МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ С.В. Казмірчук
«_____»_____ 20__ р.

На правах рукопису
УДК 004.738.52

ДИПЛОМНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «БАКАЛАВР»

Тема: Метод захисту Web-сторінок Інтернет магазину

Виконавець: Власов В.М.
Керівник: к.т.н., доцент Гулак Н.К.
Нормоконтролер: к.т.н., доцент Гулак Н.К.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки, комп'ютерної та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Бакалавр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ С.В. Казмірчук

«__» _____ 20__ р.

ЗАВДАННЯ

на виконання дипломної роботи

здобувача вищої освіти Власова Владислава Миколайовича

1. Тема: *Метод захисту Веб-сторінок Інтернет магазину*

затверджена наказом ректора від «26» квітня 2021 р. № 652/ст.

2. Термін виконання: з 10.05.2021 р. по 20.06.2021 р.

3. Вихідні дані: стандартні засоби захисту Веб-сторінок Інтернет магазину; типи атак на веб-сторінки; моделі порушників інформаційної безпеки; методи захисту Веб-сторінок від мережевих атак.

4. Зміст пояснювальної записки: аналіз стандартних засобів захисту Веб-сторінок Інтернет магазину; дослідження методів захисту Веб-сторінок Інтернет магазину від атак; розробка та тестування захисту Веб-сторінок Інтернет магазину комбінацією методів екранування та приведення до цілочисельного типу.

КАЛЕНДАРНИЙ ПЛАН

виконання дипломної роботи

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	19.04.2021	*Виконано*
2.	Аналіз літературних джерел	20.04-01.05.2021	*Виконано*
3.	Обґрунтування вибору рішення	02-03.05.2021	*Виконано*
4.	Збір інформації	03-08.05.2021	*Виконано*
5.	Дослідження сучасних систем і методик аналізу та оцінки ризиків інформаційної безпеки	09-12.05.2021	*Виконано*
6.	Розробка методики та структури системи аналізу та оцінки ризиків інформаційної безпеки	13-20.05.2021	*Виконано*
7.	Розробка алгоритму та програмного забезпечення системи аналізу та оцінки ризиків інформаційної безпеки	21-30.05.2021	*Виконано*
8.	Перевірка на антиплагіат	02.05.2021	*Виконано*
9.	Оформлення і друк пояснювальної записки	30.05-4.06.2021	*Виконано*
10.	Оформлення презентації	5-7.06.2021	*Виконано*
11.	Отримання рецензій від рецензента	8-11.06.2021	*Виконано*

Здобувач вищої освіти Власов В.М.

(підпис, дата)

Керівник дипломної роботи Гулак Н.К.

(підпис, дата)

РЕФЕРАТ

Дипломна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків і має 53 сторінки основного тексту, 24 рисунка, 5 таблиць. Список використаних джерел містить 20 найменування. Загальний обсяг роботи 54 сторінок.

Мета роботи – розробка методу захисту Веб-сторінок Інтернет магазину, на основі методів екранування та приведення до цілочисельного типу.

Проаналізовано основні моменти захисту веб-сторінок Інтернет магазину, проаналізовано веб-атаки, принцип за яким реалізуються веб-атаки та можливі варіанти захисту від них. Також наведено список інструментів для моніторингу безпеки веб-додатків. На основі комбінації методів екранування та приведення до цілочисельного типу було розроблено і протестовано метод захисту веб-сторінок Інтернет магазину.

Розроблений метод та програмне забезпечення відносяться до галузі інформаційної безпеки і можуть бути використані для підвищення рівня захищеності Web-сторінок.

Ключові слова: ризик, система аналізу і оцінки ризиків, інформаційна безпека, інформаційно-комунікаційна система, функція належності, нечіткі числа, нечіткі еталони, актив, загроза, оцінюючі компоненти, ступінь ризиків,

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

XSS – Cross Site Scripting – міжсайтовий скриптинг;

DoS – Denial of Service – відмова в обслуговуванні;

SQL – Structured query language – мова структурованих запитів;

B2B – Business-to-business– бізнес до бізнесу;

CMS – Content Management System –система управління контентом сайта;

ОС – Операційна система;

БД – База даних.

СБУ – Служба безпеки Укріїни

ВСТУП

Актуальність. В реаліях сучасності третину світового ВВП займають саме інформаційні ресурси. Інформація перетворюється на товар та фактор впливу. В останні роки в Україні та у всьому світі зростає кількість кібератак та кіберзлочинів. Кожна компанія, кожна ланка суспільства володіє конфіденційною інформацією, витоки якої призводять до колосальних збитків та непоправних наслідків.

Для протидії кіберзлочинності великого значення набуває розробка методів та систем для захисту інформації. Кожна інформаційна система може бути захищеною, але з розвитком несанкціонованого доступу цей захист стає умовним і потребує удосконалення. Тому, тема дипломної роботи є досить актуальною на сьогодні.

Метою дипломної роботи є розробка методу захисту Веб-сторінок Інтернет магазину, на основі методів екранування та приведення до цілочисельного типу.

Досягнення мети потребує розв’язання таких **задач**:

- аналіз стандартних засобів захисту Web-сторінок Інтернет магазину;
- дослідження методів захисту Web-сторінок Інтернет магазину від атак;
- розробка та тестування комбінації методів захисту Веб-сторінок Інтернет магазину.

Об’єкт дослідження: процес захисту Веб-сторінок Інтернет магазину від мережевих атак.

Предмет дослідження: методи захисту Веб-сторінок від мережевих атак.

Практична цінність на основі методів екранування та приведення до цілочисельного типу було розробленно та протестовано метод захисту інформації від мережевих атак Веб-сторінок, який може використовуватися для захисту інформації в веб-сторінках.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Статистика кіберзагроз в Україні у 2020 році

Викрадення інформації залишається пріоритетом для переважної більшості кібератак. У 30% та 42% нападів на юридичних та фізичних осіб зловмисники вимагають фінансових переваг. Високу частку фінансово мотивованих атак на людей можна пояснити регулярним масовим знищенням шкідливих програм за допомогою настирливих рекламних матеріалів, у тому числі на мобільних пристроях, зараженням неповнолітніх та іншим антивірусним програмним забезпеченням на сумнівних сайтах, а також кампаніями, що вимагають готівки, в яких зловмисники погрожують розповсюдженням компрометуючих персональних даних.

Особисті дані та дані доступу представляють великий інтерес для зловмисників, якщо вони нападають на юридичних осіб. Це пов’язано з тим, що компанії зберігають великі бази персональних даних та інформації про клієнтів. Крім того, зловмисників може зацікавити доступ до даних працівників компанії-жертви. Облікові записи в соціальних мережах також піддаються ризику, особливо якщо особистий кабінет популярний, тобто має велику кількість підписників. Користувачі, в свою чергу, нехтують безпекою особистих акаунтів: використовують нестабільні та однакові паролі, вводять ідентифікатори, не гарантуючи надійності ресурсу, публікують інформацію про себе, яка може допомогти вам вибрати пароль. Це пояснює високу частку викрадених ідентифікаторів (44%) в атаках на людей. Наприклад, шанувальники комп’ютерних ігор є серед тих людей, які перебувають у зоні підвищеного ризику хакерських атак. У другому кварталі зловмисники заманили користувачів Steam на веб-ресурси, де вони можуть безкоштовно отримати нову гру, ввівши дані облікового запису Steam. Крім того, гравці

можуть потрапити на приманку зловмисників на спеціалізованих форумах. Під виглядом чіт-кодів, упакованих в ZIP-архів, багато веб-ресурсів поширюють вірус-троян, щоб підірвати криптовалюту TurtleCoin. Дані кредитних карток та платіжні дані клієнтів зазвичай захищені криптографічними процедурами, так що зловмисникам легше дізнатися про них безпосередньо від клієнта, використовуючи методи соціальної інженерії. Як результат, 34% даних, викрадених після атак на фізичних осіб, - це дані з їх банківської картки.

На Рисунку 1.1. зображена діаграма основних об’єктів атак в нападах на фізичних та юридичних осіб.

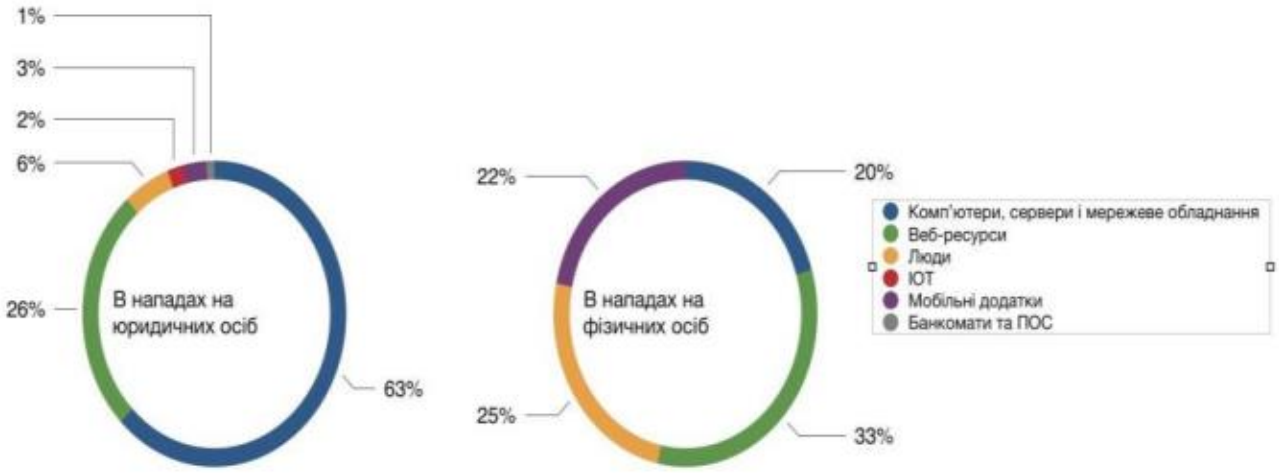

Рис. 1.1. Об'єкти атак

У II кварталі 2020 року частка цілеспрямованих атак зросла у порівнянні з I кварталом і становила 59% (у I кварталі – 47%). Частка кібер-інцидентів, в результаті яких постраждали приватні особи, становила 24%. Серед юридичних осіб зловмисники найчастіше атакують державні організації, промислові компанії, медичні організації, банки та інші організації фінансової сфери. У II кварталі відзначають атаки на організації з управління ланцюгами поставок (supply chain) на великі IT-компанії з клієнтами з різних галузей економіки.

1.2. Поняття Інтернет магазину та його основні вразливості

Інтернет-магазин (англ. Online shop або e-shop) – сайт, який здійснює торгівлю товарами за допомогою мережі Інтернет. Дозволяє користувачам онлайн, в своєму браузері або через мобільний додаток, сформувати замовлення на покупку, вибрати спосіб оплати і доставки замовлення, оплатити замовлення. При цьому продаж товарів здійснюється дистанційним способом і вона накладає обмеження на товари, що продаються. Так, в деяких країнах є заборона на інтернет-торгівлю алкоголем, зброєю, ювелірними виробами та іншими товарами (наприклад, в деяких країнах заборонена дистанційний продаж алкоголю та інших товарів, вільна реалізація яких заборонена або обмежена).

Коли інтернет-магазин створюється для підприємств, які купують у інших підприємств, це називається онлайн-покупками між підприємствами (B2B). Типовий інтернет-магазин дозволяє покупцеві переглядати асортимент продуктів і послуг компанії, переглядати фотографії або зображення продуктів, а також отримувати інформацію про характеристики продуктів і цінах.

Інтернет-магазини зазвичай дозволяють покупцям використовувати «функції пошуку» для пошуку певної моделі, бренду або товару.

По суті, інтернет-магазин - це веб-додаток, що складається з клієнтської і серверної частин, що реалізують технологію «клієнт-сервер». Клієнтська частина реалізує інтерфейс програми, відправляє запити на сервер і обробляє відповіді на ці запити. Сторона сервера отримує запит, виконує обчислення за запитом, а потім створює веб-сторінку, яку може бачити користувач. Додаток може функціонувати як клієнт інших служб, наприклад, бази даних або іншого веб-додатки, розташованого на іншому сервері.

Набагато простіше вирішити проблеми на етапі проектування, ніж вирішувати їх під час роботи готового веб-ресурсу. Дерева атак дозволяють моделювати можливі атаки і перебої в роботі ресурсів і знаходити способи їх усунення для захисту інтернет-магазину.

Більшість атак на веб-сайти здійснюються автоматично з використанням автоматичних сканерів і аналогічних інструментів або спеціалізованого програмного забезпечення, здатного виявляти «уразливості» в безпеки веб-сайту. Поява в програмному забезпеченні веб-сайту CMS робить цей веб-сайт менш безпечним. Сайт управляється http-сервером. HTTP-сервери скануються на наявність вразливостей, але вони теж завжди знаходять нові «дірки». Кожна CMS встановлюється на додаток до http сервера, додаючи уразливості CMS до існуючих. Однак той факт, що CMS робить сайт менш безпечним, не повинен бути перешкодою для створення CMS.

Рисунок 1.2. ілюструє нам статистику видів загроз для Web-сайтів

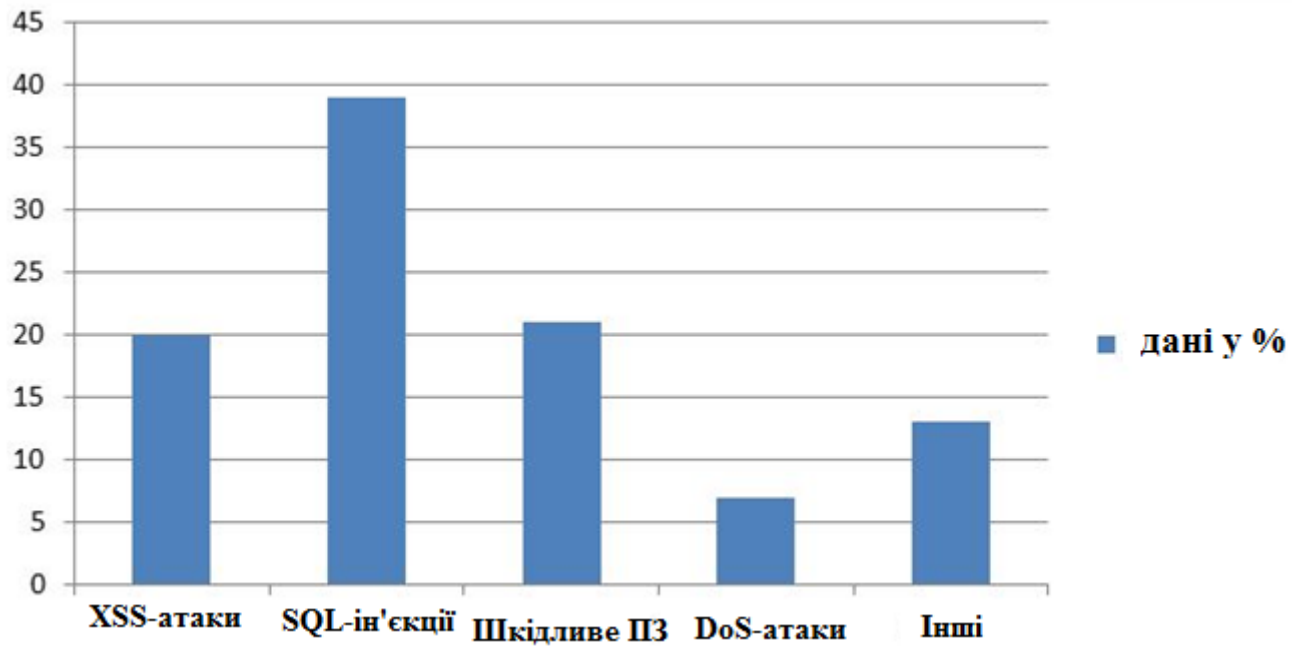

Рис. 1.2. Найбільш поширені види загроз для веб-сайтів

На практиці управління ризиками є більш важливим. Навіть безпечний Інтернет-магазин можна зламати, оскільки власник встановив пароль, який легко вгадати. Таким чином, ризики, пов'язані з використанням системи управління вмістом, можуть бути набагато нижчими, ніж ризики, пов'язані з організацією веб-ресурсу.

Безпека веб-ресурсу визначається як ступінь доцільності його механізмів захисту інформації (ідентифікація, автентифікація, контроль доступу,

реєстрація та аудит, криптографія, екранування), відповідність заходів безпеки ризикам, що існують у цьому середовищі, здатність механізмів безпеки забезпечувати конфіденційність, цілісність та забезпечувати доступність інформації. Сьогодні найефективнішим механізмом створення моделей захисту є дерево атак.

Дерева атак - це схеми, на яких показано, як можна атакувати ціль. У галузі інформаційних технологій вони використовуються для опису потенційних загроз для комп'ютерної системи та можливих методів атак, що реалізують ці загрози (рисунок 1.2).

Погроза - це сукупність умов та факторів, що визначають потенційний або фактичний ризик інциденту, який може вплинути на функціонування об'єкта або окремого захищеного веб-ресурсу. Загрози можна класифікувати з ряду причин. Зокрема, типи загроз поділяються на дві групи: умисні та природні.

Основними загрозами вважаються наступні:

- підключення до каналів зв'язку;
- несанкціонований доступ;
- розкрадання носіїв інформації.

На таблиці 1.1 наведена статистика загроз та їх розподіл щодо веб-ресурсів

№	Загроза інформаційної безпеки	% загроз	% вразливих сайтів
1	Міжсайтове кодування (XSS-атаки)	19,23	27,27
2	SQL-ін'єкція	17,25	49,35
3	Неправильна конфігурація сервера	11,09	37,36
4	Шкідливе програмне забезпечення	12,44	37,66
5	Підробка міжсайтових запитів (CSRF)	2	7,79
6	Виклик виняткових ситуацій	11,54	20,78
7	Інші	26,05	50

Таблиця 1.1

Запобігти природним загрозам майже неможливо, оскільки більшість із них є природними. Звести крадіжку медіа до нуля - надзвичайне завдання. Щоб

зменшити ризик, потрібно вжити примітивних заходів для захисту фізичних об’єктів.

На рисунку 1.3 зображена схема атак, з домогою яких зловмисник може отримати фінансову винагороду.

Рис. 1.3. Дерево атак на інтернет-магазин

Щоб уникнути атак, що частіші за інші (підключення до каналів зв'язку), ефективніше використовувати більш поширені CMS, у цьому випадку дещо вищу ймовірність того, що вразливості вже були виявлені та закриті. Однак немає жодних гарантій того, що в майбутньому в системі не буде виявлено нових «дірок».

Найкращий вибір - це система, яка постійно зростає. Якщо розробники припинять розробку системи, ймовірно, що нещодавно виявлена вразливість не буде виправлена вчасно.

Також необхідно контролювати всі оновлення системи управління вмістом з акцентом на безпеку та своєчасно встановлювати ці оновлення. Деякі сучасні системи управління вмістом, такі як "1С: Бітрікс", дозволяють автоматизувати процес перевірки та встановлення оновлень.

1.3. Аналіз програмних засобів для захисту веб-застосунків

Нижче наведено список інструментів для перевірки безпеки веб-додатків, які можуть бути використані для зниження ризику злому сайтів. Сканування допомагає підтримувати інформаційну безпеку магазину в актуальному стані.

OpenVAS сканує вузли мережі на наявність вразливостей і дозволяє управляти вразливостями. OpenVAS - це повнофункціональний сканер вразливостей. Його можливості включають неаутентифіковане тестування, автентифіковане тестування, різні високоякісні та низькорівневі Інтернет та промислові протоколи, налаштування продуктивності для широкомасштабного сканування та потужну внутрішню мову програмування для реалізації будь-якого типу тесту на вразливість. Сканер супроводжується множиною тестів на вразливість з еволюційними даними та щоденними оновленнями. Цей потік спільноти Greenbone включає понад 80 000 тестів на вразливість.

Сканер розробляється та підтримується Greenbone Networks з 2009 року. Роботи надані спільноті під відкритим кодом під загальною публічною ліцензією GNU (GNU GPL). Greenbone розробляє OpenVAS як частину їхнього комерційного сімейства продуктів для управління вразливістю "Greenbone Security Manager" (GSM). OpenVAS - це один із елементів більшої архітектури. У поєднанні з додатковими модулями з відкритим кодом він утворює рішення управління вразливістю Greenbone. Виходячи з цього, прилади GSM використовують більш широкий спектр, що покриває потреби підприємства, GVM з додатковими функціями, управління приладами та угоду про рівень обслуговування.

BeEF (Browser Exploitation Framework) дозволяє виявити слабкі місця ПЗ, використовуючи вразливості браузера. Даний інструмент автоматизації тестування додатків використовує вектори атак на стороні клієнта для перевірки безпеки. Може викликати команди браузера, такі як перенаправлення, зміна URL-адрес, створення діалогових вікон і т.д.

Google Nogotofail – засіб тестування безпеки мережевого трафіку. Перевіряє додаток на наявність відомих вразливостей TLS / SSL і неправильних конфігурацій. Сканує SSL / TLS-шифровані з'єднання і перевіряє, чи є вони

уразливими до атак. Може бути налаштований як маршрутизатор, VPN-сервер або проксі-сервер.

Observatory by Mozilla сканує ресурс на наявність проблем безпеки. Крім своїх результатів, при виборі відповідної опції, збирає і додає до звіту аналітику зі сторонніх сервісів аналізу захищеності.

WAVES – це технологія Blackbox, яка використовується для тестування веб-додатків на уразливість від SQL-ін'єкцій. Програмний засіб визначає всі точки веб-додатків, які можуть бути використані для введення SQLIA [3].

JDBC-Checker може використовуватися для запобігання атак виду, що використовують невідповідність типів у динамічно створеному ланцюжку запитів. У запитах SQL Guard та SQL Check перевіряються під час виконання, виходячи з моделі, яка виражається як граматика, яка приймає тільки легальні запити. SQL Guard розглядає структуру запиту до і після додавання введення користувача на основі моделі.

AMNESIA поєднує в собі статичний аналіз та моніторинг виконання. У статичній фазі він створює моделі різних типів запитів, які програма може створювати на законних підставах у кожній точці доступу до БД. Запити перехоплюються, перш ніж вони надсилаються в базу даних та перевіряються на статично побудовані моделі в динамічній фазі.

WebSSARI використовує статичний аналіз, щоб перевірити потоки завад від попередніх умов для чутливих функцій. Він працює на основі дезинфікованого входу, який пройшов через визначений набір фільтрів. Обмеження підходу є адекватними передумовами для чутливої функції не можна точно виразити, тому деякі фільтри можуть бути опущені.

SecuriFly - це ще один інструмент, який був реалізований для java. Незважаючи на інший інструмент, переслідує рядок замість символу для недоступної інформації. SecurityFly намагається дезінфікувати рядки запитів, які були згенеровані за допомогою введеного вкладу, але, на жаль, ін'єкція в числових полях не може зупинитися за допомогою цього підходу. Труднощі виявлення всіх джерел вводу користувача є основним обмеженням цього

підходу. Позитивне затухання не тільки фокусується на позитивному заглушенні, а не на негативному заглушенні, але також автоматично і потребує розробника втручання. Крім того, цей підхід вигідний від оцінки синтаксису, який дає розробникам механізм регулювання використання рядкових даних, заснованого не тільки на його джерелі, але також на його синтаксичній ролі в рядку запиту.

IDS використовує систему виявлення вторгнень для виявлення SQLIA, засновану на техніці навчання комп'ютера. Технологія створює моделі типових запитів, а потім під час виконання, запити, які не відповідають моделі, будуть ідентифіковані як атаки. Цей інструмент успішно виявляє атаки, але це серйозно залежить від тренувань. Іншими словами, буде сформовано багато помилкових та ложних негативів. Іншим підходом у цій категорії є SQL-IDS, який зосереджується на написанні специфікацій для веб-додатки, які описують передбачувану структуру SQL-запитів, які виробляються додатком, а також автоматичного контролю за виконанням цих SQL-запитів щодо порушень цих специфікацій.

1.4. Висновки по розділу

В даному розділі зроблений аналіз предметної області та досліджено різновид атак на Інтернет магазини,такі як: міжсайтове кодування (XSS-атаки), SQL-ін’єкція, підробка міжсайтових запитів (CSRF), та шкідливе програмне забезпечення. По статистиці кіберзагроз в Україні у 2020 році, найпоширеніша це SQL-інєкція, вона охоплює майже 52% всіх загроз. Також був зроблений аналіз програмних засобів, які використовуються для захисту веб-застосунків.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ ІСНУЮЧИХ МЕТОДІВ ЗАХИСТУ БД

2.1. SQL-ін’єкції як основний метод порушення конфіденційності інформації в БД

SQL ін'єкція – це техніка введеня коду, яка використовується для атаки неправильно написані запити вставляють інші значення для досягнення своїх цілей.

Рисунок 2.1. ілюструє нам блок-схему реалізації SQL-ін’єкції.

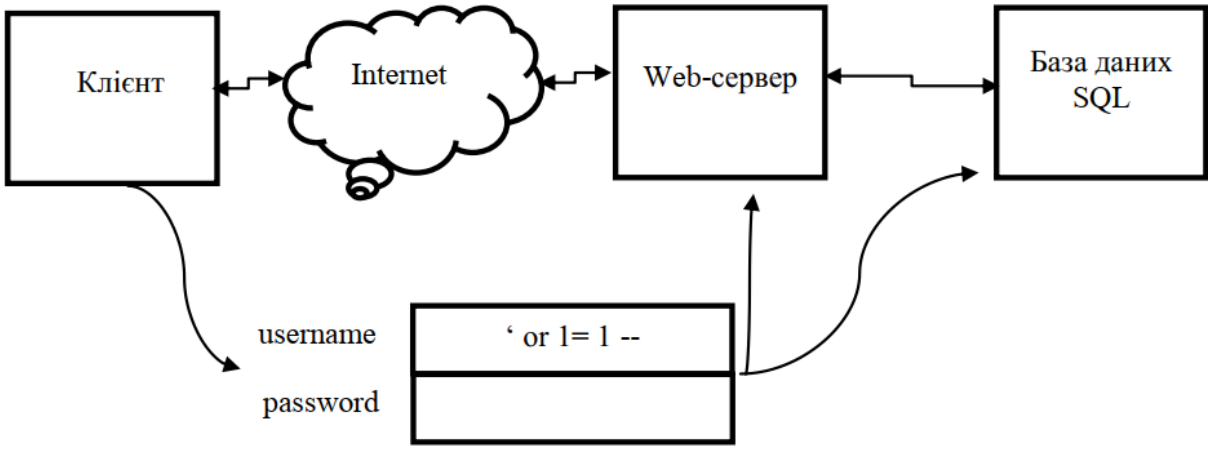

Рис. 2.1. Реалізація SQL-ін’єкції

Запити SQL - це повна програма з операторами, змінними та буквальними рядками. Ця програма створюється динамічно. На відміну від PHP-скриптів, які пишуться раз і назавжди і не змінюються на основі отриманих даних, запит SQL щоразу генерується динамічно. Тому неправильно відформатовані дані можуть перекосити або навіть змінити запит, замінивши його непередбачуваними інструкціями. Цей процес відомий як ін’єкція.

Обробка місць, зарезервованих сервером. Змінні постійно додаються до програми. Узагальнюючі символи - це загальні змінні, які вказані в запиті SQL і не змінюються на основі отриманих даних. Самі дані надходять на сервер

окремо від запиту і ніколи не проходять через нього. Дані використовуються лише на етапі виконання після виконання запиту.

На практиці це виглядає так: Коли викликається Prepare (), запит надсилається на сервер у такій формі - сервер аналізує його із заповнювачами / змінними та надсилає сигнал. Якщо все в порядку, сервер готовий отримувати дані або повідомляти про помилку. Далі дані надходять на сервер під час виконання та беруть безпосередню участь у виконанні.

Злом MS SQL починається зі збору інформації про сервер та безпосереднього пошуку сервера SQL. Для цього вам потрібно прослухати порти 1433 (прямий MS SQL) і 1434 UDP (браузер SQL). Для цього просто використовуйте Nmap із скриптом ms-sql-info. Якщо MS SQL знайдено, вам потрібно отримати доступ до сервера. MS SQL підтримує два методи автентифікації:

1. Аутентифікація Windows (Trusted Login) дозволяє працювати як користувач Windows, що перевіряється на рівні операційної системи.

2. Змішаний режим автентифікації SQL Server.

Режим автентифікації Windows є типовим, але змішаний режим набагато гнучкіший. Деякі переваги змішаного режиму:

1. Дозволяє SQL Server підтримувати старіші версії програм, а також програмне забезпечення сторонніх розробників.

2. Дозволяє підтримку інших багатоопераційних системних середовищ, не перевірених Windows.

3. Дозволяє створювати ієрархію дозволів шляхом доступу до бази даних.

Як правило, на даний момент доступ в корпоративну мережу відсутній, тобто неможливо використовувати аутентифікацію Windows. Однак є відкритий порт 1433, у цьому випадку на SQL-сервері є користувач SA (системний адміністратор), і можна зробити спробу знайти для нього пароль.

Якщо є доступ до SA, найпростіший спосіб отримати дані - отримати їх. Як правило, такий доступ не є загальним, оскільки його дуже легко зламати. Коли SA недоступний, зловмисники використовують SQL-ін’єкції в запитах

для доступу до бази даних. Наприклад, простий запит SQL, який витягує дані. Нам потрібно використовувати процедуру xp_cmdshell, але для цього потрібні права адміністратора бази даних.

Використовуючи простий SELECT, виконуємо sql-ін'єкцію. Нам потрібно створити БД та додати користувача:

```
CREATE DATABASE dbmsq;
CREATE LOGIN msadmin WITH PASSWORD = 'msadmin'
```

Наступний крок - зробити користувача sysadm,

```
USE dbmsq
ALTER LOGIN [msadmin] with default_database = [dbmsq];
CREATE USER [bob] FROM LOGIN [bob];
EXEC sp_addrolemember [db_owner], [bob];

CREATE DATABASE dbmsq;
CREATE LOGIN msadmin WITH PASSWORD = 'msadmin'
USE dbmsq
ALTER LOGIN [msadmin] with default_database = [dbmsq];
CREATE USER [bob] FROM LOGIN [bob];
EXEC sp_addrolemember [db_owner], [bob
```

Далі виключаємо можливість персоналізації та отримуємо права адміністратора:

```
ALTER DATABASE dbmsq SET TRUSTWORTHY ON
USE dbmsq
GO
CREATE PROCEDURE sp_lvlup
WITH EXECUTE AS OWNER
AS
EXEC sp_addsrvrolemember 'bob','sysadmin'
```

Процедура sp_lvlup створюється від імені OWNER, що в цьому випадку є користувачем SA. Це можливо саме тому, що БД створенна від імені db_owner,

а дана БД є довіреною, тобто TRUSTWORTHY = On. Без цієї властивості не можливо б було виконати процедуру. Активована TRUSTWORTHY – це не завжди погано. Проблеми існують тоді, коли адміністратори не моніторять привілеї та права власників БД. В результаті отримуємо доступ:

EXEC sp_configure 'show advanced options',1;

reconfigure;

‘exec sp_configure 'xp_cmdshell',1;

reconfigure sysadmin.

Отримавши доступ до потрібних нам прав – потрібно дозволити використання cmdshell:

Для найпростішого взлому, використовують доступ по RDP (Remote Desktop Protocol):

regadd "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f

exec master.dbo.xp_cmdshell 'net user root toor

У результаті, ми створили користувача в системі Windows, та відкрили доступ до сервера. Наступним кроком потрібно відкрити консоль mstsc, ввести дані, і ми отримали доступ на сервер з правами адміністратора.

2.2. Методи захисту БД від SQL-ін'єкцій

Правила, що гарантують нам захист від ін’єкцій:

1) Дані повинні використовуватися з підстановочними знаками у запитах

2) Ідентифікатори та ключові слова замінені з білого списку, попередньо зареєстрованого в нашому коді.

Практичне застосування цих правил, природно, вимагає більш детального опису. Заповнювачі використовують точні дані, всі дані слід запитувати не безпосередньо, а через якогось представника.

Більшість статей про ін’єкції не вказують на це. Однак реальність така, що окрім використання даних у запитах, ви також повинні використовувати інші речі, такі як ідентифікатори, елементи синтаксису та ключові слова. Навіть такі незначні речі, як DESC або AND, але потреба в безпеці з таких причин все ж повинна бути не менш суворою.

Давайте розберемо стандартний випадок. Ми маємо базу даних про товари, які повинні бути представлені користувачеві у вигляді таблиці HTML. Людина, яка використовує цю таблицю, може виконувати будь-які дії з даними таблиці, в результаті яких користувач може використовувати значення, а саме: атрибути та ідентифікатори. Використання значень даних безпосередньо на вимогу - це гарантоване введення. Методи форматування, до яких усі звикли тут, не допоможуть. Підготовка виразу без ідентифікатора або ключового слова не дає бажаного результату. Єдиним правильним рішенням цієї ситуації є білий список. Це загальноприйнята концепція, і майже всі досвідчені розробники легко додають її до потрібних запитів.

Для використання цього методу всі дані повинні бути записані в коді як змінні, що унеможливлює введення цього запиту.

Суть цього методу полягає в тому, що всі можливі варіанти повинні бути чітко вказані в нашому коді, і лише вони вимагаються на основі вводу користувача.

```
$order = isset($_GET['order']) ? $_GET['order'] : ''; //
$sort = isset($_GET['sort']) ? $_GET['sort'] : '';
$allowed = array("name", "price", "qty");
$key = array_search($sort,$allowed);
$orderby = $allowed[$key];
$order = ($order == 'DESC') ? 'DESC' : 'ASC';
$query = "SELECT * FROM `table` ORDER BY
```

Раніше вважалося, що плейсхолдерів для ідентифікаторів достатньо, але тепер було зроблено висновок, що ця практика має такі недоліки:

1) Якщо ім'я неправильне, запит повертає помилку.

2) Якщо ви використовуєте імена полів без перевірки, ми отримаємо інший тип ін’єкції.

Тому зараз використовуються обидва методи: спочатку отримую ідентифікатор білого списку, а потім додаю його за допомогою заповнювача - лише для того, щоб мені не потрібно було форматувати його вручну. В принципі, цієї інформації достатньо для написання цілком безпечних запитів, але в житті існують різні нюанси, що заповнювачі потрібно ретельніше аналізувати.

Перш за все, ви повинні розуміти, що існує два способи реалізації символів підстановки - серверний і клієнтський, а саме:

- У першому випадку запит та дані надсилаються окремо. Обробка виконується самою базою даних на сервері.

- У другому випадку дані створюються на стрічці запиту та замінюють зарезервовані місця безпосередньо на клієнті; створюються класичні запити SQL. Кожен із цих методів має свої недоліки та переваги. Пам'ятайте, що за замовчуванням вони працюють у другому варіанті, імітуючи роботу першого.

На цьому етапі розвитку захисту баз даних існують такі прогалини в бібліотеці:

• багатослівність

• обов'язковий

• Неможливість використання стандартних запитів SQL

• Продуктивність

Створюючи класичний запит для вибірки значень двовимірного масиву, ми повинні використовувати довгий запит, оскільки в mysql для цієї реалізації немає короткого формату. Тому для виконання цього завдання нам потрібен такий запит:

```
$data  = array();

$query = "SELECT Name, Population, Continent
FROM Country WHERE Continent=? ORDER BY
Name LIMIT 1";
$stmt->prepare($query);
$stmt->bind_param("s", $continent);
$stmt->execute();
```

Даний запит можливо скласти без використання такої кількості частин, використавши усього лиш 2 рядки:

```
$query = "SELECT Name, Population, Continent
FROM Country WHERE Continent=? ORDER BY
Name LIMIT 1"; $data  = $db-
>getAll($query,$continent);
```

Біндінг. Якщо є змінна $ _GET ['id']; і її треба прив’язати до плейсхолдеру, то це можливо зробити безпосередньо в execute () ... але лише зробивши її масивом.

Недостатність функціонала. Оператор IN (). Для створення даного запиту необхідно використати наступний запит:

```
$conts = array('Europe','Africa','Asia','North
America');

$query = "SELECT * FROM Country WHERE
Continent IN(?) ORDER BY Name LIMIT 1";
$data  = $db->getAll($query,$conts);
```

Продуктивність. Зазвичай для виразів, підготовлених на сервері, апологети наполягають на аналізі запиту лише один раз. На жаль, це не працює для веб-додатків. Результат - більше роботи там, де ви хочете заощадити. Іншим потенційним прискоренням є кешування планів запитів. І хоча ми використовуємо Prepare () для кожного запиту, база даних могла кешувати однакові дані для різних запитів. І отримайте план виконання без аналізу,

просто порівнявши рядки. За реальних навантажень готові до сервера запити втрачають швидкість звичайних запитів.

Загалом, ніхто не заважає нам покращити зручність використання існуючих бібліотек, які не витрачають час на заповнювачі. Припустимо, ви пишете обгортку для PDO, яка реалізує відсутні функції з власними заповнювачами PDO.

Як ми вже бачили раніше, класичні бібліотеки не мають достатньо корисних ігрових носіїв, а заповнювачі серверів можуть використовуватися не у всіх можливих випадках.

Створення власних заповнювачів не є складним завданням. Стандартний метод існує в PHP. Все, що вам потрібно зробити, це розрізнити різні частини запиту. Правила побудови запитів залежать від атрибутів. Тож вам потрібно визначити, який атрибут нам потрібен. Наприклад, розглянемо такий запит SQL:

Для точної заміни в базі даних нам потрібні ідентифікатори та літерали з цього запиту. У PDO це робиться за допомогою стрічки - запит передає лише стрічки, оцифровані у бажаному форматі, але це іноді може призвести до помилок.

Форматування ідентифікаторів Існує лише два правила для ідентифікаторів:

● Ідентифікатор можна викликати лише між одинарними лапками (зворотними позначками).

● Якщо такі лапки з’являються в назві, їх слід приховувати, збільшуючи вдвічі.

Чи потрібно форматувати ідентифікатори? Хіба це не потрібно в більшості випадків? Якщо запит написаний від руки, необхідність можна визначити на сайті: запит працює - не форматувати; злітає з помилкою на ідентифікаторі - необхідно форматувати. Коли ми використовуємо підстановочний знак - тобто ми динамічно додаємо ідентифікатор до запиту - потрібне форматування, оскільки ми не знаємо, яке ім’я поля замінюється у

запиті, і тому його потрібно форматувати чи ні. . . Тож ми форматуємо все. Під використанням плеєрдерів розуміється правильне форматування даних при заміні їх у запиті та допомагає нам захиститися від ін’єкцій SQL.

2.3. Сучасні методи захисту конфіденційної інформації в базах даних Microsoft SQL Server та mySQL

2.3.1. Авторизація за допомогою алгоритму TOTP

У наш час все більше підприємств використовують одноразові паролі для підвищення безпеки клієнтів та їх особистої інформації. Ці компанії в основному використовують алгоритм HOTP - який створює хеш-значення на основі певних параметрів, забезпечуючи тим самим безпеку. Але популярність алгоритму TOTP зростає - в ньому використовується алгоритм HOTP з функцією - він заснований на часі.

Авторизація за допомогою алгоритмів TOTP та HOTP є потужними сучасними методами захисту конфіденційної інформації в базах даних. Вони засновані на хеш-функції, за допомогою якої ви можете брати дані будь-якої довжини та створювати на них «відбиток пальця». Безпека алгоритмів TOTP та HOTP є засобом запобігання двом найпоширенішим атакам: атаці "людина посередині" та атаці відтворення. Їх функція полягає у використанні лічильника, який періодично змінює дані, що використовуються для генерації одноразових паролів.

2.3.2. Хеш-функція

Хеш-функція дозволяє брати дані будь-якої довжини та створювати короткий "цифровий відбиток пальця" на цих даних. Довжина значення хеш-функції не залежить від довжини вихідного тексту; якщо використовується алгоритм SHA-1, довжина цього хешу, наприклад, становить 160 біт.

Це значення завжди однакової довжини і не залежить від оригінального тексту. Хеш-функція порівнюється з виглядом кодового замка з колесами. Для всіх коліс встановлено значення "нуль", тоді ви отримуєте доступ до тексту для кожної літери коліс прокрутки відповідно до правил. Число, яке відображається

в кінці на кодовому замці, є значенням хеш-функції. Сюди входять такі інструменти, як: MD5, SHA-1.

Ідея хеш-функції полягає в тому, що вона працює лише в одному напрямку, що унеможливлює пошук документа, який повертає те саме значення за попередньо зробленим значенням хеш-функції. Це означає, що якщо ви змінили хоч одну букву, хеш теж зміниться.

2.3.3. Принципи роботи TOTP

В основі алгоритму TOTP лежить алгоритм HOTP, в якому значення лічильника підставлено величиною, залежною від часу.

Формула складається з:

T – дискретне значення часу.

X – інтервал часу, протягом якого дійсний пароль.

T – проміжок часу, необхідний для синхронізації двох сторін.

K – публічний ключ.

Current Time – поточний час.

$$T = (CurrentTime - T_0)/X$$
$$\mathrm{HOTP}(K,T) = Truncate(\mathrm{HMAC\text{-}SHA\text{-}1}(K,T))$$
$$\mathrm{TOTP} = \mathrm{HOTP}(K,T)$$

Основною відмінністю двох алгоритмів є генерація пароля на основі позначки часу, яка використовується як параметр алгоритму TOTP. Не використовується точне значення часу, а поточний інтервал, межі якого були встановлені заздалегідь (наприклад, 30 секунд).

HOTP - базується на двох важливих речах: загальній секретності та зворушливому факторі. Як частина алгоритму HmacSHA1, коефіцієнт руху створюється на основі секретного ключа. Цей алгоритм базується на подіях, тобто кожен раз, коли генерується новий OTP, коефіцієнт руху збільшується, тому згенеровані паролі повинні бути різними.

TOTP - працює подібно до HOTP, він також використовує загальний секрет і фактор руху, певний фактор створюється по-іншому. За допомогою TOTP коефіцієнт руху змінюється з часом.

Основна різниця між цими методами полягає в тому, що паролі HOTP можуть діяти необмежено довго, а TOTP доступні лише на короткий проміжок часу. З цієї причини TOTP вважається більш безпечним.

2.3.4. Безпека за допомогою HOTP

Системи безпеки, побудовані з використанням HOTP і TOTP, стійкі до поширених криптографічних атак. Авторизація за допомогою алгоритму TOTP та HOTP є ефективними сучасними методами захисту конфіденційної інформації в базах даних. Вони засновані на хеш-функції, яка дозволяє брати дані будь-якої довжини та будувати на цих даних "цифровий відбиток пальця". Безпека алгоритмів TOTP та HOTP є засобом запобігання двом найпоширенішим атакам: атаці "людина посередині" та атаці відтворення. Їх функція полягає у використанні лічильника, який періодично змінює дані, що використовуються для генерації одноразових паролів.

2.3.5. Використання Blockchain для захисту інформації

У 2008 році було вперше задокументувано blockchain, а уже в 2009 його вперше реалізували у вигляді відкритого кода, був розгорнутий як невід'ємний елемент Bitcoin, перша децентралізована цифрова валютна система для розповсюдження біткойнів за допомогою відкритого випуску ПЗ “Bitcoin peer to peer”.

Система біткойнів використовує блокчейн як розподілену книгу, яка реєструє та перевіряє всі транзакції з біткойнами у відкритій мережевій системі. Нововведенням блокчейну є його здатність уникнути подвійних витрат у транзакціях, що укладаються в повністю децентралізованій мережі без втручання довіреного зовнішнього центрального органу.

Blockchain організовує зростаючий список записів транзакцій у розширюваному блокчейні, який ієрархічно захищений криптографічними методами, щоб забезпечити міцну цілісність своїх записів транзакцій. Нові блоки можуть бути інтегровані в глобальний блокчейн лише після успішної конкуренції за децентралізованою процедурою угоди. Окрім інформації про запис транзакції, блок також містить хеш-значення всього блоку, яке можна

розглядати як криптографічне зображення, а також хеш-значення попереднього блоку, який служить криптографічним посиланням на попередній блок , служить у блокчейні. . Процес децентралізованого арбітражу забезпечується мережею, яка контролює включення нових блоків у блокчейн, протокол зчитування для надійної перевірки блокчейну та узгодженість записів транзакцій, що містяться в кожній копії блокчейну та зберігаються на кожному вузлі. Таким чином, блокчейн гарантує, що запис транзакції буде доданий до блоку і що блок, у свою чергу, буде успішно створений і заблокований у блокчейні.

Запис транзакції не можна змінити ретроспективно або скасувати. Цілісність вмісту даних у кожному блоці в ланцюжку гарантована, так що блоки, потрапивши в блокчейн, не можуть бути замінені будь-яким способом. Як результат, блокчейн служить захистом розподіленої книги, який ефективно перевіряє всі транзакції між двома сторонами відкритої мережевої системи.

У контексті біткойн-систем блокчейн використовується як безпечний, приватний та надійний публічний архів для всіх транзакцій, які обмінюють біткойни в Інтернеті. Це гарантує, що всі транзакції з біткойнами реєструються, організовуються та зберігаються в криптографічно захищених блоках, які є рядками у визначеній та стабільній формі.

Блокчейн є основним захистом для транзакцій біткойнів від багатьох поширених атак. З розвитком технології блокчейн біткойн вплинув на його застосування в інших галузях, таких як охорона здоров'я, логістика, освітня сертифікація. Екосистема блокчейну швидко зростає із збільшенням інвестицій та зацікавленості промисловості, уряду та науки у впровадженні децентралізованих систем баз даних.

Функціонально блокчейн служить розподіленою та захищеною базою даних журналів транзакцій. Коли клієнт А в мережі біткойн хоче відправити біткойни іншому клієнту В, він створює транзакцію біткойна з клієнта А. Транзакція повинна бути схвалена майнерами, перш ніж вона може бути виконана мережею біткойнів.

Майнери – це суб’єкти, які використовують ресурси власних ПК для добування певних значень хешу які не повторюються.

Для початку процесу вилучення транзакція надсилається кожному вузлу в мережі. Вузли, які є майнерами, збирають транзакції в блоці, перевіряють транзакції в блоці та відправляють блок на перевірку за допомогою MoU для отримання схвалення від мережі. Коли інші вузли перевіряють, що всі транзакції, що містяться в блоці, є дійсними, блок додається до блокчейну. Лише коли блок, що містить транзакцію, схвалений іншими вузлами і доданий до блокчейну, цей переказ біткойна з А в Б здійснюється і вважається законним.

Три основні функціональні можливості, що підтримуються реалізацією блокчейну в біткойнах:

1. Хеш-сховище.

2. Цифровий підпис.

3. Зобов’язання додати новий блок до сховища глобального ланцюга.

Завдяки правильній комбінації низки добре відомих методів безпеки, таких як хеш-ланцюг, дерево меркле, цифровий підпис, поряд з механізмами настройки, блокчейн може як запобігти подвійній емісії біткойнів, так і будь-яким змінам даних транзакцій від блокування в іншу дату.

Індикатор хешу та дерево Merkle - два основні будівельні блоки для реалізації блокчейну із зберіганням хеш-ланцюга. Тож хеш-індекс використовується для перевірки, чи дані не пошкоджені. Блок ланцюгів організований за допомогою хеш-прапорів для з'єднання блоків даних між собою. Використовуючи хеш-вказівник, який вказує на попередній блок, кожен блок вказує адресу, де зберігаються дані попереднього блоку.

Хеш збережених даних може бути публічно перевірений користувачами, щоб довести, що збережені дані не були підроблені. Якщо зловмисник намагається змінити дані в будь-якому блоці у всьому ланцюжку, щоб замаскувати підробку, супротивник повинен змінити хеш-значення всіх попередніх блоків. В результаті зловмисник припиняє втручання, оскільки він

не може замінити основну частину рядка даних, що генерується після побудови системи. Цей перший блок у ланцюжку називається блоком Genesis.

Блокчейн стійкий до несанкціонованих дій. Користувачам дозволяється повернутися до певного блоку та переглянути його з початку ланцюжка. Дерево Merkle - це бінарне дерево пошуку з вузлами, з'єднаними хеш-метриками. Ця структура даних використовується для побудови блокчейну.

Ці вузли згруповані в непересічні групи, так що, коли два вузли нижчого рівня на вищому рівні об'єднуються в один, алгоритм побудови дерева Меркла створює новий вузол даних для кожної пари вузлів нижчого рівня, що має хеш-значення кожного Вузла містить містить. Цей процес повторюється до тих пір, поки не буде досягнутий корінь дерева. Дерево Merkle має можливість запобігти фальсифікації даних, які проходять через хеш-метрики на кожному вузлі дерева.

Наприклад, якщо зловмисник намагається замінити дані на певному вузлі, хеш його батьківського вузла зміниться, і навіть якщо вони продовжать замінювати верхній вузол, їм доведеться змінити всі вузли за низхідним шляхом. вище. Це означає, що підроблені дані можна виявити, оскільки хеш-індекс кореневого вузла не відповідає збереженому хеш-індексу.

Перевага дерева Merkle полягає в тому, що воно може ефективно довести, що воно належить вузлу даних, показуючи цей вузол та всі його частини на вихідному шляху до головного вузла. Приналежність до дерева Меркле можна перевірити в логарифмічному часі, обчислюючи хеші по дорозі та перевіряючи значення хешу для коренів.

Цифровий підпис використовує криптографічний алгоритм для визначення достовірності даних. Це також хитрість, щоб переконатися, що дані не фальсифікуються.

Є три основні компоненти, що складають схему цифрового підпису. Перший компонент - це алгоритм генерації ключів, який створює два ключі, один використовується для підпису повідомлень та зберігається приватно, називається приватним ключем, а другий - загальнодоступним і називається

відкритим ключем, який використовується для перевірки того, що повідомлення має відповідний підпис та приватний ключ.

Другим основним компонентом є алгоритм підпису. Він створює підпис на вхідному повідомленні, який перевіряється цим приватним ключем.

Третім основним елементом є алгоритм перевірки. Він приймає підпис, повідомлення та відкритий ключ як вхідні дані, перевіряє підпис повідомлення за допомогою відкритого ключа і повертає логічне значення.

Правильно визначений та безпечний алгоритм підпису повинен мати дві властивості. Перше властивість є дійсним, коли підписи перевіряються. Друге властивість є дійсним, якщо підписи не змінюються. Це означає, що зловмисник із відкритим ключем не зможе замінити підписи в цих повідомленнях.

Блокчейн, що використовується в біткойнах, використовує ECDSA як схему цифрового підпису для спільних транзакцій. За допомогою ECDSA через еліптичну стандартну криву "secp256k1", для біткойн-блоку забезпечується 128-бітова безпека. ECDSA не захищає від втручання, коли обране повідомлення атакується на основі загальної групи та її стійкості до конкретної атаки. Таким чином, схема цифрового підпису, така як ECDSA, стійка до атаки вибраного повідомлення на законного суб'єкта C, метою якого є підробка фактичного підпису невидимого повідомлення M після того, як супротивник має підпис суб'єкта C, надіславши набір запити, вибрані як реакція на групу повідомлень (без повідомлення M).

Перевага використання цифрового підпису полягає в ефективній автентифікації повідомлення за допомогою PKI, щоб автор підписував повідомлення приватним ключем перед відправкою, а одержувач цього підписаного повідомлення міг використовувати ключ. повідомлення відправника для перевірки повідомлення. У більшості сценаріїв можна отримати пару ключів у довіреної третьої сторони. PKI використовується для управління відкритими ключами шляхом створення обов'язкової угоди між залученими суб'єктами.

Це зобов'язання здійснюється шляхом реєстрації та видачі сертифікатів органу з сертифікації. Процес перевірки підпису автоматично передається верифікатору підтвердження особи на основі гарантованого рівня прив'язки. Тому відкритий ключ вважається ідентичністю в цих сценаріях. Хоча блокчейн біткойнів підтримує децентралізоване управління автентифікацією, не маючи центрального органу реєстрації користувача в системі, користувачі самі генерують цю пару ключів, а користувачі можуть генерувати необмежену кількість пар ключів. клавіші. Ці ідентифікаційні дані (хеші відкритого ключа) називаються адресами в біткойнах. Оскільки не існує централізованого управління відкритими ключами, ці посвідчення фактично є псевдонімами, зібраними користувачем.

Отже, технологія блокчейн - це сучасний спосіб реалізації захисту даних. Цей метод набуває все більшого значення в галузі захисту баз даних MS SQL і застосовується лише в хмарних базах даних AZURE. На мою думку, цей метод може бути розроблений для використання в розподілених базах даних, щоб уникнути можливих порушень у надійності інформації.

2.6 Висновки по розділу

У даному розділі були розглянуті такі БД, як MS SQL та MySQL. На основі цих БД був зроблений аналіз основних методів захисту БД в контексті захисту від SQL-ін’єкцій. Ми розглянули такі методи захисту як, авторизацію за допомогою алгоритмів: TOTP та HOTP, принципи роботи цих алгоритмів. Також було розглянуто, використовування технології Blockchain, як сучасний спосіб реалізації захисту даних.

РОЗДІЛ 3.
РОЗРОБКА МЕТОДУ ЗАХИСТУ ІНФОРМАЦІЇ ВІД ВБУДОВАНИХ SQL-ЗАПИТІВ

3.1. Формалізація процесу виявлення SQL-ін’єкцій у веб-застосунках

Для спрощення розуміння домену та завдань, поставлених перед розробкою програмного забезпечення, їх потрібно моделювати, формалізувати та описувати. Для цього рекомендується методика функціонального моделювання та графічний опис процесів IDEF0 [12]. IDEF0 - методологія функціонального моделювання та графічного опису процесів, яка була розроблена для формалізації та опису процесів та систем. Особливістю IDEF0 є акцент на ієрархічному поданні об’єктів, що значно полегшує розуміння домену [13]. Діяльність програмного забезпечення представлена Рисунок 3.1 у позначенні IDEF0 та його розкладанням на різні функції за допомогою діаграм A0 та A1 у позначенні IDEF0. Ці схеми показані на Рисунках 3.1. та 3.2 [10]. Діаграма показує найбільш абстрактний опис програмного забезпечення із зазначенням предмета моделювання, цілей, яких потрібно досягти, вхідних даних, вихідних даних та механізмів - ресурсів, що виконують роботу. Діаграма A0 - це декомпозиція контекстної діаграми, вона висвітлює основні функції системи, показує потік даних між ними, вхідні та вихідні дані для різних функцій, які можуть керувати цими функціями. Діаграма A1 - це декомпозиція блоку A1 з діаграми A0 і показує дії, необхідні для виконання процесу A1 - «Виявлення ін’єкцій SQL у веб-додатках». Починаючи з найнижчого рівня декомпозиції, можна зробити висновок щодо формування необхідної архітектури програмного забезпечення та її проектування .

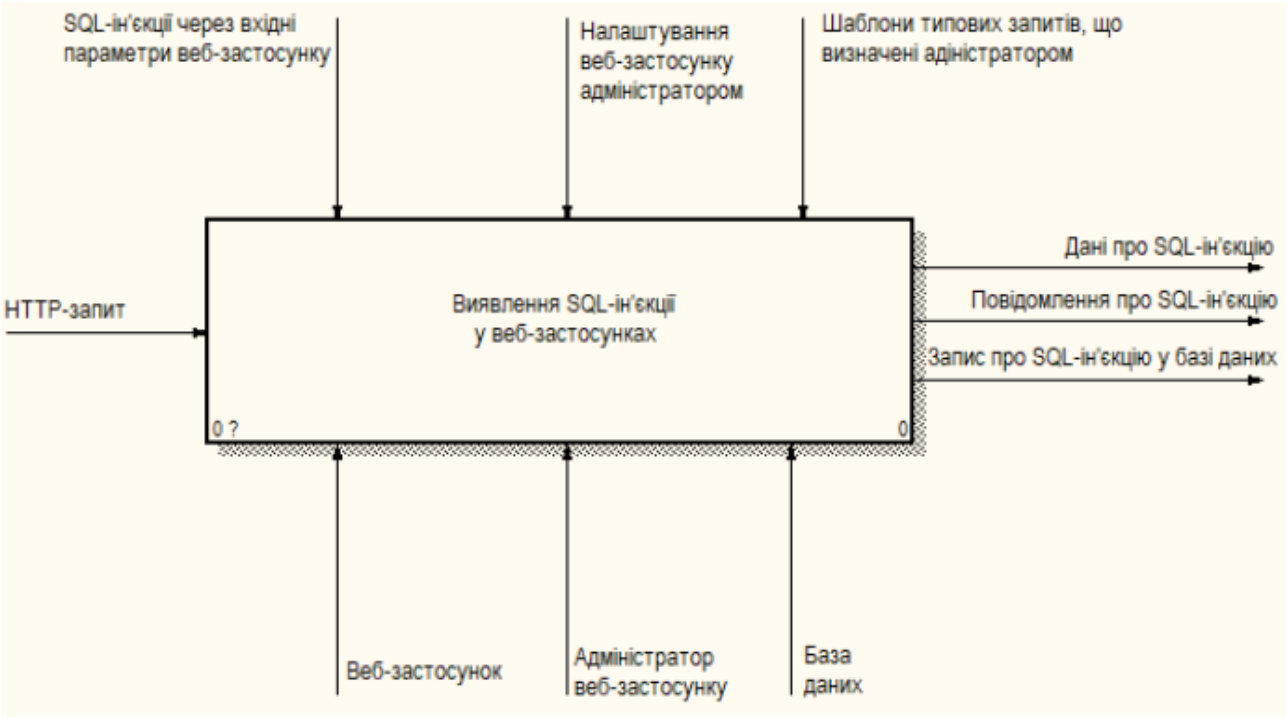

Рис. 3.1. Функціональна модель ПЗ

Відповідно до структури програмного забезпечення для виявлення ін’єкцій SQL, яка була розглянута в підрозділі 1.1 першого розділу, на рисунку 3.2 ви можете побачити найважливіші процеси, що виконуються під час роботи програмного забезпечення :

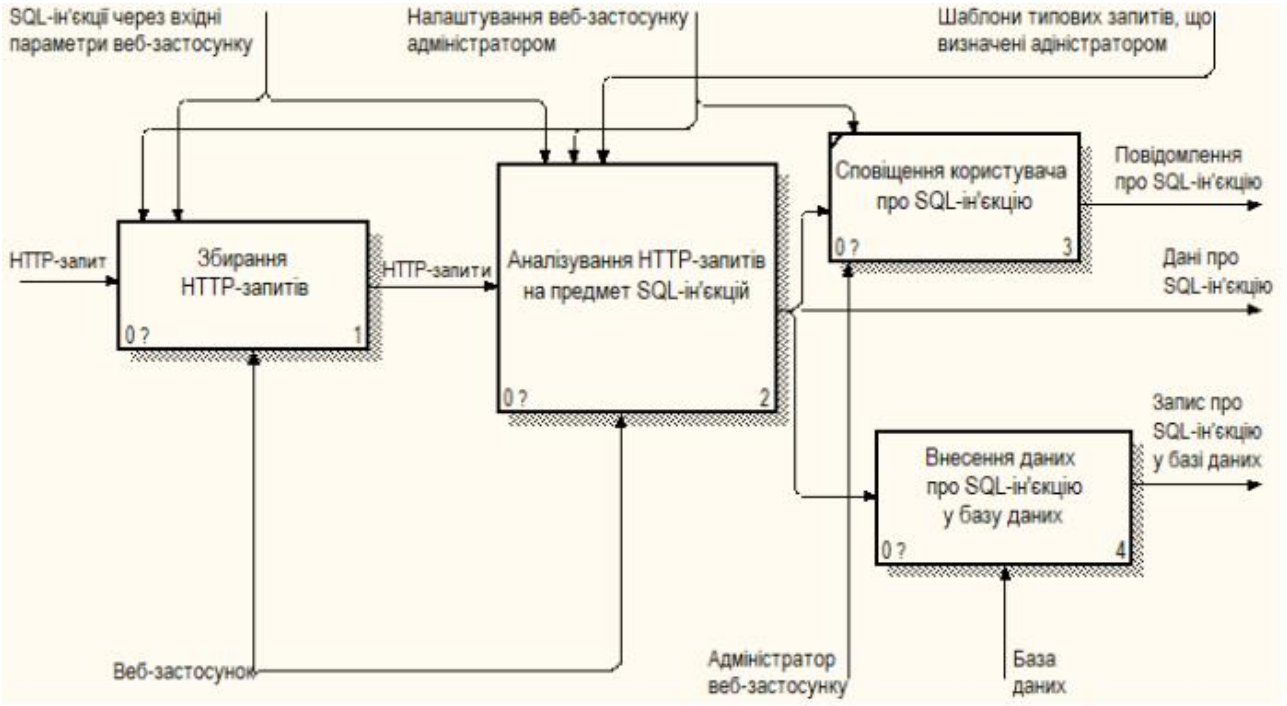

Рис. 3.2. Декомпозиція функціональної моделі ПЗ

- Збір запитів HTTP - Цей процес виконує збір первинної інформації про запити, що надсилаються до веб-програми за допомогою підсистеми збору;

- Аналіз HTTP-запитів для ін'єкцій SQL - у цьому процесі знаходить і аналізує HTTP-запити, які не відповідають стандартним шаблонам запитів, визначеним адміністратором веб-програми, використовуючи аналіз підсистеми d;

- Повідомлення про ін’єкцію SQL: Процес надсилання сповіщення про ін’єкцію SQL на поштову скриньку користувача за допомогою підсистеми доставки даних, що дозволяє адміністраторам програмного забезпечення контролювати стан веб-програми.

- Введення даних інжекції SQL в базу даних - процес, за допомогою якого інформація про інформацію про ін'єкцію SQL записується і зберігається в базі даних та в журналах файлів захищеної веб-програми.

Запис - "Запити HTTP" - сукупність запитів, надісланих до веб-програми. Ви також можете побачити механізми програмного забезпечення для виявлення ін'єкцій SQL на схемі структури бізнес-процесів:

- Програмне забезпечення для виявлення ін’єкцій SQL, яке аналізує використання довірених йому мережевих ресурсів і, у разі підозрілих або просто нетипових подій, може вжити незалежних заходів для виявлення, виявлення та усунення причини

- адміністратор веб-додатків - механізм, відповідальний за нагляд за роботою програмного забезпечення та його взаємодією з системою;

- База даних - механізм, відповідальний за зберігання та отримання даних;

Ми називаємо вихідні дані "перетвореною інформацією":

Дані введення SQL - відформатовані дані, отримані за допомогою підсистеми аналізу;

Повідомлення про ін'єкцію SQL - сформоване текстове повідомлення, надіслане підсистемою сповіщень про ін'єкції SQL із готових "Даних про ін'єкцію SQL", отриманих підсистемою аналізу;

- Запис введення SQL в базу даних - записи, створені підсистемою для зберігання інформації в базі даних.

Умови та обмеження програмного забезпечення також показані на рисунках 3.1 та 3.2 - це внутрішні правила програмного забезпечення для виявлення ін'єкцій SQL у веб-додатках.

Для більш детального опису основного процесу виявлення ін’єкцій SQL була створена діаграма в нотації IDEF, яка на рисунку 3.3, що є розкладом блоку на діаграмі A0, показує дії, необхідні для запуску процесу A1 - «Розбір HTTP-запитів на ін'єкції SQL "- буде здійснено.

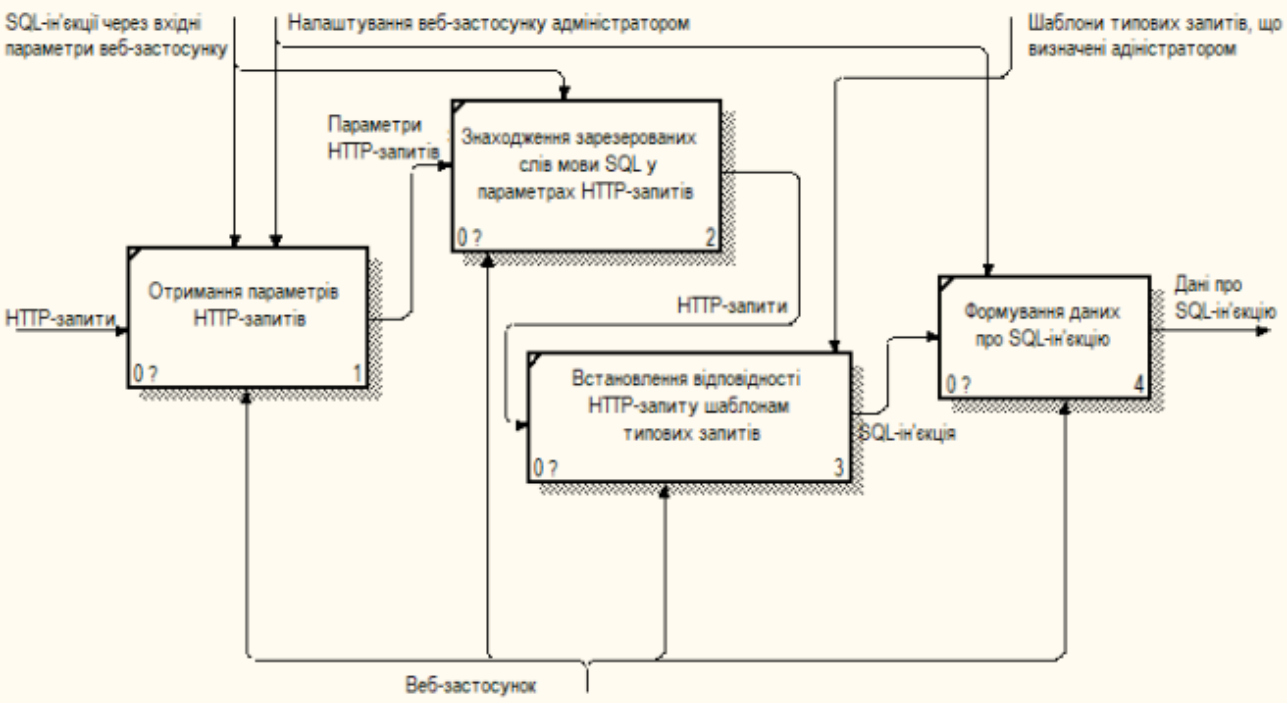

Рис. 3.3. Декомпозиція функції «Аналізування HTTP-запитів на предмет SQL-ін’єкцій»

Ця модель достатня для розуміння процесів, що лежать в основі програм45ного забезпечення для виявлення ін'єкцій SQL. Вхідними даними є "HTTP-запити". Ви також можете побачити механізми програмного забезпечення для виявлення ін'єкцій SQL на схемі структури бізнес-процесів:

- Програмне забезпечення для виявлення ін’єкцій SQL, яке аналізує використання довірених йому мережевих ресурсів і, у разі підозрілих або просто нетипових подій, може вжити незалежних заходів для виявлення, виявлення та усунення причини.

Ми називаємо вихідні дані "перетвореною інформацією":

- Дані введення SQL - відформатовані дані, отримані за допомогою підсистеми аналізу.

Також на рисунку 3.3 ви можете побачити найважливіші процеси, які виконуються під час роботи програмного забезпечення:

- Отримати параметри HTTP-запиту - Цей процес збирає вхідні параметри HTTP-запитів, які надсилаються до веб-програми через підсистему збору;

- Пошук зарезервованих слів SQL у параметрах HTTP-запитів - У цьому процесі здійснюється пошук зарезервованих слів у параметрах HTTP-запитів, і якщо такі слова знайдені, цей запит пересилається до наступного процесу та перевіряє стандартні моделі Request-request визначається веб-додатком адміністратора;

- HTTP-запит, що відповідає стандартним шаблонам запитів - процес, за допомогою якого HTTP-запит перевіряється на відповідність стандартним шаблонам запитів, визначеним адміністратором веб-програми. Якщо запит не відповідає шаблонам, тоді такий запит вважається введенням запиту SQL;

- Генерація даних ін’єкції SQL - процес, за допомогою якого генерується інформація про ін’єкцію SQL з метою передачі цих даних у процес сповіщення користувача про ін’єкцію та збереження її в базі даних.

3.2. Синтезований метод захисту інформації в БД

SQL ін'єкція є одним з найпоширеніших способів зламу сайтів, які працюють з БД. Спосіб грунтується на впровадженні в запит довільного SQL-коду.

Впровадження SQL дозволяє хакеру виконати довільний запит доБД, наприклад, прочитати зміст будь-яких таблиць, або видалити, або редагувати, додати нові дані.

Дана атака можлива тоді, коли недостатньо фільтруються вхідні дані при використанні в SQL-запитах.

Що робити, якщо в SQL-запит необхідно підставити значення рядка? Наприклад, на сайті є можливість пошуку міста по його назві. Форма пошуку передасть пошуковий запит в GET-параметр, а потім цей параметр використається в SQL-запиті:

```
$ authorName = $ _GET [ 'search'];
$ Sql = "SELECT * FROM author WHERE name LIKE ( '% $ authorName%')";
```

Проте якщо в параметрі authorName є символи лапок, то сенс запиту може кардинально змінитися. Передавши в search_text значення ') + and + (id <>' 0, виконається запит на виведення усіх міст:

SELECT * FROM author WHERE name LIKE ('%') AND (id <> '0%')).

Значення запиту змінилося, оскільки лапка в параметрі запиту використовується як контрольний символ: MySQL визначає кінець значення за допомогою наступних лапок, тому самих лапок не повинно бути.

Очевидно, що зменшення до числового типу не підходить для рядкових значень. Тож використовуйте операцію фільтра, щоб зберегти значення терміна.

Екранування додає зворотну скісну риску \ до рядка перед лапками (та іншими спеціальними символами).

Така обробка видаляє статус лапок - вони більше не визначають кінець значення і не можуть впливати на логіку оператора SQL.

Приклад:

```
fastify.route({  method: 'GET',  url: '/getAuthorByName',  handler: function (request, reply) {
    var name = request.query.authorName.replace(/(^|[^\\])"/g, '$1\\"');
    database.query('select * from fx.author where name like \'%'+ name +'%\'').then((data)=>{
        JSON.parse(reply.send(data));
    }).catch (error => reply.send(error.message));
}})
```

Запити SQL часто замінюють цілі значення, отримані від користувача. Цей ідентифікатор можна примусово зменшити до числа. Тому ми виключаємо появу небезпечних виразів. Якщо зловмисник передає цей параметр замість номера коду SQL, результат зменшення дорівнює нулю, і логіка всіх запитів SQL не змінюється.

На Рисунку 3.2 наведено приклад програмного кода, де реалізована атака з заміною цілих значень.

```
fastify.route({  method: 'GET',  url: '/getAuthor',  handler: function (request, reply) {
    var id = parseInt(request.query.id);
    database.query('select * from fx.author where Id = ' + id).then((data)=>{
        JSON.parse(reply.send(data));
    }).catch (error => reply.send(error.message));
}})
```

Рис. 3.2

Цей тип атаки введення SQL можливий, оскільки значення (дані) запиту SQL передаються разом із самим запитом. Оскільки дані не відокремлюються від коду SQL, це може вплинути на логіку всього виразу. На щастя, MySQL забезпечує спосіб передачі даних окремо від коду. Цей метод відомий як підготовлені запити.

Виконання підготовлених запитів складається з двох етапів: спочатку формується шаблон запиту - звичайне вираз SQL, але без допустимих значень, а потім окремо значення цього шаблону передаються в MySQL.

Перший етап називається підготовкою, а другий - виразом. Підготовлений запит можна виконувати кілька разів, передаючи йому різні значення.

На етапі підготовки формується SQL-запит, де замість значень стоятимуть знаки питання - наповнювачі. Ці наповнювачі пізніше будуть замінені фактичними значеннями. Шаблон запиту відправляється на сервер MySQL для аналізу і аналізу. За підготовкою слід виконання. Коли запит ініціюється, JS пов'язує фактичні значення з наповнювачами та відправляє їх на сервер. За підготовкою слід виконання. Коли запит ініціюється, JS пов'язує

фактичні значення з наповнювачами та відправляє їх на сервер, що зображено на Рисунку 3.3.

Рис. 3.3

Сервер автоматично приховує значення змінної, пов’язаної із запитом. Зв’язані змінні надсилаються на сервер окремо від запиту і не можуть впливати

```
fastify.route({  method: 'GET',  url: '/SetBirthday',  handler: function (request, reply) {
    var id = parseInt(request.query.id);
    database.query('update fx.author set Birthday=? where Id=?', [request.query.Birthday, request.query.Id]);
}})
```

на нього. Сервер використовує ці значення відразу під час виконання після обробки шаблону виразу. Прив’язані параметри не потрібно уникати, оскільки вони ніколи не вставляються безпосередньо в рядок запиту.

Нам потрібно оновити дані існуючого автора в базі даних. На Рисунку 3.4 відображається запит до бази даних, значення якого додається безпосередньо до запиту SQL:

```
fastify.route({  method: 'GET',  url: '/UpdateAuthor',  handler: function (request, reply) {
    database.query('update fx.author set Birthday = \''+ request.query.Birthday +'\', ' +
        'Name = \'' + request.query.Name + '\',' +
        'Activity =\'' + request.query.Activity + '\', ' +
        'Address =\'' + request.query.Address +'\' where Id = ' + request.query.Id);
}})
```

Рис. 3.4. Запит до бази даних

На Рисунку 3.5 наведений список авторів, який міститься в БД.

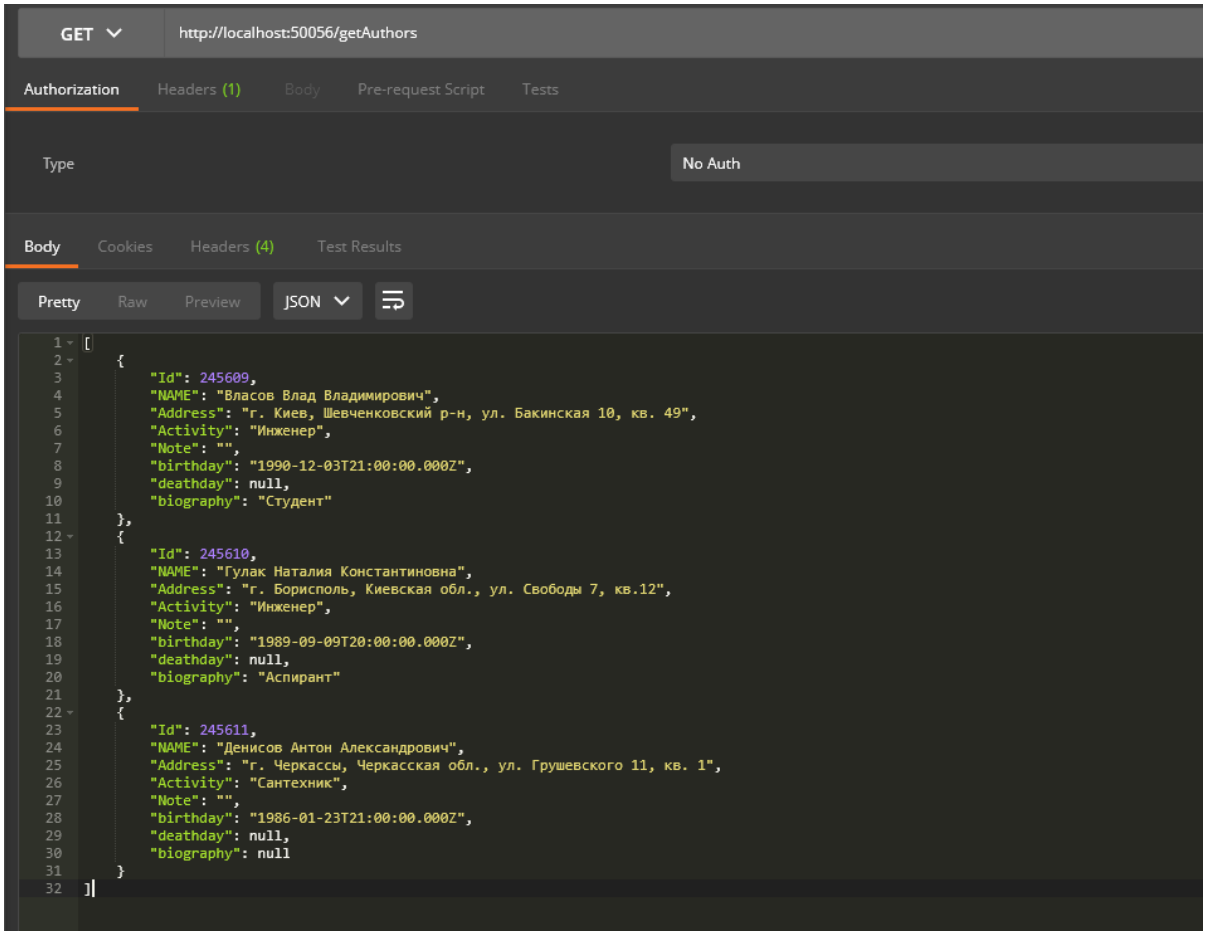

Рис. 3.5. Список авторів

Виконуємо запит на оновлення даних про автора, який має Id = 245610. Запит:

http://localhost:50056/UpdateAuthor?Id=245610&Birthday=1989-10-15

&Address=г. Киев, .Куреневский р-н, ул. Паркова 2, кв. 45

На рисунку 3.6 зображено результат оновлення:

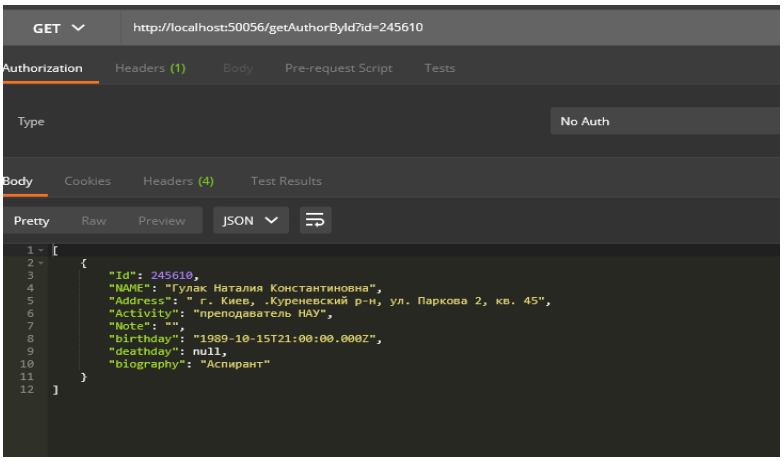

Рис. 3.6. Результат оновлення бази даних

Якщо встановити не вірні дані, то тоді буде помилка, при такому підході є вразливість до вбудованих SQL запитів.

Виконаємо один з подібних вбудованих SQL запитів, наприклад: http://localhost:50056/UpdateAuthor?Id=245610 or 1=1 &Birthday=1989-10-15& Address= г. Киев, Куреневский р-н, ул. Паркова 2, кв. 45

На рисунку 3.7 наведено результат виконання даного запиту, з якого зрозуміло, що оновилися не тільки дані автора з Id = 245610, але й всі інші записи.

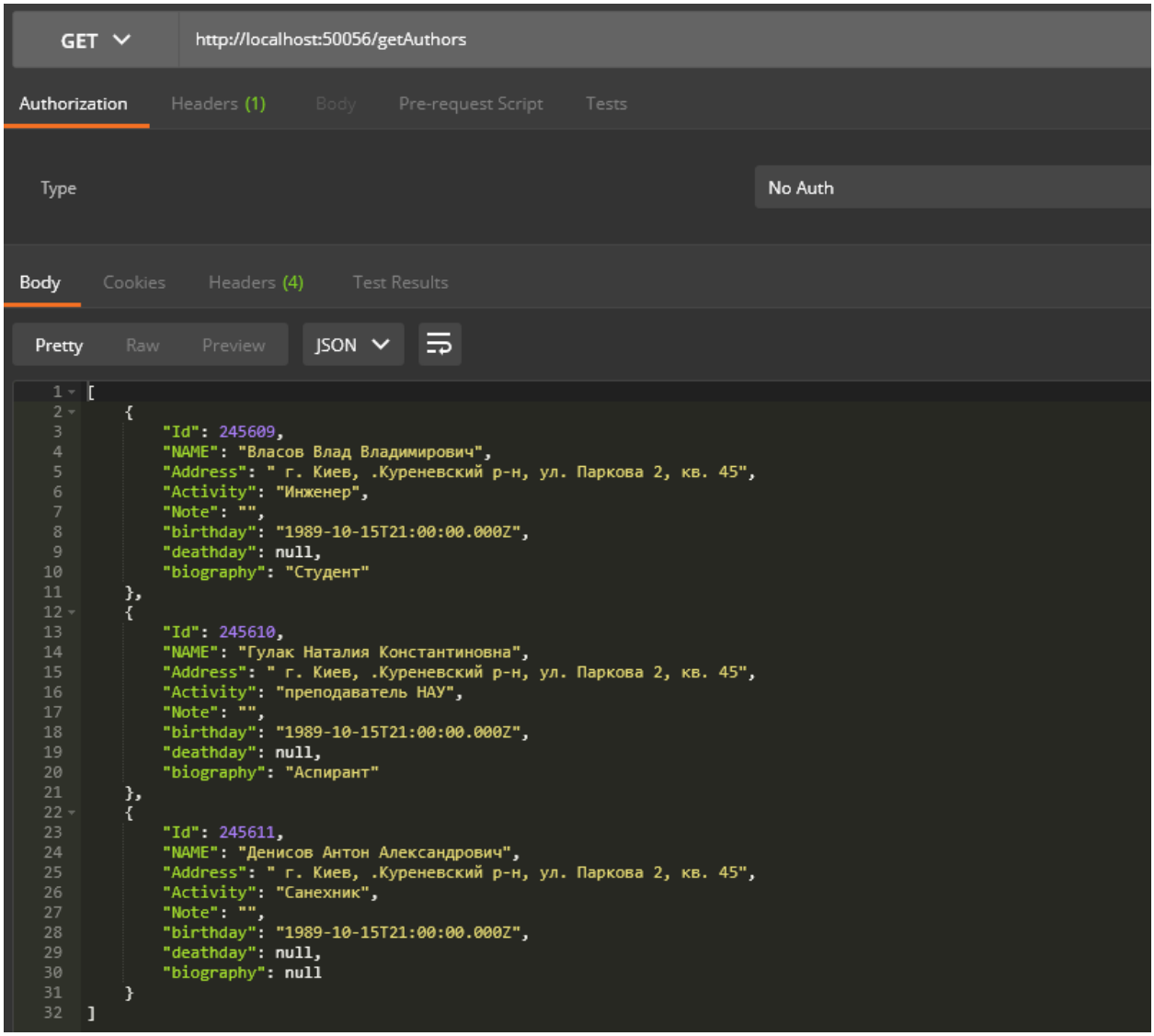

Рис.3.7. Результат виконання запиту

Проаналізовано існуючі методи захисту від вбудованих запитів SQL. Щоб повністю захистити дані від вбудованих запитів SQL, ви можете поєднати

перші два методи: екранування та зменшення цілих чисел. Тобто для створення шаблону класу, коли створюється екземпляр, конструктор якого перевіряє, фільтрує та зводить вхідні дані не лише до цілого типу, а й до бажаного типу даних, це може бути ціле число, дійсне число, текст, тип класу та інших типів. Кожна модель має власну перевірку даних. Далі напишіть метод надсилання запиту, який приймає такі параметри: ім'я команди, ім'я таблиці, в якій виконується запит, і вже перевірений екземпляр моделі, який відповідно до команди обробляє запит.

Рисунок 3.8 ілюструє новий метод для оброблення вхідних даних, а також запит до БД з уже обробленими значеннями.

```
class Author {
    constructor (aId, aBirth, aName, aActivity, aAddress) {
        this.Id = parseInt(aId);
        this.Name = aName.replace(/(^|[^\\])"/g, '$1\\"');
        this.Activity = aActivity.replace(/(^|[^\\])"/g, '$1\\"');
        this.Address = aAddress.replace(/(^|[^\\])"/g, '$1\\"');
        this.Birthday = new Date(aBirth);
    }

}


fastify.route({  method: 'GET',  url: '/UpdateAuthor',  handler: function (request, reply) {
    var author = new Author(request.query.Id,
     request.query.Birthday,
      request.query.Name,
       request.query.Activity,
        request.query.Address);

    database.query('update fx.author set Birthday = \''+ author.Birthday +'\', ' +
        'Name = \'' + author.Name + '\',' +
        'Activity =\'' + author.Activity + '\', ' +
        'Address =\'' + author.Address +'\' where Id = ' + author.Id);
}})
```

Рис. 3.8. Новий метод

Таким чином, вхідні значення обробляються відповідно, тобто вони відфільтровуються, усікаються до відповідних типів, як показано на рисунку 3.8, і правильні значення вставляються в запит. Використання методу захисту не захистить усі дані, тому вам потрібен цілісний підхід. Цей метод також дозволяє обробляти вхідні дані відповідно до ваших потреб.

Виконуємо SQL-запит:

http://localhost:50056/UpdateAuthor?Id=245610 or 1=1 &Birthday=1988-11-09& Address= г. Днипро, Днепропетровская обл., прос. Гагарина 26, кв. 18

Рисунок 3.9 ілюструє, що дані змінились у автора, із Id = 245610.

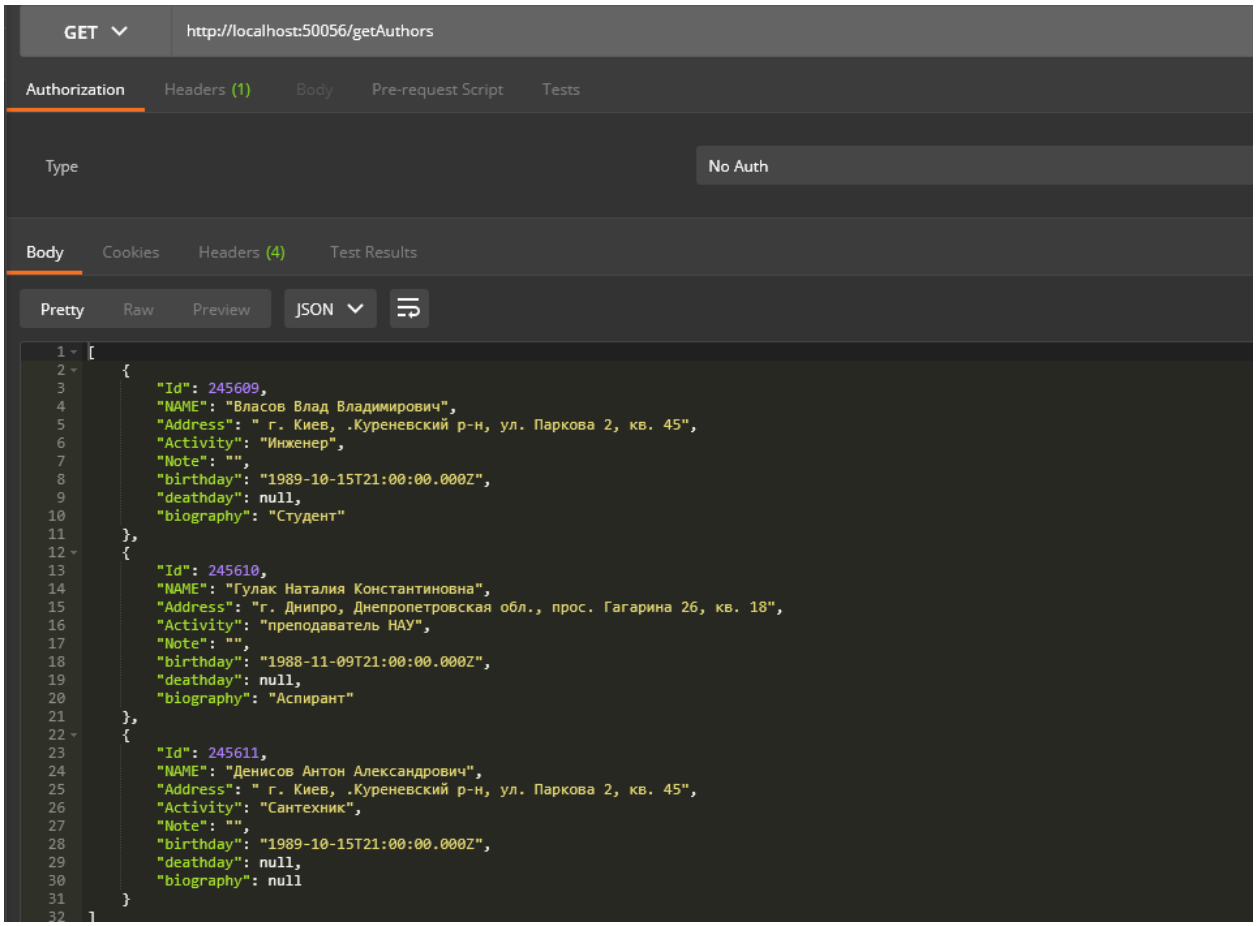

Рис. 3.9. Ілюстрація результату, що дані змінились

3.3 Рекомендації щодо захисту інформації в БД

Використовуйте ефективні технічні захисні заходи

- Централізоване управління оновленнями та виправленнями для використовуваного програмного забезпечення. Щоб правильно розставити пріоритети в планах оновлення, потрібно врахувати інформацію про поточні загрози безпеці.

● Антивірусні системи захисту з вбудованими ізольованими середовищами («пісочниця») для динамічного сканування файлів, які можуть виявляти шкідливі файли в корпоративній електронній пошті та блокувати їх, поки їх не відкриють співробітники та інші вірусні загрози. Найефективнішим є використання антивірусного програмного забезпечення, заснованого на рішеннях багатьох постачальників, які можуть виявляти приховану присутність шкідливого програмного забезпечення та виявляти та блокувати шкідливу активність у різних потоках даних - електронна пошта, мережевий та веб-трафік, сховище файлів, веб-портали. Важливо, щоб обране вами рішення дозволяло не тільки сканувати файли в режимі реального часу, але й автоматично аналізувати раніше перевірені файли для виявлення раніше виявлених загроз під час оновлення баз даних підписів.

● Рішення SIEM для своєчасного виявлення та ефективного реагування на інциденти інформаційної безпеки. Це дозволяє своєчасно виявити шкідливу діяльність, спроби зламати інфраструктуру, наявність зловмисника та негайні дії з нейтралізації загроз.

● Автоматизований аналіз безпеки та засоби виявлення вразливостей.

Брандмауери веб-додатків - як запобіжний захід для захисту веб-ресурсів.

● Поглиблені системи аналізу мережевого трафіку - для виявлення складних цілеспрямованих атак як у реальному часі, так і в збережених копіях трафіку. Завдяки цьому рішенню ви можете не тільки бачити раніше не виявлені хакерські атаки, але й відстежувати мережеві атаки в режимі реального часу, включаючи запуск зловмисного програмного забезпечення та інструментів злому, використання програмних уразливостей та атак на контролер домену. Такий підхід значно зменшить час, коли порушник крадеться в інфраструктурі, тим самим мінімізуючи ризик критичної втрати даних та порушення ділових систем та зменшуючи потенційні фінансові втрати від присутності зловмисників.

● Спеціалізовані послуги проти DDoS.

Захист даних

● Не зберігайте конфіденційну інформацію відкрито чи публічно.

● Регулярно створюйте резервні копії систем і зберігайте їх на окремих серверах, окремо від мережевих сегментів операційних систем;

● Наскільки це можливо, мінімізуйте дозволи користувачів та служб.

● використовувати різні облікові записи та паролі для доступу до різних ресурсів;

● Якщо можливо, використовуйте двофакторну автентифікацію, наприклад, для захисту привілейованих облікових записів.

Уникайте використання простих паролів

● застосовувати політику щодо паролів, яка встановлює суворі вимоги щодо мінімальної довжини та складності паролів;

● Продовження терміну корисного використання паролів (не більше 90 днів);

● Змініть паролі за замовчуванням на нові, що відповідають суворим правилам прання.

Контролюйте безпеку системи

Своєчасно оновлюйте програмне забезпечення, яке ви використовуєте, коли випускаються виправлення;

● огляд та підвищення обізнаності працівників з питань інформаційної безпеки;

● Контроль надходження небезпечних ресурсів поблизу мережі; Регулярно аналізувати ресурси, доступні для підключення через Інтернет; проаналізувати безпеку цих ресурсів та зменшити ризик уразливості використовуваного програмного забезпечення; Хорошим вибором є постійний моніторинг публікацій про нові уразливості: це дозволяє швидко виявити такі слабкі місця в ресурсах компанії та своєчасно їх усунути;

● Ефективно фільтруйте трафік, щоб мінімізувати інтерфейси мережевих служб, доступні зовнішньому зловмиснику. особливу увагу слід приділити інтерфейсам для віддаленого управління серверами та мережевими пристроями;

● Проводити регулярні тести на проникнення, щоб своєчасно виявити нові вектори атаки на внутрішню інфраструктуру та оцінити ефективність вжитих захисних заходів;

● Регулярно аналізувати безпеку веб-додатків, включаючи аналіз вихідного коду, для виявлення та виправлення вразливих місць, які можуть дозволити атаки, включаючи клієнтів додатків;

● Відстежуйте запити ресурсів на секунду, налаштовуйте сервери та мережеві пристрої для нейтралізації загальних сценаріїв атак (таких як повені TCP та UDP або декілька запитів до бази даних).

Зверніть увагу на безпеку клієнтів

● підвищити обізнаність споживачів про проблеми ІБ;

● регулярно нагадувати клієнтам про правила безпечної роботи в Інтернеті, пояснювати методи атаки та способи захисту;

● Попередити клієнтів не вводити облікові дані в підозрілих веб-ресурсах і, тим більше, не розголошувати цю інформацію третім особам електронною поштою або під час телефонного дзвінка;

● Поясніть клієнтам, що робити, якщо є підозра на шахрайство;

● Повідомляти клієнтів про випадки інформаційної безпеки.

Як постачальники захищають свою продукцію

● застосовувати всі рекомендовані заходи безпеки для забезпечення безпеки організації;

● Впроваджувати процеси безпеки протягом циклу розробки програмного забезпечення;

● Проведення регулярних перевірок безпеки програмного забезпечення та веб-додатків, включаючи перевірку вихідного коду;

● використовувати найновіші версії веб-серверів та баз даних;

● Уникайте використання бібліотек та фреймворків, які мають відомі вразливості.

Як захистити пересічного користувача

Не економте на безпеці

● використовувати лише ліцензійне програмне забезпечення;

● Використовуйте ефективний захист від вірусів на всіх пристроях.

● Своєчасно оновлюйте програмне забезпечення, коли з’являються виправлення.

Захистіть свої дані

● Зберігайте найважливіші файли не лише на жорсткому диску комп’ютера, а й на знімних носіях, зовнішніх жорстких дисках або хмарному сховищі.

● використовувати обліковий запис без прав адміністратора для щоденної роботи в операційній системі;

● Якщо можливо, використовуйте двофакторну автентифікацію для захисту електронних листів, наприклад.

Не використовуйте прості паролі

● Використовуйте складні паролі, що складаються з невеликих комбінацій літер, цифр та символів довжиною щонайменше 8 символів. За допомогою менеджера паролів (захищене сховище з новими функціями для генерації паролів) ви можете створювати та зберігати паролі;

● Не використовуйте один і той же пароль для різних систем (для веб-сайтів, електронної пошти тощо);

● Змінюйте всі паролі принаймні раз на півроку, бажано кожні два-три місяці.

Будьте пильними

● Перевірте всі вкладення, отримані електронною поштою, за допомогою антивірусного програмного забезпечення;

● Будьте обережні з веб-сайтами з фальшивими сертифікатами та пам’ятайте, що введені на них дані можуть перехоплювати зловмисники;

● Будьте гранично обережні при введенні даних для входу на веб-сайтах та при роботі з онлайн-платежами;

● Не переходьте за посиланнями на невідомі підозрілі ресурси, особливо коли браузер попереджає про небезпеку.

● Не переходьте за посиланнями зі спливаючих вікон, навіть якщо ви знайомі з компанією або продуктом, який рекламується.

3.4 Висновки по розділу

Після аналізу таких методів захисту даних, як аналіз HTTP-запитів, авторизації за допомогою алгоритму TOTP та HOTP, метода екранування, метода приведення до цілочисельного типу, які захищають базу даних від вбудованих запитів SQL було розроблено новий метод захисту в основі якого лежать методи екранування та приведення до цілочисельного типу, який може повністю захистити вхідні дані та зберегти порядок, конфіденційність та цілісність даних у базі даних. Даний розділ також містить професійний список рекомендації щодо захисту інформації в базі даних.

ВИСНОВКИ

В дипломній роботі на тему: «Методи захисту Web-сторінок Інтернет-магазину» було розглянуто та проаналізовано стандартні засоби захисту веб-сторінок Інтернет-магазину, типи атак на веб-сторінки. На основі результатів дослідження та статистичних даних було визначено найпоширеніші атаки на веб-ресурси.

Було досліджено методи захисту веб-сторінок інтернет магазину від мережевих атак. Було розглянуто переваги та недоліки стандартних методів. На основі результатів дослідження було обрано два методи захисту веб-сторінок, а саме методи екранування та приведення до цілочисельного типу.

За результатами обраних методів було розроблено та протестовано новий комбінований метод захисту інформації Web-сторінок на основі методів екранування та приведення до цілочисельного типу, який захищає вхідні дані та зберігає порядок та цілісність даних у базі даних, а це в свою чергу надало можливість удосконалити рівень захищеності інформації на Web-сторінках взагалом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Килюшева, Е. Атаки на Веб-сайты в 2016 году: боты и простые уязвимости / Е. Килюшева, Е. Гнедин // Кибербезопасность 2016- 2017: от итогов к прогнозам. – 2017. – С. 38–42.

2. Mishra, S. SQL Injection Detection Using Machine Learning: Master's Theses and Graduate Research. – USA, 2019. – 51p.

3. Literature survey on detection of web attacks using machine learning / A. Gupta [et al.] // International Journal of Scientific Research Engineering & Information Technology. – 2018. – Vol. 3. – P. 1845–1853.

4. Xiong J., Zolotov V. Fast path traversal n a relational database-based graph structure. – 12/20/2018. – US Patent App. 16/038,498.

5. Cross-site Scripting. – Режим доступа: https://www.styler.ru/styler/xss/.

6. Babiker, M. Web application attack detection and forensics: A survey / Babiker Mohammed, Karaarslan Enis, Hoscan Yasar // 6th International Symposium on Digital Forensic and Security (ISDFS). – IEEE. 2018. – P. 1–6.

7. An improved payload-based anomaly detector for web applications / Jin Xiaohui [et al.] // Journal of Network and Computer Applications. – 2018. – Vol. 106. – P. 111–116.

8. Ross, K. SQL Injection Detection Using Machine Learning Techniques and Multiple Data Sources: Master's Theses and Graduate Research. – USA, 2018. – 27p.

9. Silva, N. Network Intrusion Detection Systems Design: A Machine Learning Approach / N. Silva, D. G. Gomes // Anais do XXXVII Simposio Brasileiro de Redes de Computadores e Sistemas Distribuidos. – SBC. 2019. – P. 932–945.

10. Veni R, H. Identifying Malicious Web Links and Their Attack Types in Social Networks/ H. Veni R, H. Reddy A, C. Kesavulu // International Journal of Scientific Research in Computer Science, Engineering and Information Technology. – 2018.– P.1060–1066.

11. Fouladi, R. F. Frequency based DDoS attack detection approach using naive Bayes classification / R. F. Fouladi, C. E. Kayatas, E. Anarim // 2016 39th

International Conference n Telecommunications and Signal Processing (TSP). – IEEE. 2016. – P. 104–107.

12. Atienza, D. Neural analysis of http traffic for web attack detection / D. Atienza, A. Herrero, E. Corchado // Computational Intelligence n Security for Information Systems Conference. – Springer. 2015. – P. 201–212.

13. Goyal, B. A Competent Approach for Type of Phishing Attack Detection Using Multi-Layer Neural Network / B. Goyal, M. Bansal // International Journal of Advanced Engineering Research and Science. – 2017. – Vol. 4, no. 1. – P. 210–215.

14. Bouzida, Y. Neural networks vs. Decision trees for intrusion detection / Y. Bouzida, F. Cuppens // IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM). Vol. 28. – 2006. – P. 29–37.

15. Su MingYang. Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers/ MingYang Su // Expert Systems with Applications. – 2011. – Vol. 38. – №. 4. – P. 3492–3498.

16. A novel hierarchical intrusion detection system based on decision tree and rules-based models / A. Ahmim [et al.] // arXiv preprint arXiv:1812.09059. – 2018. – 6p.

17. Anomaly-based web application firewall using HTTP-specific features and one-class SVM / Epp Nico [et al.] // Workshop Regional de Seguran¸ca da Informa¸c~ao e de Sistemas Computacionais. – 2017. – 11p.

18. Tian, Z. A Distributed Deep Learning System for Web Attack Detection on Edge Devices / Z. Tian [et al.] // IEEE Transactions on Industrial Informatics. – 2019. – P.99–107.

19. Ye Jin. A DdoS attack detection method based on SVM in software defined network / Ye Jin [et al.] // Security and Communication Networks. – 2018. – Vol. 2018.– P. 1–8.