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

ДОПУСТИТИ ДО ЗАХИСТУ

_____

_____Завідувач кафедри

_____ С.В. Казмірчук

«_____»_____ 2021р.

На правах рукопису УДК 004.056.5(004.7)

ДИПЛОМНА РОБОТА
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ«БАКАЛАВР»

Тема: Програмний модуль захисту від SQL атак

Виконавець: А.І.Сакович

Керівник: к.т.н. О.О.Висоцька

Нормоконтролер:к.т.н. О.О.Висоцька

Київ 2021 р.

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра: Комп'ютеризованих систем захисту інформації
Освітній ступінь:Бакалавр
Спеціальність: 125 «Кібербезпека»
Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ С.В. Казмірчук
«__» _____ 2021 р.

ЗАВДАННЯ
на виконання дипломної роботи
здобувача вищої освіти Саковича Артема Ігоровича

Тема: Програмний модуль захисту від SQL атак
затверджена наказом в.о. ректора від «26»квітня 2021 р. № 652/ст.
Термін виконання: з 10.05.2021р. по 20.06.2021р.
Вихідні дані: мова програмування Python; аналіз програмного забезпечення для захисту від SQL атак; дослідження потенційної реалізації альтернативної перевірки для введення користувачем спеціального обладнання замість програмного забезпечення; впровадження метода апаратного пошуку та заміни рядків з метою видалення одного з основних векторів атак.
Зміст пояснювальної записки: захист від SQL атак,виявлення нападів SQL-атак і впровадження захисту,реалізація системи програмного забезпечення для захисту від SQL-ін'єкцій.

КАЛЕНДАРНИЙ ПЛАН

виконання дипломної роботи

№ п/п	Етапи виконання дипломноїроботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	19.04.2021	*Виконано*
2.	Аналіз літературних джерел	20.04.2021	*Виконано*
3.	Обґрунтування вибору рішення	22.04.2021	*Виконано*
4.	Збір інформації	23.04.2021	*Виконано*
5.	Аналіз та дослідження основних принципів перевірки справжності документів з використанням QR-коду	02.05.2021	*Виконано*
6.	Розробка архітектури та загальної структури системи	12.05.2021	*Виконано*
7.	Програмна реалізація системи	20.05.2021	*Виконано*
8.	Апробація роботи на II науково-технічній конференції ITSEC-2012	22.05.2021	*Виконано*
9.	Оформлення презентації	30.05.2021	*Виконано*
10.	Оформлення і друк пояснювальної записки	31.05.2021	*Виконано*
11.	Перевірка на антиплагіат	01.06.2021	*Виконано*
12.	Отримання рецензій від рецензента	04.06.2021	*Виконано*

Здобувач вищої освіти А.І Сакович

(підпис, дата)

Керівник дипломної роботи О.О. Висоцька

(підпис, дата)

РЕФЕРАТ

Дипломна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків і має 49 сторінок основного тексту, 53 рисунка, 1 таблицю, 15 сторінок додатків. Список використаних джерел містить 26 найменувань і займає 2 сторінки. Загальний обсяг роботи 77 сторінок.

Метою цієї дипломної роботи є синтезування програмного забезпечення для захисту від SQL атак.

Об'єктом дослідження є процес захисту від SQL-ін'єкцій у веб-застосунках.

Методи дослідження. Метод TCP SYN(встановлення зв'язку,в якому стан двох учасників стає несинхронізований) метод TCP CONNECT(створення з'єднання з будь-яким портом),метод FTP(поширення ПО і доступу до віддалених хостів.).

Ключові слова: SQL-ін'єкції, MS SQL, mySQL, конфіденційність, авторизації.

Практична цінність полягає у розробленні програмного модулю,який забезпечує виявлення та захист від SQL-ін'єкцій за допомогою методу сканування TCP SYN.

У процесі розробки програмного забезпечення були сформовані функціональні вимоги, принцип роботи системи, визначено архітектуру програмного комплексу та методи роботи.Здійснено тестування програми на основі скану сайтів з інтернету.

ЗМІСТ

РЕФЕРАТ 5

ВСТУП 5

РОЗДІЛ 1. ЗАХИСТ ВІД SQL АТАК 7

1.1 Введення в SQL 7

1.1 Основні види SQL-ін’єкцій 8

1.2 Причини SQL-ін'єкції 15

1.3 Методи запобігання та захисту від MySQL-ін'єкцій 17

1.4 Області застосування і де використовується SQL 21

1.5 Огляд обробки SQL 26

РОЗДІЛ 2. ВИЯВЛЕННЯ НАПАДІВ SQL АТАК І ВПРОВАДЖЕННЯ ЗАХИСТУ 27

2.1 Процес вибору ключових слів 27

2.2 Аналіз захисту від ін'єкції SQL у програмному забезпеченні 33

2.3 Безпека Web-сайтів: SQL-ін'єкція 37

2.4 Порівняння сканерів вразливостей мережі 41

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ДЛЯ ВИЯВЛЕННЯ ТА ЗАХИСТУ ВІД SQL-АТАК 49

3.1 Принцип работи сканера 49

3.2 Алгоритм роботи сканера 50

3.3 Методи сканування 51

3.4 Тестування програми 54

ВИСНОВКИ 58

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 58

ДОДАТОК 61

ВСТУП

Актуальність. Глобальний розвиток спирається на розуміння та вдосконалення технологій. Як "кам'яний вік не закінчився через відсутність каменів, а нафтовий вік закінчиться задовго до того, як у світі закінчиться нафта".Сьогодні ми живемо в тому, що багато хто називає інформаційною епохою, і нам абсолютно не загрожує втрата інформації. Існує загальне уявлення про те, що ми перевантажені даними, що робить можливість зберігання, обробки, аналізу, споживання, захисту та дії на дані першочерговою проблемою. Для великомасштабних багатонаціональних організацій, таких як фінансова галузь чи галузь охорони здоров'я, ситуація стала дуже складною та складною. Тоді виникає питання, як ми обробляємо та зберігаємо цей великий обсяг даних, зберігаючи їх секретність, надійність та доступність.

У багатьох випадках великі дані зберігаються для ефективного доступу, створюючи зв'язки в даних, а потім зберігаючи їх у базу даних, яка потім називається реляційною базою даних. Доступ до цих баз даних здійснюється через веб-сайти та API (інтерфейс прикладних програм) за допомогою мови, яка називається структурованою мовою запитів (SQL). Багато з цих веб-сайтів вразливі для хакерів, які прагнуть здійснити атаки, оскільки бэкдори залишаються відкритими через погану практику кодування, яка не відповідає хорошим стандартам безпеки. Хакери розкривають полчища ботів, щоб шукати сайти, які мають ці слабкі сторони кодування.У деяких випадках цільові веб-сайти можуть бути націлені через цінність даних, до яких хакер сподівається отримати доступ. В обох випадках довіра дизайнерам до простого написання хорошого коду виявилась недостатньою реакцією, і досконалої системи захисту не існує. Надзвичайно важливим було вирішити цю проблему та протидіяти творчим крокам зловмисника для злому систем.

Метою цієї дипломної роботи є розробка програмного модуля захисту від SQL атак,в якому завдяки обраним на основі проведеного аналізу методів захисту,забезпечується необхідний рівень захисту. Для посягнення поставленої мети необхідно розв'язати наступні задачі:

- Проаналізувати існуючі види SQL атак,причини їх виникнення та методи захисту від них.
- Розробити програмний модуль захисту від SQL атак.
- Провести тестування розробленого программного модуля виявлення та захисту від SQL атак.

Об’єктом дослідження є процеси захисту від SQL атак, зокрема у веб-застосунках.

Предметом дослідження є методи і системи захисту від SQL-атак.

Методи дослідження.Метод TCP SYN(встановлення зв’язку,в якому стан двох учасників стає несинхронізований) метод TCP CONNECT(створення з’єднання з будь-яким портом),метод FTP(поширення ПО і доступу до віддалених хостів.).

Практична цінність. Розроблено програмний модуль,який забезпечує виявлення та захист від SQL-ін’єкцій за допомогою методу сканування TCP SYN.

РОЗДІЛ 1. ЗАХИСТ ВІД SQL АТАК

1.1 Введення в SQL

SQL - простими словами, це мова програмування структурованих запитів (SQL, Structured Query Language), який використовується в якості ефективного способу збереження даних, пошуку їх частин, поновлення, вилучення з бази і видалення.

Головний інструмент оптимізації та обслуговування бази даних - ось, для чого потрібен SQL, хоча він і не обмежений цими цілями. Можливості обробки охоплюють команди визначення уявлень, вказівки прав доступу, схем відносин (в тому числі, їх видалення і зміни), взаємодія з іншими мовами програмування, перевірку цілісності, завдання початку і завершення транзакцій.

1.1 Основні види SQL-ін’єкцій

Існує 5 основних видів SQL ін'єкцій:

Класична (In-Band або Union-based). Найнебезпечніша і рідко зустрічається сьогодні атака. Дозволяє відразу отримувати будь-які дані з бази.

Error-based. Дозволяє отримувати інформацію про базу, таблицях і даних на основі виведеного тексту помилки СУБД.

Boolean-based. Замість отримання всіх даних, атакуючий може поштучно їх перебирати, орієнтуючись на просту відповідь типу true / false.

Time-based. Схожа на попередню атаку принципом перебору, маніпулюючи часом відгуку бази.

Out-of-Band. Дуже рідкісні і специфічні типи атак, засновані на індивідуальні особливості баз даних.

Самий простий приклад критично уразливого для SQLi коду виглядає так

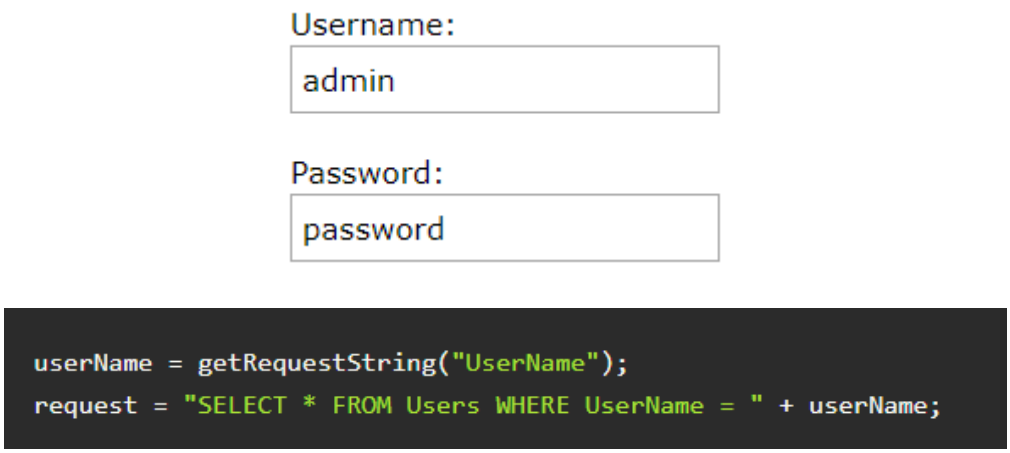

Рисунок 1.Приклад критично уразливого коду

Коментування

Використання однорядкових коментарів дозволяє ігнорувати частину запиту, що йде після ін'єкції. Наприклад, введення в уразливе поле Username запиту admin '- дозволить зайти на ресурс під адміністратором, тому що перевірка пароля буде закоментована. Звичайно, зараз такий тип вразливості зустрічається дуже рідко, але пам'ятати про неї варто.

```
SELECT * FROM members WHERE username = 'admin'--' AND password = 'password'
```

Рисунок 2.Приклад використаня однорядкових коментарів

Багаторядкові коментарі можуть впоратися з перевіркою або визначити тип бази даних. Наприклад, подібні запити обійдуть примітивний текстовий аналіз:

```
DROP/*some comment*/sampletable
DR/**/OP/*random comment to cheat*/sampletable
```

Рисунок 3.Приклад використаня багаторядкових коментарів

А деякі особливі коментарі дозволять визначити тип бази даних з метою подальшої експлуатації вразливостей:

```
/*!Если поместить код в такой комментарий - он будет исполнен только в MYSQL.
Можно даже ограничить минимальную версию*/

такой запрос вернёт ошибку деления на ноль, если MYSQL сервер выше указанной
SELECT /*!!50100 1/0, */ 1 FROM tablename
```

Рисунок 4.Особливі коментарі

Маніпуляції з рядками

Існує ряд більш просунутих способів обходити чорні списки. Наприклад, проти фільтра лапок можна використовувати конкатенацію рядків:

```
#SQL Server
SELECT login + '-' + password FROM members
#MySQL
SELECT CONCAT(login, password) FROM members
```

Рисунок 5. Конкатенація

В MySQL для обходу складних паттернів можна представляти рядки в шістнадцятковій вигляді, за допомогою функції HEX () або вводити їх посимвольний:

```
//0x633A5C626F6F742E696E69 == c:\boot.ini
SELECT CONCAT('0x','633A5C626F6F742E696E69'))

SELECT CONCAT(CHAR(75),CHAR(76),CHAR(77))
```

Рисунок 6. Рядки в шістнадцятковому вигляді

Обхід аутентифікації

Є стандартний словник, що містить в собі основні запити, для обходу вразливою форми аутентифікації. Вперше його опублікували років 10 назад і регулярно доповнюють.

```
' or 1=1
' or 1=1--
' or 1=1#
' or 1=1/*
admin' --
admin' #
admin'/*
admin' or '1'='1
admin' or '1'='1'--
admin' or '1'='1'#
admin' or '1'='1'/*
admin'or 1=1 or ''='
admin' or 1=1
admin' or 1=1--
admin' or 1=1#
```

Рисунок 7. Стандартний словник

UNION Injection

UNION це SQL-команда, що дозволяє вертикально комбінувати дані з різних таблиць в одну. Це одна з найпопулярніших і небезпечних класичних ін'єкцій.

Припустимо, на сайті є список товарів з вразливою рядком пошуку. Тоді, підібравши правильну кількість колонок і визначивши їх назву, через UNION можна вивести практично будь-які дані.

```
SELECT name, price FROM products UNION ALL SELECT name, pass FROM members

#Такой запрос позволит получить данные о таблицах и найти таблицу пользователей
UNION(SELECT TABLE_NAME, TABLE_SCHEMA FROM information_schema.tables)
```

Рисунок 8. Вивід даних через UNION

Послідовні запити

Якщо цільовий сервіс працює на SQLServer і ASP / PHP, або на PostgreSQL і PHP, можна використовувати простий знак ';' для послідовного виклику шкідливих запитів:

```
#Удаление таблицы
SELECT * FROM products WHERE productName = ""; DROP users--
#Выключение SQL Server
SELECT * FROM products WHERE productName = ""; shutdown -
```

Рисунок 9.Виклик шкідливих запитів

Можливі шкоди

Конкретних прикладів і нюансів досить багато,так що не будемо перераховувати всі. Головне, потрібно пам’ятати, що комбінуючи ці прийоми і різні специфічні функції, атакуючий може отримати повний доступ до бази і навіть командному рядку.

Error-Based

Щоб побороти цей тип атак, досить заборонити висновок помилок. Проте, можемо розібрати на прикладі, чимнам може загрожувати ігнорування цього заходу.

Послідовне виконання наступних запитів до SQL Server, дозволить визначити в тексті помилки назви стовпців:

```
' HAVING 1=1 --
' GROUP BY table.columnfromerror1 HAVING 1=1 --
' GROUP BY table.columnfromerror1, columnfromerror2 HAVING 1=1 --
.....
' GROUP BY table.columnfromerror1, columnfromerror2, columnfromerror(n) HAVI
Если ошибки перестали появляться, значит столбцы закончились
```

Рисунок 10.Визначення помилок в тексті,через запити

Сліпі ін'єкції

У більш-менш добре зробленому додатку атакуючий не побачить ні помилок, ні результату UNION-атаки. Тут приходить черга діяти наосліп.

Умовні вирази

Атаки з використанням IF і WHERE - основа сліпого методу. Вони є однією з причин, чому використовувані нами оператори повинні бути закодовані в програмі, а не генеруватися аби як. Синтаксис для різних баз буде відрізнятися:

```
#MySQL
IF(condition,true-part,false-part)
#SQL Server
IF condition true-part ELSE false-part
#Oracle
BEGIN
IF condition THEN true-part; ELSE false-part; END IF; END;
#PostgreSQL
SELECT CASE WHEN condition THEN true-part ELSE false-part END;
```

Рисунок 11. Синтаксис

Boolean-based

Якщо атакуючий все ж може отримати інформацію про наявність чи відсутність помилки з HTTP-статусу, в сервісі існує вразливість до звичайної сліпої атаки. Розглянемо запит, який дозволить нам за допомогою алгоритму бінарного пошуку посимвольно визначити назву першої таблиці і в подальшому всіх даних:

```
TRUE : SELECT ID, Username, Email FROM [User]WHERE ID = 1 AND
ISNULL(ASCII(SUBSTRING((SELECT TOP 1 name FROM sysObjects WHERE xtYpe=0x55 AND
name NOT IN(SELECT TOP 0 name FROM sysObjects WHERE xtYpe=0x55)),1,1)),0)>78--
#Этот запрос говорит нам, что ASCII-значение первого символа больше 78
#дальнейший перебор определит точное значение
```

Рисунок 12. Запит,для визначення назви першої таблиці

Time-Based

Якщо атакуючий не спостерігає жодних відмінностей у відповідях сервера, залишається повністю сліпа атака. Прикладом буде використання функцій SLEEP або WAIT FOR DELAY:

```
SELECT * FROM products WHERE id=1; WAIT FOR DELAY '00:00:15'
```

Рисунок 13. Використання функцій WAIT FOR DELAY

Звичайно, реальні приклади будуть виглядати приблизно як boolean-based, тільки true і false атакуючий буде відрізняти за часом відгуку. Недоліки такого методу очевидні. Якщо вибрати дуже маленьку затримку, буде сильний вплив сторонніх чинників типу пінга. Якщо занадто велику - атака займе дуже багато часу і її, швидше за все, зупинять.

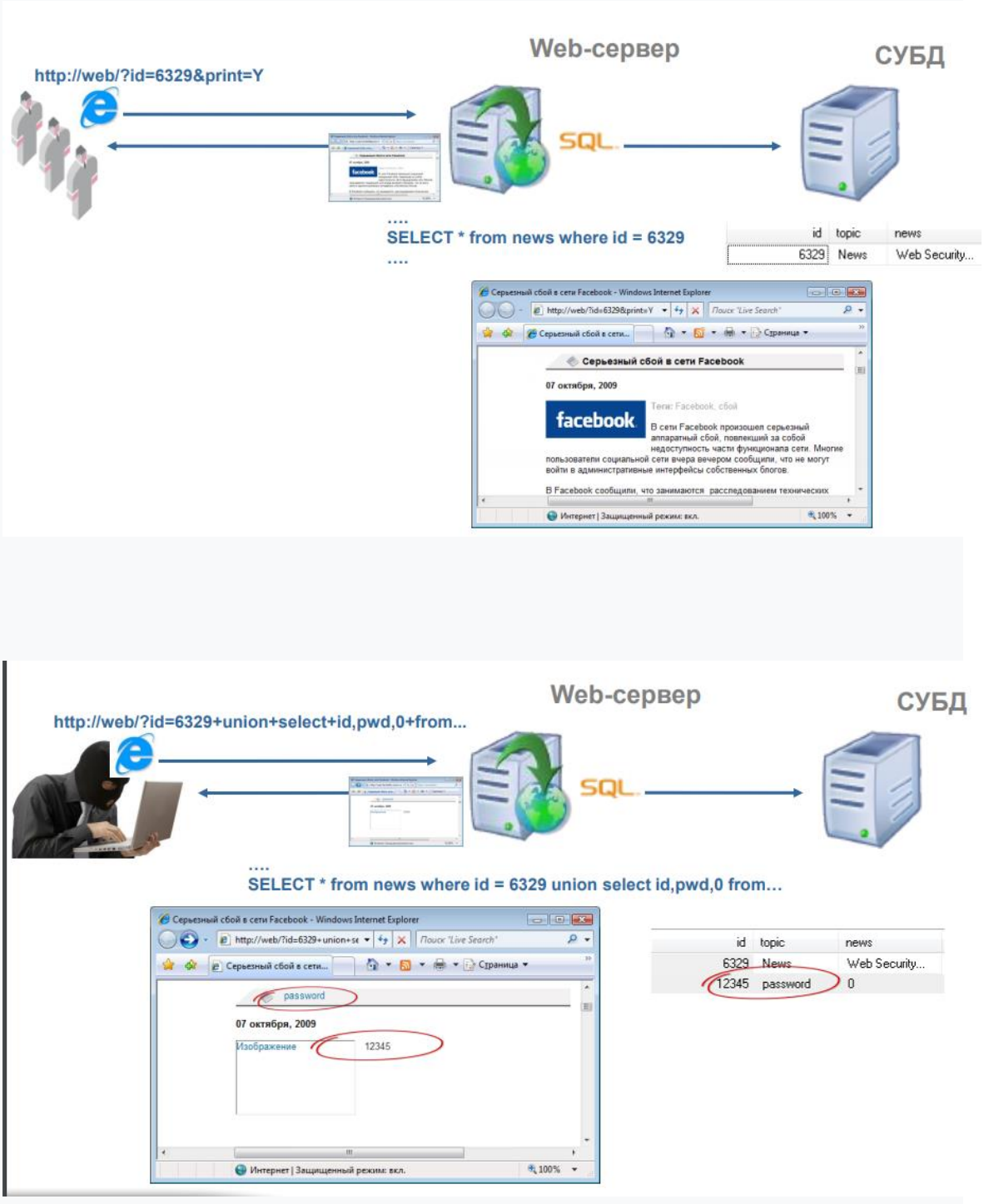

Рисунок 14. Наочний приклад впровадження операторів SQL

1.2 Причини SQL-ін'єкції

SQL-ін'єкція може виникнути в наступних випадках:

- Відсутність фільтрації
- Неправильна обробка типів
- Уразливості в базі даних сервера
- Умовні помилки

1. **Відсутність фільтрації**

Уявімо, що є модуль, який запитує адресу електронної пошти користувача, щоб відправити йому тимчасовий пароль до пошти, коли він забуває свій пароль. В цьому випадку, SQL-запит виглядатиме, приблизно, так:

```
<?php

$sqlStatement = "SELECT * FROM users WHERE username = '" + $username + "'

?>
```

Рисунок 16. Запит,який запитує адресу ел.пошти користувача

Але хакер може змінити цей запит, якщо встановить значення змінної $ email, додавши до адреси електронної пошти видалення таблиці users

```
user@hostname.com'; DROP TABLE users; SELECT * FROM customers WHERE name
LIKE '%
```

Рисунок 17. Зміненний запит

В результаті кінцевий запит буде виглядати так:

```
<?php

$sqlStatement = "SELECT * FROM users WHERE username = 'james' AND email =

?>
```

Рисунок 18. Кінцевий запит

2. **Неправильна обробка типів**

Часто ми знаємо тип даних, які хочемо отримати. Наприклад, вік клієнтів є числом, стать користувача (чоловік / дружин) рядком.

А що, якщо хтось введе $ ageValue, як:

```
20; DROP TABLE users
```

Рисунок 19. Неправильна обротка типів

В результаті SQL запит придбає такий вигляд:

```
<?php

$sqlStatement = "SELECT * FROM customers WHERE age = 20; DROP TABLE users,

?>
```

Рисунок 20. Результат

3. **Уразливості в базі даних сервера**

Хоча багато хто думає, що можуть уникнути SQL-ін'єкцій, просто використовуючи mysql_real_escape_string (), але вони не праві, на жаль. Вбудовані функції для роботи з базами даних надаються разом з мовним пакетом, а в деяких попередніх версіях MySQL є вразливість в обробці мультибайтного коду.

4. **Умовні помилки**

Використовуючи SQL-ін'єкцію користувач може легко обійти вхід в систему. Наведу приклад

```
<?php

$sqlStatement = "SELECT * FROM users WHERE username = 'james' AND passwor

?>
```

Рисунок 21. Запит,який дає обхід входу в систему

Значення 1 = 1 завжди істинно, таким чином, значення пароля вже не матиме значення і зловмисник зможе увійти в систему.

1.3 Методи запобігання та захисту від MySQL-ін'єкцій

- Використання параметризованих запитів
- Використання збережених процедур
- Застосування регулярних виразів
- Використання функцій блокування
- Відключення повідомлень про помилки
- Створення менш привілейованого користувача
- Обмеження максимального значення

1. **Використання параметризованих запитів**

Замість того, щоб підставляти значення SQL-заяв безпосередньо, підставляйте їх параметризовані значення, в такий спосіб:

```
<?php

$db_connection = new mysqli("localhost", "user", "pass", "db");
$statement = $db_connection->prepare("SELECT * FROM customers WHERE id =
$statement->bind_param("i", $id);
$statement->execute();

?>
```

Рисунок 22. Параметризовані значення

"I" - тільки цілі числа (int)

"d" - числа з плаваючою комою (double)

"s" - рядки (string)

"b" - вирушає в пакетах (blob)

2. **Використання збережених процедур**

Використання збережених процедур, теж може допомогти знизити ризик виникнення атаки. Використання процедур показано на наступному прикладі:

```
<?php

$sqlStatement = "
        CREATE PROCEDURE HUGEORDER
        (
                id INT ,
                quantity INT,
                price DECIMAL(6,2)
        )
        BEGIN
                DECLARE discount_percent DECIMAL(6,2);
                DECLARE discounted_price DECIMAL(6,2);
                SET discount_percent  =  10;
                SET discounted_price = price - discount_percent/100*price
                IF quantity > 500 THEN
                        SET discounted_price = discounted_price - 0.25 *
                END  IF;
                UPDATE fashion_products
                SET product_price = discounted_price WHERE product_id = i
                Select * from fashion_products;
        END;
";

?>
```

Рисунок 23. Використання процедур зниження ризику атаки

3. **Застосування регулярних виразів**

Регулярні вирази використовуються для того, щоб привести вхідні дані до одного шаблону. Наприклад, тут ми перевіряємо email клієнта на валідність і відкидаємо можливість для SQL-ін'єкцій.

```
<?php

if(!preg_match("/^[0-9a-z\_\.\-]+@([\-a-z0-9]+\.)+[a-z]{2,}$/i", $email))
{
        echo 'INVALID Email Address!';
        return;
}

?>
```

Рисунок 24. Перевірка на валідність

Також ми можемо користуватися вбудованими функціями PHP is_array (), is_bool (), is_double (), is_float (), is_int (), is_integer () та іншими, для перевірки даних користувача

4. **Використання функції блокування**

Використовуйте функцію mysql_real_escape_string () для обробки зовнішніх даних. наприклад:

```
<?php

$username = mysql_real_escape_string($username, $ dbLink);

?>
```

Рисунок 25. Функція обробки зовнішніх даних

Це дуже потужна вбудована функція PHP, здатна запобігти SQL-ін'єкції в більшості випадків. Ми можемо спробувати впровадити SQL-код після використання mysql_real_escape_string () і тестувати на уразливості. Ця функція відкидає безліч розумних методів атак, які використовуються зловмисниками.

5. **Відключення повідомлень про помилки**

Перш за все потрібно уникати вбудованої MySQL функції mysql_error (). Розумний нападник може вгадати деякі параметри бази даних з повідомлення про помилку, а іноді і побачити параметри з'єднання. Можемо використовувати mysql_error () тільки на стадії розробки. Але прибирати її, коли запускаємо сайт на сервері.

Також відключити звіти про помилки в PHP. Це робиться одним рядком***:***

```
<?php

// Отключить вывод ошибок
error_reporting(0);

?>
```

Рисунок 26. Відключення звітів про помилки

А краще створити власне повідомлення про помилку.

```
<?php

if(!mysql_query($statement))
{
        echo 'Извините, но сервер не доступен!';
}

?>
```

Рисунок 27. Створення власного повідомлення про помилку

В результаті, користувач не дізнається з повідомлення про помилку ніякої важливої інформації, такої як, ім'я бази даних, ім'я таблиці, ім'я користувача та інших. Тим самим ми ускладнюємо хакеру можливість дізнатися структуру SQL-запиту, використовуючи різні ін'єкції.

6. **Створення менш привілейованого користувача БД**

У більшості випадків, ви помітите, що відвідувачам не потрібно видаляти або оновлювати інформацію. Уявімо інтернет-магазин. Користувач може зробити запит (SELECT) або залишити замовлення (INSERT).

Таким чином, краще створити кілька різних користувачів. Для адміністратора надати всі привілеї, а для звичайного користувача обмежені. Приклад, з'єднання для різних користувачів:

```
<?php

$visitorDbLink = mysql_connect('host', 'general_user', 'general_user_pass
$adminDbLink = mysql_connect('host', 'admin_user', 'admin_pass');

?>
```

Рисунок 28. З'єднання для різних користувачів

Тепер можна використовувати $ visitorDbLink для регулювання доступу до бази даних для відвідувачів, і використовувати $ adminDbLink для доступу в якості адміністратора.

7. Встановлення обмеження на максимальне значення

Якщо ім'я користувача не може бути більше 10 символів, спробуємо використовувати "maxlenght" властивість:

```
<input name="username" type="text" id="username" maxlength="10" />
```

Рисунок 29. Властивість maxlenght

Однак, це не зовсім корисно, так як користувач може завантажити HTML-форму собі на комп'ютер і створити свою, більш зручну форму, яка буде оброблятися цим же скриптом. Тоді слід дублювати це обмеження на сервері:

```
<?php

$name = substr($_POST['name'],0,10);

?>
```

Рисунок 30. Скрипт для обробки форм

1.4 Області застосування і де використовується SQL

SQL використовується в:

SQL DDL

В якості мови визначення даних (DDL) він дає можливість незалежно створювати базу даних, визначати її структуру, використовувати, а потім сбрасивать по завершенню маніпуляцій.

SQL DML

В якості мови управління даними (DML) - для підтримки вже існуючих баз даних на ефективному з точки зору трудовитрат і продуктивності мовою введення, зміни та вилучення даних щодо бази даних.

SQL DCL

Як мова контролю даних (DCL), коли потрібно захистити свою базу даних від пошкодження і неправильного використання.

SQL клієнт/сервер

Відкривають єдину систему входу (SSO) з перевіркою достовірності користувача в декількох веб-додатках в рамках єдиного сеансу

SQL трирівневої архітектури

Гарантує захист інформаційної складової від несанкціонованого використання і копіювання в цифровому вигляді.

Майже всі реляційні бази даних використовують SQL. Деякі з них навіть включають абревіатуру мови в своїй назві: Microsoft SQL Server, MySQL, PostgreSQL, Non Stop SQL, SQLite. Але є і ті, хто іменується незалежно, як Oracle, DB / 2, Ingres. Є ще «NoSQL» - це збірний термін, який відносятьдо всіх нереляційних баз даних без SQL (або, коли це не єдина мова запитів).

Який вплив успішної атаки SQL-ін'єкцією?

Успішна атака з використанням SQL-ін'єкцій може призвести до несанкціонованого доступу до конфіденційних даних (наприклад, паролі, дані кредитної картки або інші особисті дані користувача). В останні роки багато гучних витоків даних стали результатом атак з використанням саме SQL-ін'єкцій, що призвело до збитків для репутації та штрафів з боку регулюючих органів. У деяких випадках зловмисник може отримати постійний канал в системи організації, що призведе до довгострокової уразливості, яка може залишатися непоміченою протягом тривалого періоду

Деякі популярні приклади впровадження SQL:

- отримання прихованих даних, де запит SQL змінюється з метою отримання додаткових результатів;
- підрив логіки додатка, де змінюється запит, щоб заважати логіці додатка;
- UNION атаки, де можна отримати дані з різних таблиць бази даних;
- вивчення бази даних, де витягується інформація про версії і структурі бази даних;
- сліпе впровадження SQL, коли результати запиту не повертаються у відповідях додатка.

На основі UNION атаки розберемо конкретний приклад.

Введення захисту на основі вразливостей

Багато вразливостей у веб-програмах використовують переваги можливостей хакера отримати доступ до цієї програми як користувач чи потенційний користувач. Таким чином, дані, введені користувачем, стають вектором в систему, який може бути використаний для здійснення бажаного використання. Кількість, різноманітність та складність цих веб-подвигів поступово збільшуються з тих пір, як вони виникли. Тут ми наводимо кілька прикладів таких експлойтів:

• SQL Injection: введення SQL - це підмножина атаки введення коду. Зловмисник вбудовує шматок коду в комп'ютерну програму за допомогою введення користувачем / введення даних. Виконання зараженої програми забезпечує зловмиснику неналежний доступ до комп'ютерної програми або програми. Хоча веб-додатки розробляються для отримання бажаного доступу до бази даних уповноваженими користувачами, зловмисник використовує їх на свою користь, щоб отримати доступ до конфіденційної інформації, що зберігається в базі даних.

• Міжсайтовий сценарій: міжсайтовий сценарій (XSS) - це атака введення коду, коли шкідливий код вводиться на веб-сайт і виконується у браузері. Зловмисник вставляє скрипт у браузер жертви, і коли відбувається доступ до скомпрометованого веб-сайту, сценарій виконується на комп’ютері жертви. За допомогою XSS зловмисник може віддалено керувати браузером об'єкта, що атакується. Це спосіб обійти концепцію стандартної операційної процедури (SOP).Наприклад, коли HTML-код генерується динамічно, а введення користувачем не дезінфікується і відображається на сторінці, зловмисник може вставити власний HTML-код на цю сторінку.

Міжсайтове підробляння запитів. Це атака, при якій зловмисник обманює користувача у виконанні дій, корисних зловмисникові під час доступу до веб-програми. Уразливість CSRF дозволяє зловмисникові змусити користувача виконувати дії, які зловмисник хоче виконати, коли користувач входить на веб-сайт. Наприклад, зловмисник може сформувати запит на переказ коштів на веб-сайт.

Вищезазначені вразливі місця - це всі випадки вразливостей на основі вхідних даних. Існує багато інших вразливих місць веб-атак, таких як помилки буфера, обхід шляху, введення коду тощо, але вони не обов'язково залежать від вкладених користувачем даних щодо використання жертви.

Відносна поширеність деяких з цих атак показано на рисунку (див.рис.31). Ці дані були отримані з блогу про хакерство та комп’ютерну безпеку від HACKING & TRICKS, написаного Ніравом Десаєм, у статті від 8

січня 2013 р. Очевидно, що SQL Injection включає значну частину (приблизно 25%) усіх веб-атак.

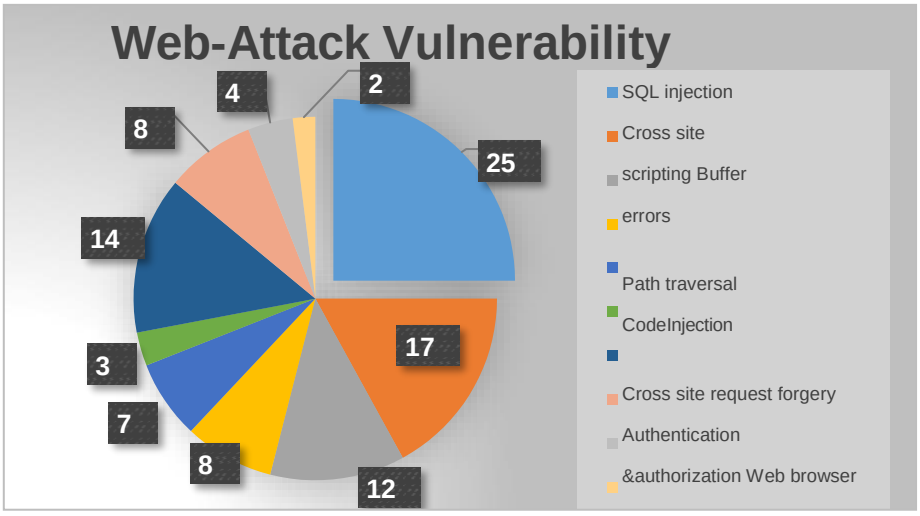

Рисунок 31.Звіт про вразливість веб-атак

Основними способами і техніками експулатації SQL-ін'єкцій,за допомогою яких зловмисник може отримати доступ до вмісту БД: техніки є універсальними, тому що застосовані для СУБД SQL, Oracle та MS SQLServer,широко використовуваних при створенні web-додатків,і працюють незалежно від мови розробки:PHP,ASP або Java.Тепер розглянемо техніки експлуатації уразливості:

- Union SQL-ін'єкція.
- Error-based SQL-ін'єкція.
- Blind SQL-ін'єкція.
- Time-based SQL-ін'єкція.
- Out-bound SQL-ін'єкція

1.5 Огляд обробки SQL

Обробка SQL відбувається у три основні етапи:

1. Розбір
2. Виконання
3. Отримання

Фаза синтаксичного аналізу: синтаксичний аналіз - це перший етап обробки запиту, який перевіряє синтаксис та семантичну достовірність оператора запиту шляхом маркерів операторів запиту . Для синтаксичного аналізу речення, щоб зробити його значущим запитом, речення потрібно проаналізувати лексично, синтаксично та семантично.

Етап виконання: Фаза виконання запиту починається з оптимізації запиту. Оптимізація полягає у формуванні відповідної послідовності, в якій триває логічний процес виконання запиту та відображається на фізичних ресурсах. Зокрема, оптимізація формує план виконання, який вибирається для підвищення продуктивності та зменшення використання серверних ресурсів. Коефіцієнт витрат для плану виконання залежить від фізичних ресурсів, таких як введення/виведення, процесор, пам'ять, кількість записів, які потрібно буде обробити. Фактичне виконання починається з оптимізованого плану та виконує відображені фізичні інструкції.

Фаза отримання: Це фаза, коли отримуються рядки результатів запиту та робляться доступними для кінцевих користувачів.

Фаза виконання визначає, які дані потрібно витягти для запиту, а фаза отримання отримує ці дані.

РОЗДІЛ 2.ВИЯВЛЕННЯ НАПАДІВ SQL АТАК І ВПРОВАДЖЕННЯ ЗАХИСТУ

2.1 Процес вибору ключових слів

Практикування ін’єкцій SQL на код Hack this site, bWAPP та код bashing.com/sql-demo, до якого можна було отримати доступ в Інтернеті, для вивчення ефекту прив’язки ключових слів присутня у атакуючих запитах. Подібно до прикладу, показаного на рисунку 32, кожне з потенційних ключових слів було переведено та перевірено, чи зможе воно викликати SQLIA.

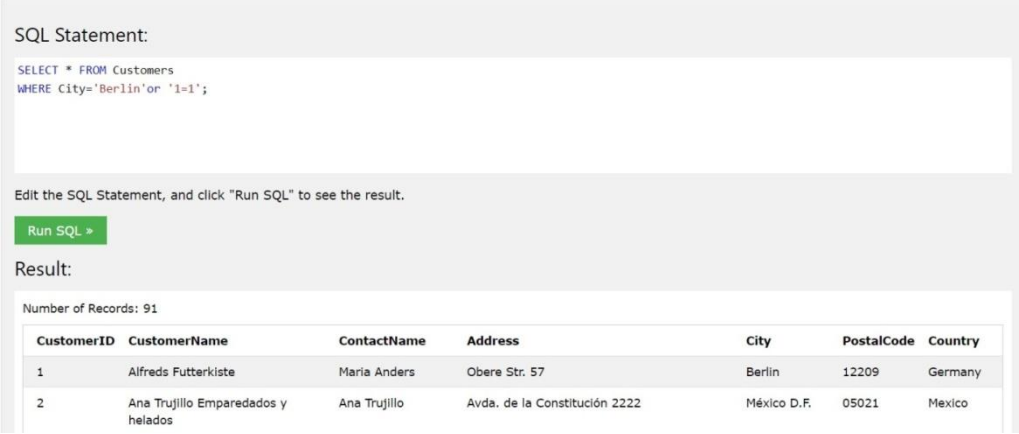

Рисунок 32. Тестування зарезервованих ключових слів SQL

Більш конкретно, база даних, яка використовується на рисунку 32 для розслідування атак введення SQL, мала загалом 91 запис для клієнтів, але лише 1 запис для клієнта, який належить до міста «Берлін». Однак завдяки атаці SQL Injection Attack, здійсненій шляхом запиту „Берлін або 1 = 1” у полі із запитом „місто”, ми змогли отримати дані всіх клієнтів у базі даних. Експериментуючи з багатьма такими запитами SQL,ми створили набір зарезервованих ключових слів, які слід дезінфікувати за введенням користувачем. У таблиці 1 ми описуємо ключові слова, які, як ми виявили, спричиняють атаки ін’єкцій SQL та вказуємо причину, через яку ці ключові слова створюють можливість для зловмисників. Зокрема, ця таблиця була сформована шляхом

складання списку з 250 ключових слів, знайдених у документації щодо зарезервованих ключових слів (Transact-SQL) на сайті Microsoft.com. Кожне ключове слово було змодельовано за допомогою веб-сайтів для аналізу SQLIA, як показано вище, для його спроможності здійснити атаку, коли вона присутня. (Зверніть увагу, що навіть якщо такі ключові слова були вставлені користувачем випадково, а не зловмисно, вони можуть спричинити серйозні проблеми. Таким чином, видалення їх із вводу користувача часто доречно, навіть якщо не передбачена атака. присутні для здійснення нападу не були включені до цього списку.)

and	and operator displays a record if all the conditions separated by and is true.
or	or operator displays a record if any of the conditions separated by or is true
between	between operator selects values within a given range.
not	not operator displays a record if the condition(s) is not true
insert	insert statement is used to insert new records in a table.
set	set operations allow the results of multiple queries to be combined into a single result set
delete	delete statement is used to delete existing records in a table.
like	like operator is used in a where clause to search for a specified pattern in a column.
in	in operator allows you to specify multiple values in a where clause,the in operator is a shorthand for multiple or conditions.
join	a join clause is used to combine rows from two or more tables
union	union operator is used to combine the result-set of two or more select statements.
into	the into creates a new table in the default filegroup and inserts the resulting rows from the query into it
"--"	"--" is used to comment the preceding statements in a query
create	create is used to create new instances of data in a database
drop	drop statement is used to drop an existing sql database.
alter	alter statement is used to add, delete, or modify columns in an existing table.
add	add statement is used for entering new enteries into existing table.
";"	";" is used as a statement terminator
all	all operator returns true if all of the subquery values meet the condition.
"""	""" is a character delimiter in sql statement
any	any operator returns true if any of the subquery values meet the condition.
exists	exists operator is used to test for the existence of any record in a subquery.
some	compares a scalar value with a single-column set of values
as	it creates copies of the table present in database table.
kill	kill is used to kill a process

Таблиця 1. Зарезервовані ключові слова та функції

Розробка схеми пошуку ключових слів

Для пошуку ключових слів у запиті, заданому користувачем, кожне слово порівнюється із збереженими ключовими словами, зазначеними в Таблиці 3. Для цієї конструкції ми вважаємо, що користувач дані були «попередньо проаналізовані», так що FPGA надає лише окремі слова. Крім того, ми припускаємо, що жодне зі слів, наданих FPGA, не перевищуватиме довжину 16 байт (16 символів). Слова довжиною менше 16 байт упаковуються з 0, щоб створити 16-байтове слово, яке буде порівнюватися з кожним із збережених ключових слів. В експериментах, проведених у цій роботі, дані, подані в машину, форматуються вручну для досягнення цих бажаних характеристик. У майбутній роботі буде розглянуто, чи можна і як синтаксичний розбір і відступ нулями ефективно включити в саму конструкцію FPGA.

На рисунку 33 наведена блок-схема високого рівня компонентів проекту "пошук і заміна рядка". Операції, які відбуваються в ланцюзі пошуку та заміни рядків, показані на рисунку 33

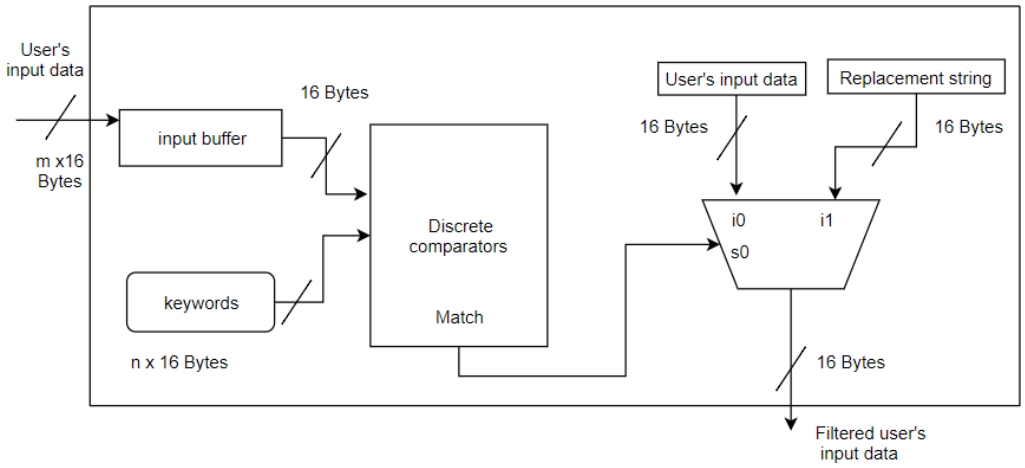

Рисунок 33.Блок-схема для пошуку рядків SQL

1. Машина приймає введені користувачем дані з кроком по одному слову по 16 байт за раз.

2. Блок порівняння перевіряє дані щодо 25 ключових слів на відповідність. Вхідні дані перевіряють кожен символ із відповідним символом ключового слова одночасно та паралельно порівнюють вхідні дані з усіма ключовими словами. Вихідний “сигнал

відповідності” створюється після порівняння даних користувача з усіма 25 зарезервованими ключовими словами.

3. Цей сигнал збігу використовується як тригер для замісної схеми, яка є мультиплексором або комутатором, який перемикається на вторинний вхід на збігу, і забезпечує нульовий рядок на виході мультиплексора замість вихідного ключового слова “заборонено” якщо знайдено відповідність ключового слова. Це призведе до видалення ключового слова із вводу користувача. В іншому випадку вихідні дані користувача просто передаються у вихідні дані.

4. Кінцевим продуктом цієї конструкції є фільтровані дані, у яких немає жодних шкідливих ключових слів.

Потрібно звернути увагу, що регістр ключових слів жорстко закодований набором з 25 ключових слів, а вхідні дані вводу даних користувача отримують нові попередньо проаналізовані дані з вхідного простору користувача кожного тактового циклу. На рисунку 34 показано технологічне відображення схеми пошуку та заміни рядків, створеної за допомогою веб-видання Altera Quartus

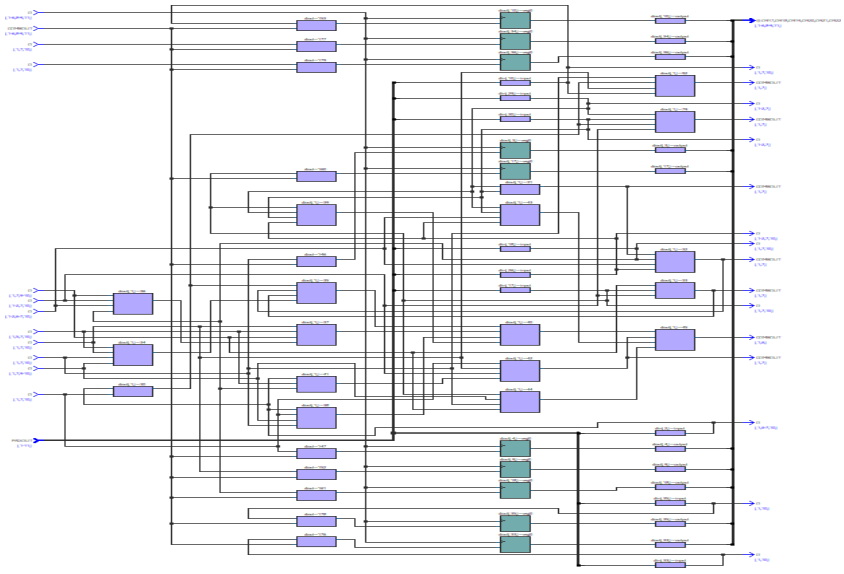

Рисунок 34. Зіставлення технологій для фіксованого збігу ключових слів

13.1. Нижче наведено більш детальний опис кожного з компонентів.

На рисунку 35 показано конструкцію дискретного компаратора для порівняння одного символу.

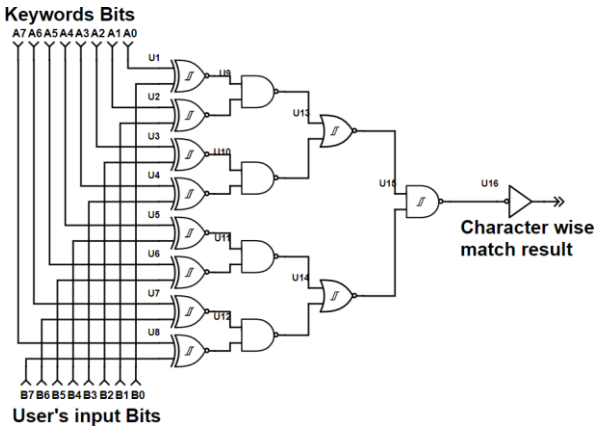

Рисунок 35. Дизайн для порівняння з одним символом

На цьому рисунку один символ вводу користувача та один символ ключового слова порівнюються побітово, і отриманий результат, який ми отримуємо, є відповіддю на стан відповідності одного байта даних. Коли 16 таких компараторів дають результат для збігу кожного символу, ми потім «I» отримуємо результати всіх результатів 1-символьних компараторів, щоб отримати остаточну відповідь для порівняння одного слова з 16 байт. Це показано на рисунок 36.

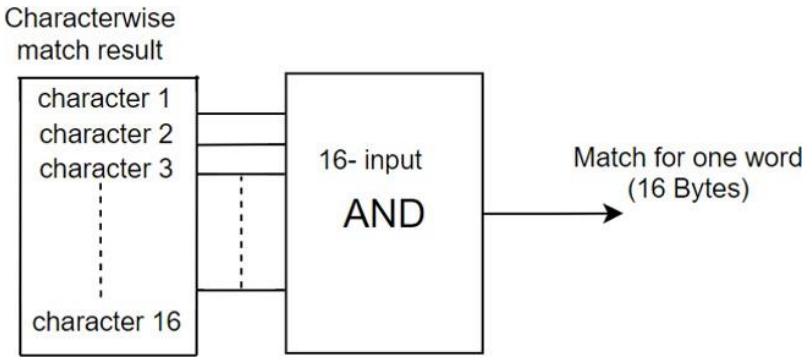

Рисунок 36. Дизайн для порівняння одним словом

Для порівняння всіх 25 ключових слів одночасно, схему на рисунку 36 тиражують 25 разів, а результати цих 25 компараторів ключових слів АБО складають, щоб отримати сигнал остаточного збігу, який використовується для заміни проблемного ключового слова символами NULL.

Оцінка реалізації

Зазначений Verilog дизайн був синтезований за допомогою програми Altera Quartus II Design Suite для програмованої логіки. Ми використовували синтезуване поведінкове кодування для проектування схеми узгодження рядків, де порівняння ключових слів здійснюється ланцюгом.

На рисунку 37 показано підмножину набору сигналів для одного перевірочного тесту запропонованої схеми. *Верхня форма* сигналу відповідає годиннику. *Друга форма*с игналу відповідає вводу користувача. *Третя форма* сигналу відповідає результату, що генерується після заміни зарезервованого ключового слова на нуль або вихідного вводу, якщо зарезервоване ключове слово не знайдено.

Четверта форма сигналу відповідає сигналу збігу, у, який стверджується, якщо виявлено збіг із зарезервованим ключовим словом

Рисунок 37. Моделювання часу на пристрої CycloneIV для відповідності ключових слів

2.2 Аналіз захисту від ін'єкції SQL у програмному забезпеченні

Щоб порівняти ефективність апаратного методу пошуку рядків із програмним підходом, можемо відтворити функціональність розробленого апаратного забезпечення мовою високого рівня C, порівнявши шаблон ключового слова та текст, по одному символу, алгоритм, реалізований на мові C.

На рисунку 38 представлена блок-схема реалізації відповідності рядків у програмному забезпеченні.

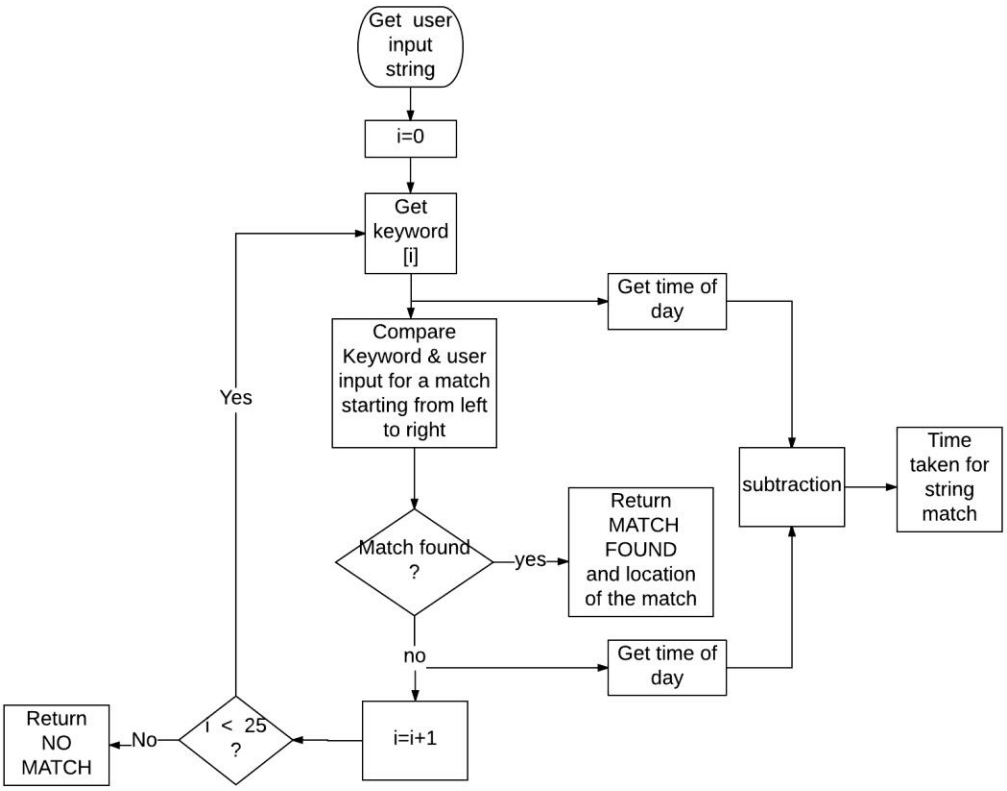

Рисунок 38. Блок-схема для програмної реалізації алгоритму пошуку рядків

Коли користувач надає введення, це порівнюється з ключовими словами. Оскільки у нас є 25 ключових слівдля порівняння, ми повторюємо процес порівняння 25 разів. Ми отримуємо час доби, коли починається збіг і коли отримується результат збігу. Віднімаючи час початку від часу закінчення, ми отримуємо загальний час, необхідний для пошуку ключових слів у введеному користувачем. Наприклад, коли

для введення даних було використано "або 1 = 1", час, який комп'ютер потребував для обробки цього рядка та пошуку ведучого "або", становив 742,3727 мкс.

Для обліку дисперсії в часі завдяки політиці планування пріоритетів ОС на сучасному процесорі проводили кожен експеримент 10 000 разів і усереднювали результат, щоб отримати реалістичну оцінку очікуваного часу. Цей експеримент був проведений на Intel i5-6500HQ, що працює на частоті 2,3 ГГц. Результат нашого порівняння для кількох різних рядків введення даних користувача показується на рисунок 39.

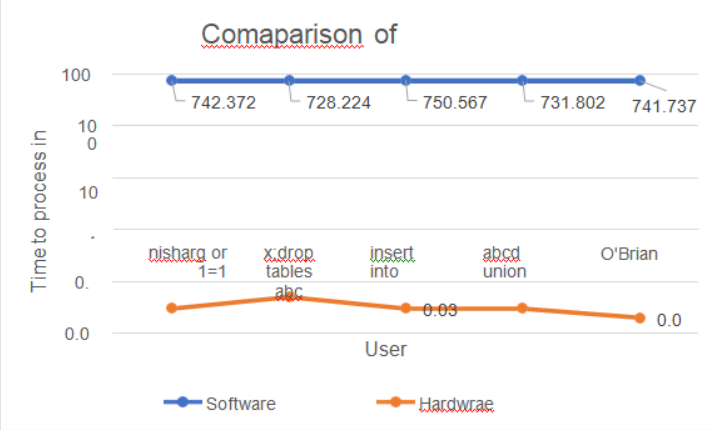

Рисунок 39. Порівняння розробленого підходу в програмному та апаратному забезпеченні

На цьому рисунку зображений результат експерименту, проведеного над програмною та апаратною реалізацією алгоритму пошуку рядків. Вісь у побудована з логарифмічною шкалою, щоб відповідати мірі продуктивності як програмного, так і апаратного забезпечення на одному графіку.

Обмеження запобігання SQLIA за допомогою методу пошуку та заміни рядків

Як видно з цього розділу, ми впровадили апарат для пошуку та заміни рядків в апаратному забезпеченні для виявлення та запобігання SQLIA. Хоча робота цього методу досить швидка, він має деякі обмеження.

Найголовніше, оскільки ми вилучаємо зарезервовані ключові слова SQL із вводу користувача, законні користувачі не можуть скористатися перевагами функцій зарезервованих ключових слів SQL для запиту бази даних. Це потенційно може бути вирішено за допомогою випадаючих меню або інших варіантів введення даних, які дозволять такі ключові слова лише в обмежених обставинах. Крім того, атаки виведення бази даних за допомогою логічно неправильних запитів запобігаються лише частково. Оскільки повідомлення про помилки використовуються для налагодження бази даних, їм не заважати виникнути, але будь-яка інформація, отримана про внутрішню базу даних, буде менш корисною для зловмисника, оскільки фактори прив'язки дезінфікуються. Поточна версія запропонованого рішення також не реалізує синтаксичний аналіз даних користувача на слова. В ідеалі цю частину роботи можна було б швидко виконати і в FPGA. Нарешті, це рішення налаштовано для конкретної системи баз даних, тобто MySQL, але її можна застосувати і до інших систем баз даних, налаштувавши ключові слова для певної системи баз даних.

Впровадження SQL кода(SQL-injection)

Метод атаки спрямовані на впровадження в їх посилають користувачем параметри SQL коду, в разі успіху зловмисник може змінити логіку запиту отримати, видалити або додати дані в базу. Існує безліч різновидів використання цієї атаки, але всі вони спрямовані на зміни призначеного для користувача запиту на свій синтаксично правильний запит.

Класичною захистом від SQL-ін'єкції є сувора фільтрація прийнятих параметрів в залежності від типу, а саме фільтрація лапок і символів "- / \ *". Більшість сучасних засобів розробки дозволяє на корені знищити цю уразливість шляхом використання ORM (об'єктне представлення

даних), зокрема компанія Microsoft пропонує своє рішення EntityFramework

2.3 Безпека Web-сайтів: SQL-ін'єкція

Веб-сайти звертаються до структурованої інформації, яка розміщена в базах даних. Користувачі взаємодіють з даними через форми для введення і відображення інформації. Наприклад, при аутентифікації вводиться логін і пароль, критерії перевіряються на сервері. У базі даних знаходиться інформація про користувачів і визначено рівень доступу. Іноді розробники сайту не враховують ситуацію, що в поле вводу інформації, можливо, поміщений текст, який порушить роботу сервера. Сервер обробляє отримані дані як задану команду. Цією уразливістю може скористатися зловмисник, наприклад, отримавши аутентифікаційні дані користувача з правами адміністратора, - така проблема отримала назву SQL-ін'єкції. Після введення логіна і пароля, дані з текстової форми передаються на сервер і підставляються в запит, поданий на- дме чином:

select*from Users where login='username'AND password='pass'

де username і pass - введений текст в поле даних на сайті.

Додаток авторизує користувача в тому випадку, якщо пара логін і пароль будуть визначені в базі користувачів. Але, замінивши username на фразу "admin '-", підсумковий запит підмінить початкову команду:

select*from Users where login='admin\'--'AND password='pass'.

І при наявності користувача з ім'ям admin сайт дозволить працювати під його обліковим записом без перевірки пароля. База проігнорує всю команду, яка стоїть після двох рисок "-", тому що в синтаксисі SQL так позначається коментар . Символ ";" відокремлює одну команду SQL від іншої, тому можливе виконання декількох команд.

Причини вразливості

Такі уразливості виникають з наступних причин:

- Динамічне побудова SQL-запитів. Найпростіший варіант для програміста - підставляти без додаткової фільтрації введені користувачем дані в готову конструкцію запиту, наприклад:

select*from users where login=’request.get Parameter(“name”)‘.

- Некоректна обробка винятків. Якщо ми спробуємо надіслати SQL-команди, що містить помилку, SQL-сервер відправить назад користувачу інформацію про неї. Зловмисник, орієнтуючись по цим повідомленням, вводить правильну команду.

- Некоректна обробка спеціальних символів. У базі даних MySQL, наприклад, коментар задається за допомогою символів "-" або "#" . Тому програма має обробляти такі символи.

- Некоректна обробка типів. При обробці символу апострофа (') закривається вразливість текстових даних. Однак в числових типах проблема залишається. У зловмисника є можливість додати команду, наприклад:

select*from money where sum=

1*union select1,user name,password from users.*

За допомогою union до вихідного запиту приєднається ще одна таблиця, в даному випадку з назвою users.

- Небезпечна конфігурація СУБД. Обліковий запис сервера додатка СУБД за замовчуванням має зайвими правами, через що база даних дозволить зловмиснику виконувати непередбачувані команди.

З точки зору програмування, задача захисту сервера від SQL-ін'єкцій вирішувана. Адміністратор повинен пам'ятати про цю уразливість і зробити правильні заходи щодо захисту.

SQL-ін'єкція є поширеною і руйнує вразливістю, тому важливо захистити додаток і сервер від даних атак. Перш за все необхідна

фільтрація даних, що надходять. Фільтрація відбувається на сервері, так як з програми може бути перехоплення і зміна інформації.

Спеціальні символи текстової інформації, наприклад, апостроф (') або коса риска (/), повинні доповнюватися косою рискою (/). Наприклад, якщо в поле username буде значення «Д'Артаньян», то в базу даних буде записано значення «Д / 'Артаньян», і помилок не виникне (рис. 40).

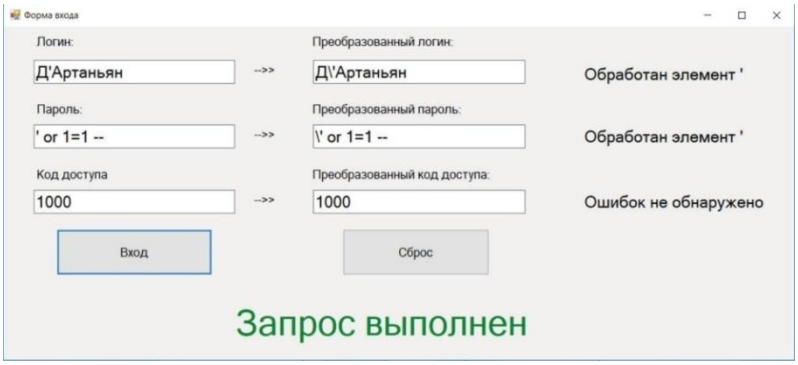

Рисунок 40. Перевірка спеціальних символів

Для перевірки чисельної інформації користуються перевіркою. Сервер поверне коректну помилку користувачеві, який в числове поле помістить текст (рис. 41).

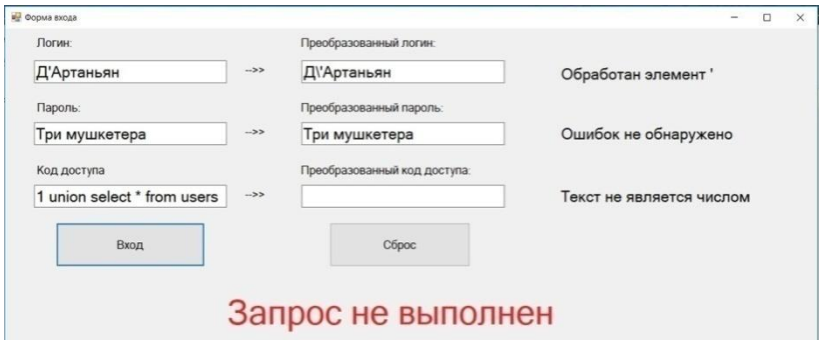

Рисунок 41. Невдала спроба доступу через числове поле.

Наводиться лістинг програми з перевірки коректності введення даних (рис. 42).

```
private bool CheckOfString(TextBox TB, TextBox TB2, Label L, string type)
{
    TB2.Text = "";
    bool flag = true;
    if (TB.Text.Length == 0) { L.Text = "Пустая строка, запрос не выполнен";
        return false; }
    if (type == "text") {
        for (int i = 0; i < TB.Text.Length; i++) {
            if (TB.Text[i] == '\'')
            { TB2.Text += "\\\'";
                flag = false;
                L.Text = "Обработан элемент \'"; }
            else if (TB.Text[i] == '\"') {
                TB2.Text += "\\\"";
                flag = false;
                L.Text = "Обработан элемент \""; }
            else if (TB.Text[i] == '\\'){
                TB2.Text += "\\\\";
                flag = false;
                L.Text = "Обработан элемент \\"; }
            else { TB2.Text += TB.Text[i]; }
        }
    } else if (type == "int")
    { try { Convert.ToInt32(TB.Text);
            TB2.Text = TB.Text; }
        catch (FormatException)
        { L.Text = "Текст не является числом";
            flag = false;
            return false; }
    } if (flag) L.Text = "Ошибок не обнаружено";
    return true; }
```

Рисунок 42. Лістинг коду перевірки і обробки введених даних

У додатку рекомендується обмежувати кількість символів для вводу інформації, а на сервері - перевіряти її. При переповненні поля запит відхиляється.

Рекомендується також відключити висновок помилок від SQL-сервера, тоді крадію доведеться «наосліп» вгадувати команди. Рекомендується вивести налаштування прав користувачів бази даних, з обмеженням прав облікного запису сервера.

2.4 Порівняння сканерів вразливостей мережі

GFI LanGuard

Одна з компаній-лідерів на ринку інформаційної безпеки - GFI Software. Компанія працює з 1992 року і за цей час заслужила репутацію надійної організації, що виробляє професійні продукти для вирішення широкого спектра IT-задач.

GFI LanGuard - програмний засіб, що забезпечує централізовану перевірку на уразливості в усій мережі. Гідність сканера в тому, що, крім виявлення відкритих портів, небезпечних налаштувань і забороненого до установки ПО, він перевіряє оновлення і патчі не тільки ОС (десктопних і мобільних, фізичних і віртуальних), але і встановленого ПО. Сканер уразливості перевіряє всі: від серверів до мережевого апаратного забезпечення, від віртуальних машин до смартфонів.

Закінчивши сканування, GFI LanGuard відправляє користувачеві повний звіт з характеристиками всіх вразливостей і інструкціями по їх ліквідації в ручному режимі. Доброзичливий інтерфейс дозволяє відразу ж закрити «провал» в захисті, виправити настройки, оновити ПЗ (включаючи установку відсутніх елементів), «знести» заборонені програми. Сканер мережевої безпеки можна налаштувати на повністю автономний режим роботи, в цьому випадку він буде запускатися по таймеру, а виправлення, дозволені адміністратором, будуть виконуватися автоматично.

Даний сканер вразливостей мережі має дуже важлива відмінність від безлічі конкурентів. Він може автоматично оновлювати застаріле програмне забезпечення та встановлювати оновлення і патчі на різні операційні системи.

Nessus

Проект був запущений ще в 1998 році, а в 2003 розробник Tenable Network Security зробив мережевий сканер безпеки комерційним. Але популярність продукту не впала. Згідно зі статистикою більше 17% користувачів вважають за краще саме Nessus. Регулярно оновлювана база вразливостей, простота в установці і використанні, високий рівень точності - його переваги перед конкурентами. А ключовою особливістю є використання плагінів. Тобто будь-який тест на

проникнення НЕ зашивається наглухо всередину програми, а оформляється у вигляді підключається плагіна. Аддони розподіляються на 42 різних типи: щоб провести пентест, можна активувати як окремі плагіни, так і всі плагіни певного типу - наприклад, для виконання всіх локальних перевірок на Ubuntu-системі. Цікавий момент - користувачі зможуть написати власні тести за допомогою спеціального скриптового мови.

Nessus - відмінний сканер вразливостей мереж. Але є у нього і два недоліки. Перший - при відключеній опції «safe checks» деякі тести на уразливості можуть привести до порушень в роботі сканованих систем. Другий - ціна. Річна ліцензія може обійтися в 42тис.грн.

Symantec Security Check

Безкоштовний сканер однойменного виробника. Основні функції - виявлення вірусів і троянів, інтернет-хробаків, шкідливих програм, пошук вразливостей в локальній мережі. Це онлайн-продукт, що складається з двох частин: Security Scan, яка перевіряє безпеку системи, і Virus Detection, яка виконує повну перевірку комп'ютера на віруси. Встановлюється швидко і просто, працює через браузер. На жаль, за підсумками багатьох тестів поступається більшості конкурентів, хоча всі свої основні функції виконує. Згідно з останніми відгуками, цей сканер мережі краще використовувати для додаткової перевірки.

XSpider

Компанія Positive Technologies працює на ринку більше 10-ти років і володіє одним з найбільших дослідницьких центрів безпеки. У цій корпорації теж є свій мережевий сканер - програма XSpider, яка, за заявою розробника, може виявити третину вразливостей завтрашнього дня. Ключовою особливістю цього сканера є можливість виявлення максимальної кількості «провалів» в мережі ще до того, як їх побачать хакери. При цьому сканер працює віддалено, не вимагаючи установки додаткового ПЗ. Відпрацювавши, сканер відправляє фахівця з безпеки повний звіт і поради щодо усунення «дірок».

Окремо відзначимо систему оновлення програмних модулів і бази вразливостей, функції одночасного сканування великої кількості робочих станцій і мережних вузлів, ведення повної історії перевірок, вбудовану документацію і механізм генерації деталізованих звітів.

QualysGuard

Багатофункціональний сканер вразливостей. Він надає великі звіти, які включають:

- оцінку рівня критичності вразливостей;
- оцінку часу, необхідного для їх усунення;
- перевірку ступеня їх впливу на бізнес;
- аналіз тенденцій в області проблем безпеки.

Фірма-розробник продукту, Qualys, Inc., широко відома в світовому середовищі IT-фахівців, у яких цей сканер користується довірою. Досить сказати, що близько 50-ти компаній зі списку Forbes «Global 100» використовують даний продукт.

Хмарна платформа QualysGuard і вбудований набір додатків дозволяють підприємствам спростити процес забезпечення безпеки і знизити витрати на відповідність різним вимогам, при цьому надаючи важливу інформацію про безпеку і автоматизуючи весь спектр завдань аудиту, комплекс-контролю і захист IT-систем і веб-додатків.

2.5 Програмні комплекси для пошуку SQL-вразливостей

Наявні для проведення SQL-ін'єкцій, програми зазвичай мають дві складові - сканування сайту на можливі уразливості і їх використання для отримання доступу до даних. Існують такі утиліти практично для всіх відомих платформ. Їх функціонал значною мірою полегшує перевірку сайту на можливість злому SQL-ін'єкцією.

Sqlmap

Дуже потужний сканер, що працює з більшістю відомих СУБД. ПідтримуєрізніметодикивпровадженняSQL-ін'єкцій. Має можливість автоматичного розпізнавання типу хеша пароля і його злому за словником. Присутній і функціонал завантаження і вивантаження файлів з сервера.Головне завдання сканера - автоматизований пошук та експлуатація sql вразливостей. Sqlmap написаний на python, а значить чудово працює на більшості сучасних операційних систем.

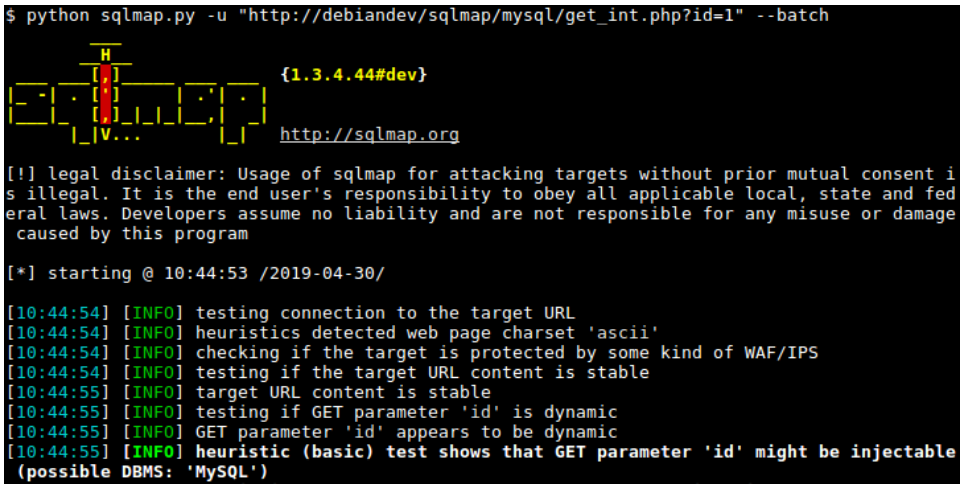

Рисунок 43. Основний вид сканера Sqlmap

Основні можливості:

✓ Підтримка MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, SQLite, Firebird, Sybase і SAP MaxDB.

✓ Підтримка наступних типів ін'єкцій: boolean-based blind , time-based blind, error-based, UNION query і stacked queries.

- ✓ Можливість підключитися до БД безпосередньо, використовуючи логін, пароль, ip, порт і ім'я бази.
- ✓ Робота з конкретним url, з списком цілей з Burp proxy або WebScarab proxy, з текстовим файлом, що містить HTTP запит або ж прямо з пошукової системи Google.
- ✓ Тестування всіх параметрів, що передаються методами GET і POST, через cookie, взаголовках User-agent і Referer і спроба їх експлуатування. Так само ви можете задати якийсь один певний параметр (и) для перевірки.
- ✓ Опціональная многопоточність, яка дозволяє сильно прискорити проведення сліпих ін'єкцій, або ж навпаки, обмежити кількість запитів на певний проміжок часу. Безліч варіантів оптимізації для прискорення.
- ✓ Можливість передавати cookie, що дозволяє проходити авторизацію на тестованому додатку або ж тестувати cookie на sql уразливості.
- ✓ Приймати і зберігати в сесії cookie, які були встановлені самим додатком, а так ж намагатися їх експлуатувати. При бажанні можна ігнорувати заголовок set-cookie.
- ✓ Аунтефікація по протоколу HTTP basic, Digest, NTLM або за допомогою сертифіката.
- ✓ Робота через проксі.
- ✓ Можливість задати довільні заголовки User-agent і Referer, або ж вибрати User-agent випадково з текстового файлу.
- ✓ Можливість змінювати рівень інформативності вихідних повідомлень (всього 7 рівнів).
- ✓ Парсинг форм, які перебувають за цільовим адресою, з метою повторної відправки запиту на приймаючий скрипт форми (action) і тестування параметрів форми.
- ✓ Гнучкість в настройки і використанні.

✓ Розрахунковий час для кожного запиту, який оновлюється в реальному часі і дозволяє пентестеру знати, скільки часу буде потрібно для отримання відповіді.

✓ Автоматично зберігає сесії (запити і відповіді, навіть частково отримані) в текстовому вигляді і в реальному часі. Це дозволяє продовжити ін'єкцію або іншу дію відразу ж, після парсинга сесії і не повторювати запити на атакується додаток.

✓ Можливість читати всі параметри з текстового файлу, а не вводити їх в ручну щораз.

✓ Підтримка реплікації БД сервера на локальну БД sqlite3.

✓ Оновлення до актуальної версії зі сховищ.

✓ Парсинг відповідей з метою знайти і показати повідомлення про помилки БД.

✓ Інтеграція з іншими інструментами для пентеста - Metasploit і w3af.

jSQL Injection

jSQLInjection – багатоплатформовий інструмент для тестування використання SQL вразливостей. Написаний на Java, тому в системі повинен бути встановлений JRE. Здатний обробляти запити GET, POST, header, cookie. Володіє зручним графічним інтерфейсом

Рисунок 44. Основний вид сканера jSQL Injection

Особливості:

- ✓ Методи GET, POST, header, cookie
- ✓ Автоматичний вибір кращого алгоритму
- ✓ Контроль многопоточності (старт / пауза / відновлення / стоп)
- ✓ Смуга прогресу
- ✓ ПоказуваннявикликівURL
- ✓ Просте ухилення
- ✓ Налаштування проксі
- ✓ Дистанційне читання файлів
- ✓ Термінал для команд
- ✓ Створення резервних копій налаштувань
- ✓ Перевірка оновлень
- ✓ Перевірка адміністративних панелей (сторінок)
- ✓ Брут-форсер (md5 mysql ...)
- ✓ Кодувальник (кодування-декодування base64 hexmd5 ...)
- ✓ Групова перевірка безлічі сайтів
- ✓ Підтримка MySQL
- ✓ Підтримує типи аутентифікації: Basic, Digest, Negotiate, NTLM і Kerberos
- ✓ Вибір типу бази даних

SQLi Dumper v.7

Дана програма - простий у використанні інструмент для пошуку і реалізації вразливостей в SQL. Працює на основі так званих Доркен. Їх списки можна знайти в інтернеті. Дорки для SQL-ін'єкцій - це спеціальні шаблони пошукових запитів. З їх допомогою можна знайти потенційно уразливий сайт через будь-який пошуковик

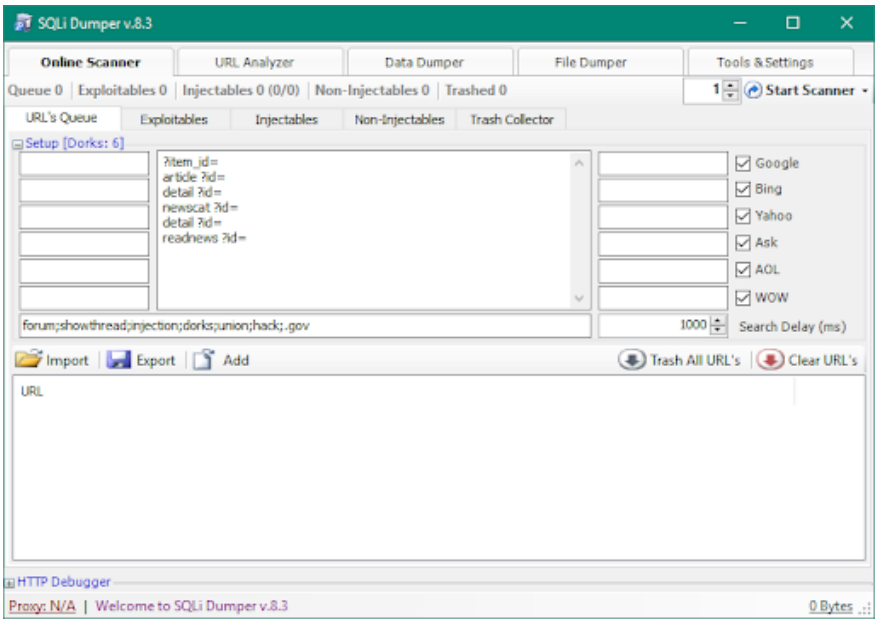

Рисунок 45. Основний вид сканера SQLi Dumper

Функціонал:

- ✓ SQL Injection;
- ✓ Operation System Function;
- ✓ Dump Database;
- ✓ Extract Database Schema;
- ✓ Search Columns Name;
- ✓ Read File (read only).

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ДЛЯ ВИЯВЛЕННЯ ТА ЗАХИСТУ ВІД SQL-АТАК

3.1 Принцип работи сканера

Зондування. Найефективніший, але повільний метод активного аналізу. Суть його полягає в тому, що сканер сам проводить спроби експлуатації знайдених вразливостей і моніторить мережу, визначаючи, де можуть пройти загрози. У процесі зондування адміністратор може підтвердити свої здогадки щодо «дірок» і вжити заходів щодо їх закриття.

Сканування. У такому режимі сканер працює максимально швидко, але проводить аналіз лише на поверхневому рівні. Тобто «дивиться» на явні «діри» і аналізує загальну безпеку інфраструктури. Відмінність цього механізму від попереднього в тому, що сканер вразливостей не підтверджує наявність уразливості, а лише попереджає про неї.

Робота сканера базується на непрямих ознаках вразливостей. Наприклад, якщо сканер аналізує протоколи прикладного рівня або API, то він визначає їх параметри і порівнює з прийнятними значеннями, заданими адміністратором. Якщо він виявить розходження значень, адміністратор отримає повідомлення про потенційну уразливості. Після цього потрібно перевірити знайдені потенційні загрози будь-якими іншими інструментами.

3.2 Алгоритм роботи сканера

1.Збір інформації з усією інфраструктурою: активні процеси, запущені додатки,порти і пристрої,які працюють, служби і т.д.

2.Пошук потенційно вразливих місць. Методом сканування (використовує непрямі ознаки уразливості без підтвердження її наявності).

3.Використання спеціальних методів моделювання атак, щоб підтвердити або спростувати наявність уразливості.

4.Формування звіту з інформацією про знайдені сайти з вразливістю.

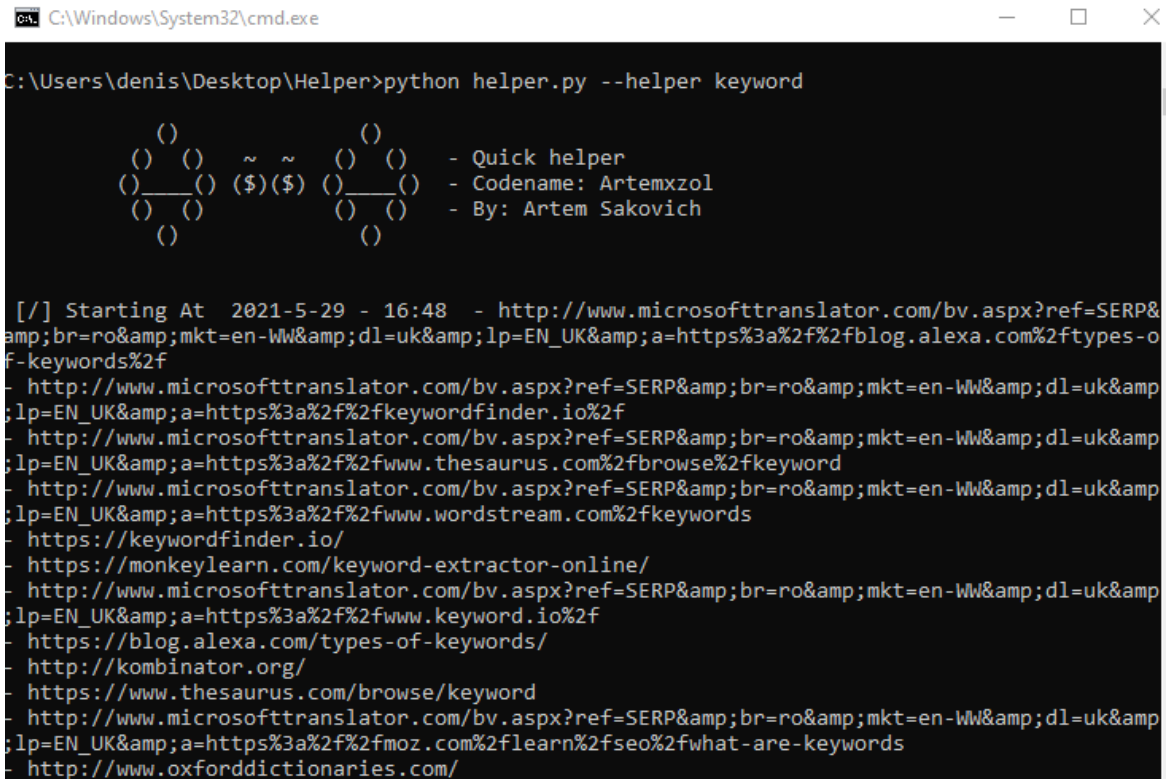

Рисунок 46. Основний вид сканера

3.3 Методи сканування

В моємо сканері було реалізовано метод TCP SYN - найбільш популярний, використовується в 95% випадків. Його називають скануванням з установкою напіввідчиненого з'єднання, так як з'єднання не встановлюється до кінця. На досліджуваний порт надсилається повідомлення SYN, потім йде очікування відповіді, на підставі якого визначається статус порту. Відповіді SYN / ACK говорять про те, що порт прослуховується (відкритий), а відповідь RST говорить про те, що прослуховується.

Якщо після декількох запитів не спадає жодної відповіді, то мережевий трафік до порту вузла призначення фільтрується засобами міжмережевого екранування.Також порт позначається як фільтрований, якщо у відповідь приходить повідомлення ICMP з помилкою досяжності (Destination Unreachable) і певними кодами і прапорами.

Також,існують інші методи,які можуть бути реалізовані в сканерах:

Метод TCP CONNECT менш популярний, ніж TCP SYN, але все-таки часто зустрічається на практиці. При реалізації методу TCP CONNECT проводиться спроба встановити з'єднання по протоколу TCP до потрібного порту з процедурою handshake. Процедура полягає в обміні повідомленнями для узгодження параметрів з'єднання, тобто службовими повідомленнями SYN, SYN / ACK, ACK, між вузлами. З'єднання встановлюється на рівні операційної системи, тому існує шанс, що воно буде заблоковано засобом захисту і потрапить в журнал подій.

UDP-сканування повільніше і складніше, ніж TCP-сканування. Через специфіку сканування UDP-портів про них часто забувають, адже

повне час сканування 65 535 UDP-портів зі стандартними параметрами на один вузол займає у більшості автоматизованих сканерів до 18 годин. Це час можна зменшити за рахунок розпаралелювання процесу сканування і рядом інших способів. Слід приділяти увагу пошуку UDP-служб, тому що UDP-служби реалізують обмін даними з великим числом інфраструктурних сервісів, які, як правило, викликають інтерес зловмисників.

Методи, які практично не використовуються:

- TCP ACK,
- TCP NULL, FIN, Xmas,
- «Ледаче сканування».

Має пряме призначення методу ACK-сканування - виявити правила засобів захисту, а також визначити фільтровані порти. У пакеті запиту при такому типі сканування встановлено тільки ACK-прапор. Відкриті та закриті порти повернуть RST-пакет, так як порти досяжні для ACK-пакетів, але стан невідомий. Порти, які не відповідають або посилають у відповідь ICMP-повідомлення Destination Unreachable з певними кодами вважаються фільтрованими.

Методи TCP NULL, FIN, Xmas полягають у відправці пакетів з відключеними прапорами в заголовку TCP. При NULL-скануванні не встановлюються жодні біти, при FIN-скануванні встановлюється біт TCP FIN, а в Xmas-скануванні встановлюються прапори FIN, PSH і URG.

Методи засновані на особливості специфікації RFC 793, згідно з якою при закритому порту вхідний сегмент, який не містить RST, спричинить за собою відправку RST у відповідь. Коли порт відкритий, відповіді не буде. Помилка досяжності ICMP означає, що порт фільтрується. Ці методи вважаються більш замкнутими, ніж SYN-сканування, однак і менш точні, тому що не всі системи дотримуються RFC 793.

«Ледаче сканування» є самим потайним з методів, оскільки для сканування використовується інший вузол мережі, який називається зомбі-вузлом. Метод застосовується зловмисниками для розвідки. Перевага такого сканування в тому, що статус портів визначається для зомбі-вузла, тому, використовуючи різні вузли, можна встановити довірчі зв'язку між вузлами мережі.

3.4 Тестування програми

Створений сканер був спрограмований на мові Python,який сканує рандомні сайти з інтернету та провіряє їх на вразливість.

Сканер має такий вигляд

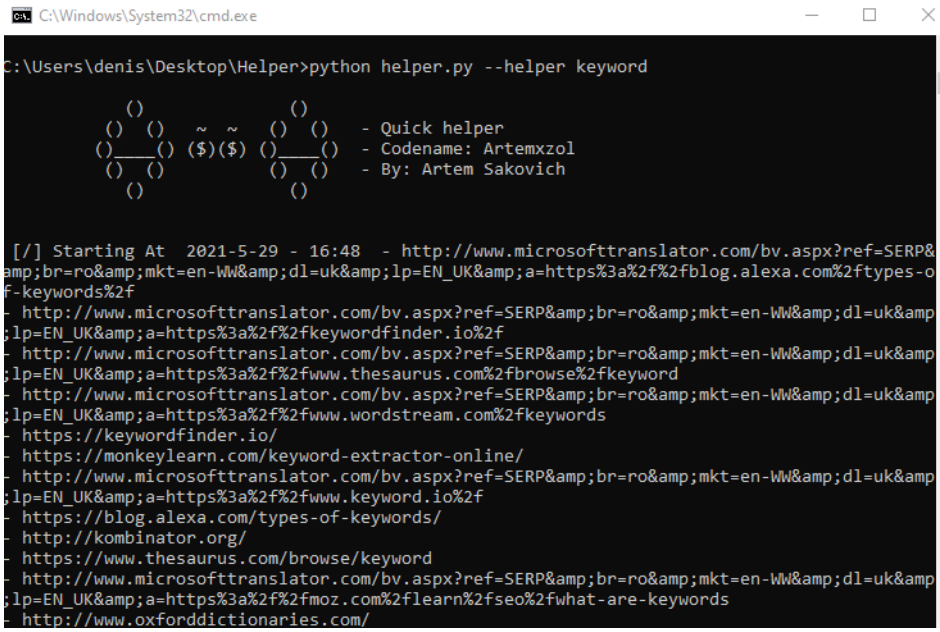

Рисунок 47. Основний вид сканера

Функції,які виконує:

- *python helper.py -h* *виводить опис та допомогу
- *python helper.py --helper keyword --scan* *сканування з SQL вразливістю
- *python helper.py --helper keyword --proxy* 0.0.0.0:1337 *сканування проксі

Спершу,переглянемо першу функцію,яка виводить опис та допомогу.Відкриваємо термінал та прописуємо:

pythonhelper.py –h

```
C:\Users\denis\Desktop\Helper>python helper.py -h
usage: helper.py --helper [keyword]  --scan

Descriptions:
 * By : Sakovich Artem
---------------------------:
- Codename : Artemxzol
+ Crawl Web Use Google helper & Bing Helper
+ with Features SQLi Scanner Vulnerability

optional arguments:
  -h, --help            show this help message and exit
  --helper [keywords]
                        Your Helper e.g inurl:.php?id=

  --proxy [proxy:port]
                        if using proxy e.g 127.0.0.1:1337
                        with auth e.g user@pass:127.0.0.1:1337

  --scan                if with Scan SQL injection Vulnerability Use This argument

C:\Users\denis\Desktop\Helper>
```

Рисунок 48. Функція опису сканера

В першому рядку,я написав назву самого кода.

В другому,описується сканування за допомогою Google та Bing з пошуком уразливостей.

Також,команди для виконання:

-h --helper --proxy --scan

Далі,розглянемо основну функцію сканера.

python helper.py --helper keyword –scan

Прописуючи команду,програма починає сканувати ран домні сайти з інтернету.

```
C:\Users\denis\Desktop\Helper>python helper.py --helper keyword --scan

          ()            ()
        ()  ()   ~  ~   ()  ()   - Quick helper
       ()____() ($)($) ()____()  - Codename: Artemxzol
        ()  ()         ()  ()    - By: Artem Sakovich
          ()            ()


 [/] Starting At  2021-5-29 - 17:21  - http://www.microsofttranslator.com/bv.aspx?ref=SERP&
amp;br=ro&amp;mkt=en-WW&amp;dl=uk&amp;lp=EN_UK&amp;a=https%3a%2f%2fwww.secretcodebreaker.co
m%2fkeyword.html
- https://support.google.com/google-ads/answer/7337243?hl=en
- https://keywordtool.io/api
- http://www.microsofttranslator.com/bv.aspx?ref=SERP&amp;br=ro&amp;mkt=en-WW&amp;dl=uk&amp
;lp=EN_UK&amp;a=http%3a%2f%2fwww.scrapebox.com%2fkeyword-scraper
- https://yoast.com/what-is-a-keyword-strategy/
- http://www.microsofttranslator.com/bv.aspx?ref=SERP&amp;br=ro&amp;mkt=en-WW&amp;dl=uk&amp
;lp=EN_UK&amp;a=https%3a%2f%2fyoast.com%2fwhat-is-a-keyword%2f
- http://www.microsofttranslator.com/bv.aspx?ref=SERP&amp;br=ro&amp;mkt=en-WW&amp;dl=uk&amp
;lp=EN_UK&amp;a=https%3a%2f%2fdocs.yoyogames.com%2fsource%2fdadiospice%2f002_reference%2f00
1_gml%2520language%2520overview%2fkeywords.html
- http://www.microsofttranslator.com/bv.aspx?ref=SERP&amp;br=ro&amp;mkt=en-WW&amp;dl=uk&amp
;lp=EN_UK&amp;a=https%3a%2f%2fblog.hubspot.com%2fmarketing%2fhow-to-do-keyword-research-ht
- https://keywordtool.io/
- https://www.wordstream.com/seo-keyword
```

Рисунок 49. Основна функція сканера

Показує сайти,захищені від вразливостей

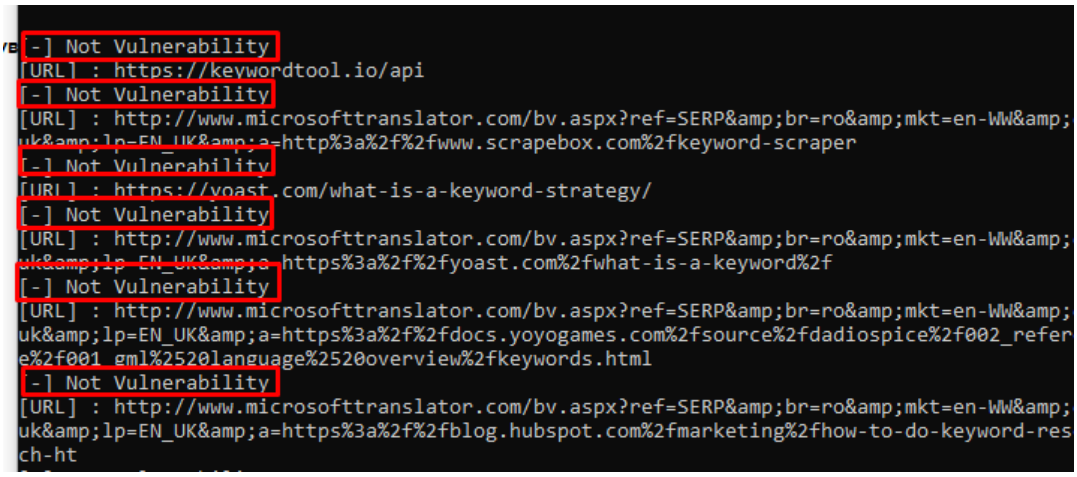

Рисунок 50. Виявлення вразливостей

Далі,починає шукати потенційно вразливі сайти,після чого,починає їх блокувати та проводити формування звіту про заблоковані сайти.

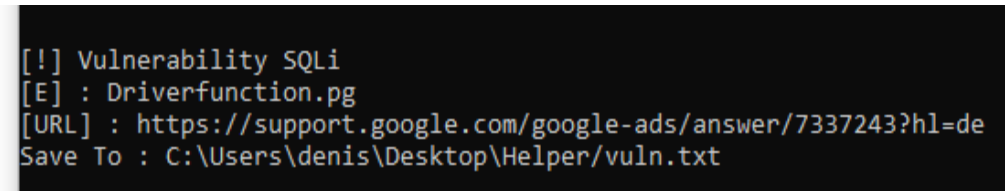

Рисунок 51(1). Не захищенні сайти

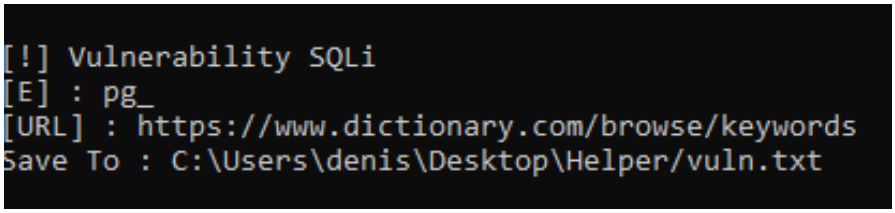

Рисунок 51(2). Не захищенні сайти

Сайти,які були заблоковані,формуються у звіт

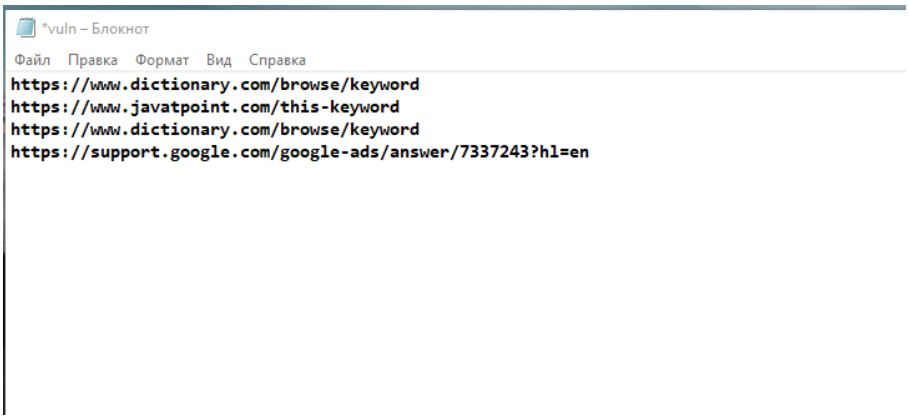

Рисунок 52.Формування звіту

Остання функція сканера,це сканування проксі

python helper.py --helper keyword --proxy 0.0.0.0:1337

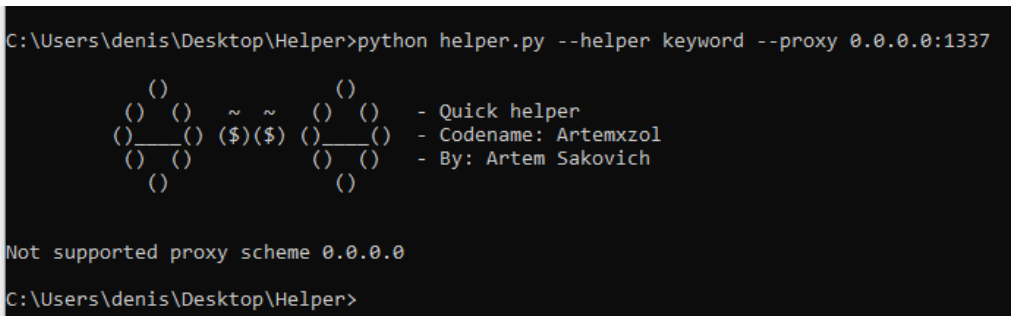

Рисунок 53.Сканування проксі

Використовував такі модулі Python

- **Termcolor**(Форматування кольорів ANSII для виводу в терміналі.)
- **Beautifulsoup4(**Фіктивний пакет**)**
- **Google**(Прив’язка Python до пошукової системи Google.)
- **Mechanicalsoup**(Бібліотека Python для автоматизації взаємодії з веб-сайтами)
- **Nyawc**(Веб-сканер)
- **Random**(модуль випадковості)
- **Sys**
- **Time**(Інструменти часу для python**)**
- **Requests (**Запити**)**

ВИСНОВКИ

В результаті дипломної роботи,були отримані такі результати:

1. Проаналізовано види SQL атак, причини їх виникнення та методи захисту від них, які дають змогу користувачу захистити свій комп'ютер від вразливостей; на основі проведеного аналізу були зроблені висновки про необхідність розробки програмного модуля виявлення та захисту від SQL-ін'єкцій.

2. Розроблено програмний модуль виявлення SQL-ін'єкцій та захисту від них, який надає змогу виявляти вразливості в веб-додатках з відкритими портами, після чого, блокувати їх.

3. Проведено тестування розробленого програмного модулю виявлення SQL-ін'єкцій та захисту від них, що дало змогу перевірити можливість використання даного програмного модулю для вирішення поставленої задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. “HACKING&TRICKS” a Blog about Hacking & Computer Security by NiravDesaihttps://tipstrickshack.blogspot.com/2013/01/list-of-vulnerability-its-tutorial.html
2. Stephen Kost (Integrity Corporation) in white paper “AN INTRODUCTION TO SQL INJECTION ATTACKS FOR ORACLE DEVELOPERS” March2007
3. СТАТИСТИКААТАКНАВЕБ-ПРИЛОЖЕНИЯIIIКВАРТАЛ2017ГОДА[Электронный ресурс]–:https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/WebApp-Vulnerabilities-2017-Q3-rus.pdf
4. Задувайло О.К.Проблема визначенняпоняття "чутлива інформація"
5. Подборка материалов по SQL Injection[Электронный ресурс]–Режим доступа:http://injection.rulezz.ru/
6.Исследование утечеки нформации за первое полугодие 2015года.[Электронный ресурс] / –:http://www.infowatch.ru/analytics/reports/16340.
7. Информационная безопасность бизнеса.Исследование текущих тенденций в области информационной безопасности бизнеса.2014.[Электронныйресурс]/–Режим доступа:http://media.kaspersky.com/pdf/IT_risk_report_Russia_2014.pdf.
8. Sandhu RaviS.,Jajodia Sushil.Data and database security and controls.Handbook of Information Security Management, Auerbach Publishers, 1993,pp.181-199.
9. Qiu M., Davis S. Database security mechanisms and implementation. IACIS,Issues in Inform.Syst.2002vol.03pp.529-534.
10. Lesov P.Database security:ahistorical perspectiv.2010.URL:http://arxiv.org/ftp/arxiv/papers/1004/1004.4022.pdf
11. Burtescu E.Database security-attacksandcontrolme-thods.Journ.of Applied Quantitative Methods2009,vol.4,no.4,pp.449-454.
12. RohillaS.,MittalP.K.Database Security:Threads and Challenges.Intern.Journ.of Advanced Research in Computer Science and Software Engineering, 2013,vol.3,iss.5,pp.810-813.
13. Потапов А.Е., Манухина Д.В., Соломатина А.С., Бадмаев А.И., ЯковлевА.В., Нилова А.С. Безопасность локальных баз данных на примере SQLServer Compact // Укр. Тамбов. ун-та. Серия: Естественные и техническиенауки.2014.№ 3.С.915-917.

14. Бортовчук Ю.В.,Крылова К.А.,Ермолаева Л.В.Информационная безопасность в современных системах управления базами данных// Современные проблемы экономического и социального развития.2010.№6.С.224-225.
15. Ткаченко Н.А. Реализация монитора безопасности СУБД MySQL в dbf /damсистемах// ПДМ.Дополнение.2014.№ 7.С.99-101.
16. Полтавцева М.А. Задача хранения прав доступа к данным в СУБД например MicrosoftSQLServer//Актуальные направления фундаментальных и прикладных исследований:матер.VМеждунар.научно-практич.конф.2015 С.118-120.
17. Баранчиков А.И., Баранчиков П.А., Пилькин А.Н. Алгоритмы и модели доступа к записям БД.М.:Горячая линия-Телеком,2011.182 с.
18. Поляков А.М.Безопасность Oracle глазами аудитора:нападение и защита.
19. Смирнов С.Н.Безопасность систем баз данных .М.:Гелиос АРВ,2007.352с.
20. Murray M.C.Databas esecurity:what students need to know. JITE :IIP ,vol.9,2010 pp.61-77.
21. Зегжда П.Д. Обеспечение безопасности информации в условиях создания единого информационного пространства // Защита информации. Инсайд.2007.№ 4 (16).С.28-33
22. Есин В. И. Безопасность информационных систем и технологий / Есин В.И.,Кузнецов А.А.,Сорока Л.С.-М.: Эден,2010.-656 с
23.https://proglib.io/p/vzlamyvaem-sayty-shpargalka-po-sql-inekciyam-2019-12-21
24.http://www.securityscripts.ru/articles/Базы%20данных/sql-injection.html
25.https://www.kp.ru/guide/setevye-skanery-bezopasnosti.html
26.https://habr.com/ru/company/pt/blog/513776/

ДОДАТОК

Лістинг програми

1. Файл helper.py

```
import sys,argparse
if sys.version[0] in '2':
  print('\n[x] Not Supported For python 2.x Please Use Python 3.x \n')
  exit()
from lib.tmp import color as _
try:
   import mechanicalsoup
   import requests
except Exception as e:
   print('\n{}[-]{} mechanicalsoup package Not Installed\n'.format(_.R,_.W))
   print('type pip3 install mechanicalsoup')
   exit()

from lib.lib import Parser
from lib.sqlscan import sqli_scan
from lib.tmp import logo
from lib.tmp import color as c

urls = []

class crawl(object):

    auth = {
```

```
  1:[
     'https://www.google.com',
     'class="r"><a href="/url\?q=(.*?)&amp',
     'fl'
  ],
  2:[
     'https://www.bing.com',
     'h=".*?" href="(h.*?")',
     "b_widePag sb_bp"
  ]
}

def __init__(
  self,
  helper,
  proxy = None
):
    self.helper = helper
    self.proxy = proxy

def Bing(self):
    bing = Parser(
      self.helper,
      crawl.auth[2][0],
      crawl.auth[2][1],
      crawl.auth[2][2],
      proxy = self.proxy
    )
    bing.request()
    for url in dir(bing):
```

```
            if 'go.microsoft.com' in url or 'bing.com' in url:
                pass
            else:
                urls.append(url)

    def Google(self):
        google = Parser(
          self.helper,
          crawl.auth[1][0],
          crawl.auth[1][1],
          crawl.auth[1][2],
          proxy = self.proxy
        )
        google.request()
        for url in dir(google):
            if 'go.microsoft.com' in url or 'bing.com' in url:
              pass
            else:
              urls.append(url)

class SQLi_Scanner(sqli_scan):
    pass

if sys.version[0] in '2':
  print('\n[x] Not Supported For python 2.x Please Use Python 3.x \n')
  exit()
fel = sys.argv[0]
par = argparse.ArgumentParser(
prog=fel,
usage="%(prog)s --helper [keyword]  --scan",
```

```
formatter_class=argparse.RawTextHelpFormatter,
description="""
Descriptions:
 * By : Sakovich Artem
----------------------------:
- Codename : Artemxzol
+ Crawl Web Use Google helper & Bing Helper
+ with Features SQLi Scanner Vulnerability

""")

par.add_argument(
'--helper',
help="""
Your Helper e.g inurl:.php?id=

""",
metavar='[keywords]',
type=str)
par.add_argument(
'--proxy',
help='''
if using proxy e.g 127.0.0.1:1337
with auth e.g user@pass:127.0.0.1:1337

''',
metavar='[proxy:port]',
type=str,
action='store',
default=None)
```

```
par.add_argument(
'--scan',help="if with Scan SQL injection Vulnerability Use This argument",action="store_true")
arg = par.parse_args()
try:
    if arg.scan:
        if arg.helper != None:
            print(logo())
            _ = crawl(arg.helper,proxy=arg.proxy)
            _.Bing()
            _.Google()
            if urls != []:
                for url in list(set(urls)):
                    print('- {}'.format(url))
                for scan in list(set(urls)):
                    try:
                        SQLi_Scanner().scan(scan)
                    except Exception as e:
                        print(e)
            else:
                print('\n{}[-]{} No Url Found !\n'.format(c.R,c.W))
        else:
            par.print_help()
    elif not arg.scan:
        if arg.helper != None:
            print(logo())
            _ = crawl(arg.helper,proxy=arg.proxy)
            _.Bing()
            _.Google()
            if urls != []:
```

```
            for url in list(set(urls)):
                print('- {}'.format(url))
        else:
            print('\n{}[-]{} No Url Found !\n'.format(c.R,c.W))
    else:
        par.print_help()
else:
    par.print_help()
except Exception as e:
    print(e)
except KeyboardInterrupt:
    exit()
```

2. Файл sqlscan.py

```
#-*- coding: utf-8 -*-
from os import getcwd as pth
from .tmp import color as _
import requests,re

class sqli_scan(object):

    @property
    def error(self):
        return {'Syntax error' : 'Syntax error',
                'Not found' : 'Not found',
                'Server Error' : 'Server Error',
                'Error Occurred While Processing Request':'Error Occurred
While Processing Request',
                "ZIMBRA-WEB-MAIL-01":"zimbra_user",
                "ZIMBRA-WEB-MAIL-02":"zimbra_ldap_password",
                "ZIMBRA-WEB-MAIL-03":"ldap_replication_password",
                "ZIMBRA-WEB-MAIL-04":"ldap_root_password",
                "ZIMBRA-WEB-MAIL-05":"ldap_nginx_password",
                "ZIMBRA-WEB-MAIL-06":"mailboxd_keystore_password",
```

```
        "ZIMBRA-WEB-MAIL-07":"zimbra_mysql_password",
        "ZIMBRA-WEB-MAIL-08":"zimbra_root_password",
        "ZIMBRA-WEB-MAIL-09":'mysql_root_password',
        "ZIMBRA-WEB-MAIL-10":'mailboxd_truststore_password',
        "ZIMBRA-WEB-MAIL-11":'ldap_postfix_password',
        "ZIMBRA-WEB-MAIL-12":'ldap_amavis_password',
        "MARIA-DB":'MariaDB server version for the right syntax',
        "MYSQL-01":'You have an error in your SQL syntax;',
        "MYSQL-02":'Warning: mysql_',
        "MYSQL-03":'function.mysql',
        "MYSQL-04":'MySQL result index',
        "MYSQL-05":'MySQL Error',
        "MYSQL-06":'MySQL ODBC',
        "MYSQL-07":'MySQL Driver',
        "MYSQL-08":'mysqli.query',
        "MYSQL-09":'num_rows',
        "MYSQL-10":'mysql error:',
        "MYSQL-11":'supplied argument is not a valid MySQL result
resource',
        "MYSQL-12":'on MySQL result index',
        "MYSQL-13":'Error Executing Database Query',
        "MYSQL-14":'mysql_',
        "MYSQL-15":'mysql_fetch_array\(\)',
        "MYSQL-16":'mysql_fetch',
        "MICROSOFT-01":'Microsoft JET Database',
        "MICROSOFT-02":'ADODB.Recordset',
        "MICROSOFT-03":'Unclosed quotation mark',
        "MICROSOFT-04":'500 - Internal server error',
        "MICROSOFT-05":'Microsoft OLE DB Provider',
        "MICROSOFT-06":'Unclosed quotes',
        "MICROSOFT-07":'ADODB.Command',
        "MICROSOFT-08":'ADODB.Field error',
        "MICROSOFT-09":'Microsoft VBScript',
        "MICROSOFT-10":'Microsoft OLE DB Provider for SQL
Server',
        "MICROSOFT-11":'Unclosed quotation mark',
        "MICROSOFT-12":'Microsoft OLE DB Provider for Oracle',
        "MICROSOFT-13":'Active Server Pages error',
        "MICROSOFT-14":'OLE/DB provider returned message',
```

```
            "MICROSOFT-15":'OLE DB Provider for ODBC',
            "MICROSOFT-16":"error '800a0d5d'",
            "MICROSOFT-17":"error '800a000d'",
            "MICROSOFT-18":'Unclosed quotation mark after the character
string',
            "MICROSOFT-19":'\[Microsoft\]\[SQL Server Native Client
11.0\]\[SQL Server\]',
            "MICROSOFT-20":'Warning: odbc_',
            "ORACLE-01":'ORA-00921: unexpected end of SQL command',
            "ORACLE-02":'ORA-01756',
            "ORACLE-03":'ORA-',
            "ORACLE-04":'Oracle ODBC',
            "ORACLE-05":'Oracle Error',
            "ORACLE-06":'Oracle Driver',
            "ORACLE-07":'Oracle DB2',
            "ORACLE-08":'Error ORA-',
            "ORACLE-09":'SQL command not properly ended',
            "DB2-01":"DB2 ODBC",
            "DB2-02":"DB2 error",
            "DB2-03":"Driver",
            "ODBC-01":"ODBC SQL",
            "ODBC-02":"ODBC DB2",
            "ODBC-03":"ODBC Driver",
            "ODBC-05":"Microsoft Access",
            "ODBC-06":"Oracle",
            "ODBC-07":"Microsoft Access Driver",
            "ODBC-08":'OLE DB Provider for ODBC',
            "POSTGRESQL-01":'Warning: pg_',
            "POSTGRESQL-02":"PostgreSql Error:",
            "POSTGRESQL-03":"function.pg",
            "POSTGRESQL-04":'Supplied argument is not a valid
PostgreSQL result',
            "POSTGRESQL-05":'PostgreSQL query failed: ERROR: parser:
parse error',
            "POSTGRESQL-06":"pg_",
            "SYBASE-01":'Warning: sybase_',
            "SYBASE-02":'function\.sybase',
            "SYBASE-03":'Sybase result index',
            "SYBASE-04":'Sybase Error:',
```

```
"SYBASE-05":'Sybase: Server message:',
"SYBASE-06":'\[Sybase\]\[ODBC Driver\]:',
"SYBASE-07":"sybase_",
"JDBC_CFM-01":'Error Executing Database Query',
"JDBC_CFM-02":'SQLServer JDBC Driver',
"JDBC_CFM-03":'JDBC SQL',
"JDBC_CFM-04":'JDBC Oracle',
"JDBC_CFM-05":'JDBC MySQL',
"JDBC_CFM-06":"JDBC Error",
"JDBC_CFM-07":"JDBC Driver",
'Not found' : 'Not found',
"PHP-ERROR-01":'Warning: include',
"PHP-ERROR-02":'Fatal error: include',
"PHP-ERROR-03":'Warning: require',
"PHP-ERROR-04":'Fatal error: require',
"PHP-ERROR-05":'ADODB_Exce2ption',
"PHP-ERROR-06":'Warning: include\(',
"PHP-ERROR-07":'Warning: require_once\(',
"PHP-ERROR-08":'function.include',
"PHP-ERROR-09":'Disallowed Parent Path',
"PHP-ERROR-10":'function.require',
"PHP-ERROR-11":'Warning: main\(',
"PHP-ERROR-12":'Warning: session_start\(\)',
"PHP-ERROR-13":'Warning: getimagesize\(\)',
"PHP-ERROR-14":'Warning: array_merge\(\)',
"PHP-ERROR-15":'Warning: preg_match\(\)',
"PHP-ERROR-16":'GetArray\(\)',
"PHP-ERROR-17":'FetchRow\(\)',
"PHP-ERROR-18":'Warning: preg_',
"PHP-ERROR-19":'Warning: ociexecute\(\)',
"PHP-ERROR-20":'Warning: ocifetchstatement\(\)',
"ASP-ERROR-01":'Version Information: Microsoft \.NET
Framework',
"ASP-ERROR-02":'ASP.NET is configured to show verbose
error messages',
"ASP-ERROR-03":"BOF or EOF",
"ASP-ERROR-04":'Unclosed quotation mark',
"ASP-ERROR-05":'Error converting data type varchar to
numeric',
```

```
            "INDEFINITE-01":'Input string was not in a correct format',
            "INDEFINITE-02":'An illegal character has been found in the
statement',
            "INDEFINITE-03":"Invalid Querystring",
            "INDEFINITE-04":"Fatal error",
            "INDEFINITE-05":"Incorrect syntax near",}

    def scan(self,url):
        self.url = url
        _vuln = 0
        __vuln = ""
        resp = requests.get(f"{self.url}"").text
        for a,b in self.error.items():
            if re.search(b,resp):
                f = open(f'{pth()}/vuln.txt','a+')
                f.write(f'{self.url}\n')
                f.close()
                _vuln += 1
                __vuln += b
            else:
                pass

        if _vuln > 0:
            print('\n')
            print(f'{_.G}[!]{_.W} Vulnerability SQLi')
            print(f'{_.Y}[E]{_.W} : {__vuln}')
            print(f'{_.B}[URL]{_.W} : {self.url}')
            print(f'Save To : {pth()}/vuln.txt')
            print('\n')
        else:
            print(f'{_.R}[-]{_.W} Not Vulnerability')
            print(f'{_.B}[URL]{_.W} : {self.url}')
```

3. Файл tmp.py

```
#-*- coding: utf-8 -*-
import sys
from time import sleep,localtime
```

```
class color(object):
    if sys.platform in ['linux','linux2']:
      R = '\033[91m'
      G = '\033[92m'
      B = '\033[94m'
      Y = '\033[93m'
      W = '\033[0m'
    else:
      R = ''
      G = ''
      B = ''
      Y = ''
      W = ''



class logo(color):

    def __init__(self):
        pass

    def __str__(self):
        _ = color
        return f'''
          ()            ()
         ()  ()   ~  ~   ()  ()   -{_.G} Quick helper{_.W}
        ()____() ($)($) ()____()  - Codename{_.W}: Artemxzol
         ()  ()        ()  ()   - {_.R}By{_.W}: Artem Sakovich
          ()            ()

        '''
def load():
   _ = localtime()
   x = [
      '|','\\','-','/'
   ]
   for y in x:
```

```
        print(f'\r [{y}]',end=' Starting At %s %s-%s-%s%s - %s:%s  ' %
(color.G,_[0],_[1],_[2],color.W,_[3],_[4]))
        sys.stdout.flush()
        sleep(0.15)
```

```
   # -*- coding: utf-8 -*-

import random
```

4. Файл useragent.py

```
user_agents = [
 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101
Firefox/64.0',
 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_2) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
 'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_2) AppleWebKit/605.1.15
(KHTML, like Gecko) Version/12.0.2 Safari/605.1.15',
 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:64.0) Gecko/20100101
Firefox/64.0',
 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36
Edge/17.17134',
 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/71.0.3578.98 Safari/537.36',
 'Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:64.0) Gecko/20100101
Firefox/64.0',
 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:64.0) Gecko/20100101
Firefox/64.0',
```

```
 'Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
 'Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0',
 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_0) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_1) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
 'Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko',
 'Mozilla/5.0 (X11; Linux x86_64; rv:64.0) Gecko/20100101 Firefox/64.0',
 'Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like
Gecko',
 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/605.1.15
(KHTML, like Gecko) Version/12.0.2 Safari/605.1.15',
 'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/71.0.3578.98 Safari/537.36',
 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:64.0) Gecko/20100101
Firefox/64.0',
 'Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:64.0) Gecko/20100101
Firefox/64.0',
 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101
Firefox/65.0',
 'Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:60.0) Gecko/20100101
Firefox/60.0',
 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14) AppleWebKit/605.1.15
(KHTML, like Gecko) Version/12.0 Safari/605.1.15',
 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/70.0.3538.102 Safari/537.36
OPR/57.0.3098.116',
 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/70.0.3538.102 YaBrowser/18.11.1.805 Yowser/2.5
Safari/537.36',
 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36
Edge/18.17763',
```

'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36 Edge/16.16299',
'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36',
'Mozilla/5.0 (iPad; CPU OS 12_1_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.0 Mobile/15E148 Safari/604.1',
'Mozilla/5.0 (Windows NT 6.1; rv:60.0) Gecko/20100101 Firefox/60.0',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_1) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.0.1 Safari/605.1.15',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36',
'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.81 Safari/537.36',
'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 YaBrowser/18.11.1.805 Yowser/2.5 Safari/537.36',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.0.3 Safari/605.1.15',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/11.1.2 Safari/605.1.15',
'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/71.0.3578.98 Chrome/71.0.3578.98 Safari/537.36',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.0.3 Safari/605.1.15',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.0.2 Safari/605.1.15',
'Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; Touch; rv:11.0) like Gecko',
'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',

'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 Safari/537.36 OPR/57.0.3098.106',
'Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0',
'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 Safari/537.36 OPR/57.0.3098.116',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:65.0) Gecko/20100101 Firefox/65.0',
'Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20100101 Firefox/64.0',
'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36',
'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:60.0) Gecko/20100101 Firefox/60.0',
'Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:64.0) Gecko/20100101 Firefox/64.0',
'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.106 Safari/537.36',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/11.1.2 Safari/605.1.15',
'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.80 Safari/537.36',
'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36',
'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.80 Safari/537.36',
'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:63.0) Gecko/20100101 Firefox/63.0',
'Mozilla/5.0 (X11; CrOS x86_64 11151.59.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.94 Safari/537.36',
'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:64.0) Gecko/20100101 Firefox/64.0',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:64.0) Gecko/20100101 Firefox/64.0',

```
 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101
Firefox/66.0',
 'Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko',
 'Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:65.0) Gecko/20100101
Firefox/65.0',
 'Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko'
 ]


def useragent():
    return random.choice(user_agents)
```