VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Jnana Sangama, Belagavi-590018, Karnataka

A PROJECT REPORT ON

“DETECTING AND REMOVING WEB APPLICATION VULNERABILITIES WITH STATIC ANALYSIS AND DATA MINING”

Submitted in partial fulfillment of the requirements for the award of the degree of
Bachelor of Engineering
in
Computer Science & Engineering

Submitted by

KAILASH (3GN18CS038)

Under the guidance of

Prof. ASHA P

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GURU NANAK DEV ENGINEERING COLLEGE
BIDAR-585403, KARNATAKA
2021-2022

VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI

GURU NANAK DEV ENGINEERING COLLEGE

BIDAR-585403

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the main project report entitled **"DETECTING AND REMOVING WEB APPLICATION VULNERABILITIES WITH STATIC ANALYSIS AND DATA MINING"** is a bonafide work carried out by **KAILASH (3GN18CS038)** in practical fulfillment of the requirements for the award of Bachelor of Engineering in **COMPUTER SCIENCE AND ENGINEERING** of the **GURU NANAK DEV ENGINEERING COLLEGE, BIDAR** during the year 2021-2022. It is certified that all the corrections/suggestions indicated for the main project had been completed satisfactorily.

Prof. ASHA P (GUIDE)	Dr. DAYANAND JAMKHANDI (HOD, CSE)	Dr. DHANANJAY MAKTEDAR (PRINCIPAL, GNDEC)

EXTERNAL VIVA:

EXAMINERS: 1) ___________________________ 2) ___________________________

ACKNOWLEDGEMENT

I would like to express my deep gratitude to our principal **Dr. DHANANJAY MAKTEDAR** of Guru Nanak Dev Engineering College, Bidar for his inspiration and support by providing good facilities to complete the main project.

My sincere thanks to **Dr. DAYANANAD J** Hod of computer science and Engineering for his whole hearted support in completion of this main project.

I am also thankful to the seminar Coordinator **Prof. ANIL KULKARNI** for their support and advise throughout the course of development of the main project.

I am in debited to my seminar guide **Prof. ASHA P** for guiding and giving timely advices and suggestions in the successful completion of the main project.

I thank to all the staff members for their support to completion of my main project, finally I express my gratefulness to all those who knowingly or unknowingly helped me in the successful completion of this main project.

KAILASH (3GN18CS038)

ABSTRACT

Although a large research effort has been going on for more than a decade, the security of web applications continues to be a challenging problem. An important part of that problem derives from vulnerable source code, often written in unsafe languages like PHP. Source code static analysis tools are a solution to find vulnerabilities, but they tend to generate false positives and require considerable effort for programmers to manually fix the code. We explore the use of a combination of methods to discover vulnerabilities in source code with less false positives. We combine taint analysis, which finds candidate vulnerabilities, with data mining, in order to predict the existence of false positives. This approach brings together two approaches that are apparently orthogonal: humans coding the knowledge about vulnerabilities (for taint analysis) versus automatically obtaining that knowledge (with machine learning, for data mining). Given this enhanced form of detection, we propose doing automatic code correction by inserting fixes in the source code. Our approach was implemented in the WAP tool and an experimental evaluation was performed with a large set of PHP applications. Our tool found 388 vulnerabilities in 1.4 million lines of code. Its accuracy and precision were approximately 5% better than PhpMinerII's and 45% better than Pixy's.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction to Project

Since its appearance in the early 1990s, the Web evolved from a platform to access text and other media to a framework for running complex web applications. These applications appear in many forms, from small home-made to large-scale commercial services (e.g., Google Docs, Twitter, Facebook). However, web applications have been plagued with security problems. For example, a recent report indicates an increase of web attacks of around 33% in 2012 [34]. Arguably, a reason for the insecurity of web applications is that many programmers lack appropriate knowledge about secure coding, so they leave applications with flaws. However, the mechanisms for web application security fall in two extremes. On one hand, there are techniques that put the programmer aside, e.g., web application firewalls and other runtime protections [12], [24], [37]. On the other hand, there are techniques that discover vulnerabilities but put the burden of removing them on the programmer, e.g., black-box testing [2], [4], [14] and static analysis [15], [17], [27].

The project explores an approach for automatically protecting web applications while keeping the programmer in the loop. The approach consists in analyzing the web application source code searching for input validation vulnerabilities and inserting fixes in the same code to correct these flaws. The programmer is kept in the loop by being allowed to understand where the vulnerabilities were found and how they were corrected. This contributes directly for the security of web applications by removing vulnerabilities, and indirectly by letting the programmers learn from their mistakes. This last aspect is enabled by inserting fixes that follow common security coding practices, so programmers can learn these practices by seeing the vulnerabilities and how they were removed.

1.2 Purpose of the Project

We explore the use of a novel combination of methods to detect this type of vulnerabilities: static analysis and data mining. Static analysis is an effective mechanisms to find vulnerabilities in source code, but tends to report many false positives (non-vulnerabilities) due to its undecidability [18]. This problem is particularly difficult with languages such as PHP that are weakly typed and not formally specified [7]. Therefore, we complement a form of static analysis, taint analysis, with the use of data mining to predict the existence of false positives. This solution combines two apparently opposite approaches: humans coding the knowledge about vulnerabilities (for taint analysis) versus automatically obtaining that knowledge (with supervised machine learning supporting data mining).

To predict the existence of false positives we introduce the novel idea of assessing if the vulnerabilities detected are false positives using data mining. To do this assessment, we measure attributes of the code that we observed to be associated with the presence of false positives, and use a combination of the three top-ranking classifiers to flag every vulnerability as false positive or not. We explore the use of several classifiers:

ID3, C4.5/J48, Random Forest, Random Tree, K-NN, Naïve Bayes, Bayes Net, MLP, SVM, and Logistic Regression. Moreover, for every vulnerability classified as false positive, we use an induction rule classifier to show which attributes are associated with it. We explore the JRip, PART, Prism and Ridor induction rule classifiers for this goal. Classifiers are automatically configured using machine learning based on labeled vulnerability data.

Ensuring that the code correction is done correctly requires assessing that the vulnerabilities are removed and that the correct behavior of the application is not modified by the fixes. We propose using program mutation and regression testing to confirm, respectively, that the fixes do the function to what they are programmed to (blocking malicious inputs) and that the application remains working as expected (with benign inputs). Notice that we do not claim that our approach is able to correct any vulnerability, or to detect it, only the input validation vulnerabilities it is programmed to deal with.

1.3 Existing System

The WAP tool is based on the analyses of the source code in the Intrusion Detection Systems. The IDS are segregated into two categories, the knowledge-based intrusion detection systems hold the database of vulnerabilities or attacks created manually, the behaviour based systems learn about attacks using labelled datasets automatically. WAP is used to detect different classes of input vulnerabilities in the source code: XSS (Cross Site Scripting), SQL Injections, Path Traversal, OS command injection and a few more. The tool contains knowledge about these vulnerabilities through manual coding. Using this taint analysis, we check if the inputs can reach a sensitive function without validation or sanitization.Hereby is an example XSS, its input points.

Entry Points	Sensitive Sinks	Sanitisation Functions
$_POST $_GET $_REQUEST $_COOKIE HTTP_POST_VARS HTTP_GET_VARS HTTP_COOKIE_VARS $_FILES $_SERVERS	Print echo die print f exit error file_put_content sfprintffile_get_contents fgetsfg etc fscanf	Htmlentitieshtmlspecialcharsstrip_tags urlencode

Table 1. Entry points, Sensitive sinks and sanitizationfunctions for XSS vulnerabilities.

The primary work of the taint analyser is to parse the source code and build an abstract syntax tree (AST). After building the AST, the taint analyzer generates the tree describing candidate vulnerable control flow paths. For each vulnerability class, the taint analysis holds different attributes related to them, i.e. Entry points, Sensitive sinks, and sanitation/validation functions. However, it tends to create a lot of false positives. False positives are vulnerabilities detected by the taint analyzer which are not true. The application has to be trained to predict newer false positives with the help of data mining as the code gets complex.

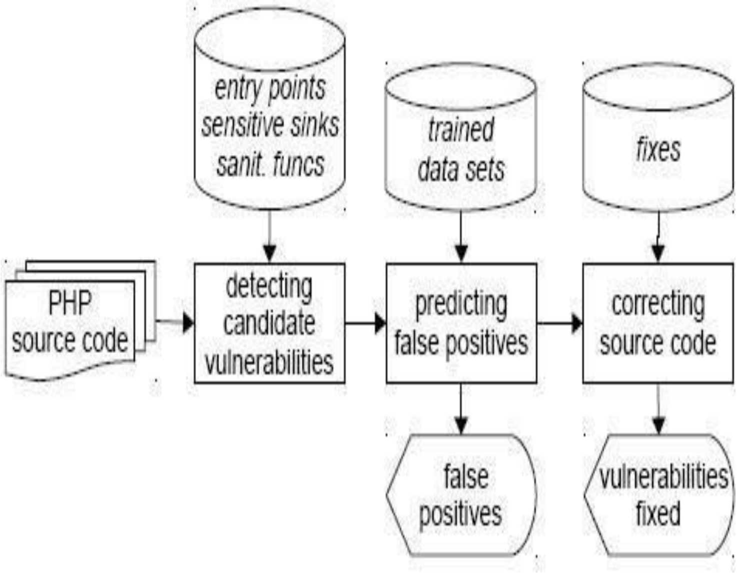


Fig 1:- Overview of the WAP tool

The false positive predictor uses the attributes from these vulnerable control flow path and different classifiers to predict if the given candidate vulnerability is a false positive or not. It uses a combination of three different classifiers, i.e. Logistic regression, random tree, support vector machine. After the vulnerabilities have been predicted, the Code Corrector is used to add the fixes to the faulty code. It looks after the fixes to be added and where they should be added. The code corrector also needs the information about the sanitation function which should be used embed the fix. This can only happen when the new classes are understood through data mining. Tainted modelis being used for performing the static and dynamic analysis. The WAP tool does not hold any specification of PHP, this is one of the drawbacks of this tool. Another drawback is that as we tested the WAP on various open source platforms, it could not parse source codes without a constructed grammar. However, with more research, this drawback has been resolved and new rules have been added. Pixy uses taint analysis as well as alias analysis while testing. The alias analysis are used to verify the existence of aliases, i.e. two or more variables which are used to denominate the same variable. The WAP tool performs global level analysis but Pixy performs only module level analysis.To detect the false positives, we use WEKA tool and perform data mining.

1.4 Proposed System

Through this project, we propose a tool that will scan applications based on PHP, to detect and remove input validation and other vulnerabilities. We use a combination of two techniques i.e. Static source code analysis and data mining in our approach. For detection of false positives, Data mining coupled with different machine learning classifiers is used. The presence of false positives is confirmed by Induction rule classifier. The single detection technique fails to provide correct results so different detection techniques are combined. But they also fail to provide entirely correct results. The generation of false positives happens after the detection of candidate vulnerabilities.The proposed system comprises of the tool which will replace the vulnerable code with fixes.Fixes are nothing but the correct code. After replacing the code by fixes, testing will be performed to check for the correct working of the system. It will check the behaviour of the application after replacing vulnerable code with fixes. The proposed system is designed for PHP applications. The system has been experimented with a number of synthetic codes in which vulnerabilities have been induced purposely.

Hardware Requirements:

System	: Pentium IV 3.5 GHz or Latest Version.
Hard Disk	: 40 GB.
Monitor	: 14’ Colour Monitor.
Mouse	: Optical Mouse.
Ram	: 1 GB.

Software Requirements:

Operating system	: Windows XP or Windows 7, Windows 8.
Coding Language	: Java / J2EE (Jsp,Servlet)
Data Base	: My Sql Server
Documentation	: MS Office
IDE	: Eclipse Galileo
Development Kit	: JDK 1.6
Server	: Tomcat 6.0

CHAPTER 2

LITERETURE SURVEY

A. *Sonam Panda, Ramani S* put forward a static examination algorithm to detect SQLCIVs. It determines the arrangements of conceivable database queries that a web application may create utilizing situation free grammars and tracks flow of data through untrusted sources into those grammars. By making use of a general meaning of SQLCIVs based on the background of unreliable substrings, we can dodge the requirement for manually inscribed policies. It was precise, notable unknown liabilities in the web applications of the real world with multiple negatives, showing the feasibility of our method.

B. *Mounika B, A. Krishna Chaitanya* projected an inevitable system that provided an option to produce an output sanitization which is robotized input approval (IPAAS) and it is displayed for avoiding XSS and SQL assaults. This method advances the safe improvement of web applications by the accomplishment of parameter extraction and different type learning techniques by applying commanding information validators at runtime.

C. *N. L. de Poel,* propounded SAFERPHP, a static investigation system used to detect the liabilities in PHP source code. This agenda employs various algorithms including; 1) To implement new type of symbolic performance to check denial-of-service liabilities. 2) A new type of infers procedural analysis to verify the application sanction policy and find misplaced checks before any complex database operations.

D. *Y.-W. Huang* anticipated anapproach which gave quick protection at a much lower cost than others since approval is confined to potentially weak segments of code. If Web SSARI classifies the use of untrusted data taking after right treatment, the code is left as it is. As per several experiments, Web SSARI carried only 0.02 percent of all the statements to be inspected with inappropriatesanitization agendas. Interestingly, Sharp and Scott implements global justification for each data submitted by the user without any complaint and without even concerning that the same validation process may be implemented by web application as well. These finally outcomes in pointless upstairs.

CHAPTER 3

SYSTEM STUDY

3.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

3.1.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.1.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources.

This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

CHAPTER 4

PRELIMINARY INVESTIGATION

The first and foremost strategy for development of a project starts from the thought of designing a mail enabled platform for a small firm in which it is easy and convenient of sending and receiving messages, there is a search engine ,address book and also including some entertaining games. When it is approved by the organization and our project guide the first activity, ie. preliminary investigation begins. The activity has three parts:

- **Request Clarification**
- **Feasibility Study**
- **Request Approval**

4.1 REQUEST CLARIFICATION

After the approval of the request to the organization and project guide, with an investigation being considered, the project request must be examined to determine precisely what the system requires.

Here our project is basically meant for users within the company whose systems can be interconnected by the Local Area Network(LAN). In today's busy schedule man need everything should be provided in a readymade manner. So taking into consideration of the vastly use of the net in day to day life, the corresponding development of the portal came into existence.

4.2 FEASIBILITY ANALYSIS

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. The different feasibilities that have to be analyzed are

- **Operational Feasibility**
- **Economic Feasibility**
- **Technical Feasibility**

4.2.1 Operational Feasibility

Operational Feasibility deals with the study of prospects of the system to be developed. This system operationally eliminates all the tensions of the Admin and helps him in effectively tracking the project progress. This kind of automation will surely reduce the time and energy, which previously consumed in manual work. Based on the study, the system is proved to be operationally feasible.

4.2.2 Economic Feasibility

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer based project. As hardware was installed from the beginning & for lots of purposes thus the cost on project of hardware is low. Since the system is a network based, any number of employees connected to the LAN within that organization can use this tool from at anytime. The Virtual Private Network is to be developed using the existing resources of the organization. So the project is economically feasible.

4.2.3 Technical Feasibility

According to Roger S. Pressman, Technical Feasibility is the assessment of the technical resources of the organization. The organization needs IBM compatible machines with a graphical web browser connected to the Internet and Intranet. The system is developed for platform Independent environment. Java Server Pages, JavaScript, HTML, SQL server and WebLogic Server are used to develop the system. The technical feasibility has been carried out. The system is technically feasible for development and can be developed with the existing facility.

4.3 REQUEST APPROVAL

Not all request projects are desirable or feasible. Some organization receives so many project requests from client users that only few of them are pursued. However, those projects that are both feasible and desirable should be put into schedule. After a project request is approved, it cost, priority, completion time and personnel requirement is estimated and used to determine where to add it to any project list. Truly speaking, the approval of those above factors, development works can be launched.

CHAPTER 5

SYSTEM DESIGN AND DEVELOPMENT

5.1 INPUT DESIGN

Input Design plays a vital role in the life cycle of software development, it requires very careful attention of developers. The input design is to feed data to the application as accurate as possible. So inputs are supposed to be designed effectively so that the errors occurring while feeding are minimized. According to Software Engineering Concepts, the input forms or screens are designed to provide to have a validation control over the input limit, range and other related validations.

This system has input screens in almost all the modules. Error messages are developed to alert the user whenever he commits some mistakes and guides him in the right way so that invalid entries are not made. Let us see deeply about this under module design.

Input design is the process of converting the user created input into a computer-based format. The goal of the input design is to make the data entry logical and free from errors. The error is in the input are controlled by the input design. The application has been developed in user-friendly manner. The forms have been designed in such a way during the processing the cursor is placed in the position where must be entered. The user is also provided with in an option to select an appropriate input from various alternatives related to the field in certain cases.

Validations are required for each data entered. Whenever a user enters an erroneous data, error message is displayed and the user can move on to the subsequent pages after completing all the entries in the current page.

5.2 MODULES

Admin

In this module, admin has to login with valid username and password. After login successful he can do some operations such as view all user, their details and authorize them , view all owners, their details and authorize them, view all attackers details(like ip address and host name), view all sql injection vulnerabilities and block them(those who are execute wrong query), view all file access vulnerabilities of users(those who are used wrong secret key) , view all blocked data owners , view unblock requests and unblock them, view all secret key requests and generate, view all users download history, view SQL Injection Vulnerabilities in chart, View number of remote vulnerabilities in chart.

- ### User

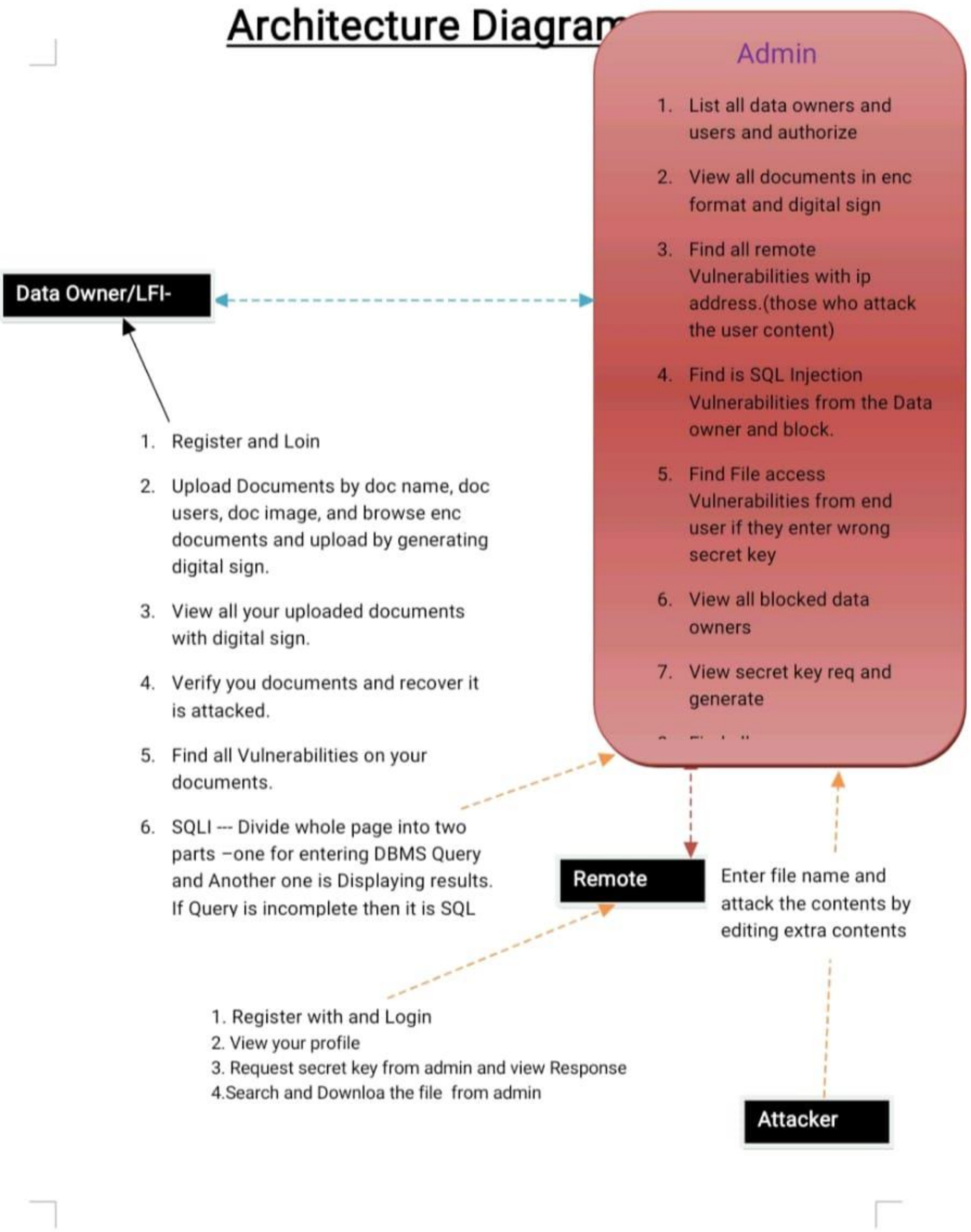
In this module, there are n numbers of users are present. User should register before doing some operations. After registration successful he can login by using valid user name and password. Login successful he will do some operations like view profile details, request secret key and view response, search documents and download by entering secret key.

- ### Data Owner

In this module, there are n numbers of owners are present. Owner should register before doing some operations. After registration successful he can login by using valid user name and password. Login successful he will do some operations like view profile details, Upload documents and generate digital sign, view uploaded documents, verify his documents and recover if it is attacked, view all on his documents like(downloads and attacked details), Execute SQL queries if query is incomplete the it is SQL Injection Vulnerabilities.

- ### Attacker

Attacker searches the documents and edit document by changing content.



1. List all data owners and users and authorize
2. View all documents in enc format and digital sign
3. Find all remote Vulnerabilities with ip address.(those who attack the user content)
4. Find is SQL Injection Vulnerabilities from the Data owner and block.
5. Find File access Vulnerabilities from end user if they enter wrong secret key
6. View all blocked data owners
7. View secret key req and generate

1. Register and Loin
2. Upload Documents by doc name, doc users, doc image, and browse enc documents and upload by generating digital sign.
3. View all your uploaded documents with digital sign.
4. Verify you documents and recover it is attacked.
5. Find all Vulnerabilities on your documents.
6. SQLI --- Divide whole page into two parts –one for entering DBMS Query and Another one is Displaying results. If Query is incomplete then it is SQL

Enter file name and attack the contents by editing extra contents

1. Register with and Login
2. View your profile
3. Request secret key from admin and view Response
4.Search and Downloa the file from admin

Fig2. Architecture Diagram

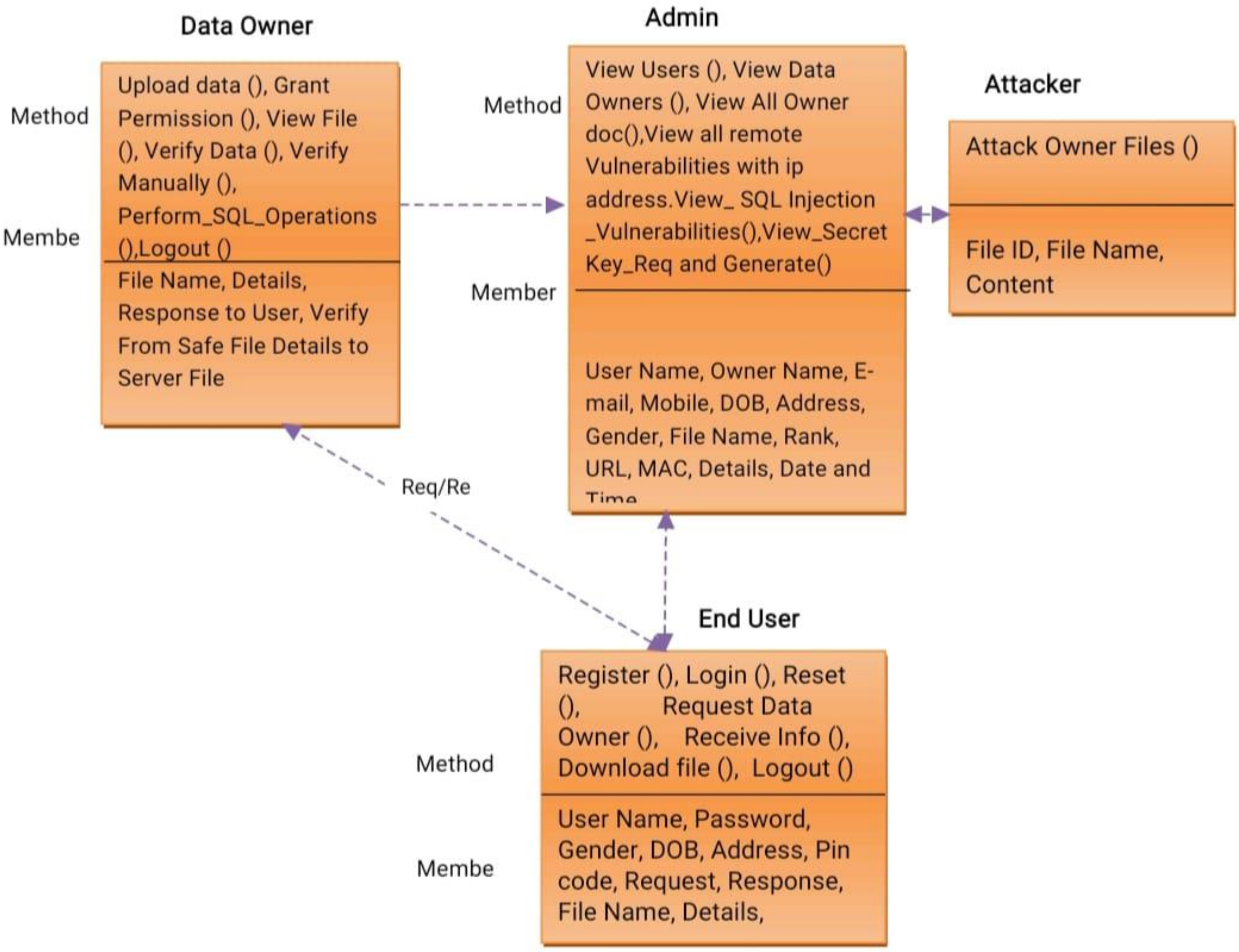

Data Owner
Method
Membe
Upload data (), Grant Permission (), View File (), Verify Data (), Verify Manually (), Perform_SQL_Operations (),Logout ()
File Name, Details, Response to User, Verify From Safe File Details to Server File
Admin
Method
Member
View Users (), View Data Owners (), View All Owner doc(),View all remote Vulnerabilities with ip address.View_ SQL Injection _Vulnerabilities(),View_Secret Key_Req and Generate()
User Name, Owner Name, E-mail, Mobile, DOB, Address, Gender, File Name, Rank, URL, MAC, Details, Date and Time
Attacker
Attack Owner Files ()
File ID, File Name, Content
Req/Re
End User
Method
Membe
Register (), Login (), Reset (), Request Data Owner (), Receive Info (), Download file (), Logout ()
User Name, Password, Gender, DOB, Address, Pin code, Request, Response, File Name, Details,

Fig 3. Class Diagram

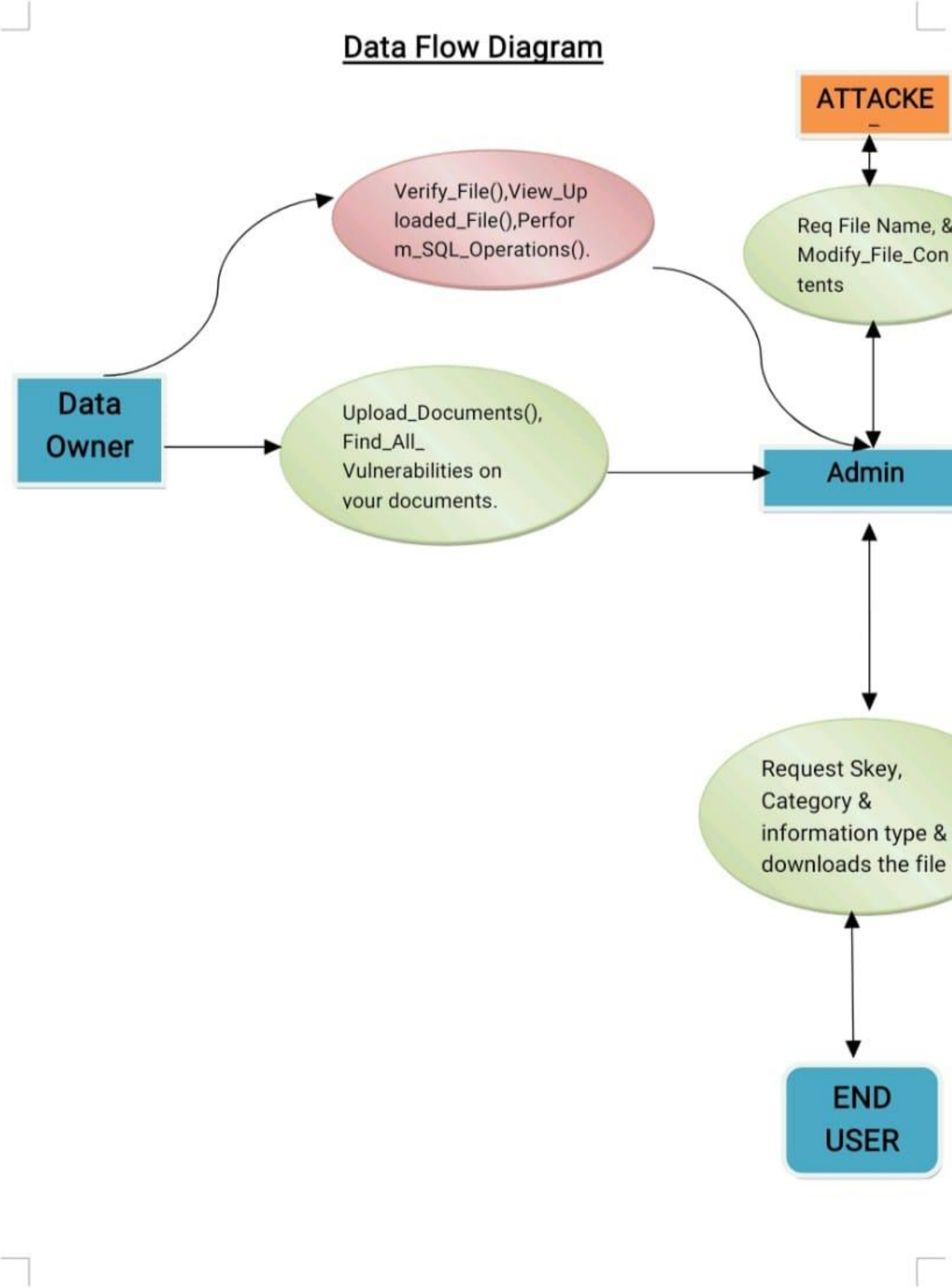

Data Flow Diagram
ATTACKE
Verify_File(),View_Uploaded_File(),Perform_SQL_Operations().
Req File Name, & Modify_File_Contents
Data Owner
Upload_Documents(), Find_All_ Vulnerabilities on your documents.
Admin
Request Skey, Category & information type & downloads the file
END USER

Fig 4. Data flow diagram

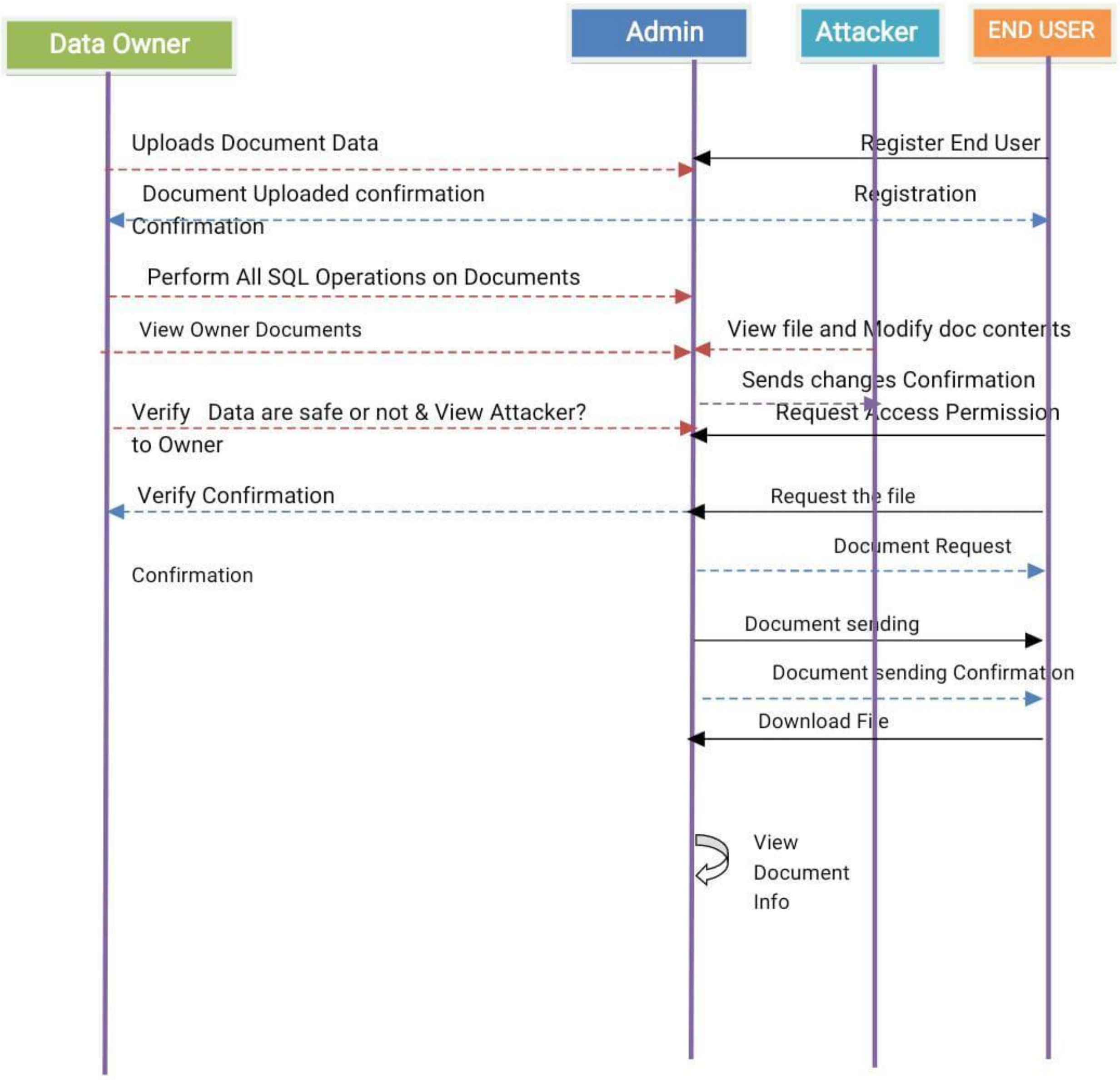


Fig 5. Sequence diagram

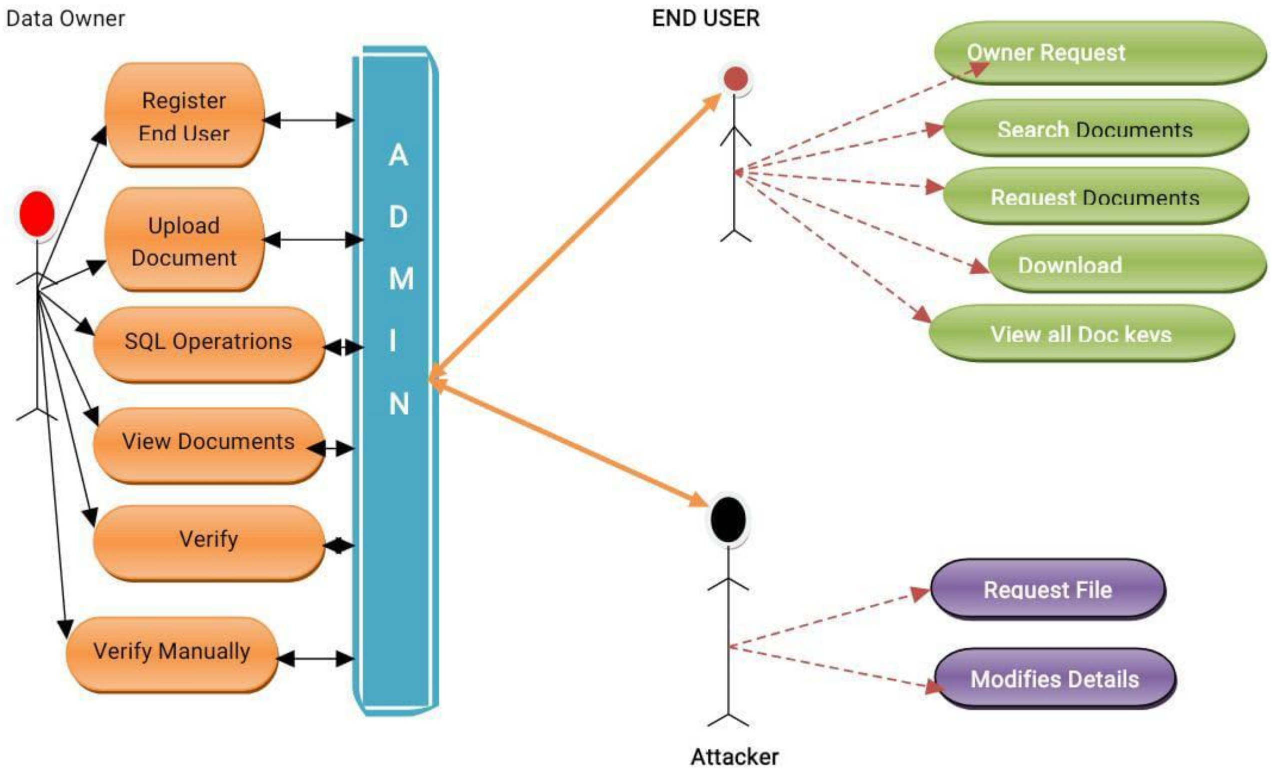


Fig 6. Use_case_diagram

5.3 OUTPUT DESIGN

The Output from the computer is required to mainly create an efficient method of communication within the company primarily among the project leader and his team members, in other words, the administrator and the clients. The output of VPN is the system which allows the project leader to manage his clients in terms of creating new clients and assigning new projects to them, maintaining a record of the project validity and providing folder level access to each client on the user side depending on the projects allotted to him. After completion of a project, a new project may be assigned to the client. User authentication procedures are maintained at the initial stages itself. A new user may be created by the administrator himself or a user can himself register as a new user but the task of assigning projects and validating a new user rests with the administrator only.

The application starts running when it is executed for the first time. The server has to be started and then the internet explorer in used as the browser. The project will run on the local area network so the server machine will serve as the administrator while the other connected systems can act as the clients. The developed system is highly user friendly and can be easily understood by anyone using it even for the first time.

CHAPTER 6

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1TYPES OF TESTS

6.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.1.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.1.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised.
Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.1.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.1.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

6.2 OTHER TESTING METHODOLOGIES

6.2.1 User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

6.2.2Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

6.2.3 Validation Checking

Validation checks are performed on the following fields.

6.2.4 Text Field

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

6.2.5 Numeric Field

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

6.2.6 Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

6.2.7 Using Live Test Data

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system.

6.2.8 Using Artificial Test Data

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package "Virtual Private Network" has satisfied all the requirements specified as per software requirement specification and was accepted.

6.3 USER TRAINING

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

6.4 MAINTAINENCE

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

6.5 TESTING STRATEGY

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

CHARTER 7

COST ESTIMATION

BUDGET	AMOUNT
SOFTWARES	6000
TRAVEL	2000
LABOUR	1000
OTHERS	2000
TOTAL	11000

CHAPTER 8

CONCLUSION

The project presents an approach for finding and correcting vulnerabilities in web applications and a tool that implements the approach for PHP programs and input validation vulnerabilities. The approach and the tool search for vulnerabilities using a combination of two techniques: static source code analysis and data mining. Data mining is used to identify false positives using a top 3 of machine learning classifiers and to justify their presence using an induction rule classifier. All classifiers were selected after a thorough comparison of several alternatives. It is important to note that this combination of detection techniques cannot provide entirely correct results. The static analysis problem is un decidable and the resort to data mining cannot circumvent this un decidability, only provide probabilistic results. The tool corrects the code by inserting fixes, i.e., sanitization and validation functions. Testing is used to verify if the fixes actually remove the vulnerabilities and do not compromise the (correct) behavior of the applications. The tool was experimented with synthetic code with vulnerabilities inserted on purpose and with a considerable number of open source PHP applications. It was also compared with two source code analysis tools, Pixy and PhpMinerII. This evaluation suggests that the tool can detect and correct the vulnerabilities of the classes it is programmed to handle. It was able to find 388 vulnerabilities in 1.4 million lines of code. Its accuracy and precision were approximately 5% better than PhpMinerII’s and 45% better than Pixy’s.

REFERENCES

[1] WAP tool website. http://awap.sourceforge.net/.

[2] J. Antunes, N. F. Neves, M. Correia, P. Verissimo, and R. Neves. Vulnerability removal with attack injection. IEEE Transactions on Software Engineering, 36(3):357–370, 2010.

[3] E. Arisholm, L. C. Briand, and E. B. Johannessen. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. Journal of Systems and Software, 83(1):2–17, 2010.

[4] R. Banabic and G. Candea. Fast black-box testing of system recovery code. In Proceedings of the 7th ACM European Conference on Computer Systems, pages 281–294, 2012.

[5] L. C. Briand, J. Wüst, J. W. Daly, and D. Victor Porter. Exploring the relationships between design measures and software quality in objectoriented systems. Journal of Systems and Software, 51(3):245–273, 2000.

[6] G. T. Buehrer, B. W. Weide, and P. Sivilotti. Using parse tree validation to prevent SQL injection attacks. In Proceedings of the 5th International Workshop on Software Engineering and Middleware, pages 106–113, Sept. 2005.

[7] N. L. de Poel. Automated security review of PHP web applications with static code analysis. Master’s thesis, State University of Groningen, May 2010.

[8] R. A. DeMillo, R. J. Lipton, and F. G. Sayward. Hints on test data selection: Help for the practicing programmer. Computer, 11(4):34–41, Apr 1978.

[9] J. Demšar. Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research, 7:1–30, Dec 2006.

[10] D. Evans and D. Larochelle. Improving security using extensible lightweight static analysis. IEEE Software, pages 42–51, Jan/Feb 2002.

[11] W. Halfond and A. Orso. AMNESIA: analysis and monitoring for neutralizing SQL-injection attacks. In Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, pages 174–183, Nov. 2005.

[12] W. Halfond, A. Orso, and P. Manolios. WASP: protecting web applications using positive tainting and syntax-aware evaluation. IEEE Transactions on Software Engineering, 34(1):65–81, 2008.

[13] J. C. Huang. Software Error Detection through Testing and Analysis. John Wiley and Sons, Inc., 2009.

[14] Y.-W. Huang, S.-K. Huang, T.-P. Lin, and C.-H. Tsai. Web application security assessment by fault injection and behavior monitoring. In Proceedings of the 12th International Conference on World Wide Web, pages 148–159, 2003.

[15] Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D.-T. Lee, and S.-Y. Kuo. Securing web application code by static analysis and runtime protection. In Proceedings of the 13th International World Wide Web Conference, pages 40–52, 2004.

[16] Imperva. Hacker intelligence initiative, monthly trend report #8. Apr. 2012.

[17] N. Jovanovic, C. Kruegel, and E. Kirda. Precise alias analysis for static detection of web application vulnerabilities. In Proceedings of the 2006 Workshop on Programming Languages and Analysis for Security, pages 27–36, June 2006.

[18] W. Landi. Undecidability of static analysis. ACM Letters on Programming
Languages and Systems, 1(4):323–337, 1992.

[19] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. IEEE Transactions on Software Engineering, 34(4):485–496, 2008.

[20] E. Merlo, D. Letarte, and G. Antoniol. Automated Protection of PHP Applications Against SQL Injection Attacks. In Proceedings of the 11th European Conference on Software Maintenance and Reengineering, pages 191–202, Mar. 2007.

[21] S. Neuhaus, T. Zimmermann, C. Holler, and A. Zeller. Predicting vulnerable software components. In Proceedings of the 14th ACM Conference on Computer and Communications Security, pages 529–540, 2007.

[22] A. Nguyen-Tuong, S. Guarnieri, D. Greene, J. Shirley, and D. Evans. Automatically hardening web applications using precise tainting. Security and Privacy in the Age of Ubiquitous Computing, pages 295–307, 2005.

[23] T. Parr. Language Implementation Patterns: Create Your Own Domain- Specific and General Programming Languages. Pragmatic Bookshelf, 2009.

[24] T. Pietraszek and C. V. Berghe. Defending against injection attacks through context-sensitive string evaluation. In Proceedings of the 8th International Conference on Recent Advances in Intrusion Detection, pages 124–145, 2005.

[25] A. Sabelfeld and A. C. Myers. Language-based information-flow security. IEEE Journal on Selected Areas in Communications, 21(1):5– 19, 2003.

[26] R. S. Sandhu. Lattice-based access control models. IEEE Computer, 26(11):9–19, 1993.

[27] U. Shankar, K. Talwar, J. S. Foster, and D. Wagner. Detecting formatstring vulnerabilities with type qualifiers. In Proceedings of the 10th USENIX Security Symposium, volume 10, pages 16–16, Aug. 2001.

[28] L. K. Shar and H. B. K. Tan. Automated removal of cross site scripting vulnerabilities in web applications. Information and Software Technology, 54(5):467–478, 2012.

[29] L. K. Shar and H. B. K. Tan. Mining input sanitization patterns for predicting SQL injection and cross site scripting vulnerabilities. In Proceedings of the 34th International Conference on Software Engineering, pages 1293–1296, 2012.

[30] L. K. Shar and H. B. K. Tan. Predicting common web application vulnerabilities from input validation and sanitization code patterns. In Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, pages 310–313, 2012.

[31] L. K. Shar, H. B. K. Tan, and L. C. Briand. Mining SQL injection and cross site scripting vulnerabilities using hybrid program analysis. In Proceedings of the 35th International Conference on Software Engineering, pages 642–651, 2013.

[32] Y. Shin, A. Meneely, L. Williams, and J. A. Osborne. Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. IEEE Transactions on Software Engineering, 37(6):772–787, 2011.

[33] S. Son and V. Shmatikov. SAFERPHP: Finding semantic vulnerabilities in PHP applications. In Proceedings of the ACM SIGPLAN 6th Workshop on Programming Languages and Analysis for Security, 2011.

[34] Symantec. Internet threat report. 2012 trends, volume 18. Apr. 2013.

[35] T. Budd et al. The design of a prototype mutation system for program testing. In Proceedings of the AFIPS National Computer Conference, pages 623–627, 1978.

Bibliography:

References for the Project Development were taken from the following Books and Web Sites.

Oracle

PL/SQL Programming by Scott Urman

SQL complete reference by Livion

JAVA Technologies

JAVA Complete Reference

Java Script Programming by Yehuda Shiran

Mastering JAVA Security

JAVA2 Networking by Pistoria

JAVA Security by Scotl oaks

Head First EJB Sierra Bates

J2EE Professional by Shadab siddiqui

JAVA server pages by Larne Pekowsley

JAVA Server pages by Nick Todd

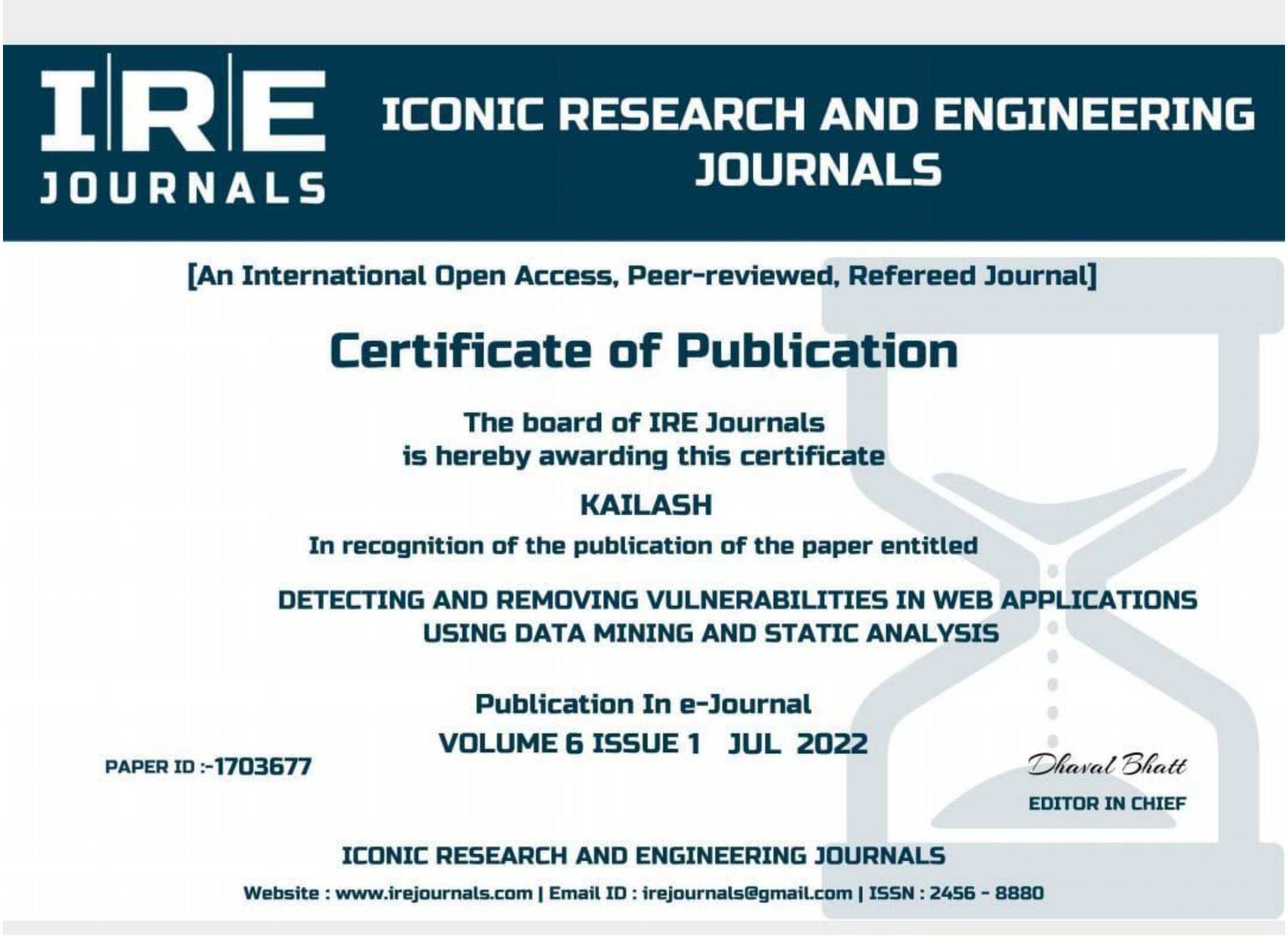
HTML

HTML Black Book by Holzner

JDBC

Java Database Programming with JDBC by Patel moss.

JOURNAL CERTIFICATE



IRE JOURNALS

ICONIC RESEARCH AND ENGINEERING JOURNALS

[An International Open Access, Peer-reviewed, Refereed Journal]

Certificate of Publication

The board of IRE Journals
is hereby awarding this certificate

KAILASH

In recognition of the publication of the paper entitled

DETECTING AND REMOVING VULNERABILITIES IN WEB APPLICATIONS USING DATA MINING AND STATIC ANALYSIS

Publication In e-Journal

VOLUME 6 ISSUE 1 JUL 2022

PAPER ID :-1703677

Dhaval Bhatt

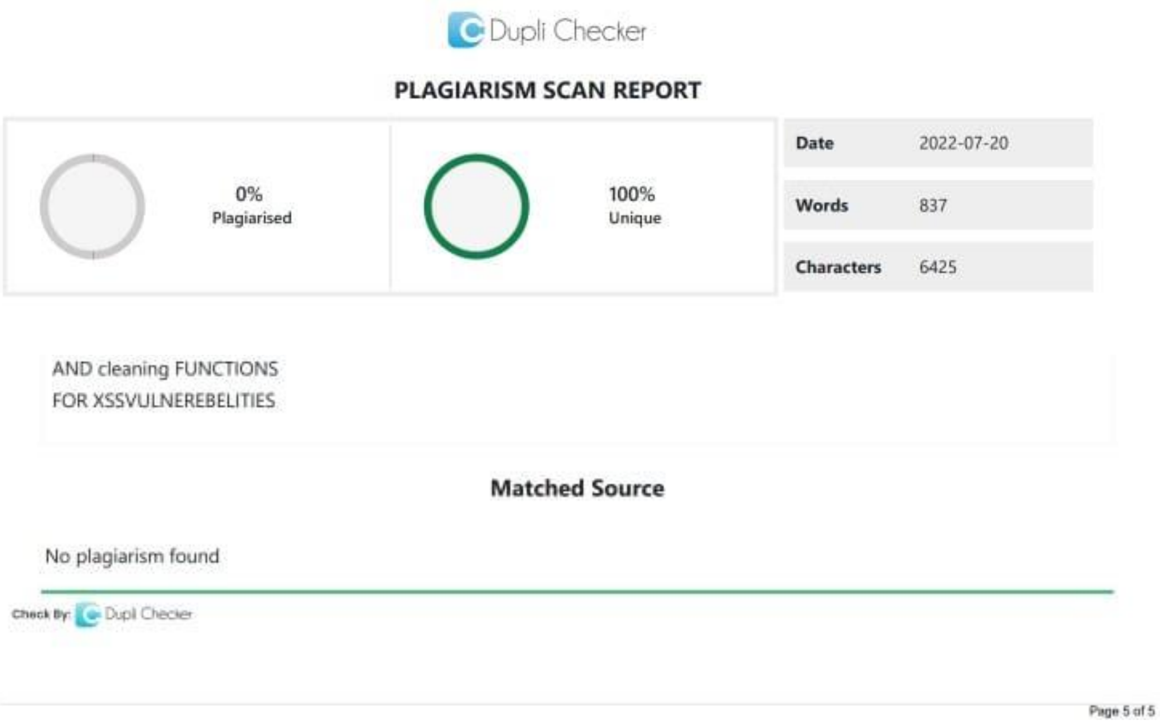
EDITOR IN CHIEF

ICONIC RESEARCH AND ENGINEERING JOURNALS

Website : www.irejournals.com | Email ID : irejournals@gmail.com | ISSN : 2456 - 8880

PLAGIARISM REPORT

PAGE 01 to 07



Dupli Checker

PLAGIARISM SCAN REPORT

0% Plagiarised

100% Unique

Date	2022-07-20
Words	837
Characters	6425

AND cleaning FUNCTIONS
FOR XSSVULNEREBELITIES

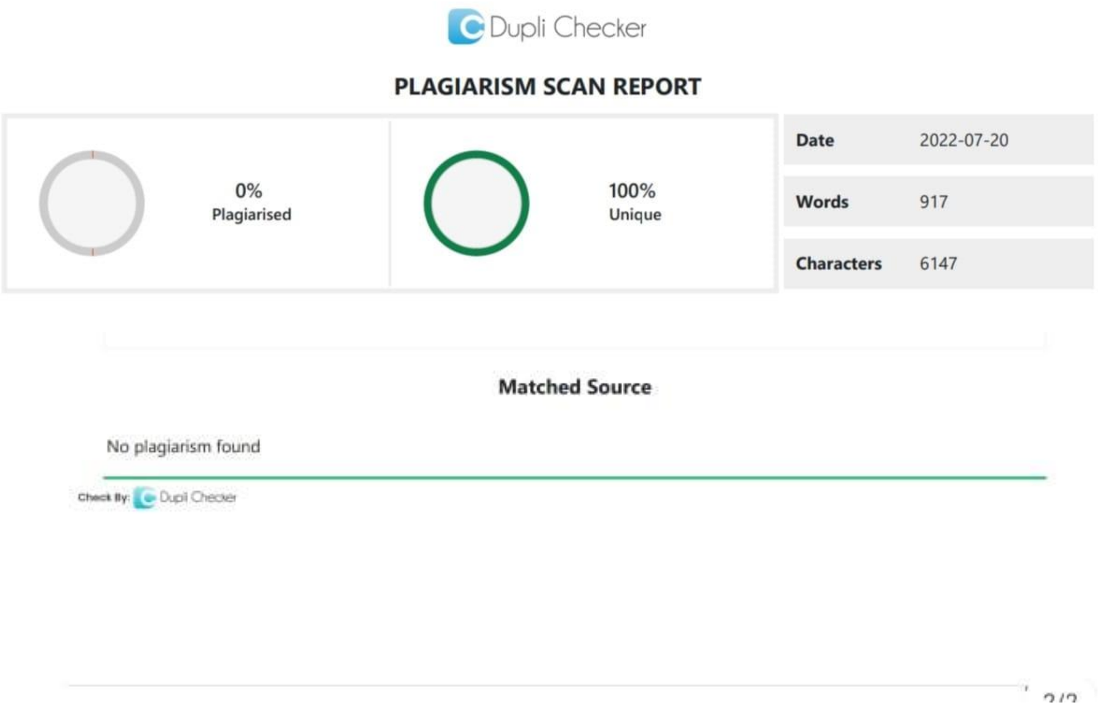
Matched Source

No plagiarism found

Check By: Dupli Checker

Page 5 of 5

PAGE 08 to 09



Dupli Checker

PLAGIARISM SCAN REPORT

0% Plagiarised

100% Unique

Date	2022-07-20
Words	917
Characters	6147

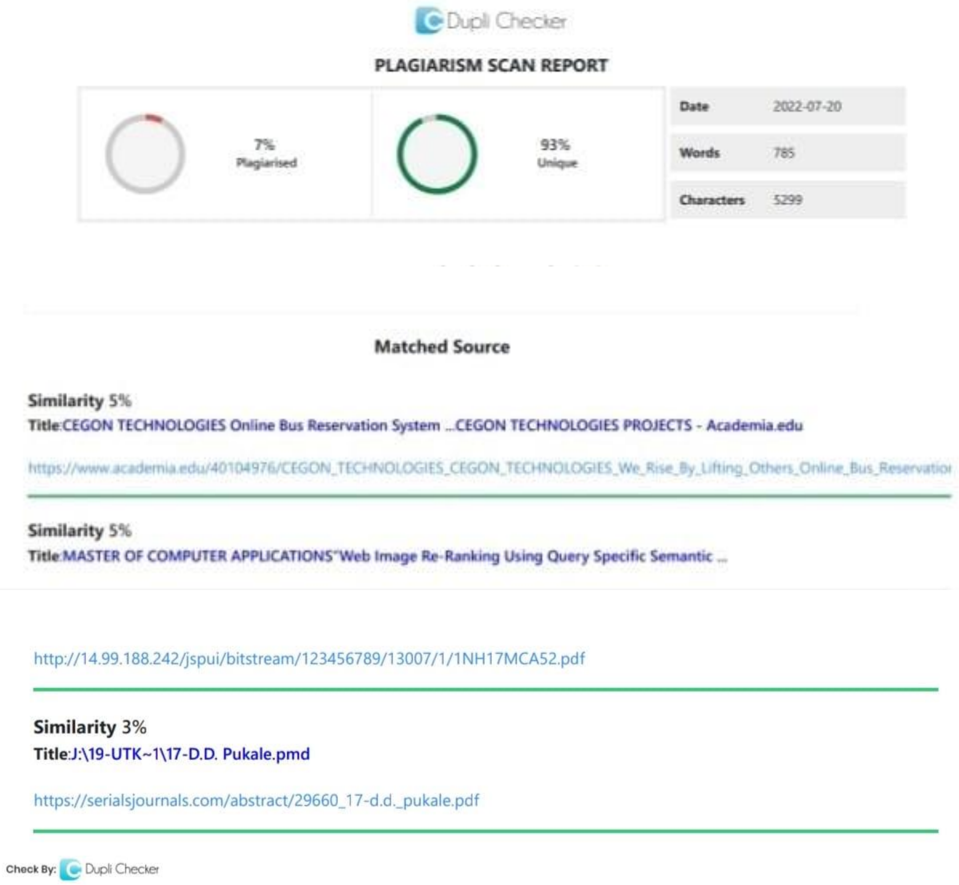
Matched Source

No plagiarism found

Check By: Dupli Checker

2/2

PAGE 10 to 12



Dupli Checker

PLAGIARISM SCAN REPORT

7%
Plagiarised

93%
Unique

Date	2022-07-20
Words	785
Characters	5299

Matched Source

Similarity 5%
Title:CEGON TECHNOLOGIES Online Bus Reservation System ...CEGON TECHNOLOGIES PROJECTS - Academia.edu

https://www.academia.edu/40104976/CEGON_TECHNOLOGIES_CEGON_TECHNOLOGIES_We_Rise_By_Lifting_Others_Online_Bus_Reservatio

Similarity 5%
Title:MASTER OF COMPUTER APPLICATIONS"Web Image Re-Ranking Using Query Specific Semantic ...

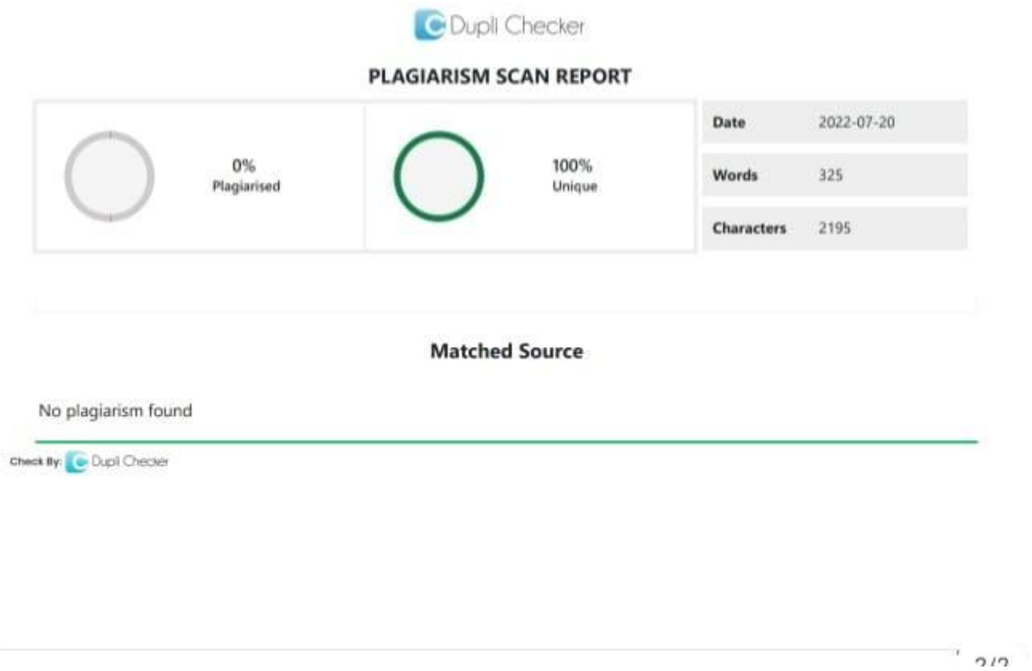
http://14.99.188.242/jspui/bitstream/123456789/13007/1/1NH17MCA52.pdf

Similarity 3%
Title:J:\19-UTK~1\17-D.D. Pukale.pmd

https://serialsjournals.com/abstract/29660_17-d.d._pukale.pdf

Check By: Dupli Checker

PAGE 13



Dupli Checker

PLAGIARISM SCAN REPORT

0%
Plagiarised

100%
Unique

Date	2022-07-20
Words	325
Characters	2195

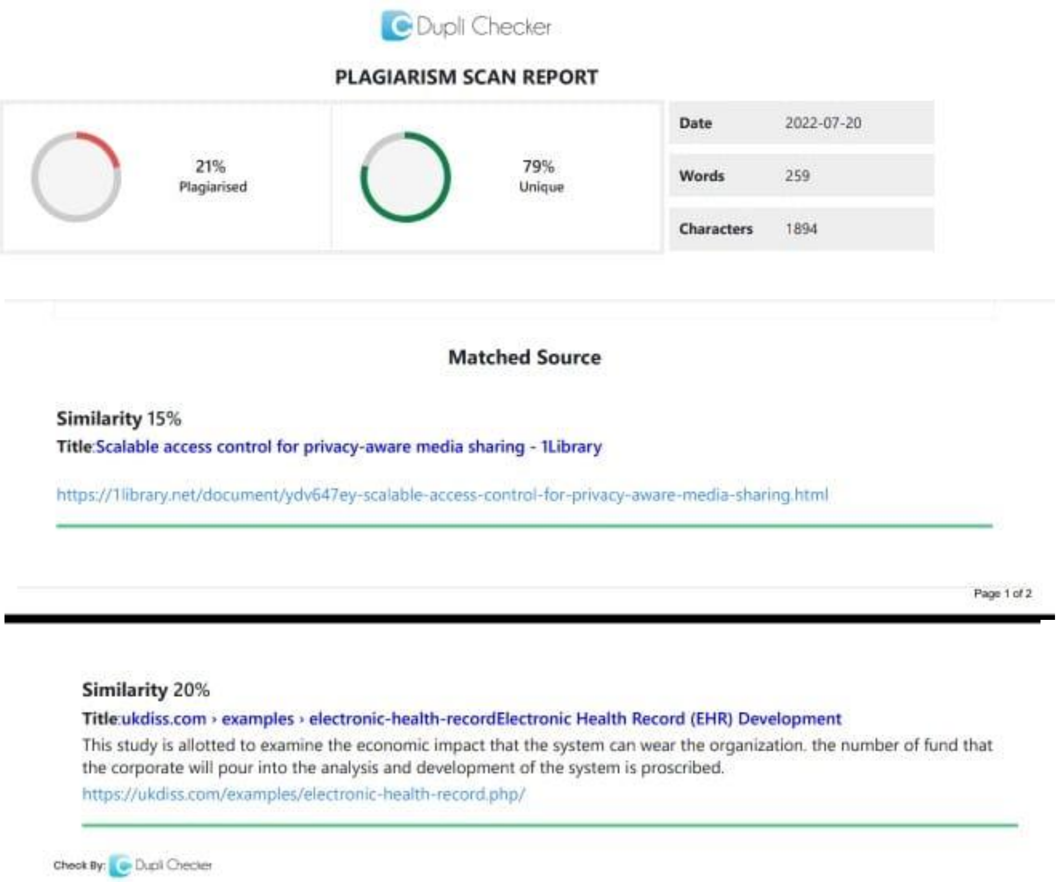
Matched Source

No plagiarism found

Check By: Dupli Checker

2/2

PAGE 14



Dupli Checker

PLAGIARISM SCAN REPORT

21% Plagiarised	79% Unique

Date	2022-07-20
Words	259
Characters	1894

Matched Source

Similarity 15%

Title:Scalable access control for privacy-aware media sharing - 1Library

https://1library.net/document/ydv647ey-scalable-access-control-for-privacy-aware-media-sharing.html

Page 1 of 2

Similarity 20%

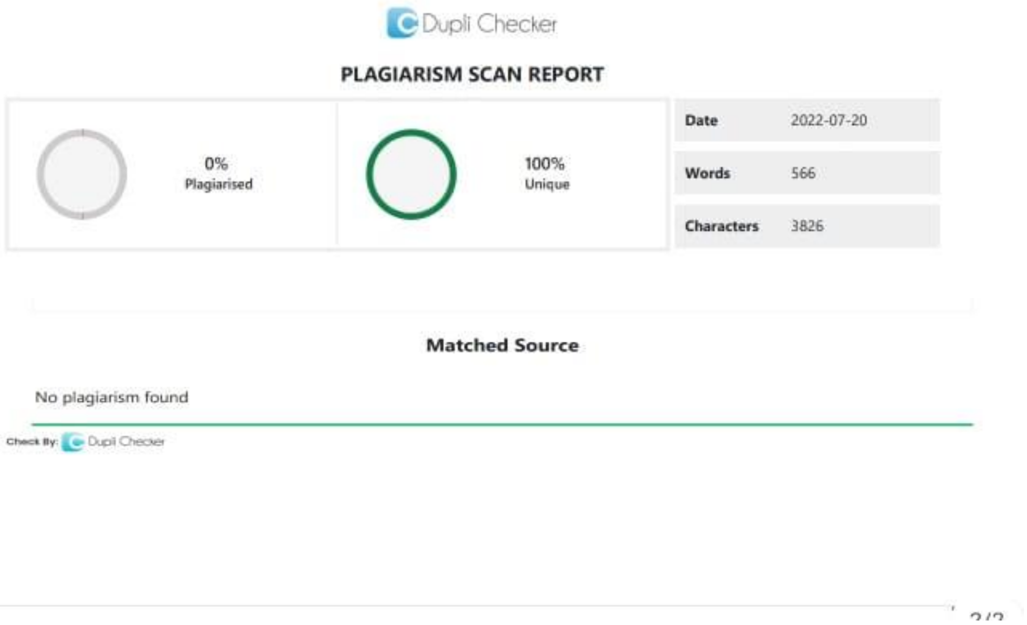
Title:ukdiss.com › examples › electronic-health-recordElectronic Health Record (EHR) Development

This study is allotted to examine the economic impact that the system can wear the organization. the number of fund that the corporate will pour into the analysis and development of the system is proscribed.

https://ukdiss.com/examples/electronic-health-record.php/

Check By: Dupli Checker

PAGE 15 to 16



Dupli Checker

PLAGIARISM SCAN REPORT

0% Plagiarised	100% Unique

Date	2022-07-20
Words	566
Characters	3826

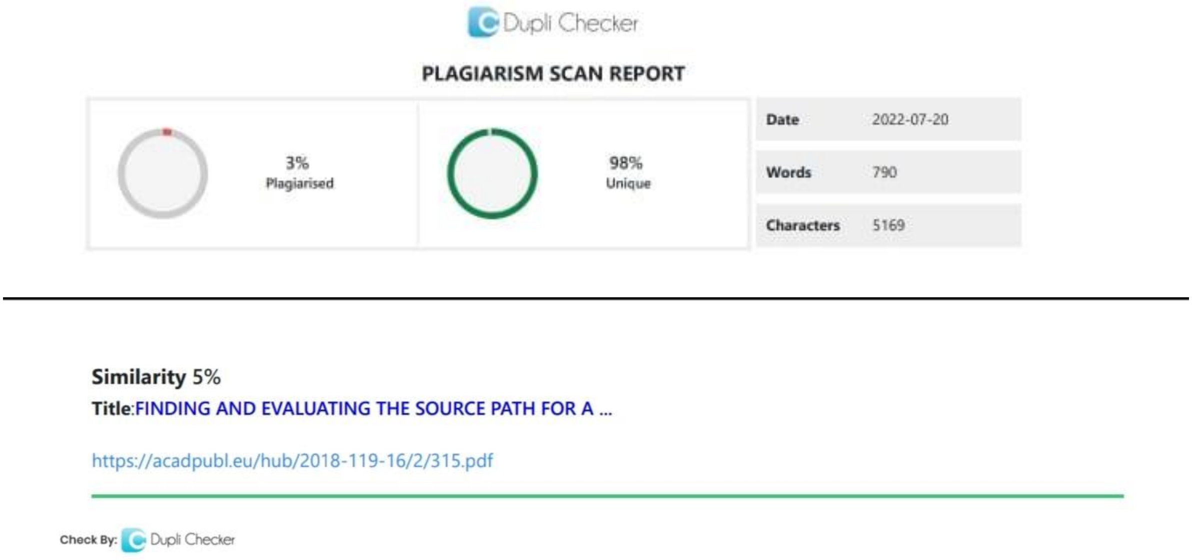
Matched Source

No plagiarism found

Check By: Dupli Checker

2/2

PAGE 17 to 18, 24



Dupli Checker

PLAGIARISM SCAN REPORT

3% Plagiarised

98% Unique

Date	2022-07-20
Words	790
Characters	5169

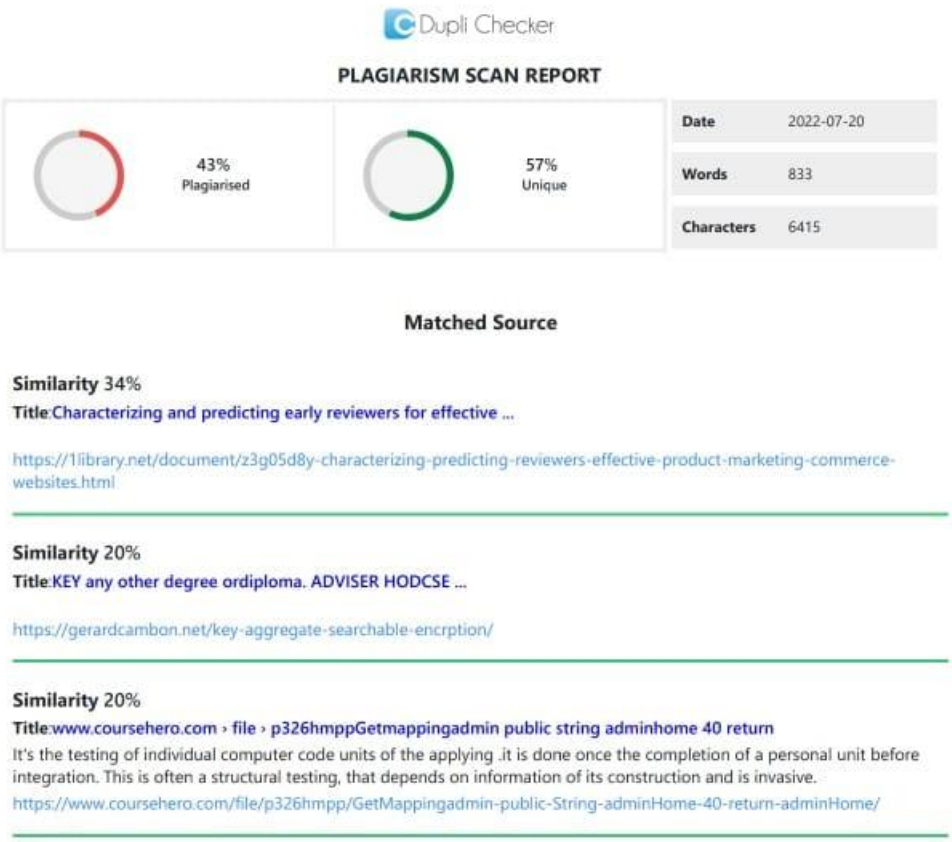
Similarity 5%

Title:FINDING AND EVALUATING THE SOURCE PATH FOR A ...

https://acadpubl.eu/hub/2018-119-16/2/315.pdf

Check By: Dupli Checker

PAGE 25 to 27



Dupli Checker

PLAGIARISM SCAN REPORT

43% Plagiarised

57% Unique

Date	2022-07-20
Words	833
Characters	6415

Matched Source

Similarity 34%

Title:Characterizing and predicting early reviewers for effective ...

https://1library.net/document/z3g05d8y-characterizing-predicting-reviewers-effective-product-marketing-commerce-websites.html

Similarity 20%

Title:KEY any other degree ordiploma. ADVISER HODCSE ...

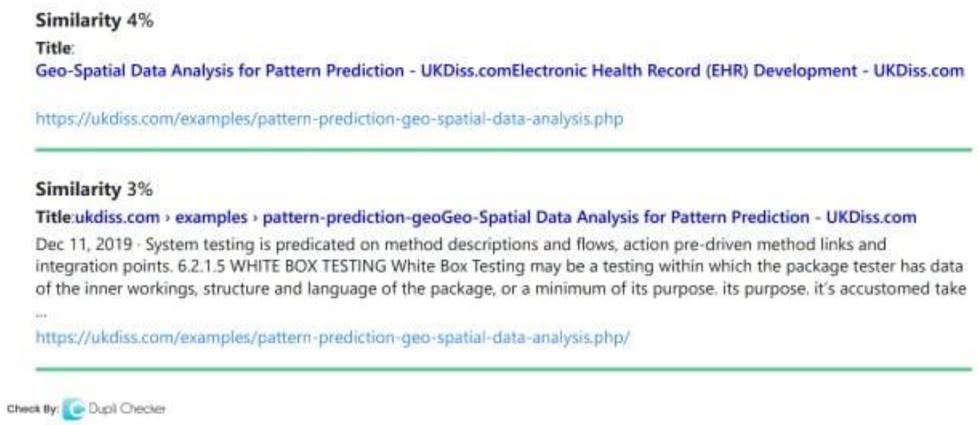
https://gerardcambon.net/key-aggregate-searchable-encrption/

Similarity 20%

Title:www.coursehero.com › file › p326hmppGetmappingadmin public string adminhome 40 return

It's the testing of individual computer code units of the applying .it is done once the completion of a personal unit before integration. This is often a structural testing, that depends on information of its construction and is invasive.

https://www.coursehero.com/file/p326hmpp/GetMappingadmin-public-String-adminHome-40-return-adminHome/



Similarity 4%

Title:
Geo-Spatial Data Analysis for Pattern Prediction - UKDiss.comElectronic Health Record (EHR) Development - UKDiss.com

https://ukdiss.com/examples/pattern-prediction-geo-spatial-data-analysis.php

Similarity 3%

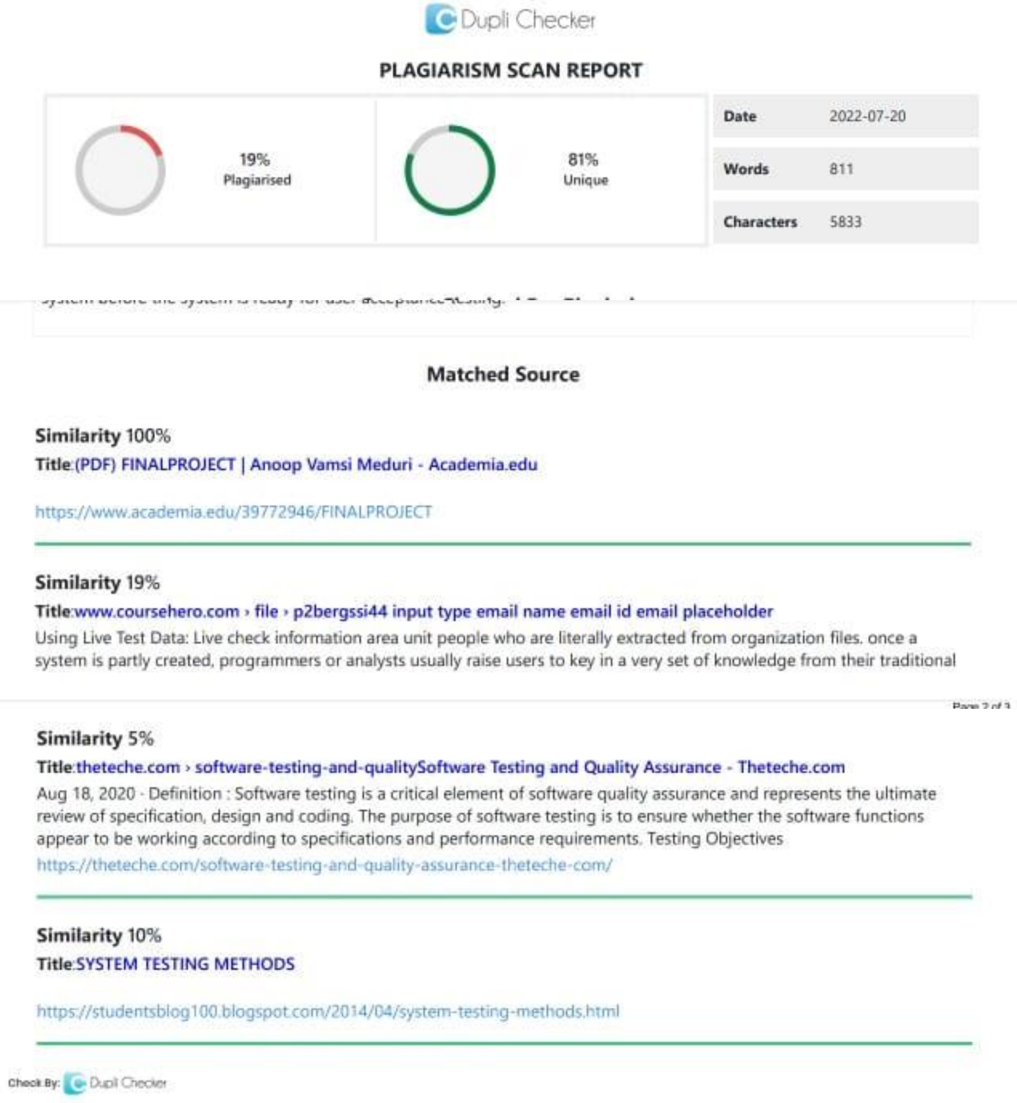
Title:ukdiss.com › examples › pattern-prediction-geoGeo-Spatial Data Analysis for Pattern Prediction - UKDiss.com

Dec 11, 2019 · System testing is predicated on method descriptions and flows, action pre-driven method links and integration points. 6.2.1.5 WHITE BOX TESTING White Box Testing may be a testing within which the package tester has data of the inner workings, structure and language of the package, or a minimum of its purpose. its purpose. it's accustomed take ...

https://ukdiss.com/examples/pattern-prediction-geo-spatial-data-analysis.php/

Check By: Dupli Checker

PAGE 28 to 30



Dupli Checker

PLAGIARISM SCAN REPORT

19% Plagiarised	81% Unique	**Date**	2022-07-20
		Words	811
		Characters	5833

system before the system is ready for user acceptance testing.

Matched Source

Similarity 100%

Title:(PDF) FINALPROJECT | Anoop Vamsi Meduri - Academia.edu

https://www.academia.edu/39772946/FINALPROJECT

Similarity 19%

Title:www.coursehero.com › file › p2bergssi44 input type email name email id email placeholder

Using Live Test Data: Live check information area unit people who are literally extracted from organization files. once a system is partly created, programmers or analysts usually raise users to key in a very set of knowledge from their traditional

Page 2 of 3

Similarity 5%

Title:theteche.com › software-testing-and-qualitySoftware Testing and Quality Assurance - Theteche.com

Aug 18, 2020 · Definition : Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The purpose of software testing is to ensure whether the software functions appear to be working according to specifications and performance requirements. Testing Objectives

https://theteche.com/software-testing-and-quality-assurance-theteche-com/

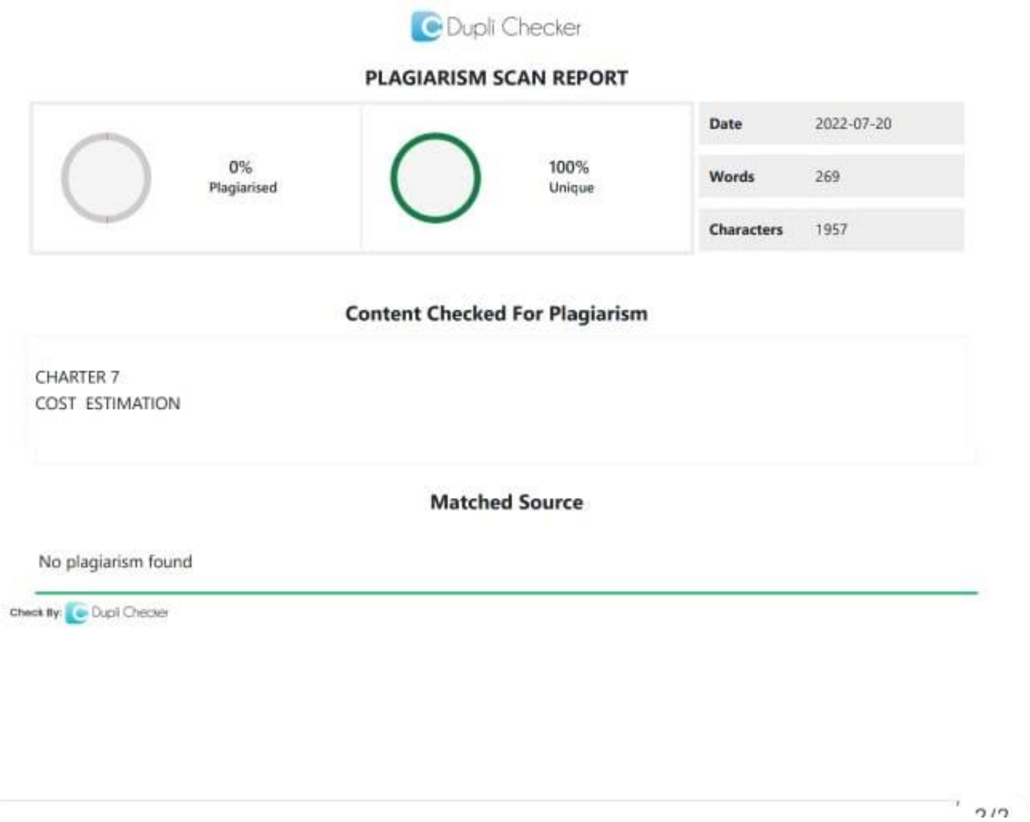
Similarity 10%

Title:SYSTEM TESTING METHODS

https://studentsblog100.blogspot.com/2014/04/system-testing-methods.html

Check By: Dupli Checker

PAGE 31 to 32



Dupli Checker

PLAGIARISM SCAN REPORT

0%
Plagiarised

100%
Unique

Date	2022-07-20
Words	269
Characters	1957

Content Checked For Plagiarism

CHARTER 7
COST ESTIMATION

Matched Source

No plagiarism found

Check By: Dupli Checker