Journal of Theoretical and Applied Information Technology
20th February 2016. Vol.84. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PROVIDING SECURITY IN CLOUD BASED NETWORKS THROUGH SOFTWARE DEFINED NETWORKS

[1]Dr.CHINTHAGUNTA MUKUNDHA, [2] Dr.I.SURYA PRABHA,[3]KARNAM SREENU
[1]Associate Professor, IT Department,SNIST, Hyderabad -501301,Telangana, India
[2] Professor, IT Department,IARE, Hyderabad -500043, Telangana, India
[3]Assistant Professor ,IT Department,SNIST,Hyderabad -501301,Telangana, India
E-mail: [1]mukundhach@gmail.com,[2] ipsurya17@gmail.com,[3]karnam.sreenu@gmail.com

ABSTRACT

Now days we have observed that the fast change in the cloud network by the Software Defined Networking (SDN) paradigm that differentiate the control plane from the data plane to give the flexibility for programmability and centralized control of the cloud networks, SDN networks not only provide simplification of cloud network management it also provides more security with SDN by implementing firewalls with in the SDNs. The demand of cloud increased day by day with the increasing of usage of cloud. The SDN is provided with OpenFlow network, cloud network states are dynamically updated and configurations are frequently changed. Open Flow accepts various Field actions that can dynamically change the packet headers. A firewall embedded in SDN can immediately enforce updated rules in the firewall policy to check security violations. Cloud computing allows all categories of users to use applications without installation and access their personal files at any system with internet access .Here we present that the way forward is to integrate SDN and fully utilize its feature to solve the security problems in cloud networks. We focus on the security aspect and investigate how to enhance the security with SDN firewalls for the cloud networks.

Keywords: *Cloud Network, Software Defined Networks,Firewalls,Security,Cloud Computing*.

1. INTRODUCTION

The Software Defined Networking (SDN) is a advanced and innovative force in the networking industry that change almost every player including network organizers, manufactures, ISPs and cloud service providers. With SDN, the low-level device configuration and management can be handled by the centralized software controller which facilitates the upgrade of operation and controllability. By Organizing and distributing the network state with a system perspective, SDN provides the administrators to mining the complex protocol specifications with agility and flexibility to control the networks. The SDN-enabled Network Functions makes it possible for the Internet and cloud service providers to deliver their services in the market through improving the service in terms of security and Quality of Service (QoS) .

OpenFlow not only shows tremendous opportunities to networking, but also gets great challenges for building SDN firewalls as follows:

- **Examining Dynamic Network Policy Updates:** In an OpenFlow network, network states are effectively changed and configurations are periodically changed. Thus, simply checking flow packet violation by monitoring the packets in behaviors in a firewall application is not effective, since flow policy abuse, which indicates exiting flow policies violate the firewall policy induced by the changes of network levels and configurations such as updating flow entries and firewall rules, should be detected and resolved in real time as well.
- **Checking Indirect Security Violations:** OpenFlow allows different Set-Field actions that can dynamically change the packet headers. Adversaries could take advantage of this feature to strategically

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

enable flow conditions that would evade security mechanisms (e.g.

firewalls). In addition, flow rules may overlay each other in a flow table, indicating intra-table dependency of flow rules. The rules in a firewall policy may also overlay each other. These terms dependencies could also be leveraged by malicious OpenFlow applications to ignore firewalls.

- **Architecture Option:** A Unify SDN firewall can immediately enforce updated rules in the firewall policy to check security abuse. However, when a flow policy violation is find, it can only reject the new flow policy or flush resident flow policy that causes the violation.
-
- If only partial packets matching a flow policy disrupt the firewall policy, eliminating the flow policy may drop fair traffic. A distributed SDN firewall may directly resolve flow policy violations by propagating and enforcing the firewall policy at each individual entry of the flow in the network. However, a distributed firewall needs a complicated abrogation and repropagation mechanism to handle dynamic policy updates.
  - **Stateful Monitoring:** At present, OpenFlow only afford very limited access to packet-level data in the controller. In addition, the OpenFlow forwarding plane is stateless and unable to actively monitor flow status without the involvement of the controller. Therefore, it is impose to fully support stateful packet inspection in SDN firewalls.

Behind the SDN structure that differentiates the control and data plane, SDN delivers four visible features to the networking field:

- **Central control and coordination:** The logically centralized Control model is an important part of the SDN architecture which mitigates the overhead from the classical distributed mechanisms based on protocols. Although the centralized approach is often questioned for its scalability, it can deliver the state and policy changes more comfortably than the distributed methods in a managed domain. The coordination feature also makes it feasible that when one of the controllers fails, other standby ones can take over the management tasks to escape service breakage, which poses a great challenge for the distributed approach.
- **Programmability:** For both the control plane and the data plane, SDN makes implementation and deployment of the new functionality faster and easier, and hence speeding up the innovation at both hardware and software level. This agility can reduce the cost for service and network providers in terms of Operational Expenditure (OPEX) as the management can be powered by SDN applications in an automatic manner. By avoiding the unnecessary replacement of the underlying hardware through software update, it can also bring down the Capital Expenditure (CAPEX) and facilitate the adoption by the cloud providers.
- **Virtualized abstraction:** The layered design of SDN hides the complexity of hardware devices from the control plane and SDN applications. Through virtualized abstraction, SDN allows the managed cloud network to be divided into virtual networks that share the same infrastructure but are governed by different policy and security requirements. Such flexibility greatly promotes the sharing, aggregation and management of available resources and enables dynamical reconfiguration and changes of policy.
- **Openness: T**he open standards of SDN such as OpenFlow help build and develop open sourced communities that attract brain power and speed up the innovation. Such openness combined with programming APIs can promote the networking research by allowing researchers to experiment with novel ideas through fast prototyping and testing. It also benefits the interoperability with the legacy infrastructure and allows different operators and providers to collaborate through the SDN framework.

Journal of Theoretical and Applied Information Technology
20th February 2016. Vol.84. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

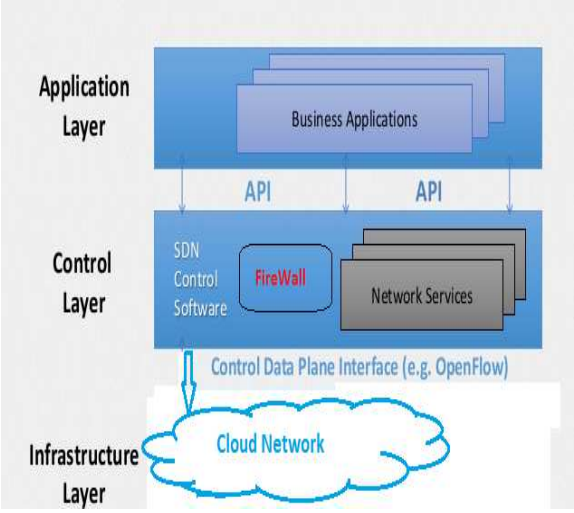

Fig1 : SDN Architecture with fire wall in Cloud Networks

As illustrated in Fig:1 of an envisioned the current trend of convergence in such networks can benefit from SDN to enhance resource utilization, network management and security in the multiservice and multi-vendor environment. The security of SDN deserves our special attention for the challenges it brings and also the opportunities to enhance the network security. In this article, we review the recent work on SDN for cloud networks and discuss how SDN solutions can improve security in such a dynamic environment. By surveying the SDN-based security solutions, we identify the design goals and describe our approach of utilizing SDN for security enhancement in cloud networks.

2. THEORETICAL BASIS AND ITERATURE REVIEW

In this section we discuss the latest research on SDN security and categorize the selected proposals in terms of their target environment into four groups: enterprise networks, cloud & data center, home & edge access, and general design. As we focus on how to utilize SDN to enhance network security, it is worth noticing that the security concern for the SDN itself is also an important topic where the existing study has identified seven threat vectors that may enable the exploit of SDN vulnerabilities and further propose a design to achieve secure and dependable SDN platform.
The work of presents the SDN-based implementation of four algorithms for Anomaly Detection System (ADS) and advocates that such programmable solution deployed in a home network environment can achieve more accurate identification of malicious activity comparing to the one that is deployed at the ISP. Their SDN-based solution can also alleviate the load of ISP from monitoring a large number of networks and promote collective detection of global network problems by feeding the monitoring results to external entities. This work requires modification of the SDN control plane. It does not consider features of local optimization, cloud network or cross-domain interoperability.

One of the most important issues related to cloud security risks is data integrity. The data stored in the cloud may suffer from damage during transition operations from or to the cloud storage provider. Cachinetal gives examples of the risk of attacks from both inside and outside the cloud provider, such as the recently attacked Red Hat Linux's distribution servers. One of the solutions that they propose is to use a Byzantine fault-tolerant replication protocol within the cloud. Hendricksetal. State that this solution can avoid data corruption caused by some components in the cloud. However, Cachinetal.Claim that using the Byzantine fault tolerant replication protocol within the cloud is unsuitable due to the fact that the servers belonging to cloud providers use the same system installations and are physically located in the same place.

According to Garfinkel, another security risk that may occur with a cloud provider, such as the Amazon cloud service, is a hacked password or data intrusion. If someone gains access to an Amazon account password, they will be able to access all of the account's instances and resources. Thus the stolen password allows the hacker to erase all the information inside any virtual machine instance for the stolen user account, modify it, or even disable its services. Furthermore, there is a possibility for the user's email(Amazon user name) to be hacked (see for a discussion of the potential risks of email), and since Amazon allows a lost password to be reset by email, the hacker may still be able to log in to the account after receiving the new reset password.

NetFuse is a new proposal for cloud and data center environment to protect it against traffic surges that origin from either security attacks,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

operator errors, or routing misconfiguration. NetFuse uses both passive listening and adaptive active query to enable effective monitoring of network status. It utilizes a multi-dimensional aggregation to find the suspicious flow clusters. To improve efficiency and responsiveness, NetFuse further adopts a toxin-antitoxin mechanism to adaptively shape the flow rate according to application feedback. The design is realized as a proxy

Between the OpenFlow switches and the controller.NetFuse follows the OpenFlow standard but does not modify the control and data plane. It delivers local optimization by relieving the heavy load from the controller such as flow redirection, delay injection, and blocking. NetFuse does consider the features of mobile and cross-domain.

CloudWatcher utilizes the advantage of SDN to build a framework that can efficiently monitor services in large and dynamic cloud networks. The proposed scripting language enables operators to employ security monitoring as a service in a convenient fashion. CloudWatcher includes four routing algorithms to optimally reroute traffic to a security monitoring node. The framework consists of three key components: device and policy manager, routing rule generator, and flow rule enforcer. CloudWatcher does not demand changes on control or data plane, and utilizes the routing algorithms to optimize security monitoring. However, it does not consider the features of mobile and cross-domain.

3. ENHANCING SECURITY IN CLOUD NETWORKS WITH SDN FIREWALLS

We propose a comprehensive framework called FLOWGUARD to facilitate accurate detection as well as flexible resolution of firewall policy violations in dynamic OpenFlow networks along with a variety of toolkits for visualization, optimization, migration, and integration of SDN firewalls.

The violation detection approach in FLOWGUARD detects violations by examining flow path space against firewall authorization space, and is capable of tracking flow paths in the entire network with respect to dynamic packet modifications, and checking rule dependencies in both flow tables and firewall policies. To support network-wide access control in an OpenFlow network, an SDN firewall needs to not only check violations at the ingress switch of each flow, but also track the flow path and then clearly identify both the original source and final destination of each flow in the network. As a preliminary solution, we leverage NetPlumber [3] as a baseline for building the flow tracking mechanism, because it offers several features that can fulfill some of our requirements for tracking flows, such as support for arbitrary header modifications, automatic rule dependency detection, and real-time response.

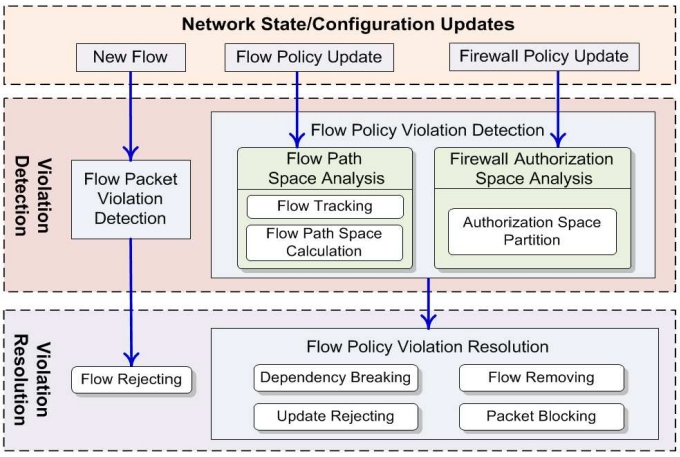

Fig 2: Flow Guard Violation Resolution Mechanism

To calculate flow path space, we abstract fields, which are needed for checking firewall policy violations, from the pattern expression of a flow rule to represent the space of

corresponding flow path. In addition, we reorganize these pair to specify a flow path space. Then, we define three kinds of spaces for representing a flow path space: (1) Incoming Space represents original header spaces of packets that can pass through the flow path; (2) Outgoing Space represents final header spaces of packets after the packets pass through the flow path; and (3) Tracked Space represents original source address and final destination address of header spaces of packets that can pass through the flow path. For accurately detecting firewall policy violations, the dependency relations between “allow” rules and “deny” rules in the firewall policy should be decoupled. We propose a concept of firewall authorization space, which represents a collection of all packets either allowed or denied by the firewall rules. We then introduce

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

a space partition approach, which represents rules with header space and performs various set operations on rules, to convert a list of firewall rules into two disjoint authorization sub-spaces, denied authorization space and allowed authorization space.

Once the space of a flow path and the firewall authorization space of the firewall policy are calculated, we identify violations through checking the tracked space of the flow path against the denied authorization space of the firewall policy. If these two spaces overlap each other, we call the overlapping space as the violated space, which indicates a firewall policy violation. There are two kinds of violations: Entire Violation (the denied authorization space includes the whole tracked space); and Partial Violation (the denied authorization space partially includes the tracked space).

For resolving various firewall policy violations in real time, we introduce a flexible and effective violation resolution mechanism. Figure 2 demonstrates how FLOWGUARD adopts various violation resolution strategies to resolve different firewall policy violations in terms of new flows and various update operations on both flow and firewall policies in OpenFlow-based networks. Since a firewall application can directly reject the new flows which violate
the firewall policy, it would be straightforward to resolve
flow packet violations.

There are four resolution strategies used for handling flow policy violations: 1. Dependency Breaking: When a new flow policy is being added into the network switches, this single flow policy may not violate the firewall policy. However, the rules in this new flow policy may overlap with the rules of other existing flow policies. Since rule dependencies could cause unexpected changes in packet headers of flows, they may lead to new firewall policy violations. Note that such kind of violations can be also incurred by other changes of network states, such as modifying flow entries and updating firewall rules. We introduce two alternative mechanisms, Flow Rerouting and Flow Tagging, for breaking rule dependencies that cause violations. 2. Update Rejecting: Adding a new flow policy, or changing or deleting an existing flow policy, can cause new entire violations. Applying this strategy, the update operation is rejected directly.3. Flow Removing: When updating (adding, changing, or deleting) a rules in the firewall policy, the firewall application examines all existing flow policies applying the updated rules and detect new entire violations. Using this strategy, all flow entries
associated with a flow path, which entirely violates the firewall policy, are removed from the network switches and 4. Packet Blocking: For any partial violation detected by the firewall application, this strategy can be applied. There may exist two ways to block packets of a flow. First, if the flow is a new flow, the firewall application only needs to block it in the ingress switch of the flow. Second, if the flow is an old flow, the firewall application needs to block the packets in both ingress and egress switches.

4. CHALLENGES AND REQUIREMENTS

Based on different environments we identify the different challenges and requirements.

- Monitoring Overhead: The OpenFlow-based monitoring

Schemes suffer from limitation in terms of high overhead and incomplete sample information. FleXam provides a good example toward this challenge.

- Multi-Access: The operational environment consists of different technologies and cloud operators leading to complex negotiation process, privacy concern, and potential conflicting policy and QoS requirement that pose a challenge to the security enforcement.
- Deployment: Although SDN has openness in its nature, any solution deployed needs to face the challenge of backward compatibility and interoperability as cloud operators need to maintain different generations of technologies and intercommunicate with other providers.

For a sound SDN design for security enhancement, we need
to meet several requirements:

- Interoperability: Handling information exchange between different elements are crucial for SDN security design for

Wireless mobile environment. The recent trend of cellular
offloading also makes this relevant since WiFi and Cellular
management are used to be separate, especially the security part. Traditional distributed protocol is incapable for this due to the complexity and privacy issues.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

• Responsiveness: Processing events in wireless mobile Networks should be timely, either in reactive or proactive
Manner. Efficient triggering and local optimization are valuable.
• Compatibility: To maximize the value of openness, using
a standard API is required, such as OpenFlow.
• Adaptation: Due to mobility and network condition changes, a design should be adaptive by monitoring and efficiently detecting events both in the network and from
User activities.

5. SYSTEM DESIGN AND USE CASES

To illustrate how to enhance security in wireless mobile networks, we propose a SDN-based framework as depicted
in Figure 1. The key elements in our framework include central controller, and security layer for cloud networks. The local agents are deployed close to the wireless edge access to meet the requirements of responsiveness, adaptation, simplicity Optimization techniques are employed by local agents including flow sampling, tracking client records and mobility profile. Instead of inserting actuation triggers in the data plane, local agents take the responsibility to adaptively query information from the underlying devices

and report to the controller. This provides transparency to both data plane and controller and hence all eviating the monitoring load on the central controller.

The central controller is the management entity running OpenFlow to control switches and OpenFlow-enabled cloud devices. To improve compatibility and simplicity, our design
adopts only the mature and open standards to manage the
network.The security layer resides on top of the control plane and aims to meet the requirements of interoperability,
compatibility, and simplicity.

Our framework aims at the wireless mobile environment.
We enhance the scalability by using local agents to process the local events. Since reaction must be fast in the wireless mobile environment, the local progressing reduces the latency from data plane to controller and therefore delivers
better performance. We further utilize the security layer to
achieve interoperability across different access domains that
are managed by different operators.

We highlight three potential use cases for our design.
• Complementing an Intrusion Detection System (IDS) - The local servers will regulating monitoring tasks by using triggers and flexible sampling to fetch necessary information from edge network and end host and feeding such information to IDS service such as Snort.
• Prevention of DoS - avoiding resource waste such as spectrum, if the malicious traffic are only dropped in the core network switches rather than dropped as early as possible. The local agent coordinates the operation by installing rules to drop packets and inform the central controller.
• Secure handoff for mobility - Proactively deliver security
keys and credentials to target networks to improve efficiency for handoff.

6. FEATURE WORK

We will further develop and integrate stateful packet inspection and analysis modules in the FLOWGUARD framework to support the implementation of stateful SDN firewalls. Furthermore, we plan to integrate our conflict detection and resolution solution into popular SDN controllers to build a robust security enforcement kernel for the controllers. We aim to provide a framework to supply a secure cloud database that will guarantee to prevent security risks facing the cloud computing community. This framework will apply multi-clouds and the secret sharing algorithm to reduce the risk of data intrusion and the loss of service availability in the cloud and ensure data integrity. In relation to data intrusion and data integrity, assume we want to distribute the data into three different cloud providers, and we apply the secret sharing algorithm on the stored data in the cloud provider. An intruder needs to retrieve at least three values to be able to find out the real value that we want to hide from the intruder. Regarding service availability risk or loss of data, if we replicate the data into different cloud providers, we could argue that the data loss risk will be reduced.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

7. CONCLUSION

The maturity of OpenFlow standards and fast development
of SDN have made the concept of SDN widely accepted by the mobile industry. For the advantages of SDN such as programmability, abstraction, and openness, we advocate
that SDN can enhance security in cloud networks through novel design. Based on our investigation of existing solutions, we highlight the key elements and sketch out our design. As part of our on-going work for the security enhancement framework, we believe this study can shed light on how we use SDN to improve network security and promote the adoption of SDN to the future wireless mobile networks.

REFERENCES:

[1] Sivaraman, Anirudh, et al. No silver bullet: extending SDN to the data plane. Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks. ACM, 2013.
[2] Kephart J O, Chess D M. The vision of autonomiccomputing[J]. Computer, 2003, 36(1): 41-50.
[3] Miller. B. The autonomic computing edge: Can you CHOP up autonomic computing? Technical report, IBM, 2008. http://www.ibm.com/developerworks/autonomic/library/ac-edge4.
[4] EFIPSANS project. http://www.efipsans.org/.
[5] ANA (Autonomic Network Architecture) Project. http://www.ana-project.org/.
[6] HAGGLE Project. http://www.haggleproject.org
[7] CASCADAS project. http://www.cascadas-project. org.
[8] BIONETS project. http://www.bionets.eu/.
[9] OpenFlow Specification 1.3.0. https://www. opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf.
[10] L. E. Li, Z. M. Mao, J. Rexford. Toward Software-Defined Cellular Networks. In Proceedings of IEEE EWSDN 2012.
[11] X. Jin, L. E. Li, L. Vanbever, J. Rexford. SoftCell: Scalable and Flexible Cellular Core Network Architecture. In Proceedings of ACM CoNEXT 2013.
[12] Gudipati, D. Perry, L. E. Li, S. Katti. SoftRAN: Software Defined Radio Access Network. In Proceedings of ACM HotSDN 2013.
[13] M. Bansal, J. Mehlman, S. Katti, P. Levis. OpenRadio: A Programmable Wireless Dataplane. In Proceedings of ACM HotSDN 2012.
[14] K. Yap, et al. Blueprint for Introducing Innovation into Wireless Mobile Networks. In Proceedings of ACM VISA 2010.
[15] Floodlight:Open SDN Controller. http://www.projectfloodlight.org.
[16] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. Implementing a distributed firewall. In Proceedings of CCS, 2000.
[17] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte. Real time network policy checking using header space analysis. In Proceedings of NSDI, 2013.
[18] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey. Veriflow: verifying network-wide invariants in real time. In Proceedings of NSDI, 2013.
[19] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker. Composing software-defined networks. In Proceedings of NSDI, 2013.
[20] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu. A security enforcement kernel for openflow networks. In Proceedings of HotSDN, 2012.
[21] S. Shirali-Shahreza and Y. Ganjali. Flexam: Flexible sampling extension for monitoring and security applications in openflow. In Proceedings of HotSDN, 2013.
[22] H. Song. Protocol oblivious forwarding: Unleash the power of sdn through a future-proof forwarding plane. In Proceedings of HotSDN, 2013.
[23] L. Yuan, H. Chen, J. Mai, C. Chuah, Z. Su, P. Mohapatra, and C. Davis. Fireman: A toolkit for firewall modeling and analysis. In Proceedings of IEEE S&P, 2006.