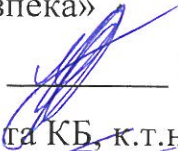
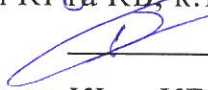


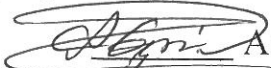
Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА
до випускної кваліфікаційної роботи бакалавра
на тему: «ПРОЕКТ ПІДСИСТЕМИ ЗАХИСТУ WEB-ОРІЄНТОВАНОГО
СЕРВІСУ ДОВІДОК»

Виконав студент 4-го курсу групи КБ-2
спеціальності 125 «Кібербезпека»

 Юрій Марущак
Керівник доцент кафедри КІ та КБ, к.т.н.

 Олександр Пірог
Рецензент завідувач кафедри КІ та КБ, к.т.н., доцент

 Андрій Єфіменко

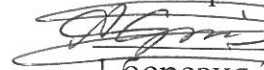
Житомир – 2022

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій Єфіменко
1 березня 2022 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Марущак Юрій Русланович

Тема роботи: «Проект підсистеми захисту web-орієнтованого сервісу довідок»
затверджена Наказом університету від 1 березня 2022 р. № 128/с

Термін здачі студентом закінченої роботи 25 червня 2022 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

Об'єктом дослідження є підсистема захисту web-орієнтованого сервісу довідок з використання технологій Laravel, Node.js, Nuxt.js, Blockchain.

Предметом дослідження є технології проектування підсистеми захисту web-орієнтованого сервісу довідок з використанням технологій Laravel, Node.js, Nuxt.js та Blockchain.

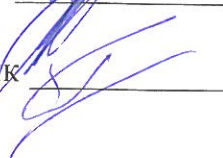
Обладнання та програмне забезпечення:

1. Мова програмування Node.js (v14.17.1);
2. Мова програмування PHP (v8.0.2);
3. Фреймворк Laravel (v9.2);
4. Linux Ubuntu (v20.04.4 LTS);

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Пірог О.В.	20.04.2022	20.04.2022
2	Пірог О.В.	20.04.2022	20.04.2022
3	Пірог О.В.	20.04.2022	20.04.2022

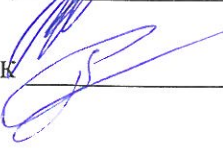
Студент  Юрій Марущак

Керівник  Олександр Пірог

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Видача завдання	01.03.2022	виконано
2	Розробка плану роботи	30.04.2022	виконано
3	Розробка розгорнутого плану	05.05.2022	виконано
4	Подача чернетки керівнику для перевірки	07.05.2022	виконано
5	Доопрацювання роботи та перевірка керівником	09.05.2022	виконано
6	Затвердження графічних матеріалів у керівника	12.05.2022	виконано
7	Проведення попереднього захисту	18.06.2022	виконано
8	Отримання відгуку керівника	21.06.2022	виконано
9	Допуск завідувача кафедри	21.06.2022	виконано
10	Подання роботи на рецензування	21.06.2022	виконано

Студент  Юрій Марущак

Керівник  Олександр Пірог

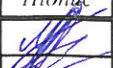
АНОТАЦІЯ

В межах випускної кваліфікаційної роботи було спроектовано сервіс на тему «Проект підсистеми захисту Web-орієнтованого сервісу довідок».

Пояснювальна записка до випускної кваліфікаційної роботи містить в собі 98 сторінок та 70 рисунків. Список використаних джерел містить 13 найменувань і займає 2 сторінки.

В роботі визначено основні завдання на розробку підсистеми, яка забезпечить захист конфіденційних даних користувача, електронних зразків документа від підробки та фальсифікації. Проаналізовано веб-вразливості та створені захисні конфігурації проти них. Розвернуто та налаштовано веб-сервер NGINX на базі якого реалізований захист від XSS та CSRF атак та витоку чутливої інформації, створені конфігурації внутрішніх сервісів, а саме: API на базі php фреймворку Laravel, який забезпечує захист від найбільш розповсюджених веб-атак на рівні додатку, реалізовані middlewares, troglings посередники для забезпечення стійкості API, написано Blockchain на мові програмування Node.js, який забезпечує верифікацію виданих документів, гарантує цілісність та безпеку від несанкціонованих змін хеш сум всередині блокчейну, також реалізований клієнтський застосунок на базі Nuxt.js для забезпечення комфортного користування сервісом та забезпеченням зв'язку з бекенд частиною. Наведені сценарії захисту від можливих атак, проілюстрована загальна побудова проекту, представлений опис версій програм, що будуть використані при побудові даного проекту. Також представлені результати тестування на вразливість до атак.

Ключові слова: ВЕБ-СЕРВЕР, ВРАЗЛИВІСТЬ, КОНФІГУРАЦІЯ, БРАНДМАУЕР, WAF, ЗАХИСТ, RBAC, IPTABLES, WEB-ДОДАТОК, АТАКА, BLOCKCHAIN.

					КІК.КР.Б – 125 – 12 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	ПРОЕКТ ПІДСИСТЕМИ ЗАХИСТУ WEB-ОРІЄНТОВАНОГО СЕРВІСУ ДОВІДОК	Літ.	Арк.	Аркуші
Розроб.		Марущак Ю.Р.		19.06				
Керівник		Пірог О.В.		19.06				
Реценз.		Сфіменко А.А.		20.06				
Н. контр.		Сфіменко А.А.		20.06				
Затверд.		Сфіменко А.А.		20.06				

ANNOTATION

As part of the final qualification work, a service on the topic "Project of the subsystem for the protection of Web-oriented help service" was designed.

The explanatory note to the final qualification work contains 98 pages and 70 figures. The list of used sources contains 13 names and occupies 2 pages.

The paper identifies the main tasks for the development of a subsystem that will protect confidential user data, electronic samples of documents from forgery and falsification. We analyzed web vulnerabilities and created protective configurations against them. Deployed and configured NGINX web server based on protection against XSS and CSRF attacks and leaks of sensitive information, created configurations of internal services, namely: API based on php framework Laravel, which provides protection against the most common web attacks at the application level, implemented middlewares, trotlings intermediaries to ensure the stability of the API, written by Blockchain in the programming language Node.js, which provides verification of issued documents, ensures integrity and security against unauthorized changes in hash amounts within the blockchain, also implemented a client application based on NUXT.js for comfortable use service and communication with the backend part. Scenarios of protection against possible attacks are given, the general construction of the project is illustrated, the description of the versions of the programs which will be used at construction of the given project is presented. The results of testing for vulnerability to attacks are also presented.

KEYWORDS: WEB SERVER, VULNERABILITY, CONFIGURATION, FIREWORK, WAF, PROTECTION, RBAC, IPTABLES, WEB APP, ATTACK, BLOCKCHAIN.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП.....	9
РОЗДІЛ 1. ТЕОРЕТИЧНІ ПОНЯТТЯ ПРО ВЕБ-СЕРВЕРА, API, API TOKENS, BLOCKCHAIN, ВЕБ ДОДАТКИ, МЕТОДИ ТА ЗАСОБИ ЇХ ЗАХИСТУ.	11
1.1 ЗАГАЛЬНІ ПОНЯТТЯ ПРО ЗАХИСТ ВЕБ-СЕРВЕРА.....	11
1.1.1 Визначення поняття безпеки веб-сервера.....	11
1.1.2 Основні шляхи захисту веб-сервера.....	11
1.2 ТЕХНОЛОГІЇ ЗАХИСТУ ВЕБ-СЕРВЕРУ ТА ЙОГО ВЕБ-ЗАСТОСУНКІВ.....	14
1.2.1 SSL/HTTP/TLS	14
1.2.2 Брандмауер ОС	15
1.2.3 WAF.....	16
1.3 ЗАГАЛЬНІ ПОНЯТТЯ ПРО ЗАХИСТ WEB API ІНТЕРФЕЙСУ ТА ЙОГО API ТОКЕНІВ	19
1.3.1 Визначення поняття безпеки WEB API та API токенів.....	19
1.3.2 Технології захисту API та забезпечення конфіденційності токенів	19
1.4 ВЕБ ДОДАТКИ ТА ЇХ ВРАЗЛИВОСТІ.....	21
1.4.1 Визначення веб-додатку	21
1.4.2 Основні вразливості веб-додатків.....	22
1.4.3 Засоби захисту веб-додатків від найбільш розповсюджених видів атак 25	
1.5 ЗАГАЛЬНІ ПОНЯТТЯ ПРО БЛОКЧЕЙН	26
1.5.1 Особливості безпеки блокчейну	27
1.5.2 Основні шляхи використання блокчейну	28
Висновок до першого розділу	29
РОЗДІЛ 2. РОЗРОБКА ТЗ ТА РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ВРАХОВУЮЧИ ЙОГО ПІДСИСТЕМУ ЗАХИСТУ	30
2.1 РОЗРОБКА ТЕХНІЧНОГО ЗАВДАННЯ	30
2.1.1 Визначення основних цілей технічного завдання, створення діаграми проекту30	
2.1.2 Створення діаграми бази даних	31
2.2 НАЛАШТУВАННЯ СЕРВЕРУ NGINX.....	34
2.2.1 Налаштування конфігураційних файлів.....	34
2.2.2 Брандмауер iptables у поєднанні з fail2ban	39
2.2.3 NAXSI/ModSecurity	42
2.3 НАЛАШТУВАННЯ API СЕРВІСУ.....	49
2.3.1 Налаштування API маршрутів, Middlewares, Trotting servісу.....	50
2.3.2 Налаштування API токенів, авторизації та автентифікації.....	53
2.3.3 Налаштування RBAC сервісу.....	56
2.4 НАЛАШТУВАННЯ BLOCKCHAIN	57
2.4.1 Створення Blockchain, Block, Document сутностей	57
2.4.2 CRYPTO-JS та ELLIPTIC	61

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

2.4.3	Налаштування HTTP серверу на Node.JS	63
	Висновок до другого розділу	64
РОЗДІЛ 3. ТЕСТУВАННЯ ТА РЕАЛІЗАЦІЯ МІР ЗАХИСТУ ВЕБ ДОДАТКУ		66
3.1	БАЗОВІ НАЛАШТУВАННЯ КОНФІГУРАЦІЙ NGINX	66
3.1.1	Приховування чутливої інформації, CSP, HSTS	66
3.2	НАЛАШТУВАННЯ HTTPS.....	70
3.2.1	Certbot	70
3.2.2	Доступ до HTTPS через брандмауер	72
3.3	ТЕСТУВАННЯ НА XSS АТАКИ	72
3.4	ВИКРАДЕННЯ ТОКЕНУ КОРИСТУВАЧА.....	75
3.5	ПІДРОБКА ДОКУМЕНТУ	77
3.6	ЗДІЙСНЕННЯ НЕПРАВОМІРНИХ ДІЙ СТОСОВНО БЛОКЧЕЙНУ	79
	Висновок до третього розділу	80
ВИСНОВКИ		81
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....		83
ДОДАТКИ.....		85

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних

РС – Робоча станція

СУБД – Система управління базами даних

ЦС – Центр сертифікації

CSP – Content Security Policy

CSRF – Cross-site Request Forgery

HSTS – HTTP Strict Transport Security

HTTP – HyperText Transfer Protocol

HTTPS – HTTP Secure

SQLi – SQL Injection

SSL – Secure Sockets Layer

TLS – Transport Layer Security

WAF – Web Application Firewall

XSS – Cross-site Scripting

RBAC - Role Based Access Control

API - Application Programming Interface

MVC – Model View Controller

ECDSA – Elliptic Curves Digital Signature Algorithm

ECC – Elliptic Curve Cryptography

DOM – Document Object Model

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Останнім часом доводиться все більше витратити особистого часу на вирішення паперових питань. У середньому, для отримання будь-якої довідки необхідно від однієї години до декількох днів, якщо взяти до уваги, коли людина знаходить за межами територіальної громади, яка видає документ то це може тривати навіть і більше. Також не менш важливим є кількість вдалих атак на веб-орієнтовані сервіси, через непрофесіоналізм організацій дуже часто під загрозою цілісності та конфіденційності персональних даних користувачів.

Існування зручних, швидкодоступних гарно захищених сервісів є безціним, адже саме вони вирішують питання витрати часу, конфіденційності, цілісності інформації користувача. Оформлення будь-якого документу це часовитратне завдання, тому краще покластись на автоматизовані та підлаштовані під цю справу веб-системи, які забезпечать цілісність та невідомість ваших онлайн документів. Згідно вищенаписаного можна сказати, що тема роботи направлена на створення проекту підсистеми захисту Web-орієнтованого сервісу довідок є **актуальною**.

Методи дослідження. В кваліфікаційній роботі було використано такі методи: експерименту, позначення проблеми, узагальнення досвіду, системного аналізу та зіставлення.

Мета і завдання роботи. Метою роботи є проектування підсистеми захисту Web-орієнтованого сервісу довідок.

Досягнення поставленої мети передбачає розв'язання наступних завдань:

- Проведення аналізу компонентів веб-безпеки їх архітектури та принципів роботи, як одного цілого;
- Аналізу наявних безпекових проблем пов'язані з веб-безпекою;
- Розробка веб-додатку підсистеми захисту Web-орієнтованого сервісу довідок.
- Проведення тестування на додавання зловмисного коду та фальсифікації інформації електронних документів, створення шаблонів та конфігурацій, щоб запобігти протестованим веб-небезпекам.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

– Перевірка ефективності створеного коду та шаблонів безпеки за допомогою тестування.

Об’єктом дослідження є підсистема захисту web-орієнтованого сервісу довідок з використання технологій Laravel, Node.js, Nuxt.js, Blockchain.

Предметом дослідження є технології проектування підсистеми захисту web-орієнтованого сервісу довідок з використанням технологій Laravel, Node.js, Nuxt.js та Blockchain.

Практичне значення роботи полягає у наступному:

– Розроблений Web-орієнтований сервіс довідок з підсистемою захисту, яка здатна забезпечити цілісність та конфіденційність даних користувачів, надати можливість електронного документообігу, виявити неправомірні дії стосовно створених документів.

Структура роботи. Робота складається зі вступу, трьох розділів, висновків, списку використаних літературних джерел та додатків. Основний зміст роботи викладено на 82 сторінках друкованого тексту. Робота містить 70 рисунків. Список використаних джерел налічує 13 джерел, що викладені на 2 сторінках. Робота містить 2 додатки.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

РОЗДІЛ 1. ТЕОРЕТИЧНІ ПОНЯТТЯ ПРО ВЕБ-СЕРВЕРА, API, API TOKENS, BLOCKCHAIN, ВЕБ ДОДАТКИ, МЕТОДИ ТА ЗАСОБИ ЇХ ЗАХИСТУ.

1.1 Загальні поняття про захист веб-сервера

1.1.1 Визначення поняття безпеки веб-сервера

Безпека веб-сервера має на увазі інструменти, технології і процеси, які забезпечують безпеку інформації (ІБ) на веб-сервері. Цей широкий термін охоплює всі процеси, які забезпечують роботу певного Інтернет-сервера відповідно до політики безпеки.

Безпека веб-сервера – це безпека будь-якого сервера, розгорнутого в домені всесвітньої павутини (приклад “Принцип функціонування веб-серверу” показано на рис.1.1.1) або в Інтернеті. Вона реалізується кількома методами та на кількох рівнях, як правило, включаючи рівень безпеки базової операційної системи (ОС), прикладний рівень безпеки та рівень безпеки мережі. Безпека ОС, яка забезпечує доступ лише авторизованим користувачам, керує критичними компонентами та службами веб-сервера. Безпека прикладного рівня забезпечує контроль над вмістом і послугами, розміщеними на веб-сервері. Безпека мережі забезпечує захист від зловживань безпеки в Інтернеті, вірусів і атак.

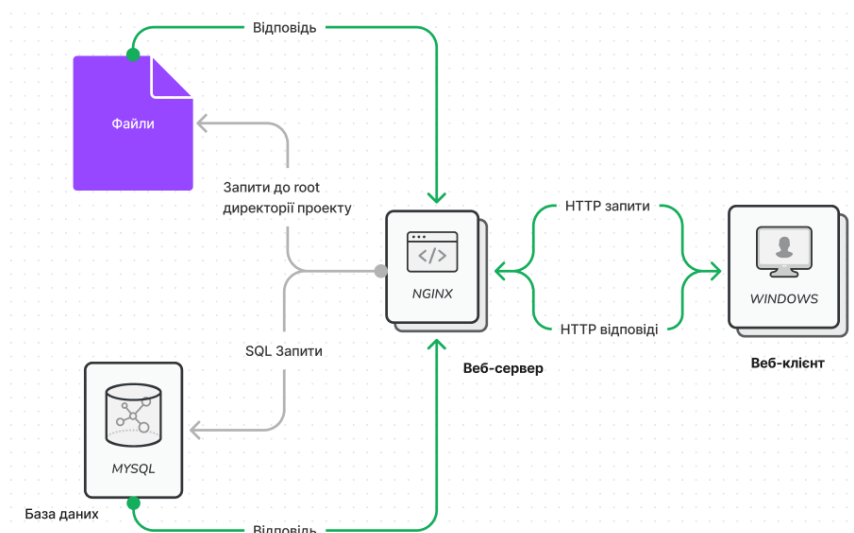


Рисунок 1.1.1 - Принцип функціонування веб-сервера

1.1.2 Основні шляхи захисту веб-сервера

Коли стоїть першочергове завдання захисту систем від різних атак та загроз, на думку приходять типові рішення, такі як брандмауери, шифрування,

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

використання останніх версій програм. Однак до захисту веб-серверу необхідно підходити не стандартно, за своєю унікальною структурою, веб-сервер може бути уразливий до багатьох атак (приклад “Стек вразливостей веб-сервера” показано на рис.1.1.2). Стек вразливостей веб-сервера безпосередньо залежить від застосунків, які є на сервері, бізнеслогіки, налаштування самого серверу, налаштування мережі, будь-яке програмне забезпечення, встановлене на Web-сервері, може бути використане злочинцем для проникнення в систему.



Рисунок 1.1.2 - Стек вразливостей веб-сервера

Веб-сервер робить вразливим не лише сама його реалізація, будь-то Apache, Nginx чи будь-яка інша, але й всі складові веб-застосунку з якими він взаємодіє.

Кожна з цих складових має свої вразливості, що безперечно збільшує кількість векторів атаки на веб-сервер.

Можна виділити загальні способи захисту Web-серверів:

1. Видалення непотрібних послуг

При встановленні системи є багато мережевих служб, включених у стандартну інсталяцію, які не будуть використовуватися, від служб віддаленого реєстру до служби сервера друку та інших функцій.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Чим більше служб працює на серверній операційній системі, тим більше портів залишається відкритим, тобто більше шляхів у мережу, якими може скористатися зловмисник.

2. Встановлення дозволів та привілеїв

Дозволи мережевих служб та дозволи на файли відіграють вирішальну роль у безпеці мережі. Якщо ваш веб-сервер зламано через програмне забезпечення мережевої служби, зловмисник може використовувати будь-який обліковий запис, який запущено мережевою службою, для виконання завдань.

Цей простий акт встановлення мінімальних привілеїв для користувачів для доступу до файлів веб-програм і серверних баз даних може сприяти запобіганню втрати або маніпуляції з даними.

3. виправлення вад у встановленому програмному забезпеченні

Нездатність підтримувати програмне забезпечення в актуальному стані з останніми може дозволити кіберзлочинцям дізнатися шляхи проникнення до вашої мережі шляхом зворотної інженерії.

4. Відокремлення та відстежування журналів сервера

У рамках регулярного тестування безпеки краще зберігати журнали сервера окремо, а також відстежувати й часто перевірити їх. Незвичайні записи у файлі журналу розкривають інформацію про спроби та успішні атаки, і їх слід досліджувати по мірі їх виникнення.

5. Встановлення брандмауєру

Програмні брандмауєри легко налаштовувати та керувати, вони захистять веб-сервер від несанкціонованого зв'язку та вторгнень.

6. Автоматизація резервного копіювання

Регулярне резервне копіювання сервера гарантує, що, якщо безпеку буде порушено, буде можливість швидко відновити дані. Автоматизація може підвищити ефективність, але ІТ-співробітник повинен перевірити, чи немає проблем, які могли перервати процес копіювання.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

1.2 Технології захисту веб-серверу та його веб-застосунків

При забезпеченні захисту Web-сервера від зловмисника можна виділити 3 основні технології: SSL/HTTP/TLS, Брандмауер ОС, WAF.

1.2.1 SSL/HTTP/TLS

Протоколи криптографії надають можливість встановлення захисних з'єднань між двома видаленими хостами.

Два найбільш часто зустріваних набори протоколів – SSL/TLS, основна задача яких полягає в забезпеченні конфіденційності переданих даних, забезпеченні цілісності даних (тобто гарантування того, що дані не були змінені або замінені під час доставки між вузлами), а так же в забезпечення ідентифікації та аутентифікації ресурсів і хостів з використанням цифрових сертифікатів.

Кожен протокол SSL/TLS включає в себе набір внутрішніх протоколів (підпротоколів).

Задачі SSL/TLS з'єднання:

- забезпечення приватності за допомогою шифрування переданих даних;
- аутентифікація за допомогою сертифікатів;
- забезпечення достовірності даних за допомогою перевірки цілісності даних.

HTTPS є безпечним розширенням HTTP. Веб-сайти, які встановлюють та налаштовують сертифікат SSL/TLS можуть використовувати протокол HTTPS для встановлення безпечного з'єднання із сервером.

Ключовий момент встановлення захищеного з'єднання по HTTPS – SSL Handshake (приклад “SSL Handshake” показано на рис.1.2.1), або «рукостискання», у процесі якого здійснюється узгодження використовуваної версії TLS, перевірка вузлів (автентифікація сервера та додаткового клієнта), алгоритмів шифрування, обмін ключами, сертифікатами та власне встановлення та початок обміну вже зашифрованими даними.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

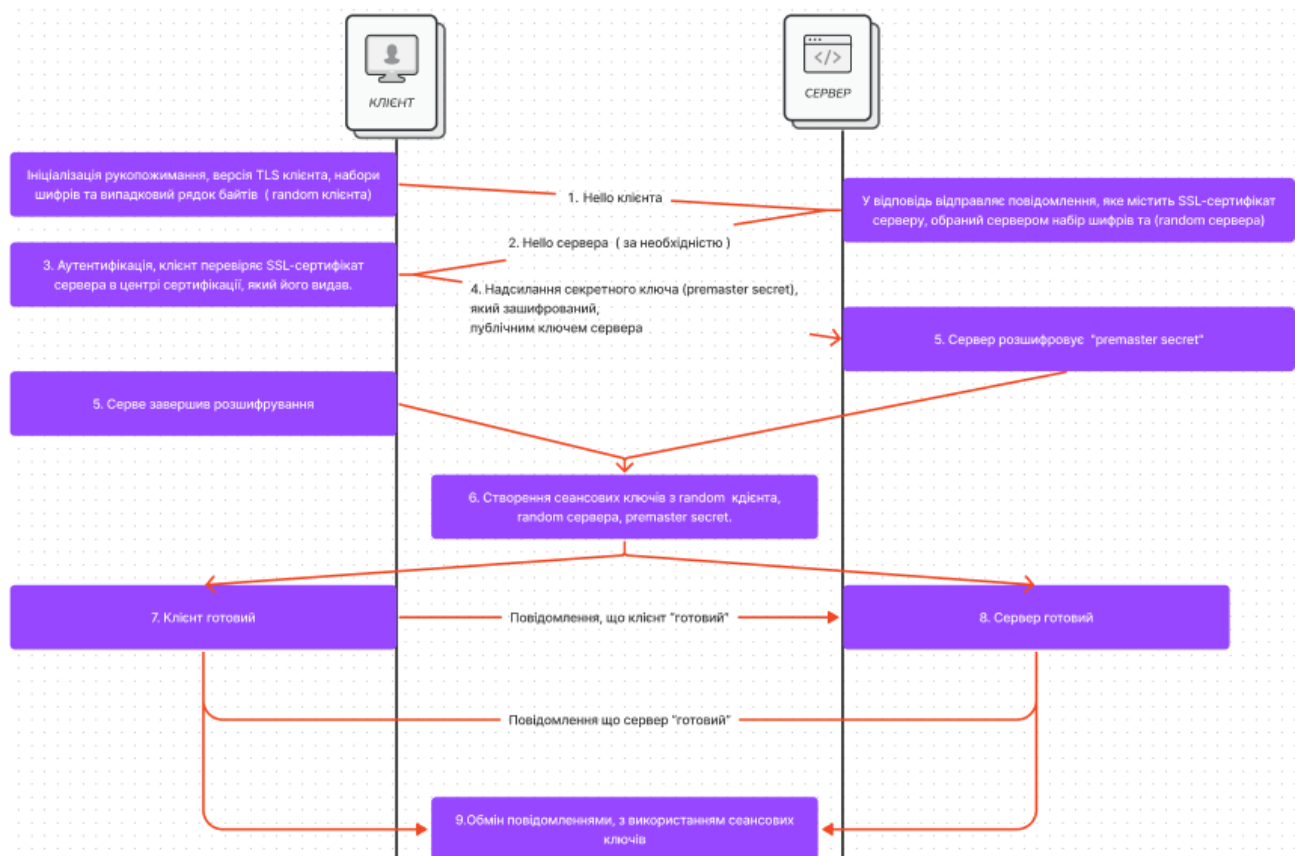


Рисунок 1.2.1 – SSL Handshake

Зазвичай SSL сертифікати, завірені авторитарними центрами сертифікації підтверджують надійність шифрування встановленого з'єднання. Самопідписані сертифікати також шифрують зв'язок між сервером і будь-якими клієнтами. Однак, оскільки вони не підписані жодним із надійних центрів сертифікації, включених до веб-браузерів, користувачі не можуть використовувати сертифікат для автоматичної перевірки ідентичності сервера.

Самопідписаний сертифікат може бути доречним, якщо відсутнє доменне ім'я, пов'язане з сервером, і для випадків, коли зашифрований веб-інтерфейс не призначений для користувача. Якщо доменне ім'я наявне, у багатьох випадках краще використовувати сертифікат, підписаний ЦС.

1.2.2 Брандмауер ОС

Брандмауер – це, по великому рахунку, набір правил. Коли пакет даних переміщується до захищеного мережевого простору або виходить із нього, його вміст (зокрема, інформація про його походження, його призначення та протокол, який він планує використовувати) перевіряється на відповідність правилам

брандмауера, щоб побачити, чи слід дозволити доступ (приклад “Використання брандмауера ОС” показано на рис.1.2.2).

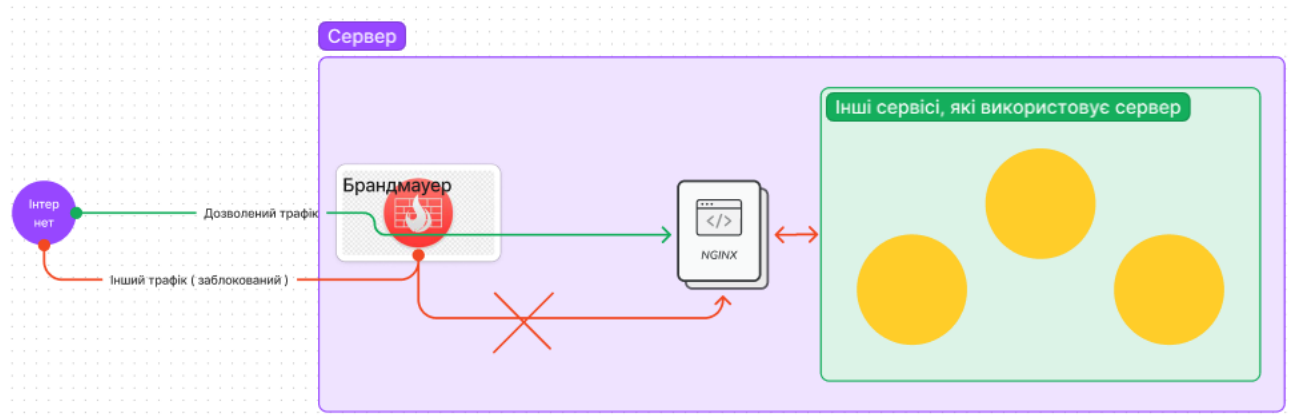


Рисунок 1.2.2 – Приклад використання брандмауера ОС

Функціональність брандмауера також доступна через апаратні пристрої, вироблені такими компаніями, як Juniper і Cisco. Ці пристрої працюють на власних ОС з унікальним синтаксисом і дизайном. Для більших корпоративних рішень із залученням сотень серверів, розподілених у кількох мережах, такі інструменти є оптимальним рішенням.

Брандмауери різних ОС PC надаються аналогічний набір правил, але зазвичай, при великій завантаженості серверу, краще покласти роль брандмауера на спеціальний мережевий пристрій.

1.2.3 WAF

Організації та користувачі все більше покладаються на веб-додатки (наприклад, веб-портали, корпоративні веб-програми, веб-рішення для автоматизації бізнесу, веб-програми для електронної комерції тощо). WAF відстежує запити HTTP/HTTPS і захищає ці веб-програми від шкідливих дій на рівні 7 моделі OSI. Отже, WAF є необхідним захистом від зростаючого числа загроз веб-безпеці.

Основна відмінність між визначенням брандмауера та брандмауера веб-додатків (WAF) полягає в тому, що брандмауер зазвичай захищає мережний і транспортний рівні (рівні 3 і 4). WAF забезпечує захист на прикладному рівні (рівень 7).

WAF зазвичай логічно розміщується між користувачами та веб-серверами (приклад “Структура та принцип роботи WAG” показано на рис.1.2.3), аналізує та порівнює мережевий трафік з базою даних вразливостей. WAF створює набір правил, призначених для захисту веб-сайту та виявлення небажаного трафіку. Зазвичай він блокує цей трафік, але також його можна налаштувати лише на моніторинг.

WAF може блокувати наступні атаки:

- Поширені атаки на веб-додатки та методи введення коду:

Ін’єкція SQL (SQLi):

Вона можна бути здійснена після введення шкідливого код у оператори SQL через веб-сторінку (наприклад, користувач дає інструкцію SQL, яка несвідомо запускатиметься у БД). Цей шкідливий код може змінювати, красти або видаляти дані БД.

Міжсайтові сценарії (XSS):

Шкідливий сценарій впроваджується в код (наприклад, HTTP, JavaScript тощо) надійного веб-сайту, що дозволяє отримати доступ до потенційно конфіденційних даних користувача, наприклад файлів cookie. Код, змінений цією атакою, виконується не на сервері, а на стороні користувача.

Віддалене включення файлів (RFI):

RFI – це процес вбудовування зовнішніх файлів через уразливості, реалізовані у веб-додатку. Якщо процес дозволяє змінити шлях до обробленого файлу (наприклад, якщо шлях включено як параметр), зловмисник може використовувати цей шлях для введення зовнішнього шкідливого файлу.

- DDoS-атаки на рівень 7 (HTTP Flood):

Вони складаються із запитів (популярні HTTP GET і DNS-запити), які призначені для споживання ресурсів програми (пам’ять, процесор, пропускна здатність). Прикладом може служити зловмисник, який постійно використовує функціональні можливості веб-сайту (надання контактної форми чи будь-яких запитів API), які, як їм відомо, спричиняють обробку БД та програм, тому базова

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

веб-служба зайнята шкідливими запитами і більше не може обробляти запити інших користувачів.

- Зловмисні боти:

Зловмисних ботів часто програмують на виконання різноманітних шкідливих завдань. Вони можуть спробувати зламати облікові записи користувачів, викрасти дані, подати безглузді дані через онлайн-форми та виконувати інші шкідливі дії. Погана активність ботів найчастіше проявляється аномальним збільшенням або зменшенням відвідувань у незвичні періоди з високим показником негайної відмови.



Рисунок 1.2.3 – Структура та принцип роботи WAF

Можна виділити наступні переваги WAF:

- Доповнює інші захисні системи, такі як брандмауери та системи запобігання вторгненням.
- Зменшить витрати на хмарну безпеку, уникаючи потреби в дорогому спеціалізованому обладнанні або персоналі з IT-безпеки.
- Фільтрує та відстежує трафік на прикладному рівні (рівень 7), що неможливо з будь-яким іншим типом брандмауера.
- Запобігає несанкціонованому переміщенню конфіденційних даних за межі програми.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

- Понизить ризик простою, крадіжки даних та порушення безпеки.
- WAF можна масштабувати для захисту від найбільших DDoS-атак на рівні 7.

1.3 Загальні поняття про захист Web API інтерфейсу та його API токенів

1.3.1 Визначення поняття безпеки WEB API та API токенів

При обговоренні безпеки WEB API, варто зауважити важливу річ, що воно надає інтерфейс для WEB додатку, тому відразу необхідно думати про ІБ на двох рівнях, а саме: доступ до API та доступ до додатку. Безпека API на теперішній час, дуже важливе питання, тому що звернення до WEB API складають приблизно 80% всього WEB-трафіку, тому кіберзлочинці все частіше орієнтуються на API-інтерфейси.

Тому у API дуже важлива правильна аутентифікація, авторизація, права доступів та інші критичні модулі, які стосуються чутливої інформації, яка передається, змінюється, видаляється, створюється через API інтерфейс.

Контроль доступу до API переважно здійснюється за допомогою спеціально виданого API токenu (приклад “Принцип функціонування API” показано на рис.1.3.1).

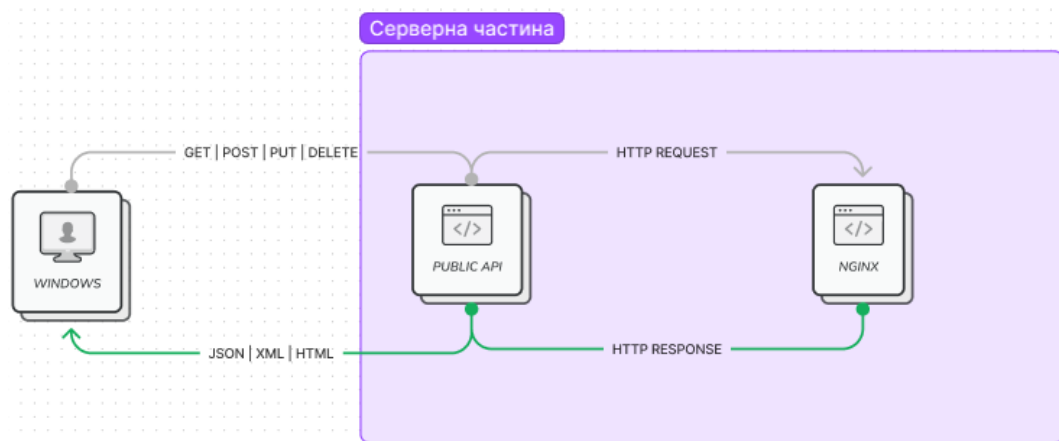


Рисунок 1.3.1 – Принцип функціонування API

1.3.2 Технології захисту API та забезпечення конфіденційності токенів

Вирішення безпекових питань, які стосуються API це доволі громізка робота, яка залежить від багатьох технологій, які використовуються у конкретному проєкті, проте можна виділити такі пункти: безпека з’єднання, контроль доступу до API, використання ключів API, клієнтське обмеження API.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

Безпека API починається з самого з'єднання HTTP. API-інтерфейси повинні надавати лише кінцеві точки HTTPS, щоб гарантувати, що всі повідомлення, які проходять через API шифруються з використанням SSL/TLS, це дозволяє клієнтам аутентифікувати службу, яка видала ключ безпеки і захищає дані API та інформацію, яка передається по ньому.

Контроль доступу до API, більшість сервісів доступні для аутентифікованих користувачів, тому що вони є приватними або потребуються реєстрації або оплати. Найбільше розповсюдженим методом аутентифікації WEB API:

1. Базова аутентифікація HTTP (HTTP Basic Authentication): данні користувача відправляються безпосередньо в заголовках HTTP в Base64 кодуванні без шифрування. Це найпростіший метод аутентифікації і найменш безпечний, оскільки чутливі дані передаються в вигляді просто тексту, тому його варто використовувати лише разом з HTTPS.
2. Веб токени JSON (JSON Web Tokens – JWT): облікові дані та інші параметри доступу відправляються у вигляді структур даних JSON. Ці токени можуть бути підписані криптографічно і є переважним способом керування доступом до API.
3. OAuth: стандартний механізм OAuth 2.0 можуть використовуватись для аутентифікації та авторизації. Цю технологію використовуються такі компанії як Google, Github, Bitbucket та інші, вона дозволяє здійснювати авторизацію клієнта через стороній сервіс, або власноруч налаштувати авторизацію через ваш сервіс сторонім додатком, має власні приватні та публічні криптологічно ключі, для шифрування API токена.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

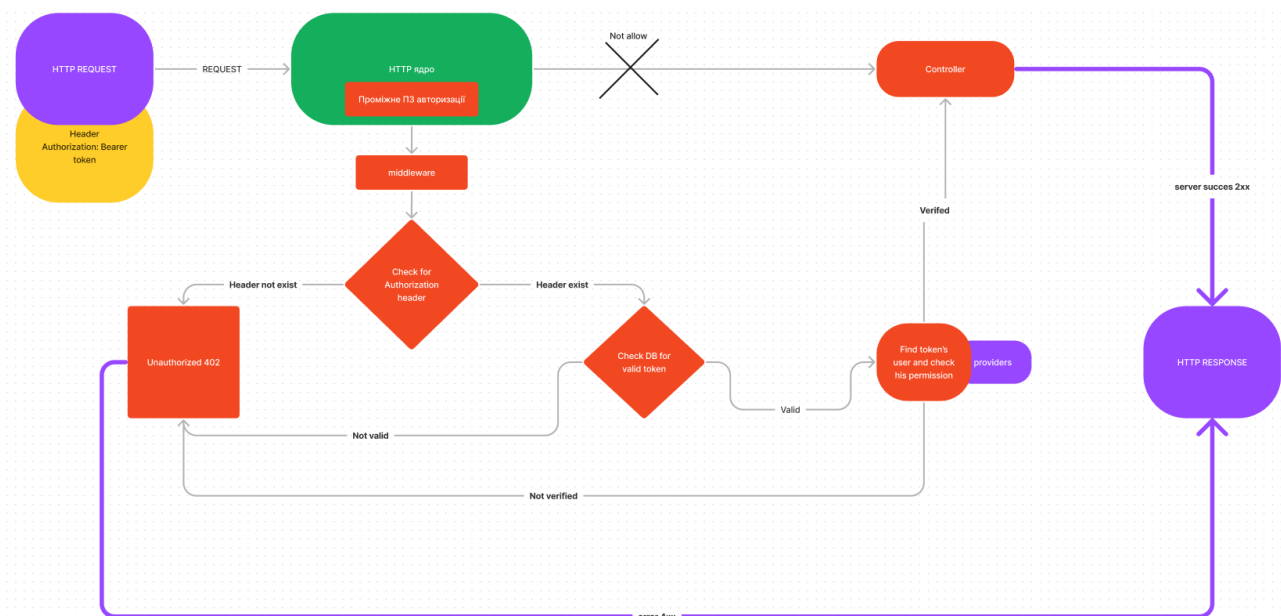


Рисунок 1.3.2 – Принцип роботи API токенів

Авторизація користувачів за допомогою ключів API (приклад “Принцип роботи API токенів” показано на рис.1.3.2), це спосіб керуванням доступом до загальнодоступних служб API для вимушеного обмеження швидкості викликів API і зменшення шкоди від атак типу “відмова в обслуговуванні”.

Клієнтські обмеження API, блокування використання служб, які не використовуються сервісом, починаючи з обмеження підтримки методів HTTP, щоб переконатись, що кіберзлочинці не можуть виконувати ніяких дій, які виходять за межі специфікації API і дозволеного рівня доступу. Наприклад, якщо API дозволяє лише запити GET, POST то інші повині бути відхилені з кодом 405 Method not allowed.

1.4 Веб додатки та їх вразливості

1.4.1 Визначення веб-додатку

Веб-додаток - це програмне забезпечення або програма, яку можна відкрити за допомогою будь-якого браузера. Зовнішній інтерфейс веб програми розробляється за допомогою таких мов програмування: HTML, CSS, Javascript, які підтримуються на будь-якому браузері (Opera, Chrome, Mozilla). У той час як для написання серверної частини (Back-end) може використовуватися будь-яка інша мова програмування або фреймворк, Python, PHP, Ruby, Java.

Основні переваги веб-застосунків:

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

1. веб-додатки можуть застосовуватися на будь-якій операційній системі (Linux, Mac, Windows), оскільки всі вони підтримують сучасні браузери
2. у зв'язку з тим, що у веб-додатку використовується той самий код порівняно з desktop додатками їх набагато легше підтримувати
3. додаток простіше програмувати оскільки він не включає багато роботи з елементами ПК(ядро,процесор, відеокарта)
4. на відміну від мобільних додатків, для веб-додатків не потрібно схвалення жодних платформ, щоб випустити свою програму
5. веб додатки це більш економний варіант для будь-якого підприємства оскільки веб-програми не вимагають підписки або покупки ліцензій, а можуть використовуватися як SaaS-сервіс, що значно дешевше

1.4.2 Основні вразливості веб-додатків

Вразливість Web-додатків залишається одним з найбільш поширених недоліків забезпечення захисту інформації. Серед різноманітної кількості небезпек можна виділити наступні

1. Низький рівень кваліфікації співробітників у питаннях з ІБ
2. Слабка парольна політика або повсюдне її невиконання, що призводить до зламу облікових акаунтів користувачів
3. Застаріле ПЗ, не дотримання своєчасного оновлення програм та версій веб додатків
4. Використання нещепний конфігурацій систем
5. Неефективне міжмережеве розмежування доступу
6. Використання сторонніх веб-рішень, недооцінка реалізації безпекових питань у них, що є серйозним ризиком загрози та вектором атаки для кіберзлочинця

Проаналізувавши наукову та спеціалізовану літературу було визначено, що всі загрози інформаційної безпеки Web-додатків можна розділити на кілька категорій:

1. Міжсайтовий скриптинг – XSS (атака на користувача спрямована на виконання в його браузері довільного сценарію), при завантаженні сторінки

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

браузер автоматично виконує код, який може бути спрямований на вилучення даних із форм для передачі персональних даних користувача зломиснику, або ж для проведення DDoS-атаки. Зломисник здійснює атаку не на пряму, а за допомогою вразливостей веб-сайту впроваджуючи свій спеціальний Java Script код. У користувачів цей код відображається як частина сайту.

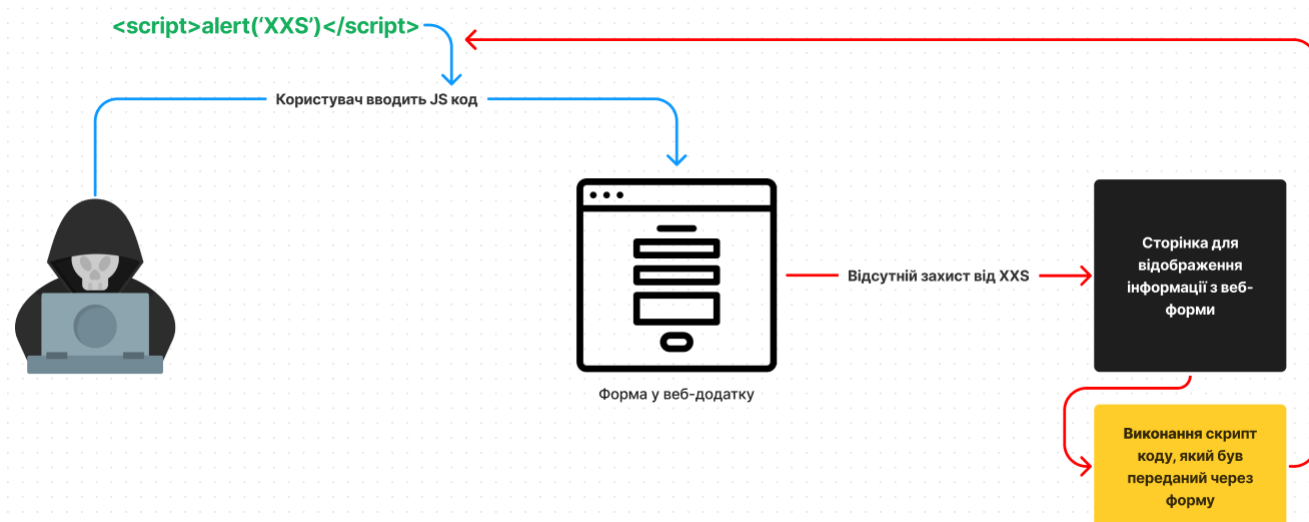


Рисунок 1.4.2 – Приклад XSS атаки

- SQL-ін'єкція – впровадження SQL-коду (приклад “SQL ін'єкція” показано на рис. 1.4.3) в тіло HTTP-запиту до веб-додатку. Коли HTTP запит використовує Query параметр, URL рядка для виконання певних SQL запитів можна здійснити підміни SQL-запиту до бази даних і дізнатися необхідну інформацію. SQL-ін'єкція типу: `SELECT field_1, field_2, field_3, field_4 FROM table WHERE field_1 =-1 UNION SELECT 1,username,password,1 FROM admin` - оскільки запису з ідентифікатором -1 завідомо не існує, з таблиці не буде вибрано жодного запису, проте в результат потраплять записи, несанкціоновано відібрані з таблиці admin внаслідок ін'єкції SQL, що призведе до витoku конфіденційних даних.

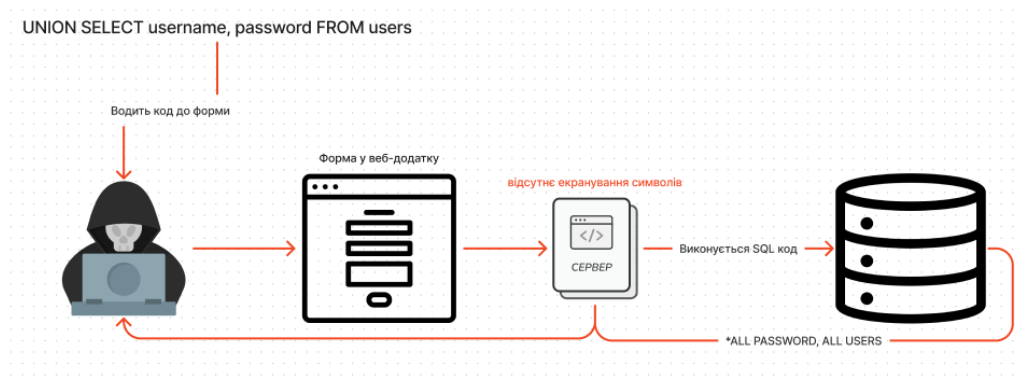


Рисунок 1.4.3 – Приклад SQL ін'єкції

3. CRLF-атака – техніка модифікації HTTP-заголовків запиту. Яка може призвести до різних атак, такі як HTTP Request Smuggling (крадіжка запиту) та HTTP Response Splitting.

1. Якщо говорити про HTTP Request Smuggling, це зазвичай відбувається, коли HTTP-запит проходить через сервер, який обробляє його і передає іншому серверу, як проксі або фایрволл. Цей тип вразливості може призвести до:

1.1 Отруєння кешу, ситуації, в якій атакуючий може змінювати записи в кеші і віддавати шкідливі сторінки (наприклад, що містять javascript) замість коректних;

1.2 Обхід фаєрволу, ситуації, в якій запит може бути створений таким чином, щоб обійти перевірки безпеки, зазвичай включає CRLF і надмірно великі тіла запитів

1.3 Крадіжка запиту, ситуації, в якій атакуючий може вкрати HttpOnly cookies та інформації про HTTP аутентифікацію. Це схоже на XSS, але не потребує взаємодії між клієнтом та хакером

2. Знаючи про HTTP Response Splitting, хакери можуть встановлювати довільні заголовки відповіді, контролювати тіло відповіді або розділяти відповідь повністю, надаючи дві відповіді замість однієї.

4. XXE (XML eXternal Entity)-атаки. При валідації XML-документа парсером (об'єктно-орієнтованою скриптовою мовою програмування, створеною для генерації HTML сторінок на веб-сервері з підтримкою CGI) за допомогою схеми DTD, всі її директиви обов'язково повинні бути виконані. Якщо

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

правилами визначено, що у вхідному документі допускаються спецсимволи XML – ризик бути атакованим дуже великий.

5. CSRF (Cross Site Request Forgery) – є атакою, яка здійснюється в той момент, коли шкідливий сайт, лист, повідомлення, програма або що-небудь інше змушує браузер користувача виконати деякі дії на іншому сайті, де користувач вже автентифікований. Найчастіше це відбувається без відома користувача (приклад “CSRF атака” рис. 1.4.4)

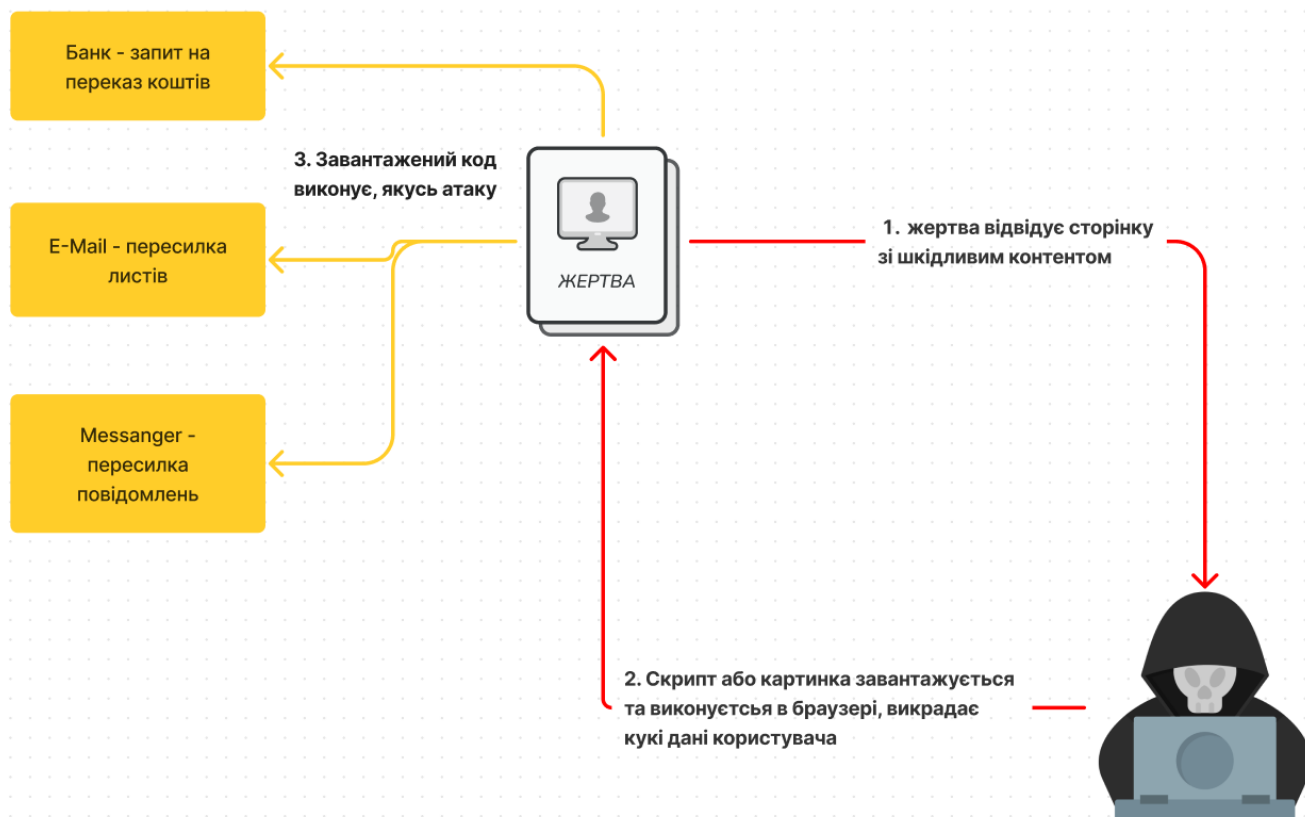


Рисунок 1.4.4 – Приклад CSRF атаки

6. DDoS-атака – велика кількість зовнішніх запитів (часто безглузвих або неправильно сформульованих) з різним трафіком (за різними портами), що викликає переповнення пам’яті та подальшу відмову в обслуговуванні системи, за цей час можна безперешкодно отримати доступ до ресурсу.

1.4.3 Засоби захисту веб-додатків від найбільш розповсюджених видів атак

Існують певні правила за допомогою, яких можна убезпечити свій Web-додаток від найбільш розповсюджених атак, розглянемо їх:

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

1. Міжсайтовий скриптинг – XSS - Заборона на вкладення HTML-сторінок, екранування спецсимволів - <-, -> |, передача всіх cookies з прапором HttpOnly.
2. SQL-ін'єкція - Повсюдне використання ORM в Web-додатку, клієнтська і серверна валідація даних, тестування Web-додатків, екранування «очищених» параметрів, що надходять в неперевірені SQL запити, тестування додатка інструментами на знаходження потенційних SQL-ін'єкцій.
3. CRLF-атака - Обов'язкове кодування CRLF-послідовності до передачі в HTTP заголовки, а також повне кодування переданих даних.
4. XXE (XML eXternal Entity)-атаки - Ретельне налаштування XML-парсерів, використання XML Schema замість DTD для валідуючого парсеру.
5. CSRF (Cross Site Request Forgery) - Встановлення HttpOnly-прапора передачі cookie, використання одноразових сесійних token і відправка прихованої форми, щоб token не можна було підробити, перевірка реферерів при HTTP запити до Web-додатку.
6. DDoS-атака - Виходу з цієї ситуації фактично немає, однак наслідки DDoS-атак і їх ефективність можна істотно знизити за рахунок правильного налаштування маршрутизатора, брандмауера і постійного аналізу аномалій в мережевому трафіку.

1.5 Загальні поняття про блокчейн

На сьогодні широка популярність й поширення технології блокчейн викликає попит на розробку додатків, що працюють на принципах цієї інновації. Повсюдне використання цієї технології робить її лише популярнішою та популярнішою. Її можна зустріти в банківській сфері, державному управлінні, юриспруденції та інших сферах.

Так, наприклад, влітку 2017 року компанії Accenture і Microsoft представили блокчейн технологію цифрового посвідчення особи. Блокчейн стартапи від Civic і UniquID Wallet дозволяють людям створювати цифрові документи, що засвідчують

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

особу, які неможливо підробити - можливо, в майбутньому вони замінять «традиційні» паспорти, водійські права та інші документи.

Саме через свою унікально побудовану систему цю технологію було використано у цьому проекті, в якості гаранта оригінальності, непідробності, засвідчення існування документу, засвідчення організації, яка видала документ та вмісту документу.

Поняття блокчейну трактується по різному, але значення визначень завжди однакове. Технічно, блокчейн на мінімальному рівні – це просто список об’єктів, які містять певну базову інформацію, таку як позначку часу, свій хеш, хеш попереднього блоку, хеші транзакцій та додаткову службову інформацію. Його дані мають бути незмінними та не підлягають зламу.

Головною одиницею блокчейну є блок. У ньому зберігається уся інформація, щодо транзакцій, які відбувались перед створенням блоку. Усі дані по транзакціям використовуються при хешуванні блоку, за допомогою різних алгоритмів хешування. В цих блоках можливо швидко верифікувати інформацію, все залежить від алгоритмів роботи блокчейну. А ще в кожному наступному блоці зберігається інформація про попередній, таким чином утворюючи безперервний ланцюжок, де хеш певного блоку залежить від всіх попередніх блоків, які були додані до блокчейну, таким чином невілюючи можливість зламу та модифікації інформації у блокчейні.

1.5.1 Особливості безпеки блокчейну

Оскільки результат хешування функції SHA-256 непередбачуваний, не існує алгоритму отримання бажаного результату, окрім випадкового перебору, який задовільняє певний умовам. Якщо хеш не задовільняє умовам, то відбувається перерахунок, та довільне змінення блоку, а саме службової інформації в заголовку – хеш перераховується. Після співпадіння варіантів певний вузол відсилає інформацію про блок іншим вузлам у системі, які в свою чергу проводять валідацію блоку, на коректне створення, якщо помилок не виявлено, то блок додається до ланцюжка і наступний блок повинен включити в себе його хеш і все починається спочатку.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

Умисна зміна інформації в будь-якій з копій або навіть у кількох копіях не буде визнана істиною, тобто правдивою, оскільки не буде відповідати правилам.

Тому саме це є одним із факторів захищеності блокчейну, підхід до якого робить блокчейн дуже захищеним способом генерування та передачі даних чи коштів через будь-який канал інтернет-з'єднання.

1.5.2 Основні шляхи використання блокчейну

На базі блокчейну можна будувати не лише щось, що пов'язане з криптовалютами, а будь-який сервіс, ця технологія може лежати в основі будь-чого. Більшість людей не знала про таку технологію як блокчейн приблизно до 2013 року, саме тоді відбувся сплеск зацікавленості до Bitcoin, а так як в основі нього лежить блокчейн, то всі розпочинаються говорити про блокчейн.

Використання цієї технології дуже допоможе у таких сферах як: медицина, страхування, збір пожертвування, сертифікація, автомобілі, голосування тощо.

У автомобілів є одометр, умовно кажучи це лічильник пройденої відстані, який на пряму впливає на ціну автомобіля, чим більше автомобіль проїхав тим він дешевший. Саме тут може допомогти блокчейн, до якого будуть надсилатись показники автомобіля, а саме інформація по одометру, що не дозволить фальсифікувати інформацію лічильника перед продажем (приклад “– Демонстрація роботи блокчейну для автомобіля” рис.1.5.1).

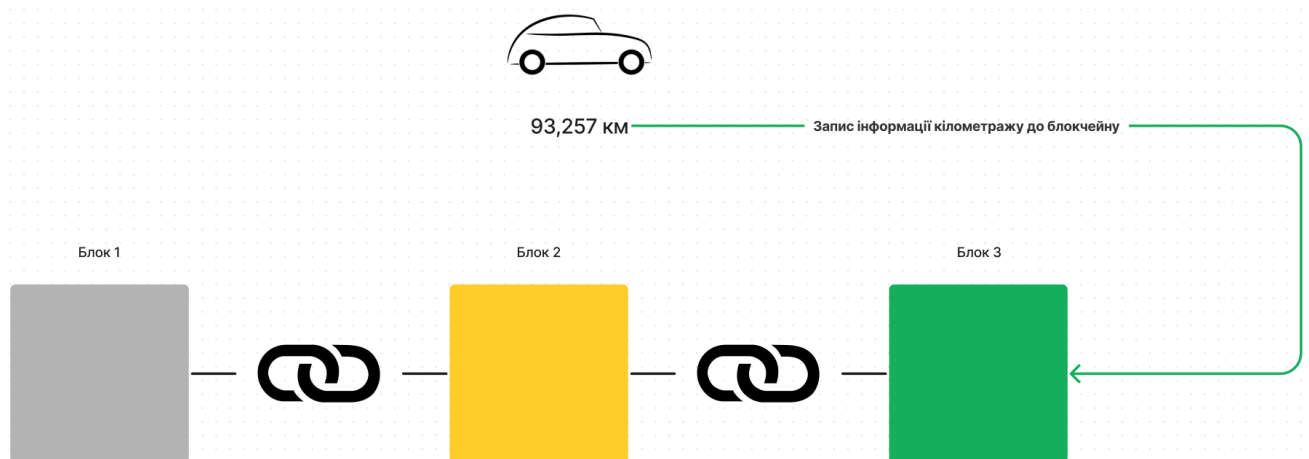


Рисунок 1.5.1 – Демонстрація роботи блокчейну для автомобіля

Голосування досить громізка річ, на теперешній час більшість країн проводить їх у паперовому варіанті, що потребує не малих фінансових витрат, щоб забезпечити всі основні умови для гідного проведення цієї події.

Уявіть, якби голосування проводилось онлайн на платформі, яка використовує блокчейн, де кожен може провести перевірку коректності голосування, де фальсифікація голосів просто неможлива і ніхто не може вплинути на чієсь рішення, саме тому блокчейн має місце і тут, але це досить комплексне завдання, яке вимагає складних рішень в алгоритмах та логіці сервісу. Наприклад, необхідно верифіковувати голос людини без порушення їхнього політики конфіденційності, а саме зберегти анонімними приватні паспортні дані. Якщо дозволяти користувачам голосувати з будь-якого пристрою, є ризик урощення цього пристрою шкідливим програмний забезпеченням, яке може вплинути на подальше голосування. І останнє на мою думку можливість сервісу протистояти DDOS атакам і не підватись відмові у обслуговуванні denial-of-service.

Саме блокчейн може зробити всі ці речі прозорими та чесними.

Висновок до першого розділу

Дослідженні основні шляхи з захисту веб-серверу, Web API, Web-додатків, методи та засоби запобігання найбільш розповсюджених атак на них, а саме було наведено частини вразливого коду Web-додатків до веб-атак, проілюстровано схеми за якими кіберзлочинець може здійснювати неправомірні дії стосовно нашого веб-сервісу. Дослідили технологію блокчейн, а саме особливості захисту та шляхи використання, де ця технологія буде незамінною.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

РОЗДІЛ 2. РОЗРОБКА ТЗ ТА РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ВРАХОВУЮЧИ ЙОГО ПІДСИСТЕМУ ЗАХИСТУ

2.1 Розробка технічного завдання

Перед тим як перейти до реалізації додатку, завжди необхідно скласти технічне завдання.

2.1.1 Визначення основних цілей технічного завдання, створення діаграми проєкту

Під час розробки технічного завдання до проєкта, стояли наступні цілі:

1. Створити сервіс через, який можна зробити онлайн замовлення довідок.
2. Забезпечити безпеку користувацьких даних.
3. Створити зручний веб-інтерфейс.
4. Створити можливість верифікувати документ, чи дійсно певна організація його видала.
5. Валідація документу на спробу модифікації.
6. Створення децентралізованого сховище, яке б зберігало в собі інформацію по всім документам, які були видані організацією.
7. Шифрування на підпис документів організацією.
8. Запобігання найбільш поширеним веб-атакам.

У сервісі будуть використані наступні технології: фронтенд фреймворк NuxtJS, бекенд фреймворк Laravel, бекенд технологія NodeJS, реляційна база даних MySQL, документо-орієнтована база даних MongoDB, децентралізоване сховище даних blockchain (приклад “Діаграма сервісів та додатків, які використовуються у проєкті” рис. 2.1.1).

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

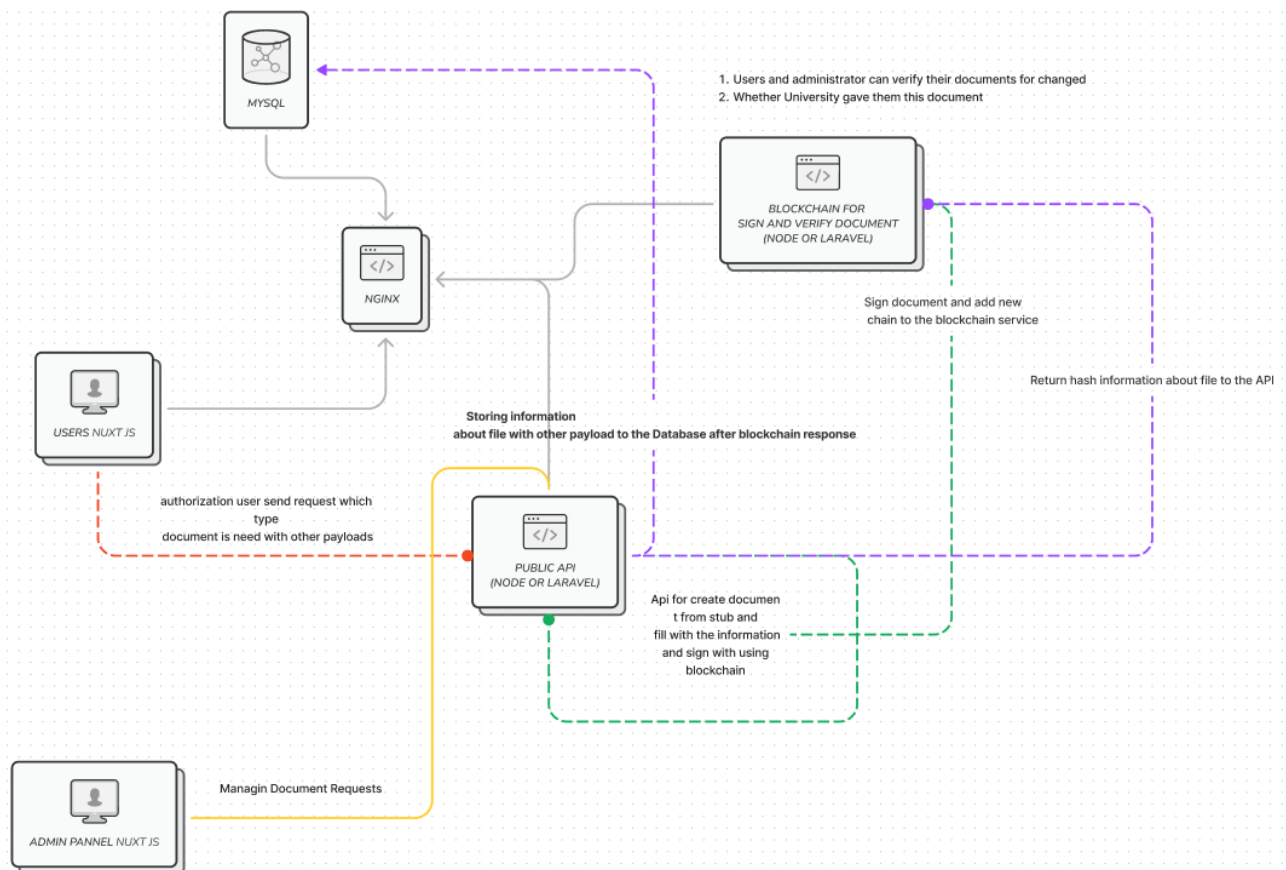


Рисунок 2.1.1 Діаграма сервісів та додатків, які використовуються у проєкті

2.1.2 Створення діаграми бази даних

Основою у цьому сервісі є довідки, але необхідно абстрагуватись від цього та прийти до того, що основною одиницею, яку буде надавати наш сервіс є документ.

Адміністратор може створювати документи, які користувач буде замовляти та створювати замовлення, також присутня сутність користувачі, це наші клієнти, які мають захешований пароль, роль та певні доступи, які їм дозволено системою.

Для автентифікації використано API токени, які володіють посиланням на користувача, також кожному токенові надаються певні доступи, що дозволяє гнучко налаштувати систему RBAC.

У системі створено наступні ролі:

1. Супер адміністратор (super_admin) – користувач з данною роллю не буде мати ніяких обмежень у використанні сервісу, ця роль буде надана лише одному чи двом користувачам, для повного контролю над веб-додатком.

2. Викладач (teacher), студент (student) – ці дві ролі будуть використовуватись виключно для студентів та викладачів, яким буде наданий обмежений список доступів та можливостей у веб-сервесі, користувачі зможуть: робити замовлення на довідку, виконувати зміну особистих даних, відстежувати процес виконання замовлення. За допомогою ролі менежер зможе відразу зрозуміти, який документ на підтвердження особи очікувати від користувача.
3. Менеджер (manager) – адміністративна роль, яка має можливість створювати, редагувати документи, переглядати, виконувати, редагувати замовлення, верифікувати відсканований документ у цифровому просторі блокчейну та всі інші можливості викладача чи студента.

Пропонується створення БД де присутні наступні сутності: certificates, certificate_orders, users, files, roles, personal_access_tokens, permissions. Давайте розберемо деякі з них:

1. Certificates та certificate_orders (приклад “Фрагмент діаграми сутність-зв’язок БД, що відповідає за документи” рис. 2.1.1) –
 - 1.1 Certificates - поле name буде відповідати за назву документу та буде обмежене у кількості символів до 191 шт.), поле slug – відповідає за унікальну рядкову назву, яка ніколи не може повторювати та обмежена до 191 символу. Кожен документ буде мати зразок, за який відповідає поле file_id, це ідентифікатор файла у веб-системі. Також для кращого розуміння термінів виготовлення замовлення присутнє поле duration, яке містити тривалість виконання документу у хвилинах.
 - 1.2 Certificate_orders – таблиця, яка відповідає за саме замовлення, яка містить у собі документ, який замовили поле certificate_id, користувача – user_id, готовий документ, який може приймати значення null або integer_fk який міститься у файловій системі сервісу, канал сповіщення на значення цього каналу channel,

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

channel_value, посилання на документ який засвідчує особу чи то студент, чи то викладач doc_file_id та block_hash – це хеш блоку в системі блокчейну, якщо він є то це означає, що документ пройшов цифрову валідацію у системі блокчейну.

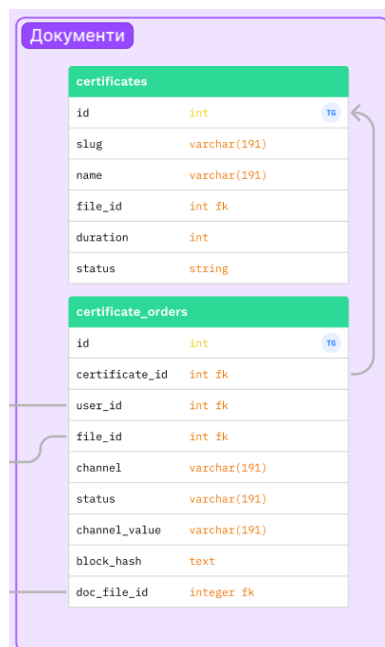


Рисунок 2.1.1 Фрагмент діаграми сутність-зв'язок БД, який відповідає за документи

- Users (приклад “Фрагмент діаграми сутність-зв'язок БД, що відповідає за користувача” рис. 2.1.2) - де користувач має ПІБ, стать, роль, фото за які відповідають відповідні пол first_name, last_name, middle_name, gender, role, file, вони несуть лише інформативну цінність, також у користувача присутній пароль за яке відповідає поле password, при створенні акаунт пароль хешується за допомогою функції password_hash, яка використовує алгоритм blowfish з підсолюванням, яка гарантує кращу безпеку паролю та гарантує, якщо у користувачів два однакових паролі то кінцевий хеш пароль буде різним, навідміну від md5 або sha1, через розвиток обчислювальних потужностей стало тривіальним використання таблиці Rainbow для зламу паролів, що зберігаються за допомогою хешу md5/sha1 коли алгоритм blowfish з підсолюванням робить дехешуванням просто марним.

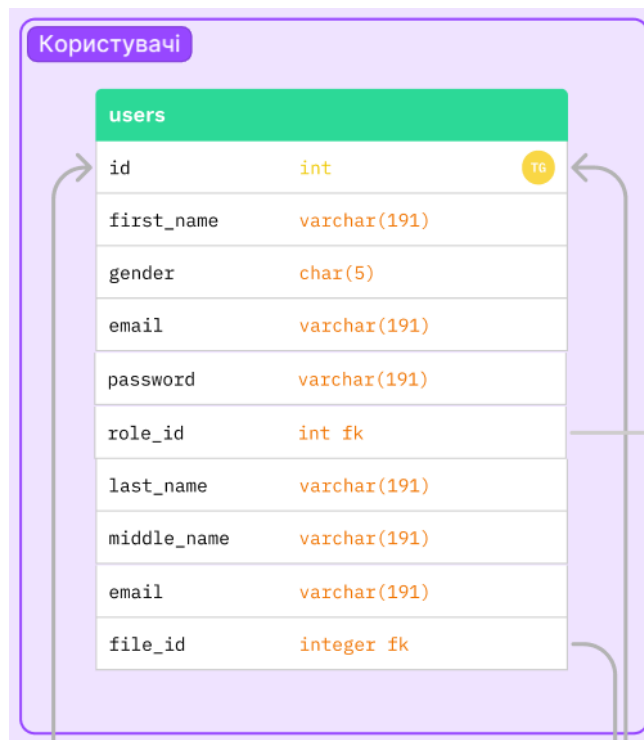


Рисунок 2.1.2 Фрагмент діаграми сутність-зв'язок БД, що відповідає за користувача

2.2 Налаштування серверу NGINX

На даний момент nginx є найпопулярнішим веб-сервером, нещодавно випередивши Apache. Він легкий, швидкий, надійний і підтримує всі основні операційні системи. Це веб-сервер вибору для Netflix, WordPress.com та інших сайтів із високою відвідуваністю. Сервер nginx може легко обробляти 10 000 неактивних HTTP-з'єднань лише з 2,5 МБ пам'яті.

В ході виконання курсового проекту сервер nginx було встановлено на ОС Linux, саме Ubuntu Server, за рахунок її відкритості, безпеки та її налаштувань а також унікальності інструментарію командних утиліт.

2.2.1 Налаштування конфігураційних файлів

Після встановлення Nginx параметри його конфігурації можна знайти у файлі nginx.conf. Це основний файл конфігурації nginx, тому більшість перевірок безпеки буде виконуватися за допомогою цього файлу. За замовчуванням nginx.conf знаходиться у [каталог встановлення nginx]/conf у системах Windows і в /etc/nginx або /usr/local/etc/nginx у системах Linux. Також може знадобитися внести деякі

зміни до файлів конфігурації віртуального хоста, які зазвичай містяться в підкаталозі sites-available.

Розглянемо типові безпекові налаштування конфігураційних файлів nginx:

1) Приховання чутливої інформації

Так само, як і Apache, за замовчуванням веб-сервер Nginx також розкриває деяку цінну інформацію, наприклад версію веб-сервера, яка може допомогти зловмиснику використати будь-яку відому помилку або вразливість, наявну в конкретній версії пакета.

Також nginx виводить код помилки, відповідний некоректно виконаному чи обробленому запиту, що може вказати зловмисникам певні налаштування сервера. Рекомендується повертати на помилки з кодом 401-405 лише код 404.

Для запобігання таким випадкам необхідно внести зміни в конфігурацію nginx за замовчуванням /etc/nginx/nginx.conf, а саме в секцію http.

Приклад налаштування /etc/nginx/nginx.conf:

```
http {  
    server_tokens off;  
    error_page 401 403 404 /404.html;  
}
```

2) Вимкнення перегляду каталогу

У nginx список каталогів за замовчуванням вимкнено, але все ж добре перевірити та підтвердити. Кожен може побачити весь вміст каталогу веб-застосунку у браузері, якщо ввімкнено перегляд каталогів, тож веб-сервер можуть розкривати деякий вміст без потреби.

Для запобігання таким випадкам необхідно внести зміни в конфігурацію nginx за замовчуванням /etc/nginx/nginx.conf, а саме в секцію location.

Приклад налаштування /etc/nginx/nginx.conf:

```
location / {  
    autoindex off;  
}
```

3) Оптимізація налаштувань SSL/TLS

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

Конфігурація nginx за замовчуванням дозволяє використовувати незахищені старі версії протоколу TLS (згідно з офіційною документацією: `ssl_protocols TLSv1 TLSv1.1 TLSv1.2`). Це може призвести до таких атак, як атака BEAST. Тому рекомендується не використовувати старі протоколи TLS і змінити конфігурацію, щоб підтримувати лише нові, безпечні версії TLS.

Для запобігання таким випадкам необхідно внести зміни в конфігурацію nginx за замовчуванням `/etc/nginx/nginx.conf`, а саме в секцію `server`.

Приклад налаштування `/etc/nginx/nginx.conf`:

```
server {
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers
ECDH+AESGCM:ECDH+AES256:ECDH+AES128:!ADH:!AECDH:!MD5;
}
```

4) CSP та захист від XSS

Політика безпеки вмісту (CSP) захищає веб-сервер від певних типів атак, включаючи міжсайтові атаки сценаріїв (XSS) та атаки ін'єкції даних. CSP можливо реалізувати, додавши приклад заголовка Content-Security-Policy (зверніть увагу, що фактичний заголовок має бути налаштований відповідно до ваших унікальних вимог).

Заголовок HTTP X-XSS-Protection підтримується IE та Safari і не є необхідним для сучасних браузерів, якщо є жорстка політика безпеки вмісту. Однак, щоб запобігти XSS у старіших браузерах (які ще не підтримують CSP), необхідно додати заголовок X-XSS Protection.

Для запобігання таким випадкам необхідно внести зміни в конфігурацію nginx для певного сайту `/etc/nginx/sites-available/xyz.conf`, а саме в секцію `server`.

Приклад налаштування `/etc/nginx/sites-available/xyz.conf`:

```
server {
    add_header X-XSS-Protection "1; mode=block";
```

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

```
add_header Content-Security-Policy "default-src 'self' http: https: data: blob:
'unsafe-inline'" always;
}
```

5) Захист від клікджекінгу

Заголовок відповіді HTTP X-Frame-Options використовується, щоб вказати, чи потрібно дозволити браузеру відображати сторінку в <frame> або <iframe>. Це може запобігти атакам клікджекінгу. Тому рекомендується увімкнути цю опцію для сервера nginx.

Для запобігання таким випадкам необхідно внести зміни в конфігурацію nginx для певного сайту /etc/nginx/sites-available/xyz.conf, а саме в секцію server.

Приклад налаштування /etc/nginx/sites-available/xyz.conf:

```
server {
    add_header X-Frame-Options "SAMEORIGIN (або DENY)";
}
```

6) Захист від перехоплення вмісту

Браузери намагаються виявити MIME-тип файлів, які надсилає веб-сервер.

Якщо зломисник завантажує шкідливий виконуваний файл на веб-сервер, який надсилає лише зображення, ідентифікація типу MIME може забезпечити певний захист. Вона повідомляє браузеру, що йому слід очікувати зображення, а не виконуваний файл. Як результат, браузер не повинен намагатися виявити MIME-тип, а повинен використовувати тільки MIME-тип, наданий веб-сервером.

Для запобігання таким випадкам необхідно внести зміни в конфігурацію nginx для певного сайту /etc/nginx/sites-available/xyz.conf, а саме в секцію server.

Приклад налаштування /etc/nginx/sites-available/xyz.conf:

```
server {
    add_header X-Content-Type-Options nosniff;
}
```

7) Вимкнення небажаних HTTP методів

Рекомендується вимкнути будь-які методи HTTP, які не будуть використовуватися і які не потрібно впроваджувати на веб-сервері. Якщо додати

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

таку умову в блок розташування файлу конфігурації віртуального хоста nginx, сервер дозволить лише методи GET, HEAD і POST і відфільтруватиме такі методи, як DELETE і TRACE.

Для запобігання таким випадкам необхідно внести зміни в конфігурацію nginx для певного сайту `/etc/nginx/sites-available/xyz.conf`, а саме в секцію `server`.

Приклад налаштування `/etc/nginx/sites-available/xyz.conf`:

```
server {  
    if ($request_method !~ ^(GET|HEAD|POST)$ ) {  
        return 405;  
    }  
}
```

8) HSTS

HTTP Strict Transport Security (HSTS) – це метод, який використовується веб-сайтами, щоб заявити, що доступ до них повинен здійснюватися лише за допомогою безпечного з'єднання (HTTPS). Якщо веб-сайт оголошує політику HSTS, браузер повинен відмовитися від усіх HTTP-з'єднань і заборонити користувачам приймати незахищені сертифікати SSL. Частіше всього це налаштування робиться паралельно перенаправлення HTTP трафіку на HTTPS.

Для запобігання таким випадкам необхідно внести зміни в конфігурацію nginx для певного сайту `/etc/nginx/sites-available/xyz.conf`, а саме в секцію `server`.

Приклад налаштування `/etc/nginx/sites-available/xyz.conf`:

```
server {  
    add_header Strict-Transport-Security "max-age=31536000; includeSubdomains;  
    preload";  
}
```

9) Доступ на основі IP-адреси

Коли доступ до будь-якого ресурсу веб-серверу nginx є небажаним для всіх окрім чітко визначених користувачів, зазвичай це обмежується на рівні секції `location`.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

Для запобігання таким випадкам необхідно внести зміни в конфігурацію nginx для певного сайту `/etc/nginx/sites-available/xyz.conf`, а саме в секцію `location`.

Приклад налаштування `/etc/nginx/sites-available/xyz.conf`:

```
location /admin {  
    allow ip_address/netmask;  
    deny all;  
}
```

10) Контроль ресурсів та обмежень

Щоб запобігти потенційним атакам DoS на nginx, необхідно встановити обмеження розміру буфера для всіх клієнтів..

Для запобігання таким випадкам необхідно внести зміни в конфігурацію nginx для певного сайту `/etc/nginx/sites-available/xyz.conf`, а саме в секцію `server`.

Приклад налаштування `/etc/nginx/sites-available/xyz.conf`:

```
server {  
    client_body_buffer_size 1K;  
    client_header_buffer_size 1k;  
    client_max_body_size 1k;  
    large_client_header_buffers 2 1k;  
}
```

2.2.2 Брандмауер iptables у поєднанні з fail2ban

Налаштування хорошого брандмауера є важливим кроком для захисту будь-якої сучасної операційної системи. Більшість дистрибутивів Linux поставляються з кількома різними інструментами брандмауера, які можна використовувати для налаштування брандмауерів. У ході виконання курсового проекту вибір пав на iptables.

Iptables – це стандартний брандмауер, включений у більшість дистрибутивів Linux за замовчуванням (сучасний варіант під назвою nftables почне замінювати його). Насправді це передня частина hook'ів netfilter на рівні ядра, які можуть маніпулювати мережевим стеком Linux. Він працює шляхом зіставлення кожного

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

пакета, який перетинає мережевий інтерфейс, з набором правил, щоб вирішити, що робити.

Брандмауер iptables працює шляхом порівняння мережевого трафіку з набором правил. Правила визначають характеристики, які повинен мати пакет, щоб відповідати правилу, і дії, які слід виконати для збігу пакетів.

Існує багато варіантів, щоб визначити, які пакети відповідають певному правилу. Є можливість зіставити тип протоколу пакету, адресу або порт джерела або призначення, інтерфейс, який використовується, його відношення до попередніх пакетів тощо.

Коли визначений шаблон збігається, дія, що виконується, називається цілю. Ціль може бути остаточним рішенням політики для пакета, наприклад прийняти або відкинути. Також можна перемістити пакет в інший ланцюжок для обробки або просто зареєструвати його входження.

Ці правила організовані в групи, які називаються ланцюжками. Ланцюжок – це набір правил, за якими пакет перевіряється послідовно. Коли пакет відповідає одному з правил, він виконує пов'язану дію і не перевіряється на відповідність іншим правилам у ланцюжку.

Користувач може створювати ланцюжки за потребою. За замовчуванням визначено три ланцюга. Це наступні:

INPUT: Цей ланцюжок обробляє всі пакети, адресовані серверу.

OUTPUT: Цей ланцюжок містить правила для трафіку, створеного сервером.

FORWARD: Цей ланцюжок використовується для роботи з трафіком, призначеним для інших серверів, які не створені на поточному сервері. Цей ланцюжок в основному є способом налаштувати сервер для маршрутизації запитів на інші машини.

Кожен ланцюжок може містити нуль або більше правил і має політику за замовчуванням. Політика визначає, що станеться, коли пакет пропускає всі правила в ланцюжку і не відповідає жодному правилу. iptables можете скинути пакет або прийняти пакет, якщо жодне з правил не відповідає.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

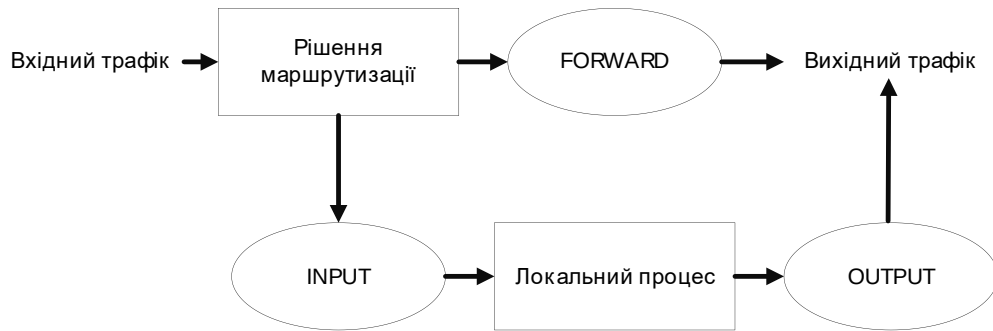


Рисунок 2.2.1 – Принцип роботи iptables

За допомогою модуля, який можна завантажити за допомогою правил, iptables також може відстежувати з'єднання. Це означає, що можна створювати правила, які визначають, що відбувається з пакетом на основі його зв'язку з попередніми пакетами. Це називається «відстеження стану», «відстеження з'єднання» або налаштування «машини станів».

Але іноді адміністратор не має наміру блокувати будь-який трафік, поки він задовольняє певній умові, наприклад не є великою кількістю повідомлень, що являє собою http-flood, чи http-auth-bruteforce. Для таких випадків використовуються утиліти, здатні обробляти журнали серверу, та знаходити і відфільтровувати незвичну поведінку за допомогою відповідного брандмауера.

Однією з таких утиліт є fail2ban, яка за замовчуванням використовує той же iptables у якості свого брандмауера. Fail2ban – це програма для аналізу журналів, яка відстежує системні журнали на наявність симптомів автоматичної атаки на сервер. Коли виявлено спробу атаки, використовуючи визначені параметри, fail2ban додає нове правило до iptables, щоб заблокувати IP-адресу зловмисника або на певний проміжок часу, або назавжди. Fail2ban також може повідомити електронною поштою про те, що відбувається атака.

Fail2ban в першу чергу зосереджений на атаках SSH, хоча його можна додатково налаштувати для роботи для будь-якої служби, яка використовує файли журналів і може бути піддана атаці. Що також є зручним, це те, що в її конфігураційних файлах попередньо визначенні налаштування та фільтри для роботи з http-серверами, тобто і з nginx.

Принцип роботи fail2ban базується на конфігураційних файлах «в'язниць», де за допомогою підключення попередньо визначених або власноруч написаних

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

файлів-фільтрів, відбувається відбір трафіку та його блокування за визначеними параметрами.

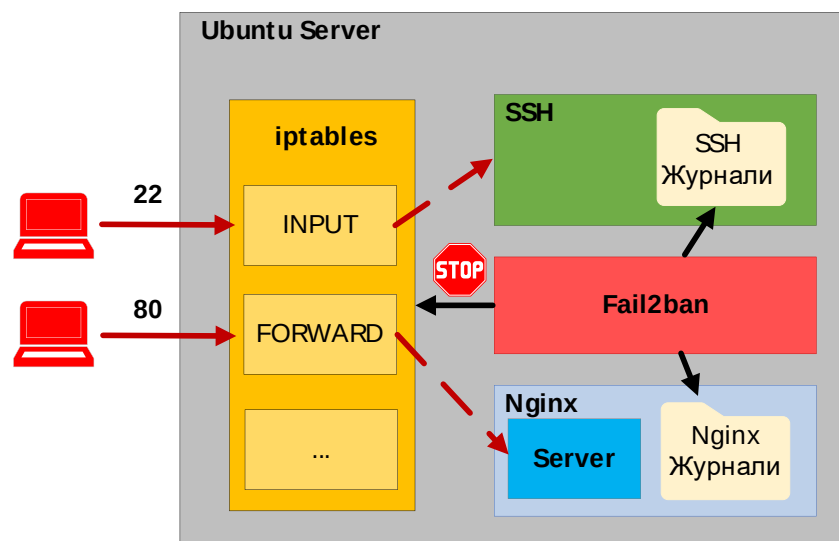


Рисунок 2.2.2 – Принцип роботи fail2ban

2.2.3 NAXSI/ModSecurity

Nginx Anti-XSS & SQL Injection (або просто NAXSI) – це WAF, спеціально розроблений для серверів Nginx.

NAXSI допомагає боротися із зловмисниками, які додають вразливі скрипти на веб-сайт. Крім того, він дозволяє уникнути введення коду в базу даних, використовувану веб-сайтами.

Працює за принципом: все, що не дозволено – заборонено. Кожен HTTP-запит (GET | PUT | POST) перевіряється на відповідність шаблонів забороняючих правил, заданих за замовчуванням у файлі `naxsi_core.rules`. Ці правила охоплюють 99% всіх можливих варіантів шкідливих запитів. Наприклад, за замовчуванням заборонені всі запити, в URI яких міститься символ подвійних лапок. Якщо для нормальної роботи програми такий символ необхідний, вручну потрібно внести відповідні винятки в білий список. Але є і зворотний бік медалі – якщо роздільні правила будуть опрацьовані погано, то NAXSI блокуватиме ще й частину легітимних запитів. Тому відповідальність за кінцевий результат розробники модуля повністю покладають на адміністратора системи. Як і більшість модулів для Nginx, NAXSI з репозиторію недоступний, тому його доведеться вручну завантажувати та компілювати.

		Марущак Ю.Р.			KIK.KP.B – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

Основні переваги NAXSI включають:

- Простий набір правил

NAXSI захищає веб-сайти за допомогою простого набору правил, використовуючи систему захисту, засновану на оцінці. Так, ґрунтуючись на цих жорстких правилах, кожному отримуваному URL-запиту виставляється оцінка. Якщо ця оцінка перевищує певне граничне значення, задане в конфігурації, NAXSI автоматично блокує запит на веб-сайт. Наприклад, коли URL-запит містить потенційно шкідливі символи, такі як «<», «/» (слеш), «drop», а також інші «небезпечні» символи та ключові слова-запити SQL, це автоматично підвищує значення оцінки. І виконання таких URL-запитів буде заблоковано на сервері.

- Підтримка білих списків

Також сильною стороною NAXSI є підтримка створення набору правил так званого білого списку. Ці правила визначають, які шаблони з потенційно шкідливими патернами і для яких додатків є прийнятними, і, відповідно, NAXSI не блокуватиме їх виконання. Щоб спростити адміністраторам завдання щодо складання набору правил для «білого» списку, NAXSI постачається з програмним інструментарієм під назвою Nxtool. Цей інструмент може автоматично вивчати трафік веб-сайту, а також створювати та доповнювати «білий» список. Слід зазначити, що цей інструментарій здатний враховувати активність користувачів. Так, якщо більше 20% користувачів постійно використовують ідентичний «небезпечний» метод для своїх запитів на веб-сайті, він буде зареєстрований як дозволений. І всі подібні запити безперешкодно проходять через веб-сервер.

- Стійкість до методів обходу WAF

Під будь-які навіть найсуворіші правила міжмережевих екранів для веб-застосунків хакери знаходять альтернативні способи їх обходу. Але NAXSI чудово справляється з нейтралізацією можливих методів обходу WAF, таких як кодування URL, об'єднання рядків у запиті тощо.

- Швидкість та легкість адміністрування

NAXSI не буде «з'їдати» більшість серверних ресурсів. Крім того, він не вимагає проведення періодичних і частих оновлень, чим вигідно відрізняється від

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

WAF ModSecurity. Відразу після установки NAXSI працюватиме безперервно і надійно, без будь-яких періодів вимушеного простою.

Хоча у NAXSI є багато переваг, у нього також є свої недоліки:

- Потрібно запустити режим навчання для кожного оновлення веб-застосунку.

NAXSI має два режими роботи: робочий (Live) та навчальний (Learning). Саме режимі навчання NAXSI не блокує запити, а вивчає поведінку користувачів, автоматично створюючи правила «білого» списку. Тому щоразу, коли в коді веб-сайту відбувається оновлення, потрібно буде запускати NAXSI в режимі навчання, щоб внести корективи в роздільні правила «білого» списку і уникнути блокування легітимного трафіку. Це створює певні додаткові проблеми, особливо якщо на веб-сайті нерідко відбуваються модифікації коду.

- Залежність від типу веб-сервера.

До суттєвого недоліку NAXSI можна віднести те, що він працює тільки з системами, реалізованими на базі Nginx. Тобто він не підходить для використання з Apache або IIS, що накладає серйозні обмеження на параметри веб-сервера. Тим не менш, NAXSI чудово працює в додатках на основі Docker. Тому NAXSI можна налаштувати для таких популярних програм, як ownCloud, Elasticsearch і т.д..

Не гіршою, а в чомусь навіть кращою, альтернативою NAXSI можна встановити WAF ModSecurity.

ModSecurity – це один з найпопулярніших WAF, який підтримує такі веб-сервери, як Apache, IIS, Nginx і т. д. Він працює з бібліотекою шкідливих шаблонів, відомих також як сигнатури. Коли URL-адреса відповідає будь-якій із сигнатур, він блокується. Хоча ModSecurity і є відмінним програмним інструментарієм для виявлення атак XSS (Cross-Site Scripting, міжсайтовий скриптинг), троянських атак тощо, він також має свої переваги та недоліки.

ModSecurity має такі переваги, як:

- Блокування стандартних атак

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

ModSecurity поставляється з набором правил Core Rule, що враховує практично всі відомі атаки в Інтернеті. Тому не потрібно буде самостійно прописувати правила для блокування вже відомих вразливостей додатків.

- Підтримка віртуального патчингу (Virtual patching)

Також ModSecurity надає можливість Virtual patching. Опція «Віртуальний патчінг» допоможе захистити сервер від атак, що використовують нещодавно виявлені вразливості. Іншими словами, як тільки з'являється нова атака на основі вразливості WordPress або PHP, просто потрібно буде встановити новий патч програми ModSecurity. Це нівелює можливість проведення подальших атак на ваші сайти за допомогою цієї вразливості. Таким чином буде отримано більше часу для оновлення кожного вразливого сайту, створеного на WordPress або PHP.

- Просунуте налаштування

Крім того, ModSecurity дозволяє писати конкретні правила для програм, розміщених на сервері. Це відкриває широкі можливості для використання власних модифікацій, коли є необхідність побудувати більш надійний захист від певних типів атак, які є найбільш критичними для веб-додатків.

Знову ж таки, у ModSecurity є і свої недоліки:

- Складне адміністрування великого набору правил

Майже всі правила ModSecurity базуються на регулярних виразах, адміністрування яких саме по собі є досить складним завданням. Крім того, що більше правил, то складніше стає коректно відфільтровувати «хороший» трафік від «поганого». Тим, хто працює з ModSecurity, доводиться періодично стикатися зі скаргами від клієнтів про те, що, крім блокування шкідливих атак, цей брандмауер для веб-застосунків часто блокує і легітимний трафік.

- Ресурсоємність

Коли налаштовано надто багато правил, ModSecurity починає використовувати надто багато ресурсів. Тому, чим більше веб-сайтів запущено на сервері, тим більше витрачатиметься ресурсів на його захист, і, відповідно, тим вище буде навантаження на сервер. Тому вибір на користь ModSecurity також багато в чому залежатиме від специфіки завдань та характеристик самого сервера.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

Nginx Anti-XSS & SQL Injection (або просто NAXSI) – це WAF, спеціально розроблений для серверів Nginx.

NAXSI допомагає боротися із зловмисниками, які додають вразливі скрипти на веб-сайт. Крім того, він дозволяє уникнути введення коду в базу даних, використовувану веб-сайтами.

Працює за принципом: все, що не дозволено – заборонено. Кожен HTTP-запит (GET | PUT | POST) перевіряється на відповідність шаблонів забороняючих правил, заданих за замовчуванням у файлі `naxsi_core.rules`. Ці правила охоплюють 99% всіх можливих варіантів шкідливих запитів. Наприклад, за замовчуванням заборонені всі запити, в URI яких міститься символ подвійних лапок. Якщо для нормальної роботи програми такий символ необхідний, вручну потрібно внести відповідні винятки в білий список. Але є і зворотний бік медалі – якщо роздільні правила будуть опрацьовані погано, то NAXSI блокуватиме ще й частину легітимних запитів. Тому відповідальність за кінцевий результат розробники модуля повністю покладають на адміністратора системи. Як і більшість модулів для Nginx, NAXSI з репозиторію недоступний, тому його доведеться вручну завантажувати та компілювати.

Основні переваги NAXSI включають:

- Простий набір правил

NAXSI захищає веб-сайти за допомогою простого набору правил, використовуючи систему захисту, засновану на оцінці. Так, ґрунтуючись на цих жорстких правилах, кожному отримуваному URL-запиту виставляється оцінка. Якщо ця оцінка перевищує певне граничне значення, задане в конфігурації, NAXSI автоматично блокує запит на веб-сайт. Наприклад, коли URL-запит містить потенційно шкідливі символи, такі як «<», «/» (слеш), «drop», а також інші «небезпечні» символи та ключові слова-запити SQL, це автоматично підвищує значення оцінки. І виконання таких URL-запитів буде заблоковано на сервері.

- Підтримка білих списків

Також сильною стороною NAXSI є підтримка створення набору правил так званого білого списку. Ці правила визначають, які шаблони з потенційно

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

шкідливими патернами і для яких додатків є прийнятними, і, відповідно, NAXSI не блокуватиме їх виконання. Щоб спростити адміністраторам завдання щодо складання набору правил для «білого» списку, NAXSI постачається з програмним інструментарієм під назвою Nxtool. Цей інструмент може автоматично вивчати трафік веб-сайту, а також створювати та доповнювати «білий» список. Слід зазначити, що цей інструментарій здатний враховувати активність користувачів. Так, якщо більше 20% користувачів постійно використовують ідентичний «небезпечний» метод для своїх запитів на веб-сайті, він буде зареєстрований як дозволений. І всі подібні запити безперешкодно проходять через веб-сервер.

- Стійкість до методів обходу WAF

Під будь-які навіть найсуворіші правила міжмережових екранів для веб-застосунків хакери знаходять альтернативні способи їх обходу. Але NAXSI чудово справляється з нейтралізацією можливих методів обходу WAF, таких як кодування URL, об'єднання рядків у запиті тощо.

- Швидкість та легкість адміністрування

NAXSI не буде «з'їдати» більшість серверних ресурсів. Крім того, він не вимагає проведення періодичних і частих оновлень, чим вигідно відрізняється від WAF ModSecurity. Відразу після установки NAXSI працюватиме безперервно і надійно, без будь-яких періодів вимушеного простою.

Хоча у NAXSI є багато переваг, у нього також є свої недоліки:

- Потрібно запустити режим навчання для кожного оновлення веб-застосунку.

NAXSI має два режими роботи: робочий (Live) та навчальний (Learning). Саме режимі навчання NAXSI не блокує запити, а вивчає поведінку користувачів, автоматично створюючи правила «білого» списку. Тому щоразу, коли в коді веб-сайту відбувається оновлення, потрібно буде запускати NAXSI в режимі навчання, щоб внести корективи в роздільні правила «білого» списку і уникнути блокування легітимного трафіку. Це створює певні додаткові проблеми, особливо якщо на веб-сайті нерідко відбуваються модифікації коду.

- Залежність від типу веб-сервера.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

До суттєвого недоліку NAXSI можна віднести те, що він працює тільки з системами, реалізованими на базі Nginx. Тобто він не підходить для використання з Apache або IIS, що накладає серйозні обмеження на параметри веб-сервера. Тим не менш, NAXSI чудово працює в додатках на основі Docker. Тому NAXSI можна налаштувати для таких популярних програм, як ownCloud, Elasticsearch і т.д..

Не гіршою, а в чомусь навіть кращою, альтернативою NAXSI можна встановити WAF ModSecurity.

ModSecurity – це один з найпопулярніших WAF, який підтримує такі веб-сервери, як Apache, IIS, Nginx і т. д. Він працює з бібліотекою шкідливих шаблонів, відомих також як сигнатури. Коли URL-адреса відповідає будь-якій із сигнатур, він блокується. Хоча ModSecurity і є відмінним програмним інструментарієм для виявлення атак XSS (Cross-Site Scripting, міжсайтовий скриптинг), троянських атак тощо, він також має свої переваги та недоліки.

ModSecurity має такі переваги, як:

- Блокування стандартних атак

ModSecurity поставляється з набором правил Core Rule, що враховує практично всі відомі атаки в Інтернеті. Тому не потрібно буде самостійно прописувати правила для блокування вже відомих вразливостей додатків.

- Підтримка віртуального патчингу (Virtual patching)

Також ModSecurity надає можливість Virtual patching. Опція «Віртуальний патчінг» допоможе захистити сервер від атак, що використовують нещодавно виявлені вразливості. Іншими словами, як тільки з'являється нова атака на основі вразливості WordPress або PHP, просто потрібно буде встановити новий патч програми ModSecurity. Це нівелює можливість проведення подальших атак на ваші сайти за допомогою цієї вразливості. Таким чином буде отримано більше часу для оновлення кожного вразливого сайту, створеного на WordPress або PHP.

- Просунуте налаштування

Крім того, ModSecurity дозволяє писати конкретні правила для програм, розміщених на сервері. Це відкриває широкі можливості для використання власних

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

модифікацій, коли є необхідність побудувати більш надійний захист від певних типів атак, які є найбільш критичними для веб-додатків.

Знову ж таки, у ModSecurity є і свої недоліки:

- Складне адміністрування великого набору правил

Майже всі правила ModSecurity базуються на регулярних виразах, адміністрування яких саме по собі є досить складним завданням. Крім того, що більше правил, то складніше стає коректно відфільтровувати «хороший» трафік від «поганого». Тим, хто працює з ModSecurity, доводиться періодично стикатися зі скаргами від клієнтів про те, що, крім блокування шкідливих атак, цей брандмауер для веб-застосунків часто блокує і легітимний трафік.

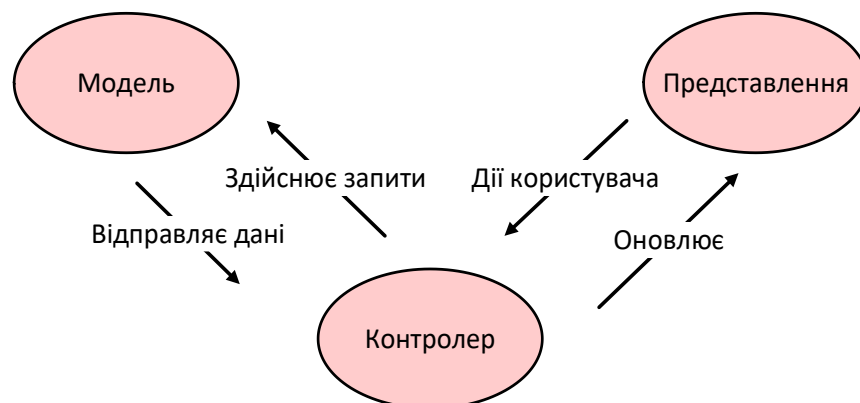
- Ресурсоемність

Коли налаштовано надто багато правил, ModSecurity починає використовувати надто багато ресурсів. Тому, чим більше веб-сайтів запуснено на сервері, тим більше витрачатиметься ресурсів на його захист, і, відповідно, тим вище буде навантаження на сервер. Тому вибір на користь ModSecurity також багато в чому залежатиме від специфіки завдань та характеристик самого сервера.

2.3 Налаштування API сервісу

API – це Application Programming Interface, це опис можливостей за допомогою, яких одна комп'ютерна програма може взаємодіяти з іншою програмою.

Сервіс побудований на основі Laravel, це безкоштовний, з відкритим кодом PHP-фреймворк і призначений для розробки вебдодатків відповідно до шаблону MVC (приклад “Принцип роботи MVC” рис. 2.3.1).



Риснок 2.3.1 – Принцип роботи MVC

MVC — це архітектура програмного забезпечення, яка відокремлює логіку домену/програми/бізнесу від решти інтерфейсу користувача. Це робиться, розділяючи програму на три частини: модель, подання та контролер.

Модель керує фундаментальними поведінками та даними програми. Він може відповідати на запити на інформацію, відповідати на інструкції щодо зміни стану своєї інформації і навіть сповіщати спостерігачів у системах, керованих подіями, коли інформація змінюється. Це може бути база даних або будь-яка кількість структур даних або систем зберігання. Коротше кажучи, це дані та управління даними програми.

Подання ефективно забезпечує елемент інтерфейсу користувача програми. Він переведе дані з моделі у форму, яка підходить для інтерфейсу користувача. Контролер отримує введення користувача та здійснює виклики до моделювання об'єктів і представлення для виконання відповідних дій.

Загалом, ці три компоненти працюють разом, щоб створити три основні компоненти MVC.

2.3.1 Налаштування API маршрутів, Middlewares, Trotting service

API маршрути – це інструкції за якими інша програма може здійснювати HTTP запити, щоб отримати інформацію, яка передбачана сервісом. Маршрути можна розділити на авторизовані та гостьові.

Доступ до авторизованих маршрутів здійснюється на основі API токenu, який ідентифікує користувача. У нашому сервісі буде налаштовано RESTful API, він характеризується за наступними критеріями:

1. Client-server – відокремлюється користувацького інтерфейсу від сховища даних, що даний можливість гнучко змінювати користувацький інтерфейс без змін в самому API, за рахунок чого відбувається спрощення у серверних компонентах.
2. Stateless – кожен запит від клієнта до серверу повинен містити всю інформацію, необхідну для розуміння запиту. Таким чином стан сеансу повністю зберігається на клієнті.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Cacheable – обмеження кешу, потребують щоб дані у відповіді на запит були очевидно чи неочевидно помічені як cacheable чи non-cacheable.
4. Uniform interface - застосовуючи принцип спільності до компонентного інтерфейсу, спрощується загальна архітектура системи та покращується видимість взаємодій.
5. Layered system - стиль шаруватої системи дозволяє архітектурі складатися з ієрархічних шарів, обмежуючи поведінку компонентів в такий спосіб, кожен компонент неспроможна “бачити” поза безпосереднього шару, з яким вони взаємодіють.
6. Code on demand (необов'язково) - REST дозволяє розширити функціональність клієнта, завантаживши та виконавши код у вигляді аплетів або скриптів. Це полегшує роботу з клієнтами за рахунок скорочення кількості функцій, необхідних для попередньої реалізації.

```
Route::group([
    'middleware' => 'auth:sanctum',
], function () {
    Route::get('user', [App\Http\Controllers\Api\User\UserController::class, 'getUser'])->name('user.get-user');
    Route::apiResource('me', \App\Http\Controllers\Api\User\UserController::class)->except(['index', 'store', 'destroy']);
    Route::apiResource('files', \App\Http\Controllers\Api\File\FileController::class)->only(['store', 'destroy']);
    Route::apiResource('certificates', \App\Http\Controllers\Api\Certificate\CertificateController::class);
    Route::apiResource('me.certificate_orders', \App\Http\Controllers\Api\Certificate\CertificateOrderController::class);
    Route::post('auth/logout', \App\Http\Controllers\Api\Auth\LogoutController::class)->name('auth.logout');
    Route::post('cancelOrder', \App\Http\Controllers\Api\Certificate\CancelCertificateController::class)->name('cancel-certificate');
});

Route::group([
    'prefix' => 'password',
], function () {
    Route::post('forget', \App\Http\Controllers\Api\Auth\ForgetPassword::class)->name('forget-password');
    Route::post('reset', \App\Http\Controllers\Api\Auth\ResetPassword::class)->name('reset-password');
});

Route::group([
    'prefix' => 'auth',
    'name' => 'auth.'
], function () {
    Route::post('login', \App\Http\Controllers\Api\Auth>LoginController::class)->name('login');
    Route::post('register', \App\Http\Controllers\Api\Auth\RegisterController::class)->name('register');
});

Route::group([
    'middleware' => ['auth:sanctum'],
    'prefix' => 'admin'
], function () {
    Route::apiResource('certificates', \App\Http\Controllers\Api\Admin\Certificate\CertificateController::class);
    Route::apiResource('certificateOrders', \App\Http\Controllers\Api\Admin\Certificate\CertificateOrderController::class);
});
```

Рисунок 2.3.2 - Реалізація API маршрутів

Middlewares – проміжне програмне забезпечення забезпечує зручний механізм для перевірки та фільтрації HTTP-запитів, що надходять у вашу програму, включає проміжне програмне забезпечення, яке перевіряє автентифікацію користувача вашої програми. Якщо користувач не автентифікований, проміжне програмне забезпечення перенаправить користувача

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

на екран входу у вашу програму. Однак, якщо користувач пройшов автентифікацію, проміжне програмне забезпечення дозволить просунути запит далі в програму. Додаткове проміжне програмне забезпечення може бути написано для виконання різноманітних завдань, крім аутентифікації. Наприклад, проміжне програмне забезпечення для реєстрації може записувати всі вхідні запити до вашої програми.

```
<?php

namespace App\Http\Middleware;

use ...

class RedirectIfAuthenticated
{
    /**
     * Handle an incoming request.
     *
     * @param Request $request
     * @param Closure(\Illuminate\Http\Request): (\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse) $next
     * @param string|null ...$guards
     * @return Response\RedirectResponse
     */
    public function handle(Request $request, Closure $next, ...$guards)
    {
        $guards = empty($guards) ? [null] : $guards;

        foreach ($guards as $guard) {
            if (Auth::guard($guard)->check()) {
                return redirect(to: RouteServiceProvider::HOME);
            }
        }

        return $next($request);
    }
}
```

Рисунок 2.3.3 - Приклад реалізації middleware

У прикладі вище, middleware проводить перевірку чи певна сутність користувача авторизована, якщо \$guard = admin, то check буде відбуватись до таблиці, яка привязана до цього гарда (приклад “ Приклад реалізації гарда” рис. 2.3.4).

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

```

'defaults' => [
    'guard' => 'web',
    'passwords' => 'users',
],
],

/*
|-----|
| Authentication Guards |
|-----|

| Next, you may define every authentication guard for your application. |
| Of course, a great default configuration has been defined for you |
| here which uses session storage and the Eloquent user provider. |

| All authentication drivers have a user provider. This defines how the |
| users are actually retrieved out of your database or other storage |
| mechanisms used by this application to persist your user's data. |

| Supported: "session" |

*/

'guards' => [
    'web' => [
        'driver' => 'session',
        'provider' => 'users',
    ],
],
],

/*
|-----|

```

Рисунок 2.3.4 - Приклад реалізації гарда

Trotting - Laravel включає потужні та настрайовані служби обмеження швидкості, які ви можете використовувати для обмеження кількості трафіку для певного маршруту або групи маршрутів. Обмеження швидкості можна викликати вручну, а можна прив'язувати до роутів чи групи роутів.

```

RateLimiter::for( name: 'api', function (Request $request) {
    return Limit::perMinute( maxAttempts: 60 )->by( key: $request->user()?->id ?: $request->ip());
});

```

Рисунок 2.3.5 - Приклад реалізації обмеження швидкості для API викликів

У прикладі “Приклад реалізації обмеження швидкості для API викликів” рис. 2.3.5, надано лише 60 запитів за хвилину, щоб не дати користувач спамити і забивати черги на сервері, якщо користувач авторизований то обмеження буде прив'язуватись до ідентифікатора, якщо ні то до ір адреси з якої здійснюється запит.

2.3.2 Налаштування API токенів, авторизації та автентифікації

Багато веб-додатків надають своїм користувачам спосіб аутентифікації за допомогою веб-додатку та «входу». У своїй основі засоби аутентифікації Laravel складаються з «гардів» і «провайдерів». Гарди визначають спосіб аутентифікації користувачів для кожного запиту. Наприклад, Laravel постачається із захистом сеансу, який підтримує стан за допомогою сховища сеансу та файлів cookie.

Для реалізації автентифікації нашого клієнтського веб-додатку, ми скористаємось готовим рішенням з використанням API TOKENS пакету Sanctum,

		Марущак Ю.Р.			KIK.KP.B – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

він забезпечує легку систему автентифікації для клієнтських машин, мобільних додатків.

Він дозволяє генерувати будь-яку кількість токенів для їхнього акаунту, що дозволяє мати доступ одночас з декількох пристроїв, токенам можна надавати доступи, щоб здійснювати обмеження стосовно дій користувача у додатку.

API токени зберігаються в БД у відповідній таблиці `personal_access_tokens` (приклад “Фрагмент діаграми сутність-зв’язок БД, що відповідає за API токени” рис. 2.3.6).



Рисунок 2.3.6 - Фрагмент діаграми сутність-зв’язок БД, що відповідає за API токени

Токен має динамічне поліморфне поле, яке забезпечує гнучку видачу токенів на основі моделей до якої належить запис. Токен – це випадковий рядок, який складається з 64 символів, які шифруються за допомогою хеш функції `sha256`, щоб приховати справжній текст токену та зберігається у БД, а користувач отримує чистий токен, хеш якого у майбутньому буде перевірятись з хешем токену, який збережений у БД.

Abilities – це рядок у форматі JSON у якому перераховані можливі доступи токену до певних модулів у програмі.

```
$token = $this->tokens()->create([
    'name' => $name,
    'token' => hash('sha256', $plainTextToken = Str::random(40)),
    'abilities' => $abilities,
]);
```

Рисунок 2.3.7 - Фрагмент коду створення API токену

Автентифікація – це підтвердження того, ким є користувач на вході. Це проходження перевірки автентичності.

Наш сервіс повністю закритий від неавтентифікованих користувачів, тому у будь-якому випадку, користувач зобов’язаний ввести пароль та логін для визначення користувача за цими параметрами (приклад “ Фрагмент коду автентифікації користувача” рис. 2.3.8).

```
class LoginController extends ApiController
{
    /**
     * Handle the incoming request.
     *
     * @param LoginRequest $request
     * @return JsonResponse
     */
    public function __invoke(LoginRequest $request)
    {
        if(!Auth::attempt($this->getCredential($request))) {
            return $this->sendError('email: ' . $request->email . ' не знайдено', [], code: 401);
        }

        return $this->sendResponse($this->createToken($request));
    }

    /**
     * @param LoginRequest $request
     * @return array
     */
    private function getCredential (LoginRequest $request): array
    {
        return $request->only( ['email', 'password']);
    }

    /**
     * @param LoginRequest $request
     * @return NewAccessToken
     */
    private function createToken (LoginRequest $request): NewAccessToken
    {
        return Auth::user()->createToken( name: DefaultSchema::TOKEN_USER_NAME);
    }
}
```

Рисунок 2.3.8 - Фрагмент коду автентифікації користувача

Авторизація – це те, що користувачу дозволяється робити після входу. Це надання і перевірка прав на вчинення будь-яких дій в системі.

У нас процес авторизації розпочинається після визначення користувача за паролем на логіном у нашій БД, а саме визначається його роль та на основі неї надаються певні дозволи до модулів чи дій (приклад “ Фрагмент коду, визначення доступів на основі ролі користувача” рис. 2.3.9).

```
/**
 * @param LoginRequest $request
 * @return NewAccessToken
 */
private function createToken (LoginRequest $request): NewAccessToken
{
    return Auth::user()->createToken( name: DefaultSchema::TOKEN_USER_NAME, $this->getPermissions());
}

/**
 * @return array
 */
private function getPermissions (): array
{
    return Auth::user()->role()->permissions->pluck( column: 'slug')->toArray();
}
```

Рисунок 2.3.9 - Фрагмент коду, визначення доступів на основі ролі користувача

		Марущак Ю.Р.			KIK.KP.B – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

Для кращого розуміння, що відбувається (приклад “Фрагмент діаграми сутність-зв’язок БД, що відповідає за ролі та їх доступи” рис. 2.3.10)

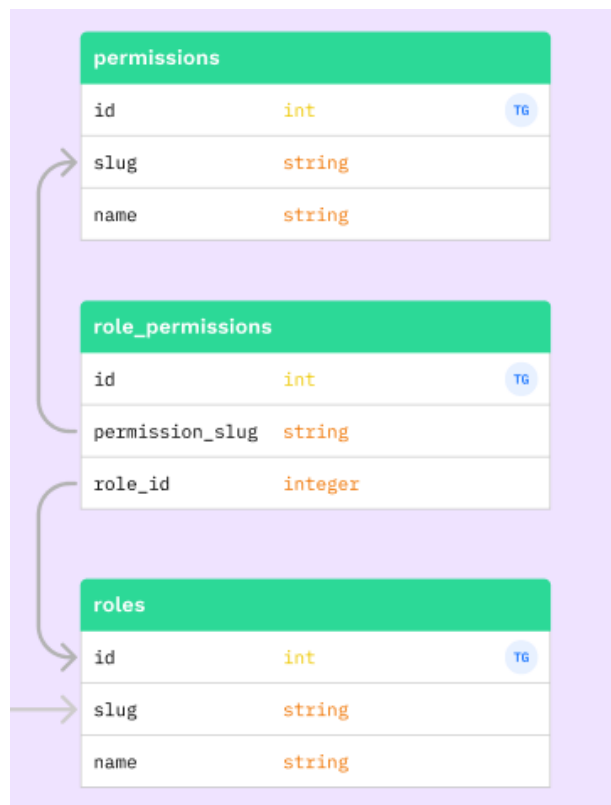


Рисунок 2.3.10 - Фрагмент діаграми сутність-зв’язок БД, що відповідає за ролі та їх доступи

На даному фрагменті, ми чітко бачимо, що у нас є ролі та доступи, де роль може мати багато доступів.

2.3.3 Налаштування RBAC сервісу

Перед тим як формувати сервіс RBAC було детально проаналізовано інтернет ресурси на опис, що таке RBAC, чому саме цей підхід до контролю над користувачами підходить саме цьому сервісу. Насамперед у нашому сервісі RBAC був модифікований шляхом додавання дозволів до певних ролей, що дозволяє гнучко та швидко додавати чи видаляти дозволи для них. Тепер коли відбувається значні зміни в бізнес моделі проекту нам не потрібно створювати нову роль чи редагувати існуючу, досить лише провести CRUD операцію з дозволами, що досить зручно та економить безліч часу.

Наша модель RBAC побудована наступним чином, користувач має роль, яка повина бути затверджена адміністратором, кожна роль має свій набір правил, доступів, дозволів (приклад “Принцип роботи RBAC у нашому сервісі” рис. 2.3.11).

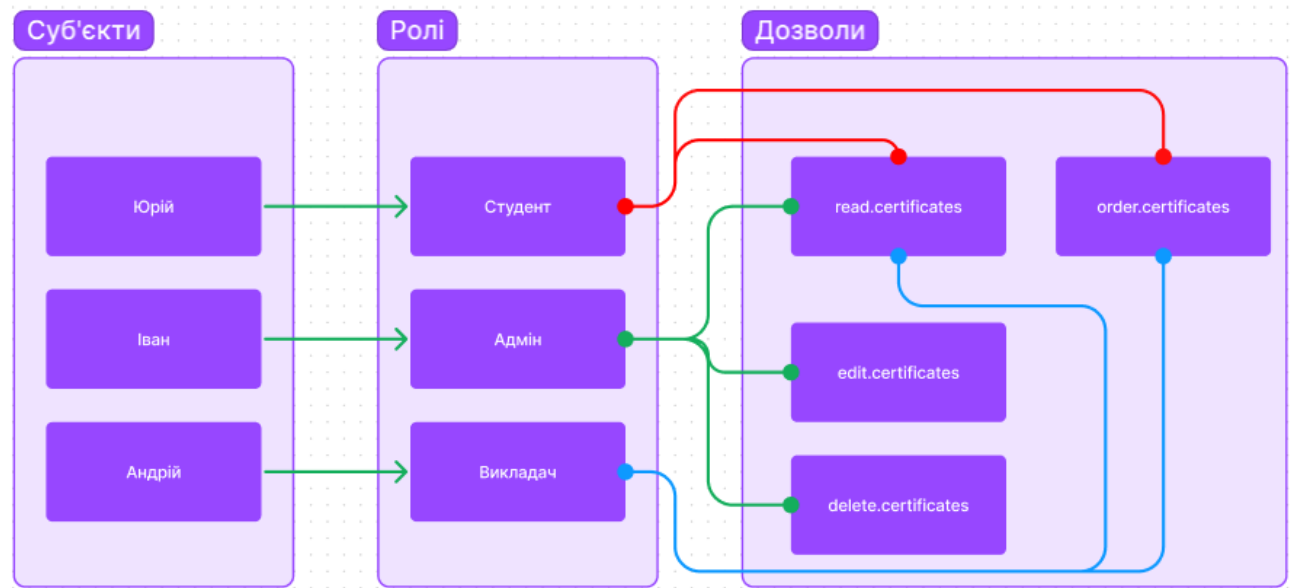


Рисунок 2.3.11 – Принцип роботи RBAC у нашому сервісі

У даній реалізації показано адміністратора, який може здійснювати CRUD операції над документами, студента та викладача, які мають наразі однакові доступи, вони можуть переглядати документи та робити замовлення.

2.4 Налаштування Blockchain

У ході виконання кваліфікаційної роботи блокчейн було написано на мові програмування Node.js, яка була створена для розробки масштабованих мережових додатків, в основі якої лежить HTTP протокол, з урахуванням потокової передачі та низької затримки. Завдяки цьому Node.js добре підходить для створення в нашій системі блокчейну.

2.4.1 Створення Blockchain, Block, Document сутностей

Сутність Blockchain говориться саме за себе, вона представляє головну надбудову над всім, що є у блокчейні, це фундамент з якого все починається при створенні цього сервісу.

Вона зберігає в собі ланцюг блоків та складність генерування документу, також в ній присутні корисні методи такі як: createGenesisBlock, getLatestBlock,

		Марущак Ю.Р.			KIK.KP.B – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

addDocument, isChainValid, verifyDocument (приклад “Фрагмент коду сутності Blockchain” рис. 2.4.1 та рис. 2.4.2).

CreateGenesisBlock – метод, який створює найперший блок у нашому блокчейні, попередній хеш якого дорівнює нулю, що є свідченням того, що цей блок дійсно перший у системі.

```
export class Blockchain {
  constructor(blocks, difficulty : number = 2) {
    this.difficulty = difficulty
    this.chain = blocks.length > 0 ? blocks : [this.createGenesisBlock()]
  }

  createGenesisBlock() {
    return new Block(new Date().getTime(), document: null, previousHash: "0")
  }

  getLatestBlock() {
    return this.chain[this.chain.length - 1]
  }

  addDocument (document, timestamp) {
    if(!document.documentHash) {
      throw new Error('Document must include document hash!')
    }
    if(!document.isValid()) {
      throw new Error('Cannot add invalid document to chain')
    }

    let block = new Block(timestamp, document, this.getLatestBlock().hash)
    block.mineBlock(this.difficulty)
    this.chain.push(block)
  }
}
```

Рисунок 2.4.1 - Фрагмент коду сутності Blockchain

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

```

isChainValid() {
  for (let i = 1; i < this.chain.length; i++){
    const currentBlock = this.chain[i]
    const previousBlock = this.chain[i - 1]

    if(!currentBlock.isValidDocument()) {
      return false
    }

    if(currentBlock.hash !== currentBlock.calculateHash()) {
      return false
    }

    if(currentBlock.previousHash !== previousBlock.hash) {
      return false
    }
  }

  return true
}

verifyDocument (signature) {
  if(!this.isChainValid()) {
    throw new Error('Chain is not valid!')
  }

  return this.chain.filter(el => el?.document?.signature === signature)
}

```

Рисунок 2.4.2 - Фрагмент коду сутності Blockchain

AddDocument – метод, який виконує функцію додавання блоку до блокчейну, він приймає сутність Document та позначку часу створення, якщо все коректно заповнено, то довжина ланцюга блоків збільшиться та до блокчейну буде додано новий блок, який буде використовуватись у майбутньому для генерування блоків.

IsChainValid – метод, який перевіряє коректність генерування блоків, він порівнює попередній хеш відносно теперішнього хеш блоку в циклі, якщо виявиться, що блок в ланцюзі згенерований невірно чи був модифікований, то метод виявить блок стосовно, якого були вчинені неправомірні дії.

VerifyDocument – метод, який перевіряє достовірність підпису документу організацією, для того, щоб впевнитися чи дійсно ця організація видала та підписала документ, якщо збігу у підписі не знайдено, це означає, що конкретний документ організація не видавала.

Сутність Block – це головна одиниця з якого складається ланцюг, він містить у собі позначку часу створення блоку, сутність Document, попередній хеш,

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

теперішній хеш, який розраховується (приклад “Фрагмент розрахунку хешу блоку” рис. 2.4.3) в момент створення блоку з урахуванням позначки часу, сутності Document, попереднього хешу, та позначку роботи, що дійсно система доклала зусиль на цифровізацію документу, так званий “proof-of-work” (приклад “Фрагмент співпадіння за шаблоном” рис.2.4.4), який генерується з урахуванням складності, до поки не відбудеться співпадіння хешу з заданим шаблоном. Блок має в собі методи генерації хешу, видобування блоку (приклад “Фрагмент видобування блоку” рис.2.4.5) та метод валідації документу, який є в блоці.

```
calculateHash () {
    return SHA256(this.timestamp + this.document + this.previousHash + this.nonce).toString()
}
```

Рисунок 2.4.3 - Фрагмент розрахунку хешу блоку

```
Хеш блоку: 00dcaf6f4d43193634e343ae7a06e3ad1063c7815e9465da6ad5d5bd1c329b25
Proof-of-work: 405
Шаблон збігу: 00
```

Рисунок 2.4.4 - Фрагмент співпадіння за шаблоном

```
mineBlock (difficulty) {
    while (this.hash.substring(0, difficulty) !== Array( difficulty + 1).join('0')) {
        this.nonce++
        this.hash = this.calculateHash()
    }
}
```

Рисунок 2.4.5 - Фрагмент видобування блоку

Сутність Document – це елемент з якого складається блок, він містить в собі хеш, який генерується (приклад “Фрагмент генерування хешу документа” рис.2.4.6) з хеш-суми вмісту певного документу, також є підпис (приклад “Фрагмент підписання документа” рис.2.4.7) з, який створюється за допомогою приватного та публічного ключа (з використанням крипто бібліотеки elliptic) певної організації та метод перевірки чи дійсно ця організація здійснювала підпис та генерування документу в системі.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

calculateHash () {
  return SHA256(this.documentHash).toString()
}

```

Рисунок 2.4.6 - Фрагмент генерування хешу документа

```

signHandler(signingKeys) {
  if(signingKeys.getPublic('hex') !== process.env.SIGN_PUBLIC_KEY) {
    throw new Error('You cannot sign documents for other companies')
  }

  const hash = this.calculateHash()
  const sign = signingKeys.sign(hash, 'base64')

  this.signature = sign.toDER( enc: 'hex')
}

```

Рисунок 2.4.7 - Фрагмент підписання документа

2.4.2 CRYPTO-JS та ELLIPTIC

Crypto-js – це набір готових стандартних і безпечних криптографічних алгоритмів, реалізованих у JavaScript з використанням найкращих практик і шаблонів. Вони швидкі та мають послідовний і простий інтерфейс. У проекті було використано алгоритм хешування SHA256 щоб: забезпечити фіксацію цілісності даних, приймати будь-яку довжину рядок та конвертувати його у фіксовану довжину, змінити дані лише в одну сторону для збереження в блокчейні.

Цей алгоритм не найшвидший, він займає золоту середину і його швидкості цілком достатньо, щоб хешувати дані та інформацію у реальному часі без великих затримок, такі як контрольну суму чи хеш файлів, його надійність також займає золоту середину, він кращий від алгоритму md5 тому що вихідний рядок у двічі більший, що забезпечую кращу стійкість до дешифрування, але в надійності він поступається своєму наступнику SHA512 який має у двічі кращу надійність, але він повільніший, так як у нашій системі інформація покривається декількома різними шарами хешування та шифрування було вирішено використовувати SHA256.

Elliptic – це бібліотека, яка використовує ECDSA, схему криптографії на основі еліптичних кривих (ECC). Створюючи публічний та приватний ключ

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

(приклад “Фрагмент коду створення ключів” рис. 2.4.8) за допомогою цієї бібліотеки та з використанням блокчейну, можна заявити право власності на документ чи будь-яку річ, яка зберігається саме у блокчейні.

```
import EC from "elliptic";
const ec = new EC.ec('ed25519')

const key = ec.genKeyPair()

const publicKey = key.getPublic('hex')
const privateKey = key.getPrivate( enc: 'hex')
```

Рисунок 2.4.8 - Фрагмент коду створення пари ключів

ECDSA – має окремі процедури для підпису та верифікації. Кожна процедура – це алгоритм, який складається з декількох арифметичних операцій. Алгоритм підпису використовує закритий ключ (приклад “Фрагмент підпису документу з використанням приватного ключа” рис. 2.4.9), алгоритм верифікації - відкритий ключ (приклад “Фрагмент верифікації документу з використанням публічного ключа” рис. 2.4.10).

Якщо взяти до уваги що ми використовуємо блокчейн то відкритий ключ це буде ідентифікатором закладу, який здійснив верифікацію документу, а закритий – це ключ, використовується для підпису документів, його повинні знати лише вибіркові особи і до рук зловмисника ні в якому разі не повинен потрапити (приклад “Результат створення пари ключів ключів” рис.2.4.9).

```
Private key 0bb0121adf574bf4eb1cee17a1418e3699496523e2f14a70464b99dcfcdebbae
Public key 044ccb4aaa651ba55162de47d0e8f5d33b5f438bfca9ede1722e2f5927c7160b694314fd31c26117fa446291babf1bfcbe1afbc3bc548d162904e7521b482c4962
```

Рисунок 2.4.9 - Результат створення пари ключів ключів

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

```

signHandler(signingKeys) {
  if(signingKeys.getPublic('hex') !== process.env.SIGN_PUBLIC_KEY) {
    throw new Error('You cannot sign documents for other companies')
  }

  const hash = this.calculateHash()
  const sign = signingKeys.sign(hash, 'base64')

  this.signature = sign.toDER( enc: 'hex')
}

```

Рисунок 2.4.10 – Фрагмент підпису документу з використанням приватного ключа

```

isValid () {
  if(!this.signature || this.signature.length === 0) {
    throw new Error('No signature in this document')
  }

  const publicKey = ec.keyFromPublic(process.env.SIGN_PUBLIC_KEY, enc: 'hex')

  return publicKey.verify(this.calculateHash(), this.signature)
}

```

Рисунок 2.4.11 - Фрагмент верифікації документу з використанням публічного ключа

2.4.3 Налаштування HTTP серверу на Node.JS

Щоб здійснити доступ до блокчейну буде використано локальну мережу у якій функціонує сервер та готовий фреймворк Express, який побудований на базі Node.js.

Express – це мінімалістичний та гнучкий веб-фреймворк для програм Node.js, що надає широкий набір функцій для мобільних та веб-додатків, маючи у своєму розпорядженні безліч службових методів HTTP та проміжних обробників, який дозволяє створити надійне, швидке API.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

```
import express from 'express'

const app = express()
const PORT = process.env.PORT || 3050

app.listen(PORT, hostname: () => {
  console.log(`Server is up on port ${PORT}`)
})
```

Рисунок 2.4.12 – Фрагмент коду запуску локального веб серверу на певному портові

Щоб приховати наш блокчейн всі запити до нього будуть здійснюватись виключно з серверу через додаток Laravel API, клієнт буде комунікувати виключно з Laravel API сервісом за допомогою роутів (приклад “Фрагмент коду роутів Node.js API” рис.2.4.13).

```
app.post( path: '/blocks', handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, Locals> , res : Response<ResBody, Locals> ) => {...})

app.get('/verifyDocument', (req : Request<P, ResBody, ReqBody, ReqQuery, Locals> , res : Response<ResBody, Locals> ) => {...})

app.post( path: '/watermarkDocument', handlers: (req : Request<P, ResBody, ReqBody, ReqQuery, Locals> , res : Response<ResBody, Locals> ) => {...})

app.get('/blocks', (req : Request<P, ResBody, ReqBody, ReqQuery, Locals> , res : Response<ResBody, Locals> ) => {...})

app.get('/blocksValid', (req : Request<P, ResBody, ReqBody, ReqQuery, Locals> , res : Response<ResBody, Locals> ) => {...})
```

Рисунок 2.4.13 – Фрагмент коду роутів Node.js API

Висновок до другого розділу

Під час виконання другого розділу були представлені можливі шаблони конфігурацій серверу NGINX для запобігання витоку чутливої інформації, захисту від XSS та CSRF атак, конфігурації внутрішніх систем, створене та чітко поставлене технічне завдання: створити сервіс через, який можна зробити онлайн замовлення довідок, забезпечити безпеку користувацьких даних, створити зручний веб-інтерфейс, створити можливість верифікувати документ, чи дійсно певна організація його видала, валідація документу на спробу модифікації, створення децентралізованого сховище, яке б зберігало в собі інформацію по всім документам, які були видані організацією, шифрування на підпис документів організацією. Створена діаграма БД, обрано систему керування доступом RBAC з

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

наступними ролями: адміністратор, менеджер, студент, викладач. Використано брандмауер iptables для забороним незахищеного HTTP трафіку. Створено API сервіс у якому було налаштовано middlewares для валідації вхідних даних у запитах, throttling сервіс, щоб захистити сервіс від DDOS атак та дуже часто повторюваних запитів. Детально розібрано функціонування та роботу сервісу блокчейну, технології та реалізовано сервіс для верифікації цілісності документів їх підпису та захист від модифікацій. Були наведені приклади з коду, які використовуються у всьому сервісі.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

РОЗДІЛ 3. ТЕСТУВАННЯ ТА РЕАЛІЗАЦІЯ МІР ЗАХИСТУ ВЕБ ДОДАТКУ

3.1 Базові налаштування конфігурацій Nginx

У нашому сервісі присутня клієнтський додаток, API сервіс та блокчейн сервіс. На приклад API сервісу, ми проведемо базові конфігурування серверу Nginx для запобігання виток інформації та для попередження найбільш розповсюджених атак на які може повпливати конфігурація серверу.

3.1.1 Приховування чутливої інформації, CSP, HSTS

Приховати інформації про версію веб-серверу саме ця дія не дасть зловмиснику швидко визначити можливі атаки, які вже є відомі на певну версію сервера Nginx, зараз при виконанні запиту до api в заголовках сервер віддає версію Nginx (приклад “Демонстрація відповіді серверу з показом версії сервера” рис. 3.1.1). Внесемо зміни до конфігурації (приклад “Фрагмент коду, щоб приховати версію веб-сервера Nginx” рис. 3.1.2).

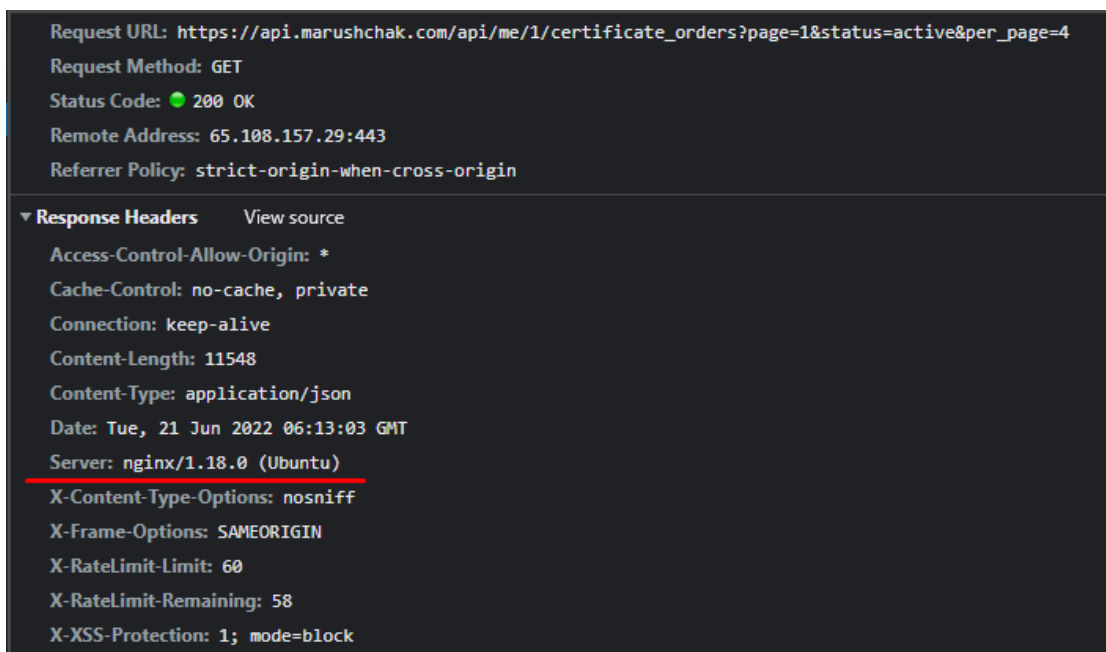


Рисунок 3.1.1 – Демонстрація відповіді серверу з показом версії сервера

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

```

GNU nano 4.8 /etc/nginx/nginx.conf Modified
worker_connections 768;
# multi_accept on;
}

http {

    ##
    # Basic Settings
    ##

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
    server_tokens off;

    # server_names_hash_bucket_size 64;
    # server_name_in_redirect off;

```

Рисунок 3.1.2 – Фрагмент коду, щоб приховати версію веб-сервера Nginx

```

Request URL: https://api.marushchak.com/api/me/1/certificate_orders?page=1&status=active&per_page=4
Request Method: GET
Status Code: 200 OK
Remote Address: 65.108.157.29:443
Referrer Policy: strict-origin-when-cross-origin

▼ Response Headers View source
Access-Control-Allow-Origin: *
Cache-Control: no-cache, private
Connection: keep-alive
Content-Length: 11548
Content-Type: application/json
Date: Tue, 21 Jun 2022 06:50:12 GMT
Server: nginx
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-RateLimit-Limit: 60
X-RateLimit-Remaining: 58
X-XSS-Protection: 1; mode=block

```

Рисунок 3.1.3 – Результат відповіді сервера після того як приховали версію Nginx у конфігураційних файлах

CSP та захист від XSS, політика безпеки вмісту (CSP) захищає веб-сервер від певних типів атак, включаючи міжсайтові атаки сценаріїв (XSS) та атаки ін'єкції даних (приклад “Фрагмент додавання захисту від XSS атак” рис. 3.1.4). Фрагмент коду “1;mode=block” – вказує на те, якщо XSS атаку виявлено то обробка сторінки буде заблокована.

		Марущак Ю.Р.			KIK.KP.B – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

```
server {
    root /var/www/api.marushchak.com/public;
    server_name api.marushchak.com www.api.marushchak.com;
    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Content-Type-Options "nosniff";

    index index.html index.htm index.php;
}
```

Рисунок 3.1.4 - Фрагмент додавання захисту від XSS атак

HSTS, коли браузер бачить цей заголовок із веб-сайту HTTPS, він «дізнається», що доступ до цього домену має здійснюватися лише за допомогою HTTPS (SSL або TLS). Він кешує цю інформацію для періоду максимального віку (зазвичай 31 536 000 секунд, що дорівнює приблизно 1 році). Необов'язковий параметр includeSubDomains повідомляє браузеру, що політика HSTS також застосовується до всіх субдоменів поточного домену. Параметр always гарантує, що заголовок встановлений для всіх відповідей, включаючи внутрішньо згенеровані відповіді на помилку. Старіші версії NGINX (до версії 1.7.5 або NGINX Plus R5) не підтримують параметр завжди і не встановлюють заголовок для внутрішньо згенерованих відповідей на помилки (приклад “Фрагмент коду встановлення HSTS режиму” рис. 3.1.5).

```
GNU nano 4.8 /etc/nginx/sites-available/api.marushchak.com
server {
    root /var/www/api.marushchak.com/public;
    server_name api.marushchak.com www.api.marushchak.com;
    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Content-Type-Options "nosniff";
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
    index index.html index.htm index.php;

    charset utf-8;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }
}
```

Рисунок 3.1.5 - Фрагмент коду встановлення HSTS режиму”

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

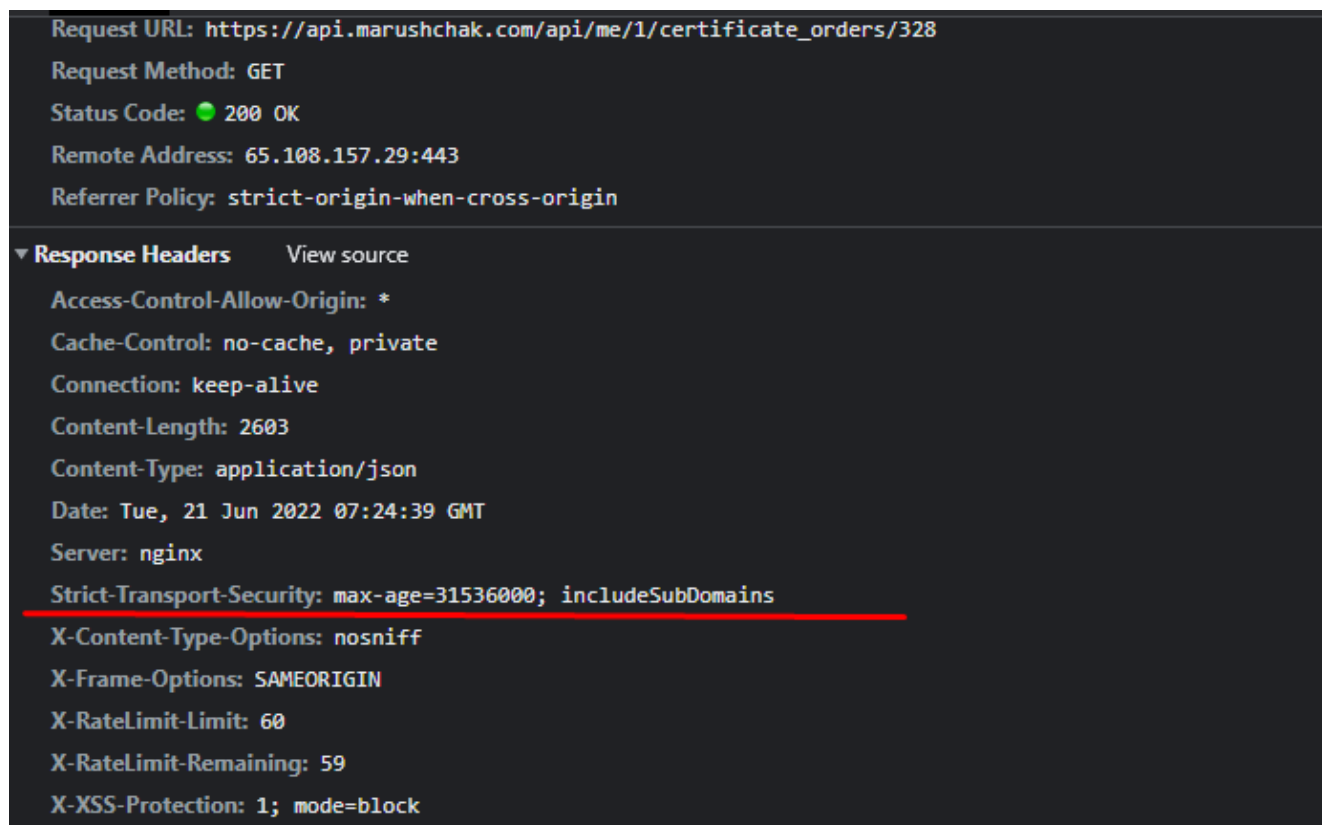


Рисунок 3.1.6 – Результат відповіді сервер з вказаним сувора політика стосовно HTTPS

Сервер застосував перенаправлення на інше посилання яке вказано в Location, якби це виконувалося у браузері, то відбудеться редірект автоматично.



Рисунок 3.1.7 – Спроба доступу до http (80) порту API сервісу

Вимкнемо редірект та спробуємо знову зробити запит до http (80), (приклад “Спроба запиту до HTTP” рис.3.1.8), як результат трафік був відкинутий сервером Nginx, тому що він не використовує HTTPS протокол.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

</html>root@ubuntu-4gb-helcurl http://api.marushchak.com/api/me/1/certificate_orders/328
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
root@ubuntu-4gb-hell-2:~#

```

Рисунок 3.1.8 – Спроба запиту до HTTP

3.2 Налаштування HTTPS

Щоб забезпечити трафік, який буде безпечний необхідно використати SSL/TLS шифрування і з цим нам допоможе протокол HTTPS, він особливо важливий у незахищених мережах (таких як публічні Wi-Fi точки доступу), оскільки будь-хто в локальних мережах може аналізувати трафік та перехоплювати чи змінювати інформацію, не захищену HTTPS.

Також без протоколу HTTPS більшість сучасний сервісів, таких як Telegram Bot, Google Smtп Server просто не працюють, тому що потребую шифрування повідомлень, щоб убезпечити дані, які передають.

Дуже важливо, використовувати SSL сертифікати, які підписані авторизованим сервісом, який уповноважений на видавати сертифікати, він засвідчує, що отримувач сертифікату — саме той, про кого йдеться у сертифікаті, але це не безкоштовно, тому ми будемо використовувати самопідписний сертифікат Certbot.

3.2.1 Certbot

Certbot — це безкоштовний програмний інструмент з відкритим вихідним кодом для автоматичного використання сертифікатів [Let's Encrypt](https://letsencrypt.org/) на веб-сайтах, які керуються вручну, щоб увімкнути HTTPS.

Для встановлення введемо команду `sudo certbot --nginx -d example.marushchak.com -d [www.example.marushchak.com](https://www.example.marushchak.com/)`, --nginx цей прапорець вказує, що ми використовуємо модуль nginx, -d вказує домені імена.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

```

root@ubuntu-4gb-hell-2:/var/www/diploma/client# sudo certbot --nginx -d example.
marushchak.com -d www.example.marushchak.com
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator nginx, Installer nginx
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for example.marushchak.com
http-01 challenge for www.example.marushchak.com
Waiting for verification...
Cleaning up challenges
Deploying Certificate to VirtualHost /etc/nginx/sites-enabled/client
Deploying Certificate to VirtualHost /etc/nginx/sites-enabled/client

Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP ac
cess.
-----
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
-----
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
Redirecting all traffic on port 80 to ssl in /etc/nginx/sites-enabled/client
Redirecting all traffic on port 80 to ssl in /etc/nginx/sites-enabled/client
-----
Congratulations! You have successfully enabled https://example.marushchak.com
and https://www.example.marushchak.com

You should test your configuration at:
https://www.ssllabs.com/ssltest/analyze.html?d=example.marushchak.com
https://www.ssllabs.com/ssltest/analyze.html?d=www.example.marushchak.com
-----

IMPORTANT NOTES:
- Congratulations! Your certificate and chain have been saved at:
/etc/letsencrypt/live/example.marushchak.com/fullchain.pem
Your key file has been saved at:
/etc/letsencrypt/live/example.marushchak.com/privkey.pem
Your cert will expire on 2022-09-09. To obtain a new or tweaked
version of this certificate in the future, simply run certbot again
with the "certonly" option. To non-interactively renew 'all' of
your certificates, run "certbot renew"
- If you like Certbot, please consider supporting our work by:
Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
Donating to EFF: https://eff.org/donate-le

```

Рисунок 3.2.1 - Результат виконання комананди `sudo certbot --nginx -d example.marushchak.com -d www.example.marushchak.com`

```

server_name www.example.marushchak.com example.marushchak.com; # managed by Certbot
return 404; # managed by Certbot


listen [::]:443 ssl; # managed by Certbot
listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/example.marushchak.com/fullchain.pem; # managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/example.marushchak.com/privkey.pem; # managed by Certbot
include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

```

Рисунок 3.2.2 - Конфігурація налаштування серверу для example.marushchak.com

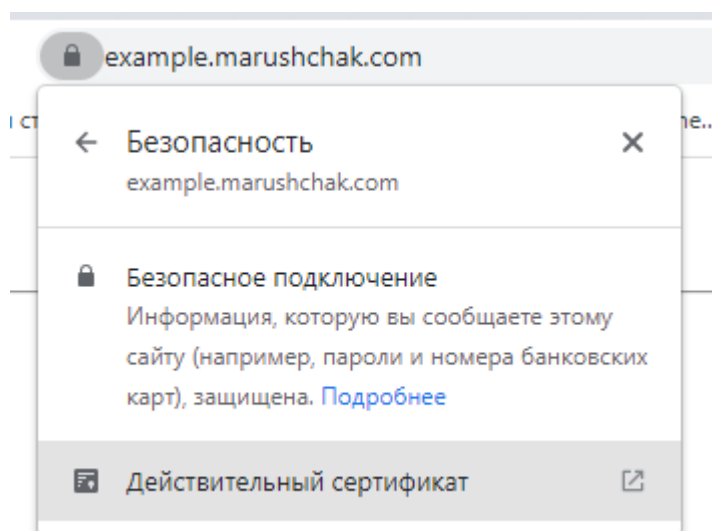


Рисунок 3.2.3 Результат встановлення SSL сертифікату

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

root@ubuntu-4gb-hell-2:/var/www/diploma/client# ls -la /etc/letsencrypt/live/example.marushchak.com/
total 12
drwxr-xr-x 2 root root 4096 Jun 11 21:35 .
drwx----- 4 root root 4096 Jun 11 21:35 ..
lrwxrwxrwx 1 root root   46 Jun 11 21:35 cert.pem -> ../../archive/example.marushchak.com/cert1.pem
lrwxrwxrwx 1 root root   47 Jun 11 21:35 chain.pem -> ../../archive/example.marushchak.com/chain1.pem
lrwxrwxrwx 1 root root   51 Jun 11 21:35 fullchain.pem -> ../../archive/example.marushchak.com/fullchain1.pem
lrwxrwxrwx 1 root root   49 Jun 11 21:35 privkey.pem -> ../../archive/example.marushchak.com/privkey1.pem
-rw-r--r-- 1 root root 692 Jun 11 21:35 README
root@ubuntu-4gb-hell-2:/var/www/diploma/client#

```

Рисунок 3.2.4 Створені SSL ключі

3.2.2 Доступ до HTTPS через брандмауер

Додано HTTPS трафік за допомогою команди `sudo ufw allow 'nginx full'`

```

rule added (v6)
root@ubuntu-4gb-hell-2:/var/www/diploma/client# sudo ufw status
Status: active

To Action From
--
Nginx Full ALLOW Anywhere
OpenSSH ALLOW Anywhere
Nginx HTTP ALLOW Anywhere
Nginx Full (v6) ALLOW Anywhere (v6)
OpenSSH (v6) ALLOW Anywhere (v6)
Nginx HTTP (v6) ALLOW Anywhere (v6)

root@ubuntu-4gb-hell-2:/var/www/diploma/client#

```

Рисунок 3.2.4 - Перегляд трафіку, який дозволений у брандмауері

Задля безпеки, заборонемо будь-який http трафік, який не використовує шифрування.

Виконаємо команду `sudo ufw delete 'Nginx HTTP'`

```

root@ubuntu-4gb-hell-2:/var/www/diploma/client# sudo ufw status
Status: active

To Action From
--
Nginx Full ALLOW Anywhere
OpenSSH ALLOW Anywhere
Nginx Full (v6) ALLOW Anywhere (v6)
OpenSSH (v6) ALLOW Anywhere (v6)

root@ubuntu-4gb-hell-2:/var/www/diploma/client#

```

Рисунок 3.2.5 - Результат виконання команди з видалення HTTP трафік

3.3 Тестування на XSS атаки

У нашому додатку присутня форма замовлення де користувач може ввести довільний текст, задумано було, що там буде пошта або телефон.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

Для розрахунку виплат на випадок безробіття сезон- ним працівникам та особам, звільненим у зв'язку із зняттям ЧАЕС з експлуатаці.

Куди відправити довідку ▼ Пошта, телефон.

Документ, який засвідчує вашу особистість

Замовити

Powered by PQINA

Рисунок 3.3.1 – Вигляд форми замовлення довідки

Потенційно вона може підлягати атаці, коли хтось захоче замовити довідку може додати скрипт, який за своїм задумом повинен був би виводити сповіщення кожного разу коли відмальовується DOM у адміністратора (приклад “Додавання зловмисного скрипту замість номеру телефона” рис. 3.2.2) та відправив заявку на замовлення.

Для розрахунку виплат на випадок безробіття сезон- ним працівникам та особам, звільненим у зв'язку із зняттям ЧАЕС з експлуатаці.

Пошта ▼ `<script>alert(JSON.stringify(document.cookie))</scr`

screen-item-3.png 69 KB Upload complete tap to undo

Замовити

Powered by PQINA

Рисунок 3.3.2 – Додавання зловмисного скрипту замість номеру телефона

Замовлення потрапило до менеджера, який спостерігаю наступну картинку (приклад “Відображення замовлення з зловмисним кодом менеджеру” рис.3.3.3).

Вітаємо! Ви увійшли до платформи довідок.

Статус замовлення ▼ На сторінці ▼

№ 332	№ 331	№ 330	№ 329
new	new	done	done
Для розрахунку виплат на випадок безробіття сезон- ним працівникам та особам, звільненим у зв'язку із зняттям ЧАЕС з експлуатаці.	Для розрахунку виплат на випадок безробіття сезон- ним працівникам та особам, звільненим у зв'язку із зняттям ЧАЕС з експлуатаці.	Для призначення допомоги з тимчасової непрацездатності, вагітності та пололів.	Для розрахунку Виплат на випадок безробіття.
Замовний: Юрій Русланович Марущак	Замовний: Юрій Русланович Марущак	Замовний: Катерина Петрівна Хвістова	Замовний: Олександр Вікторович Пірог
Роль: Студент	Роль: Студент	Роль: Менеджер	Роль: Викладач
Канал: mail	Канал: telegram	Канал: owp	Канал: mail
Значення <script>alert(JSON.stringify(document.cookie))</script>	Значення каналу: <script>alert("xss");</script>	Значення каналу: +380730962754	Значення каналу: +380730962754
Прикріплений документ: Переглянути	Прикріплений документ: Переглянути	Прикріплений документ: Переглянути	Прикріплений документ: Переглянути
Хеш файлу: Переглянути зразок	Хеш файлу: Переглянути зразок	Хеш файлу: \$2y\$10\$sz7mV4hvuLJ.G...	Хеш файлу: \$2y\$10\$UoeXdhAwjJVEI...
Відмовити Редагувати	Відмовити Редагувати	Відмовити Редагувати	Відмовити Редагувати

Рисунок 3.3.3 – Відображення замовлення з зловмисним кодом менеджеру

Зловмисний код записався до бази даних, але наша програма ніяк не реагує і не виконує його, тому що весь вивід інформації відбувається виключно після того як основні модулі виконання коду завершили свою роботу і даний код виводиться

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

виключно як звичайни текст. Але при такій роботі є ризик виконання цього скрипта на стороні серверу, якщо відображення UI частини буде лежати на стороні бекенду і не буде передбачено екранування спеціальних символів, тому слід додати функціонал на стороні API сервісу, щоб той автоматично кожного разу здійснював екранування при збігові спеціальних символів.

```
class CustomXssProtection
{
    /**
     * Handle an incoming request.
     *
     * @param Request $request
     * @param Closure(\Illuminate\Http\Request): (\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse) $next
     * @return Response\RedirectResponse
     */
    public function handle(Request $request, Closure $next): Response\RedirectResponse
    {
        $fields = $request->all();

        foreach ($fields as $key => $field) {
            $request->merge([
                $key => htmlspecialchars($field)
            ]);
        }

        return $next($request);
    }
}
```

Рисунок 3.3.4 – Фрагмент коду власного екранування символів

Відправимо ідентичну форма з відповідним кодом (приклад “Додавання зловмисного скрипту замість номеру телефона” рис. 3.2.2), але тепер на стороні серверу всі відправленні поля будуть проходити екранування тому спеціалізовані символи будуть замінені на безпечні.

Після того як код проходить екранування він більше не загрожує нашому серверу, тому що ніколи не виконається і в такому вигляді він буде лежати в БД допоки його звідти не приберуть адміністратори (приклад “Результат відмальовування зловмисного коду у DOM структурі після екранування” рис. 3.3.5).

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

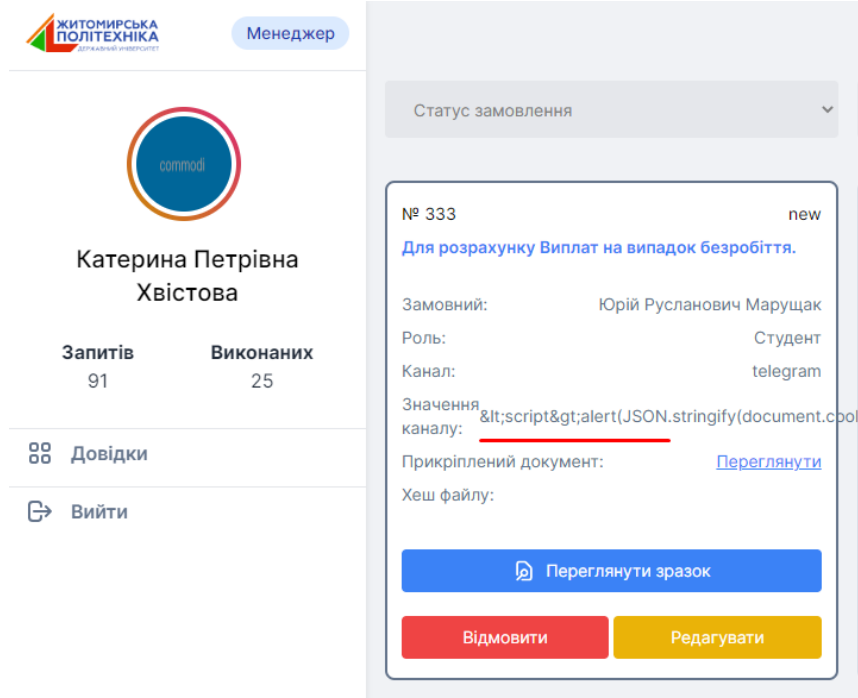


Рисунок 3.3.5 – Результат відмальовування зловмисного коду у DOM структурі після екранування

3.4 Викрадення токєну користувача

Ситуація коли зловмисник викрав токен існуючого користувача в системі і хоче від його імені розповсюджувати небезпечний код чи нашкодити якось нашому сервісу (приклад “Фрагмент викраденого токєну користувача” рис. 3.4.1). Коли є коректно сформований токен і він присутній в БД тоді у зловмисника є можливість робити всі дії, які були доступні жертві (приклад “Зловмисник потрапив у систему під чужими даними” рис. 3.4.2).

Name	Value	Domain	f	Expires / M...	Size	H	Sec...	Same..	Same..	Partiti..	Priority
GeoIP	UA:18:Zhytomyr:50.26:28.67:v4	.wikimedia....	/	Session	34		✓				Medi...
<u>user-token</u>	7%7C8COsdpm7yeVTjK8v3eZ38dCld964...	client.marus...	/	Session	54						Medi...

Рисунок 3.4.1 – Фрагмент викраденого токєну користувача

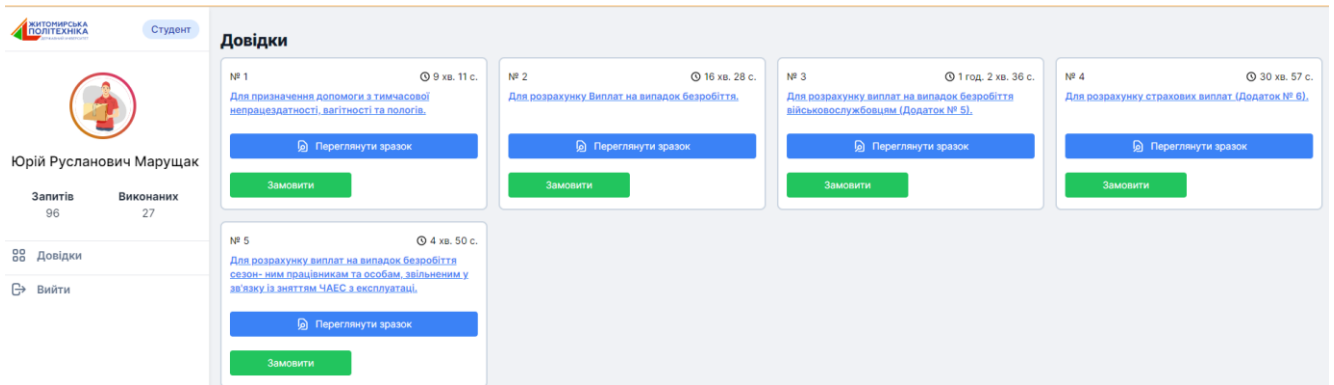


Рисунок 3.4.2 – Зловмисник потрапив у систему під чужими даними

Щоб вберегти свої користувачів від такого сценарію в момент автентифікації користувач буде прив'язуватись до IP адреси з якої користувач заходив. Буде можливість прив'язувати одного користувача до кількох адрес, якщо користувач підтвердить пароль. Якщо система бачить, що користувач змінив IP адресу, то вона автоматично перенаправляє його на авторизацію, де він може підтвердити свій пароль та пошту увівши заново (приклад “Демонстрація створеного запису з привязкою IP адреси” рис.3.4.3).



Рисунок 3.4.3 – Демонстрація створеного запису з привязкою IP адреси

Ось кіберзлочинець заволодів токеном користувач та починає виконувати запити до API сервісу, спрацьовує перевірка чи дійсно цей користувач заходив під цією адресою

```
public function handle(Request $request, Closure $next)
{
    if($request->user() && in_array($request->ip(), $request->user()->user_api_addresses->pluck('ip_address')->toArray())) {
        return $next($request);
    } else if(!$request->user()){
        return $next($request);
    }

    $request->user()->currentAccessToken()->delete();

    return (new ApiController())->sendError(error: 'Будь ласка увійдіть заново, ваша IP адреса змінилась', [], code: 401);
}
```

Рисунок 3.4.4 – Фрагмент коду з перевірки привязки користувача до IP адреси

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

Рисунок 3.4.5 – Результат перевірки, токен видалений з БД – необхідно ввести заново пароль та логін

3.5 Підробка документу

Давайте верифікуємо документ через нашу систему, в результаті верифікації документ буде доданий до блокчейн сервісу та додана автоматично згенерована водяна марка.

Користувач Юрій здійснив замовлення на довідку (приклад “Сформоване замовлення на довідку” рис. 3.4.5).

Рисунок 3.4.5 – Сформоване замовлення на довідку

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

Менеджер відповідно опрацював її, та здійснив верифікацію через блокчейн сервіс (приклад “Фрагмент верифікованої довідки” рис. 3.4.6).

Назва органу (установи)

252035, м. Київ-35, пл. Солом'янська, 1,

тел. 271-90-01, факс 234-22-22

21.06.2022 №595

ДОВІДКА

Видана майору міліції Тарнавському Тарасу Васильовичу про те, що він дійсно працює в Маруцау Юрію Руслановичу на посаді доцента кафедри криміналістики.

Довідка видана для подання до держадміністрації Шевченківського району м. Києва.

Начальник відділу по роботі

з персоналом

(підпис)

О.В.

Веревуров

Рисунок 3.4.6 – Фрагмент верифікованої довідки

This document has been approved by: Zhytomyr Polytechnic State University

3045022100d819ea9b786eedb61c1e00f04c2588c227cb854a43bb828a9253d908072d29ad022205314639e0ef06bbcbcd2aff5ca8c061e00e2dbd6c32702ab334ab6cf2adab358

Рисунок 3.4.7 – Сформована автоматично водна марка

Пройшов певний час та користувач термін дії довідки минув, тому користувач виріш підробити довідку та змінити дату створення додавши ту ж саму водну марку (приклад “Підробка дати видачі документу” рис. 3.4.6).

		Маруцак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

Назва органу (установи)
252035, м. Київ-35, пл. Солом'янська, 1,
тел. 271-90-01, факс 234-22-22
23.06.2022 №595

ДОВІДКА

Видана майору міліції Тарнавському Тарасу Васильовичу про те, що він дійсно працює в Марушау Юрію Руслановичу на посаді доцента кафедри криміналістики.

Довідка видана для подання до держадміністрації Шевченківського району м. Києва.

Начальник відділу по роботі з персоналом (підпис) О.В. Вєревуrow

Zhytomyr Polytechnic State University

Рисунок 3.4.8 – Підробка дати видачі документу

Коли користувач прийняв документ в потрібне місце його попросили підтвердити, що дійсно документ видала певна організація та дійсно ця організація його підписала через верифікаційний сервіс документів, але верифікація виявилась не вдалою (приклад “Невидала верифікація документу” рис. 3.4.7).

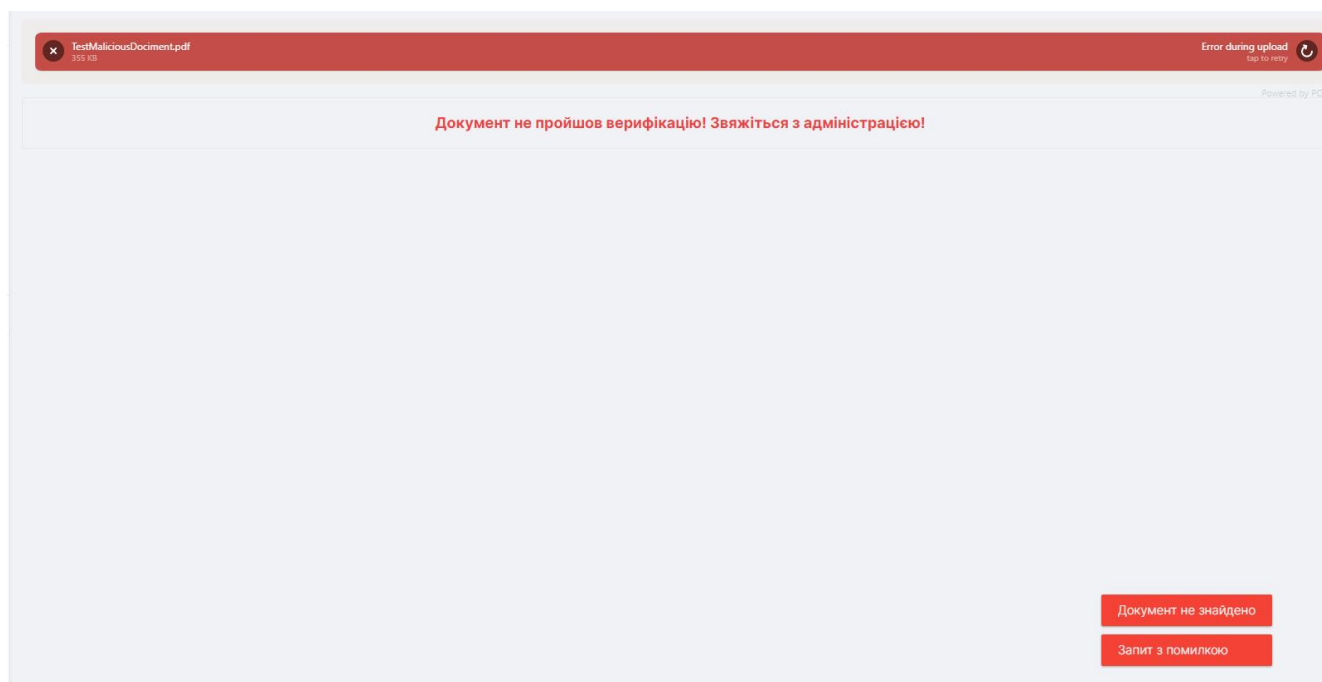


Рисунок 3.4.7 – Невидала верифікація документу

3.6 Здійснення неправомірних дій стосовно блокчейну

Зміна інформації у блокчейні це достатньо важка робота, тому що кожен наступний блок містить інформацію попереднього блоку і таким чином створюється ланцюг, який в цілому неможливу “розірвати”. Давайте спробуємо змодельовати

		Марушак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

зміну певного блоку у нашому блокчейні (приклад “Моделювання зміни інформації у блокчейні” рис. 3.4.8).

```
console.log('Не підлягав зміні зміни: ${blockchain.isChainValid()}');

console.log('Здійснюємо модифікацію хешу документ 3 блоку');

blockchain.chain[2].document.documentHash = '4ea5c508a6566e76240543f8feb06fd457777be39549c4016436afda65d2123e'

console.log('Не підлягав зміні зміни: ${blockchain.isChainValid()}');
```

Рисунок 3.4.8 – Моделювання зміни інформації у блокчейні

```
Не підлягав зміні зміни: true
Здійснюємо модифікацію хешу документ 3 блоку
Не підлягав зміні зміни: false
```

Рисунок 3.4.9 – Результат перевірки на цілісність блокчейну

Висновок до третього розділу

Під час виконання третього розділу було налаштовано сервер Nginx, а саме: приховування чутливої інформації, CSP, HSTS, налаштовано передавання трафіку за протоколом HTTPS. Проведено практичне налаштування SSL сертифікатів за допомогою.

Були проведені тестування на XSS атаку, викрадення токена користувача, неправомірна модифікація документу, несанкціоновані зміни у блокчейні. Було реалізовано заходи, щодо попередження цих атак, а саме наведені фрагменти коду, скришоти результатів виконання тестів.

ВИСНОВКИ

Дослідженні основні шляхи з захисту веб-серверу, Web API, Web-додатків, методи та засоби запобігання найбільш розповсюджених атак на них, а саме було наведено частини вразливого коду Web-додатків до веб-атак, проілюстровано схеми за якими кіберзлочинець може здійснювати неправомірні дії стосовно нашого веб-сервісу. Дослідили технологію блокчейн, а саме особливості захисту та шляхи використання, де ця технологія буде незамінною.

Були представлені можливі шаблони конфігурацій серверу NGINX для запобігання витоку чутливої інформації, захисту від XSS та CSRF атак, конфігурації внутрішніх систем, створене та чітко поставлене технічне завдання: створити сервіс через, який можна зробити онлайн замовлення довідок, забезпечити безпеку користувацьких даних, створити зручний веб-інтерфейс, створити можливість верифікувати документ, чи дійсно певна організація його видала, валідація документу на спробу модифікації, створення децентралізованого сховища, яке б зберігало в собі інформацію по всім документам, які були видані організацією, шифрування на підпис документів організацією. Створена діаграма БД, обрано систему керування доступом RBAC з наступними ролями: адміністратор, менеджер, студент, викладач. Використано брандмауер iptables для заборони незахищеного HTTP трафіку. Створено API сервіс у якому було налаштовано middlewares для валідації вхідних даних у запитах, throttling сервіс, щоб захистити сервіс від DDOS атак та дуже часто повторюваних запитів. Детально розібрано функціонування та роботу сервісу блокчейну, технології та реалізовано сервіс для верифікації цілісності документів їх підпису та захист від модифікацій. Були наведені приклади з коду, які використовуються у всьому сервісі.

Було налаштовано сервер Nginx, а саме: приховування чутливої інформації, CSP, HSTS, налаштовано передавання трафіку за протоколом HTTPS. Проведено практичне налаштування SSL сертифікатів за допомогою certbot.

Були проведені тестування на XSS атаку, викрадення токenu користувача, неправомірна модифікація документу, несанкціоновані зміни у блокчейні. Було

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

реалізовано заходи, щодо попередження цих атак, а саме наведені фрагменти коду, скришоти результатів виконання тестів.

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Стаффер М. Laravel Up & Running: A Framework for Building Modern PHP Apps / Метт Стаффер.. – 512 с. – (2). – (Бестселлери O'Reilly).
2. Stewart R. Laravel: The Ultimate Beginner's Guide to Learn Laravel Step by Step / Rufus Stewart., 2020. – (137).
3. Hoffman A. Web Application Security: Exploitation and Countermeasures for Modern Web Applications / Andrew Hoffman., 2020. – 330 с. – (1).
4. McDonald M. Web Security for Developers: Real Threats, Practical Defense / Malcom McDonald., 2020. – 216 с. – (1).
5. 6 Web Backend Security Risks to Consider in Development [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://geekflare.com/web-backend-security-risk/>.
6. Laravel Documentation [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: <https://laravel.com>
7. Why Laravel is the Recommended Framework for Secure, Mission-Critical Applications [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: <https://auth0.com/blog/why-laravel-is-the-recommended-framework-for-secure-mission-critical-applications/>.
8. An Overview of the Best Laravel Security Practices [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: <https://www.cloudways.com/blog/laravel-security/#laravel-authentication>.
9. Блокчейн [Електронний ресурс] - https://bankchart.com.ua/finansoviy_gid/investitsiyi/statti/scho_take_blokcheyn_#3
10. CRLF [Електронний ресурс] - <https://cryptoworld.su/chto-takoe-crlf-inekciya-primery-ispolzovaniya-uroki-xakinga-glava-4/>
11. REST API [Електронний ресурс] - <https://webdevblog.ru/kak-obespechit-bezopasnost-rest-api/>
12. XSS [Електронний ресурс] - <https://school38vrn.ru/uk/zashchita-ot-xss-atak-easy-hack-kak-dobyty-dannye-cherez-cross-site-scripting-inclusion.html>

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

13. NGINX [Електроний ресурс] - <https://docs.nginx.com/>

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

Клас Block:

```
import SHA256 from 'crypto-js/sha256.js'

export class Block {
  constructor(timestamp, document, previousHash = "") {
    this.timestamp = timestamp
    this.document = document
    this.previousHash = previousHash
    this.hash = this.calculateHash()
    this.nonce = 0
  }

  calculateHash () {
    return SHA256(this.timestamp + this.document + this.previousHash + this.nonce).toString()
  }

  mineBlock (difficulty) {
    while (this.hash.substring(0, difficulty) !== Array(difficulty + 1).join('0')) {
      this.nonce++
      this.hash = this.calculateHash()
    }

    console.log(`Хеш блоку: ${this.hash}`)
    console.log(`Proof-of-work: ${this.nonce}`)
    console.log(`Шаблон збігу: ${Array(difficulty + 1).join('0')}`)
  }

  isValidDocument () {
    return this.document.isValid()
  }
}
```

Клас Blockchain:

```
import { Block } from './Block.js'

export class Blockchain {
  constructor(blocks, difficulty = 2) {
    this.difficulty = difficulty
    this.chain = blocks.length > 0 ? blocks : [this.createGenesisBlock()]
  }

  createGenesisBlock() {
    return new Block(new Date().getTime(), null, "0")
  }

  getLatestBlock() {
    return this.chain[this.chain.length - 1]
  }

  addDocument (document, timestamp) {
    if(!document.documentHash) {
      throw new Error('Document must include document hash!')
    }
    if(!document.isValid()) {
      throw new Error('Cannot add invalid document to chain')
    }
  }
```

		Маруцак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пироз О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

let block = new Block(timestamp, document, this.getLatestBlock().hash)
block.mineBlock(this.difficulty)
this.chain.push(block)
}

isChainValid() {
  for (let i = 1; i < this.chain.length; i++){
    const currentBlock = this.chain[i]
    const previousBlock = this.chain[i - 1]

    if(!currentBlock.isValidDocument()) {
      return false
    }

    if(currentBlock.hash !== currentBlock.calculateHash()) {
      return false
    }

    if(currentBlock.previousHash !== previousBlock.hash) {
      return false
    }
  }

  return true
}

verifyDocument (signature) {
  if(!this.isChainValid()) {
    throw new Error('Chain is not valid!')
  }

  return this.chain.filter(el => el?.document?.signature === signature)
}
}

```

Клас Block:

```

import SHA256 from 'crypto-js/sha256.js'

import EC from "elliptic";
const ec = new EC.ec('ed25519')

export class Document {
  constructor(documentHash) {
    this.documentHash = documentHash
  }

  calculateHash () {
    return SHA256(this.documentHash).toString()
  }

  signHandler(signingKeys) {
    if(signingKeys.getPublic('hex') !== process.env.SIGN_PUBLIC_KEY) {
      throw new Error('You cannot sign documents for other companies')
    }

    const hash = this.calculateHash()
    const sign = signingKeys.sign(hash, 'base64')

    this.signature = sign.toDER('hex')
  }
}

```

		Маруцак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

isValid () {
  if(!this.signature || this.signature.length === 0) {
    throw new Error('No signature in this document')
  }

  const publicKey = ec.keyFromPublic(process.env.SIGN_PUBLIC_KEY, 'hex')

  return publicKey.verify(this.calculateHash(), this.signature)
}
}

```

Код допоміжних функцій блокчейну:

```

import EC from "elliptic";
const ec = new EC.ec('ed25519')

const key = ec.genKeyPair()

const publicKey = key.getPublic('hex')
const privateKey = key.getPrivate('hex')

console.log(`Private key ${privateKey}`)
console.log()
console.log(`Public key ${publicKey}`)

import SHA256 from 'crypto-js/sha256.js'

import EC from "elliptic";
const ec = new EC.ec('ed25519')

import {Blockchain} from '../blockchain/Blockchain.js'
import { Document } from '../blockchain/Document.js'

const myKey = ec.keyFromPrivate(process.env.SIGN_PRIVATE_KEY)
const myWalletAddress = myKey.getPublic('hex')

const blockchain = new Blockchain([], 2)

const addDocument = ({documentHash}) => {
  const document = new Document(documentHash.toString())
  document.signHandler(myKey)
  blockchain.addDocument(document, Date.now())
  return blockchain.getLatestBlock()
}

const getBlockchain = () => {
  return blockchain.chain
}

const isBlockchainValid = () => {
  return blockchain.isChainValid()
}

const verifyDocument = ({documentHash}) => {
  const document = new Document(documentHash)
  document.signHandler(myKey)

  return blockchain.verifyDocument(document.signature)
}

export {

```

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пироз О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		88

```

    addDocument, getBlockchain, isBlockchainValid, verifyDocument
  }

```

Клас для обробки API відповідей:

```

const successResponse = (data, message = 'Запит успішний', status = 200) => {
  return {
    status,
    message,
    data,
  }
}

const errorResponse = (errors = {}, message = 'Запит з помилкою', status = 400) => {
  return {
    status,
    message,
    errors,
  }
}

export { successResponse, errorResponse }

```

Код для додавання водної марки:

```

import fs from 'fs'
import { v4 as uuidv4 } from 'uuid';
import path from 'path';
import { degrees, encodeToBase64, PDFDocument, rgb } from 'pdf-lib';

const readPdf = async (pathFile, filename, signingKeys) => {
  const __dirname = path.resolve();

  const existingPdfBytes = fs.readFileSync(pathFile)
  const pdfDoc = await PDFDocument.load(existingPdfBytes)
  const pages = pdfDoc.getPages()

  const page = pdfDoc.addPage()

  const { width, height } = page.getSize()

  const approveHash = uuidv4()

  const sign = signingKeys.sign(approveHash, 'base64')

  const signature = sign.toDER('hex')

  page.drawText(`This document has been approved by: Zhytomyr Polytechnic State University`, {
    x: 10,
    y: height - 50,
    size: 12,
    color: rgb(0, 0, 0)
  })

  page.drawText(signature, {
    x: 10,
    y: height - 60,
    size: 5,
    color: rgb(0, 0, 0)
  })
}

```

		Маруцак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

```

    })

    const res = await pdfDoc.save()
    const storeFile = `verifyDocument/${filename}`;

    fs.writeFileSync(storeFile, res, (e) => {
        console.log(e);
    });

    return `${__dirname}/${storeFile}`
}

export default readPdf

```

Код роутів Node.js:

```

import 'dotenv/config'
import express from 'express'

import EC from "elliptic";
const ec = new EC.ec('ed25519')

import readPdf from './utils/pdf.js'
import { successResponse, errorResponse } from './utils/apiResponse.js'
import { addDocument, getBlockchain, isBlockchainValid, verifyDocument } from './utils/blockchain.js'
import { v4 as uuidv4 } from 'uuid';

const app = express()
const PORT = process.env.PORT || 3050

app.use(express.json())

app.post('/blocks', (req, res) => {
    if(!req.body.document_hash) {
        res.status(400).json(errorResponse({
            'document_hash': 'Обовязкове поле'
        }))
    }

    const document = addDocument({documentHash: req.body.document_hash})

    res.json(successResponse(document))
})

app.get('/verifyDocument', (req, res) => {

    if(!req.query.document_hash) {
        return res.status(400).json(errorResponse({
            'document_hash': 'Обовязкове поле'
        }))
    }

    const document = verifyDocument({documentHash: req.query.document_hash})

    if(!document.length) {
        return res.status(400).json(errorResponse({
            'document': [
                'Документ не знайдено'
            ]
        }))
    }
}

```

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пироз О.В.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    res.json(successResponse(document))
  })

app.post('/watermarkDocument', (req, res) => {

  if(!req.body.path_file) {
    return res.status(400).json(errorResponse({
      'path_file': 'Обовязкове поле'
    }))
  }

  const url = req.body.path_file
  const myKey = ec.keyFromPrivate(process.env.SIGN_PRIVATE_KEY)

  readPdf(url, uuidv4()+'.pdf', myKey).then(ress => {
    return res.json(successResponse({
      file_path: ress
    }))
  }).catch(err => {
    return res.status(400).json(errorResponse({
      'error': [
        err.message
      ]
    }))
  })
})

app.get('/blocks', (req, res) => {
  res.json(successResponse(getBlockchain()))
})

app.get('/blocksValid', (req, res) => {
  res.json(successResponse(isBlockchainValid()))
})

app.get('/test', (req, res) => {
  const myWalletAddress = myKey.getPublic('hex')
  readPdf('http://127.0.0.1:8000/certificates/uploads/4/e7b7f72b-b27a-4543-89f0-f37a3b35cf13.pdf', 'test.pdf',
  myKey).then(ress => {

    res.send('Документ створено')
  }).catch(err => {
    res.status(400).send('Помилка ' + err.message)
  })
})

app.listen(PORT, () => {
  console.log(`Server is up on port ${PORT}`)
})

```

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пироз О.В.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

Middleware перевірки айпі дареси:

```

<?php
namespace App\Http\Middleware;
use App\Http\Controllers\Api\ApiController;
use Closure;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;

class CheckUserIpAddress
{
    /**
     * Handle an incoming request.
     *
     * @param Request $request
     * @param Closure(\Illuminate\Http\Request): (\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse) $next
     * @return JsonResponse
     */
    public function handle(Request $request, Closure $next)
    {
        if($request->user() && in_array($request->ip(), $request->user()->user_api_addresses->pluck('ip_address')->toArray())) {
            return $next($request);
        } else if(!$request->user()){
            return $next($request);
        }

        $request->user()->currentAccessToken()->delete();

        return (new ApiController()->sendError('Будь ласка увідіть заново, ваша IP адреса змінилась', [], 401));
    }
}

```

Код для захисту від XSS атаки:

```

<?php
namespace App\Http\Middleware;
use Closure;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;

class CustomXssProtection
{
    /**
     * Handle an incoming request.
     *
     * @param Request $request
     * @param Closure(\Illuminate\Http\Request): (\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse) $next
     * @return Response|RedirectResponse
     */
    public function handle(Request $request, Closure $next)
    {
        $fields = $request->all();

        // htmlspecialchars
        foreach ($fields as $key => $field) {
            $request->merge([
                $key => strip_tags($field)
            ]);
        }

        return $next($request);
    }
}

```

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        });
    }

    return $next($request);
}
}

```

Код контролеру авторизації:

```

<?php
namespace App\Http\Controllers\Api\Auth;
use App\Http\Controllers\Api\ApiController;
use App\Http\Requests\Api\Auth/LoginRequest;
use App\Models\UserApiAddress;
use App\Schemes\DefaultSchema;
use Illuminate\Http\JsonResponse;
use Illuminate\Support\Collection;
use Illuminate\Support\Facades\Auth;
use Laravel\Sanctum\NewAccessToken;

class LoginController extends ApiController
{
    /**
     * Handle the incoming request.
     *
     * @param LoginRequest $request
     * @return JsonResponse
     */
    public function __invoke(LoginRequest $request)
    {
        if(!Auth::attempt($this->getCredential($request))) {
            return $this->sendError('Користувач не знайдений', [],401);
        }

        $token = $this->createToken($request);
        UserApiAddress::create(
            [
                'ip_address' => $request->ip(),
                'user_id' => Auth::user()->id,
            ]
        );
        return $this->sendResponse($token);
    }

    /**
     * @param LoginRequest $request
     * @return array
     */
    private function getCredential (LoginRequest $request): array
    {
        return $request->only('email', 'password');
    }

    /**
     * @param LoginRequest $request
     * @return NewAccessToken
     */
    private function createToken (LoginRequest $request): NewAccessToken
    {
        return Auth::user()->createToken(DefaultSchema::TOKEN_USER_NAME, Auth::user()->getPermissions());
    }
}

```

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

```
}
```

Код контролеру реєстрації:

```
<?php
namespace App\Http\Controllers\Api\Auth;
use App\Http\Controllers\Api\ApiController;
use App\Http\Requests\Api\Auth\RegisterRequest;
use App\Models\User;
use App\Schemas\DefaultSchema;
use Illuminate\Http\JsonResponse;

class RegisterController extends ApiController
{
    /**
     * Handle the incoming request.
     *
     * @param RegisterRequest $request
     * @return JsonResponse
     */
    public function __invoke(RegisterRequest $request)
    {
        $user = User::create(array_merge($this->getCredential($request), [
            'password' => bcrypt($request->get('password'))
        ]));

        $token = $user->createToken(DefaultSchema::TOKEN_USER_NAME);

        return $this->sendResponse($token);
    }

    private function getCredential (RegisterRequest $request) {
        return array_merge($request->only('email', 'first_name', 'gender', 'last_name', 'middle_name'), ['role_id' => 2]);
    }
}
```

Код контролеру відновлення пароля:

```
<?php
namespace App\Http\Controllers\Api\Auth;
use App\Http\Controllers\Api\ApiController;
use App\Models\User;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class ResetPassword extends ApiController
{
    /**
     * Handle the incoming request.
     *
     * @param Request $request
     * @return JsonResponse
     */
    public function __invoke(Request $request)
    {
        $request->validate([
            'email' => 'required|email',
            'password' => 'required|string|min:8|confirmed',
            'password_confirmation' => 'required',
            'token' => 'required|string'
        ]);
    }
}
```

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				94
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    });

$updatePassword = DB::table('password_resets')->where($this->getCredential($request))->first();

if(!$updatePassword) return $this->sendError('Упссс, щось пішло не так...', [
    'token' => [
        'Токен не знайдено'
    ]
]);

$this->updatePassword($request);
$this->deleteToken($request);

return $this->sendResponse([], 'Пароль успішно відновлено, використайте його під час авторизації!');
}

private function getCredential (Request $request) {
    return $request->only('email', 'token');
}

private function updatePassword (Request $request) {
    User::where('email', $request->get('email'))->update(['password' => bcrypt($request->get('password'))]);
}

private function deleteToken (Request $request) {
    DB::table('password_resets')->where(['email'=> $request->get('email')])>delete();
}
}

```

Код контролеру керування замовленнями:

```

<?php
namespace App\Http\Controllers\Api\Certificate;
use App\Http\Controllers\Api\ApiController;
use App\Http\Controllers\Controller;
use App\Http\Requests\Certificate\StoreCertificateOrderRequest;
use App\Models\Certificate\CertificateOrder;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;

class CertificateOrderController extends ApiController
{
    /**
     * Display a listing of the resource.
     *
     * @return JsonResponse
     */
    public function index(Request $request, $user_id)
    {
        $items = CertificateOrder::where([
            'user_id' => $user_id
        ])->when($request->get('status'), function ($q, $value) {
            return $q->where('status', $value);
        })->orderBy('id', 'desc')->paginate($request->get('per_page', 10));

        return $this->sendResponse($items);
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param StoreCertificateOrderRequest $request
     */
}

```

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пирог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		95

```

* @param $user_id
* @return JsonResponse
*/
public function store(StoreCertificateOrderRequest $request, $user_id)
{
    $item = CertificateOrder::create(array_merge(
        $request->only('certificate_id', 'channel', 'channel_value', 'doc_file_id'),
        [
            'status' => 'new',
            'user_id' => $user_id
        ]
    ));

    return $this->sendResponse($item);
}

```

```

/**
 * Display the specified resource.
 *
 * @param $user_id
 * @param $order_id
 * @return JsonResponse
 */
public function show($user_id, $order_id)
{
    $order = CertificateOrder::where([
        'user_id' => $user_id,
        'id' => $order_id,
    ]->first());

    if(!$order) return $this->sendError('Замовлення не знайдено');

    return $this->sendResponse($order);
}

```

```

/**
 * Update the specified resource in storage.
 *
 * @param Request $request
 * @param $user_id
 * @param $order_id
 * @return JsonResponse
 */
public function update(Request $request, $user_id, $order_id)
{
    $order = CertificateOrder::where([
        'user_id' => $user_id,
        'id' => $order_id,
    ]->first());

    if(!$order) return $this->sendError('Замовлення не знайдено');

    $order->update([
        'channel' => $request->get('channel'),
        'channel_value' => $request->get('channel_value'),
    ]);

    return $this->sendResponse($order);
}

```

```

/**
 * Remove the specified resource from storage.

```

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пірог О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		96

```

*
* @param int $id
* @return \Illuminate\Http\Response
*/
public function destroy($id)
{
    //
}
}

```

Код верифікації документу:

```

<?php
namespace App\Http\Controllers\Api\Verify;
use App\Facades\FileUpload\FileUploadFacade;
use App\Http\Controllers\Api\ApiController;
use App\Http\Controllers\Controller;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;
use Illuminate\Support\Facades\Storage;

class VerifyCertificateController extends ApiController
{
    public function verify (Request $request) {
        $request->validate([
            'upload_file' => ['required','mimes:pdf,docx','max:102400'],
        ]);

        $file = FileUploadFacade::upload($request->file('upload_file'), 'verify_certificates', $request->user()->id);

        $filePath = Storage::disk($file->disk)->path($file->path);

        $hashFile = hash_file('sha256', $filePath);

        $hash = hash('sha256', $hashFile);

        try {
            $result = Http::withHeaders([
                'Accept' => 'application/json'
            ]->get('http://127.0.0.1:3050/verifyDocument', [
                'document_hash' => $hash
            ]);

            $data = $result->json();

            if($result->status() === 200) {
                return $this->sendResponse($data['data']);
            }

            return $this->sendError($data['message'], $data['errors']);

        } catch (\Exception $exception){

            return $this->sendError('Документ не знайдено', [
                'error' => [$exception->getMessage()]
            ]);
        }
    }

    public function watermark (Request $request) {
        $request->validate([

```

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пироз О.В.				97
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        'file_url' => 'required|string'
    ));

    try {
        $result = Http::withHeaders([
            'Accept' => 'application/json'
        ])->timeout(60)->post('http://127.0.0.1:3050/watermarkDocument', [
            'file_url' => $request->get('file_url')
        ]);

        $data = $result->json();

        if($result->status() === 200) {
            return $this->sendResponse($data['data']);
        }

        return $this->sendError($data['message'], $data['errors']);
    } catch (\Exception $exception){

        return $this->sendError('Помилка', [
            'error' => [$exception->getMessage()]
        ]);
    }
}

```

		Марущак Ю.Р.			КІК.КР.Б – 125 – 12 – ПЗ	Арк.
		Пироз О.В.				98
Змн.	Арк.	№ докум.	Підпис	Дата		