: монолітної та мікросервісної. Тобто, використання мікросервісної архітектури покриває й монолітну архітектуру при аналізі критеріїв ефективності.

2.1 Визначення поняття ефективності

Для подальшого аналізу методів захисту потрібно визначити поняття ефективності, на основі якого сформувати критерії дослідження. Ефективність методів захисту обумовлюється їх швидкістю та надійністю, а також можливістю серверної частини ефективно їх обробляти: швидка генерація та не затратна з точки зору ресурсів обробка токенів.

Математичний вираз критерію ефективності називають цільовою функцією, оскільки її показник продуктивності і є відображенням мети аналізу операції (в нашому випадку методів захисту) [27]. Звідси випливає, що для формування критерію ефективності рішень в операції насамперед потрібно визначити поставлену мету. Далі потрібно визначення показників результатів операції, а потім сформувати критерій ефективності. Показники результатів операції, на основі яких формується критерій ефективності, прийнято називати показниками ефективності. В окремих операціях показник результату операції може прямо виступати критерієм ефективності.

Загальний вигляд математичного виразу ефективності для методів захисту від атак виду CSRF можна представити, як вектор:

$$R_i = < K_1, K_2, K_3, \dots, K_{i-1}, K_i >, i = 1, 2, \dots, n \qquad (2.1)$$

де K_i – критерій ефективності, i – кількість критеріїв у показнику ефективності.

Хоча конкретні операції досить різноманітні, існує ряд загальних принципових положень, якими необхідно керуватися при формуванні системи критеріїв ефективності рішень. Залежно від типу систем і зовнішніх впливів операції можуть бути детермінованими, імовірнісними або невизначеними. Відповідно до цього виділяють три групи показників і критеріїв ефективності функціонування систем:

– в умовах визначеності, якщо критерії відображають лише один певний результат детермінованої операції;

– в умовах ризику, якщо критерії дискретні або безперервні випадкові величини з відомими законами розподілу ймовірнісної операції;

– в умовах невизначеності, якщо критерії є випадковими величинами, закони розподілу яких невідомі.

Далі потрібно привести математичне визначення для критерію. Якщо кожне значення i-го показника якості у j-й системі, де $i = \overline{1,n}$ та $j = \overline{1,m}$, описується за допомогою деякої вихідної змінної y_i^j, то значення y_i^j характеризує міру цієї якості. Узагальненим показником якості у j-й системі буде вектор:

$$Y^j = < y_1^j, y_2^j, y_3^j, \dots, y_n^j > \tag{2.2}$$

Для подальших розрахунків треба зауважити, що завдання нормування визначається у вигляді:

$$y_i^H = \frac{y_i}{y_i^0} \tag{2.3}$$

де y_i^0 – нормована величина.

Можливі кілька підходів до вибору нормованої величини:

– y_i^0 задається на основі розрахунку середнього значення від $avg(y_1^j, \dots, y_n^j)$;

– y_i^0 = максимальне значення y_i^j, $max(y_1^j, \dots, y_n^j)$;

– $y_i^0 = max(y_1^j, \dots, y_n^j) - min(y_1^j, \dots, y_n^j)$.

Необхідна якість системи задається правилами, яким повинні задовольняти показники властивостей, а перевірка їх виконання називається оцінюванням якості системи. Тоді, критерій якості – це показник істотних властивостей системи і правило його оцінювання.

Для нашого випадку критерії ефективності являються не детермінованими величинами з відомими законами розподілу. В такому випадку критерій визначається, як:

$$K: P(Y) \geq P^{\text{т}}(Y) \tag{2.4}$$

де Y – показник якості, $P(Y)$ – ймовірність істинності висловлювання Y, $P^{\text{т}}(Y)$ – потрібна ймовірність істинності висловлювання Y.

2.2 Визначення критеріїв оцінки методів захисту

Для подальшого аналізу ефективності потрібно визначити критерії ефективності для вектора (2.1). В основі вибору критеріїв лежить мета, завдяки якій надаються вподобання до того чи іншого методу захисту від атак. З точки зору програмних інженерів та веб-систем, куди встановлюються обрані алгоритми, можемо виділити основні критерії ефективності:

– швидкість взаємодії клієнта та сервера з використанням алгоритму захисту;

– об’єм даних, що передається до серверу з використанням елементів захисту;

– навантаження серверної частини при реалізації алгоритму захисту;

– розмір супроводжуючого коду при реалізації алгоритмів захисту;

– безпека запитів та передачі даних.

Виходячи зі списку сформованих критеріїв маємо вектор ефективності:

$$R = < K_{\text{шв.}}, K_{\text{об.}}, K_{\text{нв.}}, K_{\text{к.}}, K_{\text{б.}} > \quad (2.5)$$

де $K_{\text{шв.}}$ – критерій швидкості, $K_{\text{об.}}$ – критерій об’єму даних, $K_{\text{нв.}}$ – критерій навантаження, $K_{\text{к.}}$ – критерій розміру коду, $K_{\text{б.}}$ – критерій безпеки.

2.2.1 Швидкість взаємодії

Критерій швидкості взаємодії – міра часу, що витрачається на передачу одного пакету даних від користувача до серверної частини та назад. Точність виміру швидкості часу залежить від кількості вибірки запитів до серверу та вираховується діленням кількості запитів на загальний час обміну даними.

Потрібно розуміти, що на швидкість передачі та обробки даних впливають і сторонні фактори: мережева затримка при передачі, розмір пакету даних, вибір протоколу передачі. І якщо мережевою затримкою можна знехтувати так, як така затримка стосується усіх надісланих пакетів даних, то розмір пакету даних та вибір протоколу передачі потрібно встановлювати заздалегідь, до початку проведення дослідження на вимір швидкості передачі.

Основною ідеєю вибору критерію швидкості є основоположне значення для мережевої взаємодії між клієнтом і сервером. Тобто, дана метрика має великий вплив на побудову клієнт-серверної взаємодії, на основі якої формуються нефункціональні вимоги до серверу.

Для коректного визначення даного критерію потрібно обумовити середу проведення тестових досліджень:

– пакети даних повинні бути одного розміру для кожної вибірки запитів на кожен алгоритм захисту. Це дозволить більш точно визначити швидкість для кожного окремого пакета;

– тестування швидкості повинно проходити у двох варіантах: при першому пакети даних відправляються до сервера безперервно з максимальною частотою, при другому варіанті пакети надходять із затримкою – відокремлено;

– протоколом передачі даних обрати HTTP для стабільності проведення тестового дослідження та чистоти результатів.

Також можна тестувати критерій швидкості за допомогою почергової зміни розміру пакету передачі даних. Тоді маємо потік запитів на сторону сервера, де кожен наступний пакет зворотного до поточного розміром. Таким способом можна перевіряти чисту швидкість передачі, адже сервер оптимізований на кешування (збереження результату відповіді у оперативній пам’яті) однакових запитів, що може вплинути на чистоту результатів.

Результати до кожного методу захисту від CSRF атак потрібно виводити за проведеними експериментами при різній стратегії передачі даних. Тобто, для кожного з методів захисту проводяться три визначені експерименти та формується загальна швидкість передачі.

2.2.2 Об’єм передачі даних

Критерій об’єму передачі даних – міра спроможності серверної частини обробляти об’єм даних певного розміру, що надходить за одну одиницю часу. Треба розуміти, що сервер розміщує усі запити у чергу, яку поступово оброблює по мірі спроможності. Таким чином, можливо вирахувати ефективність роботи серверної частини при навантаженні з урахуванням алгоритмів захисту від атак.

Також потрібно розуміти, що на об’єм передачі та обробки даних можуть впливати стороні фактори: швидкість передачі даних, розмір пакетів даних, протокол передачі даних, бізнес-логіка, що спрацьовує при відповідних запитах, може викликати різне навантаження.

Основною ідеєю вибору критерію об’єму передачі даних є вплив даного критерію на ефективність роботи сервера. Тобто, даний критерій є основною метрикою працездатності серверної частини, особливо при обробці великого об’єму запитів.

Для коректного визначення даного критерію потрібно обумовити середу проведення тестових досліджень:

– пакети даних повинні бути одного розміру для кожної вибірки запитів на кожен алгоритм захисту;

– пакети даних з однієї вибірки повинні запитувати однакову бізнес-логіку у сервера. Такий підхід дає більш чіткі результати дослідження для порівняння результатів між методами захисту;

– пакети даних потрібно надсилати безперервно для постійного навантаження серверної частини;

– протоколом передачі даних обрати HTTP.

Одним із варіантів тестування об’єму передачі даних може бути тестування з надходженням запитів до різних частин бізнес-логіки. Таким чином, при тестуванні об’єму передачі даних, тестовий випадок буде симулювати більш природну поведінку запитів від клієнтів до сервера.

Результати до кожного методу захисту від CSRF атак потрібно виводити за проведеними експериментами при різній стратегії передачі даних та різних запитах до сервера (при використанні різної бізнес-логіки). Тобто, для кожного з методів захисту формується загальний об’єм передачі даних на основі списку проведених експериментів:

– тестування з постійним потоком запитів малого розміру;

– тестування з постійним потоком запитів великого розміру;

– тестування передачею даних змінного розміру;

– тестування з запитами бізнес-логіки різної складності, тобто перевірка навантаження серверної логіки.

Об’єм передачі даних вимірюється кількістю пакетів, що обробляє серверна часина за одну одиницю часу. Тоді для визначення загального об’єму передачі даних для кожного методу захисту, знаходиться середнє значення від усіх проведених тестових досліджень.

2.2.3 Навантаження сервера

Критерій завантаженості сервера – міра розміру ресурсів сервера, що використовуються для обробки запитів за одну одиницю часу. Двома основними ресурсами серверної частини являються розмір пам’яті та черга, яка заповнюється по мірі надходження запитів від користувачів веб-сервісу. Розмір пам’яті; що був використаний при обробці запитів встановлюється на момент кожної одиниці часу. Розмір черги вказує на навантаження серверу у конкретний момент часу адже дані у черзі – це ті дані, що потрібно обробити у наступну ітерацію після обробки поточних запитів.

Основною ідеєю вибору критерію навантаження є вплив міри навантаження на ефективність роботи сервера. Для ефективної роботи алгоритмів захисту вони повинні спричиняти мінімальні навантаження на серверну частину. Таким чином,

вибір ефективного алгоритму також залежить від спричиненого їм навантаження на сервер.

Для коректного визначення даного критерію потрібно обумовити середу проведення тестових досліджень:

– пакети даних повинні бути одного розміру для кожної вибірки запитів на кожен алгоритм захисту;

– пакети даних з однієї вибірки повинні запитувати однакову бізнес-логіку у сервера;

– пакти даних потрібно надсилати безперервно для постійного навантаження серверної частини;

– протоколом передачі даних обрати HTTP.

Результатом впровадженого тестування буде поєднаний розмір зайнятої пам’яті на момент відповіді та розмір необробленої черги (у байтах). Такий показник буде обраховуватись до кожного пакету при відповіді, далі буде підбиватись сума всіх запитів за встановлений проміжок часу.

Результати до кожного методу захисту від CSRF атак потрібно виводити за проведеними експериментами при різній стратегії передачі даних. Тобто, для кожного з методів захисту проводяться визначені експерименти та формується загальне навантаження за одиницю часу.

2.2.4 Розмір супроводжуючого коду

Розмір супроводжуючого коду – міра об’єму реалізації алгоритму захисту при зверненні методу захисту на серверній частині. Таким чином, можливо вимірювати складність реалізації алгоритмів для серверів. Дана метрика особливо помітна при реалізації захисту на серверах з мікросервісною архітектурою.

Основною ідеєю вибору критерію розміру супроводжуючого коду є вплив даного критерію на складність реалізації і, таким чином, загальної ефективності

при впровадженні конкретного методу захисту. Тобто, даний критерій є основною метрикою простоти розробки та впровадження алгоритмів, особливо у мікросервіси з великою кількістю внутрішніх серверів.

Для коректного визначення даного критерію потрібно обумовити основну метрику: розмір доданого коду. Для коректного підрахунку розміру супроводжуючого коду потрібно вирахувати кількість доданих символів до кожної складової частини цільового веб-сервісу: клієнтської частини веб-сайту та кожного сервера, де знаходиться захисний алгоритм.

Результати до кожного методу захисту від CSRF атак потрібно виводити за нарахованою сумарною кількістю символів, використаних при реалізації алгоритму. Тобто, для кожного з методів захисту формується загальний об’єм супроводжуючого коду з урахуванням усіх доданих методів але без урахування використаних вбудованих бібліотек. Такими бібліотеками вважаються включені в мову програмування або програмну платформу модулі, які постачаються з основним пакетом для використання мови програмування. Отже, до підрахунку розміру супроводжуючого коду включаються тільки сторонні під’єднанні бібліотеки та власний код реалізації алгоритмів захисту.

2.2.5 Безпека при передачі даних

Критерій безпеки – метрика, що визначає захисні якості алгоритмів, що направлені на захист від атак на клієнт-серверну взаємодію. Дана метрика на має чіткі граничні значення виміру: 0, якщо захист від повного сценарію відсутній та 1 – якщо присутній.

Основною ідеєю вибору критерію є його безпосередня присутність в основі самого дослідження. Адже головною задачею самих методів захисту – надання необхідної безпеки при взаємодії клієнта з сервером.

Для коректного визначення даного критерію потрібно проаналізувати усі можливі варіанти атак та захисту, провести суміжний аналіз на основі знайдених методів атаки.

Для коректного виміру критерію захисту потрібно скористатись загальною формулою для оцінки критерію зі встановленими значеннями оцінки:

$$K = \left[1 - \frac{1}{n}\sum_{i=0}^{n}\frac{(y_i - y_{\text{ф}.i})}{y_i}\right] \quad (2.6)$$

де y_i – конкретний показник захисту, $y_{\text{ф}.i}$ – фактичний показник якості для y_i, n – кількість видів атак виду CSRF.

Результатом впровадженого тестування буде якісна оцінка кожного з методів захисту так, як результуючі показники якості для кожного тесту будуть виведені в коефіцієнт безпеки для відповідного алгоритму.

2.3 Оцінювання методів захисту на основі критеріїв ефективності

Загальне оцінювання методів захисту на основі приведених критеріїв ефективності проводиться з використанням вектора ефективності за формулою (2.1). Для оцінки результатів можливо використати два підходи: порівняльна оцінка на основі значень векторів, виведення залежності на основі усередненого вектора.

При підведенні оцінки на основі порівняння векторів потрібно звірити кожне значення вектора та виділити один, як найкращий по результатам оцінки. Потрібно розуміти, що для перших чотирьох значень вектора робиться зворотна оцінка показника. Тобто, чим менше показник у перших чотирьох критеріях вектора – тим більше результуюча оцінка.

При другому способі – виведені залежності, потрібно проаналізувати отримані вектори з урахуванням усередненого вектора:

$$R_m =< K_{m.\text{шв.}}, K_{m.\text{об.}}, K_{m.\text{нв.}}, K_{m.\text{к.}}, K_{m.\text{б.}} > \quad (2.7)$$

$$K_m = \frac{1}{n}\sum_{i=1}^{n} K_i \quad (2.8)$$

де R_m – усереднений вектор, що містить усереднений показник K_m для кожного критерію K_i, n – кількість векторів ефективності.

Вираховування результуючої оцінки проводиться з використанням рівняння множинної регресії:

$$Y = b_0 + b_1X_1 + b_2X_2 + \cdots + b_nX_n \quad (2.9)$$

де b_i – коефіцієнт регресії, X_i – регресор, у нашому випадку критерій ефективності, n – кількість факторів моделі.

Далі потрібно звести вектори до рівняння множинної регресії (2.9), де параметрами будуть слугувати критерії ефективності. З використанням рівняння множинної регресії можливо знайти результуючу оцінку для кожного вектора ефективності та для усередненого вектора. Потрібно ураховувати, що оцінки векторів будуть спадаючими, тобто, чим менше результуюча оцінка – тим краща ефективність. Це обумовлено чотирма першими критеріями вектора, які аналізуються зворотною оцінкою. Останній критерій оцінюється у звичайному виді, тому для коректного підрахунку множинної регресії потрібно ставити критерій у від’ємну форму. У результаті маємо оцінки векторів та усередненого вектора і тепер можемо знайти найефективніший та найбільш оптимальний вектори.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В хоті дослідження методів захисту клієнт-серверної взаємодії було вирішено розробити методи захисту від атак на базі стороннього сервісу, що допоможе більш ефективно захищати серверні частини на основі мікросервісної архітектури а також оптимізація самого коду при реалізації захисту.

Сторонній сервіс повинен задовольняти такі функціональні вимоги:

– реєстрація нових серверів для надання захисту;

– видача ключа для перевірки підписів серверам;

– валідація запитів користувача на основі домену сервера;

– реалізація двох алгоритмів захисту по вибору сервера: алгоритм клієнта-посередника та алгоритм завіерння передачі.

В результаті сервіс повинен перевіряти оригінальність пакетів та впроваджувати надійний метод захисту для серверів з мікросервісною архітектурою.

3.1 Опис архітектури сервісу

Сервіс повинен бути виконаний, як окреме серверне рішення, що буде надавати змогу іншим серверам підключати можливості даного сервісу. Для початку потрібно виділити основні точки доступу до нашого сервісу:

– доступ з веб-сторінки, де можна налаштувати власний доступ до сервісу від цільового серверного рішення. Таким чином можливо буде зареєструвати власний сервер;

– доступ з клієнтської сторони, де веб-сторінка надає запит на перевірку оригінальності запиту;

– доступ до каналу зв’язку між цільовим сервером та даним сервісом, де можна буде завірити оригінальність запиту від клієнта до сервера.

Сторонній сервіс повинен містити такі блоки логіки, як: обробка клієнтських запитів, канал зв’язку із серверами, механізм отримання та віддачі даних до клієнта, доменні таблиці для валідації запитів, логіку власної сторінки та менеджменту серверів. На рисунку 3.1 показано перераховані блоки та механізми логіки.

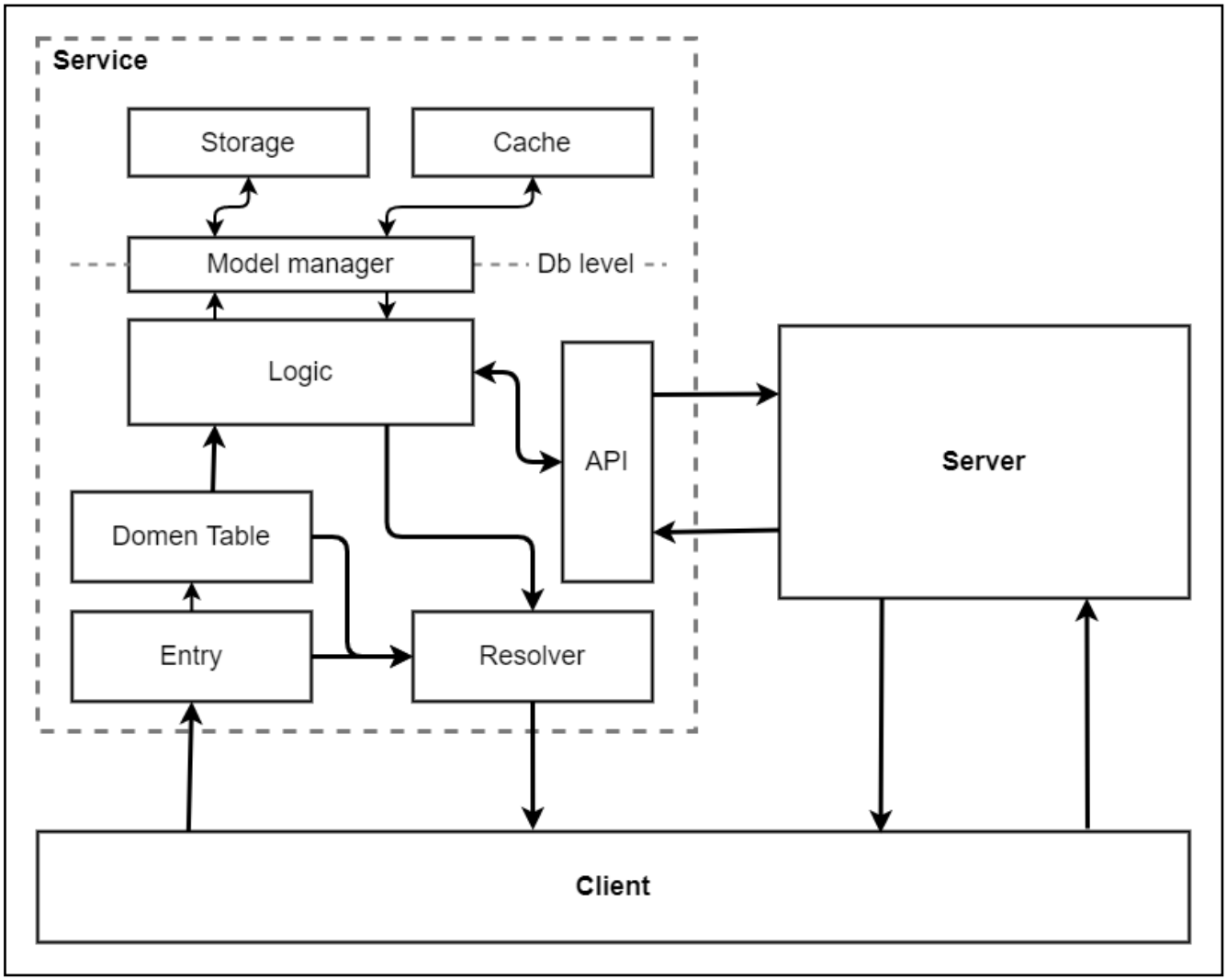

Рисунок 3.1 – Архітектура сервісної системи

Для архітектури сервісу представлені такі основні блоки:

– вхідний блок, який контролює та обробляє надіслані запити з клієнтської частини;

– таблиця доменів, що перевіряє запити на належність до відповідних доменів. Якщо запит не підходить до конкретного домену, то пакет даних повертається назад до клієнта не підписаним;

– блок основної логіки сервісу, що шифрує пакети, генерує ключі доступу та перевіряє на оригінальність запити;

– прошарок моделей даних, що спілкується з базою даних та сховищем кеш-даних [28]. Даний блок надає дані до використання основній логіці сервісу;

– відкрита частина логіки, з якою спілкуються цільові сервери для отримання підтвердження оригінальності запитів;

– блок видачі даних до клієнтської частини, який формує відповідь в залежності від результату обробки запита.

Результатом побудови архітектури сервісного рішення маємо повноцінний незалежний вузол для перевірки оригінальності запитів для цільових серверів. Така архітектура дозволяє обслуговувати велику кількість незалежних серверів, що дозволяє не створювати даний сервіс для кожного сервера окремо, а використовувати лише один екземпляр сервісу.

3.2 Опис алгоритмів захисту

В ході розробки для коректної роботи сервісу потрібно реалізувати алгоритми захисту від атак виду CSRF. Для повноцінного функціонування алгоритмів захисту з використанням стороннього сервісу потрібно встановити основні функціональні положення до алгоритмів:

– алгоритм захисту повинен використовувати криптостійкий ключ або токен, тобто витримувати спроби втручання та підбору на час, поки ключ чи токен існує;

– алгоритм повинен надійно та безперечно встановлювати оригінальність запиту та приналежний цільовий сервер до даного запиту;

– використаний ключ чи токен не повинен бути досяжним до зловмисника на протязі усього життєвого циклу.

Таким чином, при з’єднанні користувача з сервером, алгоритми стороннього сервісу повинні слугувати гарантом оригінальності запиту. Алгоритми повинні бути недосяжними до клієнт-серверної взаємодії та залишатися на стороні стороннього сервісу. А також, відкрита частина коду, що передається до сервера для коректної роботи алгоритмів та приватний ключ для дешифрування повинні бути закриті від взаємодії клієнта.

3.2.1 Алгоритм клієнта-посередника

Для повноцінної роботи сервісу основним методом захисту клієнт-серверної взаємодії від атак буде алгоритм клієнта-посередника. У даному алгоритмі основною ідеєю є підпис клієнтських пакетів сервісом у разі, якщо запит визнаний, як оригінальний. Даний алгоритм надає токен клієнту за допомогою якого клієнт у свою чергу може спілкуватись із цільовим сервером. А цільовий сервер на основі переданого токену може встановити оригінальність підпису розшифрувавши його.

Для встановлення зв’язку між стороннім сервісом та цільовим сервером алгоритм вимагає попередньої генерації ключів доступу та шифрування. Для початку, сервіс генерує пару ключів: публічний та приватний. Приватний ключ надається цільовому серверу для дешифровки, а публічний ключ зберігається у сторонньому сервісі. Також до сервісу заносяться дані стосовно домену, з яким пов’язана публічна частина ключа. Тепер для повноцінного шифрування клієнт повинен надати запит із установленого домену.

На рисунку 3.2 зображено діаграму активності алгоритму для більш наглядного опису покрокової взаємодії сервісу, клієнта та сервера. Спочатку клієнт надає первинний запит до цільового сервера, де клієнта ідентифікують,

надають ключ сесії для подальшого розпізнавання клієнта та генеровану строку, що представляє собою інформацію (або факт) для шифрування стороннім сервісом.

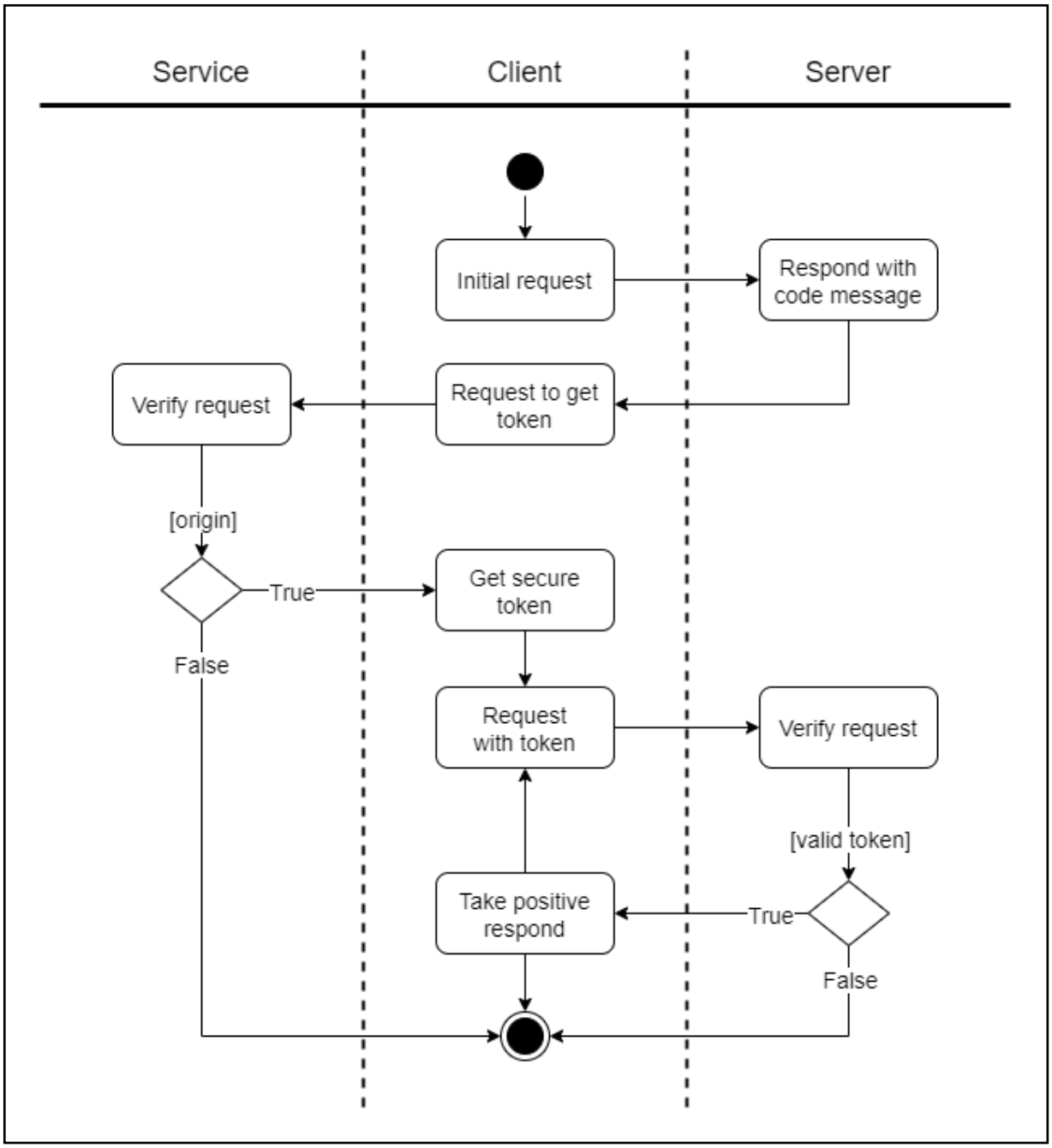

Рисунок 3.2 – Діаграма активності алгоритму клієнта-посередника

Далі дані передаються на сторону клієнта. Клієнт бере переданий факт для шифрування та надсилає його на сторону сервісу. Тоді сервіс перевіряє оригінальність запиту, порівнюючи домен з таблицею доменів. Якщо такий домен

існує, тоді передані факти зашифровуються з використанням публічного ключа, що прив’язаний до поточного домену. Таку зашифровану строку надсилають назад до клієнта, де у свою чергу наступні запити підписуються даним токеном. При отриманні запиту на стороні цільового сервера, сервер перевіряє валідність запиту намагаючись розшифрувати токен з використанням приватної частини ключа. Якщо розшифрувати дані не вийшло чи розшифровані дані не збігаються або токена зовсім немає, тоді запит вважається небезпечним та клієнту відмовляється у виконанні запиту.

На рисунку 3.3 наведено реалізацію основної логіки алгоритму: метод шифрування та метод перевірки шифру.

```
  private async getToken(): Promise<string | null> {
    const publicKey = await DomainTable.getPublicKey(this.domain);

    try {
      const encryptedData = crypto.publicEncrypt(
        {
          key: publicKey,
          padding: crypto.constants.RSA_PKCS1_OAEP_PADDING,
          oaepHash: 'sha512',
        },
        Buffer.from(this.data)
      );

      return encryptedData.toString('base64');
    } catch (error) {
      logger.error(error);
      return null;
    }
  }

  public verifyToken(token: string): boolean {
    const decryptedData = crypto.privateDecrypt(
      {
        key: this.privateKey,
        padding: crypto.constants.RSA_PKCS1_OAEP_PADDING,
        oaepHash: 'sha512',
      },
      Buffer.from(token)
    );
    return decryptedData.toString() === this.data;
  }
```

Рисунок 3.3 – Реалізація основної частини алгоритму клієнта-посередника

З точки зору захисних функцій такий алгоритм забезпечує підпис тільки тих пакетів, що являються оригіналами, та розпізнавання токена можливо лише тоді, коли підпис був виконаний парним публічним ключем до цільового сервера. Отримати токен із стороннього домену неможливо, що повністю унеможливлює атаку виду SCRF.

3.2.2 Алгоритм завірення передачі

Іще одним варіантом захисту з використанням стороннього сервісу може бути алгоритм завірення передачі. Для даного алгоритму основною є перевірка оригінальності пакета та збереження стану, за яким у подальшій взаємодії буде перевірятись валідність між сервісом та сервером. Тобто, для підтвердження оригінальності запиту цільовий сервер повинен безпосередньо звернутись до стороннього сервісу.

Для встановлення зв’язку між стороннім сервісом та цільовим сервером алгоритм, на відміну від першого, не вимагає попередньої генерації ключів доступу та шифрування. В даному алгоритмі досить уведення в доменні таблиці даних про цільовий сервер. При спілкуванні сервісну напряму з цільовим сервером використовується токен ідентифікації такий, як і в користувачів при з’єднанні з цільовим сервером. Завдяки такому токену можливо ідентифікувати сервер, що з’єднується із стороннім сервісом.

На рисунку 3.4 зображено діаграму активності алгоритму для більш наглядного опису покрокової взаємодії сервісу, клієнта та сервера.

Спочатку клієнт надає первинний запит до цільового сервера, де клієнта ідентифікують, надають ключ сесії для подальшого розпізнавання клієнта та генеровану строку, що представляє собою факт для шифрування стороннім сервісом. Далі дані передаються на сторону клієнта.

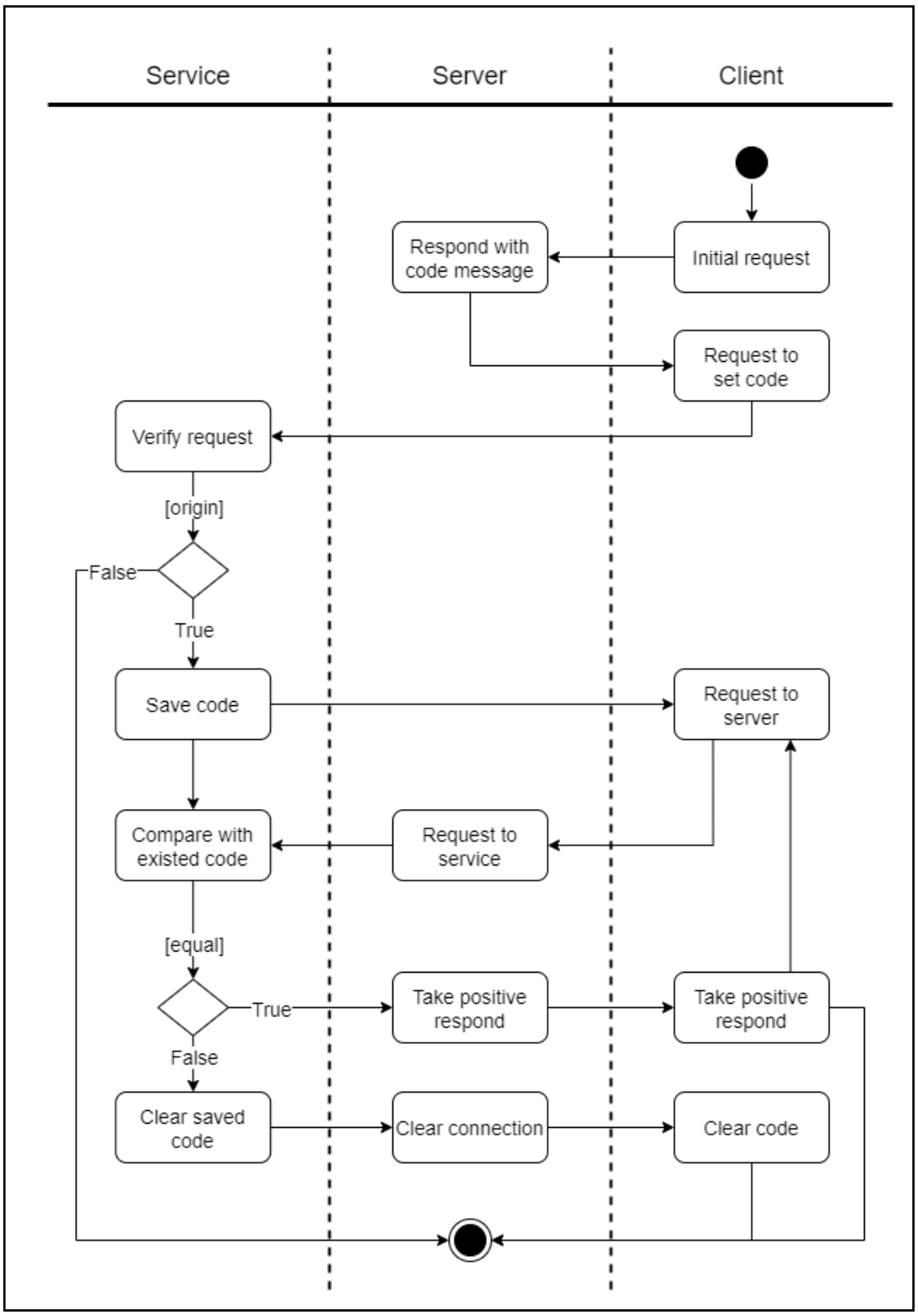

Рисунок 3.4 – Діаграма активності алгоритму завірення передачі

Після передачі даних на сторону клієнта, користувач надсилає переданий факт для шифрування на сторону сервісу. Тоді сервіс перевіряє оригінальність запиту, порівнюючи домен з таблицею доменів. Якщо такий домен існує, тоді передані факти зберігаються на стороні стороннього сервісу на час клієнтської

сесії. При наступних запитах клієнта до цільового сервера, сервер надсилає факт клієнта безпосередньо до сервісу, де порівнюються дані. Якщо факти збігаються, то сервіс надає позитивну відповідь до сервера, та якщо факти не збігаються, то вважається, що запит небезпечний.

На рисунку 3.5 наведено реалізацію основної логіки алгоритму: метод збереження переданого факту та метод перевірки переданого факту із сторони сервера та заздалегідь збереженого.

```
private async saveKey(): Promise<boolean> {
  const result = await KeyStorage.save(this.domain, this.data);
  return result;
}

static async create(domains: DomainList<URL>): Promise<Server> {
  if (await DomainTable.has(domains)) {
    const sid = await DomainTable.getServerId(domains);
    return new Server(sid);
  }

  const result = await DomainTable.setDomains(domains);
  return new Server(result.serverId);
}

async establishTokenEncoding(): Promise<KeyObject | null> {
  const { publicKey, privateKey } = crypto.generateKeyPairSync(
    'rsa',
    {
      modulusLength: 2048,
    }
  );

  const result = await DomainTable.setDomainPublicKey(publicKey)
  if (result) return privateKey;
  return null;
}

async verifyData(data: string): Promise<boolean> {
  const savedKey = await KeyStorage.getKey(this.serverId, data);
  if (!savedKey) return false;
  return true;
}
```

Рисунок 3.5 – Реалізація основної частини алгоритму завірення передачі

З точки зору захисних функцій такий алгоритм надає такий же захист, що й перший розглянутий алгоритм за виключенням атаки виду XSS. Справа у тому, що перший алгоритм схильний до атаки виду XSS, що може спричинити витік токена зв’язку між клієнтом та сервером. У такому випадку зловмисник може безпосередньо провести SCRF атаку на користувача, чий токен був викрадений. В ситуації з іншим алгоритмом ситуації з атакою виду XSS виникнути не може, адже сам токен не передається до клієнта а зберігається на стороні сервісу у виді чистих переданих фактів. Таким чином, алгоритм завірення передачі повністю не вразливий до атак виду SCRF.

Потрібно зауважити декілька випадків взаємодії стороннього сервісу та сервера для другого алгоритму. По перше, якщо токена на стороні сервісу виявлено не було, то дані із клієнтської сесії повинні очищуватись та алгоритм завірення повинен проходити із самого початку. По друге, токен, який приєднується до користувацької сесії, повинен існувати стільки ж, скільки й збережений запис про токен на стороні сервісу. Така взаємодія робиться для того, щоб оновлювати синхронно токени між сервісом та цільовим сервером.

3.3 Опис тестування алгоритмів захисту

Перевірка роботоздатності алгоритмів на предмет захисту від атак виду SCRF перевіряється симуляцією атаки з використанням вірусного сайту. Тобто, для правильної перевірки створених алгоритмів на базі стороннього сервісу потрібно реалізувати атаку на клієнта з використанням заздалегідь створеного вірусного сайту, який націлений на сервер з який спілкується клієнт.

При реалізації вірусного сайту потрібно створити прихований запит на цільовий сервер, який спрацьовує при переході клієнта на вірусний сайт. Запит повинен міститись у підготовленій формі веб-сторінки та спрацьовувати з використанням скрипту у програмному коді (див. рис. 3.6).

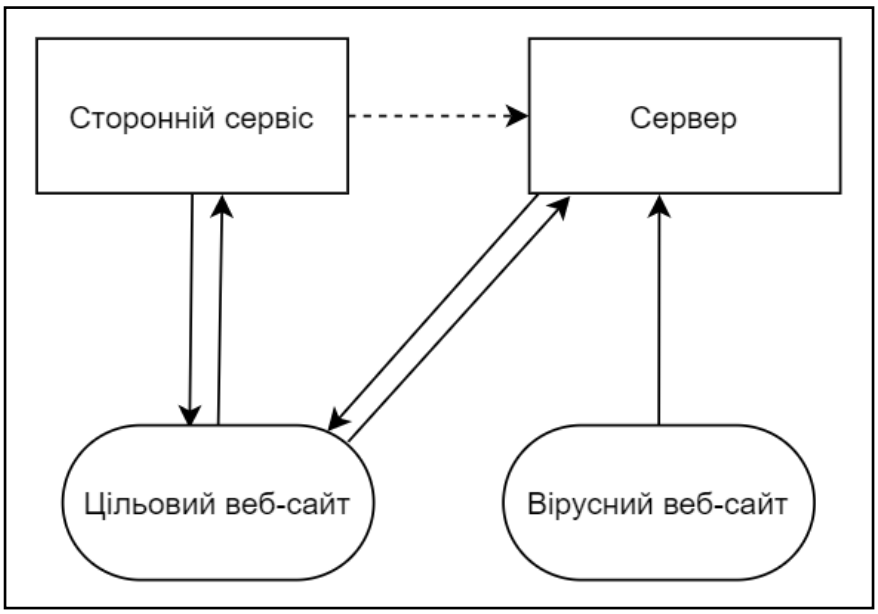

Рисунок 3.6 – Схема атаки на захищену систему із застосуванням стороннього сервісу

На рисунку 3.7 показана основна частина реалізації алгоритму атаки виду SCRF на цільовий сайт.

```
<form ="hiddenForm" method="post" hidden
  action="https://pointserver.com/api/user/transfer"
>
  <input type="text" name="toUser" value="5367" />
  <input type="text" name="amount" value="-1" />
</form>

   (function (context) {
    const form = $('#hiddenForm');
    form.on('submit', (e) => {
      e.preventDefault();
      $.ajax({
        url: 'https://point-server.com/api/user/transfer',
        method: 'post',
        data: {
          toUser: 5367,
          amount: -1,
        },
      });
      return false;
    });
    form.trigger('submit');
  })(this);
```

Рисунок 3.7 – Реалізація основної частини алгоритму атаки виду SCRF

Основною ідеєю такої атаки є створення незалежного сайту в Інтернеті, де користувачі можуть ненавмисно перейти на даний вірусний сайт і, не підозрюючи самого факту атаки, привести в дію приховану частину коду, яка виконає певні запрограмовані дії на стороні цільового сервера від особи користувача.

Потрібно зауважити, що реалізувати алгоритм можливо двома способами: перший спосіб надсилає запит не перезавантажуючи сторінки, тобто сам запит помітити не можливо, а другий – надсилає дані з перезавантаженням сторінки веб-сайту. Таким чином, реалізація способу залежить від мотивації зловмисника: чи має сенс маскувати сам факт запиту до цільового сервера, чи не має.

Сама перевірка алгоритмів проводиться переходом користувача на вірусну сторінку. Тоді запит від особи користувача надсилається на цільовий сервер, і далі перевіряється, чи виконав сервер запит, розпізнавши у надісланому запиті клієнта. Якщо сервер не виконує дій, надісланих від вірусної сторінки, то алгоритми захисту працюють коректно.

Для самої перевірки проводяться експерименти на неавторизованому та авторизованому користувачах, для переконання в працездатності не лише алгоритму захисту але й самого сервера. У таблиці 3.1 перераховані усі тестові випадки для алгоритмів стороннього сервісу.

Таблиця 3.1 – Тестові випадки для стороннього сервісу

Тестові випадки	Без алгоритмів	Алгоритми	
		клієнт-посередник	завірення передачі
Запит без токена авторизації для неавторизованого користувача з цільового веб-сайту	схвалено	схвалено	схвалено
Запит з токеном авторизації для неавторизованого користувача з цільового веб-сайту	схвалено	схвалено	схвалено
Запит без токена авторизації для авторизованого користувача з цільового веб-сайту	відхилено	відхилено	відхилено

Продовження таблиці 3.1

Тестові випадки	Без алгоритмів	Алгоритми	
		клієнт-посередник	завірення передачі
Запит з токеном авторизації для авторизованого користувача з цільового веб-сайту	схвалено	схвалено	схвалено
Запит без токена авторизації для неавторизованого користувача з вірусного веб-сайту	схвалено	відхилено	відхилено
Запит з токеном авторизації для неавторизованого користувача з вірусного веб-сайту	схвалено	відхилено	відхилено
Запит без токена авторизації для авторизованого користувача з вірусного веб-сайту	відхилено	відхилено	відхилено
Запит з токеном авторизації для авторизованого користувача з вірусного веб-сайту	схвалено	відхилено	відхилено

Результати тестування показують, що алгоритми захисту виконують свою задачу у перевірці оригінальності запиту від клієнта до сервера. Основна ідея алгоритмів цілком виконується: користувач не може отримати токен підпису пакетів, якщо він звертається із стороннього домену.

Потрібно зауважити, що фіксація виконаних алгоритмів перевіряться на стороні серверу а не самим фактом надходження відповіді, адже відповідь від крос-доменних запитів зазвичай помічається типовою помилкою «Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at https://point-server.com/api/user/transfer.». Тобто виникає помилка крос-доменної передачі, але, незважаючи на те, запит все одно виконується, однак сама відповідь не доходить до вірусного сайту.

4 ДОСЛІДЖЕННЯ МЕТОДІВ ЗАХИСТУ КЛІЄНТ-СЕРВЕРНОЇ ВЗАЄМОДІЇ ВІД АТАК ТИПУ CSRF

Відтворення та дослідження методів захисту на основі моделювання тесових випадків вимагає попереднього опису системи, що досліджується. Як описано раніше, при аналізі ефективності клієнт-серверної взаємодії з використанням алгоритмів захисту, будуть імітуватись тестові випадки навантаження серверної частини на основі мікросервісної архітектури. Також до результуючої оцінки йде критерій безпеки та розміру програмного коду.

На основі впроваджених критерії в та тестових випадків можна визначити математичну модель [29], що впроваджується у даному дослідженні. У даному випадку можна встановити два методи дослідження: імітаційне та аналітичне. Імітаційними дослідженнями, тобто дослідженнями на базі імітованих моделей об’єкта дослідження та імітованих сигналів взаємодії, в даному випадку виступають тестові експерименти пов’язані з вимірюванням навантаження цільового серверу. Аналітичні дослідження, у свою чергу, проводяться з точною фіксацією спостережуваних значень, тобто вимірювання безпеки та розміру супроводжуючого коду.

До поставленої математичної моделі також потрібно указати критерії, що її супроводжують. Дана математична модель є лінійною детермінованою динамічною моделлю та наближеною за мірою повноти опису моделі. Лінійність моделі означає, що встановлення оцінки на виборці з моделі, що досліджується, також можливо застосувати і до генеральної сукупності. Критерій детермінованості, у свою чергу, відображає процеси, для яких передбачається відсутність випадкових впливів. Для даної системи вважаємо, що стороннього впливу немає, адже тестові випадки досліджуються у контрольованій локальний середі. Дискретне моделювання досліджує систему у часті, як у нашому випадку досліджується навантаження на цільовий сервер.

4.1 Розробка методів імітації клієнт-серверної взаємодії

Для дослідження перших трьох виділених критеріїв потрібно розробити алгоритми, що будуть імітувати клієнт-серверну взаємодію [30]. Такі алгоритми будуть імітувати передачу запитів до серверної частини від декількох клієнтів, що дозволить перевірити навантаження при реальних ситуаціях.

Основними метриками до кожного пакету даних, що надходять до серверної частини, будуть: швидкість запитів, навантаження серверної частини при обробці даних та об’єм пакетів, що надходять. Таким чином, для коректного підрахунку результатів потрібно фіксувати пакет та стан сервера за кожним запитом від клієнта до сервера.

Окрім самих пакетів даних, для коректної перевірки навантаження потрібно визначити методи відправки самих пакетів, тобто визначити стратегії. Швидкість передачі та навантаження на сервер також залежить від розміру пакетів, функціоналу, що виконує сервер до певних пакетів, також стратегії кешування запитів впливає на швидкість передачі. Враховуючи усі перераховані фактори можна вивести основні стратегії при передачі даних:

– постійна передача пакетів малого розміру. Такий підхід допоможе виявити максимальну швидкість відповіді від сервера;

– постійна передача пакетів великого розміру. Дана стратегія вимірює протилежне значення до першої стратегії – мінімальна швидкість передачі для пакетів великого розміру;

– зміна розміру при передачі пакетів даних. Така стратегія націлена на обходження можливого кешування відповідей на стороні сервера. Сервер може звернутися до алгоритмів кешування відповідей, якщо до серверу надходить велика кількість однакових запитів;

– передача пакетів даних із супроводжуючим виконання великого об’єму логіки. При даній стратегії можливо перевірити навантаження сервера, що супроводжується передачею великої кількості пакетів;

– імітація реального навантаження. Для цієї стратегії використовуються усі перераховані вище варіанти тестування позмінно. Таким чином, сервер буде навантажуватись надходженням різнопланових запитів, що допоможе відобразити більш приближені до реальних випадків дані навантаження.

Також потрібно враховувати і структуру самого сервера, що може в значній мірі вплинути на результати тестових випадків. За основу сервера було обрано сервер з використанням мікросервісної архітектури (див. рис. 4.1), таким чином можливо протестувати усі перелічені алгоритми захисту на більш складному варіанті серверного рішення. На рисунку 4.1 показано, як клієнтський веб-сайт для тестування цільового серверу взаємодіє не тільки із основною частиною сервера, але й з іншими мікросервісами. Саме така поведінка найчастіше спостерігається у серверів з мікросервісною архітектурою. Для даного дослідження буде використаний сервер з 5 мікросервісами включно з основним.

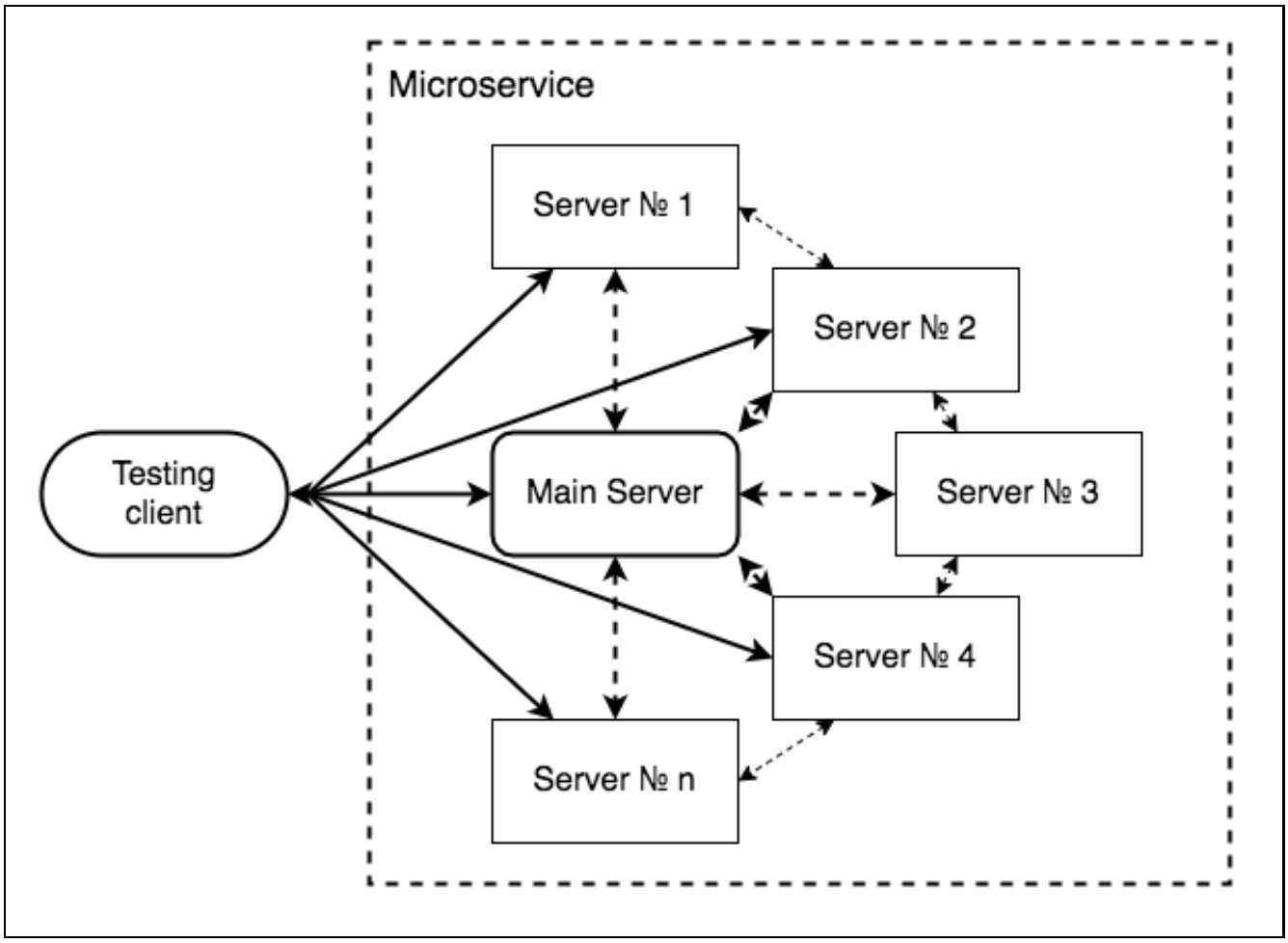

Рисунок 4.1 – Тестування сервера з мікросервісною архітектурою

Для коректного тестування методів захисту з використанням стратегій передачі даних потрібно також описати й протокол передачі даних, за яким пакети будуть надходити до серверної частини. Такий протокол передачі даних повинен бути однаковим для усіх тестових випадків задля дотримання однакових факторів впливу на стратегії тестування.

Тестування методів захисту буде проводитись з використанням HTTP протоколу передачі даних. Дані будуть передаватись з використанням методів GET і POST специфікації HTTP протоколу передачі. Основною різницею між даними методами передачі є той факт, що при запиті GET методом тіло запиту не використовується. Тобто, запит GET методом не використовує дані для передачі, пакет даних лише слугує запитом для отримання серверних даних. З іншого боку, запит методом POST може передавати дані до серверної частини.

Для усіх перерахованих стратегій передачі даних можна реалізувати один головний алгоритм передачі за допомогою якого окремі стратегії будуть керувати передачею даних. Основними схожими рисами для усіх стратегій є: протокол передачі даних, шлях передачі, пакетні дані, що передаються. Тому можна реалізувати основну логіку для усіх тестових випадків, яку у подальшому будуть використовувати тестові стратегії для клієнт-серверної взаємодії (див. рис. 4.2).

```
const resolveRequest = (path: URL, method: RequestMethod, package:
 TransferObject): Response => {
  const queryInstance = axios.create({ baseURL: baseURL });
  const request = queryInstance.request({
    method: method,
    url: path,
    data: package,
  });
  const { data, status } = request;
  return new Response({
    time: data.responseTime,
    load: data.load,
    queue: data.serverQueue,
  });
};
```

Рисунок 4.2 – Основна частина логіки передачі для стратегій тестування

З метою точності досліджень також потрібно оговорити й задані параметри самих стратегій тестування. Дані параметри будуть застосовуватись у тестових алгоритмах для передачі даних. Виділимо такі параметри для описаних стратегій:

– при стратегії передачі запитів малого розміру існує лише два параметра: розмір пакета та кількість запитів у секунду зі сторони клієнта. Даний запит буде надсилатись методом GET, тому розмір прикріплених даних дорівнює нулю, за визначенням самого GET методу. Для встановлення швидкості передачі потрібно враховувати й швидкість передачі інших стратегій, отже критерій швидкості протягом усього етапу тестування повинен бути однаковим для усіх тестових випадків;

– стратегії передачі запитів великого розміру цілком схожа на стратегію передачі малих пакетів за виключенням самого розміру пакета. Для даного дослідження встановимо розмір великого пакета даних, як 1000 кілобайт;

– при стратегії передачі запитів змішаного розміру пакети малого та великого розмірів передаються у випадковому порядку. Якщо розміри пакетів було встановлено раніше, то правило для випадкової передачі ще не встановлено. Випадковість у даному випадку буде встановлюватись по закону рівномірного розподілу [31];

– стратегія передачі запитів з виконанням великого об’єму логіки передбачає створення алгоритму на стороні сервера, що буде виконувати певну логіку для кожного запиту. Щоб уникнути ситуації кешування результату на стороні сервера потрібно створити непередбачений заздалегідь алгоритм, що буде кожен раз повертати різну відповідь. Можна використати будь-який складний алгоритм для роботи з великою кількістю даних чи алгоритм для проведення складних математичних операцій з великими числами. В ході даного дослідження було реалізовано алгоритм підрахунку дробових чисел із заданою точністю (див. рис. 4.3);

– при стратегії імітації реального навантаження на серверну частину потрібно використовувати усі попередні алгоритми у випадковому порядку.

Випадковість, як і у випадку із стратегією передачі запитів змішаного розміру, буде встановлюватись по закону рівномірного розподілу.

```
const GaussMethod = (accuracy: number): number => {
  const p = [];
  const m = [];
  const nbArctan = 3;
  let arctan;
  let Pi = 0;

  m[0] = 12; m[1] = 8; m[2] = -5;
  p[0] = 18; p[1] = 57; p[2] = 239;

  for (let i = 0; i < nbArctan; i++) {
    arctan = Math.arccot(p[i]);
    arctan *= Math.abs(m[i]);
    if (m[i] > 0) Pi += arctan;
    else Pi -= arctan;
  }

  Pi *= 4;
  return Pi;
};
```

Рисунок 4.3 – Основна частина логіки передачі для стратегій тестування

Для поданих страченій тестування потрібно оговорити ще один параметр – швидкість передачі пакетів даних, або щільність передачі. Для даного дослідження можливо виділити два основні варіанти швидкості: передача наступного пакету даних при успішному надходженні попереднього, тобто послідовна передача, та варіант паралельної передачі, де за встановлений час надсилається певна кількість пакетів не очікуючи на їх повернення. В другому випадку потрібно встановити щільність надсилання пакетів, що для нашого дослідження буде 50 запитів в одну секунду, тобто 1 запит кожні 20 мілісекунди. Така швидкість обумовлена загальною швидкістю надходження пакетів даних, що н супроводжуються складною логікою на стороні сервера.

В результаті обговорення параметрів для стратегій тестування маємо 5 стратегій та 2 режими швидкості передачі, тобто загально 10 тестових випадків до кожного алгоритму захисту від атак.

Результатами розробки системи тестування алгоритмів захисту стала програмна реалізація веб-додатку та віддаленого серверу для тестування алгоритмів (див. рис. 4.4).

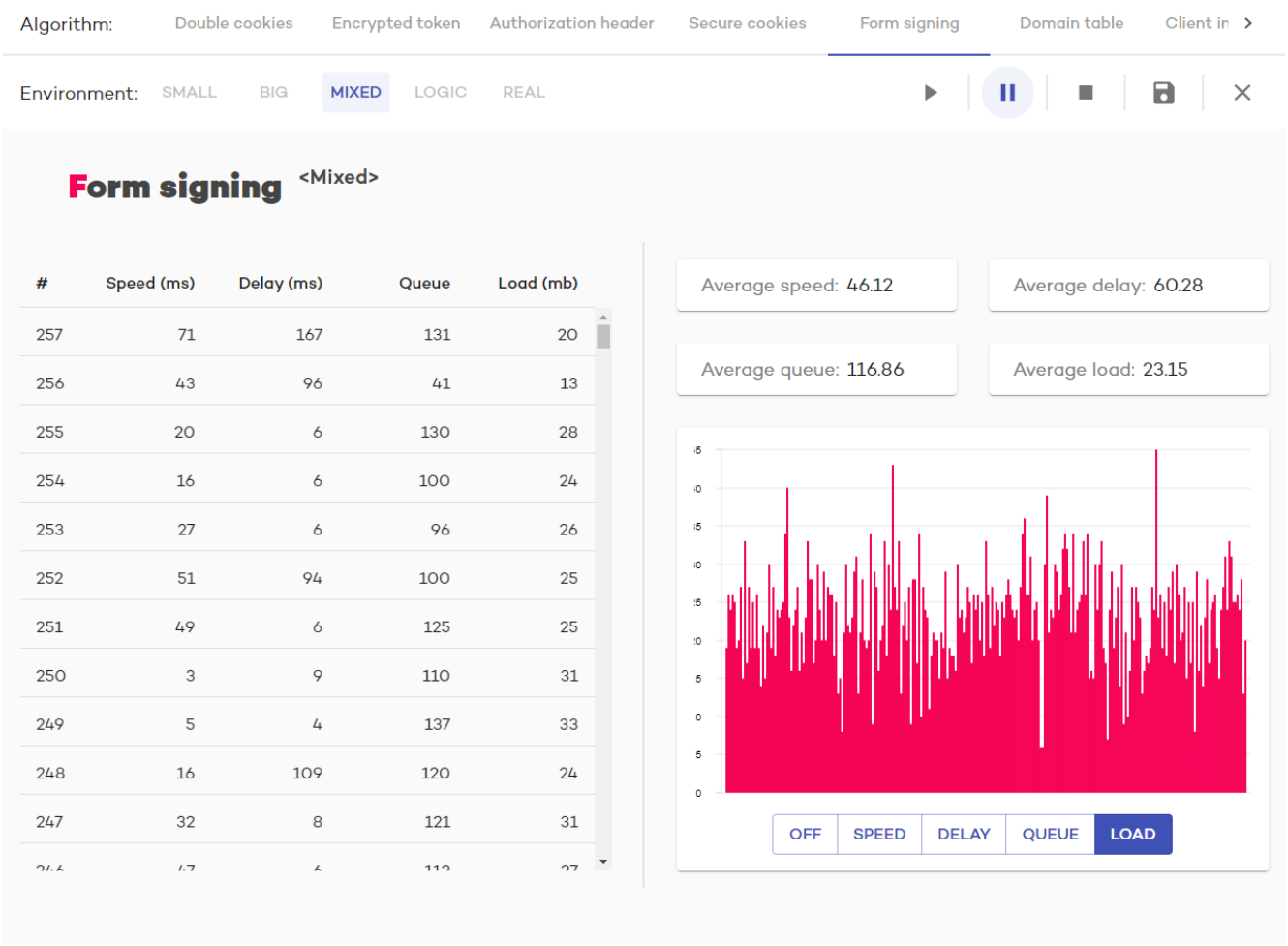

Рисунок 4.4 – Інтерфейс веб-додатку для тестування захисних алгоритмів

Веб-додаток має можливість тестувати надані алгоритми захисту передачі даних за пропонованими видами налаштувань тестової середи до кожного алгоритму.

Механізм тестування полягає у підготовці та відправленні пакетів даних до серверної частини з урахуванням обраного алгоритму та стратегії тестування. Після обробки даних сервер підготовлює відповідь надсилаючи серверні метрики, що включають витрати пам'ять сервером, довжину черги необроблених запитів на момент відправки пакета, швидкість обробки та відправлення конкретного пакета

даних. У результаті багаторазового відправлення з усіх результатів підводиться статистика з підрахунком середніх показників за увесь тестовий випадок.

Результати тестування можливо зберігати та передивлятися з більш детальною інформацією по пройденим тестам (див. рис. 4.5).

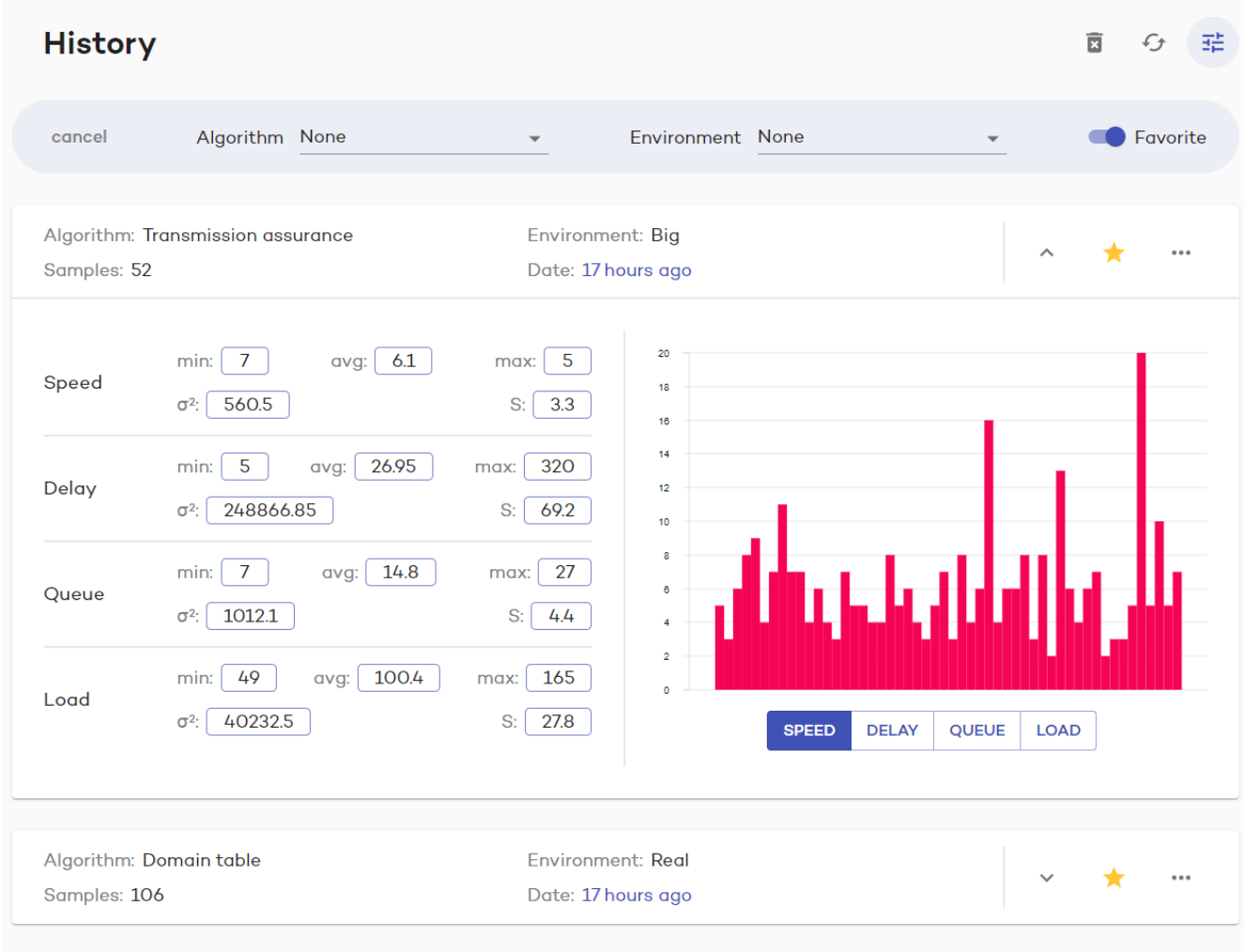

Рисунок 4.5 – Історія проведених тестів, що були збережені

Серверна сторона розподілена на незалежні частини відповідно до кожного алгоритму захисту та реалізує свій власний метод захисту до кожного пакету даних, що надходить до обробки. За контроль розподілу методів захисту відповідає створена деревовидна роутингова структура, яка дозволяє пакету даних надходити до відповідного блоку із потрібним реалізованим методом захисту від атаки виду CSRF.

4.2 Аналіз методів та алгоритмів захисту

Для порівняльної характеристики методів захисту потрібно дослідити їх показники за даними стратегіями тестування. Тестові показники будуть записані у порівняльні таблиці згідно обраних критеріїв для оцінки ефективності алгоритмів захисту. Тестування буде проводитись на реалізації раніше перерахованих алгоритмах захисту:

– cookie-файл двійної відправки (А);

– зашифрований токен (Б);

– заголовок авторизації (В);

– захисні налаштування cookie-файлів (Г);

– підписання форм-відправки (Д);

– таблиця доменів (Е);

– алгоритм клієнта-посередника (Ж);

– алгоритм завірення передачі (И).

Тестування кожного методу захисту буде проводитись з використанням п’ятьох стратегій тестування:

– стратегії передачі запитів малого розміру (1);

– стратегії передачі запитів великого розміру (2);

– при стратегії передачі запитів змішаного розміру (3);

– стратегія передачі запитів з виконанням великого об’єму логіки (4);

– при стратегії імітації реального навантаження (5).

В результаті проведення перерахованих тестів будуть встановлені виміри критеріїв ефективності. До кожного критерію ефективності буде складено порівняльну результуючу таблицю.

Першим критерієм ефективності методів захисту є критерій швидкості передачі. Даний критерій вимірювався на протязі усіх тестових випадків, а його результуюче середнє значення відображає критерій швидкості для кожного методу. Результати досліджень критерію швидкості відображено у таблиці 4.1.

Таблиця 4.1 – Результати вимірювання критерію швидкості за тестовими випадками

Методи захисту	Результати тестування швидкості стратегіями (мс.)					Загалом
	1	2	3	4	5	
Метод А	25	48	34	132	58	59,4
Метод Б	8	21	16	91	23	31,8
Метод В	7	13	8	89	22	27,8
Метод Г	6	14	10	87	26	28,6
Метод Д	52	81	59	211	106	101,8
Метод Е	6	11	8	86	21	26,4
Метод Ж	11	33	21	93	36	38,8
Метод И	12	51	27	112	45	49,4

За результатами вимірювань можна побачити, як алгоритм підписання форм-відправки (Д) найповільніший із даних. Такий результат можна пояснити методом відправки даних алгоритмом: при кожному результаті запиту алгоритм генерує та надсилає цілу сторінку, що буде значно повільніше аніж інші алгоритми.

Наступними методами захисту по повільності виявились cookie-файл двійної відправки (А) та алгоритм завірення передачі (И). Повільна взаємодія даних алгоритмів обумовлена типом передачі, при якій запит відсилається подвійно через особливості роботи самих алгоритмів.

Інші алгоритми захисту відпрацювали достатньо швидко та мають приблизно однакові оцінки. Однак, найшвидшим алгоритмом виявився метод таблиці доменів (Е). Такий результат обумовлений тим, що до самого пакету не генеруються токени ідентифікації, отже алгоритм працює швидше.

Наступний критерій ефективності є об’єм даних, що можуть бути оброблені сервером. Даний параметр, як було описано раніше, вимірюється чергою пакетів, що чекають на оброблення сервером. Результати досліджень критерію об’єму передачі даних відображено у таблиці 4.2.

Таблиця 4.2 – Результати вимірювання критерію об’єму передачі даних за тестовими випадками

Методи захисту	Об’єм необроблених даних стратегіями тестування (од.)					Загалом
	1	2	3	4	5	
Метод А	7	11	10	34	16	15,6
Метод Б	2	4	3	11	3	4,6
Метод В	1	2	2	10	2	3,4
Метод Г	0	2	1	6	1	2
Метод Д	7	19	14	51	25	23,2
Метод Е	2	3	3	9	3	4
Метод Ж	0	1	1	7	2	2,2
Метод И	4	6	5	15	4	6,8

Результати вимірювань повністю корелюють із результатами вимірювань швидкості передачі. Така поведінка обумовлена тим, що швидкість передачі впливає на чергу запитів, що чекає обробки. Таким чином найбільш неефективними виявились алгоритми підписання форм-відправки (Д) та cookie-файл двійної відправки (А).

З іншого боку, найефективнішим методом за даним критерієм виявився алгоритм захисних налаштувань cookie-файлів (Г). Такий результат обумовлений тим фактом, що налаштування cookie-файлів – це існуюча специфікація стандарту підпису cookie-файлів, отже технологія вбудована в усі новітні браузери та серверні рішення.

Наступний критерій ефективності є навантаження серверної частини, який вимірюється кількістю використаної пам’яті для обробки запитів в окремий момент часу при обробці. Результати досліджень критерію навантаження відображено у таблиці 4.3.

Таблиця 4.3 – Результати вимірювання завантаженості серверної частини при передачі даних за тестовими випадками

Методи захисту	Серверне навантаження при стратегіях тестування (кб.)					Загалом
	А	Б	В	Г	Д	
Метод А	52	95	76	361	162	149,2
Метод Б	34	63	49	249	87	96,4
Метод В	32	46	43	212	81	82,8
Метод Г	27	34	31	147	61	60
Метод Д	57	81	71	236	129	114,8
Метод Е	86	142	119	348	184	175,8
Метод Ж	33	47	39	119	46	56,8
Метод И	37	51	48	103	58	59,4

За результатами вимірювань можна побачити, як методи cookie-файл двійної відправки (А), підписання форм-відправки (Д) та таблиця доменів (Е) більш усього навантажують серверну частину. Таблиця доменів виявилась найбільш затратною в силу того, що дані по доменним зв’язкам зберігаються в оперативній пам’яті сервера.

Наступний критерій ефективності – кількість рядків коду для реалізації алгоритмів захисту. Він обумовлений трудомісткістю процесу створення та розміщення коду на багатьох мікросервісах серверної частини. Результати вимірювань рядків коду відображено у таблиці 4.4. Потрібно зауважити, що реалізація алгоритмів відбувається не тільки на серверній частині, але й на клієнтський. Тому підрахунок рядків проводиться, як на мікросервісах, так і на веб-сайті. Також реалізація алгоритмів може бути однаковою для усіх мікросервісі та просто дублюватись між собою, однак рядки реалізації усе одно підраховуються до суми.

Таблиця 4.4 – Результати вимірювання кількості рядків коду для реалізації методів захисту

Методи захисту	Кількість рядків коду		Кількість серверів	Загалом
	Клієнтська частина	Серверна частина		
Метод А	22	46	5	252
Метод Б	18	17	5	103
Метод В	17	17	5	102
Метод Г	0	6	5	30
Метод Д	0	131	1	131
Метод Е	0	42	5	210
Метод Ж	3	12	5	63
Метод И	3	12	5	63

За результатами можна побачити, що при реалізації алгоритму форм-відправки (Д), кількість серверів завжди однакова, незалежно від кількості мікросервісів. Це обумовлено тим, що відправка шаблонів сторінок завжди займається один сервер.

Останнім критерієм ефективності є захисні можливості алгоритмів. Даний критерій являється основоположним для даних методів та вимірюється кількістю атак від яких алгоритм може захистити. Аналіз захисту буде виконуватись на основі таких методів атаки виду CSRF:

– міжсайтова відправка (1): стандартна реалізація атаки через приховану форму на сторонньому вірусному сайті;

– no-referer або запит без підпису (2): навмисне видалення підпису відправника пакета даних, для унеможливлення встановлення посилання на вірусний сайт;

– spoof referer (3): підроблення підпису відправника;

– XSS ін’єкція (4): впровадження вірусного скрипту для отримання токену оригінальності та використанні його при міжсайтовій відправці;

– dangling markup (5): атака, що направлена на ін’єкцію HTML коду з метою зламати оригінальний код таким чином, щоб у результаті втручання пошкодилась форма відправки та запит був відправлений на сторону зловмисника. Таким чином зловмисник отримає необхідні дані для доступу до користувацького кабінету;

– birthday attack або атака дня народження (6): атака, що направлена на методи шифрування токену оригінальності. Якщо алгоритми не криптостійкі, то зловмисник може отримати токен користувача підбором;

– вразливий субдомен (7): дана атака потребує від зловмисника доступу до субдомена цільового серверу. Отримавши доступ, зловмисник може провести атаку міжсайтової відправки при якій цільовий сайт не зможе визначити вірусний сайт, як сайт з іншого домену, отже зловмисник не тільки виконає атаку, але й отримає відповідь від цільового сервера;

– вразливий PDF (8): атака, що передбачає загрузку файлу PDF на сервер, через який даний PDF файл можуть отримувати користувачі. При отримані такого вірусного PDF файлу, виконується вшитий вірусний код, який надсилає користувацький токен захисту до атакуючого;

– cookie injection або ін’єкція cookie-файлів (9): атака, при якій зловмисник має змогу додати до cookie-файлів захисний токен. Таким чином, цільовий сервер не зможе відлічити надісланий пакет від оригінального;

– change Content-Type (10): атака, при якій у пакеті даних змінюється заголовок на такий, що сервер приймає пакет не перевіряючи на оригінальність.

Наведені методи атак не лише реалізують вид атаки SCRF але й сприяють вилученню токена захисту, що також вражає наданий алгоритм захисту. Тобто, алгоритм захисту повинен бути таким, щоб перевірка оригінальності через токен не можливо було обійти.

У таблиці 4.5 приведено результати аналізу на захисні можливості алгоритмів перед різними реалізаціями атак виду SCRF.

Таблиця 4.5 – Результати аналізу захисту алгоритмів від наданих видів атак

Методи захисту	Захист від реалізації атаки виду CSRF										Загалом
	1	2	3	4	5	6	7	8	9	10	
Метод А	+	+	+	–	–	+	+	–	–	+	6
Метод Б	+	+	+	–	+	+	+	–	+	+	8
Метод В	+	+	+	–	+	+	+	–	+	+	8
Метод Г	+	–	–	–	–	+	–	+	–	+	4
Метод Д	+	+	+	–	–	+	+	–	–	+	6
Метод Е	+	–	–	–	+	+	–	+	+	+	6
Метод Ж	+	+	+	–	+	+	+	+	–	+	8
Метод И	+	+	+	+	+	+	+	+	+	+	10

У результаті аналізу методів захисту на захисні можливості від різних реалізацій атак маємо: алгоритм захисних налаштувань cookie-файлів (Г) виявився найбільш піддатливим до атак. Такий результат обумовлюється слабкою підтримкою даного методу захисту браузерами, отже його використання не бажане на даний час.

З іншого боку, алгоритм завірення передачі (И) найбільш безпечний, але з його реалізацією та використанням можуть виникнути проблеми. Так, як даний алгоритм потребує постійної перевірки для кожного запиту зі сторони стороннього сервісу, то він може навантажувати сервіс та зменшувати час відклику на запити. Даний алгоритм повільний при взаємодії із сервісам з декількох мікросервісів, отже може бути використаний лише при реалізації важливих взаємодій – таких, як передача коштів. Потрібно зауважити, що алгоритм не забезпечує конфіденційність передачі, а лише підтверджує оригінальність запиту.

Результати даного аналізу можуть відрізнятись залежно від середи тестування та способі реалізації алгоритмів захисту. В ході даного аналізу було

використано криптостійкі алгоритми захисту для реалізації кожного методу, також при аналізі був використаний веб-браузер «Google Chrome 51.0.2704.84».

4.3 Порівняльна оцінка метрик ефективності для методів та алгоритмів захисту

В ході аналізу методів захисту від атак виду CSRF були встановлені значення критеріїв ефективності для кожного методу відповідно. Тепер для кожного методу захисту можливо встановити вектор ефективності, заданий у формулі (2.5):

– cookie-файл двійної відправки: $R = < 59.4,\ 15.6,\ 149.2,\ 252,\ 6 >$;

– зашифрований токен: $R = < 31.8,\ 4.6,\ 96.4,\ 103,\ 8 >$;

– заголовок авторизації: $R = < 27.8,\ 3.4,\ 82.8,\ 102,\ 8 >$;

– захисні налаштування cookie-файлів: $R = < 28.6,\ 2,\ 60,\ 30,\ 4 >$;

– підписання форм-відправки: $R = < 101.8,\ 23.2,\ 114.8,\ 131,\ 5 >$;

– таблиця доменів: $R = < 26.4,\ 4,\ 175.8,\ 210,\ 6 >$;

– алгоритм клієнта-посередника: $R = < 38.8,\ 2.2,\ 56.8,\ 63,\ 8 >$;

– алгоритм завірення передачі: $R = < 49.4,\ 6.8,\ 59.4,\ 63,\ 10 >$.

Базуючись на отриманих даних можна провести порівняльну характеристику для методів захисту та виділити найкращі з поданих алгоритмів. Слід розуміти, що при порівнянні методів можливо виділити лише один метод на кожен встановлений критерій. Тобто, можливо лише обрати кращий метод за певним критерієм. Для обрання найкращого рішення з поданих можливо прибігти до двох методів: обрання найкращого за повторюваністю по критеріям чи визначення результуючої оцінки по вектору.

В таблиці 4.6 наведено порівняння векторів за критеріями ефективності:

– швидкість взаємодії (1);

– об’єм оброблюваних даних (2);

– навантаження серверної частини (3);

– розмір супроводжуючого коду (4);

– безпека запитів (5).

Таблиця 4.6 – Порівняльна таблиця методів захисту від атак виду CSRF за критеріями ефективності

Методи захисту	Критерії ефективності				
	1	2	3	4	5
Метод А	59,4	15,6	149,2	252	6
Метод Б	31,8	4,6	96,4	103	8
Метод В	27,8	3,4	82,8	102	8
Метод Г	28,6	2	60	30	4
Метод Д	101,8	23,2	114,8	131	5
Метод Е	26,4	4	175,8	210	6
Метод Ж	38,8	2,2	56,8	63	8
Метод И	49,4	6,8	59,4	63	10

З огляду на приведені результати модна встановити кращий з методів для кожного критерію:

– за критерієм швидкості найкращим методом у заданих параметрах середи буде метод з використанням таблиці доменів. Такий результат пояснюється легкістю впровадження зі сторони сервера, при якому перевірка пакетів здійснюється за допомогою таблиці доступних доменів;

– за критерієм об’єму оброблюваних даних найкращим методом захисту є захисні налаштування cookie-файлів. Такий результат обумовлений простотою імплементації алгоритму зі сторони серверу та простоти обробки зі сторони клієнта, адже дану технологію підтримують самі веб-браузери. Слід зазначити, що алгоритм клієнта-посередника також має оцінку достатньо наближену до найвищої з наданих, що може ставити даний метод на один рівень з методом захисного налаштування cookie-файлів;

– за критерієм навантаження серверної частини найкращим методом є алгоритм клієнта-посередника. Такий результат досягається самим алгоритмом роботи, адже серверна частина розвантажується на перевірку оригінальності, а перевіркою займається сторонній сервіс;

– за критерієм розміру супроводжуючого коду найкращим методом є захисні налаштування cookie-файлів. Як вказано раніше, такий результат обумовлений легкістю реалізації;

– за критерієм безпеки, основним критерієм для алгоритмів захисту, найкращим методом є алгоритм завірення передачі. Даний алгоритм, при належній реалізації зі сторони стороннього сервісу, захищає від усіх основних перерахованих способів атаки на основі CSRF атаки.

Якщо спробувати встановити найкращий алгоритм захисту на основі кількості найкращих показників по критеріям, то можна сказати, що метод захисних налаштувань cookie-файлів є найкращим з наданих. Однак даний алгоритм має найгірший показник захисту і у контексті вибору найкращого методу захисту від атак не може посісти дане місце. В такому випадку другим за частотою методом є алгоритм клієнта-посередника.

Вибір найкращого методу на основі оглядової порівняльної характеристики може бути не точним та не відповість на головне запитання – керуючись якою головною метрикою встановлюється найкращий алгоритм? Для такого підходу потрібно вивести оцінку для кожного методу захисту на основі наданих векторів.

Для цього потрібно знайти відношення кожного коефіцієнту одне до одного. Спочатку потрібно вивести усереднений вектор на основі даних векторів за формулою (2.7): $R_m =< 45.5,\ 7.725,\ 99.4,\ 119.25,\ 6.875 >$. Даний вектор є усередненим значенням від наданих векторів, отже може вважатись вектором середньостатистичного методу захисту. Якщо запропонувати шкалу оцінок для вимірювання методів захисту від 0 до 100 балів, то усереднений метод отримує 50 балів відносно інших методів. Також можна встановити найкращий вектор, тобто вектор з найкращими можливими показниками: $R_c =< 0,\ 0,\ 0,\ 0,\ 10 >$. Такі оцінки критеріїв обумовлені тим, що для перших чотирьох критеріїв правдиве

правило, що чим менше показник, тим краще метод і тим більше оцінка. Останній критерій – критерій безпеки, встановлюється від 0 до 10, де 10 – максимальна оцінка. Отже найкращий можливий метод захисту R_c має найвищу оцінку в 100 балів.

При заданих векторах методів захисту можна встановити відношення кожного критерію одне до одного з використанням рівняння множинної регресії, як показано у формулі (2.9). Встановимо рівняння регресії для заданих методів захисту по використаних критеріях ефективності:

$$Y = b_0 + b_1 K_{\text{шв.}} + b_2 K_{\text{об.}} + b_3 K_{\text{нв.}} + b_4 K_{\text{к.}} + b_5 K_{\text{б.}} \quad (4.1)$$

де b_i – коефіцієнт регресії, $K_{\text{шв.}}$ – критерій швидкості, $K_{\text{об.}}$ – критерій об'єму даних, $K_{\text{нв.}}$ – критерій навантаження, $K_{\text{к.}}$ – критерій розміру коду, $K_{\text{б.}}$ – критерій безпеки.

У результаті підрахунку коефіцієнтів регресії для нашого випадку маємо лінійне рівняння регресії:

$$Y = -27 + 0.1K_{\text{шв.}} - 1.3K_{\text{об.}} - 0.1K_{\text{нв.}} + 0.04K_{\text{к.}} + 12.6K_{\text{б.}} \quad (4.2)$$

З використанням коефіцієнтів рівняння до кожного критерію методів захисту можна встановити результуючі оцінки кожного методу відповідно:

– cookie-файл двійної відправки: $R = < 59.4,\ 15.6,\ 149.2,\ 252,\ 6 >$ = 29.5;

– зашифрований токен: $R = < 31.8,\ 4.6,\ 96.4,\ 103,\ 8 >$ = 65.5;

– заголовок авторизації: $R = < 27.8,\ 3.4,\ 82.8,\ 102,\ 8 >$ = 68;

– захисні налаштування cookie-файлів: $R = < 28.6,\ 2,\ 60,\ 30,\ 4 >$ = 19;

– підписання форм-відправки: $R = < 101.8,\ 23.2,\ 114.8,\ 131,\ 5 >$ = 10;

– таблиця доменів: $R = < 26.4,\ 4,\ 175.8,\ 210,\ 6 >$ = 37

– алгоритм клієнта-посередника: $R = < 38.8,\ 2.2,\ 56.8,\ 63,\ 8 >$ = 72;

– алгоритм завірення передачі: $R = < 49.4,\ 6.8,\ 59.4,\ 63,\ 10 >$ = 92.

Перед встановленням рейтингу для наданих методів захисту потрібно перевірити правильність виведеної формули лінійної регресії (4.2). Для цього підставимо у формулу значення середнього та максимального значень векторів: $R_m =< 45.5,\ 7.725,\ 99.4,\ 119.25,\ 6.875 >= 49$, $R_{\text{с}} =< 0,\ 0,\ 0,\ 0,\ 10 >= 99$. Дані показники наближені до необхідних та свідчать, про вірність формули для даних векторів. Однак для генеральної сукупності потрібно проаналізувати вектори за наданими параметрами системи та знайти коефіцієнт детермінації. Результати проведеного дослідження регресії наведено у таблиці 4.7.

Таблиця 4.7 – Регресійна статистика аналізу регресії

Регресійна статистика	
Множинний R	0,949624237
R-квадрат	0,928488475
Нормований R-квадрат	0,917574068
Стандартна похибка	0,152640958
Спостереження	10

У таблиці 4.8 наведено регресійний аналіз на основі даних векторів з критеріями ефективності. Даний аналіз проведений на основі поданих векторів та результуючих оцінок Y, які були виведені за формулою (4.2).

Таблиця 4.8 – Регресійний аналіз на основі критеріїв

Регресор	Коефіцієнти	Стандартна похибка	t-статистика	P-значення	Нижні 95%	Верхні 95%
Y	-26,154	0,375	-69,707	2,537E-07	-27,196	-25,113
$K_{\text{шв.}}$	0,069	0,007	10,185	0,0005	0,0503	0,088
$K_{\text{об.}}$	-1,182	0,028	-42,498	1,832E-06	-1,26	-1,105
$K_{\text{нв.}}$	-0,092	0,004	-21,398	2,820E-05	-0,108	-0,0799
$K_{\text{к.}}$	0,034	0,003	12,153	0,0003	0,026	0,0420
$K_{\text{б.}}$	12,52	0,0362	345,73	4,19935E-10	12,42	12,621

Результатом проведеного регресійного аналізу став графік нормального розподілення результуючих значень *Y* лінійного регресійного рівняння та середнього вибірки, що показано на рисунку 4.4.

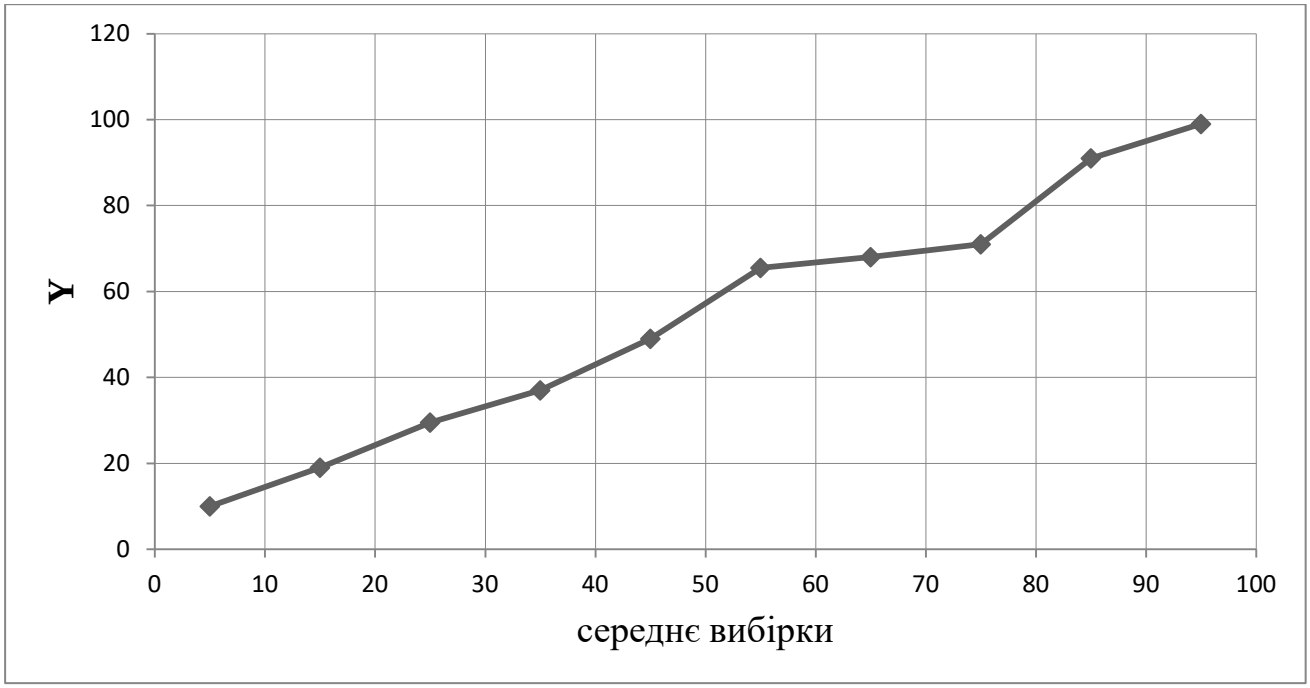

Рисунок 4.4 – Графік нормального розподілу

Результати проведеного регресивного аналізу задовольняють поставлену формулу (4.2) на основі якої були знайдені оцінки для методів захисту від атак виду CSRF. Також про адекватність моделі свідчить коефіцієнт детермінації, який дорівнює 0,92. Коефіцієнт детермінації (або R-квадрат) – це значення, що відображає адекватність моделі, тобто те, наскільки обрана модель пояснює залежність між досліджуваними параметрами, у нашому випадку критеріями ефективності. Високоякісною вважається модель, коефіцієнт детермінації якої вище або дорівнює 0,8.

За результатами проведеного аналізу можна сказати, що найкращим методом захисту є алгоритм завірення передачі, про що може свідчити його ступінь захисту від атак. Однак даний алгоритм тяжкий для реалізації та не завжди може підійти для кожного серверного рішення. Також слід зазначити, що дані результати та фінальні оцінки – значення, що були отримані при конкретних

параметрах досліджуваної моделі. Дані оцінки здебільшого стосуються серверів з мікросервісною архітектурою та для конкретної реалізації алгоритмів.

Наступним за оцінкою методом захисту є алгоритм клієнта-посередника. Даний алгоритм також посів перше місце серед порівняльної характеристики, проведеної вище. З урахуванням відносності використання алгоритму завірення передачі можна стверджувати, що алгоритм клієнта-посередника – найкращий алгоритм з двох використаних методів аналізу. Результуючі значення критеріїв ефективності даного методу обумовлений тим, що основну логіку перевірки бере на себе сторонній сервер, отже цільовий сервер розвантажується для виконання основної логіки, яку потребує клієнт.

Також слід зауважити третій по ефективності метод – алгоритм заголовків авторизації. Даний алгоритм один з найбільш розповсюджених та використовується у багатьох великих системах, що підтверджує його надійність. Даний алгоритм легкий у реалізації та не вимагає сторонніх сервісів для перевірки оригінальності. Однак потрібно розуміти, що якщо сторінки веб-сайту вразливі до атак виду XSS, то даний алгоритм втрачає свою працездатність.

ВИСНОВКИ

У результаті виконання атестаційної роботи було проведений докладний аналіз предметної області, проведено дослідження методів та алгоритмів захисту від атак виду CSRF та запропоновані захисні алгоритми.

Був проведений аналіз існуючих методів захисту, який складався із дослідження алгоритмів роботи кожного методу. На основі отриманих результатів аналізу було запропоновано алгоритми захисту з урахуванням особливостей серверних архітектур. Було проаналізовано відмінності дії алгоритмів із застосуванням різної серверної архітектури: монолітної та мікросервісної.

В результаті аналізу було встановлено критерії ефективності методів захисту, за якими можливо створити порівняльну характеристику алгоритмів. На базі поданих критеріїв були встановленні метрики та параметри тестових випадків для визначення кожного критерію окремо. В результаті було проведено тестування наданих методів захисту із встановленням метрик критеріїв для кожного окремого методу. За результатами визначених критеріїв біло проведено порівняльну характеристику та встановлені результуючі оцінки для наданих методів захисту із розробленими включно.

В процесі розробки алгоритмів захисту, з метою вирішення проблеми зменшення ефективності та збільшення навантаження на серверну частину, було вирішено використовувати сторонній сервіс, як основу для надання захисних функцій для серверної частини. Результатами реалізації стали два алгоритми на базі стороннього сервісу, що допомагають серверам із мікросервісною архітектурою розвантажити вузли передачі та обробки даних не втрачаючи при цьому в захисних здатностях відносно атак виду CSRF.

У ході роботи було виконано наступні задачі:

– встановлення та опис реалізації методів захисту від атак виду CSRF;

– визначення основних переваг та недоліків наданих методів з урахуванням різних серверних архітектур;

– пропонування, розробка та реалізація методів захисту для серверів із мікросервісною архітектурою;

– встановлення критеріїв ефективності для наданих методів захисту, встановлення метрик та параметрів для визначення математичної моделі дослідження;

– тестування наданих методів захисту з використанням обраних параметрів, виведення порівняльних тестових результатів;

– аналіз ефективності для захисних алгоритмів та визначення найбільш ефективних методів із заданими метриками та параметрами середи тестування.

В результаті аналізу і розробки поставлену задачу було виконано в заданому обсязі.

Слід зауважити, що дану тему потрібно аналізувати та досліджувати з використанням інших метрик та критеріїв, інших ситуацій в ході яких алгоритми захисту виявили себе, як неефективні для вирішення певної задачі. Необхідність досліджень даної області ґрунтується на постійному розвитку Інтернет-технологій, що у свою чергу приводить до створення нових методів реалізації атак виду CSRF.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. 100+ internet statistics and facts for 2020 [Електронний ресурс] / Портал websitehostingrating.com − Режим доступу: https://www.websitehostingrating.com /internet-statistics-facts/ – 07.10.2020 р. – Загол. з екрану.

2. 2020 Cyber Security Statistics The Ultimate List Of Stats, Data & Trends [Електронний ресурс] / Портал purplesec.us − Режим доступу: https://purplesec.us/resources/cyber-security-statistics/ – 07.10.2020 р. – Загол. з екрану.

3. Avijit Mallik, Abid Ahsan, Mhia Md. Zaglul Shahadat, Jia-Chi Tsou. Man-in-the-middle-attack: Understanding in simple words.//International Journal of Data and Network Science. 2019. №3. С. 77-92.

4. Shabnam Sharma . Study on Phishing Attacks//International Journal of Computer Applications. 2018. №182. С. 27-29.

5. B.B. Gupta, Pooja Chaudhary. Cross-Site Scripting Attacks: Classification, Attack, and Countermeasures (Security, Privacy, and Trust in Mobile Communications): навч. посіб. — США: CRC Press, 2020. — 170 с.

6. CWE-352: Cross-Site Request Forgery (CSRF) [Електронний ресурс] / Портал cwe.mitre.org − Режим доступу: https://cwe.mitre.org/data/definitions/ 352.html – 07.10.2020 р. – Загол. з екрану.

7. Cross-site request forgery (CSRF) [Електронний ресурс] / Портал portswigger.net − Режим доступу: https://portswigger.net/web-security/csrf – 07.10.2020 р. – Загол. з екрану.

8. Cross Site Request Forgery (CSRF) [Електронний ресурс] / Портал owasp.org − Режим доступу: https://owasp.org/www-community/attacks/csrf – 07.10.2020 р. – Загол. з екрану.

9. Cookie Values Used in Anti-CSRF Token [Електронний ресурс] / Портал owasp.org − Режим доступу: https://www.netsparker.com/web-vulnerability-

scanner/vulnerabilities/cookie-values-used-in-anti-csrf-token/ – 07.10.2020 р. – Загол. з екрану.

10. Martin Johns, Justus Winter. RequestRodeo: Client Side Protection against Session Riding.//OWASP AppSec Europe. 2006. Режим доступу: https://web.sec.uni-passau.de/members/martin/060531_owasp_appsec_requestrodeo_slides.pdf.

11. Aung Khant. A Most-Neglected Fact about Cross Site Request Forgery// YGN Ethical Hacker Group. 2010. С. 5.

12. EP1986395A1. Enhanced cross-site attack prevention. Германія. 2008. Режим доступу: https://worldwide.espacenet.com/patent/search/family/039673148 /publication/EP1986395A1?q=pn%3DEP1986395A1.

13. Kachko, O., Makutonina, L., Akolzina, O. Similar algorithm optimization for asymmetric encryption with the «overstretched parameters».//NTRU 2017 4th International Scientific-Practical Conference Problems of Infocommunications Science and Technology. 2017. С. 330-333.

14. Yiou Zhao. Optimizing Hash Strategy to Avoid Birthday Attack//Journal of Physics Conference Series. 2020. С. 7.

15. HttpOnly [Електронний ресурс] / Портал owasp.org − Режим доступу: https://owasp.org/www-community/HttpOnly – 25.10.2020 р. – Загол. з екрану.

16. M. Jones, J. Bradley, N. Sakimura, JSON Web Token (JWT). RFC 7519. 2015. С. 29.

17. The JSON Data Interchange Syntax. ECMA-404. Ecma International. №2. 2017. С. 7.

18. Безопасное программирование [Електронний ресурс] / Портал ptsecurity.com − Режим доступу: https://www.ptsecurity.com/upload/ptru/events /AppRoof-Offensive-Yunusov.pdf – 07.10.2020 р. – Загол. з екрану.

19. Web Storage (Second Edition) [Електронний ресурс] / Портал w3.org − Режим доступу: https://www.w3.org/TR/webstorage/ – 07.10.2020 р. – Загол. з екрану.

20. Kulbir Saini. Squid Proxy Server 3.1: Beginner's Guide: навч. посіб. — США: Packt, 2011. — 332 с.

21. Peter Sommerlad. Reverse Proxy Patterns//EuroPLoP. 2003. С. 27.

22. Cross Site Cookie Manipulation [Електронний ресурс] / Портал netsparker.com − Режим доступу: https://www.netsparker.com/blog/web-security/cross-site-cookie-manipulation/ – 07.10.2020 р. – Загол. з екрану.

23. M. Jones, B. Campbell, C. Mortimore. JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants. RFC 7519. 2015. С. 12.

24. Rowan Merewood. SameSite cookies explained [Електронний ресурс] / Портал web.dev − Режим доступу: https://web.dev/samesite-cookies-explained/ – 16.10.2020 р. – Загол. з екрану.

25. Ionut Arghire. Google Rolls Back Recently Introduced Chrome CSRF Protection [Електронний ресурс] / Портал securityweek.com − Режим доступу: https://www.securityweek.com/google-rolls-back-recently-introduced-chrome-csrf-protection – 16.10.2020 р. – Загол. з екрану.

26. Cross-Site Request Forgery Prevention Cheat Sheet [Електронний ресурс] / Портал owasp.org − Режим доступу: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html – 07.10.2020 р. – Загол. з екрану.

27. Гайдес М.А. Общая теория систем (системы и системный анализ). Вінниця: Глобус-пресс, 2005. — 201 с.

28. James Bottomley. Understanding Caching [Електронний ресурс] / Портал linuxjournal.com − Режим доступу: https://www.linuxjournal.com/article/7105 – 25.10.2020 р. – Загол. з екрану.

29. Models in Science [Електронний ресурс] / Портал stanford.edu − Режим доступу: https://plato.stanford.edu/entries/models-science/ – 25.10.2020 р. – Загол. з екрану.

30. Медовой О.Л. Контекстно-онтологічна модель та методи обчислення метрик інформаційної якості веб-орієнтованих систем. Херсон, Вестник Херсонского національного технического университета, №34, С. 162-165. 2009 р.

31. Ronald Walpole. Probability & Statistics for Engineers and Scientists: навч. посіб. — США. Prentice Hall, 2012. С. 171-172.