Collaborative Network Security in Multi-Tenant Data Center for Cloud Computing

Zhen Chen*, Wenyu Dong, Hang Li, Peng Zhang, Xinming Chen, and Junwei Cao*

Abstract: A data center is an infrastructure that supports Internet service. Cloud computing is rapidly changing the face of the Internet service infrastructure, enabling even small organizations to quickly build Web and mobile applications for millions of users by taking advantage of the scale and flexibility of shared physical infrastructures provided by cloud computing. In this scenario, multiple tenants save their data and applications in shared data centers, blurring the network boundaries between each tenant in the cloud. In addition, different tenants have different security requirements, while different security policies are necessary for different tenants. Network virtualization is used to meet a diverse set of tenant-specific requirements with the underlying physical network, enabling multi-tenant datacenters to automatically address a large and diverse set of tenants requirements. In this paper, we propose the system implementation of vCNSMS, a collaborative network security prototype system used in a multi-tenant data center. We demonstrate vCNSMS with a centralized collaborative scheme and deep packet inspection with an open source UTM system. A security level based protection policy is proposed for simplifying the security rule management for vCNSMS. Different security levels have different packet inspection schemes and are enforced with different security plugins. A smart packet verdict scheme is also integrated into vCNSMS for intelligence flow processing to protect from possible network attacks inside a data center network.

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 OpenFlow-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.

1.1 State-of-the-art network security in data center networks

Traditional security devices such as Firewalls, IDS, WAF, and other devices are deployed with the Middleboxes model inside and outside networks. With the development of cloud computing technology, the deployment of Middleboxes is facing new challenges in the large-scale data center network environment [5].

(1) In multi-tenant cases, the network boundaries are blurring.

With the increase of Internet users, the data center network topology becomes more complicated. Multiple tenants save their data in the same server, and the same tenant may store data on different servers with multiple hot backups, causing the network boundaries between each user to become blurred, as opposed to the set boundaries found in traditional physical isolation. The original static, natural, and physical boundaries within the network are replaced by dynamic and virtual logical boundaries. Hence, network security within the cloud will be more dependent on dynamic deployment, configuration, and management of security policies and security components, and more dependent on the network security system for flow and traffic awareness, decision-making, and response. Undoubtedly, this causes network management to become more complex. How to ensure network security is a challenge in such a complex network environment.

(2) The deployment location of Middleboxes also has new changes.

In traditional networks, the data of several hosts are from the same gateway, and the entire network may have several gateways. So long as security devices are deployed in network vantage points and ensure data through the gateway is safe and reliable. In the data center network, the physical vantage points have been replaced by virtual logical gateways. To protect the security of virtual logical boundaries between tenants, Firewall, IDS/IPS, and other devices are required to collaborate with the traffic controller to adapt to performance and safety requirements of each tenant or security domain. Provide security of dynamic boundaries caused by virtual machine migration, and meet the dynamic security requirements of virtual machines [6-10]. Therefore, to meet these requirements, how to properly deploy these security devices is also an important issue.

(3) Security requirements for different tenants are different.

Since different tenants have different security requirements, it is necessary to create different security policies for different tenants, and this is undoubtedly a major challenge for traditional security devices. The traditional approach is to set rules for the device where data passed through. Obviously, in a data center network, this strategy no longer applies, and now we have to address the question how to meet diverse security requirements when multiple tenants’ data pass through the same security device and how to provide effective enforcement.

(4) The migration of the virtual machine results in the switchover of security domain.

To meet the security needs of a single tenant, it needs to configure multiple security devices to control traffic and thus completely implement the tenant’s security policy. When service host migrates to other locations in the network, the topology of the network changes and appropriate security policies also migrate with the tenant host. The original security configuration of the security device will no longer work, the migration to the new security domain also requires tenants on that server to apply some configuration updates. Therefore, to protect the mapping of the security policies from the logical network to the physical network and maintain its correctness and consistency will be a major challenge to the control function of a data center network.

1.2 Software-defined network in data center network

To enable network virtualization, SDN is one of the supporting technologies to build cloud data center virtual networks. Cloud data center virtual networks need to ensure tenants’ security domains have complete and isolated network boundaries. This is not a simple network security technology, because the virtual network itself provides services for multiple tenants or basic virtual private cloud services that should be guaranteed in the implementation of the network. Due to the different virtual machines of different tenants
sharing the same physical resources, system security guarantees such as preventing the virtual machine from escaping its boundaries are also the issue of network virtualization security.

Comparing software-defined network security (e.g., OpenFlow based on SDN) with a traditional physical network, the changes are in the control mechanisms of switches, routers, and other forwarding devices. The forwarding devices worked according to a flow table issued by the controller, thereby more efficient and less costly. The controller also collects network status information, discovers network topologies, checks the network forwarding policies, and generates and releases new flow table according to status update.

Thus, handling of the first network packets header of a flow should not happen in the gateway, but in the controller. The traditional Access Control List (ACL) scheme or network packet filtering Firewall should be deployed in the controller. However, the controller usually only receives the first packet header and cannot perform an deep inspection of the network flows. The stateful Firewalls and deep packet inspection that filter packet payloads should be deployed in the data plane or in the data plane and control plane. Therefore, the network access control deployed on the physical network gateway should still be reasonable for a logical gateway.

Through Openflow protocols and controllers, data center network security based on SDN addresses the problems of Middleboxes positioning and separation of security rules by setting a flow table and guiding traffic that matches the corresponding policies to the proper security devices.

It is possible to optimize the controller by configuration and management, and update rules for dynamic migration and reconfiguration to solve security policy inconsistencies caused by Middlebox movement and virtual machine migration.

2 Related Work

2.1 Network security in data center networks

Among the security research in data center networks, SDN security is a hot topic both for academia and industry.

FRESCO\textsuperscript{[11-13]} introduces a new security application development framework. It is used to solve several key issues when implementing security service components on demand. OpenFlow is an open standard that is able to provide simplified and convenient design of complex network security applications and their integration into large networks. However, so far there remains a lack of convincing applications regarding the security aspects of OpenFlow. FRESCO provides a scripting language Application Programming Interface (API) that programmer can use to write security monitoring logic with a modular library of basic processing unit in FRESCO. A FRESCO module allows customization of stream processing rules to provide an effective response to detected networks threats.

Slick\textsuperscript{[14]} is a proposed programming framework that separates the controller and Middleboxes and provides an interface for communication. There are more and more programs running on controllers which cross data planes and control planes to call actions, and there is no complete solution to meet this requirement. While OpenFlow provides a programmable control plane, it has a relatively simple data plane. In contrast, Middleboxes are able to effectively extend the data plane and provide a complex data plane, but cannot provide effective integration with the control plane.

Slick extends data plane programmability in two aspects. First, Slick arranges complex data plane functions dynamically in the network, and also guides a subset of packet through the appropriate function processing queue that adapts to the position of the layout and response as these change over time to meet the changeable network conditions and transmission mode. Second, to achieve modularity, reusability, and integration strategy across multiple applications and network resources, Slick allows programmers to coordinate multiple actions among different entities in the data plane.

SIMPLE, based on SDN, is proposed in Refs. [15-18], and is an implementation of strategy layer that directs traffic to specific Middleboxes. Today’s networks must rely on Middleboxes to provide high throughput, high security, and efficient decision-processing capabilities. To achieve these goals, we need to ensure that traffic goes directly to the required Middleboxes queue, an operation that once requires extensive manual effort and operator’s expertise. In this respect, an SDN provides a promising option, but also introduces some functions that do not belong to the traditional L2/L3 functions, such as policy components, resource management, and packet manipulation.

SIMPLE allows network operators to specify routing policies in a logical Middlebox, and automatically
update forwarding rules according to the physical topologies, switching capacity, and resource constraints of Middleboxes to guide traffic to the proper Middleboxes queues. Under the premise of existing SDN functions and without modifying the existing implementation of Middleboxes, SIMPLE increases the flexibility of Middleboxes deployment effectively by flow oriented control, and also generates and loads new rules to maintain network stability when any Middleboxes fail and/or network transmission is overloaded.

2.2 Network virtualization with VMware NSX

VMware NSX\[^{[19]}\] is a network virtualization platform for data center networks. VMware NSX provides a network management and network security system for its software-defined data center that combines SDN and Network Functions Virtualization (NFV), and is designed to achieve the rapid deployment of multiple layers of a logical network without complicated configuration of physical devices that greatly improves the flexibility of network deployment. It provides 2-7 layers’ NFV and provides a software-based security model to achieve the decoupling of the underlying network hardware, thus making full use of the existing network infrastructure.

The core components of VMware NSX are logic switches, logical routers, NSX APIs, logical Firewalls, logical load balancing, logical VPN, etc. NSX provides a network virtualization approach that allows data center operators to treat the physical network as an on-demand system and capacity tool. The overall structure of NSX is shown in Fig. 1.

![Network virtualization platform of VMware NSX\[^{[19]}\].](image)

The scalability in NSX provides services of other network security vendors. The NSX platform uses a distributed service framework that allows multiple hosts to integrate the network service, and can easily insert new service modules using the NSX API. NSX offers a variety of logical device interfaces, but it also needs cooperation from other vendors to achieve correct mapping of logical-to-physical connections when faced with a wide variety of hardware devices.

2.3 NSX and network security

NSX aims to integrate security functions offered by other vendors. To simplify the third-party security products and integrate services, authorized vendors use a single specific API that links the security platform and the network. NSX’s Service Composer tool is used to deploy third-party Firewalls, anti-malwares, vulnerability management, data loss protection, intrusion detection, and intrusion prevention (IDS/IPS) platforms centrally.

In addition to the security measures discussed above, the network service gateways bridge the physical and virtual environments, and application delivery services include load balancing, application delivery, and WAN optimization.

There are internal stateful Firewalls in the NSX that can provide distributed Firewall detection for each virtual router port. Firewall management uses a variety of rules, audit, and inspection technologies. It also provides the logic state, and integrates routing into a third-party security platform.

Brocade VCS Gateway allows the connection of virtual and physical facilities while providing the types of applications required by a data center operator to maintain a unified network. By using Brocade VCS Fabric technology, one can use their current infrastructure to implement network management deployment. The implementation structure of Brocade VCS Gateway is shown in Fig. 2.

Cumulus Linux system provides programmable features and automated tools that use the same computing environment on network devices to quickly configure the physical network and increase cost-efficiency when adding new network capacity by reducing the difficulty of adding new devices. It provides support for network virtualization boundary functions by realizing coverage of the second layer gateway and avoiding use of the virtual network of the VXLAN tunnel endpoint, and registers NSX
gateway services to further simplify management by the controller. Combined with NSX, Cumulus Linux provides a network virtualization boundary to achieve physical workload at high levels of connectivity. The integrated solution of Cumulus NSX is shown in Fig. 3.

Dell S6000 data center switching gateway for NSX provides programmability, automation features, and scalability. S6000 is a high-performance gateway for NSX that extends a virtual network to the physical server, or connects the physical workload that is accessed by virtual LANs to the logical network via the second layer network services, also provides the migration from an existing virtual environment to the public cloud, creates a hybrid cloud, etc. The specific structure of the S6000 data center switching gateway for NSX is shown in Fig. 4.

In addition, for the security of the data center, Palo Alto presents its next generation security platform\[20, 21\]. As shown in Fig. 5, its core component is a powerful Firewall that provides identification and control functions, and is always able to detect threats while ensuring the safe operation of the program. Palo Alto PA-5000 offers a variety of defense options to network threats, such as IPS and anti-malware software to defeat known threats, and automated sandbox analysis of suspicious files to detect unknown malicious programs or APT attacks.

3 vCNSMS

In this section, we propose vCNSMS, a collaborative network management prototype system derived from CNSMS\[22-26\] 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\[27, 28\].

3.1 The principle of collaborative network security in DCN

3.1.1 Basic network topology

The Security Center and peer-UTM are deployed in the data center network. In the bootstrap stages, the peer-UTM is running a registration process for the Security Center.
The Security Center receives the registration information and displays the registered UTMs. Figure 6 shows the vCNSMS configuration and the registration process in the bootstrap stage.

### 3.1.2 Collaborative security in DCN

#### 3.1.2.1 Security center interacts with the peer-UTMs

We assume each peer-UTM manages a virtual domain for a tenant in the Data Center, and the peer-UTM is managed by commands from the Security Center. Figures 7 and 8 show the collaborative network security process in vCNSMS prototype.

1. Security Center issues rules: There is a help option for rule creation and an option for the rule issued, in web User Interface (UI).

2. Peer-UTM reports events: The Security Center needs a web interface to display the security events reported by peer-UTMs.

3. Events inform between peer-UTMs: There is a web-based UI in the peer-UTM collaborative security module. The web UI shows the different peer-UTMs to the Security Center and displays any occurred security events.

#### 3.1.2.2 Antivirus module in collaborative security

Workflow of Antivirus module is shown in Fig. 9.

---

**Fig. 6** vCNSMS configuration settings and registration process in the bootstrap stage.

**Fig. 7** Security rules issued.

**Fig. 8** Security rules interwork.

**Fig. 9** Workflow of signature database updating in Antivirus module.
(1) The Security Center imports the virus signature database: There is an option to import the virus signature database in the Security Center.

(2) The Security Center issues the virus signature database: There is an option to issue the virus signature database in the Security Center.

(3) An interface displays that the virus database has been updated: The time changes.

(4) Virus signature database synchronization between peer-UTMs is performed using the p2p mode.

3.1.2.3 Firewall module in collaborative security
Workflow of Firewall module is shown in Fig. 10.

(1) Importing Firewall rules to Security Center: The option of importing Firewall rules to the Security Center.

(2) Security Center issues the Firewall rules: The option of issuing the Firewall rules.

(3) Interface displays that the Firewall rules have been updated: There is a web UI in the peer-UTM collaborative security Firewall module. The web UI shows the new Firewall rules that will be updated.

(4) The peer-UTMs choose to implement the rules that are updated by the Security Center: In the display UI of the update rules, the peer-UTMs are allowed to let a certain rule lapse if that rule is not applicable.

3.1.2.4 Protocol control and content matching module in collaborative security
Functions and principles are the same as Section 3.1.2.3.

3.1.3 Security rule center
The security rule center in security center runs on Debian Linux, and includes a rule distribution module and an event alarm module.

The rule distribution module is divided into server and client, and the server program is a socket communication module written in Java that is manually activated in the Security Center. The client program is running in the peer-UTM Firewall and Rules control module. When the server and client programs are running normally, the Security Center can quickly transfer rules in a specific folder to the peer-UTM’s rule control module.

The event alarm module is a web application based on Apache Tomcat that opens port 443 for this service. It listens for any security event reported from the peer-UTM, and dynamically displays these on the web UI.

The rule format for the Security Center and peer-UTM communication mechanism is shown in Appendix A, in details.

3.1.4 Virtualization network based on SDN
An experimental platform based on Openflow SDN is shown in Fig. 11, and the detailed configuration is shown in Table 1. Network topology and the host IPs are allocated in the following 2 rules:

(1) Each virtual machine is bridged to eth0 (192.168.0.0) segment for easy ssh control.

(2) Another controller is also present in this section and controls Openflow switch communication.

In the Openflow switch, there are three Ethernet ports that can be connected according to experimental needs. Each client has an Ethernet port that can be connected to any Openflow switch according to experimental needs. We use Openflow in the experimental environment. Here are some details of the

Fig. 10 Workflow of rules issuing to Firewall module.

Fig. 11 Virtualization network based on SDN.
Table 1 Detailed configuration settings in virtualization network based on SDN.

<table>
<thead>
<tr>
<th>VMs</th>
<th>IP</th>
<th>Hostname</th>
<th>Host alias</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
<td>192.168.0.60</td>
<td>Saturn-controller</td>
<td>Control</td>
<td>Ubuntu 13.10 (flood light)</td>
</tr>
<tr>
<td>OpenFlow1</td>
<td>192.68.0.61</td>
<td>Saturn-openFlow1</td>
<td>Of1</td>
<td>Ubuntu 13.04</td>
</tr>
<tr>
<td>OpenFlow2</td>
<td>192.68.0.62</td>
<td>Saturn-openFlow2</td>
<td>Of2</td>
<td>Ubuntu 13.04</td>
</tr>
<tr>
<td>OpenFlow3</td>
<td>192.68.0.63</td>
<td>Saturn-openFlow3</td>
<td>Of3</td>
<td>Ubuntu 13.04</td>
</tr>
<tr>
<td>Client1</td>
<td>192.68.0.64</td>
<td>Client1</td>
<td>Client1</td>
<td>Ubuntu 13.04</td>
</tr>
<tr>
<td>Client2</td>
<td>192.68.0.65</td>
<td>Client2</td>
<td>Client2</td>
<td>Ubuntu 13.04</td>
</tr>
</tbody>
</table>

set up deployment.

(1) In Openflow 1.0, there is only a single userspace datapath. The dpctl that played an important role in previous versions is replaced by ofdatapath and ofprotocol.

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

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

(4) dpctl can still be used to view the flow table in the Openflow switch, but cannot be used to manage the datapath.

(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).

3.2 Deep security check in vCNSMS

3.2.1 Function settings

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

(1) Security rules are incrementally downloaded. In the Security Center, duplicate rules are removed from the new rule set and make the new rules available. The new rules will be added in a package and issued to the peer-UTMs.

(2) 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.

(3) 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.

3.2.2 Enhanced security functions

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

(1) 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).

(2) 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 contain the specified text string “abc” will be matched and logged.

(3) 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.

3.2.3 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 load 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 `createRegExPattern` 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.

3.3 Intelligent flow processing in vCNSMS

Intelligent flow processing is an advanced method in Refs. [29-31] 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.

3.3.1 Security level based protection policy

The security level based protection policy is shown as follows:

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

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

(3) 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.

3.3.2 Implementation of security level with security function plugins

We modify the untangle Shield module[27,32] 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).

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

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

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

(5) 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.

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

3.3.2.1 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.

3.3.2.2 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
The Shield test environment is shown in Fig. 13. 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\cite{34}.

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

The detailed parallelization scheme is given as follows:

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

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

3. 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 (`echo 1>/proc/sys/net/ipv4/ip_forward`) to enable the kernel forwarding function.

3.3.2.3 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.


(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%.

As shown in Fig. 14, 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\cite{35}.

3.3.2.4 Bottleneck analysis of Shield

We use Intels VTune tools in the compile stage of the Shield module. Running performance 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.\cite{36}. We can further improve performance with `vPF_ring`\cite{37} in virtualized environments.

4 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.

5 Future Work

Based on the practical deployment and operational experience gained from evaluating vCNSMS in a data center network, vCNSMSs security center is able to deploy more and more security rules and collect data of network events. It should be possible to detect network policy violations and intrusion with an artificial intelligence based on unsupervised learning methods. This is a promising area for exploration in the future.

Acknowledgements

This work was supported in part by the National Key Basic Research and Development (973) Program of China (Nos. 2013CB228206 and 2012CB315801), the National Natural Science Foundation of China (Nos. 61233016 and 61140320).

This work was also supported by the Intel Research Council with the title of “Security Vulnerability Analysis based on Cloud Platform with Intel IA Architecture” and Huawei Corp.

Appendix A

A3. Rule format of Protocol Control module (refer to IP-filter HOWTO)

<table>
<thead>
<tr>
<th>Rule format of Protocol Control module such as MSN (# is a comment).</th>
</tr>
</thead>
<tbody>
<tr>
<td># MySpace IM - MySpace chat client</td>
</tr>
<tr>
<td># Protocol groups: chat proprietary</td>
</tr>
<tr>
<td># MySpace IM</td>
</tr>
<tr>
<td># Written by community</td>
</tr>
<tr>
<td>TryLogin2. * ?Final$</td>
</tr>
</tbody>
</table>

Rule Format of Protocol Control module for DNS.

<table>
<thead>
<tr>
<th>Rule Format of Protocol Control module for DNS.</th>
</tr>
</thead>
<tbody>
<tr>
<td># Protocol – DNS request/reply</td>
</tr>
<tr>
<td>DNS</td>
</tr>
<tr>
<td># Protocol groups:</td>
</tr>
<tr>
<td>UDP</td>
</tr>
<tr>
<td># Port</td>
</tr>
<tr>
<td>53</td>
</tr>
<tr>
<td># Written by community</td>
</tr>
<tr>
<td>Abc</td>
</tr>
</tbody>
</table>

References


Zhen Chen et al.: Collaborative Network Security in Multi-Tenant Data Center for Cloud Computing

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.


Zhen Chen is an associate professor in Research Institute of Information Technology at Tsinghua University. He received his BEng and PhD degrees from Xidian University in 1998 and 2004, respectively. He once worked as postdoctoral researcher in Network Institute of Department of Computer Science and Technology in Tsinghua University during 2004 to 2006. He was also a visiting scholar in UC Berkeley ICSI in 2006. He joined Research Institute of Information Technology at Tsinghua University since 2006. His research interests include network architecture, computer security, and data analysis. He has published around 80 academic papers.

Junwei Cao is currently a professor and deputy director of Research Institute of Information Technology, Tsinghua University, China. He is also the director of Open Platform and Technology Division, Tsinghua National Laboratory for Information Science and Technology. His research is focused on advanced computing technology and applications. Before joining Tsinghua in 2006, Junwei Cao was a research scientist of Massachusetts Institute of Technology, USA. Before that he worked as a research staff member of NEC Europe Ltd., Germany. He got his PhD in computer science from University of Warwick, UK, in 2001. He got his MEng and BEng degrees from Tsinghua University in 1998 and 1996, respectively. He has published over 130 academic papers and books, cited by international researchers for over 3000 times. He is a senior member of the IEEE Computer Society and a member of the ACM and CCF.

Hang Li is currently a master student of Department of Computer Science and Technology at PLA Univ. of Info. & Eng. She got her BEng degree from PLA Univ. of Info. & Eng. in 2013. Her main research interest focuses on computer network security.

Peng Zhang received his BEng degree from Beijing University of Posts and Telecommunications in 2008, and PhD degree from Tsinghua University in 2013. He was a visiting assistant in Research with the Department of Computer Science, Yale University from 2012 to 2013. He is now an assistant professor in Department of Computer Science and Technology, Xi’an Jiaotong University, China. His research interests include network security and privacy, software-defined network, and information-centric networks.

Wenyu Dong is currently a master student of Department of Computer Science and Technology at Tsinghua University. He got his BEng degree from PLA Univ. of Info. & Eng, in 2009. Currently his main research interests focus on computer network security and mobile safety.

Xinming Chen is currently a PhD candidate in Dept. of Electrical and Computer Engineering, University of Massachusetts, Amherst. He received his BEng and MEng degrees from Tsinghua University, China, in 2009 and 2011, respectively. He is also a member of the Network Systems Laboratory. His research interests include Internet architecture, multi-core processing, and deep packet inspection.