



University of
Southern
Queensland

SECURITY ASSURANCE PROCESS FOR SERVICE
COMPONENT-ORIENTED APPLICATION LOGIC FOR SOCIAL
INTERACTION IN E-COMMERCE BANKING APPLICATIONS

A Thesis Submitted by

Faisal Nabi

MSc.e-BusMgmt

For the Award of

Doctor of Philosophy

2021

ABSTRACT

Application logic in e-commerce refers to features and behaviours unique to the application. Each application has its own specific handling of user inputs, user behaviour and communication with third-party components, while the weakness of component business logic is unique, there are significant web vulnerabilities that are common, impactful, and can be readily exploited. Usually, a logic weakness exists when an intruder violates legitimate application-specific functionality, against the intentions of developers.

In this research, we will investigate and discuss design flaw / logical flaw that causes business logic attack in the service-component-oriented application, at the n-tier architecture. The purpose of this research is to explore the causes of application logical flaws in service component architectural-based applications. There is clearly a need for a methodology able to deal with the logical flaws that normally do not show attack patterns or signatures, which are thereby hard to discover through automated techniques. Recent techniques to secure component-oriented applications normally focus on technical vulnerability. This can rely on security analysis and detection tools for vulnerability identification. The auditors mostly follow such policies that are based on checking a limited list of security issues/vulnerabilities. Therefore, we have observed that the technique of custom-developed business logic often falls short in its ability to discover vulnerabilities. We have also noticed a significant number of attacks recently classified as business logic attacks. Many security techniques have been introduced for service component-oriented architecture in recent years, but they are at the high level of service component-oriented architecture and do not address the middle-tier (business-tier) in component-oriented systems. The main focus is to research business logic vulnerability in the service component-oriented applications using security breach scenarios (case-study) in the banking domain, also examining the re-usability of design specification in the component. Furthermore, this approach is supported by a taxonomy of logical vulnerability in service component e-commerce, this taxonomy is validated by the proposed model in Chapter 4 B and event attack modeling in service component architecture in Chapter 5. It has a close relationship between the proposed taxonomy and the projected scenario of event attack modeling. Keeping in view this research further moves toward the logical solution of application logic.

Therefore, we propose a secure design method as a security assurance methodology, which uses social e-commerce as a modeling tool to demonstrate the features of this methodology. This method will be validated through Integration using UML modeling and system assurance process. This will be further reflected in a security feature-based UML. Sec modeling as an example B2c ATM model, demonstrated in social interactions of e-commerce component-based-application security modeling.

CERTIFICATION OF THESIS

This Thesis is the work of Faisal Nabi except where otherwise acknowledged, with the majority of the authorship of the papers presented as a Thesis by Publication undertaken by the Student. The work is original and has not previously been submitted for any other award, except where acknowledged.

Principal Supervisor: Professor. Jianming Yong

Associate Supervisor: Associate Professor. Xiaohui Tao

Student and supervisors' signatures of endorsement are held at the University

ACKNOWLEDGEMENTS

First and foremost, I would like to express my heartfelt gratitude to my creator, Allah - the most gracious, beneficial, and kind. Then, I'd want to convey my deepest gratitude and admiration to Professor Jianming Yong, my principal supervisor, for his invaluable assistance, intelligent suggestions, inspiration, and support during the study. His contributions to my PhD studies have been essential. The constructive criticism I received from him on a regular basis greatly improved the quality of my PhD study. I'd also like to convey my heartfelt gratitude to Professor Xiaohui Tao, my associate supervisor, who has always encouraged me to work hard and meet my deadlines. His valuable feedback helped me enhance my work, and he also pushed me to maintain the positive attitude I needed during the project. Moreover, two external authors also participated in one publication, and the work done by them is only 10% (Muhammad Farhan and Nauman Naseem). Without the University of Southern Queensland's (USQ) funding [USQ International Fees Research Scholarship], the research would not have been possible. In addition, I appreciated USQ's outstanding working facilities, library resources, and cutting-edge logistical support. The author would like to thank Peer Bukhari who so much supported me while completing the thesis into one document. He is my religious teacher and great Sufi saint (Peer Bukhari order).

STATEMENT OF CONTRIBUTION

The following detail is the agreed contribution of candidates and co-authors in the presented publications in this thesis:


Article I:

Nabi, Faisal and Yong, Jianming and Tao, Xiaohui (2020) A security review of event-based application function and service component architecture, IGI International Journal of Systems and Software Security and Protection, 11 (2). pp. 58-70. ISSN 2640-4265.

The first paper's overall contribution of Faisal Nabi is 90% to the concept development, data collection, research analysis and revising of the final submission.


Jianming Yong & Xiaohui Tao contributed 10%, assisted in designing the study, supervised research analysis and review the research concepts, results, and final manuscript approval.

Article II A:

Nabi, Faisal and Yong, Jianming and Tao, Xiaohui  (2020) *Classification of logical vulnerability based on group attacking method*. In: 11th International Conference on Ambient Systems, Networks and Technologies (ANT 2020), 6-9 April 2020, Warsaw Poland.

The second paper's overall contribution of Faisal Nabi is 85% to the concept development, data collection, research analysis and reviewing of current articles and deciding the eligibility for inclusion drafting the paper. Dr Jianming Yong and Xiaohui Tao contributed 15%, assisted in the review of the concept developed, approve the idea, supervised research analysis, and review the research concepts, results, and final manuscript approval.


Article II B:

Nabi, Faisal and Yong, Jianming and Tao, Xiaohui  , Muhammad Farhan, Nauman Naseem (2021) Organizing Classification of Application Logic Attacks in Component-based E-Commerce Systems, Journal of computer science Q3 validated the proposed model in Chapter 3 and developed a taxonomy for developers of J2EE platform. This helps to improve the CVC database which is used for vulnerability reporting and information

gathering. The second paper B's overall contribution by Faisal Nabi *is* 80% to the concept development, data collection, research analysis and designing system modeling and tool selection to develop the final manuscript and drafting of the paper.

Jianming Yong and Xiaohui Tao contributed 10%, assisted in the review of the concept developed, approve the idea, supervised research analysis, and review the research concepts, results, and final manuscript approval. Whereas other authors in publication B just involved in data gathering and survey so their contribution is only 5% each.

Article III:

Nabi, Faisal and Yong, Jianming and Tao, Xiaohui  (2020) A novel approach for component-based application logic event attack modeling. *International Journal of Network Security*, 22 (3). pp. 437-443. ISSN 1816-353X

The third paper's overall contribution of Faisal Nabi *is* 90% to the concept development, data collection, research analysis and designing system modeling and tool selection to develop the final manuscript and drafting of the paper.

Jianming Yong and Xiaohui Tao contributed 10%, assisted in the review of the concept developed, approve the idea, supervised research analysis, and review the research concepts, results, and final manuscript approval.

Article IV

Nabi, Faisal and Yong, Jianming and Tao, Xiaohui  (2021) Security aspects in modern service component-oriented application logic for social e-commerce systems. *Social Network Analysis and Mining*, 11 (1):22. ISSN 1869-5450.

The fourth paper's overall contribution of Faisal Nabi *is* 85% to the concept development, data collection, research analysis and proposition of a conceptual model design that follows system modeling and tool selection to develop the final manuscript and drafting of the paper.

Jianming Yong and Xiaohui Tao contributed 15%, assisted in the review of the concept developed, approve the idea, supervised research analysis, and review the research concepts, results, and final manuscript approval.

Article V

Nabi, Faisal and Yong, Jianming and Tao, Xiaohui (2021) An Approach of social interaction with software connectors & the role of façade components for secure application logic (under review).

The fifth paper's overall contribution of Faisal Nabi *is* 85% to the concept development, data collection, research analysis and proposing a conceptual model design that follows system modeling and tool selection to develop the final manuscript and drafting of the paper.

Jianming Yong and Xiaohui Tao contributed 15%, assisted in the review of the concept developed, approve the idea, supervised research analysis, and review the research concepts, results, and final manuscript approval.

TABLE OF CONTENTS

	Page
Contents	
Abstract	i
Certification of Thesis	iii
Acknowledgements	iv
Statement of Contribution	v
Table of Contents	viii
List of Figures	xiii
List of Tables	xv
List of Abbreviations	xvi
List of Publications	xix
Chapter 1. Introduction and Research Background	1
1.Introduction	1
1.1 The Scope of Research	8
1.1.2 Research Philosophy	9
1.2 Motivation and Problem Statement	10
1.2.1 Objects and Aims	10
1.2.2 Research Gap	12
1.2.3 Research Contribution	12
1.2.4 Structure of the thesis	13
Chapter 2. Literature Review & Problem Analysis	15
2.1 Current banking CBS-based system analysis	22
Chapter 3. A Security Review of Event-Based Application Function and Service Component Architecture	25
Chapter 3 research findings & Introductory note define the relationship between Chapter 1.2 and Chapter 3	25
Abstract	26
3.1 Introduction	26
3.1.1 Motivation	27
3.1.2 Problem Statement	28
3.1.3 Research Method	28
3.2 Research Background	28

3.2.1 Service Component Architecture	29
3.2.2 Specification of the Service Component	29
3.2.3 Security Properties in SCA	30
3.3 Related Work	30
3.4 Event Attack in Service Component and Composite Application	31
3.4.1 Security Features of Event-Based System	31
3.5 A Review of Event-Based Security Modeling	32
3.5.1 Security by Design Event Control Modeling	32
3.5.2 Security Component-Based Modeling SCA Approach	33
3.5.3 Event Attack Modeling Approach	34
3.5.4 Comparison of Modeling Technique and Discussion	35
3.5.4.1 Attack Analysis Model	35
3.6 Conclusion	36
References	37
Chapter 4. (A) Classification of Logical Vulnerability Based on Group Attacking Method	39
Research findings & Introductory note define the relationship between Chapter 3 and Chapter 4	39
Abstract	40
4.1 Introduction	41
4.1.1 Objective	41
4.1.2 Method	41
4.2 Related work	41
4.3 Proposed Vulnerability Classification Model	42
4.3.1 Classification of Logical Vulnerability VS Technical Vulnerability	43
4.3.2 Layer Based Software System Scenario Attack Modeling	44
4.3.3 Classifying and Categorizing Logical Vulnerabilities	46
4.4 Discussion	47
4.5 Conclusion	47
References	47
Chapter 4B. Organizing Classification of Application Logic Attacks in Component-based E-Commerce Systems	49
Abstract	49

Research Findings and concluding note, as mentioned in above Chapter * 4A*	49
Introduction	49
Research Methodology	50
Related Research Work and Taxonomic Properties	51
Previous Research Work and Classifications	53
Proposed Classification and Types of Logic Attacks	55
Conclusion	58
Chapter 5. A Novel Approach for Component-based Application Logic Event Attack modeling	61
Research Findings & Introductory note define the relationship between Chapter 4 and Chapter 5	61
Abstract	62
5.1 Introduction	62
5.2 Problem Statement	63
5.2.1 Research Philosophy	63
5.2.2 Research Gap	63
5.2.3 Current Approaches in Attack Modeling	63
5.3 Studying Case Profile & Event Attack Modeling	64
5.3.1 Component Application Logic Design Fault	64
5.3.1.1 Class of Vulnerability	64
5.3.2 Case Scenario Based Experimental Study	65
5.3.3 Theoretical Analysis of Proposed Approach	65
5.3.4 Systematical Comparison of the Proposed Scheme	66
5.3.5 Discussion	66
5.4 Related Work	66
5.5 Conclusions	66
References	66
Chapter 6. Security aspects in modern service component-oriented application logic for social e-commerce systems	69
Research Findings and Introductory note define relation between Chapter 5 and Chapter 6	69
Abstract	70
6.1 Introduction	70
6.1.1 Problem statement	71
6.1.2 Objective and contribution	73

6.2	Related work	72
6.3	Case study-based research method	73
6.3.1	The impact of flaws in application business logic	73
6.3.2	Case study-based research scope	74
6.4	A practical example: social commerce-based e-banking case study	74
6.4.1	A composite application functionality and business process	74
6.4.2	Exploitation	75
6.4.3	Security breach case summary	77
6.5	Proposed security assurance methodology	77
6.5.1	Security risk analysis	78
6.5.2	Threat modeling of application logic vulnerability	79
6.5.3	Taxonomic classification of software vulnerabilities	80
6.5.4	Component fault detection model	80
6.5.5	Modeling the application and its components without fault	82
6.5.6	Designing security by design application modeling	82
6.6	A validation and verification of method integration testing model	84
6.7	Discussion	86
6.8	Conclusion and future directions	87
	References	87
 Chapter 7. Social Interaction with Software Connectors & the Role of Façade-Based Components for Secure Application Logic		89
	Research Findings & Introductory note define the relationship between Chapter 6 and Chapter 7	89
	Abstract	91
	7.1 Introduction	91
	7.2 Research Design	93
	7.2.1 Research Motivation	94
	7.3 Review of Existing Work	95
	7.3.1 Component-based Application Logic	96
	7.3.2 Taxonomy of Software Connector	97
	7.3.3 Connector Architecture and Lifecycle	98
	7.4 Designing a Secure Façade-based Connector	99
	7.4.1 UML-based ATM Secure Façade-based Connectors Modeling	100

7.4.2 Technical analysis of data transmission Packets in the Proposed ATM Model	102
7.4.3 Centralized Security Session Façade	105
7.4.4 Security Analysis Using Session Façade	105
7.5.5 A Comparison of Software Connector Modeling Approaches	106
7.5 Discussion	107
7.6 Conclusion	108
References	108
Chapter 8. Research Results and Conclusion	111
8.1 Introduction-related thesis question and contribution to the research	111
8.2 Conclusion	119
8.2.1 Future Research Work	120
References	121

LIST OF FIGURES

Figure No.	Figure name	Page number
Figure 1.1	Architectural model of banking application organizational process	2
Figure 1.2	Organizational process of component-oriented service-based application	3
Figure 1.3a	Service Component Oriented Application Layer Model	5
Figure 1.3b	Component-based Java Service-Oriented Architecture Layer Model	5
Figure 1.4:	Architecture consists of components & services	8
Figure 1.5	Flowchart of Thesis by Publication	14
Figure 2.1 (a)	Reusable component integration pattern	21
Figure 2.1 (b)	Reusable components are binding in service integration	21
Figure 3.1	Events call application inter-communication process	28
Figure 3.2	Event request transaction diagram	32
Figure 3.3	Event-based business functional process	33
Figure 3.4	Service component architecture-based security component composite	34
Figure 3.5	Event-based subversion attack modeling	36
Figure 3.6	Analysis of attack event process model	42
Figure 4.1	The Proposed Vulnerability Classification Model	42
Figure 4.2	Classification of Vulnerability Scheme	42
Figure 4.3	Layer-Based Software System Attack Model	39
Figure 4.b.1	The Proposed Vulnerability Classification Model	42
Figure 4.b.2	Classification of Vulnerability Scheme	44
Figure 4.b.2	Layer Based Software System Attack Model	45
Figure 4.c.1	SVAM Model	51
Figure 4.c.2	Taxonomy of Software Vulnerabilities causes	53
Figure 4.c.3	Web attacks taxonomy	54
Figure 4.c.4	Application Logic Vulnerability Graph	55
Figure 4.c.5	Characterization of vulnerability	56
Figure 4.c.6	Logical vulnerabilities Vs technical vulnerabilities	57
Figure 4.c.7	Vulnerability mitigating in context software design assurance phase process	58
Figure 5.1	Component-based application logic event attack scenario	62
Figure 5.2	Attack graph with attack path against systems	64
Figure 5.3	C customer component code	65
Figure 5.4	Subversion attack event scenario	65
Figure 5.5	Event attack subversion logic scenario	66
Figure 5.6	Cyber-attack theory model	75
Figure 6.1	Customer information handling & reused component sociale-commerce in e-banking	75
Figure 6.2	Event-attack-modeling & system exploitation social e-banking	76
Figure 6.3	Subversion attack mapped through social commerce banking service flow	77
Figure 6.4	Security risk analysis model	78
Figure 6.5	Threat modeling of subversion attack	79
Figure 6.6	Taxonomic classification of software vulnerability	80
Figure 6.7	Component fault detection model (CFDM)	81
Figure 6.8	UML Sec 2.0 modeling fault detection of air control system	82
Figure 6.9	UML Sec 2.0 security by design approach multi-specification J2EE system modeling	83

Figure 6.10	V &V integration model for security by design testing	84
Figure 6.11	V &V method for fault detection in component-oriented-service	85
Figure 6.12	Model-checking process to validate the proposed method of V &V	86
Figure 7.1	Direct and Indirect component inter-communication model	93
Figure 7.2	Social ATM Network and Software Process Environment Model	94
Figure 7.3	Application Component with Business Logic	96
Figure 7.4	Connector Role in Component-Based System and Application Logic	97
Figure 7.5	Connector Model (a) Simple Architecture (b) Compound Architecture	99
Figure 7.6	Designing Façade-based Secure Connector	99
Figure 7.7	UML Base ATM Secure Façade Connector Design Model	101
Figure 7.8	ATM Network Transmission Packet Information	103
Figure 7.9	ATM Authentication Packet Transmission Process	104
Figure 7.10	Extracted Information from XML Schema Scenario Based Validation	105
Figure 8.1	Security Assurance Model Projection of Thesis Contribution	111

LIST OF TABLES

Table No.	Table name	Page Number
Table 1	Group Attacking Method ID and Vulnerability Classification	43
Table 2	Attack pattern properties	52

ABBREVIATIONS




ATM	Automated Teller Machine
ACL	Access-Control List
API	Application Programming Interface
BMI	Body Mass Index
B2c	Business-to-consumer
BCBS	e business component-based-software
COTS	Commercial off the shelf (Kind of Component)
CGI	Common gateway interface
CVE	Common Vulnerabilities and Exposures
CBS	Component based Software
CBSD	component- based-software developed
CORBA	Common Object Request Broker Architecture
CFDM	Component fault detection model
CSP	Component Service Protocol
DFT	Design for Test
EBS	Enterprise Backup Solution
EJB	Enterprise JavaBeans
ETC	Event-triggered control
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
IBM	International Business Machines Corporation

IDS	Intrusion Detection system
IPS	Intrusion Prevention System
IT	Information Technology
J2EE	Java Platform, Enterprise Edition
JSP	Java Server Pages
JMS	Java Message Service
Java RMI	Java Remote Method Invocation
MOM	Message-oriented middleware
OWASP	Open Web Application Security Project
OASIS	Organization for the Advancement of Structured Information Standards
RDA	Rapidly developed Architecture
RPC	Remote Procedure Call
SOI	Service Object Integration
SAML	Security Assertion Markup Language
SCA	service component architecture
SOAP	Service-oriented Architecture protocol
SOA	Service Oriented Architecture
SDLC	Software Development Life Cycle
SRS	Security Functional Requirement Specification
SEP	Symantec Endpoint Encryption
SED	Symantec Endpoint Decryption
UML Sec	Unified Modeling Language Security
UML	Unified Modeling Language

V&V model	Verification and Validation Model
WS	Web Services
XACML	Extensible Access Control Markup Language

LIST OF PUBLICATIONS USED IN THESIS

The work of this thesis is based on the following publications:

- i. Nabi, Faisal and Yong, Jianming and Tao, Xiaohui (2020) A security review of event-based application function and service component architecture. IGI International Journal of Systems and Software Security and Protection, 11 (2). pp. 58-70. ISSN 2640-4265.
- ii. Nabi, Faisal and Yong, Jianming and Tao, Xiaohui  (2020) Classification of logical vulnerability based on group attacking method. *In*: 11th International Conference on Ambient Systems, Networks and Technologies (ANT 2020), 6-9 April 2020, Warsaw Poland.
- iii. Nabi, Faisal and Yong, Jianming and Tao, Xiaohui  (2020) A novel approach for component-based application logic event attack modeling. International Journal of Network Security, 22 (3). pp. 437-443. ISSN 1816-353\
- iv. Nabi, Faisal and Yong, Jianming and Tao, Xiaohui, Muhammad Farhan, Nauman Naseem (2021). Organizing Classification of Application Logic Attacks in Component-based E-Commerce Systems, Journal of computer science Q3
- v. Nabi, Faisal and Yong, Jianming and Tao, Xiaohui  (2021) Security aspects in modern service component-oriented application logic for social e-commerce systems. Social Network Analysis and Mining, 11 (1):22. ISSN 1869-5450.
- vi. Nabi, Faisal and Yong, Jianming and Tao, Xiaohui (2022) An Approach of social interaction with software connectors & the role of façade components for secure application logic, International Journal Network Security (Accepted for Publication).

CHAPTER 1: INTRODUCTION AND RESEARCH BACKGROUND

1. Introduction

The field of software engineering has changed from using conventional enterprise software application techniques to Service Component Architecture (SCA) for distributed system applications. Over the last few years, this new age has evolved with the rapid growth of component-based methods for system design. Despite these advances, the implementation of existing protected technology software features in realistic e-commerce distributed systems will do little to deter intruders. While most studies have been carried out on web application services that largely use current software, middle-tier component-based software (which rapidly develops application logic), often introduces security risk opportunities (Wang et al., 2020).

Social means public interaction, in terms of technology. Current social e-commerce practices are a subset of e-commerce that focus on security framework protocols such as secure transactional protocols, cryptographic frameworks, and standards for sanitization. These procedures are widely understood to ensure the security of social media e-commerce applications. In terms of interaction through banking channels in digital form by different mediums such applications are defined as mobile, internet, ATM and mobile ATMs, which are part of social Banking. The popularity of such applications in social e-commerce through e-banking made life so advanced and fast, which made life for people more convenient. This is the reason why this study is more interesting to researchers to focus on social interaction of people in the digitalized modern world. The factor of this is the technological impact in a new area of digital banking. On the other hand, the problems of such a domain are still a challenge for researchers, this makes this study more considerable.

These are basic types of social e-banking applications medium, which is considered as a sub-set of e-commerce, also known as social e-commerce.

The building blocks of such applications are software components, which are called business components, based on Java technology, which is very widely used for social banking applications. However, other technologies are also being used in the market such as Dcom, .Net, but the main reason for the use of Java is that it is platform-

independent, which provides more highly fast service in business application solutions in social banking.

These applications are connected through the main Banking applications, which are integrated with other Banking services, such application solutions are based mostly on Java component-oriented services through loosely coupled contacts based on offered and used interfaces of every business component. This is the main integration process of such application solutions are based on component connectors as exemplified in Chapter 7.

Therefore, each type of social e-banking medium is connected through a pre-defined interface, which is connected through the main service of the banking application. The organizational process consists of business components, which are java based on different technologies such as Java in the banking domain. The software composites are EJB session bean and entity bean components as referenced in Chapter 6 Figure 9, These are building blocks based on organizational processes. These have been defined as architectural models as displayed in the following figure in the banking application domain. This process is based on banking application integration, technology integration and based on trust management.

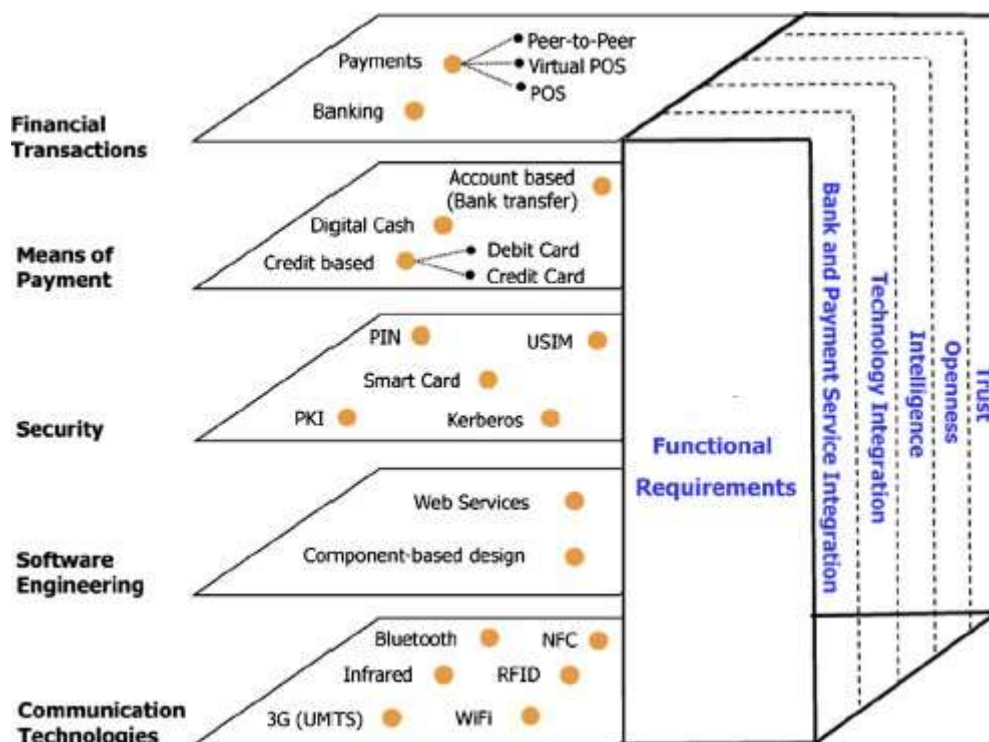


Figure 1.1: Architectural model of banking application organizational process

The banking application-based system is based on an organizational model, which is based on the following process; as displayed in Figure 1.2.

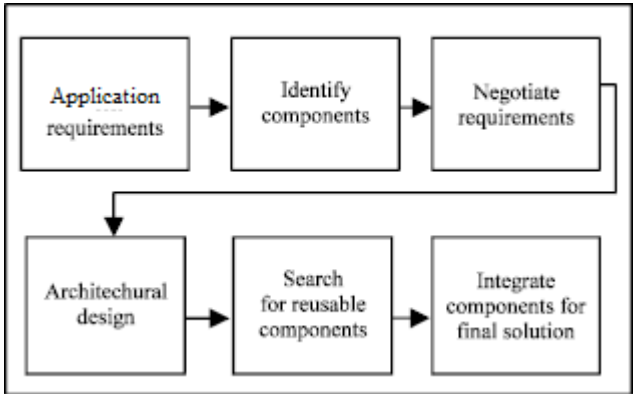


Figure 1.2: Organizational process of component-oriented service-based application

The key concern in the use of these approaches centres on software component flaws in structure and integration, which are frequently ignored when considered for business use. These issues can nullify the integrity of current information security. The weakest link in social e-commerce banking applications is the logic subversion of the part on its server-side. This occurs when the developer overlooks the business process at the time of business component integration and reuses the design specification for the current business logic (Algharabat, 2020).

Service-component-oriented social e-banking applications are being developed, with well-specified software components that are readily accessible. These are the building blocks used in-service component architecture to create services. It is necessary to follow an order in the design process of service-component-oriented applications in the social e-banking domain in which an application is designed as an organisational process once the design process is defined, which can be implemented using modular components. Each modular component is a service, which, by incorporating these services into a social e-banking framework, explains its interfaces (Wang et al., 2020).

Service component-oriented banking domain applications are built or composed of software components that are readily available and well specified. These are building blocks in a service-oriented architecture for the development of services. Within the service component-oriented application design process in the banking domain, if an application's design process is determined as an organizational task, then tasks are considered a modular components, and they need to be incorporated. Each modular element is a service that explains its interfaces by integrating these services into the banking system. The current business process relies on the integration of modules, having defined modular components (Paik et al., 2017).

System integration is not easily understandable because it is one of the most complex software design processes for complex systems, such as the banking system. The complexity of banking system integration triggers the process of continual improvement in technological and business attributes provided by the bank to satisfy the requirements of its customers. Often developed and based on different component vendors, the banking system consists of multiple platforms and design/architecture patterns. This is also due to the additional difficulty of integrating a service-oriented component-based banking application (Riad et al., 2018). Defective integration causes an attack on the business logic underpinning the process of web-based banking and causes severe financial harm. This research mainly examines business logic weaknesses in service component-oriented applications, using a real-life banking security breach scenario while explaining the component's re-usability design specification in a banking application.

The service-based architecture of components is a significant solution in modern banking, in particular, the re-use of components when developing new applications as part of distributed e-commerce systems, such as online banking services (Luhach et al., 2014). The separation of business logic (business integration logic) from implementation is the main purpose of service component architecture so that the developer can easily assemble integrated applications without knowing the implementation details. In order to achieve this, service components are constructed to allow the implementation of each service needed by business processes. Therefore, the architecture consists of 3 layers: the first is the business integration logic; the second is the service components, and the third is implementation. Business logic components are assembled independently of their implementation, even though the logic allows for implementation. In this way, the service interface remains open/available (Nurcan & Schmidt, 2015). Service component-oriented architecture infrastructure consists of the registry, a message bus, firewalls, web servers, application servers and a database. These items must be secured in a reliable service-oriented banking system. An application builder can build from existing services by using a service broker. In this process, the application builder needs to focus on the business logic, rather than the programming logic or implementation (Amirpour et al., 2016).

The service-oriented architecture is a module software development approach, whose basic artefacts are distributed, replaceable and loosely coupled components that communicate with each other using a standard contract interface, and are implemented as web services (Janssen et al. 2014; Gbaffonou et al., 2015). For security purposes, an authentication logic is not normally reusable, especially when operating with another application logic (Karimi, 2011).

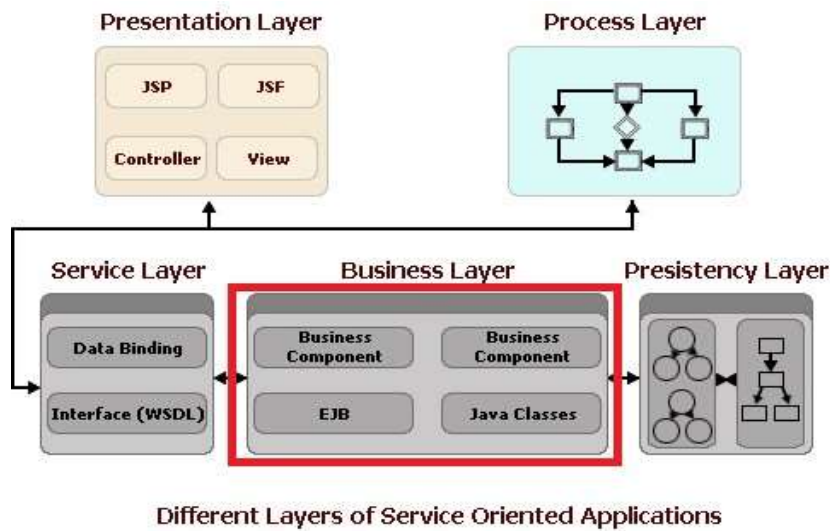


Figure 1.3 a: Service Component Oriented Application Layer Model (Gbaffonou et al., 2015)

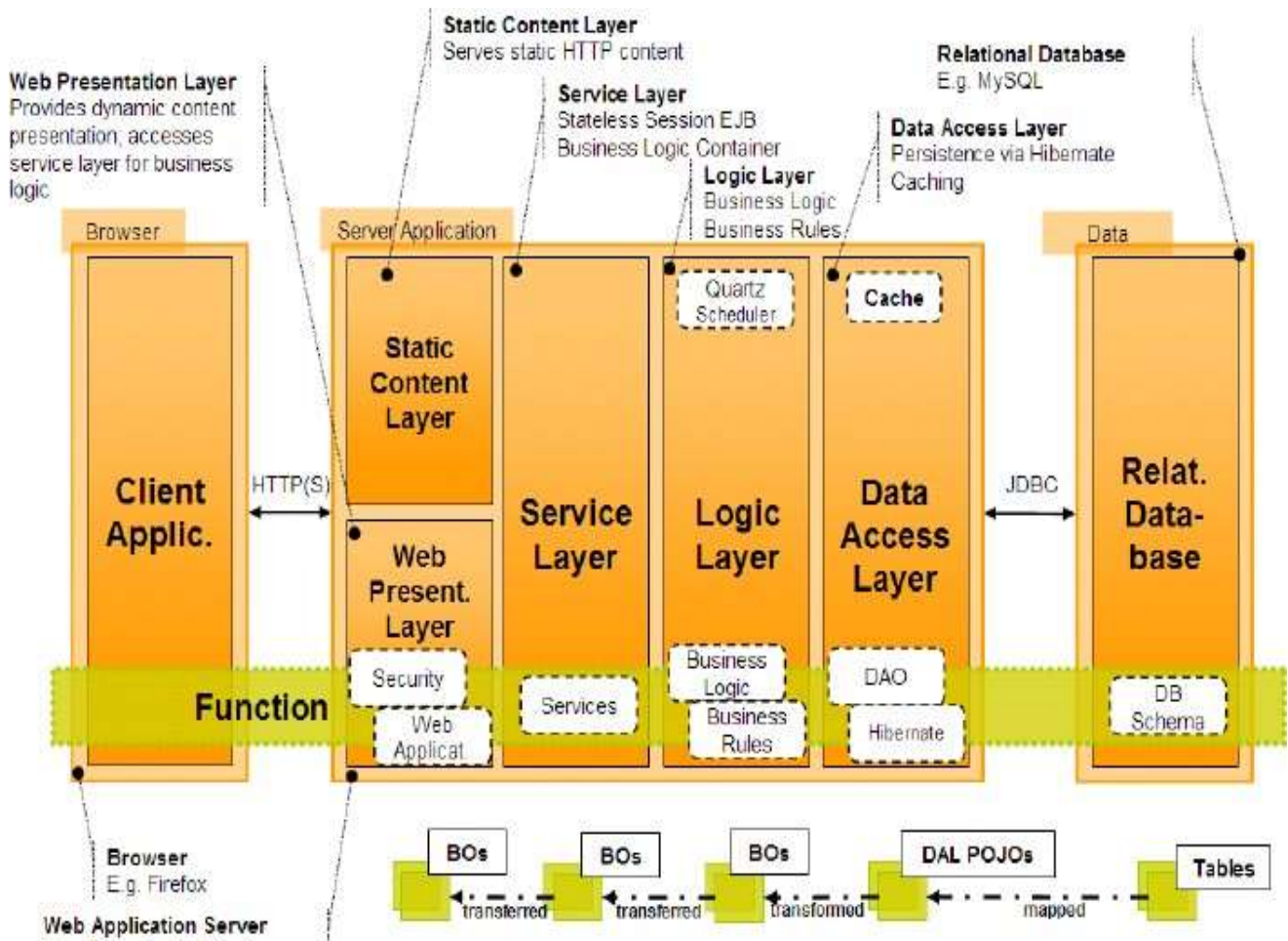


Figure 1.3 b: Component-based Java service-oriented architecture layer model (Amirpour et al., 2016)

Figure (1.3 b) explains the service component-based architecture model that clearly expresses each layer function and java beans-based business application logic. This defines the business rules, which makes application-based business logic.

As explained in the above paragraph, security is the foremost issue within the service component architecture environment. It needs to be considered carefully because components are loosely coupled. While there are several existing approaches and technologies for service security, such as WS-Security, and SAML, none of these addresses security assurance requirements for all layers within the architecture, especially at the business layer, where business logic components reside (Paik et al., 2017).

A service component-oriented architecture causes more security problems at the architecture level. Security is always a concern in the deployment of new applications, while reusability of service (components) from an existing logic of service-oriented distributed system at lower levels of system design is nevertheless always considered. However, security risks such as “design flaws” are always an issue. Security aspects are too often ignored during the development of reusable service component integration and application, causing design flaws (Amirpour et al., 2016).

Service-oriented architectures use XML-based web services, which are vulnerable to XML Signature Wrapping Attack, Coercive parsing, Oversize payload, SOAP Action Spoofing, and XML injection into SOAP middleware Hijacking. All security configuration and deployment are done at the highest level of the application. Many security tools widely in use validate the security configuration at the highest level in service-oriented online web-based banking distributed systems. In an n-tier service-component-oriented architecture, security is deployed at the client-to-web server layer, using cryptography schemes. This allows secure access control and SSL /TSL-based secure transmission of data from client to web service, but at the business layer in the middle-tier, where application logic resides. It is therefore completely lacking in systematic protection or security assurance (Luhach et al., 2014).

Current techniques are based on XML Key management specification, web services trust, XACML, Web services security, XML Digital Signature, and XML encryption.

They do provide strong security at client-server level (web-tier), that is, at layer 1 to layer 2 of the web server. They nevertheless fall short of providing a methodology that can secure middle-tier service-component oriented application logic at the business/application layer, where components are coupled, and where open contracts and designed interfaces using component offered and used interface contract design based on application logic (Wang et al., 2020 ; Nabi et al., 2020).

These security techniques (addressed above) offer only high-level service component-oriented architecture. The service component architecture has strong features like loose coupling and reusability that allow modification at run-time, while changing its system components. The drawback is, however, that services are not secured and therefore not reliable at a low level (Wang et al., 2020; Nabi et al., 2020).

The composability (CAUSE) is the best security feature provided by service-component-oriented architecture but is not secure. Therefore, any non-secure use of a service increases the risk of a security incident (Malohlava et al., 2013).

In loosely coupled services, a component or application's logic is, in practice, widely reused at the enterprise level 9 (Nabi et al., 2019). However, security becomes a significant challenge at the application server level where business components offer services: each service is its own unit of logic. The components are in a tighter and more direct coupling, which in return offers a high level of performance, as compared with the services; those are loosely coupled and usually communicate through a network, using standard protocols such as SOAP (Jiang & Willey, 2005).

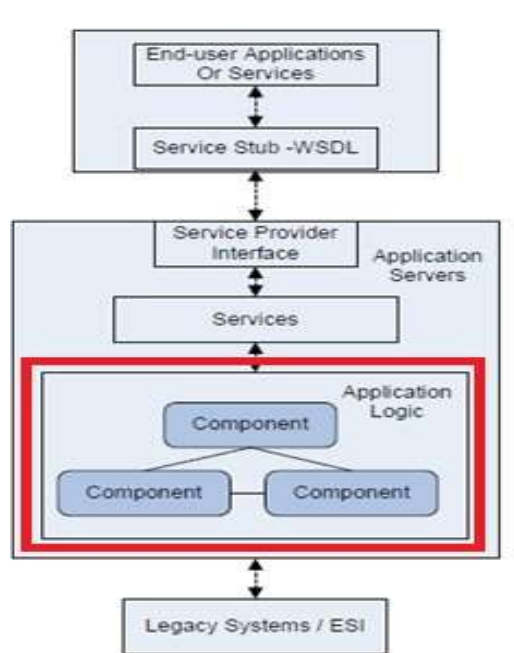


Figure 1.4: Architecture consists of components & services (Jiang & Willey, 2005).

Service component-oriented architecture opens many access points for enterprise systems - and increases vulnerabilities to the site because many of these points can be accessed through the internet. As enterprise-level security standards have changed with the advancement of security technology, the need has increased for mechanisms or methodologies that can cover core service logic security at application servers where components reside as independent services (Nabi et al., 2020).

The main example of a component-based application logic flaw is given in Chapter 6 and Chapter 4 in the form of published papers. However, subversion of logic in component-based applications may vary subject to the application domain. In banking applications, especially in social e-commerce and e-banking, the flaw is mostly caused by the reuse of component functional logic a classical example is given in Chapter 6 through a published paper that shows a real-time application logic case.

1.1 The Scope of Research

This research has significant scope in the field of a component-based banking application and security architecture, focussing especially on the middle tier where application logic security uses (SCA) design pattern. Security violation in the middle tier is a real concern and is based on a mismatch of component-design specifications within the existing logic of banking application, which may cause of subversion attack while re-using the component from existing logic to build new services (Ghassan et al., 2020; Nabi et al., 2020). It indicates

a serious violation of application integrity & security. Service-oriented component software uses two sorts of components to develop web-based banking application logic. These two components are Custom-Developed and commercial off the shelf (COTS). It is possible that they may have flaws in the design of software applications, that became apparent during the business logic integration. The CBS-based solution causes software risks that lead to logical vulnerabilities such as a component's logic subversion attack, misuse of application logic and circumvention of a component's business logic flow. All these factors together temper the application's functionality steps. In light of the research problem, we will focus on security breach subversion in an attack scenario (using the case-study method in the context of social e-commerce) related to web-based banking systems that re-use design specifications while developing new services from existing service component logic based on integration SOI methods, causing business logic vulnerability in the middle-tier of the banking architecture (Nabi et al., 2021).

1.1.2 Research Philosophy

The philosophy of research is drawn from applied science. We apply current scientific knowledge to technology, specifically, component-based software engineering theory. To ensure it is state-of-the-art, the research uses theory, knowledge, method and technique (Yaghmaie, 2017). The research philosophy also describes and explores state-of-the-art technology in component-based software design. In formulating it, we have adopted the solution for business logic weakness from applied science philosophy. At this stage, it is very important to recognise that, in the light of the research philosophy, design questions help direct research in the field of e-commerce security by ensuring that the work of the researcher is moving in the right direction and that their work is thorough and informative.

Why is a case study used as a philosophical paradigm?

The case study method is useful as a philosophical paradigm in e-commerce and it's a subset social e-commerce security studies (Laukkanen et al., 2018). This single research paradigm is particularly helpful in narrowing down a specific topic and its scope, thereby supporting philosophical investigation (Hassard & Kelemen, 2012). We have taken this approach in our research philosophy by using as a bank case study as a single research paradigm. The method used in this process is exploratory research-based case analysis as a technique to formulate the solution to business logic vulnerability phenomena. The research philosophy also improves the quality of research work.

1.2 Motivation and Problem Statement

In the service component-oriented application or enterprise approach, the application can be a combination of components that are integrated to form a particular business service or function. In service component-oriented applications, Business Process Integration (BPI) consists of business functional concern for component logic; therefore, it cannot be handled only through technical considerations, because the integration is not only based technical in nature, and mostly refers to a specific component model. However, the issue identified is the business process of component functional logic-based design solution, which is related to business components and their integration. At this stage, the focus is to consider the logical structure, where the logical problem takes place. Poor attention to design-based detecting flaws for logical structures is called business logic vulnerability (Nabi et al., 2021). We propose secure application functional process logic for e-commerce component-based applications based on security requirements of e-process and security assurance logical component behaviour specification approach to formulate and design a solution for business logic vulnerability phenomena. The first section of the methodology follows security risk analysis in the CBSD rapid business logic and defensive strategy. In addition to this, we also propose in the second section, “A security Assurance Model process” to deal with logical component-ware reusing risks in the application logic that cause logical vulnerability in e-commerce systems to encounter in such situations while reusing components from its existing application logic. This would contribute to solving identified problems. Application logic represents the translation of domain business logic that, in component-based developed application logic interoperates business processes for particular domain problems.

Therefore, it is imperative to design a solution-based methodology that will tackle and work as a security-by-design method approach for service component-oriented e-commerce applications, while considering a social e-commerce banking case study, and using a modeling methodology that helps to generate and automate vulnerability through attack scenario modeling (UML Sec & Uppaal Tool).

1.2.1 Research Objects and Aims

What sorts of problems is the research finding, in line with the research philosophy? Why is this project important, and worth doing?

The main research objects are in defining CBS-based application logical flaws in the banking system, especially when existing components-based business logic is reused.

Hence, design specification of component behaviour at the time of integration process.

Main Question: The thesis examines how to detect whether an online-based banking application service contains logical flaws in its component integration design specification.

This highlights the main object and aim that leads to the scope of this research study is to focus on investigating software application logic problems and identifying vulnerabilities that are due to a mismatch between business process specification and component ware specification at the design/architecture level while using rapid development business component-based-software approach for business application logic in e-commerce systems. Our attack patterns are more specific to what components can pinpoint vulnerability in a system design. We will only target business application Logic vulnerabilities, as explained given below questions.

Sub-Questions:

Question 1 asks how to detect design flaw-based subversion attacks in banking applications, which cause business logic vulnerability and which weaken the security of social e-commerce systems in the banking domain.

Question 2 asks how to classify and characterize two different categories of vulnerability: ie, logical and technical, that leads to a taxonomy.

Question 3 asks how to define and classify an event-attacking method revealing logical vulnerability-related flaws in service component applications at the business logic layer.

Question 4 what is the level of social interaction in component-based application logic when component design specification is reused.

Question 5 social interaction in banking application logic when subversion occurs based on black-box techniques especially social banking mediums such as ATMs?

This research addresses a burning issue in component-based service-oriented application logic security. The questions outlined above define areas, which need to be explored, yet have not been researched elsewhere, thereby indicating a clear research gap. Noting the technicality of the proposed work, findings will offer significantly increased integrity in the domain of components and service-based solutions, and therefore increased levels of assurance in security-by-design.

1.2.2 Research Gap

This research will allow us to improve the security of application business logic (Design Flaw) in-service component-oriented e-commerce applications, composed of integrated components, and responds to the research gap identified in recent research reviews (Wang et al., 2020; Nabi & Nabi, 2017; Seinturier et al., 2017). The research highlighted the significance of application logic vulnerability (allowing subversion attack, caused by design default), and the inability of vulnerability analysis or detection tools to automate this vulnerability. This project is therefore worth doing because there has been no identification of the development of a taxonomic structure of logical vulnerabilities in the middle tier of service-component oriented service. The absence of this structure often causes flaws, which COTS or home-made software components (these components are java oriented service based entity bean as shown in Figures 2.1a and 1.3b) for service integration do not address. We will investigate the problem of business logic vulnerability in the component-based rapid development of e-commerce applications while reusing the design specification of components. We propose secure application functional processing Logic Security technique for component-based e-commerce application, based on security requirement of e-business process and security assurance logical component behaviour specification approach to formulate and design a solution for business logic vulnerability phenomena. This is justified in Chapter 2 - Literature Review to explain the problem of design flaws in component integration. Moreover, the problem statement also explains the main issue in this domain to explain the extent of the research gap in service oriented business logic security.

1.2.3 Research Contribution

The research will propose a security assurance methodology for service-component-oriented business logic while reusing core service logic. The research will bridge the difference between conventional perspectives and security requirements of e-process in component-based banking systems in the context of social e-commerce. Adopting such practice will enhance security assurance embedded in the design of service component-oriented applications within the e-commerce domain, by using currently available components through the deployment of business logic into service-based systems. The second point of research contribution is to identify and propose a new classification based on the taxonomic structure of logical vulnerabilities in service component-based service in the middle-tier, which often cause design flaws due to COTS or in-house software components for service integration. The third point of contribution is to offer attack modeling in the scenario of a logical attack through event-based attack trigger component detection. The fourth contribution is to use the Sec UML modeling technique to test the methodology outlined in

Paper 5. The fifth contribution is to be further reflected as a security feature based on UML Sec modeling, exemplified by a B2c ATM model demonstrated in the context of social interaction during e-commerce software security modeling in Paper 6.

1.2.4 Structure of the thesis

Chapter 1 includes the research introduction and literature review, in the context of the analysis, the problem statement, the research gap that the thesis discusses, the aims and questions for research, the scope of research and the research contribution.

Chapter 2 is related to the literature review which is connected with the upcoming chapter in the sequence of this thesis order that has been demonstrated in the flowchart at the of this thesis. The purpose of this chapter is to highlight the main issue in-service component architecture-based rapid development application logic and related loopholes. Therefore this chapter provides ACS-based event attack connectivity with application logic.

Chapter 3 is a book review article that provides details of service component architecture (SCA) based application logic design and event-based attack in-service composite applications. This goal is accomplished by investigating and evaluating security concerns, and modeling techniques in the application of service modules, when applications create, consume and process a particular event in the application logic.

Chapter 4 is a research article that details a group-attacking method and classifies two groups of vulnerability (Technical vs Logical) in e-commerce component-based-application. This chapter consists of two sections - parts A and B, and this Paper validates the proposed model in Part A (publication).

Chapter 5 is a research article that addresses event-attack modeling in the scenario of a banking application domain. The proposed approach is based on an event-attack modeling technique that uses the Uppaal Tool to detect design flaws in e-commerce component-based applications while reusing the design specification of an existing application logic of the system.

Chapter 6 is a Paper, a research article providing a detailed case study based on problems and solutions in the context of social commerce. This is done through the use of a case study, which is a tool that allows us to reach our specialized targeted audience, and the banking case study and proposed methodology is tested through a modeling technique related to the application logic.

Chapter 7 is a research article that is designed in the context of a research-based paper that targets the modeling technique by using and incorporating security-modeling features into component service architecture in relation to expanding the research work in Paper 5. This will be further reflected as a part of security feature-based UML Sec modeling for an example B2c ATM model demonstrated in the context of social interaction of e-commerce software security modeling that justifies the secure application logic.

Chapter 8 provides the discussion and conclusion of the thesis and sets out the research findings.

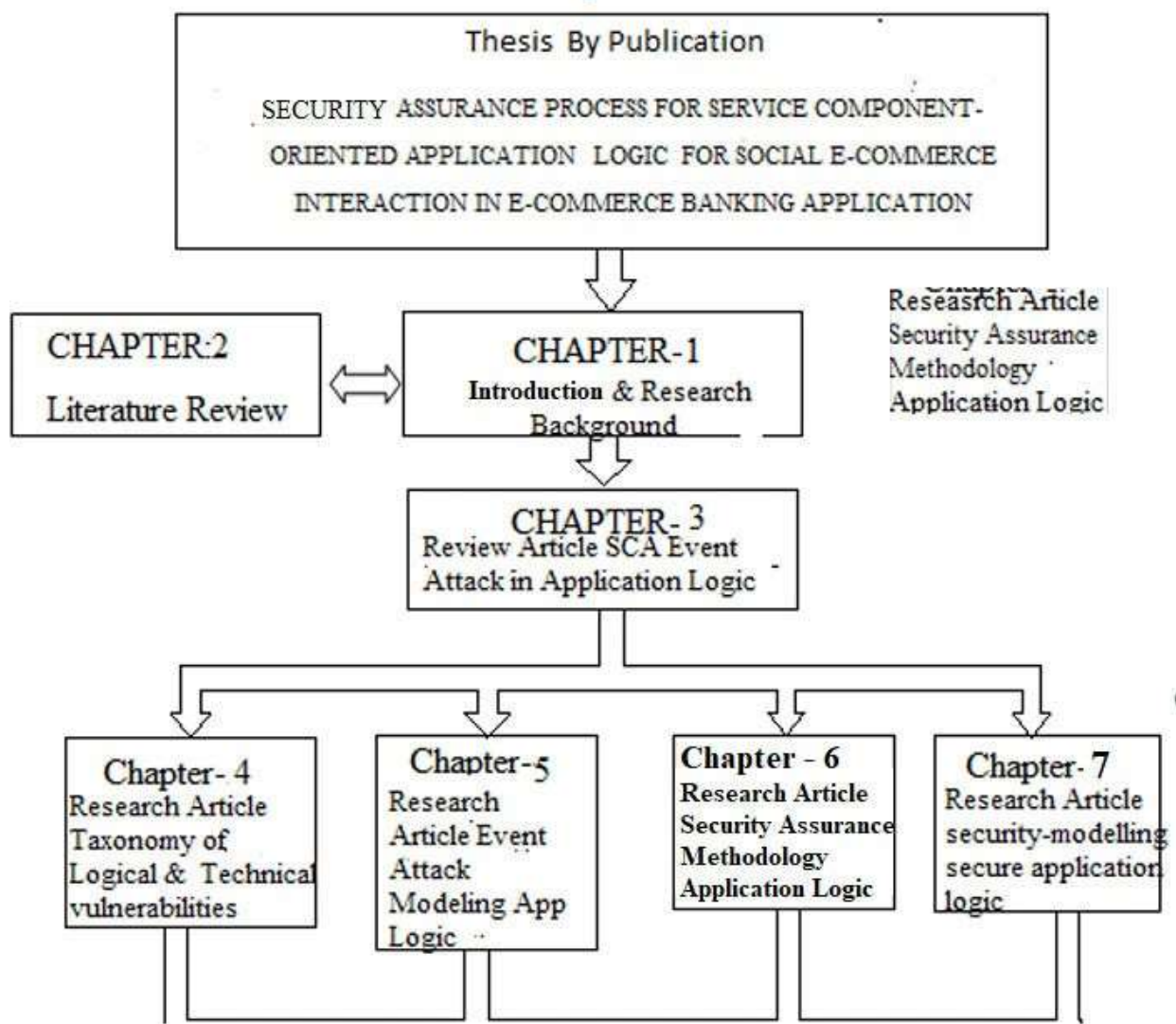


Figure 1.5 Flowchart of Thesis by Publication

CHAPTER 2: LITERATURE REVIEW AND PROBLEM ANALYSIS

In social e-commerce (a subset of e-commerce), security and privacy issues are important topics for discussion between users. E-commerce capability is one requirement of the business model of information system architecture, and its use has become increasingly prevalent. Users may, however, find themselves somewhat reluctant to engage, due to security and privacy risks. Social e-commerce has prompted a new era of information security in the banking industry (Nabi et al., 2021). Business logic development, which reuses component design specifications, is a real concern in e-commerce applications in this domain because it is mostly based on service component architecture (SCA) (Paik et al., 2017; Nabi et al., 2019; Nabi et al., 2020).

There is no clear difference between service component architecture and component-based software architecture in terms of their implementation. They both contain enhanced components, in the sense that single components have service capability, and are connected to develop new business logic (Agirre et al., 2012).

In service-component-oriented systems, the business process layer is less secure because the focus is on high-level abstraction security. This gives intruders the chance to bypass security checks in vulnerable business application logic. The service-component-oriented application contains at the middle-tier layer two building blocks: business logic & process logic (Nabi et al., 2019).

A business application is decomposed into a number of services, each of which wraps a reusable business components function and has a well-defined interface that specifies how the service may be used to perform an operation (Paik et al. 2017).

The term “business logic” refers to a particular “service”. This service can be a withdraw-payment service. The service is defined by the business component class “withdraw-payment”, which is handled through the component class business logic. Each component has business logic offered by a business component that resides within the business domain. A logical component can be defined as a sub-component or part of a sub-system; in both cases, the component keeps its own integrity. Certain steps need to be followed in order to perform a predefined action by application logic to operate the business process (Malohlava et al., 2013; Nabi et al., 2020).

The logical component-ware supports the process of application logic and of each component's business logic to develop a set of functionalities, which are then further translated into the component's business processing logic when these components are integrated into the n-tier architecture (Jiang & Willey, 2005; Agirre et al., 2012; Nabi & Nabi, 2017).

The service component architecture is a way or method by which software can be designed, in which services are provided to the components by application components. A web-based banking system is basically constructed /developed using two sorts of components at the business logic layer of the application. These two kinds of components are (1) business processing components, and (2) business entity components. The first category of components deals with a service requested by end users through the published user interface (Agirre et al., 2012; Nabi & Nabi, 2017).

They determine which function of business entity components will be called or invoked and operated. They are persistent components that keep their state stored by the application and are part of the application domain (Rodriguez et al. 2016). The coupling influences interoperability because integration impacts distributed functionality (Kalantari et al., 2013).

The research explains that recent models of web-based banking systems critically lack security properties, such as logical security. The key reasons for this are poor design and the threat of highly skilled intruders.

Recent security reference models (such as IBM's service-oriented architecture 2021) also fail to provide comprehensive security to business processes at the layer of business application logic, where services are developed through the business components (each component contains core service logic). This security model only provides authentication before gaining access to business service components, which normally occurs through the HTTP web server. Current security techniques used include model authentication, sanitization, database encryption, (IDS) and (IPS).

Service-oriented component software uses two sorts of components to rapidly develop

application logic. These two components are custom-developed and commercial of the shelf (COTS) (Jakoubi et al. 2011). It is possible that they may have flaws within the design or software application. The CBS-based software solutions can lead to logical vulnerabilities such as component's logic subversion attack, misuse of application logic and circumvention of the component's business logic flow, all of which temper the application's functionality (Nabi et al., 2020).

There are three categories of software practice vulnerability in service-component-oriented web-based banking applications (Woody, 2015; Novak & Švogor, 2016):

- (1) Flaw in design
- (2) Coding-based fault or weakness
- (3) Integration Faults of components.

Our research considers the operational vulnerabilities (middle tier) of a service-component-oriented banking application, based on the composition of components, rather than on traditional techniques of software development (Nabi et al., 2021). As we have noted, within the categories of operational vulnerability, the focus of consideration is business application logic vulnerability (which may cause design-based weaknesses), and integration faults which often circumvent a component's business logic flow operation (Nabi et al., 2020).

Unfortunately, a single flaw within sophisticated middleware can allow an intruder to bypass a strong security authentication scheme. However, we have noticed that most front-end & back-end systems contain COTS packages. Therefore, it is necessary to custom-develop software packages in middle-ware, in order to design secure business application logic (Nabi, 2005; Bentounsi et al., 2016; Nabi & Nabi, 2017). The custom developed and commercial of the shelf components example is presented in java EJB session bean and entity bean components, which are mostly made of Microsoft and Sun Java technology these, are mostly the middle tier consisting of business logic. However, such component packages are beads on the functional logic of each component that develops business logic.

The above-mentioned Diagram 2.1 depicts and illustrates the process of the functionality of java bean in business processes, as stated above in the introduction,

(Figure 1.3 b).

In recent years, we have observed that the weakest link in the server-side systems is middle-ware. A secure component-based platform regulates programme execution and manages events, while the security of interactions between application components must also be ensured. A component-based approach that addresses a variety of information system application requirements also supports the entire lifecycle of such applications, including the design, and execution phases, with a focus on their security and safety requirements, which the application integrator can define during the design phase (Nabi et al., 2019; Raed & Nripendra, 2020).

Recent innovations in the field of e-Commerce-based social media software technologies, according to Nabi et al. (2021), have offered many benefits; however, design processes often lead to a variety of challenges, from the design phase to the implementation phase. Software flaws and faults exacerbate reliability and security issues.

In component-based banking applications, banks compete by using social e-commerce-based e-banking to boost customer loyalty, gain market share, improve services and offer value-added services, increase efficiency, and cut expenses. At the same time, they must deal with security and privacy concerns relating to consumer relationships (Laukkanen et al., 2018).

When connecting with component-based banking applications, for these services security considerations are sometimes overlooked, potentially allowing unwanted access to the service. Another factor to consider is that the re-use of service logic can occasionally result in security failure (Ghassan et al., 2020).

Over here, it is important to discuss the function of components in the application logic forming state that refers to the re-use of existing logic of any component within the application in the banking domain and its security issues, especially in the context of social e-commerce-based banking.

The current development method in CBSE is a software development process that advocates building software systems from existing software components rather than creating them from the ground up. CBSE's mission is to reduce costs while delivering higher quality, systems that are more reliable. "A unit of composition with

contractually established interfaces and explicit context dependencies only,” is called the software component (Alrubae et al., 2020).

A component model in CBD defines a set of programmes called component and composition mechanisms that combine smaller components to form larger composite components. The composing mechanism determines the behaviour and structure of this larger unit. If the composition process is algebraic, that is, the composition of two or more components, a larger system can be constructed (Chicote et al., 2018).

To define a component in containment, at least two components must be combined for the composite. A composite is defined by combining the behaviour of two or more existing components (Chicote et al., 2018).

The interactions between two components are defined by a connection mechanism; a connection can be used to pass messages directly or indirectly. A third coordinated component is used to construct a composite of two components (Rana & Baz, 2020).

Component-Based Software Development (CBSD) tries to encourage software reuse in order to drastically save development time and costs. Existing solutions are wrapped in well-defined components with clear (needed and given) interfaces that allow them to connect to and interact with one another. When putting together a system from components, it is important to consider both functional and non-functional factors (Rana & Baz., 2020; Chicote et al., 2018).

Timing, dependability, safety, and resource consumption are examples of non-functional qualities. Despite their relevance, only a few component models explicitly support non-functional property specification and administration throughout the development process (Chicote et al., 2018).

In most circumstances, this assistance is limited, and unlike the well-established approach of incorporating functional qualities into interfaces, no consensus has arisen on how to manage non-functional attributes at the component and system levels (Rana & Baz., 2020).

One of the most significant characteristics of a software component is that it can be considered a sub-system that can be directly installed and run. To do this, a component must adhere to the standards of a component model and meet the requirements of a specification. Commonly acknowledged standards to explain composition and interaction are required to enable composition between separately generated components to develop application logic (Rana & Baz., 2020; Lau & Cola, 2017).

The effort required for design and execution is minimised when existing components are used in CBSE. Additional processes, such as component assessment and adaptation, are introduced to the development process. Searching for candidate components that may satisfy needed characteristics, selecting the most acceptable components from these candidates, and confirming their functionality are all part of the component evaluation (Johan & Mishra, 2019).

In software engineering, the reuse of existing work is greatly desired to save development costs and achieve high-quality software. CBD develops basic and composite components for reuse, and reuse creates composite components and systems (Johan & Mishra, 2019).

Interactions between components in a CBD system are an essential factor that can aid in the discovery of new ways to combine components. CBD-based development, which uses and reuses secure components, requires the most up-to-date development approaches for CPS building. These systems may be made up of a variety of soft and hard components that are dispersed throughout the system (Alrubae et al., 2020).

Due to the rising scale and complexity of component-based systems, developing high-quality, dependable and on-time secure software systems is difficult. Because traditional software development methodologies are not up to the task, numerous alternatives have been proposed to boost productivity and reusability during the software development process and security, but still lack automated security at the design phase. This targets the gap in security research techniques that follows a methodology for secure designing of component-based applications in the banking sector. That is free from design flaws (Johan & Mishra, 2019).

The main problem in this context is referred as a design flaw in component integration, which mostly consists of COTS and custom-made components. A real example of such binding is based on the interconnection of business components. Therefore, as a solution to this problem, service-oriented component integration places emphasis systems through the use of reusable components. The main problem of components integration is created in this architecture to solve a specific business problem. Java business component integration example is presented given below.

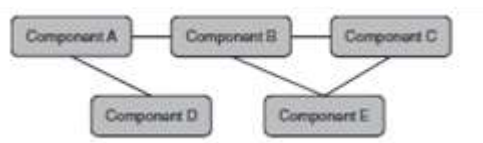


Figure 2.1 (a) reusable component integration pattern

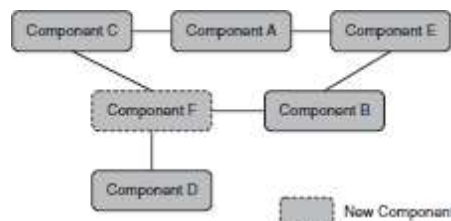


Figure 2.1 (b) reusable components are binding in service integration

They provide some generic functionality. These components can then be threaded, linked, or integrated into a specific order or configuration to meet a specific business need. There is no need to create a new computer programme if the business requirement changes. Instead, the system can be reconfigured to meet the new business requirement.

The functionality of software systems is heavily reliant on software components. The current and reusable components of a software system that has been previously debugged, validated, and rehearsed are referred to as software components. Using such components in a newly designed software system can save time, effort, and a lot of money. Security is on the rise as a result of the habit of using components in new projects. For the researchers to understand the approaches, they must first learn about the existing approaches and techniques employed for security purposes to address this problem (Migault et al., 2017).

Since no taxonomy is developed related to the business logic flow in component integration, it is represented in Chapter 4 as a publication, which gives a comprehensive contribution to the validity of the design flow problem in application business logic.

2.1: Current banking CBS-based system analysis

The current banking industry is getting benefits from this medium by using social e-commerce component-based applications. On the other hand, the re-use of component business logic is always a concern in terms of its capacity to cause logical vulnerability in such systems.

The design flaw, which causes application logic vulnerability in such systems always attracts the attention of intruders. The main reason for this is that application logic design flaws cannot be automated through traditional approaches (Nabi et al., 2020; Raed & Nripendra, 2020; Jürjens et a.2019: Rotella, 2018).

Any component-based software system's design must be well thought out and certain aspects must be taken into account from the start. There is a better chance of the system being successful if specific quality attributes of component specification match with the system application logic based on business process and designed into the architecture. There are several important characteristics to consider when designing a bank system. First and foremost, the system's performance must be excellent. Bank employees, ATM users, and bank administrators will all interact with other systems, so the new system must be fast enough to allow everyone to complete their tasks. Furthermore, the system's dependability and security are two of the most important considerations. As a result, the system must be dependable. According to NIST USA, 2021, the recent cases that have emerged about this issue that is being faced by the banking industry, while getting benefits from social interaction in component-based banking applications (NIST, 2021 <https://csrc.nist.gov/Projects/ssdf>).

Therefore, component-based software security, especially in the case of reuse design specification, while reusing existing application logic. On the other hand, there is a serious need for such a methodology that can tackle these issues, while considering

social e-commerce in the context of a sub-set of e-commerce component-based applications (Nabi et al., 2020; Raed & Nripendra, 2020)

Therefore, it is necessary to plan for strong risk management that particularly provides in-depth software security assurance, which also deals with middle-ware software. Software security problems start from design flaws, which cause system integration faults. Such flaws allow security bypass by the user when the software maliciously or accidentally gives system access that the programme does not permit (Nabi & Nabi, 2017; Ghassan et al., 2020; Nabi et al., 2021).

Security and privacy issues in the field of social media-based e-Commerce are important topics for debate among the users concerned. E-Commerce is one part of the information system architecture business model and its use has become increasingly common. However, the users may find.

The applications themselves are somewhat unwilling to suffer from risks to their security and privacy. In the banking industry, social media-based e-Commerce has prompted a new age of information security. However, the risks associated with these issues make e-commerce banking difficult. There is also no customer trust, and no visitor shops on the website and these sites will not function unless these privacy and security risks are removed. Security and privacy have social, organisational, technological, and economic implications (Raed and Nripendra 20202020: Wang et al. 2020).

Service-oriented applications in the social e-banking domain are developed with well-defined, readily available software components. These are the building blocks used to develop the services in service component architecture. It is important for the design process of service-oriented applications in the social e-banking domain to follow an order where once the design process of an application is determined as an organizational task, and then it is considered as modular components that need to be integrated. Each modular component is a service that explains its interfaces by incorporating these services into a social e-banking system.

New business processes rely on the integration of modular components that have been

identified (Nabi et al. 2020).

Researchers and developers can clearly understand that system composition and integration are the most difficult processes in software design, especially when devising complex operations such as the integration process into the social e-banking system. The complications that arise during the integration of these e-banking systems may result in a process of continuous change in the technical and business attributes provided by the bank to its customers in order to] meet their needs (Nabi et al. 2019). Social e-banking systems are frequently designed and built on a variety of component vendors with varying platforms and design/development capabilities patterns in architecture. Ongoing changes can introduce additional complexity when integrating banking applications based on service-oriented components, as design flaws in the integration process can open the door to business logic attacks, and cause severe financial damage (Nabi et al. 2020).

Therefore, there is a clear need for a methodology that is able to deal with the logical flaws that do not show attack signature and patterns and are therefore hard to discover using traditional techniques and security software tools. A modeling methodology is therefore needed that helps to generate and automate the vulnerability through attack scenario modeling (UML Sec & Uppaal Tool). This will help to deal with component design specification-based business logic and identify any design flaw that may cause a security breach in banking applications while the re-use of component business logic and avoid duplication.

CHAPTER 3: A SECURITY REVIEW OF EVENT-BASED APPLICATION FUNCTION AND SERVICE COMPONENT ARCHITECTURE

Introduction and Findings: Logical Relationship between Chapter 2 and Chapter 3

Introduction to Chapter 3

Chapter 2 features an introduction to the research and its background in the context of the analysis that follows, the problem statement, the literature gap that the thesis discusses, the aims and questions for research, and the scope of research, targetting the main question posed by the thesis. Chapter 3 covers and provides details of (SCA) service component architecture-based application logic design and event-based attacks on in-service composite applications. The purpose of this chapter is accomplished by evaluating, investigating security concerns and modeling techniques in the application of service modules when applications create, consume and process a particular event in application logic. This chapter fits logically into the thesis, because it provides the base for further research chapters that support this thesis. It explains the (SCA) service component architecture-based application logic design and event-based attack in-service composite applications.

Findings:

This chapter's findings are related to previously conducted work in component-based software rapid development that considers the security concern which put the foundation to the above title chapter and provides connectivity in terms of event-based attack modeling as a technique that is explained in the next chapter.

This paper is published in the International Journal of Systems and Software Security and Protection Volume 11 • Issue 2 • July-December 2020 IGI.

A Security Review of Event-Based Application Function and Service Component Architecture

Faisal Nabi, University of Southern Queensland, Australia

Jianming Yong, University of Southern Queensland, Australia

Xiaohui Tao, University of Southern Queensland, Australia

ABSTRACT

The term service component is derived from SCA (service component architecture) for event based distributed system design. Although service component pattern offers composite application development and support application reusability functionality. However, security in event based communication in components interaction model mostly discussed on upper layer in SCA while developing service oriented component application logic. This layer is called application business process logic layer, which produces the application's rendering logic, having being authenticated from ACL. The need for such a comprehensive security review is required in this field that could possibly elaborate the issues in composite application and Event based attack in service component architecture model. The paper achieves this target by analysing, reviewing the security issues, modelling techniques in service component application functionality, while application components, that produces, consume, and processing events.

KEYWORD

Business Processing Logic Layer, Event Attack, Reusability Component, Security, Service Component Architecture

1. INTRODUCTION

The Architecture of Service Components (SCA) framework offers a component-based model with a consistency, design and efficiency approach to loose coupling, (Service Component Architecture. <https://www.osoa.org/display/Main/Home>). A SCA part has two types of interfaces, interfaces supported and demanded. These are used for the incorporation of the service into other components and inter-service event based communication.

Component construction takes place by service interface and reference wiring. Design (Development of Individual Components), Structure (Composition of Components into Systems) and Assembly (Structure of Composite Services or Service Networks) are the key elements of SCA that provide design stability to shape structure of components and service networks.

The event-based communication model is being used more and more commonly for the development of loosely connected, distributed systems for many different industry domains, such as

DOI: 10.4018/IJSSSP.2020070104

Copyright © 2020, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

composite applications based on SCA. The Event-based systems range from distributed sensor-based systems to Comprehensive business information services, OASIS Service Component Architecture / Assembly, SCA-Assembly (2018). Compared to synchronous communication using remote procedure calls (RPC), for example, event-based communication between components offers many advantages such as high scalability and extensibility, OASIS Service Component Architecture / Assembly, SCA-Assembly (2018). Being asynchronous in nature, it allows a send-and-forget approach, i.e., a component that sends a message can continue its execution without waiting for the recipient to respond to it. In addition, the loose coupling of components achieved through the mediating middleware framework leads to increased system modularity as components can be quickly added, removed or replaced.

The development of event-based system (EBS) has become one of the popular method in terms of service component architecture, there are number of reason such as the high quality pliability, scalability and quality to being able to adjust properties of new condition. The communication system makes such advantages— implied invocation and inferred competition between components. The event management is non-determinism on the base of coordination structure in event management that is possibly cause to produce relatively inborn vulnerabilities in a process of event attack.

In composite application functionality, the Event Attacks are mostly some different type of attacks, which by manipulating the event-based communication model of the system. This can misuse, trigger and affect a target model. The Event Attacks are harder to prevent because they are treated in a way that is not different from typical normal conditions in event-based communication.

There is extensive use of event-based systems that are introduced utilise the MOM frameworks. Various types of MOM frameworks including Prism-MW, Java Message System, Java Message Service (JMS), (2016), introduce these and Carzaniga, Rosenblum, & Wolf, (2001), in applications such as web based applications or service oriented architecture-driven systems. EBSs have become popular because of its high versatility, scalability and adaptability. Such benefits are allowed by relying on component call by invoking implicitly and implied competition. In a particularly case specifically, components in event-based systems possibly not be aware of the events they publish by customers or they may not necessarily know producers.

In service component based composite application the communication method, however, it is consist on non-deterministic in the handling of events, which may introduce inherent vulnerabilities in a system called event attacks. For instance, developers can create EBSs using externally developing malicious code components and users can use malicious code component EBSs. In those instances, malicious components may cause unintended behaviour, such as by sweeping events in order to obtain unauthorized information or by manipulating data in events to compromise the functionality of the event-based system.

1.1. Motivation

Existing system analytics focused on the service component do not concentrate on event attacks or correctly identify vulnerabilities component by component, explained by SonarQube, (2017). OWASP _Orizon_Project, (2017) and Xanitizer, (2017).

Li, Bartel, Bissyandé, Klein, Traon, Arzt, Rasthofer, Bodden, Outeau and Mcdaniel (2015) showed that in specific, current system flow, analysis methods do not specifically call for components or are not feasible in order to evaluate vast quantities of components in structure. It is further investigated by the researchers that vulnerabilities that uncover Android applications to event attacks are not specifically applicable to other Event based system styles since Android is using system-specific models, such as APIs, and life-cycle components (Lu, Li, Wu, Lee & Jiang, 2012, p. 231).

The Research needs for such a comprehensive security review that could possibly elaborate the issues in composite application and service component architecture model. This can be derived through the reviewing security issues of service component application functional and reusability modelling techniques. Those are used in system design from existing application components, that produces, consume and processing events.

1.2. Problem Statement

In this research, it is found that current security efforts do not concentrate on event attacks or correctly detect component-by-component inter-communication event model. This is the required approach simulation and modelling technique that is needed for composite application and service component architecture model.

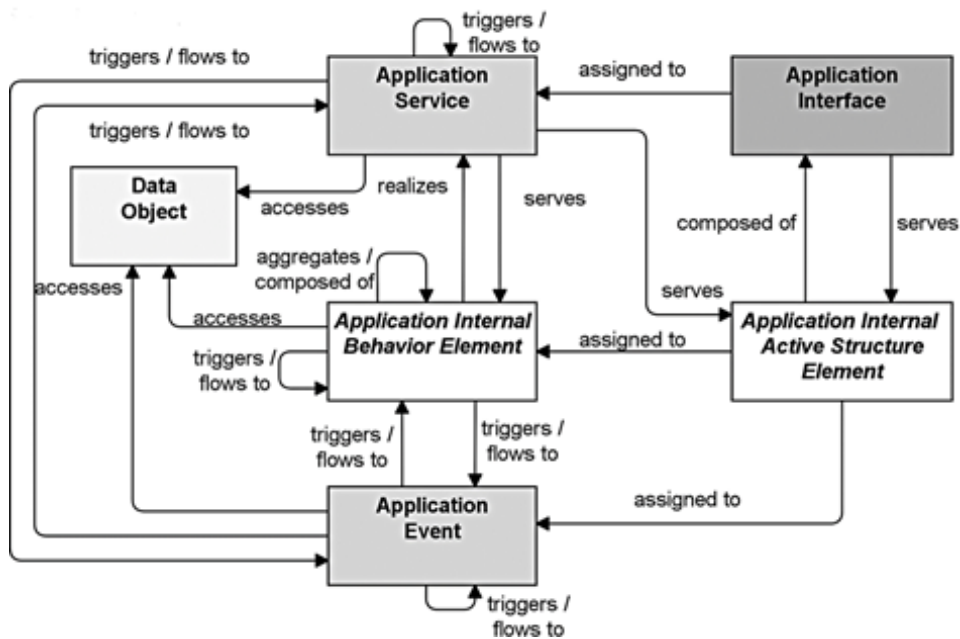
1.3. Research Method

The research consists on the empirical review of security methods in Event-based application functionality. This method analysis and review comprehensive security issue in the service component architecture, which is based on existing methods, to examine the contribution of suitable approach for event based attack solution through the three modelling techniques.

2. RESEARCH BACKGROUND

There is no clear difference between service component architecture vs component-based software architecture because by rule of implementation, it is enhancement of Components, where single components are represented as service, which is connected to develop new business logic (Agirre, Marcos & Estevez, 2012, p. 19). In Event based service oriented systems, business process layer is more unsecure because the focus is on high level abstraction security, this is reason why intruders get chance to bypass security checks and vulnerable business application logic. The service component Application contains at middle-tier layer two building blocks, Business Logic & process logic, which produces Events to call components inter-communication process. The given below Figure 1 explains the process of application service function through the interface that serve in the case of a trigger/flow to assign internal behaviour, in the result of this triggered flow the application Event is generated that correspond the component application logic as output.

Figure 1. Events call application inter-communication process



The term define business logic is to refer a particular “service”, that is generated through an Event process this service can be withdraw payment component, in which an Event call business process. This is defined by the business component class withdraw payment, that is handled through component class business logic. Each component has business logic offered by business component that is resided in business domain. The concept of logical component can be defined as sub-component or part of sub-system, in both conditions component keeps its own right. There are some certain steps need to be followed or required in order to perform a predefined action by application logic to operate business process (Nabi, 2011, p. 32). The logical component-ware supports the process of application logic and each component’s business logic to develop set of functionalities, which then further translate it into component’s Event processing logic by integrating these components in the n-tier architecture (Nabi, 2011, p. 32), (Nabi & Nabi, 2017).

The service component architecture (SCA) is a way or method by using that a software can be designed, in which services are provided to the components by application components. Web based e-commerce systems are constructed /developed using two sort of components at business logic layer in the composite application. These two kind of components are Business Processing components and Business Entity Components. Rodríguez, Zalama and González (2016) demonstrated that the first category of components deals with service that is requested by end-users through the published user-interface. They decide the function of business entity components, which definitely will be called or invoked and operated through inter-component Event based process. They are persistent components, who keep their state stored by the application and part of composite application domain.

2.1. Service Component Architecture

Service component architecture offers a programming model, based on a service component model, for building applications and solutions. It is based on the idea that business operation is delivered as a set of resources, which are combined to create solutions that meet a specific business need. Such hybrid programmes can include both new services created specifically for the product as well as business function from existing systems and replicated software as part of the synthesis. SCA includes a blueprint for both programme design and service feature development, including reuse of existing application function within SCA composites. (Memon, Hafner, & Brey, 2013) and OASIS Service Component Architecture / Assembly (SCA-Assembly 2018).

Therefore, SCA also offers a blueprint to coordinate modules that produce and consume Events and process the Events.

The SCA assembly model consists of a set of objects that characterise the structure of an SCA environment in terms of composites comprising service component assemblies and artefacts connected to the interface explain how they are linked together.

The SCA model contains following properties:

- In service component assembly model the services are, both tightly and loosely coupled;
- A model for the deployment of infrastructure capabilities to service communication, including security and transactions;
- A pattern for Event management including Published and Subscribed – a particular style of grouping components that produce and consume Events in which the processing component is decoupled from the consuming components.

2.2. Specification of the Service Component

Component interactions may also be defined by one component and use by another one service, one component provides for one service (function or computation) to the other, services requested from the other component and interactions details on which these provisions and requests are rendered can be described as Component Service Specifications. The assumptions that a provider component has on

its contractual obligations to use its services, as well as the preconditions or contingent specifications that it intends such components to enforce upon it, are explicitly stated in the service specification. (Denney & Fischer, 2009).

2.3. Security Properties in SCA

In order to be considered reliable, Service component based software must have three properties:

1. **Dependability:** Dependable software performs reliably and functions correctly in all conditions, even hostile conditions, such as when the software is attacked or runs on a malicious host;
2. **Trustworthiness:** Trustworthy software incorporates few, if any, vulnerabilities or flaws which may be deliberately explicit. (Kung-Kiu & Ukis, 2007).

Therefore, to be called trustworthy, the programme must not include any malicious reasoning that allows it to act in a malicious manner;

3. **Survivability (also referred to as “Resilience”):** Survivable — or resilient — software is software that is adequately robust to (1) either withstand (i.e., secure itself against) or accept (i.e., continue to function efficiently despite) the plurality of known attacks.

3. RELATED WORK

In EBSs, defence was based on various approaches (Aniello, Baldoni, Ciccotelli, Luna, Frontali, & Querzoni, 2014), (Petroni, Querzoni, Beraldi & Paolucci, 2016), (Shand, Pietzuch, Papagiannis, Moody, Migliavacca, Eyers & Bacon, 2011), (Srivatsa, Liu & Iyengar, 2011) and (Xenitellis, 2002, p. 149). Simeon et al have studied and identified the vulnerabilities in EBS that cause of security concerns. Systems based on events, typically, existing solutions for EBS security use encryption, static code detection and/or executive access evaluation. Encryption is not only a common software system but also EBS systems used widely for securing Events. Srivatsa, Liu and Iyengar (2011) mentioned that Guard provides solution that states the computes encryption systems in a way that each component encrypts events through network of event broker.

The process of sign and encrypt events is made via random token, whereas signature itself is authenticated and added to the event with a unique subject address. Encryption strategies however decrease the risk of keys to be stolen and can contribute to unacceptable overhead efficiency. In addition, when the component of the system is not determined, it is appropriate to provide a key distribution.

Static code analysis is a popular method for analysing security defects of the target system. Reimer, Schonberg, Srinivas, Srinivasan, Alpern, Johnson, Kershenbaum and Koved (2004) stated that SABER is a tool for static analytics that identifies recurring design errors based on instantiation of error pattern templates. (Tripp, Pistoia, Cousot, Cousot & Guarnieri, 2013) stated that Andromeda reviews the propagation of data-flow on demand by Java, NET and JavaScript support programmes. Xanitizer (2017) uses software to defects, like spikes and privacy leaks, to automatically detect them. Owasp Orizon is a computer code detector that uses a template to find bugs for J2EE web-based applications. In order to identify software vulnerabilities Sonar qube(2017) is an open-source code quality control tool. Another popular method for protecting EBSs is runtime access management. Wun and Jacobsen (2007) have suggested the regulation model and mechanism for content-based publishing / subscription schemes.

Pietzuch (2011) mentioned that DEFCon is a software functionality that uses a flow control model to monitor the dynamic, heterogeneous event processing system and prevent unintended event flows, which may violate security measures. Nonetheless, the above approaches rely more on security issues other than incident attacks. In fact, because those approaches do not help event-based

interactions structures in their entirety, the study of large web apps with a number of components may have unreliable and scaling issues.

4. EVENT ATTACK IN SERVICE COMPONENT AND COMPOSITE APPLICATION

Event attacks are the security issues, which exploit the model of event-based communication. So far, the work has reported the following types of attacks. (Lee, Nam, & Medvidovic, 2019):

- **Spoofing:** An event that spoofs a target component to manipulate the features or data of that object may be sent by the malicious component;
- **Interception:** A malicious component may intercept an event to be sent by other components and may return inappropriate messages;
- **Eavesdropping:** An incident with sensitive data that should be available only to specific components may be evoked in a malicious component;
- **Confused Deputy:** A Malicious components can indirectly access a target component through access to the other components that can have access to the target event consumer;
- **Collusion:** More than two malicious components may be merged to take advantage of targets component functionalities or services.

4.1. Security Features of Event-Based Systems

There are four Types of Event based communication security features that focus on security challenges in service component applications:

- **Event Communication Channel Analysis:** The Implicit referencing and vague interfaces in EBS obstruct the extraction of channels of event communication through which events are shared between components. In particular, it is difficult to determine where each occurrence flows into an implicit request, and ambiguous interfaces are difficult to identify explicitly. Since each MOM system offers different types of event interfaces, a comprehensive review is needed to identify the channels of events;
- **Extendable Flow Analysis:** Analyses of control data flows on each variable are necessary to check if sensitive data leak or accidental access to sensitive operations could occur. Nonetheless, flow analysis on each component may not be scalable for EBSs consisting of a large number of components with a number of methods. According to (Safi, Shahbazian, Halfond & Medvidovic, 2015) total EBSs on average include more than 35 methods to be evaluated and studied for a few hours. Although several methods of flow analysis for Android apps have been suggested, given that mobile platforms restrict apps (<https://developer.android.com/google/play/expansion-files.html>, 2017), EBS may be larger than traditional mobile apps in size and complexity;
- **Irregular Sensitive APIs:** The study of insecure flows relies on a given set of APIs that can handle private data or sensitive features. Although prior work has defined a set of sensitive Android APIs based on the supervised machine-learning approach (Lee, Nam, & Medvidovic, 2019), the set is not equally valid in other EBS domains as each framework can use different types of APIs;
- **Irregular Trusted Boundaries:** The trust boundary between components is defined according to the trust level of each component (i.e. all components of the same trust boundary have the same trust level) and event attacks take place through different confidence and trust, (Lee, Nam, & Medvidovic, 2019). Many EBSs may be using a constant type of trust boundary (i.e., Android application), but depending on the system configuration, EBSs may have types of trust boundaries.

5. A REVIEW OF EVENT BASED SECURITY MODELING

Therefore, keeping in view the above-mentioned scenario the research review presents three modelling techniques to analyse the best solution that deal with Event Attack Modeling. The first approach is Event-Based Control Modelling technique. This model helps to control the Event based attack in service component designed composite applications, which follows security by design technique.

Moreover, also discuss the second approach of security component composite in service component architecture. This illustrative the layer based security for authentication, authorization and auditing security component services to composite applications.

It is further, reviewed with most recent third research approach in the field of Event Attack modelling related to business logic subversion life cycle, due to flaw in component-based software integration oriented business process layer.

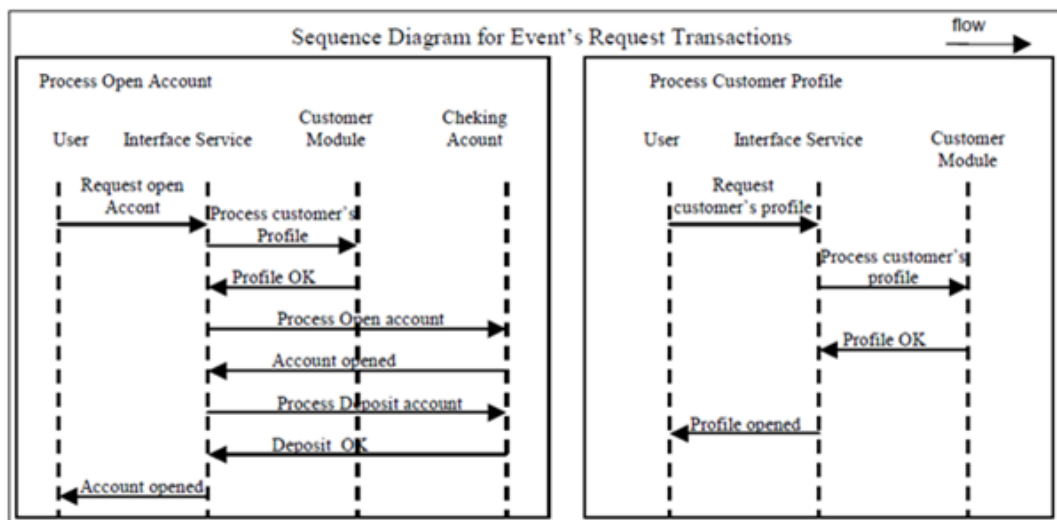
5.1. Security by Design Event Control Modelling

Over the last decade, event-triggered-control in real-time systems has been gaining increased attention. One of the most striking features is that Event-Triggered Control provides a strategy for exercising control only if required. Compared to traditional time-control systems ETC can cut the number of control tasks effectively while retaining the optimal closed loop output.

A simulation technique based on the finite automate notation can be compared to the Event Control System modelling with UML as shown in Figure 1, suitable to integrate certain logic forms that require potential transformations between different states. Every occurrence is the defining in time and space for an important event. The necessity receives it for a small number of entries, triggering a set of elementary measures and producing a state change as a result, can be called an invitation to a programme or task module (Bastos & Castro, 2008).

Figure 2 demonstrates a bank system interaction through the 'Demand Open Account' event. The programme returns the result "opened account" after a series of business transactions has been processed. Customer Profile Transactions can be triggered by the "Demand Open Account" event with two other business transactions of the account control module: the "Process Open Account" and the "Process Deposit Account." Notice that the module of the customer's business transaction

Figure 2. Event request transaction diagram

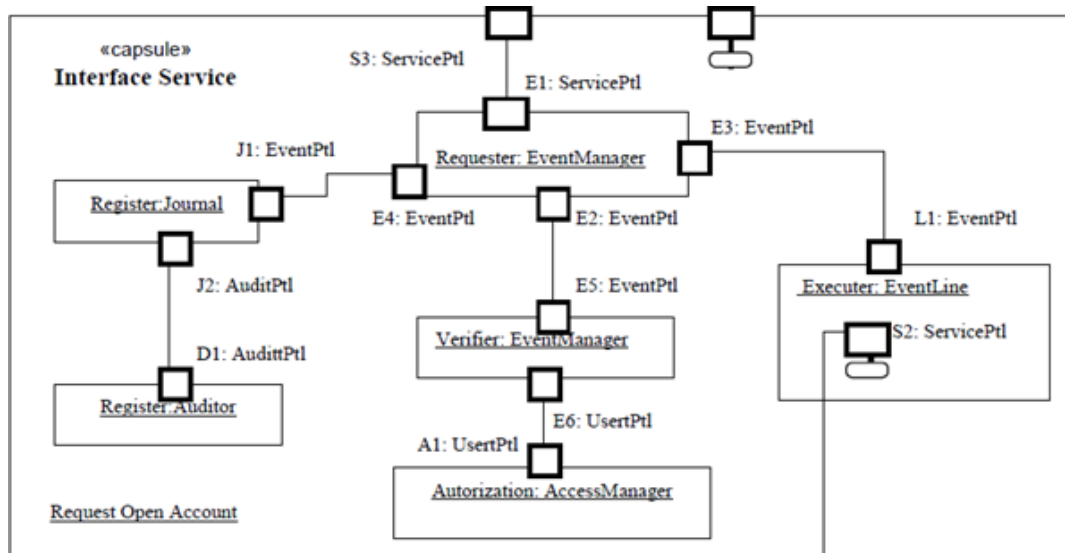


“Profile of the customer operation” can be triggered by the event “Profile of the customer request” or the event “Open account request.”

The Event Control technique involves the static and dynamic features of a programme, such as the Event reflecting a financial movement; the Item deciding the collection of products of the business (e.g., checking account, banking account, local properties, or investment); the user representing the person ordering or triggering the Event (e.g., a client or a bank branch). The Business Principle used to determine behaviour patterns for each movement of events; and the Business Transactions that are commodity transactions, identified as system procedures, carried out whenever an event is discharged.

Hence, the productivity of this model enhance the security by design technique and component service oriented feature for reuse of component based application functionality. Figure 3 clearly illustrate the business event functional communication among the inter-component event produce and consume such a processing events.

Figure 3. Event based business functional process



It is derived the modelling example as shown above that Event Control Modelling technique promotes the service component oriented application functionality and reusability of service components while system design from existing application components. The security by design technique improves the idea of reusability of service component and inter-component event-based communication process model. This model enhance the security engineering at design stage to maintenance of service component oriented system.

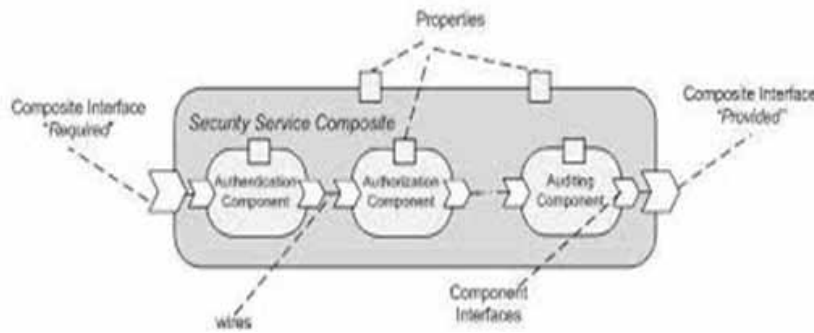
5.2. Security Component Based Modeling SCA Approach

According to the Memon Hafner and Breu (2013), approach to the design of the security Component Authorizations, and Auditing addresses the incorporation of these components into a security service composite. Service Component Architecture for security components. For instance, the Single Sign-on Authentication component implements the Sign-on protocol, and the Authorization Protocol. Apache-Tuscany recently provided an API for applying the SCA for writing composites from Java or C++ components, Service Component Architecture. <https://www.osoa.org/display/Main/Home>.

Open SOA offers a SCA Policy Platform. The architecture provides elements with security services, using Security Intents. Our approach to providing security varies from that introduced through the SCA-Policy Framework of open SOA. Because the components referred to these requirements, provide functionality for the application, whereas we are using components that provide protection functionality.

The SCA-Policy Framework also considers the implementation of security purpose as WS Policy only. Alternatively, we find the implementations best suited to the target programme and runtime environment (i.e. WS-Security Policy, J2EE Deployment Descriptor, etc.). Safety composites may be written in Service Composition Definition Language (SCDL) based on XML, (Memon Hafner & Breu, 2013).

Figure 4. Service component architecture based security component composite, (Memon Hafner & Breu, 2013)



5.3. Event Attack Modelling Approach

Nabi, Yong and Xiaohui (2020) proposed the idea of Event-Attack Modelling Technique based on Uppsala Tool. Faisal modelled the Event oriented subversive attack life cycle showing a small inter-component communication application model that dependent on chain of business component's processing logic, triggered by component-based software (CBS) faults. The Event-oriented subversion life cycle.

Figure 5. Event based subversion attack modelling, Nabi (2020)

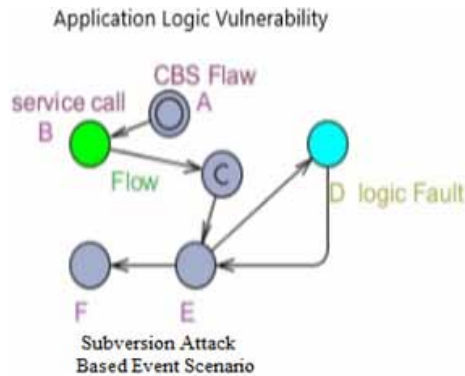


Figure 5 indicates an example for an attack event model; the class is a subversion attack based on component functionality that could be faulty in CBS. This fault could affect service calls and the functional flow that is depending on event-based communication to other objects in the system. As shown in Figure 4, C is a state, which must suit component D in order to proceed with the E before processing to the standard flow logic. The D component is therefore a logic flaw, which does not allow the service to run in accordance with the normal flow of the CBS call service, which is why the automated code & system vulnerability Scan tools cannot find such defects and these defects or faults fall within the logical vulnerability classification. This method will help detect weakness at (SDLC) early on during the design stage.

5.4. Comparison of Modeling Technique and Discussion

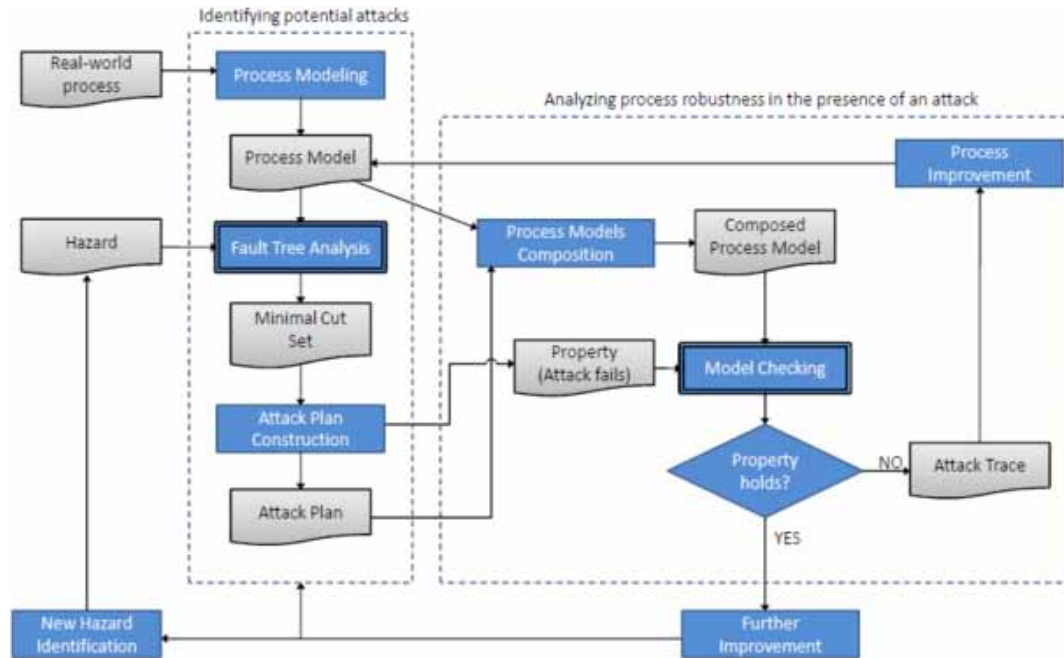
During the security review of the represented techniques, it is noticed that each technique has a valid reason to justify the claim that corresponds the issue mentioned in this paper. Analysis of security flaws are discussed in this section covers the basic idea of current problem solutions (Event based attack), which is being faced by the security auditors and administrator. The critical analysis of these three approaches describe the different techniques, such as Event Control modelling with UML that discusses the Event based system call behaviour, especially when modules are interacted in an event call process to complete a task. This approach is more likely based on design by security concept. However, the other described techniques are security components deployment in SCA while designing the security of service component based applications. The drawback of this technique is to cover the authentication layer for application access security to validate the authentication process of application but not to cover the business component inter-communication layer. This presents the very slow process of solid defence against the vulnerability identification throughout the development life cycle and while reusing existing application functionality, where Events are produced and consumed in process of inter-component- communication model.

Therefore, in the light of above presented model, it can be concluded that a combined method of these three approaches elaborate the different way of providing Event based system design security throughout different layers. However, in recent practices the most efficient technique is third technique Event attack modelling by Faisal Nabi 2020 is more appropriate, while dealing the Events based inter-component interaction process, that modelling system or application design on early stage of SDLC or re-use of Event application function. Therefore, Event Attack Modelling Technique is a useful tool, which may helpful for developers to integrate components to develop composite applications in service component architecture method. The most important function or feature of this technique is based on graphical note presentation while depicting the component functional systematic attack graph and service attack flow.

5.4.1. Attack Analysis Model

The Figure 6 illustrates the event that process the attack having being infected, the technique of *Event Attack* process model as defined given below diagram. This presents the systematic process of the vulnerability infects CBS application that refers to the fault tree analysis, which proceeds towards the attack plan construction. This is followed by model checking and generates the attack scenario that is traced through final analysis of an attack process such as DDos base attack Event process. This technique helps to modelling vulnerability in the service component architecture based applications.

Figure 6. Analysis attack event process model



6. CONCLUSION

The research paper has reviewed the comprehensively security updates in the field of component inter-communication Event base model for service component architecture. The research will help to understand the threats and types of attack that may a site can face while dealing online event base systems. Our research has put a benchmark for the current research in Event base security paradigm, which helps to design event-based systems. The contribution of this work is highlighting the current security efforts that do not concentrate on event attacks or correctly detect component-by-component inter-communication event model and produced the solution of the problem through the three-dimensional approaches analysis and comparison, which is a key contribution of this research. It also open future work to be done in this domain by the researchers, so that Event based distributed system can be more secure.

REFERENCES

- Agirre, A., Marcos, M., & Estevez, E. (2012, September). Distributed applications management platform based on Service Component Architecture. *Proceedings of the 17th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 17–21. <https://ieeexplore.ieee.org/document/6489684>
- Aniello, L., Baldoni, R., Ciccotelli, C., Luna, G., Frontali, F., & Querzoni, L. (2014, May). The Overlay Scan Attack: Inferring Topologies of Distributed Pub/Sub Systems Through Broker Saturation. *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems (DEBS)*, 107–117. <https://dl.acm.org/doi/10.1145/2611286.2611295>
- Bastos, L., & Castro, J. (n.d.). *A Event Based Layered Architecture for Bank Systems*. <https://www.semanticscholar.org/paper/A-Event-Based-Layered-Architecture-for-Bank-Systems-Bastos-Castro/9569d5930de34cec61e2bef267ff73d0af1b384c>
- Carzaniga, A., Rosenblum, D., & Wolf, A. (2001). Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3), 332–383. doi:10.1145/380749.380767
- Chin, E., Felt, A., Greenwood, K., & Wagner, D. (2011, June). Analyzing Inter-Application Communication in Android. *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 239–252. <https://dl.acm.org/doi/10.1145/1999995.2000018>
- Continuous Code Quality | SonarQube. (2017). <https://www.sonarqube.org/>
- Denney, E., & Fischer, B. (2009, July). Generating Code Review Documentation for Auto-Generated Mission-Critical Software. *Proceedings of the Third IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 19-23. <https://ti.arc.nasa.gov/m/pub/580/SMC-IT-Denney.pdf>
- Expansion Files Android Developers, A. P. K. <https://developer.android.com/google/play/expansion-files.html>, 2017. [Online; accessed August 15, 2017].
- Java Message Service (JMS). (2016). <http://www.oracle.com/technetwork/java/jms/index.html>
- Kung-Kiu, L., & Ukis, V. (2007). *On Characteristics and Differences of Component Execution Environments*. University of Manchester Preprint CSPP-41. <http://www.cs.man.ac.uk/~kung-kiu/pub/cspp41.pdf>
- Lee, Y., Nam, D., & Medvidovic, N. (2019). *Technical Report: USC-CSSE-17-801*.
- Li, L., Bartel, A., Bissyandé, T., Klein, J., Traon, Y., Arzt, S., Rasthofer, S., Bodden, E., Outeau, D., & Mcdaniel, P. (2015, May). IccTA: Detecting Inter-Component Privacy Leaks in Android App. *Proceedings of the 37th International Conference on Software Engineering (ICSE)*, 280–291. <http://2015.icse-conferences.org/>
- Lu, L., Li, Z., Wu, Z., Lee, W., & Jiang, G. (2012, October). Chex: Statically Vetting Android Apps for Component Hijacking Vulnerabilities. *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 229–240. <https://dl.acm.org/doi/10.1145/2382196.2382223>
- Memon, M., Hafner, M., & Breu, R. (2013). A Platform-independent Framework for Security Services. *Models in Software Engineering: Workshops and Symposia at Models*.
- Nabi, F. (2011). Designing a Framework Method for Secure. Business Application Logic Integrity in e-Commerce Systems. *International Journal of Network Security*, 12(1), 29–41.
- Nabi, F., & Nabi, M. (2017). *A Process of Security Assurance Properties Unification for Application Logic*. Academic Press.
- Nabi, F., Yong, J., & Xiaohui, T. (2020). A Novel Approach for Component based Application Logic Event Attack Modeling. *International Journal of Network Security*.
- OASIS Service Component Architecture/Assembly (SCA-Assembly). (n.d.). <http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-spec/v1.2/csd01/sca-assembly-spec-v1.2-csd01.pdf>
- Owasp Orizon. (2017). https://www.owasp.org/index.php/Category:OWASP_Orizon_Project
- Petroni, F., Querzoni, L., Beraldi, R., & Paolucci, M. (2016, April). Exploiting User Feedback for Online Filtering in Event-based Systems. *Proceedings of the 31st Annual ACM Symposium on Applied Computing (SAC)*, 2021–2026. <https://dl.acm.org/doi/proceedings/10.1145/2851613?tocHeading=heading77>

Pietzuch, P. (2011, September). Building Secure Event Processing Applications. *Proceedings of the First International Workshop on Algorithms and Models for Distributed Event Processing (AlMoDEP)*, 11. <https://dl.acm.org/doi/10.1145/2031792.2031794>

Rasthofer, S., Arzt, S., & Bodden, E. (2014). A Machine-Learning Approach for Classifying and Categorizing Android Sources and Sinks. *NDSS*. doi:10.14722/ndss.2014.23039

Reimer, D., Schonberg, E., Srinivas, K., Srinivasan, H., Alpern, B., Johnson, R., Kershenbaum, A., & Koved, L. (2004, July). SABER: Smart Analysis Based Error Reduction. *Proceedings of the 2004 ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, 243–251. <https://dl.acm.org/doi/abs/10.1145/1007512.1007545>

Rodríguez, M., Zalama, E., & González, I. (2016). Improving the interoperability in the Digital Home through the automatic generation of software adapters. *RIAI Rev. Iberoam. Autom. Inform. Ind.*, 13, 363–369.

Safi, G., Shahbazian, A., Halfond, W. G., & Medvidovic, N. (2015, August). Detecting Event Anomalies in Event-Based Systems. *Proceedings of the 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 25–37. <https://dl.acm.org/doi/10.1145/2786805.2786836>

Service Component Architecture. (2018). <https://www.osoa.org/display/Main/Home>

Shand, B., Pietzuch, P., Papagiannis, I., Moody, K., Migliavacca, M., Eysers, D., & Bacon, J. (2011). Security Policy and Information Sharing in Distributed Event-Based Systems. *Reasoning in Event-Based Distributed Systems*, 151–172.

Srivatsa, M., Liu, L., & Iyengar, A. (2011). Event Guard: A System Architecture for Securing Publish-Subscribe Networks. *ACM Transactions on Computer Systems*, 29(4), 10:1–10:40.

Tripp, O., Pistoia, M., Cousot, P., Cousot, R., & Guarnieri, S. (2013, March). ANDROMEDA: Accurate and Scalable Security Analysis of Web Applications. *Proceedings of the 16th International Conference on Fundamental Approaches to Software Engineering (FASE)*, 210–225. https://www.researchgate.net/publication/262170634_ANDROMEDA_accurate_and_scalable_security_analysis_of_web_applications

Wun, A., & Jacobsen, H. (2007, November). A Policy Management Framework for Content-based Publish/Subscribe Middleware. *Proceedings of the ACM/IFIP/USENIX 1907 International Conference on Middleware (Middleware)*, 368–388. <https://dl.acm.org/doi/10.5555/1785080.1785106>

Xanitizer. (2017). <https://www.rigs-it.net/index.php/product.html>

Xenitellis, S. (2002, May). Security Vulnerabilities in Event-Driven Systems. *Proceedings of the IFIP TC11 17th International Conference on Information Security: Visions and Perspectives (SEC)*, 147–160. <https://dl.acm.org/doi/10.5555/647185.719818>

Faisal Nabi is a PhD researcher at University of Southern Queensland. He has also received Honorary PhD in Computer Science from Brock University St. Catharines, Ontario, Canada. Faisal's research interests are e-commerce security and software security.

Jianming Yong (PhD) is a professor at school the of information systems at University of Southern Queensland. He has received his PhD from SwinburneUT. He is also member of IEEE professional. His research areas are Cloud Computing, Big Data Security and Privacy, Data Integration, Workflow systems, Information system security, Network management, Web service for SMEs, Digital Identity Management.

Xiaohui Tao (PhD) is a Senior Lecturer (Computing) at School of Agricultural, Computational and Environmental Sciences, University of Southern Queensland. His research interests are Ontology learning and mining, Knowledge engineering, Web intelligence, Data mining, Sentiment analysis and opinion mining, Machine learning, Information retrieval. He is also member of IEEE Professional.

CHAPTER 4 A: CLASSIFICATION OF LOGICAL VULNERABILITY BASED ON A GROUP ATTACKING METHOD

4B: ORGANIZING CLASSIFICATION OF APPLICATION LOGIC ATTACK IN COMPONENT-BASED E-COMMERCE SYSTEMS

Introduction and Findings: Relationship between Chapter 3 and Chapter 4

Introduction: Chapter 3 covers and provides details of (SCA) service component architecture-based application logic design and event-based attack in-service composite applications. Related to this, Chapter 4 covers and provides the detail about the classification of security-related issues, and further research findings in Chapter 4B, which were then proposed into groups and made into a taxonomy. This categorized the group attacking method and classification of two groups of vulnerabilities (technical vs logical) in e-commerce component-based applications. Moreover, the model presented in this chapter is validated in the Paper that is presented as 4 B. This chapter addresses the sub-questions 1 and 2 in this thesis.

Findings: This chapter provides valuable findings regarding a taxonomy in the field of component-based application logic that focuses on logical and technical vulnerabilities as explained in the publication shown in Chapter 4A. This also provides validity of the proposed model which is part of the published papers at 4A and 4B. It also validates the attack pattern in component-based software application logic.

Paper 4A was published in The 10th International Symposium on Frontiers in Ambient and Mobile Systems (FAMS 2020) held April 6-9, 2020, in Warsaw, Poland, Procedia Computer Science 170 (2020) 923–931, Elsevier.

Paper 4B, which was published in the Journal of Computer Science Q3, and validated the proposed model in Chapter 4, developed a taxonomy for developers of J2EE platform. This helps to improve the CVC database which is used for vulnerability reporting and information gathering. Journal of Computer Science 17(11):1046-1058 DOI:10.3844/jcssp.2021.1046.1058..



The 10th International Symposium on Frontiers in Ambient and Mobile Systems (FAMS 2020)
April 6-9, 2020, Warsaw, Poland

Classification of Logical Vulnerability Based on Group Attacking Method

Faisal Nabi* , Jianming Yong , Xiaohui Tao

University of Southern Queensland, QLD4350, Australia

Abstract

New advancement in the field of e-commerce software technology has also brought many benefits, at the same time developing process always face different sort of problems from design phase to implement phase. Software faults and defects increases the issues of reliability and security, that's reason why a solution of this problem is required to fortify these issues. The paper addresses the problem associated with lack of clear component-based web application related classification of logical vulnerabilities through identifying Attack Group Method by categorizing two different types of vulnerabilities in component- based web applications. A new classification scheme of logical group attack method is proposed and developed by using a Posteriori Empirically methodology.

© 2020 The Authors. Published by Elsevier B.V.

This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>)

Peer-review under responsibility of the Conference Program Chairs.

Keywords: Security Dimensions, CBS Application, Group Attack Method, Application logic, Attack Classification

* Corresponding author. Tel.: 61+0405265929; fax: +0-000-000-0000 .

E-mail address: faisal.nabi@yahoo.com

1. Introduction

The growing complexity of modern e-commerce software based on component architecture is creating many benefits for e-commerce industry. However, at the same time critical process of different available commercial of the shelf components may cause of software application logic faults and defects during the plug and play phase of application new functionality development that increases the issues of reliability and security [3]. Therefore, an approach is required to classify the issues on the base of component-based software faults and flaws categorization scheme, which then classify each attack into group attack ID through attack method. The characterization of the attack method is based on vulnerability that may cause of fault logic into an application design. The design faults or flaws are system design phase issues those cannot be mitigate through modification of few lines of component code or interface connection code [10]. The security of such problems are discussed through security dimension which reflects the system aspects and attributes. This may be affected by risk of loss in the event of cyber-attack through group attacking method. The security dimensions are divided into categories of problem where attacking method that may cause of logical vulnerability into a system. This help to the developers understand the design issues of security related system attributes. The security dimension is based on further attributes of the security system, such as security group knowledge, attack group knowledge, vulnerability category and attack boundary, and group attack method in system. These all attributes perform a major role in identifying and classifying the logical vulnerabilities based on group attack method. A group attack method explains the type of vulnerability and its attacking parameter that trigger an infected component in the case of particular event within the system. This process exploits the system security dimension. Therefore, such a scheme is needed to be developed that could characterize the two different vulnerabilities, logical and technical into groups and classification.

1.1 Objective

The research focuses the progress towards the highlighting different security dimensions of categorized vulnerability into classification of each attack with parameter that cause of triggering an exploitable event within the system. This will help to understand the further attributes of security dimension related to a system.

1.2 Method

Our research methodology focuses on a classification that separates or orders of main objects and specimens related to classes. The classifications can be derived by a priori or a Posteriori Empirically by considering the CVE vulnerability database for security breach cases [11].

2. Related work

Samaila et al. 2017, classified the vulnerability into three units by intersection each of these three units: First Units is (i) a system's weakness that cause of a flaw, Second Units is (ii) Attacker approach of attack the flaw, and Third Units is (iii) Being able to exploit the Flaw by an attacker [1] but did not proposed any classification based on these three units or categorized them into attack cause.

Krsula, 1998, defined the classification of software vulnerabilities related issues which is based on fault that is in case of faults specification, development / configuration related to software. For example execution can violate clearly defined security policy [2]. This can be mitigated through the elimination of these problem in a numerous ways, such as software patches and re-configuring the devices [5]. Krsula's classification is more likely about environmental fault which describes given below exemplified figure of taxonomy of software. However, the shortcoming of his research is, proposed scheme limitation to the software fault related environmental condition. Joshi and Singh 2014 has proposed the classification five dimensional vector of vulnerability and defined the defense, method and its impact related to target attack [3]. However, his work most likely cover the network vulnerabilities, which shortfalls about design flaw in architecture of software base application in case of component-based development. Software vulnerability occurs due to the existence of software bugs, faults and errors, which cause an unchecked buffer or race condition [4].

By the date now, there have been many different classification schemes [12, 13, 14, 15, 16] proposed targeting various parameters related to technology could affect the software production process specially in the phase of (SDLC), revelation process and attack pattern [6].

Modern classification of vulnerability models mostly targets the vent of software vulnerability, that is a single cause specially concentrate on domain specific application. It is also possible that a single vulnerability may not cause of a single reason [8]. Many different reasons can cause of a single vulnerability in a system [9]. Therefore, a single cause can be reason of different vulnerabilities in different sort of applications based on class of domain. So it is very much clear that such presentation does not classify the classification models in a holistic way or presentation. Moreover, present schemes do not provide any detail about logical vulnerability-based attack classification and group attack method. This paper covers the research gap between present classifications as stated in related work and the approach adopted in this paper “Classification of logical vulnerability “and group attack method.

3. Proposed Vulnerability Classification Model

The security dimensions are considered as aspects of system and attributes or related process that leaves its effects on security group to know system and delivers the changes to the system. This is based on understanding of the class of vulnerability and its category. The security dimensions directly impact on security group knowledge to evaluate the attack vector related to the security in network or system. This can be both logically and technically, each aspect of both can be categorized and a classification is given before mitigating the security issues.

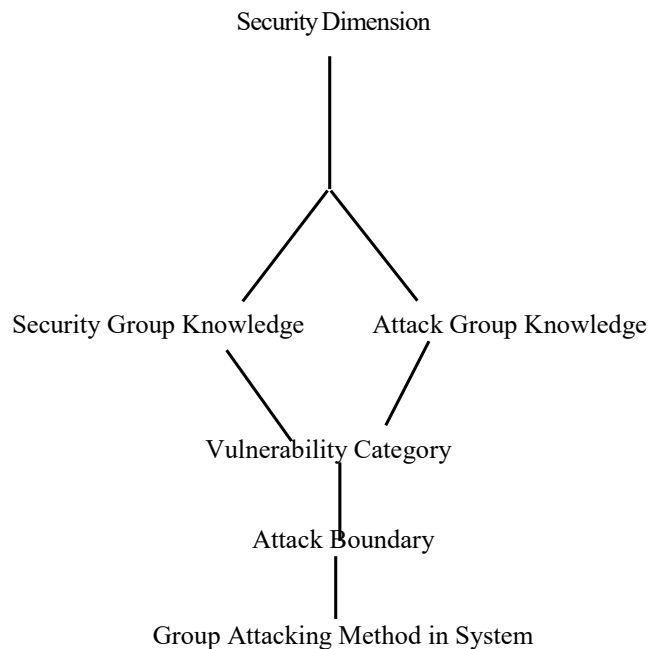


Fig. 1. The Proposed Vulnerability Classification Model

The attack group knowledge also refers to attack pattern that depends on rigorous methods of exploitation by attacker. This dimension of security based on process or set of system attributes that may be exploited in an action by attacker with means of gaining access to the system related information. The next fourth element of security dimension is vulnerability category. In this stage having evaluated by the first two process of security group and attack group knowledge gained, a vulnerability is classified and the categorized into its class of group based on exploitation technique and parameters. Once a vulnerability is categorized its attack boundary profile that is designed keeping in view the level of impact on the system in case of exploit the security function. This helps to understand the level and scale of infection or impact onto the system that became target of attack propagation.

An attack boundary is basically defined through a set of systems under attack that is controlled as a single administrative control. At this level boundaries are various, and vulnerabilities can become obvious as data object is input boundary race condition.

The group attacking method consist on attack ID, classification and attack group that simplifies the vulnerability and attacking technique, whereas group classifies the attack dimension fall under the category. The purpose of this model is to simplify the attack dimensions and way of attack fall under the category, where each vulnerability is subdivided into attack class and method, as defined in the model. Presentation of model is depicted through the table of Grouping Attack Method ID & Logical Vulnerability Classification.

Table 1. Group Attacking Method ID and Vulnerability Classification.

SN	Attack Classification	Attack method & Parameter	Attack Group & pattern	Category
1	Application Logic attack	Logic Design Fault	Exploitation of Functionality	Web Application
2	Application Logic attack	logic diversion error	Anti-Automation	Web application
3	Application logic attack	control flow error	web function exploit	Web application
4	Application Logic attack	programme logic flaw	Subversion of Logic	Web application
5	Application Logic attack	functional flow Fault	exploit the sequences of logic order	Web application
6	Application Logic attack	Design logic flaw	web Copy Cat	Web application

The logical attacks are different types of attacks with different attack methods because logical attack has to exploit the functionality that is specific to the application and its logic .This is what, defined in the above mentioned table of Grouping Attack Method ID & Logical Vulnerability Classification.

As mentioned above the main scope of this study is to focus on “application logic based vulnerabilities” problem that is because of a design flaw or fault that mismatch between design & architecture while developing component –based software application. We have classified the six vulnerabilities in the application logic and then developed the attack group and vulnerability classification to be categorized by proposed model of classification and security dimension in the light of vulnerability model that is cause of design flaw in application logic and functionality.

3.1 Classification of Logical Vulnerability VS Technical Vulnerability

In the light of our research, the proposed model would turn into be a classification & characterization of two distinctive categories of vulnerability issues /problems “Technical vs Logical Vulnerabilities”. These vulnerabilities are classified based on the attack method as mentioned in the above table of vulnerability, this classification relates to attack pattern technique. Therefore, keeping in view the proposed model of classification falls under the two categories of vulnerabilities, which have been drawn into classification tree model dividing into sub-class of attack at the application layer of ecommerce component-based software application. This depicts the detailed classification, having characterized each vulnerability by their unique signature of indemnity in the proposed scheme.

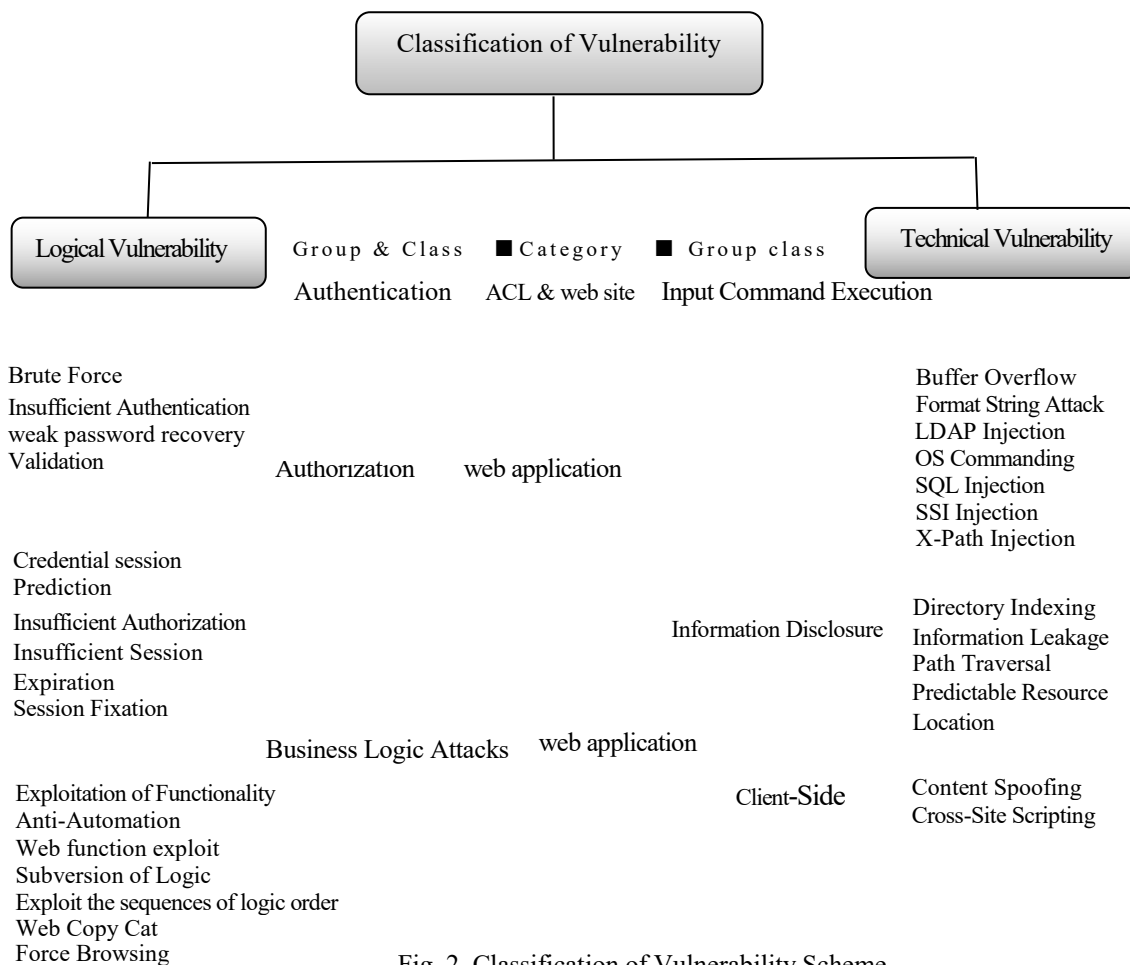


Fig. 2. Classification of Vulnerability Scheme

The proposed contribution of the classification is characterized by attack pattern and target agent in each kind of attack as mentioned in given classification scheme of application logic-based attack pattern method, vulnerability class and event triggering logical element. This is further put into attack pattern technique to classify each vulnerability in the light of attack method, such classifications are characterized in groups of attacking parameters which defines nature of vulnerability. Therefore, in the light of our detailed classification and characterizing of vulnerabilities into groups and their attacking methods, as it is defined in the above-mentioned table. It is derived that logical vulnerabilities are such vulnerabilities which cannot be mitigated through traditional approaches such as web scanning tool and vulnerabilities detection tool those are based on static analysis because web scanners only detect the implementation bugs, programming error conditions, and fault. Whereas logical vulnerabilities are based on design phased flaw of software-based application [17]. Therefore, our proposed scheme is based on classification and categorization of each logical vulnerability based on attack method, which is explained through the parameters of attack logic in above presented table. The classification with defined detailed information about each attack and related attack pattern will be helpful for the developers, having knowledge of the different attacks with technique to design new applications based on the idea of security by design technique.

3.2 Layer Based Software System Scenario Attack Modeling

Figure 3 depicts the software layer based system attack scenario to validate the above-mentioned proposed model. This figure clearly explains the role of software and service into different layers and relationships between actors of organizations and that face threats. This model help us to understand the three-dimensional layer model of software system, service, information and event; the attacker affects those and the attacks must be mitigated through defender actions.

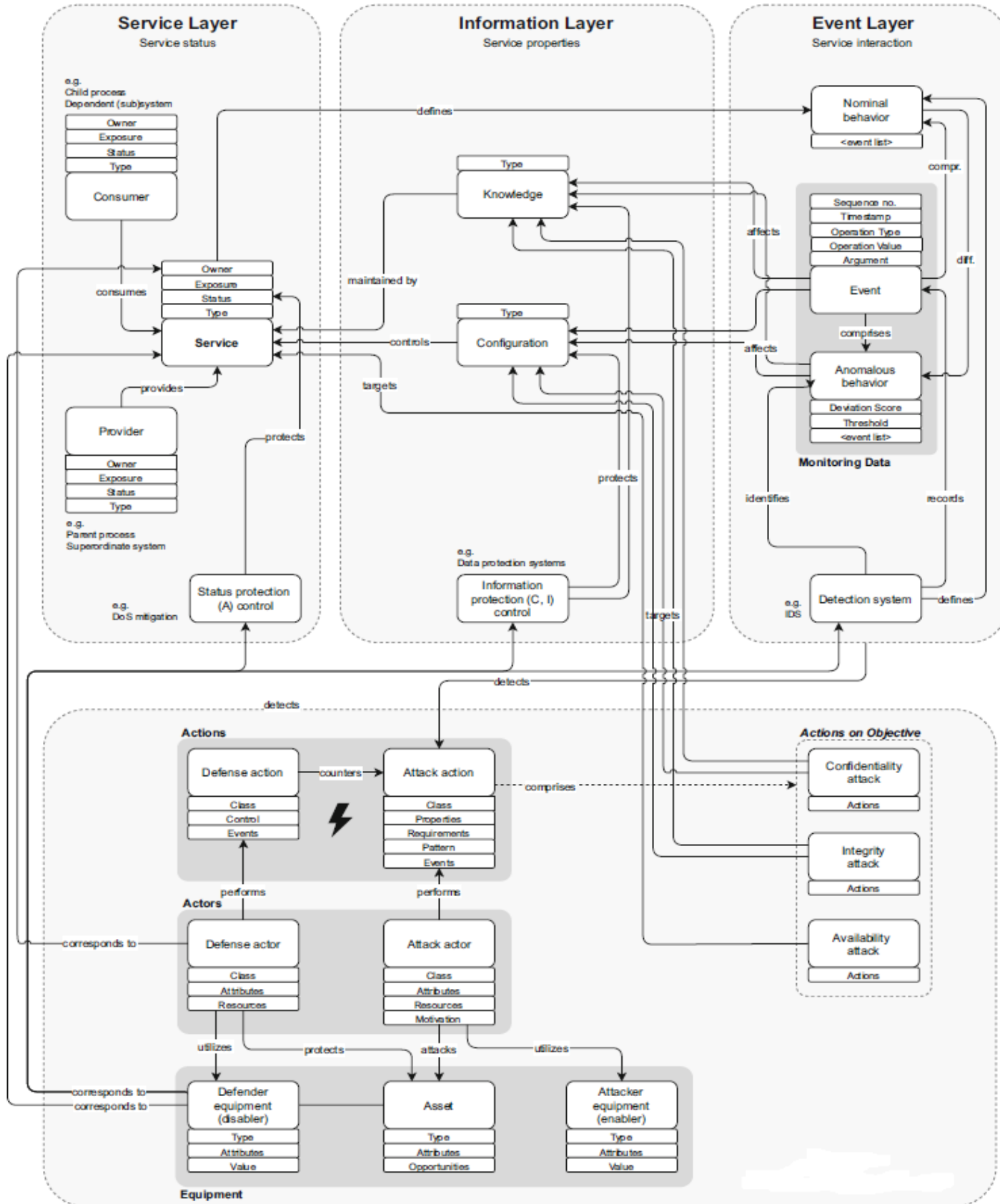


Fig. 3. Layer Based Software System Attack Model

This model classifies the vulnerability lifecycle in the layer based software system attack model. The method and tool for such modelling is UML and the aspect oriented modelling languages that support the event attack modelling through the attack surface, have been demonstrated in figure 3.

3.3 Classifying and Categorizing Logical Vulnerabilities

The group attacking parameters based nature of logical vulnerability and attack technique classification are defined according to each type of attack and characterized according to attack method based on incident reports as mentioned in method section.

1. Attack pattern Technique & Group

Exploitation of Functionality

Class logic design fault

Category: Web server service (Target Agent)

Attack Method (encoding circumvents access controls)

Logic error (attack of Cause)

Implementation level (fault logic classification)

Vulnerability Type: Exploitation of Functionality

This vulnerability class identifies the category of this attack pattern as business logic or application logic, where the attack falls under the logic design fault in the web server side target agent and the method of avoiding it is encoding circumvents access controls.

2. Attack Pattern Technique & Group

Insufficient Anti-automation

Class logic diversion error

Category: Web software (target Agent)

Attack Method (Anti-Automation)

Divert logic (attack of Cause)

Implementation level (Process logic flow classification)

Vulnerability Type: Insufficient Anti-automation

This class of attack falls under the classification of insufficient anti-automation attack pattern technique. The category of this vulnerability falls under the web application that is identified as application logic and the method is process logic flow classification.

3. Attack Pattern Technique & Group

Insufficient Process Validation

Class control flow error

Category: Web application (target Agent)

Attack Method (web function exploit)

Divert application flow control (Attack Cause)

Implementation level (application logic fault classification)

Vulnerability Type: Insufficient Process Validation

This vulnerability falls under the web application category where the attack method is web function exploited with the technique of application logic fault classification and insufficient process validation technique. This comes under the business application of logic vulnerability.

4. Attack Pattern Technique & Group

Subversion of Logic

Class programme logic flaw

Category: Server application (target Agent)

Attack Method (exploit the work flow)

Subvert application logic (attack Cause)

Implementation level (application Design logic Flaw classification)

Vulnerability Type: Subversion of logic

This vulnerability class programming logic fault falls under the category of server side application target agent, where subversion of application logic diverts the control flow of the entire application logic, the method of attack is to exploit the workflow.

5. Attack Pattern Technique

Forced Browsing

Class functional flow Fault

Category: Web logic (target Agent)

Attack Method (exploit the sequences of logic order)

Divert service flow (attack cause)

Implementation level (application function error classification)

Vulnerability Type: Force-Browsing

This class of vulnerability falls under the functional flow fault classification of attack, web logic is the target agent, and where the entire function of web logic diverts service. The method of this attack is to exploit the sequences of logic order.

6. Attack Pattern Technique

Web Copycat

Class Design logic flaw

Category: Web software application (target Agent)

Attack Method (exploit the business logic)

Application logic flow diversion (attack Cause)

Implementation level (Design Flaw classification)

Vulnerability Type: web Copy Cat

This class of vulnerability is classified as web copycat attack target agent is design logic flaw at the web software application that exploit the business logic through application logic flow diversion as an attack cause.

4. Discussion

Therefore, we have detailed the classification and characterization of vulnerabilities into groups and the methods of attacking them. From this research, it may be understood that that logical vulnerabilities cannot be mitigated through traditional approaches such as web scanning tools, and vulnerabilities detection tools that are based on static analysis. Web scanners only detect the implementation bugs, programming error conditions, and faults whereas logical vulnerabilities are based on the design-phased flaw of software based applications [17]. Therefore, our proposed scheme is based on classification and categorization of each logical vulnerability based on the attack method, which is explained through the parameters of attack logic in each case presented above. The classification with defined detailed information about each attack and the related attack pattern will be helpful for the developers, having knowledge of the different attacks with technique to design new applications based on the idea of security by design technique.

5. Conclusion

The idea of security development process is based on a proper classification of the vulnerability. It is very useful to have knowledge about the attack and its parameters, target agent, method. Since with the passage of time new technologies emerges, and the more security attacks occur software application server side, in this scenario researcher has made an effort to classify the logical vulnerabilities those are never given consideration by the research community. The proposed vulnerability classification model contributed the new classification related to group attacking method ID and vulnerability classification, which is never been done this before. The proposed model will cover the gap between previously design taxonomies based on different areas of system domain and security classifications of vulnerabilities. This will be very useful for developers to understand the two different sort of vulnerabilities, specially logical vulnerabilities, while designing applications or security by design based idea adoption by them. This model will cover the gap of logical vulnerabilities and related attack patterns, technique, method. This is a significant improvement to taxonomies a class of vulnerability that is not given much consideration by the academia.

References

- [1] Samaila, M. G., Neto, M., Fernandes, D. A. B., Freire, M. M., & Inácio, P. R. M. (2017). Security Challenges of the Internet of Things. Pp. 53–82, In J. Batalla, G. Mastorakis, C. Mavromoustakis, & E. Pallis (Eds.), *Beyond the Internet of Things. Internet of Things (Technology, Communications and Computing)* Cham: Springer. doi:10.1007/978-3-319-50758-3_3
- [2] Krsul, I. V. (1998). *Software vulnerability analysis (Doctoral dissertation)*. Retrieved from <https://dl.acm.org/citation.cfm?id=927682>
- [3] Joshi, C., & Singh, U. K. (2014). Admit-A five dimensional approach towards standardization of network and computer attack taxonomies. *International Journal of Computer Applications*, 100, 5. doi:10.5120/17524-8091.
- [4] Li, X., Chang, X., Board, J. A., & Trivedi, K. S. (2017). A novel approach for software vulnerability classification. In *Reliability and Maintainability Symposium (RAMS), 2017 Annual (1– 7)*. IEEE. doi: 10.1109/RAM.2017.7889792
- [5] Antoon, R. U. F. I. (2006). *Network Security 1 and 2 Companion Guide*. (Cisco Networking Academy).
- [6] Fournaris, A. P., PoceroFraile, L., & Koufopavlou, O. (2017). Exploiting hardware vulnerabilities to attack embedded system devices: A survey of potent microarchitectural attacks. *Electronics*, 6(3), 52. doi:10.3390/electronics6030052.
- [7] Garg, S., & Baliyan, N. (2019a). A novel parallel classifier scheme for vulnerability detection in android. *Computers and Electrical Engineering*, (Final revision submitted) doi:10.1016/j.compeleceng.2019.04.019.
- [8] Homaei, H., & Shahriari, H. R. (2017). Seven years of software vulnerabilities: The ebb and flow. *IEEE Security & Privacy*, (1), 58–65. doi:10.1109/MSP.2017.15

- [9] Sharma, C., & Jain, S. C. (2014, August). Analysis and classification of SQL injection vulnerabilities and attacks on web 18 S. GARG ET AL. applications. International Conference on Advances in Engineering and Technology Research (ICAETR), 2014 (1–6). IEEE. doi: 10.1109/ICAETR.2014.7012815
- [10] Faisal Nabi, A Process of Security Assurance Properties. Unification for Application Logic, International Journal of Electronics and Information Engineering, Vol.6, No.1, PP.40-48, Mar. 2017.
- [11] Jens L. Eftang a, Martin A. Grepl b, Anthony T. Patera, A posteriori error bounds for the empirical interpolation method, C. R. Acad. Sci. Paris, Ser. I 348 (2010) 575–579 <http://www.sciencedirect.com/> .
- [12] Hansman, S., Hunt R., “A taxonomy of network and computer attacks”. Computer and Security, vol. 24, issue 1, Feb 2005, PP. 31-43.
- [13] Simmons, C., Ellis, C., Shiva, S., Dasgupta, D., & Wu, Q. “AVOIDIT: A Cyber Attack Taxonomy”, University of Memphis, Technical Report CS-09-003, 2009. [Online]. Available:
- [14] T. Aslam, “Use of a taxonomy of Security Faults,” Technical Report 96-05, COAST Laboratory, Department of Computer Science, Purdue University, March 1996.
- [15] Scott D., Angelos S,” Towards a Cyber Conflict Taxonomy”, 5th International Conference on Cyber Conflict K. Podins, J. Stinissen, M. Maybaum (Eds.), 2013.
- [16] Lough, Daniel. “A Taxonomy of Computer Attacks with Applications to Wireless Networks,” PhD thesis, Virginia Polytechnic Institute and State University, 2001.
- [17] Marco Vieira, Nuno Antunes, and Henrique Madeira, Using Web Security Scanners to Detect Vulnerabilities in Web Services, 2009 IEEE/IFIP International Conference on Dependable Systems & Networks, <https://ieeexplore.ieee.org/abstract/document/5270294>.

Review

Organizing Classification of Application Logic Attacks in Component-based E-Commerce Systems

¹Faisal Nabi, ²Jianming Yong, ³Xiaohui Tao, ⁴Muhammad Farhan and ⁵Nauman Naseem

¹CIS, University of Southern Queensland, Australia

²CIS, USQ, Australia

³CS, USQ, Australia

^{4,5}school of IT and Engineering, MIT, Australia

Article history

Received: 06-02-2021

Revised: 12-04-2021

Accepted: 27-05-2021

Corresponding Author:

Faisal Nabi

School of Management and Enterprise, University of Southern Queensland 1 West St, Darling Heights QLD 4350, Australia

Email: faisal.nabi@yahoo.com

Abstract: This research paper addresses the topic of application logic attack taxonomy that is due to unclear and incorrect implementation in component-based applications. The issue addresses the detection and classification of two separate types of vulnerabilities in component-based applications. The paper completes this aim through organising the classification of each attack and then proposes the classification of logical vulnerabilities and discusses the two distinct forms of weakness and coding faults in the application software found in the mid-level of the framework. The most important argument is to desegregate awareness of attack patterns with boundary profile status relevant to an application logic vulnerability and possible threats. Having review of two different types of attack taxonomies, a logical vulnerability classification based taxonomy is proposed.

Keywords: E-Commerce, Web Software Application, CBS Design Flaws, Logical Attack, Vulnerability and Taxonomy, Software Security Flaw

Introduction

The implementation of advanced mechanisms for managing asynchronous events in web browsers and the advent of many frameworks for rapid prototyping of server-side components have been stimulated by the growth of emerging technologies and the shift from 'conditional' applications to Internet-based platforms (e.g., mail readers). Although new technologies have given significant funding, development, productivity and interoperability advantages, little has been done to fix security concerns. As a consequence, the web applications become more complex, the risk of abuse is increasing (Firesmith, 2005). The risk of violence also increases. An overview of the CVE vulnerability database, for example, reveals that web-based attacks rose from 25% in 2017 to 61% in 2018. The fact that component-based applications are typically accessible through designer firewalls makes it possible for developers with insufficient software protection to build server-side logic more widely under time-to-market pressure. As a result, web applications that are unsafe created and made available over the Internet, making it simple to exploit (Nabi and Nabi, 2017).

The use of best practises in industrial fields such as firewalls, encryption (SSL/TSL), vulnerability scan, security monitoring, etc. (e.g., intrusion, white box and black box) has historically been promoted by security engineering in existing systems to insure proper security. Many security papers and books are unable to provide much detail on the e-commerce framework's security specifications and most of what is written seems to stress the concept of ambiguous security objectives or concentrate on architectural constraints. Usually is either the amount required of a stated particular type of security or the safety implications of non-security. Normally, either the amount appropriate to a given security form or the safety effects of non-security specifications are addressed in security processes. Cyber attacks are essential to any component-based security assessment of e-commerce application. In this context, the characterization and classification of vulnerabilities is one of the most important fields of study. Several models suggest defining them; such models usually generally describe attacks (Nabi and Nabi, 2017) In addition, experience shows that attack profiles are highly dependent on multiple frontier conditions. This study addresses the problem of the absence of coherent vulnerabilities and

taxonomies to identify and classify two distinct vulnerability classes in the CSB's web-based e-commerce.

This is achieved by organizing the critical classifications that suggest the classification of logical vulnerabilities centred on design faults versus technological faults focused on web application deficiencies and defects at the implementation level from a security evaluation perspective of component-based software applications. Our research methodology relies on grouping that separates or orders the component-based software applications Classifications can be established as either a priority (i.e., non-empirical from an abstract model) or Posteriori Empirical by evaluating the CVE vulnerability database for security breach cases.

Research Background

A taxonomy of recurrent vulnerabilities may contribute to the organisation of today's safety-enhancing knowledge. To detect possible attacks on web application software before it is published to consumers; advanced awareness of vulnerabilities can be useful. We reviewed 25 taxonomies from 1974 to 2017 and analysed different levels of vulnerabilities, property taxonomies, web application vulnerabilities, network vulnerability taxonomy and software vulnerability taxonomy of e-commerce threat classifications before restricting the main scope of this study to address the logical problems of the web software application due to mismatch between design and architecture. However, it depends on web software application during development. Our attack patterns are more detailed to which components could recognise a device design vulnerability.

Most taxonomies have four hierarchical groups within the taxonomy: Structural flaws, environmental deficiencies and codes. We contrasted our taxonomy with the environmental defect class, which is intended to infringe the environmental standards of programmers and their software weakness.

Since most (Nabi and Nabi, 2017) researchers did not find any information on the design vulnerabilities in real-time, they could not provide any information on this vulnerability and its attack classifications.

Research Methodology

Our main objective is to develop the taxonomy of logical weakness in the application layer of distributed multiple-tier e-commerce systems, as stated in the introduction. There are several methodologies to assess the security of information communication technical infrastructure that are developed in various papers and texts, which provide a launchpad into an e-commerce

system. We have selected Masera and Nai methodology 2005 as a guide to support our methodology. The authors present in Masera *et al.* (2005) a risk management method for the assessment of complex ICT systems. This approach accepts the fact that a description of the function, components, properties and the relationship between components, assets and the outside world should be first given for the safety evaluation of a system. This can be used to identify defects that influence the system as a whole systematically.

Our research methodology is also focused on the Posteriori Empirical study of CVE vulnerability database data from various levels of e-commerce categories of web-based applications and systems (B2B) and (B2c) from 2002 to 2017. Specific groups of single characteristics are used with a set of taxonomic characters that meet the classification needs of subjective decisions. These classifications are simplest and require a clear selection criterion for individuals to be grouped. For instance, group programmes use encryption or not in their language of programming. The evaluation of potential damage to the components, their propagation to the system and subsequent attack patterns can be extracted from the evaluation of this information.

As described above, web applications and systems for e-commerce and those elements that form the basis of our methodology are strongly linked to a set of traditional computer security principles, particularly the "five pillars." We also developed a Security Vulnerability Evaluation Model focused on "Five Pillar" Computer Security Elements for component-based e-Commerce software applications and systems. This enables vulnerability to be identified and attacks to patterns that lead to our main goal of classifying logical vulnerabilities (Moore *et al.*, 2001).

In the other hand, technological flaws are due to mistake, fault and bug coding at implementation level for a software development framework. During such a process, they can be patched. Furthermore, the use of vulnerability analysis software and web application scanning tools is difficult to repair or identify faults in design. Therefore, no taxonomy provides details on the logical danger of the application layer targeting attacks and patterns related to vulnerabilities and attacks in the mid-level business application logic (the n-tier e-commerce system).

In component Web Applications and Systems, we propose the SVAM for the main computer protection attributes 'Five Columns,' as mentioned, showing the life cycle of the vulnerability and classifying the key point where the vulnerability covers two or more delicate vulnerability classes, such as 'Technical and Logical., as defined in Fig. 1.

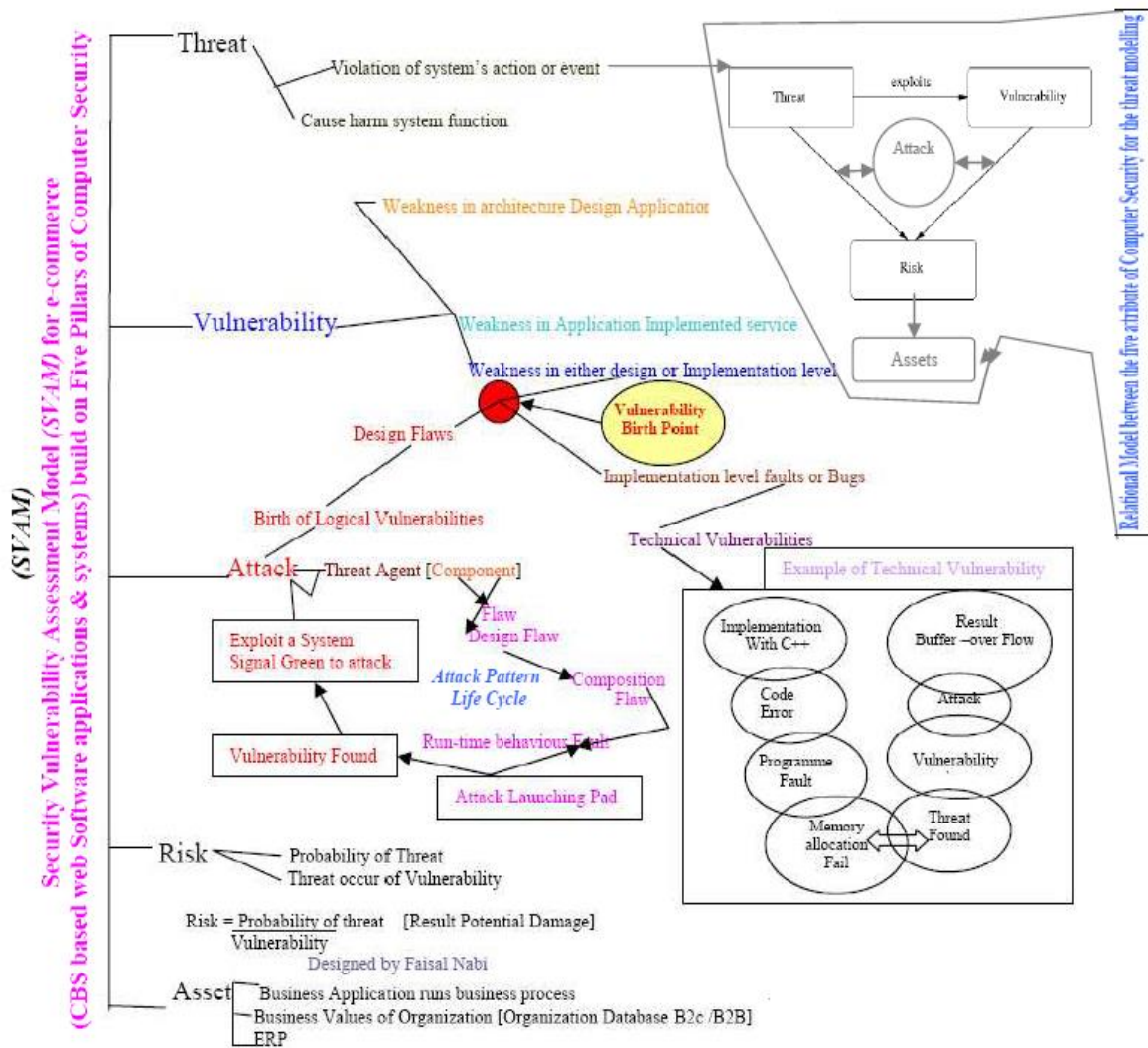


Fig. 1: SVAM model

Related Research Work and Taxonomic Properties

The theoretical analyses are categorised into taxonomy (Simpson, 1945; Moore *et al.*, 2001; Masera *et al.*, 2005), including their base, principles and procedures and standards. The grouping and/or arrangement of objects (or specimens) into groups is a classification. Non-empirically generated classifications are known as priori classifications. Empirically generated classifications are called subsequent classifications by analysing the data.

Objects, Attributes and Constraints of a System

Object: An object is an "entity" that provides or receives information and possesses a unique name and a collection of operations on it (Longley and Shain, 1990).

Attribute of Object: An object attribute is an object's data component and a derived attribute from another attribute is a later attribute's data component.

Property of Attribute: The attribute property is a property of the attribute, which can be obtained from the attribute by applying a function to the attribute.

Attribute refinement: An attribute refinement is a final refining of attributes wherein larger attributes that contributes to the identification of attributes with assumptions. The refinement attribute can-not contain an attribute element. The refinement attribute can't contain an attribute property.

Attribute Constraint: The Constraint attribute defines the ownership or collection of assumptions regarding this particular attribute.

Table 1 defines attack pattern properties.

Table 1: Attack pattern properties

Pattern name and classification	A unique, descriptive identifier for the pattern
Attack prerequisites	What conditions must exist or what functionality and what characteristics must the target Software has, or what behaviour must it exhibit, for this attack to succeed?
Description	A summary of the assault including the course of action
Related vulnerabilities or weaknesses	What specific vulnerabilities or weaknesses.
Method of attack	Which sort of attack vector utilized (e.g., malicious data entry, maliciously crafted file, Protocol corruption)?

Taxonomic Characters, Object Attributes or Features

The basis for determining a positive classification is the taxonomic character (Simpson, 1961; Glass and Vessey, 1995). These are the characteristics or attributes of the objects. These characters are sometimes referred to as characteristics, attributes or features (Simpson, 1961). Asserts the readiness and objectivity of these properties from the relevant objects.

Concept of Attack Pattern

An assault pattern is the abstraction mechanism to describe how an assault is carried out. It also describes the context in accordance with the pattern model where appropriate and then proposes, proposed ways to mitigate the attack rather than conventional patterns. In other words, a pattern of attack is an inference. In a pattern of attack, the following information is typically given.

With regard to the above-mentioned theory and concepts, discussion and references are based on principles, procedures and rules concerning the taxonomic classification of system objects, attributes, properties and characteristics. We want to first describe clearly the vulnerability of web software applications before moving towards a taxonomic contribution focused on classification and characterization of two separate vulnerability categories (Technical vs Logical).

Web Software Application Vulnerability

"The weakness of the Web application software includes misalignment between the application logic and environmental assumptions taken up in development/execution (code written) and the environment within which it is run," we define vulnerabilities in Web Application software (Nabi, 2011).

Taxonomy of Computer Program Security Flaws

A flaw can be defined as malicious or not.

Malicious Flaws

Implemented to cause a breach of the protection deliberately, such as viruses, worms, Trojan-based horses, time bombs and coded trap doors (Landwehr *et al.*, 1993).

Non-malicious Flaws: Incorporated due to missing specifications or design logic mistake.

During the software life cycle, programmes are graded by the time they are incorporated into the programme.

Defaults during development, repair or service are part of the implementation time.

Flaws are concerns that arise in software design. A vulnerability may be a flaw in the software runtime environment. In general, mitigating a defect requires much more work than just a few lines of code. The concern is not just about implementation; the idea behind it is flawed and that is why it is not implemented. For example, a design flaw that does not mitigate a simple action such as changes in array boundary (Nabi, 2005) is a sensitive business logic for an untrusted customer application (Nabi, 2005; 2011).

A Taxonomy of Security Faults

Many classification schemes for security faults have been suggested that categorise faults by different criteria as shown in Fig. 2 (Krsul, 1998; Aslam, 1995):

- Coding faults are composed of faults in the software development process that are introduced during software development. These faults are the cause of errors in programming logic and missing or incorrect requirements
- Operational faults Operational faults are called incorrect software deployment. In most situations, failures can be categorized as operational faults (Aslam, 1995)
- Environment faults occur when a programmer does not completely understand the limitations of the usable right modules or the interactions between them (Krsul, 1998)

A Taxonomy of Security Error, Faults and Failures

Error: An error is a developer mistake. It could be a typographical error, misinterpreting a specification, misunderstanding, etc. (ANSI/IEEE, 1990).

"An error can be the cause of one or more faults"

Fault: Defects can be found in the software code. In particular, the discrepancy between incorrect programming and the correct version (ANSI/IEEE, 1990).

Failures: Faulty code execution can lead to null or more failures when the failure is the [non-empty] difference between the incorrect and correct programme results (ANSI/IEEE, 1990).

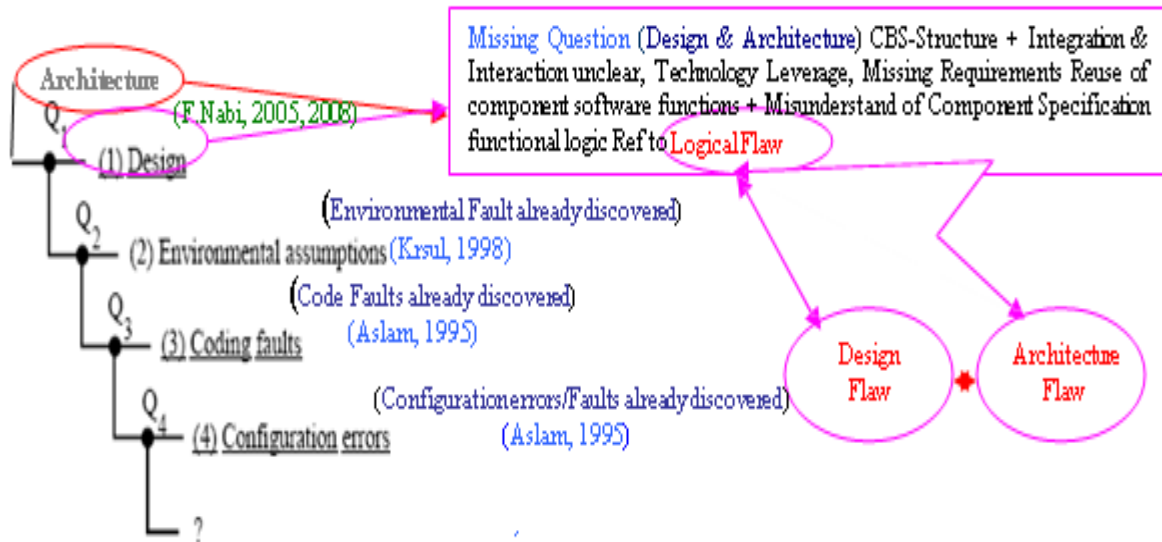


Fig. 2: Taxonomy of Software Vulnerabilities causes

Previous Research Work and Classifications

A detailed understanding of vulnerabilities can help to detect possible attacks on a software programme before they are published to customers. A taxonomy of recurring vulnerabilities can help navigate the details required to increase safety awareness. Between 1974 and 2018, we analysed 21 taxonomies and assessed various levels of vulnerability, classified property taxonomies of e-commerce risks, web application vulnerabilities, network vulnerability taxonomy and software vulnerability taxonomy before restricting the key scope of the analysis to logical attack problems. This is due to a flaw between design and architecture when designing an application with web software.

Taxonomic Classification and Review based Comparison

McPhee (1974) proposed the classification of vulnerability that falls under the category of Design flaw vulnerability, the object of the vulnerability is targeting operating system flaws.

Abbott *et al.* (1976) focus on Layered operation and features that also consider the reason is based on operating system flaws. So this taxonomy is operating system-oriented.

Bisbey and Hollingsworth (1978) Taxonomy is also single dimension targeting operating system based abstract pattern from flaw and automated search flaw. This taxonomy is also operating system-oriented.

Aslam (1995) explained the UNIX security flaw that targets the database vulnerability organization. Overall it is operating system-oriented vulnerability.

Landwher *et al.* (1993) explain the taxonomy of Operating System Flaws categorized vulnerability based on Genesis, Time of introduction and location.

Bishop (1995) explained the UNIX System and Network Vulnerabilities that focus on Effect, Minimum number of components, Source of ID.

Gray (2003) explained the layer-based vulnerability in network operational system.

Jiwnani and Zelkowitz (2004) explained the software flaws in the software development process. This taxonomy is three dimensional.

Pothamsetty and Akyol (2004) explained the Layered based vulnerability targeting the network operational protocol vulnerability.

Tsipenyuk (2005) multi-dimensional coding error-based vulnerability that causes software errors.

Weber *et al.* (2005) focused on also a layer-based software flaw that generates coding analysis and tool-based detection.

Kjaerland (2006) four-dimensional taxonomy explaining the Method of operation and impact of intrusion and its detection.

Bazaz and Arthur (2007) explained the Hierarchical vulnerability taxonomy targeting computer sources and its relation to vulnerability.

Igure and Williams (2008) explained the vulnerability class multi-dimensional attack on computer system resources and process of vulnerability.

Simmons *et al.* (2009) explains five-dimensional network taxonomy focusses on the attack vector, operational process and defense.

Cebula and Young (2010) Hierarchical taxonomy explaining the cyber-attacks and its process to generate vulnerability that cause attacks in the system.

Scott and Angelos (2013) this Hierarchical Network Taxonomy explains the Explore the relationship between events.

Joshi and Singh (2014) five-dimensional taxonomy focusing on attack entity, defence method and target, impact, which explains the nature of the attack.

Joshi *et al.* (2015) review the existing taxonomies related to computer attacks and vulnerability in the system. This mostly, targets the network-based vulnerability detection method overview.

Li *et al.* (2017) represented the software-based vulnerabilities and propose the model to mitigate the software vulnerability issues.

Chen *et al.* (2018) explained the Taxonomy of Internet-of-Things Security and Vulnerabilities that address that internet of things security wholes and related vulnerabilities in the system and applications.

Overall Review and Comparison

There is a number of vulnerabilities and attacks noted previous taxonomies which most do not concentrate on logical software vulnerabilities. This difference clearly identifies the needs for a systematic model and classification of these groups into class vulnerability against technological vulnerability. Therefore, through the vulnerability life cycle in background software process model, we introduced a new taxonomy and its implementation life cycle. This model demonstrates clearly the birth and life cycle of vulnerability.

Classification of Security Threats in e-Commerce

Generally, structural analysis allows a phenomenon to be classified. In particular, a formal e-commerce threat classification would allow managers to develop less fragile system (Álvarez and Petrović, 2003). The following classification properties are recommended for reporting accidents to incident response teams.

- The categories should be mutually exclusive (maximum one for each category) and collectively complete (each specimen should be at least one category). The various categories should be mutually exclusive (one category should be the most suitable for all specimens) and uniformly exhaustive (all

specimens should fit in at least one category). In addition, the types should be mutually exclusive

- In each category should be included specific and clear criteria for the specimens to be included in the category
- Not only security experts but also less qualified and seasoned users and administrators can benefit from intuitive and useful taxonomy
- The terminology of taxonomy should comply with existing safety terminology (which can not always be defined easily)

Classification of Web Taxonomy

Chirs and Frank (2005): Addressed a methodology for vulnerability taxonomization and an example of web services, WS architectural model of four components and their connections. It addresses two subclasses. 'Input Format and Input Origin' then contains attack flows based on a category of border state error, which is exceeding an unforeseen long input that executes arbitrary code from an attacker (programme written in C or C++). (Format and Input Origin). The authorship is the proposed Result Matrix, which is the same that (Aslam, 1995; Krsul, 1998) classifications and almost a copy thereof.

Álvarez and Petrović (2003): Entered the web attack taxonomy. Specific web categories are entry point, aim, HTTP verbs and HTTP headers, which are not covered by general taxonomies and are considered important for the precise classification of Web attacks. However, other types, such as vulnerability to site-specified values (e.g., code injection, HTML handling, etc.), will usually face taxonomies, canonicalization, overload and misspellings. Alvares differentiated & ordered the taxonomy from the point of view of the attacker. The author clarified that because of two vulnerability errors, an attacker might get access to a point that should be a web server or web application entry point looking for an attack.

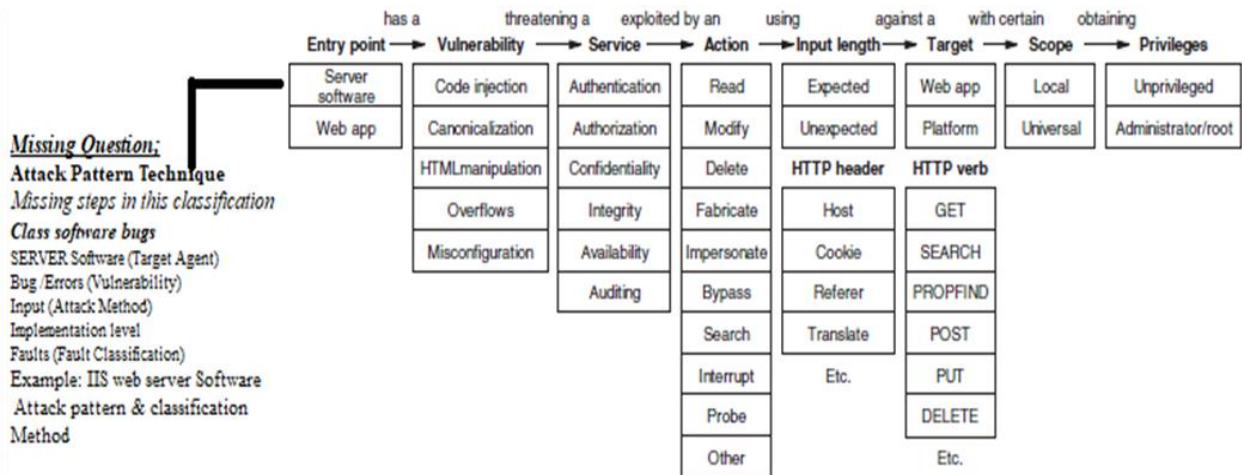


Fig. 3: Web attacks taxonomy (Álvarez and Petrović, 2003)

It reflects the widespread life cycle of a hacker attack based on the HTTP as shown on the Fig. 3. It is also incomplete taxonomy and cannot be called a classification scheme. Since attack patterns cannot be categorised, (Krsul, 1998; Aslam, 1996), classified the classification of vulnerabilities and their characterization by their attributes.

Proposed Classification and Types of Logic Attacks

There are different types of logical attacks every time and a particular application function/method must be used for taxonomy. The logical attacks are designed to interrupt the application's logical flow. The logic of implementation is the logical flow that a certain procedure is supposed to be carried out. The software logic contains examples of password recovery, account registration, auction requests and transactions for e-commerce. A website may provide a consumer with a multi-stage process to carry out a certain action properly. An attacker can bypass or use these features to cause website or users damage. As previously stated, the study focuses on the problem of "application logic-based vulnerabilities" as design and architecture differ during development of web applications. In the application logic, we find seven faults/flaws as illustrated in Fig. 4 and then a case that endorse Taxonomy as a source of reference faults for design faults.

In each type of attack, the attack pattern and target agent define the proposed taxonomy contribution. As above, graphical attack pattern methods and vulnerability classes based on application logic are logical presentation

as defined in Fig 5. This is further used to categorise each vulnerability because of an attack process, characterised in-group of attacking parameters that determine the essence of the vulnerability.

Case as a Reference: Mars Polar Landing Mission (NASA) Dec 3, 1999

The case for component-based systems and their implementations is discussed here as a reference. The case describes one of the classifications identified above of system composition failures or defects while NASA, USA, takes the component-based approach for mission-critical system development.

Reason of Project Failure

Touchdown Monitor (TDM) component failed to comply with the requirements contrasted with its functional specification based on the specification integration via contract interface, which led to an MPL device design default and task failure.

Requirement Modeled of TDM

TDM component is an MPL system software which monitors three landing legs during two downward stages.

Logical Component Information Processing

The Multi-Task Monitoring Calls TDM module receives information from the second module on the leg sensors at 100 times per second. TDM software tracks the three touchdown legs during the first process, which begins at 5 KM above Mars' Surface.

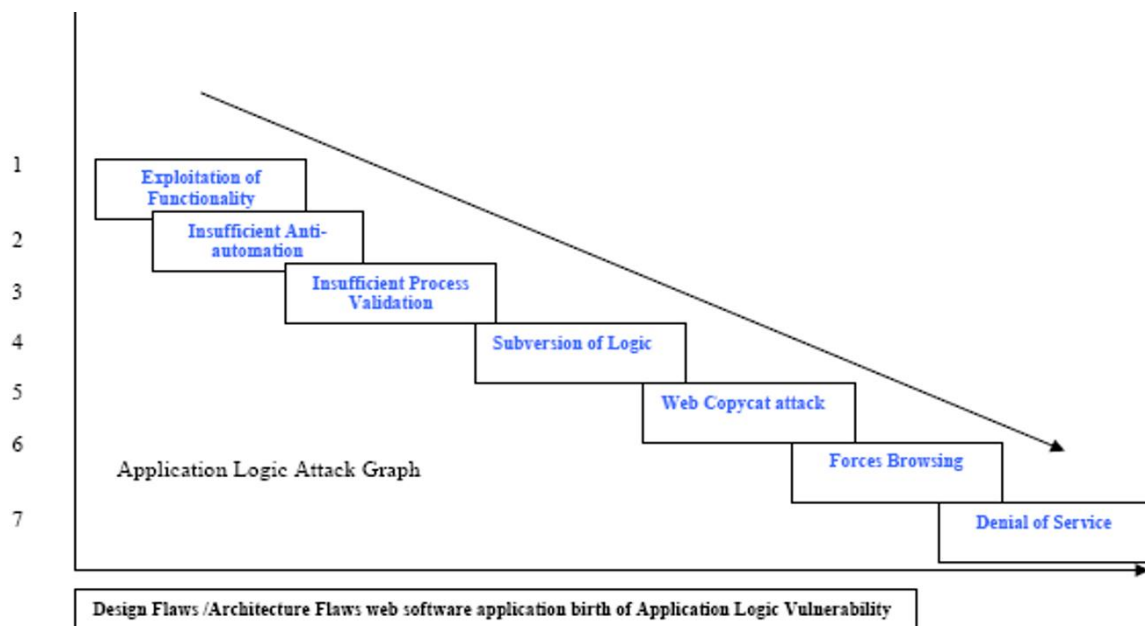


Fig. 4: Application Logic Vulnerability Graph

SVAM based Taxonomy of Logical attack, Types and Classification

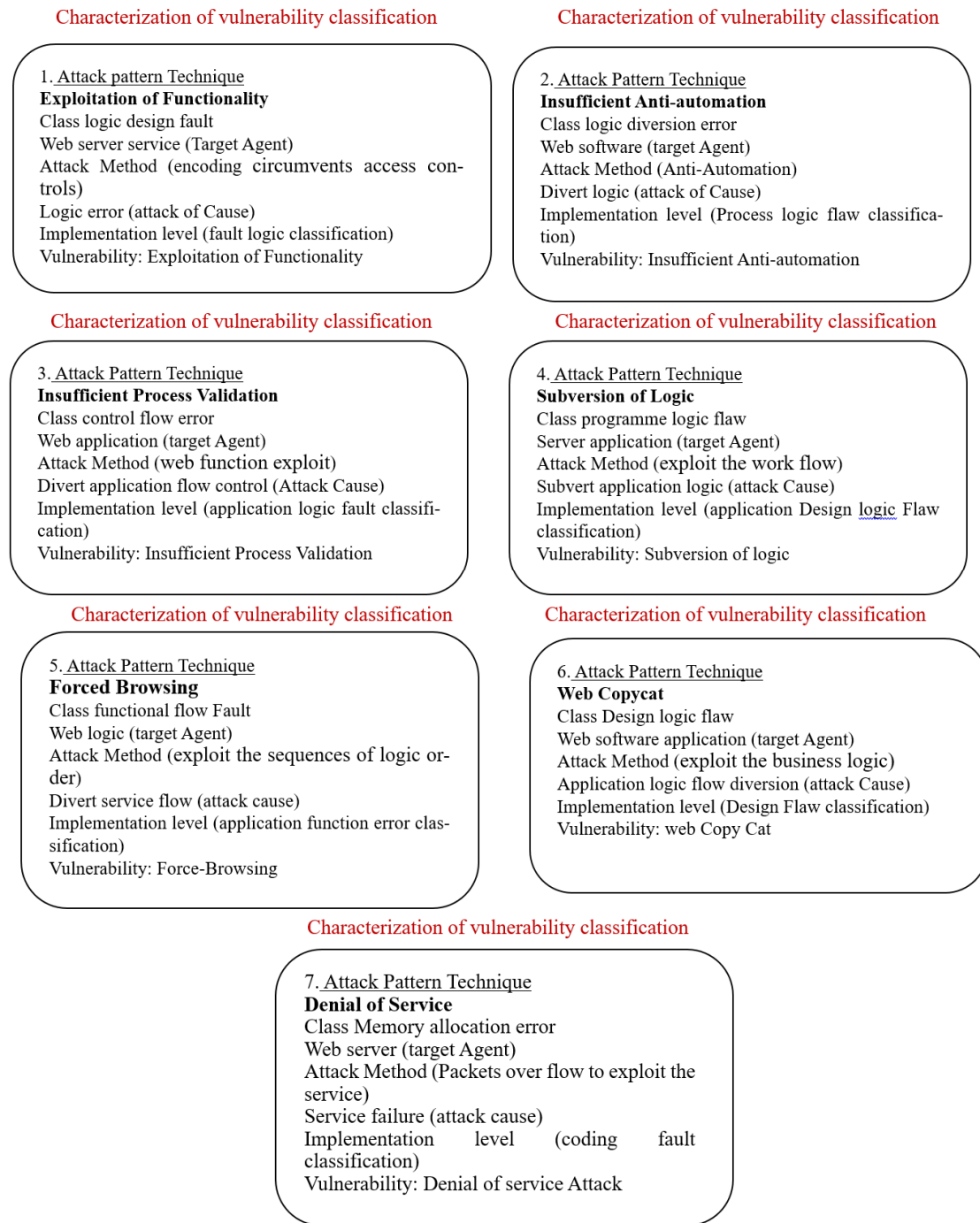


Fig. 5: Characterization of vulnerability

Application Logic of Component

Start reading at First Stage at about 5 km above the surface of Mars, TDM tracks the touchdown legs, one sensor per leg to assess touchdown.

Processing Logic Design

Developer assumed that a known possibility sensor could indicate wrong touchdown signals if-the legs locked in the deployed position. TDM software had to handle this

possible event with a-marking leg that generates a spurious signal with an inappropriate sensor on 2 consecutive sensor readings.

Second Stage

TDM was to track the remainder of the good sensor at around 40 m above the surface. When a sensor had two consecutive Touchdown reading, the TDM programme was instructed to shut down the downwind engine.

What Happened?

One or more of the sensors had 2 consecutive readings in TDM Component Memory before 40 m, leg-sensor information was processed. When MPL crossed the 40 m level, during the first step of descent, TDM changed states and read the storage associated with the leg sensor. Shutdown Engine effect.

Scientific Justification

A developer can design and enforce the requirement in various ways, but the nature of a design failure is that components cause (pre-conditioning, post-condition and invariant) infringements in performing the condition of bad data held by software variables (Chen *et al.*, 2018).

Therefore, it has been shown that the problem is not in implementation logic but in design through the application logic technique related to the logical component and its requirement specification rather than a more functional interface specification integration, which resulted in a design defect in the MPL framework and task. This defect's classification is therefore defined as a design defect, which is a logical defect identified by our vulnerability classification through SVAM (Fig. 1).

Logical vs Technical Vulnerability Classification

In view of our study, we would like to suggest a classification and characterization of the two categories of vulnerability problems/issues mentioned above (Technical Vs Logical Vulnerabilities). These are categorised as stated above in the classification of each weakness on the basis of their attack process (attack pattern technique). Therefore, by retaining the classification of two separate vulnerability types, we have drawn up a classification tree where all sub-class attacks under each vulnerability class are included. A new taxonomy is shown here with a detailed classification and distinguished by its distinctive signature in the application layer of e-commerce systems. As it is stated in Fig. 6.

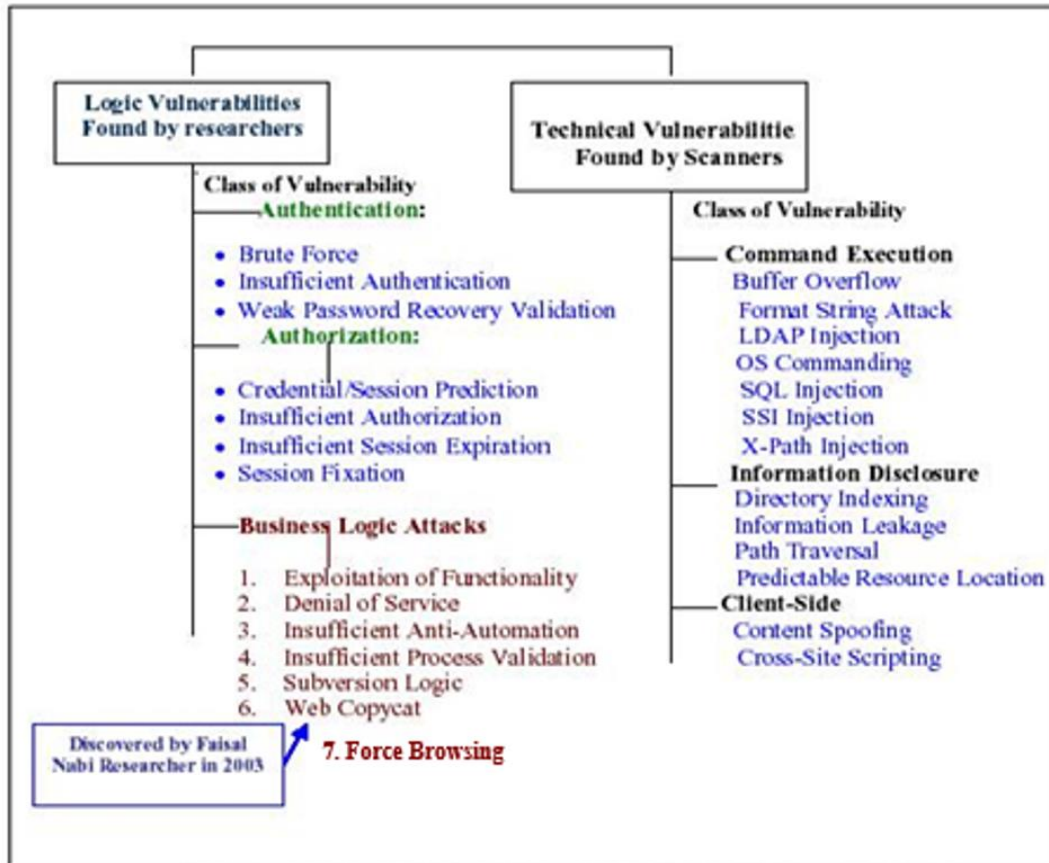


Fig. 6: Logical vulnerabilities Vs technical vulnerabilities

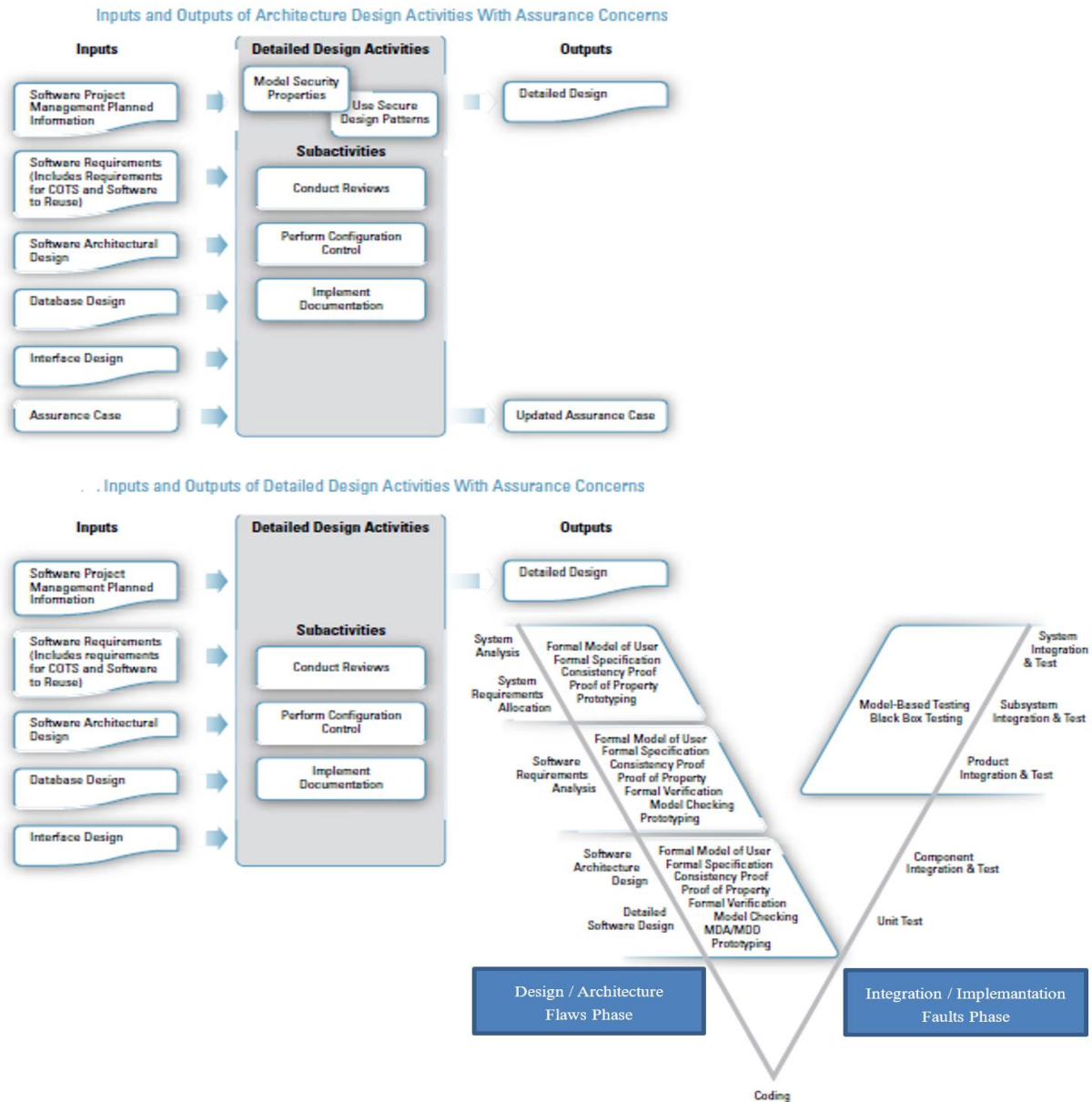


Fig. 7: Vulnerability mitigating in context software design assurance phase process

Mitigation Process in Context SDLC

Attack patterns identify typical methods of software operation. They are derived from a model proposed by the design pattern framework (Li *et al.*, 2017) that clearly shows the stages of two distinct life cycles of vulnerability, as shown in Fig. 7. The concept derived from (Joshi *et al.*, 2015) that one describes design and architecture, another one shows implementation level, each stage shows two separate causes of vulnerability, such as the design phase refers to a design flaw and architectural flaw and flaws, bugs & errors are seen in the implementation phase. By mismatching a collection of components in a system design that allows the sequence of events occurring in the attack pattern, allows the

vulnerability detecting approach is achieved. The proposed model also presents extensive information on all protected system development processes at the design and implementation levels and describes both the two distinct types of vulnerabilities. This helps to understand two distinct life cycles of vulnerability and therefore points out the closeness as stated in the Fig. 7.

Conclusion

For software developers a taxonomy is the footprint for safe system design (Johnson *et al.*, 1995). The approach taken in this article focuses in the characterization and classification of vulnerabilities of

component-based web-e-commerce applications and of logical vulnerabilities. As a result, safety awareness is increased at the outset of the development process by incorporating the proposed approach and procedure into the design phase. Risk management is required to begin early on so that the protection team can evaluate how the application logic has been strengthened. In the component development software model, we also categorised the two separate vulnerabilities and showed the birth of attack designs because of vulnerability at the various phases of the development cycle, which are helpful for developers in the adoption of protection through design technologies during software design.

Acknowledgment

This work is based on a Research in Australia cyber Banking e-commerce security Business logic issues.

Author's Contributions

All authors equally contributed in this work.

Declaration of Interest

- The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this study
- The authors declare the following financial interests/personal relationships, which may be considered as potential competing interests

Ethics Approval

There is no human and animal involved in this research therefore no need of ethical approval for this research

References

Abbott, R. P., Chin, J. S., Donnelley, J. E., Konigsford, W. L., Tokubo, S., & Webb, D. A. (1976). Security analysis and enhancements of computer operating systems. National Bureau of Standards Washingtondc inst for Computer Sciences and Technology. <https://apps.dtic.mil/sti/citations/ADA436876>

Álvarez, G., & Petrović, S. (2003, July). A taxonomy of web attacks. In International Conference on Web Engineering (pp. 295-298). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45068-8_56

ANSI/IEEE. (1990). ANSI/IEEE Standard Glossary of Software Engineering Terminology. IEEE Press. <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=159342>

Aslam, T. (1995). A taxonomy of security faults in the Unix operating system. Master's thesis, Purdue University. [http://cwe.mitre.org/documents/sources/ATaxonomyofSecurityFaultsintheUNIXOperatingSystem\[Aslam95\].pdf](http://cwe.mitre.org/documents/sources/ATaxonomyofSecurityFaultsintheUNIXOperatingSystem[Aslam95].pdf)

Bazaz, A., & Arthur, J. D. (2007, January). Towards a taxonomy of vulnerabilities. In 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07) (pp. 163a-163a). IEEE. doi.org/10.1109/HICSS.2007.566

Bisbey, R., & Hollingsworth, D. (1978). Protection analysis project final report. ISI/RR-78-13, DTIC AD A, 56816. <http://nob.cs.ucdavis.edu/bishop/papers/1999-raid/1999-vulclass/1999-vulclass.html>

Bishop, M. (1995). A taxonomy of UNIX system and network vulnerabilities. Technical Report CSE-95-10, Purdue University. <http://nob.cs.ucdavis.edu/bishop/notes/>

Cebula, J. L., & Young, L. R. (2010). A taxonomy of operational cyber security risks. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst. <http://www.sei.cmu.edu/library/abstracts/reports/10tn028.cfm>

Chen, K., Zhang, S., Li, Z., Zhang, Y., Deng, Q., Ray, S., & Jin, Y. (2018). Internet-of-Things security and vulnerabilities: Taxonomy, challenges and practice. Journal of Hardware and Systems Security, 2(2), 97-110. doi.org/10.1007/s41635-017-0029-7

Chirs, V. B., & Frank, R. J. (2005). A taxonomy methodology applied to web services. Research Report, IBM Zurich Research Laboratory. <https://dominoweb.draco.res.ibm.com/f3f9573a5c7b2db4852570750034edf2.html>

Firesmith, D. G. (2005, August). A taxonomy of security-related requirements. In International Workshop on High Assurance Systems (RHAS'05) (pp. 29-30). <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.6934&rep=rep1&type=pdf>

Glass, R. L., & Vessey, I. (1995). Contemporary application-domain taxonomies. IEEE Software, 12(4), 63-76. <https://doi.org/10.1109/52.391837>

Gray, A. (2003). An historical perspective of software vulnerability management. Information Security Technical Report, 8(4), 34-44. [doi.org/10.1016/S1363-4127\(03\)00005-0](https://doi.org/10.1016/S1363-4127(03)00005-0)

Igure, V. M., & Williams, R. D. (2008). Taxonomies of attacks and vulnerabilities in computer systems. IEEE Communications Surveys & Tutorials, 10(1), 6-19. doi.org/10.1109/COMST.2008.4483667

Jiwnani, K., & Zelkowitz, M. (2004). Susceptibility matrix: A new aid to software auditing. IEEE Security & Privacy, 2(2), 16-21.

- doi.org/10.1109/MSECP.2004.1281240
- Johnson, R., Gamma, E., Vlissides, J., & Helm, R. (1995). Design pattern: Reusable object-oriented software. Addison Wesley.
- Joshi, C., & Singh, U. K. (2014). Admit-A five dimensional approach towards standardization of network and computer attack taxonomies. International Journal of Computer Applications, 100(5), 30-36.
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.678.3355&rep=rep1&type=pdf>
- Joshi, C., Singh, U. K., & Tarey, K. (2015). A review on taxonomies of attacks and vulnerability in computer and network system. International Journal, 5(1), 742-747.
- Kjaerland, M. (2006). A taxonomy and comparison of computer security incidents from the commercial and government sectors. Computers & Security, 25(7), 522-538. doi.org/10.1016/j.cose.2006.08.004
- Krsul, I. V. (1998). Software vulnerability analysis. West Lafayette, IN: Purdue University.
<http://coast.cs.purdue.edu/pub/papers/ivan-krsul/krsul-phd-thesis.pdf>
- Landwehr, C., Bull, A. R., McDermott, P. J., & Choi, S. W. (1993). A taxonomy of computer program security flaw. Technical report, Naval Research Laboratory.
<https://cwe.mitre.org/documents/sources/ATaxonomyofComputerProgramSecurityFlawswithExamples%5BLandwehr93%5D.pdf>
- Li, X., Chen, J., Lin, Z., Zhang, L., Wang, Z., Zhou, M., & Xie, W. (2017, September). A new method to construct the software vulnerability model. In 2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCI) (pp. 225-229). IEEE.
doi.org/10.1109/CIAPP.2017.8167212
- Longley, D., & Shain, M. (1990). The Data and Computer Security Dictionary of Standards. Concepts and Terms.
- Masera, M., Fovino, I. N., & Sgnaolin, R. (2005). A framework for the security assessment of remote control applications of critical infrastructures. In Proceedings of the Twenty-Ninth ESReDA Seminar.
- McPhee, W. S. (1974). Operating system integrity in OS/VS2. IBM System Journal, 13, 230-52.
- Moore, A. P., Ellison, R. J., & Linger, R. C. (2001). Attack modeling for information security and survivability. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.
<https://apps.dtic.mil/sti/citations/ADA388771>
- Nabi, F. (2005). Secure business application logic for e-commerce systems. Computers & Security, 24(3), 208-217. doi.org/10.1016/j.cose.2004.08.008
- Nabi, F. (2011). Designing secure frame work method for e-commerce systems. Journal of Network Security, 12, 29-41.
- Nabi, F., & Nabi, M. M. (2017). A process of security assurance properties unification for application logic. International Journal of Electronics and Information Engineering, 6(1), 40-48.
<http://ijeie.jalaxy.com.tw/contents/ijeie-v6-n1/ijeie-v6-n1.pdf#page=44>
- Pothamsetty, V., & Akyol, B. A. (2004, November). A vulnerability taxonomy for network protocols: Corresponding engineering best practice countermeasures. In International Conference on Communications, Internet and Information Technology, (pp. 168-175), St. Thomas, US Virgin Islands.
https://www.researchgate.net/publication/221425438_A_vulnerability_taxonomy_for_network_protocol_s_Corresponding_engineering_best_practice_countermeasures
- Scott, D., & Angelos, S. (2013). Towards a Cyber Conflict Taxonomy. In: 5th International Conference on Cyber Conflict, (pp. 45-56).
- Simmons, C., Ellis, C., Shiva, S., Dasgupta, D., & Wu, Q. (2009). AVOIDIT: A Cyber Attack Taxonomy. University of Memphis, Technical Report CS-09-003.
- Simpson, G. G. (1945). The principles of classification and a classification of mammals. Bulletin of the American Museum of Natural History, 85. xvi+350.
<http://hdl.handle.net/2246/1104>
- Simpson, G. G. (1961). Principles of animal taxonomy. Columbia University Press, ISBN: 9780231888592.
- Tsipenyuk, K., Chess, B., & McGraw, G. (2005). Seven pernicious kingdoms: A taxonomy of software security errors. IEEE Security & Privacy, 3(6), 81-84.
doi.org/10.1109/MSP.2005.159
- Weber, S., Karger, P. A., & Paradkar, A. (2005). A software flaw taxonomy: Aiming tools at security. ACM SIGSOFT Software Engineering Notes, 30(4), 1-7.
<https://dl.acm.org/doi/abs/10.1145/1082983.1083209>

CHAPTER 5: A NOVEL APPROACH FOR COMPONENT-BASED APPLICATION LOGIC EVENT ATTACK MODELING

Introduction and Findings: Local Relationship between Chapters 4 and 5

Introductory Note: Chapters 4 A and B. These cover and provide the details about a taxonomy which categorizes the group attacking method and classification of two groups of vulnerabilities (Technical vs Logical) in e-commerce component-based applications. Within this, Chapter 5 covers and addresses the event attack modeling scenario in the banking application domain. The proposed approach is based on the event attack modeling technique by using Uppaal Tool to detect the design flaw in the e-commerce component-based application while reusing the design specification of the existing application logic of the system. The research question addressed in this chapter is based on question 3.

Findings: This chapter provides findings on the relationships with the previous two chapters as a sequence in terms of SCA and event attack modeling that is projected through case study-based modeling and projected subversion attack in banking applications while reusing design specification of reused components.

This paper is published in the International Journal of Network Security, First Online Feb. 28, 2020 (VDOI: 1816-3548-2020-00010).

A Novel Approach for Component based Application Logic Event Attack Modeling

Faisal Nabi¹, Jianming Yong¹, and Xiaohui Tao²

(Corresponding author: Faisal Nabi)

School of Management and Enterprise, University of Southern Queensland¹
West St, Darling Heights QLD 4350, Australia

School of Sciences, University of Southern Queensland, Australia²

(Email: u1104061@umail.usq.edu.au)

(Received Aug. 4, 2019; Revised and Accepted Dec. 6, 2019; First Online Feb. 28, 2020)

Abstract

An Event that targets a particular system is required to identify through a novel approach of vulnerability modeling. Current research does not support Event Attack Modeling in component based application logic vulnerabilities. To find such vulnerabilities, it is important to identify the component that triggered the Event to exploit the system. This research proposes the Event Based Attack Modeling, especially in a scenario of component based software subversion logic attack category Business Application Logic. This will help to design and reuse of component from existing application's functional logic.

Keywords: Attack Modeling; CBS reuse; E-Commerce Application; Event Attack Method; Security Modeling

1 Introduction

Event based inter-component applications interact with each other through a passing message inter-communication mechanism [12]. This controlled by a distinct component that is called the event dispatcher, which performs its role as an intermediary between components where conditions are set for the system or application. In this process data communication is called an event that is generated from input communication between components [11]. There are two more type of events, event parameters and event procedures that invoke the individual procedure called the event handlers. In an application, an event attack is occurred when any component of an application is mismatched with its design specification at integration stage. This may result of design fault, because of event-based interruption, which then can create a loophole to exploit the particular system, generated by an attack event during the inter-communication of event parameters [2].

The security vulnerability can arise in the environment that supports the event attack method. The source of the vulnerability can be based on object (component) that is

able to generate the event send without any restriction and can be easily crafted into an event sequence for other objects (components) to circumvent the entire logic [9,27].

Event Interception is a phase of condition in which a victim object is identified and intercept the events destined to it. To be able to intercept the event sent to an object permits the attacker to breach the confidentiality of one direction of object (component) communication within the system [1,2,11,27]. In recent years there have been many application attacks based on logical flaws, such as logic flaw or design faults. There is a specific strategy that is required to deal with logical vulnerabilities, such logical attacks are classified as subversion attack. This attack is occurred because of logical flaw in design component based application and its interfaced based integration fault [4,7,25,27].

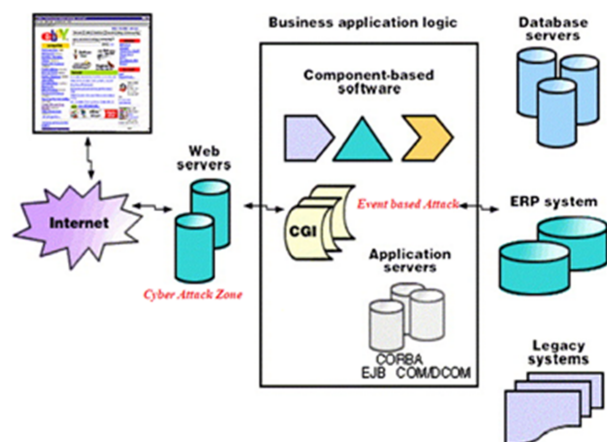


Figure 1: Component based application logic event attack scenario

Therefore, we classify this problem as an Event Attack View. In this case, specification refers to conventional attack, threat, vulnerability. This classifies the attack method, and attack model of identified vulnerability that is known as a subversion attack. In the field of cyber se-

curity Attack Event information is considered as at-tack related data that is derived from various sources. An at-tack event is defined as targeting assets by using attack method, which then exploits the functionality of application business process or circumvents the flow logic. It is very hard to detect the design flaw based vulnerabilities through traditional scanning tools; this is why such vulnerabilities never classified to deal with in terms of the application logic [5, 7].

In this research, we propose Event Based Attack Modeling for design flaw based vulnerability, called Subversion Attack (Component based application logic flaw) by using a Banking case study. The purpose of this re-search is to simplify the process of vulnerability modeling to understand the life cycle of vulnerability. This could help the developers while designing and reusing design specification of business components from existing application components and their underlining application logic. An Event Attack refers to a security problem that exploits the event based inter component communication model [5]. The definition of Event Attack: A malicious component that generates an event of circumvention in order to exploit the target's application logic or functionality. This intercepts communication by forcing the targeted component to send back an inappropriate call or calling away from application functional logic [5].

2 Problem Statement

The focus of this research is to analyze the Event at-tack model and the Subversion attack that falls in the category of business logic vulnerability. Specially considering the security breach scenario real life case study related to Barclay bank, as well as the re-usability design description of component.

The research question, how can Event Attack Modeling simplify the application logic vulnerability, subversion attack? This question is answered by the example of real time case study research method, using Event At-tack Modeling technique.

This real-life case study is a good example of a design flaw in application logic due to the reuse of a component caused component subversion. In this example, the developer reused the same component that was already incorporated in the registration functionality elsewhere within the application, violating the assumptions of the component developer. This mistake lead to the introduction of an application-level flaw that allowed an attacker to access another client's bank accounts. The approach taken to be analyzed, this problem is one that the Event Attack Modeling Technique will be able to helpful to detect design flaws and/or fault free component-based application logic in the middle tier of the n -tier architecture as depicted in Figure 1.

2.1 Research Philosophy

The research philosophy is taken as applied science that is basically an application of existing scientific knowledge to practical applications such as technology, concerning the theory of Event of inter component-communication model. It uses theory, knowledge, method and technique for a particular state of the art [28]. This discussion about Component-based State of the Art in relation to the philosophy of its application & design pattern. The research philosophy also defines and investigates about state of the art technology in Event interaction between the component software de-signs, which is adopted from an applied science philosophy to formulate a solution for business logic vulnerability. In this process, it is very important to understand that design question in the light of research philosophy, can help to conduct the research in the field of Attack Modeling & Security domain by ensuring that research-er's work is going in a right direction and their work is rigorous and insightful.

2.2 Research Gap

In the light of current research and recently studied literature review, [6, 14, 18, 21] and [17] in the domain of cyber and network vulnerability modeling. The research Gap clearly finds an interest to improve the business logic security, specially "Design Flaw" in a service oriented e-commerce applications, that is composed with integrated components. The research gap identified the significance of application logic vulnerability class and category "Subversion attack" cause of Design Flaw, because automated vulnerability analysis and detection tools cannot detect it. This is reason why such vulnerabilities are always oversighted by the application developers. The developers are always keen to reuse existing component core logic from current business logic of the system. This may often cause of mistake while integrating component code solution and designing new functionality.

2.3 Research Design and Method

This research is based on exploratory method where no scientific foundation is available for supporting techniques. The current research and literature review highlights the gap between the current approach and previously designed models or frameworks for logical vulnerabilities. Therefore, we have proposed (Event Attack Modeling) such a technique that could deal with application level logic vulnerabilities. This would help to detect early design faults at the time of integration of components and design fault free new applications. The re-search design also follow previous modeling techniques to justify the newly proposed technique. This simplifies the problem detection process and method.

2.4 Current Approaches in Attack Modeling

There have been several techniques used for vulnerability modeling. These techniques are Attack Graph [26], Attack-Vector [22], Attack-Surface [22], Diamond model [13], OWASP’s threat model [13] and Kill Chain [15]. Each technique has its own properties and speciality to identify and model the attack process path way through out the system and network. For example, Attack Graph technique is used for network related vulnerability and system exploitation modeling based on scenario of security issues. Through this technique one can identify the process and pathway of security breach cause within the network as shown in Figure 2.

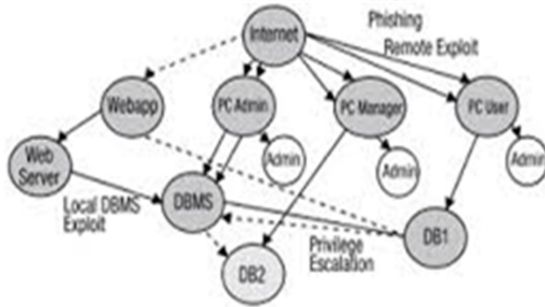


Figure 2: Attack graph with attack path against system

3 Studying Case Profile & Event Attack Modeling

This real life case is a good example of a design flaw in application logic due to the reuse of a component caused component subversion. In this example, the developer reused the same component that was already incorporated in the registration functionality elsewhere within the application, violating the assumptions of the component developer. This mistake leads to the introduction of an application-level flaw that allows an attacker to access another client’s bank accounts (component code Figure 3).

```
class CCustomer
{
    String firstName;
    String lastName;
    CDoB dob;
    CAddress homeAddress;
    long custNumber;
    ... }

```

Figure 3: C customer component code

3.1 Component Application Logic Design Fault

The registration functionality incorporated with the *CCustomer* component that consist of “(use case logic + *Process* and *Entity* Type Logic)” within the application, including core functionality. This process allows the user to authenticate and grant access to the application components such as “My Account component”, “View Balance component”, “Funds transfers component”, “Select Bank Account component, Debit Credit component and other information component. After having authenticated user itself to the application through the registration process, the same Object instantiate and saves in the session key information related to the identity. The components of application within functionally referenced information related to the **CCustomer(Component)** object in order to carry out its actions because the **CCustomer(Component)** object is candidate component (*Process* and *Entity* Type logic) within the majority of application — for example, account details shown on the main page of the user was generated based on the customer unique number that contained within this component. In the way composition or reuse of the component, code was already used within the application. It clearly shows that the developer assumption leads to a flaw in the reuse of application logic design. This caused the birth of a vulnerability to subversion attack on application business logic. It was a serious mistake and subtle to detect and exploit.

3.1.1 Class of Vulnerability

The “*Subversion Attack*” characterization of vulnerability flaw falls under the application logic, and attack method is to exploit the workflow of business logic, this process subvert business process. At implementation level it is classified as design logic flaw, which then finally characterized as “Subversion of logic” attack.

Subversion of logic. Class: Programme logic flaw;
 Server application: (Target agent);
 Attack method: (Exploit the work flow);
 Subvert application logic: (Attack cause);
 Implementation level: (Application design logic flaw classification);
 Vulnerability: Subversion of logic.

Therefore, we modeled the Event oriented subversion life cycle that displays the logic diversion of business logic in a small chain of inter-component based communication application model, caused by CBS Flaw.

The above mentioned Figure 4 displays an event attack model scenario, class is subversion attack that falls under business application logic vulnerability, based on component based software that may be flawed in CBS. This fault may have effects on service calls and flow of the function that depends on event based call to other

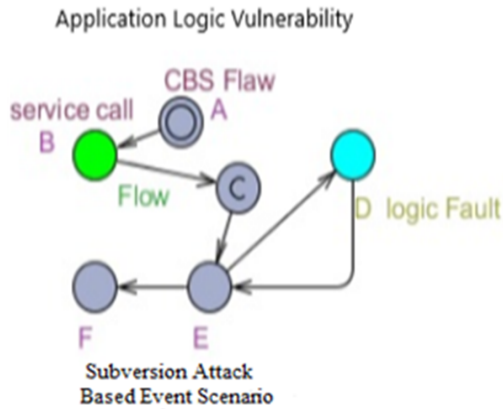


Figure 4: Subversion attack event scenario

objects within the system. As it is shown in the above Figure 4, *C* is condition that must correspond to component *D* before processing to normal application logic flow to proceed the *E*. Therefore, *D* component is a logic fault that does not let the service flow according to normal flow of CBS call service, this is reason why such faults cannot be detected by automated code & system vulnerability scanning tools, and such faults or flaws fall under the classification of logical vulnerability.

3.2 Case Scenario Based Experimental Study

We have further investigated the scenario of this attack keeping in view the above mentioned example related to a security breach of Bank case study. This is caused by a logical design flaw within the system while reusing component from existing application logic. This is called “Subvert Event based Attack” on the banking application. The developers always oversight such attacks on the application’s business logic, even though it is a serious vulnerability. It is hard to detect through code scanning and automated detection tools. Therefore, such a technique is required that could simplify the projection of this vulnerability, through the approach of Event based Attack modeling. The proposed technique seems to be a new and effective technique for early detection of such attack at design level of application.

The above-mentioned Figure 5 displays the complete life cycle of the Event Attack Model. In the model *C* indicates to a condition, If **sign-in**, Pass log in to **My Account** Condition to allow access into the system, **Else** Failed **sign in**. This is the general case of scenario system logic for sign in. However, the major mistake is done by the application developer of the banking system reused same component that was already incorporated in the registration functionality elsewhere within the application. This mistake causing subversion of logic and by pass the condition that is set on **My Account (component)** this violated the assumptions of the component developer and caused the system under attack. This attack also subvert

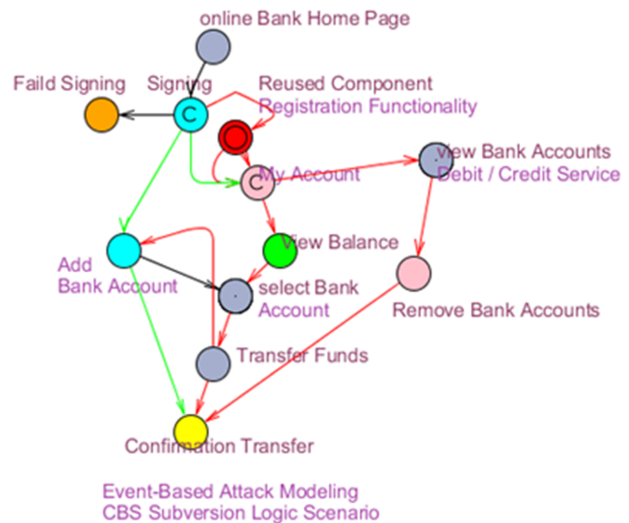


Figure 5: Event attack subversion logic scenario

the other components of the application service flow as shown in Figure 5. Any intrusion detection tool cannot detect this sort of attack known as a class of application logic subversion attack. Therefore security scanning automated software, fail to discovery and un-automate this class of vulnerability. The reused component in the application is spotted in **Red Color**, which reflects the service flow diversion and allow an Event to trigger a logical attack by passing session and controls security mechanism of an application related to other service components as displayed in Figure 5. Therefore, above model cycle of an attack is modeled through a Event Attack Modeling technique in scenario of Component-based Software subversion logic Fault.

3.3 Theoretical Analysis of Proposed Approach

In the light of cyber attack theory a successful attack relies on information to be processed by attacker, in case of when an attack is underway and it is measured by modifying as a result related to attack. Therefore, information is a most important element of any cyber at-tack theory [25].

As, it is confirmed that in the theory of cyber attack, first attack is defined and then attacker knowledge related to information parameters and configuration parameters are derived in order to mitigate the system from potential damage [10].

Therefore we formalized the theory of cyber attack into proposed approach event attack modeling. In this process, first identified the attacker and then measured the attack information parameters, through that an event is occurred as a fault logic, service component triggered to flow diversion and allow an Event trigger by passing session and controls security mechanism. This is demonstrated through scenario based event attack modeling Figure 5 that helped to diagnose the vulnerability life-cycle. This



Figure 6: Cyber attack theory model

gives the knowledge related to information attack parameters, and component configuration parameters that decides the attack vector related to vulnerability of application logic class (subversion logic attack).

Therefore, it is concluded that above mentioned technique is very useful for attack modeling in the light of cyber attack theory.

3.4 Systematical Comparison of the Proposed Scheme

The current approaches of attack modeling are based on attack graph and vector modeling techniques [22, 26], these techniques models focus on the network or system vulnerability based modeling that deals with the different attacks targeting the network [10], but the lack of software application scenario based modeling. In this, scenario an approach is immanent for application based vulnerability modeling technique. Therefore, the proposed scheme is presented, event based attack modeling that targets the service component triggered to flow diversion of application logic in component-based system. The proposed scheme is comparably sounder as compare to any other modeling technique for software based application and its core logic flow.

3.5 Discussion

We have seen that the proposed technique is very helpful in detecting the event that triggered the subversion attack within the application and its component at the integration level, which clearly depicts the vulnerability and its effects on other components of the application and underlying business logic. We also have evaluated the other techniques such as Attack Graph and Attack Vector. The Attack Graph is use to identify the vulnerability in the networks and system, and Attack Vector can provide the path way projection through hacker exploitation attempt which targets the network servers by payload or malicious input. It is also modeled through Attack Vector Modeling technique. It has been noticed that none of these techniques meet the requirement of logical attack modeling and simulation [18].

Where as proposed technique is useful to model the case scenario of banking application through Event Based

Attack modeling. That is spotted in red color the component with fault service flow, calling *C* condition **My Account** component within the application that cause exploitation.

4 Related Work

There are numbers of approaches target the security in event based inter-component applications [3, 19, 23, 24]. For example, Simeon *et al.* [27] took into account the security vulnerabilities in event-based applications and systems, explained the conditions that can be made of them, in result of inter-communication fault. In simple term, current security solutions more rely on encryption, static code analysis, and runtime ACL techniques. Whereas, on the other hand, there have been many techniques adopted to attack modeling such as the Diamond Model [13], Attack Tree [20], Attack Vector [22], Attack Surface [16], Kill Chain [15] and Attack Graph [26]. However, all of these techniques fail to address the logical vulnerabilities detection or modeling framework, because these techniques are network vulnerability modeling and address the network security issues related to the system. Therefore, such a technique needs to introduce that can deal with missing gap between application and system level vulnerability modeling. This will fill the research gap related to logical vulnerabilities in application logic (Component-based Software) [8].

5 Conclusions

Attack modeling is a most useful technique in analysing the attacks and early mitigation of the problem. This is why many techniques are introduced to deal with the attack modeling in the system network domain. The logical vulnerabilities are flaw in design or fault in logic. It is hard to detect and modeled. Therefore such a technique is required that could deal with the logical flaw based vulnerability. In this paper, we have introduced a novel approach of modeling called "Event Attack Modeling" that used Uppaal Tool to model the vulnerability and its attack flow through attack-triggered component within the application in real time scenario. This will help the developers design their application free from logical flaws and design faults, while reusing design specification of component from existing application.

References

- [1] A. A. Al-khatib, W. A. Hammood, "Mobile malware and defending systems: Comparison study," *International Journal of Electronics and Information Engineering*, vol. 6, no. 2, pp. 116–123, 2017.
- [2] H. Al-Mohannadi, Q. Mirza, A. Namanya, I. Awan, A. Cullen, J. Disso, "Cyber-attack modeling analysis techniques: An overview," *The 4th International*

- Conference on Future Internet of Things and Cloud Workshops*, 2016. DOI: 10.1109/W-FiCloud.2016.29.
- [3] L. Aniello, R. Baldoni, C. Ciccotelli, G. A. D. Luna, F. Frontali, and L. Querzoni, "The overlay scan attack: Inferring topologies of distributed pub/sub systems through broker saturation," in *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems (DEBS'14)*, pp. 107–117, 2014.
 - [4] A. Anurag, "Network neutrality: Developing business model and evidence based net neutrality regulation," *International Journal of Electronics and Information Engineering*, vol. 3, no. 1, pp. 1–9, 2015.
 - [5] Bank of England, "An introduction to cyber threat modelling", Industry report, Bank of England Publication, 2016. (<https://www.cyentia.com/library-item/an-introduction-to-cyber-threat-modelling/>)
 - [6] M. Bentounsi, S. Benbernou, M. J. Atallah, "Security-aware business process as a service by hiding provenance," *Computer Standards & Interfaces*, vol. 44, pp. 220–233, 2016.
 - [7] BSIMM, "Attack models with bsimm frameworks," 2016. (<https://www.bsimm.com/framework/intelligence/attack-models/>)
 - [8] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues in Information Warfare & Security Research*, vol. 1, pp. 80, 2011.
 - [9] S. Islam, H. Ali, A. Habib, N. Nobi, M. Alam, and D. Hossain, "Threat minimization by design and deployment of secured networking model," *International Journal of Electronics and Information Engineering*, vol. 8, no. 2, pp. 135–144, 2018.
 - [10] S. Jajodia and S. Noel, *Advanced Cyber Attack Modeling, Analysis, and Visualization*, George Mason University, Mar. 2010. (https://csis.gmu.edu/noel/pubs/2009_AFRL.pdf)
 - [11] N. J. Kim, M. S. Gong, G. S. Lee, "An attack-target-method schema for cyber attack event database," *IEEE International Conference on Electronic Information and Communication Technology (ICE-ICT'16)*, 2016. DOI: 10.1109/ICEICT.2016.7879705.
 - [12] Y. K. Lee, D. Nam, N. Medvidovic, *Identifying Inter-Component Communication Vulnerabilities in Event-based Systems*, Technical Report: USC-CSSE-17-801, 2016.
 - [13] X. Lin, P. Zavorsky, R. Ruhl, and D. Lindskog, "Threat modeling for CSRF attacks," in *IEEE 16th International Conference on Computational Science and Engineering*, vol. 3, pp. 486–491, 2009.
 - [14] A. K. Luhach, S. K. Dwivedi, C. K. Jha, "Designing and implementing the logical security framework for e-commerce based on service oriented architecture," *International Journal on Soft Computing (IJSC'14)*, vol. 5, no. 2, 2014.
 - [15] P. K. Manadhata, J. M. Wing, "An attack surface metric," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 371–386, 2011.
 - [16] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, and E. Weippl, "Dark clouds on the horizon: Using cloud storage as attack vector and online slack space," in *USENIX Security Symposium*, pp. 65–76, 2011.
 - [17] F. Nabi, "Designing a framework method for secure business application logic integrity in e-commerce systems," *International Journal of Network Security*, vol. 12, no. 1, pp. 29–41, Jan. 2011
 - [18] F. Nabi and M. M. Nabi, "A process of security assurance properties unification for application logic," *International Journal of Electronics and Information Engineering*, vol. 6, no. 1, pp. 40–48, 2017.
 - [19] F. Petroni, L. Querzoni, R. Beraldi, and M. Paolucci, "Exploiting user feedback for online filtering in event-based systems," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing (SAC'16)*, pp. 2021–2026, 2016.
 - [20] C. Phillips and L. P. Swiler, "A graph-based system for network-vulnerability analysis," in *Proceedings of the 1998 Workshop on New Security Paradigms*, pp. 71–79, 1998.
 - [21] L. Seinturier, P. Merle, R. Rouvoy, D. Romero, V. Schiavoni, and J. B. Stefani, "A componentbased middleware platform for reconfigurable service-oriented architectures," *Software Practice and Experience*, vol. 42, no. 5, pp. 559–583, 2017.
 - [22] B. Schneier, "Attack trees," *Dr. Dobb's Journal*, vol. 24, no. 12, pp. 21–29, 1999.
 - [23] B. Shand, P. Pietzuch, I. Papagiannis, K. Moody, M. Migliavacca, D. Eyers, and J. Bacon, "Security policy and information sharing in distributed event-based systems," *Reasoning in Event-Based Distributed Systems*, pp. 151–172, 2011.
 - [24] M. Srivatsa, L. Liu, and A. Iyengar, "Event-guard: A system architecture for securing publish-subscribe networks," *ACM Transactions on Computer Systems (TOCS'11)*, vol. 29, no. 4, pp. 10:1–10:40, Dec. 2011.
 - [25] A. Tayal, N. Mishra and S. Sharma, "Active monitoring & postmortem forensic analysis of network threats: A survey," *International Journal of Electronics and Information Engineering*, vol. 6, no. 1, pp. 49–59, 2017.
 - [26] United States. Joint Chiefs of Staff, *Joint Tactics, Techniques, and Procedures for Joint Intelligence Preparation of the Battlespace*, 2000. (<http://pur1.access.gpo.gov/GPO/LPS49610>)
 - [27] S. (simos) Xenitellis, "Security vulnerabilities in event driven systems," in *Proceedings, Security in the Information Society: Visions and Perspectives*, pp. 147–160, 2001.
 - [28] A. Yaghmaie, "How to characterise pure and applied science," *International Studies in the Philosophy of Science*, vol. 31, no. 2, pp. 133–149, 2017.

Biography

Faisal Nabi is a PhD researcher at University of Southern Queensland. He has also received Honorary PhD in Computer Science from Brock University St. Catharines, Ontario, Canada. Faisal's research interests are e-commerce security and software security.

Jianming Yong is Professor of school of information systems. He has received his PhD from SwinburneUT. He is also member of IEEE professional. His research areas are Cloud Computing, Big Data Security and Privacy, Data Integration, Workflow systems, Information system security, Network management, Web service for SMEs, Digital Identity Management.

Xiaohui Tao is Associate Professor in School of Sci-

ences, University of Southern Queensland, Australia. His research interests include Natural Language Processing, Text Mining, Knowledge Engineering, and Health Informatics. During his research career, Tao has gained a wealth of knowledge and experience in dealing with massive datasets and delivering solution to complex research problems, and made many contributions to Ontology Learning, Web Intelligence, Data Mining, and Information Retrieval. His research results have been published in 90+ refereed papers, many of them are on highly ranked journals such as IEEE TKDE, KBS, PRL and conferences such as ICDE, PAKDD and CIKM. He has been a Program Chair of many International Conferences and Workshops.

CHAPTER 6: SECURITY ASPECTS IN MODERN SERVICE COMPONENT-ORIENTED APPLICATION LOGIC FOR SOCIAL E- COMMERCE SYSTEMS

Introductory Note and Findings: Relationship between Chapter 5 and Chapter 6

Introductory Note: Chapter 5. covers and addresses the Event attack modeling in the scenario of the banking application domain. The proposed approach is based on the event-attack modeling technique by using Uppaal Tool to detect the design flaw in the e-commerce component-based application, while reusing the design specification of the existing application logic of the system. Chapter 6 reveals a detailed case study based problem solution in the context of social e-commerce that is used as a tool to close down the subject of targeted audience in relation to banking case study and proposed methodology that is justified through modeling technique related to application logic. The research sub-question addressed in this Chapter is Question 4. This defines logical relationships while discussing the major problem in social -e-banking, which is the main focus of this thesis.

Findings This Chapter provides a comprehensive solution as a finding that linked with previous chapter findings that social e commerce based application logic security through UML modeling and validated the research work.

This Paper was published in the Journal of Social Network Analysis and Mining (2021) 11:22 <https://doi.org/10.1007/s13278-020-00717-9> Springer-Verlag GmbH, AT part of Springer Nature 2021.



Security aspects in modern service component-oriented application logic for social e-commerce systems

Faisal Nabi¹ · Xiaohui Tao¹ · Jianming Yong¹

Received: 20 October 2020 / Revised: 3 December 2020 / Accepted: 7 December 2020 / Published online: 16 February 2021
© The Author(s), under exclusive licence to Springer-Verlag GmbH, AT part of Springer Nature 2021

Abstract

Modern practices in social commerce are a subset of e-Commerce focusing on security framework protocols such as secure transactional protocols, cryptographic schemes, and sanitization criteria. It is assumed that these practices will ensure stable social media-based e-Commerce applications. The main concern in utilizing these practices focus on software component composition, and integration flaws, which are often overlooked in their business application logic. These problems can render the effect of modern information security concepts null and void. The weakest link in social media-based e-Commerce applications is the component's logic subversion on its server side, which is caused by developers overlooking the design process. This paper addresses a unique issue in aspects of information security in application logic vulnerability called subversion attack, which can be classified as a design flaw. This kind of security flaw cannot be prevented by many traditional security mechanisms commonly used in modern e-Commerce systems. To address this issue, we propose the use of security assurance methodologies in service component-oriented applications to be utilized through threat modeling and a novel technique component fault detection model. This idea is further extended to the modeling component and its applications using a UML secure design approach. To validate the technique, the methods applied in this paper are verification and validation for security by design testing to avoid the business logic design flaw problem in rapidly built component-based social media e-Commerce applications.

Keywords Design flaws · Subversion attack · Social media-based e-commerce system · Service component architecture · Assurance & security · UML-based modeling · Business logic attacks

1 Introduction

Security and privacy issues in the field of social media-based e-Commerce are important topics for debate among the users concerned. E-Commerce is one part of the information system architecture business model and its use has become increasingly common. However, the users may find themselves somewhat unwilling to suffer from risks to their security and privacy. In the banking industry, social media-based e-Commerce has prompted a new age of information

security. However, banking via e-Commerce is unfortunately hindered by the risks associated with these issues. There is also no trust in the customer, and no visitor shop on the website, and these sites will not function if these privacy and security risks are not removed. The social, organisational, technological, and economic perspectives of these two problems, namely security and privacy (Raed and Nripendra 2020; Wang et al. 2020). Service -oriented applications in the social e-banking domain are developed with well-defined, readily available software components. These are the building blocks used to develop the services in service component architecture. It is important for the design process of service-oriented applications in the social e-banking domain to follow an order where once the design process of an application is determined as an organizational task, and then it is considered as modular components that need to be integrated. Each modular component is a service explaining its interfaces through the integration of these services in a social e-banking system. Having

✉ Faisal Nabi
faisal.nabi@usq.edu.au

Xiaohui Tao
Xiaohui.tao@usq.edu.au

Jianming Yong
jianming.yong@usq.edu.au

¹ School of Business and School of Science, University of Southern Queensland, Toowoomba, Australia

identified modular components, new business processes depend on the integration of components (Nabi et al. 2020). It can be clearly understood by researchers and developers that system composition and integration are the most complicated processes in software design, while devising complex operations such as the integration process into the social e-banking system. The complications that occur in the integration of these e-banking systems may cause a process of continuous changes in the technical and business attributes offered by the bank to its customers in order to fulfil their needs (Nabi et al. 2019a, b). The social e-banking systems are often designed and based on various component vendors, consisting of different platforms and design/architecture patterns. The continuous changes may cause extra complexity during service-oriented component-based banking application integration because design flaws in the integration process may allow business logic attacks. This is the main reason that subverting the operation of web 4.0 social commerce-based banking application processes can cause serious financial damage (Nabi et al. 2020). The focus of this research is to investigate the vulnerability of business logic in service component-oriented applications using a security breach scenario real-life case study related to social commerce-based e-banking. This real-life case study is a good example of a design flaw in application logic due to the reuse of a component causing component subversion. In this example, the developer reused the same component that was already incorporated in the registration functionality elsewhere within the application and violated the assumptions of the component developer. This mistake led to the introduction of an application-level flaw that allowed an attacker to access other clients' bank accounts. Therefore, a research question is raised here: How can the developer test if a Web 4.0 social e-Commerce e-banking application service contains logical flaws in component integration design specification? At the same time, service component architecture should be considered. Service component architecture is a solution in modern social e-banking, especially when it reuses components while designing new applications as part of social media distributed systems, such as online web 4.0 social commerce-based e-banking application services. However, security aspects are not generally considered at a low level when interacting with these services, leading to possible unauthorized access to the service. Another issue to be considered is that sometimes, reuse of service logic may cause failure. A good network defence perimeter in modern systems, such as using firewalls, honey pot, intrusion detection and other network security components should ensure that legitimate users can access the application while, at the same time, preventing illegitimate users from gaining the opportunity to attack the systems through the abuse of vulnerable social media business processes (Ghassan et al. 2020). The system logic and business process are based on

two components: business logic and flow of information (Nabi and Nabi 2017). A large majority of researchers have focused on information flow as a way to build an approach to business process security. However, issues related to e-business protection in the approaches fail to address the logic of a component's business processing during the design stage, especially when components are developed based on their business logic (Agirre et al. 2018).

In this paper, we have discussed the security aspects of challenges related to design flaws that can cause subversion attacks on component-based application logic, in n-tier applications. The paradigm of security by design technique draws attention to the complexity of the security. Security must not only be addressed retroactively by identifying and fixing security loop-holes, but must also be considered at an early stage of social commerce e-banking system development.

1.1 Problem statement

Social media is an increasing field of research activity in organizations. Social commerce is a subset of e-Commerce. Social media can be described as a phenomenon where; *"a collection of Internet-based applications, based upon the ideological and technological fundamentals of Web 4.0, which allow user generated content to be created and shared."* These applications can change how security in social commerce works by communicating, working with, interacting and exchanging information such as internet social commerce and e-banking services (Alalwan et al. 2018). This paper will investigate and discuss the security aspects of challenges related to design flaw-based subversion attacks that cause reuse design specification while developing new services from existing service component integration SOI methods, causing business logic vulnerability in the middle-tier of the social commerce e-banking architecture. Therefore, there is clearly a need for a methodology to deal with the logical flaws that normally do not show attack patterns or signatures, which is the reason it is hard to discover them through automated techniques. More recently, practices of future generation e-Commerce system development for social media, techniques to secure service-oriented applications, have mainly focused on technical vulnerabilities, which can consist of security analysis and detection tools for vulnerability identification. However, security analysis encompasses a very limited approach for service component-oriented application design and implementation (Nabi et al. 2019a, b).

1.2 Objective and contribution

This paper provides three key contributions: The first contribution is the solution to the problem of system

integration-based component reuse from existing logic. For this we have used a model-based engineering approach. The second contribution focuses on how problems in a quick design approach based on the service component or Module specification misanalysis from the logical function in future generation social e-Commerce system can lead to a security breach at the system design logic stage. This problem is illustrated by the definition of a case study involving a social commerce-based e-banking application based on a reused component. A novel approach is developed for the security assurance methodology for service component-oriented applications. The main points of this novel approach are: (1) The security risk analysis model; (2) Threat modeling; (3) The novel component fault detection technique that detects the faults in component-based design and application; (4) A UML-based modeling component and its application using a UML secure design approach. The third contribution is a novel technique of validity system integration testing (verification and validation method for security by design testing) to avoid the business logic problem customizing security assurance requirements from modern approaches for component-based applications, thus helping to detect design flaw problems cause by subversion of attack.

2 Related work

According to Abdulrahman et al. (2017), in recent decades, Internet technology has grown rapidly, having become an important element in almost every social media-based business. The social e-banking industry is one of the most important developments. In the banking industry, social e-banking is a new business model in which fixed costs of operation are reduced through the provision of unbroken banking services (Abdulrahman et al. 2017). The number of social e-banking applications in companies used by internet users is expected to rise dramatically (Laukkanen et al. 2018). Banks compete by social commerce-based e-banking to increase customer loyalty, acquire greater market share, improve services and offer value-added services, increase efficiency, and reduce operating costs. At the same time, they also face security and privacy issues customer relationship related matters.

According to Jiang et al. (2018) data security is a primary concern for both consumers and companies with the significant success of the social commerce internet trade. While the generalization of data can provide substantial protection of the privacy of even a person, over-generalized data can make the data worth little to no value. In this research, researchers have developed techniques of generalization to optimize data usability and to minimize privacy disclosure (Jiang et al. (2018); Wang et al. (2020)). Developers suggest a privacy-aware access management model for web services in social media environments based upon the fact that the permissible

degree of generalization leads to much more fine-tuned levels of access monitoring. It also discusses how a trustworthy a decision to handle a legitimate access mechanism is made, and how access management policies are continuing (Jiang et al. (2018); Wang et al. (2020)). Comprehensive experiments with both real-world and synthetic datasets illustrate the realistic and efficient privacy control model proposed.

According to Raed and Nripendra (2020), the aim of this study is to build on the understanding and influence of social exchange on the emerging social commerce. Previous research on social media indicated that certain mechanisms such as social commerce structures, social support, social presence, confidence, flow interactions and group interaction, should have a relationship with the online community. However, only some of the above-mentioned structures have already been used. This research has developed a conceptual model that considers structures from a range of theories. The results showed that building social commerce influences positive social support, confidence and social presence among community members. They also found that security and privacy are important concerns in a social commerce e-banking service, which needs to develop through new software trends (component-based software) (Raed and Nripendra 2020).

According to Nabi et al. (2020), the words ‘service component’ come from the service component architecture for distributed system design based on events. While the pattern for service components provides composite application development and reusability support. However, event-based communication in the interaction model of components was explored most within the upper SCA layer when designing logic for service-oriented application components. In this area, a robust safety evaluation is needed that could be used. This layer is called an application process logic layer that produces the application rendering logic and authenticates it from ACL (Nabi et al. 2020). Experiencing problems in composite application and event-based attack in the model architecture of the service component refers to event-based attack in application logic. This goal is accomplished by evaluating and reviewing security problems, modelling techniques of the application functionality of service components, and modeling applications that create, consume and process events.

In agreement with the methods of Zhang et al. (2018), this paper addresses a distributed approach that enables an effective and trustworthy composition of service with safe sensor network data transmission. The computation trust and data trust rules are suggested based on a model of a multi-level trust by evaluating relationships of dependence. An independent model-checker can then evaluate each target component operation. In addition, in a composite evaluation, an identity-based aggregate signature is added to ensure safe data transmission between different components. Results have shown that the

approach not only delivers efficient, reliable composition of service with complex invocation structures, but also lowers the cost of safe data transfer (Zhang et al. 2018). This can be considered as social commerce secure application service support but also limited in its approach.

According to Nabi et al. (2021), recent developments in the field of e-Commerce-based social media software technologies have brought many benefits; at the same time, however, design processes often lead to a variety of different issues, from the design phase to the implementation phase. Software faults and defects increase the problems with reliability and protection and, for these reasons, a solution for these issues is required. This paper addresses the issues of the lack of consistent classification of logical vulnerabilities in application logic relevant to components-based Web applications. The primary way to resolve these issues is to define the Group Attacking Method by categorizing two distinct forms of web-based component vulnerabilities. This research helps explain the creation of applications for social commerce through integration based on components and avoids design vulnerabilities (Nabi et al. 2021).

Seinturier et al. (2017) explain that “New Business Process” is supported by reuse of services, whether it is a business component service or an existing service of component base solution, aligning IT with business functions. Reuse of services increases the chances of active solutions by amalgamating new business process from existing services (Seinturier et al. 2017).

Current security practices are most likely to rely on traditional methods of security, which are based on a socket secure layer or a transport security layer and IDS, so deployment of these security methods targets network security problems and related issues (Nabi and Nabi 2017). Therefore, such practices are known as security functional techniques, but lack security assurance methods. Faisal also highlighted that traditional software engineering practices are not sufficiently up to date to analyze their vulnerability, including penetration of white Box and Black Box testing (Nabi and Nabi 2017). The authors further explained that traditional Intrusion detection tools or vulnerability analysis tools have often failed to detect design flaws (referred to as business logic vulnerability) through traditional security methods in social media system.

Therefore, in the light of the above-mentioned literature review and research, it is clear that there is a gap focus on improving the business logic security (Design Flaw), comprising integrated components.

3 Case study-based research method

A case study-based research plan is also called an exploratory research case for experimentation (Yaghmaie 2017). The exploratory research case study investigates a

well-defined phenomenon (business logic vulnerability) classified by a scientific detailed research formulated event-based attack modeling approach for test generation that can be tested within the research environment using an exploratory case study method. This sort of case study is very often applied as an exploratory research design in a completely new field of scientific investigation (Yin, 2016). In the light of the given explanation of exploratory case study method, the research design will be further extended by exploring a real-life case study. From that study, we take out the test design for a service-oriented social commerce Web-based banking system. The test design will follow further stages; for example, the component integration strategies at run time will be compared to its requirements and design specifications (as can be seen in Figs. 7 and 8), while modelling the component and its application. A further step is to considering attack event scenarios, which is based on connections between business components and their work flow, to analyze the security risk related to cases (as shown in Fig. 2). Therefore, considering the practice of component secure methods, multi-tier specification scenario modelling will be practiced using UML sec 2.0. The technique is further divided into 2 phases. The first phase represents a system tier, while the second phase represents a component tier. The first phase focuses on the design product, while the second phase considers the design, test and distinctive symptom specification for individual components that participate in system development (as projected in Fig. 9). The next phase is to consider the security assurance approach that helps to detect integration-based logical flaws from the system that is proved through the validation and verification (V&V) security assurance process. This process is followed by the V & V security design testing model that is theoretically justified by the designed model for security by the design testing technique. This model will validate the proposed solution called the Security by Design approach for component-oriented service and its application.

3.1 The impact of flaws in application business logic

The complexity risks in social commerce web applications are caused by the fact that developers may introduce flaws that can easily be manipulated by intruders. The key problem that is noticed most in web applications is the integration of various diverse components from a different source, such as custom-built-purpose applications and COTS components, including third party Vander products (Agirre et al. (2016). The major issue related to integration of these components is that they are not easy to reuse, which may cause application-level flaws. The presentation of this problem is more than code bugs like Buffer Overflow and is linked with business logic, for example, the right time for an authentication of canceled policies to enforce the overall security policy of the

application (Jones and Ashenden 2005). Examples like flaws that allow fraud to be committed for personal enrichment through their exploitation are shown in examples from the literature (Nabi and Nabi 2017). It is also noticed that most of the academic focus tends to be low level coding problems, even though these types of high-level software flaws account for 50% of all security software flaws.

The social commerce web application vulnerability may be described as the vulnerability of web application software, which involves mismatch of application/design software and environmental assumptions made while developing/implementing (code writing), running a system, and running the environment'. For example, the designers of a component may have built it with the assumption that all accesses are authenticated but that they can be reused in a context where pre-authentication takes place, and this reasoning is presumed to have taken place within a component. This is a simple example, but our presented case of design flaw issues of reusing components illustrates the condition when the user of the component makes assumptions that were not implemented by the developer (Xhafa et al. 2010).

3.2 Case study-based research scope

This research has significant scope in the field of component-based social commerce e-banking applications and security architecture in modern system generation, especially in the middle tier where application logic security is an important factor while using SCA design patterns. The violation in the middle tier is of real concern, based on the mismatch of component design specification with respect to the existing logic of the banking application, while re-using the component to build a new service, causing the subversion attack. The attack indicates a serious violation of application integrity and security. Service oriented component software uses two sorts of components to develop Web-based social commerce e-banking application logic. These two components are Custom-Developed/COTS. It is possible that they will have flaws in design or its software application. The CBS-based solution has caused software risks that lead to logical vulnerabilities such as the components' logic subversion attack, misuse of application logic and circumventing the components' business logic flow. All of these temper the functional steps of the flow of application.

In light of the research problems in Sect. 4, the research will focus on a security breach subversion of attack scenario (case study). This relates to the web 4.0 social commerce e-banking systems that reuses design specification while developing new services from existing service component integration SOI methods, causing business logic vulnerabilities in the middle-tier of the social commerce e-banking architecture.

4 A practical example: social commerce-based e-banking case study

This real-life case of web 4.0 social commerce e-banking is an excellent example of a design defect in software logic due to the reuse of a component design specification that causes subversion elements in the application logic. In this case, the developer reused the same features of a component code that had already been implemented elsewhere in the application for the registration functionality and violated the developer's presumptions. This error led to an application level defect that permitted an attacker to access bank accounts of other customers.

4.1 A composite application functionality and business process

The application encourages its current customers, as well as those who have not already registered for online applications. For this purpose, new users are required to provide basic personal information, such as their name, address and date of birth but not any secret information like Passwords/PINs, which would ensure a degree of assurance of their identity. When this information is processed, the application sends back a registration request to be processed. Upon successful registration, an information pack is mailed to the registered user's home address. This pack provides information about the online activation access via a telephone call to the company's call center as well as the use of a onetime password to log into the system.

The designer of the application believed that this mechanism would protect the system from unauthorized access to the application. The process of security is implemented in the three way of protection mechanism.

1. At the initial stage, some modest amount of personal data is required in defense, to judge a malicious attacker or troublesome user to attempt the beginning registration process behalf of other users.
2. During this process, a key secret transmitted out of hand the registered customer's home address. The attacker needs to have access to the victim's personal mail.
3. In order to authenticate himself, the customer is required to phone a call center in the normal way, based on personal information and selected digits from a PIN. The design is supposed to be well defended but the logical flaw is in the actual implementation of design mechanism. The personal data to be stored is based on being able to correlate this with a unique customer's identity in a company database, and the developer needs to develop the registration mechanism. The developer was eager to

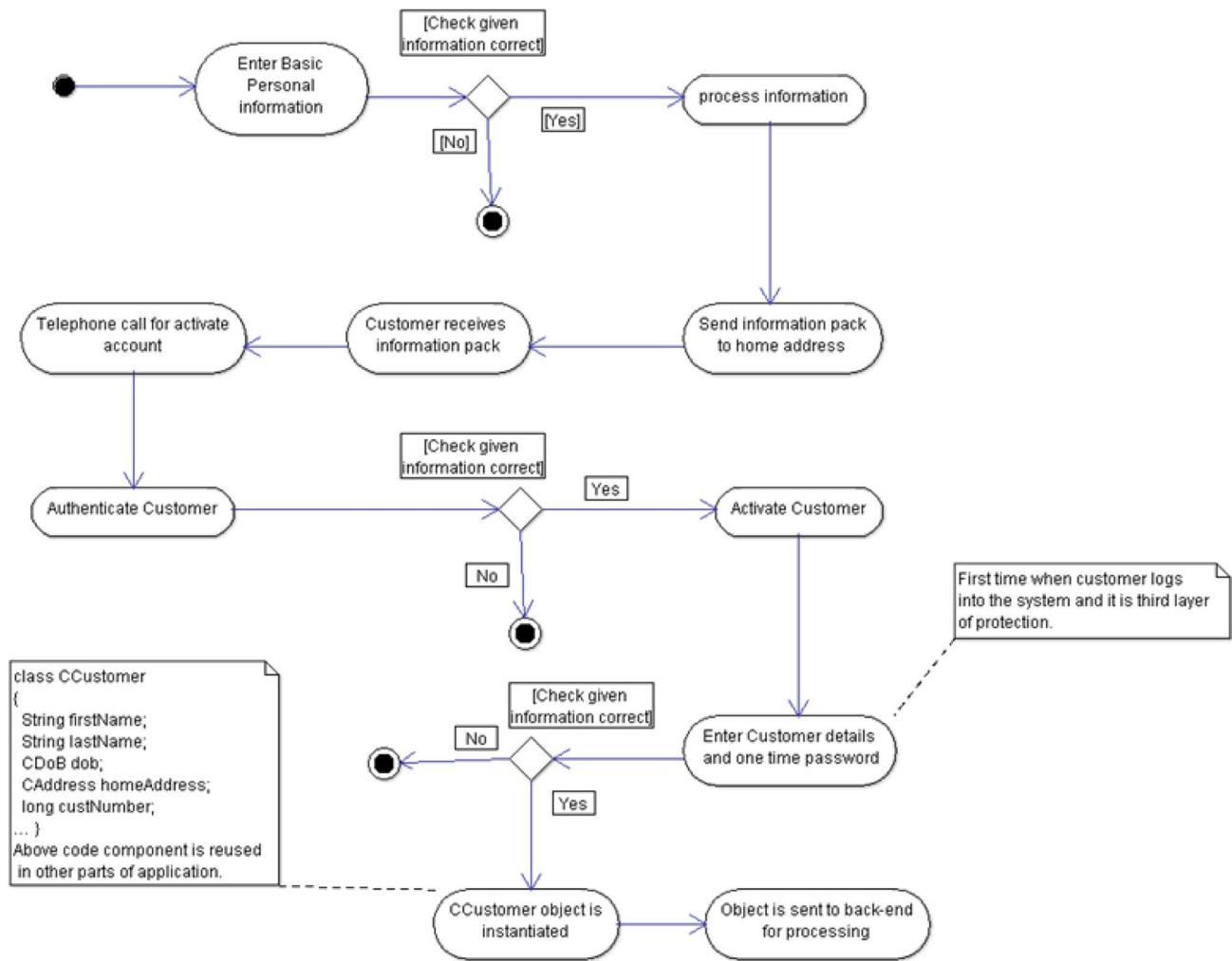


Fig. 1 Customer information handling & reused component social commerce in e-banking

reuse existing component code that was already used within the application somewhere else.

```

class CCustomer
{
String firstName;
String lastName;
CDoB dob;
CAddress homeAddress;
long custNumber;
... }
    
```

After this process was completed, as defined in Fig. 1, this object was instantiated, inhabited with provided information that is stored in the user’s session. In this process, the

application verifies the user’s details; if they match, it then retrieves that user’s unique customer number, which was used within the company’s system. This number is added in the Object with some other personal information. This object communicates with the back-end system for completion of the registration request to be processed. The developer supposes that using this code would be harmless and would not cause any security problems. However, a serious mistake caused a flaw in the actual design.

4.2 Exploitation

Figure 2 explains the registration functionality incorporated with the Customer component that consists of “(use case logic + Process and Entity Type Logic)” within the application, including core functionality. This process allows the user to authenticate and grant access to the application

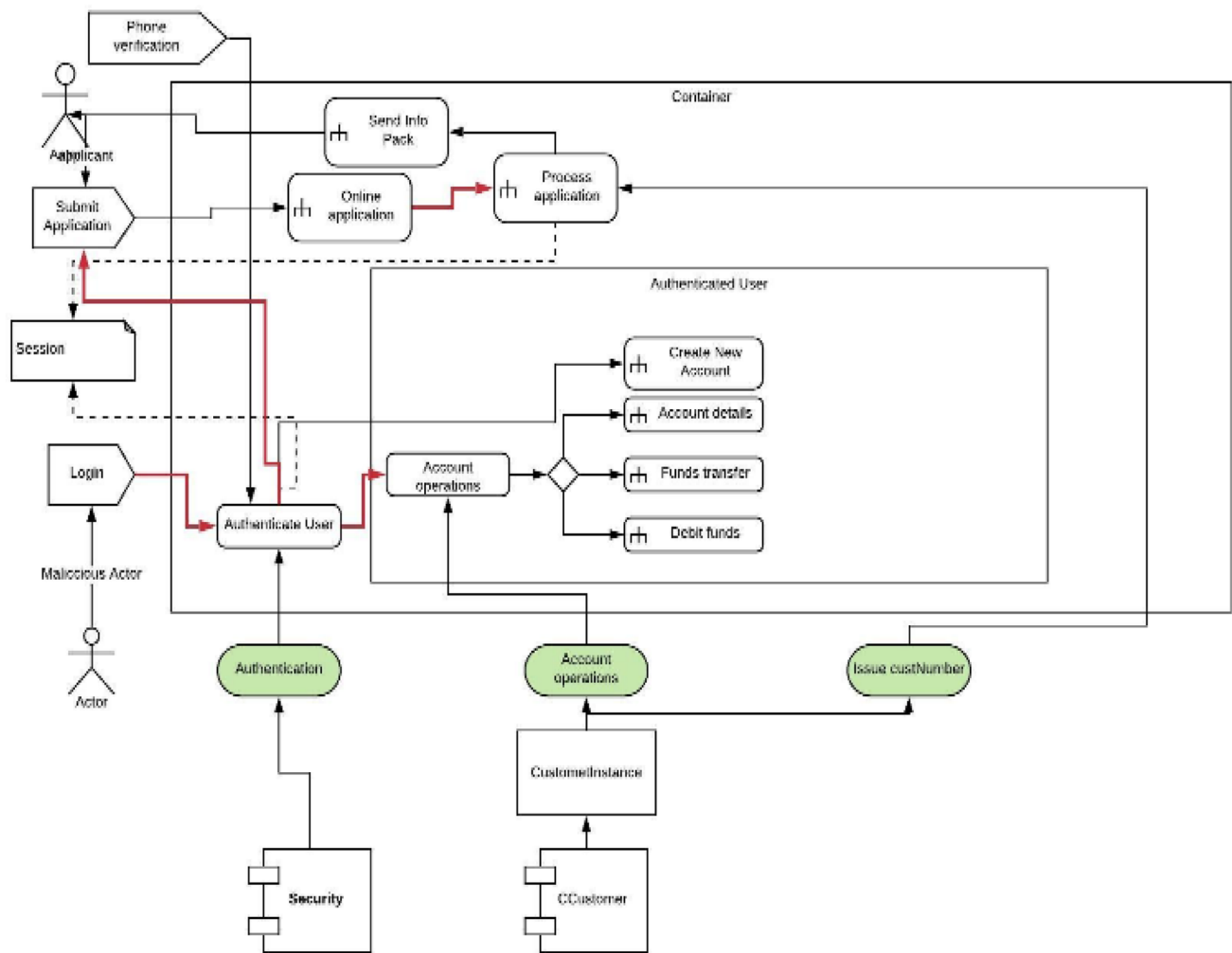


Fig. 2 Event—attack—modelling & system exploitation social e-banking

components such as “Account details component”, “Statement component”, “Funds transfers component”, “Debit component”, “Credit component” and “other information component”. After having authenticated the user to the application through the registration process, the same Object is instantiated and saved in the session key information related to the identity.

The components of the application within functionally referenced information relate to the *CCustomer (Component)* object in order to carry out its actions because *CCustomer (Component)* object is a candidate component (Process and Entity Type logic) within the majority of applications. For example, account details shown on the main page of the user were generated based on the customer’s unique number that was contained within this component. In this way, the composition or reuse of the component code was already used within the application. It clearly shows

that the developer’s assumption led to a flaw in the reuse of application logic design. This caused the birth of vulnerability to subversion attack on the application business logic. It was a serious mistake but was difficult to detect and exploit.

To exploit this flaw in the logic, an attacker may need to perform the following steps as defined in Fig. 3:

1. The first step is the “log in” process into the application using the customer’s own valid account information.
2. The result of an authenticated session, and access to the registration functionality is to try to input the some other customer’ personal information. This will result in overwriting the original “CCustomer” (Component) object in the attacker session with a new object of the targeted customer.
3. After this process, it is important to get back to application functionality and try to target another customer’s

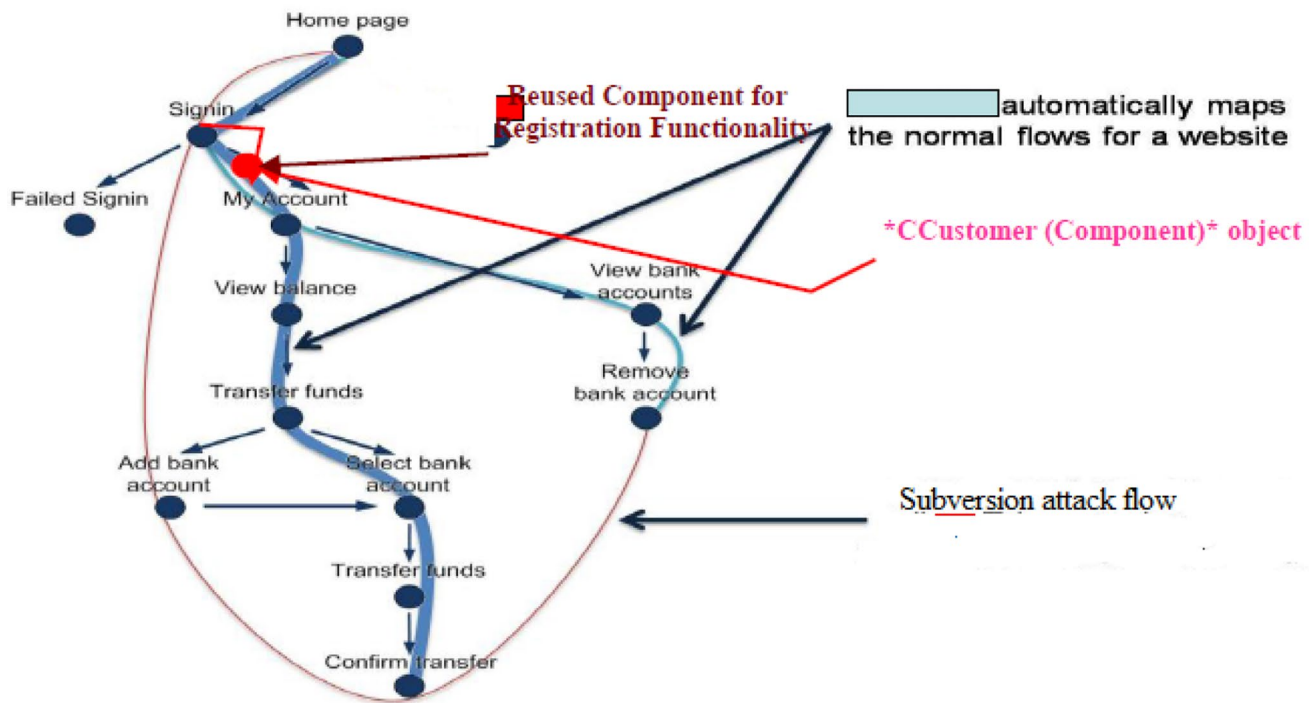


Fig. 3 Subversion attack mapped through social commerce banking service flow

account. It is hard to detect without clearly understanding the application logic as a whole and the use of different kinds of components in the application logic layer. The flawed assumption by the developer caused the subversion attack class vulnerability.

While the above-mentioned vulnerability seems to be minor, it may cause dangerous results. In fact, intruders may be relatively subtle. Access to the key application feature was assured by multilayer access controls (channel level protection), and a complete authentication session was required to pass such monitoring; fraud detection was a second security defense.

4.3 Security breach case summary

In light of the above-mentioned the question, we first addressed the problem and then, keeping the scenario of web 4.0 social commerce e-banking Application Case Study in view, proposed a security assurance methodology to overcome the problem. The vulnerability identification technique in the “Social commerce-based e-Banking Case” is accomplished by matching a sequence of components in the application development logic of the system and problem caused by overlooked integration of the business process of components during the run time logic of the application. Therefore, such a technique is necessary because

logical defects do not reveal attack patterns or signatures and thus cannot be automatically discovered. In order to solve this problem, a Threat Modelling approach is used, to project the design flaw targeting subversion attack and attacks on application logic. This approach is supported by the practice modeling component and its application, using UML for a secure design approach with a valid technique for design flaw detection (V & V method for security by design testing). Using this technique avoids the business logic problem by customizing security assurance requirements from modern approaches for component-based applications.

5 Proposed security assurance methodology

A security assurance methodology is designed to overcome the problem security aspects of challenges related to design flaws (service component-oriented application logic) in future generation social commerce-based Banking & e-Commerce applications. This methodology analyzes the security risks related to possible attacks on system design and on the base of that, three stages have been defined to deal with design-based flaw problems. The key element of this methodology consists of the formulation of well-established existing approaches that help to design a

new methodology: First (1) Threat Modelling; Second (2) Taxonomy of software Vulnerability Model; and Third (3) Component Fault Detection Models. This methodology provides further support to Modelling the Application and its Components without Fault, which then leads to Designing Security by Design Application Modeling. This methodology is proved through the validation and verification security assurance process followed by the V and V method for security by the design testing approach, which is theoretically justified by the designed model. Please note that the contingency & probability models are out of scope based on Case Scenario, which is why they are not considered part of the solution strategy, as shown in Fig. 4.

5.1 Security risk analysis

The key step in security risk assessment is to identify potential attacks on a systems design and its impact on performance, such as the above listed instance of the Web 4.0 software application based on social e-banking-based e-commerce components case, and the conceptual weakness in the business-tier application layer. The reasoning

given here focuses on a certain class of attacks on particular applications, so it will be very specific. One of the first steps in system design should be to consider potential assaults on a particular system and their effects, such as the above-noted case of the web software application based on social media-based e-commerce components, and the logical weakness found in the business application layer. The vulnerability identification technique is achieved by mismatching a sequence of components in the application design logic and problem caused by ignoring the business process integration of components. This is reflected at the time of the business process logic of the application (which can be mapped through scenario-based approach business process flow as mentioned in Fig. 3). The simple end-to-end system feature is often decomposed into sub-scenarios that define the functionality of an important component of a subsystem enabling an evaluation of the definition described in the case study. A vulnerability attack pattern can then take place in the “Event Trigger” sequence in the attack pattern, revealing the occurrence, and which component is used to exploit the vulnerability in the presented case.

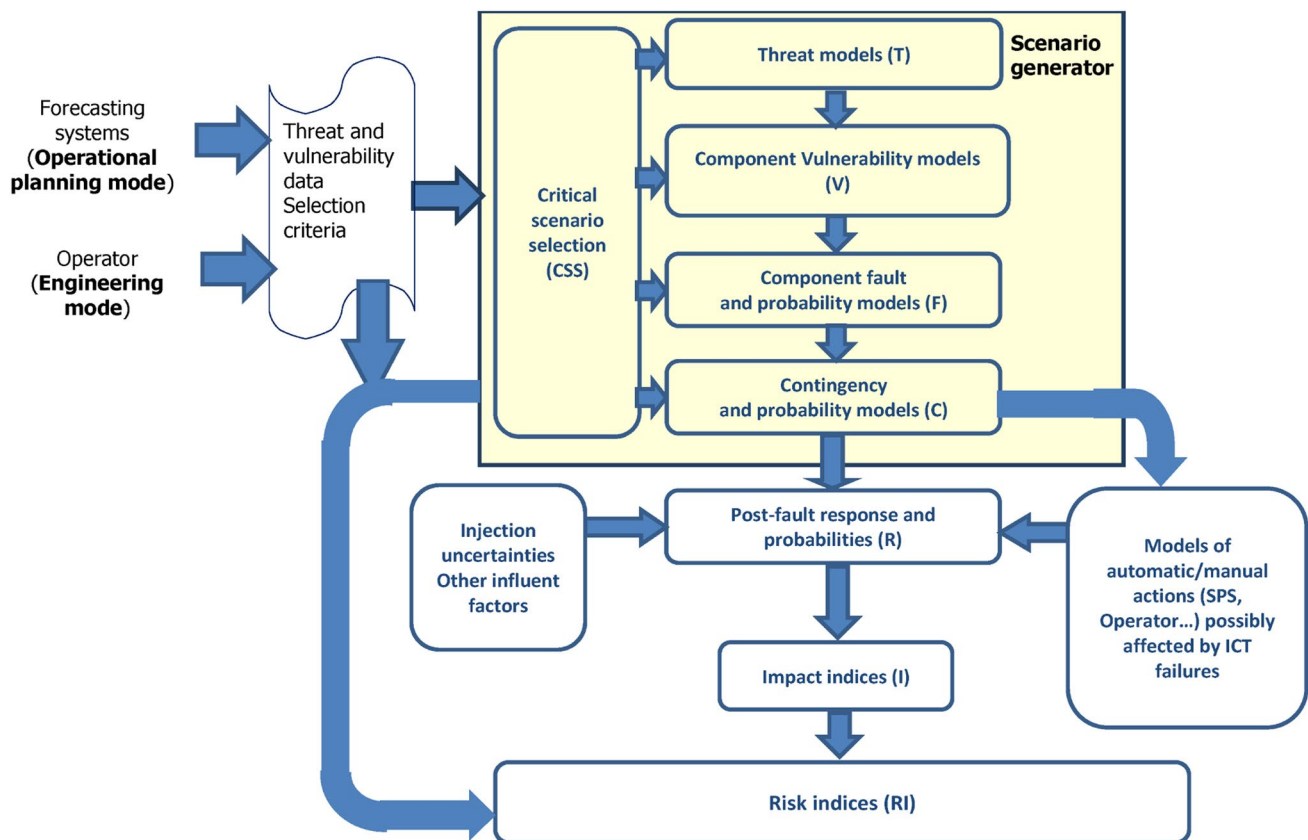


Fig. 4 Security risk analysis model

This assessment can be used to identify the necessary defensive measures and later to evaluate the system's security.

5.2 Threat modelling of application logic vulnerability

Modeling of threats is a technique that can recognize risks, attacks, vulnerabilities and countermeasures within the application scenario. In light of the above case study, we have examined a class of vulnerabilities in the application logic. These attacks, which we classified in this research as falling into the category of Design & Architecture Flaws, are based on the Logical, Design & Architectural divisions of application logic. We propose a *Method of Threat Modelling* approach to uncover the pattern of attack in the application logic from the root cause, integrating information from the case study of security breach. The threat model of an Attack Event and

Attack Target method is defined as a flaw in logic and a flaw in design. At this stage, a logical attack is defined by some attacker having access to a targeted system under the attack that enacts an illegitimate action by using an attack method (Subvert logic/Circumvent logic). A specification of an event attack method will circumvent the logical flow of an application, which may cause two further steps: one is the subversion attack; the second is circumventing the actual application's logic, resulting in a Business Logic Attack.

A major attack technique (logical error or design flaw) identifies the vulnerability and uses it to identify a threat to the components and the entire system. It must also be noted that software components and middleware from third parties constitute one of the major changes in web application systems as a security protocol, and integrity is threatened due to the design flaws in development failures as mentioned in Fig. 5.

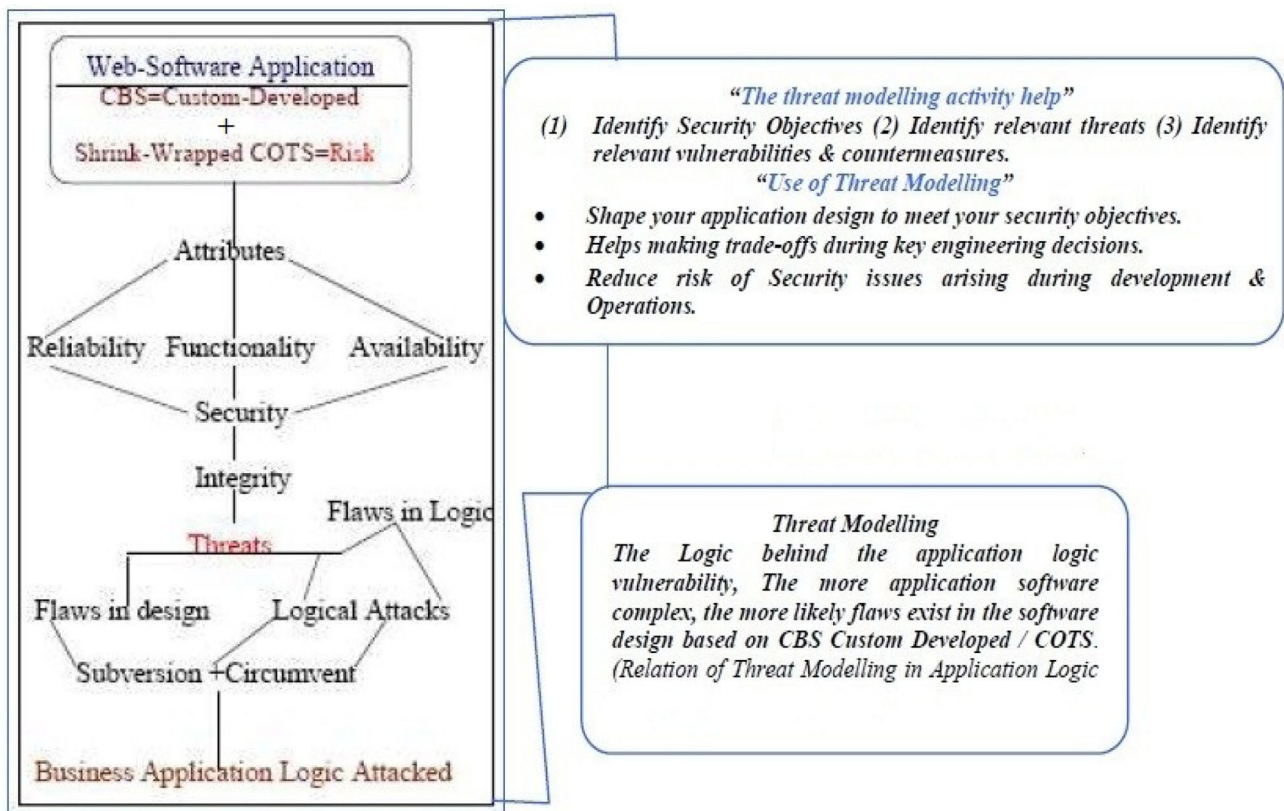


Fig. 5 Threat modelling of subversion attack

5.3 Taxonomic classification of software vulnerabilities

In software, vulnerability defects can cause security breach problems which in turn result in software loop-holes. These are (DesignFlaw, Coding Fault, Configuration Error).

In light of our research, we believe that a risk assessment in component software web applications and systems is firmly linked with some concepts traditionally derived from computer security, especially the 'five pillars': the concepts of risk, vulnerability, attack, and assets which are attributes of interest that must be defined (Nabi 2005).

According to Nabi (2011 and 2017) relates missing question to (Design & Architecture) of component structure + integration & interaction unclear model, reuse of component specification related to functional logic. Given below classification explains about discovered software security issues by other researchers contribution as mentioned in Fig. 6.

The code issues that exist in the design of the software are considered defects. A flaw in the underlying software may or may not constitute a fault. Usually, the mitigation of

a fault means much more than simply changing several lines of code (Zhang et al. 2018). The problem is not just within implementation; the underlying model is faulty. Therefore, the flaw would be included in every implementation following the model (Agirre et al. 2016). For example, in an untrusted client application, executing critical business logic is a design defect that cannot be mitigated by a single measure such as adjusting array limits.

5.4 Component fault detection model

The component fault detection is a state of the art paradigm of security by design technique. However, fault detection is a preliminary element of fault tolerance technology. Therefore, fault detection and diagnosis-based fault tolerance technologies are hard to define separately. Many approaches have been introduced, such as fault detection methods based on the system's internal data exchange and inter-component communication, but no approach covers the system's broken component risk analysis, especially when the code is not available. Therefore, under such circumstances, a technique that can model the component and system design on multi-tier specification through the UML modeling is required.

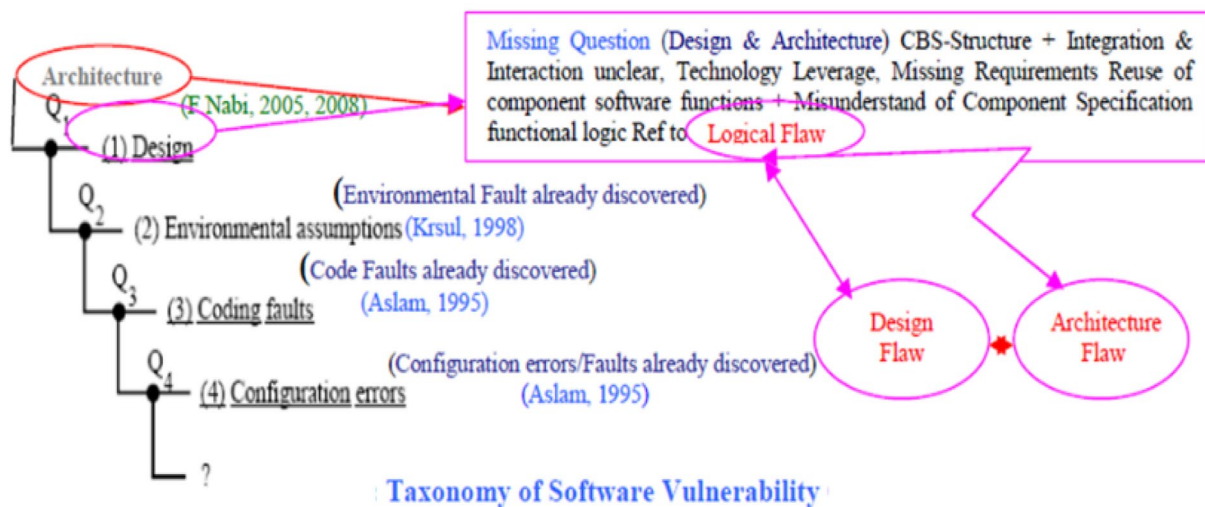


Fig. 6 Taxonomic classification of software vulnerability

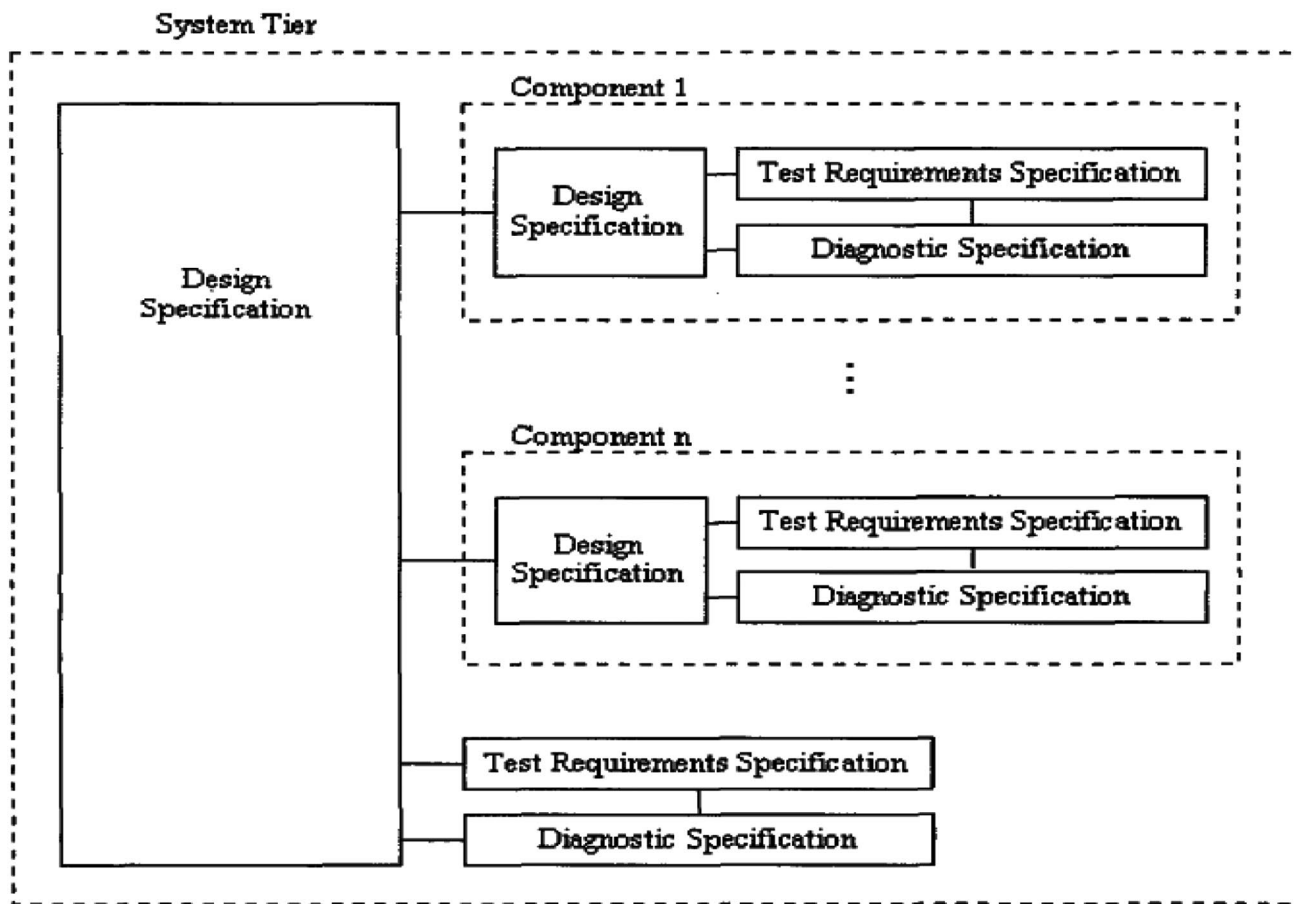


Fig. 7 Component fault detection model (CFDM)

We designed a model for a component fault detection scheme that detects faults based on component design specification, which utilizes test requirements and diagnostic specification of components, as mentioned in the diagram. It divides the model into system tier and component tier, where component 1 to component N is modeled based on stated specifications, which then test and diagnose the overall design specification of the whole model for fault detection in system design or components as mentioned in Fig. 7.

Novelty of CFDM as a practical example

In order to test the component fault detection model, it is important to address the practical testing example. For this purpose, we have chosen the Air Control system in Fig. 8. The Applicability of the above-proposed CFD model fault detection is practiced through an Air Control System. This

is based on two main components, operator user interface component and Control Station Component, in which sub-components are component 1 (ON) and Component 2 (Off) connected with a contact point where component 3 supports the system. Having the assumption of fault tolerance, the applicability of component function fails during the operation to communicate with the main component in the system as depicted in the model. That failure shows faults in the system detected through UML Sec 2.0, keeping in view a design specification based on test requirements and diagnosis specification of component and system. At this stage, where system design is considered as a whole model as depicted in the component and system fault detection model, which illustrates the practical application of faults detected in component and system.

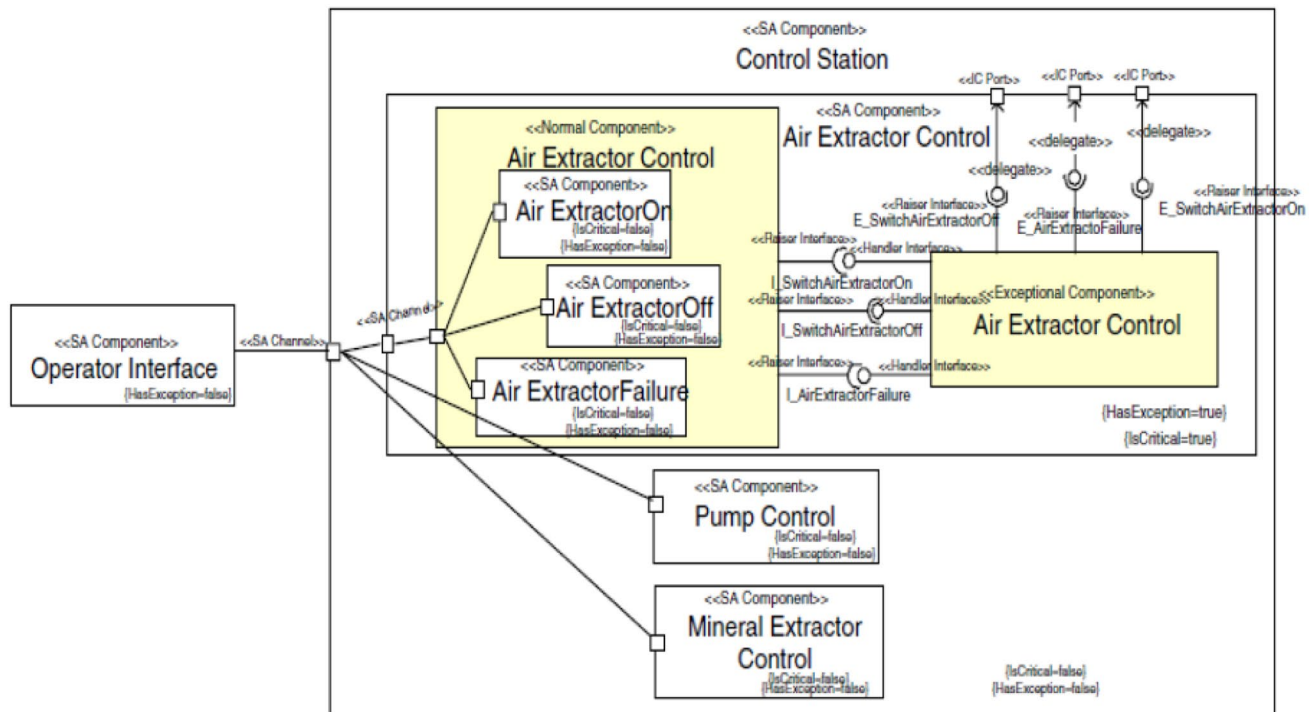


Fig. 8 UML Sec 2.0 modeling fault detection of air control system

5.5 Modeling the application and its components without fault

Component-based Software Engineering from different points of view with an emphasis on the various aspects of software design, such as various phases (design stage, component as interchangeable design parts; components reported to a particular system model in the implementation phase, run time binary packages, distributed components), industry aspects (business components, product components, COTS components), and archives. (Rodríguez et al. 2016).

In addition, we should make sure that every element of the software layout is explicit and comprehensive enough to explain the assumptions and planned functional logic in the application by the designer in order to enable the application and its components to be modelled.

Explicitly comment it is mandated that following information on all components of a system to ensure security of software and its underlining logic.

1. The intended use and purpose of the component (if the component code is usable, the functional business logic can also be expressed in the component by specifying a use contract).
2. Assumptions and logic of everything outside its direct control made by each component.

3. Reference to all client-based components using clear-cut documentation could have prevented the logic fault within the above-mentioned case, for example, online registration functionality (Note: here, the client does not apply to the client–server user-end, but to any other client-based component (code) that is considered immediately dependent upon the component logic).
4. The risk analysis consists of three key steps to ensure component-based security requirements. The first is the detection of potential system design attacks; the second is architectural risk analysis for component-based corporate logic security; and the third is component-level analysis.

5.6 Designing security by design application modeling

Designing the social commerce-based distributed system software in a tier is also extremely important since several attacks trigger design defects (subversion of attack) as mentioned in the above case study of e-Commerce applications. These logical defects usually apply not only to component-based systems but also to structural defects where component modeling is built to a set of system logic, in accordance with corporate rules relating to a particular business or sector. It is necessary to clearly define the architectural design of topology, whereby the system is designed for use

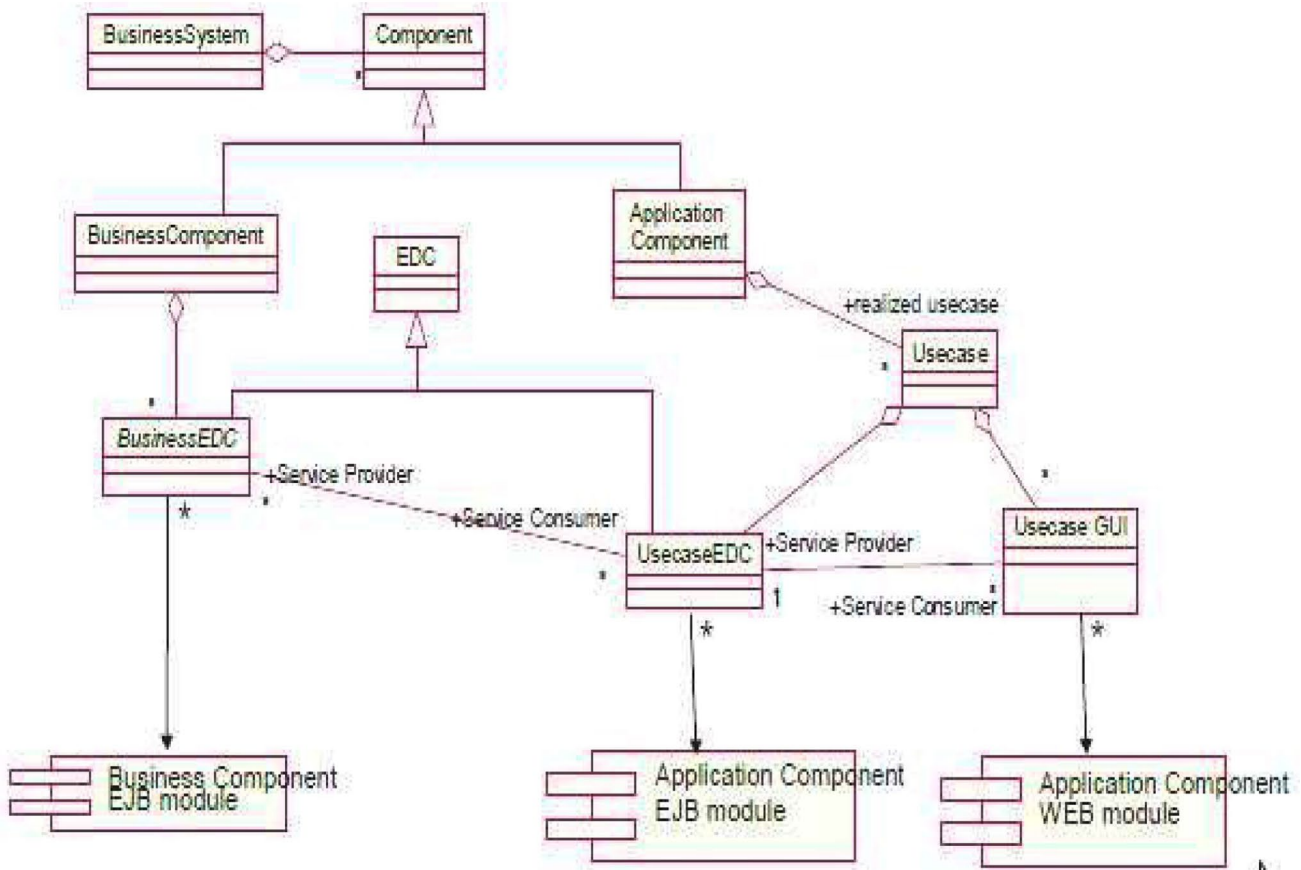


Fig. 9 UML Sec 2.0 security by design approach multi-specification J2EE system modeling

by clearly distinguishing the SDLC design level (Elio et al. 2014; Lindström et al. 2015). The second stage focuses on the strategy and policy for application logic development that concentrates on how the components need to work under a given business rule and policy. The third stage refers to the development strategy for components that use dynamic web content to customize an individual’s experience within a website and provide users with more interactive details. Dynamic content can be rendered in several ways: static HTML files, Java script, or the rendered JSP file, using the component-supported environment, such as Java servlets in J2EE by invoking business logic hosted through middleware for back-end business data access. In a normal scenario an application developer considers the approach, in which components are combined to be assembled for a particular business requirement and to create a solution. A composite application consists of new components these are created especially for business applications and existing components that are reused from other applications.

If the approach is theoretically justified, the concept of the Service Component-Oriented system development technique is a collective set of specifications that proposes a programming model for developing applications and systems. This

model promotes the previous approaches to implementing services and supporting the developing open standards like Web services.

Therefore, keeping in view theoretical justification: if we dissect a component structure, a simple component has two states; one is “Service” and other is “Reference”. A service is considered to be something that addresses the interface for a component, which keeps one or more operations, whereas a reference is a dependency based on a service (functionality) that is required by another component.

We have modeled a system (Fig. 9) in a scenario in which techniques consist of system and component at two different levels. The system level focuses on the design product, while specifications for individual components that participate in system development are modelled as UML sec design approach. This is the multi-specification architecture of a component-based system in which different layers of components are modelled based on their business rules and processes, with execution within the system. The component integration strategy at run time is compared to its requirement and design specifications.

The component-oriented programming concept promotes the implementation of service, where the security by

design technique supports the UML-based design modelling for system security at the design stage or while reusing the component from an existing application. For this purpose, we have practiced the UML-based security by design system modelling through the concept of service component-oriented software engineering. In this exercise, Business Domain Components and service Components are being modelled based on a multi-tier specification of the architectural design of topology in which the system will be designed for deployment by separating each tier. The first tier consists of the Business System Interface, which is connected with a service component (rendering logic) that corresponds to the second tier: the Business Component and Application Component. From this stage, Business EDC and sub-component, as realized use cases, invoke mid-tier service. Furthermore, this process corresponds with the back-end service and application components. These components provide Consumer Service and Business Component service provider in order to process the system function (defined as the component’s business logic), as depicted in the figure given below. The component integration strategies at run time are also clearly defined through inter-connections among the all system and application components, while considering requirement and design specifications of service description.

6 A validation & verification of method integration testing model

To validate the security assurance process, the technique would follow the V & V integration method for security by design testing the approach that would be set as a test-bed

model, which then has no need of all component realizations. Normally models are available at an earlier stage as compared to at their realization. This technique makes the process easier and makes it possible for earlier detection of design flaws, especially in the case of comparing/matching real time system testing.

In addition, adaptation and configuration is normally allowed by models as compared to realizations that are appropriate for system testing in unrelated conditions for assurance purposes. It is easier when models are used as an alternative of realizations, especially when exceptional behavior testing and broken components are conditional.

In addition, test quality is improved and the ability of models to change test conditions at a great rate improve the state of test execution. In return, not only is the allowed model-based testing simple and easy but it also lowers the chances of threat during the test process as mentioned in Fig. 10.

Theoretically justified by the design model and method of security assurance process, diagram 10 illustrates the graphical display of the V & V integration model for security by the design testing approach. It considers the components $C1$ & Cn that are only illustrated above in the figure in which component $C1$ is shown by $M1$ model, whereas Cn is shown by realization Zn . In the IMZ is infrastructure integration model, both $M1$ and Zn need to be integrated, which refers to the model and its realization, so that the equation is rendered $\{M1, Zn\} IMZ$. Therefore, this initial depiction of the system is considered for testing at an early system stage, which extracts from R & D , system requirements, and design, that are shown by the dashed arrow. The R (requirement) & D (design) systems of both specifications are captured in order to allow $M1$ model and Zn realization to be tested using

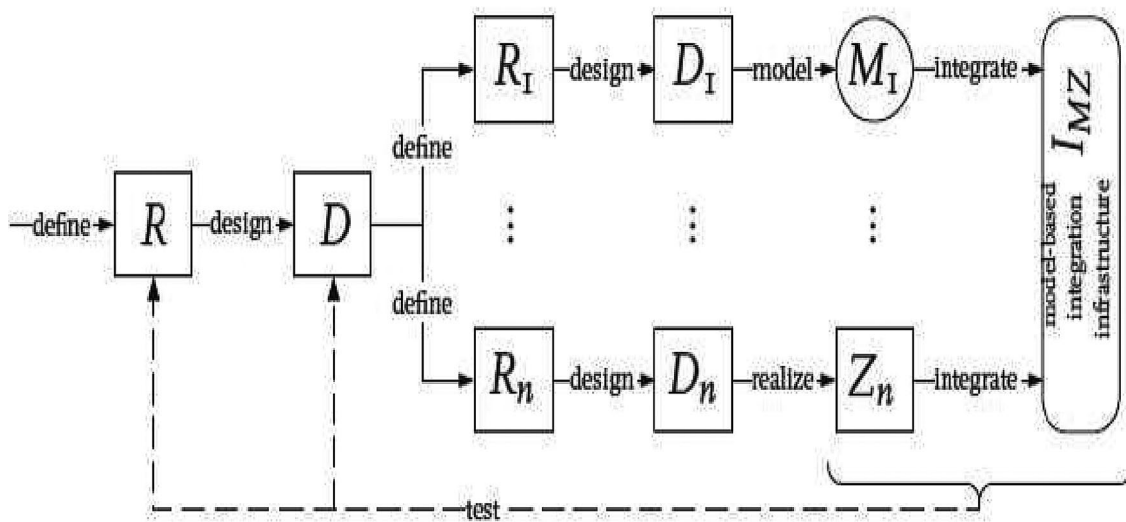


Fig. 10 V &V integration model for security by design testing

Integration of IMZ for the infrastructure integration model. The process of the model justifies above defines the security assurance service that deal with the component-oriented applications in the context of the security assurance process methodology.

Hence, in the light of the V&V model, we have exemplified a case scenario of the “Domain Model System” for applicability to the proposed model in Fig. 11, which is demonstrated with the UML modeling to detect the flaw or fault in the design. For this process, System component specification is used through the scenario-based schema to generate event modeling for the system through equation $\{M1, Zn\}$ IMZ integration testing. The diagram illustrates a component and sub-component-based system, in which faults are detected, keeping in view the equation $\{M1, Zn\}$ IMZ where all the components realization are not required, especially when exceptional behavior testing and broken components are conditions of the test. The information regarding the R & D of the Domain Model system are gathered with the

specifications of M1 and Zn realization with C1 & Cn, respectively, to be tested, using integration IMZ. The threat model also explains the fault cause of errors in the process of communication (event-based messaging) that may fail and may cause underlying threat vulnerability of the application logic. The technique of identifying vulnerability in the Domain Model System is achieved via matching a sequence of components and sub-components in a system application logic.

The graphical presentation of the above-proposed method of the V & V Integration model for security by design testing and experimental modeling of fault detection in component-oriented-service is generated through the formula in the equation $\{M1, Zn\}$ IMZ. However, it is important to explain the process through the projected model as it is defined here in Fig. 12. This model demonstrates the requirement and system design specification by formulating the properties of components that are used for realization and integration

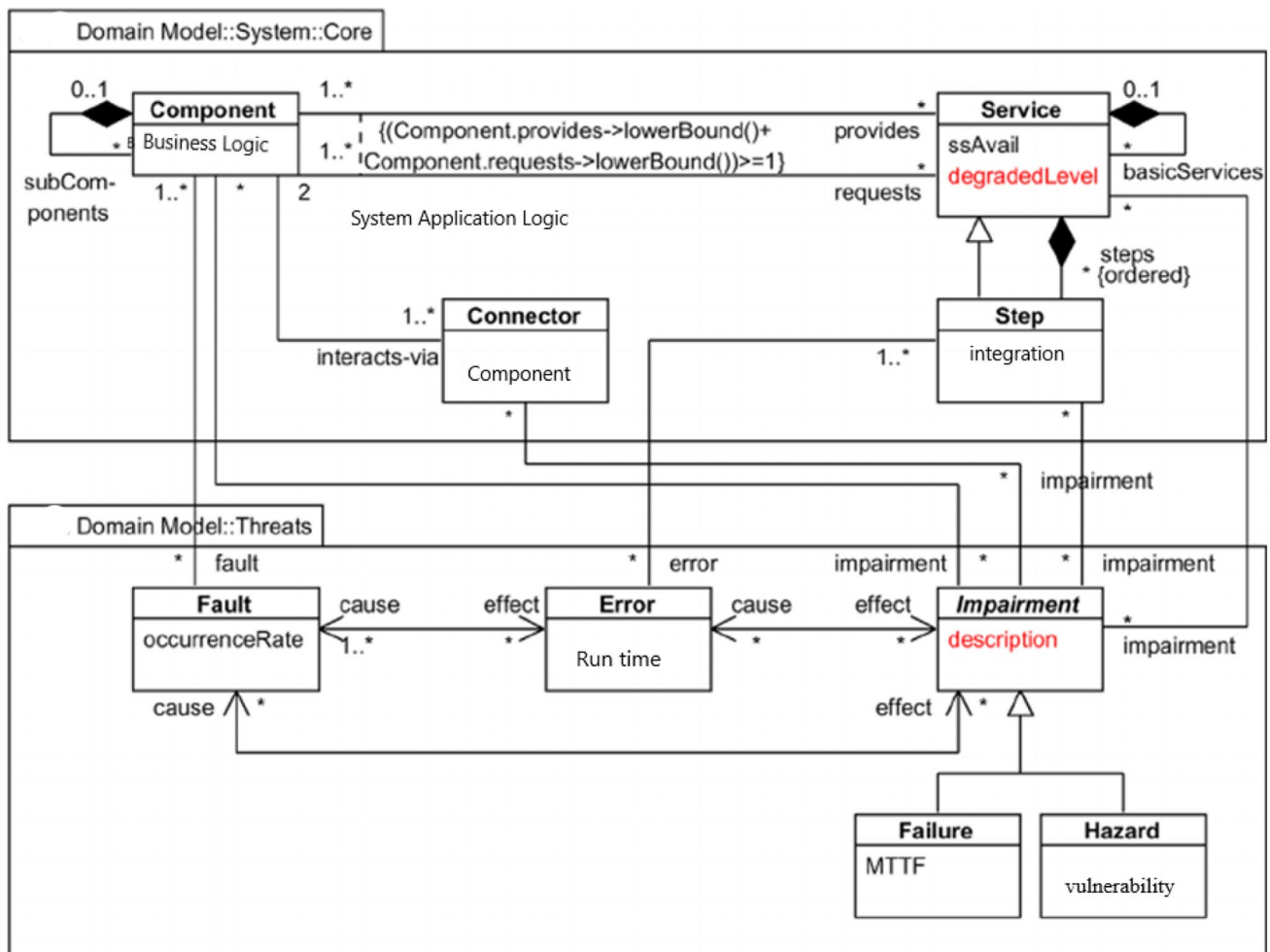


Fig. 11 V &V method for fault detection in component-oriented-service

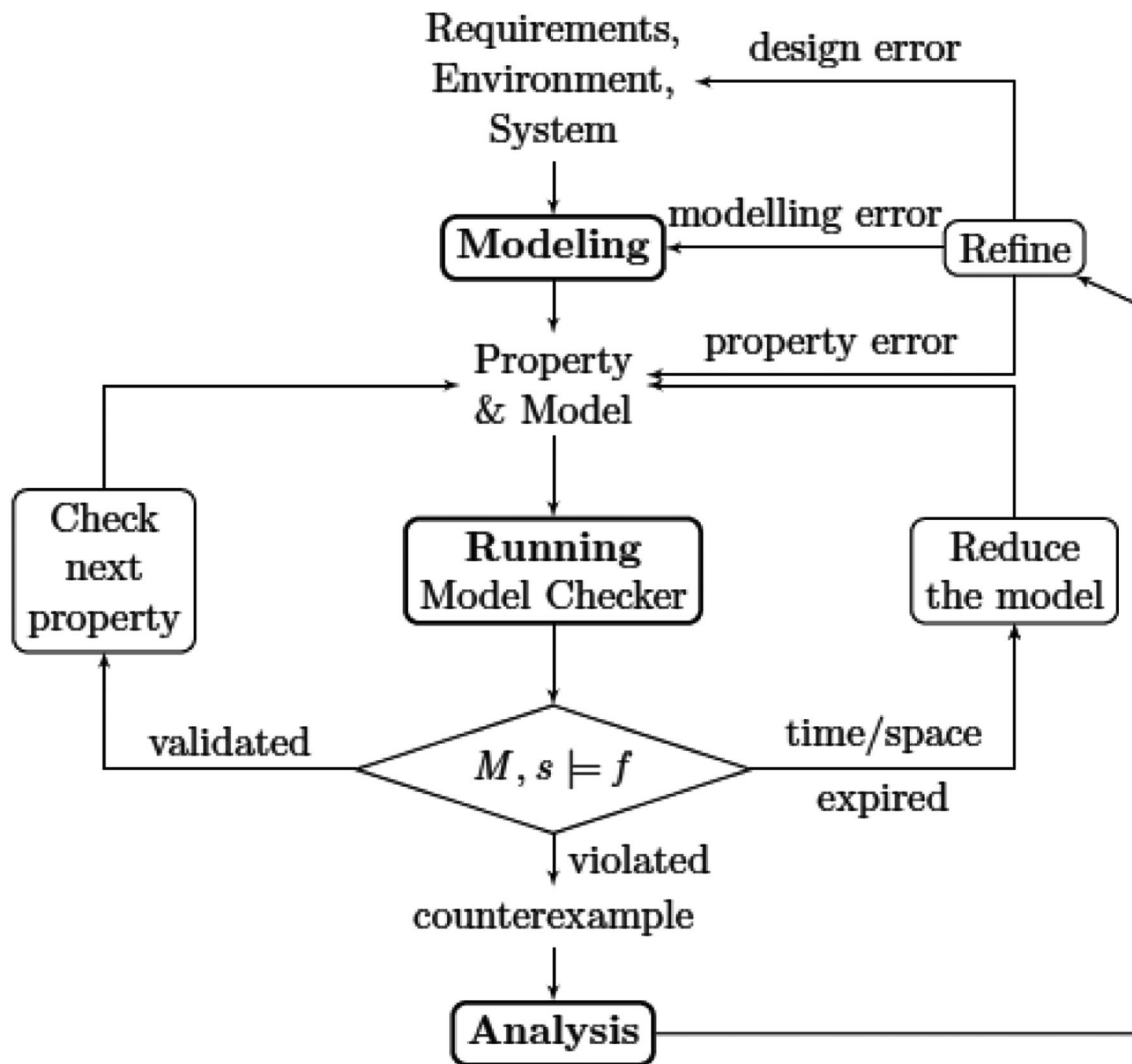


Fig. 12 Model-checking process to validate the proposed method of V &V

of the IMZ checker to be tested as defined in Fig. 12. This whole process validates the idea as explained in Sect. 6.

7 Discussion

The key point to consider is that the component itself could be considered as properly designed as long as the design specification of the component matched the n-tier architecture e-Commerce applications with regard to each layer’s boundary profile condition (functional specification requirements). This complies with the current application logic in the purpose and actions of the overall system. The concern is that the component developer and the software development company are likely to be individuals (this is the most common scenario!). In this sense, a validation process is needed to validate the design before implementing it when

incorporating components into component software artifact models and functionally processing logic. This is based on interface specifications that are unwanted. So what is required? A component software integration process that refers to security risks model in the components composition at the design stage, may be incompatible with their semantics and operational behavior! The value of this lesson may help the development of the future generation system as well to help understandings of the basic rules of component semantics and operational behavior oriented integration. Throughout a component-based software solution development, the Defense Software Community (DE Software Community) reuses components of the current system’s logic in view of the specification and role-performance of different layer components in the specific solution, such as Arian Rocket Failure (ESA).

This indicates that component software compatibility based on component software model objects was suggested, but the designer ignored component interface drive logically defined constraints. Those defined by using the component used and offered interfaces from the overall logical structure design via contract strategy during the composition. This may have caused the solution failure that depended on functionality related to particular boundary profile condition as compared to design specific conditions that caused the (ESA) Mission Arian to fail.

8 Conclusion and future directions

The lesson from this case study is that developers must always consider the key point that the component must itself be correctly designed, as far as the design and specification of the component is concerned. It must match with respect to the boundary profile condition of each layer's *functional specification requirement* within the architecture that respects existing application logic in the overall system's function as well as its behavior. Therefore, keeping in view this point, we have practiced a security assurance methodology for a service component-oriented modern generation e-Commerce social media application through threat modeling. This follows a component fault detection model novel technique with a further modeling component and its application using a UML secure design approach and developing a valid technique for design flaw detection. This will increase the level of assurance during the design component-based, rapidly developing future generations of social e-Commerce application software and deployment of business logic into the social commerce e-banking system. The proposed approach would also encourage the developers to design secure component-based applications while re-using existing components from the application's business logic and will thus ensure the security of the secure design-based modeling technique in the design method.

This research also opens a new direction for the social e-Commerce application using component-based software techniques that may help developers understand faster development and related security concerns of such system requirements. The research also provides a lesson for security modeling techniques through new methods that are addressed here. This may help the developers design secure social commerce systems at the enterprise level.

References

- Abdulrahman A, Mansour A, Noura A (2017) A model for evaluating the security and usability of e-banking platforms. *Computing* 99:519–535. <https://doi.org/10.1007/s00607-017-0546-9>
- Agirre A, Parra J, Armentia A, Estévez E, Marcos M (2016) QoS aware middleware support for dynamically reconfigurable component based IoT applications. *Int J Distribut Sensor Netw* 3:17. <https://doi.org/10.1155/2016/2702789>
- Agirre A, Armentia A, Estévez E, Marcos M (2018) A component-based approach for securing indoor home care applications. *Sensors* 18(1):46. <https://doi.org/10.3390/s18010046>
- Alalwan AA, Dwivedi YK, Rana NP, Algharabat RS (2018) Examining factors influencing Jordanian customers' intentions and adoption of internet banking: extending UTAUT2 with risk. *J Retail Consum Serv* 40:125–138. <https://doi.org/10.1016/j.jretconser.2017.08.026>
- Elio G, Karim D, Benjamin G, Eric D, Claude G (2014) A security risk assessment model for business process deployment in the cloud. In: 2014 IEEE international conference on services computing, pp 307–314. <https://doi.org/10.1109/scc.2014.48>
- Ghassan B, Achim H, Rafael Valencia G, Jun S, Asif G (2020) Towards an assessment framework of reuse: a knowledge-level analysis approach. *Complex Intell Syst* 6:87–95
- Jiang H, Zhou R, Zhang L et al (2018) Sentence level topic models for associated topics extraction. *World Wide Web*. <https://doi.org/10.1007/s11280-018-0639-1>
- Jones A, Ashenden D (2005) Risk management for computer security: protecting your network and information assets 1, 1st edn. Elsevier, Amsterdam, pp 46–57
- Laukkanen P, Sinkkonen S, Laukkanen T (2018) Consumer resistance to internet banking: postpones, opponents and rejectors. *Int J Bank Mark* 26(6):440–455
- Lindström B, Andler SF, Offutt J, Pettersson P, Sundmark D (2015) Mutating aspect-oriented models to test cross-cutting concerns. In: 2015 IEEE eighth international conference on software testing, verification and validation workshops (ICSTW). <https://doi.org/10.1109/icstw.2015.7107456>
- Nabi F (2005) Secure business application logic for e-commerce systems. *Elsevier J Comput Secur* 24(3):208–217
- Nabi F, Nabi M (2017) A process of security assurance properties unification for application logic. *Int J Electron Inform Eng* 6(1):40–48
- Nabi F, Yong J, Tao X (2019a) A novel approach for component based application logic event attack modelling. *Int J Netw Secur* 22(3):437–443
- Nabi F, Yong J, Tao X (2019b) Proposing a secure component-based-application logic and system's integration testing approach. *Int J Inform Electron Eng* 11(1):25–39
- Nabi F, Yong J, Tao X (2020) Classification of logical vulnerability based on group attacking method. In: 11th international conference on ambient systems, networks and technologies (ANT 2020), Warsaw Poland
- Nabi F, Yong J, Tao X (2021) Classification of logical vulnerability based on group attack method. *J Ubiquit Syst Pervas Netw* 14(1):19–26
- Raed SA, Nripendra PR (2020) Social commerce in emerging markets and its impact on online community engagement. *Information*. <https://doi.org/10.1007/s10796-020-10041-4>
- Rodríguez M, Zalama E, González I (2016) Improving the interoperability in the digital home through the automatic generation of software adapters. *RIAI Rev Iberoam Autom Inform Ind* 13:363–369
- Seinturier L, Merle P, Rouvoy R, Romero D, Schiavoni V, Stefani J-B (2017) A component-based middleware platform for reconfigurable service-oriented architectures. *Softw Pract Exp* 42:559–583
- Wang H, Wang Y, Taleb T, Jiang X (2020) Special issue on security and privacy in network computing. *World Wide Web* 23(2):951–957
- Khafa F, Barolli L, Papajorgji P (2010) Complex intelligent systems and their applications. Springer optimization and its applications, vol 41. Springer, New York
- Yaghmaie A (2017) How to characterise pure and applied science. *Int Stud Philos Sci* 31(2):133–149

- Yin RK (2016) Case study research design and methods. *Canad J Prog Evaluat* 1:1. <https://doi.org/10.3138/cjpe.30.1.108>
- Zhang T, Zheng L, Wang Y, Shen Y, Xi N, Ma J, Yong J (2018) Trustworthy service composition with secure data transmission in sensor networks. *World Wide Web* 21:185–200

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

CHAPTER 7: SOCIAL INTERACTION WITH SOFTWARE CONNECTORS AND THE ROLE OF FAÇADE-BASED COMPONENTS FOR SECURE APPLICATION LOGIC

Introduction and Findings: Relationship between Chapter 6 and Chapter 7

Introduction Note: Chapter 6 addresses a detailed case study-based problem solution in the context of social e-commerce that is used as a tool to close down the subject of the targeted audience in relation to the banking case study and proposed methodology that is justified through modeling techniques related to application logic. Chapter 7 covers and addresses the modeling techniques by using and incorporating security-modeling features into component service architecture to expanding the research work in Paper 5. This will be further reflected as a part of security feature-based UML Sec modeling for an example B2c ATM model demonstrated in the context of social interaction of e-commerce component-based application security modeling that justifies secure application logic. This chapter addresses -sub-question 5 and it makes a relationship to the thesis as defined in the structure of the thesis in Chapter 1

Findings: This chapter provides UML-based modeling and suggests security through façade-based function describing an example of the B2c ATM model demonstrated in the context of social interaction of e-commerce component-based application logic in terms of the social medium of application human interaction.

This paper has been accepted for publication in the International Journal of Network Security.

Social Network Analysis and Mining

Social Interaction with Software Connectors & the Function of Facade Component in Secure Application Logic

--Manuscript Draft--

Manuscript Number:	SNAM-D-21-00168
Full Title:	Social Interaction with Software Connectors & the Function of Facade Component in Secure Application Logic
Article Type:	Original Article
Corresponding Author:	faisal nabi USQ AUSTRALIA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	USQ
Corresponding Author's Secondary Institution:	
First Author:	faisal nabi
First Author Secondary Information:	
Order of Authors:	faisal nabi Jianming Yong, PhD Xiaohui Tao, PhD
Order of Authors Secondary Information:	
Funding Information:	
Abstract:	The paper represents a conceptual study of the social interaction model of ATMs by using integration in a UML model-based design for software architecture that deals with the secure component-based software protection and communication patterns through facade based connector. The facade-based connector is developed separately from software components by using the correct contact pattern between components and the required security patterns that encapsulate the connection with other secure connectors. Each secure connector consists of both security architecture and contact patterns. It is constructed as a composite component even though connectors are usually used by component-based software development to encapsulate the mechanisms of communication with components. This paper addresses the security and privacy issues related to social interaction with the ATM model that may also be encapsulated in the software application logic through a secure facade based connector that is known as a security in a connector based social e-commerce B2c application.
Suggested Reviewers:	Prof Chunsheng Yang, PhD Prof, Conseil national de recherches Canada: National Research Council Canada Chunsheng.Yang@canada.ca He is expert in social network and software security Dr.Rosisin mullins, PhD Prof, University of Wales Trinity Saint David r.mullins@uwtsd.ac.uk She is expert in information systems and security ghazanfar safdar, PhD Prof, University of Bedfordshire - Luton Campus: University of Bedfordshire ghazanfar.safdar@beds.ac.uk Security and Authentication Protocols for Communication Networks

[Click here to view linked References](#)

Social Interaction with Software Connectors & the Function of Facade Component in Secure Application Logic

Faisal Nabi, Jianming Yong, Xiaohui Tao
(Corresponding author: Faisal Nabi) faisal.nabi@yahoo.com

School of Business, and School of Science University of Southern Queensland 1 West St, Darling Heights QLD 4350, Australia.

Abstract— The paper represents a conceptual study of the social interaction model of ATMs by using integration in a UML model-based design for software architecture that deals with the secure component-based software protection and communication patterns through facade based connector. The facade-based connector is developed separately from software components by using the correct contact pattern between components and the required security patterns that encapsulate the connection with other secure connectors. Each secure connector consists of both security architecture and contact patterns. It is constructed as a composite component even though connectors are usually used by component-based software development to encapsulate the mechanisms of communication with components. This paper addresses the security and privacy issues related to social interaction with the ATM model that may also be encapsulated in the software application logic through a secure facade based connector that is known as a security in a connector based social e-commerce B2c application.

Keywords

Computer Security, Component based Software, Social e-Commerce Applications, Façade Component, Connectors Security, Security & Privacy.

1 INTRODUCTION

Modern life is far more complex than any other kind of social order, and is primarily aimed at achieving individual objectives, thus fragmenting the social order and reducing the interaction that comes with social contact. Consequently, people rely on social media (e.g. email, social media and video conferencing) to maintain contact and with the passage of time, social e-commerce is also being used, such as social interaction with the ATM banking model. This model has enabled a new era of shifting from traditional e-commerce to social e-commerce and Banking industries have adopted this model. This refers to human social interaction (Siricharoen 2019). However, the security of HCI related to social interaction is a very important issue requiring discussion among the research community. One example is social interaction using the ATM Banking security model.

In modern era of software, the component-based software development method is widely used, especially in the distributed social e-commerce system, which has developed through components. Such systems or applications encapsulate the functionality of the business logic process. Apart from that, components also need connectors, which encapsulate the inter-communication of component process (Albassam et al. 2017). The UML 2 provides notation for modelling the component ports that are provided and the necessary interfaces in CBSAs asynchronous communication or synchronous replication communication. The connector's role within the component-based application is that of a glue that encapsulates the inter-communication mechanisms to perform the functionality that supports the application logic. This provides a major role in component based applications because some components within an application are without offered or required interfaces that need connector support to glue the connection in between the component interaction within the application.

Current approaches for component-based development of social e-commerce often ignore the secure integration process designed by the security concept. There are currently two main methods for component integration. These current methods appear to use a composition process that falls into two main categories: (a) transmitting the message directly; (b) transferring the message indirectly. In these two methods, interaction of components is based directly on inter-communication which corresponds to the direct method call, performing two distinctive roles, both sender and recipient of a message. In this case, the recipient's identity is either known to the sender statically or evaluated dynamically during the execution time.

The direct message transmission method calls for a component that is tightly coupled, in which there is no need of any glue code or connector as shown in figure 1 (a). On the other hand, indirect message transmission connectors are exemplified as separate entities that are defined clearly in the component composition. They are usually glue codes or scripts, which pass messages indirectly between components. To make a connection from one component to another component, a connector provides the link, which provides a method to be notified by the former. In general, the data flow associated with the composition is isolated from the computation in the individual component, when the components are linked through the indirect message transfer figure 1 (b). This figure clearly illustrates that components do not directly call each other but connectors are called in the system method process during the loosely coupled application logic (Albassam et al.2017).

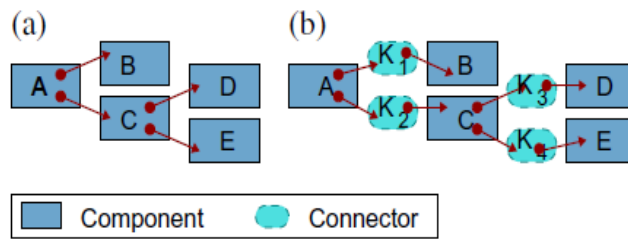


Figure 1: Direct and Indirect component inter-communication model

A pattern of security represents a common detection procedure performed by a security service. A secure connector improves the management of complex software systems by distinguishing security concerns from software architecture applications concerns. Therefore, each component determines a relatively independent application logic that differs from that of other components. A connector performs functions as a means of communication between components on behalf of the components, which encapsulating the specific type of inter-communication between components. A secure design approach is used to encapsulate security functionality separately from the software components inside the connector (Shin et al. 2018). The connector's original function is to provide a framework to exchange messages between components in the software architecture. However, the function of connectors is expanded in this paper by adding security patterns as designed by the security concept to the connectors via a secure session façade component acting with secure connectors.

The idea of secure connectors is based on component principles, which include both basic components that encapsulate security patterns and message communication patterns, in which a secure connector is a composite component. The secure inter-connectors are designed using component principles and include composite modules that encapsulate the security patterns and message transmission patterns of the secure connector. A security pattern object is encapsulated in a secure connector as a component to offer one or more security services to software components to secure business application logic.

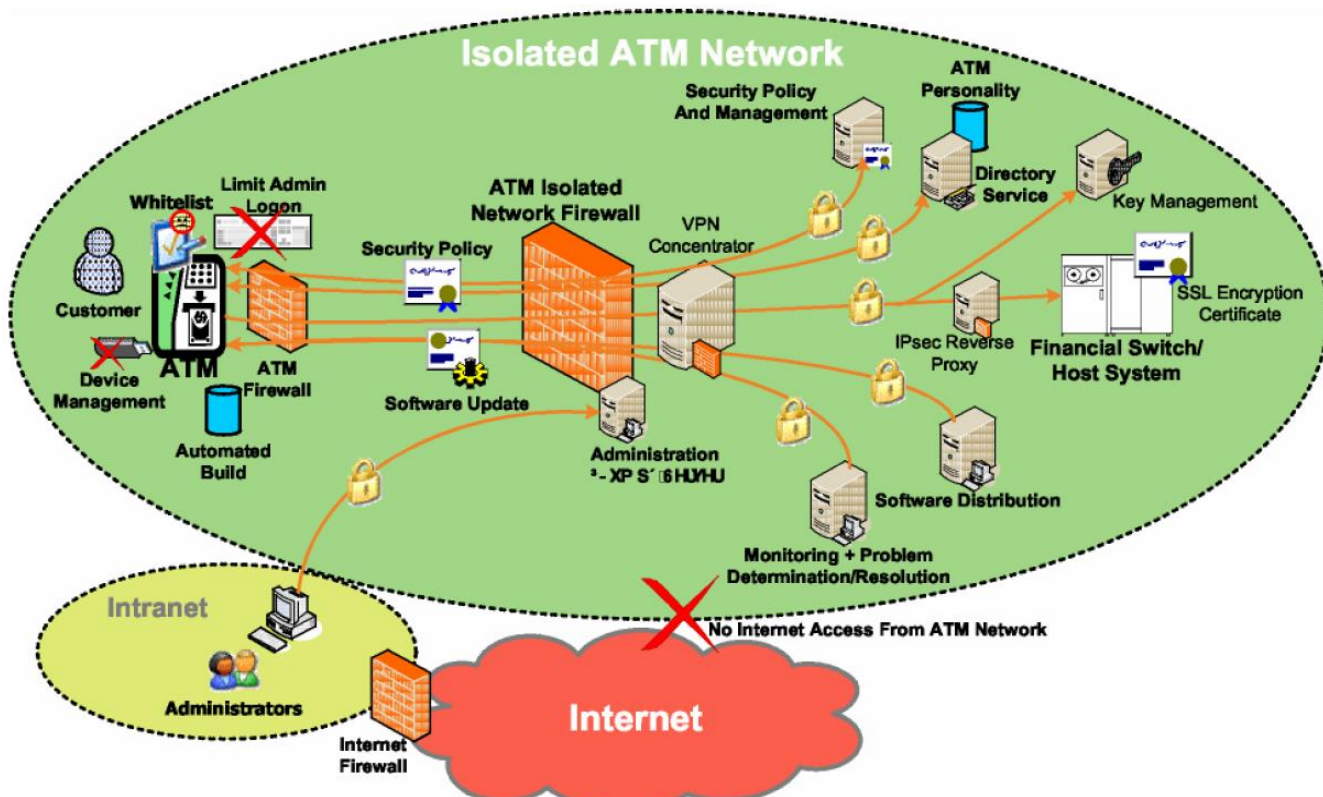
2. Research Design

In this paper, we address the role of facade-based connectors in the development of secure business application logic in the domain of social e-commerce based applications using ATMs (B2c) example. According to software architecture theory, the functionality of an application is handled by components in a CBSA (Taylor et al. 2010) for concurrent and distributed systems, whereas the emphasis here is on coordination between components. For this purpose, a comprehensive review study is carried out and a solution is suggested in section 4. This work relies on exploratory research analysis, which aims to create an idea of secure application logic, free from vulnerability, to address the complexity of the business logic phenomenon in social e-commerce systems (B2 c).

2.1 Research Motivation

In recent years, the number of ATM incidents has risen, particularly fraud involving malicious applications, including Blackbox electrical devices. Black box is a term widely used to define technically advanced electronic devices, which are directly attached to an ATM in such a manner as to enable the offender to exercise control over the operation of the ATM. The key targets of these attacks are the data, including the cardholder information. In this case, the focus is the social interaction with the ATM and the ATM dispenser which is used for the provision of cash. These attacks disrupt authentic transactions with a genuine card and a PIN. Attacks on different ATM models have been successful suppliers that run multiple ATM software versions. This paper addresses the security issues related to social e-commerce, particularly ATM Banking that may also be encapsulated in the software application logic through a facade-based connector that is called a security in connector-based application. This approach can deal with attempted fraud involving the Black Box. This form of cyber crime is a sub-set of social crime. Additionally, the Subversion Attack Vulnerability in application logic in B2c system may be the cause of a component based event called the inter communication method. This leads to a process at the time, comprising loosely coupled or tightly coupled integration based on interfaces of components as defined in figure 1. The solution defined related to this problem is modeled through B2c ATM case study fig.6-7.

Figure 2: Social ATM Network and Software Process Environment Model



3. Review of Existing Work

According to Almeida 2017, due to the broad diffusion of Web 4.0 technology, e-Commerce is gaining momentum. The focus of e-commerce applications has moved towards context awareness and personalisation in social mining, recommendations and data semantics. However, the design of such systems involves complex architectures to accommodate smart behaviours. These systems still insure substantial non-functional properties such as flexibility, performance and scalability. Therefore, secure social e-commerce applications logic is an important aspect of such systems, and this has not previously been clearly or fully discussed by the research community.

According to Shin, M. E et al. 2017, several methods have been proposed for the development of secure distributed software business applications. Many of the solutions that have been designed focus on protecting the components of business applications, in order to provide an effective security service for the component. However, less consideration has been given to connectors that can be used to provide security services for the components of business applications. In particular, social interaction based communication through connectors. The action of connectors is used to bind or encapsulate communication process between components in software architecture. At this stage, security concerns can also be encapsulated in software connectors, which are classified as protection connectors, separate from application components that contain application logic in social e-commerce applications.

According to Al-Azzani.S et al. 2012, for the simulation, the capturing and implementation of the security orientated connector approach is used. Using software connectors, the security characteristics of a system design are defined. Connectors control and apply an extensible software security contract defined by components. Connectors can assess what values the associated components should execute. Security relies on the connector core software architecture to ensure that specific security requirements are taken into account during its design. In addition to addressing the problem from an architectural point of view, their work also provides a way to define and improve these stable connectors. Therefore, an opportunity is available to work on the social interaction of connector approach, while designing component-based software for social e-commerce applications.

According to Shin, M. E et al. 2016, secure connectors are used for the implementation of a distributed business application in secure software architectures where components of business application may communicate with one another through various types of communication. This approach allows secure connectors to provide security services to business application modules built separately while focusing on separately built functional logic for

social interaction with software connectors.

An approach for complex applications by designing application needs and designs independent of security requirements and designs using the UML notation is defined in previous works (Shin, M. E et al. 2016) by the authors. In cases of security use, security specifications are identified and inserted in artefacts of security services. If a system requires security services, cases of security use are applied at extension points from a non-secure business application scenario. Nevertheless, the writers paid relatively less attention to the secure technology environments in which secure software design would suit security requirements (Shin, M. E et al 2018).

The composition of the component concept moves around the one or two components that have been combined. These encapsulate the composite properties, which can be either a tightly coupled or a loosely coupled integration. The idea of a secure connector is more likely to be designed to keep in view computer security aspects such as confidentiality, integrity, authentication, authorization, *and* non-repudiation. The need for the idea to extend over here is to protect business component based application a Facade component at the top of connector to secure the application security in social e-commerce based ATMs (B2c) model.

3.1 Component based Application Logic

The term ‘business logic’ refers to a particular “service”; this service can be a withdraw payment service. This is defined by the business component class withdraw payment, which is handled through the component class business logic. Each component has business logic offered by a business component that resides in the business domain. The concept of a ‘logical component’ can be defined as a sub-component or part of a sub-system; in both conditions, the component keeps its own identity (Nabi et al 2017). Certain steps need to be followed in order to perform a predefined action using application logic to operate a business process (Shin, M. E., et al 2016). The logical component-ware supports the process of application logic and each component’s business logic to develop set of functionalities, which then further translate it into component’s business processing logic by integrating these components in the n-tier architecture.

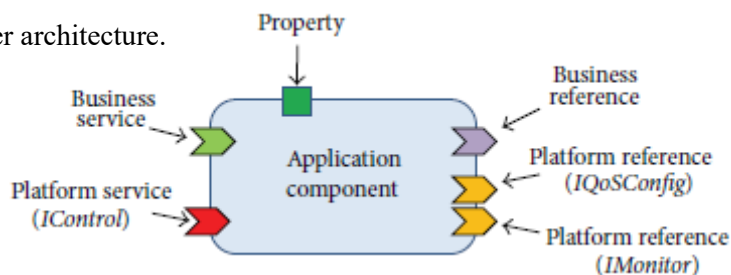


Figure 3: Application Component with Business Logic

SOA is a method of designing software, in which services are provided to the components by application components. The web based banking system is constructed /developed using two sorts of components in the business logic layer of the application. These two kinds of components are: (1) Business Processing Components; (2) Business Entity Components. The first category of components deals with service that is requested by end-users through the published user-interface. They decide the function of business entity components that definitely will be called or invoked and operated. They are persistent components that keep their state stored by the application and are part of application domain (Nabi et al 2017).

Research explains that recent approaches of the component based social e-commerce banking system present a serious lack of security properties at the level of component based secure design applications, which may be connector based logical security, The drawback here is that security is mainly designed as a component offer and the required interfaces glue based composition that has a gap of security from session development of transaction and encapsulate the functionality. At this stage, further design is needed to provide a secure session facade component approach at the top of secure connectors by encapsulating the security functionality to minimize the complexity of component based composite applications and business application logic (Nabi et al 2020).

3.2 Taxonomy of Software Connector

To properly understand the connector definition, the basic tasks a connector can perform must be defined and analysed; here, the software communication taxonomy provided in Shin et al (2017) can be used for guidance. We have chosen the connector tasks listed below, usually the core tasks, from the main service categories and the specific connector styles of this taxonomy.

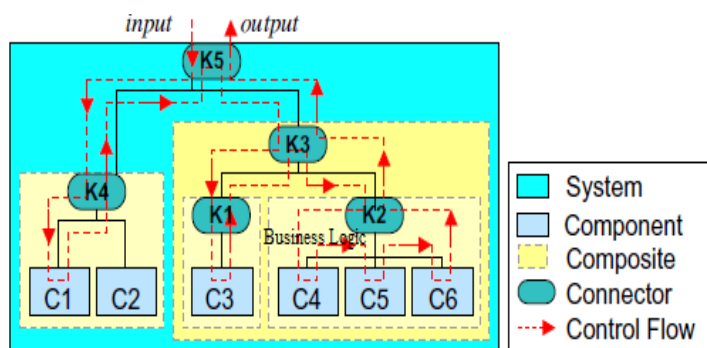


Figure 4: Connector Role in Component Based System and Application Logic

Data transmission and control: A link defines the protocols that can be used to track and/or transfer data (e.g. process call, event management and data stream). Each of these systems has certain features and characteristics,

for example local or remote procedure calls. As for RPC, it can be implemented with different types of middleware. Event management may also be based on an event server, a consolidated event queue, and so on.

Adapting the interface and converting the data: In consideration of the need to tie two (or even more) modules not originally designed to interoperate, an adapter in the connector specification is a good concept. As stated in Baker (2014), an adapter and/or data converter automatically or semi automatically generated mechanism is available to decide (and challenge) the alternative.

Coordination of connectivity and communication: Method call ordering on the interface of a system is important in general (the definition of a protocol in (Gomaa 2011)). A functional feature specification (e.g. interface, structure and design protocols (Plasil et al 1999) Wright CSP based adhesives and calculation) typically specifies the permissible orders. Another function of the connector is to manage the connector and match-enforce conformity with the system protocol (API set). For example, consider using a server node, which is used in a single-threaded environment, for a multi-client threaded environment. Intercepting correspondence: Since connectors mediate all communications between components in the network, a component communication interception frame (without being aware of participants) may help implement various filters (including cryptography applications, data compression, load control, debugging).

3.3 Connector Architecture and Lifecycle

A simple or compound connector may consist of an architecture design framework. The interior elements of a basic connector architecture are instances of primitive elements only (figure 5a). Types of primitive elements (generic forms are typically used—the interface type is permitted as well as the property parameters) (Taha et al 2017). A comprehensive specification of the semantics of each basic element type is provided by mapping the underlying context in addition to a functional specification in plain English.

"The standard features of the RPC are stub or skeleton components."

For example, their remote interface type (specified as property parameter) and the underlying application model parameterize each of the components.

For each supported application platform (CORBA, Java RMI, etc.) there is mapping of stub and skeleton element types. Internal components of a composite connector are instances of other connector types and/or components (Figure 5b). This definition allows for the development of complex connectors that represent the hierarchical existence of component interactions with hierarchically organised architectures (Nabi 2011).

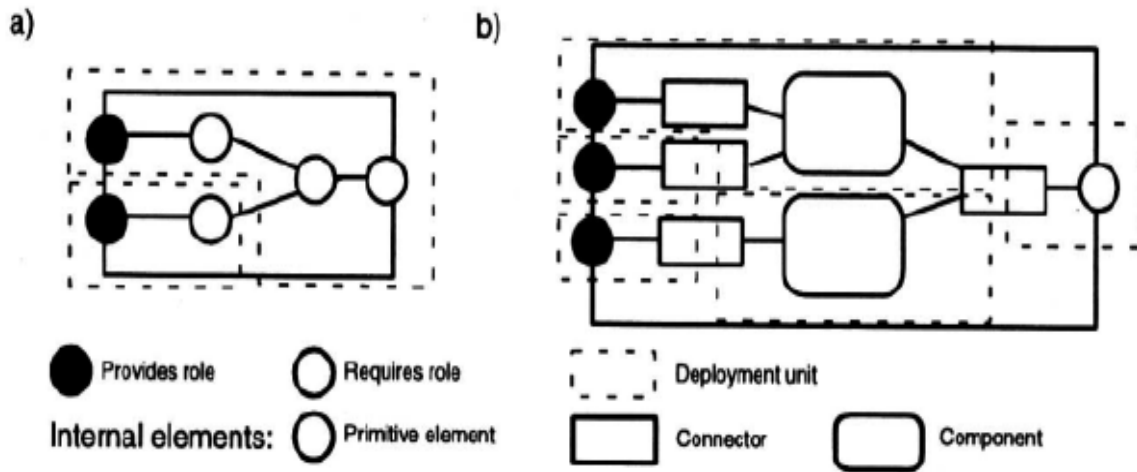


Figure 5: Connectr Model (a) Simple Architecture (b) Compound Architecture

4. An Approach Designing a Secure Facade Based Connector

Social e-commerce application model development components security services are interactive applications that can be applied through different security technologies to accomplish a security goal, such as authentication, authorization, anonymity, completeness, compatibility and non-repudiation. A protection service can be implemented using different security techniques, each addressing a particular security strategy that performs a security service. For example, you can use a symmetrical encryption protection pattern or an asymmetric authentication encryption template to construct a confidential security service (as mentioned in figure 6).

A secure connector is created separately by analysing the transmission pattern of message and security patterns given by the application components. A secure connector is a distributed connector consisting of a secured transmission sender connector and a secure receiver inter-connector that is linked with a session based facade component acting with a connector, as shown in figure 6.

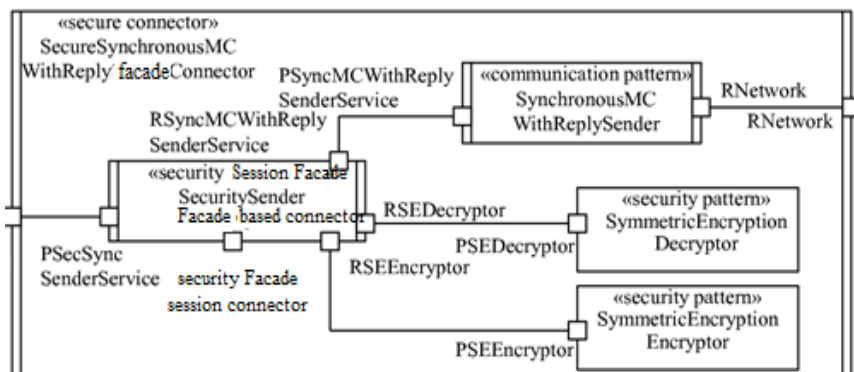


Figure 6: Designing Facade Based Secure Connector

The UML stereotype is used to identify each secure connection, which clearly identifies its position in the architecture of applications. A security controller, one or more security objects and a contact object are secure sender or receiver connectors, which are linked with the all other connectors encapsulated in by a security facade component. In this way, the secure session of transmission between the component and business logic is organised, which is displayed in figure 6. A security coordinator is designed to combine contact models and protection architectures, which are chosen to be used in a modular way, establishing the integration of the security pattern and the communication pattern through coordination among the connectors.

The secure connector may be a sender security coordinator or a receiver coordinator, as shown in figure 6. When a communication pattern component (CPC) and one or more security pattern components (SPCs) are chosen for a connector, the security sender and receiver coordinators need to be configured for every reusable secure connector. Therefore, a framework can be built for each contact style for the high-level security coordinator. For each interchangeable secure connector, the design is tailored according to the chosen security pattern(s).

4.1 UML based ATM Secure Facade Based Connectors Modeling

The example of social interaction with an ATM model is practiced through a facade based secure connector approach, demonstrating that composition connectors are n-ary connectors used to support component composition as it is explained in the below diagram. These connectors communicate and coordinate with each other through defined inter-connectors as shown in figure 7. The ATM client component establishes the session through the secure facade component, which encapsulates the secure sender service based on secure connectors and then proceeds to the process of encryption and receiver decryption to be called by the mechanism. The encryption technique is used for symmetric algorithm security pattern connector components, both encryptors and decryptors, to provide the security for the business process of the ATM service method. The approach taken as a facade session based connector condition supports the ATM process from client to server.

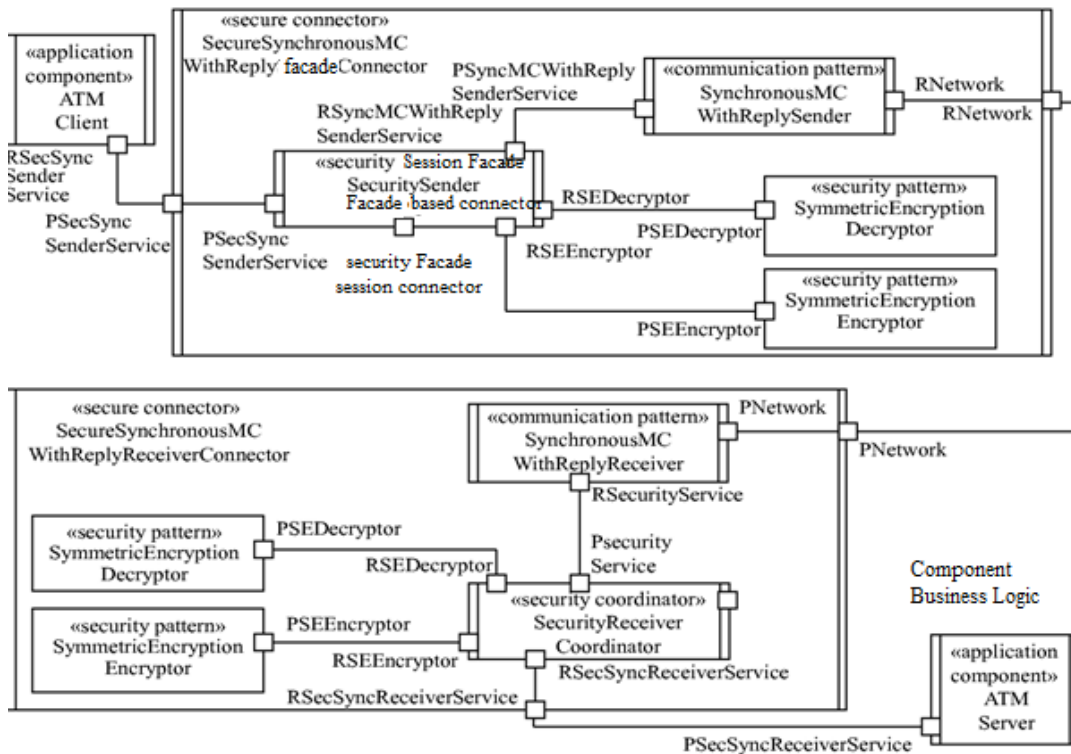


Figure 7: UML Base ATM Secure Facade Connector Design Model

The above mentioned figure 7 shows a secured synchronous response message Security Connector, which is used to validate a confidentiality service ATM application through a Personal Identification Number (PIN). The customer and ATM Server components synchronously and confidentially communicate with response sender / receiver connectors through their protected, synchronous message transmission, each encapsulating a security service cryptography object and an element for decryption protection for a confidentiality function.

A Security Sender Coordinator and a Security Receiver coordinator object, to coordinate security service objects, are included in the secure synchronous message communication with reply sender and receiver connectors. The synchronous message communication with response is encapsulated by a secure synchronous messages communication with the reply sender connector, while the secure message communication with the reply receiver connector encapsulates synchronous message communications with the response receiver template item.

From another conceptual study view, figure 7 displays an abstract and high-level layer overview of the secure façade connector with the SEP for service requests and a response, which can be used by an ATM network. This clearly illustrates the security pattern in a formulated function of connectors. The service request is encoded by SEO on the secure facade-sender connector; while the ATM client component transmits a service request to the

ATM server, so that the symmetric encryption authentication method for the SED component is chosen and then sent to the ATM server component, assuming that the ATM server component has the symmetrical encryption security pattern feature.

In the secure facade sender connector, the reply is encrypted by the SED component and sent to the ATM client component when the application logic processes any transactions made by the machine. The facade secure session manages the security for the component based application logic for the ATM service as it is depicted in diagram 7. This approach is unique in its approach as it coordinates with all connectors of the application component and their underlying security application functionality and logic to make it secure.

It may be postulated that the facade session based approach is more secure as it has the capability to accommodate the security policy for application underlying logic and other glued components, which are integrated through a secure connector approach as defined in the above diagram. The practical application of this model has been tested in a Java bean business component environment in a banking organization to test the applicability of the proposed idea of a secure facade based component acting with a connector in ATM social e-commerce (B2c).

4.2 Technical analysis of data transmission Packets in Proposed ATM Model

This process is defined as the technical analysis view while having to follow the ATM data process in the event of social interaction. The back-end process protocol used, the length of each packet and the additional information needed for each important packet is shown in figures 8 and 9. The packet number and time stamp have been clearly indicated to provide enough space for the relevant information to be displayed in the figure below. Additional packet captures will continue with the ATM startup that runs to get a clearer picture of which packets make up the initial contact with the server. It is also important to investigate the differences between startup contact packets and those that represent a customer-initiated transaction, if any exists.

Source	Destination	Protocol	Length	Info
192.168.0.7	206.71.17.21	TLSv1.2	361	Client Hello
206.71.17.21	192.168.0.7	TCP	1514	443 > 49165 [PSH, ACK] Seq=1 Ack=308 Win=35381 Len=1460 [TCP segment of a reassembled PDU]
206.71.17.21	192.168.0.7	TCP	1514	443 > 49165 [PSH, ACK] Seq=1461 Ack=308 Win=35381 Len=1460 [TCP segment of a reassembled PDU]
206.71.17.21	192.168.0.7	TLSv1.2	392	Server Hello, Certificate
206.71.17.21	192.168.0.7	TLSv1.2	396	Server Key Exchange, Server Hello Done
192.168.0.7	206.71.17.21	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
206.71.17.21	192.168.0.7	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
192.168.0.7	206.71.17.21	TLSv1.2	731	Application Data
206.71.17.21	192.168.0.7	TLSv1.2	221	Application Data
192.168.0.7	206.71.17.21	TLSv1.2	85	Encrypted Alert

Figure 8: ATM Network Transmisson Packet Information

The facade-based protected connector provides the service that is measured on each filtered data capture to contain only the packets necessary for the ATM and the processing system transaction. The overhead packets have been eliminated, still leaving the packets that handled the transaction for each of the above mentioned processes and number of transactions. During each transaction, this technique is used to measure the packets are in terms of quantity, size and volume. There are some differences with the number and size of packets in relation to the ATM startup packet. The data obtained, however, can indicate some effects on the congested section of the network connected with the ATM.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000..	192.168.0.1	192.168.0.7	DHCP	342	DHCP Offer - Transaction ID 0x61dd1155
2	0.6384..	192.168.0.1	192.168.0.7	DHCP	342	DHCP Offer - Transaction ID 0x61dd1155
3	0.6490..	192.168.0.1	192.168.0.7	DHCP	342	DHCP ACK - Transaction ID 0x61dd1155
4	39.458..	192.168.0.1	192.168.0.7	DHCP	342	DHCP Offer - Transaction ID 0x61dd1156
5	39.490..	192.168.0.1	192.168.0.7	DHCP	342	DHCP Offer - Transaction ID 0x61dd1156
6	116.85..	192.168.0.7	192.168.0.1	DHCP	342	DHCP ACK - Transaction ID 0x61dd1156
7	116.90..	192.168.0.1	192.168.0.7	DNS	81	Standard query 0x0051 A EFTDEBITATM.FNFIS.COM
8	116.91..	192.168.0.7	206.71.17.21	DNS	97	Standard query response 0x0051 A EFTDEBITATM.FNFIS.COM A 206.71.17.21
9	116.94..	206.71.17.21	192.168.0.7	TCP	62	49164 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1
10	116.94..	192.168.0.7	206.71.17.21	TCP	60	443 → 49164 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len=0 MSS=1436
11	117.47..	192.168.0.7	206.71.17.21	TCP	60	49164 → 443 [ACK] Seq=1 Ack=1 Win=33028 Len=0
12	117.50..	206.71.17.21	192.168.0.7	TLSv1.2	361	Client Hello
13	117.50..	206.71.17.21	192.168.0.7	TLSv1.2	1514	[TCP Previous segment not captured], Ignored Unknown Record
14	117.50..	192.168.0.7	206.71.17.21	TLSv1.2	392	Ignored Unknown Record
15	117.50..	192.168.0.7	206.71.17.21	TCP	60	[TCP Dup ACK 10#1] 49164 → 443 [ACK] Seq=308 Ack=1 Win=33028 Len=0
16	117.51..	206.71.17.21	192.168.0.7	TLSv1.2	396	Server Key Exchange, Server Hello Done
17	117.51..	192.168.0.7	206.71.17.21	TCP	60	[TCP Dup ACK 10#2] 49164 → 443 [ACK] Seq=308 Ack=1 Win=33028 Len=0
18	117.55..	206.71.17.21	192.168.0.7	TCP	1514	[TCP Retransmission] 443 → 49164 [PSH, ACK] Seq=1 Ack=308 Win=35381 Len=0
19	118.29..	192.168.0.7	206.71.17.21	TCP	60	49164 → 443 [ACK] Seq=308 Ack=3601 Win=32348 Len=0
20	118.32..	206.71.17.21	192.168.0.7	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
21	118.32..	206.71.17.21	192.168.0.7	TCP	60	443 → 49164 [ACK] Seq=3601 Ack=434 Win=35255 Len=0

Figure 9: ATM Authentication Packet Transmission Process

The test transactions are carried out after checking that the ATM is confirmed to the authentication server followed by the facade secure session. From the original authentication through the secure session facade connector, the ATM has an address of 192.168.0.7 and has an authentication key at IP address 206.71.17.21. These addresses are then used as buffers for each capture to a stream containing only packets and addresses from each transaction.

In order to be checked, the system process model for the above mentioned facade session based functional novel method follow checking that the ATM is authenticated to the authentication server; the user should upload the XML scheme in a specific format to the model checking process.

In a certain format, the XML schema should be given. The schema can be accessed through a computational process. The SRS tool kit takes up this description and extracts the information involved (i.e. tables, attributes and their data types, main keys and interactions), stores the extracted data as items in classes generated for the phase in comparison process between input & output data. The arrangement of information into business plans makes this goal possible The below figure 10 displays the front –end process that performs the value of the ATM model process input and output test value. In this way, the proposed test process outcome will be analysed in a secure way, as mentioned in figure 10.

Symbol	Activity name	Sequence No.	Dependency	Input	Expected output
A	Insert card	1			
B	Validate (card)	2	A	Card	True (enter the pin) False (End)
C	Enter pin	3	B	Pin	Pin
D	Pin	4	C	User types the pin	Receives the pin and send it to the server
E	Verify (pin)	5	D	Pin	True (enter amount) False (End)
F	Enter amount	6	E	Amount	Amount
G	Amount	7	F	User types the amount	Receives the amount and send it to the server
H	Check balance (amount)	8	G	Amount	True (take cash) False (End)
I	Take cash	9	H	Cash	End
J	End	10	I		

Figure 10: Extracted Information from XML Schema Scenario Based Validation

Therefore, this is a logical technique; as the adequacy of the new connector design remains unexplored, we have attempted to fill this gap through the facade software connector approach that encapsulates the functionality of all other connectors and provides a secure communication pattern in the B2c social e-commerce application development. It is also important to run a comparison of this technique through the software connector perspective rather than the network associated protocol security because the issue identified is the facade based secure approach that constitutes the idea explained above.

4.3 Centralizes Security Session Facade

The application's security policies can be operated at the facade level of the session, because this is the clients' level tier. The session facade's coarse grain exposure is harder, more realistic, than at the participating component level, to identify security policies at this level.

Secure connector supports business components that provide sophisticated control points and security, which is easier to manage for session facade. This provides coarse grain access because relatively fewer coarse grain methods have to be handled securely.

4.4 Security Analysis Using Session Facade

The approach of using the session facade is a software design pattern through which we can encapsulate the steps of the session process in a single "session call" () method. This is done at the level of session bean that delegates the multiple-steps process to encapsulate the calling functionality of secondary connectors in a frame to execute

the application processing logic. Within the session Facade implementation, the “`identifycustomer`” () secure method is a remote call, whereas “`findbycustomerID`” () and “`getpassword`” () are local calls. It is important to note that the session facade improves the performance and security to simplify the interface to the client. This increases the benefits of facade based solutions in a secure connector design approach for B2c e-commerce solutions.

5.5 A Comparison of Software Connector Modeling Approaches

The existing approaches based on some other methods (Shin et al 2018; Baker et al 2014; Shin et al 2016; Shin et al 2017; Shin et al 2016; Derdour et al 2015) shed light on the modeling of component connector interaction. Perry et al. provided a high-level description of the architecture function of software connectors (Perry et al 2014). In order to define architectural components, including connectors, Kazman et al. proposed outside canonical features (Kazman et al 2016). Finally, previous attempts had model connectors at the interdependency stage of the module.

The software modelling languages typically define relations between the modules at the process call level and the access to shared data (Derdour et al 2015). For this purpose, architecture and description languages have been specifically developed to enable development of more complex and efficient connectors, such as UML Sec. We have used the power of UMLSec for modelling the ATM B2c architectural design through a façade based component connector. This is empowered by existing ADLs primarily focused on checking the properties of modeling behaviour of each connector in the system. This helped to clearly define the connectors’ export services, their mechanics, interaction protocols and interaction usage and development constraints within the ATM example in figure 6. As the adequacy of the new connector design remains unexplored, we have attempted to fill this gap through the façade software connector approach that encapsulates the other connector functionality and provide a secure communication pattern in B2c social e-commerce application development.

5. Discussion

The approach suggested in this paper differs from other security approaches. Our approach employs secured a facade based connector that separates application issues from security issues in a secure software architecture for social distributed applications based on components. Communication problems are coordinated from security

issues within secure connectors. Suryanarayana, et al (2004) focuses on the trust management system where security services are embedded in free, unified application components in software protection technologies.

This paper provides the secure connectors with protection between components, i.e. for stable interactions between application components. The secure facade based connectors provide security services for communication with other components to the application components. They also delegate the processes of authentication, authorization, confidentiality. Integrity and non-repudiation services can be built in protected connectors or delegated to internal and external security components via secure connectors while processing the social interaction ATM (B2c) Model.

Composing and managing the interactions is not a trivial activity in component-based design. In the present state-of-the-art components-based models, the design and interface types are primarily port-to-port or method-call-based. Both styles confer a dynamic pattern because, due to the number of system calls, ports and connectors, the number of contacts will drastically increase. A simple and coherent model with such a complexity is to be avoided logical functions actions are required.

Since the facade of the session reflects the process for use cases, transaction processing on the facade level of the session is more rational. Similar to centralized security, centralized transaction management provides benefits. The facade provides a central position for the administration and interpretation of transaction power. Transaction processing on participating business components is much more relevant internally, in particular since it is smaller than the facade. However, the client has access to the Java bean business component indirectly through communication of secure connectors; those use a session facade to place the burden of demarcation on the client side tier. The function of connectors is expanded in this paper by adding security patterns as design side concept to the connectors via a secure session facade component acting with the secure connector's social interaction with the ATM B2c Model.

5.1 Research Contribution

This paper addresses the security issues that may also be encapsulated in the software application logic through secure façade based connector that is called security in connector based component application. This approach can deal with the Subversion Attack Vulnerability in application logic that may cause of component event based

calling inter communication method. This leads to a process at the time of loosely coupled or tightly coupled integration based on interfaces of components as defined in figure 1. The solution defined related to this problem through the modeling ATM case study in figure 6 and 7, which is different from other approaches in this domain.

6. Conclusion

This paper addresses the development of facade based secure connectors for the design of secure software architectures for social distributed commercial applications such as the ATM Model. The secure connectors are built separately from the business application components and take into consideration the security services that application components need, as well as the communication patterns for sending secure messages and replies between the components (if desired).

The security services provided in software environments by the application components for business applications are designed to provide secure facade based connectors. These secure connectors contain security-related artefacts for the isolation of software modules from security services. The security artefacts are allowed only when application components need the required security resources, such as authentication, authorization, protection, confidentiality and non-repudiation. Through distinguishing security concerns from implementation problems, secure facade based process encapsulating the connector's secure function that will make complex networks more easily maintained and flexible as mentioned in the ATM social e-commerce application example.

Declaration of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

The authors declare the following financial interests/personal relationships, which may be considered as potential competing interests:

Ethics approval

There is no human and animal involved in this research therefore no need of ethical approval for this research.

References

- Albassam, E., Porter, J., Gomaa, H., Menascé, D.A, 2017: DARE: A Distributed Adaptation and Failure Recovery Framework for Software Architectures. In: 14th IEEE International Conference on Autonomic Computing and Communications (ICAC), Columbus.
- Al-Azzani.S., Bahsoon.R, 2012 SecArch: Architecture-Level Evaluation and Testing for Security, in Joint Working IEEE/IFIP Conf. Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), *pp*: 51-60, ISBN 978-1-4673-2809-8.
- Almeida.F ., 2017 Concept and Dimensions of Web 4.0, International Journal of Computers and Technology, Vol 16 Issue7: pp 7040-7046 DOI: 10.24297/ijct.v16i7.6446
- Baker, C., Shin M., 2014., Aspect-Oriented Secure Connectors for Implementation of Secure Software Architecture, International Conference on Software Engineering and Knowledge Engineering (SEKE'2014), Vancouver, Canada, July 1-3.
- Derdour.M., Alti.A., Gasmi.M.,, Roose.P., 2015 Security Architecture Metamodel for Model Driven Security,Journal of Innovation in Digital Ecosystems,Volume 2, Issues 1–2, Pages 55-70.
- Gomaa.H., 2011 Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures (Cambridge University Press, Cambridge, UK.
- Kazman.R., Cervantes.H, 2016 Designing Software Architectures: A Practical Approach, Addison Wesley, p 97-102.
- Nabi.F., Yong.J., Tao.X., 2020 A Novel Approach for Component based Application Logic Event Attack Modeling, Vol. 22, No. 3, pp. 435-441.
- Nabi.F., 2011 Designing a Framework Method for Secure. Business Application Logic Integrity in e-Commerce Systems, International Journal of Network Security, Vol.12, No.1, PP.29–41, Jan. 101.
- Nabi.M.M., Nabi.f., 2017 A Process of Security Assurance Properties Unification for Application Logic “, International Journal of Electronics and Information Engineering, Vol.6, No.1, PP.40-48 .
- Plasil, F., Besta, M., Visnovsky, S.1999: Bounding Component Behavior via Protocols. In Proceedings of TOOLS USA '99, Santa Barbara, USA.

- Perry,D.E. 2014, Software Architecture and its Relevance to Software Engineering, Invited Talk. Second International Conference on Coordination Models and Languages, Berlin, Germany.
- Shin, M., Gomaa, H., Pathirage, D, 2018: A Software Product Line Approach for Feature Modeling and Design of Secure Connectors. In: The 13th International Conference on Software Technologies (ICSOFT). SciTe-Press, Porto.
- Shin, M. E., Gomaa, H., Pathirage, D., 2016. Reusable Secure Connectors for Secure Software Architecture, 15th International Conference on Software Reuse, Limassol, Cyprus,PP 25-35
- Shin, M.E., Gomaa, H., Pathirage, D.2017: Model-based design of reusable secure connectors. In: 4th International Workshop on Interplay of Model-Driven and Component-Based Software Engineering (Mod Comp), Austin.
- Shin, M. E., Gomaa, H., Pathirage, D., Baker, C., Malhotra, B., 2016. Design of Secure Software Architectures with Secure Connectors, International Journal of Software Engineering and Knowledge Engineering, Vol. 26, No. 5, pp 769–805.
- Suryanarayana.G., Erenkrantz J. R., Hendrickson S. A., Taylor R. N., 2004 PACE: An architectural style for trust management in decentralized applications, in Proc. 4th Working IEEE/IFIP Conf. Software Architecture (WICSA '04), (IEEE Computer Society, Washington, DC, USA, 2004), pp. 221–230.
- Siricharoen.W.V, 2018 Understanding Social Interaction with Human Computer Interaction (HCI) Adaptation, EAI Endorsed Transactions on Context-aware Systems and Applications 6(18):160762
- Taha, A., Trapero, R., Luna, J., Suri, N.2017: A framework for ranking cloud security services. In: International Conference on Services Computing (SCC), pp. 322–329. IEEE, Honolulu.
- Taylor, R. N.; Medvidovic, N.; Dashofy, E. M,2010 software Architecture Foundations Theory and Practice, ISBN 13: 9780470167748; Publisher: Wiley.

CHAPTER 8. RESEARCH RESULTS AND CONCLUSION

8.1 Introduction-related thesis question and contribution to the research

This figure displays the publications that informed the research questions and illustrate and highlights the main contribution of this research through this diagram that shows how application logic in banking applications can be secure. This can be done via the re-use of business component design specifications through the process of vulnerability identification, calcification and modeling. This Chapter also outlines the main goals of this process which have been covered in the following publications, that support each research thesis question as a contribution to the literature in this field.

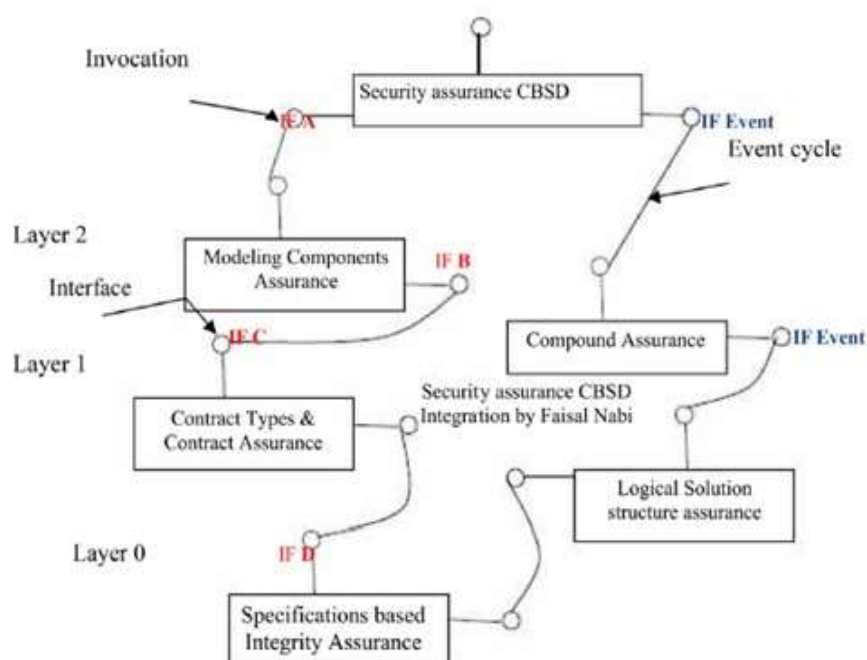


Figure 8.1: Security Assurance Model Projection of Thesis Contribution

The above-mentioned diagram explains the research questions related to the contribution made by this thesis.

For example, security assurance in the CBSD section as displayed in the figure explains the main research question and other parts represent the sub-questions related contribution/s. These five papers cover the aspects cited in the above-mentioned Diagram 8.1

- This main research question has been answered in terms of SCA that supports service-oriented application logic, in this Chapter, it is stated that existing literature that illustrates the background of modern research is significant to explore. This chapter carefully explains the nature of the problem and defines a model as a contribution which is mentioned in Figure 6. Analysis attack event process model that is a contribution to the review paper, while comparing other methods.
- Research sub-questions 1 and 2 address the component-based logical vulnerability taxonomy, the main purpose of this is to address the relationship between technical and logical vulnerability and validates the proposed model for a CBSD taxonomy, which is a major contribution as presented in Chapters 4 A and 4B. These publications discuss the contract type and contract assurance gap-based problem.
- Research Question 3 addresses the relationship to define how Question 3 was answered through proposed event-based attack modeling by using a case study-based modeling and designed solution about design flaws detection in real-world scenarios. This is presented in Chapter 5 which also demonstrates the modeling component of assurance-based integration.
- This research question 4 address the relationship between the service-oriented component-based application logic that defines application logic vulnerability detection through proposed modeling in terms of social e-banking and proposed and validated modeling for CBSD-based business logic flaws. This is the major contribution of this thesis as well as this produced in the Q 1 publication that presents the specification-based integration.
- In this thesis, research question 5 addresses social interaction in banking application logic based on connectors. A novel technique has been addressed and formulated the application-based logical vulnerability through a façade component connector that is its kind of new technique and recently accepted in IJNS Q 2 Journal. This presents logical solution assurance.

Whereas the compound assurance in Figure 8.1 presents the overall contribution of solution strategy towards displaying the complete solution in CBSD security assurance.

Therefore, it is concluded that security assurance is based on the proposed model as displayed in Figure 8.1. The research thesis results and discussion section covers each paper, with a related explanation that shows how it matches with Diagram 8.1 and its contribution to the research.

Research Thesis Results and Discussion

The research thesis discussion raised a number of points in relation to the analysis and findings regarding each of the five papers. Further analysis is offered here to clarify our research contribution.

Findings and Discussion Points

The first and most important contribution is the analysis of requirements of event-based distributed system design, where SCA (service component architecture) is used. A service component paradigm allows for composite application development and application reusability. However, while creating service-oriented component application logic, security in event-based communication in components interaction models is usually emphasized on the top layer in SCA. As highlighted by Sonar Qube (2017), existing system analytics focusing on the service component do not focus on event attacks, and nor do they appropriately detect vulnerabilities component-by-component, as demonstrated in OWASP's Orizon Project (2018) and Xanitizer (2017).

This research has explained the concerns with composite applications and event-based attacks in the service component architectural paradigm. We have accomplished this goal by analyzing, assessing, and modeling strategies in-service component application functionality, as well as in application components that create, consume, and process events.

The research paper has examined in depth the most recent security changes in the field of component inter-communication event base model for service component architecture. The study will assist users in comprehending the hazards and forms of attacks that a component-based application may encounter while working with application logic through event-based service in component architecture. Our research has established a new standard for future research in the event-based security paradigm.

The primary contribution of this research is that it identifies current security efforts that do not

focus on event attacks nor correctly detect component-by-component inter-communication event models, and it produces a solution to the problem through three-dimensional methods analysis and comparison. It also paves the way for future research in this area by researchers, with the goal of making event-based distributed systems more secure.

The thesis represents the second contribution in the field of application logic security vulnerabilities classification.

New advancements in the field of e-commerce software technology have also offered numerous benefits; yet, the development process is always fraught, from the design phase to the implementation phase. Software flaws and defects exacerbate dependability and security concerns, requiring a solution in application business logic that is based on a logical component-ware combination.

The study addresses the issue of logical vulnerability classification in component-based web applications by identifying Attack Group Method and categorizing two different types of vulnerabilities in component-based applications. Using an empirical methodology-based classification strategy for a logical group attack approach, a novel classification scheme is presented and built.

The security dimensions are features and attributes of the system that impact the security group's ability to understand and make improvements to the system. This is based on a thorough grasp of the vulnerability category and its subclasses.

The security dimensions have a direct impact on the security group's ability to analyze the attack vector in relation to an application or system security. This can be done both logically and technically, with each part of both being classified before the security issues are addressed.

Based on our findings, the proposed model offers classification and characterization of two unique categories of vulnerability issues/problems: "Technical Vulnerabilities" and "Logical Vulnerabilities." These vulnerabilities are categorized according to the attack method, which is listed in the vulnerability model, and related to the attack pattern technique.

The thesis represents the third research contribution as event-based attack modeling in the context of application logic vulnerability in a component-based banking application while re-using design specifications within existing component logic. This may cause a subversion attack in application

business logic.

Through a novel vulnerability modeling technique, an Event that targets a specific system must be identified. In component-based application logic vulnerabilities, current research does not provide event attack modeling. To detect such flaws, it is crucial to determine which component set off the Event that allowed the system to be exploited.

This study presents Event-Based Attack Modeling, which is particularly useful in the context of component-based software subversion logic attacks in the Business Application Logic category. This will aid in the creation and reuse of components based on the functional logic of current applications.

Attack modeling is a very effective technique for analyzing attacks and preventing the situation from becoming worse. As a result, a variety of strategies have been developed to deal with attack modeling in the component-based system domain. Design flaws or logical flaws are examples of logical vulnerability.

Noting the difficulty in identifying and modeling them, a technique that can deal with logical flaw-based vulnerabilities is necessary. In this work, we provide “Event Attack Modeling,” a unique modeling approach that employs the Uppaal Tool to model a vulnerability and its attack flow through an attack-triggered component within an application in a real-time scenario.

The fourth contribution of this research is that it identifies security assurance methodologies in service component-oriented applications to be utilized through threat modeling and a novel component fault-detection model.

Using a UML secure design approach, this concept is expanded to include the modeling component and its applications. The methodologies used in this research to validate the strategy include verification and validation for security by design testing in rapidly developed component-based social e-Commerce banking applications, to avoid the business logic design flaw problem. Modern social e-commerce practices are a subset of e-Commerce that emphasizes security framework protocols such as secure transactional protocols, cryptographic techniques, and sanitization criteria. These procedures are expected to ensure the stability of social e-commerce-based applications. The key challenge in designing these techniques is the composition of software components and integration flaws. The primary focus of these techniques is on software

component composition and integration problems, which are frequently overlooked in business application logic. These issues have the potential to negate the impact of modern information security approaches. The component's logic subversion on the server-side is the weakest link in social e-Commerce banking application logic security solutions.

This paper covers a specific issue in application logic security known as a subversion attack, which can be classified as a design flaw. Many traditional security techniques routinely utilized in modern e-Commerce systems cannot overcome this type of security flaw.

The security assurance methodology used in this research to validate the strategy includes verification and validation for security by design testing in rapidly developed component-based social media e-Commerce applications, to avoid the business logic design flaw problem.

The fifth research contribution is that it describes a secure social distributed applications software architecture that contains components. Our approach uses a façade-based connector that isolates application logic difficulties from security problems. This research addresses security and privacy issues related to social interactions with the ATM model that may also be encapsulated in the software application logic through a secure façade-based connector that is known to increase security in a connector-based social e-commerce (B2c) application. This will target the Subversion Attack Vulnerability found in the application logic of B2c systems and may allow a component-based approach called the inter-communication method.

This research work was developed in the context of our focusing on the modeling technique, by introducing security-modeling aspects into component service architecture in order to expand on the research work in paper 5 (Security assurance methodology). As an example, the B2c ATM model will be featured as part of the security feature-based UML Sec modeling, in turn, was demonstrated through social interactions with e-commerce software security modeling that justifies the secure application logic.

Within a secure socially distributed applications software architecture that contains components, our approach uses a façade-based connector that isolates application logic difficulties from security problems.

Secure connectors that coordinate communication have security challenges. The trust management system is the focus of this method, which combines security services with open,

unified information defence technology application components and their underlying logic. Security connectors with component security, i.e. for reliable connections among application components, are also discussed in this article. Secure façade-based connectors ensure that a component's business logic communicates with other components in the application.

They also assign authentication, permission and confidentiality procedures. Integrity and non-repudiation services can be inserted into secure connectors or transferred through secure connectors to internal and external security components during the processing of social interaction ATM (B2c). The façade serves as a focal point for the administration of transaction power and perception. Transaction processing through participating business components is far more important internally, especially because components are smaller than the façade. Through secure connector communication, the client has indirect access to the Java bean business component; these use a session façade to transfer the burden of demarcation to the client tier.

Discussion

Figure 8.1 projects that all publications based on the research questions illustrate and highlight the main contribution through this diagram, specifically how application logic in banking applications can be developed securely, while re-using a business component design specification.

This process is particularly can be used for the banking domain. However, an example of NASA project is presented, so there is the possibility of using this model for mission-critical systems may also be possible. The research findings can be applied to the development of new secure applications based on the existing business logic in the banking industry at the design, development and testing stages as presented in publication 5 (chapter 6) demonstrate the reusability of business components and validation model through Uppaal Tool and UML modeling.

This explains the role of the model of each publication which expresses the contribution of this thesis. The model Figure 8.1 expresses the process of security assurance such as design specification and contact type of component interface assurance, which then leads to specification that is based on integrity assurance which further formulates the logical structure of logical function of the component. This process defines the CBSD assurance process. This model depicts the overall process and makes links with the role defined in service component architecture-based social banking application logic.

It is important to consider that such systems need security at the design stage, so that the logical structure of the application follows the component business logic of each. Any integration fault inevitably leads to a security flaw in the design. This causes a bypass of security mechanisms in ways defined in this research, and set out in case study examples exploring social interaction in e-commerce banking applications.

Therefore, it is concluded that this research has covered the gap between design specification of business component integration faults and re-usability of business process functions of the component. The further gap that has been closed is CBSD-based social e- banking security in terms of design-based business logic, which is not discussed this before as projected in publication 5 of this thesis. Future research and further implications are subject to continue in future for those who are interested in this research.

Research Recommendations

Therefore, in this case, bank developers needs to focus on the purpose and type of behaviour specification of re-used component in terms of requirement specification in each layer (an n-tier CBS application), component functional specification boundary conditions and knowledge of its defined interfaces within the systems, and if ignored design specification for each layer component. The failure to meet the required specifications as compared to its functional specification based on design specification, for the purpose it was designed, based on its current logical component-based composition in the system. This gave birth to the design flaw in the component ware. This all process of problem generated business logic vulnerability. This is a very serious violation of the principle “specification purpose” in component-oriented logical component-ware at the time of business interface-driven integration, while ignoring usage contract type specifications. It’s also a case of “Test by Contract”, which means not only that a design specification for the component is needed for consideration but also contract establishment among the interfaces and their designed logic throughout the process. Together this creates security assurance among the interface-focused designed components behaviour through e-process while developing component-oriented business logic. It is important to consider the boundary condition of logical attack falls in between functional specification and design specification. Therefore, our proposed solution will help to mitigate the attack triggering method “Event-based-generated” e-process flow to violate business logic.

This research will recommend following the developers to create business logic free from flaws,

while using existing logic.

8.2 Conclusion

The service component architecture provides a foundation to develop application logic design and event-based communication in service composite applications. At the same time, there are security issues regarding service integration and composite application component re-use, which often suggests a design flaw in service component-oriented application logic design. The research has proved that it is difficult in service component architecture to reuse the design specifications of existing system component logic, while reusing design specifications to integrate new services through business application logic.

Specifically in the banking domain, it is evident that system design that uses existing components for another service must organize design specification and business process integration according to the business logic of each component.

Therefore, it is imperative to design a solution based on a methodology that will strengthen security. A design method approach for service component-oriented e-commerce applications can be considered in the context of social e-commerce banking case studies by using a modeling methodology that helps to generate and automate the vulnerability through attack scenario modeling (UML Sec & Uppaal Tool) as presented in this thesis.

The research findings are based on the research contribution from Papers 1 to 5 is to propose a security assurance approach for service-component-focused business logic by reusing core service logic. It will also resolve the disparity between traditional viewpoints and security requirements in e-commerce systems in the context of its sub-set social e-commerce. It will increase the degree of security assurance by design modeling such a practice. When practicing the design of service component-focused applications within the e-commerce domain we will use currently available components and deploy business logic into service-based systems.

A second contribution of the research is that it proposes and defines a new taxonomic system of logical vulnerabilities in service component-based middle-tier services, which are often the result of design defects due to COTS or in-house software modules for service integration. The third point of contribution is that attack modeling in the scenario of logical attacks by event-based attack causes feature detection. The fourth contribution is derived from the approach in paper 5,

which targets the security assurance methodology through security modeling UML - see technique. The fifth contribution will be reflected in more depth as part of security feature-based UM - see modeling for a B2c ATM social interaction model.

This thesis carefully examines a pressing issue in component-based service-oriented application logic reuse and security. Other researchers have not succeeded in solving the issues set out above. This indicates a clear research gap, which has been comprehensively addressed through this series of research publications. As a result, and given its technical nature, the proposed work when completed will be a significant achievement in the domain of components and services-based solutions, as well as strengthening their integrity and level of assurance in this domain.

8.2.1 Future Research Direction

This research opens a gateway for further research findings for researchers through the taxonomy which helps to further improve the technique in terms of flaw-free CBS-based J2EE application logic development and especially social interaction interim of social medium-based banking application security through enhancing the suggested model in this research to make more secure banking online.

References

- Algharabat, R. S., & Rana, N. P. (2020). Social commerce in emerging markets and its impact on online community engagement. *Information Systems Frontiers*, Springer, vol. 1, pages 1-22.
- Amirpour, M., Harounabadi, A., & Mirabedini, S. (2016). Service-oriented architecture assessment based on software components. *Decision Science Letters*, 5(1), 109-118.
- Agirre, A., Marcos, M., & Estévez, E. (2012, September). Distributed applications management platform based on Service Component Architecture. In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation*, 1-4.
- Bentounsi, M., Benbernou, S., & Atallah, M. J. (2016). Security-aware business process as a service by hiding provenance. *Computer Standards & Interfaces*, 44, 220-233.
- Ghassan B, Achim H, RafaelValencia G, Jun S, Asif G (2020) Towards an assessment framework of reuse: a knowledge-level analysis approach. *Complex Intell Syst* 6:87–95.
- Gbaffonou, B. A., Lapalme, J., & Champagne, R. (2015). Service-oriented architecture: a mapping study. 2015 *International Conference on Enterprise Systems*, 33-42.
- Hassard, J., & Kelemen, M. (2012). *Encyclopedia of Case Study Research*. SAGE Publications, 49-56.
- Janssen, M. (2014). Exploring the service-oriented enterprise: Drawing lessons from a case study. In *Proceedings of the 41st Annual Hawaii International Conference on Systems*, 101-101). IEEE.
- Jiang, M., & Willey, A. (2005). Architecting systems with components and services. In *IRI-2005 IEEE International Conference on Information Reuse and Integration*, 259-264.
- Jakoubi, S., Tjoa, S., Goluch, S., & Kitzler, G. (2011). Risk-aware business process management—establishing the link between business and security. In *Complex intelligent systems and their applications*, 109-135.
- Kalantari, A., Esmacili, A., & Ibrahim, S. (2013). A Service-Oriented Security Reference Architecture. *International Journal of Advanced Computer Science and Information Technology*, 1, 25-31.
- Karimi, O. (2011). Security model for service-oriented architecture. *Advanced Computing: An International Journal (ACIJ)*, 2(4).
- Laukkanen P, Sinkkonen S, Laukkanen T (2018). Consumer resistance to internet banking: postpones, opponents

and rejectors. *Int J Bank Mark* 26(6):440–455

Luhach, A. K., Dwivedi, S. K., & Jha, C. K. (2014). Designing and implementing the logical security framework for e-commerce based on service-oriented architecture. *International Journal on Soft Computing (IJSC)*, 5(2).

Malohlava, M., Hnetyuka, P., & Bures, T. (2013). Sofa 2 component framework and its ecosystem. *Electronic Notes in Theoretical Computer Science*, 295, 101-106.

Nabi, F., & Nabi, M. M. (2017). A process of security assurance properties unification for application logic. *International Journal of Electronics and Information Engineering*, 6(1), 40-48.

Nabi, F., Yong, J., & Tao, X. (2019). Proposing a secure component-based-application logic and system's integration testing approach. *International Journal of Information and Electronics Engineering*, 11(1), 25-39.

Nabi, F., Yong, J., & Tao, X. (2020). Classification of Logical Vulnerability Based on Group Attacking Method. In: 11th International Conference on Ambient Systems, Networks and Technologies (ANT), Warsaw Poland.

Nurcan, S., & Schmidt, R. (2015). Service-oriented Enterprise-Architecture for enterprise engineering introduction. *Proceedings of the 2015 IEEE 19th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations*, 88-90.

Nabi, F. (2008). Designing Secure Framework Method for Business Application Logic Integrity in e-commerce Systems. Works in Progress *24th Annual Computer Security Applications Conference* Anaheim, California, USA, 8-12.

Nabi, F. (2005). Secure business application logic for e-commerce systems. *Computers & Security*, 24(3), 208-217.

Novak, M., & Švogor, I. (2016). Current usage of component based principles for developing web applications with frameworks: a literature review. *Interdisciplinary Description of Complex Systems: INDECS*, 14(2), 253-276.

- Paik H., Lemos A.L., Barukh M.C., Benatallah B., Natarajan A. (2017) Service Component Architecture (SCA).
In: *Web Service Implementation and Composition Techniques*. Springer, Cham.
https://doi.org/10.1007/978-3-319-55542-3_8
- Raed SA, Nripendra P.R. (2020) Social commerce in emerging markets and its impact on online community engagement. Information. <https://doi.org/10.1007/s10796-020-10041-4>.
- Riad, A. M., Hassan, A., & Hassan, Q. F. (2018). Leveraging SOA in banking systems' integration. *Journal of Applied Economics Science, Romania (JAES)*, 3(2), 4-9.
- Rodriguez, M., Zalama, E., & Gonzalez, I. (2016). Improving the interoperability in the Digital Home through the automatic generation of software adapters. *Revista Iberoamericana De Automatica E Informatica Industrial*, 13(3), 363-369.
- Seinturier, L., Merle, P., Rouvoy, R., Romero, D., Schiavoni, V., & Stefani, J. B. (2017). A component-based middleware platform for reconfigurable service-oriented architectures. *Software: Practice and Experience*, 42(5), 559-583.
- Wang, H., Wang, Y., Taleb, T., & Jiang, X. (2020). Special issue on security and privacy in network computing. *World Wide Web*, 23(2), 951-957.
- Woody, C. (2015), Security Risk Management using the Security Engineering Risk Analysis (SERA) Method, presentation. Annual Computer Security Applications Conference.
- Yaghmaie, A. (2017). How to characterise pure and applied science. *International Studies in the Philosophy of Science*, 31(2), 133-149.
- Johan, S.K.; Mishra, R.K. (2019) Predicting and Accessing Security Features into Component-Based Software Development: A Critical Survey. In *Software Engineering*; Hoda, M.N., Chauhan, N., Quadri, S.M.K., Srivastava, P.R., Eds.; Springer: Singapore; pp. 287–294.
- Johan, S.K.; Mishra, R.K. (2019) A Review on Re-usability of Component Based Software Development. *Reliab. Theory Appl.* 14, 32–36.
- Lau, K.K.; Cola, S. (2017) *An Introduction to Component-Based Software Development*; World Scientific: Singapore.

- D. Migault, M. A. Simplicio, B. M. Barros et al., (2017, June) "A framework for enabling security services collaboration across multiple domains," in Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 999–1010, Atlanta, GA, USA.
- J. Jürjens, K. Schneider, J. Bürger et al., (2019) "Maintaining security in software evolution," in Managed Software Evolution, pp. 207–253, Springer, Cham, Switzerland.
- G. Wangen, C. Hallstensen, and E. Snekkenes, (2018), "A framework for estimating information security risk assessment method completeness," International Journal of Information Security, vol. 17, no. 6, pp. 681–699.
- S. E. Sahin and A. Tosun, (2019) "A conceptual replication on predicting the severity of software vulnerabilities," in Proceedings of the Evaluation and Assessment on Software Engineering, pp. 244–250, Copenhagen, Denmark.
- P. Rotella, (2018, June) "Software security vulnerabilities: baselining and benchmarking," in Proceedings of the 1st International Workshop on Security Awareness from Design to Deployment, pp. 3–10, Gothenburg, Sweden.
- A. R. S. Farhan and G. M. M. Mostafa, (2018, April) "A methodology for enhancing software security during development processes," in Proceedings of the 2018 21st Saudi Computer Society National Computer Conference (NCC), pp. 1–6, Riyadh, Saudi Arabia.
- Alrubae, A.U., Cetinkaya, D., Liebchen, G., & Dogan H., (2020) A Process Model for Component-Based Model-Driven Software Development Journal Information, 11, 302; doi:10.3390/info11060302
- Rana T. & Baz A., (2020) Incremental Construction for Scalable Component-Based Systems, Journal Sensors, 20, 1435; doi:10.3390 NIST 2021
<https://csrc.nist.gov/Projects/ssdf/s20051435>.
- Vicente-Chicote, C., Inglés-Romero, J.F., Martínez, J., Stampfer, D., Lotz, A., Lutz, M., & Schlegel, C., (2018) A Component-Based and Model-Driven Approach to Deal with Non-Functional Properties through Global QoS Metrics: 40-45.
- Mills, M., (2017) "Sharing Privately: The Effect Publication on Social Media Has on Expectation of Privacy". EBSCOhost. Journal of Media Law. Retrieved 4 April 2018.
- Nabi, F., Yong, J., & Tao, X. (2020), A security review of event-based application function and service

component architecture. *IGI International Journal of Systems and Software Security and Protection*, 11 (2). pp. 58-70. ISSN 2640-4265.

Nabi, F., Yong, J., & Tao, X. (2020), Classification of logical vulnerability based on group attacking method. In: *11th International Conference on Ambient Systems, Networks and Technologies (ANT 2020)*, 6-9 April 2020, Warsaw Poland.

Nabi, F., Yong, J., & Tao, X. (2019), A novel approach for component-based-application logic event attack modelling. *International Journal of Network Security*, VDOI: 1816-3548 (2020-00059). ISSN 1816-353X.

Nabi, F., Yong, J., & Tao, X. (2020), Security aspects in modern service component-oriented application Logic for social e-commerce systems *Social Network Analysis and Mining* <https://doi.org/10.1007/s13278-020-00717-9>.

Nabi, F., Yong, J., & Tao, X. (2021) An Approach of social interaction with software connectors & the role of façade components for secure application logic (Under Publication Process).