ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА ІНФОРМАЦІЙНОЇ ТА КІБЕРНЕТИЧНОЇ БЕЗПЕКИ

Пояснювальна записка

до бакалаврської роботи

на тему:

«ДОСЛІДЖЕННЯ ШЛЯХІВ ТА РОЗРОБКА РЕКОМЕНДАЦІЙ ЩОДО ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ RESTFUL WEB-СЕРВІСІВ»

Виконав студент 4 курсу, групи БСД-43
спеціальності 125 Кібербезпека
освітньо-професійної програми «Інформаційна та кібернетична безпека»
(шифр і назва спеціальності)
Бадзюк М.О.
(прізвище та ініціали)

Керівник _____
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Нормоконтролер _____
(прізвище та ініціали)

КИЇВ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Інститут ННІЗІ
Кафедра Інформаційної та кібернетичної безпеки
Ступінь вищої освіти Бакалавр
Спеціальність 125 Кібербезпека
Освітня програма Інформаційна та кібернетична безпека

ЗАТВЕРДЖУЮ
Завідувач кафедри ІКБ
_____ Гайдур Г.І.
“___”_____2023 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Бадзюку Максиму Олександровича
(прізвище, ім’я, по батькові)

1. Тема бакалаврської роботи: «Дослідження шляхів та розроблення рекомендацій щодо забезпечення безпеки RESTFUL web-сервісів»
керівник бакалаврської роботи _____
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «___»_____2023 року №_____.

2. Строк подання студентом бакалаврської роботи _____

3. Вихідні дані до бакалаврської роботи _____
використання REST API в мережі інтернету речей;
наукові статті про захист інформації в мережі;
технічна та література, експлуатаційна документація, нормативні документи, міжнародні стандарти.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Актуальність загроз та вразливостей в кіберпросторі.
2. Аналіз процесу реагування проблем захисту інформації в мережах інтернету.
3. Методи захисту від загроз та вразливостей.
4. Рекомендації щодо застосування методів та засобів захісту під час загроз та атак.

5. Перелік графічного матеріалу
1. Архітектури RESTful web-сервісів.
2. Загроз та та вразливостей в кіберпросторі.
3. Аналіз процесу реагування проблем захисту інформації в мережах інтернету.
4. Результати аналізу методів захисту від загроз та вразливостей.
5. Призначення та можливості рішення Wireshark та Fiddler.
6. Рекомендації щодо застосування методів та засобів захісту під час загроз та атак.
7. Висновки за результатами роботи.

6. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ зп	Назва етапів бакалаврської роботи	Строк виконання етапів бакалаврської роботи	Примітка
1.	Збір вихідних даних для дослідженням та аналіз даних та виявлення основних проблем,		
2.	Аналіз сучасних стандартів та рекомендацій.		
3.	Аналіз загроз та уразливостей в кіберпросторі.		
4.	Розроблення методів та підходів до забезпечення безпеки RESTful web-сервісів.		
5.	Оформлення результатів дослідження.		
6.	Підготовка демонстраційного матеріалу та доповіді.		

Студент _____ Бадзюк. М.О.
(підпис) прізвище та ініціали

Керівник бакалаврської роботи _____
(підпис) прізвище та ініціали

ВІДГУК РЕЦЕНЗЕНТА
на бакалаврську роботу

студента Бадзюка Максима Олександровича
на тему: «Дослідження шляхів та розроблення рекомендацій щодо забезпечення безпеки RESTFUL web-сервісів»

Актуальність: Забезпечення безпеки RESTful веб-сервісів є надзвичайно актуальною проблемою в сучасному інтернет-середовищі. REST є архітектурним стилем, який використовується для побудови розподілених систем, де веб-сервіси комунікують між собою через HTTP-протокол.

Позитивні сторони:

1. На основі проведеного аналізу в роботі було встановлено зміст процесу реагування на загроза та захисту конфідеційності даних
2. Було досліджено методи захісту та моніториг трафіку на прикладі рішення Wireshark, Fiddler.
3. Розроблення рекомендацій щодо застосування методів захисту під час розслідування загроз.
4. Текст пердставлений у належній мовній формі, він логічно послідовний та структурований. Зроблені чіткі та змістовні висновки. Графічний матеріал оформлено провесійно. Список використаної науково-технічної літератури свідчить про вміння автора ефективно користуватись джерелами, що стосуються теми його бакалаврської роботи.

Недоліки:

1. Недостатня методів шифрування RESTful веб-сервісів.
2. Експеримент захіст від загроз, бажано було провести, які відображають конкретний киберінцидент.

Висновок: Враховуючи недоліки, бакалаврська робота заслуговує оцінку **добре**, а студент **Бадзюк М.О.** – присвоєння кваліфікації: бакалавр з кібербезпеки за спеціалізацією Інформаційна та кібернетична безпека.

Якість роботи	
Виконано на замовлення підприємства	
Виконано за тематикою НДР	
Виконано з макетом	
Виконано з застосуванням ЕОМ та МПТ	√
Має практичну цінність	√
Проект-частина комплексної теми	

Підпис рецензента (_____)

Підпис засвідчую
Підпис особи, що засвідчує (_____)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

ПОДАННЯ
ГОЛОВІ ДЕРЖАВНОЇ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ БАКАЛАВРСЬКОЇ РОБОТИ

Направляється студент Бадзюк М.О. до захисту бакалаврської роботи
(прізвище та ініціали)
спеціальності 125 Кібербезпека
освітньо-професійної програми Інформаційна та кібернетична безпека
(шифр і назва спеціальності)
на тему: «Дослідження шляхів та розроблення рекомендацій щодо застосування засобів багатовимірного візуального аналізу під час розслідування кіберінцидентів».
Бакалаврська робота і рецензія додаються.

Директор інституту _____ Савченко В.А.
(підпис) (прізвище та ініціали)

Довідка про успішність

Бадзюк М.О. за період навчання в інституті
(прізвище та ініціали студента)
ННІЗІ з 2018 року по 2022 рік повністю виконав навчальний план за напрямом підготовки, спеціальністю з таким розподілом оцінок за:
національною шкалою: відмінно____%, добре____%, задовільно____%;
шкалою ECTS: A____%; B____%; C ____%; D____%; E____%.

Секретар інституту, факультету (відділення)_____ Журенко О.В.
(підпис) (прізвище та ініціали)

Висновок керівника бакалаврської роботи

Студент ***Бадзюк М.О.*** обрав тему роботи, метою якої було дослідження шляхів та розроблення рекомендацій щодо забезпечення безпеки RESTFUL web-сервісів. Перелік використаних джерел свідчить про вміння студентом розбиратись в наукових питаннях та застосовувати їх при дослідженнях. Під час виконання бакалаврської роботи Бадзюк М.О. показав добру теоретичну та практичну підготовку. Роботу виконував сумлінно, акуратно та вчасно за планом.
Все це дозволяє оцінити виконану бакалаврську роботу студента Бадзюка Максима Олександровича на оцінку **«добре»** та присвоїти йому кваліфікацію: бакалавр з кібербезпеки за спеціалізацією Інформаційна та кібернетична безпека.

Керівник бакалаврської роботи_____ _____.
(підпис) (прізвище та ініціали)
“____”_____2023 року

Висновок кафедри про бакалаврську роботу

Бакалаврська робота розглянута. Студент Бадзюк М.О.
(прізвище та ініціали)
Завідувач кафедри Інформаційної та кібернетичної безпеки
(назва)

_____ Гайдур Г.І.
(підпис) (прізвище та ініціали)

“____”_____2023 року

РЕФЕРАТ

Текстова частина бакалаврської роботи: 57 сторінок, 16 рисунків, 1 таблиця, 12 джерел.

Об'єкт дослідження – реагування на загрози та вразливість в RESTful web-сервісів.

Предмет дослідження – шляхи та рекомендації щодо забезпечення безпеки RESTful web-сервісів.

Мета роботи – Дослідження та розробка рекомендацій, спрямованих на забезпечення безпеки RESTful web-сервісів. Основними завданнями дослідження є аналіз потенційних загроз безпеці, ідентифікація ризиків, розробка та впровадження методів та технік для підвищення безпеки RESTful web-сервісів.

Методи дослідження – опрацювання літератури за даною темою, аналіз експлуатаційної документації, міжнародних стандартів та їх порівняння, проведення експерименту.

В роботі приведено основні відомості про рекомендацій щодо забезпечення безпеки RESTFUL web-сервісів. Проаналізовано різні види загроз та наведено їх класифікацію.

Досліджено методи та засоби захисту від угроз та аналізу забезпечення безпеки RESTFUL web-сервісів.

Досліджено можливості програмного комплексу Wireshark та Fiddler.

На основі досліджень проведених в роботі розроблено рекомендації щодо застосування методів та засоби захисту від угроз та аналізу забезпечення безпеки RESTFUL web-сервісів.

ЗМІСТ

Стор.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

REST– Representational State Transfer

RBAC – Role-Based Access Control

ABAC – Attribute-Based Access Control

RT– Request Tampering

DOS – Disk Operating System

DDoS – Distributed Denial-of-Service

MITM – Man-in-the-Middle

SSL – Secure Sockets Layer

TLS – Transport Layer Security

SOAP – Simple Object Access Protocol

JSON – JavaScript Object Notation

XML – eXtensible Markup Language

TTPS – HyperText Transfer Protocol Secure

XSS – Cross-Site Scripting

CSRF – Cross-Site Request Forgery

ВСТУП

У рамках дослідження будуть проаналізовані різні методи та стратегії, які можуть бути використані для забезпечення безпеки RESTful веб-сервісів. Будуть розглянуті питання автентифікації та авторизації, включаючи різні методи аутентифікації, такі як токени та OAuth2. Також будуть проаналізовані різні методи контролю доступу до ресурсів та методів, такі як RBAC та ABAC.

Крім того, у дослідженні будуть проаналізовані різні типи атак, на які можуть бути піддані RESTful веб-сервіси, такі як атаки на міжсайтове виконання скриптів (XSS), SQL-ін'єкції та інші. Будуть розглянуті різні методи та стратегії для запобігання цим атакам та забезпечення високого рівня безпеки RESTful веб-сервісів.

Отже, дане дослідження буде корисним для будь-яких розробників веб-додатків, які працюють з RESTful веб-сервісами та бажають забезпечити високий рівень безпеки своїх продуктів. Результати дослідження можуть бути використані для розробки рекомендацій та методів забезпечення безп еки RESTful веб-сервісів, які можуть бути використані в реальних проектах. Також дослідження може бути корисним для менеджерів проектів, які відповідають за безпеку продукту та хочуть бути в курсі останніх тенденцій та стратегій у галузі безпеки веб-додатків.

Дослідження шляхів та розробка рекомендацій щодо забезпечення безпеки RESTful веб-сервісів важливе не тільки з погляду захисту конфіденційної інформації та запобігання атакам, але й для збереження довіри користувачів та підтримки репутації бізнесу. Тому, враховуючи вагу цього питання, дослідження може стати важливим кроком для забезпечення високої якості та безпеки веб-додатків.

Важливість безпеки веб-додатків та сервісів стає все більш актуальною в світі швидкого розвитку технологій та зростаючої кількості кібератак. RESTful веб-сервіси є одним з найпоширеніших форматів веб-сервісів, які використовуються для обміну даними між різними системами. Однак, як і у будь-якому іншому типі

веб-додатків, існують ризики, пов'язані з безпекою, які можуть призвести до втрати конфіденційної інформації, викрадення даних користувачів, псування даних та інших проблем.

Дослідження шляхів та розробка рекомендацій щодо забезпечення безпеки RESTful веб-сервісів допоможе виявити найбільш поширені вразливості та ризики, пов'язані з цим типом веб-сервісів, а також надати рекомендації щодо їх запобігання та забезпечення безпеки. Результати дослідження можуть бути корисні для розробників, які створюють RESTful веб-сервіси, а також для тих, хто займається тестуванням та аудитом безпеки веб-додатків. Дослідження може включати огляд літератури та вивчення останніх тенденцій у галузі безпеки RESTful веб-сервісів, а також аналіз відомих вразливостей та ризиків, пов'язаних з цим типом веб-сервісів. Дослідження також може включати практичні випробування на прикладі реальних RESTful веб-сервісів для визначення можливих вразливостей та ризиків.

1 АНАЛІЗ ПРОБЛЕМИ ЗАХИСТУ RESTFUL WEB-СЕРВІСІВ

1.1 Аналіз основних загроз та вразливості RESTful web-сервісів

RESTful web-сервіси піддаються різним загрозам і вразливостям, що можуть призвести до порушення безпеки та конфіденційності даних. Основні загрози та вразливості RESTful web-сервісів:

Атаки на міжсайтовий запит на підняття привілеїв (XSPA): XSPA - це атака, при якій зловмисник використовує підроблений запит, щоб отримати доступ до конфіденційної інформації або отримати підвищені привілеї доступу.

Основна ідея атаки XSPA полягає у використанні вразливостей у політиці безпеки браузера, який дозволяє виконувати запити на інші порти, ніж той, на якому розташований веб-сервер. Зловмисник використовує JavaScript або інші технології для виконання таких запитів через браузер потерпілого користувача.

Основні наслідки атаки XSPA можуть включати:

1. Несанкціонований доступ до сервісів, які прослуховують на інших портах, що може призвести до витоку конфіденційної інформації.

2. Запуск внутрішніх сервісів з вразливостями, які можуть бути використані для подальшого злому або атак на інші системи.

Атака Request Tampering є одним із типів атак на RESTful веб-сервіси, при якій зловмисник змінює вміст або параметри запитів, що надсилаються до сервера, з метою отримання несанкціонованого доступу або впливу на поведінку сервісу.

Процес атаки RT. може включати наступні етапи:

1. Спостереження: Зловмисник спостерігає і аналізує комунікацію між клієнтом і сервером, щоб зрозуміти структуру та параметри запитів.

2. Зміна запитів: Зловмисник змінює параметри запитів, включаючи значення, заголовки, шляхи URL або методи, з метою виконання несанкціонованих дій або отримання чутливої інформації.

3. Перехоплення відповідей: Зловмисник може також перехопити

відповіді від сервера, змінити їх або вплинути на них перед поверненням до клієнта.

Деякі наслідки атаки Request Tampering можуть включати:

- Несанкціонований доступ до конфіденційних даних або ресурсів сервера.
- Виконання дій від імені автентифікованого користувача, наприклад, зміна пароля або внесення змін в профіль.
- Зміна логіки додатку або порушення його нормальної роботи.

Неавторизований доступ: Незаконне отримання доступу до ресурсів сервісу, що може призвести до розкриття конфіденційної інформації або недозволених дій з даними.

Атаки перехоплення: Перехоплення передачі даних між клієнтом і сервером, що може призвести до зловживання аутентифікаційними даними або модифікації переданих даних.

Вразливості введення даних: Недостатня перевірка та очищення даних, надсиланих клієнтами, може призвести до вразливостей, таких як SQL-ін'єкції або виконання зловмисних скриптів на сервері.

Недостатня аутентифікація та управління доступом: Недостатньо сильні механізми аутентифікації та контролю доступу можуть дозволити зловмисникам використовувати незахищений API або обмежувати доступ до ресурсів.

Відмова анатка DoS: Спроби перевантажити сервер запитами, що призводять до відмови в обслуговуванні для легітимних користувачів.

Бати участь в атаці можуть будь-які пристрої, які підключаються до мережі Інтернет і вміють відправляти запити: смартфони, смарт-годинник, побутова техніка, хостингові сервери. Але знайти і організувати тисячу охочих провести атаку важко. Тому хакери перетворюють комп’ютери звичайних користувачів у “зомбі” і керують ними здалеку. Зазвичай це роблять за допомогою вірусів. Такі віруси не вимагають нічого від користувача і не видають себе, тому власник пристрою може навіть не підозрювати, що бере участь в атаці.

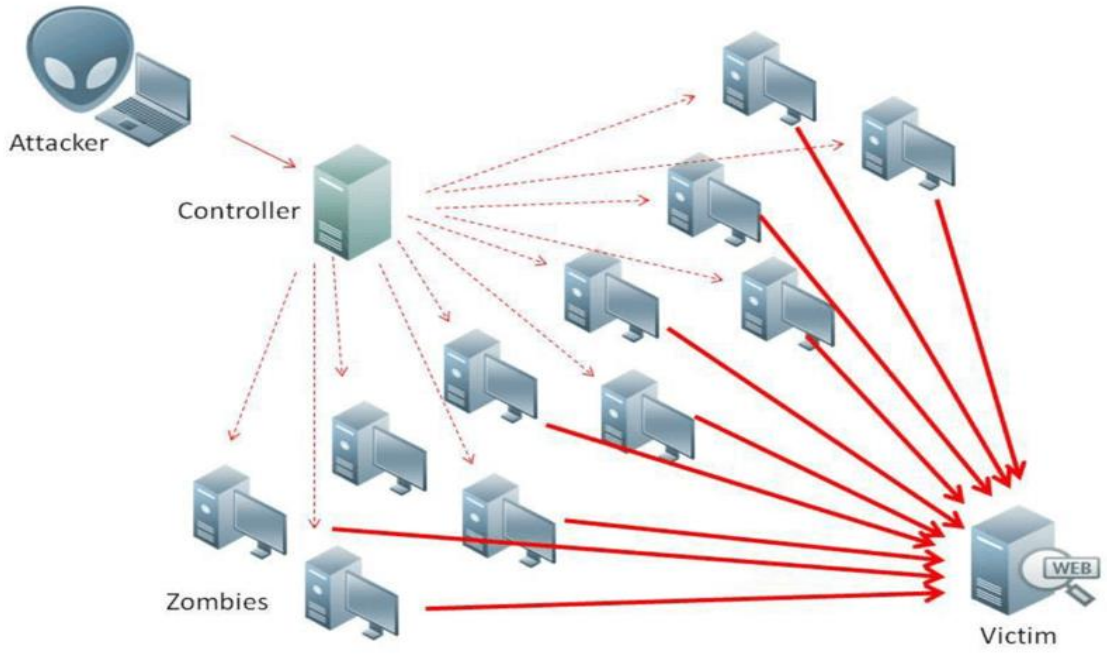

Рис. 1.1 Схема DDos атаки

Під час DDoS-атаки низка ботів або ціла бот-мережа перевантажує веб-сайт чи службу запитами HTTP й трафіком. По суті, кілька комп’ютерів намагаються захопити один пристрій, щоб реальні користувачі втратили до нього доступ. Як наслідок можуть виникати затримки в обслуговуванні або інші перебої.

Також під час атаки хакери також можуть проникнути у вашу базу даних і отримати доступ до конфіденційної інформації. DDoS-атаки можуть скористатися вразливостями системи безпеки та націлюватися на будь-яку кінцеву точку, яка загальнодоступна в Інтернеті.

Атаки "відмова в обслуговуванні" можуть тривати кілька годин або навіть днів. Одна така атака може спричинити порушення кількох типів. Ці атаки можуть загрожувати особистим і робочим пристроям.

Типи DDoS-ата:

Існує три основні категорії DDoS-атак: об’ємні атаки, атаки на рівні протоколу й атаки на прикладному рівні.

1. Під час об’ємної атаки зловмисник переповнює мережевий рівень тим, що спочатку видається нормальним, – трафіком. Це найпоширеніша форма DDoS-атаки. Прикладом такої атаки може слугувати "Посилення DNS-сервера" (DNS

amplification), коли через відкриті DNS-сервери цільовий об’єкт переповнюється трафіком із відповідей на запити DNS.

2. Атака на рівні протоколу призводить до переривання роботи служби, використовуючи вразливості на 3 й 4 рівнях. Прикладом такої атаки є SYN-атака, яка використовує всі доступні серверні ресурси, що робить сервер недоступним.

3. Атака ресурсного (прикладного рівня) спрямовується на веб-програми й зриває передавання даних між хостами. До таких атак належать порушення протоколу HTTP, внесення SQL-ін’єкції, міжсайтові сценарії та інші атаки рівня 7.

Кіберзловмисники можуть виконувати одну або кілька атак на мережу. Наприклад, атака може розпочатися в одному вигляді, а потім перетворитися на іншу загрозу або поєднатися з попередньою, щоб завдати шкоди системі.

До кожної категорії можна віднести безліч кібератак. Кількість нових кіберзагроз буде невпинно зростати, оскільки зловмисники створюють нові способи завдати шкоди.

Вразливості сторонніх бібліотек та фреймворків: Використання ненадійних або застарілих сторонніх компонентів може створювати вразливості, які можуть бути використані зловмисниками для атак.

Недостатня захист від перекриття атак: Відсутність заходів захисту від атак, таких як перекриття аутентифікаційних сесій або подвійні аутентифікаційні механізми, може використовуватися зловмисниками для незаконного доступу до ресурсів.

Вразливості сторонніх компонентів: Використання сторонніх бібліотек, фреймворків або інших компонентів може створювати вразливості, які можуть бути використані для атак.

Недостатня моніторинг та логування: Недостатній контроль та аналіз журналів подій можуть призвести до пропущення підозрілих активностей або затримки у виявленні інцидентів.

Недостатня безпека комунікації: Використання незахищених каналів зв'язку або відсутність шифрування може призвести до перехоплення або зловживання передачі даних.Недостатній контроль версій та конфігурації: Недостатній контроль

версій і конфігурацій сервісу може призвести до використання застарілих та вразливих компонентів або неправильних налаштувань, що збільшує ризик атак.

Недостатня усвідомленість персоналу: Недостатня підготовка та усвідомлення персоналу щодо безпеки може призвести до недоумовлення з приводу найкращих практик, викладених відповідними документами, а також до введення людських помилок, що збільшує ризик загроз.

1.2 Аналіз в RESTful web-сервісів перехоплення пакетів

Аналіз в RESTful web-сервісів перехоплення пакетів включає в себе відстеження та аналіз мережевого трафіку, що передається між клієнтом та сервером. Для цього можна використовувати різноманітні інструменти, наприклад, Wireshark або Fiddler.

Wireshark - це безкоштовний інструмент для перехоплення та аналізу мережевого трафіку. Він дозволяє перехоплювати пакети, які відправляються між клієнтом та сервером, та докладно аналізувати їх зміст. За допомогою Wireshark можна перевірити, які HTTP-запити та відповіді відправляються між клієнтом та сервером, а також перевірити, які заголовки HTTP використовуються.

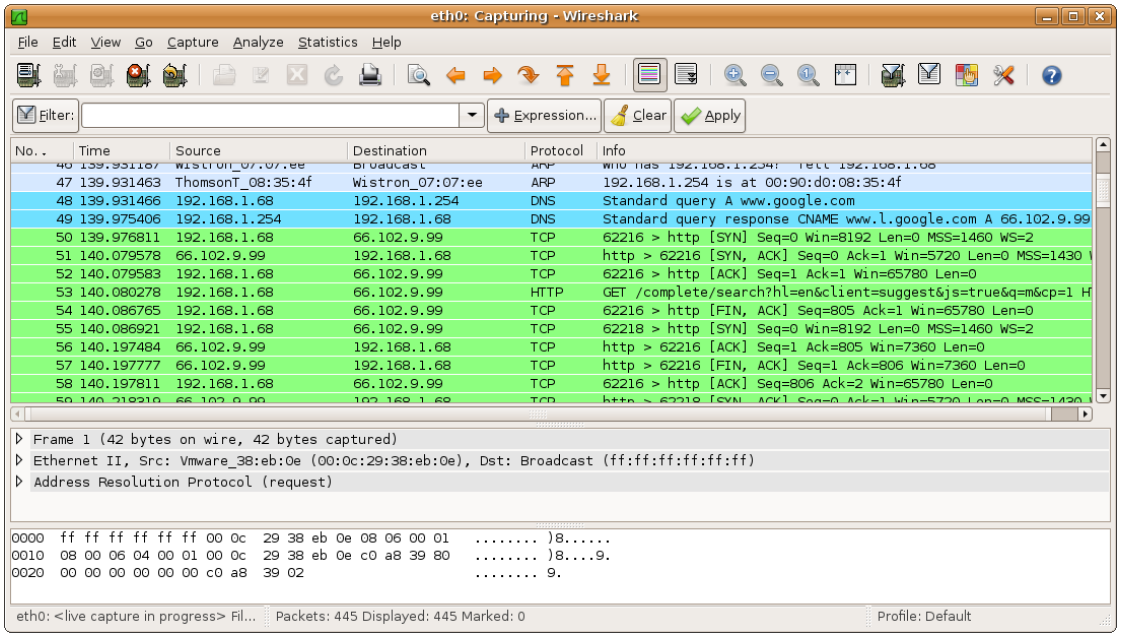

Рис. 1.2. Wireshark аналізу мережевого трафіку та моніторинг

Fiddler - це інструмент для перехоплення та аналізу HTTP-запитів та відповідей. Він дозволяє перехоплювати HTTP-запити та відповіді, які відправляються між клієнтом та сервером, та докладно аналізувати їх зміст. За допомогою Fiddler можна перевірити, які заголовки HTTP використовуються, та які параметри відправляються в тілі запиту.

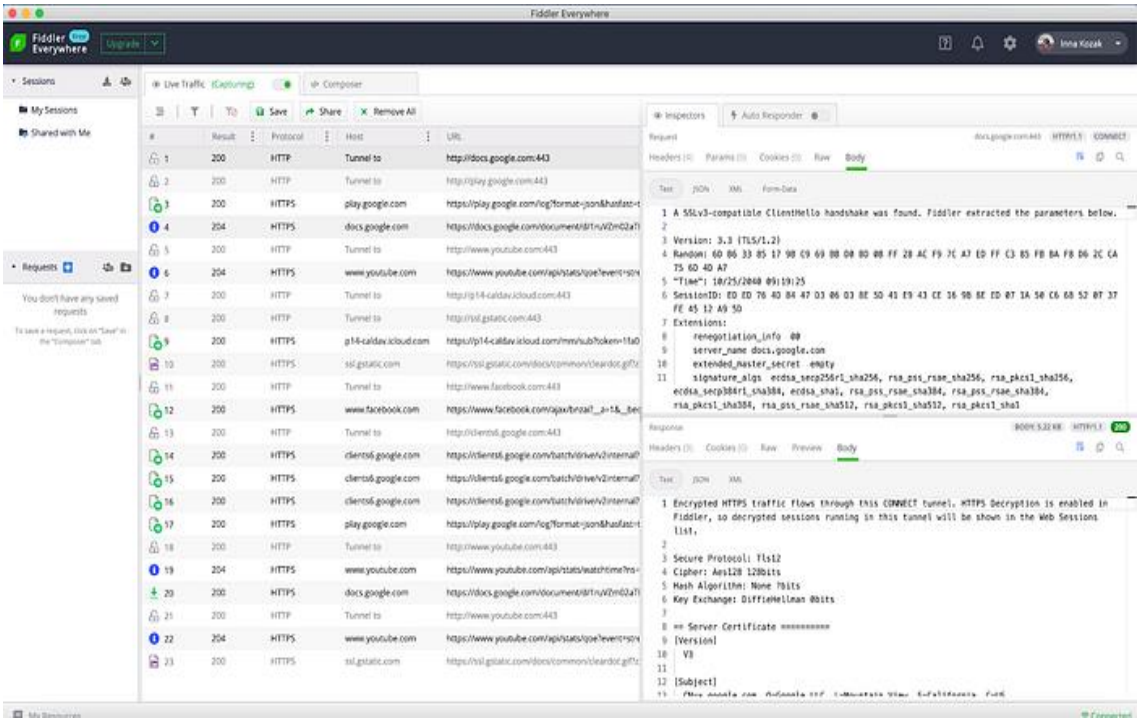

Рис. 1.3. Fiddler аналізу мережевого трафіку та моніторинг

Вони надають зручний інтерфейс для створення запитів та аналізу відповідей, і дозволяють швидко перевірити різні сценарії взаємодії з сервером.

При перехопленні пакетів можна отримати доступ до HTTP запитів та відповідей, що надсилаються між клієнтом та сервером. Це дає можливість аналізувати різні аспекти взаємодії, такі як:

параметри запиту: можна перевірити, які параметри передаються в запиті, і як вони обробляються на сервері;

формат відповіді: можна перевірити, які типи відповідей повертає сервер (наприклад, JSON або XML), і як вони обробляються на клієнті;

коди статусу: можна перевірити, які коди статусу повертає сервер, і як вони обробляються на клієнті.

Атака MITM є серйозною загрозою для безпеки REST API і передбачає

втручання зловмисника в комунікацію між клієнтом і сервером. Під час MITM-атаки зловмисник отримує контроль над комунікаційним каналом і може перехоплювати, модифікувати або навіть відхиляти передачу даних між двома сторонами.

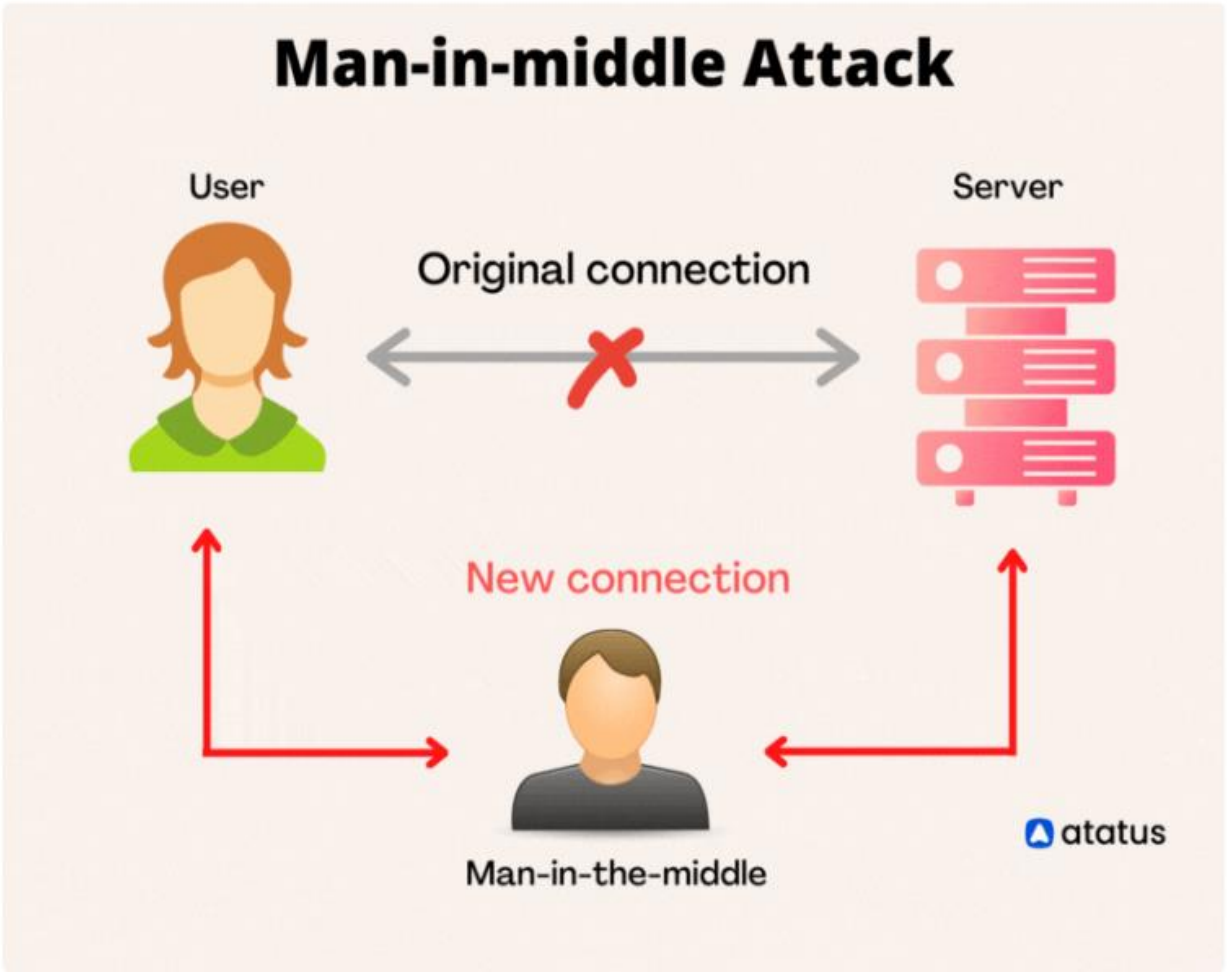

Рис. 1.4. Схема атака посередника

Окрім того, під час аналізу веб-сервісів можна звернути увагу на такі аспекти:

- Аутентифікація та авторизація: можна перевірити, які механізми аутентифікації та авторизації використовуються на сервері, та як вони застосовуються до запитів від клієнта;
- Обмеження доступу: можна перевірити, які обмеження доступу до ресурсів встановлені на сервері, та як вони застосовуються до запитів від клієнта;
- Обробка помилок: можна перевірити, як сервер обробляє помилки, що виникають під час взаємодії з ним, та як він повідомляє про ці помилки клієнту.

Таким чином, аналіз в RESTful web-сервісів перехоплення пакетів дозволяє зрозуміти, як саме взаємодіє клієнт з сервером, виявити можливі проблеми, та покращити якість та надійність веб-сервісу.

1.3 Аналіз XSS та CSRF атак по RESTful web-сервісів

XSS - це атака, при якій зловмисник вводить код скрипту на веб-сторінці, який виконується в браузері користувача. Якщо web-сервіс не валідує введені дані користувача перед відображенням їх на сторінці, зловмисник може ввести шкідливий код скрипта, який виконається в браузері жертви.

Етапи XSS атаки:

1 Зловмисник створює спеціальний код, який вбудовується в вхідні дані, які будуть передані на сервер через RESTful API.

2 Коли користувач взаємодіє зі сторінкою, яка використовує вхідні дані, на його браузер завантажується скомпрометований код, який може виконувати шкідливі дії, такі як крадіжка cookie-файлів, перенаправлення на інші сторінки, відправка конфіденційної інформації з браузера на сервер зловмисника.

3 Якщо користувач має права на виконання дій в системі, то скомпрометований код може виконувати ці дії від імені користувача, що може призвести до виконання несанкціонованих дій.

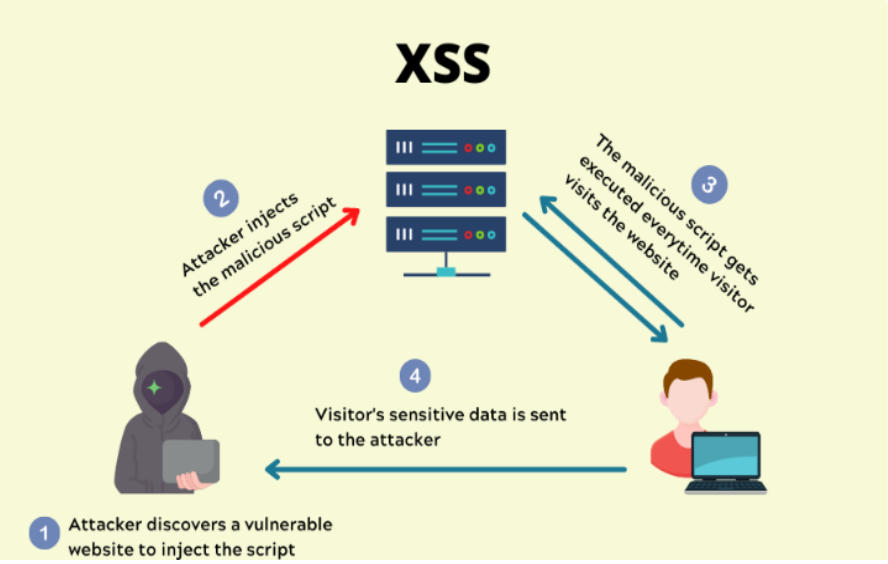

Рис. 1.5. Схема атаки XSS

Типи XSS-атак:

- Stored XSS - уразливість полягає в тому, що зловмисник зберігає шкідливий скрипт на сервері. Коли користувач звертається до сторінки з цим скриптом, він виконується.
- Reflected XSS - уразливість полягає в тому, що зловмисник використовує спеціальну посилання або форму, яка містить скомпрометовані дані. Ці дані відображаються на сторінці та виконуються в браузері користувача.
- DOM-based XSS - уразливість полягає в тому, що скрипт виконується в об'єктній моделі документу (DOM) на стороні клієнта. Зловмисник може впливати на DOM, вбудовуючи вхідні дані в скрипт, який виконується на сторінці.
- Blind XSS - уразливість полягає в тому, що зловмисник може вбудовувати шкідливий код в вхідні дані, які не відображаються на сторінці, але використовуються на сервері.

CSRF- це атака, при якій зловмисник намагається змусити авторизованого користувача виконати дію в системі, яку він не хоче виконати. Наприклад, зловмисник може надіслати листа-спаму з посиланням на зловмисну сторінку, яка містить форму, яка буде відправляти запит на виконання певної дії в системі. Якщо користувач авторизований в системі, його браузер автоматично виконає запит на виконання дії, і зловмисник зможе отримати доступ до конфіденційних даних або виконати несанкціоновані дії.

Етапи CSRF атаки:

1 Зловмисник створює спеціальний код, який вбудовується на зловмисну сторінку.

2 Коли користувач відвідує зловмисну сторінку, скрипт автоматично відправляє запит на RESTful API з підробленим CSRF-токеном, що змушує сервер виконати несанкціонований запит.

3 Якщо запит був успішно виконаний, то зловмисник може отримати доступ до конфіденційної інформації, виконати дії від імені користувача, змінити чи видалити дані в системі.

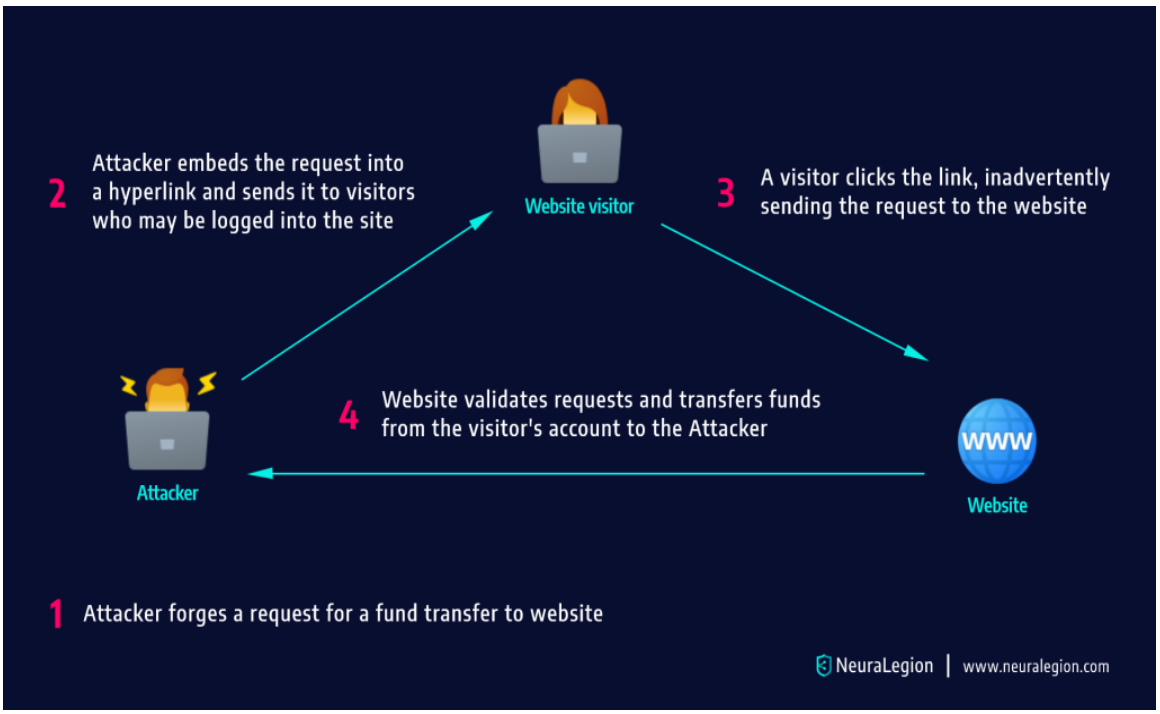

Рис. 1.6. Схема атаки CSRF

Типи CSRF атаки:

- GET-атака: в цьому випадку зловмисник надсилає посилання на вразливий ресурс, що містить запит HTTP GET, який автоматично виконується на сервері з уразливим додатком. Цей тип атаки може бути особливо небезпечним, оскільки браузер автоматично відкриває посилання, що призводить до виконання запиту без знання користувача.
- POST-атака: цей тип атаки полягає в тому, що зловмисник створює форму на своєму сайті, яка надсилає запит HTTP POST до вразливого додатку, виконуючи несанкціоновані дії в системі без знання користувача.
- PUT-атака: цей тип атаки використовує запит HTTP PUT для внесення змін до існуючого ресурсу на сервері.
- DELETE-атака: цей тип атаки використовує запит HTTP DELETE для видалення існуючого ресурсу на сервері.
- Ajax-атака: цей тип атаки використовує технологію Ajax для відправки запиту на сервер без оновлення сторінки, що робить атаку менш помітною для користувача.

XSS та CSRF-атаки є одними з найбільш поширених вразливостей веб-

додатків. Ці атаки можуть призвести до втрати конфіденційної інформації, втрати грошей, виконання несанкціонованих дій в системі та інших наслідків.

1.4 Аналіз атаки Buffer Overflow на RESTful web-сервісів

Атака Buffer Overflow є одним з найбільш поширених типів атак на програмне забезпечення, включаючи RESTful web-сервіси. Ця атака використовується для незаконного переповнення буфера пам'яті з метою виконання шкідливого коду або отримання несанкціонованого доступу до системи.

Основний принцип атаки Buffer Overflow полягає у введенні більшого обсягу даних у буфер, ніж він може вмістити. Це може статися, коли програмне забезпечення не виконує достатньої перевірки розміру введених даних. Як результат, перевищений обсяг даних переповнює буфер і може перезаписати важливі дані або вказівники на виконання коду.

У випадку RESTful web-сервісів, Buffer Overflow може виникати внаслідок некоректного оброблення вхідних даних, переданих у запиті. Наприклад, якщо вхідні дані не валідуються або обмежуються у межах допустимих розмірів, атакувач може надіслати запит зі спеціально сформованими даними, які призведуть до переповнення буфера і виконання шкідливого коду.

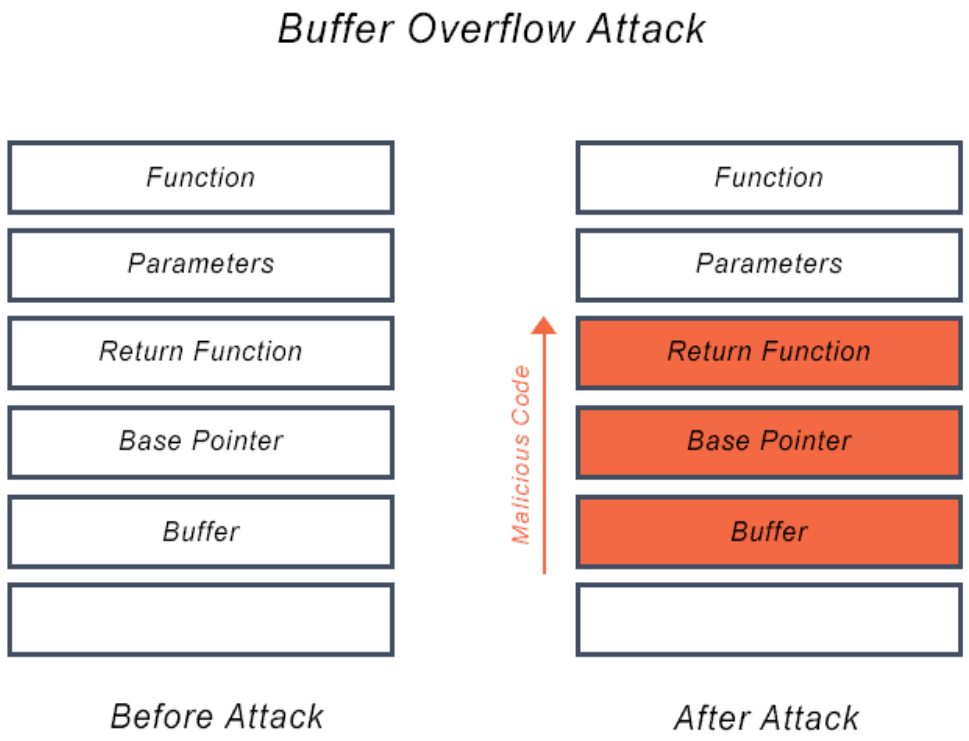

Рис. 1.7. Схема Buffer Overflow

Атака Buffer Overflow може мати серйозні наслідки, включаючи:

- Виконання шкідливого коду: Зловмисник може внедрити та виконати свій власний шкідливий код, що може призвести до небажаних наслідків, таких як отримання незаконного доступу до системи, крадіжка конфіденційних даних або виконання шкідливих дій на сервері.
- Відмова в обслуговуванні: Переповнення буфера може призвести до відмови в обслуговуванні системи або краху, оскільки це може призвести до виконання некоректного коду або виконання надмірно великих операцій, що використовують багато ресурсів.
- Компрометація конфіденційності: Шкідливий код, що виконується через Buffer Overflow, може отримати доступ до конфіденційної інформації, такої як паролі, ключі шифрування, персональні дані користувачів або інша чутлива інформація.

- Виконання привілейованих операцій: Зловмисник, якщо вдасться виконати шкідливий код через Buffer Overflow, може отримати привілеї адміністратора або іншого привілейованого користувача, що дасть їм повний контроль над системою.
- Зловживання вразливостей: Зловмисник може використовувати вразливість Buffer Overflow як початкову точку для виконання інших атак або зловживання інших системних слабкостей, що може призвести до додаткових компрометацій або злому системи. Виконання шкідливого коду: Зловмисник може внедрити та виконати свій власний шкідливий код, що може призвести до небажаних наслідків, таких як отримання незаконного доступу до системи, крадіжка конфіденційних даних або виконання шкідливих дій на сервері.
- Відмова в обслуговуванні: Переповнення буфера може призвести до відмови в обслуговуванні системи або краху, оскільки це може призвести до виконання некоректного коду або виконання надмірно великих операцій, що використовують багато ресурсів.
- Компрометація конфіденційності: Шкідливий код, що виконується через Buffer Overflow, може отримати доступ до конфіденційної інформації, такої як паролі, ключі шифрування, персональні дані користувачів або інша чутлива інформація.
- Виконання привілейованих операцій: Зловмисник, якщо вдасться виконати шкідливий код через Buffer Overflow, може отримати привілеї адміністратора або іншого привілейованого користувача, що дасть їм повний контроль над системою.

Зловживання вразливостей: Зловмисник може використовувати вразливість Buffer Overflow як початкову точку для виконання інших атак або зловживання інших системних слабкостей, що може призвести до додаткових компрометацій або злому системи

1.5 Аналіз атаки Information Leakage на RESTful web-сервісів

Атака Information Leakage в RESTful web-сервісах відбувається, коли

зловмисник отримує чутливу інформацію, яку не повинен бути здатен отримати, шляхом витоку даних з веб-сервісу. Це може мати серйозні наслідки для безпеки та конфіденційності даних. Ось докладний аналіз атаки Information Leakage:

- Витік конфіденційних даних: Атака Information Leakage може призвести до витоку конфіденційних даних, таких як особиста інформація користувачів, паролі, номери кредитних карт або інші конфіденційні дані. Якщо зловмисник отримує доступ до такої інформації, це може призвести до крадіжки ідентичності, фінансових втрат або інших негативних наслідків.
- Розкриття бізнес-логіки: Атака Information Leakage може також розкрити бізнес-логіку або внутрішню структуру веб-сервісу. Зловмисник може отримати доступ до інформації про роутинг, параметри запитів, обробку даних або інші деталі, що можуть допомогти відновити або зловживати функціональністю сервісу.
- Зловживання вразливостей: Атака Information Leakage може дати зловмисникам інформацію про вразливості веб-сервісу. Зловмисник може аналізувати отриману інформацію, щоб знайти слабкі місця в системі та використовувати їх для подальших атак, таких як вторгнення, перехоплення сесій або злому системи.
- Порушення вимог регуляторних стандартів: Атака Information Leakage може призвести до порушення вимог регуляторних стандарті.
- Підкопання довіри до сервісу: Якщо користувачі відкриваються до атак Information Leakage, це може підірвати їх довіру до веб-сервісу. Якщо відомо, що сервіс витікає конфіденційні дані або недостатньо захищений, користувачі можуть уникати використання цього сервісу, що може негативно вплинути на репутацію та бізнесовий успіх організації.
- Витік внутрішньої інформації: Атака Information Leakage може спричинити витік внутрішньої інформації про інфраструктуру, налаштування серверів, баз даних, ключі шифрування та іншу чутливу інформацію, яка повинна залишатися конфіденційною. Це може зробити сервіс більш вразливим до подальших атак.

- Зниження ефективності оборонних заходів: Якщо зловмисник отримує інформацію про оборонні заходи та механізми безпеки, використовувані веб-сервісом, він може швидше розробити стратегію атаки, уникнути виявлення або обійти заходи захисту.
- Витік системної інформації: Атака Information Leakage може розкрити системну інформацію про операційну систему, версію серверного програмного забезпечення, використовувані протоколи та інші деталі, які можуть бути використані для подальшого сканування та атак на систему.

Рис. 1.8. Крадіжка даних та інформації

1.6 Визначення ролі і місця RESTful веб-сервісів інформаційній системі

RESTful веб-сервіси є важливою складовою сучасних інформаційних систем. Вони дозволяють забезпечувати обмін даними між різними компонентами системи, що працюють на різних платформах та мовах програмування. RESTful веб-сервіси можуть використовуватися для забезпечення комунікації між різними додатками та сервісами, для обміну даними між сервером та клієнтом, для реалізації мікросервісної архітектури та багатьох інших випадків використання.

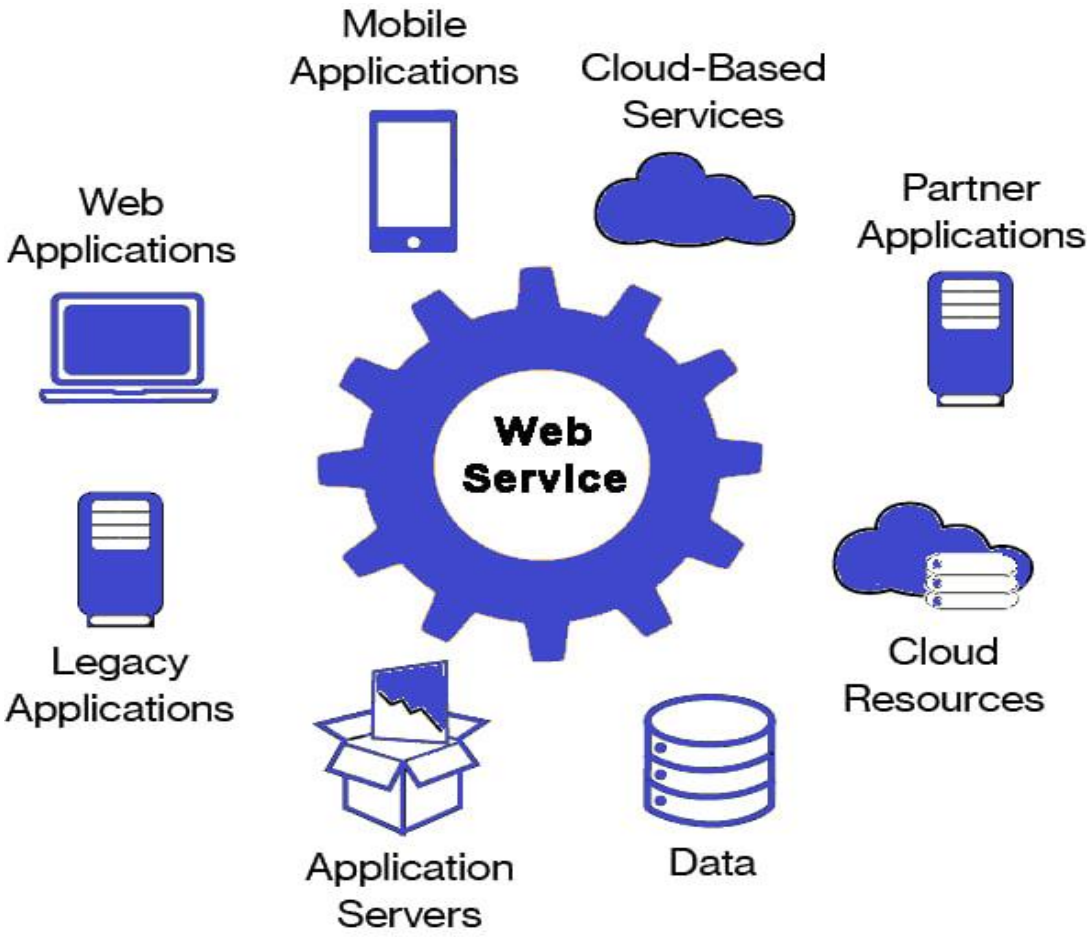

Рис. 1.9. Схема роль місця RESTful веб-сервісів інформаційній системі.

Основною роллю RESTful веб-сервісів в інформаційній системі є забезпечення інтеграції різних компонентів та сервісів. Завдяки стандартизованому протоколу HTTP та формату даних JSON або XML, RESTful веб-сервіси дозволяють легко і ефективно обмінюватися даними між різними додатками та сервісами, незалежно від платформи та мови програмування, що використовуються.

RESTful веб-сервіси також можуть забезпечувати безпеку та захист даних в інформаційній системі. За допомогою протоколу HTTPS, аутентифікації користувачів та шифрування даних, RESTful веб-сервіси дозволяють зменшити ризик витоку даних та забезпечити безпеку обміну даними.

Таким чином, RESTful веб-сервіси грають важливу роль в інформаційних системах, забезпечуючи інтеграцію та обмін даними між різними компонентами системи, а також забезпечуючи безпеку та захист даних.

Крім того, RESTful веб-сервіси дозволяють створювати гнучкі та масштабовані інформаційні системи. За допомогою RESTful веб-сервісів можна легко додавати нові функції та можливості до системи, не впливаючи на існуючі компоненти. Також RESTful веб-сервіси дозволяють масштабувати систему в залежності від потреб користувачів та об'єму даних.

У зв'язку з ростом кількості та складності інформаційних систем, зростає також і значення безпеки та захисту RESTful веб-сервісів. Недостатня захищеність може призвести до витоку даних, порушення конфіденційності, цілісності та доступності даних. Тому, важливо дослідити шляхи та розробити рекомендації щодо забезпечення безпеки RESTful веб-сервісів, що дозволить зменшити ризики та підвищити рівень захисту інформації в інформаційній системі.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ТА ЗАСОБІВ ЗАХИСТУ В RESTFUL WEB-СЕРВІСІВ

2.1 Аналіз засобів захисту

Основними засобами захисту RESTful web-сервісів є наступні:

Аутентифікація - цей механізм дозволяє перевірити, що клієнт, який намагається отримати доступ до ресурсів RESTful web-сервісу, є дійсною особою або системою, яка має право на доступ. Для цього можуть використовуватися різноманітні методи аутентифікації, такі як базова аутентифікація, формулярна аутентифікація, токени, OAuth та інші.

Авторизація - цей механізм дозволяє обмежити доступ до ресурсів RESTful web-сервісу для певних користувачів або груп користувачів на основі їх ролей або інших параметрів. Для цього можуть використовуватися різноманітні методи авторизації, такі як рольова авторизація.

Різниця між аутентифікацією і авторизацією

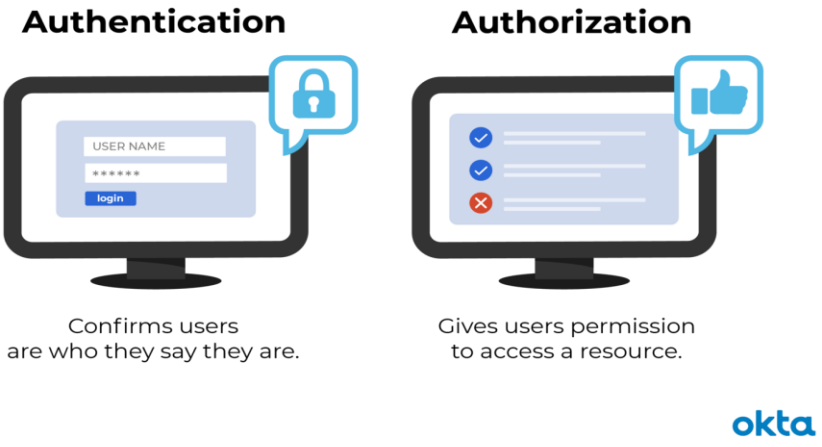

Рис. 2.1. Різниця між Аутентифікація і Авторизація

Наприклад, користувач хоче прочитати свої свіжі листи, що прийшли на email. Зайшовши на сайт пошти, він поки що бачить лише рекламу і новини. Але

свої листи він зможе прочитати лише після введення та відправки свого логіну та паролю. До цього моменту – ні-ні. Система ж не знає, хто він. Ось це і є процес аутентифікації.

Що може перевіряти система:

- чи існує користувач з таким ім’ям,
- чи збігається введений пароль з його обліковим записом.
- може запитувати ще наявність сертифіката, IP-адресу або додатковий код верифікації.

Якщо користувач пройшов аутентифікацію – він дійсно той, ким здається.

Після цього починається процес авторизації. І тут з поштою все просто: всі можуть переглядати листи і документи, редагувати їх і створювати нові.

А ось в соцмережах, сайтах або на форумах, часто відвідувачі належать до певної групи.

Тут авторизація допомагає системі визначити, що дозволено тим чи іншим користувачам:

- право писати повідомлення користувачу,
- право додавати в повідомлення певні матеріали.
- право переглядати фотографії інших користувачів.
- право на редагування або копіювання документів.

Авторизація перевіряє наявність прав на конкретні дії. Це відбувається не тільки під час входу в систему, але і при будь-якій спробі вчинити будь-які маніпуляції з даними.

Валідація вхідних даних - цей механізм дозволяє перевірити коректність та правильність даних, які надходять від клієнта. Недостатня валідаці кація вхідних даних може призвести до вразливостей у веб-додатку, таких як SQL-ін'єкції, XSS-атаки та інші. Для валідації вхідних даних можуть використовуватися різноманітні методи, такі як регулярні вирази, фільтри, обмеження та інші.

Захист від DoS атак є важливим аспектом розробки RESTful web-сервісів. Ось декілька стратегій та методів, які можна використовувати для захисту від DoS-атак в RESTful web-сервіси:

1. Обмеження швидкості запитів (Rate limiting): Встановлення обмежень на швидкість запитів від одного клієнта або IP-адреси допомагає запобігти перевантаженню сервера. Можна використовувати алгоритми, такі як Token Bucket або Leaky Bucket, для контролю швидкості запитів.

2. Перевірка на валідність запитів (Request validation): Валідація та фільтрація запитів допомагає виявити та відхилити недійсні або шкідливі запити. Це може включати перевірку формату даних, розміру запиту, а також використання списків дозволених методів та ресурсів.

3. Кешування (Caching): Використання кешування може допомогти зменшити навантаження на сервер шляхом збереження результатів запитів і повторного використання їх для майбутніх запитів. Це особливо ефективно для статичних або рідко змінюваних даних.

4. Доступне та масштабоване розподілене середовище (Scalable and distributed environment): Побудова масштабованих та розподілених систем дозволяє розподіляти навантаження між кількома серверами та протистояти DoS-атакам. Застосування навантажувального балансування (load balancing) та автоматичного масштабування (auto-scaling) допомагає забезпечити високу доступність та стійкість до навантаження.

5. Детекція та блокування аномальної активності: Використання систем детекції аномалій та аналізу журналів допомагає виявити та блокувати небезпечну або неординарну активність, яка може бути ознакою DoS-атаки. Це може включати спостереження за частотою запитів, незвичайними шаблонами поведінки або іншими аномаліями.

6. Гнучкість на рівні сервера (Server-level resilience): Використання технік, таких як встановлення максимального часу обробки запиту, обмеження ресурсів для окремих запитів та механізмів відновлення після збоїв допомагає забезпечити більшу стійкість до DoS-атак.

7. Моніторинг та аналіз: Постійний моніторинг навантаження, ресурсів сервера та журналів активності дозволяє вчасно виявляти незвичайну або надмірну активність, що може бути пов'язана з DoS-атаками. Аналіз даних допомагає

виявити тенденції та патерни атак для подальшого удосконалення захисту.

Захист від вразливостей - цей механізм дозволяє захистити RESTful web-сервіс від вразливостей, таких як XSS, SQL-ін'єкції, CSRF та інші. Для цього можуть використовуватися різноманітні техніки, такі як фільтрація вхідних даних, використання безпечних форматів даних, захист від міжсайтових скриптів та інші.

Моніторинг та журналювання - ці механізми дозволяють відстежувати та аналізувати поведінку RESTful web-сервісу та його користувачів для виявлення підозрілих дій, атак та інших проблем. Для цього можуть використовуватися різноманітні інструменти моніторингу та журналювання, такі як системи відстеження подій, системи аналізу ведення журналу та інші.

Інструменти для моніторингу та журналювання RESTful веб-сервісів включають:

- Prometheus: Це система моніторингу з відкритим вихідним кодом, яка збирає дані про стан системи та метрики з різних джерел, включаючи RESTful API. Вона дозволяє встановлювати правила спостереження та аналізувати дані для моніторингу продуктивності та стану веб-сервісу.
- Grafana: Це інструмент візуалізації даних з відкритим вихідним кодом, який працює у поєднанні з Prometheus або іншими системами моніторингу. За допомогою Grafana можна створювати настроювані інтерактивні панелі та графіки для відображення метрик, логів та інших даних, що стосуються RESTful API.
- ELK Stack (Elasticsearch, Logstash, Kibana): Це комплекс інструментів для журналювання та аналізу логів з відкритим вихідним кодом. Elasticsearch використовується для зберігання та індексації лог-даних, Logstash - для обробки та нормалізації лог-файлів, а Kibana - для візуалізації та аналізу журнальних даних. ELK Stack може бути налаштований для отримання та аналізу логів RESTful API.
- Splunk: Це комерційний інструмент для моніторингу, пошуку та аналізу даних з різних джерел, включаючи лог-файли RESTful веб-сервісів. Він надає можливості для збору, індексації, пошуку та візуалізації лог-даних, а також для створення звітів та сповіщень.

Ці інструменти допомагають відстежувати та аналізувати метрики

продуктивності, стану та логи RESTful веб-сервісів. Вибір конкретного інструменту залежить від ваших вимог.

Крім перерахованих механізмів захисту, також існують спеціалізовані інструменти для тестування вразливостей RESTful web-сервісів, які дозволяють виявляти потенційні проблеми та ризики в системі. Такі інструменти можуть бути корисними для виявлення вразливостей, які можуть бути недостатньо очевидними під час розробки та тестування системи.

Деякі інструменти для тестування вразливостей RESTful веб-сервісів включають:

- Burp Suite: Це один з найпопулярніших інструментів для тестування вразливостей веб-додатків. Він має спеціальні функції для тестування RESTful API, такі як перехоплення та модифікація HTTP-запитів, сканування на наявність вразливостей і перевірка автентифікації.
- OWASP ZAP (Zed Attack Proxy): Це безкоштовний інструмент з відкритим вихідним кодом, розроблений спільнотою OWASP. Він має функціонал для сканування веб-додатків на вразливості, включаючи RESTful API. За допомогою ZAP можна перехоплювати та модифікувати запити, аналізувати відповіді сервера та перевіряти вразливості.
- Nessus: Це комерційний інструмент для сканування вразливостей, який також може бути використаний для тестування RESTful API. Він має широкий набір функцій для виявлення різних типів вразливостей, включаючи наявність веб-сервісів і їх аналіз.
- Postman: Це розширена платформа для розробки та тестування API, включаючи RESTful сервіси. Він дозволяє створювати, відправляти та тестувати HTTP-запити, а також автоматизувати тестування за допомогою сценаріїв. Postman може бути використаний для перевірки правильності і безпеки RESTful API.

Ці інструменти надають широкий спектр можливостей для тестування вразливостей RESTful веб-сервісів. При використанні будь-якого з них важливо дотримуватись найкращих практик тестування безпеки, а також враховувати контекст вашого додатку та конкретні вимоги.

Окрім технічних засобів захисту, також важливо дотримуватися добрих практик розробки та застосування стандартів безпеки. Наприклад, дотримання принципів RESTful архітектури може допомогти зменшити ризики вразливостей системи.

Захист RESTful web-сервісів є важливим завданням для забезпечення безпеки веб-додатків. Використання різноманітних засобів та технік захисту, таких як автентифікація, авторизація, валідація даних, обмеження швидкості та захист від вразливостей, може допомогти зменшити ризики вразливостей та забезпечити безпеку RESTful web-сервісу.

Рис. 2.2. Найкращі методи безпеки WebAPI

Крім технічних заходів, також важливо враховувати соціальні аспекти захисту RESTful web-сервісів. Наприклад, важливо забезпечувати надійність паролів та протидіяти соціальному інжинірингу шляхом навчання користувачів виявляти та уникати шахрайства.

Також важливо регулярно оновлювати та вдосконалювати захист RESTful web-сервісів. Розробники повинні використовувати нові технології та засоби захисту, щоб забезпечити відповідність системи до поточних стандартів безпеки. API розроблено для автоматизованого доступу без взаємодії з користувачем, тому особливо важливо переконатися, що всі введені дані є дійсними та очікуваними. Будь-які запити, які не відповідають специфікації API, мають бути відхилені. Ось деякі загальні вказівки щодо перевірки вхідних даних REST API :

- Використовуйте вбудовані функції перевірки та кодування, де вони доступні.
- За замовчуванням усі параметри, об’єкти та інші вхідні дані вважаються ненадійними.
- Завжди перевіряйте розмір запиту, довжину та тип вмісту.
- Використовуйте жорстку типізацію для параметрів API (якщо підтримується).
- *Щоб запобігти SQL-ін’єкції , уникайте створювати запити вручну – замість цього використовуйте параметризовані запити.
- Значення параметрів і введення рядків у білий список, де це можливо (замість чорного списку).
- Реєструйте та відстежуйте всі помилки перевірки введення, щоб виявити спроби введення облікових даних.

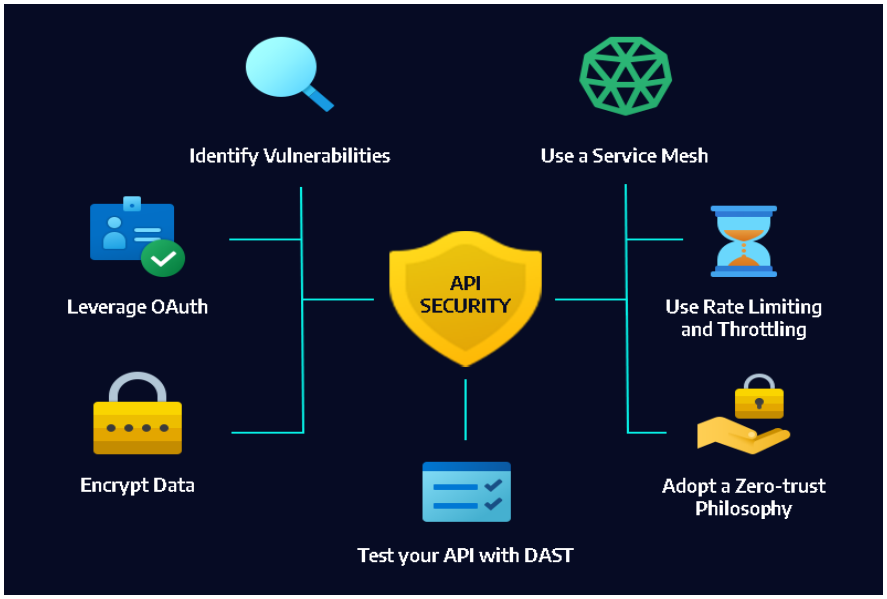

Рис. 2.3. Схема API безпеки

Узагалі, захист RESTful web-сервісів є складним та постійно розвиваючимся процесом, який вимагає знань та досвіду в галузі кібербезпеки. Проте дотримання найкращих практик та використання сучасних засобів захисту може допомогти забезпечити високий рівень безпеки та надійності системи.

2.2 Використання захисних механізмів проти XSS та CSRF атак

XSS та CSRF є двома видами вразливостей веб-додатків, які можуть бути використані зловмисниками для злому сайтів та крадіжки конфіденційної інформації. Щоб захистити ваш сайт від цих атак, можна використовувати різні захисні механізми.

Для захисту від XSS можна використовувати такі методи:

- Екранування вхідних даних: цей метод полягає в перетворенні спеціальних символів, які можуть бути використані в XSS атаках, в їх HTML-еквіваленти.
- Використання Content Security Policy (CSP): цей механізм дозволяє обмежити джерела, з яких можуть завантажуватись скрипти, стилі та інші ресурси. Це дозволяє зменшити можливість XSS атак, оскільки зловмисники не зможуть виконати свій код на вашому сайті.

Для захисту від CSRF можна використовувати такі методи:

- Використання токенів: при кожному запиті на сервер включається унікальний токен, який використовується для перевірки, що запит прийшов від легітимного користувача.
- Перевірка заголовків HTTP-запиту: сервер може перевірити заголовки запиту, щоб переконатися, що він прийшов від легітимного джерела.
- Використання робочого потоку (SameSite): цей механізм дозволяє обмежити доступ до куків (cookies) з інших доменів, що зменшує можливість CSRF атак.

Також, для підвищення захисту вашого веб-додатку від XSS та CSRF атак, можна виконувати наступні дії:

- Регулярно оновлювати версії використовуваних бібліотек та фреймворків. Це дозволяє виправляти вразливості та вдосконалювати захист.
- Не довіряти вхідні дані та перевіряти їх на наявність потенційно шкідливих символів. Також, не варто виводити вхідні дані користувачів без екранування.

- Використовувати HTTPS для захищеної передачі даних між клієнтом та сервером. Це дозволяє зменшити можливість перехоплення даних та підробки запитів.
- Регулярно аудитувати свій код на вразливості та вдосконалювати захист на основі результатів аудиту.

Нарешті, важливо розуміти, що захист від XSS та CSRF атак - це постійний процес, і він повинен бути включений у процес розробки веб-додатку з самого початку. Забезпечення безпеки повинно бути настільки ж важливим, як і функціональність вашого додатку.

Для захисту від CSRF атак можна використовувати токени безпеки, які генеруються сервером та передаються в запиті на зміну стану. Це дозволяє переконатись, що запит на зміну стану відправлений саме користувачем, а не зловмисником.

Важливо пам'ятати про те, що XSS та CSRF атаки можуть бути складними для виявлення та розуміння їх наслідків. Тому, якщо ви маєте підозру на наявність вразливості в вашому веб-додатку, краще звернутись до фахівців зі знанням цих атак та методів захисту від них.

Крім того, не забувайте про важливість навчання та підвищення своєї освіти в області кібербезпеки. Чим більше ви знаєте про можливі загрози та методи їх запобігання, тим більше шансів у вас захистити свій веб-додаток від атак.

Узагальнюючи, захист від XSS та CSRF атак - це складний та постійний процес, який потребує використання різноманітних методів та технологій. Включення захисту від цих атак в процес розробки веб-додатку та його постійне вдосконалення дозволять зберегти ваш додаток від можливих наслідків цих вразливостей.

Звернути увагу на додаткові заходи безпеки, які можна використовувати для підвищення рівня захисту від XSS та CSRF атак.

Наприклад, Content Security Policy (CSP) - це механізм безпеки, який дозволяє контролювати, звідки можуть завантажуватись ресурси на вашому веб-сайті. Ви можете вказати список джерел, з яких дозволяється завантажувати ресурси, таких

як скрипти, стилі, зображення тощо. Це допоможе запобігти XSS атакам, де зловмисник намагається вставити шкідливий код на вашому сайті.

Також, HTTP Only cookies - це додатковий захист для захисту від CSRF атак. Якщо HTTP Only cookie встановлено, то JavaScript не може отримати доступ до значення цього cookie. Це означає, що зловмисник не зможе отримати доступ до значення cookie, навіть якщо він зміг запустити шкідливий JavaScript-код на вашому сайті.

Забезпечувати постійний моніторинг вашого веб-додатку на наявність вразливостей. Для цього можна використовувати автоматичні інструменти сканування веб-додатків, які допоможуть виявити потенційні вразливості та запропонувати способи їх виправлення. Також, важливо регулярно оновлювати всі компоненти вашого веб-додатку, включаючи бібліотеки та фреймворки, для забезпечення максимальної безпеки.

Також, слід зазначити, що регулярне тестування на проникнення (penetration testing) може допомогти виявити потенційні вразливості в вашому веб-додатку та забезпечити його захист від XSS та CSRF атак. Penetration testing - це процес тестування системи з точки зору зловмисника з метою знайти потенційні вразливості та слабкі місця в безпеці, а також визначити, як зловмисники можуть використати ці вразливості для своїх користей.

Нарешті, якщо ваш веб-додаток побував під XSS або CSRF атакою, вам необхідно діяти швидко та відповідально. Спочатку необхідно відключити доступ до веб-додатку, щоб зупинити можливу подальшу експлуатацію вразливостей зловмисниками. Потім слід зібрати якомога більше інформації про атаку, включаючи джерело, спосіб і розмір завданої шкоди. Якщо необхідно, слід внести необхідні зміни в код веб-додатку та відновити доступ до нього.

Отже, захист від XSS та CSRF атак - це важливий елемент захисту вашого веб-додатку від шкідливих впливів. Необхідно використовувати захисні механізми, застосовувати безпечні методи програмування, контролювати вхідні дані та регулярно тестувати безпеку вашого веб-додатку. Це допоможе зберегти ваш веб-додаток від можливих загроз та забезпечити його успішну роботу.

Додатковою заходом захисту може бути використання Content Security Policy (CSP). Це механізм, який дозволяє зазначити список джерел, з яких дозволено завантажувати ресурси (такі як скрипти, стилі, зображення тощо). Він також дозволяє заборонити виконання коду з джерел, які не були вказані в списку дозволених джерел. Використання CSP може допомогти запобігти XSS-атакам, оскільки він обмежує виконання віддаленого скрипту, який може бути включений в HTML-код.

Крім того, для захисту від CSRF-атак можна використовувати техніку відправки токена (token-based technique). Це означає, що на сторінці форми вставляється токен, який згенерований на сервері та пов'язаний з конкретним сеансом користувача. При відправці запиту на сервер, включаючи запити на виконання дій з додатком, токен також відправляється. Сервер перевіряє, чи збігаються токени на сервері та в запиті, та дозволяє або відхиляє запит в залежності від результату перевірки. Це допомагає запобігти CSRF-атакам, оскільки зловмисники не зможуть відправити запит на сервер без правильного токена.

Нарешті, слід зазначити, що важливо не тільки застосовувати заходи захисту, але й регулярно перевіряти їх ефективність та підтримувати їх в актуальному стані. Наприклад, якщо ви використовуєте CSP, слід перевіряти, що список дозволених джерел ще дійсний та включає всі необхідні джерела. Також, слід регулярно оновлювати сервер та застосовувати патчі безпеки для мінімізації ризику атак.

Загалом, для захисту від XSS та CSRF атак важливо використовувати комбінацію захисних механізмів та перевіряти свій код на вразливості регулярно.

2.3 Аналіз можливостей SSL/TLS шифрування в RESTful веб-сервісів

SSL/TLS шифрування є одним з найпоширеніших та найефективніших засобів захисту веб-сервісів, в тому числі RESTful. Це забезпечує шифрування даних, які передаються між клієнтом та сервером, та запобігає їх зловживанню або перехопленню.

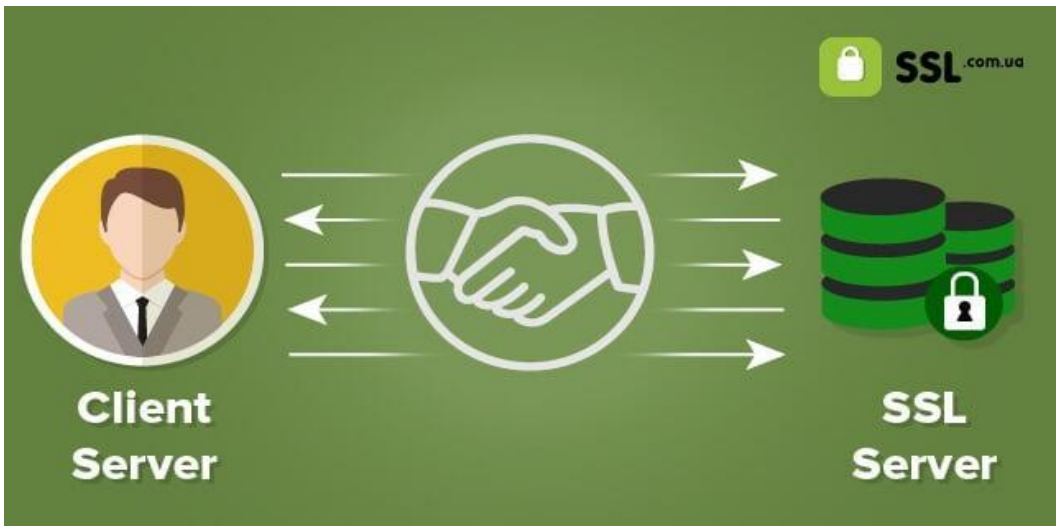

Рис. 2.4. Схема SSL шифрування

SSL/TLS використовує два ключі - приватний та публічний - для шифрування та розшифрування даних. Клієнт та сервер обмінюються публічним ключем та використовують його для шифрування та розшифрування даних, які передаються між ними.

Один з головних переваг SSL/TLS шифрування полягає у тому, що воно забезпечує автентифікацію сервера. Клієнт може перевірити, що він спілкується з правильним сервером, і що дані не будуть перехоплені або модифіковані зловмисниками. Це забезпечує підвищений рівень безпеки для RESTful веб-сервісу.

SSL/TLS шифрування також дозволяє забезпечити конфіденційність даних, які передаються між клієнтом та сервером. Це особливо важливо для веб-сервісів, які передають чутливу інформацію, таку як особисті дані користувачів, фінансові дані тощо.

SSL/TLS також може бути використано для захисту веб-сервісів від атак типу "человік-в-середині" (man-in-the-middle), коли зловмисник перехоплює та модифікує передані дані між клієнтом та сервером. За допомогою SSL/TLS шифрування, клієнт може перевірити, що він спілкується з правильним сервером, і що дані не будуть перехоплені або модифіковані зловмисниками.

Проте, SSL/TLS шифрування не є універсальним засобом захисту RESTful веб-сервісів, і існують деякі можливі проблеми, пов'язані з його використанням.

Одна з таких проблем полягає у тому, що SSL/TLS шифрування може зменшити швидкість взаємодії між клієнтом та сервером. Це може бути особливо важливим для веб-сервісів, які мають велику кількість запитів, оскільки кожен запит повинен бути шифрований та розшифрований.

Іншою можливою проблемою є те, що SSL/TLS шифрування може бути піддається атакам, таким як атака типу "часова петля" (time-division cryptanalysis), коли зловмисник збирає статистику про час, необхідний для шифрування та розшифрування даних. Ця проблема може бути вирішена шляхом використання сильніших шифрувальних алгоритмів, таких як AES.

Також важливо зазначити, що SSL/TLS шифрування не захищає веб-сервіси від інших можливих атак, таких як атаки на SQL-ін'єкції або кросс-сайт скриптінг. Для захисту від цих атак можуть бути використані додаткові заходи безпеки, такі як перевірка вхідних даних та належна конфігурація сервера.

Узагальнюючи, SSL/TLS шифрування є потужним засобом захисту RESTful веб-сервісів, який забезпечує шифрування даних, автентифікацію сервера та конфіденційність даних. Проте, важливо пам'ятати, що воно не є універсальним засобом захисту та не захищає веб-сервіс від всіх можливих атак. Для забезпечення максимальної безпеки веб-сервісу можуть бути використані інші засоби захисту, такі як перевірка вхідних даних, механізми автентифікації та авторизації, контроль доступу та інші.

Крім того, SSL/TLS шифрування має певні недоліки, такі як зниження швидкості взаємодії та можливість атак типу "часова петля". Для розв'язання цих проблем можуть бути використані сильніший шифрувальний алгоритм, оптимізація сервера та належна конфігурація системи безпеки.

Також важливо зазначити, що для захисту веб-сервісу можуть бути використані інші протоколи безпеки, такі як OAuth та OpenID Connect, які забезпечують безпеку авторизації та аутентифікації, а також підтримують різні рівні авторизації.

Узагальнюючи, SSL/TLS шифрування є ефективним засобом захисту RESTful веб-сервісів, який забезпечує конфіденційність даних та автентифікацію

сервера. Проте, він не є універсальним засобом захисту та не захищає веб-сервіс від всіх можливих атак. Для максимальної безпеки можуть бути використані інші засоби захисту, такі як перевірка вхідних даних, механізми авторизації та контролю доступу.

Розглянемо їх переваги та недоліки:

SSL плюси:

1. Широке застосування: SSL був широко використовуваний протоколом для захищеного з'єднання веб-сайтів, електронної пошти, віртуальних приватних мереж (VPN) тощо.

2. Відносна простота: SSL протокол відносно простий у налаштуванні та використанні.

3. Підтримка багатьох версій: SSL підтримує кілька версій протоколу, що дозволяє сумісність з різними клієнтськими програмами та серверами.

Мінуси:

1. Вразливості: SSL страждає від деяких серйозних вразливостей, таких як POODLE, BEAST та інші, які можуть бути використані для атак на безпеку.

2. Застарілість: Офіційна підтримка SSL була припинена, і багато організацій переходять на використання TLS.

TLS плюси:

1. Покращена безпека: TLS має вдосконалені механізми шифрування та аутентифікації, що робить його більш стійким до атак і забезпечує вищий рівень безпеки.

2. Підтримка сучасних шифрувань: TLS підтримує більш сильні алгоритми шифрування та хешування, що дозволяє забезпечити кращу захист інформації.

Мінуси:

1. Сумісність: Іноді можуть виникати проблеми з сумісністю між різними версіями TLS, особливо при спілкуванні зі старими системами, які не підтримують більш нові версії TLS.

2. Складніша настройка: TLS може вимагати більшої настройки та

конфігурації, особливо при використанні додаткових функцій, таких як взаємна аутентифікація або розширені механізми шифрування. Це може бути складніше для непрофесійних користувачів.

3. Швидкодія: У порівнянні зі старішими версіями SSL, TLS може мати трохи більший накладний обсяг, що може призводити до деякого зниження швидкодії з'єднання.

4. Сумісність зі старими системами: Деякі старі клієнтські програми або сервери можуть не підтримувати більш нові версії TLS, що може створювати проблеми з їх взаємодією.

Враховуючи ці переваги та недоліки, TLS став більш популярним та рекомендованим протоколом для захищеного з'єднання, оскільки він пропонує покращену безпеку та підтримує сучасні криптографічні алгоритми. SSL залишився в минулому і рекомендується його не використовувати через виявлені вразливості та відсутність активної підтримки.

2.4 Аналіз аутентифікації перевірити ідентичність користувача за медотодів HTTP Basic або Digest аутентифікація, OAuth, OpenID

Розглянемо три методи аутентифікації - HTTP Basic/Digest аутентифікація, OAuth та OpenID.

HTTP Basic/Digest аутентифікація - це простий метод аутентифікації, який використовується в браузерах та веб-серверах. У цьому методі користувач надсилає свої ідентифікаційні дані (логін та пароль) у вигляді заголовка HTTP запиту до сервера. У разі HTTP Basic аутентифікації, дані передаються у вигляді незашифрованого тексту, що є небезпечним з точки зору безпеки. У разі Digest аутентифікації, дані передаються у зашифрованому вигляді, але цей метод також має свої недоліки та може бути підірваний атаками типу "middle man".

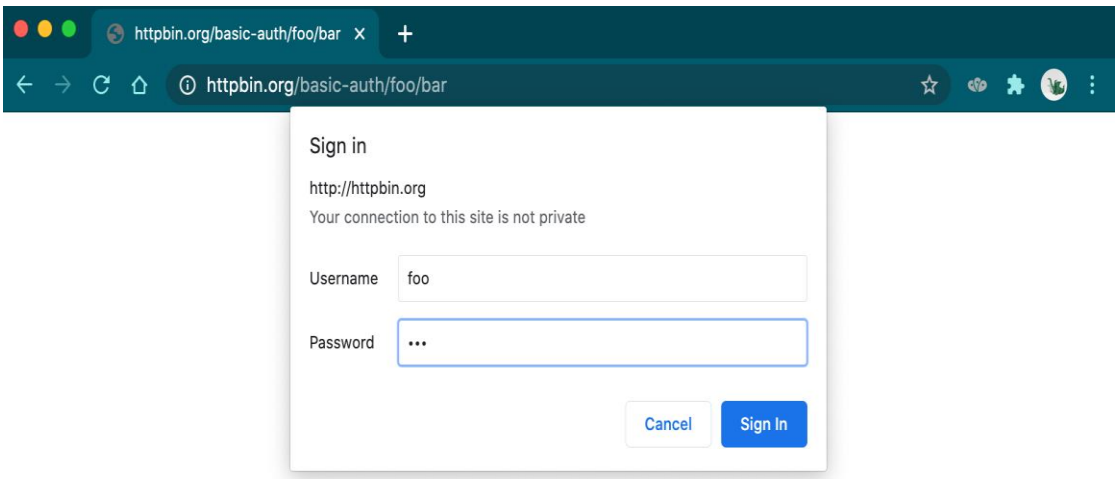

Рис. 2.5. Аутентифікація

OAuth - це відкритий протокол, який використовується для дозволу користувачам надавати обмежений доступ до своїх ресурсів без необхідності передавати свій пароль. Цей протокол забезпечує безпеку та конфіденційність інформації, оскільки передача даних відбувається за допомогою токенів доступу, які можуть бути обмінені між користувачем та додатком. OAuth дозволяє використовувати внутрішні авторизаційні системи, такі як Google, Facebook, Twitter та так далі.

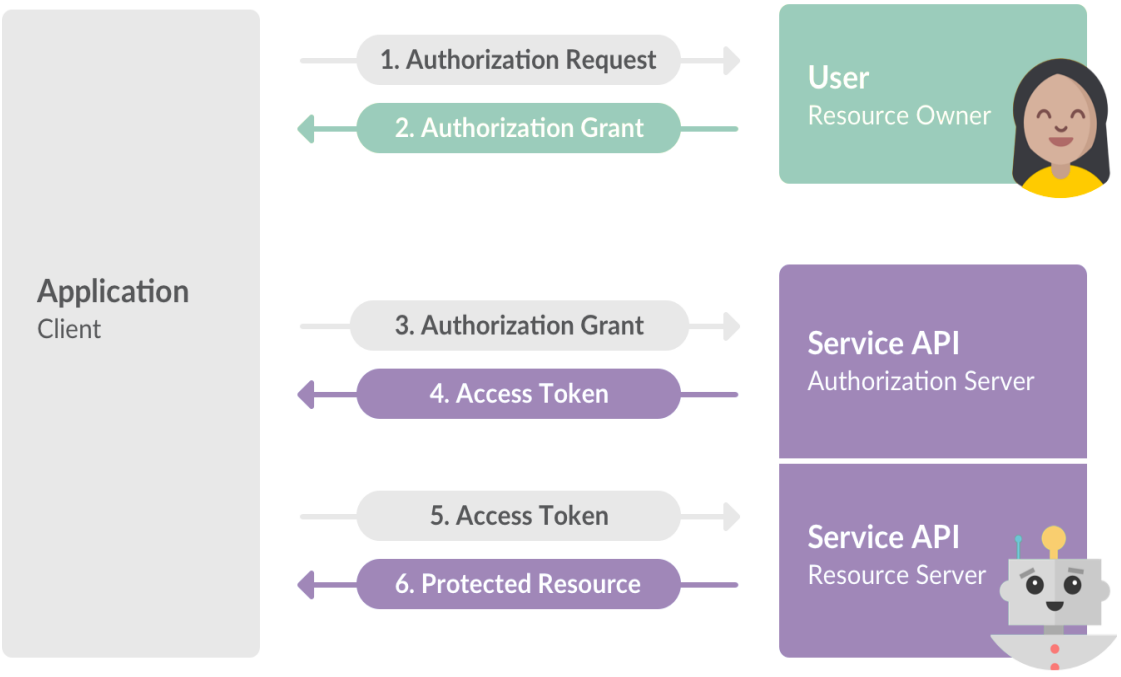

Рис. 2.6. Робочий процес OAuth

OpenID - це ідентифікатор користувача, який використовується для аутентифікації в різних онлайн-сервісах. Цей метод дозволяє користувачам використовувати один OpenID для входу в різні онлайн-сервіси без необхідності створювати окремі облікові записи для кожного з OpenID дозволяє зменшити ризик зламу облікового запису, оскільки користувач не має потреби створювати та запам'ятовувати багато паролів для різних сервісів. Якщо користувач вже має OpenID, він може використовувати його для входу в будь-який сервіс, який підтримує OpenID.

Отже, для аутентифікації можна використовувати різні методи залежно від потреб користувачів та характеру веб-сайту чи додатку. HTTP Basic/Digest аутентифікація можуть бути використані у простих веб-сайтах, але для більш складних систем краще використовувати OAuth або OpenID, які забезпечують більш високий рівень безпеки та конфіденційності.

Наприклад, OAuth може бути використаний у соціальних мережах або інших додатках, які потребують доступ до особистих даних користувача, таких як контакти, фото чи музика. OpenID може бути використаний у веб-сайтах, які потребують відвідувачам реєстрації чи входу, наприклад, інтернет-магазинах, соціальних мережах, форумах чи блогах.

Важливо також звернути увагу на те, що необхідно забезпечувати безпеку та захист даних користувачів під час аутентифікації. Для цього можуть бути використані додаткові методи, такі як захист від SQL-ін'єкцій, захист від Cross-Site Scripting (XSS), використання SSL-з'єднань та інші.

Отже, вибір методу аутентифікації залежить від потреб користувачів та характеру веб-сайту чи додатку. Під час реалізації аутентифікації варто дотримуватись найкращих практик безпеки, щоб захистити довіру користувачів та їхні особисті дані.

Крім того, варто забезпечити відповідний рівень безпеки при зберіганні та передачі інформації про користувачів. Для цього можна використовувати шифрування, захист від SQL-ін'єкцій, захист від Cross-Site Scripting (XSS), використання SSL-з'єднань та інші методи захисту.

Також варто наголосити на тому, що додаткові фактори аутентифікації, такі як двофакторна аутентифікація, можуть значно підвищити рівень безпеки. При двофакторній аутентифікації користувачеві необхідно вказати не тільки логін та пароль, але й додатковий фактор, такий як одноразовий пароль, SMS-повідомлення або відбиток пальця. Це значно ускладнює заволодіння доступом до аккаунта навіть у випадку, якщо зловмисник вже знає логін та пароль користувача.

2.5 Аналіз захист від атаки Buffer Overflow в RESTful web-сервісах

Захист від атаки Buffer Overflow в RESTful web-сервісах вимагає впровадження наступних заходів безпеки:

1. Валідація та обмеження розміру даних: Перевіряйте та валідуйте вхідні дані, щоб забезпечити, що вони відповідають очікуваному формату і обмеженням. Обмежуйте розмір буферів та вхідних полів, щоб уникнути переповнення.

2. Використання безпечних функцій та бібліотек: Використовуйте безпечні функції та бібліотеки, які мають вбудовані механізми захисту від переповнення буфера, такі як функції для безпечного копіювання рядків або обробки вхідних даних.

3. Використання механізмів пам'яті з обмеженнями: Використовуйте механізми пам'яті з обмеженнями, які дозволяють виділяти та використовувати пам'ять з урахуванням максимального розміру буферів і даних.

4. Перевірка на вразливості: Виконуйте перевірку на вразливості вашого програмного забезпечення, включаючи сканування вразливостей і проведення кодового аудиту, щоб виявити можливі проблеми з Buffer Overflow.

5. Проведення тестування на проникнення: Виконуйте тестування на проникнення, щоб виявити можливі вразливості в системі і перевірити їх на Buffer Overflow. Це дозволить вам виявити і виправити проблеми до їх експлуатації зловмисниками.

6. Постійне оновлення програмного забезпечення: Забезпечуйте регулярне оновлення всього використовуваного програмного забезпечення,

включаючи фреймворки, бібліотеки та компоненти, щоб уникнути використання відомих вразливостих, включаючи ті, що стосуються Buffer Overflow.

7. Використання принципів безпечного програмування: Дотримуйтеся принципів безпечного програмування, таких як перевірка меж та валідація вхідних даних, уникання використання небезпечних функцій, правильне керування пам'яттю і уникання використання старих або застарілих методів.

8. Впровадження механізмів виявлення та запобігання атак: Використовуйте системи виявлення та запобігання вторгнень (IDS/IPS), які можуть виявити небезпечні активності, пов'язані з Buffer Overflow, і прийняти відповідні заходи для запобігання атак.

9. Обмеження прав доступу: Забезпечуйте обмеження прав доступу до важливих ресурсів та функцій. Це допоможе уникнути можливості використання успішної атаки Buffer Overflow для отримання несанкціонованого доступу до конфіденційної інформації або привілейованих операцій.

10. Навчання та свідомість персоналу: Проводьте навчання та свідомість персоналу щодо безпеки програмного забезпечення та усвідомлення ризиків, пов'язаних з Buffer Overflow. Це дозволить забезпечити відповідальне використання коду та застосування найкращих практик безпеки.

2.6 Аналіз політики Same-origin policy в RESTful веб-сервісів

Політика Same-origin policy є важливою політикою безпеки для RESTful веб-сервісів. Вона встановлює обмеження на взаємодію між веб-додатками, що запобігає несанкціонованому доступу до ресурсів з інших джерел. Основні аспекти політики Same-Origin включають наступне:

1. Політика Same-Origin: За замовчуванням браузери використовують політику Same-Origin для обмеження взаємодії між веб-додатками на основі походження. Це означає, що веб-додатки можуть здійснювати запити до ресурсів тільки з однакового початкового походження (схема, домен і порт).

2. Cross-Origin Resource Sharing (CORS): Якщо веб-додаток потребує

здійснити запит до ресурсу з іншого початкового походження, то використовується механізм Cross-Origin Resource Sharing (CORS). Цей механізм включає наявність спеціальних заголовків у запитах і відповідях, які дозволяють серверу з іншого походження передати ресурс коректному веб-додатку.

3. Застосування CORS у RESTful веб-сервісах: При розробці RESTful веб-сервісів важливо належним чином налаштувати CORS, якщо дозволяється взаємодія з іншими джерелами. Це включає встановлення відповідних заголовків у веб-сервісі, що дозволяють або обмежують запити з інших початкових походжень.

4. Захист від Cross-Site Request Forgery (CSRF): Для запобігання атакам типу CSRF в RESTful веб-сервісах рекомендується використовувати механізми захисту, такі як включення токенів аутентифікації у запити або перевірка джерела запиту.

Загалом, політика Same-Origin в RESTful веб-сервісах допомагає зменшити ризик несанкціонованого доступу до ресурсів і захищає веб-додатки від атак, таких як Cross-Site Scripting (XSS) і Cross-Site Request Forgery (CSRF). Вона обмежує взаємодію між веб-додатками, забезпечуючи, що запити можуть бути здійснені лише з однакового початкового походження.

При розробці RESTful веб-сервісів важливо правильно налаштувати політику Same-Origin та CORS, враховуючи конкретні вимоги і обмеження системи. Це може включати встановлення заголовків CORS у веб-сервері для контролю доступу, визначення списку дозволених джерел або використання токенів авторизації для перевірки дозволу на запити з інших джерел.

Налаштування політики Same-Origin і використання CORS допомагають забезпечити безпеку і надійність RESTful веб-сервісів, знижуючи ризик вразливостей і атак. Однак, важливо також враховувати інші аспекти безпеки, такі як аутентифікація, авторизація, захист від вразливостей введення даних та моніторинг системи для виявлення підозрілої активності.

Таким чином, політика Same-Origin для RESTful веб-сервісів впливає на взаємодію клієнтських додатків (наприклад, JavaScript додатків), які виконують запити до цих сервісів з браузера. Вона обмежує можливість здійснювати запити з

одного походження до іншого.

Перевірки походження застосовуються браузером у кожному випадку потенційної взаємодії між елементами з різних джерел. Це включає, але не обмежується:

- Код JavaScript і об’єктна модель документа (DOM), наприклад, сторінка не може отримати доступ до вмісту свого iframe, якщо вони не мають того самого походження.
- Файли cookie, наприклад ваш сеансовий файл cookie для певного сайту, не можуть бути надіслані на сторінку іншого походження. Однак у випадку файлів cookie схема та порт не оцінюються, а лише домен/субдомен.
- Виклики AJAX (XmlHTTPRequest).

Однак SOP не усуває повністю взаємодію між різними джерелами. Браузер оцінює, чи може ця взаємодія становити загрозу, і якщо ні, вона допускається. Наприклад:

- Зазвичай ви можете писати між джерелами. Наприклад, ви можете створювати перехресні посилання та надсилати форми з іншого джерела.
- Зазвичай можна вбудовувати між джерелами. Наприклад, ви можете використовувати вміст з іншого джерела в iframe (якщо це дозволяє X-Frame-Options) або вставити img , css або сценарій з іншого сайту.
- Однак читання між джерелами зазвичай блокується. Це часто означає, що ви можете надіслати перехресний запит, але не можете прочитати відповідь.

У деяких випадках ви можете послабити жорстку хватку SOP і дозволити певну перехресну взаємодію, наприклад, між різними доменами, які обидва належать вам. У таких випадках є кілька способів переконатися, що SOP не перешкоджатиме можливостям взаємодії веб-додатків між доменами.

Сама по собі політика однакового походження підвищує безпеку, але її недостатньо для запобігання всім атакам міжсайтової підробки запитів (CSRF) , які в основному є спробою скористатися різними джерелами. Тому анти-CSRF-токени все ж слід використовувати як додаткову форму захисту. SOP також абсолютно марний як метод захисту від міжсайтового сценарію (XSS), оскільки він мав би

обмежувати завантаження сценаріїв з різних сайтів, і це повністю перешкоджало б функціональності веб-додатків.

Таким чином, хоча SOP є ефективним засобом захисту від очевидного, його не можна вважати надійним механізмом захисту. Однак це механізм, який веб-розробники повинні розуміти та використовувати

2.7 Аналіз ABAC та RBAC і роль безпеки

RBAC (Role-Based Access Control) - це метод, в якому доступ до ресурсів контролюється на основі ролей, які присвоюються користувачам. Кожна роль має визначений набір прав доступу до різних ресурсів, а користувачам присвоюються ролі в залежності від їхніх обов'язків та функцій в організації.

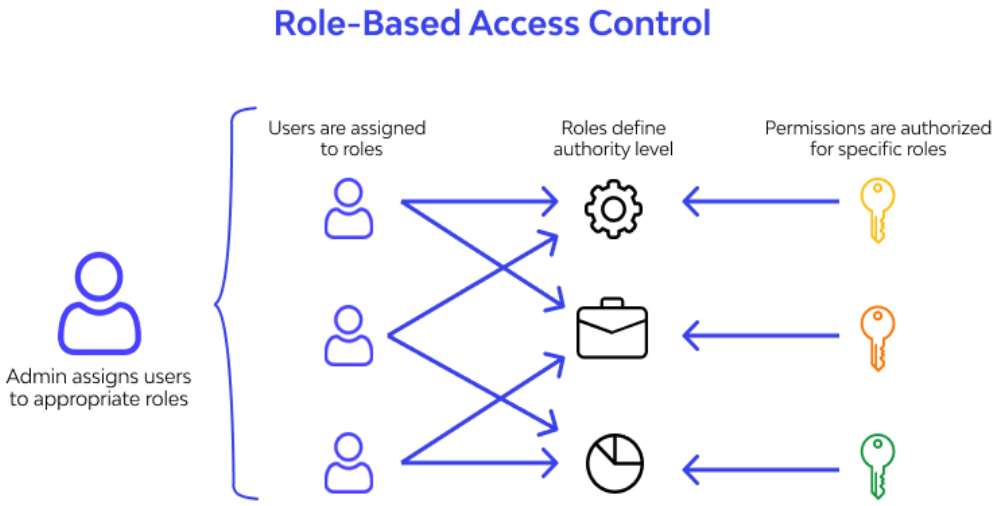

Рис. 2.7. Схема RBAC

ABAC (Attribute-Based Access Control) - це метод, в якому доступ до ресурсів контролюється на основі атрибутів користувача та контексту запиту. Наприклад, атрибутами можуть бути роль, час, місцезнаходження, тип пристрою, атрибути запиту тощо. На основі цих атрибутів система приймає рішення про надання або відмову у доступі до ресурсів.

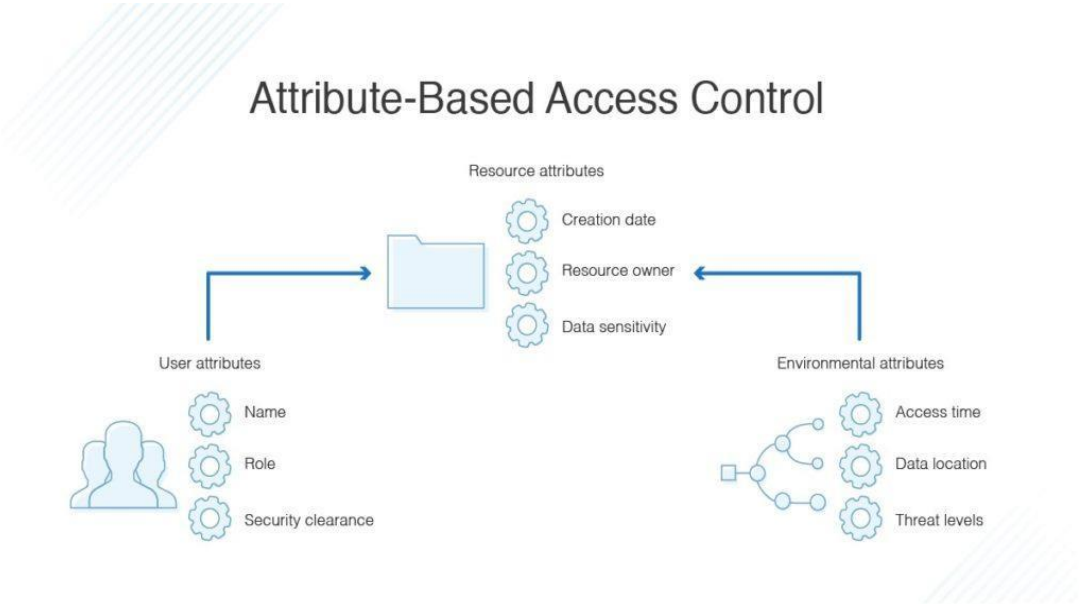

Рис. 2.8. Схема ABAC

RBAC та ABAC - це два різні підходи до керування доступом до ресурсів.

У RBAC, доступ до ресурсів контролюється на основі ролі, яку має користувач. В системі, що використовує RBAC, ролі надаються користувачам в залежності від їх потреб та відповідальності. Ролі визначають, які ресурси можуть бути доступні для користувача, і які дії він може здійснювати з цими ресурсами.

ABAC, з іншого боку, використовує атрибути користувача та контексту, щоб приймати рішення щодо доступу до ресурсів. У системі з ABAC, кожен запит на доступ до ресурсів аналізується на основі набору атрибутів, таких як роль користувача, географічне положення, час доступу та інші фактори. Застосування ABAC дозволяє більш гнучкий та точний контроль доступу до ресурсів, оскільки атрибути користувача можуть бути дуже різними.

Загалом, який з цих методів краще використовувати для безпеки RESTful web-сервісів залежить від конкретних потреб та вимог вашого проекту. Однак, обидва методи можуть бути успішно використані для забезпечення безпеки RESTful web-сервісів.

У RBAC, доступ до ресурсів контролюється на основі ролі, яку має користувач. В системі, що використовує RBAC, ролі надаються користувачам в залежності від їх потреб та відповідальності. Ролі визначають, які ресурси можуть

бути доступні для користувача, і які дії він може здійснювати з цими ресурсами.

ABAC, з іншого боку, використовує атрибути користувача та контексту, щоб приймати рішення щодо доступу до ресурсів. У системі з ABAC, кожен запит на доступ до ресурсів аналізується на основі набору атрибутів, таких як роль користувача, географічне положення, час доступу та інші фактори. Застосування ABAC дозволяє більш гнучкий та точний контроль доступу до ресурсів, оскільки атрибути користувача можуть бути дуже різними.

Загалом, який з цих методів краще використовувати для безпеки RESTful web-сервісів залежить від конкретних потреб та вимог вашого проекту. Однак, обидва методи можуть бути успішно використані для забезпечення безпеки RESTful web-сервісів.

Узагалі, забезпечення безпеки RESTful web-сервісів - це складний та багатоаспектний процес, який потребує розуміння різних механізмів безпеки та їх взаємодії. Застосування RBAC та ABAC, разом із застосуванням SSL / TLS та інших заходів безпеки, може допомогти забезпечити безпеку RESTful web-сервісів та захистити їх від різних видів загроз.

RESTful web-сервісів. Розглянемо плюси та мінуси кожного підходу:

Плюси ABAC:

- Більш гнучкий підхід до контролю доступу, оскільки використовується багато атрибутів для визначення прав доступу.
- Дозволяє забезпечити точний контроль доступу до конкретних ресурсів, оскільки можна встановлювати правила доступу для конкретних атрибутів.
- Забезпечує більшу гранулярність контролю доступу, оскільки можна визначати права доступу на основі багатьох атрибутів.

Мінуси ABAC:

- Складність налаштування та управління правилами доступу, оскільки використовуються багато атрибутів для визначення прав доступу.
- Може призвести до зниження продуктивності системи, оскільки використовується багато атрибутів для визначення прав доступу.

Плюси RBAC:

- Простий у налаштуванні та управлінні системою контролю доступу, оскільки базується на ролях.
- Дозволяє забезпечити широкі права доступу для ролей, що допомагає спростити управління доступом до ресурсів.
- Дозволяє забезпечити більшу швидкодію системи, оскільки базується на простішому механізмі контролю доступу.

Мінуси RBAC:

- Обмежена гранулярність контролю доступу, оскільки базується на ролях, що може призвести до необхідності використання додаткових механізмів контролю доступу.
- Можливість зламування системи через компрометацію ролі або перехоплення ідентифікатора ролі.

Отже, кожен підхід до контролю доступу має свої переваги та недоліки, і обраний підхід залежить від конкретної ситуації та потреб користувачів. ABAC може бути кращим вибором у випадку, коли необхідно забезпечити більшу гранулярність контролю доступу та точний контроль над ресурсами, в той час як RBAC може бути кращим вибором у випадку, коли необхідно швидко та просто управляти доступом до ресурсів.

Може бути корисним використовувати комбінацію різних підходів до контролю доступу для забезпечення максимальної безпеки. Наприклад, можна використовувати RBAC для надання базових прав доступу на рівні ролей, а потім використовувати ABAC для забезпечення більшої гранулярності на основі додаткових атрибутів, таких як місцезнаходження, час, тип пристрою та інших факторів.

В будь-якому випадку, важливо враховувати потреби користувачів та вимоги безпеки при виборі підходу до контролю доступу. Також важливо регулярно перевіряти та оновлювати правила доступу для забезпечення максимальної безпеки системи.

3 РЕКОМЕНДАЦІЇ ЩОДО ЗАХИСТУ WEB-СЕРВІСІВ

3.1 Функції та можливості обраного рішення

Ось порівняна чому вибрати REST над іншими аналогами: SOAP, JSON-RPC, XML-RPC.

Призначення:

REST API - це архітектурний стиль, що використовується для побудови мережевих додатків на основі протоколу HTTP. Він використовує HTTP-методи, такі як GET, POST, PUT та DELETE, для доступу до ресурсів, що використовуються в додатку.

SOAP - це протокол обміну повідомленнями, який використовується для обміну даними між різними додатками, що працюють на різних платформах та мовах програмування.

JSON-RPC та XML-RPC - це протоколи віддаленого виклику процедур (RPC), які використовуються для виклику методів на віддаленому сервері.

Концепт підходу:

REST API - базується на принципах архітектури REST, що передбачає використання уніфікованого інтерфейсу та безстанних запитів.

SOAP - передбачає використання XML-повідомлень та має складну структуру, яка може включати в себе інформацію про безпеку, транзакції та інші аспекти.

JSON-RPC та XML-RPC - передбачають виклик віддалених процедур за допомогою мережевого протоколу та формату повідомлень.

Залежність від стану:

REST API - безстанний, тобто кожен запит містить всю необхідну інформацію для обробки.

SOAP - може бути залежним від стану, оскільки повідомлення можуть містити інформацію про стан процесу.

JSON-RPC та XML-RPC - можуть бути залежними від стану, оскільки передача даних відбувається під час виклику методу на віддаленому сервері.

Кешування:

REST API - може використовувати кешування, яке дозволяє зменшити навантаження на сервер та прискорити роботу додатка.

SOAP - має можливість використовувати кешування за допомогою HTTP-заголовків.

JSON-RPC та XML-RPC - не мають вбудованої підтримки кешування.

Формат повідомлень:

REST API - використовує різні формати даних, такі як JSON, XML, або HTML.

SOAP - використовує XML-формат повідомлень

JSON-RPC та XML-RPC - не мають вбудованої підтримки безпеки.

Продуктивність:

REST API - може бути швидким та продуктивним, оскільки використовує безстанні запити та може використовувати кешування.

SOAP - має складну структуру та може бути повільним в порівнянні з іншими протоколами.

JSON-RPC та XML-RPC - можуть бути швидкими та продуктивними за умови оптимальної конфігурації.

Безпека:

REST API - може використовувати різні методи захисту, такі як автентифікацію та авторизацію.

SOAP - має вбудовану підтримку безпеки, таку як шифрування та підписування повідомлень.

JSON-RPC та XML-RPC - не мають вбудованої підтримки безпеки.

Протокол передачі:

REST API - використовує протокол HTTP.

SOAP - може використовувати різні протоколи, такі як HTTP, SMTP, або JMS.

JSON-RPC та XML-RPC - можуть використовувати різні протоколи, такі як HTTP, HTTPS, або TCP/IP.

Рекомендовано для:

REST API - рекомендується для створення мережевих додатків з простим та інтуїтивно зрозумілим інтерфейсом.

SOAP - рекомендується для обміну повідомленнями між різними додатками, що працюють на різних платформах та мовах програмування, та вимога безпеки та надійності взаємодії.

JSON-RPC та XML-RPC - рекомендується для взаємодії між додатками, що працюють на одній платформі та мові програмування, та потребують швидкого та простого інтерфейсу.

Переваги:

REST API - має простий та інтуїтивно зрозумілий інтерфейс, що дозволяє швидко розробляти та впроваджувати додатки. Також, використання безстанних запитів та можливості кешування дозволяє забезпечити високу продуктивність та масштабованість

SOAP - має вбудовану підтримку безпеки та надійності, що дозволяє забезпечити високу якість взаємодії між додатками, які працюють на різних платформах та мовах програмування.

JSON-RPC та XML-RPC - мають простий та швидкий інтерфейс, що дозволяє швидко та ефективно взаємодіяти між додатками на одній платформі та мові програмування.

Таблиця 3.1

Таблиця оцінки порівняння програмного забезпечення

Критерій	**SOAP**	**REST**	**JSON**	**XML**
Призначення	4	3	2	2
Концепт	3	4	2	1
Підхід	3	4	4	4
Залежність від стану	1	4	3	4
Кешування	2	4	2	4

Формат повідомлень	1	4	1	1
Безпека	3	4	2	2
Проду тивність	4	2	3	1
Протокол передачі	4	2	3	2
Рекомендовано для	4	2	3	1
Переваги	3	2	1	4

Виходячи з результатів об’єктом мого дослідження та удосконалення обираємо REST API так, як ця технологія являється більш гнучкою, та простою до удосконалення.

3.2 Рекомендації щодо безпеки restful web-сервісів

Використовуйте HTTPS: HTTPS забезпечує шифрування даних, що передаються між клієнтом та сервером, тим самим забезпечуючи конфіденційність і цілісність даних.

Обмежте доступ до критичних ресурсів: Для забезпечення безпеки RESTful веб-сервісу обмежте доступ до критичних ресурсів, таких як бази даних та конфігураційні файли. Використовуйте права доступу та автентифікацію, щоб гарантувати, що лише авторизовані користувачі мають доступ до цих ресурсів.

Валідуйте вхідні дані: Валідація даних забезпечує, що вхідні дані, що передаються до веб-сервісу, відповідають певним вимогам, що допомагає запобігти атакам внесення неправильних даних (наприклад, SQL-ін’єкції).

Використовуйте механізми обмеження спроб атак: Механізми обмеження спроб атак дозволяють обмежувати кількість запитів до веб-сервісу, що можуть бути виконані з однієї IP-адреси. Це зменшує ризик DDoS-атак та спроби перебору пароля. Забезпечуйте моніторинг безпеки: Моніторинг безпеки дозволяє виявляти

потенційні загрози та дії, що вказують на атаки на ваш веб-сервіс.

Перевіряйте наявність вразливостей: Перевірка веб-сервісу на наявність вразливостей дозволяє виявляти можливі дії з боку зловмисників та зменшує ризики для безпеки даних.

Перевіряйте використання Cookies: Cookies можуть зберігати чутливу інформацію, таку як ідентифікатори сеансів, тому важливо забезпечити безпеку використання cookies шляхом встановлення атрибутів cookie, які забезпечують безпеку, такі як HttpOnly та Secure. HttpOnly дозволяє обмежити доступ до cookies з боку скриптів JavaScript, тоді як Secure забезпечує, що cookies передаються лише через HTTPS.

Проводьте тестування на проникнення: Тестування на проникнення є ефективним способом виявлення потенційних слабких місць у веб-сервісі. Воно дозволяє виявити та виправити вразливості перед тим, як вони будуть використані зловмисниками.

Застосовуйте принцип найменшого доступу: Принцип найменшого доступу вимагає надавати доступ до ресурсів лише тим користувачам, які потребують цього для виконання своїх завдань. Це зменшує ризик несанкціонованого доступу та можливість виконання шкідливого коду.

Захистіть конфігураційні файли: Конфігураційні файли містять конфіденційну інформацію, таку як ключі API та паролі. Захистіть конфігураційні файли, щоб запобігти несанкціонованому доступу до цієї інформації. Використовуйте захист паролів та шифрування даних, щоб забезпечити їх безпеку.

Регулярно оновлюйте програмне забезпечення: Регулярне оновлення програмного забезпечення забезпечує захист від відомих вразливостей та забезпечує стійкість до атак. Перевіряйте регулярність оновлення програмного забезпечення та виконуйте оновлення якомога швидше.

Перевіряйте безпеку сторонніх бібліотек: Сторонні бібліотеки можуть містити вразливості та бути джерелом шкідливого коду. Перевіряйте безпеку сторонніх бібліотек перед їх використанням та перевіряйте оновлення для забезпечення відповідності найновішим стандартам безпеки.

Використовуйте двофакторну аутентифікацію: Двофакторна аутентифікація забезпечує додатковий рівень безпеки, вимагаючи від користувача введення додаткового коду або використання іншого пристрою для підтвердження своєї особи. Використовуйте двофакторну аутентифікацію для підвищення безпеки веб-сервісу.

Навчайте своїх користувачів: Навчайте своїх користувачів про безпеку веб-сервісу та навчіть їх використовувати найкращі практики безпеки. Це допоможе забезпечити максимальний рівень безпеки вашого веб-сервісу та запобігти можливим атакам.

ВИСНОВКИ

1) Завдання дослідження шляхів та розробка рекомендацій щодо забезпечення безпеки RESTful web-сервісів є важливим для забезпечення безпеки даних, які передаються між різними системами.

2) У результаті дослідження було виявлено, що основними загрозами для безпеки RESTful web-сервісів є аутентифікація та авторизація, вразливості з боку клієнтів, відсутність контролю доступу до ресурсів та збереження конфіденційної інформації.

3) Для забезпечення безпеки RESTful web-сервісів рекомендується використовувати протокол HTTPS, щоб забезпечити шифрування даних, а також використовувати методи аутентифікації та авторизації, такі як JWT або OAuth 2.0. Крім того, слід використовувати заходи безпеки на рівні коду, такі як перевірка на відсутність вразливостей та обмеження прав доступу до ресурсів.

4) Таким чином, забезпечення безпеки RESTful web-сервісів є важливим завданням для забезпечення безпеки даних. Рекомендації, що наведені в дослідженні, можуть допомогти розробникам створювати безпечні сервіси та захистити дані від несанкціонованого доступу.

ПЕРЕЛІК ПОСИЛАНЬ

1. REST [Електронний ресурс]. – Режим доступу до ресурсу:https://uk.wikipedia.org/wiki/REST

2. Аутентифікація і авторизація: що це і в чому відмінність [Електронний ресурс]. – Режим доступу до ресурсу: https://qagroup.com.ua/publications/autentyfikatciia-i-avtoryzatciia/

3. Securing IoT Devices and SecurelyConnecting the Dots Using REST API andMiddleware [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/8777334.

4. Same-origin policy [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

5. What Is SSL, TLS, and HTTPS? [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://blog.gigamon.com/2019/09/06/gigamons-guide-to-communications-security-what-is-ssl-tls-and-https/

6. Що таке DDoS-атака? [Електронний ресурс]. – Режим доступу до ресурсу: https://www.microsoft.com/uk-ua/security/business/security-101/what-is-a-ddos-attack

7. API Security - Threats & Best Practices [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.atatus.com/blog/api-security-threats-best-practices/

8. API Security: The Complete Guide to Threats, Methods & Tools [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://brightsec.com/blog/api-security/

9. How to ensure REST API security [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.invicti.com/blog/web-security/rest-api-web-service-security/

10. How to secure your REST API from attackers by Ivan Novikov

[Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://hakin9.org/how-to-secure-your-rest-api-from-attackers/

11. Безвугляк М. С. ЗАХИСТ ІНФОРМАЦІЇ В МЕРЕЖІ ІНТЕРНЕТУ РЕЧЕЙ НА ОСНОВІ REST API / М. С. Безвугляк, В. В. Курдеча // 64 ПЕРСПЕКТИВИ ТЕЛЕКОМУНІКАЦІЙ / М. С. Безвугляк, В. В. Курдеча. – Київ, 2020. – С. 213–215.

12. Безвугляк М. С. ОСОБЛИВОСТІ ТЕХНОЛОГІЇ REST API В IoT / Максим Сергійович Безвугляк // ПЕРСПЕКТИВИ РОЗВИТКУ ІНФОРМАЦІЙНО-ТЕЛЕКОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ ТА СИСТЕМ / Максим Сергійович Безвугляк. – Київ, 2020. – С. 362.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)