Title:

A Combined Approach of Fine Role-Based Access Control and Dynamic/Static Parse Tree Comparison to Mediate SQL Injection Attacks within a Selected West African Case System and Context.

EVANS DOGBE

A Master's Thesis Submitted in Partial Fulfilment of the Requirements for the Award of Master of Information and Communications Technology

Durban University of Technology (DUT)

Durban –South Africa

DEDICATION

I dedicate this work to my daddy who passed away before I started this thesis for his faith and contribution to my education with my mother, Elizabeth Dogbe.

DECLARATION

Name of student: **Evans Dogbe**

Student ID: 21242738

I, hereby, declare that this research project is the result of my own research, except for quotations and summaries which have been duly acknowledged.

Signature: …. ..... Date: Apr 28, 2020……

(**Author**)

Signature: … Date: Apr 28, 2020……

Prof. Richard C. Millham

(**Supervisor**)

ACKNOWLEDGEMENTS

I would like to thank Prof Richard Millham for his contributions to this research, and I would like to thank all my family members for their patience and prayers. I would like to thank DUT for allowing me to study in South Africa.

ABSTRACT

Business legacy systems, when migrated to the Web, often face increased chances of Structured Query Language (SQL) injection attacks; these attacks are compounded when this system lacks proper security mechanisms and security training for its staff. This study seeks to determine how the researcher's new theory of amalgamating two established techniques for defence namely; fine-grained Role-Based Access Control (RBAC) and static/dynamic parse tree comparison; can be combined to form a single centralized defence in order to effectively mitigate SQL injection attacks in a web-based environment, using a selected recently migrated legacy system as an exemplar. This proposed defence first involves redefining existing RBAC security to a fine-grained RBAC to act as the first tier of defence. Those queries, legitimate or not, which successfully pass through the first tier are analysed by the second tier of defence that is designed to both do a static and dynamic parse tree analysis and comparison of the queries in order to identify legitimate queries from illegitimate queues. During the study, it was discovered that the basic RBAC in control system and the fine grained RBAC could only mitigate a fraction of the selected test cases and thereby generated a number of false positives but no false negatives. However, those false positives were successfully identified and mitigated by the second tier of static/dynamic parse tree comparison. As such the measurement of performance using precision, recall and f-measure were determined in three cases namely basic RBAC defence in control with 31% precision,100% recall and f-measure of 32%; Fine grained RBAC without dynamic parse tree comparism with 54% precision ,100% recall and f-meaure of 54% and hybrid defence of fine grained RBAC and dynamic parse tree comparism with 100 % precision with a 100 % recall and f-measure of 100% with the test cases used in a repeated experimentation. However extensive real-world testing might expose weaknesses not observed during experimentation and such is the recommendation of the study.. This entire

approach is centralized in a security aspect in order to easily incorporate it into vulnerable newly migrated legacy systems to the web which requires minimal training of security staff for deployment. The hybrid was then tested using a case sample system that represents the West African context of inadequate security mechanisms and poor staff training. Standard test cases were used to test each defence tier in the hybrid as well as the individual tiers. This testing detected and halted illegitimate SQL queues and demonstrated this aspect's effectiveness and suitability for the West African context.

Contents

ACRONYMS AND ABBREVIATIONS

GPRTU: Ghana Private Road Transport Union

RBAC: Role-Based Access Control

SQLIA: Structured Query Language Injection Attacks

ICT: Information and Communication Technology

PT: Program Tracing

TRIG: Triggers

eCommerce: Electronic Commerce

LAN: Local Area Network

WAN: Wireless Area Network

ANNEXURES (LIST OF TABLES, FIGURES, GRAPH LABELS)

List of Figures and Their Names in Thesis

List of Tables and Their names in Thesis

Label for Graph in Thesis

1 CHAPTER ONE

1.1 Introduction

Ghana, a western African country, has a population of about 30.6 million people(Worldometers 2019).According to (Agyeman 2013), the majority of Ghanaians cannot afford to buy and maintain personal vehicles. As a result of this fact, many Ghanaians rely on public transportation systems. To ensure quality transportation service delivery, the safety of passengers, unity of vehicle owners and their drivers, a nationally recognized union manages bus stations where both vehicle drivers and passengers conduct business. The nationally recognized association that oversees private commercial vehicle owners and their drivers is the Ghana Private Road Transport Union (GPRTU), which is a member of the Ghana Trades Union Congress (an organization for all workers groups). The union plays a significant role in the governance of public transportation services in Ghana. Commuters rely on the efficient and effective service of this union to make their way to and from work, therefore, making them a major driver of the Ghanaian economy. The GPRTU manages stations, drivers, vehicles and passengers for both inter-city and intra-city transportation. The union serves as a monitoring system to allow vehicle owners to check on their drivers. In the past, all of the operations of the union were manually managed. However, with the increased introduction of computer software technology, the union is slowly moving its manual operations to digital platforms. The volume of cars and owner tracking with the number of tickets sold have led to some busy bus stations creating their personal stand-alone systems to manage their work load. To manage new demands from drivers to buy tickets online and to coordinate inter-city travel and rates, standalone systems although still functional and contained a lot of data and high switchover costs, were migrated to the web. The security of migrated independent systems must be enhanced to meet new threats that this web migration poses (Dogbe, Millham and Singh 2013).

1.2 General Objective of the Study

The aim of this study is to determine how fine-grained Role-Based Access Control (RBAC) and dynamic/static parse tree comparison can be combined to prevent SQL injection attacks in a web-based environment for a selected legacy system, which has been recently migrated. A legacy system will be used as an exemplar to test the efficiency of this approach.

1.2.1 Specific Objectives of the Study

To achieve the above aim, the following objectives will be addressed;

- Determine the most applicable SQL injection attacks on the migrated legacy system in its new Web-based environment.
- Develop a fine-grained Role-Based Access Control (RBAC) security mechanism that uses task, operation and data partition to access web resources to replace the basic RBAC security approach currently used in the legacy system.
- Develop a combined fine-grained RBAC and dynamic/static parse tree comparison into a single defence shield against SQLIA for the Sub –Saharan Africa context.
- Centralize this hybrid defence shield security into an element for easy deployment, administration and reuse within the West African context.

1.3 Problem Statement

There are over 4.5 billion internet users around the world (InternetWorldStats 2019). From this number, it can be inferred that there are over 4.5 billion potential hackers with continuous open access to web applications online in general as no user can easily be excluded as a potential hacker. With so many potential hackers, according to (Olmstead 2018), global cybercrime costs at least 45 billion dollars annually. Therefore, given the high cost of cybercrime, there is a genuine concern about the kind of security solutions that could be made available and be managed by different organizations across the world.

Some of the solutions that have been proposed by experts to resolve the hacking techniques that have been identified are not easy to employ in the West African context. Therefore, this study will seek to find a manageable approach that seeks to help curb and contain one of the most common hacking techniques, namely Structured Query Language Injection Attacks (SQLIA), which has been known to plague online systems for decades (Olmstead 2018).

The general problem that this research seeks to investigate is which solutions provide efficiency, cost-effectiveness and ease of implementation of SQLIA attack defence in the web environment within the context of West African systems and constraints. As stated earlier, legacy systems that migrate to the web environment need further security protection as the threat base of these systems exponentially increase given increased user access.

1.4 The Relevance of the Study

There is a continuous increase in African education, both formally and informally, in the use of computers and the design of computer software. It is often the case that, web applications designed in Africa are not adequately sophisticated to guarantee their security or at least enhance the web security of those applications. Due to lack of education and funds, security concerns are not given much consideration in the design of online systems or applications, and therefore, these systems are usually vulnerable to attack. However, this research will not address the reasons why these problems exist or solutions for them. Nevertheless, the study focuses on how to resolve a small selected part of the system vulnerability problem (SQLIA) using efficient but simple techniques with ease of implementation requirements so that inexperienced programmers can still be equipped to handle some of the security issues that plague online applications.

1.5 Background of the Legacy System

The legacy system of the GPTRU was a system designed primarily to help to assist the association with the tracking of drivers who have bought tickets and to significantly reduce the cost of paper usage in the management of drivers and vehicle owners at the Achimota station. The system used a password restriction based on Role-Based access to determine which screens a user with a specific role could be granted access to use. For example, if you were a clerk tasked with inter-city transportation, then when you log into the system, you would only see screens that are directly associated with your role and not the entire system. This legacy system reduced the use of paper and also allowed tickets to be printed in the office with a strategy to easily track who, when and what kinds of tickets were purchased and the routes they were purchased for at a glance. However, it did not reduce the queues the drivers had to join every morning to get tickets and also it did not provide mobile access to both drivers and vehicle owners (Dogbe, Millham and Singh 2013).

In the recent past, West Africa has generally had an increasing flood of mobile phone and smartphone technology. According to GhanaWeb (2018), 29 million Ghanaians out of a population of 30.6 million Ghanaians (Worldometers 2019) use 34 million mobile phones to assess the web. As a result, it is not uncommon to find some of the latest and most expensive phones being used even in the remote parts of the country. However, the availability of mobile technology has not been used to provide web-based functionality to drivers, vehicle owners

and passengers by the GPRTU. The reason for this non-adoption is mostly economic but also includes security issues. However, the Achimota branch of the GPRTU has migrated its legacy system to a web-based system. Nevertheless, this newly-migrated system, based on the original windows-based system uses Role-Based Access as the only defence mechanism for authentication to prescribed screens that correspond to the legitimate passwords for those screens allowed in that role. There were no additional layers provided to protect the system over the internet (Dogbe, Millham and Singh 2013).

1.6 Significance of the Study

The findings of this study could be beneficial in alleviating one of the most common kind of cybercrimes, SQLIA and its aftermath, on the internet today. The approach used in this study affords inexperienced programmers, primarily in third world countries to try other ways of securing their systems without comprehensive knowledge in database systems. The study will look at how to secure data through the application software using the combined benefits of fine-grained RBAC and dynamic/static parse tree comparisons. In addition, the idea of the centralization of security into one mechanism offers ease of implementation with minimal customization to other similar applications in terms of the degree of security provided. For the managers of GPRTU, they will have a security mechanism that protects them from one of the most common attacks on the internet. With this protection, they are then confident to migrate more of their operations online. The online functionality affords drivers ease to buy their tickets and prevent queues from being formed in the station for ticket purchase. Also, vehicle owners can check on the activities of their drivers with ease.

1.7 Structure of the Thesis

This dissertation is structured into five chapters. Chapter One outlines the problem statement, which is the foundation of the study. The significance of undertaking the study is also highlighted in the chapter. The background to the study, along with specific literature related to the study, is discussed. The research objectives of the study are also examined in this chapter.

Chapter Two comprises the literature review, which explores legacy systems and its problems, especially in terms of system migration to the web. It examines the different system defences available and the various types of threats these systems face. One of these threats, SQLIA, is discussed in-depth along with a critical examination of the available defence strategies and tools in terms of their strengths and weaknesses. One such defence, a hybrid multi-tiered

approach, is explored with respect to its effectiveness and usefulness within the West Africa context.

Chapter Three presents the -research methodology. This chapter addresses the research methodology and design used to conduct the study. The focus is on the design and description of the techniques used in the study. It explains how the SQLIA injection test cases were selected and outlines the various categories of SQLIA. Also, the chapter expounds on the procedure adopted to conduct the testing of the techniques outlined and the metrics used to ascertain whether or not success is achieved using a case system as an example.

Chapter Four presents the analysis of the study's results. The analysis and interpretation of data from this study's experiments are presented in this chapter. The study employs tables and a graph to represent the data's findings. Chapter Five comprises the findings and conclusions of the study. This chapter discusses the findings, how the objectives of the study were met, and offers recommendations for future research.

2 CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter evaluates related work to identify an effective SQLIA defence mechanism that is feasible for the West African context. The chapter starts with a brief introduction to the contextual framework of ICT development in West Africa in general, and then it moves on to describe the growth of ICT and its impact generally for businesses in West Africa, particularly Ghana. Also, the introduction of SQL injection attacks and their categorizations together with their defence and gaps are discussed. Finally, a suitable defence model that suits the Western African context is proposed.

2.2 Contextual Framework

Information and communication technologies (particularly computers and the Internet) are important resources for socio-economic advancement in both developed and developing countries (Helo, Gunasekaran and Rymaszewska 2017). However, developing countries are faced with enormous challenges in their ability to utilize these digital resources for their growth and development. Some of the challenges they face include infrastructural constraints (Paul and Das 2017), inadequate implementation of basic security requirements in software, and inadequately skilled IT professionals, among other limitations (Patrick and Fields 2017). In spite of these challenges, there is a high demand for Internet access in order to access these digital platforms for basic operations such as data processing and the offering of services to clients through eCommerce websites (Amankwah-Amoah 2019).

This high demand for Internet access, according to (Amankwah-Amoah 2019), is primarily because of the introduction of smartphones in West Africa. This introduction has encouraged more businesses to adopt a web presence, via the creation of a web client business interface, to serve the growing number of online users (Amankwah-Amoah 2019). Technically, web interfaces interact with web users, and their input is processed and stored into databases. These databases form an integral part of web applications. However, it is sometimes a challenge to identify legitimate users who interact with backend databases from illegitimate users (Liu et al. 2019). Given that trade secrets and critical customer biographical and financial information

could potentially be stored in these databases, these systems must be able to identify legitimate users from illegitimate users. Computers cannot on their own without instruction, manage the specific interests of business stakeholders as far as data security is concerned. This is owing to a broader information technology concept proposed by De Martin (2019), in his study as GIGO, which stands for garbage in, garbage out. In that, computers are limited to doing what they are designed to do, and hence, flawed designs lead inevitably to flawed outcomes. Therefore, the successful designing of quality security-conscious web applications is highly dependent on the skill and knowledge base of developers; developers in West Africa lack this knowledge and skill base (Dogbe, Millham and Singh 2013).

For example, a less secure web application design may allow crafted injection and malicious update, which may cause confidential data to be accessed by unauthorized users. The growth of less secure web-enabled applications is almost parallel to an increase in attacks targeting web applications due to irregular security updates and in most cases due to bad password management policies or very weak passwords (Serianu 2017).

This suggests that the lack of updated security measures and weak password policies have exposed web applications and business databases online to different kinds of attacks and have increased the relevance of the measures needed to mitigate the damage.

An apparent reason accounting for the increased number of attacks is an increase in web applications handling traditional day to day business transactions such as booking a bus ticket, buying pizza online, etc. Consequently, attacks increase as hackers have more potential targets (Kshetri 2019).

However, the increase in attacks may also be attributed to the ease of Structured Query Language Injection Attacks (SQLIA) (Dogbe, Millham and Singh 2013). Although one could make all the security updates needed for the newest operating system managing the web servers running a business' website and have the best password policies in place, it will all be redundant if an attacker succeeds in modifying the code of the website or mobile application in order to ignore the password restrictions entirely (Serianu 2017).

SQLIA has been around for a long time, but the defences require expertise and the resources for the implementation needed to secure websites is mostly ignored in impoverished countries that have their priorities mainly and rightly on other pressing issues of survival such as food security, political stability, electricity, and supply of clean water (Hunter 2006). However, given that such places are the easiest targets usually because most people cannot afford or keep

up with the changing security requirement knowledge base that is necessary to run robust operations online. As a result, another approach must be found that best suits the West African context to resolve this problem (See Growth of ICT in Ghana in Section 2.3).

In order to explore and experiment with the efficacy of a proposed inexpensive security design for developing countries, a case system from a West African country, notably Ghana, was chosen as the basis for testing of our security model. Ghana, out of all the developing countries in West Africa, was selected because of the ease of access and availability of an unsecured system provided by a typically well-known organization that handles the main transportation system in Ghana, namely GPRTU. It is the overall goal of the researcher to find out how less resource-endowed businesses can mitigate damages to their databases from selected well-known SQLIA via an easily configurable security model.

Different defence techniques need to be evaluated against cost and technical implementation constraints to determine their feasibility for this West African context to achieve the overall goal of the study.

2.3 Growth of Information Communication Technology (ICT) In Ghana

Information and Communications Technologies (ICTs) is a term that includes any communication device or application, such as online banking and eCommerce (Agboh 2015). Generally, ICT is any technology that is used to support information gathering, processing, distribution and use (Alfred and Bett 2019).

ICT has greatly influenced the teaching and learning institutions from high schools to universities. It is now a common practice for students to perform research for their assignments, read literature on subjects of interest and type out their reports using computers. In some urban schools, students can even submit their assignments online. Therefore, the youth are not left behind in computer usage and are also targets for businesses that sell services to this type of demography (Dei 2018). However, the aspect of ICT, which is considered in this dissertation is online business applications implemented through the use of ICT and not educational sector, although the solutions discussed and found could be applied in other sectors.

Businesses in Ghana have witnessed an impressive growth in utilizing ICT for their operational needs and have improved sales and customer services through online interactions. It is common to find in most businesses using computers or smartphones for sending and receiving emails,

responding to queries on social media platforms, writing blog posts on services and reading comments and service recommendations as well as complaints from customers (Agboh 2015).

Another interesting development of ICT growth is the level of mobile phone and mobile technology usage. Two decades ago, mobile phones and mobile technology were almost non-existent, but in recent times, mobile phones and mobile technology usage have grown by 128% per cent because for a population of 30.6 million Ghanaians, there were over 35 million registered active mobile phones in 2016 alone. This figure is estimated to go up in the next decade (Laary 2016). This growth has led businesses to rely on the web and mobile phones to a considerable extent to deliver services to the majority of their customers either through online business applications or through mobile phone interactions. Another reason why businesses are adopting ICT solutions for their traditional business operations such as customer communication is that it is relatively cheap compared to the manual system in the long term, and information dissemination to their customers is almost instantaneous (Agboh 2015).
Due to the pervasiveness of ICT in all spheres of life, including business, there is a steady cultural change in Ghana with respect to ICT adoption.

2.3.1 Societal Cultural Change towards ICT Developments

The massive introduction and popularity of mobile phones and technologies, have considerably changed the lives of Ghanaians. The traditional method of writing letters and posting them has become obsolete in recent times as most people communicate through inexpensive texts using their mobile phones (Laary 2016).

Also, according to Boaheng et al. (2019), the ability to share information using mobile phones has partly developed the computer skills of Ghanaians who even dwell in rural areas. It suggests that most people could use eCommerce solutions without the need for extensive training. Given their newly-acquired digital skills and familiarity with eCommerce solutions, they now expect to use online applications rather than traditional manual systems to receive their services from vendors.

Therefore, one could say that as a result of continuously benefiting from mobile technology usage, the expectations and social culture of ordinary Ghanaians have seen a paradigm shift, which have consequently opened new business avenues to Ghanaians.

2.3.2 Business Opportunities Created as a Result of Societal Change towards the Use of ICT

The proliferation of ICT has provided a unique opportunity for small and medium-sized enterprises to conduct business electronically, be more competitive, and conduct business in a global environment(Leckson-Leckey, Osei and Harvey 2011). The key outcome is the opportunity that has been created for businesses to expand their business operations as a result of the growth in mobile phone usage, online and social media activities, and increase in the basic computer skills of ordinary Ghanaians. Businesses (both small and medium enterprises) have realized the positive impact of ICT as it helps to increase sales revenue; provide better communication via e-mail and mobile phone; and offer the automation of manual processes, leading to faster response times (Agboh 2015). Because of the ICT and societal changes experienced by Ghanaians, they can now instantaneously advertise their products and services directly to their customers through inexpensive text messages, YouTube channels, and their websites usually for a small fee.

Given the cultural changes and impact ICT has had on the country and its businesses, the government has also made some strides towards the adoption of ICT.

2.3.3 Government Legislation and Regulatory Frameworks for ICT Development

Mangesi (2019) indicates that Ghana is part of the first African countries to reform their ICT sector and establish the necessary regulatory framework to support the growth of the sector. Since 1990, the government of Ghana has liberalized the telecommunications sector with the aim of enabling the private sector to actively participate in the provision of communication services to the Ghanaian populace. These services include internet and mobile communication to increase access and coverage, introduce value-added services, and boost consumer access to state-of-the-art technologies (Mangesi 2019). This increase in access and coverage of internet and mobile communication meant increased infrastructural development across the country, and the introduction of competition within the telecommunications sector, which has lowered tariffs for consumers. Also, the government has put in place general regulations and restrictions on the types of pricing structures and services that can be offered by these telecommunication organizations. These regulations are generally designed to protect consumers from exploitation by these giant mobile companies.

2.4 Legacy Systems

Despite the advancement of ICT in the world, most organizations in Africa still rely on outdated systems, often called legacy systems. Some organization continue to use these systems because of their initial investment in them, and their reliance on them to manage their organizations' core internal business processes and functions (Pei Breivold 2019). However, these systems are surrounded by faster and better modernized systems that interact with multiple technologies in the business environment; consequently, putting pressure, in terms of data access and processing, on legacy systems to deliver and work with these modern systems. Typically, these legacy systems perform core and critical business functions that have very expensive replacement and switchover costs, which make it almost impossibile to replace these systems (Kumar and Abdul 2019). When old systems, which were originally designed to be standalone, are moved onto a new system such as a networking environment that includes a new interface, there are often problems with the new demands and threats posed by the new interface and environment (Sadotra and Sharma 2017). One of the reasons for these problems is that legacy systems are designed to operate as standalone systems with their own data manipulation, rules and function calls (Millham 2010). They are not usually upgraded in terms of security to handle the wider domain of users who input data as well as threats from new forms of system attacks, particularly online. However, the dynamic business environment requires systems to be integrated to share information and process information at faster rates with some level of security for data (Parnami, Jain and Sharma 2019).

In addition to the involvement of government in ICT development nationally, there are many businesses whose core business functions are still being run by old systems or legacy systems (Udunwa Ugonna Anthony *et al.* 2019)

2.4.1 Legacy system migration

Legacy system migration is a costly procedure that carries a definite risk of failure (Kumar and Abdul 2019). Even though there are common cost and risk factors associated with running businesses both in developed and developing countries (Kumar and Abdul 2019), developing nations usually face more cost and risk factors in the migration of legacy systems. This situation is due to low IT investment and skill levels for ensuring that the systems are secured. The cost and risk factors make it unfeasible for businesses to consider the migration of their legacy systems as a possibility.

Agboh (2015) indicates that businesses operating on a small scale suffer a severe scarcity of resources (e.g., financial and ICT skills) compared to larger organizations. Therefore, incurring additional costs and risks to migrate a legacy system to an online system for these smaller businesses is not a feasible option. In addition, the migration of a legacy system to an online system brings with its increased risks of being hacked due to the high exposure that online systems face. Consequently, given the increased threats and the West African context of low skill and limited budgets, the solutions for mitigating online security risks must consider easily configurable components that require very minimal training and skills to implement so that small businesses can have the opportunity to be secured online. The most common security threat encountered by businesses online is SQLIA (Taylor and Sakharka 2019).

2.5 Understanding SQL Injection Attack (SQLIA) Online

Out of the online security attack threats faced by businesses, the commonest is a code injection technique known as Structured Query Language injection attacks (SQLIA) (Sadotra and Sharma 2017). There are different definitions for SQLIA in the literature. According to Hartley (2012), SQL injection attack (SQLIA) is a database security attack or threat in which a segment of SQL code is inserted or appended into application/user input parameters that are later passed to a back-end SQL server for parsing and execution. Another definition offered by Agarwal and Sirsikar (2019) says that an SQL injection attack is an attack that exploits a security vulnerability in the database layer, network layer or at the application input end of an application (through the use of database access mechanisms such as queries).

Agarwal and Sirsikar's (2019) definition is more comprehensive than that of Hartley (2012). This is because depending on how the attacker performs the SQLIA, it might not involve using an application input parameter at all (more details would be given on this later in the section).

Before attempting to understand SQLIA, it is essential to understand the fundamentals of thin client and n-tier-server architecture, commonly used in web applications.

Consider, the diagram in Figure 1

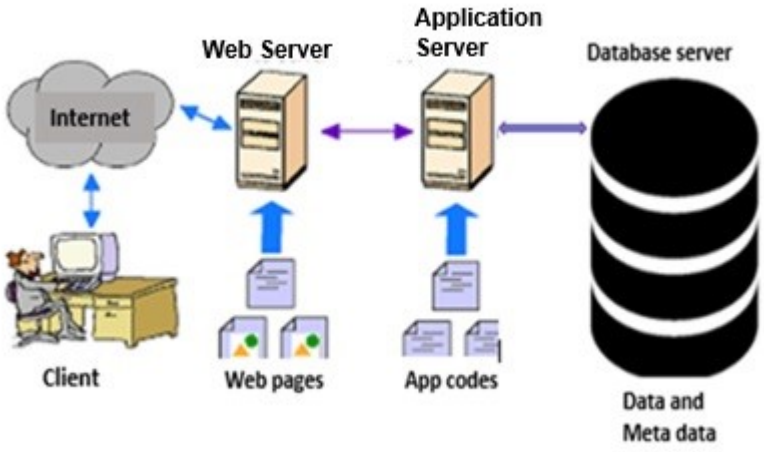


Fig 1 Understanding SQLIA environment

In Figure 1, the user with a desktop computer and its browser serve as a client. When a client processes data on a different system other than their own, then it is called a thin-client (Marmignon 2019). This thin-client is the commonest architecture used in West Africa (Kshetri 2019) and, forms the exemplar, which represents mobile-based n-tier systems that were used in this research. For ease of understanding of how things work in this structure, the fundamental jobs of each server are separated, even though in practice they would be configured to run on one system in most cases. As illustrated in Figure 1, the client accesses forms on a website through a browser, and the pages requested are provided to the client by the web server through the internet. Then, processing of the data entered by the client is done on the application server through a network connection to the webserver. Afterwards the application server, in turn, may have back and forth communication with the database server to complete its tasks through a network connection and all processed results are sent back to the client that made the request.. Figure 1 shows three contact points of data in the architecture, namely the client's end(browser), the network connections (internet and LAN or Wan) and the database end. SQLIA attacks may happen through all three data point contacts

Using the thin-client architecture, a potential attacker could try to infiltrate the system by injecting codes through the user interface of the client in the form of a client-side attack

scenario. This type of infiltration is the focus of this research, and serves as an example of Hartley's definition of SQLIA.

Alternatively, the attacker could try to infiltrate the system by monitoring and capturing unsecured network packets using tools like Wireshark (Sikos 2019). This tool, among others, allows the attacker to see the queries sent to the database server in plain text. Therefore, it can potentially be altered by the attacker to compromise the system (in the network connection attack scenario, which is an example of Agarwal and Sirsikar's definition). This could simply be stopped by encrypting all the network communications; this is, however, outside the scope of this study. Last but not least, the attacker could try to infiltrate the database server using a guest account, for instance and, through SQLIA try to expand resource privileges from that guest account to perform unauthorized data manipulations through a direct attack on the database (this is another example of Agarwal and Sirsikar's definition). This could be prevented in most cases and at least minimized by developing strict roles and rules on the database, and also by strictly following schedules for upgrading outdated databases with constantly available security patches from database vendors. However, the exploration of this option is out of the scope of this study, and so would not be further discussed.

2.6 SQLIA via Client or Application End

SQLIA that happens through the client-side could happen through a web application or simply a browser; this is the commonest form of SQLIA Hartley (2012). An attacker could inject a special code that has meaning to the database and, in combination with the original developer's query, can alter the results or intent of the original developer. Thereby tricking the database into giving off information or executing unintended commands that are not from the original developer.

To understand how this type of attack uses a web form, consider the diagram in Figure 2

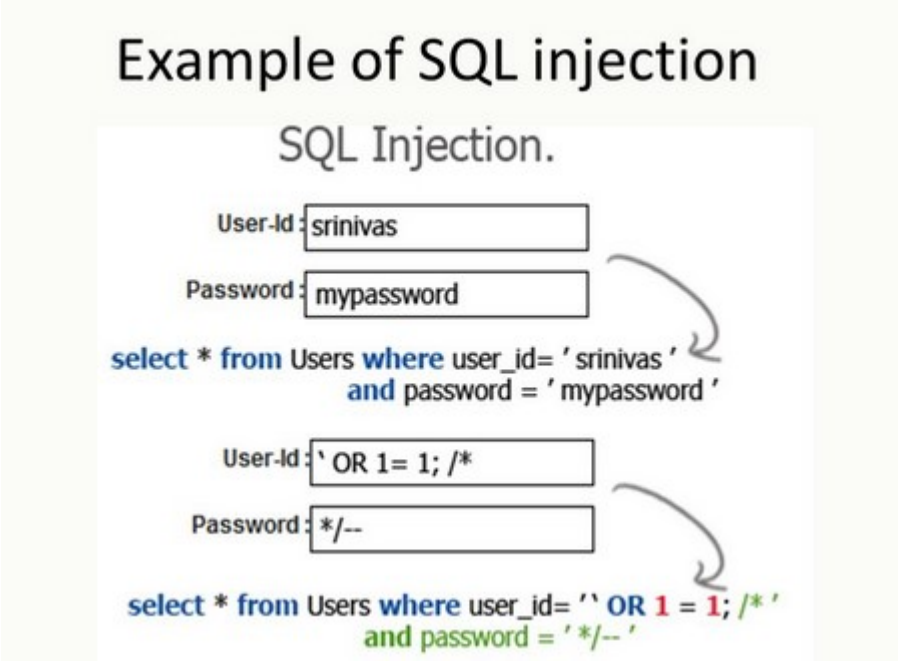


Figure 2 (Example of SQLIA); source: (Sadotra and Sharma 2017)

In Figure 2, the user is expected to enter a user-id and password as shown in the first part of the diagram (user-id: *srinivas* and password: *mypassword).* If the user does that, then the application uses these entered fields to form a resultant query in the database, which would be (select * from Users where user_id= 'srinivas' and password = 'mypassword'), returning true if the system finds a user that matches the user_id and password; "srinivas and mypassword", respectively in the database. However, if the user (attacker) decides to enter in the text boxes for input "' OR 1=1; /* " as user_id and " */ -- " as password (which is a tautology and comment SQLIA attack), then the resultant query will be (*select * from Users where user_id = ' ' OR 1 =1; /* ' and password =' */--* '). When interpreted by the database, if the username is nothing or has no match in the database (which would return as false) but because of the OR if 1=1 (which is always true) then the system returns a result as a match. The comment "—" after the password input tells that database to ignore any other SQL statements which would cause the rest of the legitimate query to be ignored, including matching the password to the user-entered password. In short, the where clause in the query allows filtering of records for matches in the database. The OR 1=1[tautology] in the clause allows every comparison in the database to be true regardless of whether any other comparison is correct or not. The '--' makes the parser to ignore any other text and is used to bypass any remaining legitimate part of the query (comment attack). This effectively means that the database engine will always evaluate

it to be true for any possible comparisons it would make against the user's table for matches; consequently, it will show all records in the user's table. This will grant the attacker access to all user names and passwords, and then the attacker can do unimaginable damage to the database with zero restrictions.

There are many variations of this attack and ways the attacker can use to steal and perform malicious activities on the system. The capacity to detect these types of attacks depends on the skills of the developer and the available supporting defence mechanisms. However, in the Western African context (see Section 2.2 Contextual Framework on West African Context), developers have limited skills and support structures to implement preventive structures that are complex to detect and prevent such an attack, and SQLIA from this point forward will be assumed to be in this context per this research.

2.7 Categories of SQLIA Attack

There are different types of SQLIA. The types of SQLIA are discussed using a two-step process as a simple methodology for a potential attacker to follow.

Step 1: Information discovery (system enumeration or fingerprinting)

Step 2: Information altering, stealing, insertion and deletion operations.

In Step 1, the attack runs operations to learn more about a database such as finding the exact table and column names used by the original developer in the database to tailor his attacks to be more effective. Step 2 is where the original code of the developer is changed to allow unauthorized access into the database to steal or cause harm to the system.

Furthermore, SQL Injection attacks are grouped into three main categories in this research, namely: SQL Manipulation, Code injection and Function call injection. It should however be noted that there is no consensus on which types of attacks belong to particular categories in literature hence there is some overlap. Other researchers could classify them differently.

Consider Table 1(Sqlia categories)

Category of SQLIA	**Attack Type**
SQL manipulation	Union, tautologies and comments
Code injection	Sql server execute commands
Function call injection	Stored procedure attacks

Table 1: Category of SQLIA

2.7.1 SQL Manipulation

It is the process of modifying existing SQL statements by using different operations such as UNION (the database keyword used to combine SQL results of queries) to append original queries(Dadashzadeh 2020). SQL manipulation is the commonest category of SQLIA attacks because it requires minimal skills from the attacker as well as a weak defence system on side of the host. This happens to be the commonest threat experienced by West African based developers in terms of SQLIA (Lartey 2019). Therefore, the SQLIA attacks discussed by this study fall under this category.

2.7.2 Code Injection

One type of the code injection attacks is to append a SQL Server EXECUTE command to a vulnerable SQL statement(Rahman *et al.* 2020). The execute command allows the attacker to execute MS-Windows system commands, which can compromise the system (assuming it is a Windows platform which is the commonest platform used by businesses in West Africa) and allow free navigation of the resident computer housing the application (George, Jacob and James 2019). This can, however, be easily avoided by disabling the ability of the SQL server to run windows commands by default on the database server. The exemplar database already has this filter so it will be not considered further in the study.

2.7.3 Function Call Injection

It is the process of inserting various database functions such as system stored procedures into a vulnerable SQL statement. These function calls can make an operating system call or manipulate data in a database. However, these calls can very easily be filtered out at runtime (Kucherov, Braunschvig and Fliess 2019). Nevertheless, this is outside the scope of this study due to the ease of filtering.at runtime

In conclusion, SQL manipulation is the commonest because it requires a considerable amount of skill to filter and detect legitimate queries from illegitimate ones and will be the focus of the study since the other categories as easy to filter out.

2.8 Types of SQLIA and their Categories

There are several types of attacks. This section discusses some types of attacks and indicate the categories they fall under, and the aspect of the simple attack methodology that is used in this study (see Section 2.7 above).

2.8.1 Piggy-Backed Queries Attacks

Piggy-backed queries are queries that are injected as additions of originally legitimate queries to compromise a database (Varshney and Ujjwal 2019). It is one of the commonest methods of attacking systems since all the attacker needs to do is to add on to legitimate queries masquerading as the original query to the application. In this type of attack, an attacker injects add-on data manipulation operation keywords such as Union, Insert, etc., which are to manipulate or add records to the results of an original query. This type of attack falls under Code injection and string manipulation categories.

For example:

"SELECT * FROM Users WHERE UserName = “; lNSERT

lNTO Users (Continent, Name) VALUES (‘WestAfrica, 'ItDoesntMatter’) --"

The query uses semi-colons to separate queries via user parameters. This query inserts the record in the table. Once the table’s name is known, other manipulation operations like UPDATE and DELETE records in the table can also be performed in similar ways. It is worth noting that there is an overlap of SQL attacks in most cases, and these are not exclusive in themselves.

2.8.2 Tautologies and Comments

This type of attack has been explained already using Figure 2. This type of attacks do not exist on their own, they are usually combined with different types of attacks.

2.8.3 Logically Incorrect Queries

This type of attack falls under the SQL manipulation category in which an attacker can get an advantage from error messages generated from the database server (Agarwal and Sirsikar 2019). This technique is mainly used during the enumeration phase by the hacker (Step 1 of attack methodology). Enumeration is the exploratory and information gathering phase an effective hacker needs to go through to exploit a target successfully. However, this is only possible when the debugging option is left switched on by default. A developer can easily switch it off but this requires IT skills and knowledge that might not easily be available to developers in West Africa.

For example, the following query is used to extract table and column names for enumeration purposes:

SELECT * FROM Users WHERE Username = “HAVING1='1'; -- and Password='ItDoesNtMatter'.

With the help of this query, an attacker gets an error message like *"Column 'Users.UserID' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause* “. This error message shows the table name to be Users with a column name User ID. Through this method and a variety of these types of queries, an attacker can extract all the tables and columns needed to compromise the database of a website.

2.8.4 Union Attack Queries

This type of attack falls under the Code Injection and SQL manipulation category (George, Jacob and James 2019). Union Query adds a malicious code or injects a code with a legitimate query to get table information from a database server. It adds additional information to a legitimate clause to get other database information from the database server or can be used in conjunction with other forms of SQLIA for malicious purposes. Through this type of attack, an attacker can extract data type information of a column.

For example:

SELECT * FROM Users WHERE UserName = “UNION

SELECT SUM (Username) from Users-- BY Use rid

HAVING 1='1'; -- and Password='ItDoesntMatter'

An error message such as "Operand data type varchar is invalid for sum operator" Is displayed. In this error message, it is obvious that the Username has a VARCHAR Data Type. Therefore, using this type of attack, an attacker can easily discover and then extract all data types of columns found on the tables of the backend database. The union keyword can be used to get other forms of data from the database as well.

2.8.5 Stored Procedure Attacks

This type of attack falls under a function call injection category (Kucherov, Braunschvig and Fliess 2019). In this attack, an attacker can execute a built-in procedure using illegitimate values as parameters.

For example, a stored procedure in SQLserver2005 (pirate versions are available throughout Ghana and is more representative of the current situation (Yie 2019)) is given below:

CREATE PROCEDURE spDisplayTableRecords

```
(
@select NVARCHAR(500),
@NameOfTableNVARCHAR(500),
@WhereClauseNVARCHAR(500)
AS
BEGlN
EXEC (SELECT' +@select+
'FROM '+@NameOfTable +
'WHERE ' + @WhereClause
)
END
GO
EXEC spDisplayTableRecords '* FROM
INFORMATION SCHEMA.TABLES FOR XML RAW--','
NATE','DOGBE'
```

This procedure is vulnerable to injection attacks that utilize the parameter to get meta-data about the table and database structures, in the form of XML, by the use of the database system table, lNFORMATION_SCHEMA.TABLES.

An attacker can get XML data, which contains information of table and database schema as in the following example.

"<row TABLE CATALOG=" Users"

TABLE SCHEMA="dbo" TABLE NAME="User info"

TABLE TYPE="BASE TABLE" />"

In general, attacks that rely on using Exec on a built on string query using parameters, even though it is allowed by database interpreters or compilers, is a bad programming practice that can result in systems being compromised (Taylor and Sakharka 2019). The knowledge of not

using this poor procedure solves the problem, but this knowledge is unlikely to be applied in West African designed applications due to the lack of skills and current information on security systems. Also, having knowledge of just the data type and parameters limit the type of attacks that can be mounted.

2.8.6 Alternate Encodings

In this type of attack, an attacker manipulates the query by using a similar alternative encoding (such as hexadecimal, ASCII, and Unicode), which is used by the application to pass parameters from page to page (Raman 2019). This attack modifies the URL and its parameters in an attempt to trick the database to give off privileged information. The objective of these attackers is to bypass the filters in browsers or web applications which would normally filter out some of the commands and encoding techniques inserted by a hacker. This approach heavily depends on the notion that the developer will pass information from page to page using the URL but if good programming skills are used, such as not depending on URLs to pass information between pages but instead using cookies and session variables, then the attacker's attempts become fruitless since the system does not rely on the URL for data in decisions. Hence, the SQLIA is never interpreted or executed against the database. Also, with frequent browser updates, the encodings are easily filtered out, and therefore, the attacker needs to find new ways to attack the system. The exemplar does not parse information from page to page using URL parameters but instead uses session variables; as such, due to of this system's non-use of URL parameters, none of the SQLIA attacks will be successful in yielding any meaningful damage against the control system and proposed security mechanism design. Consequently, this type of attack is out of scope for this research.

2.8.7 Inference Based Attacks

This attack falls under code injection, and buffer overflows category (Manjunatha and Kempanna 2019). Due to this attack, an application or database may behave differently based on the result of the queries. These inference attacks can be categorized into two types, BLIND Injection and TIMING Injection.

2.8.8 Blind injection Attacks

In the case where developers hide error details by checking the "no debug" option so that attackers cannot get help from the database, the attacker, when using the enumeration type of SQLIA, faces a generic page provided by the developer instead of an error message as is expected (Manjunatha and Kempanna 2019). An attacker can still steal data by asking a series of True/False questions through SQL statements. For example, assuming URL

“http://evansdogbe.com/item.jsp?itemID=10“ corresponds to a valid page that displays some form of result from the backend database called “p”. In principle, if an attacker appends a logic preposition such as tautology statements, then depending on the response of the system, the attacker can know if an unknown is true or false. Assuming any result displayed on the page after an attack is different from result “p”, which was earlier “q”, then using the same example with a logic preposition we have; “http://evansdogbe.com/item.jsp?itemID= 10 and 1=1”. If the system displays the same page as p then it is vulnerable to SQLIA; the developer could try more meaningful attacks other than “1=1” like USER_NAME() = ‘dbo’ in place of 1=1 to attempt to know if the user is an administrator. In this case, if that is true, then the result of p is displayed else the result of q is displayed. Knowing this an attacker could intuitively guess names and then check with the system whether they are true or false; this could potentially compromise the system through the knowledge slowly gained. This, however, is heavily dependent on the URL attack, which is out of scope for the present study, and it could be filtered out as a remedy.

2.8.9 Timing Injection Attacks

A timing attack allows an attacker to gather information from a database by observing timing delays in the database’s responses. This type of attack works even in secure web applications, as they rely on the delay that occurs due to the execution of an injection statement and not on the output of the application (Mishra 2019). It is very similar to blind injection as an attacker can measure the time a page takes to load to determine if the injected statement is true. Firstly, the attacker opens a legitimate page of the web application and observes the time it takes to load. After he gets to know this, the attacker tries to reload the same page with alterations and questions for the system, typically in the URL, and use tautologies and time delay commands. If the web page takes longer to load than the initial time recorded, the hacker gets to know that the questions/alterations are true in the instance of no difference and false in the instance of significant time difference with the initially recorded average page load time. The delays are used to brute force the application to give out information not originally intended by the developer (Mishra 2019). This is heavily dependent on the URL and could be filtered out with an exhaustive list; hence, it is out of scope for this study.

2.9 Defence and Prevention of SQL Injection Attacks

There are many ways to prevent SQL Injection attacks. Prevention depends on the correctness of the input value, which is supplied by the client or user at the coding level. These techniques force the client to enter correct data and can be barred to enter illegal values that are harmful

to the database server. This type of prevention can be done at both sides whether at client-side or server-side, but SQL injection cannot be prevented completely with this technique as one cannot guarantee the completeness of allowable data input. The tools and techniques for detecting and preventing SQL injection are given below.

2.9.1 Runtime Monitoring Technique

AMNESIA technique is used for detecting SQL injection attacks over web applications (Halfond and Orso 2005). This technique works on both a static approach and runtime monitoring, and it is perhaps the most cited tool in this category by 674 academics worldwide, according to Google Scholar as of November 2019. It detects injected queries before they are executed on the database server using a model-based approach. This approach has two parts; the first part has a static part that automatically builds legal queries with placeholders (models for user input) using program analysis. The second part builds dynamically generated queries from actual user input and then compares them (query from actual user input vs query with placeholders for user input) for matches with their static counterparts, and then mismatches are deemed as attacks. If queries violate the model, then this approach prevents the execution of the queries on the database server. This technique, however, requires the developer to know how to configure it on the server to work properly and also requires knowledge of the inner workings of the tool so that if it breaks, they could fix it (Dogbe, Millham and Singh 2013). All of these are disadvantages, and the skill required is highly unlikely to be available considering the West African context.

2.9.2 VIPER for Detecting SQL Injection Attacks

In this tool, the technique used to detect SQL injection attacks is by employing practical methods that are not guaranteed to be perfect but sufficient to gain knowledge that is instrumental for compromising the target or web application. This is known as heuristic-based approach (Alsahafi 2019). This approach is broken into 4 phases. The first phase involves a web application structure discovery that is done through the navigation and downloading of web pages of a site as well as following all its hyperlinks. The second phase consists of the determination of all the input parameters and their location on specific pages, basically identifying the possible entry points for SQLIA, also known as hotspots. In the third phase, the tool injects standard queries from its knowledge base, which are not dependant on the application, with the objective of causing an error and then matching the error message using regular expressions in its knowledge base to determine which database, SQL dialect and details of the database are being used by the target. The last phase logs all the activities and determines

which attacks were successful and which pages are vulnerable etc. and then refines the attack; after this phase three is re-initiated. This integration of phase 3 and 4 are done with the knowledge presented by the error messages using 15 patterns of error messages in 5 different databases until at least parts of the database structure is known. The main disadvantage of this tool is that it can only work with five database management systems (DBMS) and can only identify 15 patterns of error messages produced by this DBMS. It is not an exhaustive method. The tool was designed using Perl and requires skill for developer to use it; these skills are not likely to be easily available in West Africa.

2.9.3 Parse Tree Validation Technique

This technique maps out the query structure of all static queries in a web application as it breaks the queries into tokens and categorizes them according to their types as being either a SQL keyword or an input parameter into a tree structure (Dogbe, Millham and Singh 2013). It repeats the same process at runtime for all resulting queries after inputs are provided by a user and compares the runtime queries with the static ones to determine whether they have changed in structure. A change in the structure indicates a possible attack and no change is taken to mean that no attacks have occurred. The generation of the model structure of the queries during programme compilation is called static analysis, and the generation of resultant queries at runtime is called dynamic analysis. The main advantage of this approach is that it is relatively easy to understand and implement by developers. However, the main disadvantage is that it needs a safe application environment for it to be effective. For example, the network that runs the application needs to be encrypted, so that safe queries are not altered by hackers executing a man in the middle attack after they have been analysed. Alteration by the man in the middle can be used in any tool and so we assume that the network will be encrypted. There are also many free solutions online for encrypting network traffic (Yassein, Aljawarneh and Wahsheh 2019) that is available to West Africans, and therefore, the reason for our assumption.

2.9.4 SQLrand Approach

This approach is proposed by Stephen W. Boyd and Angelos D. Kero(2017), and like AMNESIA, it is also well respected with 516 citations by academics worldwide according to Google scholar. The principle used by these authors is to provide a tool that helps developers to append SQL standard keywords with random numbers that are difficult to guess by a potential attacker (Boyd and Keromytis 2004). After the application is appended with randomized queries, a proxy is set with instructions for de-randomizing the queries in the web application and the results of the de-randomized queries are sent to the database for

computation. The proxy itself does not give any error messages, and if attacks are performed at runtime, the proxy can easily identify them since the attacker is unlikely to guess the random numbers attached to each SQL keyword used in his attack. One of the problems of this approach is that it cannot prevent attacks that are stored procedure based (Alsahafi 2019). In addition, other problems with this approach are that it assumes that de-randomization queries sent to the database are safe. However, if the connection between the proxy and database used for computation is compromised, then SQLIA can be fairly easily introduced.

2.10 Classification of SQLIA Defence

This section groups the methods of defence used against SQLIA into two; classification by detection principle and classification by analysis method. The classification by detection method is concerned with the identification of SQLIA in a broad sense of security. It, therefore, covers other security vulnerabilities other than SQLIA, while classification by analysis method is primarily focused on SQLIA. The methods are discussed with a focus on the West African context.

2.10.1 Classification by Detection Principle

The detection principle focuses on the concepts behind how existing security vulnerabilities, including SQLIA in a given web application, are identified. The principles are categorized as signature-based and tainted data flow (Li et al. 2019).

2.10.1.1 Signature-Based Detection

This principle uses regular expressions to describe attack patterns known as signatures. The techniques that implement this principle build and maintain a list of all possible attack signatures and during runtime all user input statements are compared with the list of signatures to identify and avoid SQLIA attack patterns (Li et al. 2019). However, developing good regular expressions to identify attack patterns requires skills and experience to detect all possible attacks. Hence, this requirement might not be suitable for defence in the West African context. Additionally, the timely identification of an attack and its inclusion on the list of signatures is key to the effectiveness of this defence method. This might produce high false-positive and high false negatives since some injections might not be detected promptly, and some legitimate queries may not be allowed to execute. Hence, this principle may not be the most suitable approach to defence.

2.10.1.2 Tainted Data Flow Detection

Tainted data is input from the user, which is always treated as malicious. The main principle behind tainted data flow detection is to detect whether a tainted data has reached a vulnerable point within an application (Talukder et al. 2019). This vulnerable point is called sensitive sinks. There are two variants of the tainted approach.

2.10.1.2.1 Positive Tainting

The approach identifies and marks trusted data user inputs to make sure queries are built using only trusted data. It only allows trusted data to form semantically correct SQL queries such as SQL keywords and operators (Talukder et al. 2019).

2.10.1.2.2 Negative Tainting

The approach identifies and marks un-trusted data from untrusted sources instead of trusted data (Talukder et al. 2019). In that variables manipulated by users (saved user inputs value from forms), which are used to construct SQL statements are marked as potentially dangerous or untrusted without sanitization. The approach tracks outputs that are built as a result of untrusted data sources (variables that can be manipulated); however, neither positive nor negative tainting can guarantee the completeness because the approach relies on identifying all possible positive user inputs (positive tainting) or all possible negative user input (negative tainting). The percentage that is not identified inevitably becomes false positive or false negatives, making this approach not very effective. Hence, this is not suitable for the West African context

2.10.2 Classification by Analysis Method

This section is categorized into two categories. The first category describes different analysis methods, which are defensive coding (known as secure programming), static analysis techniques, dynamic techniques and hybrid analysis (Feng et al. 2019). The second category selects the analysis methods that are most suitable for the West African context based on two criteria; firstly, additional implementation infrastructure (any supporting resource in the technique environment outside of the technique direct resource) and skill (development and deployment); while the second criteria looks at the efficacy of the technique (high false positive and false negative). These analysis methods are discussed based on their popularity and the empirical review available on their robustness in deployment and their technical requirement from the literature (Feng *et al*. 2019).

2.10.2.1 Secure Programming to Develop a Defence against SQLIA

This technique uses API (Application Programme Interface) that enables a programmer to generate SQL statements automatically. This approach prevents a web developer from using string manipulation, which can cause SQLIA in poorly formulated string statements (Feng et al. 2019). Secure programming is also referred to as defensive coding. It allows the developer to use pre-compiled functions that help build SQL statements to be used with place holders as inputs. The advantage is that the input values are treated by the application as input with data types and not part of the SQL statements such as the keywords used to build SQL statements as seen in parameterized queries (Feng et al. 2019). Therefore, any input attack is simply seen as a string and not as part of the programme code because of pre-compilation. The main disadvantage of this is that it requires skill and knowledge to properly use it, and therefore, it is not likely suitable for our context.

2.10.2.2 Static Analysis Technique to Develop a Defence against SQLIA

This technique examines the source code, binary code, byte code, and configuration files to analyse data flow at runtime without executing the code (Agarwal and Sirsikar 2019). The analysis involves a complete perusal of the entire source code for vulnerabilities using known patterns of data flow which might be exploited. A scanned report generated with respect to vulnerabilities might lead to numerous warnings that are vague and difficult to interpret by programmers. It is also a time-consuming process as it must be done for every code change.

2.10.2.3 Dynamic Analysis Technique to Develop a Defence against SQLIA

This technique involves observing and analysing software behaviour during runtime with the objective of detecting bugs and vulnerabilities using actual dataflow and execution path (Kuykendall et al. 2019). The rationale behind this is to simulate possible test case scenarios to check for SQLIA within the execution path at runtime. A scanned report is then generated on the codes with possible vulnerabilities. The main disadvantage of this process is that one cannot argue completeness, as there could be unconventional ways through which the application could be still assessed that were not thought of at the time of testing.

2.10.2.4 Hybrid Analysis Technique to Develop a Defence System against SQLIA

The combination of both the static and dynamic analyses is known as hybrid analysis (Dogbe, Millham and Singh 2013). In this combination, analysis begins with the static phase where a

grammar-based structure of the legitimate queries of the application is modelled with placeholders for user input, and then during runtime, all resultant queries with user input are modelled using the same method and grammar as the static phase but with the placeholders of the static replaced by the actual user input. Both query structures are compared side by side and any violation is reported as an SQLIA (Dogbe, Millham and Singh 2013).

2.10.2.5 Selection of the most Suitable Analysis Methods

Selection is based on the additional infrastructure and skills required to support a given technique during the deployment and what prerequisite skills are needed for deployment and maintenance of the technique. The selection was further grouped into two levels in order to select the most suitable analysis method. The levels are discussed below.

2.10.2.5.1 First Level Selection

Table 2 refers to the first level of the selection and the criteria used (additional implementation infrastructure and skills).

Table 2: First Level Selection Techniques and Criteria

	Criteria	
Techniques	**Additional implementation infrastructure**	**Skills in (development and deployment)**
Secure programming	Generally required	• Intensive development • Intensive deployment
Static analysis	Generally not required	• Fairly intensive development • Intensive deployment
Dynamic analysis	Less	• Fairly intensive development • Minimal deployment
Hybrid analysis	No additional	Minimal skills

Source: (Aliero *et al.* 2016)

As indicated in Table 2, the use of secure programming techniques generally requires additional implementation infrastructure in terms of internet connectivity to download secure class libraries or API; the purchase of additional storage devices for the downloaded class libraries; and intensive development skills for testing and hosting of class libraries on servers.

Also, it requires knowledge on how attacks are mounted and how to put up a defence against it, using defensive programming to develop secure programmes. Generally, these requirements lead to the high cost of deployment and maintenance; therefore, making this approach unsuitable for inclusion in the second level categorization.

Generally, static analysis techniques do not require additional resources such as storage devices, but they require intensive deployment skills to interpret the numerous warnings typically generated when legacy data is scanned. Therefore, static analysis may be tedious and time-consuming to deploy. It might also require continuous updates of new patterns to enhance the static analysis tool. Also, the static analysis technique requires fairly intensive development skills in terms of source code analysis. Hence, internet connectivity is imperative in this case and is unsuitable in the West African context.

On the other hand, dynamic analysis techniques require less additional implementation infrastructure and high deployment skills compared to other techniques such as static analysis. The developer could use the debugger to step trace his codes to ensure that everything is behaving as expected; however, it requires a great level of understanding to be effective. While the hybrid Analysis requires no additional implementation infrastructure and minimal skills. In view of this, the static analysis, dynamic analysis, and hybrid analysis are techniques considered for the second level of the categorization.

2.10.2.5.2 Second Level Selection

Selection involves the techniques and their efficacy in terms of false-positive, negative generation and any other weaknesses where applicable. Table 3 refers to the second level selection and criteria (focusing on the efficacy).

Table 3: Second Level Selection Techniques and Criteria

Techniques	Efficacy
Static Analysis	High False Positive
Dynamic Analysis	It has no vulnerability detection outside the programme execution path; hence, the possibility for some false negatives.
Hybrid Analysis	Little overhead for doing both analyses. It combines the best of both static and dynamic types of analyses. It has greater coverage than both static and dynamic analyses individually. It has no known false positives or negatives.

Source: (Aliero et al. 2016)

As indicated in Table 3, the static analysis technique has high false positives (Aliero et al. 2016). Therefore, tools based on this technique might report a valid data as invalid or an SQLIA, which makes this technique not efficacious for deployment.

The dynamic analysis technique during runtime checks for SQLIA within the execution path of a given test case and might not detect vulnerability outside its execution path (Aliero et al. 2016). In view of this, the dynamic analysis technique does not perform full analysis of the entire source code. Therefore, in terms of efficacy, the dynamic analysis technique is not a strong defence against SQLIA.

The hybrid analysis technique and related tools stands-out as the most suitable technique for the West African context, taking into consideration both the first and second categorisation. In reference to Table 2 and 3, the hybrid technique requires no additional infrastructure, which requires minimal skill and has no known false-positives or negatives according to the literature.

Based on the popularity and empirical review of the robustness of the AMNESIA technique, it is the most dominant hybrid analysis tool in the literature. Hence, we will discuss this tool next.

2.10.2.6 Tools using Hybrid

Amnesia uses two key techniques in analysing SQLIA; these are static and dynamic analysis techniques. The static part automatically builds a model of legitimate queries using programme analysis, while the dynamic part inspects and authenticates the queries that have been dynamically created (Halfond and Orso 2005). In the use of Amnesia, initial configuration requires the modification of scripts by the programmer. This still requires a pre-knowledge of XML and hands-on skills with script handling on web servers which might be a challenge within the West African context. This is because most institutions focus on the theoretical aspect of basic programming, and web server management and configuration (Asabere *et al*. 2017)

A less demanding hybrid technique with ease of understanding could perhaps be static and dynamic/static parse tree comparisons. In this technique, the static phase defines the query structure of SQL statements in an application into a model that depicts a tree with each fragment of SQL as tokens or nodes. Then in the dynamic phase, the same is done to dynamic queries with actual user input. The nodes of both the static and dynamic phases are matched for token position, token type, token name, etc., any mismatch is considered as an SQLIA. This technique is very easy to implement as it utilises very basic programming constructs such as 'if' statements and object-orientation concepts like collections. This technique could be easily maintained compared to AMNESIA, which requires fairly good server-side scripting knowledge and so on as explained earlier.

2.11 Summary

This chapter reviewed relevant literature and identified a gap in current tools and techniques available to prevent SQLIA in less secure web applications. The hybrid analysis technique was determined to be the most effective technique for defence in the West African context. The current resource constraints of businesses in Ghana make this tool (hybrid analysis) suitable to mitigate attack in less secure web applications. The next chapter discusses the methodology to adopt in developing an easy configurable single security component for less secure web applications taking into consideration the limited resource of the West African context.

3 CHAPTER THREE

RESEARCH METHODOLOGY

Introduction

The methodology of research is the procedure that a researcher uses to arrive at credible findings. In this chapter, the criteria for the selection of the research design, research method and its justification, the procedure used to arrive at credible findings, selection of a case system and suitable test cases as well as attack goals, strategies of inquiry, detailed description of proposed solution, limitation of the study and summary of chapter are discussed.

3.1 Criteria for Selecting Aspects of Research Design and Worldview

There are four questions that help guide a researcher in clearly understanding the problem domain within a research design. These questions are 1) What theory of knowledge is embedded in the theoretical perspective (denoted as epistemology), which informs the research (either objectivism or subjectivism)? 2) What is the theoretical perspective (in other words, what is behind the methodology in question such as critical theory, interpretive or positivism and post-positivism)? 3) What is the methodology (that is strategy or plan of action that links methods to outcomes) that governs the choice and use of methods (e.g. experimental research, survey research, ethnography, etc.)? and 4) What methods (termed as procedures or techniques) are chosen and utilised (such as experiments, surveys, etc)? (Creswell and Miller 2009).

If a problem and the approach to discover solutions are scientific, then the three identified aspects of research design may be the theoretical perspective, strategy of inquiry, and specific methods. These methods may be positivism, qualitative/quantitative/mixed methods and experimentation, respectively (Creswell and Miller 2009). Given that the problem and research objectives for this study are a scientific one, the strategy of inquiry and specific methods will be as discussed above.

3.2 The Strategies of Inquiry

There are three main types of strategies of inquiry; these are qualitative, quantitative and mixed methods (Creswell and Miller 2009). However, some scholars refer to them as research methodologies *(*Christensen and Waraczynski 1988*)*. This section will define these three types of strategies and discuss different forms of these strategies under each type.

Qualitative research (*referred here as a strategy of inquiry*) is an in-depth study of a small group of people in order to gain a deeper understanding of a phenomenon being studied. It usually involves a choice of a sample of people affected by the said phenomenon being studied, while methods such as interviews and subjective interpretation of results are commonly used (Creswell and Miller 2009).

Quantitative research (*referred here as a strategy of inquiry*) is a means of testing objective theories by examining the relationship between variables. These variables produce tangible effects that can be numerically and objectively interpreted. Researchers focus on deduction, from specific to general study of a phenomenon. This approach offers protection from individual bias, which controls and greatly minimises alternative explanations for cause and effect. This minimisation of alternative explanations is often done by controlling other variables that may affect outcomes while manipulating and observing the effects of one variable. This experiment is repeated numerous times in other to eliminate the effects of any unknown interfering variables. The findings of this kind are usually generalizable and replicable (Creswell and Miller 2009). This is the preferred method for this research primarily because it allows demonstration of cause and effect with limited biases (see 3.3 Selection of a Research Method and Justification).

Mixed methods (*referred here as a strategy of inquiry*) are a means of inquiry that combines both qualitative and quantitative methods to study problems in research. Here, the inquirer relies on the strengths of both techniques. Findings from these kinds of methods are more reliable than those of quantitative or qualitative strategy of inquiry in general. However it involves extra time since data is collected and analysed separately by both techniques (Creswell and Miller 2009)

3.3 Selection of a Research Method, Experimentation Steps and Justification

The research problem is "What could be the influence of the combined techniques of RBAC and dynamic/static parse tree comparisons as a defence model against SQLIA?" Hence, experimentation is chosen to be the procedure to use to find the solution to this problem and the strategy of inquiry used would be quantitative.

Experiments are the means by which cause and effect can be established in the scientific domain (Creswell and Miller 2009). Experimentation, as a quantitative strategy of inquiry, holds all other known variables constant while manipulating a single variable while observing the effect of the manipulated variable in a system. The independent variable (IV) is thought of

as the cause and the dependent variable (DV) as the effect (Creswell and Miller 2009). The independent variable, in this case, is the defence of the chosen system for the proof of concept of the study and the dependent variable is the test attack cases used to exploit the system, which is represented by the SQL attack variables.

There is only one variable to manipulate that is the SQL attack variable. Observation is done through tracing the program code in the application to see how the attack variable is executed and the effect is observed through logging the database damage using triggers in the database to determine the occurrence of changes before and after the execution of the attack variable.

The research steps in our thesis involve the following:

Step 1: Select a case system to test its defence model. This system represents a system used in West Africa, which initially used a basic RBAC for its security but had been recently migrated to the web where it is more vulnerable to SQLIA attacks.

Step 2: Based on this system's business rules and use cases, we determine the normal legitimate screen access requests per given user role. We then test both the control system that has no defence but a basic RBAC and the test system with the defence system, to determine if all legitimate access requests would be allowed. The goal of this step is to determine if the defence produces false positives and stops legitimate access requests.

Step 3: Based on the system's business rules and use cases, we determine illegitimate screen access requests per given user role. As there are multiple screens, the number of illegitimate screen access requests per user role can be a lot so we only tested a representative sample (particularly, those involving a finer grain of access). We then test the control system without the defence but a basic RBAC, and the test system with the defence system including fine-grained RBAC to determine if any of these illegitimate access requests would be allowed. The goal of this step is to determine if the control system's basic RBAC will stop all illegitimate access requests and whether the test system's fine-grained RBAC will stop all illegitimate access requests.

Step 4: Select variants of three of the commonest SQLIA (namely, Comment, Union and tautology attacks) to exploit the case system with attack goals of enumeration (database information discovery such as table names, column names etc.), gain login details, privilege expansion, same level attacks and where possible alter data in a database.

Step 5: Use triggers' and program tracing to observe in detail how the attacks behave and determine whether they were successful, log the damages caused, or where the defence contained the attack.

Step 6: Using selected SQLIA test cases, attack the case system without the proposed defence (control system) and use step five to record findings. The purpose of this step is to record the effects of these SQLIA test cases against the control system.

Step 7: Using selected SQLIA test cases, attack the case system with the proposed defence (test system) and use step five to record findings. The purpose of this step is to determine the efficacy of this proposed defence in the test system against these SQLIA test cases.

Step 8: Measure the potential performance cost of the proposed hybrid defence by comparing the performance cost of the control system, which does not contain the defence, with the performance cost of the test system, which contains the defence through a function in the security aspect that feeds the attacks and records the time for executions of the attacks in steps 2, 3, 6, and 7.

Step 6: Analyse the findings of these steps and their effects on both the control and test systems and draw conclusions.

3.4 Selection of a Case System Example and Justification

A case example was used to validate the effectiveness of the proposed security model as it offers a real-world context for the model to be tested. The representative case example used is a public management transportation system, specifically the system used by the GPTRU as indicated in Chapter One. This system is representative because it was recently migrated to the web and, thus, its risk of being attacked had increased as opposed to its initial business-only standalone confined environment. This presents an important opportunity for this research. The system offers the inquirer a chance to test the proposed security model of combined techniques of static/dynamic analysis of parse trees and Role-Based access control techniques.

The tools used for this research were SQL server 2005 because it is the most popular system used in most organisations and universities in Ghana ((Sadeghian, Zamani and Ibrahim 2013))..

3.5 Selection of SQL Injection Test Cases

The SQLIA test cases will cover three of the most common SQL injection Attacks, which are comment, tautology and union attacks ((Sadeghian, Zamani and Ibrahim 2013)). These types of SQLIA attacks were chosen because they are the commonest and most understood kinds of attacks. Comment, tautology and union attacks have been cited 468 times according to google scholar to be the highest citations in common SQLIA attacks. Therefore, if the proposed model is able to defend the system against these attacks, it would validate the concept. These three common types of attacks have been combined in various ways to attempt three attack goal levels of security breaches labelled as:

- Login attacks
- Same level privilege attacks
- Privilege expansion attacks

3.5.1 Login Attacks

This basically involves SQLIA attempts to gain unauthorised access to users and their roles arbitrarily.

3.5.2 Same Level Privilege Attacks

These attacks involve a hacker attempting to gain unauthorized access to data resources of other users at the same level.

For example, when a legitimate staff logs in and tries to access another staff's data resources, then these attacks are called same level privilege attacks.

3.5.3 Privilege Expansion Attacks

This attack uses a low user privilege to gain access to higher user privileges. For instance, deleting a database as a simple user.

The source of these SQLIA test cases is the Open Web Application Security Project (OWASP) (Sadeghian, Zamani and Ibrahim 2013). It provides databases to find unbiased, real-world test cases of SQLIA.

3.6 Framework of Security Model Analysis

The security model is designed as a security mechanism within the system whereby users through the internet send their requests on the web application interface as seen in Figure 3.

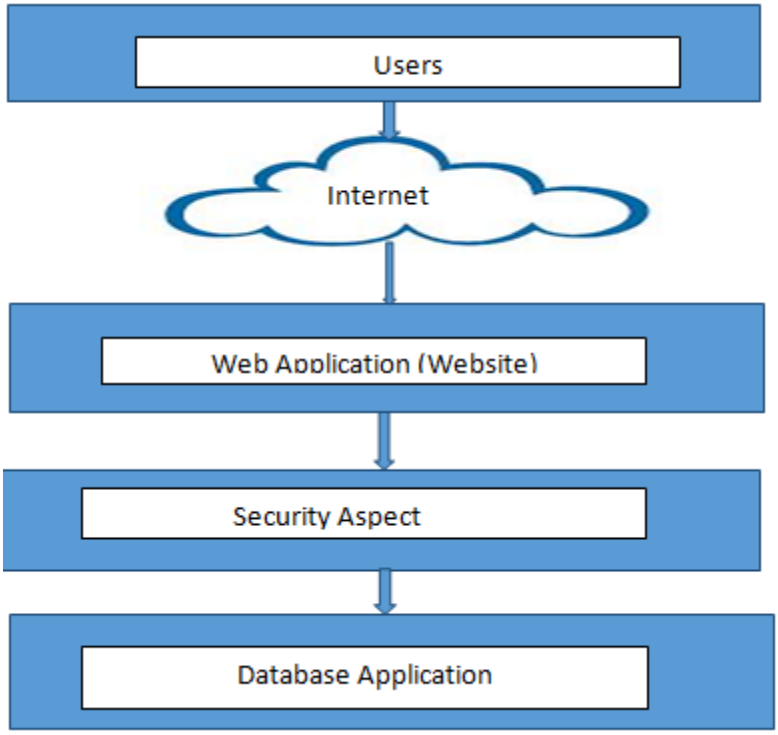


Figure 3: Framework of Security Model

As shown in Figure 3 above, these requests create traffic, which is then channelled to the security mechanism for verification and validation. The security mechanism makes a decision to pass the request unto the database to obtain the appropriate response when authentication and validation of the request are passed or sends the user to the login screen when it fails. The major constraint observed in this study was that all traffic to the database are forced through the security mechanism by the application interface and not direct manipulation at the database end as per the scope of the study. This step ensured that SQLIA did not bypass the security between the application and the database. The security mechanism implemented a two tied defence against SQLIA, which are fine-grained RBAC and static/dynamic analyses of parse trees.

3.7 Security Mechanism of the Framework

The security mechanism contains two techniques that are used as a hybrid defence: fine-grained role base access control technique and dynamic/static parse tree comparisons. Each of these will be discussed below.

3.7.1 Fine-Grained Role Base Access Control Technique

This technique was used in a security mechanism to attempt to give the least privilege access to execute a job function within the system as shown in Figure 4.

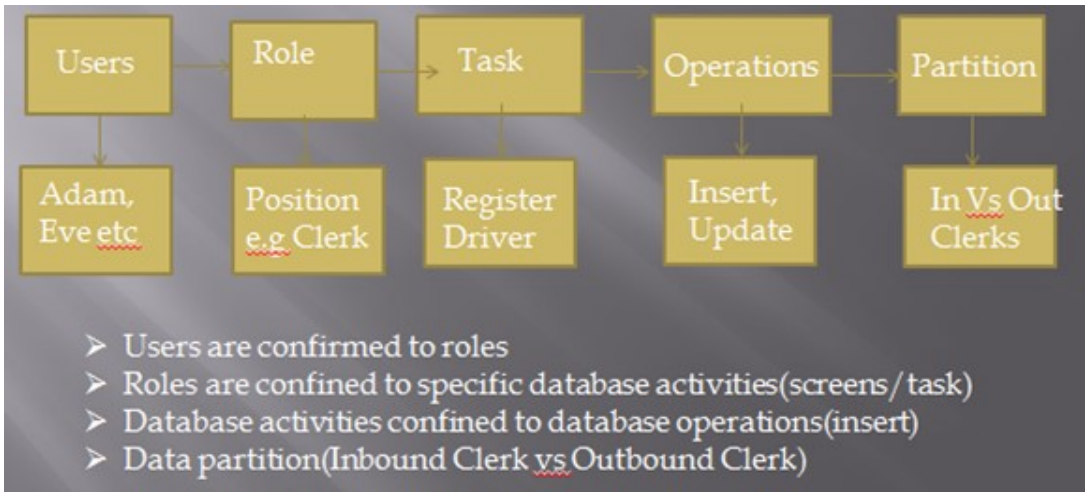


Figure 4: Privilege Access

The least access that allows the execution of job roles gained as explained above is known as privileged access. Privilege access also requires enforcing access to the database only if the user in question is using the valid Role, executing the subset of valid tasks (pages of application) for that chosen role, invoking any of the subset of valid operations under the chosen task, triggering the valid database access such as insert, update or select for the chosen operation and under the valid data partition.

3.7.2 Hybrid or Static/Dynamic Analysis of Parse Trees Technique

Usually, this technique allows a programmer to store intended SQL queries in table form in the memory before or at runtime of the application and these are then compared to runtime resultant queries with user input called dynamic queries in order to determine if a match exists. This attempt at matching was used to disregard any resultant dynamic query that does not match the static query.

Figures 5 and Figure 6 illustrate an example of a static parse tree and a dynamic parse tree, respectively.

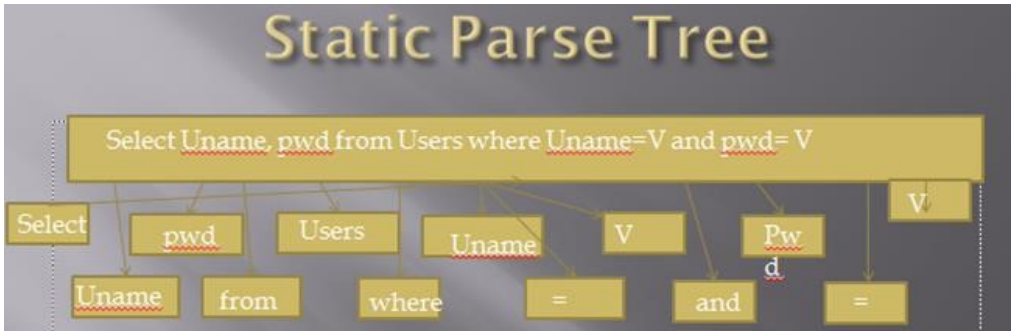


Figure 5: Static Parse Tree

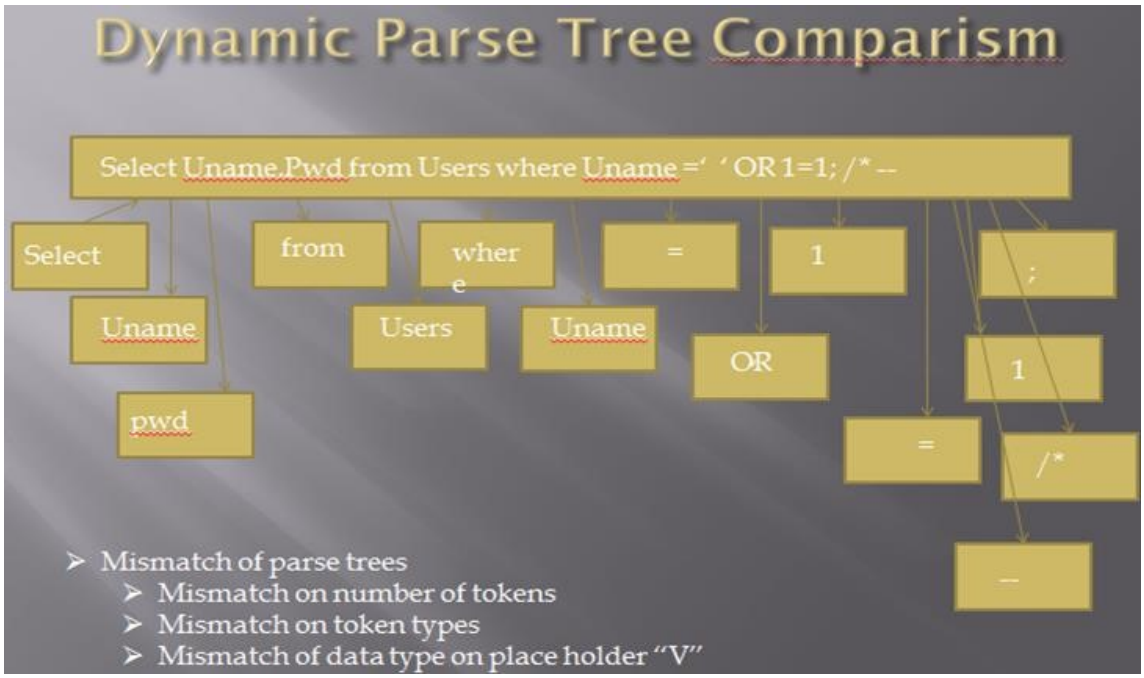


Figure 6: Dynamic Parse Tree

Basically, all static queries are programmer intended queries and dynamic queries could be programmer intended valid queries or hacker modified queries. This technique was chosen because, theoretically, it allows the application to distinguish between valid queries and invalid ones.

The primary problem of SQLIA detection, as far as the inquirer is concerned, is simply a missing mechanism to help the application to detect valid queries from invalid ones.

Therefore, the reason for the choice of hybrid analysis is because Static/Dynamic analysis (hybrid analysis) offers a simple, inexpensive and understandable way to differentiate between programmer intended queries (static queries) and hacker intended queries (dynamic queries). However, not all dynamic queries are SQLIA queries. It could be a mix of valid and invalid queries; both static and dynamic queries are constructed as trees. Trees are simply the different

parts of the selected query after being broken down into tokens which form the leaves of the tree. Each SQL query consists of separated strings/symbols called tokens, which can represent names of tables, columns, SQL commands (select, update and delete), from, where, order by and operands such as user input. The user input as shown in the dynamic parse tree diagram in Figure 6 is represented by "v" and the data type of the value of this letter of the dynamic tree is checked to match that of the static parse tree. A mismatch of this results in an attack and the security mechanism prevents the query from being executed.

In addition, legitimate queries are pulled from a simple database table to construct the static parse tree model, while a function within the security mechanism is used to construct its corresponding dynamic parse tree for comparison. This security requires very little configuration and maintenance with a low-level skill base.

3.7.3 Algorithm of Security Aspect

Algorithm code for Security Aspect

Main Program()

{

Start

//Initialise global variables

Global mismatch variables // initialise all possible mismatches in parsetree to false

blnPermitted = false

strConn as database connection

str= " " as String //query string

1. Load UserFormAccesses into a tree collection of objects with nodes as strUser, strOp, strAccess, strDataPartition //Load from database all user access roles to screens or pagenames as strUser e.g "Admin", strOp e.g "prepayment", strAccess e.g "insert Driver", strDataPartition e.g "Outbound"
2. LoadExistingQueries //Load all static queries in application into collection of trees from database under their users and operations or tasks.
3. Form access invoked
4. Call SecurityAspect(strUser as String, strOp As String, strAccess As String, strDataPartition as String, Str as querystring)
5. If blnPermitted =False then
    a. Display Access denied
    b. Close connection
    c. Exit program
6. If blnPermitted = true then
7. Continue program execution
    }

End Program

SecurityAspect (strUser as String, strOp As String, strAccess As String, strDataPartition, Str as querystring) // Function definition
Start Function

1. Start TimeCounter
2. blnAccessAllowed= VerifyRoleOpAccess(strOp As String, strAccess As String, strDataPartition As String) As Boolean //check if op passed RBAC restrictions
3. if blnAccessAllowed = false then
    a. blnPermitted =false
    b. IdentitfyRbacPositionMisMatch // where in the Rbac the mismatch happened and record into a database table
    c. Record TimeCounter
    d. exit SecurityAspect
4. End If
5. Initialise str as query attempting to access database
6. Initialise blnchkQuery As Boolean
7. blnchkQuery = CompareExistingToDefinedQueries(str, strOp) //check only for queries under that specific task represented by strOp.
8. If blnchkQuery = true Then
    a. blnPermitted=true
    b. Continue program execution
9. Else
    a. Display mismatch in global variables // display where in the parse tree the mismatch occurred and record in database
    b. Display TimeCounter
    c. Close connection to database
    d. Exit SecurityAspect

VerifyRoleOpAccess(strUser as String, strOp As String, strAccess As String, strDataPartition As String) As Boolean //Function definition
Start function

1. UserFormAccesses // load legitimate form accesses from database
2. Compare strUser, StrOp, strAccess, strDataPartition to be consistent with records of database of Rbac table
3. If consistent return true else return false

End function

CompareExistingToDefinedQueries(strqry,strOp) as boolean // Function definition
Start Function

1. initialise timecounter
2. Initialise all global mismatch variables to false
3. Check to ensure current strOp matches one of the strOp in existing static collection. // make sure current strop, representing the current operation, matches recorded strop, representing the correct operation, in static query collection as trees
4. Find which query matches the current query of strqry under strOp as task of static query collection loaded at startup of entire program.
    i. The match must be in tokenname, tokenposition,tokentype within the subset of queries under strOp

1. Loop through the queries for matches up to total number of queries under strOp -1
2. If match is found then
    a. Stop timecounter and record in database
    b. return true and exit function
3. Else continue looping until all the subset of queries under that strOp is exhausted minus One.
4. If at the end of subset of queries under strOp, no matches are found then
5. Compare current query of strqry with the last query remaining under strOp for a match in tokenname,tokenposition,tokentype
    a. If match is found then
        i. Stop timecounter and record in database
        ii. return true and exit function
    b. Else
        i. set where mismatch occurred in comparison e.g tokenname mismatch etc in global variables, stop and record timecounter value
        ii. return false
        iii. exit security aspect

End Function
End Function // end security Aspect as a function

3.7.4 Performance Measurement Metrics

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known(Visa *et al.* 2011). It allows the visualization of the performance of an algorithm. To make the most out of our security model we introduce three performance measures namely precision, recall and f-measure. Applying these performance measures to our research redefines them as follows:

Precision: Of the form accesses and queries identified as attacks by the systems security, what fraction of them were actually attacks through program tracing and triggers on the database end?

Recall: Of the form accesses and queries that are actually attacks, how many are identified as attacks by the system security through program tracing?

F-measure(Fβ): This is the mathematical value that determines the comparative performance between two systems with different values of precision and recall. For example consider two systems; 1 and 2. The Fβ helps determine which of the two systems performs better. Consider the following values as precision and recall:

System 1
- Precision = 70%
- Recall= 60%

System 2
- Precision = 80%
- Recall = 50%

β value is greater if the value of precision is more important in our test for performance which is the case in this research and the β is less if recall is more important in our measurement. For this

research β was assumed to be 0.95 threshold. In other words, the research did not want to classify an attack as being a legitimate access and the threshold for that was very high, with 1.0 meaning Zero tolerance for errors in precision.

Precision, Recall and F-measure (Fβ) is calculated as follows:

Let Precision be P,

Recall be R,

F-measure be Fβ,

True Positives be TP,

False Positives be FP,

Then;

$P=TP/(TP+FP)$

$R=TP/TP+FN$

$F\beta= 1/(\beta x\ 1/P +(1-\beta)x\ 1/R)$

Then

Confusion Matrix becomes:

Table 15 (Confusion Matrix Model)

	P'(predicted)	n'(predicted)
P(Actual)	True Positive	False Negative
n(Actual)	False Positive	True Negative

3.8 Limitations of the Framework

The major constraint observed in this study particularly with the security model was that all traffic to the database was forced through the security mechanism and no direct manipulation of the database was covered as that was outside the scope of the study. Also there was no extentive testing available other than common testcases from literature used.

3.9 Summary

This chapter described the methodology adopted for the study, the bases for the choice of methodology and framework for the security model analysis. The next chapter will discuss the analysis and implementation of the proposed security model.

4 CHAPTER FOUR

DESIGN, ANALYSIS, AND IMPLEMENTATION OF SYSTEM

4.1 Introduction

This chapter discusses the testing of the proposed model. It determines to what extent, if any, the test system is an improvement over the control system using triggers and application code monitoring or program tracing as metrics. The system chosen as the case study, hereafter referred to as the control system, and the proposed model, hereafter referred to as the test system are tested by using different security threats that would try to compromise the confidentiality and integrity of existing data.

The chapter is organised as follows:

1. A narrative of the business rules of the control system (describes the business rules and scenarios of the control system).
2. Job roles and functions (the list of job roles within the system chosen the case study).
3. Model of the control system with SQLIA (provides description of the system and its environment).
4. Model of the Test System with SQLIA.
5. Based on the control system's business rules and used cases, we determine the standard legitimate screen access requests per given user role. We then test both the control system, without a defence system but basic RBAC, and the test system, with a defence system, to determine if legitimate access requests would be allowed.
6. Based on the case system's job roles, we determine illegitimate screen access requests per given user role. We then test both the control system, without the defence but with basic RBAC, and the test system, with the defence system including fine-grained RBAC, to determine if any illegitimate access requests would be allowed.
7. The control system, which utilises basic RBAC, is attacked using selected SQLIA test cases and the approximate execution times of these attacks are recorded.
8. The test system, which utilises fined grained RBAC and dynamic/static parse tree comparison defence, is attacked using selected SQLIA test cases and the approximate execution times of these attacks are recorded.
9. The performance differences in Sections 7 and 8 are compared to determine whether any performance degradation occurred due to the use of the security defence.

10. Analyse the findings of the tests, in terms of successful access and their effects on both the control and test systems, and conclusions are drawn.
11. Finally calculate the precision, recall and f-measure of control system with basic RBAC, test system with Fine grained RBAC and test system with fined grained RBAC and dynamic/static parse tree comparison defence
12. A summary of the chapter is given.

4.1.1 Narrative of Business Rules of Control System (Business Rules and Scenarios of Operation of the Case System)

The case system has different processes that are performed by different groups of users. These different groups of users are categorised as roles within the system. The different roles within the system are ascribed different job functions as per the business rules of the organisation. The organisation has simple rules for its operations.

The registration of drivers that operate within the city is carried out by an employee, hereafter called inbound clerk, while the employee who registers those outside the city is called outbound clerk.

The transactions are audited by employees who are hereafter called trustees. Trustee 1 updates fares for both inbound and outbound drivers, whereas Trustee 2 just views the reports of all the transactions. Trustee 2 audits the actions of Trustee 1 through reports. In order to know how the rules work, consider the following job scenarios

Scenario 1: An intra-city driver purchases a station ticket

System roles and response: The inbound clerk(IC) fills out a form on a computer with the driver's bio data and contacts, and then accesses the ticket screen to choose ticket price and route via radio buttons. IC confirms ticket purchase and prints out a receipt that is given to the intra-city driver.

Scenario 2: An inter-city driver purchases a station ticket.

System roles and responses: The outbound clerk (OC) performs the same actions as the IC clerk except he/she is serving an inter-city driver.

Scenario 3: Registration of intra-city cars.

System roles and responses: The inbound clerk (IC) fills out a form on a computer with intra-city drivers' or car owner's contact details. Also, the intra-city driver or car owner can fill out the forms online through his account that was created when he/she was first registered by IC.

Scenario 4: Registration of inter-city cars.

System roles and responses: The outbound clerk (OC) fills out a form on a computer with inter-city driver's or car owner's and contacts details. Also, the inter-city driver or car owner can fill out the forms online through his account that was created when he/she was first registered by OC.

Scenario 5: Update route fares.

System roles and responses: Trustee 1 fills out a form on a computer using the fares screen and chooses the appropriate routes via a combo box, specifies new amounts and confirms changes via a submit button.

Scenario 6: Audit route fares

System roles and responses: Trustee 2 views fares screen and can choose different dates for reports on fares via a date user control element on the screen and can view the report in a grid.

4.1.2 Job Roles and Functions (List of Job Roles within the System Chosen the a Case Study)

Table 13 indicates the different groups of roles and their job functions according to the business rules of the organisation under study.

Table 13 Roles and Job Functions

Roles	Full name	Job Function
Ic	Inbound Clerk	Registers drivers, car owners and cars of inbound drivers, issues ticket to drivers, and updates drivers' prepayment status
Oc	Outbound Clerk	Registers drivers, car owners and cars for outbound drivers, issues ticket to drivers, and updates drivers' prepayment status

T1	Trustee One	Updates the fares for both inbound and outbound drivers and reads reports on all transactions
T2	Trustee Two	Reads only reports of transactions
Driv	Driver	Buys ticket, prepays for tickets, and registers biodata and contact details

Source: Field Data, 2015

Each job function is linked to one or more forms in the system. Consequently, these different forms are linked to various roles. Table 14 illustrates the various job roles and the form access requests allowed as per the business rules of the selected case example system.

Table 14 Roles and Form Access

Job Role	**Form Access**
Ic	Registration, ticket and prepayment
Oc	Registration, ticket and prepayment
T1	Updates fares and reports
T2	Reports
Driv	Ticket

Source: Field Data, 2015

4.1.3 : Model of Control System with SQLIA (Provides Description of the System and its Environment)

4.2 Model of Control System

The control system is a web client-server system consisting of a web application and an SQL database as the backend database. The control system has only a basic Role-Based access control system as its method of security and accesses records in the database for different users of the system. Relying on the basic Role-Based access control system is common in Ghana (Lartey 2019); however, it is vulnerable to web-based attacks such as SQL injection attacks.

The model of this system is depicted in Figure 7.

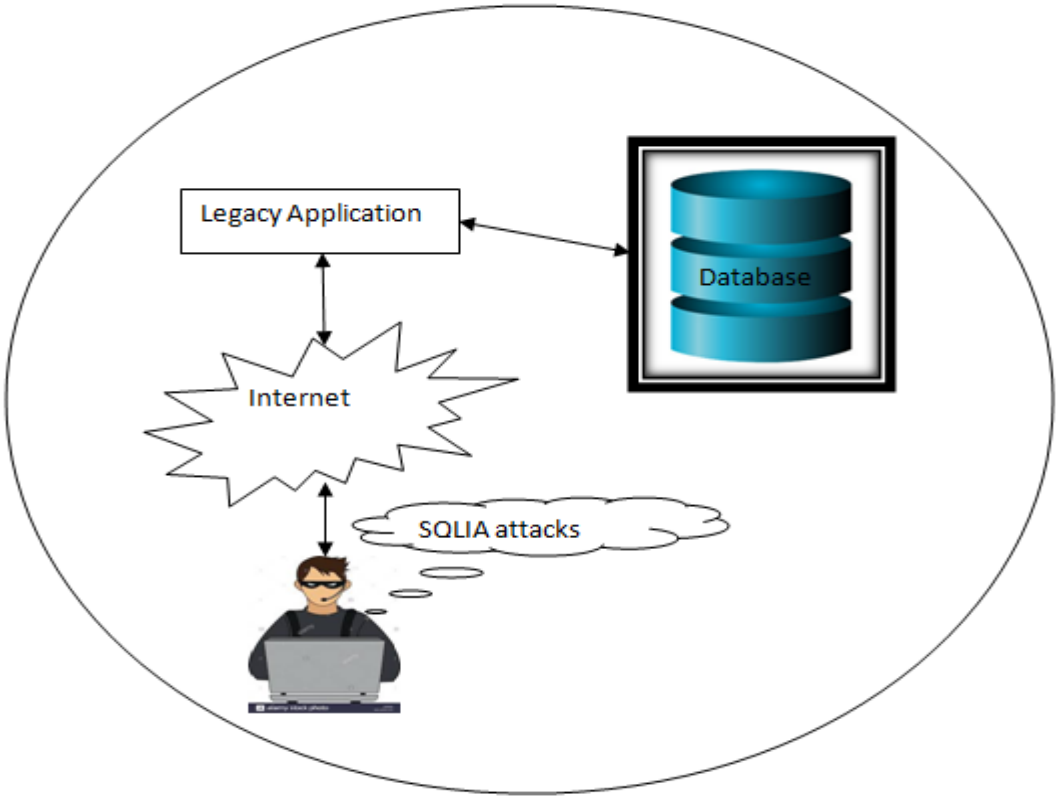
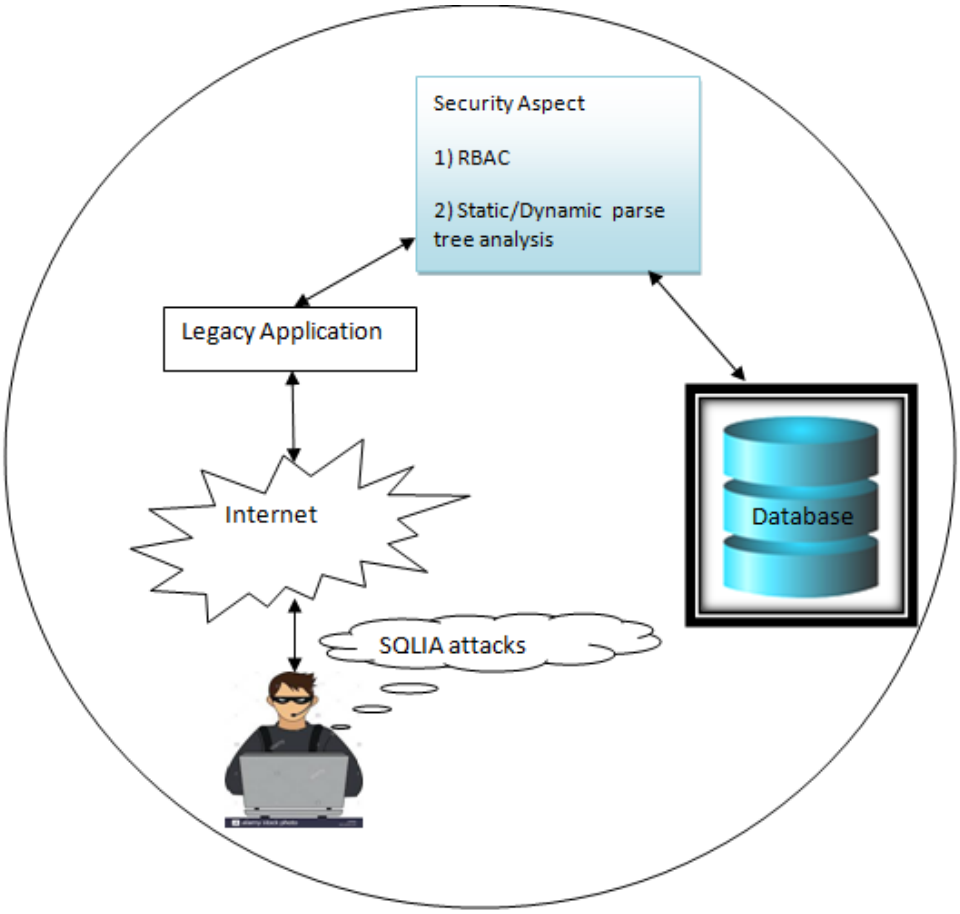


Figure 7 Model of Control System with SQLIA

4.2.1 Model of the Test System with SQLIA

Similar to the control system, the test system consists of a web application and an SQL database as the backend database. However, the test system has a security mechanism that is designed to implement a hybrid of RBAC and Dynamic/static parse tree comparisons. The user starts the process of interaction by making a request through a typical web interface on the internet; the request is received by the legacy system which in turn passes it on to the security mechanism for analysis which allows only valid requests to interact with the database. In this system, the security mechanism is centralised and, therefore, filters all traffic going and coming into the database.

The model of the test system is depicted in Figure 8



4.2.2 Normal Legitimate Screen Access requests per given User Roles

In this section, based on this case system's job rules and functions, we determine the normal legitimate screen access requests granted per given user roles. We then test both the control system with a basic RBAC, and the test system with the proposed defence system to determine whether all legitimate access requests would be allowed.

Based on the business rules and users, **Table 4** depicts the normal legitimate screen access requests per given user.

Table 4 (Normal Legitimate Screen Access Requests)

Job Role	Form Access	Allowed Form Access by Data Partition
Ic	Registration, ticket and prepayment	Inbound driver's registration, inbound car owners' registration, inbound ticket sales, inbound prepayment for drivers' status update, inbound prepayment for car owners' update status and inbound car registrations
Oc	Registration, ticket and prepayment	Outbound driver's registration, outbound car owner's registration, outbound ticket sales, outbound prepayment for drivers' status update, outbound prepayment for car owners' status update and outbound car registrations
T1	Update fare and reports	Fare update for both inbound and outbound routes, create and view reports
T2	Reports	View reports for both inbound and outbound routes
Driv	Ticket	Buy ticket, prepay for ticket, register biodata and contact details

Table 4: Normal Determined Legitimate Screen Access requests

Testing of the control and test systems with a basic RBAC and fine-grained RBAC using selected legitimate form access requests generated the same results, which are recorded in Table 5 below.

Table 5 Normal Legitimate Selected Screen Access requests of Control and Test System with Basic RBAC and Fine-grained RBAC, respectively

Job Role	**Form Access**	**Allowed Form Access by Data Partition**	**Program Tracing Status**
Ic	Registration, ticket and prepayment	Inbound driver's registration	Successful
		Inbound car owner's registration	Successful
		Inbound ticket sales	Successful
		Inbound prepayment for driver's status update	Successful
		Inbound prepayment for car owner's update status	Successful
		Inbound car registration	Successful
Oc	Registration, ticket, prepayment	Outbound driver's registration	Successful
		Outbound car owner's registration	Successful
		Outbound ticket sales	Successful
		Outbound prepayment for driver's status update	Successful
		Outbound prepayment for car owner's status update	Successful
		Outbound car registration	Successful
T1	Update Fare and reports	Fare update for inbound routes	Successful
		Fare update for outbound routes	Successful
		Create reports	Successful

		View reports	Successful
T2	Reports	View reports on inbound routes	Successful
		View reports on outbound routes	Successful
Driv	Ticket	Buy tickets	Successful
		Prepay for tickets	Successful
		Register biodata and contact details	Successful

4.2.3 Testing of Illegitimate Form Access Requests for both Control and Test Systems

In this section, we determine from the business rules and used cases the illegitimate form access requests granted per user/role for both the control and test systems, and then test the control and test systems separately to record their differences, if any. The illegitimate form access requests of the control and test systems are depicted in Table 6 below:

Table 6 Selected Illegitimate Screen Access Requests of the System according to Business Roles for the Control and Test Systems (These are Representative Tests and not an Exhaustive Test List)

Job Role	Form Access	Disallowed Form Access by Data partition	Attack Goal
Ic	Registration, ticket and prepayment	Outbound driver's registration, outbound car owner's registration, outbound ticket sales, outbound prepayment for drivers' status update, outbound prepayment for car owner's status update and outbound car registrations	Same level privilege attacks
Oc	Registration, ticket and prepayment	Inbound driver's registration, inbound car owner's registration, inbound ticket sales, inbound prepayment for driver's status update, inbound prepayment for car owner's update status	Same level privilege attacks

		and inbound car registrations	
T1	Update Fare, Reports	Register inbound cars and register drivers	Privilege expansion attacks
T2	Reports	Fare update for both inbound and outbound routes, and create Reports	Privilege expansion attacks
Driv	Ticket	Update fare	Privilege expansion attacks

Testing of selected illegitimate form access requests for the control system with Basic RBAC is depicted in Table 7.

Table 7 Testing of Selected Illegitimate Form Access Requests

Job Role	**Form Access**	**Disallowed Form Access by Data Partition**	**Program Tracing Status**
Ic	Registration, ticket and prepayment	Outbound driver's registration	Successful
		Outbound car owners	Successful
		Outbound ticket sales	Successful
		Outbound prepayment for driver's status update	Successful
		Outbound prepayment for car owner's status update	Successful
		Outbound car registration	Successful
Oc	Registration, ticket and prepayment	Inbound driver's registration	Successful
		Inbound car owner's registration	Successful
		Inbound ticket sales	Successful
		Inbound prepayment for driver's status update	Successful
		Inbound prepayment for car owner's update status	Successful

		Inbound car registration	Successful
T1	Update fares and reports	Register inbound cars	Fail
		Register drivers	Fail
T2	Reports	Update inbound fares	Fail
		Update outbound routes	Fail
Driv	Ticket	Update fares	Fail

The results from the programme tracing recorded in Table 7 show that all the operations of outbound clerks could be done by inbound clerks and the reverse was found to be true during testing. In addition, Trustee 1 and Trustee 2's roles and illegitimate access requests were correctly detected. Therefore, the control system lacks data partition defences.

After the testing of illegitimate form access requests for the control system, we ran the same test on the test system and recorded the results through programme tracing, which involves inserting breakpoints. Next, the programme execution in code, and physical observations of the test system was conducted with the form access requests as shown in Table 8.

Table 8 Testing of Selected Illegitimate Form Access requests in the Test System with Fine RBAC

Job Role	**Form Access**	**Disallowed Form Access by Data partition**	**Program Tracing status**
Ic	Registration, ticket and prepayment	Outbound driver's registration	Fail
		Outbound car owners	Fail
		Outbound tickets sales	Fail
		Outbound prepayment for driver's status update	Fail
		Outbound prepayment for car owner's status update	Fail
		Outbound car registrations	Fail
Oc	Registration, ticket and prepayment	Inbound driver's registration	Fail
		Inbound car owner's registration	Fail
		Inbound tickets sales	Fail
		Inbound prepayment for driver's status update	Fail
		Inbound prepayment for car owner's update status	Fail
		Inbound car registration	Fail
T1	Update fares and reports	Register inbound cars	Fail
		Register drivers	Fail
T2	Reports	Update inbound fares	Fail
		Update outbound routes	Fail

Driv	Ticket	Update fare	Fail

Table 8

Table 8 shows that the fine-grain RBAC was successful in stopping the data partition tests of reversal of inbound and outbound clerks' roles unlike that of the control system, where they failed to be detected. In addition, the table shows that the test system was able to stop other selected illegitimate form access requests of other users like Trustee 1, Trustee 2 and drivers trying to perform tasks outside their job functions and scope.

4.2.4 Testing Control System and Measuring Performance with SQLIA Suite

Table 9 depicts the SQLIA test suite, with descriptive comments, for our testing both control and test systems. The value of "0x3c62723e" is the hex for the return key, and it is used to enable nice formatting during programme tracing and viewing of results on the direct pages of the exemplar system during the fingerprinting or enumeration phase of the attacks.

Table 9 SQLIA Test Case Suite

No.	SQL Statements	Comments	Attack Goals
1.	Select name, description, price from tickets where category=1 union select all group_concat (table_name) from information_schema.tables where table_schema=database()- -	Finding out all table names in the database	Enumeration
2.	Select name, description, price from tickets where category=1 UNION select all group_concat(column_name,0x3c62723e) from information_schema.columns where table_name=0x61646d696e6973747261746f7273 - -	Finding out all columns in the specific table such as the administrators table in hex and 0x3c62723e is the hex for the return key; this allows for nice formatting on the page for reading the results of the attack.	Enumeration
3.	Select name, description, price from tickets where category=1 UNION select all group_concat(column_name,0x3c62723e) from	Finding out all columns in the specific	Enumeration

	information_schema.columns where table_name=0x4c6f67696e - -	table such as the Login table in hex and 0x3c62723e is the hex for the return key' this allows for nice formatting on the page for the results	
4.	Select name, description, price from tickets where category=1 UNION select all group_concat(column_name,0x3c62723e) from information_schema.columns where table_name=0x636172 - -	Finding out all columns in the car table in hex	Enumeration
5.	Select name, description, price from tickets where category=1 UNION select all group_concat(column_name,0x3c62723e) from information_schema.columns where table_name=0x647269766572 - -	Finding out all columns in the driver table in hex	Enumeration
6.	Select name, description, price from tickets where category=1 UNION select all group_concat(column_name,0x3c62723e) from information_schema.columns where table_name=0x6f776e6572 - -	Finding out all columns in the owner table in hex	Enumeration
7.	Select name, description, price from tickets where category=1 UNION select all group_concat(column_name,0x3c62723e) from information_schema.columns where table_name=0x7261746573 - -	Finding out all columns in the rates table in hex	Enumeration
8.	Select name, description, price from tickets where category=1 UNION select all group_concat(column_name,0x3c62723e) from information_schema.columns where table_name=0x7469636b6574 - -	Finding out all columns in the ticket table in hex	Enumeration
9.	Select name, description, price from tickets where category=1 UNION select all group_concat(column_name,0x3c62723e) from information_schema.columns where table_name=0x5072657061796d656e74 - -	Finding out all columns in the prepayment table in hex	Enumeration

10.	Select name, description, price from tickets where category=1 UNION select all group_concat(column_name,0x3c62723e) from information_schema.columns where table_name=0x726567697374726174696f6e - -	Finding out all columns in the registration table in hex	Enumeration
11.	Select Role From Login where Username = 'x'; insert into Login values ('hacker','hackpass','ic',null); insert into Login values ('david','hackpass','oc',null); insert into Login values ('t1','hackpass','t1',null); insert into Login values ('t2','hackpass','t2',null); -- and Password = ‘’	Inserting values into the login table	Login attacks
12.	Select Role From Login where Username = 'hacker'; Insert into Car(OwnerId,DriverID,Make,Model,PlateNumber) values (‘3’, ‘3’, ‘BMW’, ‘2012’, ‘gh4667’) -- and Password = ‘’	Inserting values into car table	Same level privilege attack
13.	Select Role From Login where Username = 'hacker';Insert into Driver (DriverName,BoxNumber,City,Region,Phone,LicenseType) values (‘king’, ‘4847’, ‘durban’, ‘KZN’,‘0737664635’, ‘b’) -- and Password = ‘’	Inserting values into the driver table	Same level privilege attack
14.	Select Role From Login where Username = 'hacker';Insert into Owner (OwnerName, BoxNumber, City, Region, Phone) values (‘king’, ‘8487’, ‘capetown’,’eastern’,’b’) -- and Password = ‘’	Inserting values into the owner table	Same level privilege attack
15.	Select Role From Login where Username = 'hacker';Insert into Prepayment (DriverID,Payment) values (‘5’,5000099) -- and Password = ‘’	Making illegal payments	Same level privilege attack
16.	Select Role From Login where Username = 'hacker';Insert into Rate (RateID, InTownRate,OutTownRate) values (45444,448837) -- and Password = ‘’	Illegal changing of the rates for routes	Privilege expansion attack
17.	Select Role From Login where Username = 'hacker';insert into Registration(RegID,CarID,CarNumber,Trandate,Route) values (1000,1000,1000,'fh553',2013/01/5,'in') -- and Password = ‘’	Adding a car record to the registration records	Same level privilege attack
18.	Select Role From Login where Username = 'hacker';Insert into Ticket(DriverID,CarID,TicketDate,Amount,TransportType) values (1000,1000,2012/05/07,57786,'in') -- and Password = ‘’	Adding a free ticket	Same level privilege attack

19.	Select Role From Login where Username = 'hacker'; update Login set Username = 'john' where Username = ‘hacker’-- and Password = ‘’	Illegal change of login details	Login attacks
20.	Select Role From Login where Username = ‘john’; update Car set Make = ‘2006’ where Make = ‘2009’ -- and Password = ‘’	Modifying car records on file	Same level privilege attack
21.	Select Role From Login where Username = ‘john’; update Driver set DriverName = ‘John’ where DriverName= “Tom” -- and Password = ‘’	Modifying driver records	Same level privilege attack
22.	Select Role From Login where Username = ‘john’; update Owner set OwnerName = ‘hacker’ where OwnerName= “Adam”-- and Password = ‘’	Modifying owner record	Same level privilege attack
23.	Select Role From Login where Username = ‘john’; update Prepayment set Payment = 10000000000 where Payment= 10 -- and Password = ‘’	Making fake prepayment	Same level privilege attack
24.	Select Role From Login where Username = ‘john’; update Rate set OutTownRate = 10 where OutTownRate=100 ” -- and Password = ‘’	Modifying intra city rates for routes	Privilege expansion attack
25.	Select Role From Login where Username = ‘john’; Update Registration set PlateNumber= ‘Ng500’ where PlateNumber= ‘Ng100’ -- and Password = ‘’	Updating the registration table	Same level privilege attack
26.	Select Role From Login where Username = ‘john’; Delete from Rate -- and Password = ‘’	Deleting records in the rates table	Privilege expansion attack
27.	Select Role From Login where Username = ‘john’; Delete from Car -- and Password = ‘’	Deleting all records in the car table	Same level privilege attack
28.	Select Role From Login where Username = ‘john’; Delete from Owner -- and Password = ‘’	Deleting records in the owner table	Same level privilege attack
29.	Select Role From Login where Username = ‘john’; Drop Table Rate -- and Password = ‘’	Deleting the rate table entirely	Privilege expansion attack
30.	Select Role From Login where Username = ‘john’; Drop Table Prepayment -- and Password = ‘’	Deleting the prepayment table entirely	Privilege expansion attack

31.	Select Role From Login where Username = ‘hacker’ OR 1= 1; /*’ and Password = ‘y*/--‘	Login as inbound clerk	Login attack
32.	Select Role From Login where Username = ‘david’ OR 1=1; /*’ and Password = ‘*/--‘	Login as outbound clerk	Login attack
33.	Select Role From Login where Username = ‘t1’ OR 1=1;/*’ and Password = ‘*/--‘	Login as Trustee 1	Login attack
34.	Select Role From Login where Username = ‘t2’ OR 1=1;/*’ and Password = ‘*/--‘	Login in as Trustee 2	Login attack

Source: Field Data, 2015

Table 10 indicates the recorded results for the tests conducted in the control system with a set of selected SQLIA statements. In monitoring the effect of SQLIA, database triggers and programme tracing were used. However, when measuring performance, a function within the security mechanism was used to feed in the SQLIA and record the performance obtained. Even though in all the cases the Basic RBAC stopped the first parts of the queries, the second part still executed and did damage to the database from the programme tracing and triggers created to monitor operations. The query numbers (Query No) corresponds to the SQLIA number in Table 9. The results are shown in Table 10 below.

Table 10 The Results of SQLIA on the Control System with Basic RBAC

Query No	Status	Columns Affected	Rows Affected	Comments
1	Successful	-	-	9 tables were reported on the page of the control system through programme tracing (PT)
2	Successful	4	2	2 records with 4 columns were affected by triggers (TRIG) and programme tracing (PT)
3	Successful	4	5	5 records of users with 4 columns were affected by TRIG and PT
4	Successful	5	3	3 records of cars with 5 columns were affected by TRIG and PT
5	Successful	6	3	3 records of drivers with 6 columns were affected by TRIG and PT
6	Successful	5	3	3 records of owners with 5 columns were affected by TRIG and PT
7	Successful	3	1	1 record of rates with 3 columns were affected by TRIG and PT
8	Successful	5	6	6 records of tickets with 5 columns were affected by TRIG and PT
9	Successful	2	6	6 records of prepayment with 2 columns were

				affected by TRIG and PT
10	Successful	5	3	3 records of registration with 5 columns were affected by TRIG and PT
11	Successful	4	4	4 records of users as inbound, outbound, trustee 1 and 2 inserted with 4 columns were affected by TRIG and PT
12	Successful	5	1	1 record of cars with 5 columns were affected by TRIG and PT
13	Successful	6	1	1 record of drivers with 6 columns were affected by TRIG and PT
14	Successful	5	1	1 record of owners with 5 columns were affected by TRIG and PT
15	Successful	2	1	1 record of prepayment made with 2 columns were affected by TRIG and PT
16	Successful	3	1	1 record of rate changed with 3 columns were affected by TRIG and PT
17	Successful	5	1	1 record of registration was added with 5 columns, which were affected by TRIG and PT
18	Successful	5	1	1 record of a free ticket was added

				with 5 columns, which were affected by TRIG and PT
19	Successful	4	1	1 record of fake user was added to login with 4 columns, which were affected by TRIG and PT
20	Successful	1	1	1 record of a car with a make of 2006 was changed to 2009, affecting 1 column by TRIG and PT
21	Successful	1	1	1 record of changing a driver's name was made with 1 column, which was affected by TRIG and PT
22	Successful	1	1	1 record of changing an owner's name to hacker was made with 1 column affected by TRIG and PT
23	Successful	1	1	1 record of changing a prepayment to 10 billion from 10 Ghana cedis was made with 1 column affected by TRIG and PT
24	Successful	1	1	1 record of changing an outbound rate from 10 to 100 Ghana cedis with 1 column was affected by TRIG and PT

25	Successful	1	1	1 record of changing a car's number plate was made with 1 column affected by TRIG and PT
26	Successful	4	4	4 records of Login tables were deleted with 4 columns affected by TRIG and PT
27	Successful	5	3	3 records of car tables were deleted with 5 columns affected by TRIG and PT
28	Successful	5	3	3 records of owner tables were deleted with 5 columns affected by TRIG and PT
29	Successful	3	2	The entire rate table was deleted by TRIG and PT
30	Successful	2	4	The entire prepayment table was deleted from TRIG and PT
31	Successful	1	1	Login as Inbound Clerk was made affecting 1 row and 1 column with TRIG and PT
32	Successful	1	1	Login in as Outbound Clerk was made affecting 1 row and 1 column with TRIG and PT
33	Successful	1	1	Login in as Trustee 1 was made affecting 1 row and 1 column by TRIG and PT

34	Successful	1	1	Login in as Trustee 2 was made affecting 1 row and 1 column by TRIG and PT

Testing of Test System with SQLIA Suite

After testing the control system with SQLIA, the test system was tested with the proposed defence. Table 11 indicates the recorded results for the selected SQLIA tests carried out in the test system of Fine-grained RBAC and static/dynamic query. Although in all cases the Fine RBAC stopped the first parts of the queries, the second part were still executed; however, they detected in the static/dynamic query comparison hybrid tier of the defence system and prevented from programme tracing of monitoring operations. The query numbers (Query No) corresponds to the SQLIA number in Table 9 of the test suite. The test suite was exactly the same as what was used in the control system.

The results are shown in Table 11.

Table 11 Results of SQLIA on Test System with Hybrid Defence

Query No	Status form	Columns Affected	Rows Affected	Comments
1	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the UNION keyword token from TRIG and PT
2	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the UNION keyword token from TRIG and PT
3	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the UNION keyword token from TRIG and PT
4	Fail	-	-	SQLIA was caught by

				Static /Dynamic parse tree comparison; it mismatched on the UNION keyword token from TRIG and PT
5	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the UNION keyword token from TRIG and PT
6	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the UNION keyword token from TRIG and PT
7	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the UNION keyword

				token from TRIG and PT
8	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the UNION keyword token from TRIG and PT
9	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the UNION keyword token from TRIG and PT
10	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the UNION keyword token from TRIG and PT
11	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it

				mismatched on the INSERT keyword token from TRIG and PT
12	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the INSERT keyword token from TRIG and PT
13	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the INSERT keyword token from TRIG and PT
14	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the INSERT keyword token from TRIG and PT
15	Fail	-	-	SQLIA was caught by Static

				/Dynamic parse tree comparison; it mismatched on the INSERT keyword token from TRIG and PT
16	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the INSERT keyword token from TRIG and PT
17	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the INSERT keyword token from TRIG and PT
18	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the INSERT keyword

				token from TRIG and PT
19	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the UPDATE keyword token from TRIG and PT
20	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the UPDATE keyword token from TRIG and PT
21	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the UPDATE keyword token from TRIG and PT
22	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it

				mismatched on the UPDATE keyword token from TRIG and PT
23	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the UPDATE keyword token from TRIG and PT
24	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the UPDATE keyword token from TRIG and PT
25	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the UPDATE keyword token from TRIG and PT
26	Fail	-	-	SQLIA was caught by Static

				/Dynamic parse tree comparison; it mismatched on the DELETE keyword token from TRIG and PT
27	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the DELETE keyword token from TRIG and PT
28	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the DELETE keyword token from TRIG and PT
29	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the DROP keyword token from TRIG and PT

30	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the DROP keyword token from TRIG and PT
31	Fail	-	1	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the OR keyword token from TRIG and PT
32	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the OR keyword token from TRIG and PT
33	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the OR keyword token from TRIG and PT

34	Fail	-	-	SQLIA was caught by Static /Dynamic parse tree comparison; it mismatched on the OR keyword token from TRIG and PT

4.2.5 Performance Time Degradation Assessment

Following the testing of the test system of fine-grained RBAC and hybrid defence of static/dynamic parse tree comparison, we determined the difference in the execution times for the control system of basic RBAC and test system of hybrid defence of static/dynamic parse tree comparison with the SQLIA suite in Table 9.

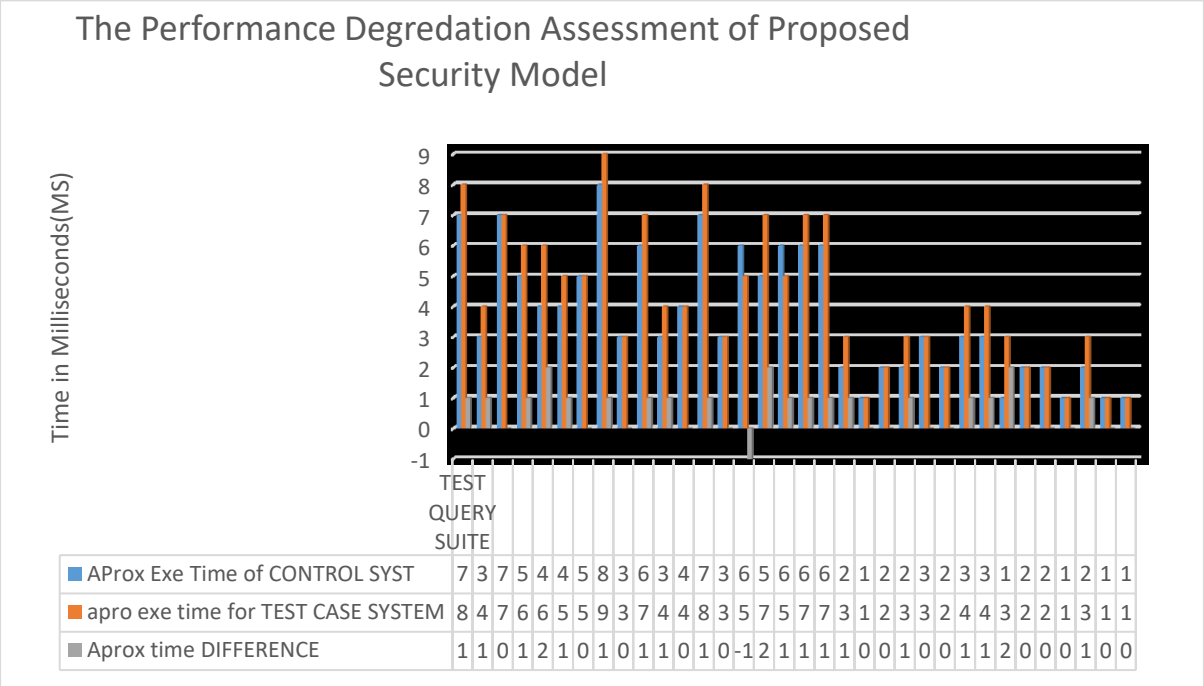
In measuring performance, a function within the security mechanism was used to feed in the SQLIA and to record the performance results obtained.

Table 12 shows the results obtained for performance.

Test Query No	Approximate Execution Time of Control System (Ms)	Approximate Execution Time of Test System (Ms)	Approximate Execution Time of Difference (Ms)
1	7	8	1
2	3	4	1
3	7	7	0
4	5	6	1
5	4	6	2
6	4	5	1
7	5	5	0
8	8	9	1
9	3	3	0
10	6	7	1
11	3	4	1
12	4	4	0
13	7	8	1
14	3	3	0
15	6	5	-1
16	5	7	2
17	6	5	1
18	6	7	1
19	6	7	1
20	2	3	1
21	1	1	0
22	2	2	0
23	2	3	1
24	3	3	0
25	2	2	0

26	3	4	1
27	3	4	1
28	1	3	2
29	2	2	0
30	2	2	0
31	1	1	0
32	2	3	1
33	1	1	0
34	1	1	0
Total Time (SUM)	126	145	21
Total Average (SUM/NO Queries)	3.71	4.26	0.62

As shown in Table 12, there was no significant difference in execution times for the testing of the SQLIA suite in the control and test systems since the average time after running the 34 queries was under a second. Graph 1.0 illustrates this in a graphical representation. It depicts a graphical representation of the results and assessments.



Graph 1.0 The Performance Degradation Assessment of the Proposed Security Model

4.2.6 Analysis of Findings for both the Control and Test Case Systems

As depicted in Table 9, the SQLIA suite went through the control system; the program tracing revealed that the basic RBAC stopped the first part of the query that had a role as well as the password mismatch, however, it still executed the second part of the query which contained the attack as a piggyback query with variants of comment, tautology, union and straight forward queries attack (basically any query could potentially run without problems). The triggers revealed that, in all instances, there were rows and columns that were affected by the attack without any problems. The initial query tests were responsible to enumerate the system and discovery table, columns, etc to exploit. Tables that were outrightly deleted by the hacks during the control test were recreated before the test system was tested to ensure the consistency of results. The overall test results for the test system show that the defence was able to stop all attacks of the SQLIA suite while the control system could not.

4.2.7 Analysis of Findings for better performance with precision, recall and f-measure

Confusion Matrix for Basic RBAC in control system

Normal and test cases data recorded, 21 True Positives , 46 False Positives, 5 True Negatives, 0 False Negatives. Consider Table 16 below:

Table 16 Confusion matrix for Basic RBAC model of Control system

	P'(predicted)	n'(predicted)
P(Actual)	True Positive (TP= 21)	False Negative (FN=0)
n(Actual)	False Positive (FP=46)	True Negative (TN=5)

P=TP/(TP+FP) ⇔ 21/(21+46)= 0.3134 ⇔ 31 % precision

R=TP/TP+FN ⇔ 21/(21+0) = 1 ⇔ 100 % recall

If β=0.95 Then

Fβ= 1/(βx 1/P +(1-β)x 1/R) ⇔ 1 /(0.95x1/0.31+(1-0.95)x1/1) = 0.3215

Confusion Matrix for fine RBAC in Security Aspect

Normal and test cases data recorded, 21 true positives , 18 false positives, 33 true negatives, 0 false negatives.

Consider table 17 below:

Table 17 (Confusion Matrix for fine RBAC in Security Aspect)

	P'(predicted)	n'(predicted)
P(Actual)	True Positive (TP= 21)	False Negative (FN=0)
n(Actual)	False Positive (FP=18)	True Negative (TN=33)

P=TP/(TP+FP) ⇔ 21/(21+18)= 0.5385 ⇔ 54 % precision

R=TP/TP+FN ⇔ 21/(21+0) = 1 ⇔ 100 % recall

If β=0.95 Then

Fβ= 1/(βx 1/P +(1-β)x 1/R) ⇔ 1 /(0.95x1/0.54+(1-0.95)x1/1) = 0.5527

Hence since Fβ of Basic Rbac of 0.3215 < Fβ of Fine RBAC in Security Aspect of 0.5527

Fine RBAC performed better with a β of 0.95 or 95 % precision threshold.

Confusion Matrix for fine RBAC and Dynamic Parse Tree in Security Aspect

Normal and test cases data recorded, 21 true positives, 0 false positives, 34 true negatives, 0 false negatives. Consider table 18 below:

Table 18 (Confusion Matrix for fine RBAC and Dynamic Parse Tree in Security Aspect)

	P'(predicted)	n'(predicted)
P(Actual)	True Positive (TP= 21)	False Negative (FN=0)
n(Actual)	False Positive (FP=0)	True Negative (TN=34)

P=TP/(TP+FP) ⇔ 21/(21+0)= 1 ⇔ 100 % precision

R=TP/TP+FN ⇔ 21/(21+0) = 1 ⇔ 100 % recall

If β=0.95 Then

If β=0.95 Then

Fβ= 1/(βx 1/P +(1-β)x 1/R) ⇔ 1 /(0.95x1/1+(1-0.95)x1/1) = 1 ⇔ 100 %

Hence since Fβ of Basic Rbac of 0.3215 < Fβ of Fine RBAC in Security Aspect of 0.5527 < Fβ of hybrid defence of Fine RBAC and Dynamic Parse Tree of 1

The hybrid defence of Fine RBAC and Dynamic Parse Tree performed better than the assumed β of 0.95 or 95 % precision threshold to 100% precision. It must be noted however that even though these results were consistent after repeated testing with the same test cases, a real world based and

more extensive testing could exposure the security aspect in ways the study could not demonstrate and might generate different results.

4.2.8 Chapter Summary

The chapter presented the results obtained for testing of the proposed model. Generally, it was observed that the proposed model was successful in stopping illegitimate data access requests due to the fine-grained RBAC model, and in detecting and stopping SQLIA. The corresponding form access requests and SQLIA attacks were mostly successful except in a few exceptions of form access requests attacks, which did not involve data partitions that were detected and stopped by the control system. Lastly, the performance measure of using precision, recall and f-measure indicated a progressive strengthing of security as the precision of the basic RBAC model in the control system of 31% increased to 54 % with the addition of the fine grained RBAC defence and finally increased to 100% with the addition of static/dynamic parse tree comparism to fine grained RBAC with 100% recall in all cases and 32 %, 55%, and 100 % f-measure respectively. In addition, the time cost of the performance of using the defence model was negligible, hence, showing the proposed defence does not add any meaningful performance time cost to potential systems that might use it for future works in this context.

5 CHAPTER FIVE

DISCUSSION AND RECOMMENDATION

5.1 Introduction

This chapter discusses how the research objectives were met, findings, limitations of study, recommendations, and future work. The study sought to develop a feasible and efficient SQLIA defence to address the problem of security in legacy system migration to the web environment within the West African Context.

5.2 How the Objectives of the Study were met

In this section, we outline how the objectives of the study were met

a) *Determine the most applicable SQL injection attacks on a migrated legacy system in its new Web-based environment*

   Based on the literature review, we selected the three commonest SQLIA attacks and based on these three types, developed a test suite of SQLIA, which were customised for our exemplar system. In the literature review, we examined different SQLIA types, such as blind SQLIA, but these types were ruled out due to various reasons, including easy defence (such as filtering) and high time consumption for the hacker.

b) *Develop a fine-grained Role-Based Access Control (RBAC) security mechanism that uses task, operation, and data partition access to web resources to replace the basic RBAC security approach currently used in the legacy system*

   A fine-grained RBAC security mechanism was developed using these aspects of task, operation and data partition based on the business rules of the exemplar. As indicated in the results, this finer granularity of RBAC was able to stop certain illegitimate form access requests than the basic RBAC in the original system was unable to do.

c) *Develop a combined fine-grained RBAC and dynamic/static parse tree comparison into a single defence shield against SQLIA for the Sub –Saharan African context*

   After developing a fine-grained RBAC, this RBAC was combined with a dynamic/static parse tree comparison mechanism in order to form a hybrid two-tiered security shield. As indicated from the results, this shield was able to stop all SQLIA, from our suite of test cases, without false positives, while the control system, which

lacked this shield allowed SQLIA to attack the database except for a few illegitimate form access requests that it was able to stop.

d) *Centralize this hybrid defence shield security into a mechanism for easy deployment, administration and reuse within the West African context*

This newly developed shield was incorporated into a standalone system, which can easily be attached to many web-migrated legacy systems. As it relies on existing business information already incorporated into database tables, it is relatively easy to configure and requires very minimal skills than other SQLIA tools and no additional infrastructure. Furthermore, by relying on existing information within database tables, the degree of administration required for this tool is relatively low. This is because the shield is integrated into a standalone security system that pulls required information from existing database tables; hence, it can easily be reused in many different legacy systems. Its low skill base requirement and relatively easy deployment characteristics are highlighted in Table 2.1 in the literature view of this study. Consequently, due to the system's ease of deployment, reuse, low administrative cost, and low skill requirement, it is ideal for the West African context

5.3 Discussion of Findings

From the results, the fine-grained RBAC allowed all legitimate form access requests to proceed; hence, no false positives were obtained. Naturally, the basic RBAC control system allowed the same access requests as these access requests were part of its design.

The fine-grained RBAC, with its data partition, stopped ALL illegitimate form access requests from processing. The control system with the basic RBAC lacked the fine granularity of a data partition; consequently, it was unable to stop all illegitimate form access requests and allowed a few to be processed.

However, the hybrid fine-grained RBAC with the static/dynamic parse tree comparison defence stopped all SQLIA test cases. On the other hand, the control system, which did not this type of defence system, did not stop all SQLIA test cases, which resulted in damage of the database.

Measurement of the performance of form access requests and SQL query evaluation indicated a progressive precision gain in security defence in basic RBAC of 31% rose to 54 % with the addition of fine grained RBAC and finally to 100% with the application of the hybrid defence of fine grained RBAC and static/dynamic parse tree comparisms with recall of 100% in all

cases and f-measure of 32%, 55% and 100% respectively. There were no significant measurable performance time degradation. Performance was measured using the security component through a function within it that feeds the SQLIA test suite and measures the times of execution for performance records.

Given that the legitimate queries were pulled from a simple database table to construct the static parse tree model, which used a function within the security component to construct its corresponding dynamic parse tree for comparison. This security system requires very little configuration and maintenance with low- skill levels.

Although programme tracing made testing slower, it allowed the researcher to determine exactly where in the defence mechanism a particular SQLIA was stopped.

5.4 Limitations

The applicability of the proposed security model was tested on a legacy system that had been newly migrated unto a web-based environment. The testing of the system was limited by the representativeness of the test cases and limited types of SQLIA cases used. Although SQLIA selected were few, they represented the commonest types of SQLIA which the system might encounter. Also, this research was conducted through laboratory tests, field tests were carried out. Hence, practical problems and attacks that might be encountered in the field are unknown. The SQLIA attacks in the study were done at only the application end. This approach was used with the assumption that other SQLIA entry points such as the network and database would be encrypted and properly firewalled, respectively.

This research made a case for generalization in that the West African context consisted of a low skill base, poor infrastructure, and required easy deployment solutions, which are cheap. Consequently, a security mechanism that met these conditions was developed. However, it was realised that, in a few isolated cases, these conditions might not apply.

5.5 Recommendations

Based on the findings, the following recommendations were made:

1) Given this security mechanism's requirement of minimal skills, easy configuration, easy deployment, and no additional infrastructure as well as its efficacy for defending against a selected suite of SQLIA test cases, it is proposed to be ideal for systems in the West African context.

2) GPRTU can use this security defence to defend its system from hackers.

5.6 Future Work

More types of SQLIA and a larger pool of SQLIA test cases should be utilised on different systems for a more comprehensive testing.

Given the number of screens, only a representative number of screens per user role could be tested and not the complete set. With automated testing, exhaustive testing of all screen access requests per job role could be performed.

This research involved the use of one system; consequently, it cannot be ascertained how the mechanism would fare in a high traffic networked environment with multiple simultaneous SQLIA attacks.

In using this mechanism, field testing should be conducted in order to ensure that the findings of this study are consistent with those systems in the field and any possible security gaps identified by users should be addressed. In addition, field testing would enable the mechanism to be tweaked for real-world problems.

6 REFERENCES:

Agboh, D. K. 2015. Drivers and challenges of ICT adoption by SMES in Accra metropolis, Ghana. Journal of Technology Research, 6: 1.

Agboh, D. K. 2015. Drivers and challenges of ICT adoption by SMES in Accra metropolis, Ghana. Journal of Technology Research, 6: 1.

Agyeman, W. 2013. Measurement of service quality of “Trotro” as public transportation in Ghana: A case study of the city of Kumasi. In: Proceedings of 32nd Southern African Transport Conference (SATC 2013). Citeseer, 8-11.

Aliero, M. S., Ardo, A. A., Ghani, I. and Atiku, M. 2016. Classification of Sql Injection Detection And Prevention Measure. IOSR Journal of Engineering, 6 (02)

Amankwah-Amoah, J. 2019. Technological revolution, sustainability, and development in Africa: Overview, emerging issues, and challenges. Sustainable Development,

Amoako, Y. A., Awuah, B., Larsen-Reindorf, R., Awittor, F. K., Kyem, G., Ofori-Boadu, K., Osei-Bonsu, E. and Laryea, D. O. 2019. Malignant tumours in urban Ghana: evidence from the city of Kumasi. BMC cancer, 19 (1): 267.

Asabere, N., Togo, G., Acakpovi, A., Torgby, W. and Ampadu, K. 2017. AIDS: An ICT model for integrating teaching, learning and research in Technical University Education in Ghana. International Journal of Education and Development using ICT, 13 (3)

Ayebeng Botchway, E. and Yeboah-Boateng, E. O. 2019. IoT Readiness of Project Management Teams Within Local Government Organizations in Ghana. International Journal of Civil Engineering and Technology, 10 (07)

Boaheng, J. M., Amporfu, E., Ansong, D. and Osei-Fosu, A. K. 2019. Determinants of paying national health insurance premium with mobile phone in Ghana: a cross-sectional prospective study. International journal for equity in health, 18 (1): 50.

Boaheng, J. M., Amporfu, E., Ansong, D. and Osei-Fosu, A. K. 2019. Determinants of paying national health insurance premium with mobile phone in Ghana: a cross-sectional prospective study. International journal for equity in health, 18 (1): 50.

Boaheng, J. M., Amporfu, E., Ansong, D. and Osei-Fosu, A. K. 2019. Determinants of paying national health insurance premium with mobile phone in Ghana: a cross-sectional prospective study. International journal for equity in health, 18 (1): 50.

Boyd, S. W. and Keromytis, A. D. 2004. SQLrand: Preventing SQL injection attacks. In: Proceedings of International Conference on Applied Cryptography and Network Security. Springer, 292-302.

Creswell, J. W. and Miller, G. A. 2009. Research Methodologies and the Doctoral Process. New Directions for Higher Education, 99: 33-46.

Dadashzadeh, M. 2020. A simpler approach to set comparison queries in SQL. Journal of Information Systems Education, 14 (4): 1.

De Martin, J. C. 2019. Ethical and Socially-Aware Data Labels. In: Proceedings of Information Management and Big Data: 5th International Conference, SIMBig 2018, Lima, Peru, September 3-5, 2018, Proceedings. Springer, 320.

Dei, D.-G. J. 2018. Assessing the Use of Ict in Teaching and Learning in Secondary Schools. Library Philosophy and Practice,

Dogbe, E., Millham, R. and Singh, P. 2013. A combined approach to prevent SQL Injection Attacks. In: Proceedings of 2013 Science and Information Conference. IEEE, 406-410.

Donya, R. K. and Kumah, A. 2011. Cellular phone usage and productivity among employees in a Ghanaian SME: an assessment. International Journal of Computing and ICT Research, 5 (1): 21-34.

Düntsch, I. and Gediga, G. 2020. Indices for rough set approximation and the application to confusion matrices. International Journal of Approximate Reasoning, 118: 155-172.

Feng, K., Gu, X., Peng, W. and Yang, D. 2019. Moving Target Defense in Preventing SQL Injection. In: Proceedings of International Conference on Artificial Intelligence and Security. Springer, 25-34.

George, T. K., Jacob, K. P. and James, R. K. 2019. A Proposed Framework Against Code Injection Vulnerabilities in Online Applications. Journal of Internet Technology, 20 (1): 83-96.

Gyaase, P. O., Gyamfi, S. A., Kuranchie, A. and Koomson, F. S. 2020. The Integration of Information and Communication Technology in Pre-University Education in Ghana: A Principal Component Analysis. In: Handbook of Research on Diverse Teaching Strategies for the Technology-Rich Classroom. IGI Global, 109-123.

Halfond, W. G. and Orso, A. 2005. AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks. In: Proceedings of Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering. ACM, 174-183.

Helo, P., Gunasekaran, A. and Rymaszewska, A. 2017. Role of technology in servitization. In: Designing and Managing Industrial Product-Service Systems. Springer, 57-71.

Hunter, L. M. 2006. Household strategies in the face of resource scarcity in coastal Ghana: are they associated with development priorities? Population research and policy review, 25 (2): 157-174.

Jain, S. and Pais, A. R. 2011. Model Based Approach to Prevent SQL Injection Attacks on .NET Applications. International Journal of Computer Science & Informatics, 1 (11)

Kumar, V. and Abdul, M. 2019. Data Migration From Legacy Systems To Koha: A Practical Approach. Library Philosophy and Practice: 1-17.

Landgrebe, T. C. and Duin, R. P. 2008. Efficient multiclass ROC approximation by decomposition via confusion matrix perturbation analysis. IEEE transactions on pattern analysis and machine intelligence, 30 (5): 810-822.

Leckson-Leckey, G. T., Osei, K. A. and Harvey, S. K. 2011. Investments in information technology (IT) and bank business performance in Ghana. International Journal of Economics and Finance, 3 (2): 133-142.

Li, Q., Wang, F., Wang, J. and Li, W. 2019. LSTM-Based SQL Injection Detection Method for Intelligent Transportation System. IEEE Transactions on Vehicular Technology, 68 (5): 4182-4191.

Liu, X., Li, B., Chen, H., Sun, Z., Liang, Y.-C. and Zhao, C. 2019. Detecting Pilot Spoofing Attack in MISO Systems With Trusted User. IEEE Communications Letters, 23 (2): 314-317.

Manjunatha, K. and Kempanna, M. 2019. RAMIFICATION ANALYSIS OF SQL INJECTION DETECTION IN WEB APPLICATION. International Journal of Computer Science and Information Security (IJCSIS), 17 (7)

Millham, R. 2010. Migration of a legacy procedural system to service-oriented computing using feature analysis. In: Proceedings of 2010 International Conference on Complex, Intelligent and Software Intensive Systems. IEEE, 538-543.

Millham, R. and Dogbe, E. 2011. Aspect-oriented security and exception handling within an object oriented system. In: Proceedings of 2011 IEEE 35th Annual Computer Software and Applications Conference Workshops. IEEE, 321-326.

Parnami, P., Jain, A. and Sharma, N. 2019. Toward Adapting Metamodeling Approach for Legacy to Cloud Migration. In: Ambient Communications and Computer Systems. Springer, 275-284.

Patrick, H. and Fields, Z. 2017. A need for cyber security creativity. In: Collective Creativity for Responsible and Sustainable Business Practice. IGI Global, 42-61.

Paul, M. and Das, A. 2017. Health Informatics as a Service (HIaaS) for developing countries. In: Internet of Things and Big Data Technologies for Next Generation Healthcare. Springer, 251-279.

Pei Breivold, H. 2019. Towards factories of the future: migration of industrial legacy automation systems in the cloud computing and Internet-of-things context. Enterprise Information Systems: 1-21.

Rahman, R. U., Verma, R., Bansal, H. and Tomar, D. S. 2020. Classification of Spamming Attacks to Blogging Websites and Their Security Techniques. In: Encyclopedia of Criminal Activities and the Deep Web. IGI Global, 864-880.

Raman, R. H. A. 2019. Enhanced Automated-Scripting Method for Improved Management of SQL Injection Penetration Tests on a Large Scale. In: Proceedings of 2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE). IEEE, 259-266.

Sadeghian, A., Zamani, M. and Ibrahim, S. 2013. SQL injection is still alive: a study on SQL injection signature evasion techniques. In: Proceedings of 2013 International Conference on Informatics and Creative Multimedia. IEEE, 265-268.

Sadotra, P. and Sharma, C. 2017. SQL Injection Impact on Web Server and Their Risk Mitigation Policy Implementation Techniques: An Ultimate solution to Prevent Computer Network from Illegal Intrusion. International Journal of Advanced Research in Computer Science, 8 (3)

Serianu. 2017. Demystifying Africa's Cyber Security Poverty Line. Available: https://www.serianu.com/downloads/AfricaCyberSecurityReport2017.pdf (Accessed

Talukder, M. A. I., Shahriar, H., Qian, K., Rahman, M., Ahamed, S., Wu, F. and Agu, E. 2019. DroidPatrol: A Static Analysis Plugin For Secure Mobile Software Development. In: Proceedings of 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC). IEEE, 565-569.

Taylor, C. and Sakharka, S. 2019. BEST PAPER AT SIGCSE 2019 IN THE CURRICULUM INITIATIVES TRACK:') DROP TABLE textbooks;--: an argument for SQL injection coverage in database textbooks. ACM Inroads, 10 (2): 58-64.

Taylor, C. and Sakharka, S. 2019. BEST PAPER AT SIGCSE 2019 IN THE CURRICULUM INITIATIVES TRACK:') DROP TABLE textbooks;--: an argument for SQL injection coverage in database textbooks. ACM Inroads, 10 (2): 58-64.

Varshney, K. and Ujjwal, R. 2019. LsSQLIDP: Literature survey on SQL injection detection and prevention techniques. Journal of Statistics and Management Systems, 22 (2): 257-269.

Visa, S., Ramsay, B., Ralescu, A. L. and Van Der Knaap, E. 2011. Confusion Matrix-based Feature Selection. MAICS, 710: 120-127.

Yassein, M. B., Aljawarneh, S. and Wahsheh, Y. A. 2019. Survey of Online Social Networks Threats and Solutions. In: Proceedings of 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT). IEEE, 375-380.