МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2023 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

СТВОРЕННЯ ЗАХИЩЕНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ СФЕРИ ОХОРОНИ ЗДОРОВ'Я

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21910112

Виконав студент 4-го курсу, групи 401
_____***К. О. Колодяжний***
«20» червня 2023 р.

Керівник: д-р техн. наук, проф.
_____***І. М. Журавська***
«20» червня 2023 р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____20____р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Колодяжному Кирилу Олексійовичу.

1. Тема кваліфікаційної роботи «Створення захищеної інформаційної системи для сфери охорони здоров'я».

Керівник роботи Журавська Ірина Миколаївна, д-р техн. наук, проф.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» ____ 20__ р. № _____

2. Строк представлення кваліфікаційної роботи студентом «__» _____ 20__ р.

3. Вхідні (початкові) дані до роботи: архітектура інформаційної системи сфери охорони здоров'я.

Очікуваний результат роботи: захищенні механізми автентифікації та авторизації.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– аналіз архітектури інформаційної системи в сфері охорони здоров'я та конкретно механізмів надання ідентифікації та авторизації;

– виявлення вразливих елементів в механізмах автентифікації та авторизації і створення рекомендацій по їх усуненню;

– моделювання та вибір способів створення механізмів автентифікації та авторизації;

– розроблення механізмів автентифікації та авторизації на основі створених рекомендацій та обраних технологій.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини «Вимоги до роботи із комп'ютерними пристроями».

7. Консультанти:

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва Анна Олександрівна, канд. техн. наук, доцент кафедри екології Медичного інституту ЧНУ імені Петра Могили	

Керівник роботи __д-р техн. наук, проф. Журавська І. М._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Колодяжний К. О._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_23_» ____листопада____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: Створення захищеної інформаційної системи для сфери охорони здоров'я.

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	27.10.2022	29.10.2022	Виконано
2	Отримання завдання на виконання БКР	25.11.2022	25.11.2022	Виконано
3	Складання календарного плану роботи на весь період виконання БКР	10.12.2022	10.12.2022	Виконано
4	Отримання завдання на переддипломну практику	01.05.2023	01.05.2023	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	01.05.2023	14.05.2022	Виконано
6	Розробка звіту з переддипломної практики	15.05.2023	17.05.2023	Виконано
7	Виконання БКР: аналіз архітектури інформаційної системи, створення рекомендацій до існуючих вразливостей, огляд існуючих технологій, розробка ПЗ	15.05.2023	19.06.2023	Виконано
8	Попередній захист БКР на засіданні комісії кафедри	29.05.2023	30.05.2023	Виконано
9	Доробка та остаточне оформлення БКР	02.06.2023	19.06.2023	Виконано
10	Подання БКР рецензенту	15.06.2023	17.06.2023	Виконано
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	19.06.2023	22.06.2023	Виконано
12	Захист БКР перед екзаменаційною комісією (ЕК)	26.06.2023	29.06.2023	Виконано

Розробив студент Колодяжний К. О. _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи д-р техн. наук, проф. Журавська І. М _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«_09_» _____12_____ 2022 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента групи 401 ЧНУ ім. Петра Могили

Колодяжного Кирила Олексійовича

Тема: «Створення захищеної інформаційної система для сфери охорони здоров'я»

Актуальність роботи полягає у необхідності постійного підвищення рівня захищеності інформаційних систем, що набувають все більшого поширення у сфері охорони здоров'я.

Об'єкт роботи – процес надання доступу до інформаційної системи.

Предмет роботи – механізми автентифікації та авторизації в комп'ютерних системах (КС).

Метою бакалаврської кваліфікаційної роботи є надання рекомендацій щодо покращення захисту КС шляхом впровадження запропонованих безпечних механізмів автентифікації та авторизації.

Пояснювальна записка складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та двох додатків.

У першому розділі був проведений аналіз механізмів автентифікації та авторизації користувачів інформаційної системи в сфері охорони здоров'я та.

У другому розділі було здійснено огляд вразливостей проаналізованих механізмів та розроблено рекомендації для їх подальшого використання при розробці.

У третьому розділі обрано технології, котрі будуть використовуватись при розробці, а також виконано моделювання програми, умовної центральної бази даних та ролей користувачів.

У четвертому розділі описано процес розробки програмного забезпечення та створено механізми автентифікації та авторизації.

В результаті створено захищену інформаційну систему для сфери охорони здоров'я та розроблено рекомендації щодо покращення захисту КС.

Бакалаврська кваліфікаційна робота містить: сторінок 67, рисунків 57, додатків 2, джерел 25.

Ключові слова: *автентифікація, авторизація, інформаційна система, сфера охорони здоров'я, контроль доступу, JWT, .NET, Microsoft SQL Server, безпечні механізми ідентифікації.*

ABSTRACT

of the bachelor's degree student of the group 401 in Petro Mohyla Black Sea National University

Kolodiazhnyi Kyrylo Oleksiiovych

«Creating a secure information system for the healthcare sector»

The relevance of the work lies in the need to constantly increase the level of security of information systems that are becoming more and more widespread in the healthcare sector.

Object of work – the process of providing access to the information system.

Subject of work – mechanisms of authentication and authorization in computer systems (CS).

The purpose of the bachelor's qualification work is offering recommendations for improving CS protection by implementing the proposed secure authentication and authorization mechanisms.

The explanatory note consists of an introduction, four chapters, conclusions, references and two appendices.

The first section analyzes of the authentication and authorization mechanisms by users in the healthcare information system.

The second section reviews the vulnerabilities of the analyzed mechanisms and develops recommendations for their further development.

The third section selects the technologies that will be used in the development, as well as modeling the application, the conditional central database, and user roles.

The fourth section describes the software development process and creates authentication and authorization mechanisms.

As a result, a secure information system for the healthcare sector has created and recommendations for improving the protection of the CS has developed.

The bachelor's qualification work contains: pages 67, figures 57, appendices 2, sources 25.

Key words: *authentication, authorization, information system, healthcare, access control, JWT, .NET, Microsoft SQL Server, secure identification mechanisms*.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АСУ	–	автоматизована система управління
БД	–	база даних
ЕМЗ	–	електронний медичний запис
ЕН	–	електронне направлення
ІЛС	–	інформаційно-логічна система
ІПС	–	інформаційно-пошукова система
ІС	–	інформаційної системи
МІС	–	медична інформаційна система
НСД	–	несанкціонований доступ
ПМД	–	первинна медична допомога
СМД	–	спеціалізована медична допомога
ЦБД	–	центральна база даних

ABAC	–	attribute-based access control
ACL	–	access control lists
ASVS	–	application security verification standard
CAPTCHA	–	completely automated public turing test to tell computers and humans apart
CSRF	–	cross site request forgery
IP	–	Internet protocol
ISP	–	Internet service provider
NIST	–	national institute of standards and technology
OWASP	–	open worldwide application security project
RBAC	–	role-based access control
SSO	–	single sign-on
SQL	–	structured query language
NoSQL	–	non structured query language
XSS	–	cross site scripting

ВСТУП

З кожним днем кількість користувачів мережі Інтернет зростає, тому кількість пацієнтів, котрі прагнуть керувати своєю медичною інформацією та спілкуватися з лікарями за допомогою медичних застосунків зростає. Через збільшення пацієнтів, потреба в надійних заходах безпеки для захисту конфіденційних медичних і персональних даних набуває першочергового значення. Оскільки інформаційна система сфери охорони здоров’я включає в собі різні види даних, їхній рівень конфіденційності відрізняється: починаючи з персональних карток пацієнтів і закінчуючи кількістю необхідних ліків для закупівлі. Через це, інформаційні системи мають велику кількість користувачів з різними рівнями доступу, що збільшує потребу у їхньому контролі. На жаль, інформаційні системи охорони здоров’я не застраховані від витоків даних, які стали більш поширеними і гучними в останні роки [3; 4].

Персональні данні пацієнтів, лікарів, фармацевтів, агентів та інших користувачів а також функціонал інформаційних систем мають велику цінність для зловмисник. Отримавши доступ, зловмисник може використовувати це для:

– збору та зміни інформації. Отримавши несанкціонований доступ (НСД) до інформаційної системи (ІС), зловмисник має можливість для збору та/або зміні будь-якої інформації о користувачах з різними правами (працівники, адміністраторі, пацієнти тощо), налаштувань цієї та інших підключених до неї інформаційних систем та інших даних, до яких можливо отримати доступ різного рівня;

– вимагання коштів. Отримавши НСД до ІС та маючи права, зловмисник може створювати тиск на власників ІС. Це може бути як популярний зараз метод шифрування даних, так і часткове публікування отриманих чутливих даних для демонстрації їх володіння та можливості публікування всього обсягу.

НСД можливо отримати через різні вразливості інформаційної системи. Серед них, основними є отримання доступу до існуючих акаунтів користувачів та подальші запити на отримання інформації від їхнього імені.

Здобування доступу до акаунтів існуючих користувачів також ділиться на різні атаки. В рамках даної кваліфікованій роботи, буде проаналізовано атаки на механізми автентифікації та авторизації та розроблено більш надійні з урахуванням можливих ризиків.

Актуальність – механізми автентифікації та авторизації належать до критичних функцій та їхній аналіз необхідний для безпечного використання інформаційної системи у сфері охорони здоров’я.

Завдання які мають бути вирішеними:

– аналіз атак на механізм надання сесійних токенів авторизації;

– знаходження механізму, котрий буде відповідати всім вимогам безпеки;

– розробка, на основі проаналізованих даних, більш безпечного механізму.

Об’єкт дослідження – процес надання доступу до інформаційної системи.

Предмет дослідження – механізми автентифікації та авторизації в комп’ютерних системах (КС).

Метою роботи є надання рекомендацій щодо покращення захисту КС шляхом впровадження запропонованих безпечних механізмів автентифікації та авторизації.

Методи, що мають бути використані: мова програмування .NET для розробки, та система управління базами даних Microsoft SQL Server, для роботи з даними користувачів.

Основні положення, запропоновані для підвищення рівня безпеки медичних ІС, пройшли апробацію під час XXV Всеукраїнської науково-методичної конференції «Могилянські читання – 2022» (м. Миколаїв, 7-11 листопада 2022 р.). За результатами опубліковано тези доповіді [5].

1 АНАЛІТИЧНИЙ ОГЛЯД МЕХАНІЗМІВ АВТЕНТИФІКАЦІЇ І АВТОРИЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ СФЕРИ ОХОРОНИ ЗДОРОВ’Я

1.1 Аналіз механізмів автентифікації та авторизації у вебзастосунках

Всі інформаційні системи, котрі працюють з важливими даними, мають протоколи безпеки від зовнішніх та внутрішніх ризиків. Протоколами безпеки від зовнішніх ризиків приділяють особливу увагу, через їхнє розташування у відкритій мережі Інтернет. Одним з перших механізмів захисту інформації є ідентифікація користувача, або автентифікація, та перевірка його доступу до даних, або авторизація.

Автентифікація – це процес перевірки особи користувача або системи, щоб переконатися, що вони є тими, за кого себе видають. У контексті комп'ютерних наук автентифікація є важливим аспектом інформаційної безпеки і використовується для контролю доступу до таких ресурсів, як комп'ютерні системи, мережі, програми та дані.

Автентифікація може бути здійснена за допомогою різних методів, зокрема:

– ім'я користувача та пароль: найпоширеніша форма автентифікації, коли користувач вводить ім'я користувача та секретний пароль, який знає лише він;

– багатофакторна автентифікація: Цей метод передбачає використання двох або більше факторів автентифікації, таких як щось, що користувач знає (наприклад, пароль), щось, що у нього є (наприклад, смарт-карта або токен безпеки), або щось, чим він є (наприклад, біометричне сканування відбитків пальців або обличчя);

– автентифікація на основі сертифікатів: Цей метод використовує цифрові сертифікати для перевірки особи користувача або системи. Сертифікати видаються довіреною третьою стороною, наприклад, центром

сертифікації, і використовуються для підтвердження особи користувача або системи;

– єдиний вхід (Single sign-on, SSO): Цей метод дозволяє користувачам увійти в систему один раз і отримати доступ до декількох систем або застосунків без необхідності щоразу вводити свої облікові дані (рис. 1.1).

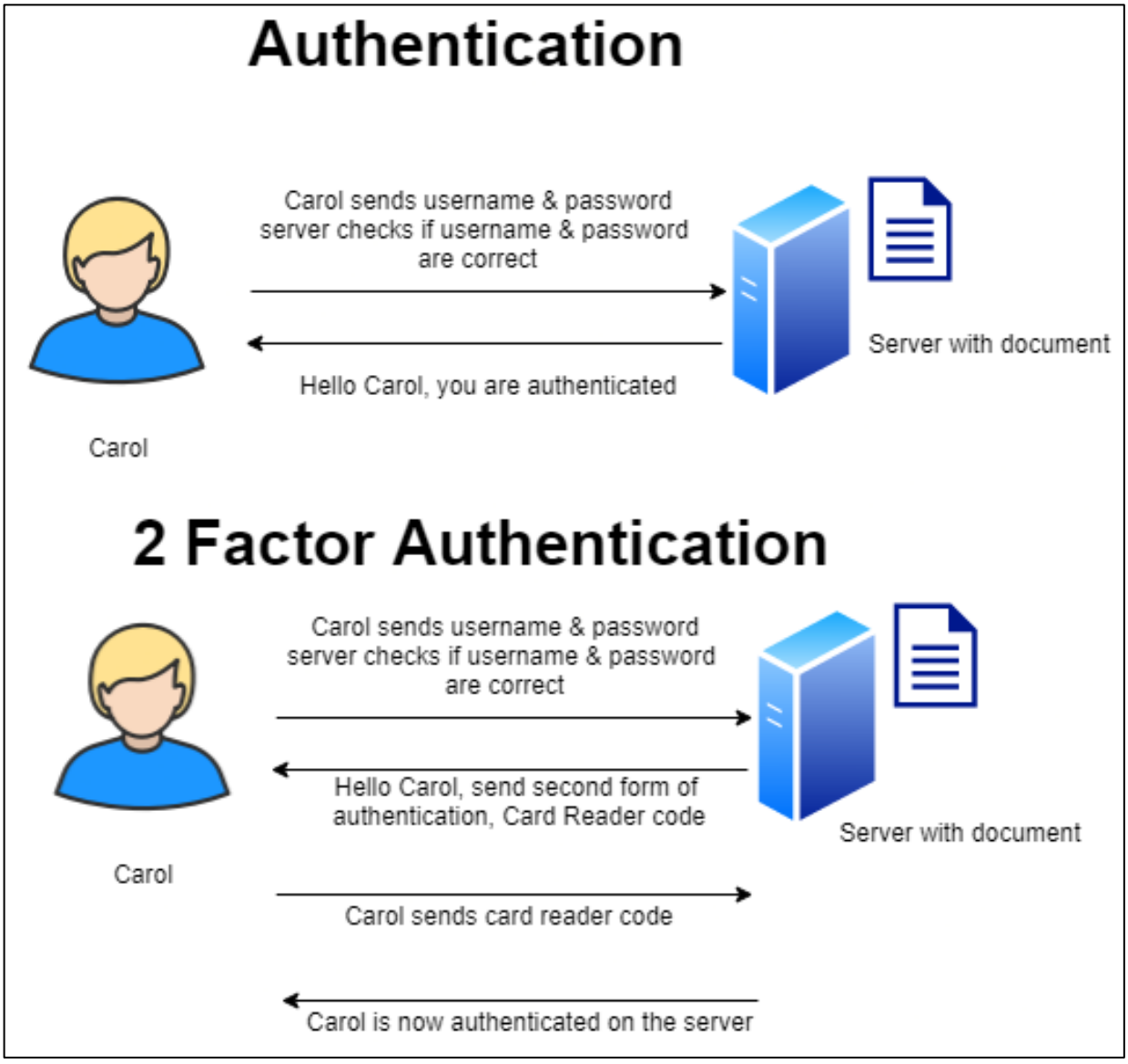

Рисунок 1.1 – Ілюстрація однофакторної та багатофакторної автентифікації

Отже, автентифікація є першим етапом для отримання користувачеві доступу. Найчастіше вебзастосунки використовують перший метод: надання імені користувача та паролю, бо він простіший – потрібно знати лише два значення, та через це зручніший – для отримання доступу не потрібно мати

сертифікати або надавати додаткові значення для підтвердження користувача. Як тільки застосунок підтверджує вірність наданих даних користувачем, починається етап авторизації.

Авторизація, в свою чергу є процесом визначення того, чи має користувач або система відповідний рівень доступу до певного ресурсу, наприклад, комп'ютерної системи, мережі, програми або даних. Іншими словами, авторизація – це процес надання або заборони доступу до ресурсу на основі ідентифікації користувача та його дозволів.

Авторизація є важливим аспектом інформаційної безпеки і використовується для забезпечення дотримання політик контролю доступу. Політика контролю доступу - це правила, які визначають, хто може отримати доступ до яких ресурсів і за яких умов. Авторизація гарантує, що тільки авторизовані користувачі можуть отримати доступ до певних ресурсів, і що їм надається відповідний рівень доступу на основі їхньої ролі, дозволів та інших факторів.

Авторизація може бути реалізована за допомогою різних механізмів, таких як (рис. 1.2):

– списки контролю доступу (Access control lists, ACL): Цей метод передбачає асоціювання кожного ресурсу зі списком користувачів або груп, які мають право доступу до нього, разом з їхнім рівнем доступу;

– контроль доступу на основі ролей (Role-based access control, RBAC): Цей метод передбачає призначення користувачів на ролі, а потім призначення дозволів для цих ролей. Користувачам надається доступ на основі їхньої ролі, а не індивідуальної ідентичності;

– контроль доступу на основі атрибутів (Attribute-based access control, ABAC): Цей метод передбачає надання доступу на основі різних атрибутів, таких як посада користувача, місцезнаходження або відділ.

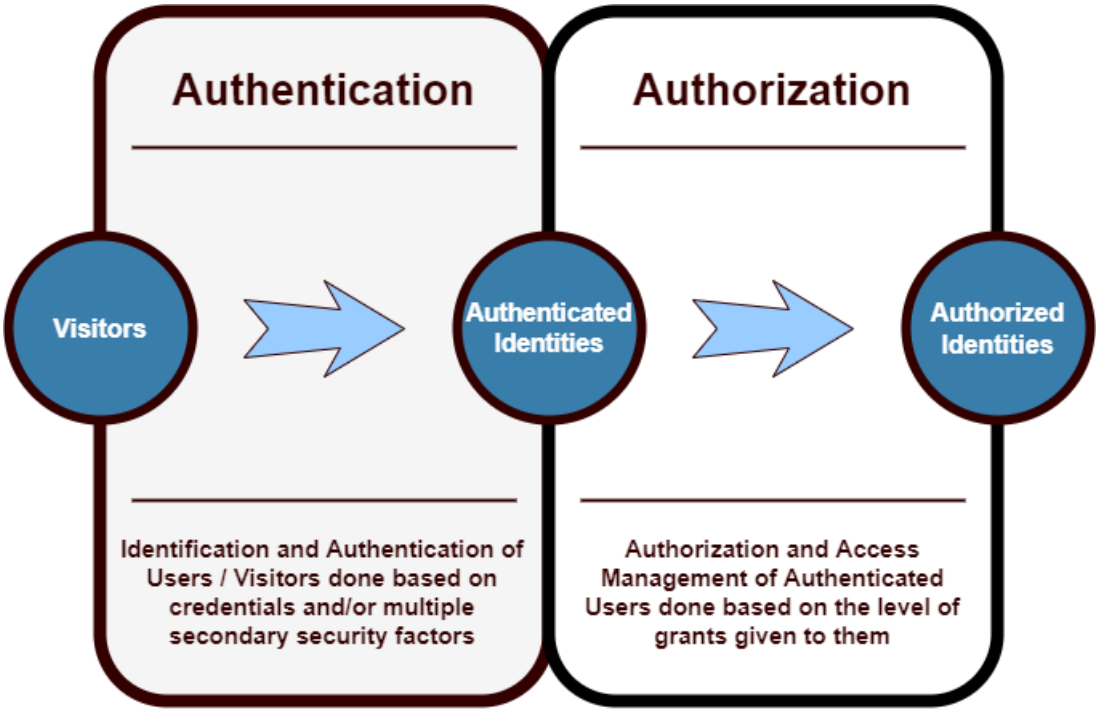

Рисунок 1.2 – Визначення ролі користувача за допомогою автентифікації та авторизації

Після цього, застосунок надає доступ до тих даних, до котрих роль користувача має доступ. У вебзастосунках користувач отримає файл cookie.

Файл cookie – це невеликий фрагмент даних, який сервер надсилає веббраузеру користувача (рис. 1.3). Браузер може зберігати файл cookie і надсилати його назад на той самий сервер при наступних запитах [6].

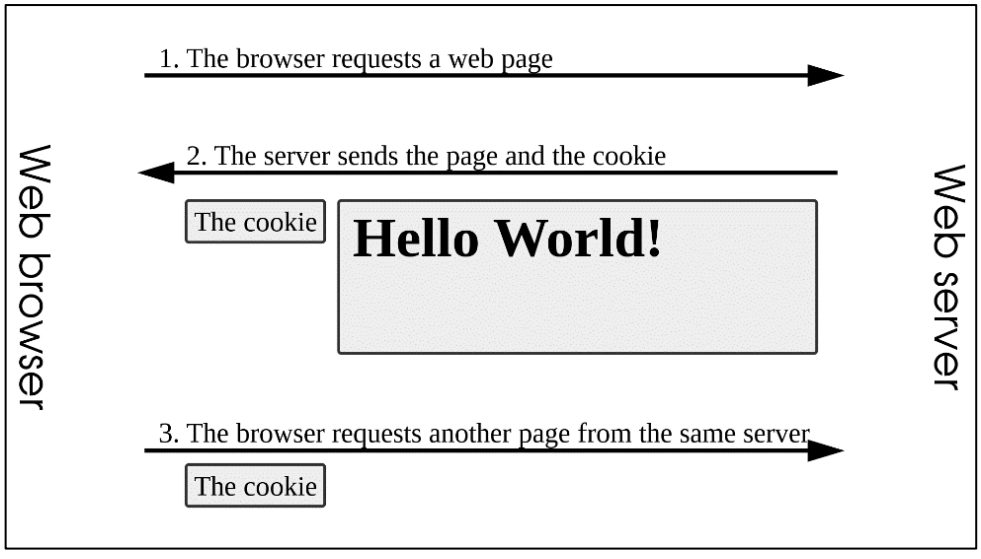

Рисунок 1.3 – Ілюстрація надання файлу cookie веббраузеру користувача

Файли cookie можна розділити на дві основні групи: сесійні та постійні.

Сесійні файли cookie є тимчасовими файлами котрі створюються коли користувач відвідує вебзастосунок, і видаляються, коли користувач закриває браузер або виходить зі свого акаунту, тобто завершає сеанс. Сесійні файли cookie використовуються для збереження інформації про стан між вебсервером і браузером користувача під час одного сеансу перегляду і, як правило, використовуються для автентифікації користувача та запам'ятовування його налаштувань.

Постійні файли cookie зберігаються на пристрої користувача протягом більш тривалого періоду часу, зазвичай кілька тижнів або місяців. Їх використовують для запам'ятовування налаштувань і параметрів користувача протягом декількох сеансів, і часто використовуються для персоналізованої реклами та відстеження.

Різниця між ними є лише у конфігурації. І від цієї конфігурації залежить конфіденційність даних користувача, котра є першим завданням інформаційної системи для охорони здоров’я.

1.2 Аналіз будови та призначення медичної інформаційної системи для сфери охорони здоров’я

Впровадження електронної системи охорони здоров'я Ehealth в системі охорони здоров'я України призвело до значної трансформації способу надання медичних послуг. Ehealth – це електронна система охорони здоров'я, яка здійснила революцію в галузі охорони здоров'я, забезпечивши платформу для ефективного управління даними про пацієнтів (рис. 1.4). Система дозволяє медичним працівникам отримувати доступ до інформації про пацієнта з будь-якого місця і в будь-який час, підвищуючи якість надання медичної допомоги пацієнтам. Зазначена електронна система Ehealth має двокомпонентну архітектуру, в якій користувач через відповідні медичні інформаційні системи (МІС) взаємодіє з центральною базою даних (ЦБД) [7].

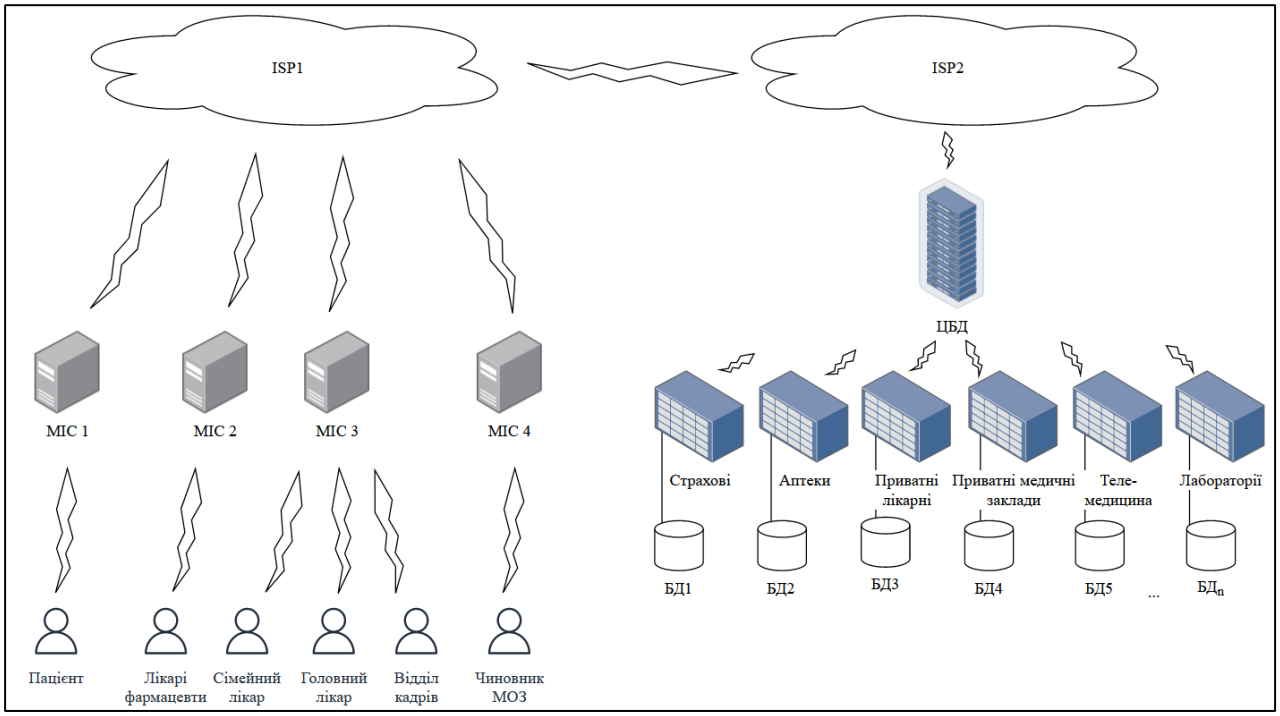

Рисунок 1.4 – Архітектура системи Ehealth

ЦБД є інформаційно-телекомунікаційною системою, яка містить передбачені законодавством реєстри, програмні модулі, інформаційну систему НСЗУ, в частині, необхідній для реалізації державних фінансових гарантій та інше [7].

МІС, в свою чергу також є інформаційно-телекомунікаційною системою, котра дає змогу автоматизувати роботу суб’єктів господарювання у сфері охорони здоров’я, створювати, переглядати, обмінюватися інформацією в електронній формі, зокрема з центральною базою даних (у разі підключення) [7].

Медичні інформаційні системи розрізняються за наступними класифікаціями [8]:

а) процес збору та обробки інформації:

1) автоматизовані – частина операцій по обробці та збірці інформації виконується людиною;

2) автоматичні – людина не приймає жодної участі у процесах;

б) тип інформаційної бази МІС:

1) системи, котрі оперують даними;

2) системи, котрі оперують знаннями. Ці системи є експертними, бо базуються на знаннях експертів, а результати роботи схожі на аналітичні дії експертів;

в) тип вирішуваних завдань:

1) інформаційно-довідкові – системи автоматизованого пошуку, вимірювальні системи;

2) інформаційно-логічні – діагностичні системи; системи прогнозу; системи моніторингу;

3) автоматизовані системи управління.

Інформаційно-логічна система (ІЛС) перетворює інформацію для отримання нової, наявної в інформаційному масиві.

В автоматизованих системах управління (АСУ) вирішується та приймається управлінські рішення.

Інформаційно-пошукові системи (ІПС) поділяються на фактографічні та документальні:

– фактографічні системи відповідають за масиви фактичних даних. Друковані довідники, каталоги, технічні паспорти є аналогами таких систем. Однак в ІПС фактичні дані зберігаються в базах даних (БД) і являють собою таблиці;

– документальні системи займаються з інформацією у вигляді документів. Як приклад, документальна ІПС може бути різні картотеки (бібліографічні, історії хвороби тощо). Під час пошуку, документальна система видає або номера документів, або список заголовків, або адреси зберігання заданих документів. Оцінку котра була знайдені в шуканих документів робить саме користувач.

Керуючі системи збирають інформацію про об'єкт управління, обробляють дані, передають проаналізовану інформацію в орган управління та формують керуюче рішення.

МІС можна також класифікувати за ієрархічним принципом, що відповідає багаторівневій структурі охорони здоров'я, і зазвичай розподіляють за чотирма рівнями:

1) базовий (або клінічний) рівень (лікарі різного профілю);

2) рівень лікувально-профілактичного закладу (поліклініка, стаціонар, диспансер, швидка допомога тощо);

3) територіальний рівень (профільні і спеціалізовані медичні служби і регіональні органи керування);

4) державний рівень (державні заклади та органи управління).

В межах кожного рівня класифікація МІС здійснюється за функціональним принципом, тобто відповідно до цілей і задач, що розв'язуються системою.

Функціонал МІС можна поділити на дві групи. Функціонал для установ Первинної медичної допомоги (ПМД) та Спеціалізованої медичної допомоги (СМД). До першої установи входить:

1) адміністративний модуль первинної медичної допомоги:
   - реєстрація постачальника медичних послуг, підрозділів, користувачів;
   - висновок капітаціонних договорів;
2) робоче місце лікаря ПМД:
   - робота з деклараціями, висновки декларацій;
   - електронні медичні записи;
   - імунізація;
   - виписка електронного рецепта «Доступні ліки»;
   - електронне направлення (ЕН);
3) електронний рецепт;
4) медичний висновок про тимчасову непрацездатність;
5) доступ до даних ПМД:
   - доступ до даних, які містять елементи обмежувальних груп;

– доступ до даних, які містять елементи заборонених станів (сервісів).

До функціоналу установ СМД входить:

1) функціонал обліку наданих послуг:
   – амбулаторія;
   – стаціонар;
2) адміністративний модуль вторинної медичної допомоги;
3) робоче місце лікаря СМД:
   – електронні медичні записи (ЕМЗ);
   – імунізація;
   – діагностичні звіти;
   – ЕМЗ і ЕН для неідентифікованих пацієнтів;
   – виписування електронних направлень;
   – обробка і погашення електронних направлень;
   – ЕМЗ стаціонар: надходження;
   – ЕМЗ стаціонар: виписка;
4) робота з записами про ідентифікованих пацієнтів;
5) робота з записами про неідентифікованих пацієнтів;
6) приєднання записів неідентифікованого пацієнта до ідентифікованого;
7) доступ до даних:
   – доступ до даних, що містять елементи обмежувальних груп;
   – доступ до даних, що містять елементи заборонених станів (сервісів);
8) медичні висновки:
   – про народження дитини;
   – про тимчасову непрацездатність;
9) виписування електронного рецепту;
10) план лікування.

Описаний функціонал та доступ до даних є найважливішими компонентами МІС. Як зображено на рис. 1.4, зовнішній доступ до мережі Інтернет має лише МІС, тому і є найбільш вразливим компонентом інформаційної системи. Для отримання доступу до МІС, користувач повинен знати лише адресу в Інтернет мережі. Після отримання адреси, лише механізми автентифікації та авторизації захищають дані від не автентифікованого користувача, або атакуючого.

Слід зазначити що на теперішній час в Україні до ЦБД вже підключено більше 35 МІС, і багато з них (23 з 36) мають доступ до даних [9]. І так як програмний код, як наслідок, і механізми автентифікації та авторизації створенні по різному а також використовують різні функції.

Різниця між механізмами авторизації та автентифікації МІС, потенційно надають можливість знаходження найслабкіших функцій в певних МІС. Таким чином, проведення дослідження вразливостей та слабких місць у подібних механізмів і надалі створення більш захищених механізмів автентифікації та авторизації є найважливішою задачею.

Висновки до розділу 1

У першому розділі був проведений аналіз механізмів автентифікації та авторизації. З’ясовано, як саме відбувається ці процеси, їх існуючи типи та їх важливість у сучасному світі.

Також, було проаналізовано будову інформаційної системи у сфері охорони здоров’я, а саме системи Ehealth. Було визначено частину системи, котра є найбільш вразливою, через свою відкритість до мережі Інтернет. Виявлено, що ця частина є медичною інформаційною системо, котра має обширний функціонал, який залежить від її типу. Також, багато з вже підключених МІС мають повний доступ до важливих даних, що робить їх ще більш критичною інформаційною системою у інфраструктурі охорони здоров’я.

Було визначено роль, функції та важливість різних типів МІС у інформаційній системі у сфері охорони здоров’я.

На основі проведеного аналізу було встановлено який саме механізм треба досконально розглянути, провести аудити безпеки та впровадити міри для захисту цих механізмів. Надалі ця інформація допоможе в розробці більш захищених інформаційних систем.

2 ВИЗНАЧЕННЯ СЛАБКИХ ЕЛЕМЕНТІВ ТА ВРАЗЛИВОСТЕЙ В МЕХАНІЗМАХ АВТЕНТИФІКАЦІЇ І АВТОРИЗАЦІЇ

2.1 Специфікації та методології на основі яких будуть розроблені рекомендації

У сучасному цифровому світі, де конфіденційність та безпека інформації є критичними аспектами, механізми автентифікації та авторизації відіграють важливу роль у захисті доступу до систем та даних, особливо у сфері охорони здоров’я. Проте, незважаючи на їхню важливість, ці механізми часто стають об'єктом атак, що призводить до порушення конфіденційності, цілісності та доступності вебзастосунків та їхньої інформації.

В цьому розділі буде виявлено типові вразливості у механізмах автентифікації (на основі паролю) та авторизації, і запропоновано їхнє вирішення. Рекомендації по усуненню вразливостей буде розроблено згідно специфікацій NIST та методології OWASP:

а) NIST (National Institute of Standards and Technology) є агентством Сполучених Штатів Америки, яке займається стандартизацією, вимірюваннями та технологічним дослідженням. Воно має широкий спектр обов'язків, серед яких розробка стандартів, забезпечення точних вимірювань та дослідження нових технологій. Одним з важливих документів, розроблених NIST, є специфікація NIST SP 800-63. Ця специфікація, що має назву "Digital Identity Guidelines" (Директиви щодо цифрових ідентифікаторів), надає рекомендації щодо впровадження та керування цифровими ідентифікаторами та аутентифікацією. SP 800-63 визначає основні принципи та вимоги для захисту цифрових ідентифікаторів, які використовуються для ідентифікації та автентифікації користувачів в різних інформаційних системах. Вона містить рекомендації стосовно використання паролів, багаторівневої аутентифікації, методів ідентифікації та керування цифровими ідентифікаторами. Специфікація NIST SP 800-63 складається з трьох частин: SP 800-63A, SP 800-63B та SP 800-

63С. Кожна з цих частин спрямована на різні аспекти цифрових ідентифікаторів та аутентифікації:

1) NIST SP 800-63A (Digital Identity Guidelines: Enrollment and Identity Proofing) встановлює рекомендації для процедур реєстрації та підтвердження особи при видачі цифрових ідентифікаторів;

2) NIST SP 800-63B (Digital Identity Guidelines: Authentication and Lifecycle Management) надає вказівки щодо аутентифікації користувачів та управління цифровими ідентифікаторами;

3) NIST SP 800-63C (Digital Identity Guidelines: Federation and Assertions) містить рекомендації для створення і використання цифрових ідентифікаторів у федеративних системах, де автентифікація здійснюється через довірені сторони;

б) OWASP (Open Web Application Security Project) є незалежною організацією, котра зосереджується на покращенні безпеки вебзастосунків та програмного забезпечення загалом. Метою OWASP є популяризація усвідомлення про безпеку програмного забезпечення та надання ресурсів, інструментів і рекомендацій для побудови безпечних програмних застосунків. OWASP розробляє різні матеріали та проєкти, які допомагають виявляти, виправляти та запобігати вразливостям у вебзастосунках. Одними з їх найвідоміших рекомендацій є:

1) OWASP Application Security Verification Standard: Цей стандарт містить детальний перелік контролів безпеки, які слід перевіряти під час розробки та тестування вебзастосунків. Він надає рекомендації щодо різних аспектів безпеки, включаючи аутентифікацію, авторизацію, перевірку вводу, захист від XSS та CSRF, криптографічні операції та інші;

2) OWASP Secure Coding Practices – Quick Reference Guide: Цей документ містить короткий перелік рекомендацій щодо безпечного програмування. Він надає загальні поради та кращі практики щодо використання безпечних функцій, обробки введення, обробки помилок,

керування пам'яттю та інших аспектів, що допомагають уникати вразливостей;

3) OWASP Cheat Sheet Series – це набір коротких і конкретних документів, що містять практичні поради та рекомендації щодо різних аспектів безпеки програмного забезпечення. Цей набір створений OWASP з метою надати корисну інформацію розробникам, тестувальникам та аудиторам безпеки, які шукають швидкий та зрозумілий довідник по різноманітним темам безпеки.

2.2 Вразливості механізму автентифікації

Неавтентифікований доступ. Ця вразливість означає можливість отримання доступу до системи, застосунку або ресурсу без надання будь-яких форм ідентифікації чи автентифікації. Тобто, користувач може взаємодіяти з вебзастосунком, не надаючи облікових даних, такі як ім'я користувача і пароль, або будь-який інший вид ідентифікуючого токену.

Відсутність перевірки автентифікації, навіть на певній сторінці несе високі ризики вебзастосунку, оскільки дозволяє будь-кому потенційно отримати НСД до конфіденційної інформації або виконувати дії у вебсистемі.

Якщо взяти для прикладу ІС в сфері охорони здоров’я, то подібна вразливість надає можливості зловмиснику отримати доступ до повної інформації користувачів чи певного функціоналу адміністрування вебзастосунків, модулів або систем без будь-яких перешкод. Без належної аутентифікації неможливо перевірити ідентичність або наміри користувача, що ускладнює застосування контролю доступу і захист від зловмисних дій.

Вирішенням є перевірка аутентифікації при кожному запиті до будь-яких не публічних сторінок, документів, плагінів тощо.

Недосконалий захист від brute-force атак. Brute-force або атака грубої сили полягає в систематичному переборі всіх можливих комбінацій імен користувачів і паролів з метою отримання НСД до ІС або/та облікового запису.

Зловмисники використовують автоматизовані інструменти або скрипти, для систематичної генерації та перебору різних комбінацій, поки не буде знайдено правильні облікові дані [10].

Зазвичай, якщо зловмисники не мають список користувачів вебзастосунку, атака тоді поділяється на два етапи. Перший етап визначає першу частину облікових даних – імена користувачів.

Процес підбору імені користувачів полягає в систематичному вгадуванні дійсних значень з подальшою метою отримання НСД до ІС або облікового запису. Впізнавані шаблони, такі як адреси електронної пошти або загальновживані імена користувачів, такі як «admin», «alex», «anton» тощо, роблять імена користувачів особливо вразливими для підбору. Особливо, якщо вебзастосунок має не правильні конфігурації, та надає корисну інформацію під час обробки запитів. Це може бути:

а) помилка, котра надає чутливу інформацію о статусі наданого імені користувача (рис. 2.1);

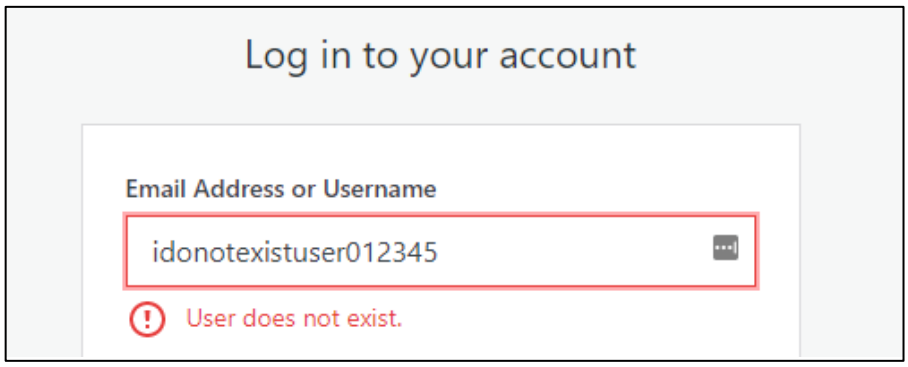

Рисунок 2.1 – Вебзастосунок інформує про неіснуюче ім’я користувача

б) аналіз використаного часу вебзастосунком для обробки запиту (рис. 2.2);

Payloads	Code	Reason	RTT	Size Resp. Header
americas	200	OK	751 ms	129 bytes
analyzer	200	OK	497 ms	129 bytes
anaheim	200	OK	470 ms	129 bytes
autodiscover	200	OK	447 ms	129 bytes
announcements	200	OK	424 ms	129 bytes

Рисунок 2.2 – Вебзастосунок обробив запит з іменем користувача «americas» довше ніж інші

в) інші різниці між відповідями: коди помилок або/та хедери HTTP, змінені HTML сторінки тощо (рис. 2.3).

Payloads	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
asterix	200	OK	315 ms	129 bytes	2,888 bytes
root	200	OK	550 ms	129 bytes	2,886 bytes
guest	200	OK	547 ms	129 bytes	2,886 bytes
carlos	200	OK	542 ms	129 bytes	2,886 bytes
admin	200	OK	509 ms	129 bytes	2,886 bytes
test	200	OK	508 ms	129 bytes	2,886 bytes
auction	200	OK	364 ms	129 bytes	2,886 bytes
austin	200	OK	363 ms	129 bytes	2,886 bytes
auth	200	OK	362 ms	129 bytes	2,886 bytes

Рисунок 2.3 – Відповідь вебзастосунку на запит автентифікації з іменем користувача «americas» відрізняється на 2 байти

Другий етап настає після формування списку існуючих користувачів або визначення одного імені користувача. Результатом другого етапу буде знаходження другої частини облікового запису – паролю (рис. 2.4):

Payloads	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
joshua	302	Found	327 ms	199 bytes	0 bytes
000000	200	OK	345 ms	129 bytes	2,888 bytes
1111	200	OK	329 ms	129 bytes	2,888 bytes
111111	200	OK	339 ms	129 bytes	2,888 bytes
11111111	200	OK	323 ms	129 bytes	2,888 bytes
112233	200	OK	316 ms	129 bytes	2,888 bytes
121212	200	OK	360 ms	129 bytes	2,888 bytes
123123	200	OK	338 ms	129 bytes	2,888 bytes

Рисунок 2.4 – Наданий пароль «joshua» з переліку є правильним

Результатом другого етапу буде отримання доступу до переліченого акаунту або акаунтів та завершення brute-force атаки. Як приклад, відповідь вебзастосунку проілюстровано на рис. 2.5, надані дані облікового запису є правильними та вебзастосунок присвоює сесійний ідентифікатор:

```
HTTP/1.1 302 Found
Location: /my-account
Set-Cookie: session=38BbLUo4rTfs7Z0HDNp1E6wTDKHmXJgv; Secure;
HttpOnly; SameSite=None
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 0
```

Рисунок 2.5 – Відповідь вебзастосунку та отримання доступу до акаунту

Рекомендаціями для покращення механізму автентифікації від атак перелічення є [11-13]:

а) запровадження механізмів CAPTCHA: кожен запит користувача повинен містити відповідь на CAPTCHA вправу. Впроваджений механізм допоможе відлічати автоматичні та мануальні запити;

б) блокування облікового запису на певний час: якщо користувач, який намагається отримати доступ до облікового запису, перевищує певний поріг невдалих спроб входу, вебзастосунок буде тимчасово блокувати вхід в обліковий запис, щоб запобігти подальшому НСД;

в) блокування акаунта зі зростаючою затримкою: Після певної кількості невдалих спроб входу, встановлення затримки на наступну спробу авторизації. Збільшуйте час очікування для кожної невдалої спроби, щоб запобігти атакам грубої сили. Наприклад, спочатку затримка буде встановлена в 30 секунд, та надалі збільшена до години, коли обліковий запис наближається до максимальної кількості невдалих спроб підряд;

г) білі списки IP-адрес: обробка запитів на автентифікацію лише з білого списку довірених IP-адрес, з яких користувач успішно пройшов автентифікацію раніше або був зареєстрований. Це допоможе запобігти НСД з невідомих або підозрілих джерел;

д) чорні списки IP-адрес: реалізований механізм чорних списків IP-адрес користувачів, буде фіксувати та вносити IP-адресу зловмисника до чорного списку та повністю забороняти доступ;

е) фільтрація відповідей: вебзастосунок повинен реалізовувати фільтрацію відповідей під час процесу автентифікації. Важливо переконатися, що відповідь вебзастосунку не виявляє жодних відмінних характеристик або не надає диференційованої інформації на основі дійсності наданих облікових даних. Відповідь, включаючи тіло, заголовки та час відповіді, повинна залишатися послідовною незалежно від того, успішною була спроба автентифікації чи ні.

Слабкі механізми фільтрації та валідації даних. При розробці захисту від атак перелічення та самих механізмів автентифікації треба приділяти особливу увагу фільтрації та валідації даних. В противному випадку зловмисник може отримати НСД різними шляхами (рис. 2.6).

```
POST /login HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: [redacted]
Content-Type: text/plain;charset=UTF-8
Origin: [redacted]
Content-Length: 47
Connection: close
Cookie: session=DY03GrAlyw2NvrocqnI5CCG8mRiz9mR4

{
  "username":"carlos",
  "password":[
    "123456",
    "password",
    "12345678",
    "qwerty",
    "123456789",
    "12345",
    "1234",
    "111111"
  ]
}
```

Рисунок 2.6 – Запит в тілі якого вставлений список можливих паролів

Через те, що зловмисник відправив один запит, вебзастосунок, якщо список не містить правильного паролю, зарахує це як одну неправильну спробу входу. Якщо ж правильний пароль присутній в списку – вебзастосунок надасть ідентифікатор сеансу (рис. 2.7).

Рисунок 2.7 – Вебзастосунок знайшов правильний пароль та надав ідентифікатор сеансу зловмиснику

Окрім слабких механізмів захисту від перелічення, вебзастосунок може мати інші вразливості в механізмі автентифікації:

а) ін’єкції в SQL-запити (рис. 2.8) [14];

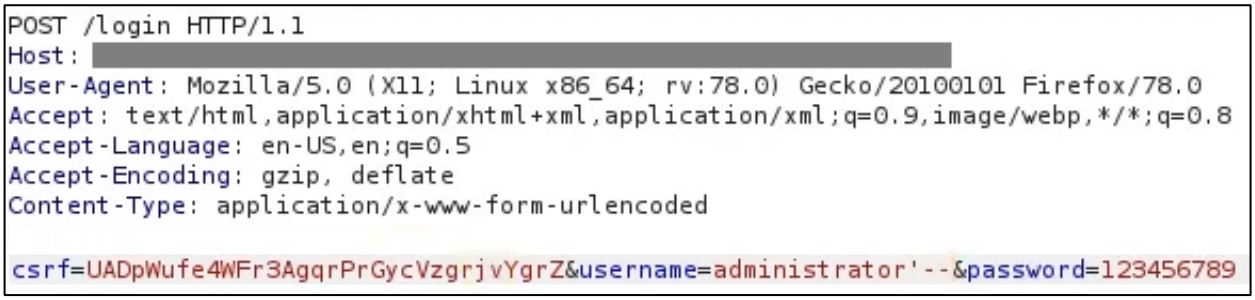

Рисунок 2.8 – Значення «‘--» дозволить обійти перевірку паролю шляхом коментування SQL-запиту

б) ін’єкції в NoSQL-запити (рис. 2.9) [15];

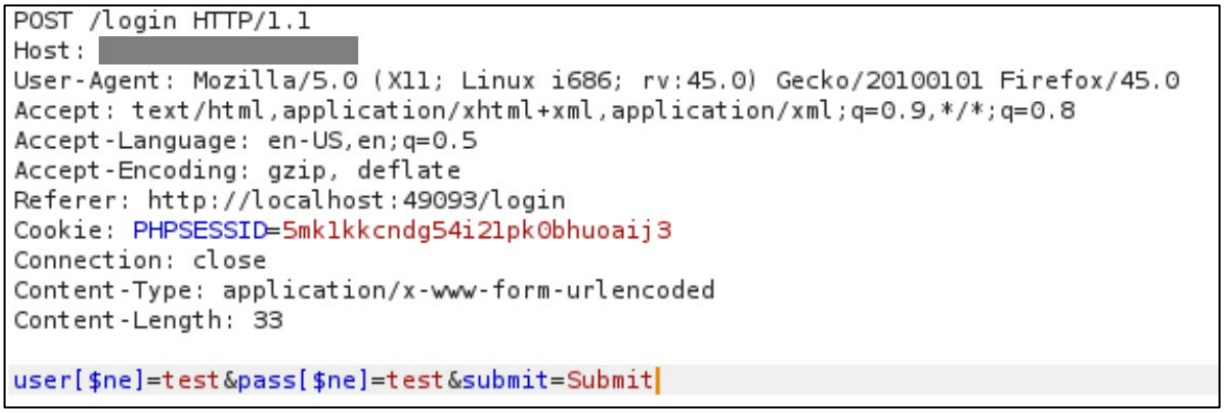

Рисунок 2.9 – Доданий оператор *$ne* до параметрів *user* та *pass* дозволить отримати доступ шляхом порушення запиту до бази даних

в) та багато інших методів, в залежності від вебзастосунків. Як приклад, можливість отримання доступу до вебпанелі керування відеокамер Hikvision, шляхом надання статичного коду автентифікації в посиланні (рис. 2.10) [16; 17].

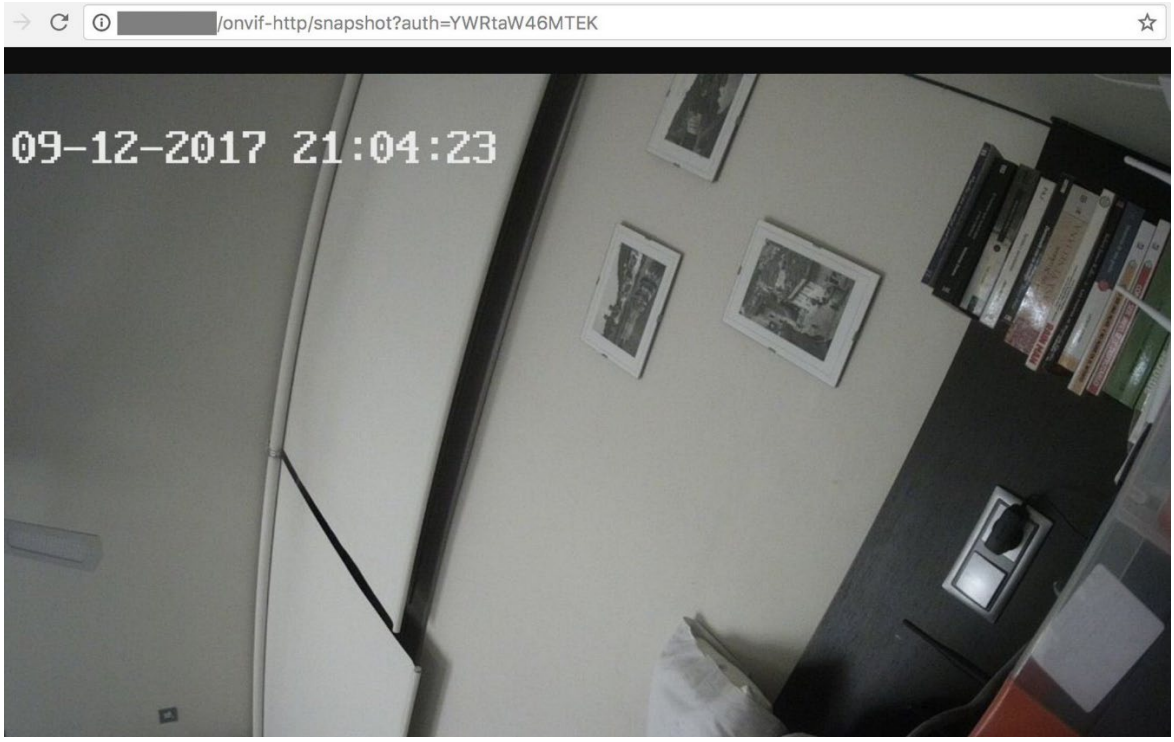

Рисунок 2.10 – Отримання доступу до вебкамери шляхом надання статичного коду автентифікації

2.3 Вразливості механізму авторизації

Механізми авторизації або механізми контролю доступу можна поділити на кілька категорій залежно від їхнього характеру і реалізації.

Вертикальний контроль доступу встановлює ієрархію доступу на основі рівнів привілеїв. Кожен користувач має визначений рівень доступу, який обмежує його здатність переглядати або змінювати певні ресурси. Цей механізм гарантує, що низько рівневі користувачі не можуть отримати доступ до конфіденційної інформації або функцій, які призначені для вищих рівнів привілеїв.

Горизонтальні елементи управління доступом встановлюють права доступу на основі ролей. У цьому підході користувачі групуються в ролі з певними привілеями, і кожна роль має доступ до певних ресурсів. Наприклад, адміністратор має повний доступ до всіх функцій, тоді як звичайні користувачі можуть мати обмежений доступ до деяких функцій.

Контекстно-залежні елементи керування доступом враховують додаткові фактори, такі як час, місцезнаходження, тип пристрою або стан користувача. Застосовуються політики, які змінюються залежно від контексту, що дозволяє динамічно контролювати доступ до ресурсів в залежності від поточних умов і обставин (рис. 2.11).

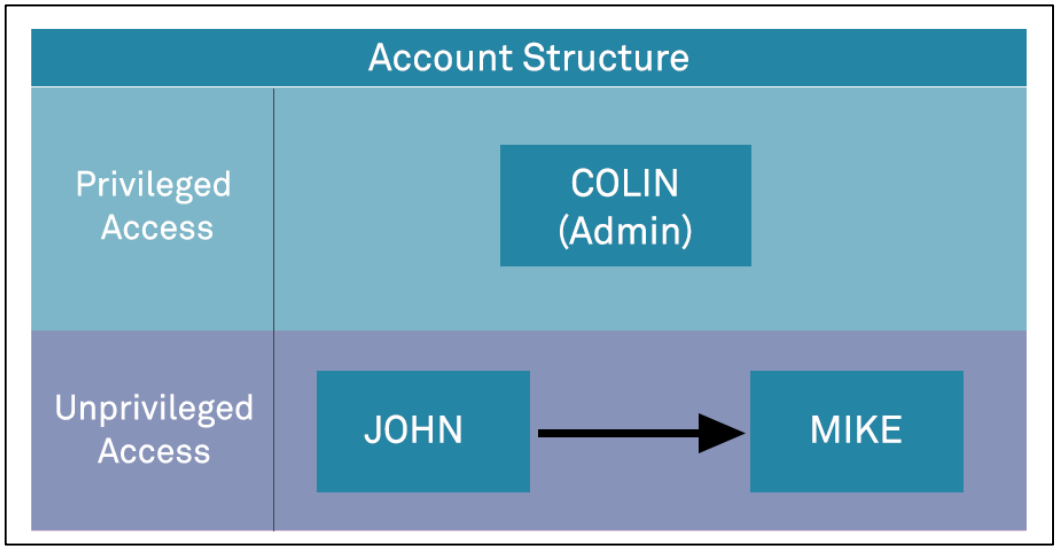

Рисунок 2.11 – Приклад горизонтального та вертикального контролю доступу

Якщо вебзастосунок має неправильні конфігурації контролю доступів або контроль взагалі відсутній, користувач може отримати НСД до інформації та/або функціоналу. Такі дії мають назву ескалацією привілеїв. Різниця між горизонтальною та вертикальною ескалацією привілеїв, є лише в критичності отриманого НСД.

Прикладом горизонтальної ескалації є отримання доступу до карток інших пацієнтів, тобто користувачів того ж рівня, шляхом перелічення значень (рис. 2.12). Червоні пунктирні лінії позначають НСД до карток:

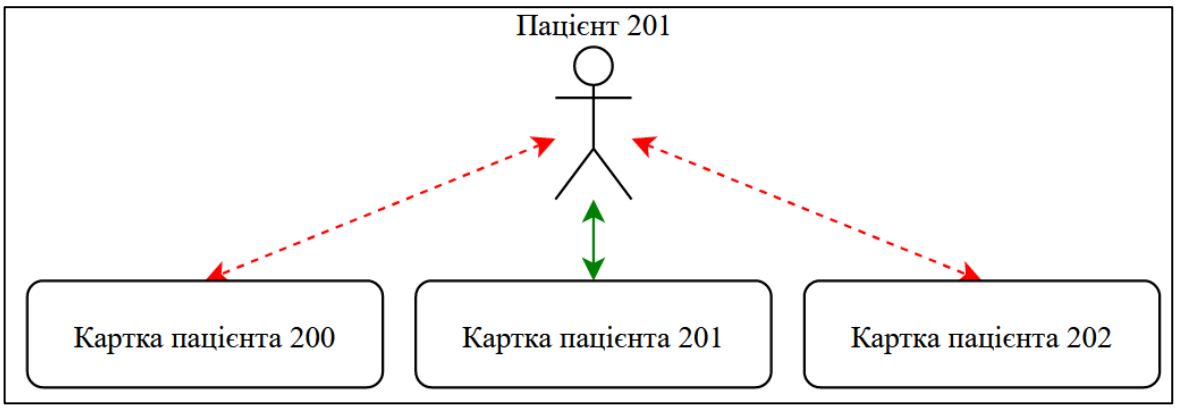

Рисунок 2.12 – Приклад горизонтальної ескалації прав

Вертикальна ескалація привілеїв є більш критичною, адже зловмисник може отримати НСД до функцій, які призначені для вищих рівнів привілеїв (рис. 2.13). Червоні пунктирні лінії позначають НСД до функціоналу:

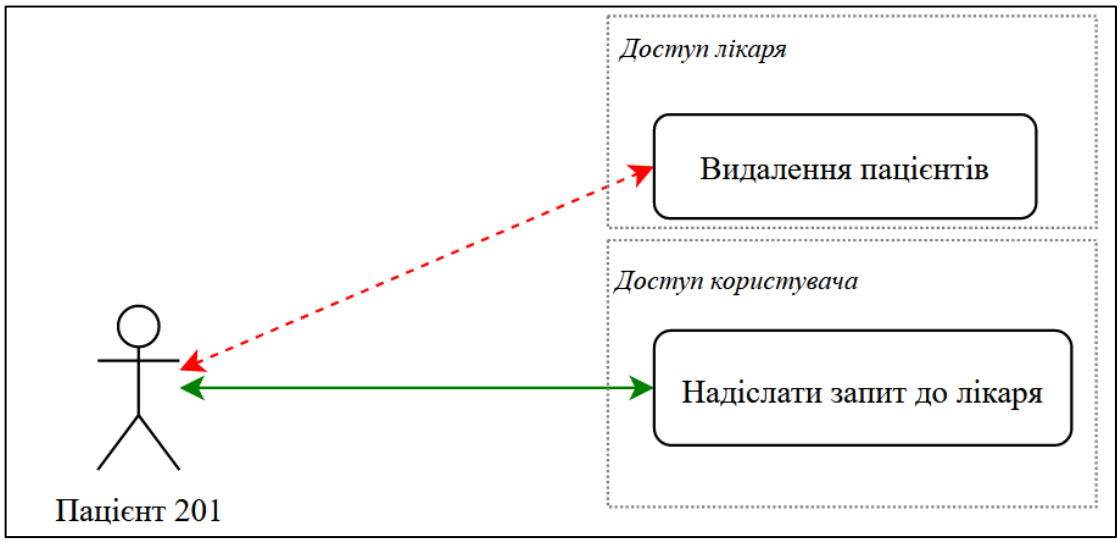

Рисунок 2.13 – Приклад вертикальної ескалації прав

Рекомендації по усуненню цих вразливостей можна сформулювати наступним чином [18]:

а) ніколи не слід покладатися виключно на обфускацію як на засіб для контролю доступу;

б) доступ до ресурсів має бути заборонений за замовчуванням, якщо тільки вони явно не призначені для публічного доступу;

в) слід застосовування уніфікованого механізму контролю доступу для всього вебзастосунку, де є така можливість;

г) на рівні коду розробники повинні бути зобов'язані декларувати дозволений доступ до кожного ресурсу, забороняючи доступ за замовчуванням;

д) проведення комплексного аудиту та ретельного тестування засобів контролю доступу для перевірки їхньої відповідності запланованій функціональності.

2.4 Вразливості механізму управління сеансами

Сеанс користувача починається, коли входить в систему, і закінчується, при виході з системи або через бездіяльність. Керування сеансами має вирішальне значення для підтримки безпеки та цілісності сеансів користувачів і запобігання НСД.

Однак проблеми з керуванням сеансами можуть призвести до вразливостей, які можливо використати для отримання НСД.

Необмежений термін дії сеансу – вразливість, котру зловмисник, який отримав доступ до ідентифікатору сеансу користувача, може використати для отримання доступу навіть після того, як користувач вийшов з системи або закрив браузер. Сеанси повинні мати визначений термін дії, щоб гарантувати, що неактивні сеанси будуть завершені через певний проміжок часу.

Небезпечні ідентифікатори сеансу – атака на основі передбачення значень ідентифікатора сеансу, який надалі дозволить зловмиснику обійти схему автентифікації вебзастосунку.

Проаналізувавши та зрозумівши процес генерації токена сеансу, зловмисник може передбачити дійсне значення ідентифікатора сеансу та отримати НСД (рис. 2.14).

```
Set-Cookie: PHPSESSID=33342d61646d696e; path=/; HttpOnly
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
```

Рисунок 2.14 – Вебзастосунок надав ідентифікатор сеансу

На першому кроці зловмисникові потрібно зібрати кілька дійсних значень ідентифікатора сеансу, які використовуються для ідентифікації автентифікованих користувачів.

Після того, як зловмисник отримав ці значення, він повинен зрозуміти структуру ідентифікатора та яка саме інформація використовується для його створення. Це можуть бути певні алгоритми, шифрування або хешування, які використовуються застосунком для захисту ідентифікатора сеансу. Знання про ці механізми може допомогти зловмиснику розшифрувати або розгадати його.

У деяких слабких реалізаціях використовуються ідентифікатори сеансів, які мають передбачувані або недостатньо випадкові значення. Наприклад, ідентифікатор може складатися з імені користувача, мітки часу або його IP-адреси, що робить ідентифікатор вразливим до атак методом перебору. Інші неправильні практики включають використання відкритого тексту або слабкого кодування, наприклад як base64.

Зловмисник може застосовувати метод грубої сили, спробуючи згенерувати та перевіряти різні значення ідентифікатора сеансу, доки не отримає доступ до застосунку. Якщо токени сеансу легко вгадуються або мають послідовний шаблон, зловмисники можуть успішно генерувати правильні токени і вдавати себе за легітимних користувачів. Це стає особливо проблематичним, коли токени сеансу є короткими або не мають достатньої випадковості, оскільки це робить їх більш піддатливими до атак.

На прикладі отриманого ідентифікатору (рис. 2.14), після перевірки ентропії згенерованих ідентифікаторів, діаграма демонструє, що механізм генерації є слабким. При генерації стійких ідентифікаторів, кожен біт має мати близько 100 % значення випадковості (рис. 2.15).

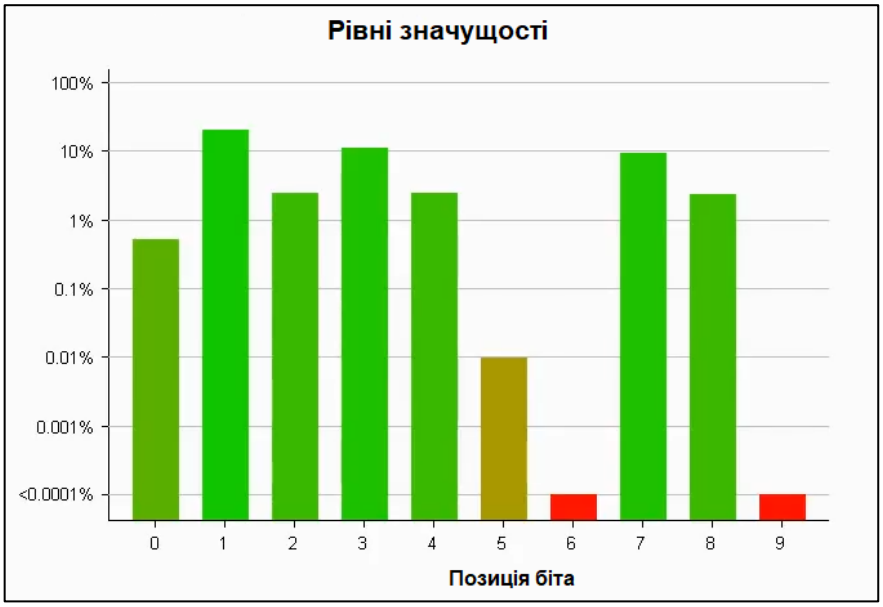

Рисунок 2.15 – Перевірка ентропії ідентифікаторів

Переконавшись, що ідентифікатор є слабким, відбувається спроба його декодування (рис. 2.16).

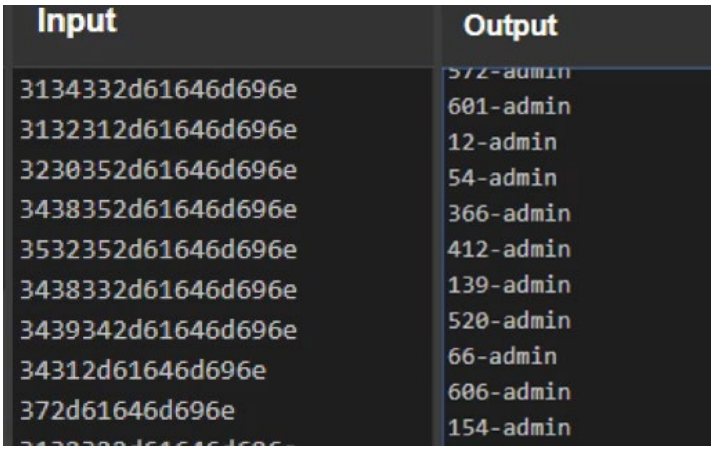

Рисунок 2.16 – Результат декодування ідентифікаторів з base64 та hex-кодування

Маркери сеансу, такі як ідентифікатори сеансу або файли cookie, повинні генеруватися з використанням надійних криптографічних алгоритмів і не повинні містити будь-якої передбачуваної або конфіденційної інформації. Якщо маркери сеансу передбачувані або їх можна легко вгадати, зловмисник може підробити або вгадати дійсні маркери, щоб видати себе за законного користувача.

Окрім генерації складних ідентифікаторів, вебзастосунок повинен додавати атрибути для сесійних файлів cookie, за для безпеки користувача [19]:

а) Secure: вказує на те, що файл cookie повинен передаватись лише через захищене з'єднання, таке як HTTPS. Це запобігає перехопленню cookie зловмисниками, оскільки вони не зможуть отримати доступ до нього під час передачі по незахищеному з'єднанню;

б) HttpOnly: його використання дозволяє обмежити доступ до cookie з боку JavaScript. Це забезпечує захист від XSS атак, де зловмисник викрадає файл cookie через скрипти, які впровадженні на вебсторінку;

в) SameSite: визначає, чи дозволено браузеру відправляти cookie в запитах, які не походять з того ж джерела (вебзастосунку). Це допомагає запобігти атакам типу "Cross-Site Request Forgery" (CSRF), коли зловмисник використовує сайт-жертву для виконання небажаних дій в ім'я користувача.

Описані атрибути мають бути встановлені на сесійних файлах cookie задля безпеки користувача (рис. 2.17).

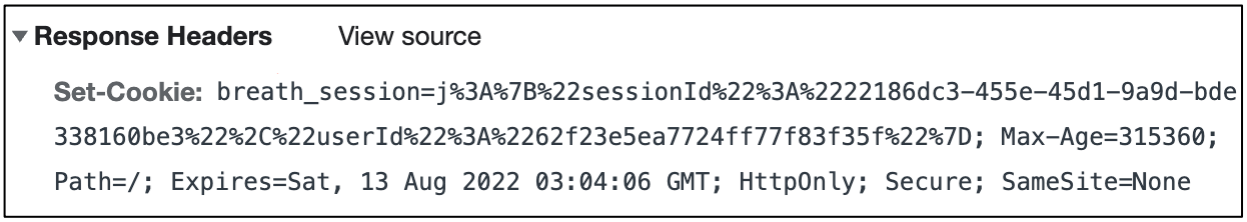

Рисунок 2.17 – Використання атрибутів при наданні сесійного файлу cookie

Висновки до розділу 2

Надані рекомендації по усуненню потенційних вразливостей в механізмах аутентифікації та авторизації є суттєвим фактором для проєктування нових та більш досконалих. Проілюстровані слабкі елементи при генеруванні ідентифікаторів демонструють необхідність використання більш складніших та надійніших механізмів генерації.

Слід зазначити, що архітектура процесів автентифікації та авторизації будуть будуватись одразу на зазначених рекомендацій.

3 МОДЕЛЮВАННЯ ТА ТЕХНІЧНЕ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Вибір механізмів автентифікації та авторизації

Ідентифікатор сеансу є найголовнішою частиною механізму автентифікації та авторизації. Розглянуті вразливості в другому розділі демонструють важливість обрання безпечних механізмів котрі будуть надавати ці ідентифікатори. Також, слід звернути увагу на те, що система Ehealth має велике навантаження та величезні бази даних, котрі постійно потребують додаткових ресурсів.

Для описаних потреб, можуть підійти наступні стандарти:

– JSON Web Tokens (JWT);
– Simple Web Tokens (SWT);
– Security Assertion Markup Language Tokens (SAML).

Однак, кількість збережених значень SAML (рис. 3.1) є більшою за значення JWT (рис. 3.2), а тому і довжина SAML є більшою:

Рисунок 3.1 – Закодований SAML

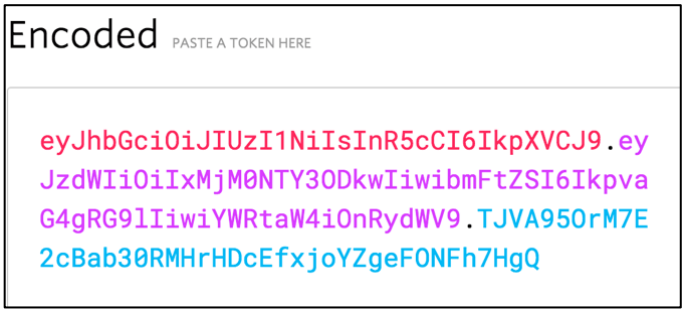

Рисунок 3.2 – Закодований JWT

З точки зору безпеки, SWT може бути симетрично підписаний лише спільним секретним ключем за допомогою алгоритму Hash-based Message Authentication Code (HMAC). Однак токени JWT і SAML можуть використовувати для підписання пару публічних/приватних ключів у вигляді сертифіката X.509.

Але треба зазначити, що безпечно підписати XML за допомогою XML Digital Signature без створення потенційних вразливостей в безпеці механізму дуже складно в порівнянні з простотою підписання JSON. Тобто процес підписання XML є складним і потенційно може призвести до появи вразливостей, таких як:

– XML External Entity (XXE) атака або атака зовнішніми сутностями XML: XML-документи можуть містити відносні або абсолютні посилання на зовнішні сутності, які можуть бути використані для отримання чутливої інформації з сервера або виконання віддаленого коду;

– expansion-атака або атака розширення: XML-документи можуть бути розширені за допомогою сутностей для створення великих обсягів даних, що може призвести до переповнення пам'яті та збоїв системи при обробці;

– signature wrapping attacks (атаки підміни підпису): під час цієї атаки, зловмисники можуть спробувати обійти механізми верифікації цифрового підпису або підмінити його, змінивши структуру XML-документа таким чином, щоб підпис залишився незмінним, але вміст документа був змінений, що призведе до прийняття недійсних або змінених документів як дійсних та порушити автентичність та цілісність.

Через просту структуру та відсутність вбудованих механізмів, JavaScript Object Notation (JSON) документ є більш вигідним варіантом.

Тому, було обрано для використання JWT, як ідентифікатор.

3.2 JSON Web Token

JSON Web Token (JWT) – це відкритий стандарт (RFC 7519), який визначає компактний і автономний спосіб безпечної передачі інформації між сторонами у вигляді об'єкта JSON. Ця інформація може бути перевірена і їй можна довіряти, оскільки вона підписана цифровим підписом. JWT можуть бути підписані за допомогою секретного ключа (за допомогою алгоритму HMAC) або пари публічних/приватних ключів за допомогою RSA (Rivest, Shamir та Adleman) або Elliptic Curve Digital Signature Algorithm (ECDSA).

Хоча JWT можуть бути зашифровані, щоб також забезпечити секретність між сторонами, більш безпечним буде використовувати підписані токени. Підписані токени дозволяють перевірити цілісність тверджень, що містяться в ньому, в той час як зашифровані токени приховують ці твердження від інших сторін. Коли токени підписуються за допомогою пар відкритих/закритих ключів, підпис також засвідчує, що тільки сторона, яка володіє закритим ключем, є тією, хто підписав токен.

JWT складається з трьох частин, котрі закодовані в base64 та відокремленні крапкою (рис. 3.2) [20]:

– заголовок: містить метадані про JWT, такі як: alg, тобто використаний алгоритм підпису та typ, або тип токену. Він представлений як JSON-об'єкт (рис. 3.3);

Рисунок 3.3 – Заголовок JWT

– тіло: містить вимоги або твердження про автентифікованого користувача або інші дані. Твердження можуть включати таку інформацію, як особистість користувача, ролі, дозволи та додаткові дані, що стосуються програми (рис. 3.4);

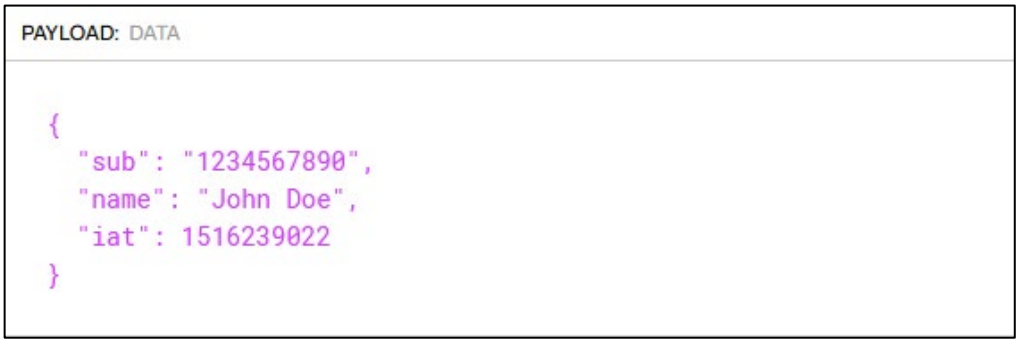

Рисунок 3.4 – Тіло JWT

– підпис: створюється шляхом поєднання закодованого заголовка, закодованого тіла і секретного ключа за допомогою певного алгоритму. Підпис забезпечує цілісність JWT і підтверджує, що він не був підроблений (рис. 3.5).

Рисунок 3.5 – Підпис JWT

Слід зауважити, що існують також JSON Web Token Claims. JWT Claims є частиною структури JWT, яка містить інформацію про суб'єкта, визначає права доступу та надає додаткові властивості, пов'язані з токеном. Вони визначаються як JSON-об'єкт, який містить ключі та значення. JWT Claims поділяються на три категорії:

а) registered claims: попередньо визначені стандартні поля, які використовуються для передачі стандартної інформації про токен. Деякі з них включають:

– «iss» (Issuer): визначає видавця токену;

– «sub» (Subject): визначає суб'єкта токену (користувача або ідентифікатор);

– «exp» (Expiration Time): вказує час, коли токен стає недійсним;

– «iat» (Issued At): вказує час, коли токен був створений;

– «aud» (Audience): визначає призначення (одержувача) токену;

б) public claims: власні поля, які використовуються для передачі додаткової інформації, специфічної для застосунку або домену, що використовує токен. Вони можуть бути використані згідно потреб і вимог застосунку;

в) private claims: спеціальні власні поля, які використовуються для обміну приватною інформацією між двома сторонами, які домовились про це. Вони не мають попереднього визначення і використовуються на власний розсуд розробника застосунку.

Таким чином, JWT Claims можуть бути додані до токену при його створенні, розшифровані і подальшій перевірці, а також при використанні. Вони дозволяють передавати інформацію про користувача, контролювати доступ та встановлювати додаткові контекстні дані, пов'язані з JWT [21].

Вибір алгоритму підпису для JWT важливий аспект при роботі з токенами. Існує декілька алгоритмів, котрі можна використовувати для підпису. Обирати алгоритму слід від потреб інформаційної системи.

Слід зазначити, що для підпису JWT завжди використовується Secure Hash Algorithm (SHA) – сімейство криптографічних хеш-функцій, розроблених Агентством національної безпеки (АНБ) США. Ці хеш-функції приймають на вхід (повідомлення) і видають на виході фіксований розмір (хеш-значення), який зазвичай представлений у вигляді послідовності цифр або букв. На даний

момент, існує три версії хеш-функції: SHA-1, SHA-2 та SHA-3, але з 2022 року хеш-функція SHA-1 визнана слабкою [22], тому безпечно використовувати лише SHA-2 та SHA-3.

Хеш-функції SHA-2 широко використовуються, оскільки вони мають кілька важливих властивостей:

– стійкість до колізій: Хеш-функції SHA-2 (нп., SHA224, SHA256, SHA384 і SHA512) призначені для мінімізації ймовірності того, що два різні вхідні дані дадуть однакове хеш-значення (колізія). Ця властивість забезпечує цілісність даних, що хешуються;

– детермінований вихід: При однакових вхідних даних хеш-функція SHA-2 завжди видає одне і те ж хеш-значення. Ця властивість має вирішальне значення для перевірки цілісності підпису JWT;

– одностороння функція: Обчислювально неможливо відновити вихідні дані за хеш-значенням. Ця властивість гарантує, що вихідні дані залишаються конфіденційними.

У зв’язці з SHA-2, в JWT-підписах використовують наступні алгоритми:

– HMAC: алгоритми, такі як HMAC-SHA256, HMAC-SHA384 і HMAC-SHA512, використовують хеш-функції групи SHA-2 в поєднанні з секретним ключем. HMAC забезпечує цілісність і автентичність повідомлення, генеруючи код автентифікації повідомлення на основі хешу з використанням секретного ключа;

– RSA: алгоритми, такі як RS256, RS384 і RS512, використовують хеш-функції SHA разом з асиметричними парами ключів (відкритий і закритий ключі). RSA забезпечує механізм цифрового підпису, в якому відправник підписує JWT за допомогою свого приватного ключа, а одержувач перевіряє підпис за допомогою відкритого ключа відправника;

– ECDSA: алгоритми, такі як ES256, ES384 і ES512, використовують хеш-функції SHA в поєднанні з криптографією еліптичної кривої. ECDSA

забезпечує цифрові підписи, подібні до RSA, але з меншою довжиною ключа та швидшими обчисленнями.

Різниця між 256, 384 і 512-бітним хешем полягає в довжині вихідного хешу. Більша довжина хешу, як правило, означає більший простір для пошуку і, таким чином, більшу захищеність від потенційних атак. Однак, більша довжина хешу також призводить до більшого розміру підпису і дещо повільніших обчислень.

Кожен алгоритм безпечно використовувати, відштовхуючись від потреб. Для інформаційної системи в сфері охорони здоров’я, потрібна як безпека, так і коротка довжина токену, для швидших обрахувань. Вирішенням є ES256, тобто ECDSA з хеш-функцією SHA256.

Принцип алгоритму ECDSA базується на використанні еліптичних кривих для генерації цифрових підписів. Основна ідея полягає у використанні математичних властивостей еліптичних кривих для створення публічного та приватного ключів, а також для підпису та перевірки повідомлень. Публічний ключ використовується для верифікації підписів, а приватний ключ – для створення підпису повідомлення.

Для створення підпису, повідомлення спочатку хешується, тобто перетворюється в унікальний вихідний код фіксованої довжини. За допомогою приватного ключа обчислюється підпис шляхом застосування математичних операцій до хешу повідомлення та приватного ключа. Отриманий підпис додається до повідомлення.

Для перевірки підпису, отримувач бере публічний ключ користувача, якому належить підписане повідомлення, та знову хешує повідомлення. Потім використовуючи публічний ключ, перевіряє відповідність підпису хешу повідомлення. Якщо підпис співпадає, то повідомлення вважається достовірним, інакше – недостовірним (рис. 3.6).

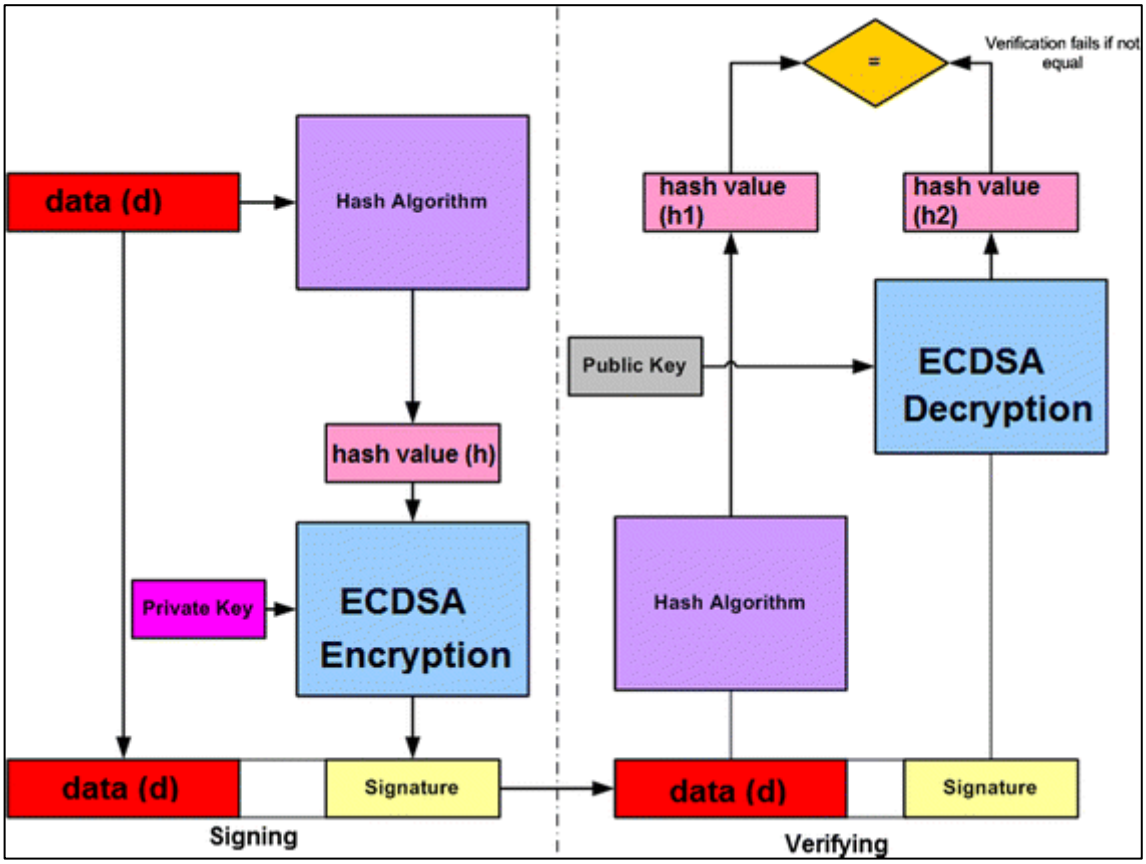

Рисунок 3.6 – Етапи підписання та верифікації ECDSA

Визначившись зі структурою JWT, та алгоритмом підписання, доцільно перейти до демонстрації використання токенів під час автентифікації (рис. 3.7) та авторизації (рис. 3.8). Це буде проілюстровано шляхом діаграм, а саме діаграм послідовності.

Діаграма послідовності – це графічний інструмент, що використовується для візуалізації та моделювання взаємодії між об'єктами або компонентами системи в часовій послідовності. Діаграма складається з вертикальних ліній, які представляють об'єкти, та горизонтальних стрілок, які показують передачу повідомлень між ними. Це дозволяє візуально зрозуміти послідовність взаємодії та взаємозв'язки між об'єктами системи.

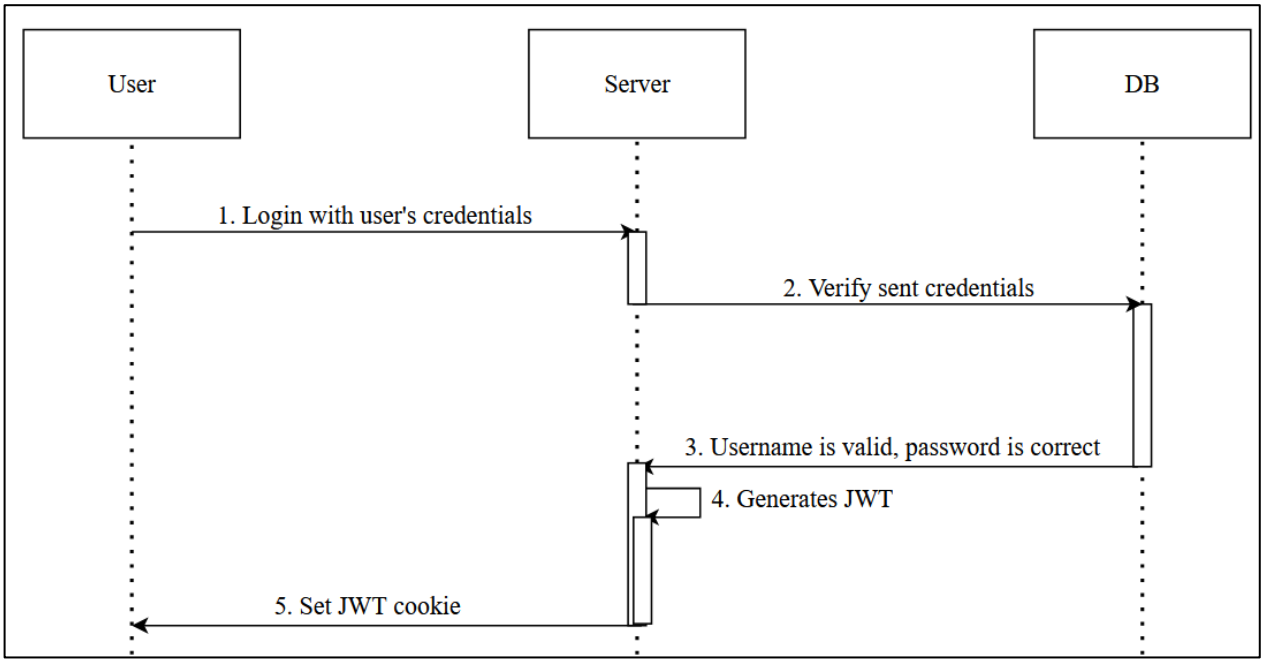

Рисунок 3.7 – Діаграма послідовності автентифікація користувача з JWT-токеном шляхом надання облікових даних

На даній діаграмі послідовності зображена успішна автентифікація користувача:

1) користувач надає облікові дані, такі як ім’я користувача та пароль акаунта;

2) вебзастосунок робить запит до бази даних з перевіркою, чи існує такий користувач та чи правильний пароль було вказано;

3) в даному випадку облікові данні були надані вірні, так як отриманні дані з БД є вірними та підтверджують існування користувача;

4) після отримання позитивних результатів, вебзастосунок генерує JWT-токен;

5) після генерації JWT-токена вебзастосунок надає його користувачеві у вигляді файлу cookie.

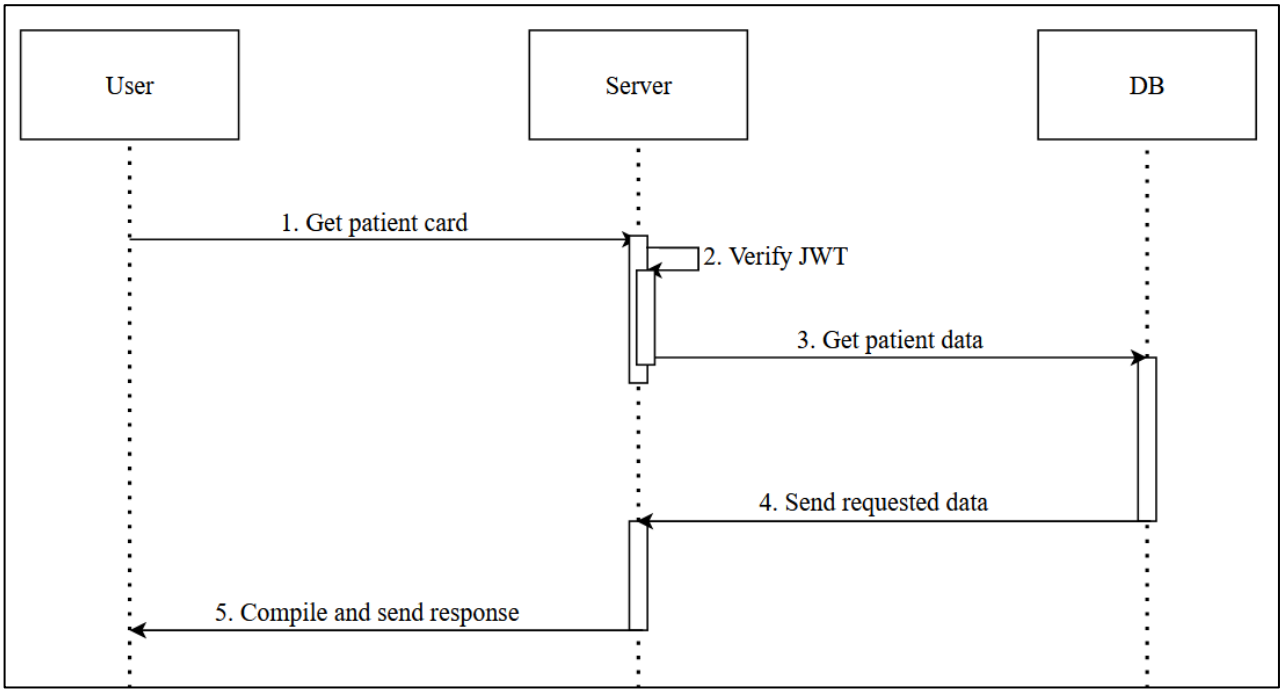

Рисунок 3.8 – Діаграма послідовності контролю доступу користувача з JWT-токеном

На даній діаграмі зображено механізм контролю авторизації або доступу користувача. В даному випадку, JWT-токен є правильним, та запитані дані існують в БД:

1) користувач робить запит на отримання картки пацієнта;

2) вебзастосунок перевіряє, чи справжній JWT-токен (чи існує користувач, чи є його роль: лікар, чи справжній підпис);

3) після отримання позивних результатів, вебзастосунок робить запит до БД;

4) запитані дані існують в БД, та відправлені до вебзастосунку;

5) отримані данні та інша інформація генерується в одну відповідь та відсилається користувачеві.

3.3 Огляд технологій

Для реалізації інформаційної системи для сфери охорони системи було обрано мову програмування C#, а саме фреймворк .NET.

.NET – це платформа розробки програмного забезпечення, створена компанією Microsoft. Вона надає інфраструктуру для побудови, розгортання та виконання різноманітних застосунків на різних платформах, таких як Windows, macOS і Linux.

Головні компоненти .NET включають компілятори, середовище виконання Common Language Runtime (CLR) та бібліотеки класів.

Компілятори .NET дозволяють розробникам писати код на різних мовах програмування, таких як C#, Visual Basic.NET і F#. Вони перетворюють вихідний код у проміжний код, який може бути виконаний в середовищі CLR.

CLR є середовищем виконання, яке керує виконанням програм на платформі .NET. Воно відповідає за керування пам'яттю, збірку сміття, безпеку, обробку виключень та інші операції системного рівня.

Бібліотеки класів .NET є великою стандартною набором компонентів і функціональностей, які розробники можуть використовувати при створенні своїх програм. Вони надають широкий спектр можливостей, включаючи роботу з мережами, базами даних, графікою, шифруванням та багато іншого.

Деякі з переваг .NET включають кросплатформність, що означає, що розроблені застосунки можуть працювати на різних операційних системах, наявний широкий набір функцій у стандартній бібліотеці, існує підтримка різних мов програмування, що дозволяє розробникам вибирати мову залежно від вимог проєкту, а також програміст має розширену екосистему інструментів для розробки, таких як інтегровані середовища розробки (англ. Integrated Development Environment – IDE), налагоджувачі та інші допоміжні інструменти.

Загалом, .NET є потужним і гнучким інструментом, котрий підходить до розробки ІС. Для розробки було обрано середовище Rider IDE.

Rider – це кросплатформне IDE, сумісне з Windows, macOS та Linux. Це забезпечує однаковий досвід розробки на різних операційних системах, дозволяючи розробникам працювати на їхній улюбленій платформі.

IDE надає набагато більший набір функцій та інструментів у порівнянні з Visual Studio Enterprise, включаючи редагування коду, навігацію, інтелектуальне завершення коду, рефакторинг, налагодження, інтеграцію контролю версій та вбудований термінал. Ці функції підвищують продуктивність розробника та спрощують процес розробки.

Rider підтримує широкий спектр мов програмування, таких як C#, VB.NET, F#, JavaScript, TypeScript, HTML, CSS, а також різні фреймворки, включаючи .NET, ASP.NET, Unity, Xamarin і Blazor. Така універсальність дозволяє розробникам працювати над різноманітними проєктами, використовуючи улюблену мову та фреймворк.

Також Rider включає в себе розширені можливості аналізу коду, виявлення потенційних проблем, виконання перевірок коду та пропонування швидких рішень. Він допомагає виявляти помилки, вузькі місця та "запахи" коду, сприяючи створенню чистого та зручного для підтримки коду. IDE також пропонує повний набір інструментів рефакторингу для покращення структури та дизайну коду.

IDE підтримує розширюваність за допомогою плагінів і розширень, що дозволяє розробникам розширювати його функціональність і пристосовувати його до своїх конкретних потреб розробки. JetBrains пропонує процвітаючу екосистему плагінів, надаючи додаткові інструменти, фреймворки та мовну підтримку на застосунок до вбудованих функцій IDE.

JetBrains має багаторічну історію надання частих оновлень та вдосконалень своїх продуктів, і Rider отримує вигоду від цієї безперервної підтримки. Ці оновлення забезпечують сумісність з новітніми технологіями, вирішують проблеми та впроваджують нові функції. JetBrains також забезпечує чуйну підтримку клієнтів і активно взаємодіє зі спільнотою розробників.

Таким чином, Rider – це потужне середовище розробки, що пропонує повний набір функцій, широку підтримку мов і фреймворків, а також

бездоганну інтеграцію з екосистемою JetBrains. Вона надає розробникам продуктивне та ефективне середовище розробки для різних проєктів.

Microsoft SQL Server було обрано для реалізації ЦБД. MS SQL Server – це сервер реляційних баз даних, розроблений корпорацією Microsoft. Сервер баз даних є, додатком до бази даних, який використовується для зберігання даних, а інші програмні застосунки отримують і зберігають дані за допомогою SQL. MS SQL Server має наступні переваги:

– легке встановлення: Всі продукти Microsoft легко встановлюються за допомогою процедури інсталяції в один клік та зрозумілого графічного інтерфейсу з великою кількістю інструкцій для неспеціалістів. MS SQL Server містить всі ці характеристики і має надзвичайно зручний інтерфейс встановлення, на відміну від інших серверів баз даних, які вимагають складних конфігурацій командного рядка. В основному, щоб завантажити MS SQL Server, вам потрібен мережевий фреймворк, мінімум 1 Гб пам'яті та система NTFS;

– покращена продуктивність: MS SQL Server містить чудові можливості стиснення та шифрування, що призводить до покращення функцій зберігання та пошуку даних;

– безпека: Сервер MS SQL вважається одним з найбезпечніших серверів баз даних зі складними алгоритмами шифрування, що робить практично неможливим злом рівнів безпеки, встановлених користувачем. Сервер MS SQL не є сервером баз даних з відкритим вихідним кодом, що знижує ризик атак на сервер баз даних;

– відмінний механізм відновлення даних: Сервер MS SQL повністю усвідомлює важливість розміщених даних, тому містить багато складних функцій, які дозволяють відновлювати та відновлювати дані, які були втрачені або пошкоджені.

Через відсутність графічного інтерфейсу, використання, корегування та тестування застосунку буде виконуватись за допомогою Postman.

Postman – це популярний інструмент, що використовується розробниками для тестування, документування та обміну Application Programming Interface (API). Він спрощує процес розробки API, надаючи зручний інтерфейс для створення HTTP-запитів, управління та організації кінцевих точок API, а також автоматизації робочих процесів тестування.

За допомогою Postman розробники можуть легко надсилати запити, перевіряти та підтверджувати відповіді, налаштовувати автентифікацію та створювати автоматизовані набори тестів.

Він також дозволяє створювати детальну документацію по API і ділитися колекціями з членами команди. Postman підтримує різні методи HTTP, типи запитів і формати даних, що робить його цінним інструментом для розробки і тестування API.

Висновки до розділу 3

В третьому розділі були розглянуті можливі варіанти стандартів та з міркувань безпеки та швидкості обрано бібліотеку JSON Web Token (JWT) для створення (підписання) і підтвердження (перевірки) токенів, які використовуються для аутентифікації/авторизації користувачів.

Було описано структуру токенів, обрано надійний алгоритм для підпису, а саме ES256, а також проілюстровано механізми автентифікації та авторизації, з використанням JWT.

Також, було обрано мову програмування, середовище, фреймворк та СУБД.

4 ОПИС РЕАЛІЗАЦІЇ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Обравши мову програмування, середовище, фреймворк, СУБД та надійний алгоритм для створення JWT-токенів було розроблено інформаційну систему згідно зі сформульованими рекомендаціями щодо покращення захисту КС шляхом впровадження запропонованих безпечних механізмів автентифікації та авторизації.

4.1 Опис програмної реалізації

4.1.1 Опис архітектури

Інформаційна система була реалізована як API (рис. 4.1), адже головною задачею є покращення механізмів захисту, а не розробка user-friendly інтерфейсу.

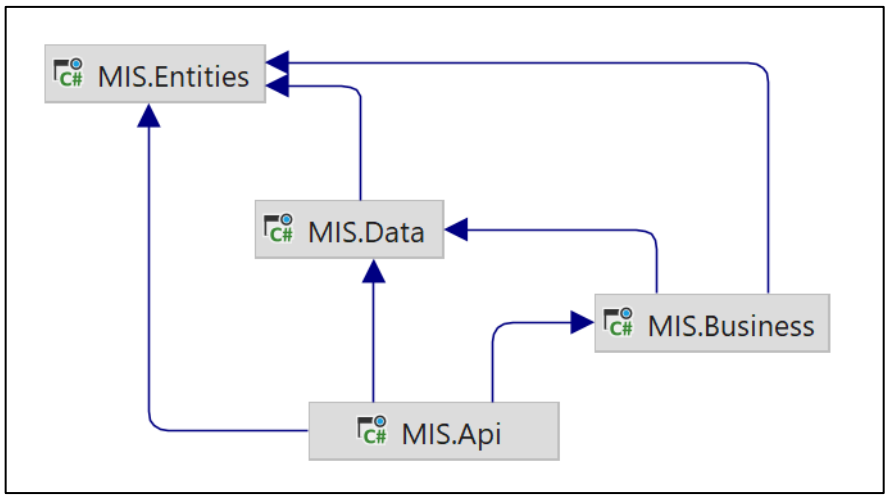

Рисунок 4.1 – Діаграма структури створеного проєкту

MIS.API відповідає за:

– реалізацію шляхів або routes, до яких будуть звертатись користувачі;

– реалізацію перевірки автентифікації та авторизації шляхом валідації JWT;

– реалізацію обробки помилок.

MIS.Business відповідає за:

– реалізацію будь-яких користувацьких функцій;

– реалізацію та вся конфігурація JWT (який секрет та алгоритм використовується, що передається в тілі, тощо).

MIS.Entities відповідає за:

– реалізацію сутностей.

MIS.Data відповідає за:

– деяку окрему конфігурацію сутностей.

Надалі, в окремих підрозділах, будуть розглядатись окремі частини реалізацій.

Згідно з рис. 1.4, MIC є окремою інформаційною системою, та окремим об’єктом від ЦБД. Через це, база даних створеної інформаційної системи містить приблизну інформацію, котра використовується в IC Ehealth (рис. 4.2).

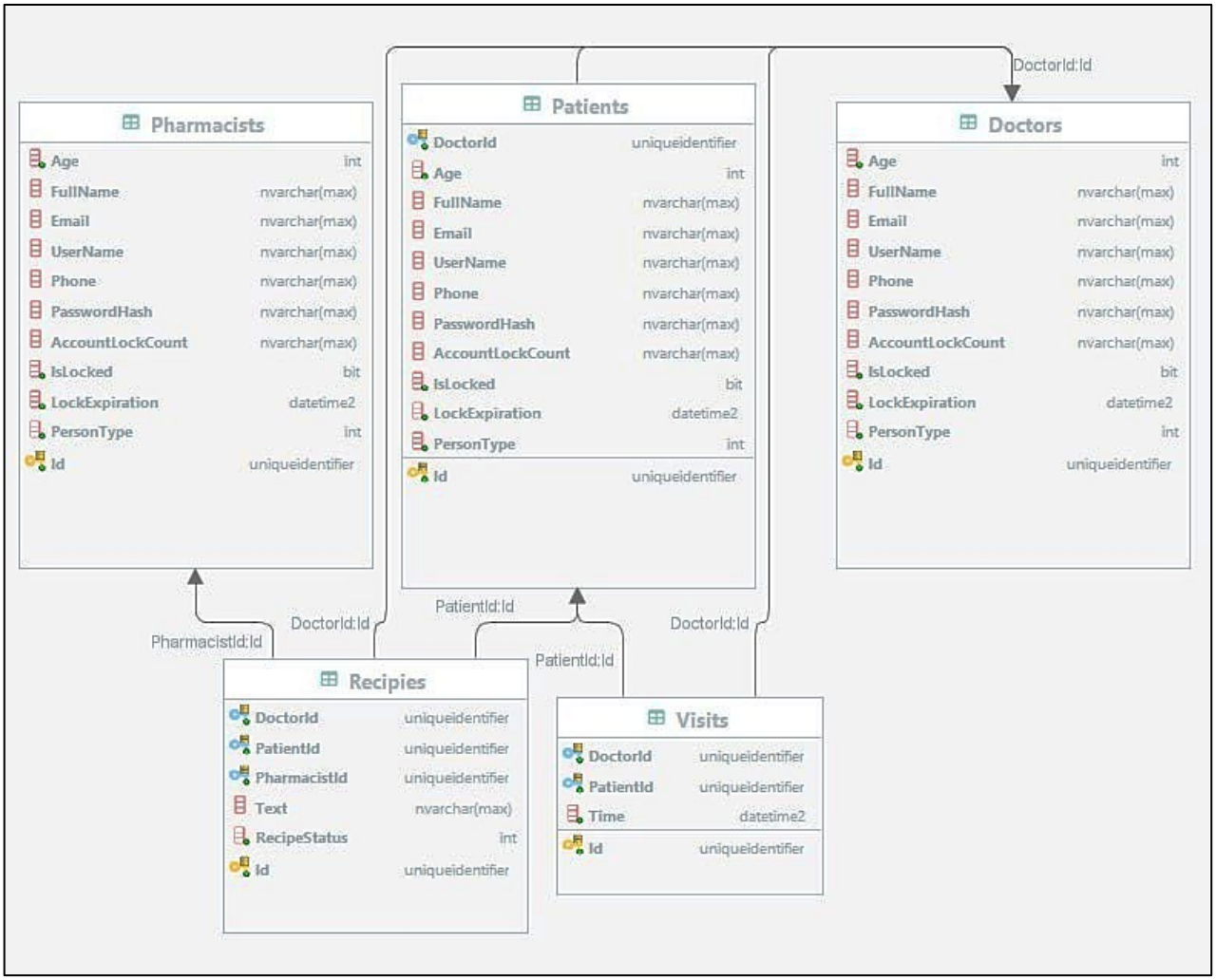

Рисунок 4.2 – Структура розробленої бази даних

Розроблена база даних містить 5 таблиць: Pharmacists, Patients, Doctors, Recipies та Visits.

Таблиця Pharmacists, Patients та Doctors мають однакові аргументи. Однак значення PersonType регулює роль користувачів: фармацевт, пацієнт та лікар.

Створені ролі краще проілюструють механізм контролю доступу, адже різні користувачі мають доступ до різних функцій та даних.

Неавтентифікований користувач має право на:

– реєстрацію;
– вхід до акаунту.

Користувач з роллю пацієнт може виконувати наступні дії:

– перегляд своєї медичної картки;
– запис до певного лікаря;
– зареєструватись на прийом;
– перегляд стану рецепта.

Користувач з роллю лікар може виконувати наступні дії:

– перегляд своїх пацієнтів;
– випис рецептів.

Користувач з роллю фармацевт може виконувати наступні дії:

– надання згоди на рецепт.

Також, задля безпеки акаунтів користувачів, було створено аргументи AccountLockCount, IsLocked, LockExpiration, котрі використовуються в механізмі захисту від brute-force атак, котрий буде розглянуто далі.

4.1.2 Опис автентифікації та авторизації

Якщо користувач не має акаунту, йому треба зареєструватися. Слід зазначити, що сам процес реєстрації має умовний характер, адже в ІС Ehealth процес є більш складним через використання інших державних установ. Запит та відповідь застосунку зображена на рис. 4.3.

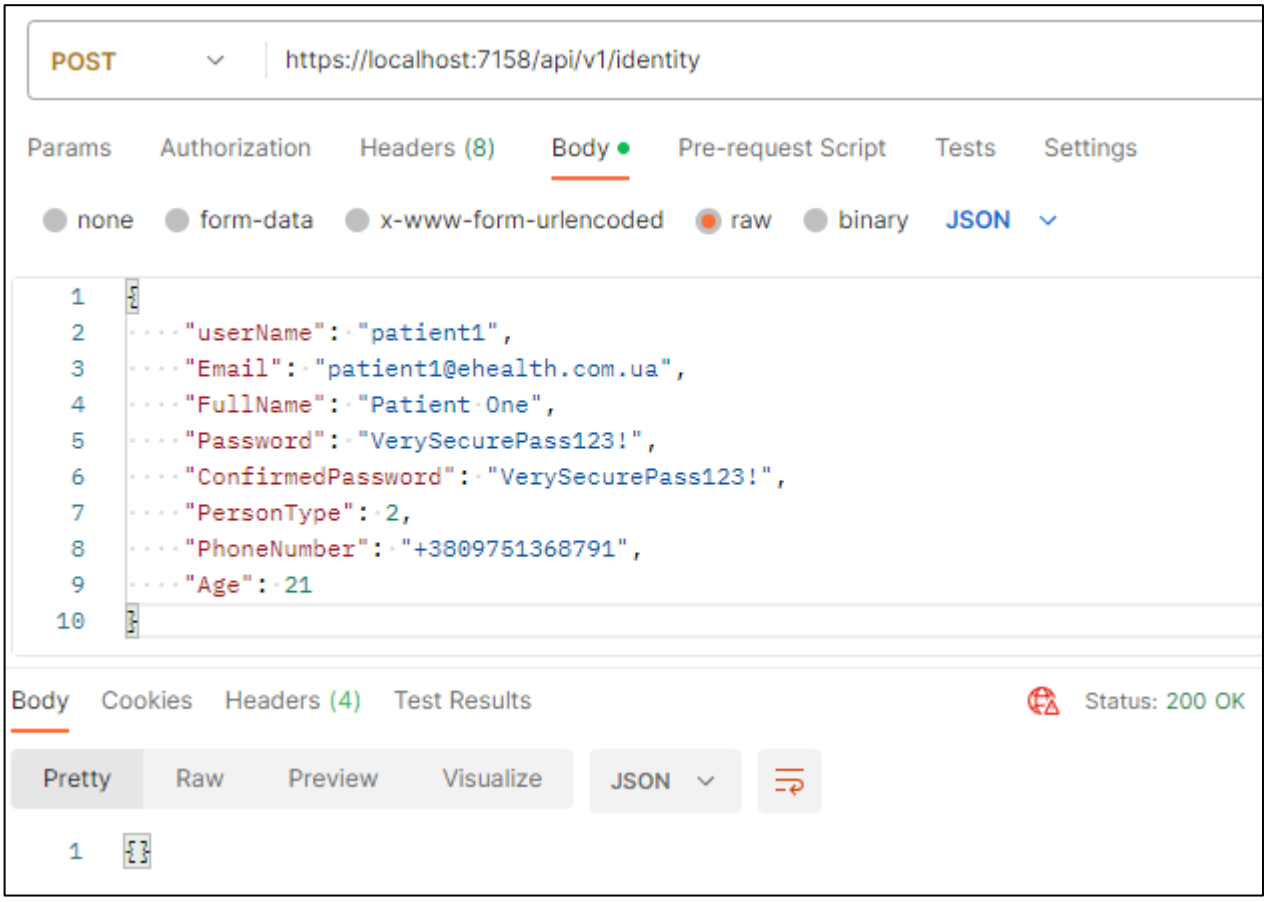

Рисунок 4.3 – Користувача з роллю пацієнт було зареєстровано

Після реєстрації користувач може пройти автентифікацію (рис. 4.4):

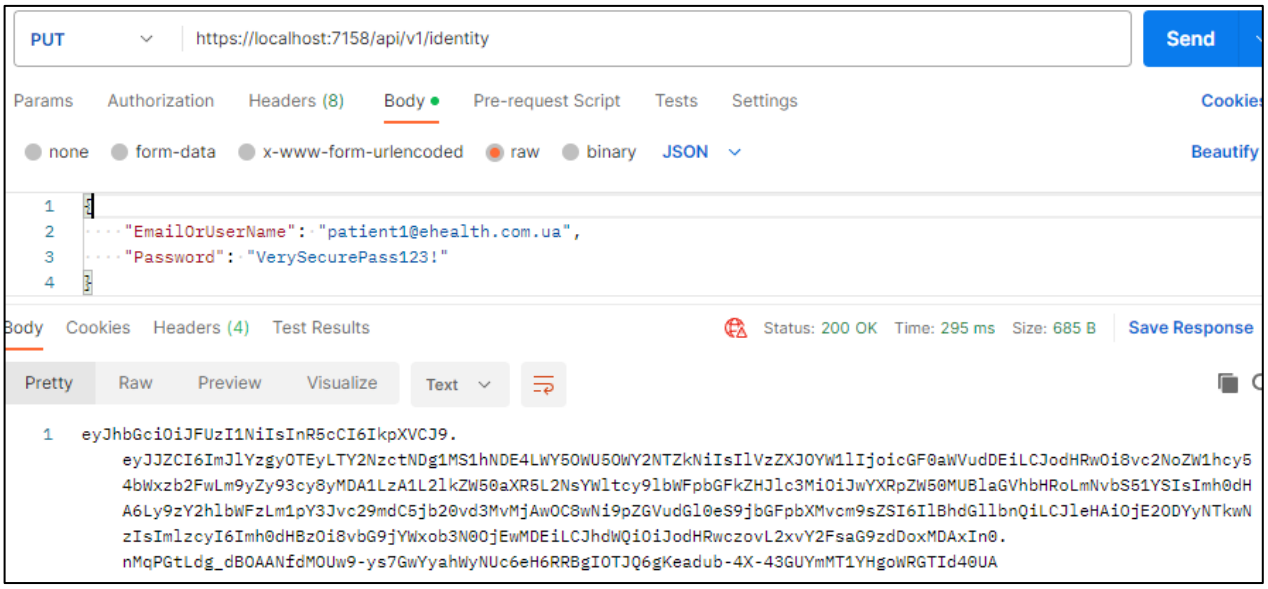

Рисунок 4.4 – Проходження автентифікації

Як тільки застосунок перевірив дійсність наданих даних, користувачу надсилається JWT-токен. Слід розглянути, як саме він реалізований (рис. 4.5).

```
public static void AddAuth(this IServiceCollection services, IConfiguration configuration)
{
    var jwtSection = configuration.GetSection( key: "Jwt");
    services.Configure<JwtSettings>(jwtSection);
    var jwtSettings = jwtSection.Get<JwtSettings>();
    var key = ECDsa.Create(ECCurve.NamedCurves.nistP256);
    jwtSettings.Key = key;
    services.AddSingleton<IJwtService, JwtService>();
    services.AddSingleton(jwtSettings);

    services.AddAuthorization();
    services.AddAuthentication( configureOptions: options =>
    {
        options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
        options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
        options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
    }).AddJwtBearer(o :JwtBearerOptions =>
    {
        o.SaveToken = true;
        o.TokenValidationParameters = new TokenValidationParameters
        {
            ValidateIssuer = true,
            ValidIssuer = jwtSettings.Issuer,
            ValidateIssuerSigningKey = true,
            IssuerSigningKey = new ECDsaSecurityKey(key),
            ValidateAudience = true,
            ValidAudience = jwtSettings.Audience,
            ValidateLifetime = true
        };
    });
}
```

Рисунок 4.5 – Реалізація механізмів автентифікації та авторизації

Як вже було розглянуто раніше, для підпису токену потрібен секрет, котрий в нашому випадку вказується на 6-му рядку. Але у .NET є можливість одразу вставити функцію для генерації секрету. Таким чином, можливо уникнути впровадження секрету як рядка, адже це несе ризик для вебзастосунку у разі витоку програмного коду [23]. Також, як тільки вебзастосунок буде перезапущено, старий секрет не буде дійсним через автоматичну генерацію нового. цей рядок коду створює екземпляр класу ECDsa

з використанням еліптичної кривої NIST P-256. Була створено ще одна функція, котра містить конфігурацію для генерації токенів (рис. 4.6).

```
public Task<string> GenerateJwtToken(Person user, IEnumerable<string> roles)
{
    List<Claim> claims = new()
    {
        new Claim(type: "Id", value: user.Id.ToString()),
        new Claim(type: "UserName", value: user.UserName),
        new Claim(type: ClaimTypes.Email, value: user.Email)
    };
    claims.AddRange(collection: roles.Select(r :string => new Claim(type: ClaimTypes.Role, value: r)));

    var jwtToken = new JwtSecurityToken(
        _jwtSettings.Issuer,
        _jwtSettings.Audience,
        claims,
        expires: DateTime.UtcNow.AddMinutes(_jwtSettings.AccessTokenExpirationMinutes),
        signingCredentials: new SigningCredentials(new ECDsaSecurityKey(_jwtSettings.Key),
            algorithm: SecurityAlgorithms.EcdsaSha256));

    return Task.FromResult(new JwtSecurityTokenHandler().WriteToken(jwtToken));
}
```

Рисунок 4.6 – Функція генерації JWT

Функція створює окремі claims, котрі містять персональні дані користувача (Id, UserName, Email), термін закінчення валідності токену (60 хвилин від дати підписання), емітента тощо. Обраним алгоритм ECDSA з хеш-функцією SHA-256 було встановлено в змінній «signingCredentials». Таким чином, згенерований токен буде мати наступний вигляд після декодування (рис. 4.7).

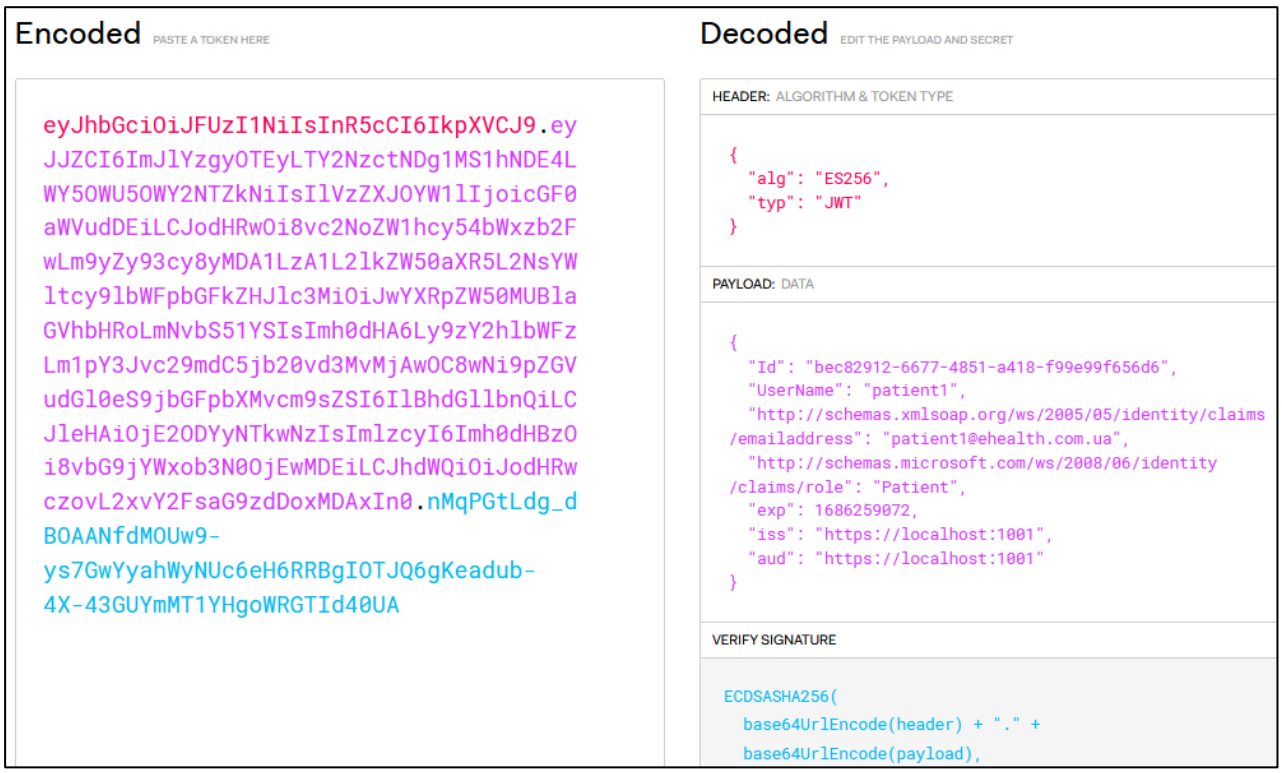

Рисунок 4.7 – Вигляд згенерованого JWT-токена після декодування

Надалі цей токен використовується в хедері Authorization наступним чином (рис. 4.8).

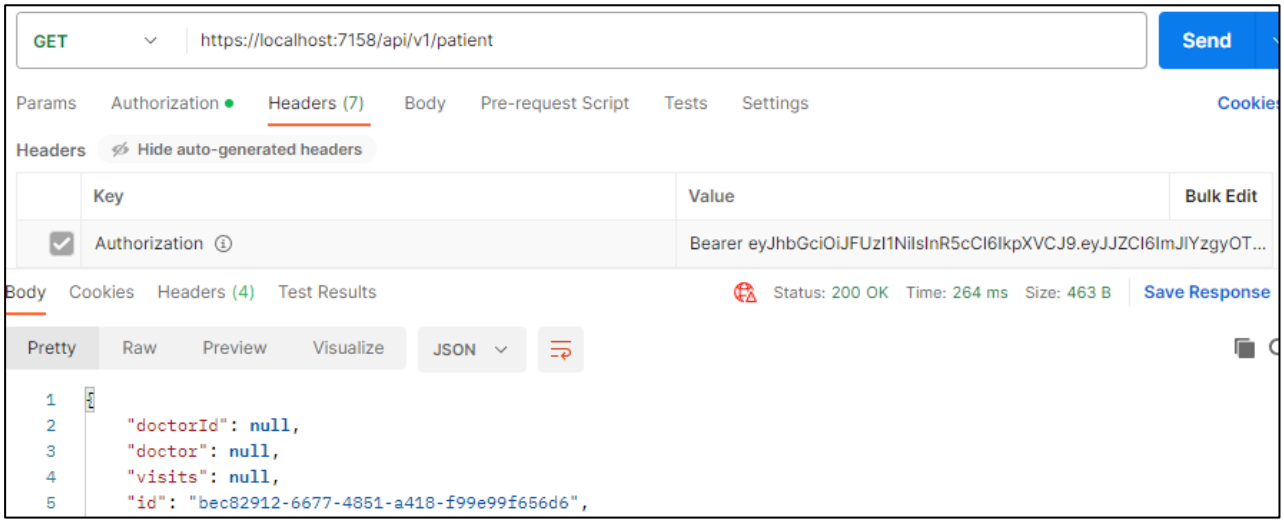

Рисунок 4.8 – Приклад використання JWT-токена

Якщо ж користувач не має токену, то механізм авторизації не пропустить запит (рис. 4.9).

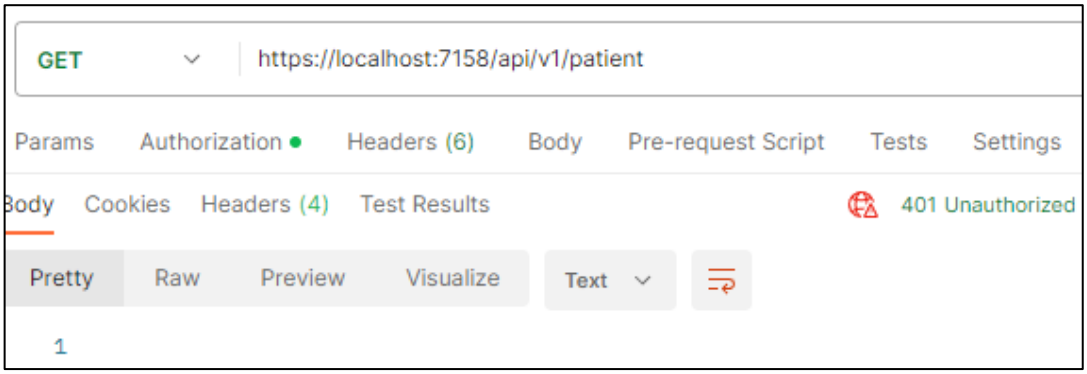

Рисунок 4.9 – Застосунок відповів кодом «401 Unauthorized», що свідчить про відсутність авторизації

4.1.3 Опис інших функцій МІС

Медична інформаційна система має певний цикл дій. Він включає в себе дії всіх користувачів. Як тільки пацієнт, лікар та фармацевт зареєстровані в застосунку, пацієнт може записатись до певного лікаря (рис. 4.10).

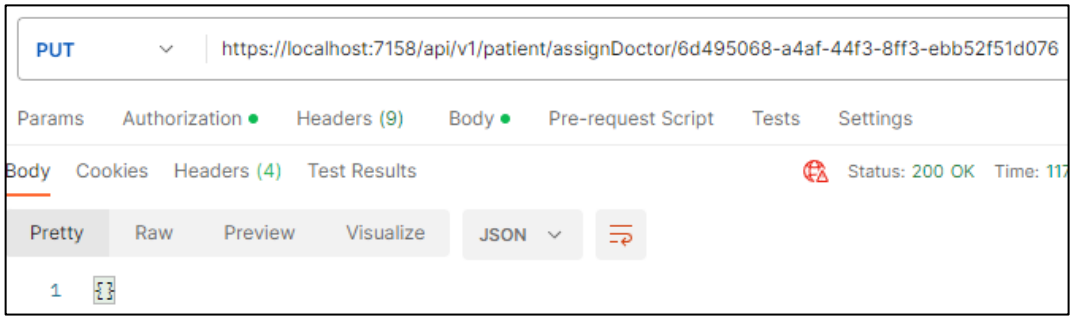

Рисунок 4.10 – Пацієнт успішно записався до «лікаря 1»

Після визначення лікаря, пацієнт може назначити дату зустрічі (рис. 4.11).

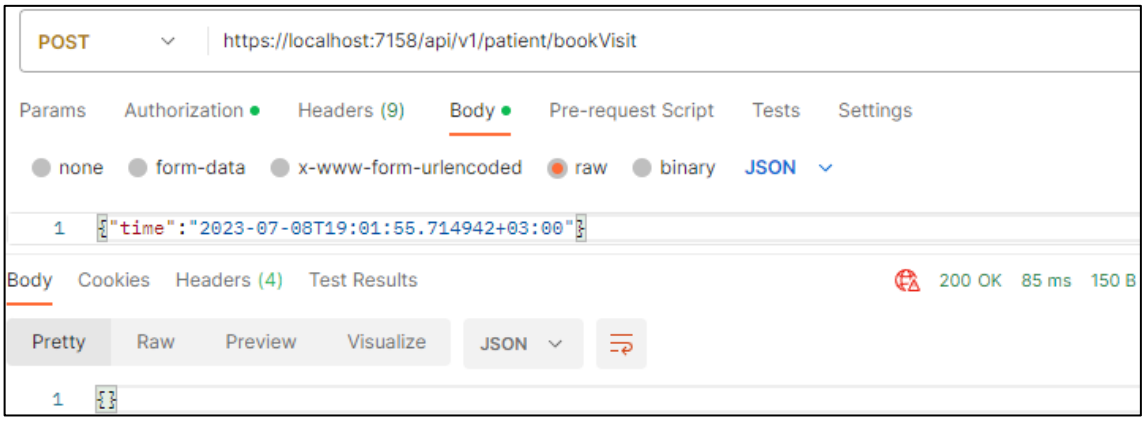

Рисунок 4.11 – Пацієнт успішно назначив дату зустрічі до «лікаря 1»

Користувач з роллю «лікар», може переглядати медичну картку своїх пацієнтів, котра містить всю їхню інформацію та майбутні візити (рис. 4.12).

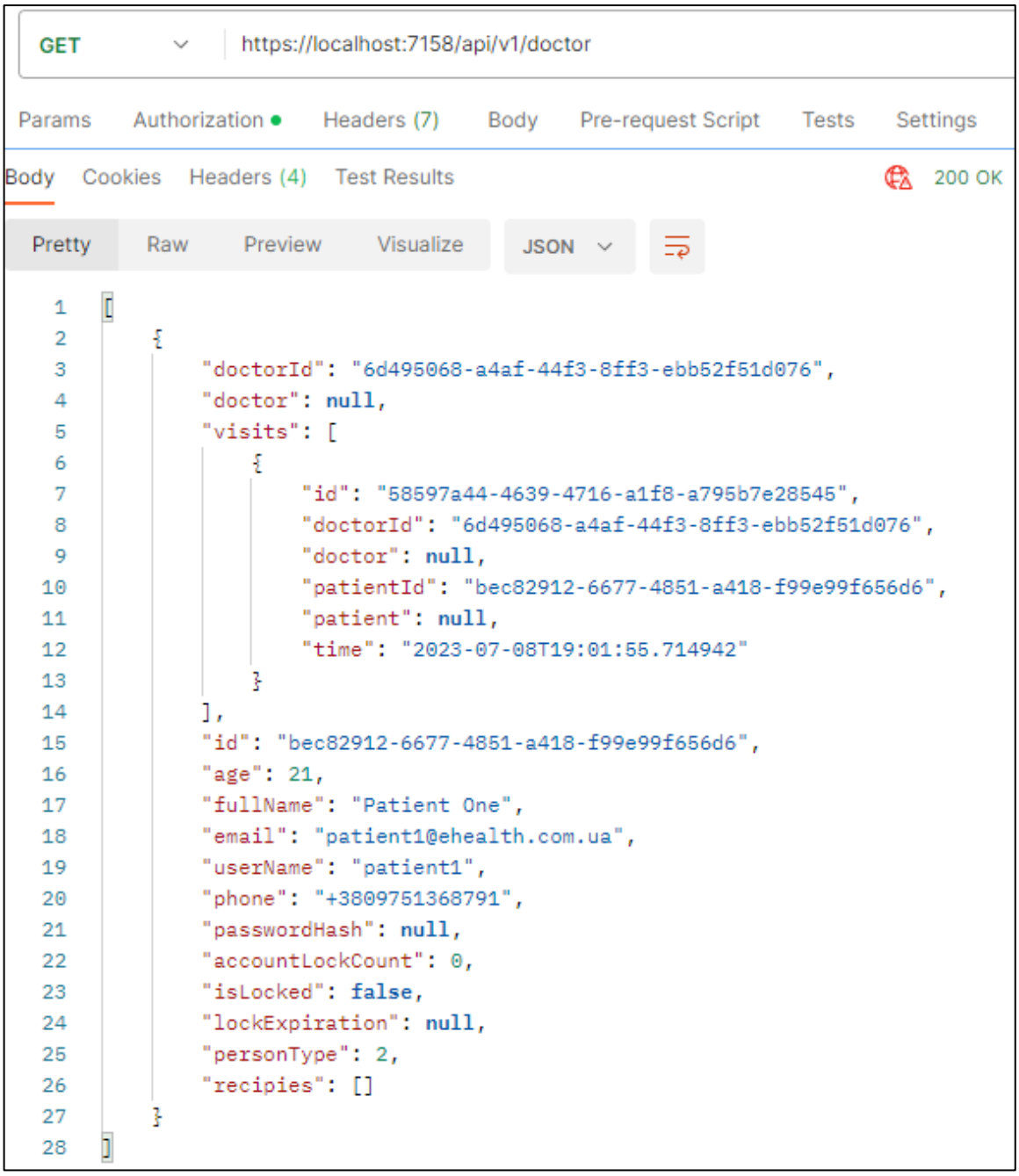

Рисунок 4.12 – Вигляд медичної картки «пацієнта 1»

Після проведення зустрічі, «лікар» виписує рецепт «пацієнту 1», де вказує його ідентифікатор та ідентифікатор «фармацевта 1», котрий зможе надати право на використання цього рецепту надалі (рис. 4.13).

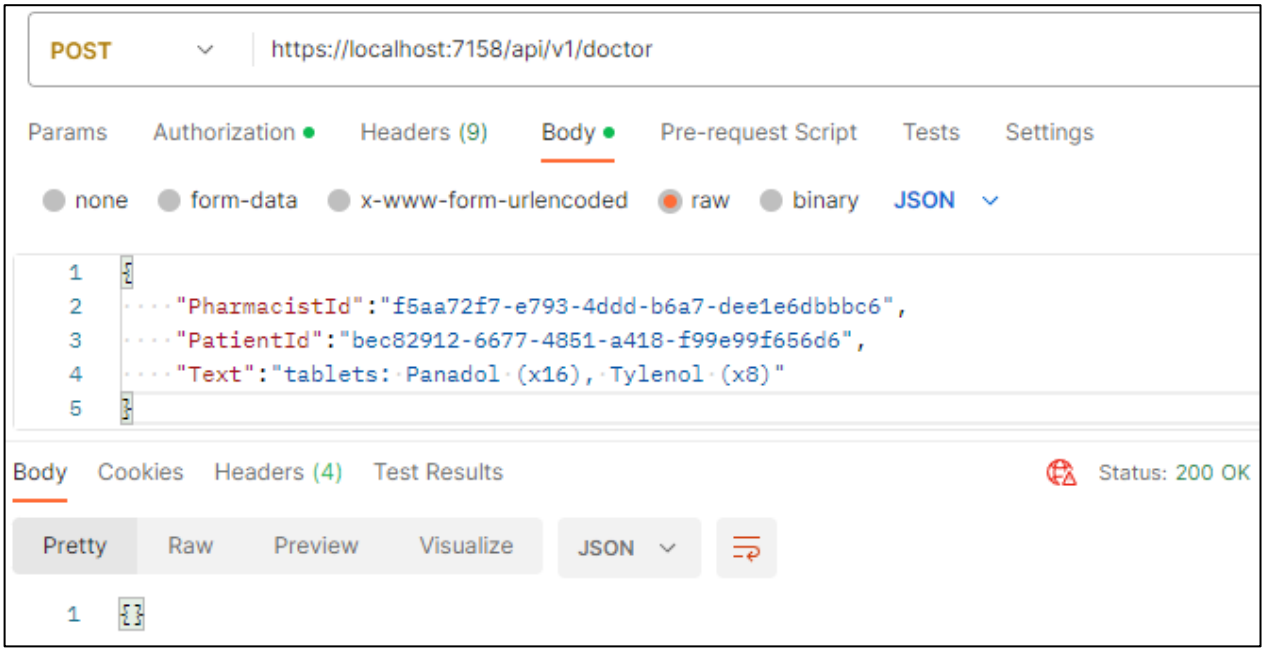

Рисунок 4.13 – Лікар 1 виписав рецепт пацієнту 1

Фармацевт надає дозвіл на отриманий рецепт (рис. 4.14).

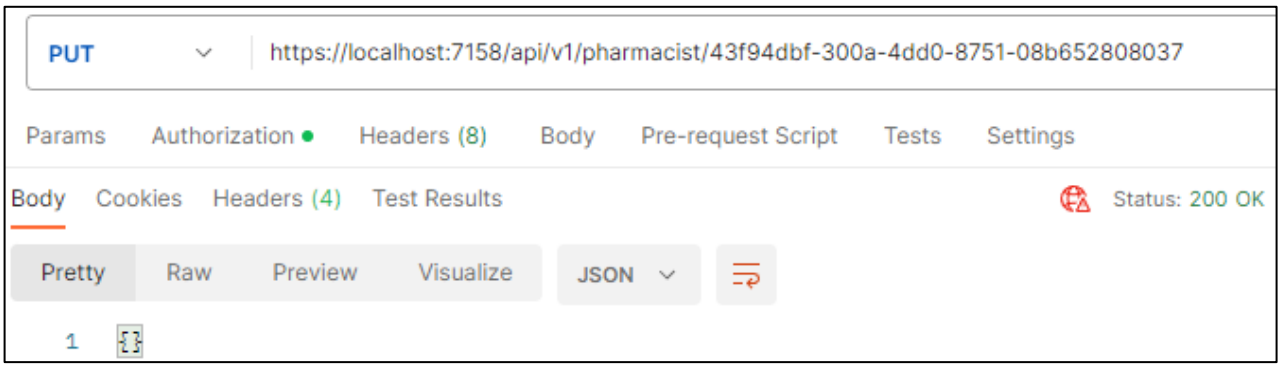

Рисунок 4.14 – «Фармацевт 1» надав дозвіл на використання рецепту

Фінальним етапом є отримання позитивного статусу змінної «recipeStatus» (рис. 4.15). Далі пацієнти знову записуються до лікаря, назначають зустріч та отримують рецепт. Ці дії передають складність та зв’язаність інформаційної системи Ehealth.

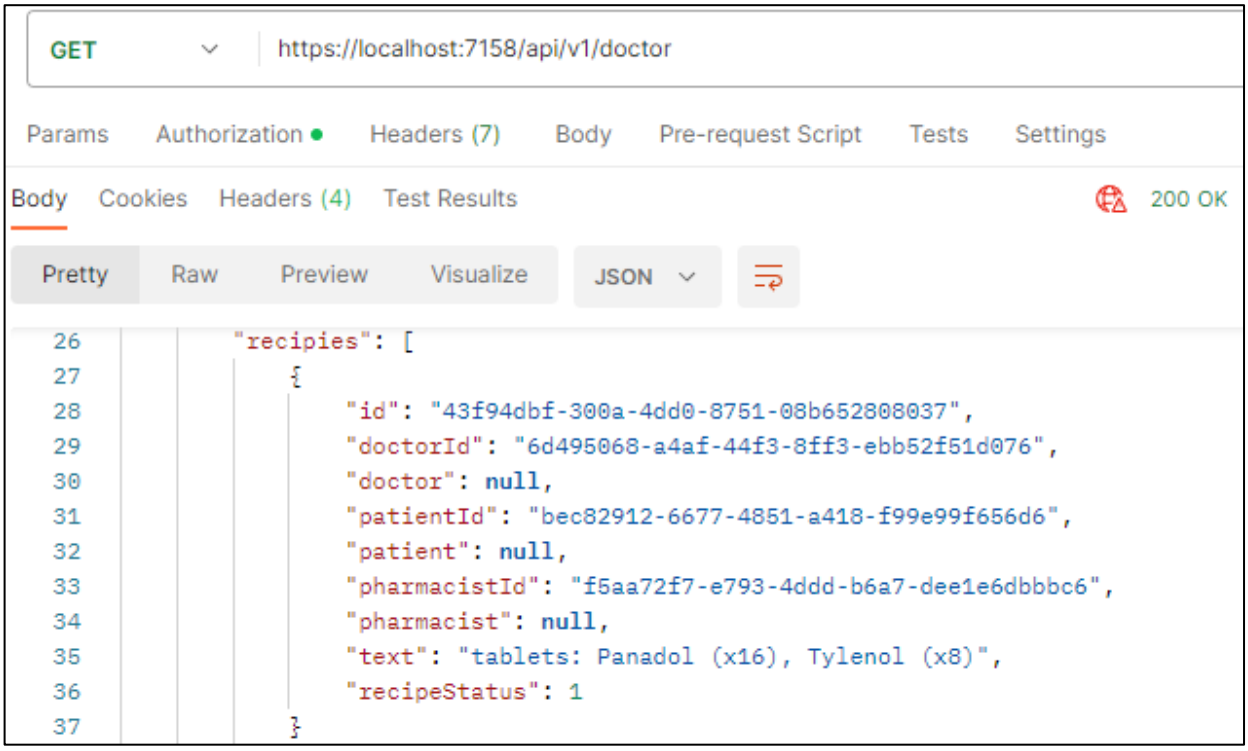

Рисунок 4.15 – Рецепт надано

4.1.4 Опис механізмів захисту

Одним з механізмів був розроблений механізм захисту від brute-force атак (рис. 4.16).

```
if (user is null || (user.AccountLockCount == AccountLockCount && user.LockExpiration > DateTime.Now))
    throw new ApplicationException(ErrorMessage);

var result = _passwordHasher.VerifyHashedPassword(user,
    hashedPassword: user.PasswordHash,
    providedPassword: request.Login.Password);
if (result != PasswordVerificationResult.Success)
{
    if (++user.AccountLockCount == AccountLockCount)
    {
        user.LockExpiration = DateTime.Now.AddHours(5);
        user.IsLocked = true;
    }

    _dbContext.People.Update(user);
    await _dbContext.SaveChangesAsync(cancellationToken);

    throw new ApplicationException(ErrorMessage);
}
```

Рисунок 4.16 – Реалізація механізму захисту від brute-force атак

Як тільки кількість наданих невірних облікових даних дорівнюватиме встановленої кількості, а саме5 невірних спроб, то акаунт користувача блокується на 5 годин [24]. Така політика була обрана через критичність інфраструктури. При такому блокуванні користувач не зможе увійти в свій акаунт навіть з вірними обліковими даними, поки 5 годин не пройдуть (рис. 4.17).

AccountLockCount	5
IsLocked (Hex)	true
LockExpiration	2023-06-09 06:21:30.0179244

Рисунок 4.17 – Значення змінних в базі даних

Реалізовано також було механізм захисту від переліковування користувачів, шляхом надання однієї відповіді незалежно від того, чи існує вказане ім’я користувача (рис. 4.18), чи ні (рис. 4.19).

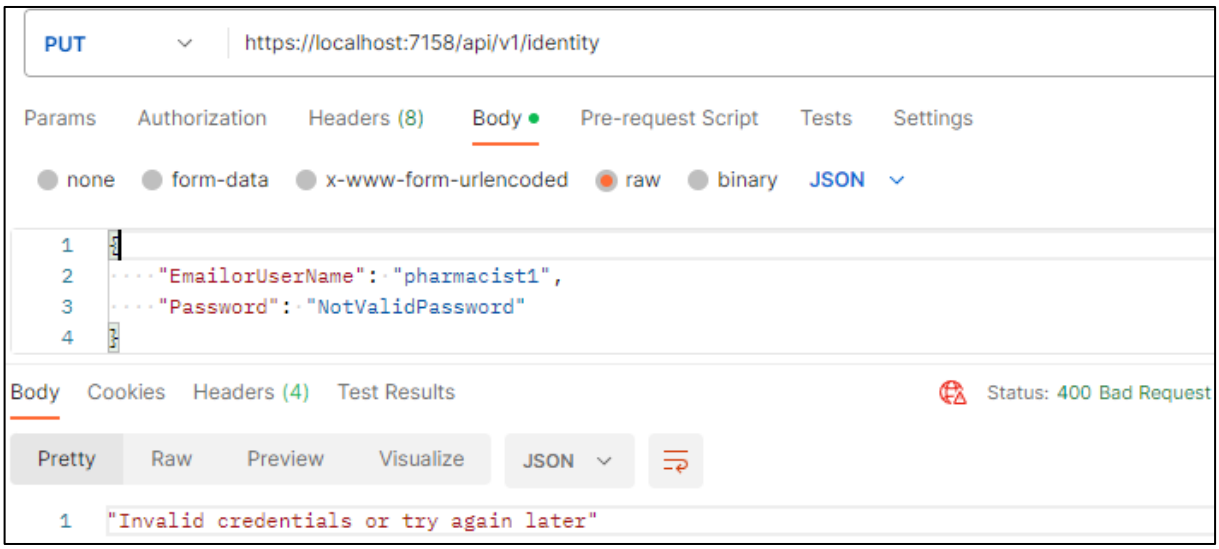

Рисунок 4.18 – Відповідь застосунку при правильному імені користувача

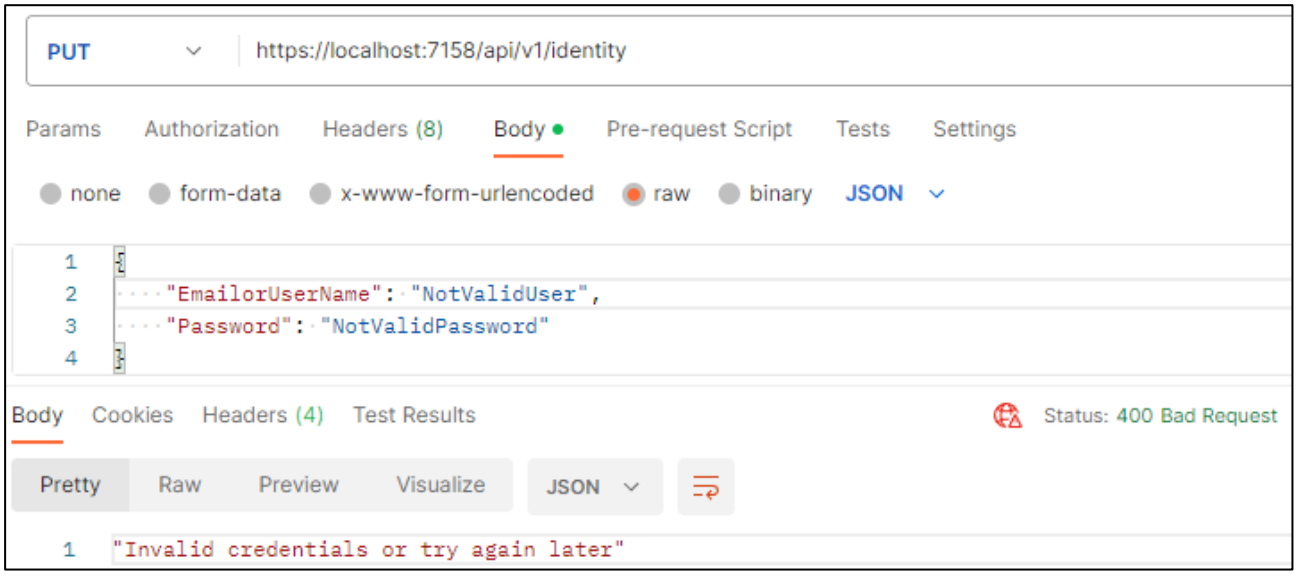

Рисунок 4.19 – Відповідь застосунку при неправильному імені користувача

Також під час створення інформаційної системи використовувались безпечні запити до бази даних Microsoft SQL Server за допомогою LINQ [25] (рис. 4.20).

```
public async Task<Patient> Handle(ShowCardQuery request, CancellationToken cancellationToken)
{
    var patient = await _dbContext.Patients.FirstOrDefaultAsync( predicate: p :Patient => p.Id == request.Id,
        cancellationToken); // Task<Patient>
    patient.PasswordHash = null;
    return patient;
}
```

Рисунок 4.20 – Приклад безпечного запиту до бази даних

4.2 Тестування інформаційної системи

Під час тестування вебзастосунку, було проведено ряд заходів для визначення слабких місць як у захищених механізмах, так і у звичайних. Таким чином, можна зазначити, що застосунок не має вразливостей у генерації та валідації JWT-токена, блокує обліковий запис користувача при brute-force атаках, не видає корисну інформацію при переліченні імен користувачів.

Механізм авторизації не пропускає запити неавтентифікованих користувачів, через відсутність JWT-токена. Так як кожен токен містить

ідентифікатор користувача (рис. 4.7), то без токену застосунок не зможе надати інформацію, адже при валідації токену бере з нього ідентифікатор та порівнює.

Однак, не усюди застосунок використовує ідентифікатор з токену. Як раз в такому місці було знайдено вразливість, котра дозволяла фармацевту надавати згоду рецептам навіть котрі не містили його ідентифікатор.

Вразливість було виявлено після виконання наступних кроків:

1) створення другого фармацевта (рис. 4.21);

	1	2
Id	b040a22a-3ed9-4d4b-bb13-03d9f6ff842b	f5aa72f7-e793-4ddd-b6a7-dee1e6dbbbc6
Age	31	45
FullName	Pharmacist Two	Pharmacist One
Email	pharmacist2@ehealth.com.ua	pharmacist1@ehealth.com.ua

Рисунок 4.21 – Існування двох фармацевтів

2) надання запиту на дозвіл видачі рецепта для «фармацевта 2» від «лікаря» (рис. 4.22);

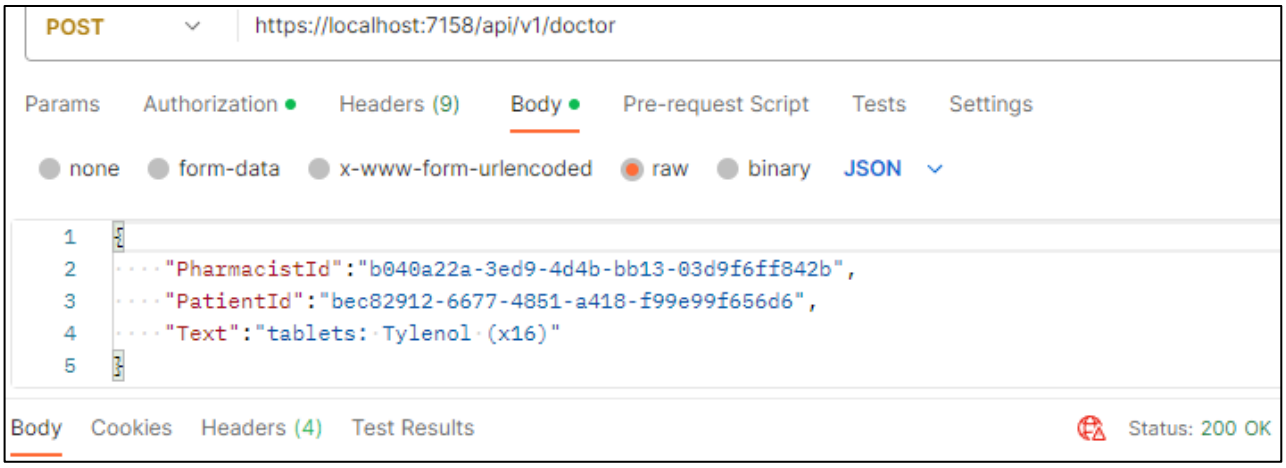

Рисунок 4.22 – Створено запит на новий рецепт для другого фармацевта

3) виконання входу від імені «фармацевта 1» (рис. 4.23);

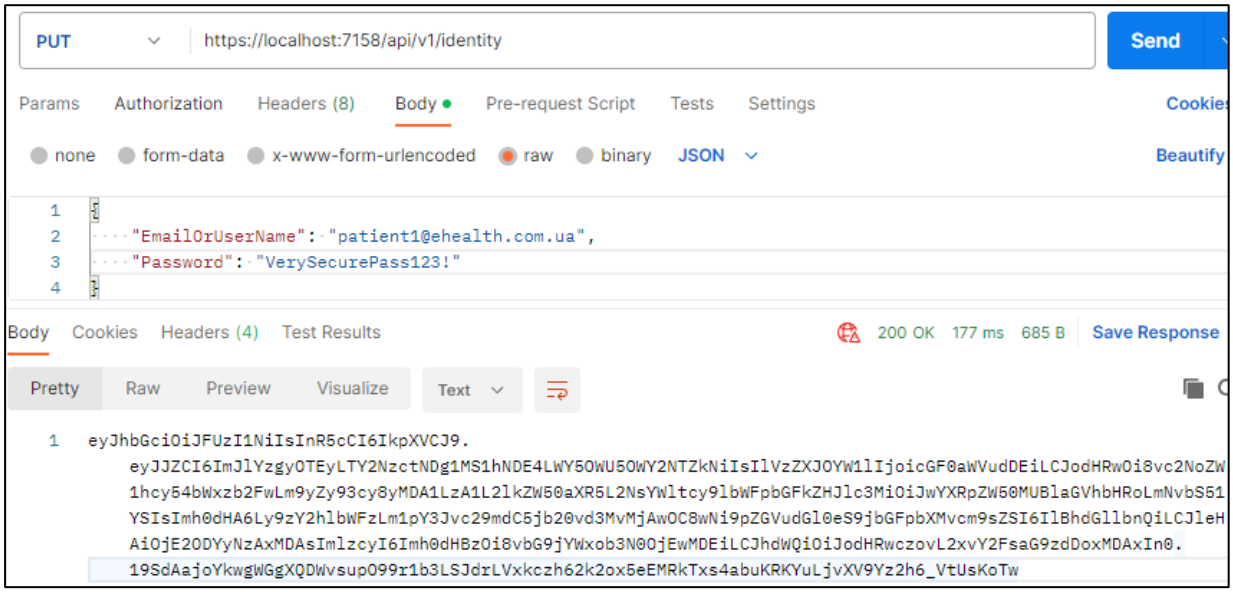

Рисунок 4.23 – Виконання входу та отримання JWT

4) успішне виконання запиту на надання дозволу на рецепт, котрий не був надісланий «фармацевту 1» (рис. 4.24).

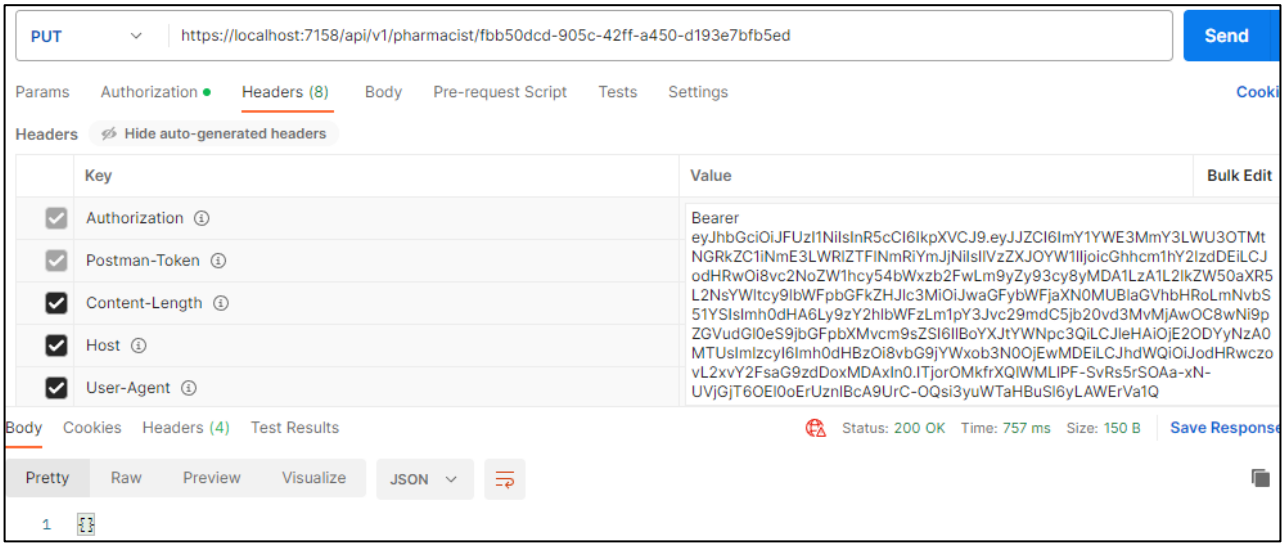

Рисунок 4.24 – Рецепт надано від імені іншого фармацевта

Вразливість мала місце через слабкий механізм контролю, котрий не перевіряв, чи співпадає ідентифікатор фармацевта, який підписує, та ідентифікатор, котрого вказано в рецепті. Було додано наступні рядки коду (рис. 4.25).

```
if (recipe.PharmacistId != request.PharmacistId)
    throw new AuthenticationException( message: "Access denied, access control error");
```

Рисунок 4.25 – Перевірка ідентифікаторів при підписання рецепту

Таким чином, вразливість в механізмі контролю було усунуто (рис. 4.26).

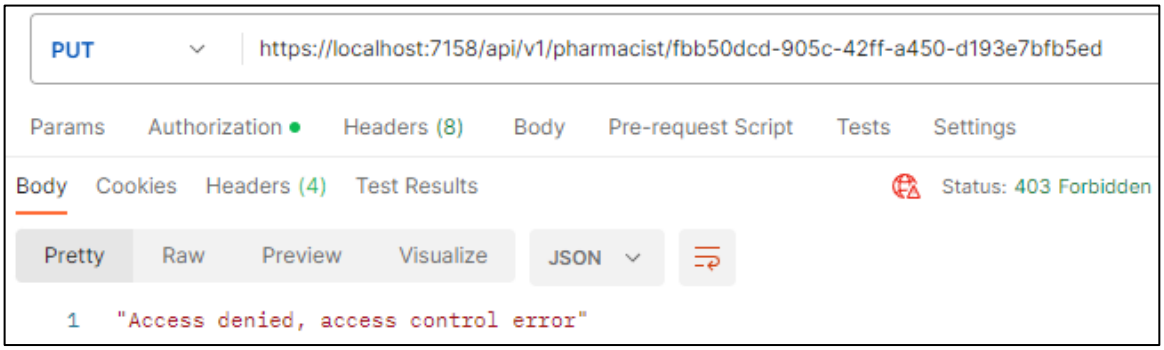

Рисунок 4.26 – Виправлена поведінка застосунку під час надання згоди на рецепт

При ретельному перегляді коду було знайдено подібну вразливість. Вона дуже схожа на минулу, але мала місце під час надання рецепту. Застосунок не перевіряв, чи записаний пацієнт у лікаря, ідентифікатор котрого вказується в рецепті. Було додано наступні рядки коду (рис. 4.27).

```
if (patient.DoctorId != doctor.Id)
    throw new AuthenticationException( message: "Access denied, access control error");
```

Рисунок 4.27 – Перевірка ідентифікаторів при підписання рецепту

Таким чином, ще одну вразливість в механізмі контролю було виправлено (рис. 4.28).

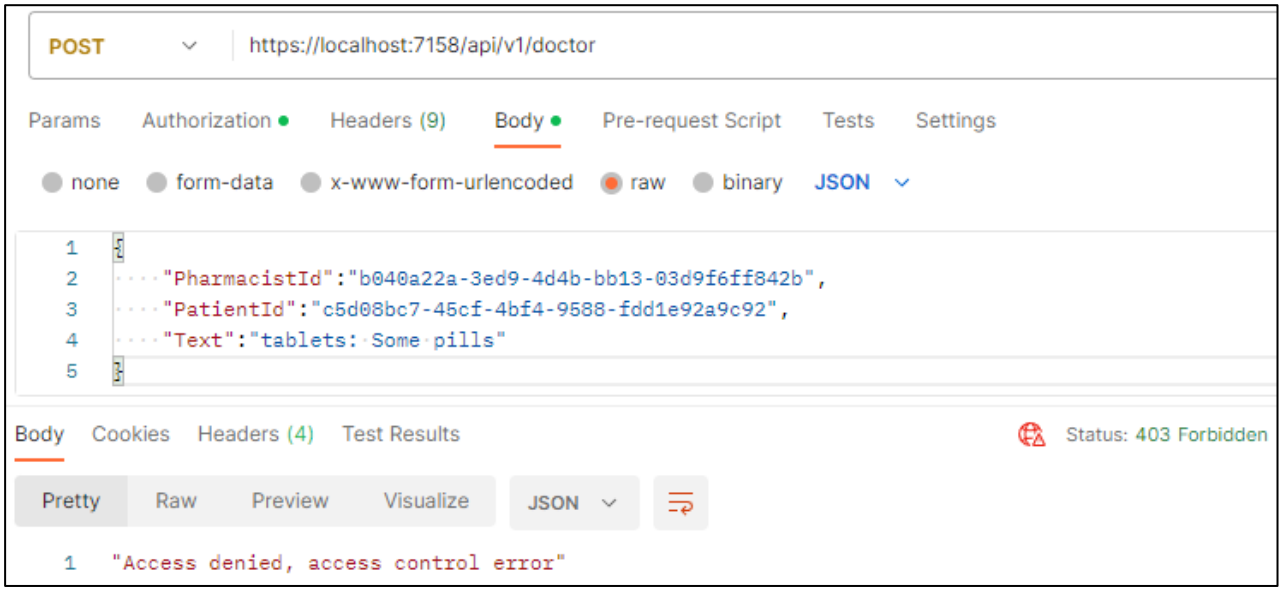

Рисунок 4.28 – Виправлена поведінка застосунку під час надання рецепту

Висновки до розділу 4

Було розглянуто архітектуру застосунку та бази даних, котра була створена для функціонування застосунку, адже в ІС Ehealth ЦБД є окремою інформаційною системою.

Детально оглянуто реалізацію генерації, надання та валідації JWT. Описано повний цикл дій в інформаційної системи та вказано додатково розроблені механізми захисту на основі створених рекомендацій.

В результаті тестування розробленої інформаційної системи, були знайдені вразливості в механізмі контролю доступу, котрі надавали можливість фармацевтам надавати згоду будь-яким рецептам, а лікарям – виписувати рецепти пацієнтам, котрі не зареєстровані у цього лікаря. Окрім цього вразливості в інших механізмах не було знайдено, а до механізму авторизації внесені зміни для виправлення вразливостей.

ВИСНОВКИ

Під час виконання бакалаврської кваліфікаційної роботи було створено захищену інформаційну систему для сфери охорони здоров’я.

Був проведений аналіз архітектури системи Ehealth, і визначено потенційно слабкий елемент, а саме медичні інформаційні системи. Було проаналізовано механізми автентифікації та авторизації.

Розглянуті типові вразливості механізмів автентифікації та контролю доступу продемонстрували необхідність створення нових механізмів на основі рекомендацій щодо виправлень та правильних конфігурацій. Створені рекомендації відповідають специфікацій NIST та методології OWASP.

Для реалізації ідентифікаторів сеансу були розглянуті різні варіанти, та обрано технологію JSON Web Token. Було детально розглянуто структуру токену та обрано метод шифруванням підпису ESHA256, а саме алгоритм ECDSA з хеш-функцією SHA-256. Також було створено діаграми послідовності для ілюстрування етапів обробки запитів користувача.

Були обрані технології та програми для розробки, які підходять для реалізації запланованого застосунку. Треба зауважити, що створення центральної бази даних та інших компонентів було виконано методом Grey Box, адже увага приділялась медичній інформаційній системі.

В результати роботи отриманий застосунок може бути впроваджений в інформаційну систему Ehealth або взятий за основу для подальших розробок згідно з вимогами до певної МІС вже з розробленими механізмами автентифікації та авторизації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Shameli-Sendi A. An efficient security data-driven approach for implementing risk assessment. Journal of Information Security and Applications. 2020. Vol. 54, no. 7:102593. DOI: 10.1016/j.jisa.2020.102593.

2. If healthcare doesn't strengthen its cybersecurity, it could soon be in critical condition. URL: https://www.weforum.org/agenda/2021/11/healthcare-cybersecurity/ (Last accessed: 01.05.2023).

3. Page C. NextGen Healthcare says hackers accessed personal data of more than 1 million patients. Publ. May 8, 2023. URL: https://techcrunch.com/2023/05/08/nextgen-healthcare-data-breach/ (Last accessed: 10.05.2023).

4. Healthcare data breaches: Insights and implications. Healthcare (Basel) / Seh A. H. та ін. ; 2020 May. Vol. 13, no. 8(2):133. DOI: 10.3390/healthcare8020133.

5. Колодяжний К. О., Журавська І. М. Розробка захищеної інформаційної системи для сфери охорони здоров’я. Могилянські читання – 2022 : тези доп. XXV Всеукр. наук.-метод. конф. Миколаїв, 7–11 листоп. 2022 р. Миколаїв : Чорном. нац. ун-т ім. Петра Могили, 2022. С. 31–33.

6. Using HTTP cookies. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies (Last accessed: 01.05.2023).

7. Електронна система охорони здоров'я в Україні. URL: https://ehealth.gov.ua/ (дата звернення: 02.05.2023).

8. IT-технології в медицині. URL: https://naukam.triada.in.ua/index.php/konferentsiji/42-dvanadtsyata-vseukrajinska-praktichno-piznavalna-internet-konferentsiya/462-it-tekhnologiji-v-meditsini (дата звернення: 04.05.2023).

9. Підключені до ЦБД медичні інформаційні системи. URL: https://ehealth.gov.ua/pidklyucheni-do-ehealth-mis/ (дата звернення: 02.05.2023).

10. Session management cheat sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.ht

ml (Last accessed: 06.05.2023).

11. NIST special publication 800-63B. Digital identity guidelines, authentication and lifecycle management / Grassi P. A., et al. 2017. 69 p. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf (Last accessed: 12.05.2023).

12. Blocking brute force attacks. URL: https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks (Last accessed: 13.05.2023).

13. If healthcare doesn't strengthen its cybersecurity, it could soon be in critical condition. URL: https://www.weforum.org/agenda/2021/11/healthcare-cybersecurity/ (Last accessed: 13.05.2023).

14. Hoffman A. Web application security. Exploitation and countermeasures for modern web applications. 1st ed. O'Reilly Media, 2020. 331 p.

15. NoSQL injection. Fun with objects and arrays. URL: https://owasp.org/www-pdf-archive/GOD16-NOSQL.pdf (Last accessed: 14.05.2023).

16. Hikvision cameras. URL: https://www.cisa.gov/news-events/ics-advisories/icsa-17-124-01 (Last accessed: 14.05.2023).

17. CVE-2017-7923. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-7923 (Last accessed: 14.05.2023).

18. Access control vulnerabilities and privilege escalation. URL: https://portswigger.net/web-security/access-control (Last accessed: 15.05.2023).

19. Bortz A., Barth A., Czeskis A. Origin cookies: Session integrity for web applications. URL: https://www.abortz.net/papers/session-integrity.pdf (Last accessed: 16.05.2023).

20. Introduction to JSON web tokens. URL: https://jwt.io/introduction (Last accessed: 20.05.2023).

21. JSON web token claims. URL: https://auth0.com/docs/secure/tokens/json-web-tokens/json-web-token-claims (Last accessed: 20.05.2023).

22. NIST retires SHA-1 cryptographic algorithm. URL:

https://www.nist.gov/news-events/news/2022/12/nist-retires-sha-1-cryptographic-algorithm (Last accessed: 21.05.2023).

23. Saad E., Mitchell R. Web Security Testing Guide v4.2. OWASP, 2020. 465 p.

24. ISO/IEC 27002:2022(E). Information security, cybersecurity and privacy protection – Information security controls. Switzerland, 2022. 164 p.

25. Security Considerations (Entity Framework). URL: https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/ef/security-considerations?redirectedfrom=MSDN (Last accessed: 02.06.2023).

ДОДАТОК А

Код програмного застосунку

JwtService.cs

```
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using MIS.Entities;
using Microsoft.IdentityModel.Tokens;
using MIS.Business.Settings;

namespace MIS.Business.JwtService
{
   public class JwtService : IJwtService
   {
      private readonly JwtSettings _jwtSettings;

      public JwtService(JwtSettings jwtSettings)
      {
         _jwtSettings = jwtSettings;
      }

      public Task<string> GenerateJwtToken(Person user, IEnumerable<string> roles)
      {
         List<Claim> claims = new()
         {
            new Claim("Id", user.Id.ToString()),
            new Claim("UserName", user.UserName),
            new Claim(ClaimTypes.Email, user.Email)
         };
         claims.AddRange(roles.Select(r => new Claim(ClaimTypes.Role, r)));

         var jwtToken = new JwtSecurityToken(
            _jwtSettings.Issuer,
            _jwtSettings.Audience,
            claims,
            expires: DateTime.UtcNow.AddMinutes(_jwtSettings.AccessTokenExpirationMinutes),
            signingCredentials: new SigningCredentials(new ECDsaSecurityKey(_jwtSettings.Key),
               SecurityAlgorithms.EcdsaSha256));

         return Task.FromResult(new JwtSecurityTokenHandler().WriteToken(jwtToken));
      }
   }
}
```

ServiceCollectionExtensions.cs

```
using System.Reflection;
using System.Security.Cryptography;
using FluentValidation;
using MediatR;
```

```
using MIS.Business.JwtService;
using MIS.Business.Settings;
using MIS.Business.Validation;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;

namespace MIS.Api.Extensions
{
   public static class ServiceCollectionExtensions
   {
      public static void AddValidation(this IServiceCollection services, Assembly
validatorsAssembly)
      {
         AssemblyScanner.FindValidatorsInAssembly(validatorsAssembly)
            .ForEach(item => services.AddScoped(item.InterfaceType, item.ValidatorType));
         services.AddScoped(typeof(IPipelineBehavior<,>),
typeof(PipelineValidationBehavior<,>));
      }

      public static void AddAuth(this IServiceCollection services, IConfiguration configuration)
      {
         var jwtSection = configuration.GetSection("Jwt");
         services.Configure<JwtSettings>(jwtSection);
         var jwtSettings = jwtSection.Get<JwtSettings>();
         var key = ECDsa.Create(ECCurve.NamedCurves.nistP256);
         jwtSettings.Key = key;
         services.AddSingleton<IJwtService, JwtService>();
         services.AddSingleton(jwtSettings);

         services.AddAuthorization();
         services.AddAuthentication(options =>
         {
            options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
            options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
            options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
         }).AddJwtBearer(o =>
         {
            o.SaveToken = true;
            o.TokenValidationParameters = new TokenValidationParameters
            {
               ValidateIssuer = true,
               ValidIssuer = jwtSettings.Issuer,
               ValidateIssuerSigningKey = true,
               IssuerSigningKey = new ECDsaSecurityKey(key),
               ValidateAudience = true,
               ValidAudience = jwtSettings.Audience,
               ValidateLifetime = true
            };
         });
      }
   }
```

```
}
```

JwtSettings.cs

```
using System.Security.Cryptography;

namespace MIS.Business.Settings
{
   public class JwtSettings
   {
      public string Issuer { get; set; }
      public string Audience { get; set; }
      public ECDsa Key { get; set; }
      public int AccessTokenExpirationMinutes { get; set; }
   }
}
```

ДОДАТОК Б

Матеріали апробації роботи

Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили

«МОГИЛЯНСЬКІ ЧИТАННЯ – 2022:
Досвід та тенденції розвитку суспільства в Україні:
глобальний, національний та регіональний аспекти»

XXV Всеукраїнська науково-практична конференція

ТЕЗИ

Комп'ютерні науки.

Технічні науки

Миколаїв, 7–11 листопада 2022 року

Миколаїв – 2022

УДК 004.62:004.9

Колодяжний К. О.,
бакалавріат,
Журавська І. М.,
д-р техн. наук, професор, в.о. завідувача кафедри КІ,
ЧНУ ім. Петра Могили, Миколаїв, Україна

РОЗРОБКА ЗАХИЩЕНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ СФЕРИ ОХОРОНИ ЗДОРОВ'Я

Інформаційні системи охорони здоров'я мають прямий та найважливіший вплив на людське життя. Їх неправильне створення, конфігурування, та адміністрування створює вразливість та може надати несанкціонований доступ до критичної інформації та важливих функцій, що може визвати не тільки перебої в роботі, а й матиме наслідки для здоров'я пацієнтів.

Рисунок 1 – Архітектура МІС

На теперішній час в системі охорони здоров'я України впроваджена та функціонує електронна система охорони здоров'я Ehealth. Зазначена електронна система Ehealth має двокомпонентну архітектуру, в якій користувач через відповідні медичні інформаційні системи (МІС) взаємодіє з центральною базою даних (ЦБД). МІС забезпечує можливість створення, перегляду, обміну інформацією та документами між державними, медичними та фармацевтичними реєстрами.

На теперішній час в Україні до ЦБД вже підключено більше 30 МІС.

ЗМІСТ

Застосування захищеної від НСД інформаційної системи для сфери охорони здоров'я не повністю, але зменшить процент можливих успішних атак. Її головною функцією буде впровадження функції тайммінгу для різних факторів МІС, які здійснюють виконання, моніторинг та аналіз процесів, що відбуваються в інформаційній системі закладу охорони здоров'я.

УДК 004.85

Миронюк О. М.,
магістрант,
Сіденко Є. В.,
канд. техн. наук, доцент, в.о. завідувача кафедри ПС,
ЧНУ ім. Петра Могили, Миколаїв, Україна

ІНТЕЛЕКТУАЛЬНА СИСТЕМА РОЗПІЗНАВАННЯ ТА ПОДОЛАННЯ ПЕРЕШКОД ТРАНСПОРТНИМИ ЗАСОБАМИ

У наш час більшість автомобільних концернів сьогодні працюють зі створення автономного транспорту. Автомобілі, автобуси, безпілотні таксі, колісні дрони для служби доставки та інші – сьогодні все частіше можна почути про подібні речі. Однак транспортні засоби зі справжнім автопілотом досі мають характер дослідницьких прототипів.

За оцінками різних компаній та організацій, повністю автоматизований транспорт займатиме значну частку серед пересувних засобів на дорогах світу вже у 2025–2050 роках. Це означає, що автомобілі будуть не тільки пересуватися самостійно, але й зможуть спілкуватися між собою за допомогою систем типу «Car-to-Car», а також з навколишньою інфраструктурою, світлофорами, центрами дорожнього регулювання. Величезна кількість електронних систем та технологій необхідна для роботи системи автоматичного пілотування транспортного засобу, частина з яких вже сьогодні ефективно використовується у передових транспортних засобах.

У теперішній час автомобілі стали майже невід'ємною частиною для людини. Вони надають масу переваг, таких як, незалежність від громадського транспорту, свобода переміщення та інші. Але з кожним роком людина вимагає від виробників автомобілів все більше нових та вдосконалених технологій, які мають спростити водіння та додати ще більше комфорту. Тому в машино-будівництві намагаються розробити все кращі технології, серед яких є і автомобільний штучний інтелект,

Але треба зауважити, що зі збільшенням кількості таких МІС зростає не тільки кількість сервісів для користувача, але й загрози для персональної та комерційної інформації користувачів, медичних закладів та бізнесу через наявні в таких МІС вразливості.

Подібні вразливості можуть бути використані для проведення наступних атак:

– збір та зміна інформації. Отримавши несанкціонований доступ (НСД) до інформаційної системи (ІС), зловмисник має можливість для збору та/або зміні будь-якої інформації о користувачах з різними правами (працівники, адміністратори, пацієнти тощо), налаштувань цієї та інших підключених до неї інформаційних систем та інших даних, до яких можливо отримати доступ різного рівня;

– вимагання коштів. Отримавши НСД до ІС та маючи права, зловмисник може створювати тиск на власників ІС. Це може бути як популярний зараз метод шифрування даних, так і часткове публікування отриманих чутливих даних для демонстрації їх володіння та можливості публікування всього обсягу.

Для підвищення захищеності МІС доцільно запроваджувати використання самодостатніх (містять в собі інформацію про клієнта та набір його прав) токенів авторизації та автентифікації. Перспективним напрямком вдосконалення зазначених токенів є створення їх з коротким життєвим циклом. В такому разі кожен вузол реєструється як клієнт і отримує унікальний ідентифікатор. Вузол отримує тимчасову пару токенів, підписану мережевим шлюзом. Використання токенів з коротким життєвим циклом дозволяє вирішити проблему блокування вузлів в розподіленій системі – таким чином інформацію про блокування вузла не потрібно поширювати, достатньо дочекатися закінчення терміну дії його ключа. Для забезпечення цілісності даних пропонується підписувати тіло повідомлення та URI адресата комбінованим ключем на основі токена та включати отриманий хеш в заголовок запиту. Таким чином сервер зможе верифікувати цілісність запиту. Даний підхід можна визначити як гібрид OAUTH1 та OAUTH2+JWT специфікацій. На основі інформації про клієнта з токена запиту генерується журнал аудиту.

Для розробки ІС буде обрано мову програмування Python, через її можливість подальшого масштабування, та імплементування. Через потребу використовувати документи як об'єкт бази даних, було обрано документо-орієнтовану нереляційну система керування Mongodb.

Доцільно буде створити окрему бібліотеку на мові програмування Python для більш зручного додавання нових функцій та використання при розробці ІС.

32