International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

Centralized Context-Aware Firewall configuration in Virtual Network

[1]Mohd Shoaib Adeeb, [2]Mir Abdul Aziz Khan, [3]Mohammed Zaid Uddin, [4]Dr. Mohammed Jameel Hashmi

[1,2,3] B.E. Students, Department of CSE, ISL Engineering College (OU), Hyderabad, India.

[4] Associate Professor, HOD CSE, Department of CSE, ISL Engineering College (OU), Hyderabad, India.

ABSTRACT

Modern virtualized networks require dynamic and automated security configurations to avoid vulnerabilities caused by manual setups. This project presents a system that generates and configures firewall rules automatically based on high-level Network Security Requirements (NSRs) specified by administrators. Initially, the administrator uploads a Security Graph (SG), which contains a list of Access Points (APs) representing logical network nodes. Next, the administrator defines NSRs by specifying the source AP, destination AP, and action (ALLOW or DENY) to control network traffic behavior. The system processes these NSRs to compute the optimal placement of firewall instances, generate a minimal and anomaly-free set of firewall rules, and enforce the required security policies with default behaviors like whitelisting or blacklisting. This approach formally guarantees the correctness of the solution, i.e., that all security requirements are satisfied, and it minimizes the number of needed firewalls and firewall rules. This methodology is extensively evaluated using different metrics and tests on both synthetic and real use cases, and compared to the state-of-the-art solutions, showing its superiority.

1- INTRODUCTION

THE Network Functions Virtualization (NFV) and Software Defined Networking (SDN) paradigms increased agility in network configuration, opening the possibility for users to dynamically request the creation of virtual Service Function Graphs, more commonly known as Service Graph (SG), generalization of the Service Function Chain (SFC) concept. The SG is a new level of abstraction that has been enabled by decoupling computing and physical infrastructures. It is the logical representation of a virtual network, independent from the physical infrastructure where it is embedded. A problem which arises in this context is how to enforce the Network Security Requirements (NSRs) that a SG should satisfy, like data protection and isolation. Traditionally, this task is in charge of a security manager, typically different from the net work manager who defines the logical topology of the service. This separation of roles, if combined with miscommunication or lack of technical knowledge about the domain field of the other person, can lead to the enforcement of incorrect security controls [3]. Furthermore, the configuration of the Network Security Functions (NSFs) is commonly performed manually. This approach not only entails slower reaction when attacks on the network are detected, but it is also prone to human errors, which can lead to the introduction of vulnerabilities. For example, among the hundreds of rules that need to be defined in order to enforce a certain isolation policy, the security manager might miss one ,so making isolation not actually effective. Focusing attention on the control of traffic forwarding, the core NSFs used for this purpose are firewalls. If in traditional scenarios a single point-of-control for packet filtering was usually placed between the local area to protect and the external network where attacks could come from, the flexibility provided by NFV and SDN is encouraging distributed architectures, where more instances are allocated between different service functions, thus being able to address more complex security requirements and to improve efficiency and scalability. Designing and managing these complex architectures requires automation, because positioning and configuring virtual instances manually can likely lead to incorrect or non-optimal solutions, in addition to taking excessive time. Instead, automation, paired with formal verification techniques, is the key for computing provably correct and optimized solutions rapidly enough. Unfortunately, the problem of automating network security configuration has not been sufficiently addressed in the literature so far, despite it represents a key aspect in facing the constantly increasing cybersecurity attacks. Initially, the problem was less pronounced in traditional networks, even though already perceived, because of their intrinsic limitations (e.g., difficulty in allocating multiple firewalls, or creating complex topologies). Later, the advent of network virtualization sharpened the sensibility for this problem but its high complexity restrained the evolution of the state-of the-art solutions, which are all partial. In particular, no prior formal method exists to automatically find an optimum allocation and configuration of virtual firewalls in a given SG. Consequently, automated allocation and configuration of distributed firewalls is still an open research issue. In this context, this article proposes a new methodology that addresses the open issues. The goal is to provide an automatic way to allocate packet filters– the most common and traditional firewall technology– in a SG defined by the service designer, and to create firewall rules automatically ,so as to satisfy the specified security requirements. The method is based on a formal model which provides assurance that the final solution really satisfies the security requirements (correct ness-by-construction). Optimality represents another core aspect of our approach: minimizing the number of firewalls to be allocated and the number of rules to be configured in each one of them increases the performance of the overall architecture, while reducing its cost in terms of employed resources. All this is achieved by a careful formulation of the problem as a partial weighted Maximum Satisfiability

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP41-48

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

Modulo Theories (MaxSMT) problem, which can be solved automatically, providing a solution, if one exists, that is formally guaranteed to satisfy all the hard logical constraints defined in the problem, and that is an optimum one, according to the optimization criteria defined in the problem. Our preliminary ideas about the proposed approach were presented at the NOMS2020conference.

The way traffic flows, network functions, and network security requirements are formalized in the MaxSMT problem is totally new with respect to the preliminary ideas. The new models have been redefined in terms of flows, rather than packets, with the goal of achieving better performance and scalability. All models and algorithms are described with full technical details. The experimental validation has been performed much more extensively than in, by testing the method not only with synthetic topologies, but also with real topologies of production networks. Com pared to the preliminary idea, the new framework shows greatly increased performance. Amore extensive survey of the related literature has been carried out, and a different, more articulated, clarifying example is used here to better highlight the advantages of our proposal with respect to alter native solutions or manual strategies

2- LITERATURE SURVEY

3- Title: "Improving the formal verification of reachability poli cies in virtualized networks.

Author: D. Bringhenti, G. Marchetto, R. Sisto, S. Spinoso, F. Valenza, and J. Yusupov.

Year: 2021.

Description:

Network Function Virtualization (NFV) and Software Defined Networking (SDN) are new emerging paradigms that changed the rules of networking, shifting the focus on dynamicity and programmability. In this new scenario, a very important and challenging task is to detect anomalies in the data plane, especially with the aid of suitable automated software tools. In particular, this operation must be performed within quite strict times, due to the high dynamism introduced by virtualization. In this article, we propose a new network modeling approach that enhances the performance of formal verification of reachability policies, checked by solving a Satisfiability Modulo Theories (SMT) problem. This performance improvement is motivated by the definition of function models that do not work on single packets, but on packet classes. Nonetheless, the modeling approach is comprehensive not only of stateless functions, but also stateful functions such as NATs and firewalls. The implementation of the proposed approach achieves high scalability in complex networked systems consisting of several heterogeneous functions.

Title: "Benchmarking open source NFV MANO systems: OSM and ONAP.

Author: G. M. Yilma, F. Z. Yousaf, V. Sciancalepore, and X. P. Costa

Year: 2020

Description:

With the increasing trend in softwarization and virtualization of network functions and systems, NFV management and orchestration (MANO) solutions are being developed to meet the agile and flexible management requirements of virtualized network services in the 5G era and beyond. In this regard, ETSI ISG NFV has specified a standard MANO system that is used as a reference by vendors as well as open-source MANO projects. These MANO systems are inherently very complex and have a direct impact on the overall performance of NFV systems. However, unlike traditional networking functions and systems, there are no well-defined test methods and KPIs based on which the performance of the NFV-MANO system can be tested, validated and benchmarked. Given the absence of formal MANO specific evaluation techniques based on which the performance and features of a MANO system can be quantified, and compared against, we introduce in this paper a formal benchmarking methodology and KPIs for MANO systems. For illustration purposes, we analyze and compare the performance of the two most popular open-source NFV MANO projects, namely ONAP and OSM, using a complex open-source virtual customer premises equipment (vCPE) VNF. Our results show the current features support, performance to be expected and gaps to be covered in future releases.

Title: Adaptive network slicing in multi-tenant 5GIoTnetworks.

Author: A. Matencio-Escolar , J. M. Alcaraz-Calero , P. Salva-Garcia , J. B. Bernabe, and Q. Wang.

Year: 202

Description:

The Fifth Generation (5G) mobile networking coupled with Internet of Things (IoT) can provide innovative solutions for a wide range of uses cases. The flexibility of virtualized, softwarized and multi-tenant infrastructures and the high performance promised by 5G technology are key to cope with the deployment of the IoT use cases demanded by various vertical businesses. Such 5G IoT use cases incur challenging Quality of Service (QoS) requirements especially connectivity for millions of IoT devices to achieve massive Machine-Type Communication (mMTC). In addition, network slicing is a key enabling technology in 5G multi-tenant networks to create logical virtualized networks for delivering customized solutions to meet diverse QoS requirements. This work presents a 5G IoT framework with network slicing capabilities able to manage a vast number of heterogeneous IoT network slices dynamically on demand. The proposed solution has been empirically tested and validated in five realistic vertical-oriented IoT use cases. The achieved results demonstrate a excellent stability, isolation and scalability while being able to meet extreme QoS requirements even in the most congested and stressful scenarios.

Title: Enabling cyber-attack mitigation techni ques in a software defined network.

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP41-48

Author: A. Pasias, T. Kotsiopoulos, G. Lazaridis, A. Drosou, D. Tzovaras, and P. G. Sarigiannidis.
Year: 2021.
Description:
Software Defined Networking (SDN) is an innovative technology, which can be applied in a plethora of applications and areas. Recently, SDN has been identified as one of the most promising solutions for industrial applications as well. The key features of SDN include the decoupling of the control plane from the data plane and the programmability of the network through application development. Researchers are looking at these features in order to enhance the Quality of Service (QoS) provisioning of modern network applications. To this end, the following work presents the development of an SDN application, capable of mitigating attacks and maximizing the network's QoS, by implementing mixed integer linear programming but also using genetic algorithms. Furthermore, a low-cost, physical SDN testbed was developed in order to evaluate the aforementioned application in a more realistic environment other than only using simulation tools.
Title: Self-organization and resilience for net worked systems: Design principles and open research issues.
Author: S. Dobson, D. Hutchison, A. Mauthe, A. Schaeffer-Filho , P. Smith, and J. P. G. Sterbenz,
Year: 2019
Description:
Networked systems form the backbone of modern society, underpinning critical infrastructures such as electricity, water, transport and commerce, and other essential services (e.g., information, entertainment, and social networks). It is almost inconceivable to contemplate a future without even more dependence on them. Indeed, any unavailability of such critical systems is - even for short periods - a rather bleak prospect. However, due to their increasing size and complexity, they also require some means of autonomic formation and self-organization. This paper identifies the design principles and open research issues in the twin fields of self-organization and resilience for networked systems. In combination, they offer the prospect of combating threats and allowing essential services that run on networked systems to continue operating satisfactorily. This will be achieved, on the one hand, through the (self-)adaptation of networked systems and, on the other hand, through structural and operational resilience techniques to ensure that they can detect, defend against, and ultimately withstand challengesMETHODOLOGY
This project focuses on the automated generation and management of firewall configurations in virtualized network environments based on administrator-defined Network Security Requirements (NSRs). The system starts by allowing the user to upload a Security Graph (SG), which defines the logical structure of the network in terms of Access Points (APs). Administrators then specify communication policies by defining the source AP, destination AP, and action (either ALLOW or DENY) for each communication flow. The system automatically processes these security rules to determine where firewall instances should be placed within the network and generates optimized, minimal, and anomaly-free sets of firewall rules for each firewall. The approach supports both whitelisting and blacklisting models for traffic management and ensures that all user-defined security policies are formally satisfied while minimizing resource consumption. Designed with scalability in mind, the project also prepares for future enhancements such as conflict detection, dynamic visualization of network topology, real firewall system integration, and multi-user management features, making it a comprehensive solution for modern network security automation.
User Interface Design
To connect with server user must give their username and password then only they can able to connect the server. If the user already exits directly can login into the server else user must register their details such as username, password, Email id, City and Country into the server. Database will create the account for the entire user to maintain upload and download rate. Name will be set as user id. Logging in is usually used to enter a specific page. It will search the query and display the query.
User Authentication Module
This module provides a secure login system for administrators to access the tool. It ensures that only authorized users can manage and configure the network security settings. Admin credentials are verified before granting access to the main dashboard.
Security Graph (SG) Upload Module
In this module, users upload the network's logical structure, known as the Security Graph, which defines all the Access Points (APs) in the virtual network. The system processes the uploaded data and prepares it for further configuration.
NSR Definition Module
This module allows administrators to define Network Security Requirements by selecting the source AP, destination AP, and the desired action (ALLOW or DENY). It captures all communication policies needed to enforce network security.
Firewall Allocation Module
The firewall allocation module decides the best locations within the network (Access Points) to place firewall instances. It ensures that firewalls are optimally positioned to enforce all specified security rules using the least number of resources.
Firewall Rule Optimization
Firewall rule optimization is the process of improving firewall configurations to make them more **efficient**, **correct**, and **easy to manage**.
In any network, firewalls are used to allow or block specific types of traffic based on defined rules. However, as networks grow larger and more dynamic, firewall rule sets often become **very large, complex, and redundant**. This can cause problems like slow packet processing, wasted memory, and even security vulnerabilities due to

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP41-48

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 2s, 2025

mistakes.

The main goal of firewall rule optimization is to **minimize** the number of rules while still **preserving the correct behavior** of the firewall. This involves detecting and removing **redundant rules**, **conflicting rules**, and **unnecessary overlaps**. For example, if two rules allow the same traffic, they can be merged into one. If a rule is never reached because a previous rule already matches the traffic, it can be safely removed.

Optimizing firewall rules not only makes the firewall faster but also **improves security**, because a smaller and cleaner rule set is easier to audit and verify. In modern systems, optimization also tries to **reduce resource usage** like CPU, memory, and storage on the devices running firewalls, especially important in virtualized cloud environments.

Different techniques are used for firewall rule optimization, including **heuristic methods**, **graph-based models**, and **formal methods** like **SAT solving** or **SMT solving**. In advanced systems, optimization happens automatically during the design of the network, rather than being manually adjusted after problems are noticed.

Thus, firewall rule optimization is a critical step for building **secure, high-performance, and scalable** networks, particularly in environments using **virtual networks and dynamic topologies**.

4- REQUIREMENTS

ENGINEERING HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shoulds what the system do and not how it should be implemented.

HARDWARE

- PROCESSOR : PENTIUM IV 2.6 GHz, Intel Core 2 Duo.
- RAM : 512 MB DD RAM
- MONITOR : 15” COLOR
- HARD DISK : 40GB

SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

- Front End : J2EE (JSP, SERVLET)
- Back End : MY SQL 5.5
- Operating System : Windows 10
- IDE : Eclipse

FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs. The outsourced computation is data is more secured.

User Request

- **Input**: The user submits a request for data or files.
- **Behavior**: The system processes the request and checks the user's credentials for valid access.
- **Output**: If the request is valid, access to the data is granted. Otherwise, an error message is displayed.

File Details

- **Input**: User selects a file and requests to view its details.
- **Behavior**: The system fetches file metadata such as name, size, and type.
- **Output**: The file details are displayed to the user, allowing them to view more information before proceeding with further actions (such as downloading or requesting keys).**NON-FUNCTIONAL REQUIREMENTS**

The major non-functional Requirements of the system are as 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 a 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

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP41-48

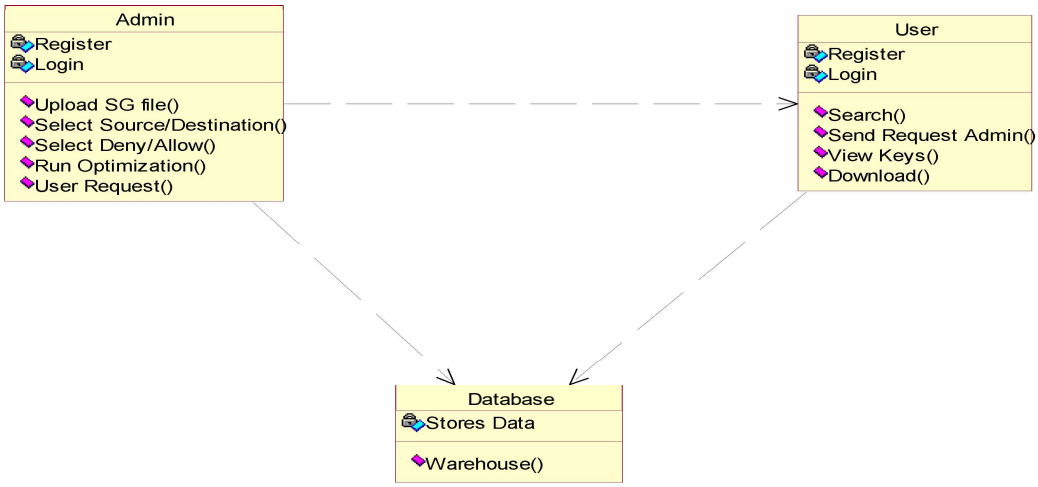
International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2s, 2025

5-DESIGN ENGINEERING

Design Engineering deals with the various UML [Unified Modeling Language] diagrams for the implementation of a project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into a representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into a product

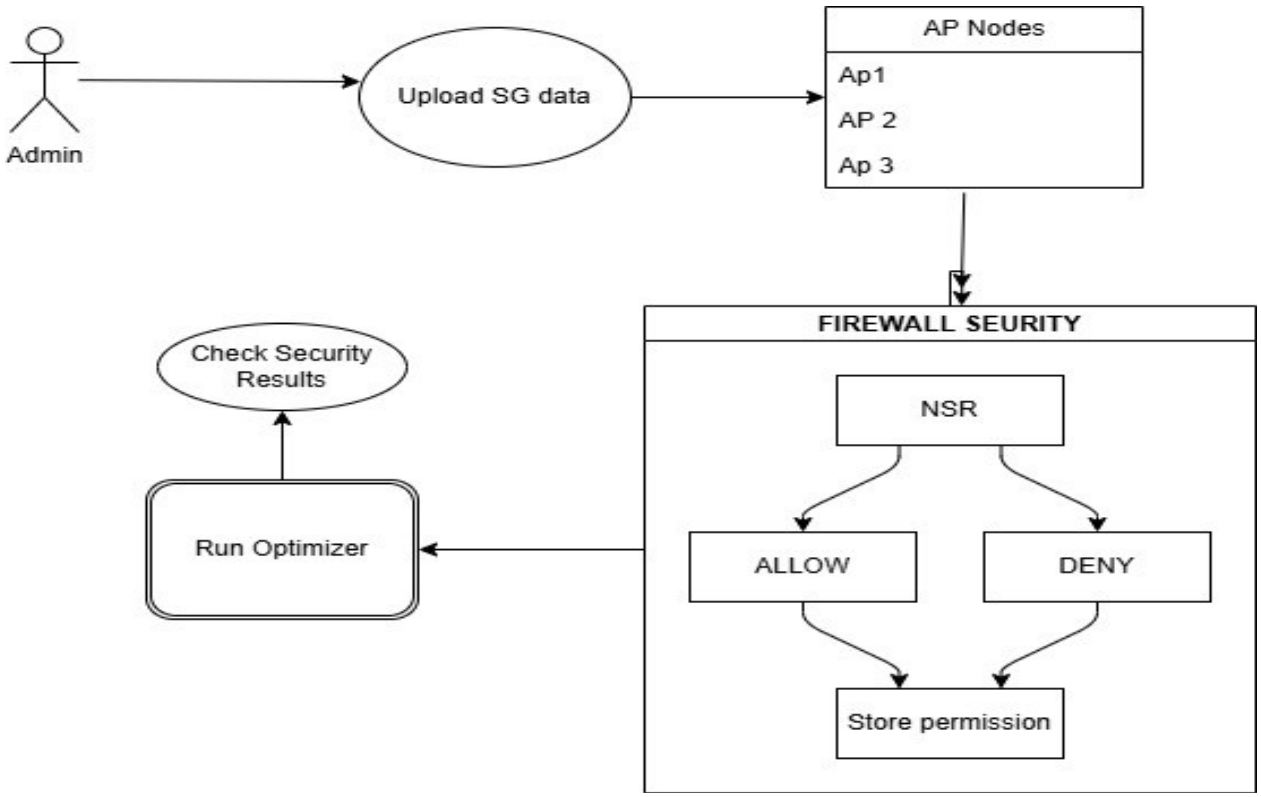


This class diagram represents how the classes with attributes and methods are linked together to perform the verification

SYSTEM ARCHITECTURE:

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP41-48

International Journal of Information Technology & Computer Engineering
ISSN 2347-3657
Volume 13, Issue 2s, 2025



6-IMPLEMENTATION

LIST:

Lists are implemented in the JCF via the java.util.List interface. It defines a list as essentially a more flexible version of an array. Elements have a specific order, and duplicate elements are allowed. Elements can be placed in a specific position. They can also be searched for within the list. Two concrete classes implement List. The first is java.util.ArrayList, which implements the list as an array. Whenever functions specific to a list are required, the class moves the elements around within the array in order to do it. The other implementation is java.util.LinkedList. This class stores the elements in nodes that each have a pointer to the previous and next nodes in the list. The list can be traversed by following the pointers, and elements can be added or removed simply by changing the pointers around to place the node in its proper place.

SET:

Java's java.util.Set interface defines the set. A set can't have any duplicate elements in it. Additionally, the set has no set order. As such, elements can't be found by index. Set is implemented by java.util.HashSet,java.util.LinkedHashSet, and java.util.TreeSet. HashSet uses a hash table. More specifically, it uses a java.util.HashMap to store the hashes and elements and to prevent duplicates. Java.util.LinkedHashSet extends this by creating a doubly linked list that links all of the elements by their insertion order. This ensures that the iteration order over the set is predictable. java.util.TreeSet uses a red-black tree implemented by a java.util.TreeMap. The red-black tree makes sure that there are no duplicates. Additionally, it allows Tree Set to implement java.util.SortedSet. The java.util.Set interface is extended by the java.util.SortedSet interface. Unlike a regular set, the elements in a sorted set are sorted, either by the element's compareTo() method or a method provided to the constructor of the sorted set. The first and last elements of the sorted set can be retrieved, and subsets can be created via minimum and maximum values, as well as beginning or ending at the beginning or ending of the sorted set. The SortedSet interface is implemented by java.util.TreeSet

java.util.SortedSet is extended further via the java.util.NavigableSet interface. It's similar to SortedSet, but there are a few additional methods. The floor(), ceiling(), lower(), and higher() methods find an element in the set that's close to the parameter. Additionally, a descending iterator over the items in the set is provided. As with SortedSet, java.util.TreeSet implements NavigableSet.

MAP:

Maps are defined by the java.util.Map interface in Java.

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP41-48

Maps are simple data structures that associate a key with a value. The element is the value. This lets the map be very flexible. If the key is the hash code of the element, the map is essentially a set. If it's just an increasing number, it becomes a list. Maps are implemented by java.util.HashMap, java.util.LinkedHashMap, and java.util.TreeMap. HashMap uses a hash table. The hashes of the keys are used to find the values in various buckets. LinkedHashMap extends this by creating a doubly linked list between the elements. This allows the elements to be accessed in the order in which they were inserted into the map. TreeMap, in contrast to HashMap and LinkedHashMap, uses a red-black tree. The keys are used as the values for the nodes in the tree, and the nodes point to the values in the map

THREAD:

Simply put, a *thread* is a program's path of execution. Most programs written today run as a single thread, causing problems when multiple events or actions need to occur at the same time. Let's say, for example, a program is not capable of drawing pictures while reading keystrokes. The program must give its full attention to the keyboard input, lacking the ability to handle more than one event at a time. The ideal solution to this problem is the seamless execution of two or more sections of a program at the same time

CREATING THREAD:

Java's creators have graciously designed two ways of creating threads: implementing an interface and extending a class. Extending a class is the way Java inherits methods and variables from a parent class. In this case, one can only extend or inherit from a single parent class. This limitation within Java can be overcome by implementing interfaces, which is the most common way to create threads

7-CONCLUSION

In this article, we have designed an automated system for configuring network security, specifically focusing on firewall rule optimization and placement within a virtual network environment. By leveraging innovative algorithms like the Partial Weighted Maximum Satisfiability Modulo Theories (Max-SMT), the system ensures that security requirements are enforced correctly while rminimizing the resource consumption required for firewall rules and placement. The integration of features like dynamic security graph handling, user-specific policy enforcement, and secure data access management enhances the overall security posture of virtual networks. Additionally, the system offers scalability and flexibility for future extensions, such as anomaly detection, multi-user support, and real-time firewall configuration updates. Ultimately, this system automates and streamlines the process of enforcing network security policies, making it more efficient and reliable, particularly in complex.

While the current system provides an effective solution for automating firewall rule optimization and placement, several future enhancements can further improve its capabilities and user experience. One key enhancement could be the integration of anomaly detection to identify and resolve conflicting rules or potential security vulnerabilities automatically. Additionally, dynamic topology visualization could be introduced to provide network administrators with real-time graphical representations of firewall placements and traffic flow, making it easier to monitor and adjust configurations.Multi-user support and role-based access control could also be implemented, allowing different users (e.g., administrators, MSPs, owners, and regular users) to interact with the system based on their specific privileges. Furthermore, real-time updates and the ability to handle security configuration changes on-the-fly would ensure the system remains adaptive in responding to emerging threats. Integration with existing firewall configuration exporters could allow seamless deployment of the system into real-world environments, further enhancing its practical utility. Lastly, file upload support for security graphs would enable users to directly import network security data, improving flexibility and ease of use. These enhancements will ensure the system remains scalable, adaptable, and capable of meeting the ever-evolving demands of modern network security

REFERENCES

[1] J. Halpern and C. Pignataro, “Service function chaining (SFC) architecture,” *RFC 7665*, pp. 1–32, 2015.

[2] J. Garay, J. Matias, J. Unzilla, and E. Jacob, “Service description in the NFV revolution: Trends, challenges and a way forward,” *IEEE Commun. Mag.*, vol. 54, no. 3, pp. 68–74, Mar. 2016.

[3] S. Singh, Y. Jeong, and J. H. Park, “A survey on cloud computing security: Issues, threats, and solutions,” *J. Netw. Comput. Appl.*, vol. 75, pp. 200–222, 2016.

[4] R. Mijumbi et al., “Management and orchestration challenges in network functions virtualization,” *IEEE Commun. Mag.*, vol. 54, no. 1, pp. 98–105, Jan. 2016.

[5] D. Bringhenti et al., “Improving the formal verification of reachability policies in virtualized networks,” *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 713–728, Mar. 2021.

[6] Y.-M. Kim and M. Kang, “Formal verification of SDN-based firewalls by using TLA+,” *IEEE Access*, vol. 8, pp. 52100–52112, 2020.

[7] Y. Bartal, A. Mayer, K. Nissim, and A. Wool, “Firmato: A novel firewall management toolkit,” *ACM Trans. Comput. Syst.*, vol. 22, no. 4, pp. 381–420, 2004.

[8] P. Verma and A. Prakash, “FACE: A firewall analysis and configuration engine,” in *Proc. IEEE/IPSJ Int. Symp. Appl. Internet*, 2005, pp. 74–81.

[9] J. D. Guttman, “Filtering postures: Local enforcement for global policies,” in *Proc. IEEE Symp. Secur. Privacy*, 1997, pp. 120–129.

[10] M. A. Rahman and E. Al-Shaer, “Automated synthesis of distributed network access controls: A formal framework with refinement,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 2, pp. 416–430, Feb. 2017.

[11] A. Gember-Jacobson et al., “Automatically repairing network control planes using an abstract representation,” in *Proc. 26th Symp. Oper. Syst. Princ.*, 2017, pp. 359–373.

[12] C. Basile et al., “Adding support for automatic

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP41-48

enforcement of security policies in NFV networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 707–720, Apr. 2019.
[13] D. Bringhenti et al., "Automated optimal firewall orchestration and configuration in virtualized networks," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, 2020, pp. 1–7.
[14] A. Mayer, A. Wool, and E. Ziskind, "Fang: A firewall analysis engine," in *Proc. IEEE Symp. Secur. Privacy*, 2000, pp. 177–187.
[15] J. Govaerts, A. K. Bandara, and K. Curran, "A formal logic approach to firewall packet filtering analysis and generation," *Artif. Intell. Rev.*, vol. 29, no. 3/4, pp. 223–248, 2008.
[16] D. Ranathunga, M. Roughan, P. Kernick, and N. Falkner, "The mathematical foundations for mapping policies to network devices," in *Proc. 13th Int. Joint Conf. E-Bus. Telecommun.*, 2016, pp. 197–206.
[17] N. B. Y. B. Souayeh and A. Bouhoula, "A fully automatic approach for fixing firewall misconfigurations," in *Proc. 11th IEEE Int. Conf. Comput. Inf. Technol.*, 2011, pp. 461–466.
[18] K. Adi, L. Hamza, and L. Pene, "Automatic security policy enforcement in computer systems," *Comput. Secur.*, vol. 73, pp. 156–171, 2018.
[19] A. Tsuchiya et al., "Software-defined networking firewall for industry 4.0 manufacturing systems," *J. Ind. Eng. Manage.*, vol. 11, 2018, Art. no. 318.
[20] P. Salva-Garcia et al., "Towards automatic deployment of virtual firewalls to support secure mMTC in 5G networks," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2019, pp. 385–390.
[21] S. Dobson et al., "Self-organization and resilience for networked systems: Design principles and open research issues," *Proc. IEEE*, vol. 107, no. 4, pp. 819–834, Apr. 2019.
[22] J. Zhang et al., "Enabling efficient service function chaining by integrating NFV and SDN: Architecture, challenges and opportunities," *IEEE Netw.*, vol. 32, no. 6, pp. 152–159, Nov./Dec. 2018.
[23] E. J. Scheid et al., "INSpIRE: Integrated NFV-based intent refinement environment," in *Proc. IFIP/IEEE Symp. Integr. Netw. Serv. Manage.*, 2017, pp. 186–194.
[24] Y. Han et al., "An intent-based network virtualization platform for SDN," in *Proc. 12th Int. Conf. Netw. Serv. Manage.*, 2016, pp. 353–358.
[25] A. S. Jacobs et al., "Refining network intents for self-driving networks," *Comput. Commun. Rev.*, vol. 48, no. 5, pp. 55–63, 2018.
[26] Z. Hao, Z. Lin, and R. Li, "A SDN/NFV security protection architecture with a function composition algorithm based on trie," in *Proc. 2nd Int. Conf. Comput. Sci. Appl. Eng.*, 2018, Art. no. 176.
[27] Y. Liu et al., "A dynamic composition mechanism of security service chaining oriented to SDN/NFV-enabled networks," *IEEE Access*, vol. 6, pp. 53918–53929, 2018.
[28] A. Pasias et al., "Enabling cyber-attack mitigation techniques in a software defined network," in *Proc. IEEE Int. Conf. Cyber Secur. Resilience*, 2021, pp. 497–502.
[29] A. Matencio-Escolar et al., "Adaptive network slicing in multi-tenant 5G IoT networks," *IEEE Access*, vol. 9, pp. 14048–14069, 2021.
[30] M. Yoon, S. Chen, and Z. Zhang, "Minimizing the maximum firewall rule set in a network with multiple firewalls," *IEEE Trans. Comput.*, vol. 59, no. 2, pp. 218–230, Feb. 2010.
[31] N. Schnepf et al., "Rule-based synthesis of chains of security functions for software-defined networks," *Electron. Commun. EASST*, vol. 76, pp. 1–19, 2018.
[32] E. Al-Shaer et al., "Conflict classification and analysis of distributed firewall policies," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 10, pp. 2069–2084, Oct. 2005.
[33] F. Valenza et al., "Classification and analysis of communication protection policy anomalies," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2601–2614, Oct. 2017.
[34] J. Gil-Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.
[35] K. Hida and S. Kuribayashi, "Joint deployment of virtual routing function and virtual firewall function in NFV-based network with minimum network cost," in *Proc. 21st Int. Conf. Adv. Netw.-Based Inf. Syst.*, 2018, pp. 333–345.

.

DOI: https://doi.org/10.62647/IJITCE2025V13I2sPP41-48