SECURITY IN CLOUD USING CIPHERTEXT POLICY BASED ACCESS IN SDN

P.Nandini1,R.oviya2,P.veeralakshmi3

[1] Department of Information Technology, Prince Shri Venkateshwara Padmavathy Engineering College,Chennai,India

[2] Department of Information Technology, Prince Shri Venkateshwara Padmavathy Engineering College,Chennai,India

[3]Assistant professor,Department of Information Technology, Prince Shri Venkateshwara Padmavathy Engineering College,Chennai,India.

ABSTRACT

The software-defined networks (SDN) and OpenFlow development made easier to manage policy-driven network management. Before this, policy management are handled through the manual configuration where the author has to send code to solve the problems. The devices has made the things complexity and takes more time to transfer messages. They used low level language to configure the system. Manual configuration handled by network controller is being replaced by an automated approach of all network devices by SDN. We propose OpenSec, an OpenFlow-based security framework that allows a network security operator to implement security policies written in human-readable language. Using OpenSec, the openflow controller converts security policies into a series of openflow messages and automatically react when malicious traffic is detected. OpenSec allows for automated reaction to security alerts based on pre-defined network policies. The user data are encrypted using RC4 algorithm and stored in public cloud named CloudMe.

Keywords—*Software-defined networking, OpenFlow, network security, policy-based network management, policy specification.*

1.INTRODUCTION

With the advent of software-defined networks (SDN) [2], efforts to automate and simplify network operation have become popular. Software-defined networking (SDN) is an approach to computer networking that allows network administrators to programmatically initialize, control, change, and manage network behavior dynamically via open interfaces [1] and abstraction of lower-level functionality. SDN is meant to address the fact that the static architecture of traditional networks doesn't support the dynamic, scalable computing and storage needs of more modern computing environments such as data centers. This is done by decoupling or disassociating the system that makes decisions about where traffic is sent (the SDN controller, or control plane) from the underlying systems that forward traffic to the selected destination that is the data plane. In SDN, the complexity of the network shifts towards the controller and brings simplicity and abstraction to the network operator. As we move away from manual configuration at each device ,we get closer to automated implementation of network policies and rules. SDN decouples the control plane from the data plane and migrates the former to a logically centralized software-based network controller .OpenFlow (OF) is considered one of the first software-defined networking(SDN) standards. It originally defined the communication protocol in SDN environments that enables the SDN Controller to directly interact with the forwarding plane of network devices such as switches and routers, both physical and virtual (hypervisor-based), so it can better adapt to changing business requirements.

We propose OpenSec, an OpenFlow-based network security framework that allows campus operators to implement security policies across the network. Suppose a campus operator needs to mirror incoming web traffic to an intrusion detection system (IDS) and e-mail traffic to a spy- ware detection device. Our goal is to leverage SDN to allow the operator to write a high-level policy to achieve this, instead of having to manually configure each device. Furthermore, suppose the IDS detects malicious traffic and the sender needs to be blocked from accessing the network. Instead of having the operator configure the edge router to manually disable access to the source, we are interested in blocking the sender automatically. OpenSec provides an abstraction of the network, that the operators can focus on specifying human-readable security policies, instead of on configuring all the network devices to achieve the desired security. OpenSec consists of a software layer running on the top of the network controller, as well as multiple external devices that perform security services (such as firewall, intrusion detection system (IDS), encryption, spam detection, deep packet inspection (DPI) and others) and report to the controller. The main goal of OpenSec is to allow network operators to describe security policies for specific flows. The policies include a description of the flow, a list of security services that can apply to the flow and how to react in case malicious content that is found.

Three design requirements that are taken consideration. First, policies should be human-readable. Simplicity is one of the main goals of our framework and although current work has focused on creating human-readable policies. Second, the data plane traffic should be processed by the processing units such as the network devices, middle boxes or any other hardware that provides security services to the network. Third, the framework should react to security alerts automatically to reduce human intervention when suspicious traffic is detected. Next, we focused on evaluation of four metrics. First, the time needed by OpenSec to implement security policies based on the number of switches, processing units and existing policies in the system. Second, we measured the delay needed by the framework to react to alerts raised by processing units. Third, the benefits of automated blocking. Fourth, the trade-off of moving middle- boxes away from the data path and mirroring traffic as opposed to doing in-line processing.

2. RELATED WORK

2.1. Policy-Based Management Without SDN

A significant amount of work has focused on policy specification, policy refinement, conflict detection, and policy analysis in networks. Policy-based management (PBM) has also been applied to network management and security. Agrawal et al. provide an overview of how policy- based management can be applied to networked systems. In particular, they explain how Policy Management for Autonomic Computing (PMAC) can be applied to network management. In a nutshell, PMAC is a generic policy middleware supports extensive and flexible policy languages. Also, a Policy Definition Tool (PDT) should be provided to allow users to create and modify policies. Finally, an automated manager is responsible for collecting policies and implementing them. Rubio-Loyola et al. [15] propose a method to refine policies in policy-based management systems. Policy refinement allows to derive low-level enforceable policies from high-level guidelines. The authors provide a list of steps needed to convert high-level goals into low-level policies and describe a framework that supports all the required steps. Charalambides et al. [16] address conflict resolution in PBM, a crucial aspect when managing a system using policies. Indeed, as the authors point out, when several policies coexist it is likely to encounter that two or more policies give a different output for the same input. This study addresses the problem of conflict resolution when using policies to provide Quality of Service (QoS). OpenSec is similar to these methods in that it proposes a centralized system capable of receiving policies as input and analyzing them, checking for conflicts and implementing them. In this work, however, we provide a detailed explanation of how policies can be converted into OpenFlow messages to update the forwarding rules dynamically. Also, the policies used in OpenSec are low-level specifications because they already include a list of OpenFlow matching fields that should be used. Thus, the main contribution of OpenSec is the auto- mated administration of processing units and dynamic reaction to security alerts using SDN, as opposed to deriving low-level policies from high-level goals.

2.1.Policy-Based Network Management Using SDN

With the advent of SDN, the field of network management has evolved to become more dynamic [21]–[23]. Casado et al. proposed Ethane [4]. In Ethane, an operator creates a pol- icy using the Flow-based Security Language (FSL) to create a high-level access control list. Ethane allows an operator to write an access control policy with good granularity while still using high-level language (for example, using “testing nodes” instead of a subnet mask). Although OpenSec does not focus on referring to network objects by name, we do provide a broader set of security services besides access control. Also, OpenSec includes a reactive component to security alerts. When anomalous traffic is detected, OpenSec can modify traffic rules as specified by the policy, which adds a reactive component missing in Ethane. Foster et al. propose Frenetic [24], a programming language to program OpenFlow-based networks. Frenetic provides an interface to query traffic information. Frenetic can also be used to create a policy to react to network events. In our opinion, Frenetic focuses on simplifying how to program network events and how to retrieve traffic information. OpenSec focuses on hiding such complexity and allowing a security operator to work at a higher level, since it was designed to implement and enforce security policies, rather than to provide another mechanism to handle events sent by the network switches. Finally, Bari et al. [25] proposed Policy Cop, an autonomic QoS policy enforcement framework for SDN. This framework allows to specify service level agreements (SLAs) to implement and enforce QoS in an OpenFlow-based network. The step-by-step method used by Policy Cop to convert policies into flow rules is similar to that of OpenSec. However, our focus is on reacting to network security alerts instead of QoS violations.

3. MOTIVATION

The main goal of OpenSec is to be a human-friendly, dynamic and automated security framework. Next, we describe design requirements of our framework: reacting automatically to security events .

3.1. Reacting Automatically to Security Events

The second goal of OpenSec is to enable automated reaction to security events. When a middlebox detects suspicious traffic, it issues a security alert. Traditionally, these alerts are received by a network operator who then decides how to react to them. With OpenSec we aim at automating this reaction so that the framework either blocks the traffic, or simply alerts the operator of the detected malicious traffic. The reason why this is feasible is because, in general, an operator can plan ahead of time how critical a flow is based on the service provided through that flow. In a production network, for example, an operator would want a denial of service attack to be stopped as soon as possible. In contrast, a testing environment could be less critical. Thus, we designed OpenSec to allow the operator to specify a head of time what the automated reaction should be, so that in case of malicious traffic, the human participation is minimized.

TABLE I

SYNTAX TO CREATE POLICIES USING OPENSEC

	Value	Description
Flow	VLAN, macScr, macDst, macInPort, macOutPort, ipSrc, ipDst, 14Port	Uses OpenFlow match fields to describe a flow
Service	Encrypt, IDS, DPI, spam, DDoS	Identify a security service that should be applied to the flow
React	Notify-only, quarantine, block	Determine how to react if the service reports malicious content

4.OPENSEC COMPONENTS

OpenSec is an SDN framework capable of forwarding flows to security processing units based on policies and to automatically react to events raised by these middleboxes. Using this framework, security devices such as intrusion detection middle boxes, firewalls or encryption units can be removed from the main data path between the LAN and the Internet. OpenSec leverages a smart control plane to allowend-users to direct only part of the traffic to these security units.

System model

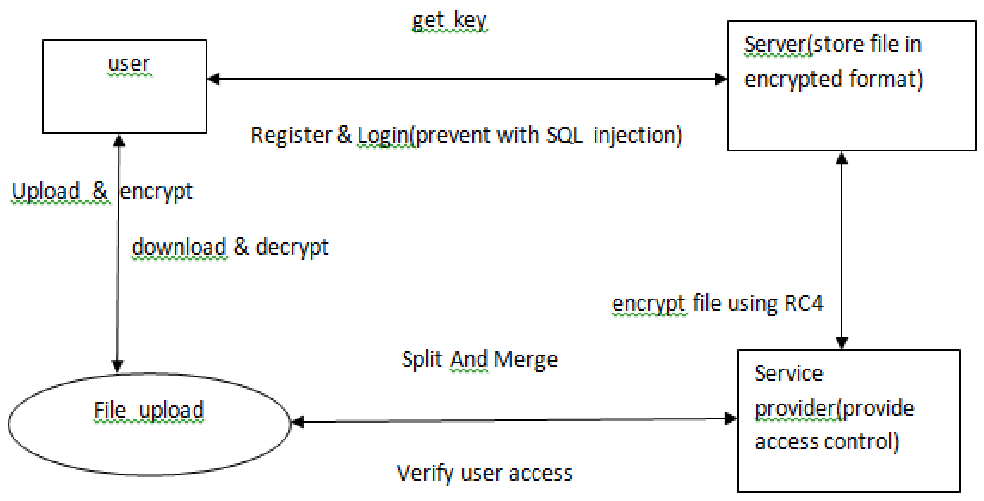
get key
user
Server(store file in encrypted format)
Register & Login(prevent with SQL injection)
Upload & encrypt
download & decrypt
encrypt file using RC4
Split And Merge
File upload
Service provider(provide access control)
Verify user access

Fig 4

4. 1 Identity Management

The identity management is divided into two roles: users and the security manager as follows. Users: Users can encrypt each key from his block and his own key. They can split files into blocks, encrypt them with the key, followed by signing the resulting encrypted blocks and creating the storage request. For each file, this key will be used to decrypt and rebuild the original file during the retrieval phase. The user also uses single sign-on to access each block with a compact signature scheme. Security Manager: Three roles are offered by the security manager. First, it can authenticate users during the storage/retrieval phase. Second, it can access control. Third, it can encrypt/decrypt data between users and their cloud.

4.2 Intrusion Detection and prevention

In this module automatic Intrusion detection system (IDS), encryption, deep packet inspection (DPI) and report the results to the controller. The main goal of OpenSec is to allow network operators to describe security policies for specific flows. The policies include a description of the flow, a list of security services that apply to the flow and how to react in case malicious content is found. The reaction can be to alert only, or to quarantine traffic or even block all packets from a specific source. Hence we have considered automatic intrusion detection and alerting network operator automatically when intruder tries brute force, SQL injection and wrapping attack.

4.2.1 Brute Force:

A password and cryptography attack that does not attempt to decrypt any information, but continue to try a list of different passwords, words, or letters. For example, a simple **brute-force attack** may have a dictionary of all words or commonly used passwords and cycle through those words until it gains access to the account. A more complex brute-force attack involves trying every key combination until the correct password is found. Due to the number of possible combinations of letters, numbers, and symbols, a brute force attack can take a long time to complete. The higher the type of encryption used (64-bit, 128-bit or 256-bit encryption), the longer it can take.

4.2.2 SQL Injection:

SQL Injection is one of the most widely exploited web application vulnerability of the web era. SQL Injection is used by hackers to steal data from online businesses' and organizations' websites. This web application vulnerability is typically found in web applications which do not validate the user's input. As a result, a malicious user can inject SQL statements through the website and into the database to have them executed. If a web application is vulnerable to SQL injection, a hacker is able to execute any malicious SQL query or command through the web application. This means he or she can retrieve all the data stored in the database such as customer information, credit card details, social security numbers and credential to access private areas of the portal, such as the administrator portal. By exploiting an SQL injection it is also possible to drop (delete) tables from the database. Therefore with an SQL Injection the malicious user has full access to the database.

4.2.3 Wrapping attack:

The attack uses a method known as XML signature wrapping and shows vulnerabilities while executing the web service request. In wrapping attack, the attacker tries to insert the malicious element in the SOAP (Simple Object Access Protocol) message structure in Transport Layer Service (TLS) and after inserting the malicious code, fake content of the message is copied into the server and while executing, cloud server working is interrupted by the attacker.

4.3 ENCRYPTION

In our system we proposed ciphertext-policy attribute-based encryption (CP-ABE) to address this problem, and give the first construction of such a scheme. In our system, a user's private key will be associated with an arbitrary number of attributes expressed as strings. On the other hand, when a party encrypts a message in our system, they specify an associated access structure over attributes. A user will only be able to decrypt a cipher-text if that user's attributes pass through the cipher-text's access structure.

4.4 KEY SEED MECHANISM

Time based Group Key Management algorithm for cryptographic cloud storage applications, which uses the proxy re-encryption algorithm to transfer major computing task of the group key management to the cloud server. So, the proposed TGKM scheme greatly reduces the user's computation and storage overhead and makes full use of cloud server to achieve an efficient group key management for the cryptographic cloud storage applications. Moreover, we introduce a key seed mechanism to generate a time-based dynamic group key which effectively strengthens the cloud data security. Our security analysis and performance evaluations both show that the proposed TGKM scheme is a secure and efficient group key management protocol for the cloud storage applications with low overheads of computation and communication.

4.5 SPLIT AND MERGE:

Cloud Storage usually contains business-critical data and processes, hence high security is the only solution to retain strong trust relationship between the cloud users and cloud service providers. Thus to overcome the security threats, this paper proposes multiple cloud storage. Thus the common forms of data storage such as files and databases of a specific user is split and stored in the various cloud storages(e.g.Cloud A and Cloud B). Databases consists of tables, rows and columns. Databases are easy to store in multiple cloud storages. Our application will act as a combiner and store different parts of the table such as rows and columns in multiple clouds using Vertical fragmentation and horizontal fragmentation. These rows and columns will be encrypted using RC4 (Stream Cipher) encryption algorithm. During response our application combines the data and sends to the verifier. Files are stored in multiple clouds using cryptographic data splitting. The file is split into fragments and stored in distinct cloud servers with encrypted key. Thus once the authorized token for the specific file is requested, searchable encryption allows keyword search on encrypted data and combines the fragments. This is sent to the verifier.

4.6 CLOUD STORAGE

Multiple cloud system include more than one cloud. We are storing the files in multiple cloud for secure storage purpose. The main advantage of storing in multiple cloud is for security and ease access. The user files are split and stored in multiple cloud servers namely Cloud A and Cloud B. The user can access the respective file by giving the appropriate key.

5. ALGORITHM

RC4 is a stream cipher, symmetric key algorithm. The same algorithm is used for both encryption and decryption as the data stream is simply XORed with the generated key sequence. The key stream is completely independent of the plaintext used. It uses a variable length key from 1 to 256 bit to initialize a 256-bit state table. The state table is used for subsequent generation of pseudo-random bits and then to generate a pseudo-random stream which is XORed with the plaintext to give the ciphertext

Algorithm 1. Parsing a file into a policy object

Data: new file path path

```
//Parse new policy from file Policy policy = new Policy( )
 lines = readFile(path)
 for each line f in lines do
       if line starts with ‘Flow’ then
                  //Create a match based on fields match = createMatch(line);
 end
 if line starts with ‘Service’ then
                //save codes of units (DPI, DoS,...) servicesCollection
               = getServices(line);
 end if line starts with ‘React’ then
             //Remember expected reaction
 reaction = getReaction(line);
 end
```

Algorithm 2. Implementing a policy

Data: Policy policy

```
//Implement policy in network
servicesCollection = policy.getServicesCollection();
for each service u in unitsCollection do
       //For each service code, find the unit
       Unit unit = unitManager.getUnit(service);
       //Get DPID, inPort and match to create a flow rule
       dpid = unit.getDPID( );
       inPort = unit.getInPort( );
       match = policy.getMatch( );
```

```
        //Get input port from existing rule
        inputPort = database.findInputPort(match);
        //Get next available VLAN tag
        vlanTag = database.getNextTagAvailable();
        //Update existing rule
writeFlowMod(matchon:inputPortandmatch,actions:
add vlan tag, output to port inPort);
end
 policy.setMatch(match);
 policy.setUnits(units);
 policy.setReaction(reaction);
policy.setVlan(getNextVLAN( ));
end
```

6. CONCLUSIONS

In this paper we presented OpenSec, an OpenFlow-based framework that allows network operators to describe security policies using human-readable language and to implement them across the network. OpenSec acts as a virtual layer between the user and the complexity of the OpenFlow controller and automatically converts security policies into a set of rules that are pushed into network devices. OpenSec also allows network operators to specify how to automatically react when malicious traffic is detected. Our evaluation shows several advantages of OpenSec. First, moving the analysis of traffic away from the controller and into the processing units makes our framework more scalable. Even when the load is high, the controller is not a bottleneck. Second, OpenSec is a first step towards moving the security controls away from the core of the network.

7. REFERENCES

[1] A. Lara and B. Ramamurthy, "OpenSec: A framework for implementing security policies using OpenFlow," in Proc. IEEE Globecom Conf., Austin, TX, USA, Dec. 2014, pp. 781–786.

[2] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using OpenFlow: A survey," IEEE Commun. Surveys Tuts., vol. 16, no. 1, pp. 493–512, Feb. 2014.

[3] A. Lara, A. Kolasani, and B. Ramamurthy, "Simplifying network man- agement using software defined networking and OpenFlow," in Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS), Bangalore, India, Dec. 2012, pp. 24–29.

[4] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," SIGCOMM Comput. Commun. Rev., vol. 37, no. 4, pp. 1–12, Oct. 2007.

[5]H.KimandN.Feamster,"Improvingnetwork management with software defined networking," IEEE Commun. Mag., vol. 51, no. 2, pp. 114–119, Feb. 2013.

[6] N. McKeown et al., "OpenFlow: Enabling innovation in campus net- works," SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Mar. 2008.

[7] A. Voellmy, H. Kim, and N. Feamster, "Procera: A language for high- level reactive network control," in Proc. Workshop Hot Topics Softw. Defined Netw. (HotSDN), Helsinki, Finland, Aug. 2012, pp. 43–48.

[8] S. Shin and G. Gu, "CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security mon- itoring as a service in clouds?)," in Proc. 20th IEEE Int. Conf. Netw. Protocols (ICNP), Austin, TX, USA, Oct. 2012, pp. 1–6.

[9] R. Hofstede, L. Hendriks, A. Sperotto, and A. Pras, "SSH compromise detection using NetFlow/IPFIX," ACM SIGCOMM Comput. Commun. Rev., vol. 44, no. 5, pp. 20–26, 2014.

[10] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, "FRESCO: Modular composable security services for software defined networks," in Proc. Netw. Distrib. Syst. Sec. Symp. (NDSS), San Diego, CA, USA, Feb. 2013, pp. 1–16.

[11] A. K. Bandara, E. C. Lupu, and A. Russo, "Using event calculus to formalise policy specification and analysis," in Proc. IEEE Workshop Policies Distrib. Syst. Netw., Lake Como, Italy, Jun. 2003, pp. 26–39.

[12] A. K. Bandara, E. C. Lupu, J. Moffett, and A. Russo, "A goal-based approach to policy refinement," in Proc. IEEE Workshop Policies Distrib. Syst. Netw., Yorktown Heights, NY, USA, Jun. 2004, pp. 229–239.

[13] A. K. Bandara, A. Kakas, E. C. Lupu, and A. Russo, "Using argumenta- tion logic for firewall policy specification and analysis," in Large Scale Management of Distributed Systems. New York, NY, USA: Springer, 2006, pp. 185–196.

[14] N. Foster et al., "Frenetic: A network programming language," in Proc. ACM SIGPLAN Int. Conf. Funct. Program., Tokyo, Japan, Sep. 2011, pp. 279–291.

[15] M. Bari, S. Chowdhury, R. Ahmed, and R. Boutaba, "PolicyCop: An autonomic QoS policy enforcement framework for software defined net- works,"in Proc.IEEESDNFutureNetw.Serv.(SDN4FNS),Trento,Italy, Nov. 2013, pp. 1–7.

[16]V.Sekar,S.Ratnasamy,M.K.Reiter,N.Egi,andG.Shi,"Themiddlebox manifesto: Enabling innovation in middlebox deployment," in Proc. 10th ACM Workshop Hot Topics Netw., 2011, pp. 21 :1–21:6.

[17] Big Switch Networks. (2016). Floodlight [Online]. Available: www.projectfloodlight.org/floodlight/

[18] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," SIGCOMM Comput. Commun. Rev., vol. 43, no. 4, pp. 7–12, Aug. 2013.

[19] A.TootoonchianandY.Ganjali,"HyperFlow: Adistributedcontrolplane for OpenFlow," in Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw., San Jose, CA, USA, Apr. 2010, p. 3.

[20] N. Farrington et al., "Helios: A hybrid electrical/optical switch archi- tecture for modular data centers," SIGCOMM Comput. Commun. Rev., vol. 41, no. 4, pp. 339–350, Aug. 2010.

[21] F. Yonghong, B. Jun, W. Jianping, C. Ze, W. Ke, and L. Min, “A dor- mant multi-controller model for software defined networking,” Commun. China, vol. 11, no. 3, pp. 45–55, Mar. 2014.

[22] R. Sherwood et al., “Carving research slices out of your production net- works with OpenFlow,” SIGCOMM Comput. Commun. Rev., vol. 40, no. 1, pp. 129–130, Jan. 2010.