International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 01 | Jan 2020 www.irjet.net p-ISSN: 2395-0072

Collaborative Network Security in Data Center for Cloud Computing

Yashveer singh[1], Anurag Chandna[2], Sagar Choudhary[3]

[1,2,3]*Department of Computer Science and Engineering, Roorkee College of Engineering, Roorkee*

Abstract - Data centers are simply centralized locations where computing and networking equipment is concentrated for the purpose of collecting, storing, processing, distributing or allowing access to large amounts of data. They have existed in one form or another since the advent of computers.

In the days of the room-sized behemoths that were our early computers, a data center might have had one supercomputer. As equipment got smaller and cheaper, and data processing needs began to increase -- and they have increased exponentially -- we started networking multiple servers (the industrial counterparts to our home computers) together to increase processing power. We connect them to communication networks so that people can access them, or the information on them, remotely. Large numbers of these clustered servers and related equipment can be housed in a room, an entire building or groups of buildings. Today's data center is likely to have thousands of very powerful and very small servers running 24/7.

Key words: *data center network; network security; software defined network; collaborative network security; multi-tenant; network virtualization; intelligent flow processing; cloud computing*

1. INTRODUCTION

A cloud data center is an infrastructure that supports internet services. A cloud data center may be defined from a variety of perspectives, and the most popular ones are categorized by IaaS, PaaS, and SaaS proposed by the NIST[1] and include public cloud, private cloud, hybrid cloud, and other different categories. Other categories include computing, networking, storage from a system's perspective or using (in use), archiving (at rest), and transmission (in motion) from a data perspective. Specific to the cloud network, there are different characteristics of a cloud (remote access), within a cloud, and between cloud networks. The debut of VMware NSX provides the virtualization of networks with Software-Defined Network (SDN) inside a data center. The Google B4 network [2] also uses an Open Flow-based SDN[3, 4] to implement all the interconnections among cloud data centers in different locations.

The challenge posed by SDN is the dynamic characteristic of network boundaries that is the "twin" of the network topology with expanded flexibility provided by virtualization. In other words, the original static, natural, and physical boundaries within the traditional network are replaced by the dynamic and virtual logical boundaries of SDN.

2. VCNSMS

In this section, we propose vCNSMS, a collaborative network management prototype system derived from CNSMS[11-15] for multi-tenant data center networks. In our experiments, we demonstrate that vCNSMS can be integrated into virtualized cloud environments. vCNSMS is a prototype system with a home-brewed version of an untangle open source multifunction gateway.

Ofdatapath is the implementation of a datapath. It is responsible for forwarding packets based on the flow table and communicates through of protocol and floodlight.

Ofprotocol is a middle program that controls ofdatapath and is controlled by floodlight.

3. Results

In Open flow 1.0, there is only a single user space data path. The dpct that played an important role in previous versions is replaced by of data path and of protocol.

VMs	IP	Hostname	Host alias	System
Controller	192.168.0.60	Saturn-controller	Control	Ubuntu 13.10 (flood light)
OpenFlow1	192.68.0.61	Saturn-openFlow1	Of1	Ubuntu 13.04
OpenFlow2	192.68.0.62	Saturn-openFlow2	Of2	Ubuntu 13.04
OpenFlow3	192.68.0.63	Saturn-openFlow3	Of3	Ubuntu 13.04

Client1	192.68.0.64	Client1	Client1	Ubuntu 13.04
Client2	192.68.0.65	Client2	Client2	Ubuntu 13.04

Table 1: Detailed configuration settings in virtualization Network based on SDN.

Ofdatapath is the implementation of a data path. It is responsible for forwarding packets based on the flow table and communicates through of protocol and floodlight.

Of protocol is a middle program that controls of data path and is controlled by floodlight.

dpctl can still be used to view the flow table in the Open flow switch, but cannot be used to manage the data path.

To change the behavior of the Open flow switch, the component (C or python) must pass through floodlight, and the component will be loaded with floodlight. (Specific usage can be referred to from the main page for of data path, of protocol, and dpctl).

4. Deep security check in vCNSMS

4.1.1 Function settings

1. The Security Center, centrally manages security rules, collects the feedback information from the rule deployment, and stores the data into the security log.

2. Security rules are incrementally downloaded. In the Security Center, duplicate rules are removed from the new rule set and make the new rules available.

3. Firewall module. Firewall rules will be downloaded from the Security Center and loaded in the Firewall module, and the corresponding functions will be activated. Security events are collected and returned back to the Security Center.

4. UDP content filtering module. It blocks or drops the specified types of data packets under the current rules. The format for UDP rules and content matching is presented in Appendix A.

5. To change the behavior of the Openflow switch, the component (C or python) must pass through floodlight, and the component will be loaded with floodlight. (Specific usage can be referred to from the main page for ofdatapath, ofprotocol, and dpctl).

4.1.2 Enhanced security functions

1. Protocol Control module in the prototype system mainly enforces UDP protocol rules.

2. UTMs routinely update rules. UTMs regularly obtain Firewall rules. The configuration information includes the rules of content inspection for UDP and the blacklist based on the content filtering (source and destination addresses, ports, protocol, URL).

3. Filtering based on the content of a UDP packet. According to the rules, any packets that contain specified signatures will be matched. For example, the DNS requests' packets that contains the specified text string "abc" will be matched and logged.

4. Blocking function with a blacklist. Block rule based on "quintuple" filtering, e.g., block the network connection with a command such as "TCP 1.2.3.4:443". The Firewall module can quickly apply the update rule.

4.1.3 Implementation in peer-UTM

1. **Update Firewall rules** The administrator puts a new Firewall rule file in the specified folder named FirewallRule in the Security Center. The daemon program periodically checks this folder, and if there is a new set of rules in the folder, the Security Center will send announcement messages to all online peer-UTMs to inform them of a new rule update. A peer-UTM receives messages from the Security Center and compares these with its own rules' serial number in the rule database. If there is an update, the peer-UTM will send a download request to the Security Center for the updated rules. The corresponding rules are then downloaded to the local rule database and are activated in real time.

2. Firewall module blocks a specified network flow. The Firewall module provides a blocking function of rules based on "quintuple" filtering, and sends the corresponding log info to the Security Center through an announcement message too. After receiving the message, the web UI of Security Center will present this information in real-time.

3. UDP content filtering. UDP content filtering module uses the same mechanism as the Firewall module, and has the specified tag indicated in the announcement message. When a peer-UTM receives the announcement from the Security Center, it will send a download request message to the Security Center for the rules. When the rules are downloaded and activated, the peer-UTM will call the getPatterns function in the LoadPatterns class to activate the rules loaded in UDP content filtering module. (We modify the prototype system to load multiple UDP rules.) When the UDP packet is parsed, the findMatch() function in the EventHandler class will be called to match the content and rules. The create RegExPattern function in the PatternFactory class is called to process the content of the UDP packet. The JAVA regular expression matching and matcher functions in the java.util.regex class are then used to match the rules and filter out any specified content. Finally, the UDP packets that contain specified content will be dropped.

4.2 Intelligent flow processing in vCNSMS

Intelligent flow processing is an advanced method in Refs. [18-20] for intrusion detection. Intelligent flow processing in vCNSMS is based on the smart packet verdict scheme. In this section, we propose the security level based protection policy for intelligent flow processing in vCNSMS.

4.2.1 Security level based protection policy

The security level based protection policy is shown as follows:

(1) Security level: Red, Yellow, Orange, and Green.

(2) Security level: Red, Yellow, Orange, and Green.

(3) Intelligence flow processing: According to the security level, different security rule sets and packet verdict schemes are used with different performances and load costs.

(4) Multi-function security gateway: UTM can configure different security plugins on the demand of different security levels, and incurs different processing costs. The working mode of peer-UTMs can also be categorized into Red, Yellow, Orange, and Green configurations.

4.2.2 Implementation of security level with security function plugins

We modify the untangle Shield module[16, 21] and enhance it with the smart packet verdict algorithm. The detailed functions are as follows.

(1) Security plugins' rule set: S1 (urgent), S2 (important), S3 (less important), and S4 (trivial).

Security plugin module: plugin 1, plugin 2, plugin 3, , plugin N.

(2) Security plugin module's processing: Tagging each packet with a Block mark, Pass mark, or Suspect mark.

(3) Packet verdict on network packets is based on a scoring system.

(4) Packet processing obeys the following strategy: Let the benign flows pass as soon as possible;

Recheck the suspected flow with additional rule sets or security plugins; provide a verdict on the packet with the context, i.e., current level of security, such as Red or Green.

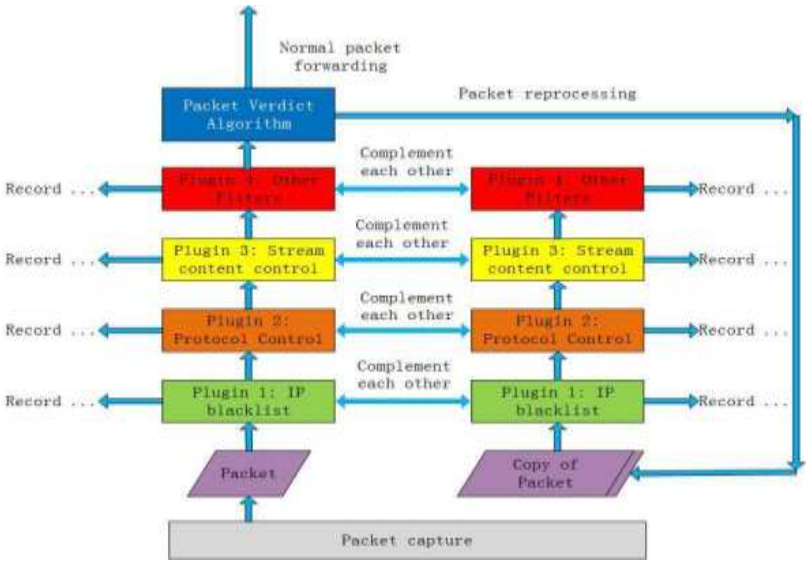
(5)Packet verdict processing also takes traffic load and performance penalties into consideration.

4.2.3 Smart packet verdict scheme with untangle Shield

Originally, Shield is an untangle module whose function is to prevent DoS attacks. Shield reads a packet from nfqueue and starts four threads including a main thread, an event dispatch thread, an admission packet processing thread, and a frontend microhttpd daemon. The main task of Shield is to collect packet processing threads, and there is no parallel structure in Shield. Figure 12 shows the smart packet verdict scheme used in vCNSMS.

4.2.4 Process level parallelization of Shield

We separate the untangle Shield module and enhance it with our smart packet verdict scheme. For performance improvement, we also modify Shield to integrate xtables[22] for parallelization. The Shield test environment is shown in Fig.



4.3 The parallelization scheme of Shield may be accomplished in two ways:

(1) Use multiple NFQUEUEs and open multiple Shield processes that perform the same as the Snort inline program[23].

(2)Rewrite the Shield code to implement a dispatcher for the corresponding queue in the program to achieve multi-threaded processing.

4.3 The detailed parallelization scheme is given as follows:

Install xtables-addons-1.12-nslab that is an NFQUEUEX module with the shunt function.

(1) Specify iptables -A FORWARD -j NFQUEUEX –queue-num 4. The last parameter establishes four NFQUEUEs in this example.

(2) Start four Shield processes, with different queue numbers and port numbers assigned to each.

To achieve the forward operation, it needs not only to set routing table of client1 and client2, but also to run the command (echo1>/proc/sys/net/ipv4/ip forward) to enable the kernel forwarding function. In this section, we propose the security level based protection policy for intelligent flow processing in vCNSMS.

4.3.1 Performance of Shield parallelization

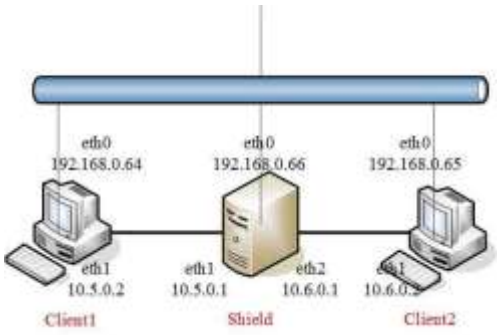
Because of the amount of traffic is throttled from a single IP, UDP packets sent by traffic generator SmartBit will lead the original Shield to block these packets. To avoid this, we modify Shield to make all packets pass through. The performance measured in this manner should reflect the capability of Shield module.

The test environment is described as follows:

(1) Hardware: Intel(R) Core(TM) 2 Quad CPU Q9400 @ 2.66 GHz, 4 cores, 2 GB memory, 8 Gbit ports.

(2) Software: Ubuntu 13.04, Shield untangle module.

vCNSMS is based on the smart packet verdict scheme.



(3) SmartBit settings: 4-way UDP streams with different IP and Ports. Packet loss ratio is collected in each experiment. Each experiment is conducted in a 1-Gbps link with varied input traffic ratio from 10% to 100%.

Five parallel Shield processes can handle 400-500 Mbps. The CPU usage is about 40%. Memory consumption is quite low. As Shield only deals with the headers of packets in this test, the performance result is expectable, but it is slower than the snort.

4.3.2 Bottleneck analysis of the shield

We use Intel VTune tools in the compile stage of the shield module. Running performance evaluation experiments of the shield module process useful information of the function calls inside the shield module. The most time consuming operation in packet handling are concentrated in bar fight net nfqueue read() and handle packet(), wherein the packet verdict scheme is running. In addition, the debug function is time consuming.

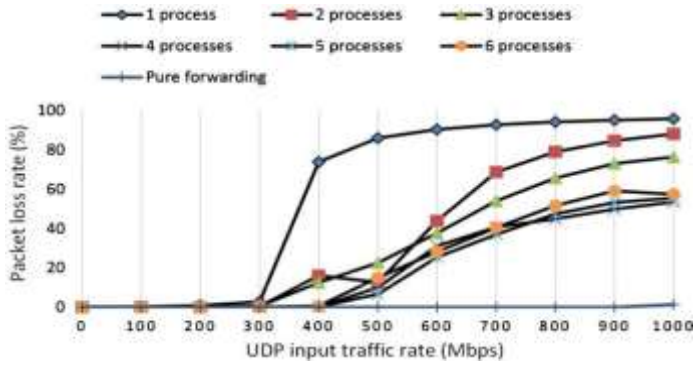
As we run the Shield module in a virtual machine, performance issue in virtualized environments is a critical and open problem, which has already been discussed in Ref. [25]. We can further improve performance with vPF ring[26] in virtualized environments.

Evaluation experiments of the Shield module produces useful information of the function calls inside the Shield module. The most time consuming operations in packet handling are concentrated in barfight net nfqueue read() and handle packet(), wherein the packet verdict scheme is running. In addition, the debug function is time consuming.

As we run the Shield module in a virtual machine, performance issue in virtualized environments is a critical and open problem, which has already been discussed in Ref. [25]. We can further improve performance with vPF ring[26] in virtualized environments.

5. Conclusions

In this paper, we propose vCNSMS to address network security in multiple-tenants data center and demonstrate vCNSMS with a centralized collaborative scheme. vCNSMS can further integrate a smart packet verdict scheme for packet inspection to defend from possible network attacks inside the data center network. An SDN-based virtualization network in a data center can deploy vCNSMS for flexibility and scalability to protect multiple tenants with different network policies and security requirements. The smart packet verdict scheme can be further investigated and improved with parallelization.



6. References

[1] NIST definition of cloud computing, http:// csrc.nist.gov/publications/PubsNISTIRs.html, 2007.

U. S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, Hozle, S. Stuart, and A. Vahdat, B4: Experience with a globally-deployed software defined WAN, in Proc. ACM SIGCOMM 2013 Conference on SIGCOMM, Hong Kong, China, 2013, pp. 3-14.

[2] J.D. Liu, A. Panda, A. Singla, B. Godfrey, M. Schapira, and S. Shenker, Ensuring connectivity via data plane mechanisms, presented at 10th USENIX Symposium on Networked Systems Design and Implementation, Lombard, IL, USA, 2013.

[3] J. D. Liu, B. H. Yan, S. Shenker, and M. Schapira, Data-driven network connectivity, in Proc.10th ACM Workshop on Hot Topics in Networks, New York, USA, 2011, p. 8.

[4] Qihoo 360 Internet Security Center, Development trend of enterprise security in the internet ages, http:// www.gartner.com/technology/mediaproducts/pdfindex.jsp? g=Qihoo issue1, 2013.

[5] X. M. Chen, B. P. Mu, and C. Zhen, NetSecu: A collaborative network security platform for in-network security, in Proc. 3rd International Conference on Communications and Mobile Computing, Qingdao, China, 2011, pp. 59-64.

[6] D. H. Ruan, C. Lin, Z. Chen, and J. Ni, Handling high speed traffic measurement using network processors, presented at International Conference on Communication Technology, Guilin, China, 2006.

[7] J. Ni, C. Lin, and Z. Chen, A fast multi-pattern matching algorithm for deep packet inspection on a network processor, presented at the IEEE International Conference on Parallel Processing, Xi'an, China, 2007.

[8] Z. Chen, C. Lin, J. Ni, D.H. Ruan, B. Zheng, Y. X. Jiang, H. Peng, Y. Wang, A. A. Luo, B. Zhu, Y. Yue, Y. Ren, AntiWorm NPU-based parallel bloom filters for TCP/IP content processing in giga-Ethernet LAN, in Proc. the IEEE International Conference on Communications, 2006, pp. 2118-2123 and F.

[10] Bro, http://www.bro-ids.org, 2013.

[11] F. Han, Z. Chen, H. Xu, H. Wang, and Y. Liang, A collaborative botnets suppression system based on overlay network, International Journal of Security and Networks, vol. 7, no. 4, pp. 211-219, 2012.

[12] Z. Chen, F. Han, J. Cao, X. Jiang, and S. Chen, Cloud computing-based forensic analysis for collaborative network security management system, Tsinghua Science and Technology, vol. 18, no. 1, pp. 40-50, 2013.

[13] X. Chen, K. Ge, Z. Chen, and J. Li, AC-Suffix-Tree: Buffer free string matching on out-of-sequence packets, in Proc. 2011 ACM/IEEE Seventh Symposium on Architectures for Networking and Communications Systems, IEEE Computer Society, Brooklyn, NY, USA, 2011, pp. 36-44.

[14] T. Li, F. Han, S. Ding, and Z. Chen, LARX: Large-scale antiphishing by retrospective data-exploring based on a cloud computing platform, in Proc. IEEE 20th International Conference on Computer Communications and Networks, Maui, HI, USA, 2011, pp. 1-5.

[15] B. Mu, X. Chen, and Z. Chen, A collaborative network security management system in metropolitan area network, in Proc. IEEE 3rd International Conference on Communications and Mobile Computing, Qingdao, China, 2011, pp. 45-50.

[16] Untangle open source appliance, https://gitorious.org/ untangle, 2013.

[17] Y. D. Lin, R. H. Hwang, and F. Baker, Computer Networks: An Open Source Approach. McGraw-Hill, February 2011.

[18] Y. D. Lin, H. Y. Wei, and S. T. Yu, Building an integrated security gateway: Mechanisms, performance evaluations, implementations, and research issues, IEEE Communications Surveys & Tutorials, vol. 4, no. 1, pp. 2-15, 2002.

[19] Y. D. Lin, C. W. Jan, P. C. Lin, and Y. C. Lai, Designing an integrated architecture for network content security gateways, Computer, vol. 39, no. 11, pp. 66-72, 2006.

[20] C. N. Lu, C. Y. Huang, Y. D. Lin, and Y. C. Lai, Session level flow classification by packet size distribution and session grouping, Computer Networks, vol. 56, no. 1, pp. 260-272, 2012.

[21] D. Morris, J. Irwin, and R. Scott, Methods and systems for reputation based resource allocation for networking, US Patents US20070043738A1, February 22, 2007.

[22] Xtables-addons, http://xtables-addons.sourceforge.net, 2013.

[23] Snort-inline, http://snort-inline.sourceforge.net, 2009.

[24] Snort, http://www.snort.org, 2010.

[25] M. J. Schultz and P. Crowley, Performance analysis of packet capture methods in a 10 Gbps virtualized environment, in Proc. IEEE 21st International Conference on Computer Communications and Networks, Munich, Germany, 2012, pp. 1-8.

[26] A. Cardigliano, L. Deri, J. Gasparakis, and F. Fusco, vPFRING: Towards wire-speed network monitoring using virtual machines, in Proc. ACM SIGCOMM Conference on Internet Measurement Conference, Berlin, Germany, 2011, pp. 533-548.

[27] Zhen Chen and Junwei Cao, TSINGHUA SCIENCE AND TECHNOLOGY I S S Nll1 0 07 - 0 2 14ll0 9/ 1 0llp p 8 2-9 4 Volume 19, Number 1, February 2014.