

RISK AND AVOIDANCE STRATEGY FOR BLOCKING MECHANISM OF SOFTWARE-DEFINE NETWORKING BASED SECURITY SERVICE

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

By

KATAM PRUTHVI RAJ GOUD (REG.NO-37110332)

&

DASARI SANJEEV RAO (REG.NO-37110173)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC JEPPIAAR
NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119

MARCH - 2021

11
12
13



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **KATAM PRUTHVI RAJ GOUD (REG.NO - 37110332)** and **DASARI SANJEEV RAO (REG.NO - 37110173)** who carried out the project entitled "**RISK AND AVOIDANCE STRATEGY FOR BLO-CKING MECHANISM OF SOFTWARE-DEFINE NETWORKING BASED SECURITY SERVICE**" under my supervision from March 2021 to April 2021.

Internal Guide

Dr.Judji,M.E.,Ph.d.,

Head of the Department

Dr. S. VIGNESHWARI M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

internalExaminer

ii

External Examiner

i ii iii

DECLARATION

I **KATAM PRUTHVI RAJ GOUD(**REG.NO - 37110332**) and DASARI SANJEEV RAO (**REG.NO - 37110173**)** hereby declare that the Project Report entitled "**RISK AND AVOIDANCE STRATEGY FOR BLOCKING MECHANISM OF SOFTWARE-DEFINE NETWORKING BASED SECURITY SERVICE**" is done by me under the guidance of **Dr.Judgi,M.E.,Ph.d.,** Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE



ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean, School of Computing, Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. Judgi, M.E., Ph.d.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Software-Defined Network (SDN) is the dynamic network technology to address the issues of traditional networks. It provides centralized view of the whole network through decoupling the control planes and data planes of a network. Most SDN-based security services globally detect and block a malicious host based on IP address. However, the IP address is not verified during the forwarding process in most cases and SDN-based security service may block a normal host with forged IP address in the whole network, which means false-positive. In this paper, we introduce an attack scenario that uses forged packets to make the security service consider a victim host as an attacker so that block the victim. We also introduce cost-effective risk avoidance strategy.

**TABLE OF
CONTENTS**

CH APT ER NO	5	SYSTEM ANALYSIS
TITL E	5	5.1 FEASIB ILITY STUDY
		5.1.1 ECON OMIC FEASIB ILITY
	ABSTRACT	
	LIST OF FIGURES	5.1.2 OPERA TIONA L FEASIB ILITY
1	INTRODUCTION	5.1.3 TECHN ICAL FEASIB ILITY
2	LITERATURE SURVEY	
3	AIM AND SCOPE	
	3.1 AIM OF PROJEC T	
3	3.2 SCOPE AND OBJECTI VE	
	3.3 PURPO SE	
4	EXPERIMENTAL METHODOLOGY & ALGORITHMS	

PAGE V VIII

	7	
	8	
1		1
2	11	2
	11	
	11	
7	12	
7		

6	SYSTEM REQUIREMENTS SPECIFICATION	
	6.1 INTRODUCTION	13
	6.2 PURPOSE	14
	6.3 PUNCTIONAL REQUIREMENTS	14
	6.4 NON-FUNCTIONAL REQUIREMENTS	15
	6.5 SDLC METHODOLOGIES	16
7	RESULTS AND DISCUSSION	
	7.1 SCREENSHOTS	20
8	CONCLUSION AND FUTURE WORK	
	8.1 CONCLUSION	36
	8.2 FUTURE WORK	36
	REFERENCES	37
	APPENDIX	39
	SOURCE CODE	39

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
FIGURE 4.1	PROJECT ARCHITECTURE	8
FIGURE 6.5.1	SPIRAL MODEL	19
FIGURE 7.1.1	HOME PAGE	20
FIGURE 7.1.2	LOGIN PAGE	21
FIGURE 7.1.3	REGISTRATION PAGE	22
FIGURE 7.1.4	ABOUT PROJECT PAGE	23
FIGURE 7.1.5	CONTACT US PAGE	24
FIGURE 7.1.6	CONTROLLER HOME PAGE	25
FIGURE 7.1.7	NETWORK HOST PAGE	26
FIGURE 7.1.8	SWITCH CONFIGURATION PAGE	27
FIGURE 7.1.9	DATA REQUEST PAGE	28
FIGURE 7.1.10	BLOCKED HOST PAGE	29
FIGURE 7.1.11	HOST HOME PAGE	30
FIGURE 7.1.12	HOST INFORMATION PAGE	31
FIGURE 7.1.13	SHARE DATA PAGE	32
FIGURE 7.1.14	SENT DATA PAGE	33
FIGURE 7.1.15	RECEIVED DATA PAGE	34
FIGURE 7.1.16	CHANGE PASSWORD	35

1. INTRODUCTION

1 INTRODUCTION

Software-Defined Network (SDN) is the dynamic network technology to address the issues of traditional networks. It provides centralized view of the whole network through decoupling the control planes and data planes of a network. The control planes are responsible for policy creation and its implementation and the data planes are responsible for packet forwarding. This separation makes the network more programmable. Recently, SDN-based security services like FRESCO, Avant-guard, Floodguard, Of-guard and FlowRanger were proposed. A centralized SDN-based security service can manage network resource and dynamically add or delete security rules. Most SDN-based security services detect and block a malicious host based on IP address. It doesn't consider a flaw of Internet protocol that source IP address is not checked during the forwarding process and the centralized management system blocks the IP address in the whole network. As a result, malicious packets with forged IP address can cause a normal host to be blocked. In this, we introduce an attack scenario that uses forged packets to make the security service consider a victim host as an attacker so that block the victim. An attacker sends malicious packets, whose source IP address is a victim's IP address, to SDN. The security service then detects these packets and considers the victim malicious. Consequently, the security service generates flow rules that blocks the victim and the victim is blocked globally in the network. To prevent this attack, we also introduce cost-effective risk avoidance strategy. With our strategy, SDN-based security service blocks malicious IP address within the switch port from which the packet comes. This can prevent false positives from causing problems with availability of normal hosts.

1

1
1

2. LITERATURE SURVEY

SDN Based Firewall

In today's evolving technology, the need to adapt is the only necessity. The same is applicable when we are talking about the field of Network and Security i.e. the SDN technology. SDN is the newest evolving topic in the field of Networking. There are many problems faced by the traditional networks. These are solved by SDN. It is a way of making networks more manageable and more dynamic. It is very helpful in realizing the whole network beforehand thus giving a clear idea of the topologies required and the issues which can be solved even before setting the network up. But as in a new technology comes up it brings in new threats and issues. Thus our aim is to build a firewall over this new technology to protect the integrity of it. In this paper we focus on designing and developing a Firewall for our SDN controller which in turn performs the function of safe guarding the network from inside as well as the outside internet. To perform experiment, we have installed Mininet emulator for creating network topologies. In this paper, we present the implementation details of the firewall application. The current internet network is well set and huge in terms of topologies and its connections all over the world. But it has its drawbacks too. The data and the control plane being together yet makes it difficult in finding optimality in terms of routing, maintaining security and much more. Also adding or removing even a single router makes us change the whole network and is tedious. In a view of finding solution the SDN technology was introduced in 2005 where in the control plane gets totally removed and what we get is a controller i.e. the brain of the network and a bunch of routers that are meant for handling data and routing them. The new brain so formed is called as the SDN controller in the world of SDN. A controller responsible for taking the routing, protocol decisions whenever a packet arrives at its door. Our project aims at building a firewall over this controller that will in turn help the controller to take the decisions regarding the packets that arrive at the controller. The controller accompanied with the firewall will have the ability to accept, drop, or reject the incoming packets thus ensuring the safety of the system from malicious packets. The Firewall thus created will stand in between the controller and outside internet. Also it has the ability to handle the flow of packets shown by the controller thus providing an all rounder functionality where the controller is the mega brain and has the record

of every bit of data irrespective whether it flows in or out of network. Also the firewall which is coded in Python, provides a simplified GUI for the users to set up the firewall as per his/her convenience and also it will be able to save the system from any malicious attack by an intended attacker. Thus the project aims at emulate a network topology. The implementation includes the respective emulation of the network along with the firewall. The remainder of this paper is organized as follows, Section 2 presents the comparison of this project with the existing system, followed by Section 3 which states the proposed system and the implementation of each block of the system, the Conclusion is shown in section 4 and in the last section the references are presented.

Software-Defined Networking: A Comprehensive Survey

The Internet has led to the creation of a digital Society, where (almost) everything is connected and is accessible From anywhere. However, despite their widespread adoption, Traditional IP networks are complex and very hard to manage. It is both difficult to configure the network according to predefined Policies, and to reconfigure it to respond to faults, load And changes. To make matters even more difficult, current Networks are also vertically integrated: the control and data Planes are bundled together. Software-Defined Networking (SDN) Is an emerging paradigm that promises to change this state of Affairs, by breaking vertical integration, separating the network's Control logic from the underlying routers and switches, promoting (logical) centralization of network control, and introducing the Ability to program the network. The separation of concerns Introduced between the definition of network policies, their Implementation in switching hardware, and the forwarding of Traffic, is key to the desired flexibility: by breaking the network Control problem into tractable pieces, SDN makes it easier to Create and introduce new abstractions in networking, simplifying Network management and facilitating network evolution. In this paper we present a comprehensive survey on SDN. We Start by introducing the motivation for SDN, explain its main Concepts and how it differs from traditional networking, its roots, And the standardization activities regarding this novel paradigm. Next, we present the key building blocks of an SDN infrastructure Using a bottom-up, layered approach. We provide an in-depth Analysis of the hardware infrastructure, southbound and northbound Apis, network virtualization layers, network operating Systems (SDN controllers), network programming languages, and Network applications. We also look at cross- layer problems such As debugging and troubleshooting. In an effort to anticipate the Future

evolution of this new paradigm, we discuss the main Ongoing research efforts and challenges of SDN. In particular, We address the design of switches and control platforms – with A focus on aspects such as resiliency, scalability, performance, Security and dependability – as well as new opportunities for Carrier transport networks and cloud providers. Last but not Least, we analyze the position of SDN as a key enabler of a Software-defined environment.

FRESCO: Modular Composable Security Services for Software-Defined Networks

OpenFlow is an open standard that has gained tremendous interest in the last few years within the network community. It is an embodiment of the software-defined networking paradigm, in which higher-level flow routing decisions are derived from a control layer that, unlike classic network switch implementations, is separated from the data handling layer. The central attraction to this paradigm is that by decoupling the control logic from the closed and proprietary implementations of traditional network switch infrastructure, researchers can more easily design and distribute innovative flow handling and network control algorithms. Indeed, we also believe that OpenFlow can, in time, prove to be one of the more impactful technologies to drive a variety of innovations in network security. Open- Flow could offer a dramatic simplification to the way we design and integrate complex network security applications into large networks. However, to date there remains a stark paucity of compelling OpenFlow security applications. In this paper, we introduce FRESCO, an OpenFlow security application development framework designed to facilitate the rapid design, and modular composition of OF-enabled detection and mitigation modules. FRESCO, which is itself an OpenFlow application, offers a Click-inspired [19] programming framework that enables security researchers to implement, share, and compose together, many different security detection and mitigation modules. We demonstrate the utility of FRESCO through the implementation of several well-known security defenses as OpenFlow security services, and use them to examine various performance and efficiency aspects of our proposed framework. OpenFlow (OF) networks distinguish themselves from legacy network infrastructures by dramatically rethinking the relationship between the data and control planes of the network device. OpenFlow embraces the paradigm of highly programmable switch infrastructures, enabling software to compute an optimal flow routing decision on demand. For modern networks, which must increasingly deal with host virtualization and dynamic application migration, OpenFlow may offer the agility needed to handle dynamic network orchestration beyond that which traditional networks can achieve. For an OpenFlow switch, the data plane is

made programmable, where flows are dynamically specified within a flow table. The flow table contains a set of flow rules, which specify how the data plane should process all active network flows. In short, OpenFlow's flow rules provide the basic instructions that govern how to forward, modify, or drop each packet that traverses the OF-enabled switch. The switch's control plane is simplified to support the OpenFlow protocol, which allows the switch to communicate statistics and new flow requests to an external OpenFlow network controller. In return, it receives flow rules that extend its flow table ruleset. An OF controller is situated above a set of OF-enabled switches, often on lower-cost commodity hardware. It is the coordination point for the network's flow rule production logic, providing necessary flow rule updates to the switch, either in response to new flow requests or to reprogram the switch when conditions change. As a controller may communicate with multiple OF switches simultaneously, it can distribute a set of coordinated flow rules across the switches to direct routing or optimize tunneling in a way that may dramatically improve the efficiency of traffic flows. The controller also provides an API to enable one to develop OpenFlow applications, which implement the logic needed to formulate new flow rules. It is this application layer that is our central focus.

FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks

This paper addresses one serious SDN-specific attack, i.e., data-to-control plane saturation attack, which overloads the infrastructure of SDN networks. In this attack, an attacker can produce a large amount of table-miss packet_in messages to consume resources in both control plane and data plane. To mitigate this security threat, we introduce an efficient, lightweight and protocol-independent defense framework for SDN networks. Our solution, called FLOODGUARD, contains two new techniques/modules: proactive flow rule analyzer and packet migration. To preserve network policy enforcement, proactive flow rule analyzer dynamically derives proactive flow rules by reasoning the runtime logic of the SDN/OpenFlow controller and its applications. To protect the controller from being overloaded, packet migration temporarily caches the flooding packets and submits them to the OpenFlow controller using rate limit and round-robin scheduling. We evaluate FLOODGUARD through a prototype implementation tested in both software and hardware environments. The results show that FLOODGUARD is effective with adding only minor overhead into the entire SDN/OpenFlow infrastructure. Software-Defined Networking (SDN) [21] has quickly emerged as a new promising technology for future networks, and its reference implementation, OpenFlow, is becoming widely used in

recent years 1. SDN presents a physically distributed but logically centralized controlled networking framework. By decoupling the control plane from the data plane, SDN is designed to support fine-grained network management policies. Current OpenFlow implementations use a “southbound” protocol. When a switch receives a new flow for which there is no matching flow rules installed in the flow table (we call it a “table-miss” in this paper), the data plane will ask the control plane for actions.

A Request Prioritizing Algorithm for Controller DoS Attacks in Software Defined Networks

Software Defined Networking (SDN) introduces a new communication network management paradigm and has gained much attention from academia and industry. However, the centralized nature of SDN is a potential vulnerability to the system since attackers may launch denial of services (DoS) attacks against the controller. Existing solutions limit requests rate to the controller by dropping overflowed requests, but they also drop legitimate requests to the controller. To address this problem, we propose FlowRanger, a buffer prioritizing solution for controllers to handle routing requests based on their likelihood to be attacking requests, which derives the trust values of the requesting sources. Based on their trust values, FlowRanger classifies routing requests into multiple buffer queues with different priorities. Thus, attacking requests are served with a lower priority than regular requests. Our simulation results demonstrate that FlowRanger can significantly enhance the request serving rate of regular users under DoS attacks against the controller. To the best of our knowledge, our work is the first solution to battle against controller DoS attacks on the controller side. Software Defined Networks (SDN) introduces a new communication network management paradigm and has gained plethoras attention from academia and industry due to its fast growing potential in the market. One primary goal of SDN is to allow a network controller, called the control plane, to overlook and manage the entire network by configuring routing mechanisms for underlying switches. The switches, also called the data plane, are solely responsible for data forwarding according to their forwarding tables. Routing computation and network management are handled by the controller. The entries of forwarding tables on switches are determined and managed by the controller through a secured communication protocol between switches and the controller.

3. AIM AND SCOPE

3.1 AIM OF PROJECT

To Design a Responsive web Page for "Risk and avoidance strategy for blocking mechanism of software-define networking based security services".

3.2 SCOPE AND OBJECTIVE

Software-Defined Network(SDN) is the dynamic network technology to address the issues of traditional networks. It provides centralized view of the whole network through decoupling the control planes and data planes of a network. Most SDN-based security services globally detect and block a malicious host based on IP address. However, the IP address is not verified during forwarding process in most cases and SDN-based security services may block a normal host with forged IP address in the whole network, which means false-positive. In this paper, we are introduced an attack scenario that uses forged packets to make the security service consider a victim host as an attacker so that block the victim. We also introduce cost-effective risk avoidance strategy. Software-Define Network(SDN) is the dynamic network technology to address the issue of traditional networks. It provides centralized view of the whole network through de-coupling the control planes and data planes of a network. The control planes are responsible for policy creation and its implementation and the data planes are responsible for packet forwarding. This separation makes the network more programmable.

3.3 PURPOSE

We propose risk and avoidance strategy to prevent IP forging attack. The attack described above occurs when the SDN-based security service blocks unverified IP address in the whole network. Source validation is a good way to prevent this attack but it needs complex deployment steps and high cost. So, we focus on the specification of the blocking range. We use Open-Flow protocol described in II.A to specify the blocking range. Using Open-Flow protocol, we can figure out OFPP IN PORT, from which the malicious packets come, and set the specific blocking range. Applying our risk avoidance strategy. SDN-based security service blocks IP address when both IP address and port information meet the detection criteria. As off-switch drops malicious packet based on IP address and OFPP IN PORT, we can prevent IP forging attack from blocking a normal host.

4 EXPERIMENTAL METHODOLOGY AND ALGORITHMS

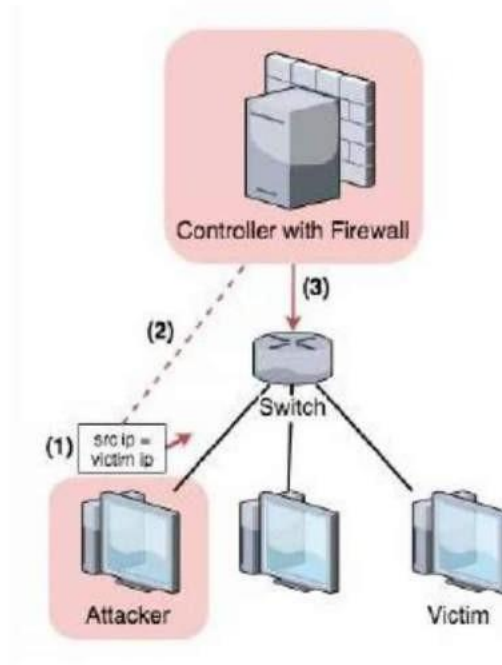


Figure 4.1 Project Architecture

IP forging attack, which we introduce in this , is using the flaw of SDN-based security service. An attacker sends malicious packets, whose source IP address is a victim's IP address, to SDN. The security service then detects these packets and considers the victim malicious. Consequently, the security service generates flow rules that blocks the victim and the victim is blocked globally in the network. In this section, we first analyze the risk of SDN-based security service. And then we define vulnerable component, attacker and victim with NVD's CVSS v3.0. Finally, we introduce IP forging attack scenario and evaluate the scenario with CVSS v3.0.

A. Risk of blocking mechanism of SDN-based security service ISO/IEC 15408-2:2008 defines the content and presentation of the security functional requirements to be assessed in a security evaluation using ISO/IEC 15408. Based on ISO/IEC 15408-2:2008, risk occurs when asset, vulnerability and threat are combined. Fig. shows the relationship of asset, vulnerability, threat and risk. • Asset: Assets are entities that someone places value upon. In our SDN environment, asset can be hosts. Impairment of the assets commonly includes loss of asset confidentiality,

integrity and availability. • **Vulnerability:** Vulnerability is weakness that can be exploited by a threat agent or an attacker. Most SDN-based security services have two flaws that can cause vulnerabilities. 1) SDN-based security service uses Internet protocol that does not verify source IP address during the forwarding process. 2) SDN-based security service blocks IP address in the whole network. • **Threat:** A threat consists of an adverse action performed by a threat agent on an asset. An attacker can exploit vulnerabilities above with IP forging attack. The attacker uses forged packets to make the security service consider a victim host as the attacker so that block the victim. • **Risk:** Due to the vulnerability that SDN-based security service centrally blocks IP address in the whole network, an attacker can do IP forging attack to a normal host. This lets the loss of asset availability.

B. Definition of vulnerable component A vulnerable component defined in CVSS v3.0 represents characteristics of thing that is vulnerable . A vulnerable component is the entire network in which the SDN-based security service runs. As described in II.B, SDN-based security service centrally monitors and controls network resources. Malicious packets are automatically detected by the centralized security service in the whole network, so an attacker doesn't have to send packets directly to it. Instead, the attacker can send packets in the network. The security service generates flow rules after detection and distributes them to switches. Consequently, the rules generated based on the detected packets will affect the whole network. In short, the attack will also affect the whole network. This characteristic is the key point of this attack and makes the attack surface wider. A vulnerable component is the SDN-based security service that detects and blocks malicious actions. A principle of our attack is to make security service consider a victim as an attacker so that block the victim. Security services utilize the global view for detecting and blocking malicious actions. These kinds of security services can be our target.

C. Definition of attacker and victim An attacker knows victims IP address. An attacker only needs to know the IP address of a victim because IP forging attack is network-based attack. An attacker can send spoofed packets to SDN network. IP forging attack sends spoofed packets in the network. So first, an attacker makes spoofed packets. Second, an attacker should be able to send packets to SDN. An attacker knows the actions to be detected by SDN-based security services. IP forging attack utilizes blocking mechanism in SDN-based security services and an attacker should know the mechanism or signature for detecting and blocking malicious actions. A victim is in SDN-based security service area. IP forging attack utilizes the centralized SDN-based

security service188International Conference on Advanced Communications Technology(ICACTION) ISBN 979-11-88428-02-1 ICACTION2019 February 17 ~ 20, 2019Fig. 3. IP Forging Attack scenariomechanism. Therefore, to block a victim, a victim has to be within the area of the SDN-based security service.

D. IP Forging Attack Scenario Fig. 3 shows the scenario of IP Forging attack as follows. 1) An attacker generates packets whose source IP address is a victim's IP address. The packets can contain various malicious payloads that should be detected by security services. Then the attacker sends packets to any host in the SDN. 2) When the attacker sends malicious packets to any host through SDN, the security service detects the packets using policies. The security service considers the victim as an attacker and generates a flow rule which blocks the victim. 3) The controller distributes the generated flow rule to switches. Each switch then updates the flow rule into its flow table. 4) As a result, the victim is blocked at every switch, which means it is blocked in the whole network.

E. Impact CVSS(the Common Vulnerability Scoring System) provides a way to capture the principal characteristics of a vulnerability and produce a numerical score reflecting its severity. According to CVSS v3.0, the vulnerable component is the SDN-based security service and the impact component is the victim host. Evaluation details are as follows. Attack Vector: IP forging attack is remotely exploitable, so Attack Vector is Network. Attack Complexity: Attack Complexity is High because IP forging attack needs configuration settings to enable detection and blocking threats in the SDN-based security service. But unlike MiTM attack, IP forging attack can be done by simply one step.

Privileges Required: Privileges Required is None. The attacker only needs to send spoofed packets to SDN, so it is possible without privilege of the vulnerable component. User Interaction: User Interaction is also None because IP forging attack doesn't require any user interaction. Scope: When the victim is an important server or critical device, IP forging attack can affect many systems. So Scope is Changed. Confidentiality Impact: Confidentiality Impact is None. Integrity Impact: Integrity Impact is None. Availability Impact: A IP forging attacked host is completely unavailable, so Availability Impact is High.

$$10^7$$

5 SYSTEM ANALYSIS

5.1 FEASIBILITY STUDY

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

5.1.1 ECONOMIC FEASIBILITY

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

5.1.2 OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?



This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

5.1.3 TECHNICAL FEASIBILITY

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipments have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source.

6 SYSTEM REQUIREMENTS SPECIFICATION

6.1 INTRODUCTION

A **Software Requirements Specification (SRS)** – a requirements specification for a software system – is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development life cycle domain, typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

- **Business requirements** describe in business terms *what* must be delivered or accomplished to provide value.
- **Product requirements** describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
- **Process requirements** describe activities performed by the developing organization. For instance, process requirements could specify specific methodologies that must be followed, and constraints that the organization must obey.

Product and process requirements are closely linked. Process requirements often specify the activities that will be performed to satisfy a product requirement. For example, a maximum development cost requirement (a process requirement) may be imposed to help achieve a maximum sales price requirement (a product requirement); a requirement that the product be maintainable (a Product requirement) often is addressed by imposing requirements to follow particular development styles

6.2 PURPOSE

An systems engineering, a **requirement** can be a description of *what* a system must do, referred to as a Functional Requirement. This type of requirement specifies something that the delivered system must be able to do. Another type of requirement specifies something about the system itself, and how well it performs its functions. Such requirements are often called Non-functional requirements, or 'performance requirements' or 'quality of service requirements.' Examples of such requirements include usability, availability, reliability, supportability, testability and maintainability.

A collection of requirements define the characteristics or features of the desired system. A 'good' list of requirements as far as possible avoids saying *how* the system should implement the requirements, leaving such decisions to the system designer. Specifying how the system should be implemented is called "implementation bias" or "solution engineering". However, *implementation constraints* on the solution may validly be expressed by the future owner, for example for required interfaces to external systems; for interoperability with other systems; and for commonality (e.g. of user interfaces) with other owned products.

In software engineering, the same meanings of requirements apply, except that the focus of interest is the software itself.

6.3 FUNCTIONAL REQUIREMENTS

6.3.1 MODULES

Home

Network Host

Accept request

Switch Configuration

Data Requests

Blocked Hosts

¹⁴11

11

Logout

Host:

Home

Status

Share Data

View sent Data

Received Data

Change password

Logout

Software Requirements:

- Operating system : Windows7
- Coding Language : J2EE
- Server : Tomcat
- Tool : My Eclipse 8.6
- Data Base : Oracle 10 g

Hardware Requirements:

- System : Intel i3 Processor
- Hard Disk : 500 GB
- Ram : 8GB

6.4 NON FUNCTIONAL REQUIREMENTS

6.4 NON FUNCTIONAL REQUIREMENTS

follows

Usability

The system is designed with completely automated process hence there is no or less user intervention.

Reliability

The system is more reliable because of the qualities that are inherited from the chosen platform java. The code built by using java is more reliable.

Performance

This system is developing in the high level languages and using the advanced front-end and back-end technologies it will give response to the end user on client system with in very less time.

Supportability

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is having JVM, built into the system.

Implementation

The system is implemented in web environment using struts framework. The apache tomcat is used as the web server and windows xp professional is used as the platform.

Interface the user interface is based on Struts provides HTML Tag

6.5 SDLC METHODOLOGIES

Spiral Model

The spiral model is similar to the incremental model, with more emphases placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements is gathered and risk is assessed. Each subsequent spirals builds on the baseline spiral. Requirements are gathered during the planning phase. In the risk analysis phase, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. Software is produced in the engineering phase, along with testing at the end of the phase. The evaluation phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral. In the spiral model, the angular component represents progress, and the radius of the spiral represents cost. Spiral Life Cycle Model.

13₁₆

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, "A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

- The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system.
- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved by a fourfold procedure:
 1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
 2. Defining the requirements of the second prototype.
 3. Planning an designing the second prototype.
 4. Constructing and testing the second prototype.

- At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.
- The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
- The final system is constructed, based on the refined prototype.
- The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

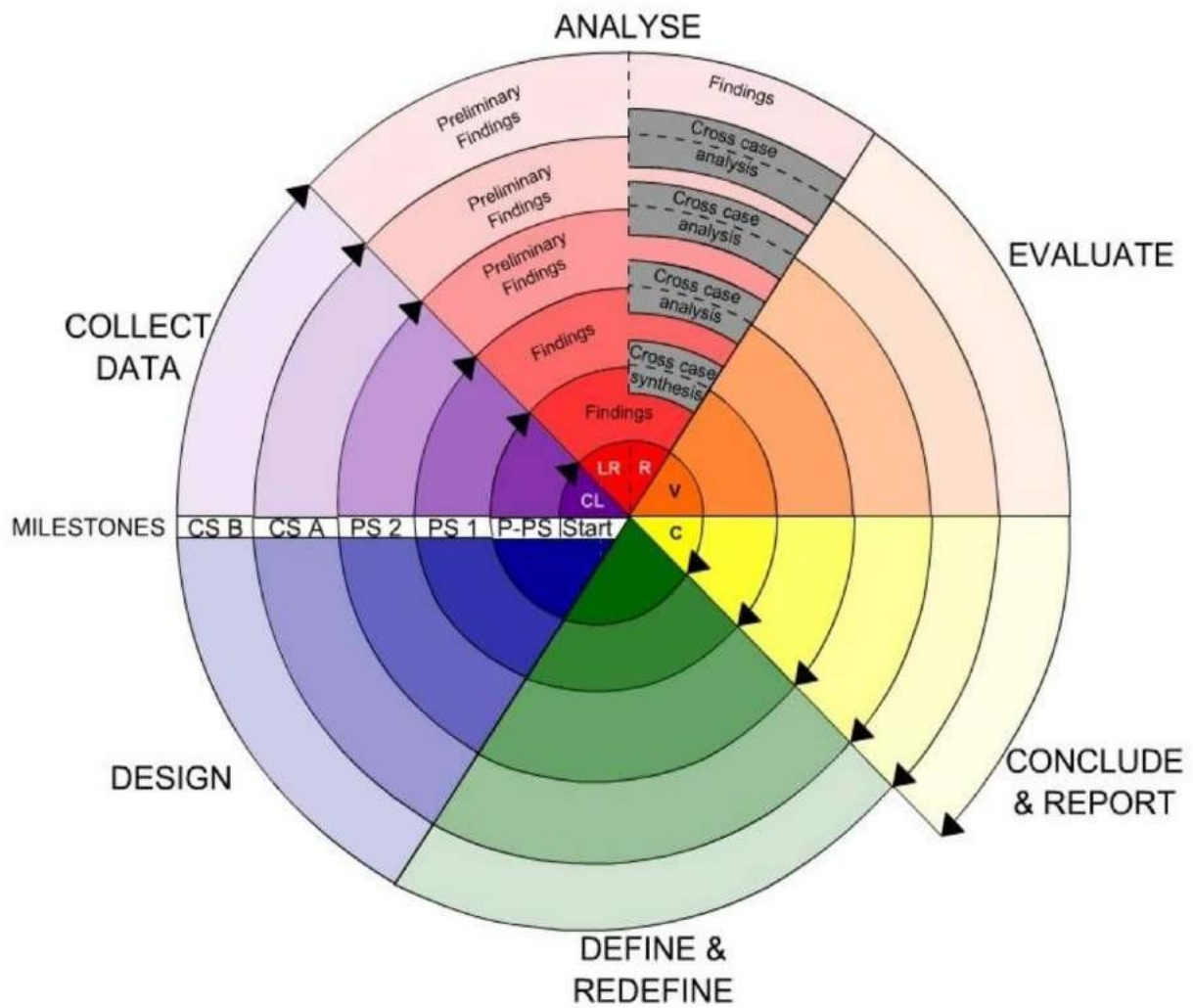


Figure 6.5.1 Spiral Model

Advantages

- High amount of risk analysis
- Good for large and mission-critical projects.
- Software is produced early in the software life cycle.

16
19

7.1 SCREENSHOTS



Figure 7.1.1 HOME PAGE



Fig:Login Page

Figure 7.1.2 LOGIN PAGE

SignUp Page Object Browser +

← signbp: Not secure | rambabu8086/Risk%26Avoidance_Strategy_for_BlockingMechanism_of_SDN_Security_Services/signup.jsp

Risk And Avoidance Strategy For Blocking Mechanism of SDN based security services

[Home](#) [Login](#) [Register](#) [About_us](#) [Contact_us](#)

Register Here

Host Id	enter host_id(example@gmail.com)
Host Name	enter host name
Password	enter password
Email	enter email
Host IP	enter ip(ex -192.168.0.0)
Register	



Figure 7.1.3 REGISTRATION PAGE



Figure 7.1.4 About Project Page

2023



Figure Contact Us page

2241



Figure 7.1.6 Controller Home Page



Figure 7.1.7 Network Host Page



Figure 7.1.8 Switch Configuration Page

Data Requests Object Browser +

← → ↻ ⓘ Not secure | rambabu8086/Risk%26Avoidance_Strategy_for_BlockingMechanism_of_SDN_Security_Services/getData ☆ ⓘ

Risk And Avoidance Strategy For Blocking Mechanism of SDN based security services

[Home](#) [Network Host](#) [Switch Cfg](#) [Data Requests](#) [Blocked Hosts](#)

Welcome ! controller@gmail.com | [Logout](#)

Data Forwarding Requests

ID	Source Host	Source IP	Switch Port	Receiver Host	Data	Status
1	rahul@gmail.com	192.168.1.2	1656	ram@gmail.com	Hello	Data Sent
2	rahul@gmail.com	192.168.1.2	1656	raja@gmail.com	Hii	Data Sent
3	rahul@gmail.com	192.168.1.2	1656	ram@gmail.com	Hello	Data Sent
4	rahul@gmail.com	192.168.45.12	1656	ram@gmail.com	hello ram	Data Sender Blocked
5	ram@gmail.com	192.168.45.12	6381	raja@gmail.com	hello	Data Sent

Figure 7.1.9 Data Request Page



Figure 7.1.10 Blocked Host Page

29
26



Figure 7.1.11 Host Home Page

30

27



F



Figure 7.1.13 Share Data Page

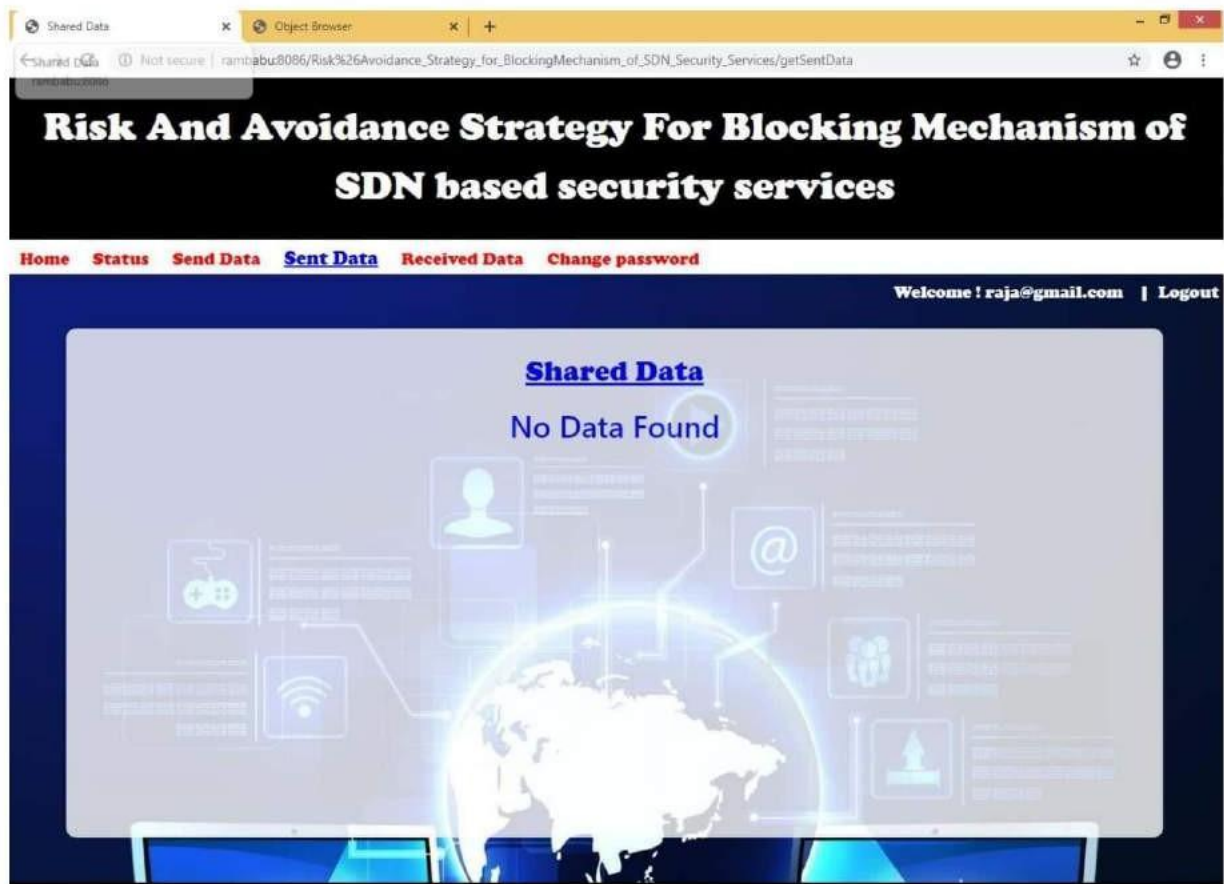


Figure 7.1.14 Sent Data Page

30³³

Received Data Object Browser

Not secure | rambabu:8086/Risk%26Avoidance_Strategy_for_BlockingMechanism_of_SDN_Security_Services/getReceivedData

Risk And Avoidance Strategy For Blocking Mechanism of SDN based security services

[Home](#) [Status](#) [Send Data](#) [Sent Data](#) [Received Data](#) [Change password](#)

Welcome : raja@gmail.com | [Logout](#)

Received Data

ID	Source IP	Sender Host	Switch Port	Data
2	192.168.1.2	rahul@gmail.com	1656	Hi!
5	192.168.45.12	ram@gmail.com	6381	hello



Figure Received Data Page

Change Password Object Browser + Not secure | rambabu8086/Risk%26Avoidance_Strategy_for_BlockingMechanism_of_SDN_Security_Services/host_changePass.jsp

Risk And Avoidance Strategy For Blocking Mechanism of SDN based security services

[Home](#) [Status](#) [Send Data](#) [Sent Data](#) [Received Data](#) [Change password](#)

Welcome : raja@gmail.com | [Logout](#)

Change Password

Old Password

enter old password

New Password

enter new password

Re-enter Password

re-enter new password

Change Password

Figure Change Password

8.1 CONCLUSION

We propose the IP forging risk of SDN-based security service. SDN-based security service blocks unverified IP address in the whole network and this can cause the loss of assets availability. Avoidance strategy for this attack uses Open-Flow Protocol to specify the blocking range. With this strategy, SDN-based security service blocks malicious IP address with in the switch port from which the packet comes and we can prevent the loss of asset availability with cost-effective and simple way.

8.2 FUTURE WORK

As future work, we will verify our risk avoidance strategy through experiments with SDN-based security services.

REFERENCES

- [1] P. Patel, Implementing software-defined network (SDN) based firewall, 2016.
- [2] M. Rouse, Programmable Network (PN), 2013.
- [3] S. W. Shin, P. Porras, V. Yegneswara, M. Fong, G. Gu, and M. Tyson, Fresco: Modular composable security services for software- defined networks, in 20th Annual Network Distributed System Security Symposium. NDSS, 2013.
- [4] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, Avant-guard: Scalable and vigilant switch flow management in software-defined networks, in Proceedings of the 2013 ACM SIGSAC conference on Computer communications security. ACM, 2013, pp. 413424.
- [5] H. Wang, L. Xu, and G. Gu, Floodguard: A dos attack prevention extension in software-defined networks, in 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. IEEE, 2015, pp. 239250.
- [6] W. Haopei, X. Lei, and G. Guofei, Of-guard: a dos attack prevention extension in software-defined networks, USENIX Open Network Summit, 2014.
- [7] L. Wei and C. Fung, Flowranger: A request prioritizing algorithm for controller dos attacks in software defined networks, in Communications (ICC), 2015 IEEE International Conference on. IEEE, 2015, pp. 5254 5259.
- [8] J. Jeong, J. Seo, G. Cho, H. Kim, and J.-S. Park, A framework for security services based on software-defined networking, in Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on. IEEE, 2015, pp. 150153.
- [9] R. I.-T. Y.3300, Framework of Software-Defined Networking, 2014.
- [10] Open Networking Foundation, SDN Architecture, ONF TR-521, 2014.
- [11] Open Networking Foundation, OpenFlow Switch Specification (Version 1.5.1), ONF TS-025, 2015.
- [12] M. [27] Boucadair and C. Jacquenet, Software-defined networking: A perspective from within a service provider environment. No. RFC 7149. 2014.
- [13] J. Wu, J. Bi, X. Li, G. Ren, K. Xu, and M. Williams, A source address validation architecture (sava) testbed and deployment experience, Tech. Rep., 2008.
- [14] D. Jacobs, Addressing SDN security challenges means securing the controller, 2017.
- [15] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, Security in software defined networks: A survey, IEEE Communications Surveys Tutorials, vol. 17, no. 4, pp. 23172346, 2015.

- [16] NVD, Common Vulnerability Scoring System v3.0: Specification Document
- [17] ISO/IEC 15408-2:2008
- [18] Douglas D., and Victor R. Basili. A Comparative Analysis of Functional Correctness. No. TR-921. MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE, 1980.
- [19] Kellerer, Wolfgang, Arsany Basta, and Andreas Blenk. "Flexibility of Networks: a new measure for network design space analysis?" arXiv preprint arXiv:1512.03770 (2015).
- [20] G. Yao and et al., Source address validation solution with Open-Flow/NOX architecture, in Proceedings of the 19th ICNP, 2011, pp. 712.
- [21] S. Scott-Hayward and et al., A Survey of Security in Software Defined Networks, IEEE Communications Surveys Tutorials, vol. 18, no. 1, pp. 623654, 2016.
- [22] Chen, Guolong, et al. "SAVSH: IP source address validation for SDN hybrid networks." Computers and Communication (ISCC), 2016 IEEE Symposium on. IEEE, 2016.

APPENDIX

SOURCE CODE :

```
package com.nt.dbutils;

import java.sql.Connection;

import java.util.ResourceBundle;

import oracle.jdbc.pool.OracleDataSource;

public class DatabaseUtils {

    static OracleDataSource ds =null;

    static ResourceBundle bundle;

    static{

        try{

            bundle =ResourceBundle.getBundle("com/nt/dbproperties/database");

            ds = new OracleDataSource();

            ds.setURL(bundle.getString("url"));

            ds.setUser(bundle.getString("username"));

            ds.setPassword(bundle.getString("password"));

        }catch (Exception e) {

            e.printStackTrace();

        }

    }

}
```



```

    }

    public Connection getConnection() throws Exception{

        Connection con =ds.getConnection();

        return con;

    }

```

```

}package com.nt.dao;

```

```

import java.sql.Connection;

```

```

import java.sql.PreparedStatement;

```

```

import java.sql.ResultSet;

```

```

import java.util.ArrayList;

```

```

import java.util.List;

```

```

import com.nt.dbutils.DatabaseUtils;

```

```

import com.nt.model.Host;

```

```

import com.nt.model.HostData;

```

```

import com.nt.model.OpenFlowSwitch;

```

```

public class ControllerDAO extends DatabaseUtils {

```



```

public List<Host> getHost()throws Exception{

    List<Host> list =new ArrayList<Host>();

    Connection con =getConnection();

    String query ="SELECT
HOST_ID,HOST_NAME,HOST_IP,SWITCH_PORT,HOST_EMAIL,NW_STATUS,PASSWO
RD,STATUS FROM NETWORK_HOST INNER JOIN LOGIN_DATA ON
NETWORK_HOST.HOST_ID=LOGIN_DATA.USERNAME";

    PreparedStatement ps =con.prepareStatement(query);

    ResultSet rs =ps.executeQuery();

    while(rs.next()){

        Host h =new Host();

        h.setHost_id(rs.getString(1));

        h.setHost_name(rs.getString(2));

        h.setHost_ip(rs.getString(3));

        h.setSwitch_port(rs.getInt(4));

        h.setHost_email(rs.getString(5));

        h.setNw_status(rs.getString(6));

        h.setPsw(rs.getString(7));

        h.setStatus(rs.getInt(8));

        list.add(h);

    }

    con.close();
}

```



```
        return list;
    }
}
```

```
public List<Host> getBHost()throws Exception{

    List<Host> list =new ArrayList<Host>();

    Connection con =getConnection();

    String query ="SELECT
HOST_ID,HOST_NAME,HOST_IP,SWITCH_PORT,NW_STATUS
FROM
NETWORK_HOST WHERE NW_STATUS='BLOCKED'";

    PreparedStatement ps =con.prepareStatement(query);

    ResultSet rs =ps.executeQuery();

    while(rs.next()){

        Host h =new Host();

        h.setHost_id(rs.getString(1));

        h.setHost_name(rs.getString(2));

        h.setHost_ip(rs.getString(3));

        h.setSwitch_port(rs.getInt(4));

        h.setNw_status(rs.getString(5));

        list.add(h);

    }

    con.close();

    return list;
}
```



```
}
```

```
public OpenFlowSwitch getSwitch()throws Exception{

    Connection con =getConnection();

    String query="SELECT
ID,SWITCH_NAME,PORT_NUMBERS,MAX_PORT,ASSIGNED_PORTS,TYPE FROM
SWITCH_CFG";

    PreparedStatement ps =con.prepareStatement(query);

    ResultSet rs =ps.executeQuery();

    OpenFlowSwitch s =new OpenFlowSwitch();

    if(rs.next()){

        s.setSwitch_name(rs.getString(2));

        s.setPort_numbers(rs.getString(3));

        s.setMax_port(rs.getInt(4));

        s.setAssigned_ports(rs.getInt(5));

        s.setType(rs.getString(6));

    }

    con.close();

    return s;

}
```

40

43

```
public int acceptRequest(String host,int port) throws Exception{
```

```
    Connection con =getConnection();
```

```
    con.setAutoCommit(false);
```

```
    int i1 =updateLogin(host, con);
```

```
    int i2=updateNWHost(host, port, con);
```

```
    int i3=updateSwitch(port, con);
```

```
    if(i1>0&& i2>0&& i3>0){
```

```
        con.commit();
```

```
        return 1;
```

```
    }else{
```

```
        con.rollback();
```

```
        return 0;
```

```
    }
```

```
}
```

```
public int updateSwitch(int port,Connection con)throws Exception{
```

```
    String query="SELECT PORT_NUMBERS,ASSIGNED_PORTS FROM  
SWITCH_CFG WHERE ID=1";
```

```
    String query1="UPDATE SWITCH_CFG SET  
PORT_NUMBERS=?,ASSIGNED_PORTS=? WHERE ID=1";
```

```
    PreparedStatement ps =con.prepareStatement(query);
```



```

ResultSet rs =ps.executeQuery();

int a_port=0;

String ports=null;

if(rs.next()){

    ports =rs.getString(1);

    a_port=rs.getInt(2);

    PreparedStatement ps1 =con.prepareStatement(query1);

    ps1.setString(1,ports+","+port);

    ps1.setInt(2, a_port+1);

    int r =ps1.executeUpdate();

    if(r>0)

        return 1;

    else

        return 0;

}

return 0;

}

}

public int updateNWHost(String host,int port,Connection con)throws Exception{

```



```
String query ="UPDATE NETWORK_HOST SET  
NW_STATUS='ACTIVE',SWITCH_PORT=? WHERE HOST_ID=?";
```

```
PreparedStatement ps =con.prepareStatement(query);
```

```
ps.setInt(1,port);
```

```
ps.setString(2, host);
```

```
return ps.executeUpdate();
```

```
}
```

```
public int updateLogin(String user,Connection con)throws Exception{
```

```
String query="UPDATE LOGIN_DATA SET STATUS=1 WHERE  
USERNAME=?";
```

```
PreparedStatement ps =con.prepareStatement(query);
```

```
ps.setString(1,user);
```

```
return ps.executeUpdate();
```

```
}
```

```
public String getPorts()throws Exception{
```

```
String query="SELECT PORT_NUMBERS FROM SWITCH_CFG WHERE  
ID=1";
```

```
Connection con =getConnection();
```

```
PreparedStatement ps =con.prepareStatement(query);
```

```
ResultSet rs =ps.executeQuery();
```

4⁴³⁶

```

String ports =null;

if(rs.next()){

    ports =rs.getString(1);

}

con.close();

return ports;

}

```

```

public String getHostIP(String host_id)throws Exception{

    String  query="SELECT  HOST_IP  FROM  NETWORK_HOST  WHERE
HOST_ID=?";

    Connection con =getConnection();

    PreparedStatement ps =con.prepareStatement(query);

    ps.setString(1,host_id);

    ResultSet rs =ps.executeQuery();

    String ip =null;

    if(rs.next()){

        ip =rs.getString(1);

    }

    con.close();

    return ip;
}

```



```
}
```

```
public List<HostData> getData()throws Exception{

    List<HostData> list =new ArrayList<HostData>();

    Connection con=getConnection();

    String query="SELECT
ID,SENDER_HOST,INFO,SOURCE_IP,PORT,RECEIVER_HOST,STATUS      FROM
HOST_DATA";

    PreparedStatement ps =con.prepareStatement(query);

    ResultSet rs =ps.executeQuery();

    while(rs.next())

    {

        HostData d =new HostData();

        d.setId(rs.getInt(1));

        d.setSender_host(rs.getString(2));

        d.setInfo(rs.getString(3));

        d.setSource_ip(rs.getString(4));

        d.setPort(rs.getInt(5));

        d.setReceiver_host(rs.getString(6));

        d.setStatus(rs.getInt(7));

        list.add(d);

    }

}
```



```
    }  
  
    con.close();  
  
    return list;  
}
```

```
public int takeAction(HostData d)throws Exception{  
  
    Connection con =getConnection();  
  
    con.setAutoCommit(false);  
  
    int i =updateStatus(d.getId(),2, con);  
  
    int i2=blockPort(d.getSender_host(), d.getPort(), con);  
  
    if(i>0&& i2>0){  
  
        con.commit();  
  
        return 1;  
  
    }else{  
  
        con.rollback();  
  
        return 0;  
  
    }  
}  
  
public int blockPort(String host_id,int port,Connection con)throws Exception{
```

49
46

```
String query="UPDATE NETWORK_HOST SET NW_STATUS='BLOCKED'  
WHERE HOST_ID=? AND SWITCH_PORT=?";
```

```
PreparedStatement ps =con.prepareStatement(query);  
  
ps.setString(1,host_id);  
  
ps.setInt(2,port);  
  
int result =ps.executeUpdate();  
  
return result;  
  
}
```

```
public int updateStatus(int id,int status)throws Exception{
```

```
String query="UPDATE HOST_DATA SET STATUS=? WHERE ID=?";  
  
Connection con =getConnection();  
  
PreparedStatement ps =con.prepareStatement(query);  
  
ps.setInt(1,status);  
  
ps.setInt(2,id);  
  
int result=ps.executeUpdate();  
  
con.close();  
  
return result;  
  
}
```

```
public int updateStatus(int id,int status,Connection con)throws Exception{
```



```

String query="UPDATE HOST_DATA SET STATUS=? WHERE ID=?";

PreparedStatement ps =con.prepareStatement(query);

ps.setInt(1,status);

ps.setInt(2,id);

int result=ps.executeUpdate();

return result;

}

```

```

public HostData getData(int id)throws Exception{

    Connection con=getConnection();

    String query="SELECT ID,SENDER_HOST,SOURCE_IP,PORT FROM
HOST_DATA WHERE ID=?";

    PreparedStatement ps =con.prepareStatement(query);

    ps.setInt(1,id);

    ResultSet rs =ps.executeQuery();

    HostData d =new HostData();

    while(rs.next())

    {

        d.setId(rs.getInt(1));

        d.setSender_host(rs.getString(2));
    }
}

```



```
        d.setSource_ip(rs.getString(3));

        d.setPort(rs.getInt(4));

    }

    con.close();

    return d;

}

}package com.nt.dao;


import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;


import com.nt.dbutils.DatabaseUtils;

import com.nt.model.Host;

import com.nt.model.HostData;


public class HostDAO extends DatabaseUtils{


    public Host getStatus(String host_id)throws Exception{
```



```

String query="SELECT
HOST_ID,HOST_NAME,HOST_IP,SWITCH_PORT,HOST_EMAIL,NW_STATUS,PASSWO
RD,STATUS FROM NETWORK_HOST INNER JOIN LOGIN_DATA ON
NETWORK_HOST.HOST_ID=LOGIN_DATA.USERNAME WHERE HOST_ID=?";

Connection con =getConnection();

PreparedStatement ps =con.prepareStatement(query);

ps.setString(1,host_id);

ResultSet rs =ps.executeQuery();

Host h=new Host();

if(rs.next()){

    h.setHost_id(rs.getString(1));

    h.setHost_name(rs.getString(2));

    h.setHost_ip(rs.getString(3));

    h.setSwitch_port(rs.getInt(4));

    h.setHost_email(rs.getString(5));

    h.setNw_status(rs.getString(6));

    h.setPsw(rs.getString(7));

    h.setStatus(rs.getInt(8));

}

con.close();

return h;

}

```



```

public int shareData(HostData d)throws Exception{

    int result=0;

    Connection con =getConnection();

    String query="INSERT INTO
HOST_DATA(SENDER_HOST,INFO,SOURCE_IP,PORT,RECEIVER_HOST,STATUS)
VALUES(?,?,?,?,?,0)";

    PreparedStatement ps =con.prepareStatement(query);

    ps.setString(1,d.getSender_host());

    ps.setString(2,d.getInfo());

    ps.setString(3,d.getSource_ip());

    ps.setInt(4,getPort(d.getSender_host()));

    ps.setString(5,d.getReceiver_host());

    result =ps.executeUpdate();

    con.close();

    return result;

}

```

```

public List<HostData> getSentData(String host_id)throws Exception{

    List<HostData> list =new ArrayList<HostData>();

    Connection con=getConnection();

```



```
String query="SELECT  
ID,SENDER_HOST,INFO,SOURCE_IP,PORT,RECEIVER_HOST,STATUS FROM  
HOST_DATA WHERE SENDER_HOST=?";
```

```
PreparedStatement ps =con.prepareStatement(query);
```

```
ps.setString(1,host_id);
```

```
ResultSet rs =ps.executeQuery();
```

```
while(rs.next())
```

```
{
```

```
    HostData d =new HostData();
```

```
    d.setId(rs.getInt(1));
```

```
    d.setSender_host(rs.getString(2));
```

```
    d.setInfo(rs.getString(3));
```

```
    d.setSource_ip(rs.getString(4));
```

```
    d.setPort(rs.getInt(5));
```

```
    d.setReceiver_host(rs.getString(6));
```

```
    d.setStatus(rs.getInt(7));
```

```
    list.add(d);
```

```
}
```

```
con.close();
```

```
return list;
```

```
}
```

```

public List<HostData> getReceivedData(String host_id)throws Exception{

    List<HostData> list =new ArrayList<HostData>();

    Connection con=getConnection();

    String query="SELECT
ID,SENDER_HOST,INFO,SOURCE_IP,PORT,RECEIVER_HOST,STATUS      FROM
HOST_DATA WHERE RECEIVER_HOST=? AND STATUS=1";

    PreparedStatement ps =con.prepareStatement(query);

    ps.setString(1,host_id);

    ResultSet rs =ps.executeQuery();

    while(rs.next())

    {

        HostData d =new HostData();

        d.setId(rs.getInt(1));

        d.setSender_host(rs.getString(2));

        d.setInfo(rs.getString(3));

        d.setSource_ip(rs.getString(4));

        d.setPort(rs.getInt(5));

        d.setReceiver_host(rs.getString(6));

        d.setStatus(rs.getInt(7));

        list.add(d);

    }

    con.close();
}

```

```
        return list;
    }
}
```

```
public int getPort(String host_id)throws Exception{

    Connection con =getConnection();

    int port=0;

    String query="SELECT SWITCH_PORT FROM NETWORK_HOST WHERE
HOST_ID=?";

    PreparedStatement ps =con.prepareStatement(query);

    ps.setString(1,host_id);

    ResultSet rs=ps.executeQuery();

    rs.next();

    port=rs.getInt(1);

    con.close();

    return port;

}
```

```
public List<String> getHosts(String host_id)throws Exception{

    List<String> list =new ArrayList<String>();

    String query = "SELECT HOST_ID FROM NETWORK_HOST WHERE
HOST_ID NOT IN (?) AND NW_STATUS='ACTIVE'";
```



```

        Connection con =getConnection();

        PreparedStatement ps =con.prepareStatement(query);

        ps.setString(1,host_id);

        ResultSet rs =ps.executeQuery();

        while(rs.next())

            list.add(rs.getString(1));

        con.close();

        return list;

    }

```

```

public String getNWStatus(String host_id)throws Exception{

    Connection con=getConnection();

    String query="SELECT NW_STATUS FROM NETWORK_HOST WHERE
HOST_ID=?";

    PreparedStatement ps =con.prepareStatement(query);

    ps.setString(1,host_id);

    ResultSet rs =ps.executeQuery();

    rs.next();

    String status=null;

    status=rs.getString(1);

    con.close();

```

58

55

```
        return status;
    }
}
```

```
public int changePassword(String host_id,String psw) throws Exception{

    int result=0;

    String query="UPDATE LOGIN_DATA SET PASSWORD =? WHERE
USERNAME =?";

    Connection con =getConnection();

    PreparedStatement ps =con.prepareStatement(query);

    ps.setString(1, psw);

    ps.setString(2, host_id);

    result =ps.executeUpdate();

    con.close();

    return result;

}
```

```
public String getPassword(String user)throws Exception{

    String psw =null;

    String query="SELECT PASSWORD FROM LOGIN_DATA WHERE
USERNAME =?";

    Connection con =getConnection();

    PreparedStatement ps =con.prepareStatement(query);

    ps.setString(1, user);
```



```

        ResultSet rs =ps.executeQuery();

        rs.next();

        psw =rs.getString(1);

        con.close();

        return psw;

    }

}package com.nt.dao;


import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;


import com.nt.dbutils.DatabaseUtils;

import com.nt.model.Host;


public class LoginDAO extends DatabaseUtils{


    public int login(String user,String psw,String role)throws Exception{

        int result=0;

        String query ="SELECT COUNT(*) FROM LOGIN_DATA WHERE
        (USERNAME=? AND PASSWORD=? AND ROLE=? AND STATUS=1)";

```



```

        Connection con =getConnection();

        PreparedStatement ps =con.prepareStatement(query);

        ps.setString(1,user);

        ps.setString(2,psw);

        ps.setString(3,role.toLowerCase());

        ResultSet rs =ps.executeQuery();

        rs.next();

        result =rs.getInt(1);

        con.close();

        return result;

    }

```

```

private int registerHost(String user,String psw,Connection con)throws Exception{

        String          query="INSERT          INTO
LOGIN_DATA(USERNAME,PASSWORD,ROLE,STATUS) VALUES(?,?, 'sdn host',0)";

        PreparedStatement ps =con.prepareStatement(query);

        ps.setString(1,user);

        ps.setString(2,psw);

        return ps.executeUpdate();

    }

```



```

public int addNwHost(Host host)throws Exception{

    int result=0;

    String query="INSERT INTO
NETWORK_HOST(HOST_ID,HOST_NAME,HOST_IP,HOST_EMAIL,NW_STATUS)
VALUES(?,?,?,',INACTIVE')";

    Connection con =getConnection();

    con.setAutoCommit(false);

    result= registerHost(host.getHost_id(),host.getPsw(), con);

    if(result>0){

        PreparedStatement ps =con.prepareStatement(query);

        ps.setString(1,host.getHost_id());

        ps.setString(2,host.getHost_name());

        ps.setString(3,host.getHost_ip());

        ps.setString(4,host.getHost_email());

        int r =ps.executeUpdate();

        if(r>0){

            con.commit();

            return 1;

        }else{

            con.rollback();

            return 0;

```

5₆9₂

```
        }  
    }else{  
        con.rollback();  
        return 0;  
    }  
}  
  
}
```