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

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра безпеки інформації та телекомунікацій

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню бакалавра

студента *Бородкіна Владислава Григорійовича*
академічної групи *125-20-1*
спеціальності *125 Кібербезпека*
спеціалізації[1]
за освітньо-професійною програмою *Кібербезпека*
на тему *Захист вебресурсів від SQL-загроз за допомогою програмного додатку на Python*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Кагадій Т.С.			
розділів:				
спеціальний	ст. викл Начовний І.І.			
економічний	к.е.н., доц. Пілова Д.П.	85	добре	
Рецензент				
Нормоконтролер	ст. викл. Мєшков В.І.			

Дніпро
2024

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

«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавра

студенту ***Бородкіну Владиславу Григорійовичу*** **академічної групи** ***125-20-1***
(прізвище ім'я по-батькові) (шифр)

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

на тему ***Захист вебресурсів від SQL-загроз за допомогою програмного додатку на Python***

затверджену наказом ректора НТУ «Дніпровська політехніка» від 23.05.2024 № 469-с

Розділ	Зміст	Термін виконання
Розділ 1	Аналіз сучасних проблем кібербезпеки та постановка задачі щодо захисту баз даних від SQL-атак.	15.03.2024
Розділ 2	Дослідження та вибір оптимальних мов програмування, баз даних і фреймворків для розробки захищеного додатку, розробка та тестування цього додатку.	10.05.2024
Розділ 3	Розрахунок витрат, оцінка можливого збитку від атаки та аналіз економічної ефективності системи.	11.06.2024

Завдання видано _____ Тетяна КАГАДІЙ
(підпис керівника) (ім'я, прізвище)

Дата видачі: 01.04.2024р.
Дата подання до екзаменаційної комісії: 28.06.2024р.
Прийнято до виконання _____ Владислав БОРОДКІН
(підпис студента) (ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 88 с., 23 рис., 1 табл., 4 додатка, 20 джерел.

Об'єкт досліджень: вебресурси та їх вразливість до SQL-загроз.

Предмет розробки: методи та засоби захисту вебресурсів від SQL-ін'єкцій.

Мета роботи: розробка та впровадження ефективних методів захисту вебресурсів від SQL-загроз, які зможуть запобігти несанкціонованому доступу до баз даних, забезпечити цілісність та конфіденційність інформації.

В першому розділі розглянуто стан питання, постановка задачі кваліфікаційної роботи, проведено аналіз нормативно-правової бази у сфері кібербезпеки. Розглянуто основні проблеми кібербезпеки, класифікація SQL-атак, описано сучасні тенденції у сфері кібербезпеки та основні загрози для інформації для вебресурсів.

В спеціальній частині описана загальна характеристика мов програмування, баз даних, та фреймворків, досліджені основні вразливості для вебресурсів. Розроблено програмне рішення для захисту від SQL-загроз та проведено його тестування.

В економічному розділі розраховані основні показники, які показують економічну доцільність та ефективність впровадження розроблених методів захисту інформації. Проведено розрахунок витрат на розробку політики безпеки інформації, оцінка величини можливого збитку від атаки, та визначено загальний ефект від впровадження системи інформаційної безпеки.

Практична цінність розробки полягає забезпеченні ефективного захисту вебресурсів від SQL-ін'єкцій, що запобігає несанкціонованому доступу до баз даних та забезпечує цілісність і конфіденційність інформації.

БЕЗПЕКА ІНФОРМАЦІЇ, КІБЕРБЕЗПЕКА, SQL-ІН'ЄКЦІЇ, ЗАХИСТ ВЕБРЕСУРСІВ, ЕКОНОМІЧНА ДОЦІЛЬНІСТЬ.

ABSTRACT

Explanatory note: 88 pp., 23 pic, 1 table, 4 app, 20 sources,

Object of research: web resources and their vulnerability to SQL threats.

Development subject: methods and tools for protecting web resources from SQL injection attacks.

The purpose of the project: development and implementation of effective methods for protecting web resources from SQL threats to prevent unauthorized access to databases, ensuring the integrity and confidentiality of information.

In the first section, the nutritional system, the formulation of the problem of qualified work, and an analysis of the regulatory framework in the field of cybersecurity were analyzed. The main problems of cybersecurity, the classification of SQL attacks are reviewed, current trends in the field of cybersecurity and the main threats to information for web resources are described.The special part describes the general characteristics of programming languages, databases, and frameworks, and examines the main vulnerabilities for web resources. A software solution for protection against SQL threats was developed and tested.

The economic section calculates the main indicators showing the economic feasibility and effectiveness of implementing the developed information security methods. It includes the calculation of costs for the development of the security policy, the assessment of the potential damage from an attack, and the overall effect of implementing the information security system.

The practical value of the development is to provide effective protection of web resources against SQL injections, which prevents unauthorized access to databases and ensures the integrity and confidentiality of information.

INFORMATION SECURITY, CYBERSECURITY, SQL INJECTIONS, WEB RESOURCE PROTECTION, ECONOMIC FEASIBILITY.

СПИСОК УМОВНИХ СКОРОЧЕНЬ

API - Application Programming Interface;

CPU - Central Processing Unit;

CSS - Cascading Style Sheets;

DOS - Denial-Of-Service Attack

DNS - Domain Name System;

DHCP - Dynamic Host Configuration Protocol;

FTP - File Transfer Protocol;

GUI - Graphical User Interface;

GPU - Graphics Processing Unit;

HTML - Hyper Text Markup Language;

HTTP - Hyper Text Transfer Protocol;

IDE - Integrated Development Environment;

LAN - Local Area Network;

RAM - Random Access Memory;

SSL - Secure Sockets Layer;

SDK - Software Development Kit;

URL - Uniform Resource Locator;

VPN - Virtual Private Network;

WAN - Wide Area Network;

XML - Extensible Markup Language;

СУБД - система управлінням базами даних.

ЗМІСТ

ВСТУП

Тема кваліфікаційної роботи є актуальною в сучасному світі, все більше компаній впроваджують інформаційні технології в свій бізнес, в своє повсякденне життя, де інформаційні технології відіграють вирішальну роль у різних сферах життя.

Вебресурси стали невід'ємною частиною нашої діяльності, забезпечуючи зручний доступ до інформації та послуг. Але зростання кількості вебресурсів та їх популярність також призвели до збільшення кількості кібератак, зокрема SQL-ін'єкцій, які є одними з найпоширеніших та найнебезпечніших типів загроз. SQL-ін'єкції дозволяють злодіям отримати доступ інформації, редагувати або навіть повністю знищити важливі та не дуже данні. Такі атаки можуть призвести до фінансових втрат, порушення репутації або правових наслідків. Тому забезпечення захисту інформації є ключовим аспектом забезпечення безпеки вебресурсів.

Метою розробки є впровадження методів захисту інформації від SQL-загроз, які будуть запобігти несанкціонованому доступу до баз даних, забезпечуючи цілісність та конфіденційність інформації.

Відповідно до мети було поставлено наступні завдання:

- дослідити проблематику кібербезпеки;
- розглянути проблему SQL-загроз та їхню класифікацію;
- проаналізувати мови програмування;
- проаналізувати бібліотеки;
- проаналізувати бази даних;
- розробити програмне рішення;
- протестувати розроблене програмне рішення.

РОЗДІЛ 1. СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Проблеми кібербезпеки

Сьогодні людина може отримувати та надсилати будь-яку інформацію: відео, електронну пошту лише натисканням кнопки, але чи замислювався він/вона коли-небудь про те, наскільки безпечно ця інформація передається іншій особі без розтоку даних? Правильна відповідь полягає в кібербезпеці. Сьогодні понад 61% галузевих обмінів здійснюється в Інтернеті, тому ця сфера є необхідною умовою високої якості безпеки для прямих і найкращих обмінів. Таким чином, кібербезпека стала актуальною проблемою. Ступінь кібербезпеки не обмежується лише перевіркою даних в IT-індустрії, але й різними сферами, такими як кіберпростір і так далі. Покращення кібербезпеки та забезпечення того, що необхідні системи даних є життєво важливими для безпеки та фінансового процвітання кожної країни.

Створення безпечнішого Інтернету (і захист Інтернет-клієнтів) стало важливим для вдосконалення нового управління як законодавча стратегія. Боротьба з кіберзлочинністю потребує широкомасштабної та більш безпечної практики. Окремі оцінки самі по собі не можуть утримати жодного злочину; вкрай важливо, щоб юридичні органи мали змогу ефективно розслідувати кіберзлочини та висувати звинувачення. Нині численні країни та адміністрації вводять суворі правила кібербезпеки, щоб уникнути втрати деяких життєво важливих даних. Кожна з них має бути оснащена цією кібербезпекою та врятувати себе від цих зростаючих кіберзлочинів [1].

Кібербезпека стосується як незахищеності, створеної цим новим простором, так і практикою чи процедурами, які роблять його (поступово) безпечним. Це натякає на безліч навчань і заходів, як спеціалізованих, так і неспеціалізованих, спрямованих на забезпечення біоелектричного стану та інформації, яку він містить і транспортує від усіх можливих загроз. Це дослідження має на меті зібрати всю інформацію та огляд, пов'язаний з

кіберзлочинністю, надати історичні факти та підготувати звіти про проаналізовані дані про різні атаки, про які повідомлялося всюди за останні п'ять років. На основі проаналізованої інформації потрібно надати всі контрзаходи, які організації можуть вжити для забезпечення покращеної безпеки, яка б підтримувала захист організацій від атак хакерів і забезпечувала кібербезпеку, щоб уникнути всіх ризиків.

Інтернет – це інфраструктура, що найшвидше розвивається сьогодні. У сучасному технічному середовищі багато нових технологій змінюють людство. Але через нові технології ми не можемо ефективно захистити нашу особисту інформацію, тому кількість кіберзлочинів різко зростає щодня. Більшість транзакцій як комерційних, так і особистих здійснюються за допомогою засобів онлайн-транзакцій, тому важливо мати досвід, який потребує високої якості безпеки, підтримуючи кращу прозорість для всіх і безпечніші транзакції. Отже, кібербезпека - це остання проблема. Передові технології, такі як хмарні послуги, мобільні телефони, електронна комерція, інтернет-банкінг та багато іншого, вимагають високих стандартів і безпечнішого процесу безпеки. Усі інструменти та технології, задіяні для цих транзакцій, зберігають найбільш конфіденційну та важливу інформацію користувача. Тому забезпечення необхідної безпеки для них є дуже важливим. Поліпшення кібербезпеки та захист конфіденційних даних та інфраструктури є важливими для безпеки кожної країни.

Тренди кібербезпеки

Кібербезпека відіграє вирішальну роль у сфері технологій даних. Захист даних став найбільшою проблемою в наш час. Кібербезпека. Основна річ, яка займає рейди, - це кіберзлочини, які крок за кроком надзвичайно зростають. Різні адміністрації та організації вживають багатьох заходів для запобігання цим кіберзлочинам. Крім того, різні заходи кібербезпеки все ще викликають величезне занепокоєння багатьох. Одні з основних тенденцій, які змінюють кібербезпеку:

Ризик нападу на вебпрограми з метою розділення інформації чи поширення шкідливого коду залишається. Кіберзлочинці передають свій код за допомогою хороших вебсерверів, якими вони обмінювалися. У будь-якому випадку інформаційні атаки, значна частина яких потрапляє на обговорення ЗМІ, також становлять значний ризик. Наразі люди потребують більш незвичайного акценту на безпеці вебсерверів а також вебдодатків. Вебсервери є головним місцем для цих кіберзлочинців, щоб отримати інформацію. Таким чином, слід надійно використовувати додаткову безпечну програму, головним чином серед життєво важливих обмінів разом, щоб не стати кар'єром для цих забруднень.

Мобільні мережі

Ризик нападу на вебпрограми з метою розділення інформації чи поширення шкідливого коду залишається. Кіберзлочинці передають свій код за допомогою хороших вебсерверів, якими вони обмінювалися. У будь-якому випадку інформаційні атаки, значна частина яких потрапляє на обговорення ЗМІ, також становлять значний ризик. Наразі люди потребують більш незвичайного акценту на захисті вебсерверів, а також вебдодатків. Вебсервери є головним місцем для цих кіберзлочинців, щоб отримати інформацію. Таким чином, слід надійно використовувати додаткову безпечну програму, головним чином серед життєво важливих обмінів разом, щоб не стати кар'єром для цих забруднень.

Шифрування

Це метод кодування повідомлень, тому програмісти не можуть його перевірити. У шифруванні повідомлення кодується за допомогою шифрування, перетворюючи його на змішаний фігурний вміст. Зазвичай він завершується використанням «ключа шифрування», який демонструє, як має бути закодовано повідомлення. Шифрування на рівні першої створеної контрольної точки забезпечує захист інформації та її надійність. Додаткове використання шифрування створює більше проблем у кібербезпеці. Шифрування використовується для забезпечення інформації під час подорожей, наприклад, для

обміну інформацією за допомогою систем (наприклад, Інтернет, онлайн-бізнес), мобільних телефонів тощо.

ADP і цілеспрямовані атаки

Advanced Persistent Threat (APT) - це цілий вимір кіберзлочинного програмного забезпечення. Наприклад, IPS або вебфільтрація мали ключовий вплив на розрізнення таких цілеспрямованих нападів. Оскільки зловмисники стають сміливішими та використовують все більш сумнівні методи, безпека мережі повинна поєднуватися з іншими перевагами безпеки для виявлення нападів. Таким чином, необхідно відновити наші процедури безпеки, щоб протистояти більшій кількості небезпек, які з'являться пізніше. Згодом наведене вище є частиною візерунків зміна сутності кібербезпеки на планеті. Основні мережеві загрози показано на рисунку 1.1

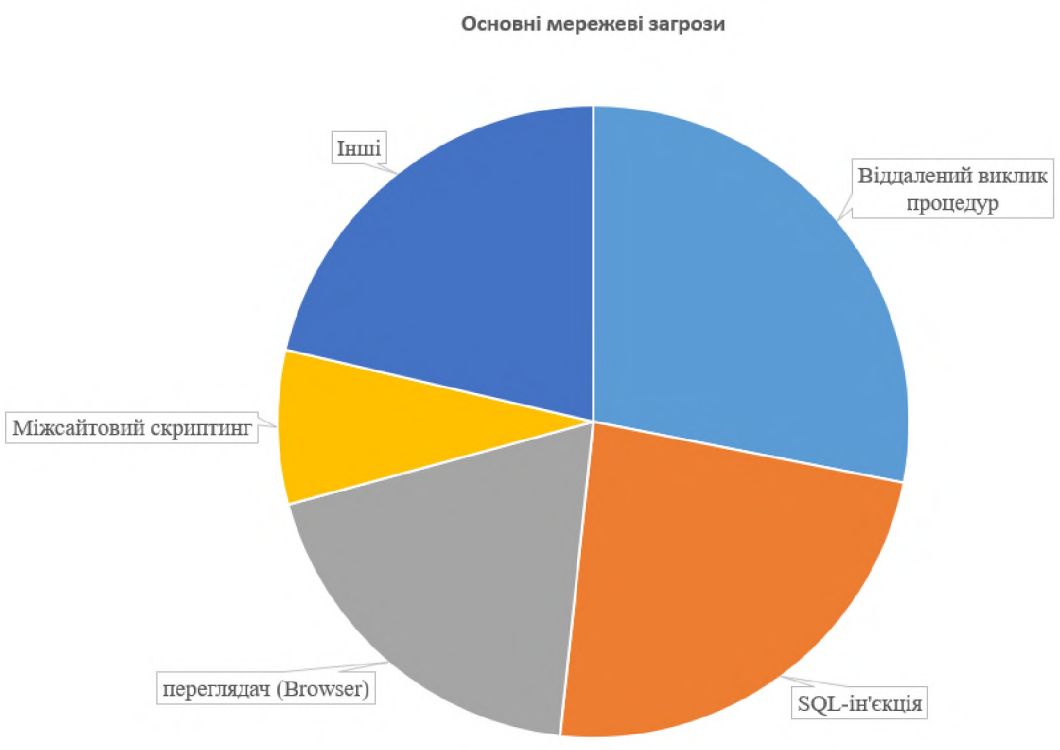

Рисунок 1.1 - Загрози кібербезпеці

Роль соціальних мереж у кібербезпеці

Соціальні мережі перетворюються на стиль життя для деяких людей. Зазвичай їх використовують для підтримки зв'язку, планувати події, ділитися своїми фотографіями та коментувати останні новини. Вони замінили електронну пошту та телефон, вимагаючи від нас багато часу. Однак, як і будь-що інше в Інтернеті, необхідно знати про небезпеки. Комп'ютери, мобільні телефони та різні гаджети є безцінними активами, які дають людям будь-якого віку надзвичайну здатність підключатися та співпрацювати з будь-яким куточком світу. Люди можуть робити це різними способами, включаючи використання соціальних медіа або мережевих сайтів.

Завдяки соціальним мережам люди можуть ділитися роздумами, фотографіями, вправами або будь-якою частиною свого життя. Вони можуть привнести невідомий погляд на життя інших, незалежно від того, чи живуть вони поблизу чи на іншій земній кулі. На жаль, ці мережі представляють небезпеку пристрою. Оскільки соціальні медіа майже надійно використовуються більшістю з них, це стало чудовою сценою для кіберзлочинців для злому особистих даних і отримання значної інформації [4].

Організації повинні переконатися, що вони так само швидко розпізнають небезпеки, посилено реагують і утримуються від будь-якого розриву. Згодом люди повинні вжити відповідних заходів, зокрема щодо керування соціальними мережами, щоб уникнути втрати своїх даних. Здатність людей передавати дані мільйонній групі осіб є основою точного тесту, який соціальні мережі пропонують організаціям. Тим не менш, дозволяючи будь-кому розповсюджувати фінансово делікатні дані, соціальні медіа додатково дають можливість виявляти неправдиві дані. Швидке поширення неправдивої інформації в соціальних мережах є однією з небезпек, що зростає. Хоча соціальні медіа можуть використовуватися для кіберзлочинів, ці організації не можуть відмовитися від використання соціальних медіа, оскільки вони відіграють важливу роль у

приверненні уваги організації. Натомість вони повинні мати механізми, які інформуватимуть їх про ризик, щоб виправити це до того, як буде завдано будь-якої фактичної шкоди. У будь-якому випадку, організації повинні розуміти це та розуміти сенс розбиття даних головним чином у соціальних обговореннях і створювати хороші плани безпеки, щоб уникнути небезпек. Потрібно укласти контракт із соціальними медіа, використовуючи конкретні плани та правильні технології.

Кібертероризм

Термін «тероризм» може означати незаконне використання влади або жорстокості проти людей з метою загрожувати адміністрації або її мешканцям та асоціаціям, які можуть мати на меті створення політичного чи шкідливого сайту [10]. Тероризм трансформувався зі звичайної структури в кібертип інноваційного тероризму, визнаного кібертероризмом. Це залишається актуальною проблемою сучасного суспільства. Мало того, що боротьба з тероризмом відстає, поточні кіберзлочинні напади стають все сильнішими та конфронтаційними. Цей тероризм є використанням кібернетичного слова для нападу на основні фундаменти, від яких повністю залежить присутність асоціацій і країн, що може призвести до його закриття.

Компоненти кібертероризму

Кілька атак, як кібертероризм, мають кілька частин, які були виділені численними дослідниками-спостереженнями в дослідницькій мережі. Як зазначають Самуель і Осман (2014), у своїй гіпотетичній моделі визнають п’ять розділів, за якими вони класифікуються як «кібертероризм»; мета насильства, натхнення та відданість місії, яка має бути виконана, коли такий інцидент має місце, вплив, інструменти, які використовуються для направлення такого нападу і атакуючих, область, яка є природою як стратегія діяльності. Він може впевнено знати, знаючи профіль діяльності, яка спонукає до дій винних.

Критичною проблемою «кібертероризму» є мотивація для здійснення таких дій в інтернеті, які призводять до шкоди людям та їхньому майну. Терористи в усьому світі розвивають кіберсвіт, маючи серйозний стимул як етап, за допомогою якого вони можуть здійснювати незвичайні атаки. Юнос і Ахмад (2014) стверджують, що з використанням інновацій у сфері інформації та комунікації терорист може завдати більшої шкоди або спричинити республіці неприємні умови через переривання необхідних адміністрацій. Вони також зазначають, що «терорист у кіберпросторі» завдає більшої шкоди та спустошення, ніж традиційні методи тероризму.

Анонімність природи Інтернету

Анонімність є ключовим елементом, до якого прагне кожен злий винуватець з метою, щоб його персонажа не можна було впізнати після здійснення кіберзлочину. Інтернет – це захищений домен, який є такою ж прихованою сценою для терористів, оскільки вони можуть залишатися невідомими, тому їх особистість не може бути відома.

Злом

Загальним терміном усіх видів несанкціонованого доступу до будь-якої організації мережі «комп'ютерної системи» є злом, який може статися в будь-якій структурі. Велика кількість цих хакерів використовує «грубу силу», тобто підбирає комбінації для кожної окремої літери та цифру яку можна уявити, доки вони не отримають пароль.

Комп'ютерні віруси

Ці віруси розкидані по системі, щоб виконувати шкідливі дії. Це може включати отримання адміністративних прав, крадіжку інформації або навіть порушення роботи системи.

Перевірка пароля

«Кібертерорист» може використовувати один із методів, наприклад, перехоплення пароля, як спосіб здійснення своєї «кібератаки» на різні країни та

великі організації, щоб спричинити їх крах і контролювати їхні системи. Сніффер паролів - це програма, яка використовується для моніторингу мережевого трафіку та перехоплення всіх паролів, що передаються через системний інтерфейс.

Наслідки кібертероризму

Кібертероризм – це специфічний вид кібернебезпеки та атаки, яка має багато наслідків, коли спрямована проти країн та організацій. Деякі наслідки кібертероризму визначаються наступним чином:

Вторгнення в дані:

Кібертероризм може знищити інформаційну цілісність, роблячи інформацію недостовірною та недоступною. Швидкість поширення кібертероризму серед організацій та інформаційних систем країн спричинила масу труднощів через втрату важливої інформації, яку зазвичай важко відновити.

Напад на бізнес:

Через кібертероризм організації можуть втратити мільярди доларів. Дані банку можуть бути атаковані або зламані терористами, які отримують несанкціонований доступ до фінансових рахунків, що може призвести до втрати мільйонів доларів і навіть до банкрутства банку.

Втрата життя:

Кібертероризм спричинив багато випадків загибелі людей та зруйнував багато життів, викликаючи психічні травми у постраждалих родин. Кібертероризм може призводити до загибелі та завдавати серйозної шкоди, як показують численні авіаційні аварії, спричинені кібернападами на комп'ютери та мережі.

Довіра споживачів під загрозою:

Розвиток будь-якої організації залежить від довіри, яку клієнти мають до неї. Кібертероризм підриває цю довіру, що може негативно вплинути на відносини між організацією та її клієнтами.

Кібербезпека в електронному урядуванні:

Електронне урядування є зусиллям урядів покращити відносини зі своїми громадянами через Інтернет. Існуючі та потенційні загрози в сфері кібербезпеки є одними з найскладніших викликів 21 століття. Для забезпечення безпеки електронного урядування необхідні найкращі практики захисту даних, налаштування політик, практик та методів безпеки, а також використання технологій безпеки. Відкрита приватна організація є важливою частиною кібербезпеки в електронному урядуванні, оскільки вони можуть зручно вирішувати проблеми координації. Потужні заходи із запобігання кіберзлочинам та притягнення до відповідальності є критично важливими в умовах сучасних ІКТ.

Справа про викрадення Касперського

«Найбільш резонансна» справа про кіберстеження, переслідування та викрадення включала Івана Касперського, сина адміністратора та генерального директора російської «Лабораторії Касперського», однієї з провідних компаній з кібербезпеки у світі. Івана Касперського викрали з метою отримання викупу в 2011 році, коли він йшов на роботу зі свого московського житла. За повідомленнями російських ЗМІ, злочин організувала досвідчена пара, яка залучила свою дитину та двох її друзів як «силову підтримку». Викрадачі переслідували Касперського та його кохану деякий час, спостерігаючи за його звичками та виявивши, що він не мав охорони. Ймовірно, зловмисники отримали всі необхідні дані з профілю Касперського на Вконтакте, відомій російській соціальній мережі. Касперський був змушений зателефонувати своєму батькові, щоб передати вимоги викрадачів про викуп. Можливо, викрадачі використовували подібний бездротовий зв'язок для доставки їжі або мали доступ до геолокаційних даних.

Практичний приклад Uber

Витоки даних трапляються щодня в багатьох місцях, але ризик витоку даних не обов'язково залежить від кількості, він також може залежати від ризику

та шкоди, яку це завдає доходам компанії та впливу на користувачів або власників облікових записів. Одним із найбільших недавніх витоків даних був інцидент з Uber.

Одна з нещодавніх великих кібератак – розкриття особистої інформації близько 57 мільйонів користувачів Uber і 600 000 водіїв Uber. Найгірша частина цієї атаки полягає в тому, як Uber впорався з проблемою, що стало уроком для багатьох компаній, чого не слід робити. Наприкінці 2016 року лише два хакери змогли отримати особисті дані користувачів, включаючи імена, номери телефонів та адреси електронної пошти. Вони також викрали інформацію про 600 000 водійських прав. Хакери отримали доступ до облікового запису GitHub Uber через сторонній хмарний сервіс. Завдяки інформації, знайденій на GitHub, хакери знайшли спосіб отримати доступ до даних користувачів Uber в AWS. Uber заплатив цим двом хакерам 100 000 доларів, щоб вони назавжди знищили всі отримані дані та не повідомили користувачів чи регуляторів про викрадену інформацію.

Uber також підтвердив, що дані були знищені, спираючись на гарантії, які вони отримали від хакерів. Згідно з американськими законами, про будь-яке порушення слід повідомляти владі, а не платити хакерам. Такий підхід Uber змусив інших хакерів шантажувати Netflix, погрожуючи опублікувати телевізійні шоу, якщо компанія не заплатить гроші. Майже в 49 штатах діють закони, що вимагають повідомляти про порушення безпеки, і після судових слухань Uber погодився виплатити 20 мільйонів доларів, щоб погасити претензії FTC. На це відреагували не лише США, але й інші великі країни, такі як Великобританія, Італія та Філіппіни. Порушення Uber відрізняється від звичайних порушень, оскільки компанія намагалася приховати інцидент і не попередила владу та користувачів.Новий генеральний директор Uber, отримав кілька спірних проблем лише щодо юридичного питання, а також критику за сексуальні домагання, недоплату водіям і багато іншого.

Деякі рішення щодо кібербезпеки та кібертероризму описані так:

– методи кібербезпеки. Деякі методи можна використовувати для покращення кібербезпеки;

– контроль доступу та «безпека паролем»: ідея пароля та імені користувача є основним методом забезпечення даних. Це можуть бути основні заходи щодо кібербезпеки;

– документи, які отримуються, мають бути перевірені перед передачею. Необхідно впевнитися, що вони надходять з надійного джерела та не були змінені. Перевірка цих записів зазвичай здійснюється за допомогою антивірусного програмного забезпечення, наявного на пристроях. Для захисту пристроїв від вірусів також необхідне надійне антивірусне ПЗ.

– антивірусне програмне забезпечення: це комп'ютерна програма, яка класифікує, уникає та намагається завдати шкоди шкідливим програмам, наприклад, вірусам і хробакам. Більшість «антивірусних програм» містять функцію «автоматичного оновлення», яка дозволяє програмі завантажувати профілі нових вірусів з метою перевірки нових вірусів, коли вони їх знаходять;

– сканери зловмисного програмного забезпечення: це програмне забезпечення, яке зазвичай фільтрує кожен із записів і архівів, поточних у структурі, на предмет мстивого коду або руйнівних вірусів [10]. Віруси, хробаки, а також троянські коні є екземплярами «шкідливого програмного забезпечення», яке регулярно збирається та називається шкідливим програмним забезпеченням;

– брандмауер: «програма програмного забезпечення» або обладнання, яке допомагає стежити за хакерами, інфекціями та всіма типами хробаків, які намагаються проникнути на ПК через Інтернет. Усі дані, які передаються туди й назад через Інтернет, проходять через сучасний брандмауер, який переглядає кожен окреме повідомлення та перешкоджає, які не відповідають вимогам безпеки та класифікують їх як загрозу та намагаються заблокувати систему та

контролювати дії. Відтепер брандмауери беруть на себе важливу роботу з розпізнавання шкідливих програм.

Запобігання кібертероризму

Спроможність запобігати кібертероризму залежить від здатності надійно перевіряти кіберпростір. Кібербезпека має інтригуючу паралель із тероризмом. Обидва однобокі. Гарантувати безпеку інформації, даних і листування вражаюче складніше, ніж зламати структуру. Зловмисник має невід'ємну перевагу як у звичайних терористичних діях, так і в кібератаках. З огляду на атаки, які підтримуються державою, труднощі набагато більші. Уряди повинні гарантувати, що їхні правила не сприяють кіберзлочинам, а також мають бути повністю актуалізовані та відповідати нормам безпеки. Важливо, щоб країни вжили заходів для забезпечення того, щоб їхнє каральне та технічне законодавство було достатнім для вирішення труднощів, пов'язаних із кіберзлочинами.

Доступність, конфіденційність і цілісність інформації в будь-яких асоціаціях є важливими, і необхідно докладати зусиль, щоб гарантувати їх виняткову безпеку, оскільки це значний кіберресурс, який робить кожну асоціацію стійкою та тим часом залежною від неї. Інформація, введена «кібертерористом», є чимось поза межами записів, що може включати повідомлення, вебдодатки, вебсторінки та інші незамінні операційні системи.

1.2 Класифікація SQL-атак

SQL Injection - це метод використання бази даних вебдодатку. Це робиться шляхом введення операторів SQL як вхідного рядка для отримання несанкціонованого доступу до бази даних. SQL-ін'єкція - це серйозна вразливість, яка призводить до високого рівня компрометації - як правило, можливість виконання будь-якого запиту до бази даних. Це атака на вебпрограму, яка підключається до серверних баз даних і дозволяє обійти брандмауер.

Зловмисник користується перевагою незахищеного коду та поганої перевірки вхідних даних для виконання неавторизованих команд SQL[4-7].

Мета SQL Injection Attack (SQLIA) полягає в тому, щоб налаштувати систему бази даних на виконання шкідливого коду, який може розкрити конфіденційну інформацію. Таким чином, SQL-ін'єкція - це несанкціонований доступ до бази даних.

Ін'єкційна атака першого порядку

Ін'єкційні атаки першого порядку - це випадки, коли зловмисник отримує потрібний результат миттєво шляхом прямої відповіді від програми, з якою він взаємодіє, або через інший механізм відповіді, як-от електронна пошта тощо.

Ін'єкційна атака другого порядку

Атака ін'єкції другого порядку - це реалізація шкідливого коду, який впроваджується в програму зловмисником, але не активується негайно програмою[1]. Шкідливі дані вводяться зловмисником у систему або базу даних. Це використовується для непрямого запуску SQLIA, який використовується пізніше. Зловмисник зазвичай покладається на те, де вхідні дані будуть використані згодом, і таким чином створює свою атаку. Ін'єкція другого порядку залишає клопітку роботу з виявлення та запобігання. Це пояснюється тим, що точка ін'єкції відрізняється від точки, де насправді вона проявляється.

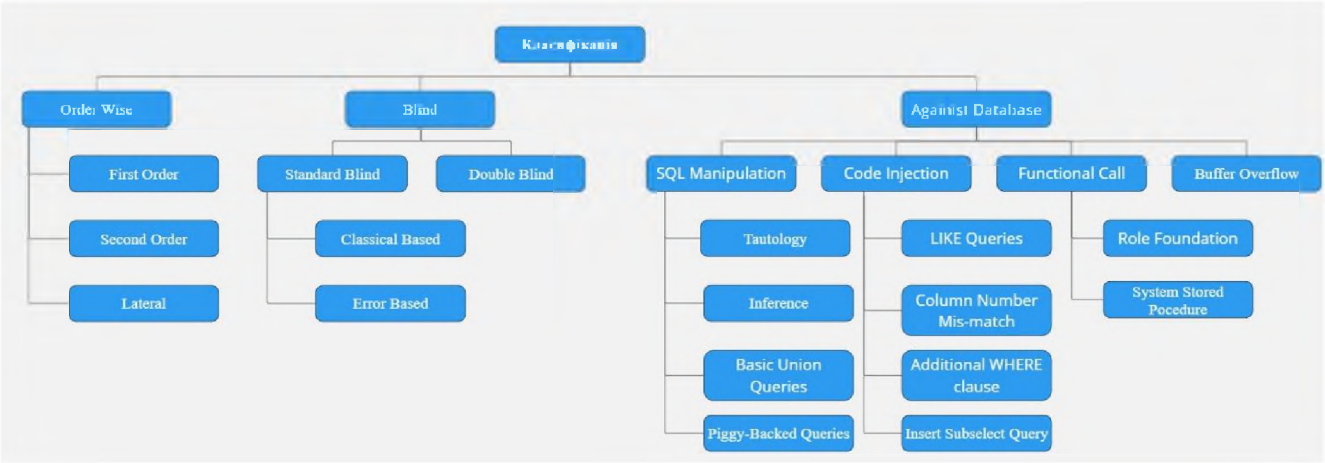

Рисунок 1.2 – Класифікація SQL-ін'єкцій

Бічна ін'єкційна атака

Це техніка, яка використовується для віддаленого злому баз даних Oracle. Атака використовує деякі поширені типи даних, зокрема ДАТА та НОМЕР, які не потребують введення від користувача, тому зазвичай не вважається придатним для експлуатації. Зловмисник може маніпулювати базою даних, просто використовуючи трохи творчого кодування [2].

Класичний SQL-ін'єкційна атака

Класична експлуатація вразливостей SQL-ін'єкції дає можливість об'єднати два SQL-запити. Це виправдовує мету отримати додаткові дані з певної таблиці. Отримати доступ до інформації з системи керування базами даних стане легше за допомогою класичні SQL-ін'єкції.

Сліпа атака SQL-ін'єкції

Якщо вебпрограма вразлива до SQL-ін'єкції, але результати ін'єкції приховані для зловмисника, використовується сліпий SQLA. Відображена сторінка може відрізнятися від оригінальної. Це залежить від результатів логічного оператора, вставленого в законний оператор SQL, викликаний для цієї сторінки. Сліпий SQLIA дозволяє агенту загрози зробити висновок про конструкцію бази даних через оцінювальні вирази, які поєднуються з операторами, які завжди оцінюються як істинні, і операторами, які завжди оцінюються як хибні. Сліпе впровадження SQL можна класифікувати як:

Стандартна жалюзі

Іноді неможливо використати вразливість SQL Injection за допомогою класичної техніки, яка передбачає застосування оператора <union> і роздільника<";">. У цій ситуації реалізовано стандартний сліпий SQLIA[3]. Це дозволяє записувати та читати файли та отримувати дані з таблиць за умови, що лише записи читаються символ за символом. Цей вид нападу можна додатково класифікувати наступним чином:

Класичний на основі сліпого впровадження SQL. Ця атака базується на аналізі логічного виразу істини/хибності. Якщо вираз істинний, то вебдодаток повертатиме певний вміст, а якщо він хибний, то додаток повертатиме вміст [3].

Сліпий на основі помилок SQL ін'єкції. Сліпа SQL-ін'єкція на основі помилок є найшвидшою технікою використання SQL-ін'єкції. Головною перевагою цього методу є те, що цінну інформацію різних СУБД можна зберігати в повідомленнях про помилки в разі отримання недопустимого виразу SQL. Цю техніку можна використовувати, якщо будь-яка помилка обробки виразу SQL, яка сталася в СУБД, повертається назад уразливою програмою[3].

Подвійне сліпе впровадження SQL – це техніка, при якій всі повідомлення про помилки та вразливі запити виключаються зі сторінки, яку повертає вебпрограма, а результати запиту не впливають на повернуту сторінку. Експлуатація такого роду уразливості використовує лише часові затримки під час обробки запитів SQL. Тобто, якщо SQL-запит виконується негайно, це означає, що він невірний, але якщо він виконується із N-секундною затримкою, тоді це вважається правильним запитом [3].

Проти бази даних

На основі атаки на систему керування базою даних, SQL-атаку можна класифікувати наступним чином:

Маніпуляції SQL. Найпоширенішим типом атаки SQL Injection є маніпулювання SQL. Зловмисник намагається змінити існуючий оператор SQL[13]. Цю атаку маніпуляції SQL можна класифікувати як:

Тавтологія. Метою атаки на основі тавтології є впровадження коду в один або кілька умовних операторів, щоб оцінка завжди була істинною. У цьому типі SQLIA зловмисник використовує ін'єктоване поле, яке використовується в умові WHERE запиту, тобто запити завжди повертають результати після оцінки умовного параметра WHERE. Усі рядки в таблиці бази даних, на які спрямований запит, повертаються під час перетворення умови в тавтологію. Приклад: у цьому

прикладі зловмисник надсилає « ' ' або 1=1 - -» для поля введення логіна (введення, подане для інших полів, не має значення). Отриманий запит:

```
SELECT accounts FROM users WHERE login='' or 1=1 -- AND pass='' AND
pin=''
```

Код, введений в умовний оператор (АБО 1=1), перетворює весь WHERE речення в тавтологію.

Під час цієї атаки запит модифікується у форму дії, яка виконується на основі відповіді на запитання true/false про значення даних у базі даних. У цьому типі ін'єкції зловмисник намагається атакувати сайт, який достатньо захищений, щоб не забезпечити прийнятний зворотний зв'язок через повідомлення про помилку бази даних, коли ін'єкція вдалася. Зловмисник повинен використовувати інший метод, щоб отримати відповідь від бази даних, оскільки повідомлення про помилки бази даних йому недоступні. У цій ситуації зловмисник вводить команди на сайт, а потім спостерігає, як змінюється функція/відповідь вебсайту. Уважно спостерігаючи за мінливою поведінкою сайту, зловмисник може екстраполювати не тільки вразливі параметри, але й додаткову інформацію про значення в базі даних. Дослідники повідомили, що за допомогою цих методів їм вдалося досягти швидкості вилучення даних 1 Б/с.

Наприклад розглянемо дві можливі ін'єкції в поле входу. Перший - «legalUser' і 1=0 - -», а другий - «legalUser' і 1=1 - ». Результатом цих ін'єкцій є наступні два запити:

```
SELECT accounts FROM users WHERE login='legalUser' and 1=0 -- ' AND
pass='' AND pin=0
SELECT accounts FROM users WHERE login='legalUser' and 1=1 -- ' AND
pass='' AND pin=0ВИБРАТИ облікові записи ВІД користувачів WHERE
login='legalUser'і 1=1 -- ' AND pass='' AND pin=0
```

Перший сценарій: Програма безпечна, і введення для входу перевірено правильно. У цьому випадку обидві ін'єкції повернули б повідомлення про помилку входу, і зловмисник зрозумів би, що параметр входу не є вразливим.

Другий сценарій: Програма небезпечна, і параметр входу вразливий до впровадження. Зловмисник надсилає першу ін'єкцію та отримує повідомлення про помилку входу, оскільки воно завжди оцінюється як false. Однак зловмисник не знає, чи сталося це тому, що програма правильно перевірила введені дані та заблокувала спробу атаки, чи сама атака спричинила помилку входу. Потім зловмисник надсилає другий запит, який завжди оцінюється як істина. Якщо в цьому випадку немає повідомлення про помилку входу, зловмисник знає, що атака пройшла успішно і що параметр входу вразливий до ін'єкції.

Основні запити об'єднання

За допомогою цієї техніки зловмисник обманом змушує сервер повернути дані, які розробники не передбачали. У цій атаці зловмисник використовує вразливий параметр, щоб змінити набір даних, що повертається для певного запиту. Зловмисники роблять це, вводячи оператор такої форми: UNION SELECT <решта введеного запиту>. Оскільки зловмисник повністю контролює другий/введений запит, він може використовувати його для отримання інформації з указаної таблиці. У результаті цієї атаки база даних повертає набір даних, який є об'єднанням результатів початкового першого запиту та результатів ін'єктованого другого запиту

Підкріплені запити

У цьому SQLIA зловмисники не мають на меті змінити запит; вони намагаються включити нові та чіткі запити в оригінальний запит. Ця база даних отримує численні запити SQL і може бути надзвичайно шкідливою. Приклад: якщо зловмисник вводить «'; скинути таблицю користувачів - -» у поле пропуску, програма генерує запит:

```
SELECT accounts FROM users WHERE login='doe' AND pass=''; drop table
users -- ' AND pin=123
```

Після виконання першого запиту база даних розпізнає роздільник запиту (";") і продовжить виконання другого запиту. Виконання другого запиту призведе до видалення таблиці «користувачів», що, ймовірно, пошкодить цінну інформацію

Введення коду

Атаки з впровадженням коду намагаються додати додаткові оператори SQL або команди до існуючого оператора SQL. Цей тип атаки часто використовується проти програм Microsoft SQL Server, але рідко працює з базою даних Oracle[13]. Цю атаку можна класифікувати як:

Запити LIKE

Зловмисник намагається маніпулювати SQL, використовуючи подібні запити. Отже, введення користувача, включене в параметр запиту LIKE, може підірвати запит, ускладнити відповідність LIKE і в багатьох випадках запобігти використанню індексів, що суттєво сповільнює запит. З кількома ітераціями скомпрометований запит LIKE може викликати атаку на відмову в обслуговуванні через перевантаження бази даних[4][11]. приклад:

```
    $sub = mysql_real_escape_string ("%щось"); // ще %щось mysql_query
(«ВИБЕРІТЬ * З повідомлень, ДЕ тема, ЯК '{$sub}%'»).
```

Номер стовпця невідповідність

Щоб зрозуміти цей тип ін'єкції, розглянемо приклад. Нехай вихідний запит виглядає так:

```
SELECT product_name FROM all_products WHERE product_name, наприклад
"&Chairs&"
```

Тоді атака буде:

```
SELECT product_name FROM all_products WHERE product_name, наприклад ''
UNION ALL SELECT 9,9 FROM SysObjects WHERE '' = ''
```

Наведений вище запит видає помилки, які вказують на невідповідність кількості стовпців і їх типів даних в об'єднанні таблиці SysObjects і стовпців, указаних за допомогою 9. Помилка «Невідповідність типу операції» головним чином виникає через те, що тип даних невідповідність у реченні Union, викликана введеним рядком. Інша помилка, яку можна побачити: «Усі запити в інструкції SQL, що містить оператор UNION, повинні мати однакову кількість виразів у своєму цільовому списку» через те, що кількість стовпців не збігається.
Після численних спроб і помилок заявка це може вдатися.

```
SELECT product_name FROM all_products WHERE product_name, наприклад ''
UNION ALL SELECT 9,9, 9,' Text', 9 FROM SysObjects WHERE '' = ''
```

Набір результатів наведеного вище запиту відображатиме всі рядки в таблиці SysObjects, а також відображатиме постійні значення рядків для кожного рядка в таблиці SysObjects, визначеної в запиті [4].

Додаткове речення WHERE

Випадок виникає, коли в операторі SL може бути додаткова умова WHERE, яка додається після вставленого рядка.

```
SELECT firstName, LastName, Title from Employees WHERE City = ' "&
strCity &" ' AND Country = 'Ukraine' "З першої спробипісля введення рядка
кінцевий запит виглядатиме так:
SELECT firstName, Last Name, Title from Employees WHERE City =
'NoSuchCity' UNION ALL Виберіть OtherField з OtherTable WHERE 1 = 1 AND
Country = 'USA'
```

Наведений вище запит призведе до такого повідомлення про помилку: Недійсна назва стовпця «Країна», оскільки «Інша таблиця» не має стовпця під назвою «Країна». Якщо серверною частиною є MS SQL Server, проблему можна вирішити за допомогою «;--», щоб позбутися решти рядка запиту [4].

Вставити – Підвибір запиту.

Використання розширеного запиту на вставку може допомогти зловмиснику отримати доступні всі записи бази даних.

```
INSERT INTO tableName значення (' " ' + (SELECT TOP 1 Fieldname FROM
tableName ) + ' ' , 'qwe@qwe.com' , '240402364')
```

Там, де ця інформація відображається користувачеві, зазвичай розміщуються сторінки, на яких користувачам дозволено редагувати інформацію користувача. У наведеній вище атаці перше значення в FieldName відображатиметься замість імені користувача. Якщо TOP 1 не використовується, з'явиться повідомлення про помилку «підвибір повернув забагато рядків». Зловмисник може переглядати всі записи, використовуючи NOT у пункті [4].

Функціональна ін'єкція виклику

Ін'єкція виклику функції – це вставка функцій бази даних у вразливий оператор SQL. Ці виклики функцій можна використовувати для викликів операційної системи або маніпулювання даними в базі даних[13]. Це можна класифікувати як:

Основа ролі

Багато компаній надають прес-релізи через свій портал. Зазвичай запити користувачів на прес-реліз виглядатимуть так:

Відповідний оператор SQL, який використовується програму буде виглядати так:

Виберіть заголовок, опис, дату випуску, тіло з pressRelease WHERE pressRelaseID=5

Сервер бази даних повертає всю необхідну інформацію, що відповідає 5-му прес-релізу. Ця інформація форматується програмою на сторінці HTML і надається користувачеві.

Якщо введений рядок дорівнює 5 I 1 = 1, а програма все ще повертає той самий документ, тоді програма чутлива до атаки SQL-ін'єкцій[4].

Система збереженої процедури

У цьому типі SQLIA зловмисник намагається виконати збережені процедури, присутні в базі даних. Якщо запущений внутрішній сервер відомий, зловмисник здійснює свою атаку, щоб використати збережену процедуру. Якщо він здатний успішно ввести рядок SQL, то збережені процедури не безпечніші. Доступ до процедур системного сховища залежить від привілеїв доступу користувача програми до бази даних. У більшості випадків вихідні дані можуть бути відсутні на екрані, як це було б у випадку звичайного оператора SQL під час виконання збереженої процедури успішно[4-6].

Переповнення буфера

У кількох базах даних було виявлено переповнення буфера. Деякі функції бази даних сприйнятливі до переповнення буфера, що може бути використано через атаку SQL-ін'єкції в базі даних без виправлень. Більшість застосування і вебсервери не можуть обробити втрату з'єднання з базою даних через переповнення буфера. Зазвичай вебпроцес зависає, доки не буде розірвано з'єднання з клієнтом, що робить це дуже ефективною атакою на відмову в обслуговуванні[13].

Вище була проведена класифікувція SQL Injection Attack, яка може бути ефективна проти дій зловмисників. Це може допомогти у виправленні або принаймні зменшити ймовірність появи вразливості, яка може завдати шкоди безпеці бази даних. Проводиться детальне вивчення кожної атаки, щоб класифікувати атаку SQL Injection Attack. За допомогою цих категорій галузеві експерти та розробники можуть забезпечити більший захист базам даних вебдодатків.

1.3 Постановка задачі

Вебсайти стали невід'ємною частиною нашого життя, надаючи зручний доступ до інформації та послуг. Але стрімке зростання їх популярності спричинило збільшення кіберінцидентів, серед яких SQL-ін'єкції посідають одне з перших місць за небезпечністю та поширеністю.

SQL-ін'єкції дають можливість зловмисникам отримати несанкціонований доступ до інформації, змінити або й повністю знищити важливі дані. Такі атаки можуть мати серйозні наслідки, фінансових втрат, шкоди репутації та юридичних проблем. Тому забезпечення захисту інформації стає ключовим аспектом кібербезпеки вебресурсів.

Метою кваліфікаційної роботи є розробка методів захисту інформації від SQL-загроз, які запобігають несанкціонованому доступу до баз даних та гарантували цілісність та конфіденційність інформації.

Для досягнення поставленої мети було сформульовано наступні завдання: провести дослідження актуальних проблем кібербезпеки, вивчити проблему SQL-загроз та їх класифікацію, проаналізувати мови програмування, бібліотеки та бази даних, які використовуються для розробки вебсайтів, розробити програмне рішення для захисту від SQL-ін'єкцій, провести тестування розробленого програмного рішення.

1.4 Висновки

У цьому розділі було розглянуто існуючі практики у галузі інформаційної безпеки SQL-баз даних. Цей аналіз виявив, що незважаючи на значні зусилля та досягнення у забезпеченні безпеки даних, як і раніше, існують значні вразливості та виклики. Зокрема, основні проблеми включають недостатню захищеність програмного забезпечення від SQL-ін'єкцій, слабкі паролі, неправильну конфігурацію баз даних, відсутність шифрування даних, а також недостатню обізнаність користувачів про сучасні методи захисту. Необхідність постійного

оновлення та удосконалення методів безпеки стає критично важливою, оскільки зловмисники також розвивають свої техніки та знаходять нові способи обійти захист. Також було розглянуто різні види SQL-атак, що представляють загрозу для вебресурсів. SQL-ін'єкції є одними з найпоширеніших та найбільш небезпечних методів атаки, які дозволяють зловмисникам отримати несанкціонований доступ до баз даних, маніпулювати даними або навіть знищити їх. Таким чином, захист вебресурсів від SQL-загроз вимагає ретельного підходу та уваги до актуальних проблем.

РОЗДІЛ 2. СПЕЦІАЛЬНА ЧАСТИНА

2.1 Аналіз мов програмування

Python - це високорівнева, інтерпретована, динамічна мова програмування, яка здатна вирішувати різноманітні завдання, від простих сценаріїв до складних вебдодатків та наукових обчислень. Ось детальний огляд переваг і недоліків використання Python:

Переваги Python

Простий синтаксис: Python має зрозумілий і простий для вивчення синтаксис, що робить його ідеальним вибором для початківців та професіоналів. Широка підтримка спільноти: Python має велику та активну спільноту, яка регулярно розвиває бібліотеки та фреймворки, що полегшують розробку програм. Багато бібліотек: Python має велику екосистему бібліотек для різних цілей, таких як робота з базами даних, обробка даних, машинне навчання, веброзробка тощо. Портативність: Python підтримується на багатьох платформах, включаючи Windows, macOS і різні дистрибутиви Linux, що робить його універсальним і доступним для розробників. Швидкість розробки: Python дозволяє розробникам швидко прототипувати та впроваджувати програми завдяки своїй простоті та зручному синтаксису.

Недоліки Python:

Виконання на великих об'ємах даних: У порівнянні з деякими іншими мовами програмування, такими як C++ або Java, Python може бути повільним у виконанні на великих об'ємах даних.

Неідеальна підтримка паралельного програмування: Хоча Python має модулі для паралельного програмування, такі як multiprocessing та threading, вони не завжди ефективні через особливості інтерпретатора Python [13].

Не підходить для деяких задач високої продуктивності: У деяких областях, де вимагається висока продуктивність, таких як вбудовані системи або ігри, Python може бути не найкращим вибором через свою інтерпретовану природу.

Незважаючи на ці недоліки, Python залишається однією з найпопулярніших мов програмування завдяки своїм перевагам у сфері швидкості розробки, доступності та потужності.

JavaScript (JS) - це мова програмування, яка використовується для розробки вебдодатків, включаючи фронтенд і бекенд, а також для створення мобільних додатків та додатків для IoT (Internet of Things). Ось детальний огляд переваг і недоліків використання JavaScript:

Переваги JavaScript:

Вебдодатки та інтерактивність: JavaScript використовується для створення інтерактивних елементів вебсторінок, таких як анімація, форми, ігри та інші візуальні ефекти.

Широка підтримка браузерами: JavaScript підтримується всіма сучасними браузерами, що робить його ідеальним для розробки вебдодатків, які працюють на будь-якому пристрої.

Node.js для серверного програмування: JavaScript може бути використаний для розробки серверних додатків за допомогою платформи Node.js, що дозволяє використовувати той же код як на фронтенді, так і на бекенді.

Багатофункціональність: JavaScript є мовою високого рівня, яка має багато вбудованих функцій і бібліотек для роботи з різними типами даних, включаючи рядки, масиви, об'єкти тощо.

Широке використання: JavaScript використовується не лише для веброзробки, але й для мобільних додатків (за допомогою фреймворків, таких як React Native або Ionic) та для розробки додатків для IoT.

Недоліки JavaScript:

Перенесення помилок на сторону клієнта: Оскільки JavaScript виконується на стороні клієнта, помилки в коді можуть призвести до некоректної роботи вебсторінки або додатку.

Інтерпретована мова програмування: JavaScript є інтерпретованою мовою, що може призвести до меншої швидкодії в порівнянні з компільованими мовами програмування.

Крос-браузерна сумісність: Існує проблема крос-браузерної сумісності, коли деякі функції JavaScript можуть працювати по-різному на різних браузерах.
Безпека: JavaScript може бути вразливим до атак, таких як внідрення коду (injection) або cross-site scripting (XSS) атаки, якщо не враховувати правила безпеки програмування.

Хоча JavaScript має свої недоліки, він залишається однією з найбільш популярних мов програмування завдяки своїм перевагам у сфері веброзробки, доступності та розширюваності.

PHP (Hypertext Preprocessor) - це мова програмування, яка використовується для розробки вебдодатків та вебсайтів. Ось детальний огляд переваг і недоліків використання PHP:

Переваги PHP:

Простота вивчення: PHP має простий і зрозумілий синтаксис, що робить його досить легким для вивчення, навіть для початківців.

Широке використання: PHP є однією з найпоширеніших мов програмування для веброзробки і використовується для створення різних типів вебдодатків, від простих блогів до великих електронних комерційних платформ.
Багатофункціональність: PHP має велику кількість вбудованих функцій та розширень, що полегшує роботу з базами даних, обробкою форм, створенням зображень тощо.

Зручність інтеграції: PHP добре інтегрується з різними СУБД, такими як MySQL, PostgreSQL, SQLite тощо, що дозволяє розробникам легко працювати з базами даних.

Велика спільнота: PHP має велику та активну спільноту розробників, що забезпечує широкий доступ до документації, форумів підтримки та багато інших ресурсів.

Недоліки PHP:

Безпека: PHP має деякі вразливості, такі як вразливості типу переповнення буфера та вразливості SQL-ін'єкції, що можуть стати причиною атак на вебдодатки.

Низька продуктивність: У порівнянні з деякими іншими мовами програмування, такими як Java або C++, PHP може мати меншу продуктивність та швидкодію, особливо для великих та складних додатків.

Неявна типізація: PHP має слабку підтримку статичної типізації, що може призвести до непередбачуваного поведінки програми та помилок.

Нестабільність функціоналу: Історично PHP мала проблеми зі стабільністю деяких функцій та розширень, що може призвести до проблем при оновленні або міграції додатків на нові версії PHP.

Хоча PHP має свої недоліки, вона залишається однією з найпопулярніших мов програмування для веброзробки завдяки своїм перевагам у сфері доступності, багатофункціональності та широкому використанню.

Ruby - це динамічна, об'єктно-орієнтована мова програмування, яка була створена в Японії в 1995 році Юкіхіро Мацумото.

Переваги Ruby:

Простий та читабельний синтаксис: Ruby має природній та зрозумілий синтаксис, що робить код легким для читання та розуміння. Це сприяє швидкому розвитку та підтримці проектів.

Об'єктно-орієнтованість: Ruby повністю об'єктно-орієнтована мова, де все є об'єктом. Це спрощує роботу з даними та створення більш структурованих програм.

Багата бібліотека: Ruby має велику кількість готових бібліотек та фреймворків, які значно полегшують розробку різноманітних додатків, включаючи вебдодатки, мобільні додатки та інші.

Ruby on Rails: Ruby має відомий та потужний вебфреймворк Ruby on Rails, який прискорює розробку вебдодатків та надає велику кількість зручних інструментів та бібліотек для роботи з базами даних, маршрутизації, аутентифікації тощо.

Динамічна типізація: Ruby має динамічну типізацію, що означає, що змінні не повинні бути явно оголошені з типом даних, що полегшує написання коду та зменшує кількість помилок.

Недоліки Ruby:

Швидкодія: У порівнянні з деякими іншими мовами програмування, такими як C++ або Go, Ruby може бути повільним виконуваним мовом через його інтерпретований характер.

Ресурсомісткість: Ruby може вимагати багато ресурсів системи, що може призвести до проблем з продуктивністю на великих проектах або під навантаженням.

Крос-платформенність: Хоча Ruby підтримується на багатьох платформах, розгортання додатків на Windows може викликати проблеми через різницю у середовищах.

Помірна популярність: Хоча Ruby має велику та віддану спільноту, вона може бути менш популярною, ніж інші мови програмування, що може вплинути на доступність ресурсів та документації.

Хоча Ruby має свої недоліки, вона залишається потужним та популярним інструментом для розробки різноманітних додатків, зокрема вебдодатків, завдяки своїм перевагам у сфері простоти, гнучкості та багатофункціональності.

Go, також відома як Golang, - це мова програмування, розроблена в компанії Google, яка поєднує в собі ефективність компільованих мов, таких як C або C++, з простотою і сучасністю скриптових мов програмування. Ось детальний огляд переваг і недоліків використання Go:

Переваги Go:

Швидкість виконання: Go володіє вражаючою швидкістю виконання завдяки своїй компільованій природі, що робить його ідеальним вибором для високонавантажених додатків та сервісів.

Ефективність використання ресурсів: Go відомий своєю ефективністю використання ресурсів системи, що дозволяє економити пам'ять і CPU ресурси на великих проектах.

Простий синтаксис: Синтаксис Go простий та легкий для вивчення, що робить його популярним серед початківців та досвідчених розробників.

Конкурентність: Go має вбудовану підтримку конкурентності, що дозволяє легко створювати паралельні процеси та оброблювати великий обсяг одночасних запитів.

Стандартна бібліотека: Go має потужну стандартну бібліотеку, яка включає в себе багато корисних інструментів для роботи з мережами, шифруванням, обробкою JSON і багато іншого.

Недоліки Go:

Брак обратної сумісності: З часом Go може змінюватися, що може призвести до проблем з обратною сумісністю між версіями, особливо для великих проектів.

Молода екосистема: Хоча Go має швидко зростаючу спільноту та екосистему, вона може бути менш розвиненою, ніж у деяких інших мов програмування.

Неідеальна підтримка generics: У Go не було повноцінної підтримки generics до версії 1.18, що може призвести до більш складного коду у деяких випадках [15].

Менша кількість бібліотек: Незважаючи на швидке зростання, екосистема Go все ще може мати менше бібліотек порівняно з іншими мовами програмування.

Не зважаючи на ці недоліки, Go залишається потужним та ефективним інструментом для розробки різноманітних додатків, особливо у сферах високонавантажених систем та обробки одночасних запитів.

2.2 Аналіз баз даних

MySQL - це одна з найпопулярніших відкритих систем управління базами даних (СУБД), яка широко використовується для зберігання та керування реляційними базами даних.

1. Надійність та швидкість:

MySQL відома своєю високою надійністю та швидкодією, що робить її популярним вибором для великих вебсайтів, систем управління контентом та корпоративних додатків.

Вона оптимізована для роботи з великим обсягом даних та високими навантаженнями, що дозволяє їй ефективно обробляти тисячі запитів за короткий час.

2. Розширюваність:

MySQL підтримує різні методи розширення, включаючи горизонтальне та вертикальне масштабування, реплікацію та розподілені транзакції. Це дозволяє

розгортати бази даних MySQL на серверах з великою кількістю ядер та пам'яті для забезпечення високої продуктивності та доступності.

3. Функціональність:

MySQL має широкий набір функцій, включаючи підтримку транзакцій, індексацію даних, зберігані процедури, тригери, представлення, операції з JSON, регулярні вирази та інші. Вона підтримує різні типи даних, такі як числа, рядки, дати, часи, зображення, географічні дані, що робить її універсальним рішенням для різних типів додатків.

4. Легкість використання:

MySQL має простий та зрозумілий SQL-синтаксис, який робить легкою роботу з базою даних для розробників усіх рівнів.

Вона підтримує велику кількість інструментів адміністрування, таких як phpMyAdmin, MySQL Workbench, що полегшує управління базою даних.

5. Відкритість та спільнота:

MySQL - це відкрита СУБД з активною спільнотою розробників, яка надає регулярні оновлення, виправлення помилок та нові функції.

Вона має широку документацію та велику кількість ресурсів для навчання та підтримки.

В цілому, MySQL є потужним, надійним та широко використовуваним рішенням для зберігання та управління реляційними базами даних у різних типах програм.

PostgreSQL - це потужна об'єктно-реляційна система управління базами даних (СУБД), яка відкрита, гнучка та розширювана. Ось детальний огляд особливостей PostgreSQL:

1. Надійність та стабільність:

PostgreSQL відома своєю високою надійністю та стабільністю. Вона має вбудовану підтримку транзакцій, журналізації та відновлення після збоїв, що робить її ідеальним рішенням для критичних застосунків.

2. Розширюваність:

PostgreSQL підтримує різні методи розширення, включаючи горизонтальне та вертикальне масштабування, реплікацію, розподілені транзакції та розподілені обробки запитів.

Це дозволяє гнучко налаштовувати та розгортати бази даних для відповіді на різні потреби та обсяги даних.

3. Функціональність:

PostgreSQL має багатий набір функцій, включаючи підтримку тригерів, збережених процедур, виразів, віконних функцій, географічних типів даних, операцій з JSON та інші.

Вона підтримує також різні типи даних, такі як рядки, числа, дати, часи, зображення, географічні дані тощо.

4. Сумісність:

PostgreSQL підтримує стандарт SQL ANSI, що робить її сумісною з багатьма іншими СУБД і дозволяє легко переносити додатки з однієї системи на іншу.

Вона також підтримує різні рівні ізоляції транзакцій, включаючи серіалізовану ізоляцію, що забезпечує високий рівень консистентності даних.

5. Розвиток та спільнота:

PostgreSQL має активну та велику спільноту розробників, яка надає регулярні оновлення, виправлення помилок та нові функції.

Вона має докладну документацію, різноманітні ресурси для навчання та підтримки, а також різноманітні розширення та допоміжні інструменти.

Узагальнюючи, PostgreSQL є потужною та надійною СУБД, яка підходить для різних типів додатків та завдань. Вона забезпечує широкий набір функцій, високу продуктивність та надійність, що робить її відмінним вибором для великих та критичних застосунків.

MongoDB - це документоорієнтована система управління базами даних (NoSQL), яка використовує JSON-подібні документи для зберігання даних. Ось детальний огляд особливостей MongoDB:

1. Схема документів:

MongoDB використовує гнучку схему документів, що дозволяє зберігати дані у вигляді JSON-подібних об'єктів, які можуть мати різні поля та структури.

Це дозволяє легко змінювати структуру даних та додавати нові поля без необхідності зміни схеми бази даних.

2. Горизонтальне масштабування:

MongoDB підтримує горизонтальне масштабування, що дозволяє розгортати бази даних на кількох серверах та автоматично розподіляти навантаження між ними.

Це дозволяє підтримувати великі обсяги даних та високу доступність в розподілених середовищах.

3. Швидкодія та продуктивність:

MongoDB відома своєю високою швидкодією та продуктивністю завдяки використанню індексації даних, кешування та іншим оптимізаціям.

Вона також підтримує запити до бази даних у форматі мови запитів MongoDB (MongoDB Query Language), яка дозволяє швидко та ефективно отримувати дані.

4. Гнучкість та розширюваність:

MongoDB має широкий набір функцій, таких як підтримка транзакцій, агрегаційних функцій, текстового пошуку, геопросторових операцій, операцій з масивами та інші.

Вона також підтримує різні формати даних, такі як документи, масиви, геодані та інші, що дозволяє зберігати різнорідні дані.

5. Спільнота та інструменти:

MongoDB має велику та активну спільноту розробників, яка надає регулярні оновлення, виправлення помилок та нові функції.

Вона має також широкий вибір інструментів для адміністрування, моніторингу та розробки, таких як MongoDB Compass, MongoDB Atlas, MongoDB Shell та інші.

Узагальнюючи, MongoDB є потужною та гнучкою системою управління базами даних, яка підходить для різних типів додатків та завдань. Вона надає широкий набір функцій, високу швидкодію та надійність, що робить її відмінним вибором для розробки сучасних додатків.

Redis - це швидка та потужна система управління даними типу ключ-значення, яка часто використовується для кешування, сесій та повідомлень у вебдодатках. Ось детальний огляд особливостей Redis:

1. Простота та ефективність:

Redis має простий та легкий у використанні інтерфейс команд, який дозволяє швидко зберігати та отримувати дані у форматі ключ-значення.

Вона працює у пам'яті, що дозволяє отримувати швидкий доступ до даних та виконувати операції з ними з великою швидкодією.

2. Типи даних:

Redis підтримує різні типи даних, включаючи рядки, хеші, списки, множини та сортовані множини.

Це дозволяє зберігати різнорідні дані та використовувати їх для різних цілей, від кешування до обробки потоків даних.

3. Висока доступність:

Redis підтримує реплікацію та шардування, що дозволяє розгортати кластери Redis на кількох серверах та забезпечувати високу доступність та надійність. Вона також має вбудовану підтримку механізму моніторингу та автоматичного відновлення, що дозволяє швидко виявляти та виправляти збої.

4. Підтримка транзакцій:

Redis підтримує транзакції, що дозволяє виконувати групу команд атомарно та забезпечувати консистентність даних.

Вона також підтримує різні операції над множинами даних, такі як об'єднання, перетин, різниця тощо.

5. Гнучкість та розширюваність:

Redis має широкий вибір розширень та модулів, які дозволяють розширювати його функціональність та використовувати його для різних цілей. Вона також має велику спільноту розробників, яка надає різноманітні інструменти та бібліотеки для роботи з Redis.

Узагальнюючи, Redis є потужним та ефективним інструментом для кешування, сесій та повідомлень у вебдодатках. Вона надає швидку доступність до даних, гнучкість та високу доступність, що робить її відмінним вибором для великих та складних проектів.

2.3 Аналіз фреймворків

Flask - це легкий та гнучкий мікрофреймворк для розробки вебдодатків на мові програмування Python. Розроблений Арміном Ронхейзером у 2010 році, Flask став популярним інструментом для створення вебдодатків через свою простоту, гнучкість та широкий функціонал.

Однією з головних переваг Flask є його легкість використання. Він має мінімальний набір залежностей та вбудовані можливості, що дозволяє розробникам швидко розгортати вебдодатки без зайвого навантаження або складності.

Flask також володіє простим та інтуїтивним синтаксисом, що робить його дуже легким для вивчення та використання. Це особливо корисно для початківців у веброзробці, а також для проектів, які вимагають швидкого прототипування або розробки MVP.

Незважаючи на свою простоту, Flask має великий функціонал і можливості для розширення. Він підтримує роботу з шаблонами Jinja2 для відображення

сторінок, вбудовану обробку запитів та маршрутизацію, роботу з формами, керування сесіями та багато іншого.

Однією з ключових особливостей Flask є його модульність та розширюваність. Він має велику кількість розширень, які додають до нього додатковий функціонал, такий як автентифікація користувачів, робота з базами даних, взаємодія з API, інтеграція з іншими сервісами та багато іншого.

Загалом, Flask є потужним та гнучким інструментом для розробки вебдодатків на мові Python. Він володіє простотою та гнучкістю, що робить його відмінним вибором для широкого спектру проектів, від простих лендінгів до складних вебсервісів.

Node.js - це вільна, відкрита платформа, яка використовується для виконання коду JavaScript на серверному боці. Розроблена Райаном Далем та представлена у 2009 році, Node.js стала одним з найпопулярніших інструментів у сфері серверної розробки завдяки своїм унікальним характеристикам та перевагам [20].

Однією з ключових особливостей Node.js є його асинхронний та подійно-орієнтований характер. Виконання операцій у Node.js зазвичай здійснюється асинхронно, що дозволяє ефективно обробляти велику кількість одночасних запитів без блокування потоків. Це робить його ідеальним вибором для розробки високопродуктивних та масштабованих додатків, таких як вебсервери, чат-додатки та API.

Node.js базується на движку JavaScript V8, розробленому Google для браузера Chrome. Це означає, що Node.js володіє високою швидкодією та ефективністю виконання коду. Велика спільнота розробників та активна підтримка забезпечують постійний розвиток та вдосконалення платформи.
Node.js має широкий вибір модулів та пакетів, доступних через пакетний менеджер npm. Це дозволяє розробникам швидко розширювати функціонал своїх

додатків та використовувати готові рішення для різноманітних завдань, від роботи з базами даних до роботи з мережевими запитами.

Крім того, Node.js підтримує розробку за допомогою сучасних підходів, таких як серверні рендерери React або Vue, що дозволяє створювати універсальні (isomorphic) додатки, які можуть виконуватися як на клієнтському, так і на серверному боці.

Узагальнюючи, Node.js - це потужна та ефективна платформа для розробки серверних додатків на мові JavaScript. Вона надає розробникам зручний та ефективний інструмент для створення високопродуктивних та масштабованих додатків з використанням асинхронного програмування та широкого вибору модулів і бібліотек.

React.js - це бібліотека JavaScript для створення інтерфейсів користувача. Вона була розроблена Facebook і випущена в 2013 році. React відомий своєю декларативною та компонентною природою, що робить розробку вебдодатків ефективною та зручною.

Однією з ключових особливостей React є використання віртуального DOM (Document Object Model). React використовує віртуальний DOM для ефективного оновлення сторінок без перезавантаження вебсторінки. Він відстежує зміни у стані додатку та автоматично оновлює відповідні елементи інтерфейсу користувача, забезпечуючи швидку та плавну відповідь на дії користувача.

Ще однією важливою особливістю React є його компонентна модель. Вебдодатки, розроблені з використанням React, складаються з невеликих та незалежних компонентів, які можуть бути легко перевикористані та модифіковані. Це робить код більш структурованим, легшим для розуміння та підтримки.

React також використовує концепцію властивостей (props) та стану (state). Властивості передаються компонентам ззовні та не змінюються, тоді як стан визначається внутрішньо і може змінюватися залежно від дій користувача або

інших факторів. Ця роздільність дозволяє створювати динамічні та інтерактивні компоненти.

Однією з сильних сторін React є його активна та велика спільнота. Існують тисячі додаткових бібліотек, компонентів та інструментів, які розроблені спільнотою для полегшення розробки на React. Також існують популярні фреймворки, такі як Next.js та Gatsby.js, які розширюють можливості React та додають новий функціонал.

Узагальнюючи, React - це потужна та популярна бібліотека для створення інтерфейсів користувача. Вона надає зручні та ефективні засоби для розробки вебдодатків з використанням декларативного підходу, компонентної моделі та віртуального DOM. Реакт відомий своєю швидкістю, продуктивністю та великою спільнотою, що забезпечує постійний розвиток та підтримку.

Laravel - це високорівневий фреймворк для розробки вебдодатків на мові програмування PHP. Він був розроблений Тейлором Отвеллом і вперше випущений у 2011 році. Laravel використовує сучасні технології та практики розробки для спрощення процесу створення вебдодатків та забезпечення швидкого розгортання проектів.

Однією з ключових особливостей Laravel є його елегантний та зручний синтаксис. Він надає широкий набір зручних інструментів та функціоналу для роботи з базами даних, маршрутизації, сесіями, аутентифікацією, кешуванням та багато іншого, що робить розробку додатків швидкою та простою.

Ще однією важливою особливістю Laravel є його модульність та розширюваність. Він має велику кількість готових компонентів та розширень, які дозволяють розробникам швидко додавати новий функціонал до своїх додатків. Крім того, Laravel надає зручний механізм для створення та використання власних пакетів.

Фреймворк також надає широкий набір інструментів для тестування коду, що дозволяє розробникам забезпечувати якість своїх додатків та впевненість в

їхній роботі. Laravel підтримує автоматизовані тести одиниць, інтеграційні тести та інші види тестування, що дозволяє розробникам швидко виявляти та виправляти помилки в коді.

Однією з найбільш популярних особливостей Laravel є його фреймворк для створення вебдодатків - Laravel Eloquent ORM. Eloquent забезпечує зручний та ефективний спосіб взаємодії з базами даних, використовуючи сучасний синтаксис та підтримуючи багато різних видів відносин.

Узагальнюючи, Laravel - це потужний та прогресивний фреймворк для розробки вебдодатків на мові PHP. Він надає зручний та елегантний спосіб для створення високоякісних та швидких додатків, забезпечуючи розробникам широкий набір інструментів та функціоналу. Laravel заслужено вважається одним з найкращих фреймворків для розробки вебдодатків у світі PHP.

Angular - це популярний фреймворк для розробки вебдодатків, розроблений командою Google. Angular надає потужний набір інструментів та функціоналу для створення високоякісних та складних клієнтських додатків з використанням мови програмування TypeScript.

Однією з ключових особливостей Angular є його компонентна архітектура. Додатки, розроблені на Angular, будуються з використанням компонентів, які є незалежними, перевикористовуваними та легко розширюваними. Компоненти дозволяють розбити додаток на малий, легко керований блоки коду, що спрощує розробку та підтримку проектів.

Angular також володіє потужним модульним системою. Він дозволяє розробникам організовувати свої додатки у вигляді модулів, що містять компоненти, сервіси та інші ресурси. Це дозволяє створювати масштабовані та добре організовані додатки, які легко розширювати та модифікувати.

Однією з основних особливостей Angular є його двостороннє зв'язування даних. Це означає, що зміни в моделі даних автоматично відображаються у

користувацькому інтерфейсі та навпаки, що робить розробку інтерактивних додатків більш простою та ефективною.

Angular також надає широкий набір інструментів для роботи з HTTP запитами, роутінгом, формами, аутентифікацією та іншими аспектами веброзробки. Вбудовані модулі та компоненти дозволяють швидко розгортати різноманітні функції та фічі у додатках.

Крім того, Angular має велику спільноту розробників та активну підтримку з боку Google. Існують тисячі додаткових бібліотек, модулів та інструментів, розроблених спільнотою, які допомагають розширювати можливості фреймворку та вдосконалювати розробку додатків.

Узагальнюючи, Angular - це потужний та ефективний фреймворк для розробки вебдодатків на мові TypeScript. Він надає зручний та сучасний підхід до розробки, широкий набір інструментів та функціоналу, що дозволяє розробникам швидко створювати високоякісні та масштабовані додатки.

Express.js - це легкий та гнучкий вебфреймворк для розробки вебдодатків на платформі Node.js. Він є одним з найпопулярніших інструментів для створення серверних додатків з використанням JavaScript, завдяки своїй простоті, швидкості та розширюваності.

Однією з ключових особливостей Express є його мінімалістичний підхід. Фреймворк надає лише базовий набір функцій для створення вебсервера, такий як маршрутизація, обробка запитів та відправка відповідей. Це дозволяє розробникам більш вільно контролювати та налаштовувати свої додатки, не перевантажуючи їх зайвим функціоналом.

Express використовує простий та інтуїтивно зрозумілий синтаксис, що робить його легким для вивчення та використання. Це особливо корисно для початківців у розробці вебдодатків, а також для швидкого прототипування та розробки MVP.

Однією з сильних сторін Express є його розширюваність. Він має велику кількість модулів та розширень, які дозволяють розробникам додавати новий функціонал до своїх додатків. Наприклад, для роботи з шаблонами можна використовувати модуль "ejs" або "pug", для автентифікації - модуль "passport", для роботи з базами даних - модуль "mongoose" або "sequelize", тощо.

Express також підтримує роботу з middleware - це функції, які мають доступ до об'єктів запиту та відповіді, і можуть виконувати певні дії, оброблювати дані або модифікувати запит перед тим, як він досягне маршруту. Це дозволяє розробникам створювати легко змінювані та перевикористовувані компоненти, що спрощує розробку складних додатків.

Узагальнюючи, Express.js - це легкий та гнучкий вебфреймворк для розробки серверних додатків на платформі Node.js. Він надає простий та ефективний спосіб створення вебдодатків з використанням JavaScript, забезпечуючи розробникам зручний інструмент для створення швидких та масштабованих додатків.

Ruby on Rails (зазвичай скорочено як Rails) - це високорівневий фреймворк для швидкої та ефективної розробки вебдодатків на мові програмування Ruby. Розроблений Девідом Генем Генсом та випущений у 2004 році, Rails швидко став одним з найпопулярніших фреймворків для веброзробки завдяки своїй простоті, продуктивності та конвенції над конфігурацією.

Однією з ключових особливостей Rails є його принцип "Конвенція над конфігурацією" (Convention over Configuration, або CoC). Цей підхід дозволяє розробникам швидко розгортати проекти, забезпечуючи стандартизацію та автоматизацію багатьох аспектів розробки, таких як маршрутизація, ORM (Object-Relational Mapping), тестування тощо.

Rails також використовує паттерн проектування Model-View-Controller (MVC), що дозволяє розділити логіку додатка на три складові: моделі (Models),

представлення (Views) та контролери (Controllers). Це спрощує структуру додатка, робить код більш організованим та підтримуваним.

Rails має велику кількість вбудованих модулів та бібліотек, які дозволяють розробникам швидко додавати різноманітний функціонал до своїх додатків. Наприклад, ActiveRecord - це ORM, який дозволяє взаємодіяти з базою даних за допомогою звичних методів Ruby, а ActionView - це модуль для створення HTML та інших форматів виводу.

Rails також надає потужні інструменти для тестування коду. Вбудований фреймворк для тестування дозволяє розробникам створювати та виконувати різні види тестів, такі як юніт-тести, інтеграційні тести, функціональні тести тощо, що забезпечує високу якість та стабільність додатків.

Загалом, Ruby on Rails - це потужний та продуктивний фреймворк для розробки вебдодатків, який надає розробникам зручні та ефективні інструменти для створення швидких та масштабованих додатків. Він є популярним вибором для стартапів, компаній різних розмірів та проектів різної складності.

Таким чином, для розробки програмного рішення було обрано мову Python, фреймворк Flask та СУБД SQLite.

2.4 Модель загроз

Згідно з національним стандартом ТЗІ 2.5-010-03, модель загроз є основою для аналізу та ідентифікації потенційних ризиків, які можуть вплинути на інформаційну безпеку вебресурсів. У контексті захисту від SQL-загроз, модель загроз включає різноманітні небезпеки, що виникають через вразливості системи[20].

Основні категорії загроз:

1. Несанкціонований доступ до даних

Однією з найгостріших загроз є можливість несанкціонованого доступу до даних. Зловмисники можуть скористатися вразливостями в системах управління

базами даних (СУБД) для отримання доступу до конфіденційної інформації. Основні методи включають: SQL-ін'єкції - введення шкідливих SQL-кодів через вебформи, URL-параметри або URL-рядок, недостатня аутентифікація і авторизація - використання слабких паролів або недосконалої системи контролю доступу.

2. Порушення цілісності даних

Цілісність даних може бути порушена через несанкціоновані зміни або знищення даних. Загрози включають: SQL-ін'єкції для модифікації даних - зловмисники можуть використовувати вразливості для зміни даних у базі, шкідливе програмне забезпечення - використання вірусів або троянів для внесення змін у базу даних.

3. Витік інформації

Витік інформації є серйозною загрозою, яка може призвести до розголошення конфіденційної інформації. Загрози включають: доступ до нешифрованих даних - витік інформації через невикористання шифрування при зберіганні або передачі даних, використання вразливих протоколів зв'язку - передача даних без використання безпечних протоколів (наприклад, HTTP замість HTTPS).

4. Відмова в обслуговуванні (DoS)

Атаки, спрямовані на зниження доступності системи, можуть бути здійснені шляхом перевантаження СУБД або вебсервера. Основні методи включають: Перевантаження запитами: Надмірна кількість SQL-запитів, які призводять до перевантаження сервера. Використання ресурсомістких запитів: Створення запитів, що вимагають великої кількості обчислювальних ресурсів, що призводить до гальмування системи.

5. Шкідливий вплив на продуктивність

Проблеми з продуктивністю можуть бути викликані виконанням довготривалих або складних запитів, які впливають на загальну швидкодію

системи. Основні загрози: Довготривалі запити: Запити, що вимагають великої кількості часу для виконання, що знижує швидкодію системи. Неефективні запити: Запити з неефективними операціями, що займають значну частину ресурсів бази даних.

2.5 Модель порушника

Згідно з НД ТЗІ 2.5-010-03, модель порушника визначає типи потенційних загроз, їх мотивацію, можливості та методи атаки.[20] Для захисту від SQL-загроз потрібно враховувати такі типи порушників:

1. Зовнішні зловмисники.

Хакери мають високий рівень технічних знань і досвід у використанні вразливостей у системах управління базами даних. Зазвичай атакують з метою отримання конфіденційної інформації або порушення цілісності даних. Кібертерористи можуть використовувати SQL-ін'єкції або інші методи для здійснення атак, які можуть призвести до значних збитків або збоїв у роботі системи.

2. Внутрішні загрози

Співробітники можуть використовувати свої привілегії для несанкціонованого доступу до даних або їх модифікації. Це може бути як навмисне, так і ненавмисні дії, що призводять до порушення безпеки. Неуважні користувачі можуть випадково допустити вразливості, не дотримуючись правил безпеки або шляхом несвідомого виконання шкідливих дій.

3. Злочинці з метою фінансової вигоди

Фінансові зловмисники: Мета яких отримання прибутку через викрадення або шантаж даних, використання інформації для здійснення фінансових злочинів. Кіберзлочинці з метою крадіжки даних використовують техніки SQL-ін'єкцій для витоку або крадіжки великих обсягів інформації.

4. Скрипт-кидки та автоматизовані атаки

Автоматизовані інструменти використовують скрипти або спеціалізоване програмне забезпечення для автоматизованого проведення атак, що можуть включати масові SQL-ін'єкції. Скрипт-киди можуть бути використані для автоматизованого запуску SQL-ін'єкцій на великій кількості цілей з метою виявлення вразливостей.

2.6 Розробка додатку

Створимо простий вебдодаток на Flask з використанням SQLAlchemy для взаємодії з базою даних. Почнемо з імпорту необхідних модулів:

Імпортуємо os для роботи з файловою системою та налаштування змінних середовища.

Імпортуємо необхідні класи та функції з Flask: Flask, render_template, request, redirect, url_for.

Імпортуємо SQLAlchemy з Flask для взаємодії з базою даних.

Імпортуємо text з SQLAlchemy для виконання SQL-запитів.

```
import os
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy.sql import text
```

Створюємо екземпляр додатку Flask, використовуючи Flask(__name__). Встановлюємо конфігурацію бази даних за допомогою app.config. Наприклад, використовуємо SQLite як базу даних, шлях до якої зберігаємо у змінній середовища або вказуємо явно.

Створюємо екземпляр SQLAlchemy, передаючи йому наш додаток Flask. Описуємо модель бази даних як клас, який наслідується від db.Model. У цьому класі визначаємо поля таблиці як атрибути класу.

Створюємо маршрути для обробки різних HTTP-запитів (наприклад, GET та POST).

Використовуємо функцію render_template для відображення HTML-сторінок.

Обробляємо дані з форм, використовуючи request.form.

Використовуємо функції redirect та url_for для перенаправлення користувачів після обробки запитів.

Використовуємо app.run для запуску серверу у режимі відладки, що зручно під час розробки.

```
app = Flask(__name__, instance_relative_config=True)
app.config.from_mapping(
    SECRET_KEY='dev',
    SQLALCHEMY_DATABASE_URI='sqlite:///' +
os.path.join(app.instance_path, 'demo_app.sqlite'),
    SQLALCHEMY_TRACK_MODIFICATIONS=False,
```

Для початку створюємо екземпляр Flask-додатку, використовуючи Flask(__name__, instance_relative_config=True), що дозволяє зберігати конфігураційні файли у папці instance. Потім налаштовуємо додаток за допомогою app.config.from_mapping. Вказуємо кілька ключових параметрів конфігурації, таких як секретний ключ SECRET_KEY для захисту даних сесії, а також параметри для підключення до бази даних SQLAlchemy.

У нашому випадку використовуємо SQLite як базу даних, визначаючи шлях до файлу бази даних за допомогою os.path.join(app.instance_path, 'demo_app.sqlite'). Це дозволяє розмістити базу даних у спеціальній папці instance, що знаходиться поруч з додатком, але не входить до репозиторію версій. Також вимикаємо відстеження модифікацій SQLAlchemy за допомогою SQLALCHEMY_TRACK_MODIFICATIONS=False, що зменшує накладні витрати і попереджає про можливі конфлікти.

Це забезпечує базову конфігурацію нашого Flask-додатку та підключення до бази даних, дозволяючи нам далі працювати з моделями бази даних та маршрутизацією запитів.

```
class User(db.Model):
    __tablename__ = 'users'
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50), nullable=False)
    password = db.Column(db.Text, nullable=False)
```

Описуючи модель User, створюємо клас, що наслідується від db.Model, вказуючи, що цей клас відповідає таблиці в базі даних. Таблиця отримує назву users, яку визначаємо за допомогою атрибуту __tablename__.

Модель містить три поля:

Поле id, який є первинним ключем і визначається як стовпець типу Integer з параметром primary_key=True.

Змінна з назвою username, стовпець типу String з обмеженням довжини у 50 символів, що не може бути порожнім (nullable=False).

Змінна з назвою password, стовпець типу Text, також з обмеженням nullable=False.

Ця модель описує основні характеристики користувача в нашій системі, включаючи унікальний ідентифікатор, ім'я користувача та пароль. В подальшому можна буде використовувати цей клас для створення, читання, оновлення та видалення записів користувачів у базі даних.

```
@app.route('/')
def home():
    return render_template('home.html')

@app.route('/sql-injection-demo', methods=['GET', 'POST'])
def sql_injection_demo():
    users = None
    if request.method == 'POST':
        username = request.form['username']
        query = text(f"SELECT * FROM users WHERE username =
'{username}'")
        result = db.session.execute(query)
        users = result.fetchall()
    return render_template('sql_injection_demo.html', users=users)
```

Створюємо маршрут /sql-injection-demo, який обробляє як GET, так і POST запити. У функції sql_injection_demo було ініціалізовано змінну users як None, яка буде використовуватися для зберігання результатів запиту до бази даних.

Якщо метод запиту POST - отримуємо введене значення username з форми за допомогою request.form['username']. Потім створюємо SQL-запит за допомогою text, який вибирає всі записи з таблиці users, де значення username відповідає введеному значенню.

Виконуємо запит через сесію бази даних за допомогою db.session.execute(query) і зберігаємо всі отримані результати у змінну users за допомогою result.fetchall().

Після обробки запиту, незалежно від його типу, повертаємо шаблон sql_injection_demo.html, передаючи йому змінну users, яка містить результати запиту або None, якщо запит не було виконано. Цей шаблон повинен бути розташований у папці templates і використовувати передані дані для відображення користувачів. Цей приклад демонструє потенційну вразливість до SQL-ін'єкції, оскільки введене значення username безпосередньо включається в SQL-запит без належної обробки або перевірки. Для уникнення цієї вразливості варто використовувати параметризовані запити або ORM-засоби, такі як SQLAlchemy.

```
@app.route('/parameterization-demo', methods=['GET', 'POST'])
def parameterization_demo():    # Demonstrating parameterized query as
SQLite does not support stored procedures
    users = None
    if request.method == 'POST':
        username = request.form['username']
        query = text("SELECT * FROM users WHERE username = :username")
        result = db.session.execute(query, {'username': username})
        users = result.fetchall()
    return render_template('stored_procedure_demo.html', users=users)
```

Створюємо маршрут /parameterization-demo', який обробляє як GET, так і POST запити. У функції stored_procedure_demo демонструється використання параметризованого запиту, оскільки SQLite не підтримує збережені процедури.

На початку ініціалізуємо змінну users як None, яка буде використовуватися для зберігання результатів запиту до бази даних. Якщо метод запиту POST, отримуємо введене значення username з форми за допомогою request.form['username'].

Для безпечного виконання SQL-запиту використовуємо параметризований запит з бібліотеки SQLAlchemy. Створюємо запит за допомогою text("SELECT * FROM users WHERE username = :username"), де :username є параметром, який буде замінено фактичним значенням. Виконуємо запит через сесію бази даних за допомогою db.session.execute(query, {'username': username}), передаючи значення параметра у вигляді словника. Отримані результати зберігаємо у змінну users за допомогою result.fetchall().

Після обробки запиту повертаємо шаблон stored_procedure_demo.html, передаючи йому змінну users, яка містить результати запиту або None, якщо запит не було виконано. Цей шаблон повинен бути розташований у папці templates і використовувати передані дані для відображення користувачів.

Використання параметризованих запитів допомагає уникнути вразливостей до SQL-ін'єкцій, забезпечуючи безпечну роботу з введеними користувачем даними.

```
@app.route('/orm-demo', methods=['GET', 'POST'])
def orm_demo():
    users = None
    if request.method == 'POST':
        username = request.form['username']
        users = User.query.filter_by(username=username).all()
    return render_template('orm_demo.html', users=users)
```

Створюємо маршрут /orm-demo, який обробляє як GET, так і POST запити, для демонстрації використання ORM (Object-Relational Mapping) з SQLAlchemy у Flask-додатку.

У функції orm_demo ініціалізуємо змінну users як None, яка буде використовуватися для зберігання результатів запиту до бази даних. Якщо метод

запиту POST, отримуємо значення username з форми за допомогою request.form['username'].

Для виконання запиту до бази даних використовуємо ORM-засоби SQLAlchemy. Викликаємо метод query на класі User для створення запиту і використовуємо метод filter_by, щоб відфільтрувати результати за полем username. Метод all повертає всі знайдені записи, які відповідають умові. Отримані результати зберігаємо у змінну users.

Після обробки запиту повертаємо шаблон orm_demo.html, передаючи йому змінну users, яка містить результати запиту або None, якщо запит не було виконано. Цей шаблон повинен бути розташований у папці templates і використовувати передані дані для відображення користувачів.

Використання ORM дозволяє працювати з базою даних на більш високому рівні абстракції, забезпечуючи більш зручний та безпечний спосіб виконання запитів і обробки даних.

```
@app.route('/add-user', methods=['GET', 'POST'])
def add_user():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        new_user = User(username=username, password=password)
        db.session.add(new_user)
        db.session.commit()
        return redirect(url_for('home'))
    return render_template('add_user.html')
```

Ми створюємо маршрут /add-user, який обробляє як GET, так і POST запити, для додавання нових користувачів до бази даних.

У функції add_user, якщо метод запиту POST, отримуємо значення username та password з форми за допомогою request.form['username'] та request.form['password']. Створюємо новий об'єкт User з отриманими значеннями, використовуючи клас User. Додаємо новий об'єкт до сесії бази даних за

допомогою db.session.add(new_user). Після цього викликаємо db.session.commit(), щоб зберегти зміни в базі даних.

Після успішного додавання нового користувача, перенаправляємо користувача на головну сторінку за допомогою redirect(url_for('home')).

Якщо метод запиту GET, повертаємо HTML-шаблон add_user.html, який відображає форму для введення імені користувача та пароля. Цей шаблон повинен бути розташований у папці templates.

Цей код забезпечує зручний спосіб додавання нових користувачів до системи через вебінтерфейс, використовуючи можливості ORM SQLAlchemy для взаємодії з базою даних.

```
if __name__ == '__main__':
    app.run(debug=True)
```

Завершуючи створення нашого Flask-додатку, додаємо блок if __name__ == '__main__':, щоб забезпечити правильний запуск додатку.

Цей блок перевіряє, чи запускається скрипт безпосередньо (а не імпортується як модуль), і якщо це так, викликає метод app.run(debug=True). Параметр debug=True включає режим відладки, який забезпечує кілька корисних функцій для розробників:

Автоматичне перезавантаження: Додаток автоматично перезапускається при внесенні змін до коду, що значно спрощує процес розробки.

Вивід детальних помилок: У випадку виникнення помилок під час виконання, режим відладки надає детальний трейсбек помилок прямо у веббраузері, що допомагає швидко знаходити і виправляти проблеми.

Отже, весь додаток, включаючи створення екземпляра Flask, налаштування бази даних, визначення моделей, маршрутизацію та запуск серверу, виглядає так:

Імпортуємо необхідні модулі.

Створюємо екземпляр Flask-додатку та налаштовуємо його.

Ініціалізуємо SQLAlchemy.

Визначаємо моделі бази даних.

Створюємо маршрути та функції обробки запитів.

Запускаємо сервер у режимі відладки.

Це забезпечує основну структуру та функціональність для подальшого розвитку вашого вебдодатку на Flask.

2.7 Тестування програми

Відкриваючи сторінку, бачимо привітання на головній сторінці, а також головне меню зверху сторінки.

Додаток містить такі сторінки як «Демонстрація SQL-ін'єкцій», «Демонстрація параметризацій», «Демонстрація ORM», а також форму для додавання нових користувачів у базу.

Рисунок 2.1 – Головна сторінка

Відкриємо першу сторінку:

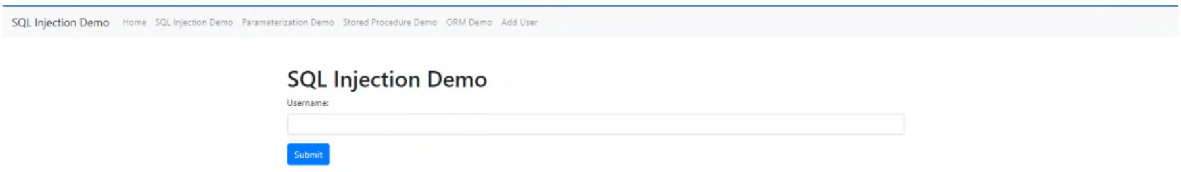

Рисунок 2.2 – Демонстрація SQL-ін’єкцій

Бачимо форму для введення імені користувача. Дана форма має продемонструвати нам можливості SQL вразливостей.

Для того, щоб перевірити її роботу, спершу додамо записи у базу даних на відповідній сторінці:

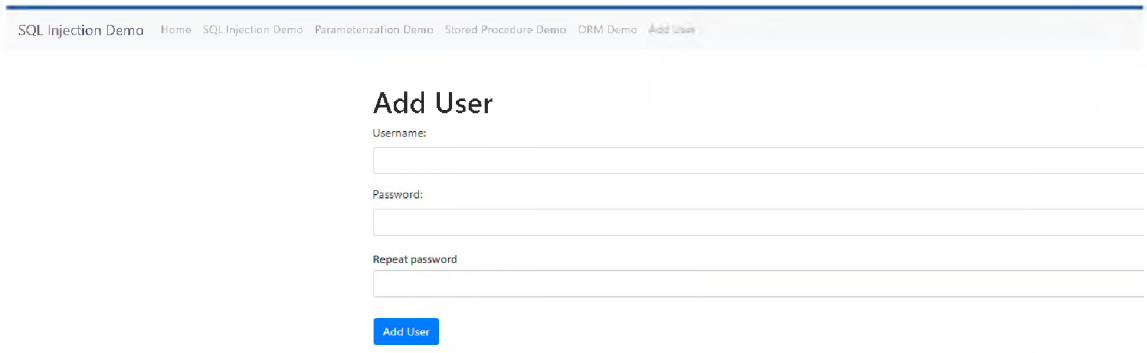

Рисунок 2.3 – Додавання користувача

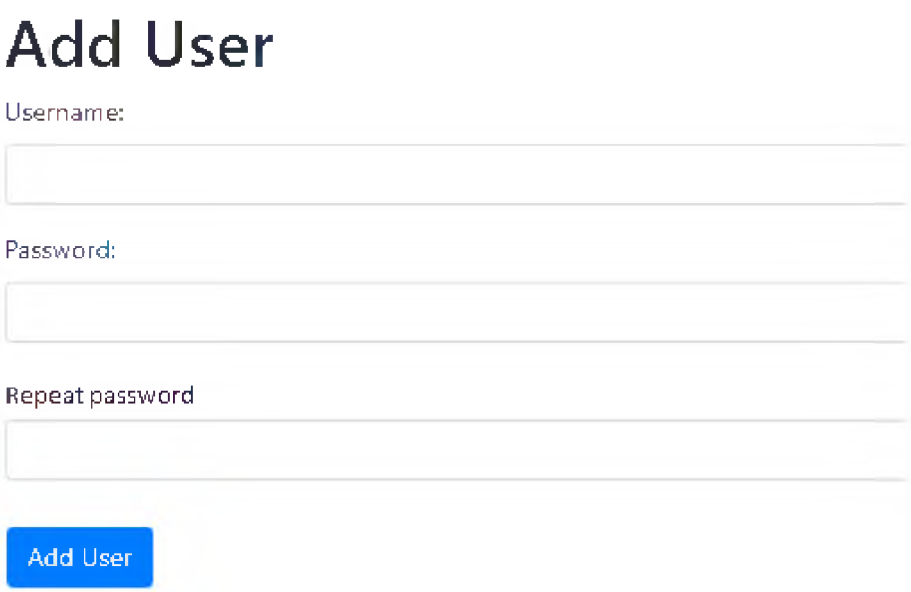

Рисунок 2.4 – Додавання користувача

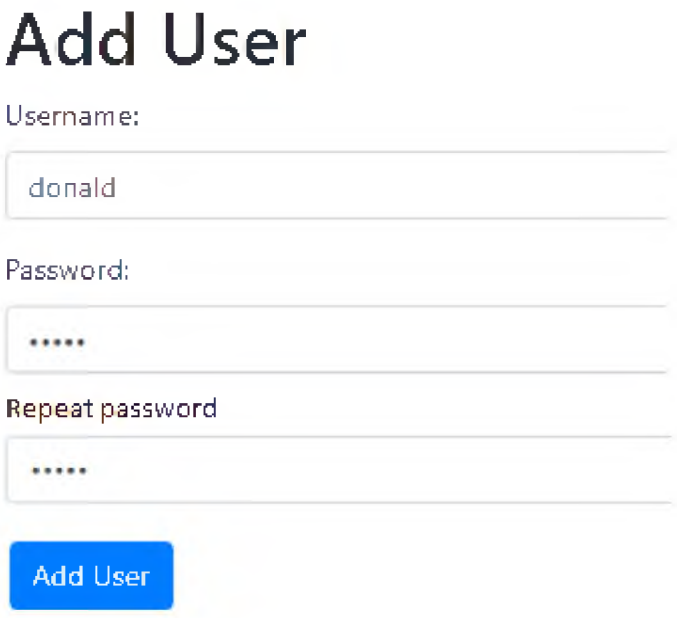

Рисунок 2.5 – Додавання користувача

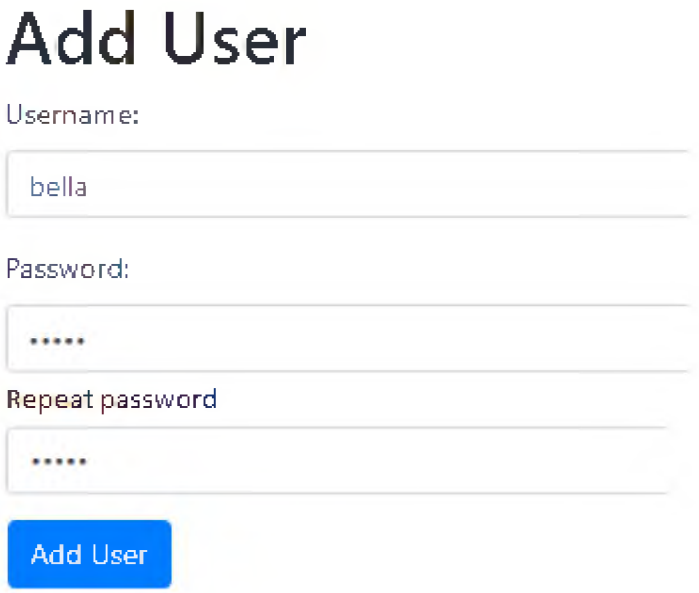

Рисунок 2.6 – Додавання користувача

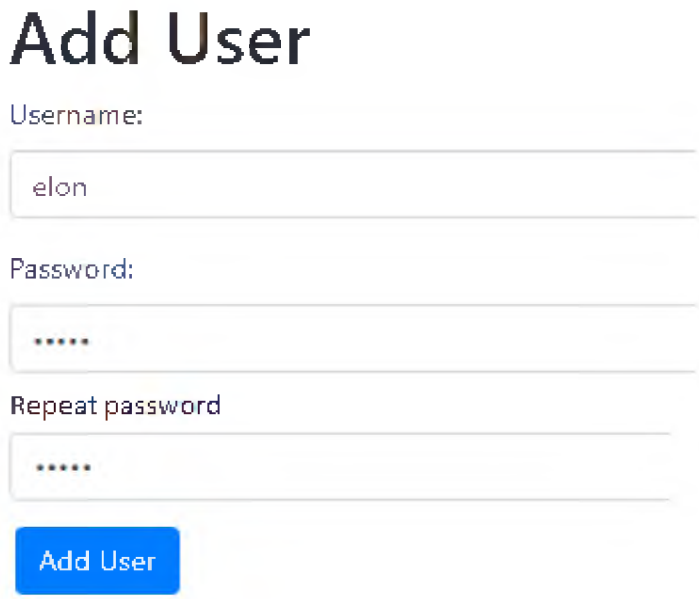

Рисунок 2.7 – Додавання користувача

Далі, коли наша база с користувачами заповнена, можемо повернутись на сторінку з ін’єкціями:

Спочатку, вводимо коректні дані:

Рисунок 2.8 – Запит користувача

Система повертає нам відповідний вивід:

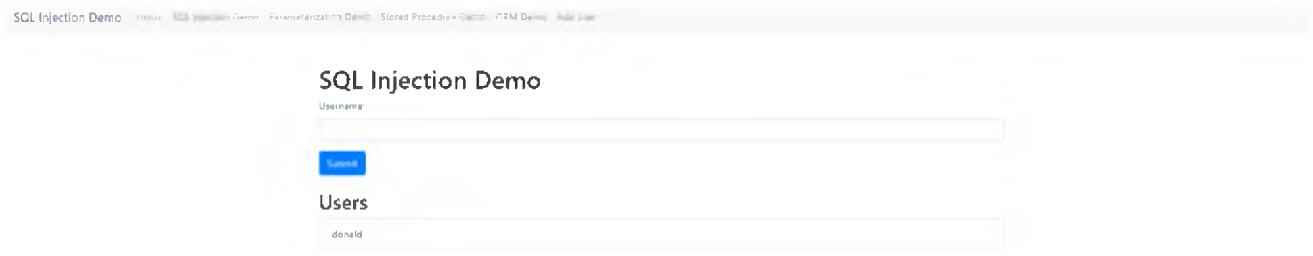

Рисунок 2.9 – Коректна відповідь системи

Тобто система повертає список користувачів, у яких ім’я «donald». Тепер спробуємо зробити SQL-ін’єкцію:

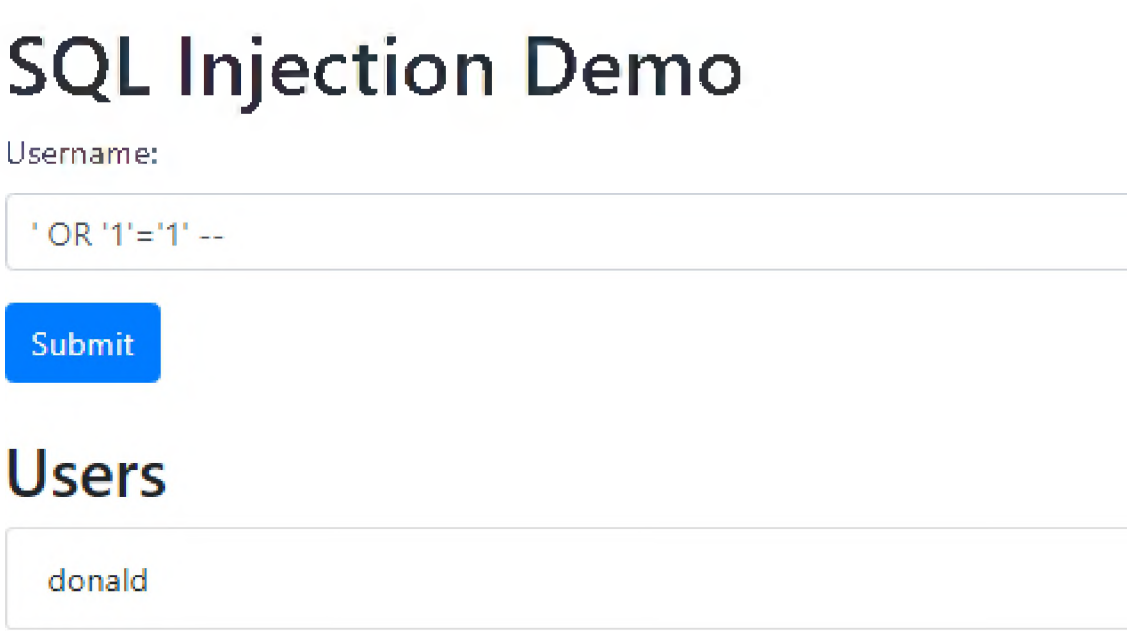

Рисунок 2.10 – SQL-ін’єкція

Отримаємо наступну відповідь:

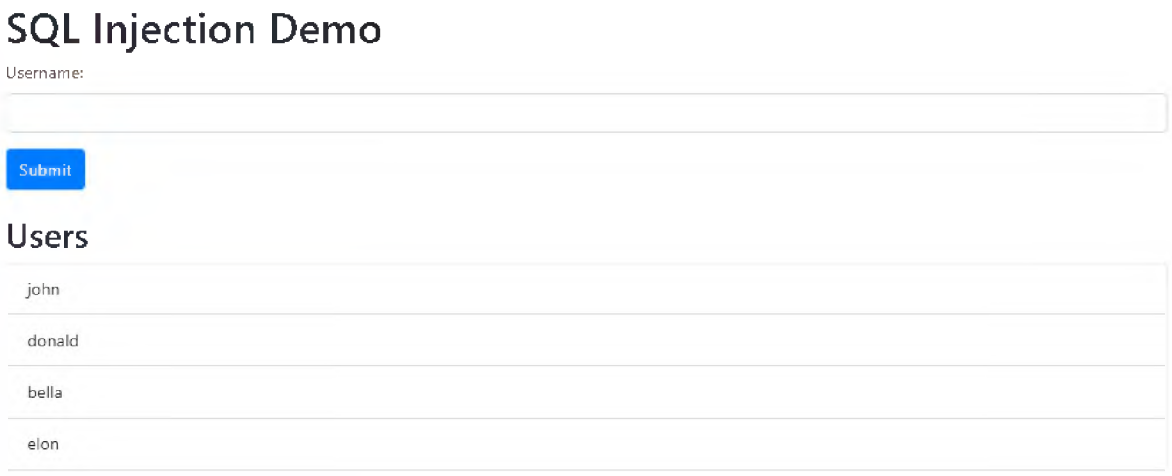

Рисунок 2.11 – Відповідь скомпрометованої системи

Таким чином, зламана система поверне список усіх користувачів. Так відбувається тому, що той код, надісланий у форму було додано у наступний запит:

SELECT * FROM users WHERE username = '{username}'

Далі, перейдемо на сторінку з параметризацією:

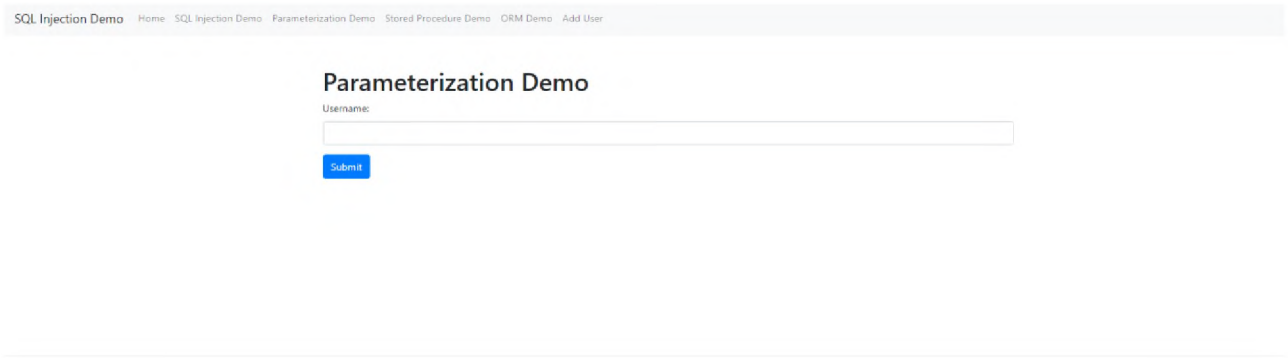

Рисунок 2.12 – Демонстрація параметрізації

Введемо коректні дані у форму:

Рисунок 2.13 – Введення коректного запиту

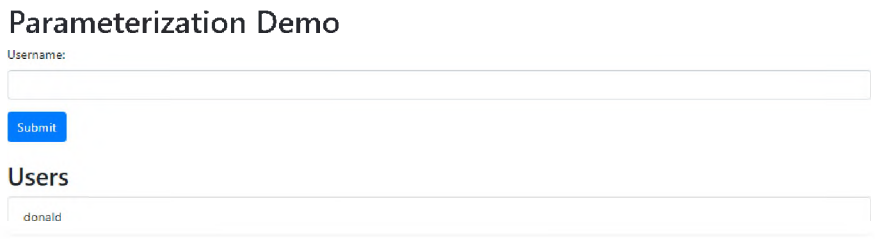

Рисунок 2.14 – Відповідь системи

Бачимо, що тут були повернені користувачі, які мають ім’я, яке було написане у формі.

Спробуємо зробити SQL-ін’єкцію:

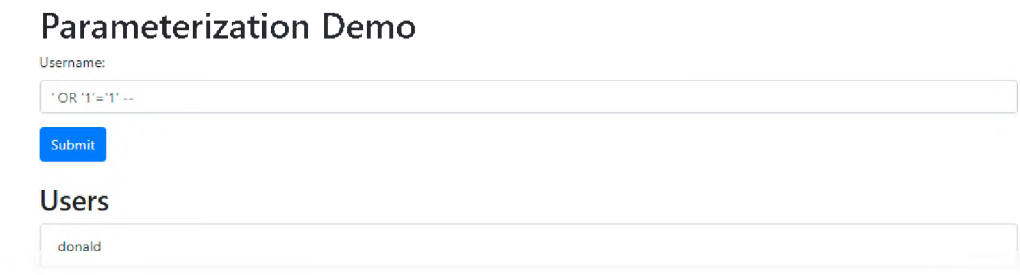

Рисунок 2.15 – SQL-ін’єкція

Рисунок 2.16 – Безпечна відповідь системи

У відповідь бачимо, що система нічого не вивила. Тобто тут була зупинена спроба взлому.

Перейдемо на сторінку «ORM Demo»:

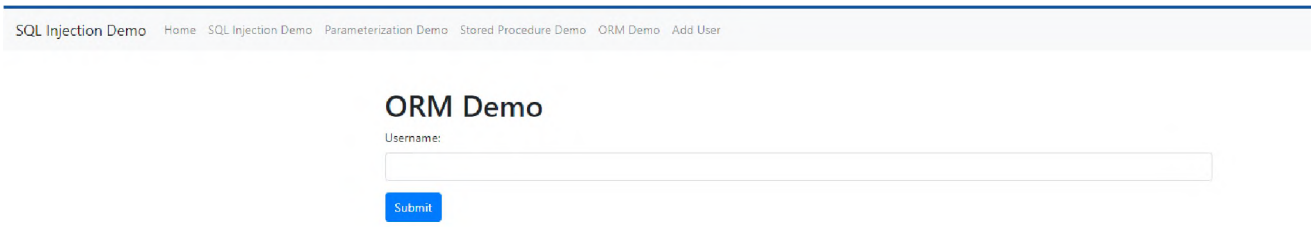

Рисунок 2.17 – ORM Demo

Введемо коректні дані:

Рисунок 2.18 – Введення коректних даних

Отримуємо відповідь:

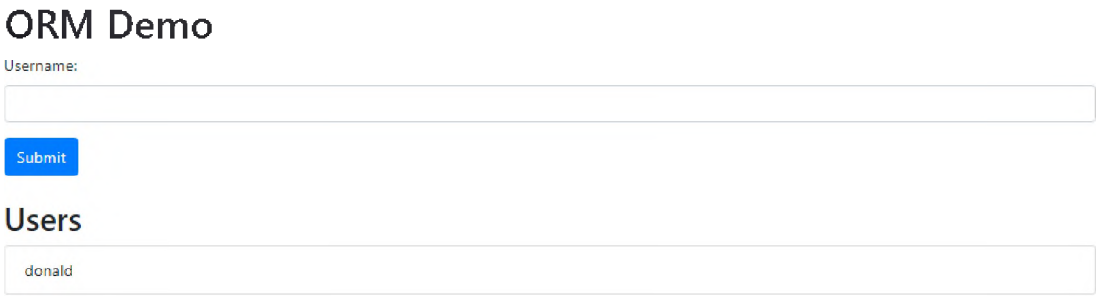

Рисунок 2.19 – Відповідь системи

Спробуємо здійснити SQL-ін’єкцію:

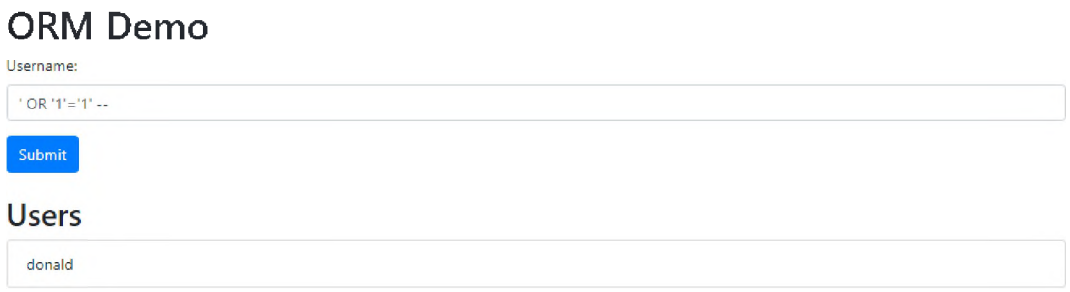

Рисунок 2.20 – Спроба SQL-ін’єкції

Отримуємо відповідь:

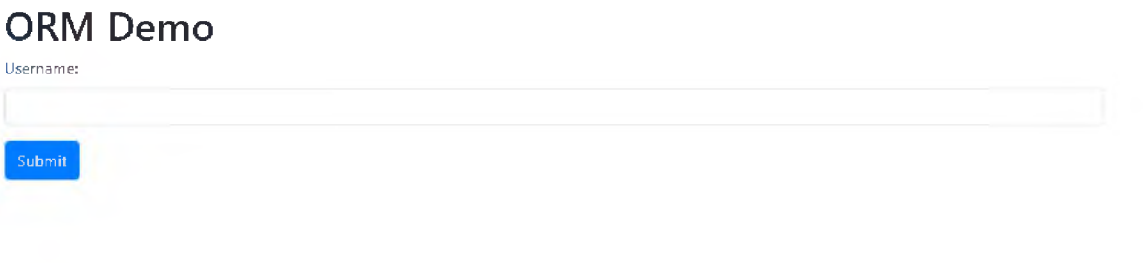

Рисунок 2.21 – Безпечна відповідь системи

Таким чином бачимо, що тут також була припинена спроба атаки на вебресурс.

У підсумку, було розроблено і протестовано додаток, що демонструє основні засоби боротьби із SQL-загрозами для вебресурсів.

2.8 Висновки

У цьому розділі був проведений аналіз мов програмування, їх недоліки та переваги, аналіз бази даних MySQL та обґрунтування вибору саме цієї бази даних. Використання Python у поєднанні з такими фреймворком як Flask, дозволяє створювати ефективні рішення для захисту вебресурсів. Були розглянуті

Моделі загроз та порушників, розроблені згідно з НД ТЗІ 2.5-010-03, та на основі цього документа був розроблений додаток та проведене тестування програми, що забезпечує виявлення та усунення вразливостей, тим самим підвищуючи загальну безпеку вебресурсів.

РОЗДІЛ 3. ЕКОНОМІЧНА ЧАСТИНА

Метою економічної частини кваліфікаційної роботи є аналіз економічної доцільності впровадження програмного забезпечення для захисту вебресурсів від SQL-атак. Адже при витратах на розробку, впровадження та підтримку, що значно перевищують їх ефективність та користь, вони можуть бути визнані недоцільними.

Аналіз економічної доцільності відбувається в декілька кроків. На першому підраховується капітальні витрати на впровадження програмного забезпечення для захисту вебресурсів та закупівлю необхідних технічних засобів для його реалізації. Після визначення усіх витрат необхідно провести вирахування показників ефективності вкладень та зробити висновок про економічну доцільність проекту.

3.1 Розрахунок витрат на впровадження програмного забезпечення для захисту вебресурсів від SQL-атак

Витрати на впровадження та підтримку програмного забезпечення для захисту вебресурсів від SQL-атак складаються з капітальних (одноразових) витрат та витрат для підтримки дієздатності та технічного супроводження нововведень (розраховуються для періоду – 1 рік).

3.2. Розрахунок капітальних (фіксованих) витрат

Щоб розрахувати вартість створення оновлення автентифікації (Кпа), необхідно розрахувати вартість машинного часу (Смч) і помножити на кількість витрачених для оновлення політики автентифікації годин. Програмне забезпечення для використання віддаленого доступу на віртуальний робочий стіл постачається компанією VMware безкоштовно, за умови використання сервісів компанії Amazon.

- P = 750 Вт/*год* = *0,75 кВт/*год (номінальна потужність серверу Dell PowerEdge R640, на якому виконувались роботи).

- Ce = 4.32 грн/кВт (вартість електроенергії).
- Первісна вартість серверу становить 25,000 грн.

Сервер у користуванні вже 2-й рік. Цей технічний засіб амортизується за прямолінійним методом і корисний період його використання становить 5 років. Ліквідаційна вартість технічного засобу 5,000 грн, отже сума щорічних амортизаційних відрахувань становить 4,000 грн, а норма амортизації (На) дорівнює 16%. На початок цього року залишкова вартість склала:

$$\Phi_{зал} = 25{,}000 - 4{,}000 = 21{,}000 \text{ грн}$$

Далі потрібно розрахувати вартість одної години машинного часу ПК ($С_{мч}$):

$$Смч = P \cdot t_{нал} \cdot Ce + \frac{Ф_{зал} \cdot Н_а}{F_р} + \frac{К_{лпз}}{F_р}$$

де:

- P – встановлена потужність серверу, кВт;
- Ce – тариф на електричну енергію, грн/кВт·година;
- $Ф_{зал}$ – залишкова вартість серверу на поточний рік, грн;
- $Н_а$ – річна норма амортизації на сервер, частки одиниці;
- $К_{лпз}$ – вартість ліцензійного програмного забезпечення, грн;
- Fp – річний фонд робочого часу (за 40-годинного робочого тижня Fp=1920).

Вартість ліцензійного програмного забезпечення (SQL Injection Shield від компанії Imperva) становить 30,000 грн на один сервер/рік.

Розрахунок вартості машинного часу:

$$Смч = 0.75 \times 4.32 + \frac{21{,}000 \times 0.16}{1920} + \frac{30{,}000}{1920} = 20.61 \text{ грн/год}$$

Розрахунок вартості створення оновлення автентифікації ($К_{па}$):

$$Кпа = Смч \times 100 = 20.61 \times 100 = 2{,}061 \text{ грн}$$

Витрати на розробку та впровадження включають витрати на розробку програмного забезпечення, інсталяцію та конфігурацію системи. Нижче наведені необхідні витрати:

- Витрати на розробку програмного забезпечення:
- Зарплата розробників (погодинна ставка 400 грн, 100 годин):
- Розробка = 400 грн/год × 100 год = 40,000 грн
- Витрати на інсталяцію та конфігурацію системи:
- Зарплата системних адміністраторів (погодинна ставка 300 грн, 50 годин):
- Інсталяція та конфігурація = 300 грн/год × 50 год = 15,000 грн

Таким чином, витрати на розробку та впровадження програмного забезпечення складають:

- Витрати на розробку програмного забезпечення: 40,000 грн
- Витрати на інсталяцію та конфігурацію системи: 15,000 грн
- Загальні витрати на розробку та впровадження: 55,000 грн
- Загальна сума капітальних витрат становить 57,061 грн.

3.3 Розрахунок річних поточних (експлуатаційних) витрат

Поточні витрати включають витрати на підтримку дієздатності та теRхнічне супроводження програмного забезпечення для захисту вебресурсів. У нашому випадку це витрати на електроенергію, ліцензійне програмне забезпечення, а також технічне обслуговування та підтримку системи. Нижче наведені необхідні витрати:

1. Вартість електроенергії:
   - Сервер Dell PowerEdge R640 споживає 750 Вт (0.75 кВт).
   - Вартість електроенергії: 4.32 грн/кВт·год.
   - Річний фонд робочого часу: 1920 год.

Розрахунок вартості електроенергії:

$$В_{електроенергії} = 0.75 \text{ кВт} \times 4{,}32 \text{ грн/кВт} \cdot \text{год} \times 1920 \text{ год} = 6{,}220 \text{ грн/рік}$$

2. Ліцензійне програмне забезпечення:

- Вартість ліцензійного програмного забезпечення (SQL Injection Shield від компанії Imperva): 30,000 грн/рік.

3. Технічне обслуговування та підтримка:

- Технічне обслуговування та підтримка програмного забезпечення (оплата праці IT-фахівців, погодинна ставка 500 грн, 24 год на рік):

$$В_{техпідтримка} = 500 \text{ грн/год} \times 24 \text{ год} = 12{,}000 \text{ грн/рік}$$

Таким чином, річні поточні витрати складають:

Таблиця 4.1 – Річні поточні (експлуатаційні) витрати

Стаття витрат	Сума, грн/рік
Вартість електроенергії	6,220
Ліцензійне програмне забезпечення	30,000
Технічне обслуговування та підтримка	12,000
Загальна сума поточних витрат	48,220

Таким чином, загальні річні поточні витрати на підтримку програмного забезпечення для захисту вебресурсів від SQL-атак становлять 48,220 грн на рік.

3.4 Оцінка величини можливого збитку від атаки

Для оцінки економічної доцільності можна не враховувати всі загрози та втрати від них. Якщо розглянути тільки деякі загрози, і втрати від них перевищать видатки на впровадження політик безпеки – це вже буде означати доцільність

інвестицій. Для приблизної оцінки збитку за рік розглянемо ті ситуації, що мають найбільшу вірогідність виникнення:

1. Помилки співробітників:

- Вірогідність реалізації загрози через джерело загрози АВн1 та вразливість С2 – 0,173.
- Вірогідність реалізації загрози через джерело загрози АЗв3 та вразливість С2 – 0,115.
- Середня ймовірність реалізації R1: $R1 = \frac{0,173+0,115}{2} = 0,144$

2. Відсутність фізичної захищеності робочого місця:

- Вірогідність реалізації загрози через джерело загрози АВн1 та вразливість С6 – 0,173.
- Вірогідність реалізації загрози через джерело загрози Т6 та вразливість С6 – 0,138.
- Середня ймовірність реалізації R2: $R2 = \frac{0,173+0,138}{2} = 0,156$

3. Використання незахищеного інтернет-з'єднання:

- Вірогідність реалізації загрози через джерело загрози АВн1 та вразливість О5 – 0,384.
- Вірогідність реалізації загрози через джерело загрози Т6 та вразливість О5 – 0,307.
- Вірогідність реалізації загрози через джерело загрози АЗв3 та вразливість О5 – 0,256.
- Середня ймовірність реалізації R3: $R3 = \frac{0,384+0,307+0,256}{3} = 0,316$

Припустимо 3 ситуації, уявімо, що кожна відбудеться раз на рік з врахуванням ймовірнісної характеристики:

1. Перехоплення інформації про стандартне замовлення – витрати на повторне уведення інформації ($П_{ви}$) та втрати від зниження обсягу продажів (V):
$В1 = Пви + V = 400 + 30\,000 = 30\,500$ грн

Витрати на повторне уведення інформації ($П_{ви}$): 500 грн

$$Пви = \frac{\sum Зс}{F} \cdot tви$$

де:

- ∑Зс – сума заробітних плат всіх співробітників, залучених до процесу;
- Тви – Час витрачений на введення інформації;
- F – Загальний фонд робочого часу;

$$Пви = \frac{50000}{100} \cdot 1 = 500 \text{ грн}$$

Втрати від зниження обсягу продажів (V): 30 000 грн

2. Атака на комп'ютер менеджера – викрадення конфіденційної інформації про майбутні 3 замовлення:

$$В2 = Пви + V = 1000 + 30\,000 \times 3 = 91\,000 \text{ грн}$$

Витрати на повторне уведення інформації (Пви): 1000 грн

$$Пви = \frac{5000}{100} \cdot 2 = 1000 \text{ грн}$$

Втрати від зниження обсягу продажів (V) за майбутні 3 замовлення: 30 000 × 3 = 90 000 грн

3. Хакерська атака на комп'ютерну мережу – витрати на відновлення працездатності вузла або сегмента мережі ($П_{пв}$), заміна устаткування ($П_{зч}$), втрати від простою ($П_{п}$):

$$В3 = Пп + Пв + V = 3000 + 2000 + 240{,}000 = 245{,}000 \text{ грн}$$

Витрати на відновлення працездатності вузла або сегмента мережі ($П_{пв}$):300 грн

$$Ппв = \frac{\sum Зо}{F} \cdot tв$$

де:

- $\sum З_о$ – сума заробітної плати адміністраторів;
- $t_в$ – Час на відновлення після атаки;

- F – Загальний фонд робочого часу;

$$\text{Ппв} = \frac{15000}{50} \cdot 1 = 300 \text{ грн}$$

Заміна устаткування (Пзч): 2,000 грн

Втрати від простою (Пп): 3,000 грн

$$\text{Пп} = \frac{\sum \text{Зс}}{F} \cdot t\text{п}$$

де:

- $\sum З_с$ – сума заробітної плати співробітників атакованого вузла;
- $t_п$ – Час простою вузла сегмента корпоративної мережі внаслідок атаки.
- F – Загальний фонд робочого часу;

$$\text{Пп} = \frac{50000}{100} \cdot 6 = 3000 \text{ грн}$$

Сумарний можливий збиток за рік становитиме:

$$\text{В} = \text{В1} + \text{В2} + \text{В3} = 30{,}500 + 91{,}000 + 245{,}000 = 284{,}600 \text{ грн}$$

Таким чином, можливий збиток від атаки значно перевищує витрати на впровадження заходів безпеки, що підтверджує економічну доцільність впровадження програмного забезпечення для захисту вебресурсів від SQL-атак.

3.5 Загальний ефект від впровадження системи інформаційної безпеки

Загальний ефект від впровадження системи інформаційної безпеки визначається з урахуванням ризиків порушення інформаційної безпеки і становить:

$$\text{Е} = \sum \text{Вi} \cdot R\text{i} - \text{С}$$

де:

- B – загальний збиток від атаки на вузол або сегмент корпоративної мережі, тис. грн;
- R – очікувана імовірність атаки на вузол або сегмент корпоративної мережі, частки одиниці;
- C – щорічні витрати на експлуатацію системи інформаційної безпеки, тис. грн.

$$E = (50{,}021.9 \times 0.144 + 150{,}065.6 \times 0.156 + 400{,}850 \times 0.316) - 5{,}498$$
$$= 151{,}783.95 \text{ грн}$$

3.6 Визначення та аналіз показників економічної ефективності системи інформаційної безпеки

$$ROSI = \frac{E}{K}$$

де:

- E – загальний ефект від впровадження системи інформаційної безпеки, тис. грн;
- K – капітальні інвестиції за варіантами, що забезпечили цей ефект, тис. грн.

$$ROSI = \frac{151{,}783.95}{57.061} = 2.66$$

3.7 Висновки

Виходячи з отриманих розрахунків та загального огляду, можна зазначити, що впровадження програмного забезпечення для захисту вебресурсів було економічно доцільним. Загальний економічний ефект від впровадження системи інформаційної безпеки, розрахований в цьому розділі, дорівнює додатному значенню (151,783.95 грн), що підтверджує, що корисний економічний ефект

перевищує витрати. Також важливим показником є коефіцієнт повернення інвестицій. У нашому випадку він більше одиниці, що демонструє прибутковість вкладень.

ВИСНОВКИ

Захист вебресурсів від SQL-загроз є важливою і актуальною проблемою в у сучасному цифровому світі. Вебресурси стали не лише невід'ємною частиною нашого щоденного життя, але й ключовим елементом діяльності практично будь-якої компанії чи установи. Вони забезпечують доступ до інформації, надають можливість здійснювати різноманітні послуги, обмінюватися даними та взаємодіяти з користувачами. Однак поширене використання також робить вебресурси мішенню для зловмисників, які прагнуть отримати несанкціонований доступ до даних або завдати шкоду. Відповідно, захист вебресурсів від різноманітних загроз, зокрема від SQL-ін'єкцій, стає важливим завданням для будь-якої компанії або організації.

У першому розділі кваліфікаційної роботи було розглянуто проблематику сучасної кібербезпеки та основні тенденції. Також, було детально проаналізовано питання SQL-загроз, наведено їхню класифікацію.

Другий розділ був присвячений аналізу засобів розробки, зокрема мовам програмування, фреймворкам та базам даних.

У третьому розділі даної роботи було розроблено систему, що демонструє основні можливості SQL-ін’єкцій та засоби їхнього знешкодження.

SQL-ін'єкції є однією з найпоширеніших та найнебезпечніших атак на вебдодатки. Вони дозволяють зловмисникам виконувати SQL-запити на базі даних вебсайту без будь-яких обмежень, що може призвести до витоку конфіденційної інформації, порушення цілісності даних або навіть до повного контролю над системою. У зв'язку з цим розробка та впровадження ефективних методів захисту від цих загроз стає нагальною потребою. Велика кількість інцидентів, пов'язаних зі зломом вебсайтів через SQL-ін'єкції, підкреслює важливість розуміння цього типу атак та вжиття відповідних заходів забезпечення безпеки.

Дослідження в галузі захисту вебресурсів від SQL-загроз відіграє ключову роль у забезпеченні безпеки та надійності вебсервісів. Воно дозволяє виявляти потенційні вразливості та розробляти ефективні стратегії захисту, що сприяє підвищенню рівня безпеки в інтернеті. Такі дослідження допомагають уникнути фінансових втрат, зберегти довіру користувачів та підтримати репутацію компанії. Тому важливість цієї теми не може бути переоцінена, оскільки безпека вебресурсів є основою успішного функціонування сучасного цифрового світу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Даніель, Шац; Джулі, Стіна. «До більш репрезентативного визначення кібербезпеки». Журнал цифрової криміналістики, безпеки та права.

2. Роуз, Маргарет. «Визначення соціальної інженерії». Технічна ціль.

3. Шац, Даніель; Башруш, Рабіх; Уолл, Джулі. «До більш репрезентативного визначення кібербезпеки». Журнал цифрової криміналістики, безпеки та права.

4. Стівенс, Тім. «Глобальна кібербезпека: нові напрями в теорії та методах». Політика та управління.

5. Millman, Renee. «Нове поліморфне шкідливе програмне забезпечення обходить три чверті AV-сканерів». SC Magazine Великобританія.

6. Сканнелл, Кара. «Шахрайство з електронною поштою генерального директора коштує компаніям 2 мільярди доларів». Financial Times.

7. Марсель, Себастьян; Ніксон, Марк; Лі, Стен, ред. Посібник із біометричної боротьби зі спуфінгом: надійна біометрія під спуфінгом Напади(PDF). Лондон: Springer.

8. Опитування Gartner понад 2 000 головних інформаційних офіцерів виявляє необхідність для підприємств у 2022 році ухвалювати бізнес-композицію. STAMFORD, Конн. 18 жовтня 2021 року. Режим доступу: https://www.gartner.com/en/newsroom/press-releases/2021-10-18-gartner-survey-of-over-2000-cios-reveals-the-need-for-enterprises-to-embrace-business-composability-in-2022

9. Опитування Gartner понад 2 000 головних інформаційних офіцерів виявляє наявність для підприємств у 2022 році ухвалювати бізнес-композицію. СТАМФОРД, Конн. 18 жовтня 2021 року. Режим доступу: https://www.gartner.com/en/newsroom/press-releases/2021-10-18-gartner-survey-of-over-2000-cios-reveals-the-need-for-enterprises-to -embrace-business-composability-in-2022

10. Accenture, Стан кібербезпеки 2021, «Як вирівнювання безпеки та бізнес створює кіберстійкість». Режим доступу: https://www.accenture.com/_acnmedia/PDF-165/Accenture-State-Of-Cybersecurity-2021.pdf

11. Bodin, L. D., Gordon, L. A. та Loeb, M. P. Оцінка інвестицій в інформаційну безпеку за допомогою АНАЛІТИЧНОГО ПРОЦЕСУ ІЄРАРХІЇ. Повідомлення ACM, 48(2)

12. Arora, A., Hall, D., Piato, C. A., Ramsey, D. та Telang, R. (2004) Вимірювання ризик-орієнтованої вартості IT-рішень з інформаційної безпеки. IT Professional, 6(6)

13. Bojanc, R. та Jerman-Blažič, B. Економічний модельний підхід до управління ризиками інформаційної безпеки. International Journal of Information Management

14. NIST Спеціальна публікація 800-39 Управління ризиками інформаційної безпеки

15. NIST Спеціальна публікація 800-30 Посібник з проведення ризик-оцінки [Текст]

16. NIST Спеціальна публікація 800-53 Ревізія 3 Рекомендовані заходи безпеки для федеральних інформаційних систем і організацій [Текст]

17. Чи-Чун Ло, Ван-Цзя Чен. Гібридна процедура оцінки ризиків інформаційної безпеки, яка враховує взаємозалежність між контролем. Експертні системи з додатками

18. Sheng-Li Si, Xiao-Yue You, Hu-Chen Liu, Ping Zhang, Техніка DEMATEL: систематичний огляд сучасних досліджень методології та розробки, / Sheng-Li Si, Xiao-Yue You, Hu-Chen Liu, Ping Zhang // Математичні. проблеми в інженерії

19. Сааті Томас Л. Прийняття рішень в умовах залежності і зворотного зв'язку: Аналітичні мережі / Томас Сааті // М.: Видавництво.

20. Державна служба спеціального зв'язку та захисту інформації України. (2003). НД ТЗІ 2.5-010-03. Методика аналізу захищеності інформації від несанкціонованого доступу в інформаційно-телекомунікаційних системах. Київ.

ДОДАТОК А. Відомість матеріалів кваліфікаційної роботи

№	Формат	Найменування	Кількість листів	Примітка
1	А4	Реферат	2	
2	А4	Список умовних скорочень	1	
3	А4	Зміст	2	
4	А4	Вступ	1	
5	А4	1 Розділ	23	
6	А4	2 Розділ	39	
7	А4	3 Розділ	9	
8	А4	Висновки	2	
9	А4	Перелік посилань	3	
10	А4	Додаток А	1	
11	А4	Додаток Б	1	
12	А4	Додаток В	1	
13	А4	Додаток Г	1	

ДОДАТОК Б. Перелік документів на оптичному носії

Пояснювальна записка.docx

Пояснювальна записка.pdf

Презентація.pptx

ДОДАТОК В. Відгук керівника економічного розділу

Економічний розділ виконаний відповідно до вимог, які ставляться до кваліфікаційних робіт, та заслуговує на оцінку 85б. («добре»).

Керівник розділу _____ Дар`я ПІЛОВА

(підпис) (ім'я, прізвище)

ДОДАТОК Г. Відгук керівника кваліфікаційної роботи

ВІДГУК

на кваліфікаційну роботу студента групи 125-20-1

Бородкіна Владислава Григорійовича

На тему: Захист вебресурсів від SQL-загроз за допомогою програмного додатку на Python.

Пояснювальна записка складається з титульного аркуша, завдання, реферату, списку умовних скорочень, змісту, вступу, трьох розділів, висновків, переліку посилань та додатків, розташованих на 88 сторінках та містить 23 рисунка, 1 таблицю, 20 джерел та 4 додатка.

Студент показав достатній рівень володіння теоретичними положеннями з обраної теми, показав здатність формувати власну точку зору (теоретичну позицію).

Робота оформлена та написана грамотною мовою, відповідає вимогам положення про систему запобігання та виявлення плагіату у Національному технічному університеті «Дніпровська політехніка». Містить необхідний ілюстрований матеріал. Автор добре знає проблему, уміє формулювати наукові та практичні завдання і знаходить адекватні засоби для їх вирішення.

В цілому робота задовольняє усім вимогам і може бути допущена до захисту, а його автор заслуговує на оцінку «_____».

Керівник кваліфікаційної роботи проф. Кагадій Т.С.

Керівник спец. розділу ст. викл Начовний І.І.