Customized Network Security for Cloud Service

Customized Network Security for Cloud Service

Jin He, Kaoru Ota, *Member, IEEE*, Mianxiong Dong, *Member, IEEE*,
Laurence T. Yang, *Senior Member, IEEE*, Mingyu Fan,
Guangwei Wang, and Stephen S. Yau, *Life Fellow, IEEE*

Abstract—Modern cloud computing platforms based on virtual machine monitors (VMMs) host a variety of complex businesses which present many network security vulnerabilities. In order to protect network security for these businesses in cloud computing, nowadays, a number of middleboxes are deployed at front-end of cloud computing or parts of middleboxes are deployed in cloud computing. However, the former is leading to high cost and management complexity, and also lacking of network security protection between virtual machines while the latter does not effectively prevent network attacks from external traffic. To address the above-mentioned challenges, we introduce a novel customized network security for cloud service (CNS), which not only prevents attacks from external and internal traffic to ensure network security of services in cloud computing, but also affords customized network security service for cloud users. CNS is implemented by modifying the Xen hypervisor and proved by various experiments which showing the proposed solution can be directly applied to the extensive practical promotion in cloud computing.

Index Terms—Network security, FDCs, unified management, customized network security service, packet delay, throughput

1 Introduction

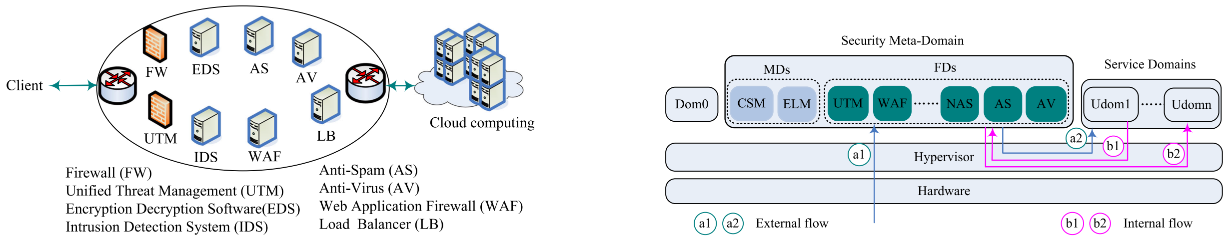
Cloud computing has emerged as one of the most influential paradigms in the IT industry, and has attracted extensive attention from both academia and industry. Reduced costs and capital expenditures, increased operational efficiencies, scalability, and flexibility are regarded as benefits of cloud computing. Although the great benefits brought by cloud computing paradigm are exciting for IT companies, academic researchers and potential cloud users, security problems of cloud computing become serious obstacles which, without being appropriately addressed, will limit extensive applications and utilization of cloud computing in the future. In cloud computing, network security [8], [9], [17], [46], [53] is believed to be one of the prominent security concerns, and it poses the same deadly threat as data security and privacy disclosure. Furthermore, as stated by National Vulnerability Database [29], there are 84 network vulnerabilities discovered in cloud computing by February 2013, all of which strongly threaten network security of cloud computing. In addition, there is sufficient evidence [27] that a large number of data destruction or tampering or forgery in cloud computing still come from malicious network attacks.

In recent years, there have been a number of relative efforts [1], [18], [22], [28], [47], [48], [52] in probing into data security and privacy in cloud computing, and tremendous progress has been maintained. However, these outcomes are based on an assumption that there has been secure network of cloud computing, and if the assumption is got rid of, the above achievements would come to be naught. Further, some researchers pay much attention to certain types of network security in cloud computing. For example, Lin et al. [19] have placed network inspection detection system into a privileged virtual machine (VM) to verify all packets received by the cloud platform. However, this approach has an unavoidable drawback: the privileged VM causes serious performance bottlenecks. Wu et al. [50] focus on the security of virtual network in virtualized environment and solve network security between VMs by Firewall. However, it is powerless for attacks from malicious external traffic. McAfee Security-as-a-Service [34] merely focuses on Email and Web protection in cloud computing, Imperva Cloud [41] and Du et al. [5] provide Distributed Denial-of-Service (DDoS) protection service, and Krishnan et al. [14] attach importance to intrusion detection system in cloud computing. Huawe security products [30] also only provides a single type of network security service for cloud computing. The preceding solutions are provided for a single type of service protection or detection in cloud computing (e.g., Web or E-mail), and they are lacking of integrated comprehensive protection for multi-service cloud.

Since cloud computing hosts multi-type network-based service which requires a desired sequence of multiple middleboxes together to protect their network security. For example, Web service needs Firewall and Web Application Firewall chains (FW-WAF) to protect network security.

- *J. He, M. Fan, and G. Wang are with School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, P.R. China. E-mail: hejin_some@163.com, ff98@uestc.edu.cn, gwwanguestc@hotmail.com.*
- *K. Ota and M. Dong are with Department of Information and Electronic Engineering, Muroran Institute of Technology, Muroran, Hokkaido 050-8585, Japan. E-mail: {ota, mx.dong}@csse.muroran-it.ac.jp.*
- *L. T. Yang is with School of Information and Communication Engineering, University of Electronic Science and Technology of China, China, and Department of Computer Science, St Francis Xavier University, Canada. E-mail: ltyang@gmail.com.*
- *S.S. Yau is with Computer Science and Engineering at Arizona State University, Tempe, AZ 85281. E-mail: yau@asu.edu.*

Manuscript received 23 Jan. 2016; revised 12 June 2016; accepted 8 July 2016. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Kaoru Ota.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSC.2017.2725828



(a) (b)

Fig. 1. Architecture comparison between the traditional architecture and CNS, (a) the traditional architecture, (b) CNS architecture, requiring external or internal traffic to traverse a desired sequence of FDs before accessing servers in service domains.

Thus, single network security service is unable to meet network security requirements for cloud computing. Considering the above shortcomings of single network security service, both industries and academies put many efforts on alternative solutions. In industry, traditional architecture [40] [10], [42] from Fig. 1a is regarded as current prevalent solution for multi-type service cloud, requiring large-scale security middleboxes, which leads to high costs [42], high complexity, and serious performance overhead. Besides, the architecture does not effectively prevent attacks between VMs [33], [48]. In academia, recent efforts [6], [13], [24] and [35] well combine middleboxes with SDN to protect enterprise network security and provide a flexible scalability and resource optimization for middleboxes. However, they lack of automatic security rules configuration and unified log management for middleboxes, and cannot provide appropriate cloud security.

Since above-mentioned efforts is inappropriate or defective to protect network security of cloud computing, the CNS system is presented that which adopts novel approach to eliminate or mitigate the disadvantages with promising benefits for cloud computing—reduced expenditure for infrastructure, personnel and management, pay-by-use, etc. As shown in Fig. 1b, the scheme is put forward in which security middleboxes are placed in cloud computing instead of at front-end of cloud computing so as to prevent malicious attacks from external and internal traffic. This will end mutual attacks between VMs for the traditional architecture. For security requirements in which cloud users' service is placed in cloud computing, CNS offers customized network security service to meet on-demand network security service. CNS also offers automatic security rules configuration and unified log management for middleboxes so as to lower complexity management and costs for cloud provider. Note that security capabilities or optimization algorithm of each device or middlebox [12] is not enhanced under this approach, but a more affordable and convenient protection service is provided.

In summary, our main contributions are as follows:

- *Innovative architecture* A novel flexible effective architecture for network protection of cloud computing is proposed. Based on best knowledge, a systematic approach to provide on-demand unified solution for network security protection of cloud computing is advocated.
- *Preventing attacks from external and internal traffic* CNS prevents network attacks not only from external traffic but also attacks from internal traffic so as to ensure network security of cloud users' service.
- *Customized network security service* So long as cloud users understanding their service security hosted on cloud computing raise security requirements, CNS can provide network security protection for their service.
- *Low cost and complexity* CNS provides virtual middlebox with automatic security rules configuration, and offers unified log UI for a cloud user and cloud administrator. By this approach, cloud providers pay lower price to provide cloud users with safe and trusted security service. Accordingly, cloud users also have access to low-cost service fees.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 provides an overview of the CNS design. Section 4 gives implementation details of the entire system. Section 5 presents various experimental results for evaluating system impact and performance. The paper is concluded Section 6.

2 Related Work

This section presents literature review on several research areas related to CNS, including cloud-based single network security service and cloud-based integrated security service.

Cloud-based single security service. Focus on providing the security for a certain type of service, preventing certain types of attacks, or optimizing a certain type of middleboxes.

Cloud computing + IDS: In recent years, instruction detection system (IDS) for cloud computing has become research focus for numerous experts studies. For vulnerabilities of a cloud system and compromising virtual machines to deploy further large-scale DDoS, NICE [3] has proposed a multi-phase distributed vulnerability detection, measurement, and countermeasure selection mechanism, which is constructed on attack graph based analytical models and reconfigurable virtual network-based countermeasures to significantly improve attack detection and mitigate attack consequences. Because of distributed nature, grid and cloud computing environments can become the targets which intruders look for and become possible vulnerabilities to exploit. Meanwhile, it requires more than user authentication with passwords or digital certificates and confidentiality in data transmission to provide the security in a distributed system. Vieira et al. [49] have integrated knowledge and behavior analysis to detect specific intrusions. Regardless of host-based IDS, network-based IDS, knowledge-based IDS, or behavior-based IDS, Modi et al. [25] have surveyed different intrusions affecting availability, confidentiality and integrity of cloud resources and service and have recommended IDS/IPS positioning in cloud environment to achieve desired security in the next generation networks.

Cloud computing + DOS: One of the most serious threats to cloud computing itself comes from HTTP Denial of Service or XML-Based Denial of Service attacks, Chonka et al. [2] have offered a solution to trace back through our Cloud TraceBack (CTB) to find attack source, and have introduced the use of a back propagation neutral network, which was trained to detect and filter such attack traffic.

The above research can only provide a single type of network security, they could do nothing for integrated network security service.

Cloud-based integrated network security service. provide integrated service with security protection (such as enterprise, data center, cloud computing). Existing research lays particular emphasis on middleboxes coupled with cloud computing or SDN.

Cloud computing + middleboxes: Salah et al. [38] focus on integrating the most popular types of middleboxes (e.g., IDSs, distributed denial-of-service (DDoS), FW, etc), which aims at offering an integrated set of security service for cloud computing. However, this brings a huge challenge to configurate security rules and manage so many middleboxes. CNS not only provides comprehensive security services, but also facilitates the provision of management and configuration. APLOMB, Embark [15] and Yuan et al. [51] considered that current middlebox infrastructure is expensive and complex to manage, and generates new failure modes of networks, it outsources enterprise middlebox processing to the cloud, solves security problems faced by modern enterprises. CNS as security provider on the cloud can provide APLOMB with outsourcing security services.

SDN + middleboxs: Cloudwatcher [43], which provides monitoring service for large and dynamic cloud networks, automatically detours network packets to be inspected by pre-installed network security devices. Compared to CNS, this work is lack of log and event unified management and detailed analysis of filtering rules on the middlebox, and does not solves middlebox hotspots on FDCs. CoMb [39] addresses key resource management and implementation challenges that arise in exploiting benefits of consolidation in middlebox deployments, but this work is almost difficult to achieve CoMb system due to middleboxes' closed system and incompatible architecture, and large development costs. SIMPLE [35], based on a SDN-based policy enforcement layer, takes an explicit stance to work within the constraints of legacy middleboxes and existing SDN interfaces, ensuring that the traffic is directed through the desired sequence of middleboxes and overcoming significant manual effort and operator expertise. However, this work is lack of log and event unified management and detailed analysis of filtering rules on the middlebox, and does not provide security service for cloud security.

Cloud computing + SDN + middleboxs: Split/Merge [37] can be dynamically scaled out (or in) virtual middleboxs in cloud computing, and enables load-balanced elasticity: Per-flow state may be transparently split between many replicas or merged back into one. However, this work mainly focuses on how to dynamically scaled out (or in) virtual middleboxes, and it does not provide customized network security service in cloud computing according to cloud user's security requirements and unified management.

The above cloud-based integrated network security service is lack of perfect fusion among middleboxes, cloud computing and SDN, neither provides customized network security for cloud service, nor considers maintenance costs and management complexity.

3 Design

Before the CNS design is demonstrated, it is envisioned that hardware platform, hypervisor and VMs on cloud computing are trusted and what is focused is network security of service in cloud computing. The CNS design dedicates three aspects:

- *Preventing malicious attacks from external and internal traffic*: As shown in Fig. 1b, CNS prevents network attacks from both external traffic and internal traffic to ensure network security of service domains. Whenever accessing to service domains, external traffic or internal traffic needs to pass through a desired sequence of filer domains (FDs) (e.g., FW–WAF) to prevent malicious attacks (it is also called VMs filter domains). The specific design and implementation are presented in §3.1 and §4.3.
- *Customized network security service*: Most cloud users known clearly about security requirements for the service in service domains and prefer specific measures according to their requirements. CNS adds corresponding security rules into FDs on a sequence of FDs path and forwards traffic to go through this sequence, which ensures network security. The specific design and implementation are presented in §3.2 , §4.1 and §4.3.
- *Reducing cost and complexity*: It reduces device hardware cost by migrating middleboxes to VMs in cloud computing. Furthermore, automatic analysis about cloud users' customized network security requirements and unified log management from FDs lower management complexity and costs. This section is presented in §3.3, §4.1 and §4.2.

Unlike MtoVM [7], [8] that migrates all middleboxes to the same VM, The CNS system migrates each middlebox to a separate VM. As the comparison experiment demonstrates in (§5.2), CNS gets much better performance than MtoVM. Before the design is introduced, the notation of a desired sequence of FDs is defined as filter domain chain.

Definition 1 (FDC). *Filter domain chain (FDC) represents a desired sequence of filter domains, and traffic must go through FDC to ensure their network security before arriving at servers in service domains. For example, FDC (FW → WAF) of web traffic goes through Firewall and WAF.*

3.1 Component

As shown in Fig. 1b, CNS consists of the following several components: a system domain (dom0), MDs, FDs, service domains and virtual switch (vSwitch).

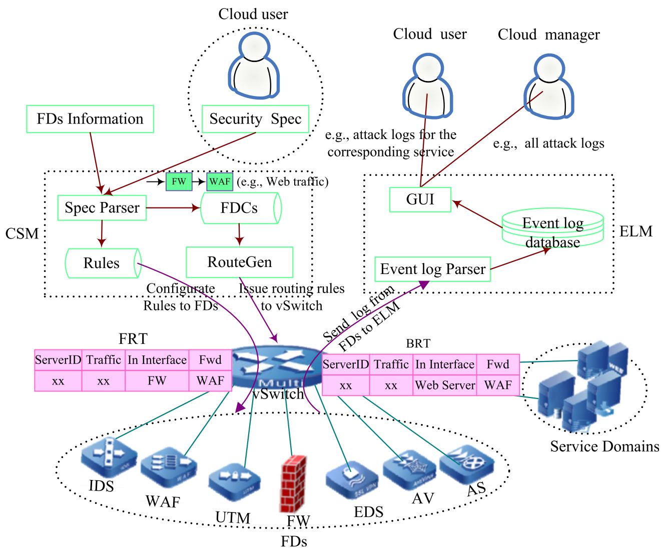


Fig. 2. CNS design.

- *Dom0* We weaken dom0 privileges, it does not have the permission to create/start and stop/destroy any domain in FDs. However, these permissions are reserved: it still has the privileges to operate every domain in service domains and management domains (MDs) and manage resources, including scheduling time-slices and I/O quotas.
- *MDs* are composed of cental security management domain (CSM) and event and log management domain (ELM). CSM has permission to create/start and stop/destroy any domain in FDs, manage and control FDs, and provide security inspection path for incoming/outgoing traffic of service domains. ELM stores and manages security events and logs from FDs and provides audit inquiry and attack statistics for cloud users and cloud administrator.
- *FDs* are a real network security inspection performer comprised of various virtual middleboxes. Network security inspection (e.g., anti-virus, filtering), decryption and encryption are realized by FDs, and this ensures that incoming/outgoing traffic to/from service domains are secure and trusted. FDs flexibly provide service domains with different security inspections according to security needs of different network-based service (called customized network security service).
- *Service Domains* hosts multiple types of service (e.g., FTP server, Web server) owned by cloud users.
- *vSwitch* receives forwarding rules from MD and forwards external and internal traffic through FDs to be filtered and inspected.

MDs and FDs cooperate jointly to provide customized network security service for cloud users, of which MDs provide incoming and outgoing traffic of service domains with their corresponding FDCs as inspection path and FDs perform security inspection when these traffic goes through FDCs. The focus of customized network security service lies in the fact that different service in service domains corresponds to different FDCs. For example, as shown in Fig. 3b, the FTP server corresponds to its FDCs, while Web server has corresponding FDCs in Fig. 3c. Due to different security requirements, the same type of service also has different FDCs. For example, the encrypted Email traffic passes through its corresponding FDCs (FW–EDS–SSL/VPN), whereas the non-encrypted Email passes through FW–EDS.

Fig. 1b shows that external and internal traffic must traverses their corresponding FDCs before arriving at service domains. When external traffic accesses the service in service domains, it is subjected to security inspection through a1, and then forwarded to service domains through a2; Internal traffic can not directly access service domains, and it must go through its corresponding FDCs (b1 and b2).

3.2 Customized Network Security Service

CNS provides customized network security service according to cloud users' various security requirements. Cloud users who know clearly about security requirements of their services in service domains only need to fill their security requirements in accordance with security spec template provided by cloud provider, and then deliver it to CNS. All the rest will be accomplished by CNS which automatically generates corresponding FDCs and security rules according to users' security spec and adds corresponding security rules into filter domains on FDCs path. The traffic must pass through FDCs to be inspected so as to ensure network security before arriving at cloud users' services.

As shown in Fig. 2, spec parser in the CSM analyzes users' spec and generates *FDCs* in both directions (incoming

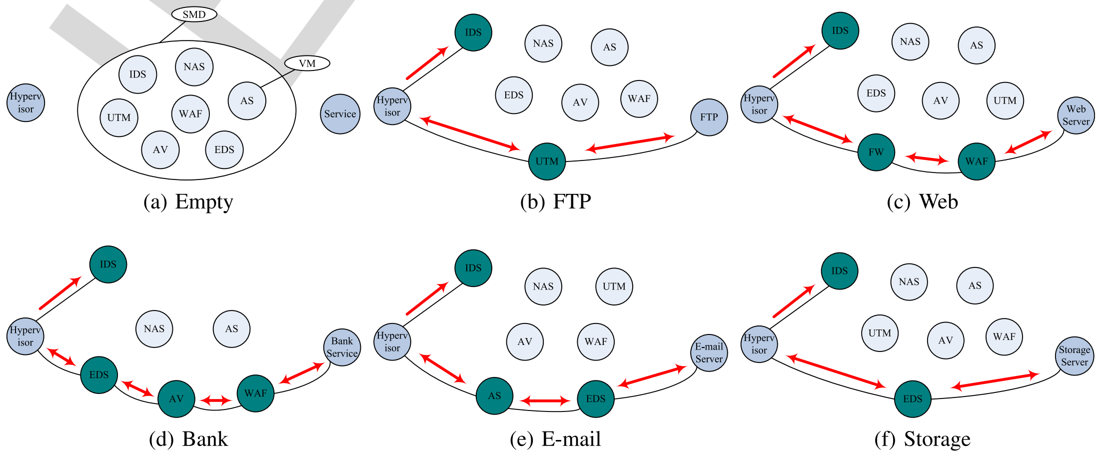


Fig. 3. Examples of customized network security service for different services.

TABLE 1
Corresponding Customized Network Security Service for Security Requirements of Different Protected Servers

Service Name	Attack	Filter Domains	Filter Domain Chains
FTP server	Session hijacking, Bounce attack, etc.	UTM	Fig. 3b
Web server	DDOS HTTP-DOS SQL injection XSS, etc	FW+WAF	Fig. 3c
Bank service	Date interception, SQL injection, Virus attack, etc	EDS+AV+WAF	Fig. 3d
E-mail server	Date interception, Spam mail, etc	EDS+AS	Fig. 3e
Storage service	Date interception, etc	EDS	Fig. 3f

TABLE 2
Actors and Operations in the Privilege Model

Log type	Cloud Administrator	Cloud User
System logs	✓	
Audit logs	✓	
Attack logs	✓	own services ✓
Statistical reports	✓	own services ✓

Each ✓ in the table denotes that the actor can perform the corresponding operation.

and outgoing traffic) and corresponding *security rules*. These security rules are issued to FDs on FDCs path and incoming and outgoing traffic pass through these security rules on their FDCs to be filtered and inspected. For example, Web server in service domains utilizes FW and WAF to protect its network security. After analysis, security rules protecting Web server are configurated to the FW and the WAF, and FDC of its incoming traffic is FW → WAF, FDC of its outgoing traffic is WAF → FW due to the fact that most network security middelboxs are stateful and need to process both directions of a session for correctness. Refer to §5.1 for detailed content and analysis of security spec.

There are many ways to realize the function that traffic from/to service domains must go through their corresponding FDCs, a more concise way is on based on forwarding rules [23]. RouteGen in the CSM converts FDCs into forwarding rules placed in vSwtich (§5.3). In order to fast find forwarding rules, the vSwtich contains two forwarding tables: Forward Route Table (FRT) and Backward Route Table (BRT). Forwarding rules of incoming traffic is placed in the FRT, forwarding rules of outgoing traffic is placed in the BRT. The above Web server is considered as an example of forwarding: when a client accesses the web server, the vSwtich inquires forwarding rules from the FRT and Web incoming traffic is first forwarded to the FW, then the WAF, finally arrives at the web server; outgoing traffic is forwarded oppositely. In the following, a few examples of customized network security services are enumerated.

Example. It is assumed that a cloud user inquires cloud provider to provide network security of both network-layer (e.g., data link layer, network layer) and application-layer (e.g., website) for Web server in service domains. The CSM analyzes users' security requirements in conjunction with FDs topology: FW is used to protect network-layer security so as to avoid DDOS attack, UDP and ICMP flood, etc, and the WAF is used to protect application-layer security so as to avoid SQL injection, cross-site scripting attacks, etc. Therefore, an ordered combination of the FW and the WAF is adapted to protect network security of Web server required by cloud user. The specific FDCs are shown in Fig. 3c. Table 1 envisages the situation in which cloud users raise security requirements for various servers suffering from network attacks and CNS provides corresponding solutions shown in Fig. 3.

3.3 Unified Management

CNS provides unified management for FDs in terms of unified configuration management and unified log management. In the traditional architecture, administrators have to face much tedious configuration management from independent vendors and different types of middleboxes. In cloud computing, if the same problem as the traditional architecture cannot be solved appropriately, it is almost an impossible task for cloud administrators to configurate and manage such a large diversity of FDs. As shown in Fig. 2, CNS can provide automated configuration and unified management to overcome these issues.

Automatic configuration CSM automatically analyzes user security spec in conjunction with the FDs topology, and then generates security rules directly configured into corresponding FDs and corresponding FDCs directly delivered to vSwitch by the method of forwarding rules, this process does not require human intervention (except for post-adjustment for special rules).

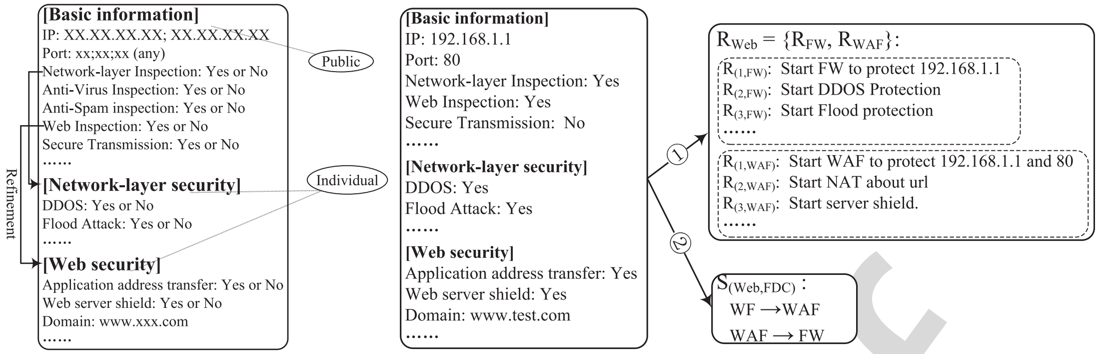
Unified log management ELM manages and counts all the logs (e.g., system logs, audit logs, attack logs) generated by FDs, and generates statistical reports based on attack logs. Logs from FDs are sent to ELM, analyzed by log analysis module in ELM, and placed in log database. To easily query logs and attack statistics, ELM provides cloud administrator and cloud users with GUI, and offers respective access privileges for different users shown in Table 2. Cloud administrator can access all the logs and statistics with high privileges, while cloud user can only access corresponding statistics and these logs are recorded when their service is under attack.

4 Implement

The above design elaborates the principle of CNS and this section presents the implementation of CNS in detail. First, CNS automatically analyzes customized security requirement spec required by cloud users, thereby generating corresponding security rules and FDCs. Second, unified logs management proves to be conducive to facilitating event and log query for cloud administrator and cloud user. Finally, the FDCs implement is presented by forwarding rules.

4.1 Customized Network Security Service Implementation

According to cloud users security requirements, CNS generates the corresponding security rules and forwarding rules to ensure services' network security. Section 3.2 shows the



(a) Customized security template (b) Security spec analysis of Web service

Fig. 4. (a) Cloud provider provides cloud users with customized security spec template in which cloud users fill in security requirements according to service security requirements. (b) CNS analyzes Web security spec from cloud users and generates filter domain chains of traffics in both direction and security rules.

principle design of customization of network security services, its focus is reflected in the implementation of security spec analysis.

Security spec template: As shown in Fig. 4a, cloud provider provides cloud users with customized security spec template, in which they fill in security requirements according to service requirements placed on service domains. The filled security spec is transmitted after encrypted in order to avoid being tampered by malicious cloud administrator [16]. The following explains some items in the template and presents some descriptive language for spec analysis. For the sake of clarity, we list some used symbols in Table 3.

IP and port: They represent protected object. IP and port fields in the basic information can be filled with one or more IP and port pairs, that is, one or more protected servers.

Algorithm 1. Spec Analysis Algorithm

1: // *Initialize corresponding security rules and FDC of each service.*
2: **for** *each* $S_i \in$ S **do**
3: $R_i \leftarrow \phi$
4: $S_{(i,fdc)} \leftarrow \phi$
5: **end for**
6: $R \leftarrow \phi$
7: **for** *each* $S_i \in$ S **do**
8: // *Analyze the protected Si requiring security rules and FDC.*
9: **for** *each* $m_j \in$ M **do**
10: // *Add a security rule to the corresponding middelbox.*
11: **while** *each* r_k *is yes* **do**
12: $R_{m_j} \leftarrow R_{(k,m_j)} \cap R_{m_j}$
13: **end while**
14: // *Security rules protected by* S_i.
15: $R_i \leftarrow R_{m_j} \cap R_i$
16: // *Add the needed middlebox to FDC.*
17: **if** *inspection item in base information is yes* **then**
18: $S_{(i,fdc)} \leftarrow S_{(i,fdc)} \cap m_j$
19: **end if**
20: **end for**
21: $R \leftarrow R_i \cap R$
22: **end for**

Network-layer, Anti-Virus, Anti-Spam, Web inspection and Secure transmission: These items in the base information are important parameters to determine which virtual middleboxes to provide protection for S security requirements, and each item has a corresponding virtual middlebox. For example, R_{FW} and R_{WAF} are respectively expressed as FW security rules and WAF security rules to protect Website server S_{web}, that is, S_{web} needs security rules R_{web} = {R_{FW}, R_{WAF}} to protect its network security.

Network-layer and Web security: They are the refinement of network-layer and Web inspection items in the basic information. Specifically, network-layer protection includes DDOS and flood attack etc. If any item in network-layer security is activated, $R_{(i,FW)}$ is used to express it, R_{FW} = $\{R_{(1,FW)}, R_{(2,FW)} \cdots R_{(n,FW)}\}$ indicates that FW consists of multiple rules $R_{(i,FW)}$. Similarly, R_{WAF} = {$R_{(1,WAF)}$, $R_{(2,WAF)}$ $\ldots R_{(n,WAF)}$}.

CSM accepts filled and encrypted spec from a cloud user. Spec parser in CSM first decrypts the security spec, analyzes it, and then generates FDCs of traffics in both direction and security rules for the protected service domains. We show the pseudocode about the spec analysis in algorithm 1. First, security rules and FDCs of the protected objects are initialized. That is, corresponding security rules of all the protected servers are set as R_i = {ϕ} and R = {ϕ} from 2 lines to 6 lines, and corresponding FDCs of S_i is set as NULL, i.e., $S_{(i,fdc)}$ = {ϕ}. Second, security rule $R_{(k,m_j)}$ is configured to corresponding virtual middlebox m_j to protect S_i according

TABLE 3
Spec Analysis Algorithm Needs the Symbol and its Explanation

Symbol	Repression
S	The set of servers filled in security spec;
S_i	A server that is a specific IP and port pair;
R	A collection of security rules to protect S, Multiple R_i protect S by R, $R = \{R_i \| i \in 1 \ldots n\}$;
R_i	A collection of security rules to protect S_i, and it may be dispersed in one or more virtual middleboxes on its corresponding FDCs path;
M	The set of virtual middleboxes;
m_j	Any one of M;
R_{m_j}	Security rules which m_j contains to protect S_i;
$R_{(k,m_j)}$	Security rule configured to the corresponding virtual middlebox m_j to protect S_i;
$S_{(i,fdc)}$	Corresponding FDCs of S_i;

<LogType><FDID><EventID><ServerID><SrcIP>
<SrcPort><DestIP><DestPort><Protocol><Description>

Fig. 5. Log format.

to 'yes' items in spec (lines 11-13). The same operation is performed for all the involved virtual middleboxes (lines 9-20). Third, these corresponding virtual middleboxes which provide S_i with network security are added into FDCs to protect S_i (lines 17-19). Finally, R_i is composed of security rules provided by one or more virtual middleboxes to protect S_i (lines 7-22).

Web example: Fig. 4b presents Web security spec provided by a cloud user and specific content generated by analysis. The left side in Fig. 4b shows that Web security spec enables two items in base information: network-layer and Web inspection. That is, Web server needs network-layer and Web application-layer security protection. It is obvious that a combination of FW and WAF meets security requirements of web server: First, corresponding FDCs of Web server are $S_{(Web,fdc)}$: FW $\rightarrow$ WAF and WAF $\rightarrow$ FW; Second, R_{Web} = $\{R_{FW}, R_{WAF}\}$ is configured to protect S_{Web} on FDCs path. All items in network-layer security detail Web network-layer security requirements, and $R_{(2,FW)}$ and $R_{(3,FW)}$ represent specific security rules. Similarly, all the items from Web security clarify application-layer ones and specific analytical results present $R_{(2,WAF)}$, $R_{(3,WAF)}$ etc.

4.2 Log Unified Management

Log unified management is also perceived as the CNS research emphasis. If each FDs has its own management user interface (UI) as what traditional way does, it is impossible for cloud computing administrators to log in so many UIs to view attack logs and statistics information due to massive and tedious work.

Furthermore, most of the servers in service domains may require multiple FDs to protect them, inspected attack logs scattering in multiple FDs do not form integral statistics and management, therefore, it is essential for log unified management.

Algorithm 2. Log Classification Algorithm

1: *// Classify every log.*
2: **if** *every log l* **then**
3: *// l is a system log.*
4: **switch** $(l_{.logtype})$
5: **case** *system log*:
6: $L_{ca} \leftarrow L_{ca} \cap l$
7: *// l log belong m_i logs.*
8: **if** $L_{.fdid} = (m_i \in M)$ **then**
9: $L_{m_i} \leftarrow L_{m_i} \cap l$
10: **end if**
11: *break*
12: *// l is a attack or statistic log.*
13: **case** *attack log and statistic log*:
14: $L_{ca} \leftarrow L_{ca} \cap l$
15: *// l log belong cu_i user.*
16: **if** $l_{.serverID} = (cu_i \in CU)$ **then**
17: $L_{cu_i} \leftarrow L_{cu_i} \cap l$
18: **end if**
19: *break*
20: **end switch**
21: **end if**

TABLE 4
Log Classification Algorithm Needs the Symbol and its Explanation

Symbol	Repression
l	A log;
CU	All the cloud users;
L_{ca}	Log and statistics database queried by cloud administrator;
cu_i	The ith cloud user;
L_{cu_i}	Log and statistics database only queried by the ith cloud user;
L_{m_i}	Log and statistics database from the ith virtual middlebox;

For events, logs and system information from FDs, ELM performs unified management to provide cloud computing administrators with convenient management and query. In order to easily identify and standardize all logs from FDs, FDs need to abide by a unified log format shown in Fig. 5. LogType indicates log type (e.g., attack log, system log); FDID is denoted as unique FD identifier; EventID is denoted as event identifier (e.g., attack number); ServerID indicates certain domain in service domains as unique identifier to facilitate server log information statistics; SrcIP, SrcPort, DestIP, DestPort, Protocol represent quintuple flow; Description represents detailed information of the event.

After ELM receives logs, these logs are classified by log parser in order to provide access on the basis of actor permissions in Table 2. Parameters in Table 4 are introduced to facilitate the description of log classification algorithm. Algorithm 2 offers log classification. A log l arrives at ELM, If l is a system log, l is added into L_{ca} (lines 6); If l belongs to a log of m_i, l is added into the corresponding L_{m_i} (lines 8-10); If l is an attack log, l is added into L_{ca} (lines 14); if l is generated due to the attacked server owned by the ith cloud user to be attacked, l is added into L_{cu_i} (lines 16-18). After classification, cloud administrator queries all the logs from FDs, cloud users only query their owner logs generated by corresponding middleboxes when their owner servers are being attacked.

4.3 FDCs Load Balancing Implementation

We have considered load balancing of each middlebox in FDCs, and avoid each middlebox becoming a hospot.

FDCs is important part to realize customized network security service for each service in service domains. RouteGen converts FDCs and the FDs topology into forwarding rules and issues these rules to the vSwitch. CSM is considered as a SDN controller, and vSwitch is responsible for forwarding packets to/from FDs and service domains according to forwarding rules delivered by CSM. That is, network security inspection is achieved on the basis of forwarding rules in the vSwitch.

Traffic accessing to a server in service domains is divided into two types of traffic: external traffic from Internet and internal traffic between service domains in cloud computing. Incompetence internal or external traffic must go through corresponding FDCs to ensure nework security of

TABLE 5
The List of Open Source Security Softwares

Product Name	Open Source Software
FW	IPFire [11]
WAF	ModSecurity [26]
SSL/VPN	OpenSSL [32]
AS	PacketFence [44]

service domains. By default, forwarding rules from external traffic have been stored in two route tables (FRT and BRT), while there is no forwarding rules from internal traffic in the mentioned route tables. The main reason goes that there is very little communication between service domains. If forwarding rules are added into route tables, it will result in larger route table and take longer time to look up corresponding forwarding rules from the route table, which would lead to performance degradation. If the communication is established between service domains, the default route in the vSwitch forwards the first packet between them to CSM. RouteGen in the CSM generates forwarding rules by FDCs of the accessed server and issues these rules to vSwich, and subsequent traffic is forwarded in accordance with forwarding rules in the vSwitch.

During FDCs generation process, we have to consider two natural requirements: (1) Each chain FDC should have enough virtual middelboxes assigned to it, so that we retain sufficient freedom to achieve near-optimal load balancing subsequently. (2) We ensure that we have sufficient degrees of freedom; e.g., each FDC will have a guaranteed minimum number of distinct physical sequences and that no middlebox becomes a hotspot.

$$Minimize\ max\{Load_{m_j}\},\ subject\ to \tag{1}$$

$$\forall c : \sum_{c \subset Path_{FDCs}} P_{c,m_j} = 1 \tag{2}$$

$$\forall j : Load_{m_j} = \sum_{c: m_j \in Path_{FDCs}} \frac{P_{c,m_j} \times T_c \times Footprint_{c,m_j}}{ProcCap_{m_j}} \tag{3}$$

$$\forall c, j : P_{c,m_j} \in [0, 1] \tag{4}$$

$$\forall j : MiddleboxUsed_{m_j} = \sum_{m_j \in Path_{FDCs}} UsedNumber_{mj} \tag{5}$$

$$\forall j : MaxMiddleboxOccurs \geq MiddleboxUsed_{m_j} \tag{6}$$

Thus, we can consider the management problem in terms of deciding the *fraction of traffic* belonging to each chain c (c $\subset$ FDCs) that each virtual middleboxes m_j has to process. Let P_{c,m_j} denote this fraction and let T_c denote the volume of traffic for each chain c. The optimization problem can be expressed by the linear program shown in Eqs. (1)–(6). Eq. (2) simply specifies a coverage constraint so that the fractional responsibilities across the virtual middleboxes on the path for each c add up to 1. Eq. (3) models the stress or load on each virtual middlebox in terms of the aggregate processing costs (i.e., product of the traffic volume and the footprints) assigned to this virtual middlebox. Here, $m_j \in Path_{FDCs}$ denotes that virtual middlebox m_j is on the routing path for the traffic in T_c. At the same time, we want to make sure that no virtual middlebox becomes a hotspot; i.e., many chains FDCs rely on a specific virtual middlebox. Thus, we model the number of chosen sequences in which a middlebox occurs and also the maximum occurrences across all middleboxes in Eqs. (5) and (6) respectively. Our objective is to minimize the value of MaxMiddleboxOccurs to avoid hotspots.

To summarize, CNS presents three important characteristics: 1) *preventing malicious attacks from external and internal traffic:* CNS prevents network attacks from external and internal traffic to ensure network security of service domains 2) *Customized network security service:* After cloud users put forwards security requirements according to their own server characteristics, CNS can be well adapted to meet security service requirements. 3) *Complexity and cost:* CNS can realize automatic configuration and management in accordance with cloud users' security spec without human intervention, which includes security rules configuration and FDCs and forwarding rules generation, and provide cloud administrators with unified logs management. Besides, CNS can provide cloud user with low-cost security service with respect to hardware and management costs of middleboxes.

5 Evaluation

In this section, there are four goals of the evaluation:

- valuate system benchmarks of CNS.
- evaluate the cost of CNS and the traditional architecture.
- evaluate maintenance and management complexity between CNS and the traditional architecture.
- evaluate performance between CNS and MtoVM, between CNS-unbind-core and CNS-bind-core and between with and without CNS.

Experimental environment Cloud platform is conducted on a Dell Server with 8 core, 3.42 GHz Intel CPU, 16GB memory. IXIA [31] and `iperf` are considered as a performance test instruments. The XEN hypervisor version is 3.4.2, and the dom0 system is fedora 16 with kernel version 2.6.31. We used a 64bit fedora Linux with kernel version 2.6.27 as our guest OS, and the vSwitch bandwidth is 1 Gigabit Ethernet; CNS uses open source security softwares shown in Table 5. For the next step, four simulation environments are installed.

- *MtoVM simulation environment:* Four kinds of open softwares in Table 5 are moved to the same VM.
- *CNS-unbind-core simulation environment:* Each software is moved to a separate VM in FDs.
- *CNS-bind-core simulation environment:* Each software is moved to a separate VM in FDs, and each VM is bound to a core, namely, each virtual middlebox runs on a separate core.
- Without security protection.

5.1 System Benchmarks

We focus on four key metrics here: the time to analyze spec, the time to install filter rules, the time to install forwarding rules, the total communication overhead at the

TABLE 6
Time and Control Traffic Overhead to Install Customized Network Security Service

Middlebox Number of each FDC	Time to Analyze Spec(ms)	Time to Install Filter Rules(ms)	Time to Install Forwarding Rules(ms)	Overhead (KB)
1	3.1	5.1	1.1	6
2	3.2	6.3	1.2	10
3	3.2	7.6	1.2	14
4	3.3	8.5	1.3	22
5	3.5	10.1	1.3	30
6	3.5	14.8	1.5	38
7	3.6	18.9	1.6	47
8	3.8	23.3	1.6	67
9	3.8	25.7	1.7	89
10	3.9	32.0	1.9	108

controller, and the maximum load on any middlebox or link in the network relative to the optimal solution. We begin by running the topology from Fig. 2 on the Emulab testbed. We did the comparison experiments according to the middleboxes number of FDCs, whose results shown are shown in Table 6.

Time to analyze spec. Table 6 shows the time taken by CNS to proactively analyze spce according to the middleboxes number from 1 to 10 of FDCs. The time to analyze increases from 3.1 ms to 3.9 ms as middlebox number in FDCs increases, but the increase is acceptable without large fluctuations. The main causes here is that the controller spends more time to analyzes more items in spec as middleboxes number of FDCs increases.

Time to install filter rules. Table 6 shows the time taken by CNS to install the filter rules for the FDCs. The time to install changes from 5.1 ms to 32.0 ms as the middlebox number of FDCs. The main bottleneck here is the controller spends mort time to install more filter rules and send more filter rules to each middlebox in FDC. We can reduce this to 70 percent overall with multiple parallel sending filter rules to middlebox.

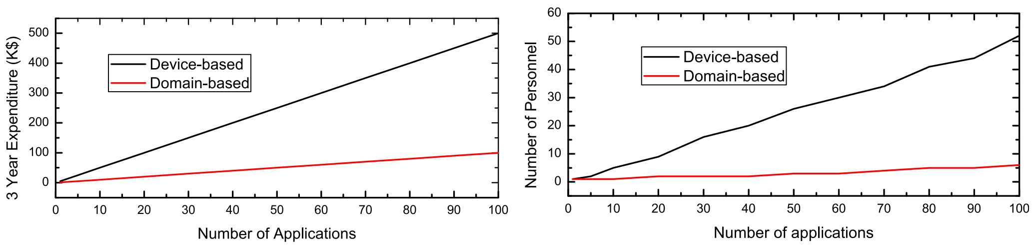
Time to install forwarding rules. The time to install forwarding rules is very short and almost does not change as as the middlebox number of FDCs. The main causes here is that it takes short time for the controller to analyze forwarding rules and sends forwarding rules only to vSwtich.

Controllers communication overhead. The table also shows the controllers communication overhead in terms of Kilobytes of control traffic to/from the controller to install filter and forwarding rules. Note that there is no other control traffic during normal operation. These numbers are consistent with the total number of rules that we need to install.

5.2 Cost and Complexity

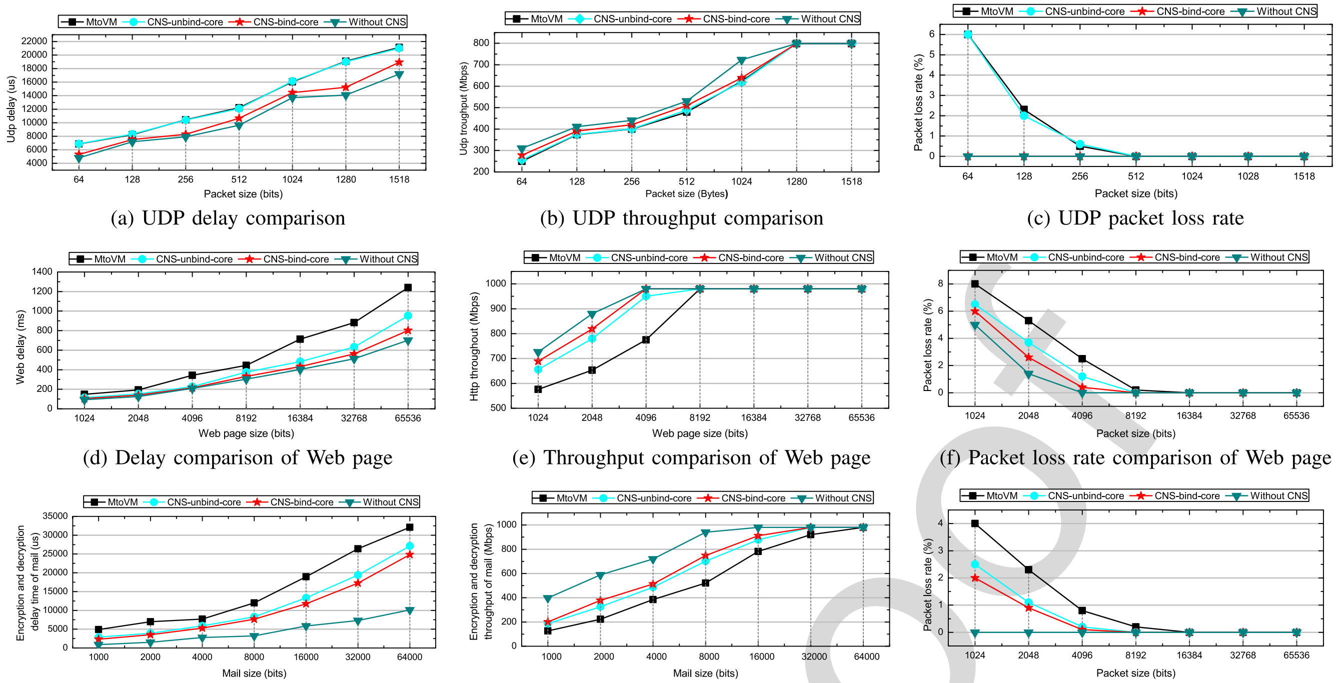
Cost. Since middleboxes (labeled as device-based) and FDs (labeled as domain-based) from independent vendors and different types of security devices or software have distinctive costs, only rough estimation rather than accurate assessment is conducted. Thus, thus the average cost of all the middleboxes and FDs are considered as their cost. Device-based and domain-based cost can be drawn according to benchmark cost [38], [42]. It can be seen from Fig. 6a that device-based cost is five times as that of domain-based. That is, the average cost of a middlebox is about \$5,000, while the average cost of an FD is only \$1,000. This saves the cost to a deep extent.

Complexity. Complexity focuses on configuration, maintenance and management. CNS provides security spec with automatic analysis without human intervention (except for strategy adjustment) so as to avoid security rules configuration complexity. This is especially useful for complex network security service which requires multiple middleboxes to meet full security protection, Complex network security service is a much difficult task which takes a lot of time, taking manual configuration and interactions between rules into consideration. In view of post-maintenance and post-management, the traditional architecture (labeled as device-based) is facing the complex and tedious work. For example, APLOMB [42] has conducted a survey of 57 enterprise network administrators and it is found that managing many heterogeneous middleboxes require broad expertise and consequently a large management team. Even small networks with only tens of middleboxes typically require a management team of 6-25 personnel. Unlike the traditional architecture, CNS proposes and implements automatic generation of security rules and FDCs and forwarding rules, and offers unified logs management and query. Fig. 6b presents the comparative data of management personnel between CNS and the traditional architecture in the light of the complexity: For CNS, only a few personnel are required to maintain and manage FDs, while the traditional architecture needs a management and maintenance team with large-scale personnel who rapidly grows as the number of applications increases. Especially, when the number of middleboxes reaches 100, the traditional architecture requires 50 personnel, whereas CNS only require 4 personnel.



(a) Cost comparison (b) Complexity comparison

Fig. 6. Cost and complexity comparison and between CNS and the tradition architecture.



(a) UDP delay comparison
(b) UDP throughput comparison
(c) UDP packet loss rate
(d) Delay comparison of Web page
(e) Throughput comparison of Web page
(f) Packet loss rate comparison of Web page
(g) E-mail delay comparison
(h) E-mail throughput comparison
(i) E-mail packet loss rate

Fig. 7. The performance comparison results between four cases: the traditional architecture, CNS-unbind-core, CNS-bind-core and without CNS.

5.3 Performance Discussion

In this section, the CNS performance is evaluated with the following goals:

- Why is CNS employed rather than MtoVM? The two are compared to prove the conclusion, and the reasons are analyzed.
- How to improve the CNS performance? What shall be done to overcome the difficulties?
- Performance overhead with CNS is evaluated to determine whether the overhead is acceptable.

Evaluation purposes are achieved by three sets of comparative experiments.

- *The first experiment* is that NCSS-unbind-core has been compared with MtoVM in term of system performance, and a better solution from comparison results can be selected , and the reasons which affect system performance are analyzed.
- *The second experiment*, some factors affecting CNS performance are overcome, and optimization results from the comparison between NCSS-unbind-core and CNS-bind-core is viewed.
- *The third experiment*, CNS performance overhead is evaluated, and related measurements on both the case with CNS-bind-core and the case without CNS-bind-core in cloud computing are performed, and whether the overhead is within acceptable range is assessed.

In order to response the comprehensive performance, every experiment presents nine sets of comparative data from three aspects of performance: latency, throughput, and packet loss rate witch are important indicator [21] of system performance about network security. The following present the three experiments.

The Comparison between MtoVM and CNS-unbind-core This is the first experiment to compare MtoVM with CNS-unbind-core about latency, throughput and packet packet loss rate. Their performances are compared by three types of service: UDP forwarding (FW), Website (FW-WAF), and Email service (FW-SSL/VPN).

In order to obtain comprehensive and correct performance assessment, UDP packets are employed with different sizes to evaluate system performance of MtoVM and CNS-unbind-core. Figs. 7a and 7b) presents experimental results about latency and throughput. In stress measurement environment, regardless of packet size from 64bit to 1528 bit, 500 Mbit/s throughput is always kept to observe packet loss. The experimental result is shown in Fig. 7c. In short, Figs. 7a, b, and Fig. 7c indicates two points: First, latency and throughput of UDP forwarding increase with the increase of their sizes. Second, MtoVM and CNS-unbind-core present the same performance, and the main reason is that UDP packets only traverse FW on the VM instead of all the virtual middleboxes on the VM although MtoVM employs multiple virtual middleboxes from Table 5 on a VM. Because CNS-unbind-core employs each virtual middlebox on a stand-alone VM, udp packets with CNS-unbind-core just go through FW according to FDC of UDP service. Therefore, MtoVM and CNS-unbind-core demonstrate the same performance in term of udp forwarding.

Website access [20], [45] based on TCP protocol needs FW and WAF to protect its network security. The experiment method is similar to the UDP forwarding except for http traffic and packets with larger size from 1024 bit to 65,536 bit. Figs. 7d, e, and Fig. 7f shows our experimental results: First, the relationship between packet size and performance is similar to the UDP forwarding. Second, regardless of latency or throughput or packet loss rate,

TABLE 7
CNS Performance Overhead Comparing to no Protective Measures in Cloud Computing

Access method	Performance	Max (%)	Min (%)	Avg (%)
UDP packet	Latency	9.1	4.2	6.4
	Throughput	13.4	0	8.8
	Packet loss rate	0	0	0
Web page	Latency	18.9	12.4	16.3
	Throughput	4.2	0	0.7
	Packet loss rate	6	0	0.9
Encrypted mail	Latency	15.7	9.2	13.1
	Throughput	5.1	0	0.8
	Packet loss rate	2	0	0.3

CNS-unbind-core is far higher than MtoVM in terms of performance.

The main reason goes like the following: the advantage of the MtoVM is that the entire inspection and filtering of Web traffic are performed only on a single VM, and this avoids the overhead of inter-VM communication and cache invalidations which may arise as shared state is accessed by multiple cores; compared with CNS-unbind-core, the MtoVM also has its own disadvantage which cloud incur overhead due to context switches and potential contention over shared resources on a single VM, especially, filtering rules and feature matching require a lot of CPU resources. Since CNS-unbind-core employs FW and WAF respectively on a stand-alone VM. Therefore, the advantages and disadvantages of CNS-unbind-core are opposite to MtoVM. A conclusion is drawn from Figs. 7d, e, and Fig. 7f) that resource contention and context switches extend greater impact than inter-VM communication and cache invalidations for MtoVM and CNS-unbind-core. If CNS is used on multiple-core virtual platform to perform parallel inspection, it is possible to overcome resource contention (especially, CPU resource) competition, which significantly improves system performance.

The importance of CPU resources for performance impact between MtoVM and CNS-unbind-core is further confirmed by e-mail encryption and decryption requiring more CPU resources. As shown in Fig. 3e, e-mail needs AS and SSL/VPN to protect its network security. Figs. 7g, h, and Fig. 7i shows CNS-unbind-core has a better performance than MtoVM in term of latency, throughput and packet loss rate. Even in the worst case, latency of MtoVM is twice than CNS-unbind-core at 64,000 bits.

In summary, regardless of UDP forwarding, Website access, Email access, Fig. 7 has showed that CNS-unbind-core realizes far higher system performance than that of the MtoVM. The main reasons is that a large number of rules and feature matching requires a lot of CPU resources which extend a greater impact than the overhead of inter-VM communication and cache invalidations for system performance. Therefore, CNS-unbind-core rather than MtoVM is adopted, which can achieve better system performance.

The Comparison between CNS-unbind-core and CNS-bind-core Resource competition, especially CPU resources, context switches, inter-VM communication and cache invalidations are regarded as main factors that affect system performance. CNS-unbind-core takes full advantage of system resources, (especially, CPU resources) and overcomes context switches over shared resources on a single VM. Since inter-VM communication between FDs makes full use of hardware-assisted I/O virtualization techniques such as single root I/O Virtualization (SR-IOV) [4] and self-assisted devices [36], it can reduce I/O virtualization overheads and achieve good performance. Therefore, inter-VM communication overhead is considered. However, multi-core scheduling constantly switches between multiple VM to lead to corresponding cache invalidations, which causes system performance degradation. In order to overcome cache invalidations, each FDs is binded to a CPU core, thus overcoming the disadvantage of cache invalidations. Fig. 7 shows our experimental results, as can be seen from nine sets of data that CNS-bind-core reflects a more superior performance than CNS-unbind-core.

The Comparison both With And Without CNS-bind-core in Cloud Computing. This is the third experiment, there are cases with the CNS and cases without the CNS in cloud computing. Fig. 7 presents the experimental comparison results indicating that the case without CNS-bind-core are more efficient than ones with CNS-bind-core. Although the case without employing CNS-bind-core to protect network security achives higher efficiency than one with CNS, it may lead to incalculable losses if no protective measures are taken to protect cloud computing security. Therefore, it is essential to protect network security of cloud computing so as to defend various attacks from the network. Even if CNS-bind-core is selected to protect cloud computing security, it is necessary to consider whether its performance overhead can be accepted. The following three experiments are still used to evaluate the performance impact with CNS-bind-core from three aspects: UDP forwarding, Website and Email service.

For UDP forwarding, Figs. 7a, b, and Fig. 7c shows NetSecCC gives little impact on system performance (specific performance overhead is shown in Table 7), compared with the case without NetSecCC, NetSecCC imposes 6.4 percent of average latency overhead (ranging from 4.4 to 9.1 percent) and 8.8 percent of average throughput drop (ranging from 0 to 13.4 percent). Packet loss rate suffers from the impact of security inspection and filtering. It is inevitable for these performance overhead to inspect and filter UDP traffic. Since UDP traffic must go through FW to be inspected and filtered before being forwarded to the UDP server in service domains. During the process, traffic is required to match hundreds of filtering rules in FW. This will take some time and result in increased latency and decreased throughput. Compared wtih MtoVM and CNS-unbind-core, CNS-bind-core has made tremendous progress.

For Website access, The results of this experiment showed in Figs. 7d, e, and Fig. 7f present CNS-bind-core has related impact on system performance. Compared with the case without CNS, latency is more affected, while throughput is hardly affected. Table 7 further illustrates that 16.3 percent of average latency overhead (ranging from 12.4 to 18.9 percent), 0.7 percent of average throughput drop (ranging from 0 to 4.2 percent), 0.9 percent of average overhead of packet loss rate (ranging from 0 to 6 percent). The main reason is like this: Web traffic must go through FW and WAF to be inspected and filtered before being forwarded to the Website server in service domains. In this process, traffic is required to match hundreds of filtering rules in FW and thousands of signatures in WAF, which will take some time and hence result in the increased latency and the decreased throughput. In the case without CNS, Web traffic directly accesses to the Website server to avoid inspection in terms of system overhead. Therefore, compared with the cases without CNS, latency becomes longer with CNS, throughput suffers from the impact of latency. However, overall system performance with CNS is within the acceptable range.

For e-mail access, the results of this experiment showed Figs. 7g, h, and Fig. 7i) present encrypted emails with CNS are affected. The impact of longer latency, lower throughput, and bigger packet loss with CNS is mainly caused by the reason that emails must be forwarded through AS and SSL/VPN as shown in Fig. 3e. In addition, the encrypted emails require encryption processing. This will take some time and lead to performance degradation. Compared with the case without CNS, specific data with CNS on the performance overhead are shown in Table 7: the average cost of latency is 13.1 percent (ranging from 9.2 to 15.7 percent), the average cost of throughput is 0.8 percent (ranging from 0 to 5.1 percent), and the average cost of packet loss rate is 0.3 percent (ranging from 0 to 2 percent). For security services, the preceding performance overhead is acceptable.

In summary, it is found that CNS-unbind-core is a more preferred method in terms of performance by comparing both MtoVM and CNS-unbind-core. On the basis of CNS-unbind-core, it is further optimized to produce more efficient CNS-bind-core and offer more efficient customized network security service. At the same time, by the comparison of the case with CNS-bind-core and the case without CNS-bind-core in cloud computing, it is found that CNS-bind-core can provide adequate network security protection for cloud computing without sacrificing the high price of system performance.

6 Conclusion

Main problems caused by todays cloud security are high costs and performance overhead, and management complexity, especially the lack of customized network security services. In this paper, we introduced a innovative architecture called CNS, which provides customized network security for security needs of suitable cloud services as well as the qualitative benefits with respect to low performance overhead, easy to maintenance and management, and reduction in middleboxes costs. Further, we gave a specific and detailed examples and algorithms in the process of implementation in order to leverage these benefits in practice. Next, we use CNS to offer customized network security service for big data, enterprise and outsourcing security through in-depth research.

Acknowledgments

This work is partially supported by JSPS KAKENHI Grant Number JP16K00117, JP15K15976, and KDDI Foundation.

References

[1] K. Alhamazani, R. Ranjan, P. P. Jayaraman, et al., "Cross-layer multi-cloud real-time application QoS monitoring and benchmarking as-a-service framework," *IEEE Trans. Cloud Comput.*, p. 1, 2015.

[2] A. Chonka, Y. Xiang, W. Zhou, and A. Bonti, "Cloud security defence to protect cloud computing against HTTP-DoS and XML-DoS attacks," *J. Netw. Comput. Appl.*, vol. 34, no. 4, pp. 1097–1107, 2011.

[3] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "Nice: Network intrusion detection and countermeasure selection in virtual network systems," *IEEE Trans. Dependable Sec. Comput.*, vol. 10, no. 4, pp. 198–211, Jul./Aug. 2013.

[4] Y. Dong, X. Yang, J. Li, G. Liao, K. Tian, and H. Guan, "High performance network virtualization with SR-IOV," *J. Parallel Distrib. Comput.*, vol. 72, no. 11, pp. 1471–1480, 2012.

[5] P. Du and A. Nakao, "DDoS defense as a network service," in *Proc. IEEE Netw. Operations Manage. Symp.*, 2010, pp. 894–897.

[6] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul, "Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags," in *Proc. 11th USENIX Conf. Netw. Syst. Des. Implementation*, 2014, pp. 533–546.

[7] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[8] J. He, M. Dong, K. Ota, M. Fan, and G. Wang, "NetSecCC: A scalable and fault-tolerant architecture for cloud computing security," *Peer-to-Peer Netw. Appl.*, vol. 9, no. 1, pp. 67–81, 2016.

[9] J. He, M. Dong, K. Ota, M. Fan, and G. Wang, "NSCC: Self-service network security architecture for cloud computing," in *Proc. IEEE 17th Int. Conf. Comput. Sci. Eng.*, 2014, pp. 444–449.

[10] J. He, M. Dong, K. Ota, M. Fan, and G. Wang, "PNSICC: A novel parallel network security inspection mechanism based on cloud computing," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, 2015, pp. 402–415.

[11] IPFire. (2017). [Online]. Available: http://www.ipfire.org/

[12] D. Joseph and I. Stoica, "Modeling middleboxes," *IEEE Netw.*, vol. 22, no. 5, pp. 20–25, Sep./Oct. 2008.

[13] A. Krishnamurthy, S. P. Chandrabose, and A. Gember-Jacobson, "Pratyaastha: An efficient elastic distributed SDN control plane," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 133–138

[14] D. Krishnan and M. Chatterjee, "An adaptive distributed intrusion detection system for cloud computing framework," in *Recent Trends in Computer Networks and Distributed Systems Security*, Berlin, Germany: Springer, 2012, pp. 466–473.

[15] C. Lan, J. Sherry, R. A. Popa, S. Ratnasamy, and Z. Liu, "Embark: Securely outsourcing middleboxes to the cloud," in *Proc. 11th USENIX Conf. Netw. Syst. Des. Implementation*, 2016, pp. 255–273.

[16] C. Li, A. Raghunathan, and N. K. Jha, "A trusted virtual machine in an untrusted management environment," *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, pp. 472–483, Oct.–Dec. 2012.

[17] H. Li, M. Dong, X. Liao, and H. Jin, "Deduplication-based energy efficient storage system in cloud environment," *Comput. J.*, vol. 58, no. 6, pp. 1373–1383, 2014.

[18] H. Li, M. Dong, K. Ota, and M. Guo, "Pricing and repurchasing for big data processing in multi-clouds," *IEEE Trans. Emerg. Topics Comput.*, vol. 4, no. 2, pp. 266–277, Apr.–Jun. 2016.

[19] C.-H. Lin, C.-W. Tien, and H.-K. Pao, "Efficient and effective NIDS for cloud virtualization environment," in *Proc. IEEE 4th Int. Conf. Cloud Comput. Technol. Sci.*, 2012, pp. 249–254.

[20] Y. L. Liu, X. Y. Bai, G. Chen, et al., "Policy-based runtime performance evaluation and validation of web services," *Acta Electronica Sinica*, vol. 38, no. 2A, pp. 182–187, 2010.

[21] M. R. Lyu and L. K. Y. Lau, "Firewall security: Policies, testing and performance evaluation," in *Proc. 24th Annu. Int. Comput. Softw. Appl. Conf.*, 2000, pp. 116–121.

[22] Y. Ma, L. Wang, A. Y. Zomaya, D. Chen, and R. Ranjan, "Task-tree based large-scale mosaicking for massive remote sensed imageries with dynamic DAG scheduling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 2126–2137, Aug. 2014.

[23] N. McKeown, et al., "Openflow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[24] M. Menzel, R. Ranjan, L. Wang, S. U. Khan, and J. Chen, "Cloudgenius: A hybrid decision support method for automating the migration of web application clusters to public clouds," *IEEE Trans. Comput.*, vol. 64, no. 5, pp. 1336–1348, May. 2015.

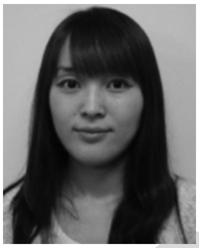
[25] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in cloud," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 42–57, 2013.

[26] ModSecurity. [Online]. Available: http://www.modsecurity.org/

[27] A. Mohammed, S. Sama, and M. Mohammed, *Enhancing Network Security in Linux Environment*," PhD thesis, Halmstad University, Halmstad, Sweden, 2012.

[28] A. Nguyen, H. Raj, S. Rayanchu, S. Saroiu, and A. Wolman, “Delusional boot: Securing hypervisors without massive re-engineering,” in *Proc. 7th ACM Eur. Conf. Comput. Syst.*, 2012, pp. 141–154.
[29] NVD. (2018). [Online]. Available: http://nvd.nist.gov/
[30] Huawei Security. (2017). [Online]. Available: http://e.huawei.com/en/products/enterprisenetworking/security
[31] IXIA. (2017). [Online]. Available: http://www.ixiacom.com/
[32] OpenSSL. (2018). [Online]. Available: http://www.openssl.org/
[33] M. Pearce, S. Zeadally, and R. Hunt, “Virtualization: Issues, security threats, and solutions,” *ACM Comput. Sur.*, vol. 45, no. 2, 2013, Art. no. 17.
[34] McAfee SaaS Email Protection and Web Protection. (2017). [Online]. Available: http://www.mcafee.com/us/products/security-as-a-service/index.aspx
[35] Z. A. Qazi, C. C. Tu, L. Chiang, et al., “Simple-fying middlebox policy enforcement using SDN,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 27–38, 2013.
[36] H. Raj and K. Schwan, “High performance and scalable I/O virtualization via self-virtualized devices,” in *Proc. 16th Int. Symp. High Performance Distrib. Comput.*, 2007, pp. 179–188.
[37] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield, “Split/merge: System support for elastic execution in virtual middleboxes,” in *Proc. 10th USENIX Conf. Netw. Syst. Des. Implementation*, 2013, pp. 227–240.
[38] K. Salah, J. M. Alcaraz Calero, S. Zeadally, S. Al-Mulla, and M. Alzaabi, “Using cloud computing to implement a security overlay network,” *IEEE Sec. Privacy*, vol. 11, no. 1, pp. 44–53, Jan./Feb. 2013.
[39] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, “Design and implementation of a consolidated middlebox architecture,” in *Proc. 9th USENIX Conf. Netw. Syst. Des. Implementation*, 2012, Art. no. 24.
[40] V. Sekar, S. Ratnasamy, M. K. Reiter, N. Egi, and G. Shi, “The middlebox manifesto: Enabling innovation in middlebox deployment,” in *Proc. 10th ACM Workshop Hot Topics Netw.*, 2011, pp. 21.
[41] Imperva Cloud DDos Protection Service. (2016). [Online]. Available: http://www.imperva.com/products/wsc_cloud-ddos-protection-service.html
[42] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, “Making middleboxes someone else’s problem: Network processing as a cloud service,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 13–24, 2012.
[43] S. Shin and G. Gu, “Cloudwatcher: Network security monitoring using openflow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?)” in *Proc. 20th IEEE Int. Conf. Netw. Protocols*, 2012, pp. 1–6.
[44] SpamAssassin. (2017). [Online]. Available: http://spamassassin.apache.org/
[45] A. Cilardo, L. Coppolino, A. Mazzeo, and L. Romano, “Performance evaluation of security services: An experimental approach,” in *Proc. 5th EUROMICRO Int. Conf. Parallel, Distrib. Net.-Based Process. (PDP’07)*, Feb. 2007, pp. 387–394.
[46] S. Subashini and V. Kavitha, “A survey on security issues in service delivery models of cloud computing,” *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 1–11, 2011.
[47] J. Szefer and R. B. Lee, “A case for hardware protection of guest VMs from compromised hypervisors in cloud computing,” in *Proc. 2nd Int. Workshop Sec. Privacy Cloud Comput.*, Jun. 2011, pp. 248–252..
[48] J. Szefer and R. B. Lee, “Architectural support for hypervisor-secure virtualization,” *SIGARCH Comput. Archit. News*, vol. 40, no. 1, pp. 437–450, Mar. 2012.
[49] K. Vieira, A. Schulter, C. B. Westphall, and C. M. Westphall, “Intrusion detection for grid and cloud computing,” *Professional*, vol. 12, no. 4, pp. 38–43, 2010.
[50] H. Wu, Y. Ding, C. Winer, and L. Yao, “Network security for virtual machine in cloud computing,” in *Proc. 5th Int. Conf. Comput. Sci. Convergence Inf. Technolo.*, 2010, pp. 18–21.
[51] X. Yuan, X. Wang, J. Lin, and C. Wang, “Privacy-preserving deep packet inspection in outsourced middleboxes,” in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[52] F. Zhang, J. Chen, H. Chen, and B. Zang, “Cloudvisor: Retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization,” in *Proc. 23rd ACM Symp. Operating Syst. Principles*, 2011, pp. 203–216.
[53] D. Zissis and D. Lekkas, “Addressing cloud computing security issues,” *Future Generation Comput. Syst.*, vol. 28, no. 3, pp. 583–592, 2012.

Jin He received the BS and MS degrees in computer science and technology from the University of Electronic Science and Technology of China (UESTC), in 1998 and 2005, respectively. He is currently working toward the PhD degree in the Department of Computer Science, UESTC. He was a senior engineer with Hua Wei, China, from 2005 to 2008. He leaded two teams (unified threat management team and Web application firewall team) as the director of research and development at link-trust Co. Ltd. from 2008 to 2011. His research interests include virtualization, operating system, and cloud security.



Kaoru Ota received the BS degree in computer science and engineering from the University of Aizu, Japan, in 2006, the MS degree in computer science from Oklahoma State University, in 2008, and the PhD degree in computer science and engineering from the University of Aizu, Japan, in 2012, respectively. She is currently an assistant professor in the Department of Information and Electronic Engineering, Muroran Institute of Technology, Japan. From March 2010 to March 2011, she was a visiting scholar with the University of Waterloo, Canada. Also she was a Japan Society of the Promotion of Science (JSPS) re-search fellow with Kato-Nishiyama Lab at Graduate School of Information Sciences at Tohoku University, Japan from April 2012 to April 2013. Her research interests include wireless networks, cloud computing, and cyber-physical systems. She has received best paper awards from ICA3PP 2014, GPC 2015, IEEE DASC 2015, and IEEE VTC 2016. She serves as an editor of the *IEEE Communications Letter*, the *Peer-to-Peer Networking and Applications* (Springer), *the Ad Hoc & Sensor Wireless Networks*, *the International Journal of Embedded Systems* (Inderscience), as well as a guest editor of the *IEEE Wireless Communications*, the *IEICE Transactions on Information and Systems*. She is currently a research scientist with A3 Foresight Program (2011-2016) funded by Japan Society for the Promotion of Sciences (JSPS), NSFC of China, and NRF of Korea. She is a member of the IEEE.

Mianxiong Dong received the BS, MS and PhD degrees in computer science and engineering from the University of Aizu, Japan. He is currently an associate professor in the Department of Information and Electronic Engineering, Muroran Institute of Technology, Japan. Prior to joining Muroran-IT, he was a Researcher at the National Institute of Information and Communications Technology (NICT), Japan. He was a JSPS research fellow in the School of Computer Science and Engineering, The University of Aizu, Japan, and was a visiting scholar with BBCR Group, University of Water-loo, Canada supported by JSPS Excellent Young Researcher Overseas Visit Program from April 2010 to August 2011. He was selected as a foreigner research fellow (a total of three recipients all over Japan) by NEC C&C Foundation in 2011. His research interests include wireless networks, cloud computing, and cyber-physical systems. He has received best paper awards from IEEE HPCC 2008, IEEE ICESS 2008, ICA3PP 2014, GPC 2015, IEEE DASC 2015, and IEEE VTC 2016. He serves as an editor of the *IEEE Communications Surveys and Tutorials*, *the IEEE Network*, *the IEEE Wireless Communications Letters*, *the IEEE Cloud Computing*, *IEEE Access*, and *the Cyber-Physical Systems* (Taylor & Francis), as well as a leading guest editor of the *ACM Transactions on Multimedia Computing*, *the Communications and Applications*, *the IEEE Transactions on Emerging Topics in Computing* (TETC), *the IEEE Transactions on Computational Social Systems* (TCSS), *the Peer-to-Peer Networking and Applications* (Springer) and Sensors. He has been serving as the program chair of IEEE SmartCity 2015 and Symposium Chair of IEEE GLOBECOM 2016, 2017. He is currently a research scientist with A3 Foresight Program (2011-2016) funded by Japan Society for the Promotion of Sciences (JSPS), NSFC of China, and NRF of Korea. He is a member of the IEEE.

Laurence T. Yang received the BE degree in computer dcience and technology from Tsinghua University, China, and the PhD degree in computer science from the University of Victoria, Canada. He is a professor in the Department of Computer Science, St. Francis Xavier University, Canada. His research interests include parallel and distributed computing, embedded and ubiquitous/pervasive computing, big data. His research has been supported by the National Sciences and Engineering Research Council, and the Canada Foundation for Innovation. He is a senior member of the IEEE.

Minyu Fan received the PhD degree in computer science from Xi'nan Jiao Tong University, in 1996, and was a visiting scholar with Queen University, United Kingdom, in 2005. She is a professor of computerscience at UESTC. Her research interests include operating systems, distributed systems, and systems security.

Guangwei Wang is a senior software engineer, and a teacher of Computer Scienceat UESTC. His research interests include information security and distributed systems.

Stephen S. Yau received the BS degree in electrical engineering from National Taiwan University, and the MS and PhD degrees in electrical engineering from the University of Illinois, Urbana-Champaign. He was with the University of Florida, Gainesville, and Northwestern University, Evanston, IL. He is currently a professor of Computer Science and Engineering and the director of the Information Assurance Center with Arizona State University, Tempe. His research interests include cyber trust, cloud computing, software engineering, service-based systems, and parallel and distributed computing systems. He was the president of the IEEE Computer Society and a member of the Board of Directors of the IEEE and the Computing Research Association, and is a fellow of the American Association for the Advancement of Science. He is a fellow of the IEEE.