Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра безпеки інформації та телекомунікацій

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

студента *Грінченка Антона Сергійовича*

академічної групи *125-18-1*

спеціальності *125 Кібербезпека*

спеціалізації[1]

за освітньо-професійною програмою *Кібербезпека*

на тему *Підвищення рівня безпеки під час обміну JWT в ASP.NET Core*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи				
розділів:				
спеціальний	ст.викл. Саксонов Г.М.			
економічний	к.е.н., доц. Романюк Н.М.			

Рецензент				

Нормоконтролер				

Дніпро
2022

ЗАТВЕРДЖЕНО:
завідувач кафедри
безпеки інформації та телекомунікацій
_____ д.т.н., проф. Корнієнко В.І.

«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавра

студенту *Грінченку Антону Сергійовичу* **академічної групи** *125-18-1*
(прізвище ім'я по-батькові) (шифр)

спеціальності *125 Кібербезпека*

за освітньо-професійною програмою *Кібербезпека*

на тему *Підвищення рівня безпеки під час обміну JWT в ASP.NET Core*

затверджену наказом ректора НТУ «Дніпровська політехніка» від 18.05.2022 № 268-с

Розділ	Зміст	Термін виконання
Розділ 1	Аналіз кіберзагроз, а саме усіх актуальних на даний момент вразливостей у комп'ютерних системах, також розгляд існуючих рішень.	04.02.2022 – 25.02.2022
Розділ 2	Проектування і програмна реалізація системи підвищення безпеки робочих процесів з обмеженням доступу до критичних ресурсів.	22.03.2022 – 08.05.2022
Розділ 3	Розрахунок витрат на розробку програмного забезпечення та визначення економічної ефективності данного рішення.	13.05.2022 – 10.06.2022

Завдання видано _____ Саксонов Г.М.
(підпис керівника) (прізвище, ініціали)

Дата видачі: _____

Дата подання до екзаменаційної комісії: _____

Прийнято до виконання _____ Грінченко А.С.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 60с., 8 рис., 7 табл., 54 джерела.

Об’єктом дослідження є поняття безпеки процесів під час передачі JWT між сервером та web-сторінкою клієнта.

Предметом дослідження є методи і засоби підвищення рівня безпеки процесів передачі та зберігання JWT.

Мета роботи полягає в створенні власної програмної реалізації системи підвищення безпеки процесів обміну JWT між сервером та клієнтом, а, також, підвищення безпеки зберігання JWT.

Перший розділ представляє собою огляд основних понять предметної області.

У другому розділі проведено проектування і розробку системи підвищення безпеки процесів обміну та зберігання JWT с використанням Cookies.

Третій розділ присвячено економічному аналізу доцільності розробки системи підвищення безпеки процесів передачі за обробки JWT.

Підвищення рівня безпеки під час обміну JWT в ASP.NET Core.

ABSTRACT

Explanatory note: 60 pages, 8 figures, 7 tables, 54 sources.

The object of research is the concept of process security during the transfer of JWT between the server and the client's web page.

The subject of the research is methods and means to increase the level of security of JWT transfer and storage processes.

The purpose of the work is to create its own software implementation of the system to increase the security of JWT exchange processes between the server and the client, as well as to increase the security of JWT storage.

The first section is an overview of the basic concepts of the subject area.

In the second section, the design and development of a system to increase the security of JWT exchange and storage processes using Cookies.

The third section is devoted to the economic analysis of the feasibility of developing a system to improve the security of transmission processes for JWT processing.

Improve security when sharing JWT in ASP.NET Core.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ОС — Операційна система

NIST — National Institute of Standards and Technology

SPA — Single Page Application

JWT —Json Web Token

API — Application Programming Interface

SSE — Server Side Event

ЗМІСТ

ВСТУП

В сучасному світі, при тенденції на розвиток науково-технічного прогресу і глобальну діджиталізацію, питання безпечності доступу до інформації стає все гостріше.

Кожного дня з'являються нові незаконні засоби отримати конфіденційну інформацію та перехвати ключів авторизації та аутентифікації.

У випадку Web-серверної авторизації найвразливішим місцем системи є передача та зберігання токнів доступу до інформаційної системи.

Виходячи з усього вищесказаного, можна зробити висновок про високу актуальність поняття підвищення безпеки процесів передачі та зберігання токенів у мережі.

Об'єктом дослідження є поняття безпеки процесів передачі та обробки JWT у середовищі ASP.NET.

Предметом дослідження є методи і засоби безпеки процесів передачі токенів у Web-додатку.

Мета роботи полягає в створенні власної програмної реалізації системи підвищення безпеки процесів передачі, обробки та зберігання JWT у середовищі ASP.NET.

РОЗДІЛ 1
ОСНОВНІ АСПЕКТИ ТЕОРЕТИЧНОЇ БАЗИ

1.1 Json Web Token

Веб-токен JSON — це запропонований Інтернет-стандарт для створення даних з необов'язковими підписами та/або додатковим шифруванням, корисне навантаження якого включає JSON, який оголошує низку претензій. Токени підписуються приватним секретом або відкритим/ приватним ключем.

Наприклад, сервер може згенерувати токен з оператором «увійти як адміністратор» і надати його клієнту. Потім клієнт може використовувати цей маркер, щоб підтвердити, що він увійшов як адміністратор. Маркер може бути підписаний приватним ключем однієї сторони (зазвичай сервера), щоб будь-яка сторона згодом могла перевірити, що маркер є актуальним. Легітимність токена також можна перевірити, якщо контрагент має відповідний відкритий ключ якимось відповідним і надійним способом. Закладки розроблені так, щоб вони були компактними, безпечними для URL і придатними для використання, особливо в контексті єдиного входу у веб-переглядачах (SSO). Претензії JWT зазвичай можуть використовуватися для передачі автентифікованих облікових даних користувача або будь-якого іншого типу претензій, що вимагається бізнес-процесом, між постачальником облікових даних і постачальником послуг.

JWT покладається на інші стандарти на основі JSON: веб-підпис JSON і веб-шифрування JSON.

Header

Визначає, який алгоритм використовується для створення підпису

HS256 вказує, що цей маркер підписаний за допомогою HMAC-SHA256.

Типовими криптографічними алгоритмами, які використовуються, є HMAC з SHA-256 (HS256) і підпис RSA з SHA-256 (RS256). JWA (веб-алгоритми JSON) RFC 7518 пропонує багато іншого як для аутентифікації, так і для шифрування.

```
{
 "alg": "HS256",
 "typ": "JWT"
}
```

Payload

Містить набір претензій. Специфікація JWT визначає сім зареєстрованих імен претензій, які є стандартними полями, які зазвичай входять до маркерів. Залежно від призначення маркера, зазвичай також включаються спеціальні вимоги.

У цьому прикладі є стандартна претензія Issued At Time (iat) і спеціальна претензія (loggedInAs).

```
{
 "loggedInAs": "admin",
 "iat": 1422779638
}
```

Signature

Надійно перевіряє маркер. Підпис обчислюється шляхом кодування заголовка та корисного навантаження за допомогою Base64url Encoding RFC 4648 та об'єднання двох разом із роздільником крапки. Потім цей рядок запускається через криптографічний алгоритм, зазначений у заголовку, в даному випадку HMAC-SHA256. Кодування Base64url подібне до base64, але використовує інші небуквенно-цифрові символи та пропускає заповнення.

```
HMAC_SHA256(
 secret,
 base64urlEncoding(header) + '.' +
 base64urlEncoding(payload)
```

)

Три частини кодуються окремо за допомогою Base64url Encoding RFC 4648 і об'єднуються за допомогою крапок для створення JWT:

```
const token = base64urlEncoding(header) + '.' + base64urlEncoding(payload) + '.' + base64urlEncoding(signature)
```

Наведені вище дані та секрет «секретного ключа» створює токен:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJsb2dnZWRJbkFzIjoiYWRtaW4iLCJpYXQiOjE0MjI3Nzk2Mzh9.gzSraSYS8EXBxLN_oWnFSRgCzcmJmMjLiuyu5CSpyHI

Отриманий маркер можна легко передати в HTML і HTTP.

Використання

Під час аутентифікації, коли користувач успішно входить в систему, використовуючи свої облікові дані, буде повернуто веб-токен JSON, який потрібно зберегти локально (зазвичай у локальному або сеансовому сховищі, але також можна використовувати файли cookie), замість традиційного підходу створення сеансу на сервері та повертає файл cookie. Для неконтрольованих процесів клієнт також може аутентифікуватися безпосередньо, створюючи та підписуючи власний JWT із попередньо спільним секретом і передавати його службі, сумісній з OAuth, таким чином:

```
POST /oauth2/token?
Content-type: application/x-www-form-urlencoded
grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&assertion=eyJhb...
```

Якщо клієнт передає дійсне затвердження JWT, сервер згенерує access_token, дійсний для здійснення викликів до програми, і передасть його назад клієнту:

```
{
 "access_token": "eyJhb...",
 "token_type": "Bearer",
 "expires_in": 3600
}
```

Коли клієнт хоче отримати доступ до захищеного маршруту або ресурсу, користувальницький агент повинен надіслати JWT, як правило, в заголовку HTTP авторизації, використовуючи схему Bearer. Вміст заголовка може виглядати так:

Authorization: Bearer eyJhbGci...<snip>...yu5CSpyHI

Це механізм аутентифікації без стану, оскільки стан користувача ніколи не зберігається в пам'яті сервера. Захищені маршрути сервера перевірять дійсний JWT в заголовку авторизації, і якщо він присутній, користувачеві буде дозволено отримати доступ до захищених ресурсів. Оскільки JWT є автономними, вся необхідна інформація є там, що зменшує необхідність багаторазового запиту до бази даних.

Уразливі місця

Веб-токени JSON можуть містити стан сеансу. Але якщо вимоги проекту допускають скасування сеансу до закінчення терміну дії JWT, служби більше не можуть довіряти твердженням маркерів лише за допомогою маркера. Щоб підтвердити, що сеанс, збережений у маркері, не скасовано, підтвердження маркерів має бути перевірено у сховищі даних. Це робить токени більше не без громадянства, підриваючи головну перевагу JWT.

Консультант з безпеки Тім Маклін повідомив про вразливості в деяких бібліотеках JWT, які використовували поле alg для неправильної перевірки маркерів, найчастіше, приймаючи маркер alg=none. Хоча ці вразливості були виправлені, Маклін запропонував повністю відмовитися від підтримки поля alg, щоб запобігти подібній плутанини в реалізації. Тим не менш, нові вразливості alg=none все ще зустрічаються в дикій природі, причому чотири CVE, подані в період 2018-2021 років, мають цю причину.

При правильному дизайні розробники можуть усунути вразливості алгоритму, вживаючи запобіжних заходів:

- Ніколи не дозволяйте лише заголовку JWT здійснювати перевірку
- Знайте алгоритми (уникайте залежності лише від поля alg)

- Використовуйте ключ відповідного розміру

1.2 SPA

Односторінковий додаток (SPA) — це веб-додаток або веб-сайт, який взаємодіє з користувачем шляхом динамічного відображення поточної веб-сторінки з новими даними з веб-сервера, замість стандартного методу веб-браузера, який завантажує цілі нові сторінки. Метою є швидші переходи, завдяки чому веб-сайт буде більше схожий на рідну програму.

У SPA оновлення сторінки ніколи не відбувається; натомість весь необхідний код HTML, JavaScript та CSS або витягується браузером із завантаженням однієї сторінки, або відповідні ресурси динамічно завантажуються та додаються на сторінку за потреби, як правило, у відповідь на дії користувача.

Історія

Походження терміну односторінковий додаток неясна, хоча ця концепція обговорювалася принаймні ще в 2003 році. Стюарт Морріс, студент програмування в Університеті Кардіффа, Уельс, написав веб-сайт Self-Contained на slashdotslash.com з тими ж цілями та функціями у квітні 2002 року, а пізніше того ж року Лукас Бірдо, Кевін Хекман, Майкл Пічі та Кліффорд. Yeh описав реалізацію односторінкової програми в патенті США 8,136,109.

JavaScript можна використовувати у веб-браузері для відображення інтерфейсу користувача (UI), запуску логіки програми та зв'язку з веб-сервером. Доступні самостійні безкоштовні бібліотеки, які підтримують створення SPA, зменшуючи кількість коду JavaScript, який розробники повинні писати.

Технічні підходи

Існують різні методи, які дозволяють браузеру зберігати одну сторінку, навіть якщо програмі потрібен зв'язок із сервером.

Хеш документів

Автори HTML можуть використовувати ідентифікатори елементів, щоб показати або приховати різні розділи HTML-документа. Потім,

використовуючи CSS, автори можуть використовувати селектор `#target`, щоб показати лише той розділ сторінки, до якого перейшов браузер.

Аякс

Станом на 2006 рік найпомітнішою технікою, яка використовувалася, був Аякс. Ajax передбачає використання асинхронних запитів до сервера для даних XML або JSON, наприклад за допомогою JavaScript XMLHttpRequest або більш сучасного fetch() (з 2017 року), або застарілого об'єкта ActiveX. На відміну від декларативного підходу більшості фреймворків SPA, з Ajax веб-сайт безпосередньо використовує JavaScript або бібліотеку JavaScript, таку як jQuery, для маніпулювання DOM та редагування елементів HTML. Крім того, Ajax був популяризований такими бібліотеками, як jQuery, який забезпечує простіший синтаксис і нормалізує поведінку Ajax в різних браузерах, які історично мали різну поведінку.

WebSockets

WebSockets – це двонаправлена технологія зв'язку клієнт-сервер у реальному часі, яка є частиною специфікації HTML5. Для зв'язку в реальному часі їх використання перевершує Ajax з точки зору продуктивності і простоти.

Події, надіслані сервером

Події, надіслані сервером (Server sive events) – це техніка, за допомогою якої сервери можуть ініціювати передачу даних клієнтам браузера. Після встановлення початкового з'єднання потік подій залишається відкритим, поки клієнт не закриє його. SSE надсилаються через традиційний HTTP і мають різноманітні функції, яких WebSockets не мають за проектом, наприклад автоматичне повторне підключення, ідентифікатори подій і можливість надсилати довільні події.

Плагіни для браузера

Незважаючи на те, що цей метод застарів, асинхронні виклики до сервера також можуть бути здійснені за допомогою технологій плагінів браузера, таких як Silverlight, Flash або Java-аплети.

Транспортування даних (XML, JSON і Ajax)

Запити до сервера зазвичай призводять до повернення необроблених даних (наприклад, XML або JSON) або нового HTML. У випадку, коли сервер повертає HTML, JavaScript на клієнті оновлює часткову область DOM (об'єктна модель документа). Коли повертаються вихідні дані, часто для перекладу необроблених даних у HTML використовується процес XML / (XSL) на стороні клієнта (а у випадку JSON — шаблон), який потім використовується для оновлення часткової області DOM. .

Пошукова оптимізація

Через відсутність виконання JavaScript на сканерах деяких популярних веб-пошукових систем SEO (пошукова оптимізація) історично представляла проблему для загальнодоступних веб-сайтів, які бажають прийняти модель SPA.

У період з 2009 по 2015 рік Google Webmaster Central запропонував, а потім рекомендував «схему сканування AJAX» з використанням початкового знака оклику в ідентифікаторах фрагментів для сторінок AJAX із збереженням стану (#!). Сайт SPA має впровадити особливу поведінку, щоб забезпечити можливість вилучення релевантних метаданих пошуковою системою. Для пошукових систем, які не підтримують цю схему хешування URL-адрес, хешовані URL-адреси SPA залишаються невидимими. Багато авторів, включаючи Дженні Теннісон з W3C, вважали ці URI проблемними, оскільки вони роблять сторінки недоступними для тих, у кого JavaScript не активований у своєму браузері. Вони також порушують заголовки посилання HTTP, оскільки браузерам заборонено надсилати ідентифікатор фрагмента в заголовку Referer. У 2015 році Google відмовився від їхньої популярної пропозиції сканування AJAX.

Крім того, програми можуть відображати завантаження першої сторінки на сервері та наступні оновлення сторінки на клієнті. Це традиційно складно, оскільки код візуалізації може знадобитися написати іншою мовою або фреймворком на сервері та в клієнті. Використання шаблонів без логіки,

перехресна компіляція з однієї мови на іншу або використання однієї мови на сервері та клієнті може допомогти збільшити кількість коду, який можна спільно використовувати.

У 2018 році Google представив динамічний рендеринг як ще один варіант для сайтів, які бажають запропонувати сканерам важку версію сторінки без JavaScript для цілей індексування. Динамічне відтворення перемикається між версією сторінки, яка відтворюється на стороні клієнта, і попередньо відтвореною версією для певних користувацьких агентів. Цей підхід передбачає, що ваш веб-сервер виявляє сканери (через агент користувача) і направляє їх на засіб візуалізації, з якого вони потім обслуговують просту версію вмісту HTML.

Оскільки сумісність із SEO не є тривіальною в SPA, варто зазначити, що SPA зазвичай не використовуються в контексті, де індексація пошукових систем є або вимогою, або бажаною. Випадки використання включають програми, які показують приватні дані, приховані за системою аутентифікації. У тих випадках, коли ці програми є споживчими продуктами, часто для цільової сторінки програми та маркетингового сайту використовується класична модель «перемалювання сторінки», яка надає достатньо метаданих для того, щоб програма відображалася як звернення в пошуковому запиті. Блоги, форуми підтримки та інші традиційні артефакти перемальовування сторінок часто розміщуються навколо SPA, які можуть заповнювати пошукові системи відповідними термінами.

Станом на 2021 рік і Google зокрема, SEO-сумісність для звичайного SPA є простою і вимагає лише кількох простих умов для виконання. Також доступний практичний посібник для більш просунутого SPA, який використовує вибіркове попереднє відтворення.

Один із способів збільшити кількість коду, який можна спільно використовувати між серверами та клієнтами, — це використовувати мову шаблонів без логіки, як-от Mustache або Handlebars. Такі шаблони можна відтворювати з різних мов хоста, наприклад, Ruby на сервері та JavaScript на

клієнті. Однак просто спільний доступ до шаблонів зазвичай вимагає дублювання бізнес-логіки, яка використовується для вибору правильних шаблонів і заповнення їх даними. Відтворення з шаблонів може мати негативний вплив на продуктивність, коли оновлюється лише невелика частина сторінки — наприклад, значення введеного тексту у великому шаблоні. Заміна всього шаблону може також порушити вибір користувача або положення курсора, де оновлення лише зміненого значення може не бути. Щоб уникнути цих проблем, програми можуть використовувати прив'язки даних інтерфейсу користувача або детальні маніпуляції з DOM, щоб оновлювати лише відповідні частини сторінки, а не повторно відображати цілі шаблони. рейтинги.

1.3 Обмін JWT в ASP.NET

Для встановлення аутентифікації за допомогою токенів у методі ConfigureServices у виклик services.AddAuthentication передається значення JwtBearerDefaults.AuthenticationScheme. Далі за допомогою методу AddJwtBearer() додається конфігурація токена.

Для конфігурації токена застосовується об'єкт JwtBearerOptions, який дає змогу за допомогою властивостей налаштувати роботу з токеном. В даному випадку використані такі властивості:

- RequireHttpsMetadata: якщо дорівнює false, то SSL при надсиланні токена не використовується. Однак даний варіант встановлений тільки для тестування. У реальному додатку все ж таки краще використовувати передачу даних по протоколу https.
- TokenValidationParameters: Параметри валідації токена - складний об'єкт, що визначає, як токен буде валідуватися. Цей об'єкт у свою чергу має багато властивостей, які дозволяють налаштувати різні аспекти валідації токена. Але найважливіші властивості: IssuerSigningKey - ключ безпеки, яким підписується токен, і ValidateIssuerSigningKey - чи треба валідувати ключ безпеки. Ну і крім того, можна встановити ряд інших властивостей, таких як потрібно валідувати видавця та споживача токена, термін життя токена, можна встановити назву claims для ролей та логінів користувача тощо.

Після проведенний дій можливо використовувати авторизацію на основі токенів. За замовчуванням у ASP.NET Core відсутні вбудовані можливості створення токена. І в даному випадку можна або скористатися сторонніми рішеннями (наприклад, IdentityServer або OpenIdDict), або створити свій механізм.

1.4 Основні проблеми існуючого підходу

Ідентифікація по JWT (JSON Web Token) – це досить одноманітний, узгоджений механізм авторизації та автентифікації між сервером та клієнтами. Переваги JWT у тому, що він дозволяє нам менше керувати станом та добре масштабується. Не дивно, що авторизація та автентифікація за його допомогою все частіше використовується у сучасних веб-додатках.

При розробці програм з JWT часто виникає питання: де і як рекомендується зберігати токен? Якщо ми розробляємо веб-додаток, у нас є два найпоширеніші варіанти:

HTML5 Web Storage (localStorage or sessionStorage)

Cookies

Порівнюючи ці способи, можна сказати, що вони обидва зберігають значення в браузер клієнта, обидва досить прості у використанні і є звичайним сховище пар ключ-значення. Різниця полягає у середовищі зберігання.

Web Storage (localStorage/sessionStorage) доступний через JavaScript у тому ж домені. Це означає, що будь-який JavaScript код у вашому додатку має доступ до Web Storage, і це породжує вразливість до cross-site scripting (XSS) атак. Як механізм зберігання Web Storage не надає жодних способів убезпечити свої дані під час зберігання та обміну. Ми можемо його використовувати лише для допоміжних даних, які хочемо зберегти під час оновлення (F5) або закриття вкладки: стан та номер сторінки, фільтри тощо.

1.5 Постановка задачі

Основна здача даної роботи полягає в тому, щоб підвищити рівень безпеки обміну JWT для запобіганню перехопленю та використанню токенів злоумисниками, для цього необхідно виконати ряд задач, а саме:

- розробити базовый додаток (сервер та сторінку користувача);
- налаштувати базу данних;
- інтегрувати систему підпису JWT;
- сконфугірувати сервер для підтримки опрацювання Cookies;
- розробити систему аутинтифікації за допомогою Cookies;
- налаштувати безпечну обробку та зберігання файлів Cookies.

РОЗДІЛ 2
ПРОЕКТУВАННЯ І РЕАЛІЗАЦІЯ ВЛАНСОГО ПІДХОДУ

2.1 Модель покращення підходу до передачі JWT в ASP.NET

Токени можуть передаватись через файли cookie браузера. Файли cookie, які використовуються з прапором httpOnly, не піддаються XSS. httpOnly – це прапорець для доступу до читання, запису та видалення cookies тільки на сервері. Вони не будуть доступні через JavaScript на клієнті, тому клієнт не знатиме про токен, а авторизація повністю оброблятиметься на стороні сервера.

Ми також можемо встановити secure прапор, який гарантуватиме, що cookie передається тільки через HTTPS. Зважаючи на ці переваги, мій вибір упав на cookies.

Файли cookie HTTP (також звані веб-куки, файли cookie Інтернету, файли cookie браузера або просто файли cookie) — це невеликі блоки даних, створені веб-сервером, коли користувач переглядає веб-сайт, і розміщуються на комп'ютері користувача чи іншому пристрої веб-браузером користувача. Файли cookie розміщуються на пристрої, який використовується для доступу до веб-сайту, і протягом сеансу на пристрої користувача може бути розміщено більше одного файлу cookie.

Файли cookie виконують корисні, а іноді й важливі функції в Інтернеті. Вони дозволяють веб-серверам зберігати інформацію про стан (наприклад, товари, додані в кошик для покупок в інтернет-магазині) на пристрої користувача або відстежувати його активність у веб-перегляді (зокрема натискання певних кнопок, вхід в систему або запис відвіданих сторінок у минуле). Їх також можна використовувати для збереження для подальшого використання інформації, яку користувач раніше ввів у поля форми, наприклад, імена, адреси, паролі та номери платіжних карток.

Файли cookie для аутентифікації зазвичай використовуються веб-серверами для автентифікації того, що користувач увійшов у систему та за допомогою якого облікового запису він увійшов. Без файлу cookie користувачам потрібно було б аутентифікувати себе, увійшовши в систему на кожній сторінці, що містить конфіденційну інформацію, до якої вони хочуть отримати доступ . Безпека аутентифікаційного файлу cookie загалом залежить від безпеки веб-сайту, який видає, і веб-браузера користувача, а також від того, чи зашифровані дані cookie. Уразливості безпеки можуть дозволяти зловмиснику зчитувати дані файлів cookie, використовувати їх для отримання доступу до даних користувача або використовувати для отримання доступу (з обліковими даними користувача) до веб-сайту, якому належить файл cookie.

Файли cookie для відстеження, і особливо сторонні файли cookie, зазвичай використовуються як способи складання довгострокових записів історії перегляду окремих осіб — потенційна проблема конфіденційності, яка спонукала законодавців Європи та США вжити заходів у 2011 році. Європейське законодавство вимагає, щоб усі веб-сайти, націлені на країни-члени Європейського Союзу, отримували «інформовану згоду» від користувачів перед тим, як зберігати несуттєві файли cookie на своєму пристрої.

Сеансовий файл cookie

Сеансовий файл cookie (також відомий як файл cookie в пам'яті, тимчасовий файл cookie або непостійний файл cookie) існує лише у тимчасовій пам'яті, поки користувач переміщується по веб-сайту. Файли cookie сеансу закінчуються або видаляються, коли користувач закриває веб-браузер. Файли cookie сеансу ідентифікуються браузером за відсутністю призначеного їм терміну дії.

Постійний файл cookie

Термін дії постійного файлу cookie закінчується в певну дату або через певний проміжок часу. Протягом тривалості життя постійного файлу cookie,

встановленого його розробником, його інформація буде передаватися на сервер щоразу, коли користувач відвідує веб-сайт, якому він належить, або щоразу, коли користувач переглядає ресурс, що належить цьому веб-сайту, з іншого веб-сайту (наприклад, рекламу).).

З цієї причини постійні файли cookie іноді називають файлами cookie для відстеження, оскільки рекламодавці можуть використовувати їх для запису інформації про звички перегляду веб-сторінок користувача протягом тривалого періоду часу. Однак вони також використовуються з «законних» причин (наприклад, щоб користувачі залишалися ввійшли в свої облікові записи на веб-сайтах, щоб уникнути повторного введення облікових даних під час кожного відвідування).

Безпечний файл cookie

Захищені файли cookie можна передавати лише через зашифроване з'єднання (тобто HTTPS). Вони не можуть передаватися через незашифровані з'єднання (наприклад, HTTP). Це зменшує ймовірність крадіжки файлів cookie через підслуховування. Файл cookie стає безпечним, якщо до нього додається позначка Secure.

Файл cookie лише для HTTP

Клієнтські API, такі як JavaScript, не можуть отримати доступ до файлів cookie лише для http. Це обмеження усуває загрозу викрадення файлів cookie за допомогою міжсайтових сценаріїв (XSS). Однак файл cookie залишається вразливим до атак міжсайтового відстеження (XST) та підробки міжсайтових запитів (CSRF). Файл cookie отримує цю характеристику шляхом додавання до файлу cookie прапора HttpOnly.

Файл cookie на тому самому сайті

У 2016 році Google Chrome версії 51 представив новий тип файлів cookie з атрибутом SameSite. Атрибут SameSite може мати значення Strict, Lax або None. З атрибутом SameSite=Strict браузери надсилатимуть файли cookie лише до цільового домену, який є таким же, як і вихідний домен. Це ефективно пом'якшить атаки підробки міжсайтових запитів (CSRF). З

SameSite=Lax браузери надсилатимуть файли cookie із запитами до цільового домену, навіть якщо він відрізняється від вихідного домену, але лише для безпечних запитів, таких як GET (POST є небезпечним), а не сторонніх файлів cookie (всередині iframe). Атрибут SameSite=Немає дозволить сторонні (міжсайтові) файли cookie, однак більшість браузерів вимагає атрибута Secure для SameSite=Немає файлів cookie.

Файл cookie на тому самому сайті включено в новий проект RFC для «Cookies: Механізм управління станом HTTP» для оновлення RFC 6265 (якщо затверджено).

Chrome, Firefox, Microsoft Edge почали підтримувати файли cookie одного сайту. Ключом розгортання є обробка наявних файлів cookie без визначеного атрибута SameSite. Chrome обробляє ці файли cookie так, ніби SameSite=None, це дозволить забезпечити роботу всіх веб-сайтів/програм, як раніше. У лютому 2020 року Google збирався змінити це значення за замовчуванням на SameSite=Lax, ця зміна призведе до поломки тих програм/ веб-сайтів, які покладаються на сторонні чи міжсайтові файли cookie, але без визначеного атрибута SameSite. Враховуючи значні зміни для веб-розробників та обставини COVID-19, Google тимчасово скасував зміну файлів cookie SameSite.

Сторонні файли cookie

Зазвичай атрибут домену файлу cookie відповідатиме домену, який відображається в адресному рядку веб-переглядача. Це називається основним файлом cookie. Однак сторонні файли cookie належать до домену, відмінному від того, що показано в адресному рядку. Цей тип файлів cookie зазвичай з’являється, коли веб-сторінки містять вміст із зовнішніх веб-сайтів, наприклад рекламні банери. Це відкриває потенціал для відстеження історії перегляду користувача і часто використовується рекламодавцями, щоб показати релевантну рекламу кожному користувачеві.

Наприклад, припустимо, що користувач відвідує www.example.org. Цей веб-сайт містить рекламу від ad.foxytracking.com, яка під час завантаження

встановлює файл cookie, що належить домену реклами (ad.foxytracking.com). Потім користувач відвідує інший веб-сайт www.foo.com, який також містить рекламу від ad.foxytracking.com і встановлює файл cookie, що належить цьому домену (ad.foxytracking.com). Зрештою, обидва ці файли cookie будуть надіслані рекламодавцю під час завантаження його реклами або відвідування його веб-сайту. Потім рекламодавець може використовувати ці файли cookie для створення історії перегляду користувача на всіх веб-сайтах, на яких є реклама від цього рекламодавця, за допомогою поля заголовка посилання HTTP.

Станом на 2014 рік деякі веб-сайти встановлювали файли cookie, доступні для читання для понад 100 сторонніх доменів. У середньому один веб-сайт встановлював 10 файлів cookie, при цьому максимальна кількість файлів cookie (перших і сторонніх) сягала понад 800.

Більшість сучасних веб-переглядачів містять налаштування конфіденційності, які можуть блокувати сторонні файли cookie, а деякі тепер блокують усі сторонні файли cookie за замовчуванням — станом на липень 2020 року такі браузери включають Apple Safari, Firefox і Brave. Safari дозволяє вбудованим сайтам використовувати API доступу до сховища для запиту дозволу на встановлення сторонніх файлів cookie. У травні 2020 року Google Chrome представив нові функції для блокування сторонніх файлів cookie за замовчуванням у режимі анонімного перегляду для приватного перегляду, зробивши блокування необов'язковим під час звичайного перегляду. Це ж оновлення також додало можливість блокувати власні файли cookie. Chrome планує почати блокувати сторонні файли cookie за замовчуванням у 2023 році.

Суперкукі

Суперкукі — це файли cookie з походженням домену верхнього рівня (наприклад, .com) або загальнодоступного суфікса (наприклад, .co.uk). Звичайні файли cookie, навпаки, мають походження з певного доменного імені, наприклад example.com.

Суперкуки можуть бути потенційною проблемою безпеки і тому часто блокуються веб-браузерами. Якщо його розблокувати браузер, зловмисник, який контролює шкідливий веб-сайт, може встановити суперкукі і потенційно порушити або видати законні запити користувачів на інший веб-сайт, який має той самий домен верхнього рівня або загальнодоступний суфікс, що й шкідливий веб-сайт. Наприклад, суперкукі з джерелом .com може зловмисно вплинути на запит, зроблений до example.com, навіть якщо файл cookie походить не з example.com. Це можна використовувати для підробки входу або зміни інформації користувача.

Публічний список суфіксів допомагає зменшити ризик, який становлять суперкукі. Список загальнодоступних суфіксів — це ініціатива між постачальниками, яка має на меті надати точний та актуальний список суфіксів доменних імен. Старі версії браузерів можуть не мати оновленого списку, і тому будуть вразливі до суперкукі з певних доменів.

Інші види використання

Термін «суперкукі» іноді використовується для технологій відстеження, які не покладаються на файли cookie HTTP. Два таких механізму «суперкукі» були знайдені на веб-сайтах Microsoft у серпні 2011 року: синхронізація cookie, яка відродила файли cookie MUID (унікальний ідентифікатор машини), і файли cookie ETag.[40] Через увагу ЗМІ Microsoft пізніше відключила цей код. У дописі в блозі 2021 року Mozilla використовувала термін «суперкукі» для позначення використання кешу браузера як засобу відстеження користувачів на різних сайтах.

Використання

Управління сесією

Спочатку файли cookie були введені для того, щоб користувачі могли записувати товари, які вони хочуть придбати, під час навігації по веб-сайту (віртуальний «кошик» або «кошик для покупок»). Сьогодні, однак, вміст кошика для покупок користувача зазвичай зберігається в базі даних на сервері, а не в файлі cookie на клієнті. Щоб відстежувати, якого користувача

призначено до якого кошика, сервер надсилає клієнту файл cookie, який містить унікальний ідентифікатор сеансу (як правило, довгий рядок випадкових літер і цифр). Оскільки файли cookie надсилаються на сервер із кожним запитом клієнта, цей ідентифікатор сесії буде надсилатися назад на сервер щоразу, коли користувач відвідує нову сторінку веб-сайту, що дає серверу знати, який кошик для покупок відображати користувачеві.

Іншим популярним використанням файлів cookie є вхід на веб-сайти. Коли користувач відвідує сторінку входу на веб-сайт, веб-сервер зазвичай надсилає клієнту файл cookie, що містить унікальний ідентифікатор сеансу. Коли користувач успішно входить в систему, сервер запам'ятовує, що цей конкретний ідентифікатор сеансу пройшов автентифікацію, і надає користувачеві доступ до своїх послуг.

Оскільки файли cookie сеансу містять лише унікальний ідентифікатор сеансу, це робить кількість особистої інформації, яку веб-сайт може зберегти про кожного користувача, практично необмеженою — веб-сайт не обмежується обмеженнями щодо того, наскільки великими можуть бути файли cookie. Файли cookie сеансу також допомагають покращити час завантаження сторінки, оскільки обсяг інформації в файлі cookie сеансу невеликий і вимагає малої пропускної здатності.

Персоналізація

Файли cookie можна використовувати для запам'ятовування інформації про користувача, щоб показувати відповідний вміст цьому користувачеві з часом. Наприклад, веб-сервер може надіслати файл cookie, що містить ім'я користувача, яке було останнє використане для входу на веб-сайт, щоб він міг бути заповнений автоматично під час наступного входу користувача.

Багато веб-сайтів використовують файли cookie для персоналізації на основі уподобань користувача. Користувачі вибирають свої переваги, вводячи їх у веб-форму та надсилаючи форму на сервер. Сервер кодує налаштування в файлі cookie та надсилає файл cookie назад у браузер. Таким чином, щоразу, коли користувач отримує доступ до сторінки на веб-сайті, сервер може

персоналізувати сторінку відповідно до уподобань користувача. Наприклад, пошукова система Google колись використовувала файли cookie, щоб дозволити користувачам (навіть незареєстрованим) вирішувати, скільки результатів пошуку на сторінці вони хочуть бачити. Крім того, DuckDuckGo використовує файли cookie, щоб дозволити користувачам встановлювати параметри перегляду, наприклад кольори веб-сторінки.

2.2 Проектування структури бази даних

На рисунку 2.1 зображено схему бази, яка відображає її внутрішню структуру.

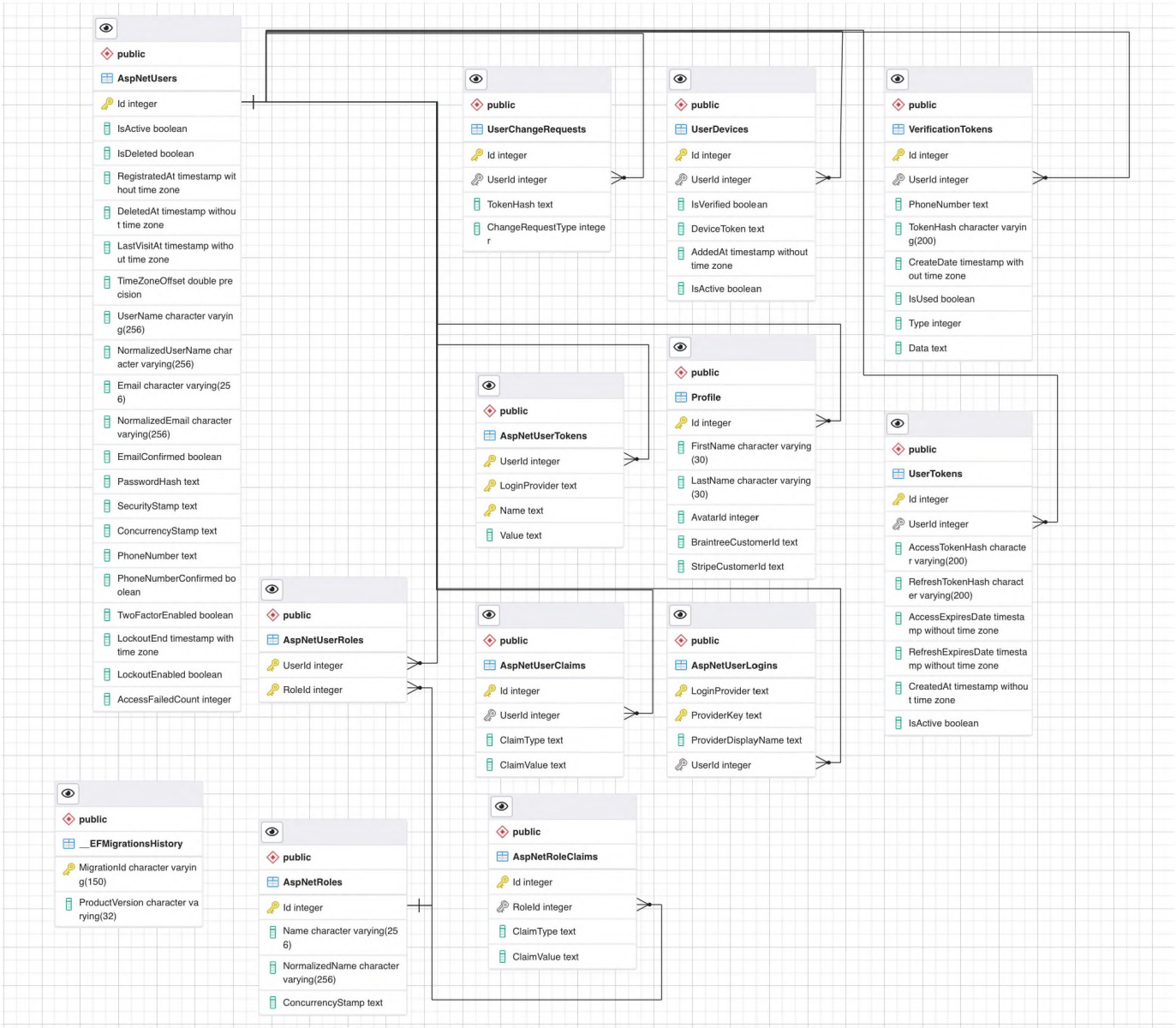

Рисунок 2.1 — Схема бази даних

База даних містить 13 сутностей, а саме:

- AspNetRoleClaims, яка зберігає данні про Id, RoleId, ClaimType, ClaimValue;

- AspNetRoles, яка зберігає данні про Id, Name, NormalizedName, ConcurrencyStamp;
- AspNetUserClaims, яка зберігає данні про Id, UserId, ClaimType, ClaimValue;
- AspNetUserLogins, яка зберігає данні про LoginProvider, ProviderKey, ProviderDisplayName, UserId;
- AspNetUserRoles, яка зберігає данні про UserId, RoleId;
- AspNetUserTokens, яка зберігає данні про UserId, LoginProvider, Name, Value;
- AspNetUsers, яка зберігає данні про Id, IsActive, IsDeleted, RegisteredAt, DeletedAt, LastVisitAt, TimeZoneOffset, UserName, NormalizedUserName, Email, NormalizedEmail, EmailConfirmed, PasswordHash, SecurityStamp, PhoneNumber, PhoneNumberConfirmed, TwoFactorEnebled, LockoutEnd, LockoutEnebled, AccessFailedCount;
- Profile, яка зберігає данні про Id, FirstName, LastName, AvatarId, BraintreeCustomerId, StripeCustomerId;
- UserChangeRequests, яка зберігає данні про Id, UserId, TockenHash, ChangeRequestType;
- UserDevices, яка зберігає данні про Id, UserId, IsVerified, DeviceToken, AddedAt, IsActive;
- UserTokens, яка зберігає данні про Id, UserId, AccessTokenHash, RefreshTokenHash, AccesExpiresDate, RefreshExpiresDate, CreatedAt, IsActive;
- VerificationTokens, яка зберігає данні про Id, UserId, PhoneNumber, TokenHash, CreateDate, IsUsed, Type, Data;
- __EFMigrationsHistory, яка зберігає данні про MigrationId, ProductVersion.

2.3 Реалізація Custom Middleware

Основна ідея - це реалізувати Custom Middleware для вставки токена у вхідний HTTP-запит. Після авторизації користувача ми зберігаємо cookie під певним ключем, наприклад: ".AspNetCore.Application.Id". Рекомендується задавати назву, ніяк не пов'язану з авторизацією або токенами, - у цьому випадку cookie з токеном будуть виглядати як якась непримітна системна константа AspNetCore програми. Такий вищий шанс, що зловмисник побачить багато системних змінних і, не розібравшись, який механізм авторизації використовується, піде далі.

Далі потрібно вставити цей токен у всі наступні HTTP-запити. Для цього ми напишемо кілька рядків коду Middleware. Це не що інше, як HTTP-pipeline.

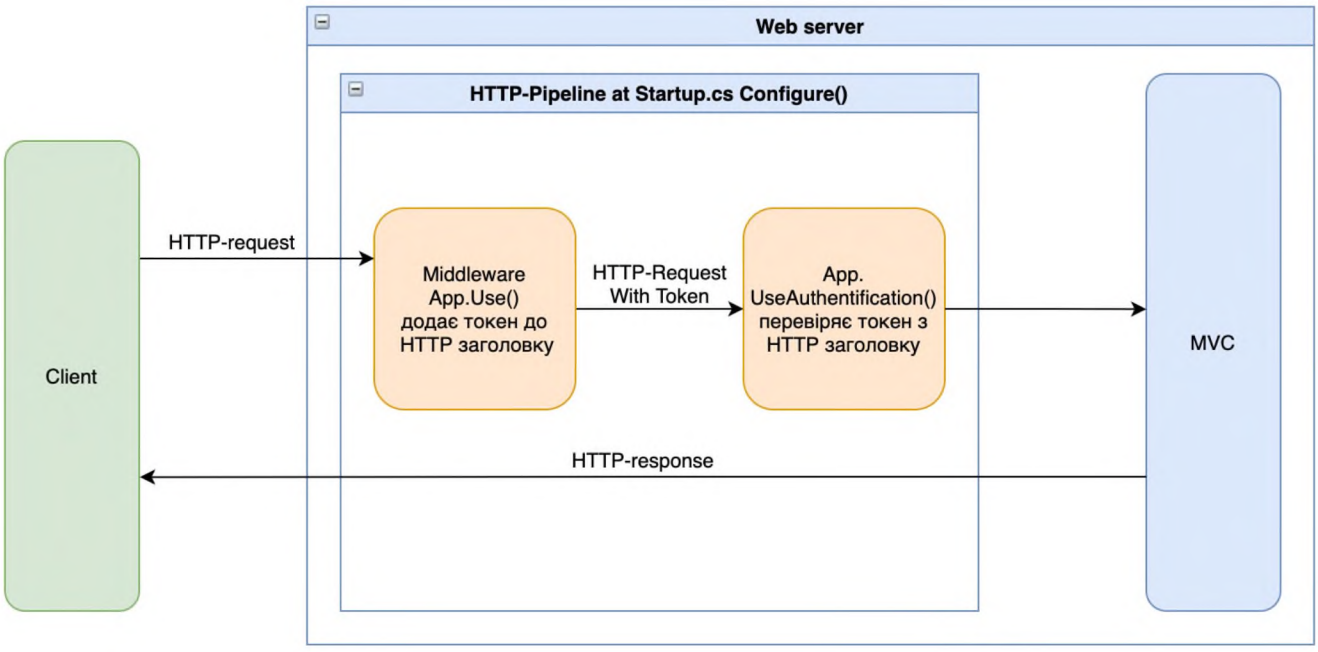

Рисунок 2.2 — HTTP-pipeline

Конфігурація

```
app.Use(async (context, next) =>
{
    var token = context.Request.Cookies[".AspNetCore.Application.Id"];
    if (!string.IsNullOrEmpty(token))
        context.Request.Headers.Add("Authorization", "Bearer " + token);

    await next();
});
app.UseAuthentication();
```

Для того, щоб записати значення в cookies, нам достатньо додати наступний рядок у логіку авторизації:

```
if (result.Succeeded)
    HttpContext.Response.Cookies.Append(".AspNetCore.Application.Id",
token,
    new CookieOptions
    {
        MaxAge = TimeSpan.FromMinutes(60)
    });
```

За допомогою наших cookie-policy ці cookie автоматично відправляться як httpOnly та secure. Не потрібно перевизначати їхню політику в cookie options.

У CookieOptions можна встановити MaxAge, щоб вказати час життя. Це корисно вказувати разом з JWT Lifetime при випуску токена, щоб cookie зникала після часу. Інші властивості CookieOptions настроюються залежно від вимог проекту.

Для забезпечення більшої безпеки раджу додати до Middleware наступні заголовки:

```
context.Response.Headers.Add("X-Content-Type-Options", "nosniff");
```

```
context.Response.Headers.Add("X-Xss-Protection", "1");
context.Response.Headers.Add("X-Frame-Options", "DENY");
```

- Назва X-Content-Type-Options використовується для захисту від уразливостей типу MIME sniffing. Ця вразливість може виникнути, коли сайт дозволяє користувачам завантажувати контент, однак користувач маскує певний тип файлу як щось інше. Це може дати зловмисникам можливість виконувати сценарії cross-site scripting або компрометувати веб-сайт.
- Всі сучасні браузери мають вбудовані можливості фільтрації XSS, які намагаються зловити уразливості XSS до того, як сторінка буде повністю відображена нам. За замовчуванням вони включені в браузері, але користувач може бути хитрішим і відключити їх. Використовуючи заголовок X-XSS-Protection, можна фактично сказати браузеру ігнорувати те, що зробив користувач, і застосовувати вбудований фільтр.
- X-Frame-Options повідомляє браузеру, що якщо ваш сайт розміщено всередині HTML-фрейму, то нічого не відображати. Це дуже важливо при спробі захистити себе від спроб clickjacking-злому.

2.4 Аналіз результатів

У першому розділі була проведена велика аналітична робота, за результатами якої ми отримали уявлення про існуючі підходи до програмної реалізації та актуальні проблеми та вразливості системи. Після цього було запропоновано та реалізовано вирішення поставленної проблеми у середовищі ASP.NET.

Рисунок 2.3 — Результат авторизації

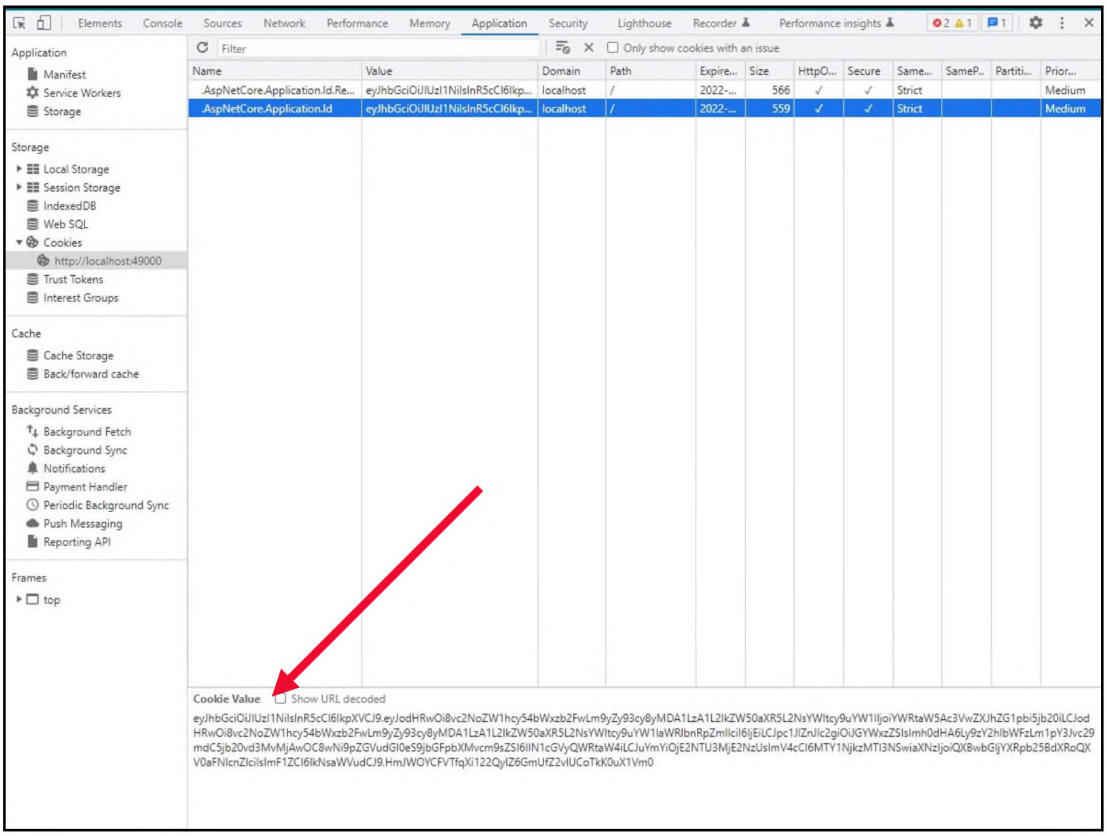

Рисунок 2.4 —Збережений JWT у файлі Cookies

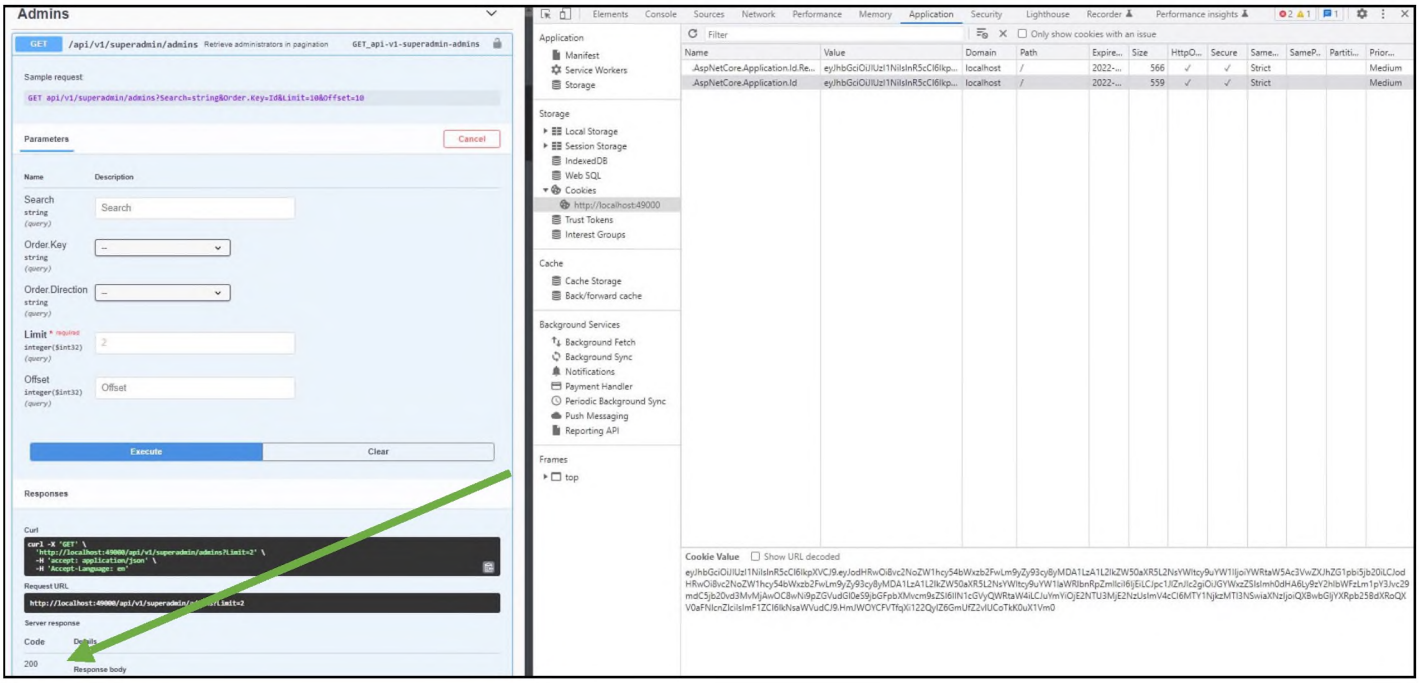

Рисунок 2.5 — Приклад успішного запиту на сервер з актуальним Cookies

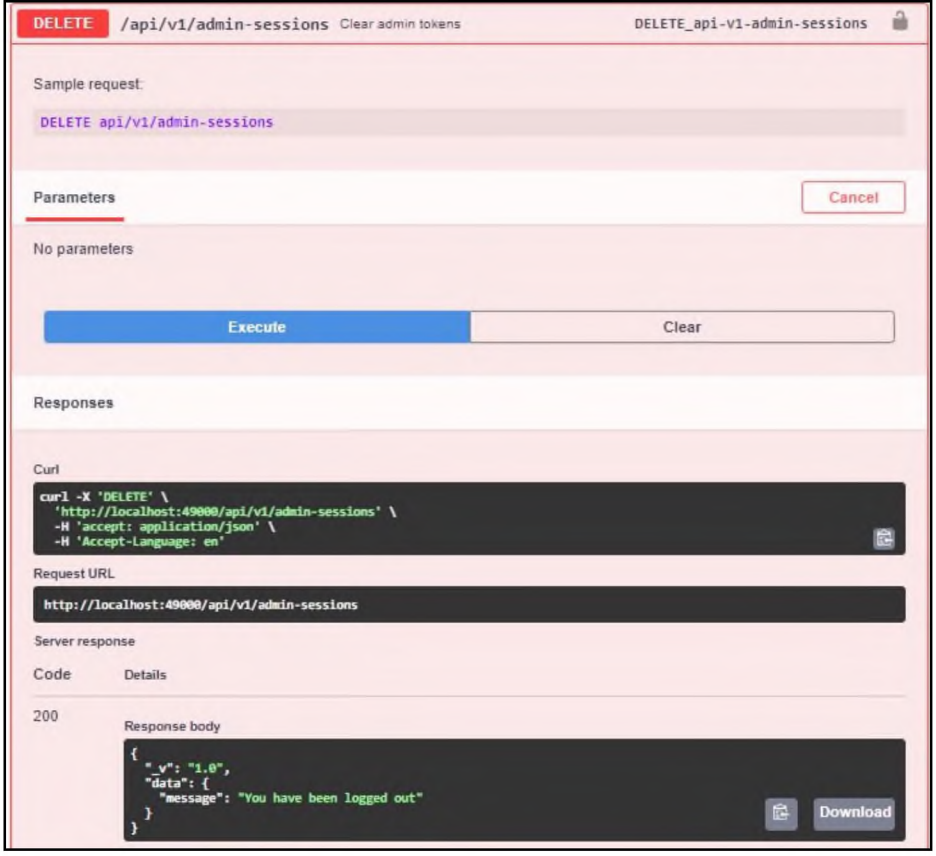

Рисунок 2.6 — Запит на видалення файлу Cookies та JWT

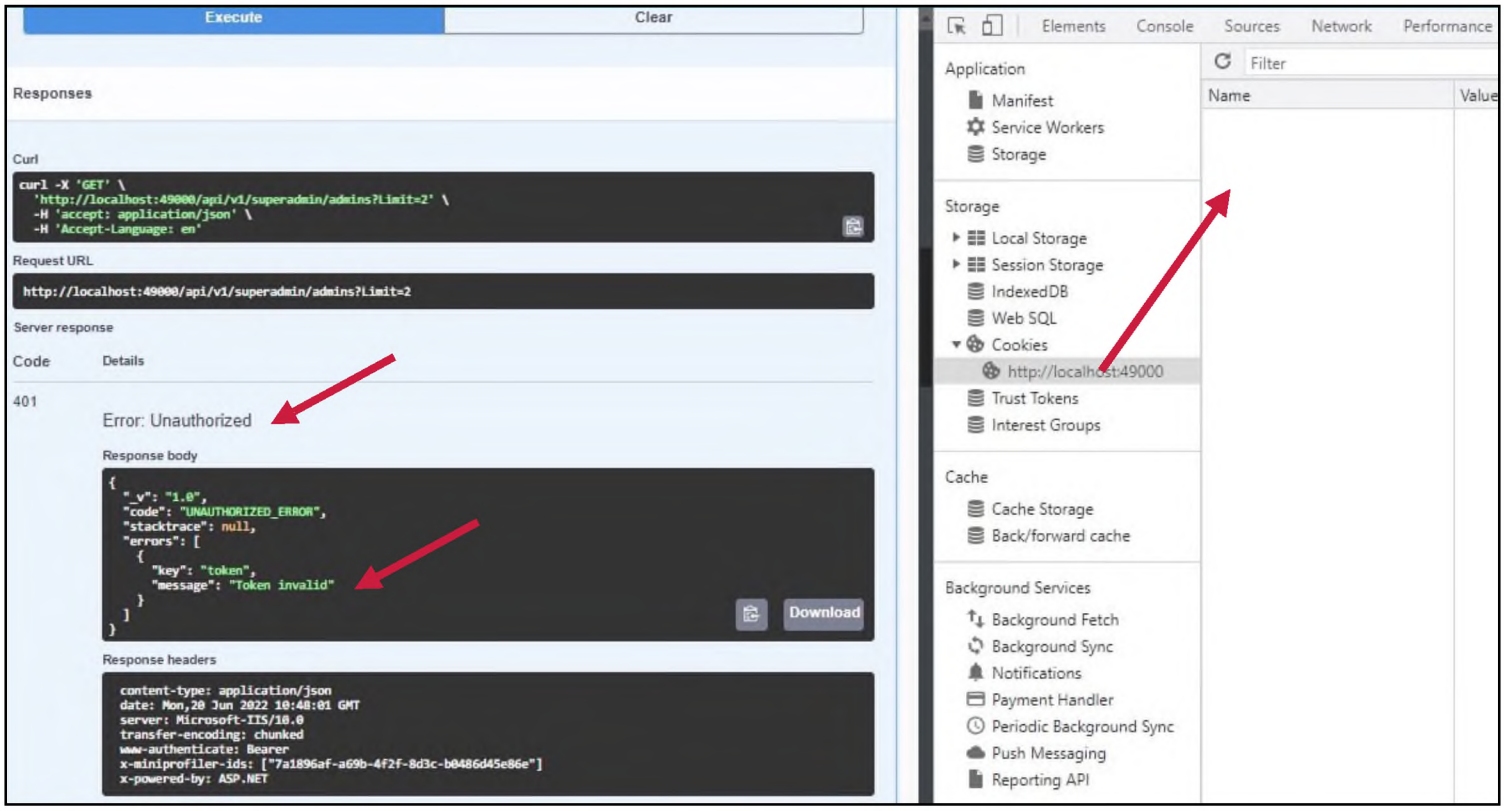

Рисунок 27 — Приклад запиту на сервер без актуального Cookies та JWT

```
#region Cookie auth

app.UseCookiePolicy(new CookiePolicyOptions
{
    MinimumSameSitePolicy = SameSiteMode.Strict,
    HttpOnly = HttpOnlyPolicy.Always,
    Secure = CookieSecurePolicy.Always
});

app.Use(async (context, next) =>
{
    var token = context.Request.Cookies[".AspNetCore.Application.Id"];
    if (!string.IsNullOrEmpty(token))
        context.Request.Headers.Add("Authorization", "Bearer " + token);

    await next();
});

#endregion
```

Рисунок 2.8 — Частина коду, яка відповідає за обробку Cookies

РОЗДІЛ 3
ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ

В даному розділі дипломної роботи проводиться економічне обґрунтування доцільності розробки програмного забезпечення. Зокрема, здійснюється розрахунок витрат на розробку програмного забезпечення, експлуатаційних витрат, ціни споживання проектного рішення. В заключній частині визначаються показники економічної ефективності нового програмного продукту, обґрунтовуються відповідні висновки.

Розроблене програмне забезпечення призначене для підвищення безпечності процесу обміну токенами між клієнтом та сервером.

3.1 Розрахунок витрат на розробку програмного забезпечення

Витрати на розробку і впровадження програмних засобів (К) включають:

$$К = К_1+К_2 \quad (3.1)$$

де $К_1$ - витрати на розробку програмних засобів, грн.

$К_2$ - витрати на відлагодження і дослідну експлуатацію програми рішення задачі на ЕОМ, грн.

- Витрати на розробку програмних засобів включають:
- витрати на оплату праці розробників;
- витрати на відрахування у спеціальні державні фонди (Вф,);
- витрати на куповані вироби (Кв);
- витрати на придбання спецобладнання для експериментальних робіт (Об);
- накладні витрати (Н);
- інші витрати (Ів).

Витрати на оплату праці розробників проекту визначаються за формулою:

$$З = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} t_{ij} C_{ij} \quad (3.2)$$

де nij - чисельність розробників і-ої спеціальності j-го тарифного розряду, які приймають участь в проектуванні, чол.;

t_{ij} - час, який затрачений на розробку проекту співробітника і-ої спеціальності j-го тарифного розряду, днів;

C_{ij} - денна заробітна плата і-ої спеціальності j-го тарифного розряду, грн.;

$$C_{ij} = \frac{C_{ij}^{0}(1+h)}{p} \quad (433)$$

де C_{ij} - основна місячна заробітна плата розробника і-ої спеціальності j-го тарифного розряду, грн.;

h - коефіцієнт, що визначає розмір додаткової заробітної плати;

p - середня кількість робочих днів у місяці (21).

Таблиця 3.1.

Вихідні дані для розрахунку витрат на оплату праці

№	Посада виконавців	Місячний оклад, грн.	Погодинна ставка, грн./ година
1	Керівник диплому, асистент	8500	52,7
2	Консультант з економіки	9000	55,9
3	Студент	0	0

Таблиця 3.2.

Розрахунок витрат на оплату праці

№	Спеціальність розробника	Час розробки, години	Погодинна заробітна плата, грн.	Витрати на розробку, грн.
1	Керівник диплому, асистент	18.5	52,7	974,95
2	Консультант з економіки	0.5	55,9	25,45
3	Студент	0	0	0
	Разом			1000,4

Величину відрахувань у спеціальні державні фонди визначають у процентному співвідношенні від суми основної та додаткової заробітної плати. Згідно діючого нормативного законодавства сума відрахувань у спеціальні державні фонди складає 22% від суми заробітної плати:

$$Вф=22: 100*З \quad (3.4)$$

$$Вф=0,22*1000,4=220 \text{ грн.}$$

Таблиця 3.3.

Розрахунок витрат на куповані вироби

№ п/п	Найменування купованих виробів	Одиниця виміру	Ціна за одиницю	Кількість купованих	Сума, грн.
1	Папір (формат А4)	500 листів	100,00	1	100,00
2	Зошит	Шт.	8,70	1	8,70
3	Диск (CD-RW)	Шт.	10,50	2	21,00
Всьог	129,70 грн				

При розробці даного програмного забезпечення спеціальне обладнання не використовувалось, тому витрати на спеціальне обладнання відсутні.

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими процентами до витрат на оплату праці:

$$H = \frac{30}{100} З \tag{3.5}$$

Н = 0,3*1000,4=300,12 грн.

Інші витрати відображають видатки, які не враховані в інших статтях витрат. Вони розраховуються за встановленими процентами до витрат на оплату праці:

$$І_в = 10:100*З \tag{3.6}$$

Ів=0,1*768,35=100,04 грн.

Витрати на розробку програмного забезпечення розраховуються за формулою:

$$К = З + Вф + Кв + Об + Н + Ів \tag{3.7}$$

К =1000,4+200+129,70+0+300,12+100,04=1730,26 грн.

Витрати на відлагодження і дослідну експлуатацію програмного забезпечення визначаються за формулою:

$$К_2 = S_{Мг} * t_{Від} \tag{3.8}$$

де $S_{Мг}$ - вартість однієї машино-години роботи конкретного типу ЕОМ, грн./год.;

$t_{Від}$ - машинний час, витрачений на відлагодження і дослідну експлуатацію програмних засобів, год.

Загальна кількість днів роботи на ЕОМ рівна 60 днів. Середній щоденний час роботи на ЕОМ - 6 год., тому:

$$t_{від} = 60*6 = 360 \text{ год.}$$

За занальними даними для ЕОМ типу IBM PC/AT $S_{Мг}$ = 5 грн.

Отже:

$$К_2 = 5,0*360 = 1800 \text{ грн.}$$

Таблиця 3.4.

Кошторис витрат на розробку програмного забезпечення

№	Найменування елементів витрат	Сума витрат, грн.
1	Витрати на оплату праці	1000,04
2	Відрахування у спеціальні державні фонди	200
3	Витрати на куповані вироби	129,7
4	Накладні витрати	300,12
5	Інші витрати	100,04
6	Витрати на відлагодження і дослідну експлуатацію програмного забезпечення	1800
	Всього	3529,9

3.2 Визначення експлуатаційних витрат

Для оцінки економічної ефективності розроблюваного програмного продукту слід порівняти його з аналогом, тобто існуючим програмним забезпеченням ідентичного функціонального призначення.

Експлуатаційні одноразові витрати по програмному забезпеченню і аналогу включають вартість підготовки даних і вартість машино-годин

роботи ЕОМ (за час дії програми):

$$E_{\text{П}} = E_{1\text{П}} + E_{2\text{П}} \tag{3.12}$$

де $E_{\text{П}}$ - одноразові експлуатаційні витрати на проектне рішення (аналог), грн.;

$E_{1\text{П}}$ - вартість підготовки даних для експлуатації проектного рішення (аналогу), грн.;

$E_{2\text{П}}$ - вартість машино-годин роботи ЕОМ для виконання проектного рішення (аналогу), грн.

Річні експлуатаційні витрати $B_{e\text{П}}$ визначаються за формулою:

$$B_{E\text{П}} = E_{\text{П}} * N_{\text{П}} \tag{3.13}$$

де $N_{\text{П}}$ - періодичність експлуатації проектного рішення (аналогу), раз/рік.

Вартість підготовки даних для роботи на ЕОМ визначається за формулою:

$$E_{1\text{П}} = \sum_{l=1}^{L} n_1 t_1 c_1 \tag{3.14}$$

де 1 - номери категорій персоналу, який приймає участь у підготовці даних (1=1,2,...L);

n_1, - чисельність співробітників 1-ої категорії, чол.;

t_1, - трудоємність роботи співробітників 1-ої категорії по підготовці

даних, год.;

c_1 — середнього динна ставка співробітника 1-ої категорії з врахуванням додаткової заробітної плати та відрахувань у спеціальні державні фонди, грн./ год.

$$c_1 = \frac{c_1^0(1+b)}{m} \qquad (3.15)$$

де c_1^0 - основна місячна заробітна плата працівника 1-ої категорії, грн.;

b - коефіцієнт, який враховує додаткову заробітну плату і відрахування у спеціальні державні фонди;

m - кількість робочих годин у місяці, год.

Для роботи з даними як для проектного рішення так і аналогу потрібен один працівник, основна місячна заробітна плата якого складає: с° = 6700 грн. Тоді, враховуючи додаткову заробітну плату:

$$c_1 = \frac{6700(1+0.57)}{162} = 64{,}9 грн/год$$

Трудомісткість працівника по підготовці даних для проектного рішення складає 1 год., для аналога 1,5 год.

Таблиця 3.7.

Розрахунок витрат на підготовку даних та реалізацію проектного рішення на ЕОМ

№	Час роботи співробітників, год.	Середньогодинна заробітна плата,	Витрати , грн.
	Проектне рішення		
1	1	64,9	64,9
	Аналог		
1	1,5	64,9	97,4

Витрати на експлуатацію ЕОМ визначається за формулою:

$$E_{2П} = t * S_{МГ} \tag{3.16}$$

де t - витрати машинного часу для реалізації проектного рішення (аналогу), год.;

$S_{МГ}$ - вартість однієї машино-години роботи ЕОМ, грн./год.

$E_{2п} = 1 * 5,0 = 5,0$ грн.

$E_{2а} = 1,5 * 5,0 = 7,5$ грн.

Еп = 64,9 + 5,0 = 69,9 грн.

Еа = 97,4 + 7,5 = 104,9 грн.

$В_{еп} = 64,9 * 252 = 16354,8$ грн.

$В_{еа} = 97,4 * 252 = 24544,8$ грн.

3.3 Розрахунок ціни споживання проектного рішення

Ціна споживання - це витрати на придбання і експлуатацію проектного рішення за весь строк його служби:

$$Цс_п=Ц_п+В_{епрv} \tag{3.17}$$

де $Ц_п$ - ціна придбання проектного рішення, грн.:

$$Ц_П = К(1+\frac{П_Р}{100}) + К_0 + К_к \tag{3.18}$$

де Пр - норматив рентабельності;

$К_о$ - витрати на прив'язку та освоєння проектного рішення на конкретному об'єкті, грн.;

$К_к$ - витрати на доукомплектування технічних засобів на об'єкті, грн.;

$Ц_п$=3527,38*(1+0,3) = 4585,6 грн.

$В_{епрv}$ - теперішня вартість витрат на експлуатацію проектного рішення (за весь час його експлуатації), грн.:

$$B_{епрv} = \sum_{t=0}^{T} \frac{B_{еП}}{(1+R)^t} \tag{3.19}$$

де $В_{еп}$ - річні експлуатаційні витрати, грн.;

T - строк служби проектного рішення, років;

R - річна ставка проценту банківського

$$B_{епрv} = \sum_{t=1}^{5} \frac{16354,8}{(1+0.16)^t} = 55550.4грн.$$

$$B_{епрv} = \sum_{t=1}^{5} \frac{24544,8}{(1+0.16)^t} = 80366.8грн.$$

Вартість аналога становить 5000 грн., тому отримуємо:

$Ц_{сп} = 4585,6 + 55550,4 = 60136$ грн.

$Ц_{са} = 5000 + 80366,8 = 85366.8$ грн.

3.4 Оцінка можливих збитків

Для розрахунку вартості такого збитку можна застосувати наступну спрощену модель оцінки для умовного підприємства.

Необхідні вихідні дані для розрахунку:

tn — час простою сервера внаслідок атаки, 1 година;

tв — час відновлення після атаки персоналом, що обслуговує сервер, 2 години;

tви — час повторного введення загубленої інформації співробітниками атакованого вузла 2 години;

Зо — заробітна плата обслуговуючого персоналу (адміністраторів та ін.), 5500 грн./міс.;

Зс — заробітна плата співробітнишв атакованого вузла, 11000 грн./міс.;

Чо — чисельність обслуговуючого персоналу (адміністраторів та ін.), 1 особа;

Чс — чисельність співробітнишв атакованого вузла, 7 осіб.;

О — обсяг прибутку атакованого вузла, 2 млн грн. у рік;

І — число атакованих сегментів корпоративної мережі, 1;

N — середне число атак на рік, 10.

Упущена вигода від простою атакованого сервера мессенджеру:

$$U=Пп+Пв+V=11740,4,$$

де Пп — оплачувані втрати робочого часу та простої співробітників атакованого вузла або сегмента корпоративної мережі, грн;

Пв — вартість відновлення працездатності вузла або сегмента корпоративної мережі (переустановлення системи, зміна конфігурації та ін.), грн;

V — втрати від зниження обсягу продажів за час простою атакованого вузла або сегмента корпоративної мережі, грн.

Втрати від зниження продуктивності співробітників атакованого вузла являють собою втрати їхньої заробітної плати (оплата непродуктивної праці) за час простою внаслідок атаки:

$$П_n = \frac{\sum Зс}{F} * t_n,$$

Пп = ((11000 *7)/176)*3 = 1312,5 грн,

де F — місячний фонд робочого часу (при 40-а годинному робочому тижні становить 176 ч).

Витрати на повторне введення інформації Пви розраховуються виходячи з розміру заробітної плати співробітників атакованого вузла або сегмента корпоративної мережі Зс, які зайняті повторним введенням втраченої інформації, з урахуванням необхідного для цього часу tви:

Пви = ((11000 *7)/176)*4 = 1750 грн,

Витрати на заміни устаткування або запасних частин можуть скласти 3200

Тоді витрати на відновлення працездатності вузла:

Пв = 1312,5 + 1750 + 125 = 1875 грн.

Таким чином, загальний збиток від атаки складе:

В = 1 * 14 * 13764,4 = 192700 грн.

3.5 Визначення показників економічної ефективності

Економічний ефект в сфері проектування рішення:

$$Е_{пр}=Ц_а-Ц_п \quad (3.21)$$

$$Е_{пр} = 5000,0 – 4585,6 = 414,4 \text{ грн.}$$

Річний економічний ефект в сфері експлуатації:

$$Е_{кс}=В_{еа}-В_{еп}$$

$$Е_{кс} = 24544,8 - 16354,8 = 8190 \text{ грн.}$$

Додатковий економічний ефект у сфері експлуатації:

$$\Delta E_{екс} = \sum_{t=1}^{T} E_{екс}(1+R)^{T-t}$$

$$\Delta E_{екс} = \sum_{t=1}^{5} 8190*(1+0,16)^{5-t} = 56323.7 грн .$$

Сумарний ефект складає:

$$E = E_{пр} + \Delta E_{екс} = 414,4 + 56323.7 = 56738,1 грн$$

Таблиця 3.8.

Показники економічної ефективності проектного рішення

№	Найменування	Одиниці вимірювання	Значення показників	
			Базовий варіант	Новий варіант
1	Капітальні вкладення	Грн.	-	3527,38
2	Ціна придбання	Грн.	5000,0	4585,6
3	Річні експлуатаційні витрати	Грн.	5922,0	3956,4
4	Ціна споживання	Грн.	24544,8	16354,8
5	Економічний ефект в сфері проектування	Грн.	-	414,4
6	Річний економічний ефект в сфері експлуатації	Грн.	-	8190
7	Додатковий ефект в сфері експлуатації	Грн.	-	56738,1
8	Сумарний ефект	Грн.	57152,5	

3.6 Висновки

В даному розділі проведено розрахунок витрат на розробку проектного рішення. Здійснено порівняння з існуючим аналогом, і цим показано, що дане проектне рішення має переваги в порівнянні з аналогами, зокрема: надійність, простота використання, гнучкість, зручність. Згідно проведеного економічного обґрунтування дане проектне рішення є конкурентноздатним. Крім того, отримано додатній економічний ефект у розмірі 56738,1грн. і тому розробка і впровадження цього проектного рішення є економічно доцільними.

ВИСНОВКИ

Мета роботи полягала у створенні власного програмного рішення для підвищення рівня безпеки передачі, зберігання та обробки токенів у середовищі ASP.NET.

Для Досягнення мети було виконано наступні завдання:

— Проведення аналізу основних понять предметної області;

— Підбір найкращого технічного рішення;

— Вибір середовища реалізації технічного рішення;

— Проектування бази даних;

— Розробка Web-серверу;

— Реалізація технічного рішення підвищення рівня безпеки передачі та збереження токенів.

Завдяки чіткому виконанню даних завдань, в результаті було отримано повнофункціональний Web-сервер з більш безпечними алгоритмами передачі, обробки та збереження токенів доступу до критичних ресурсів, який повністю виконує закладений в нього функціонал.

У першому розділі була проведена велика аналітична робота, за результатами якої ми отримали уявлення про існуючі підходи до програмної реализації та актуальні проблеми та вразливості системи. Після цього було запропоновано та реалізовано вирішення поставленної проблеми у середовищі ASP.NET.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "What are cookies? What are the differences between them (session vs. persistent)?". Cisco. 17 July 2018. 117925.
2. Vamosi, Robert (14 April 2008). "Gmail cookie stolen via Google Spreadsheets". News.cnet.com. Archived from the original on 9 December 2013. Retrieved 19 October 2017.
3. "What about the "EU Cookie Directive"?". WebCookies.org. 2013. Archived from the original on 11 October 2017. Retrieved 19 October 2017.
4. "New net rules set to make cookies crumble". BBC. 8 March 2011. Archived from the original on 10 August 2018. Retrieved 21 June 2018.
5. "Sen. Rockefeller: Get Ready for a Real Do-Not-Track Bill for Online Advertising". Adage.com. 6 May 2011. Archived from the original on 24 August 2011. Retrieved 2 June 2011.
6. "Where cookie comes from :: DominoPower". dominopower.com. Archived from the original on 19 October 2017. Retrieved 19 October 2017.
7. Raymond, Eric (ed.). "magic cookie". The Jargon File (version 4.4.7). Archived from the original on 6 September 2017. Retrieved 8 September 2017.
8. "Why are internet cookies called cookies?".
9. Schwartz, John (4 September 2001). "Giving Web a Memory Cost Its Users Privacy". The New York Times. Archived from the original on 18 November 2011. Retrieved 19 February 2017.
10. Kesan, Jey; and Shah, Rajiv; Deconstructing Code Archived 2018-08-19 at Archive-It, SSRN.com, chapter II.B (Netscape's cookies), Yale Journal of Law and Technology, 6, 277–389
11. Kristol, David; HTTP Cookies: Standards, privacy, and politics, ACM Transactions on Internet Technology, 1(2), 151–198, 2001 doi:10.1145/502152.502153 (an expanded version is freely available at [https://web.archive.org/web/20140716051321/http://arxiv.org/abs/cs.SE/

0105018 Archived 2014-07-16 at the Wayback Machine arXiv:cs/0105018v1 [cs.SE]])

12. "Press Release: Netscape Communications Offers New Network Navigator Free On The Internet". Archived from the original on 7 December 2006. Retrieved 22 May 2010.
13. "Usenet Post by Marc Andreessen: Here it is, world!". 13 October 1994. Archived from the original on 27 April 2011. Retrieved 22 May 2010.
14. Kristol, David M. (November 2001). "HTTP Cookies". ACM Transactions on Internet Technology. 1 (2): 151–198. arXiv:cs/0105018. doi:10.1145/502152.502153. ISSN 1533-5399. S2CID 1848140.
15. US 5774670, Montulli, Lou, "Persistent client state in a hypertext transfer protocol based client-server system", published 1998-06-30, assigned to Netscape Communications Corp.
16. Hardmeier, Sandi (25 August 2005). "The history of Internet Explorer". Microsoft. Archived from the original on 1 October 2005. Retrieved 4 January 2009.
17. Jackson, T (12 February 1996). "This Bug in Your PC is a Smart Cookie". Financial Times.
18. "Rfc2109".
19. "Setting Cookies". staff.washington.edu. 19 June 2009. Archived from the original on 16 March 2017. Retrieved 15 March 2017.
20. The edbrowse documentation version 3.5 said "Note that only Netscape-style cookies are supported. However, this is the most common flavor of cookie. It will probably meet your needs." This paragraph was removed in later versions of the documentation Archived 2017-03-16 at the Wayback Machine further to RFC 2965's deprecation.
21. Hodges, Jeff; Corry, Bil (6 March 2011). "'HTTP State Management Mechanism' to Proposed Standard". The Security Practice. Archived from the original on 7 August 2016. Retrieved 17 June 2016.

22."Set-Cookie2 - HTTP | MDN". developer.mozilla.org. Retrieved 8 March 2021.

23.Microsoft Support Description of Persistent and Per-Session Cookies in Internet Explorer Archived 2011-09-25 at the Wayback Machine Article ID 223799, 2007

24."Maintaining session state with cookies". Microsoft Developer Network. Archived from the original on 14 October 2012. Retrieved 22 October 2012.

25."'SameSite' cookie attribute, Chrome Platform tatus". Chromestatus.com. Archived from the original on 9 May 2016. Retrieved 23 April 2016.

26.Goodwin, M.; West (20 June 2016). "Same-Site Cookies draft-ietf-httpbis-cookie-same-site-00". tools.ietf.org. Archived from the original on 16 August 2016. Retrieved 28 July 2016.

27."Using the Same-Site Cookie Attribute to Prevent CSRF Attacks". www.netsparker.com. Retrieved 5 April 2021.

28."Require "Secure" for "SameSite=None". by miketaylr · Pull Request #1323 · httpwg/http-extensions". GitHub. Retrieved 5 April 2021.

29."Browser Compatibility Testing of 'SameSite' cookie attribute".

30."SameSite Cookie Changes in February 2020: What You Need to Know". Chromium Blog. Retrieved 5 April 2021.

31."Temporarily rolling back SameSite Cookie Changes". Chromium Blog. Retrieved 5 April 2021.

32."Third party domains". WebCookies.org. Archived from the original on 9 December 2014. Retrieved 7 December 2014.

33."Number of cookies". WebCookies.org. Archived from the original on 9 December 2014. Retrieved 7 December 2014.

34.Statt, Nick (24 March 2020). "Apple updates Safari's anti-tracking tech with full third-party cookie blocking". The Verge. Retrieved 24 July 2020.

35."Firefox starts blocking third-party cookies by default". VentureBeat. 4 June 2019. Retrieved 24 July 2020.

36. Brave (6 February 2020). "OK Google, don't delay real browser privacy until 2022". Brave Browser. Retrieved 24 July 2020.
37. Protalinski, Emil (19 May 2020). "Chrome 83 arrives with redesigned security settings, third-party cookies blocked in Incognito". VentureBeat. VentureBeat. Retrieved 25 June 2020.
38. Goel, Vinay (24 June 2021). "An updated timeline for Privacy Sandbox milestones". Google Chrome Official Blog. Retrieved 13 October 2021.
39. "Learn more about the Public Suffix List". Publicsuffix.org. Archived from the original on 14 May 2016. Retrieved 28 July 2016.
40. Mayer, Jonathan (19 August 2011). "Tracking the Trackers: Microsoft Advertising". The Center for Internet and Society. Archived from the original on 26 September 2011. Retrieved 28 September 2011.
41. Vijayan, Jaikumar. "Microsoft disables 'supercookies' used on MSN.com visitors". Archived from the original on 27 November 2014. Retrieved 23 November 2014.
42. Englehardt, Steven; Edelstein, Arthur (26 January 2021). "Firefox 85 Cracks Down on Supercookies".
43. Angwin, Julia; Tigas, Mike. "Zombie Cookie: The Tracking Cookie That You Can't Kill". ProPublica. Retrieved 1 November 2020.
44. Stolze, Conrad (11 June 2011). "The Cookie That Would Not Crumble!". 24x7 Magazine. Retrieved 1 November 2020.
45. Peng, Weihong; Cisna, Jennifer (2000). "HTTP Cookies, A Promising Technology". ProQuest. Online Information Review. ProQuest 194487945.
46. Jim Manico quoting Daniel Stenberg, Real world cookie length limits Archived 2013-07-02 at the Wayback Machine
47. Lee, Wei-Bin; Chen, Hsing-Bai; Chang, Shun-Shyan; Chen, Tzung-Her (25 January 2019). "Secure and efficient protection for HTTP cookies with self-verification". International Journal of Communication Systems. 32 (2): e3857. doi:10.1002/dac.3857. S2CID 59524143.
48. Rainie, Lee (2012). Networked: The New Social Operating System. p. 237

49. IETF HTTP State Management Mechanism, Apr, 2011 Obsoletes RFC 2965
50. "Persistent client state HTTP cookies: Preliminary specification". Netscape. c. 1999. Archived from the original on 5 August 2007.
51. "Cookie Property". MSDN. Microsoft. Archived from the original on 5 April 2008. Retrieved 4 January 2009.
52. Shannon, Ross (26 February 2007). "Cookies, Set and retrieve information about your readers". HTMLSource. Archived from the original on 24 August 2011. Retrieved 4 January 2009.
53. "HTTP State Management Mechanism, The Path Attribute". IETF. March 2014. Archived from the original on 1 May 2011. Retrieved 12 May 2011.
54. "RFC 6265, HTTP State Management Mechanism, Domain matching". IETF. March 2014. Archived from the original on 1 May 2011. Retrieved 12 May 2011.

ДОДАТОК А. Відомість матеріалів кваліфікаційної роботи

№	Формат	Найменування	Кількість листів	Примітки
Документація				
1	А4	Реферат	2	
2	А4	Перелік умовних скорочень	1	
3	А4	Зміст	1	
4	А4	Вступ	1	
5	А4	Аналіз предметної області	19	
6	А4	Проектування і програмна реалізація	17	
7	А4	Економічне обгрунтування доцільності розробки	13	
8	А4	Висновки	1	
9	А4	Перелік використаних джерел	10	
10	А4	Додаток А	1	
11	А4	Додаток Б	1	
12	А4	Додаток В	1	
13	А4	Додаток Г	1	

ДОДАТОК Б. Перелік документів на оптичному носії

1.ГрінченкоАС_ПЗ-125-18-1.pdf

2.Диплом_ГрінченкоАС_125-18-1.pdf

3.Презентація_ГрінченкоАС_125-18-1.pdf

4.Програмні коди.zip

ДОДАТОК В. Відгук керівника економічного розділу

Керівник розділу _____ _____

(підпис) (Прізвище, ініціали)

Додаток Г. Відгук керівника кваліфікаційної роботи

В І Д Г У К

на кваліфікаційну роботу студента групи 125-18-1 Грінченка А.С.

на тему: «Підвищення рівня безпеки під час обміну JWT в ASP.NET Core.»

Пояснювальна записка складається зі вступу, трьох розділів і висновків, розташованих на 60 сторінках.

Мета роботи є актуальною, оскільки вона спрямована на підвищення безпеки процесів при отриманні доступу до інформаційних систем, що є гострою проблемою сьогодення.

При виконанні роботи автор продемонстрував добрий рівень теоретичних знань і практичних навичок. На основі аналізу проблеми та можливостей середи програмування ASP.NET, завдяки цьому було сформульовано найголовнішу проблему під час використання ПО, яка була вирішена завдяки програмній реалізації.

Практична цінність роботи полягає у тому, що віднайдений програмний засіб є повноцінною можливістю уникнення викрадення критичної інформації для будь яких структур.

Рівень запозичень у кваліфікаційній роботі не перевищує вимог «Положення про систему виявлення та запобігання плагіату».

В цілому робота задовольняє усім вимогам, а її автор Грінченко А.С. заслуговує на оцінку « » та присвоєння кваліфікації «Бакалавр з кібербезпеки» за спеціальністю 125 Кібербезпека.

Керівник роботи,

ст.викладач **Саксонов Г.М**