ISSN: 1813-162X (Print) ; 2312-7589 (Online)

Tikrit Journal of Engineering Sciences

available online at: http://www.tj-es.com

Ablahd AZ, Dawood SA. Using Flask for SQLIA Detection and Protection. *Tikrit Journal of Engineering Sciences* 2020; **27**(2): 1- 14.

Using Flask for SQLIA Detection and Protection

Ann Z. Ablahd[*1]
Suhair A. Dawwod[2]

[1]Computer Engineering Dept. / Technical College / Northern Technical University/ Kirkuk-Iraq

[2]Management Information System Dept. /Administration & Economic/ College/ Mosul-Iraq

Keywords:

SQL injection, Flask, vulnerability, Web application, Python, Django

ARTICLE INFO

Article history:

Received 25 October 2019
Accepted 25 April 2020
Available Online 09 May 2020

ABSTRACT

At present the web applications are used for most of the life activities, these applications are affected by an attack called (Structure Query Language Injection Attack) SQLIA due to the vulnerabilities of the web application. The vulnerabilities of the web application are increased because most of application developers do not care to security in designing.SQL injection is a common attack that infects a web application. The attacker adds (Structured Query Language) SQL code to web page for accessing and changing victim databases.The vital step in securing the database and detecting such an attack in web apps is preparing a tool. Many researchers propose different ways for detection and prevention of such as an attack. In this paper a tool it proposed using a powerful micro-framework web application designer called Flask in Python 3.7 to detect and prevent such attacks. The proposed system is called SQLIAD. SQLIAD analyzed a web application on-line.

DOI: http://doi.org/10.25130/tjes.27.2.01

استخدام الفلاسك للكشف والحماية من هجوم حقن لغة الاستعلام المنظمة

آن زكي أبلحد / قسم هندسة تقنية الحاسوب , الكلية التقنية الهندسية كركوك ,الجامعة التقنية الشمالية , العراق
سهير عبد داؤد / قسم نظم المعلومات الادارية , كلية الادارة والاقتصاد , جامعة الموصل , العراق

الخلاصة

يتم استخدام تطبيقات الويب في الوقت الحاضر لمعظم أنشطة الحياة . ان هذه التطبيقات تتأثر بهجوم من قبل الهكرز وذلك بسبب نقاط الضعف التي تلحق بها والناتجة عن قلة الخبرة والضعف في تصميمها , لأن معظم مطوريها لا يهتمون بالأمان في عملية التصميم. ان هذا الهجوم يطلق عليه الحقن باستخدام لغة الاستعلام المنظمة. هذا النوع من الهجوم يهدف الى اضافة احد رموز لغة الاستعلام المعروفة للمبرمجين الى قواعد البيانات وتغييرها في الحاسبة الضحية . الخطوة الحيوية في تأمين قاعدة البيانات واكتشاف مثل هذا الهجوم في تطبيقات الويب والحماية منه تم اعداد اداة برمجية باستخدام الفلاسك في لغة البايثون النسخة 3.7 . اقترح الباحثين طرقا مختلفة للكشف عن هذه الهجمات والوقاية منها. اقترحت اداة للتحليل والكشف والوقاية من هذه الهجمات سميت SQLIAD . في هذه الاداة استخدم إطار دقيق وكفوء من ضمن اطارات لغة البايثون وهو الفلاسك في تصميم صفحات الويب.ان عملية التحليل والكشف تجرى عبر الانترنت.

الكلمات الدالة: فلاسك, ثغرة, بايثون, ديجانكو,,حقن لغة الاستعلام.

* Corresponding Author: E-mail: dranne@ntu.edu.iq , Tel: 07702386040

1. INTRODUCTION

Web security is the most important topic of web applications due to the rapid developments in the internet and because it is valuable and most popular for attacks. Different web applications react with the database background systems, which stored the most sensitive information (like bank account,…etc). There are two types of web applications, static and dynamic. In the static web application, it is difficult to alter the content of a web application, it can only be done by a webmaster. But a dynamic web application is much more complex than a static web application, due to updating data by users each time. This has a general administration control that is controlled by administrators in modifying the content of web applications including images and text. The Structured Query Language (SQL) is a standard computer programming language used for storing, retrieving and manipulating relational stored database. This language

is used to access, edit, modify and delete data. Different websites are stored as SQL databases in the server.
There is a dangerous attack that infects a vulnerability web application called SQL injection attack (SQLIA). This attack infects dynamic web applications and it can lead to modify sensitive and confidential information.
This attack sends malicious inputs to database-driven web applications. These inputs are used by applications in building dynamic SQL queries, and have the ability to alter query semantic structures, due to lacking separation between control and SQL database. The attacker attempts to append malicious SQL commands by inserting special variables into web applications. This leads to steal, damage, edit and delete the main databases server. There are many companies which became victims of that attack and lost a lot of money. To improve the web application security, an online tool was designed using Django, Flask tools in python3.7 for detecting and preventing SQL injection attack. This on-line tool is called SQLIAD.

2. RELATED WORKS

Huan [1] examined input by "Tainted Data Tracking" research, which is a used precondition. The drawback of this research is that the technique is assumed accurately expressed for sensitive functions. Boyd [2] introduced “SQL Randomization” this research is based on the randomization of the instruction set, by appending a random number after the keyword of SQL used in building the statement of SQL. In the runtime, the parser of SQL finds that the injected SQL key does not have an appended number and will stop the attack. Kema [3] introduces *“Novel-Specification Based Methodology”.* This research compares generating with predefined SQL statements that is done at compilation time. Junjin [4] introduces "An approach for SQL injection vulnerability detection". In this research, inputs are compared for legitimate and attacker SQL query structure where done. The drawback of this research, it is impossible to make a set of legal inputs in a large web application. Kim [5] introduces "Injection attack detection using the removal of SQL query attribute values". In this research dynamic and static analysis is used. It designed a function for detecting the generated SQL queries at runtime for normal users and compares the static generated queries by normal user with the dynamically generated queries by attacker. The drawback of this technique is the source code and prepared learning is needed. Ryan Turner [6] introduces "How can I prevent SQL Injection in PHP" Using sqlmap of python2.3. But this research is slow in analysis and execution. The proposed system used Django and flask tools with a new version of python (3.7) that leads to prepare an easy-to-use and powerful application to detect and prevent a dangerous attack SQLI from a web application.

3. THE ARTITECTURE OF WEB APPLICATION

A web application is a framework which consists of interactions between many applications, middleware, and database systems on the web, with ensuring that many applications run simultaneously [7].
After typing the URL at the address bar of Browser, this is a request for a web browser. This request is sent to the server for getting responses as files, to be executed. Fig.1. represents the structure of web application.

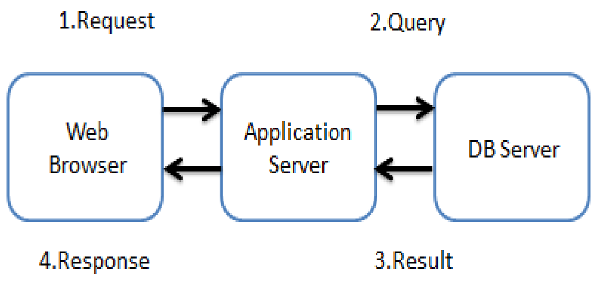
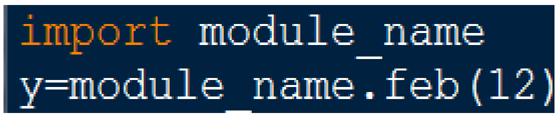


Fig.1. Web application structure

The Web Application can be accessed by an Internet Web Browser. It is a computer application written in different languages (JavaScript, Java, Html… etc.) to be common for executable with a different Browser. [8] Web application supports a dynamic basic database-driven that consists of server-side script web pages, which is written in one programming language like java to extract database information depending on dynamic interactions. [9] Many web applications believe that the input of the user is legitimate to build queries of SQL to access databases [8]. These applications are more susceptible to SQLI attacks.

4. PYTHON

It is a dynamic programming language, created in the beginning of the 1990s by Guido Rossum [10]. Python is an interpreter boost multi programming model involving functional and Object-Oriented Programming. A python program consists of various modules. A module is a python file that contains python statements with extension (py) [11]. The data in python is represented as an object. The syntax called "Decorators" is used in designing SQLIAD that allows transforming function to others easily. It is potential in python to produce a module reference and imported with other modules. It is possible in python to import different modules with the command (import module-name) after using this command a reference of this module will be created with module-name. It imports classes, functions, and variables. The access of each part of the module is done by specifying the name of that part separated by a dot (.)see Fig.2. [2]



```
import module_name
y=module_name.feb(12)
```

Fig.2. Import statement

The proposed system used a parameter passing technique. Basically, in python have to pass with reference using object and passing by value.
Python is the most powerful language for finding a vulnerability in a web application when compared with other languages [13]. Table 1 shows the ranking of different languages in finding the vulnerability of web applications [14].

Table 1
Ranking of different languages in detecting web application vulnerability, after [14].

Language Rank	SQL Injection detecting Ranking
1- Python	100
2- C	99.7
3- Java	99.5
4- C^{++}	97.1
5- R	87.7
6- C#	87.7
7- JAVAScript	85.6
8- PHP	81.2
9- GO	75.1
10- Swift	73.7

A new line is used as a delimiter in a python program between two statements. A "#" symbol used to defines comment in python.

5. DJANGO

Django is an open-source python free web framework. The primary goal of Django is to ease designing a complex web application supported with databases driven and low cost [5]. The power here in reusing the applications easily by using URL for linking it together. Django has the ability to work as optional administrative in creating, updating, reading, and deleting generate dynamic interfaces, through configuration by admin. [15]

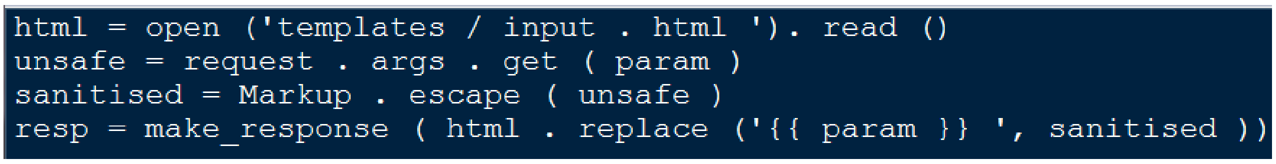
Django has the ability to force developer for adding functions in a different way. The way of building Django forces the developer for adding functions by using a specific way. The Django application contains the following modules:

- The main modules, that the application starts code executing.
- The test modules, app testing.
- The view modules mean app visualization.
- The URLs module, mapping URLs for viewing.
- The model's module, models of databases.
- The apps module means apps nested.

Most of the well-known frameworks or web sites used Django like Instagram, Mozilla, and Flask. The library of Django is generated using the python command "pip install Django".

6. FLASK

It is a micro web framework in python3. Flask is very powerful customizable when chosen for developing web applications. Also, it is very flexible when it used by developers with databases, easy to design packages and server development [16]. The nature of Flask is that by a small limit of code you can design a webpage. The power of Flask is the capability of developers to customize all things. In normal python there was an input () command used to input data and eval (value) command to hand the input without filtration from any annoyance [17]. But after using Flask, it has different means of input rather than the input () command with sanitization of input data used in protection from SQLI. Flask provided by Markup class that uses string replace () command to sanitizing user input. The Markup class contains an escape function responsible for sanitizing the input by escaping the Html markup. Fig.3. shows sanitizing of user input by Flask.

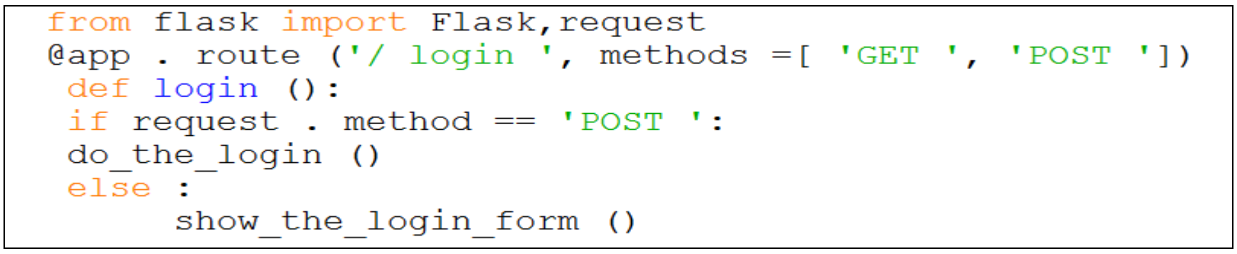


```
html = open ('templates / input . html '). read ()
unsafe = request . args . get ( param )
sanitised = Markup . escape ( unsafe )
resp = make_response ( html . replace ('{{ param }} ', sanitised ))
```

Fig.3. sanitizing of user input by Flask

There are many functions in Flask that are used in this proposed system (Routing, Requests, Responses, Templates) functions.
The Routing function that is used in flask (app.route) decorator to bind functions to the URL path. For e.g., if the path is '/' that means to get the output from the root path. Another parameter of (app.route) decorator like (GET,POST) where "GET requests" used to fetch server information, but POST request used to send the data to used server. Fig.4. shows an example of (GET, POST) requests.

The reason for the chosen Flask tool in this proposed system is simple and easy on development and testing web applications.



```
from flask import Flask,request
@app . route ('/ login ', methods =[ 'GET ', 'POST '])
 def login ():
 if request . method == 'POST ':
 do_the_login ()
 else :
      show_the_login_form ()
```

Fig.4. login form using a POST request

A templates engine used by Flask called Jinja2 which keeps the security of application. These templates will escape all dangerous input that is used in SQLIAD. Django needs a way for generating dynamic HTML, a common way of generating HTML pages is using template. A template in python is a way of representing data in different attractive forms.

7. SQLALCHEMY TOOL

SQLAlchemy is a toolkit used in operating directly with SQL databases. The aim of using this tool is to access databases inactive and high performance with simple python code [18]. This tool consists of several dependence components organized into layers. The major parts are "Object Relational Mapper" (ORM), "SQL Expression Language"; the second part may be used independently of ORM. The Expression of SQL remains as a part of public API (Application Programming Interface) as it can be used within queries and configuration of Object-Relational.

There are three separate layers (illustrated in Fig.5.) of SQLAlchemy: Core, ORM, and Dialects. The ORM is introduced and a new user can begin by this section with a high level of SQL. Core is the second section; it is the heart of this tool; it is the core between the integration of databases, services description, and Expression of SQL language. The third section is Dialects that is a documentation reference for all defined database and provided with a backend DBAPI. To generate a library of SQLAlchemy, the command "pip install SQLAlchemy" was used through the prepared SQLIAD app. This tool is also used to prevent SQL Injection attacks by using parameterized queries with (ORMs) mappers.

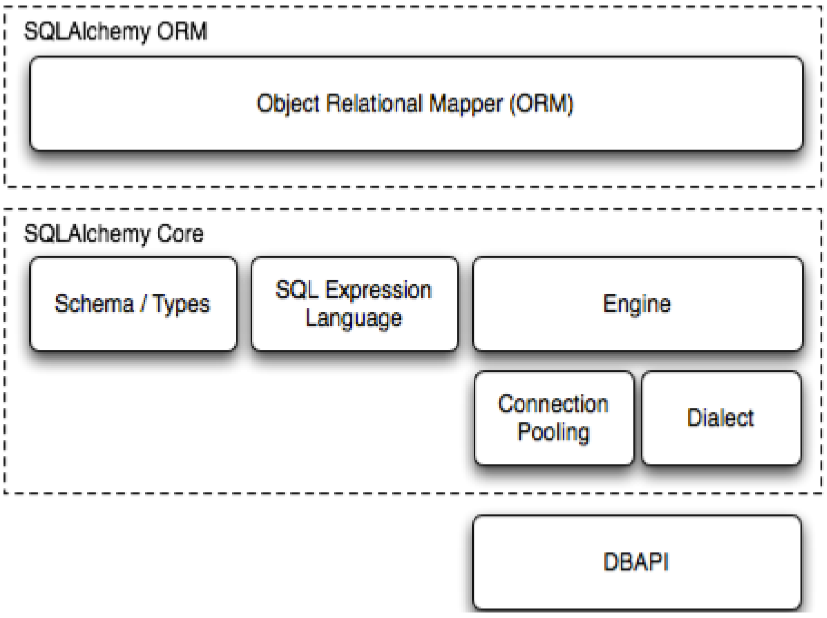


Fig.5. layers of SQLAlchemy tool

8. SQL INJECTION

The injection vulnerability is the most important part of the Open Web Application Security Project (OWASP) and is still number one of web application risks [19], where the input of users used without sanitizing by application. [20]

There are many attacks which infect web application, Fig.6. represents a diagram of the attack rate in web applications [20]. The most dangerous attack is SQL Injection (SQLI). This attack infects web application, by lack of variable input filtering. It attempts to access the database table of websites if the webserver didn't have any protection against this attack [10][9].

This proposed system focuses on the SQLI attack by solving the weak user input by detecting and preventing this type of attack. The user input passed dynamically in web application to create SQL statement Fig.7. describes the steps of URL (Uniform Resource Locator) requested. After being realized by the attacker, the web application is vulnerable to SQLI, he will inject a query of SQL commands to try stealing important information of the victim [21].

The problem is characterized by setting up a number of rules which define the problem [11]. The program is then converted into a model that represents the flow of the program. The solution to the problem will then be gradually approached by an algorithm that applies the defined rules on the model. The algorithm wills stop when it cannot get closer to an answer. [17]. There are many serious risks resulted in a SQL injection attack as follows:

- Hacking person account.
- Stealing, copying data, changing the configuration of application, viewing, and deleting sensitive, private data (users' profiles).
- Allowing users to log in to the application as another user even administrator.
- Modifying database structure and deleting database application tables.
- Controlling databases server and running its commands.

To prevent the above serious risks, it led to prepare a protection system against this type of attack. The proposed system (SQLIAD) aims to improve the user input by filtering it from malicious attacks on a dynamic web application using AST tools in python 3.7., before posting to the server. Many developers of web applications neglecting the high risks of SQLI on confidentiality and security of stored data. SQLIAD tool generates a report for each web application test to declare this web vulnerable or not.

Many webpages accept user input such as search item, password, and username to be used in building an SQL query, posted to the database. Such input, if it is malicious code for e.g. it can delete specific information within client records webpage tables.

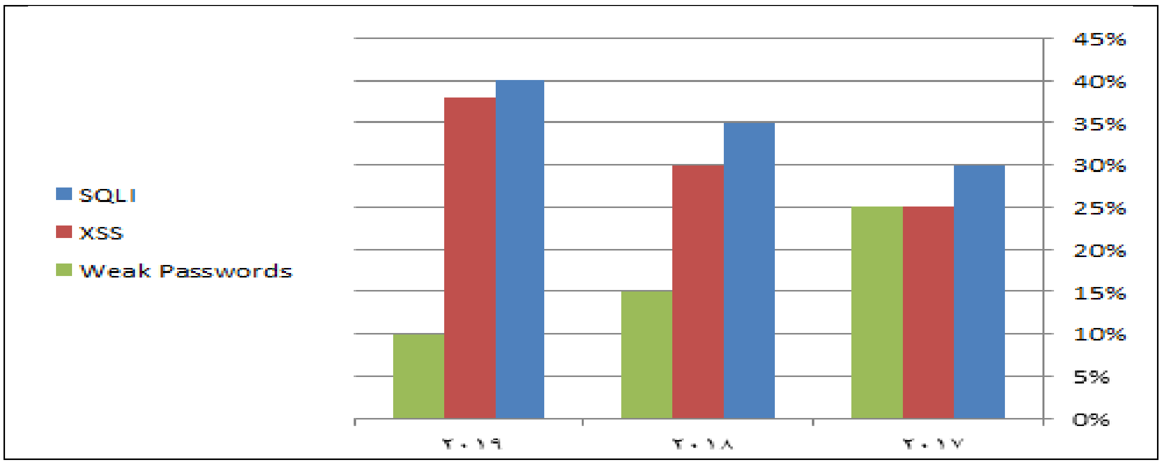


Fig.6. most dangerous attack in web application

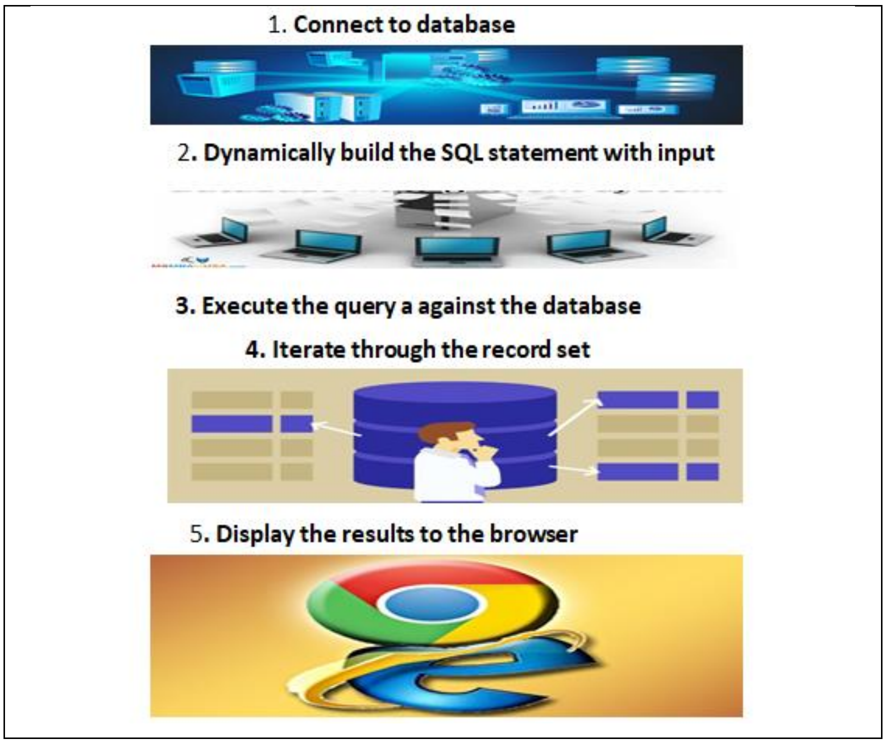

1. Connect to database
2. Dynamically build the SQL statement with input
3. Execute the query a against the database
4. Iterate through the record set
5. Display the results to the browser

Fig.7. steps of URL requested

9. ANATOMY OF SQLI

The query of SQL has arguments that tell the database system to return the required records. These arguments sent by the user in the form of field name, URL, etc. There are two stages of SQLI:

- Research Stage: The attacker submits different unexpected values as arguments, and observes the application's reaction to determine the attack.
- Attack Stage: Attackers provide an input argument to query of SQL to command part of it, while the database will execute that command as modified data by attacker.

10. TYPES OF SQL INJECTION

There are different types of SQLI attacks Fig.8. shows these types. The attacker extracts server data by exploiting the vulnerabilities of websites. All these attack types depend on retrieving databases on-conditions (T/F), errors on timing.

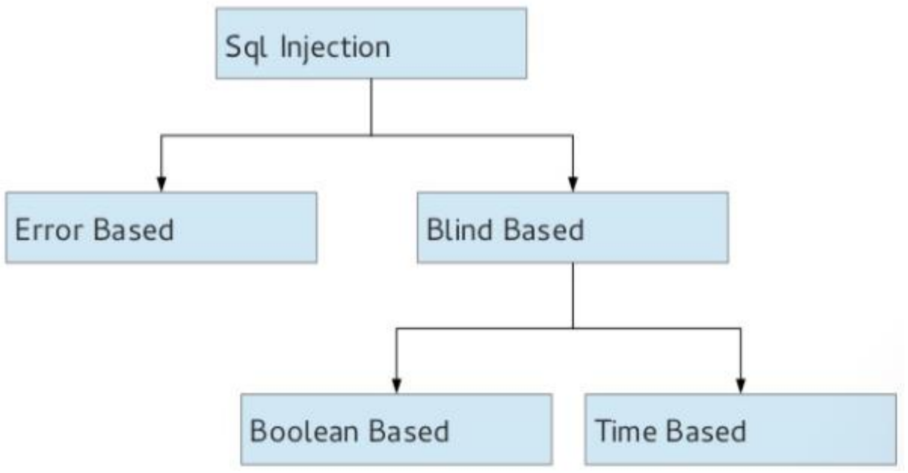


Fig.8. types of SQLI attack

Such as:

1. Error Based: In this type of SQLI the attacker can fetch details of database tables such as names and other contents of visible database errors, this can be identified by attacker on the production servers. To prevent such attacks, it should avoid displaying error database messages, which prevents the attacker from fetching this information, for e.g. the following request will return an error.[18]

https://example.com/index.php?id=1+and(select 1 from (select count (*),
cocat ((select (concat(database())) From information_schema.tables LIMIT 0,1),
Floor(rand(0)*2))x FROM information_schema.tables GROUP BY x)a)

Duplicate entry 'database1' for key 'group_key'
Request query

Error response

2. Blind Based divided into two parts:

a. Boolean Errors: In this type of attack the web browser does not display error through failing of SQL query which the attacker inserts a condition to be false to SQL query to extract data from database system. Such condition will be inserted to check the website vulnerable. If the website loaded normally that gives an indicator to the attacker that this website is good for SQLI attacking. To confirm this doubt, the attacker will send another request such as the following:

https://xyz.com/index.php?id=1+AND+1=2

This request is false; if a responded website does not work normally with that command, it means this is a vulnerable website to the SQLI attack.

b. Time-Based Query: The attacker here will instruct the database for a waiting period of time before responding. If the website rejects that and loads without waiting it means that the page is not vulnerable. The query of SQL here like Boolean attack but the difference is sleep function in a query. For e.g., if the sleep time=3 seconds, the attacker instructs the database to sleep for 3 seconds like the following:

https://xya.com/index.php?id=1+AND+IF(version()+LIKE+'8%',sleep(3),false)

To detect this attack a proposed tool was designed to connect all the information of user input with the sensitive code to find the place of danger.[19]

11. AST MODULE

AST is a mnemonic of "Abstract Syntax Trees". AST is a built-in module in Python. AST is a data structure that lets the user check, analyze, edit code easily and process the python abstract grammar. In this module, the source code of programming language is converted to an abstract tree. The node of the tree stands for different states in the source code. The most important class used in SQLIAD and derived from the base class ast.AST parser is "ast.NodeVisitor" class.

Class "ast.NodeVisitor" is used in parsing tree node to call visitor function for each node found, the return value forwarded to visit () function. If the returned value is "none", the node will remove from tree, otherwise, the node will be replaced by the return value.

In SQLIAD this article was used in detecting SQLI vulnerabilities in web applications using python3.7. This module contains a parsing method for constructing AST from the existing python code string. The standard library of AST is generated by (pip install ast) command. The source code of the web app is parsed to generate AST tree and CFG flow graph using the AST module. Fig.9. shows an example of using AST. AST traversed systematically to turn by CFG (Control Flow Graph) components into a standard control flow graph. Fig.10. represents some rules of the AST tool.

CFG is a directed graph where the specific place in the program represents a node, while the option of the code is represented by edges of the graph. Fig.11. illustrated the conversion from AST to expected CFG.

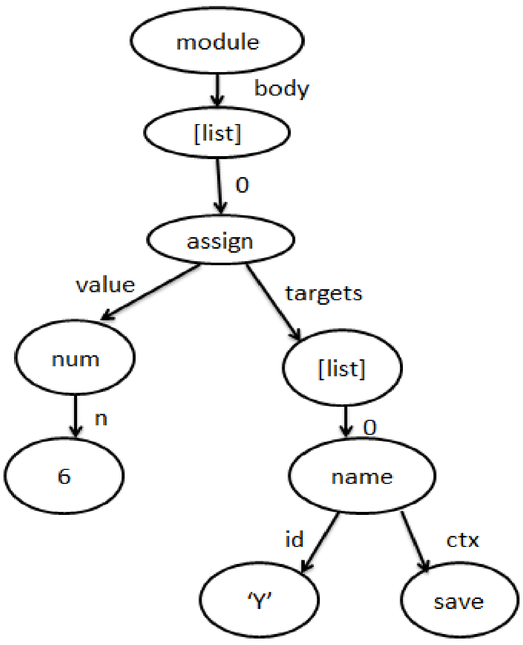
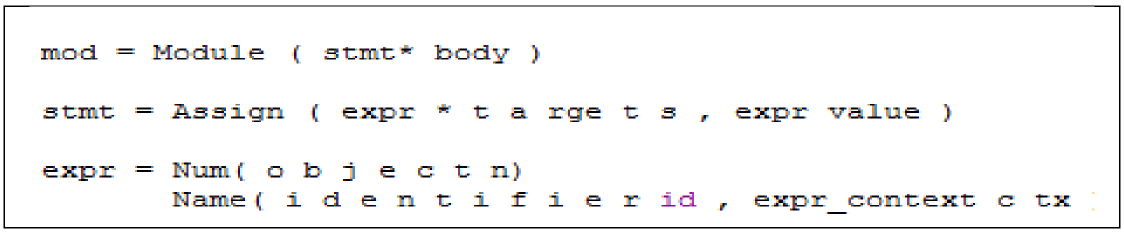


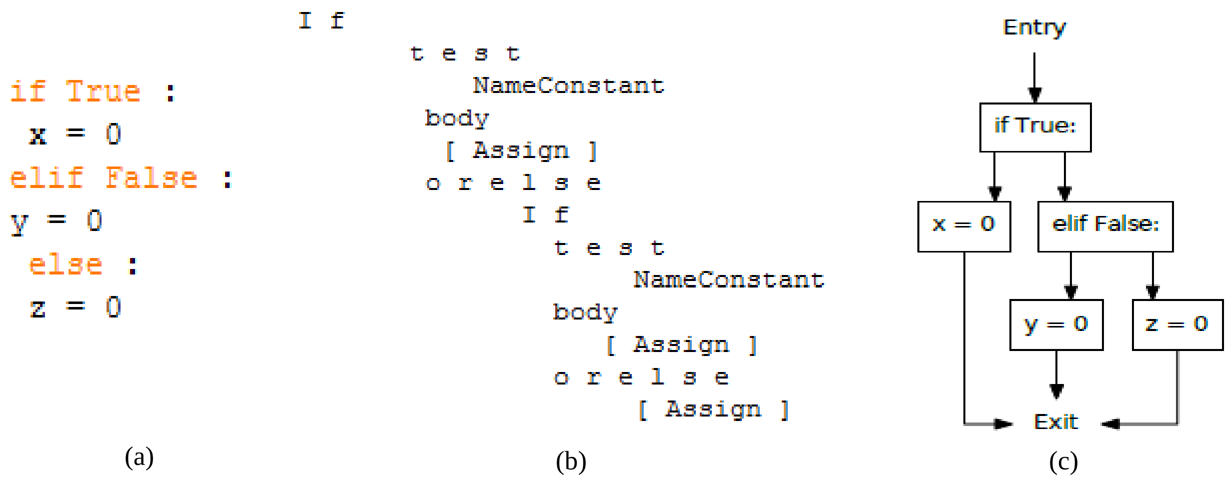
Fig.9. syntax tree for code (x=6) in python



```
mod = Module ( stmt* body )

stmt = Assign ( expr * t a rge t s , expr value )

expr = Num( o b j e c t n)
       Name( i d e n t i f i e r id , expr_context c tx
```

Fig.10. some of AST rules

In this example AST visualized as a tree, the first node contains a test node. The body and or else body is implemented by a nodes list. NameConstant nodes considered as conditions, but all statements considered as assign nodes. The predecessor given nodes denoted by "pred(n)" command while the successor's nodes denoted by "succ(n)" command. The CFG has two entries like any directed graph, first called "entry node", second called "exit node". The working with AST needs to represent the relations between expressions, operator, functions, and objects of python language.



```
if True :
 x = 0
elif False :
y = 0
 else :
 z = 0
```

(a)

```
I f
       t e s t
           NameConstant
        body
          [ Assign ]
        o r e l s e
              I f
                t e s t
                     NameConstant
                body
                   [ Assign ]
                o r e l s e
                     [ Assign ]
```

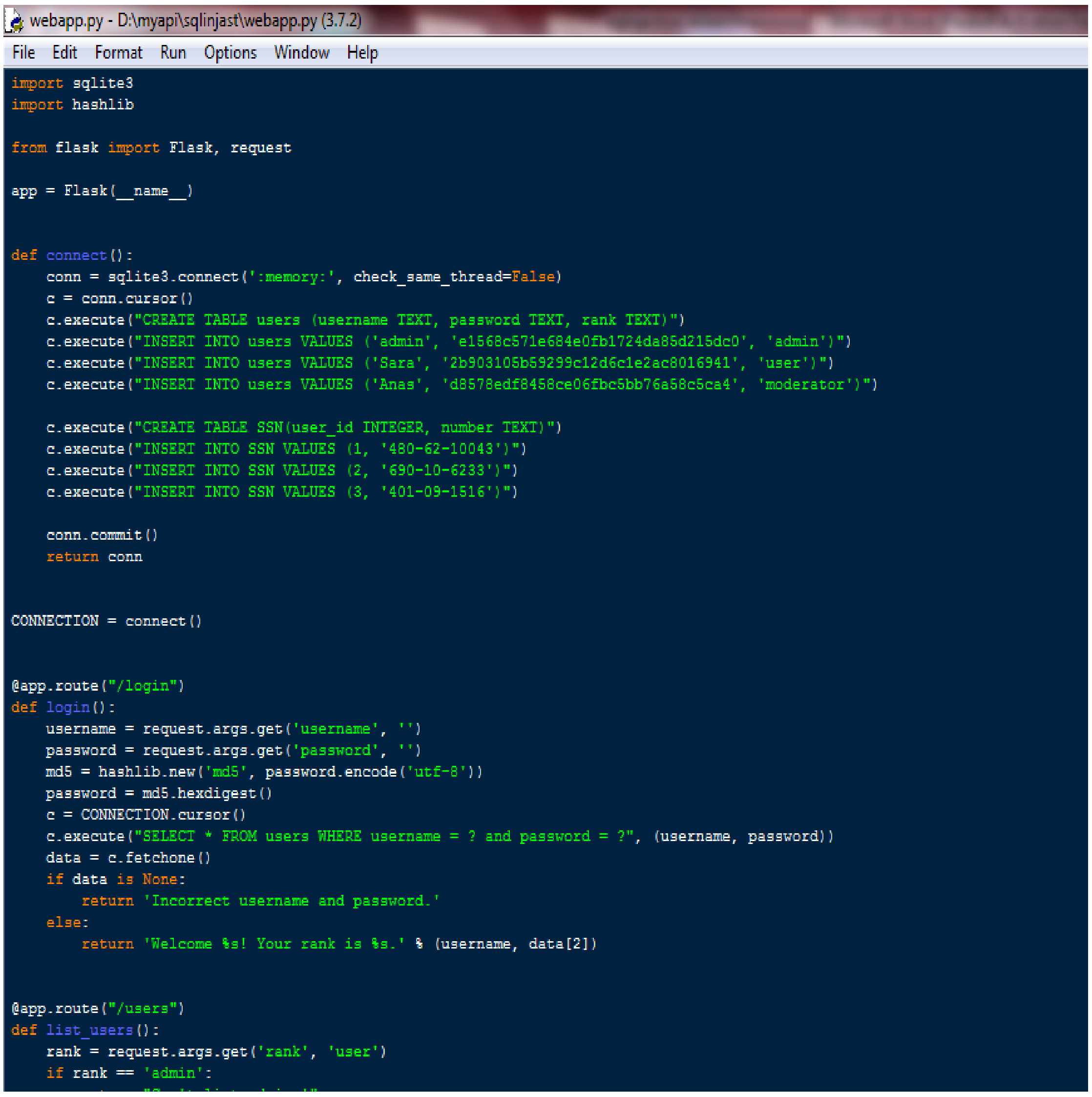
(b)

(c)

Fig.11. example code converted to AST and CFG form, (a) Example Code (b) AST form (c) expected CFG

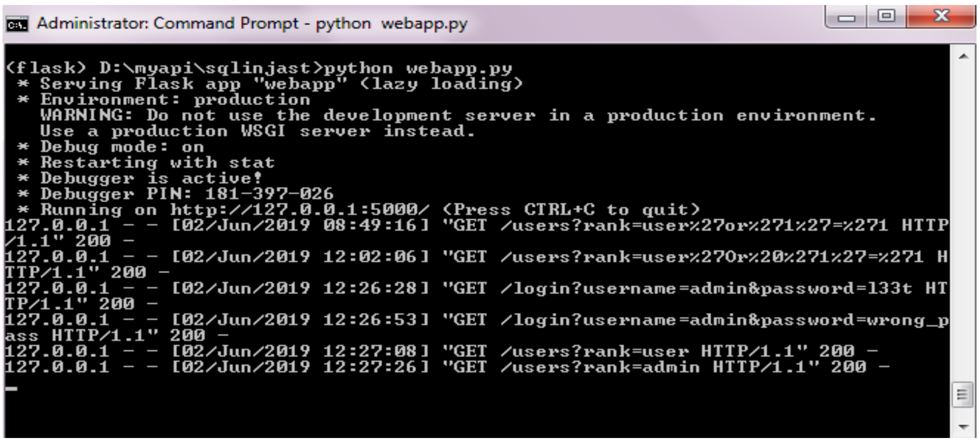
12. SQLIAD PROPOSED SYSTEM

To demonstrate SQLIAD at first it designed a simple web application using a flask tool in python (see Fig.12.). After setting up many tools like flask ("pip install flask"), sqlite3 ("pip install sqlite3"), Django ("pip install Django") and installing Virtual Environment using ("pip install virtual envwrapper-win") after that a virtual environment was created using ("mkvirtualenv myproject") and activate command used to activate a virtual environment. A command ("python webapp.py") used for browsing designed web applications. See Fig.13. The output of calling designed web application using explorer like in Fig.14.


webapp.py - D:\myapi\sqlinjast\webapp.py (3.7.2)
File Edit Format Run Options Window Help

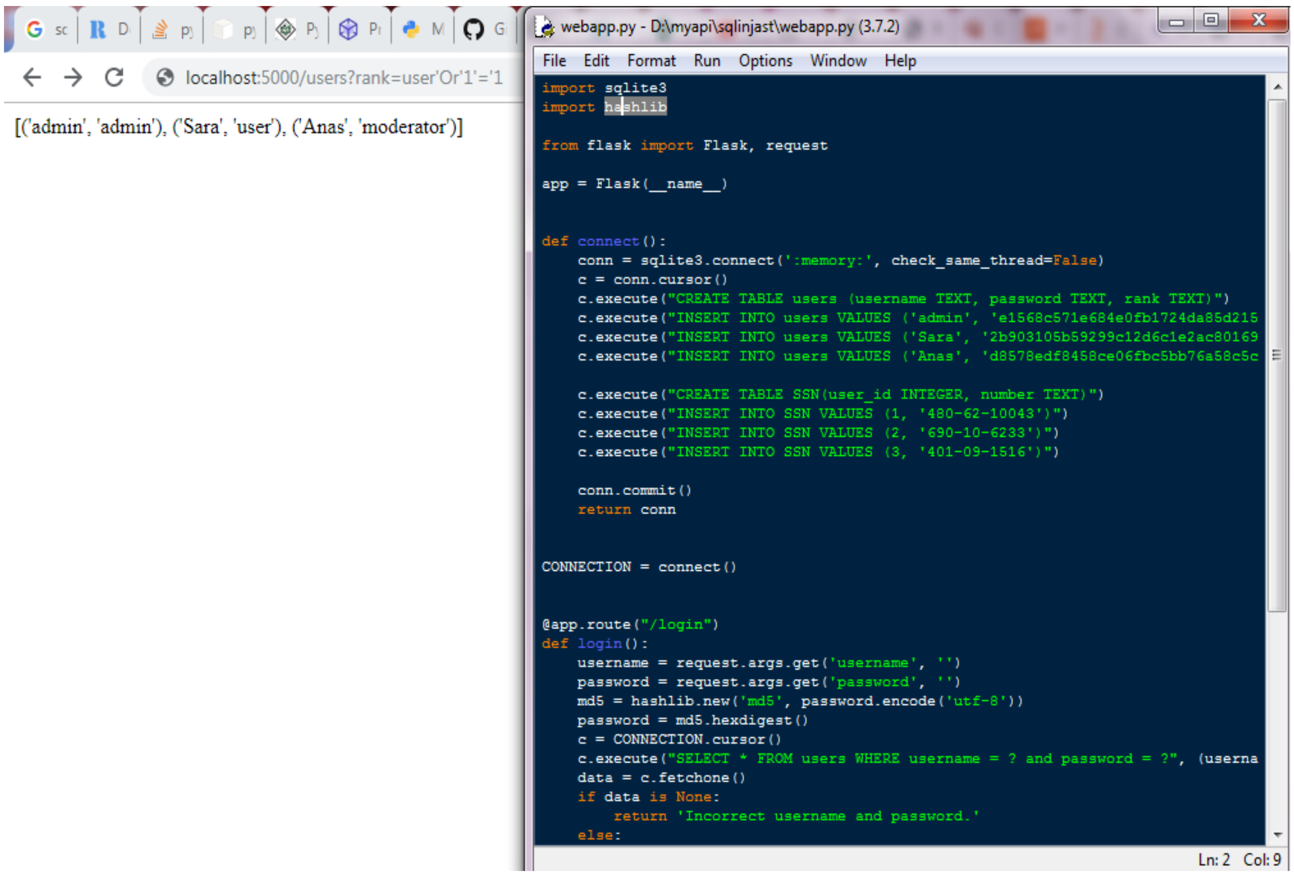
```python
import sqlite3
import hashlib

from flask import Flask, request

app = Flask(__name__)


def connect():
    conn = sqlite3.connect(':memory:', check_same_thread=False)
    c = conn.cursor()
    c.execute("CREATE TABLE users (username TEXT, password TEXT, rank TEXT)")
    c.execute("INSERT INTO users VALUES ('admin', 'e1568c571e684e0fb1724da85d215dc0', 'admin')")
    c.execute("INSERT INTO users VALUES ('Sara', '2b903105b59299c12d6c1e2ac8016941', 'user')")
    c.execute("INSERT INTO users VALUES ('Anas', 'd8578edf8458ce06fbc5bb76a58c5ca4', 'moderator')")

    c.execute("CREATE TABLE SSN(user_id INTEGER, number TEXT)")
    c.execute("INSERT INTO SSN VALUES (1, '480-62-10043')")
    c.execute("INSERT INTO SSN VALUES (2, '690-10-6233')")
    c.execute("INSERT INTO SSN VALUES (3, '401-09-1516')")

    conn.commit()
    return conn


CONNECTION = connect()


@app.route("/login")
def login():
    username = request.args.get('username', '')
    password = request.args.get('password', '')
    md5 = hashlib.new('md5', password.encode('utf-8'))
    password = md5.hexdigest()
    c = CONNECTION.cursor()
    c.execute("SELECT * FROM users WHERE username = ? and password = ?", (username, password))
    data = c.fetchone()
    if data is None:
        return 'Incorrect username and password.'
    else:
        return 'Welcome %s! Your rank is %s.' % (username, data[2])


@app.route("/users")
def list_users():
    rank = request.args.get('rank', 'user')
    if rank == 'admin':
```

Fig.12. web application code using python Flask tool



```
(flask) D:\myapi\sqlinjast>python webapp.py
 * Serving Flask app "webapp" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 181-397-026
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [02/Jun/2019 08:49:16] "GET /users?rank=user%27or%271%27=%271 HTTP
/1.1" 200 -
127.0.0.1 - - [02/Jun/2019 12:02:06] "GET /users?rank=user%27Or%20%271%27=%271 H
TTP/1.1" 200 -
127.0.0.1 - - [02/Jun/2019 12:26:28] "GET /login?username=admin&password=l33t HT
TP/1.1" 200 -
127.0.0.1 - - [02/Jun/2019 12:26:53] "GET /login?username=admin&password=wrong_p
ass HTTP/1.1" 200 -
127.0.0.1 - - [02/Jun/2019 12:27:08] "GET /users?rank=user HTTP/1.1" 200 -
127.0.0.1 - - [02/Jun/2019 12:27:26] "GET /users?rank=admin HTTP/1.1" 200 -
_
```

Fig.13. report of Http status code



```
localhost:5000/users?rank=user'Or'1'='1

[('admin', 'admin'), ('Sara', 'user'), ('Anas', 'moderator')]
```

webapp.py - D:\myapi\sqlinjast\webapp.py (3.7.2)

```python
import sqlite3
import hashlib

from flask import Flask, request

app = Flask(__name__)


def connect():
    conn = sqlite3.connect(':memory:', check_same_thread=False)
    c = conn.cursor()
    c.execute("CREATE TABLE users (username TEXT, password TEXT, rank TEXT)")
    c.execute("INSERT INTO users VALUES ('admin', 'e1568c571e684e0fb1724da85d215
    c.execute("INSERT INTO users VALUES ('Sara', '2b903105b59299c12d6c1e2ac80169
    c.execute("INSERT INTO users VALUES ('Anas', 'd8578edf8458ce06fbc5bb76a58c5c

    c.execute("CREATE TABLE SSN(user_id INTEGER, number TEXT)")
    c.execute("INSERT INTO SSN VALUES (1, '480-62-10043')")
    c.execute("INSERT INTO SSN VALUES (2, '690-10-6233')")
    c.execute("INSERT INTO SSN VALUES (3, '401-09-1516')")

    conn.commit()
    return conn


CONNECTION = connect()


@app.route("/login")
def login():
    username = request.args.get('username', '')
    password = request.args.get('password', '')
    md5 = hashlib.new('md5', password.encode('utf-8'))
    password = md5.hexdigest()
    c = CONNECTION.cursor()
    c.execute("SELECT * FROM users WHERE username = ? and password = ?", (userna
    data = c.fetchone()
    if data is None:
        return 'Incorrect username and password.'
    else:
```

Fig.14. displaying designed web application within Explorer

13. ALGORITHM OF SQLIAD

The algorithm of finding SQLI attacks (see Fig.15) in a web app is as follow:
Step1: Get URL to be source code.

Step2: Analyze source code using AST (Abstract Syntax Trees) module to get CFG ("<u>C</u>ontrol <u>F</u>low <u>G</u>raph ").
Step3: The output of step2 will input to Framework Adapter.
Step4: Analyze the result from step3.
Step5: All Vulnerabilities will be spit out and reported with the output.

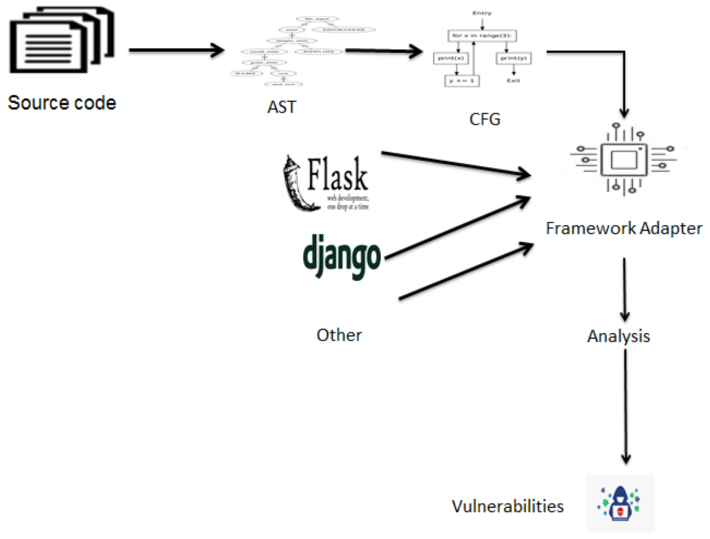


Fig.15. Algorithm of the proposed system to detect SQLI attack

The module AST is a base class for every AST node class, contains many subclasses like (ast_helper), ast.parse (f.read) , (generate_ast), (FrameworkAdaptor) (ast.NodeVisitor),…,etc. .Each class is responsible to do some functions. Python used the library of AST tools. The class (ast_helper) used for parsing (splitting) the source code of test web app into a list of tokens, to produce AST abstract tree. The AST tree is a collection of nodes linked together by edges, based on python language grammar. Sent to ast.parse(source.read()) to generate graph form CFG file, as an object tree. The (ast.NodeVisitor) track all the nodes of the AST tree to call visit (node) function for every tree node, to return a value. The NodeVisitor class used for tree scanning. The user input is called source code that parsed to AST to presents each file by it. Fig.16. shows a simple example of converting from AST to graph. By using CFG (Control Flow Graph) command the AST traversed to CFG form. The output of CFG sends to Flask adaptor (is an effective tool to test client requests without running Flask server). The fixed-point algorithm was applied to the resulting CFG to produce the resulting nodes. At the end of the invitation of CFG, the precise potential vulnerabilities information found by analysis. Through designing SQLIAD a Framework Adaptor class was used for converting data from one structure to others. It is contained with a list of CFGs. This adapter used Django and Flask in implementation. In Flask all web apps can be defined as decorator function. There was a Flask Adapter tool that goes to functions of all webpages for finding the defined function by @app.route() decorator for adding to the CFGs list. The outcome is a CFG list that covers all needed code application analysis. Fig.17. represents the code of FlaskAdapter. The analysis in step (4) used a flexible fixed-point algorithm applied on the CFG list, the fixed point means perform iterate analysis of dataflow until nothing changing, this statement is known as a fixed point.

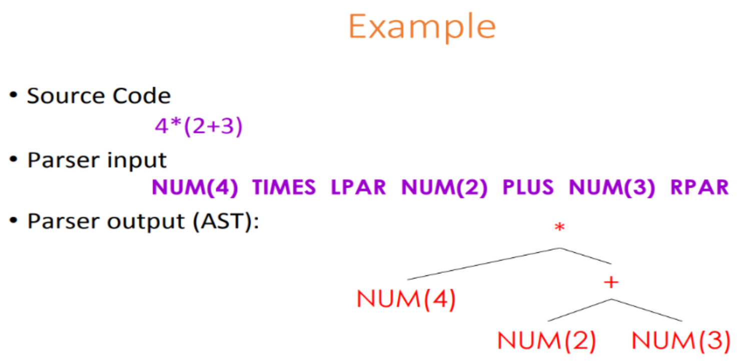
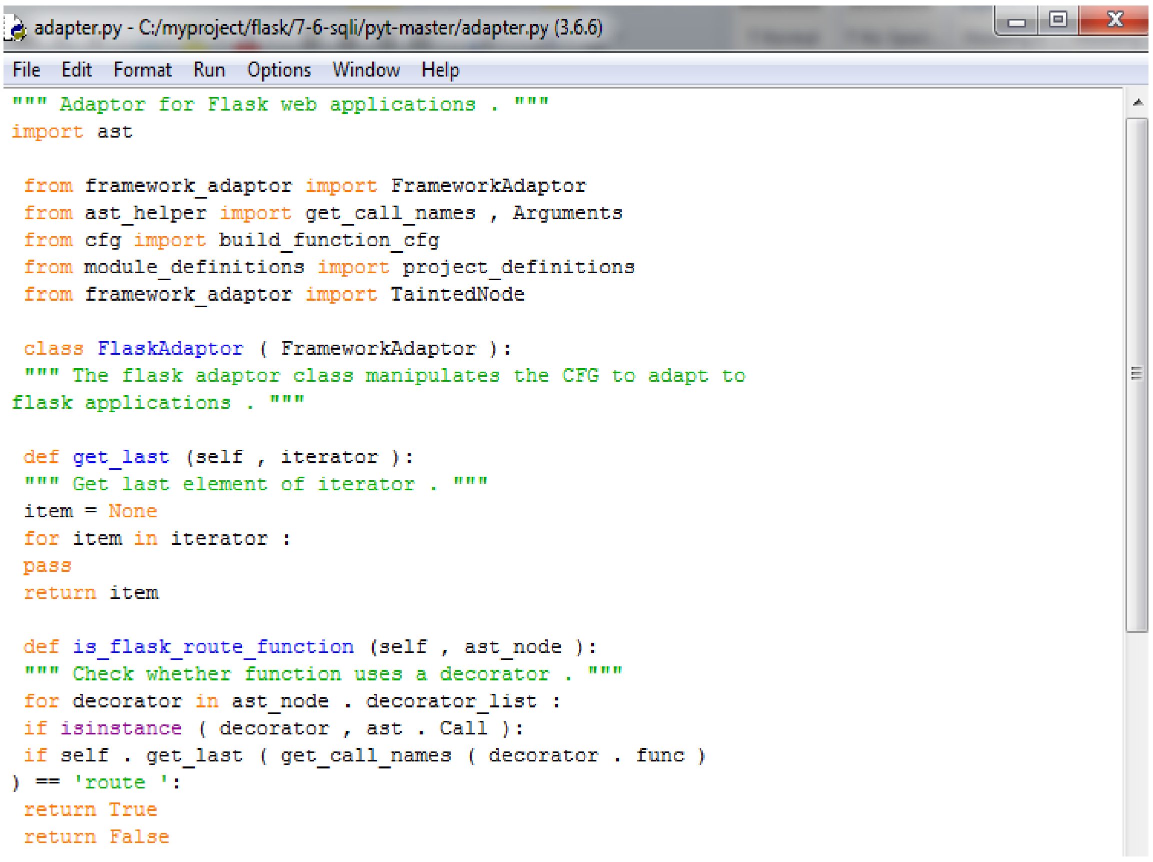


Fig.16. example of converting AST to graph


adapter.py - C:/myproject/flask/7-6-sqli/pyt-master/adapter.py (3.6.6)
File Edit Format Run Options Window Help

```
""" Adaptor for Flask web applications . """
import ast

 from framework_adaptor import FrameworkAdaptor
 from ast_helper import get_call_names , Arguments
 from cfg import build_function_cfg
 from module_definitions import project_definitions
 from framework_adaptor import TaintedNode

 class FlaskAdaptor ( FrameworkAdaptor ):
 """ The flask adaptor class manipulates the CFG to adapt to
flask applications . """

 def get_last (self , iterator ):
 """ Get last element of iterator . """
 item = None
 for item in iterator :
 pass
 return item

 def is_flask_route_function (self , ast_node ):
 """ Check whether function uses a decorator . """
 for decorator in ast_node . decorator_list :
 if isinstance ( decorator , ast . Call ):
 if self . get_last ( get_call_names ( decorator . func )
) == 'route ':
 return True
 return False
```

Fig.17. FlaskAdapter code

14. DETECT VULNERABILITIES IN A WEBSITE

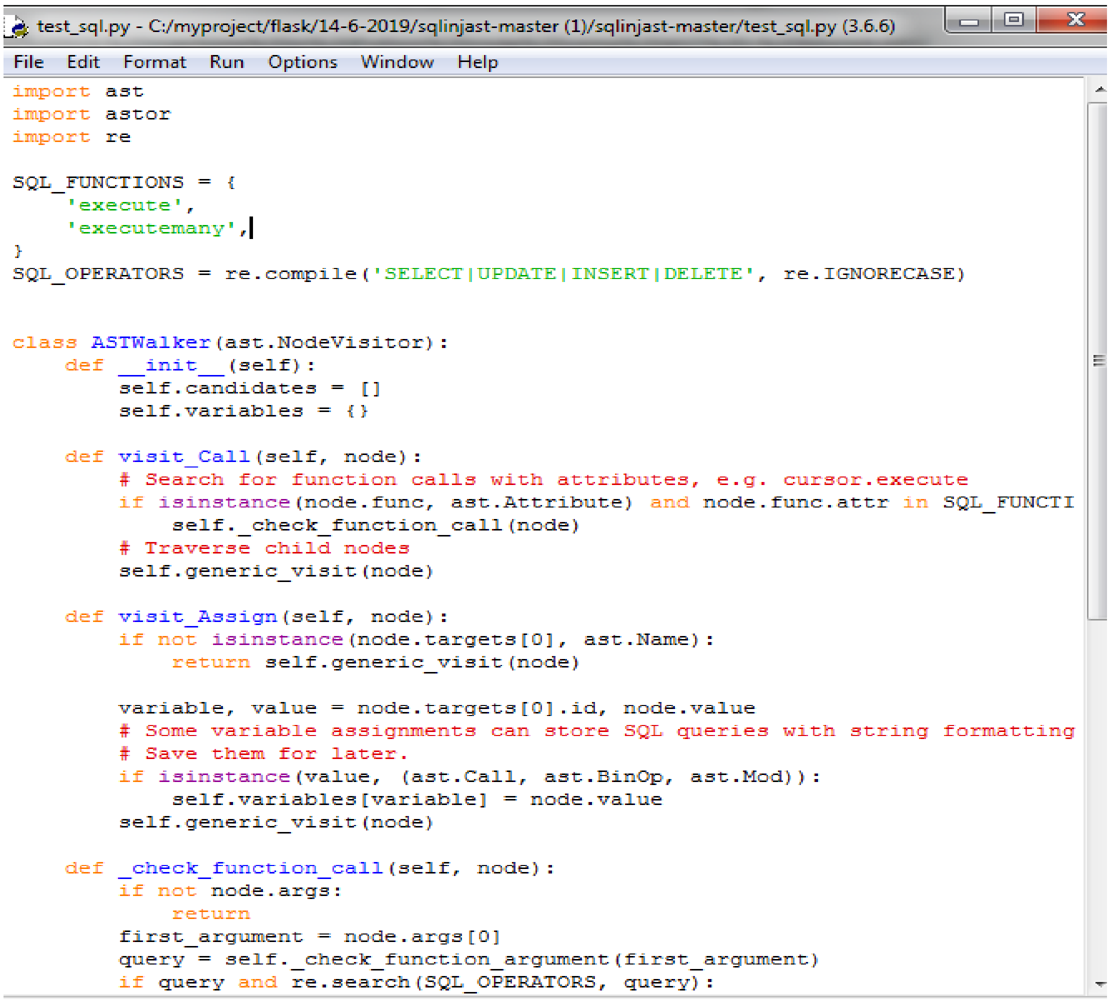
The SQL injection means that the input user reaches to the database query. A special tool was created to detect dangerous vulnerabilities by connecting information for sensitive code and user input. By building such a tool, a technique for static analyzing source code, to get some properties where used. The aim here is to track input and decide if it is harmful or not. The function that filters the input is called sanitizer.To represent this idea a python model was designed by setting up several tools that solved this problem. The python program is converted to another model using a special algorithm to implement the flow of the program.

The mistake that causes SQL Injection is using the formatting string in statement of SQL. The functions (execute, executemany) used in the SQLIAD app for searching for a string formatting to prevent SQL injection. There are (3) methods used in python for formatting string to avoid any injection code. Fig.18. explains the ways of python for formatting string.

```
c.execute("Select usernamerank FROM users Where rank ={'0'}".format(rank))
c.execute("SELECT username,rank FROM users WHERE rank='%s'" %rank)
c.execute(f"SELECT username,rank FROM users WHERE rank ='{rank}'")
```

Fig.18. formatting string in python

The "execute (query,Vars=None)" command in python used to execute the specified query. The Executemany(query,var_list) will execute database queries against all parameters in the vars-list sequence. (See Fig.19.).



```
test_sql.py - C:/myproject/flask/14-6-2019/sqlinjast-master (1)/sqlinjast-master/test_sql.py (3.6.6)
File  Edit  Format  Run  Options  Window  Help
```

```python
import ast
import astor
import re

SQL_FUNCTIONS = {
    'execute',
    'executemany',
}
SQL_OPERATORS = re.compile('SELECT|UPDATE|INSERT|DELETE', re.IGNORECASE)


class ASTWalker(ast.NodeVisitor):
    def __init__(self):
        self.candidates = []
        self.variables = {}

    def visit_Call(self, node):
        # Search for function calls with attributes, e.g. cursor.execute
        if isinstance(node.func, ast.Attribute) and node.func.attr in SQL_FUNCTI
            self._check_function_call(node)
        # Traverse child nodes
        self.generic_visit(node)

    def visit_Assign(self, node):
        if not isinstance(node.targets[0], ast.Name):
            return self.generic_visit(node)

        variable, value = node.targets[0].id, node.value
        # Some variable assignments can store SQL queries with string formatting
        # Save them for later.
        if isinstance(value, (ast.Call, ast.BinOp, ast.Mod)):
            self.variables[variable] = node.value
        self.generic_visit(node)

    def _check_function_call(self, node):
        if not node.args:
            return
        first_argument = node.args[0]
        query = self._check_function_argument(first_argument)
        if query and re.search(SQL_OPERATORS, query):
```

Fig.19. code of SQLIAD app in python

15. SQLIAD RESULTS

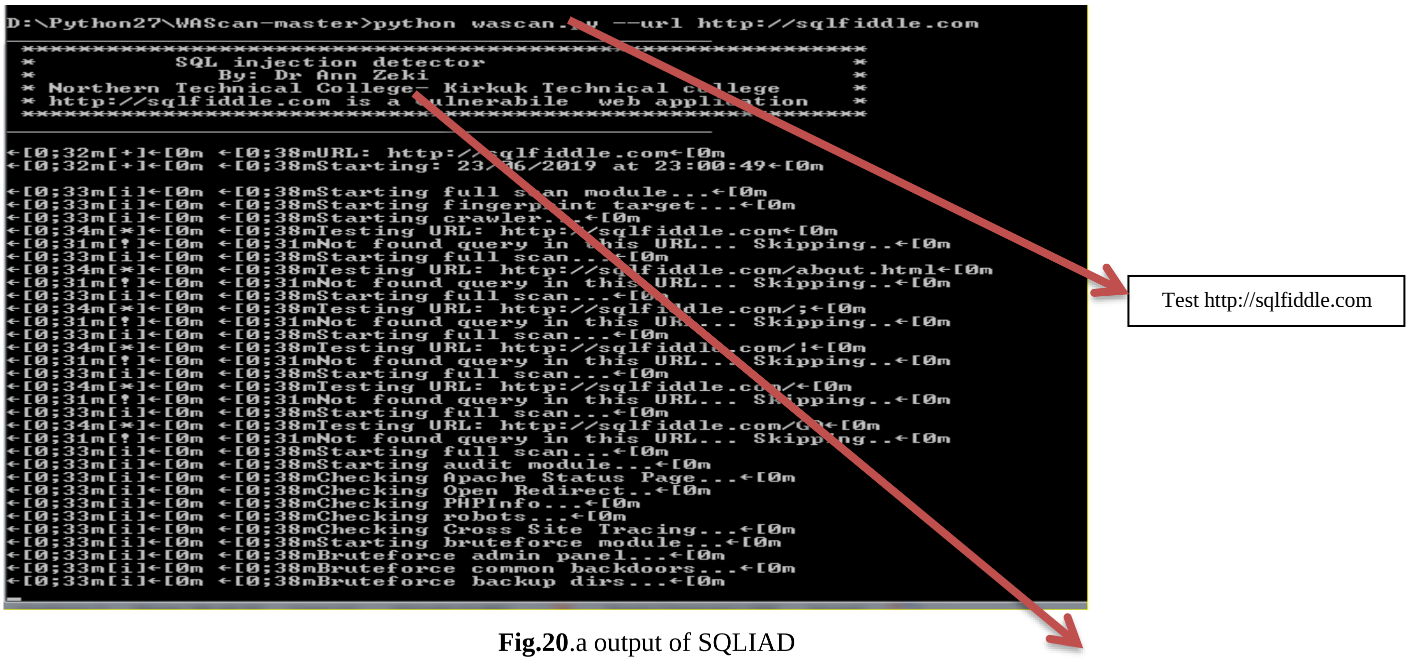
This section represents running the proposed system. SQLIAD is a command-line tool that needs one argument, file path name with some parameters to be analyzed like "python wascan.py webapp_path".

Because of the dangerous SQL injection attack, proposed developed tools were prepared (SQLIAD) to create a parser detector of SQL injection. These tools are smart and easy to use that takes a web app, analyze it and detect all type of SQL injection.

The steps of running SQLIAD module to get output by generating a parsing report of a web application to declare, if it is a vulnerability web application or not as follows:

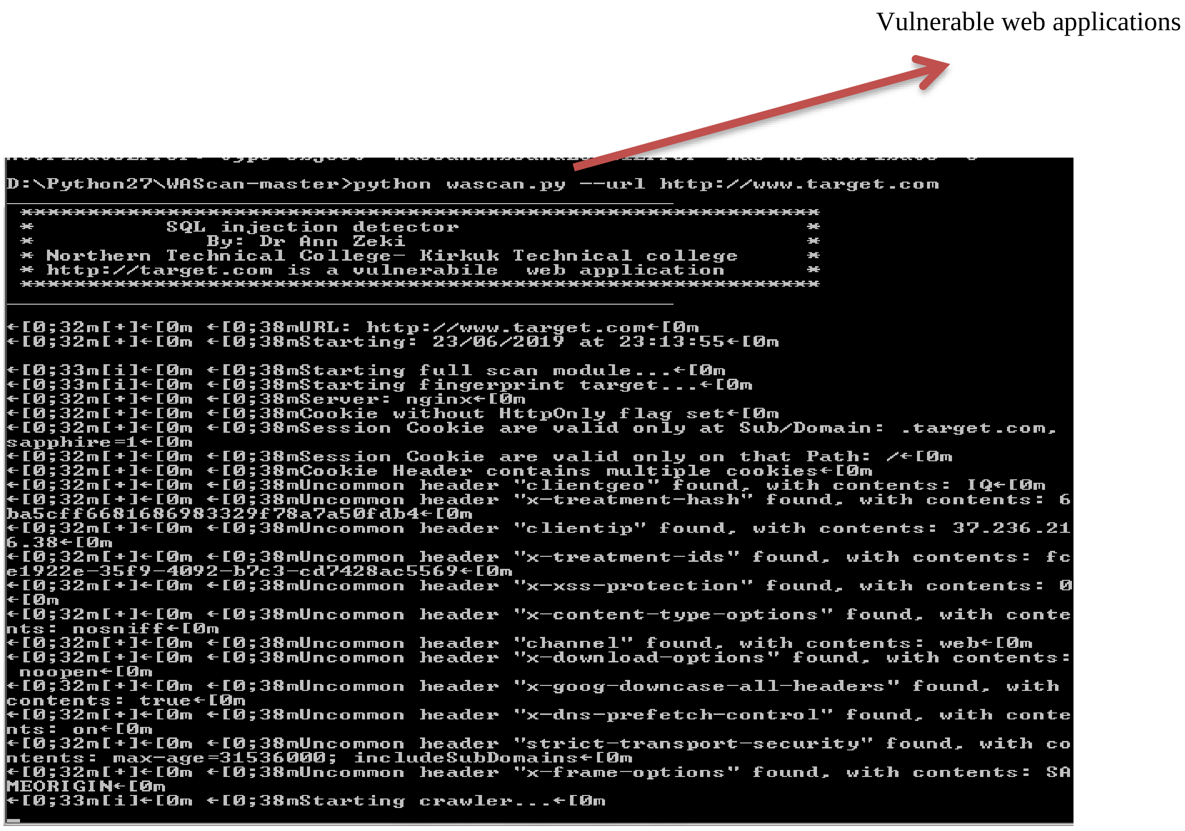
- Parsing command line with argument.
- Generating AST ("Abstract Syntax Tree").
- Generating CFG ("Control Flow Graph") from generated AST.
- The generated CFG pass to Framework Adapter, which mark the argument functions as tainted sources.
- Analysis and modify reaching definition by knowing the reaching definitions.
- Finding the vulnerabilities by vision how and where sources reach.
- Remove all suspicious vulnerabilities.
- Output analysis report into the console.

The output of SQLIAD like in Fig.20a Fig.20b illustrated reports of testing http://sqlfiddle.com , http://target.com vulnerable web applications.



```
D:\Python27\WAScan-master>python wascan.py --url http://sqlfiddle.com
 ***********************************************************************
 *              SQL injection detector                                 *
 *                 By: Dr Ann Zeki                                     *
 * Northern Technical College- Kirkuk Technical college                *
 * http://sqlfiddle.com is a vulnerabile  web application              *
 ***********************************************************************

←[0;32m[+]←[0m ←[0;38mURL: http://sqlfiddle.com←[0m
←[0;32m[+]←[0m ←[0;38mStarting: 23/06/2019 at 23:00:49←[0m

←[0;33m[i]←[0m ←[0;38mStarting full scan module...←[0m
←[0;33m[i]←[0m ←[0;38mStarting fingerprint target...←[0m
←[0;33m[i]←[0m ←[0;38mStarting crawler...←[0m
←[0;34m[*]←[0m ←[0;38mTesting URL: http://sqlfiddle.com←[0m
←[0;31m[!]←[0m ←[0;31mNot found query in this URL... Skipping..←[0m
←[0;33m[i]←[0m ←[0;38mStarting full scan...←[0m
←[0;34m[*]←[0m ←[0;38mTesting URL: http://sqlfiddle.com/about.html←[0m
←[0;31m[!]←[0m ←[0;31mNot found query in this URL... Skipping..←[0m
←[0;33m[i]←[0m ←[0;38mStarting full scan...←[0m
←[0;34m[*]←[0m ←[0;38mTesting URL: http://sqlfiddle.com/;←[0m
←[0;31m[!]←[0m ←[0;31mNot found query in this URL... Skipping..←[0m
←[0;33m[i]←[0m ←[0;38mStarting full scan...←[0m
←[0;34m[*]←[0m ←[0;38mTesting URL: http://sqlfiddle.com/!←[0m
←[0;31m[!]←[0m ←[0;31mNot found query in this URL... Skipping..←[0m
←[0;33m[i]←[0m ←[0;38mStarting full scan...←[0m
←[0;34m[*]←[0m ←[0;38mTesting URL: http://sqlfiddle.com/←[0m
←[0;31m[!]←[0m ←[0;31mNot found query in this URL... Skipping..←[0m
←[0;33m[i]←[0m ←[0;38mStarting full scan...←[0m
←[0;34m[*]←[0m ←[0;38mTesting URL: http://sqlfiddle.com/GO←[0m
←[0;31m[!]←[0m ←[0;31mNot found query in this URL... Skipping..←[0m
←[0;33m[i]←[0m ←[0;38mStarting full scan...←[0m
←[0;33m[i]←[0m ←[0;38mStarting audit module...←[0m
←[0;33m[i]←[0m ←[0;38mChecking Apache Status Page...←[0m
←[0;33m[i]←[0m ←[0;38mChecking Open Redirect..←[0m
←[0;33m[i]←[0m ←[0;38mChecking PHPInfo...←[0m
←[0;33m[i]←[0m ←[0;38mChecking robots...←[0m
←[0;33m[i]←[0m ←[0;38mChecking Cross Site Tracing...←[0m
←[0;33m[i]←[0m ←[0;38mStarting bruteforce module...←[0m
←[0;33m[i]←[0m ←[0;38mBruteforce admin panel...←[0m
←[0;33m[i]←[0m ←[0;38mBruteforce common backdoors...←[0m
←[0;33m[i]←[0m ←[0;38mBruteforce backup dirs...←[0m
```

Fig.20.a output of SQLIAD



```
D:\Python27\WAScan-master>python wascan.py --url http://www.target.com
 *******************************************************************
 *            SQL injection detector                               *
 *               By: Dr Ann Zeki                                   *
 * Northern Technical College- Kirkuk Technical college            *
 * http://target.com is a vulnerabile  web application             *
 *******************************************************************

←[0;32m[+]←[0m ←[0;38mURL: http://www.target.com←[0m
←[0;32m[+]←[0m ←[0;38mStarting: 23/06/2019 at 23:13:55←[0m

←[0;33m[i]←[0m ←[0;38mStarting full scan module...←[0m
←[0;33m[i]←[0m ←[0;38mStarting fingerprint target...←[0m
←[0;32m[+]←[0m ←[0;38mServer: nginx←[0m
←[0;32m[+]←[0m ←[0;38mCookie without HttpOnly flag set←[0m
←[0;32m[+]←[0m ←[0;38mSession Cookie are valid only at Sub/Domain: .target.com,
sapphire=1←[0m
←[0;32m[+]←[0m ←[0;38mSession Cookie are valid only on that Path: /←[0m
←[0;32m[+]←[0m ←[0;38mCookie Header contains multiple cookies←[0m
←[0;32m[+]←[0m ←[0;38mUncommon header "clientgeo" found, with contents: IQ←[0m
←[0;32m[+]←[0m ←[0;38mUncommon header "x-treatment-hash" found, with contents: 6
ba5cff66816869833329f78a7a50fdb4←[0m
←[0;32m[+]←[0m ←[0;38mUncommon header "clientip" found, with contents: 37.236.21
6.38←[0m
←[0;32m[+]←[0m ←[0;38mUncommon header "x-treatment-ids" found, with contents: fc
e1922e-35f9-4092-b7c3-cd7428ac5569←[0m
←[0;32m[+]←[0m ←[0;38mUncommon header "x-xss-protection" found, with contents: 0
←[0m
←[0;32m[+]←[0m ←[0;38mUncommon header "x-content-type-options" found, with conte
nts: nosniff←[0m
←[0;32m[+]←[0m ←[0;38mUncommon header "channel" found, with contents: web←[0m
←[0;32m[+]←[0m ←[0;38mUncommon header "x-download-options" found, with contents:
 noopen←[0m
←[0;32m[+]←[0m ←[0;38mUncommon header "x-goog-downcase-all-headers" found, with
contents: true←[0m
←[0;32m[+]←[0m ←[0;38mUncommon header "x-dns-prefetch-control" found, with conte
nts: on←[0m
←[0;32m[+]←[0m ←[0;38mUncommon header "strict-transport-security" found, with co
ntents: max-age=31536000; includeSubDomains←[0m
←[0;32m[+]←[0m ←[0;38mUncommon header "x-frame-options" found, with contents: SA
MEORIGIN←[0m
←[0;33m[i]←[0m ←[0;38mStarting crawler...←[0m
```

Fig.20.b output of SQLIAD

After analyzing http://Facebook.com , SQLIAD declares this page is a vulnerable web application with a blind SQL injection. See Fig.21.

16. CONCLUSION

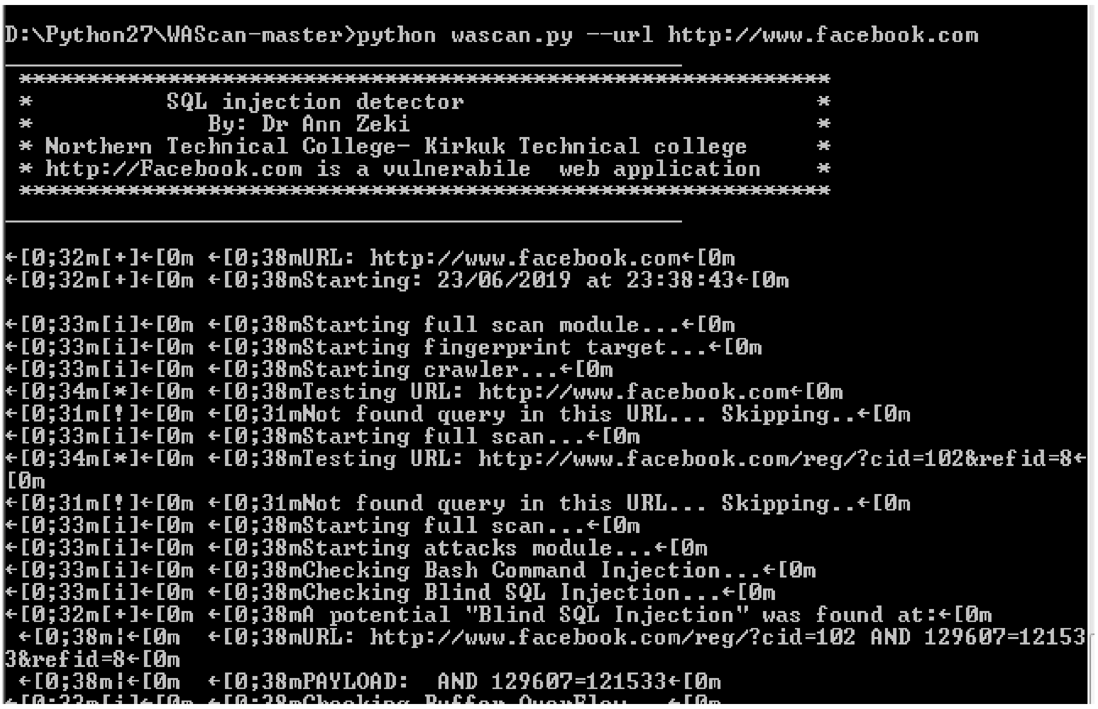
SQL injection is a modern attack on a web application; it is not commonly known to the general world. It takes a lot of time for understanding what is it, and how it can be detected and protecttion from it.

Because of the wide vulnerability of web applications and the problems caused by it, an SQLIAD tool was prepared to detect and protect the web application from any attack like SQL injection. A python3.7 programming language with a Flask tool was used in designing SQLIAD. SQLIAD were used to test and analyze many different web applications.

SQLIAD was evaluated, to be able in finding all types of SQLI attack and it is flexible in updating and easy to apply in any web application.

17. ACKNOWLEDGMENTS

This work is impossible without the help of OWASP group, YouTube, and many helpers of python documents. I am grateful to all people who participated in designing and assisted in writing this paper. I would also like to thank the members of the Tikrit Journal for their effort.



```
D:\Python27\WAScan-master>python wascan.py --url http://www.facebook.com
 ____________________________________________________________________________
 ************************************************************
 *          SQL injection detector                          *
 *              By: Dr Ann Zeki                             *
 * Northern Technical College- Kirkuk Technical college     *
 * http://Facebook.com is a vulnerabile  web application    *
 ************************************************************
 ____________________________________________________________________________

←[0;32m[+]←[0m ←[0;38mURL: http://www.facebook.com←[0m
←[0;32m[+]←[0m ←[0;38mStarting: 23/06/2019 at 23:38:43←[0m

←[0;33m[i]←[0m ←[0;38mStarting full scan module...←[0m
←[0;33m[i]←[0m ←[0;38mStarting fingerprint target...←[0m
←[0;33m[i]←[0m ←[0;38mStarting crawler...←[0m
←[0;34m[*]←[0m ←[0;38mTesting URL: http://www.facebook.com←[0m
←[0;31m[!]←[0m ←[0;31mNot found query in this URL... Skipping..←[0m
←[0;33m[i]←[0m ←[0;38mStarting full scan...←[0m
←[0;34m[*]←[0m ←[0;38mTesting URL: http://www.facebook.com/reg/?cid=102&refid=8←
[0m
←[0;31m[!]←[0m ←[0;31mNot found query in this URL... Skipping..←[0m
←[0;33m[i]←[0m ←[0;38mStarting full scan...←[0m
←[0;33m[i]←[0m ←[0;38mStarting attacks module...←[0m
←[0;33m[i]←[0m ←[0;38mChecking Bash Command Injection...←[0m
←[0;33m[i]←[0m ←[0;38mChecking Blind SQL Injection...←[0m
←[0;32m[+]←[0m ←[0;38mA potential "Blind SQL Injection" was found at:←[0m
 ←[0;38m¦←[0m  ←[0;38mURL: http://www.facebook.com/reg/?cid=102 AND 129607=12153
3&refid=8←[0m
 ←[0;38m¦←[0m  ←[0;38mPAYLOAD:  AND 129607=121533←[0m
←[0;33m[i]←[0m ←[0;38mChecking Buffer OverFlow...←[0m
```

Fig.21. analyzing report of http://Facebook.com

REFERENCES

[1] Huang, Yao-Wen, et al. Securing web application code by static analysis and runtime protection Proceedings of *the 13th international conference on World Wide Web*. ACM ,2004.

[2] Boyd, Stephen W., and Angelos D. Keromytis. Preventing SQL injection attacks. *International Conference on Applied Cryptography and Network Security*. Springer Berlin Heidelberg,2004.

[3] Kemalis, Konstantinos, and Theodores Tzouramanis. SQL-IDS: a specification-based approach for SQL-injection detection. *Proceedings of the ACM*, 2008.

[4] Junjin, Mei. An approach for SQL injection vulnerability detection. ITNG'09. *International Sixth Conference on. IEEE*. Justin Clarke. (2009). SQL Injection attacks and defense. Burlington, Mass:Syngress Pub,2009.

[5] J. Kim. Injection Attack Detection Using the Removal of SQL Query Attribute Values. *Proc. of the International Conference on Information Science and Applications (ICISA)*, Jeju Island, Korea,2011.

[6] Ryan Turner, *Python Programming book*, Kindle Edition. Sqlmap tutorial for beginners hacking with SQL injection. http://www.binarytides.com/sqlmap-hacking-tutorial, 2018.

[7] Alfantookh, Abdulkader. An automated universal server-level solution for SQL injection security flaw. *International Conference on Electrical and Computer Engineering (ICEEC'04)*, 2004.

[8] Chen xiao bing, Zhang Han yu, Luo Liming. Research on the technology of SQL injection attacks and detection. *Comput Eng Appl.*,2007.

[9] Fu, Xiang, et al. A static analysis framework for detecting SQL Injection vulnerabilities. *Computer Software and Applications Conference.31st Annual International*.Vol. 1**.** IEEE,2007

[10] Gerand Swinnen, *Teach python3 book*, first edition, 2013.

[11] Grinberg, Miguel, Flask web development: developing web applications with python. *O'Reilly Media Inc*, 2014.

[12] Halfond, William G., Jeremy Viegas, and Alessandro Orso. A classification of SQL-injection attacks and countermeasures. *Proceedings of the IEEE International Symposium on Secure Software Engineering*. Vol**. 1.** IEEE, 2006.

[13] *MATEC Web of Conference* 173.https://doi.org/10.1051/matecconf/2018173, 2018.

[14] Suraj Natarajan, Melody Moh Recommending News Based on Hybrid User Profile Popularity Trends and Location, *CTS*, 2016.

[15] Tao Han. Research on SQL injection detection method based on analytic tree - Harbin *Institute of Technology*, 2013.

[16] The OWASP Top Ten Project. https://www.owasp.org/index.php/Category: OWASP_Top_Ten_Project, 2018.

[17] Tian Y J, Zhao Z M, Zhang H C. Second-order SQL Injection Attack Defense Model, Netinfo Security, 2014.

[18] Valeur, Fredrik,Darren Mutz, and Giovanni Vigna. A learning-based approach to the detection of SQL attacks. *International Conference on Detection of Intrusions and Malware*, and Vulnerability Assessment. Springer Berlin Heidelberg ,2005.

[19] Wahid Rajeh,Alshreef Abed.Anovel three-tier SQLi detection and mitigation scheme for cloud environments, *ICECIS*, 33-37, 2017.

[20] WuShao Hua, Chen Shu bao.Web Attack Detection Method Based on Support Vector Machine Computer ,2015.

[21] Zhuang Chen, Min Guo, Lin zhou. Research on SQL injection detection technology based on SVM,2018.