Security Isolation Strategy Mechanism for Lightweight Virtualization Environment

Qian LIU[1], Xiao-Yong LI[1,a], Jian DU[2] and Gang LIU[3]

[1]*Beijing Jiaotong University, School of Computer and Information Technology, 100044, Beijing*
[2]*China Transport Telecommunications & Information Center, 100011, Beijing*
[3]*China Railway Corporation Transport Bureau Information Sector, 100844, Beijing*

Abstract. For cloud service providers, lightweight virtualization is a more economical way of virtualization. While the user is worried about the safety of applications and data of the container, due to the container sharing the underlying interface and the kernel, therefore the security and trusted degree of lightweight virtualization container isolation mechanism is critical for the promotion of lightweight virtualization service. Because the user cannot directly participate in the process of the construction and management of container isolation mechanism, it is difficult for them to establish confidence in the security and trusted degree of container isolation mechanism. Based on the research and analysis of system credible and virtualization isolation mechanism, this paper puts forward a set of lightweight virtualization security isolation strategy mechanism, divides lightweight virtualization container storage address space into several parts, puts forward the definition of lightweight virtualization security isolation, gives the formal description and proof of container security isolation strategy, and combines with related technology to verify the feasibility of lightweight virtualization security isolation strategy mechanism. The mechanism has important guiding significance for cloud services providers to deploy container security isolation.

1 Introduction

These Cloud computing generally adopt virtualization technology to support themselves, the traditional virtualization technologies include Xen hypervisor, VMWare, and Linux kernel Virtual Machine (KVM) and so on, these virtualization technology can achieve extension, control computing resources, and can securely isolate Virtual Machine (VM), while the resources they consume is large, and the performance and economic benefits they bring are relatively low. With the increased number of cloud users, the demand for physical resources is growing at an order of magnitude, so from the perspective of reducing the hardware cost, people hope for the appearance of a more economical virtualization solutions.

In recent years, operating system-level virtualization which based on the Linux Containers (LXC) technology is becoming more and more popular, such lightweight virtualization has many advantages compared to the traditional way of virtualization. First of all, container can start in the second level, it is much faster compared to the traditional way of virtualization. Second, container run on a shared operating system kernel, this allows multiple virtual environment to share the underlying operating

[a] Corresponding author: 14120354@bjtu.edu.cn

system interface and a public host kernel, therefore the system resource utilization is very high, and virtual machine density that single server can support is big, so as cloud service providers can have better profit space.

The problem lightweight virtualization bringing is users worrying about whether the strength of container isolation mechanism is safe enough, and whether or not they can ensure the security of user's application and data. So the method of lightweight virtualization technology attracting more users is to prove the effectiveness of their isolation, enabling users to establish enough confidence level for the services it provides.

At present domestic research, the lightweight virtualization field focus on implementation method, lacking of basic research such as container isolation mechanism security policies. This paper proposes a safety isolation strategy mechanism under lightweight virtualization environment with the help of lightweight virtualization containers technology and virtualization isolation mechanism, aims to solve the problem of container security isolation.

Innovation point of this article is: 1) based on the characteristics of lightweight virtualization, putting forward the address space division of lightweight virtualization storage resources and the definition of isolation; 2) adopting the trusted computing technology, turning it into the formal security isolation strategy, providing guidelines for developing container, and improving the user's confidence in lightweight virtualization technology.

In this paper, the first chapter introduces the research status of the related technology, including the current research status of the system trust and virtualization isolation mechanism; The second chapter embarks from dividing the address space of lightweight virtualization storage resources, researches lightweight virtualization security isolation strategy mechanism, proposes lightweight virtualization security isolation definition, formally describe container security isolation strategy; The third chapter certificates the strength of these container security isolation strategy; The forth chapter analyzes in detail the technical feasibility of above-mentioned lightweight virtualization security isolation strategy mechanism. the publisher.

2 Correlation Study

2.1 Trusted System

Trusted computing group described credible as "an entity' behavior is always carried out in accordance with the expected targets and the way", whether the system is trusted or not, it can guaranteed from two aspects: one is the system to act in accordance with the code of conduct, only within the scope of the code of conduct, the behavior of the system is considered credible. Second, conduct itself needs to meet the requirements of people's confidence level.

For the first assurance requirements, through the trust root and trust chain mentioned in the basic consider of the trusted compute to ensure that the system runs in accordance with the code of conduct, Trusted Computing Group (TCG)put forward the Trusted Computing platform [1-3][technical specification.

Technology needed by second assurance requirements is relatively a bit more complicated, such as through the precise formal verification [4]to describe behavior standards, such as ensure that the source (developed in-house or a third party)of the specification is transparent controllable[5-7]: if from developed in-house , the user can through process management, code size control methods to minimize or eliminate potential errors or bugs in code of conduct; if from a third party, the user can through other ways to evaluate the credibility of the code of conduct [8-10], such as the evaluation of others, such as solve the operation control issues of the user program with reference monitor [11,12]. System credible guarantee are shown in Figure 1 below:

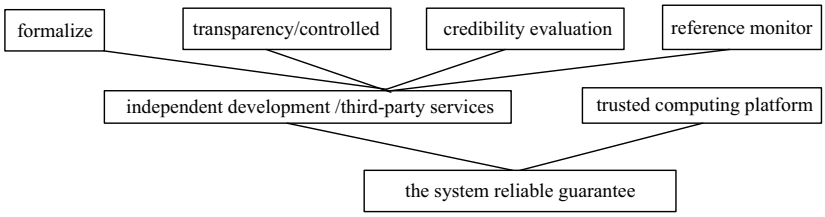

Figure 1. The system reliable guarantee.

2.2 Virtualization Isolation

Traditional virtualization is also called heavyweight virtualization, typical heavyweight virtualization methods includes KVM, Xen, VMWare, VirtualBox and other virtualization, etc. The machine which traditional virtual machines (VMs) running on is usually referred to as the host machine, the virtual machine is often referred to as the client. VMs are created and ran by the Virtual Machine Monitor (VMM) .The VMM has three basic characteristics: 1) the equivalence, the VMM provides an environment almost equaled with the host machine;2) the effectiveness, the program running in the virtual machine environment shows the performance of the minimum punishment;3) safety, VMM can completely control system resources.

In the early study of isolation, Rushby proposed "separation kernel" [13] on a single machine hosting multiple isolation environment to protect context. In 2008, on the basis of prior work experience, Rushby [14]summed up three kinds of the isolation technique: spatial separation, this kind of isolation is a direct implementation for resources which without sharing policy framework, each component of resource in physics are independent; Temporal isolation makes resources in a certain period of time only used by a component, wipes clean after using it, and then assigns them to another component, so always; Encryption isolation uses encryption signatures or calibration approach to strengthen the protection of reading and writing.

In the study of spatial segregation, Chen haibo et al. [15]of Fudan University used the nested virtualization technology to solve virtual security problem under the mega tenancy environment of cloud computing, putted forward the overall solution Cloudvisor, the main method of this mechanism was using the security protection of virtualization layer to achieve the isolation of resource management. In the study of temporal isolation, David Lie et al. [16] of Stanford university designed a kind of CPU architecture based on "the Execute - only the Memory" (XOM).In this kind of CPU architecture, data in cache and memory include tags used to bind data to a specific XOM isolated partition and MAC value for integrity protection .In the study of encryption isolation, Sahai et al.[17] proposed a called Attribute-based Encryption (ABE) Encryption access control method, which can provide fine-grained control for sharing data, but can only support threshold access control policy. To show more flexible access control policy, Bethencourt et al.[18]further putted forward the key-policy ABE (KP - ABE) and ciphertext-poicy ABE (CP - ABE) two types of ABE mechanism.

Lightweight virtualization is also called operating system-level virtualization, based on Linux Containers (LXC) technology. The earliest container technology can be traced back to the chroot tool series of Unix operating system. UNIX seventh edition in 1979 introduced the chroot (Change Root) mechanism. Chroot is a process making the specified directory as the root directory, all of its file system operation can only in the specified directory, so as not to harm the host system. Chroot existed the flaw like classic jump out of the holes, and it provided no segregation for any resources such as process and network .Early container implementation technologies also include the FreeBSD jail on FreeBSD operating system and Linux - VServer on GNU/Linux, and Sun Solaris Containers on the Solaris operating system. These techniques have been developed very mature, but do not have any of their containers integrated into the mainstream Linux kernel, therefore the LXC is currently the only operating-system virtualization technology for the Linux.

LXC was incorporated into the kernel thread from the beginning of Linux version 3.8, it packages what needed by the operation of the application into a container, such as the external environment, internal code, components and package dependency, allocates available hardware resources for different container through the common Application Programming Interface(API) and other tools, creates independent sandbox environment for running application, makes Linux users easily create and manage system or application container. By using Namespace and control group (Cgroups) function, LXC provides applications with an independent operating system environment.

In 1992, aiming to solve the problem of resource isolation based on the process, Namespace [19] have been proposed for the first time, it is an operating system level environment isolation method for Linux to achieve the lightweight virtualization (containers) service. Process under the same Namespace can sense the change of each other, and ignore the process's change from the external, so that the container in the process feels as if itself in an independent system environment, so as to achieve the goal of independence and isolation. Namespace like PID, NET, IPC, MNT, UTS and USER, segregate the process, the network, the message, the file systems, the UTS (" UNIX Time - sharing System ") and the USER space of the container.

3 Lightweight virtualization security isolation strategy mechanism

To simplify the problem, this paper hasn't take the above time isolation, encryption isolation into consideration, only focus on the study of lightweight virtualization storage resource address space isolation in the study of the lightweight virtualization space isolation .The host creates a new container meaning to assign it an independent storage resource address space.

3.1 Lightweight virtualization security isolation definition

Lightweight virtualization system includes two kinds of subject, respectively is: 1) the Host, the Host machine provides system services for containers, such as container disk read and write, network transmission, etc.;2) Containers, information cannot read and write between different containers.
Based on the structure of lightweight virtualization system, system storage address space can be divided into: 1) the host storage address space, which all belonged to the host machine.2) the container storage address space, each container has their own separate storage address space. Among them, based on access permission, the host storage address space can further be divided into three categories: the first kind of address space is closely related to all containers privacy protection, any container cannot read and write the content of this address space portion; The second type of address space has nothing to do with privacy of specific container, all containers can read the content of this portion, but can't write, such as public image resource information, etc.; The third type of address space is closely related to the privacy protection of particular container, in addition to the host machine and the container, other containers can't read and write, such as the container configuration information.

The letter S represents subject of the system, $S=\{Ho、Co_1、Co_2、\ldots、Co_n\}$, of which: Ho represents the host machine, $Co_i\ (1\le i\le n)$ represents container i. The letter L represents system storage address space, $L=\{H、C_1、C_2、\ldots、C_n\}$ denotes a division, of which: H denotes the storage address space corresponding to the host machine, $C_i\ (1\le i\le n)$ denotes the storage address space corresponding to the container i.

$H=\{H_x、H_{ro}、H_{1c}、H_{2c}、\ldots、H_{nc}\}$ represents a division of H, of which: H_x denotes the first kind of address space of H, any containers can not read or write on this part of the address space; H_{ro} denotes the second category of address space in H, the content of the address space for this part can be read by all containers, but no container can conduct write operation; $H_{ic}\ (1\le i\le n)$ denotes the third

type of address space, only the container i can read and write the contents of this part of the address space.

The above discussion towards the relation of system storage address space partitioning is shown in Figure 2.

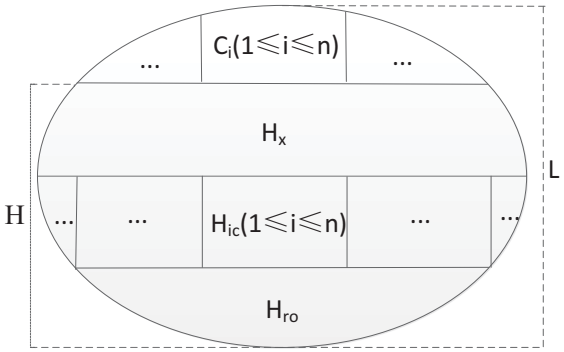

Figure 2. Lightweight virtualization storage address space division relations.

$R:S\to 2^L$ denotes the storage address space which the subject can read, $W:S\to 2^L$ denotes the storage address space which the subject can write.

In system, the direct information exchange between the two subject show up as a subject write information to the storage address space where another subject can read, for any two different subject s and t, if $R(s)\cap W(t)\neq\varnothing$, then t can pass information to s. Such as a container in the system through the system call request the host machine services, is through writing system call parameters, then these parameters are read by the host kernel so as to complete the service request.

Isolation between the two subject means there is no direct or indirect information exchange between them, but in this article, the container isolation we discussed is relatively weak on demand, because all containers need to exchange information with the host machine, to apply for the system service. Given this, lightweight virtualization system isolation requirements between the container is limited to no direct exchange of information between any two containers, for this reason, this paper gives a container isolation definition as follows:

Definition 1. In lightweight virtualization structure, security isolation between containers refers to:

For $\forall i,j,1\le i\le n,1\le j\le n,i\neq j$, have: $R(Co_i)\cap W(Co_j)=\varnothing$

Definition 1 refers to the meaning of no one container can directly read other containers address space, or they can be used to realize information exchange for both reading and writing.

3.2 Lightweight virtualization security isolation policy

Based on Figure 2 about lightweight virtualization storage address space partition method, this paper puts forward a set of security isolation rules, in order to realize the security isolation between different containers.

Rule 1: $\forall Co_i\left(1\le i\le n\right)\in S\Rightarrow R(Co_i)\in C_i\cup H_{ro}\cup H_{ic}$

Rule 2: $\forall Co_i\left(1\le i\le n\right)\in S\Rightarrow W(Co_i)\in C_i\cup H_{ic}$

Rule 1shows the address space that any containers $i\left(1\le i\le n\right)$ can read include the container storage address space corresponding to the container i itself $C_i\left(1\le i\le n\right)$,the address space H_{ro} in H

which can be read but can't be write by all containers, and the address space $H_{ic}\left(1 \le i \le n\right)$ in H which can only be read or write by the corresponding container.

Rule 2 shows the address space that any containers $i\left(1 \le i \le n\right)$ can write include the container storage address space corresponding to the container i itself $C_i\left(1 \le i \le n\right)$, and the address space $H_{ic}\left(1 \le i \le n\right)$ in H which can only be read or write by the corresponding container.

If a lightweight virtualization system adopt as shown in figure 2 storage address space partition method, and satisfy the rules 1 and 2, then the system's isolation between all the containers is security, namely for $\forall i, j, 1 \le i \le n, 1 \le j \le n, i \ne j,\ R\left(Co_i\right) \cap W\left(Co_j\right) = \left(C_i \cup H_{ro} \cup H_{ic}\right) \cap \left(C_j \cup H_{jc}\right) = \varnothing$.

4 Lightweight virtualization security isolation strategy mechanism security

As mentioned above, isolation strategy mechanism is guaranteeing different containers isolated in the host machine level, but the host machine will inevitably consciously or unconsciously run some unsafe applications, including the program code which containing trojans, worms and other malicious code, the vulnerability of the these unsafe applications may tamper with mechanism, and result a certain destruction to the effectiveness of the mechanism.

Basing on the analysis of the system safety mentioned above, this chapter will ensure the safety of isolation strategy mechanism from two aspects, on the one hand is the safety of isolation strategy mechanism itself, by using formal formula to prove that isolation strategy set is meeting the requirement of container security isolation definition; On the other hand is the security isolation strategy mechanism at run time, with the help of white list of trusted computing platform technology. The guarantee of mechanism safety is shown in Figure 3:

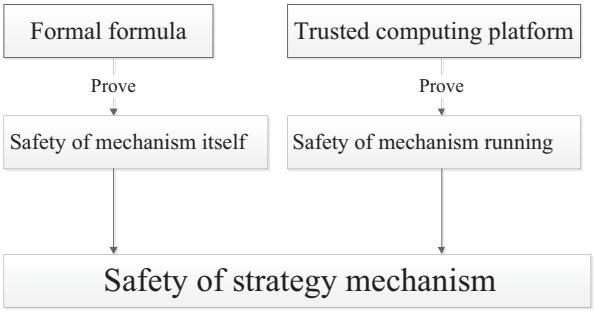

Figure 3. The guarantee of mechanism safety.

4.1 Safety of isolation strategy mechanism itself

As mentioned above, isolation strategy mechanism is guaranteeing different containers isolated in the host machine level, but the host machine will inevitably consciously or unconsciously run some unsafe applications, including the program code which containing trojans, worms and other malicious code, the vulnerability of the these unsafe applications may tamper with mechanism, and result a certain destruction to the effectiveness of the mechanism.

According to lightweight virtualization security policy, the readable storage address space of the container i is $R(Co_i)=\{C_i、H_{ro}、H_{ic}\}$, the writable storage address space of the container j is $W(Co_j)=\{C_j、H_{jc}\}$.

And by the system storage address space partition, we know $H.C_1.C_2.\ldots.C_n$ is a partition of the lightweight virtualization storage resource address space L, that is to say $L=H\cup C_1\cup\ldots\cup C_n$.We know $H_x.H_{ro}.H_{1c}.H_{2c}.\ldots.H_{nc}$ is a partition of the host storage address space H, that is $H=H_x\cup H_{ro}\cup H_{1c}\cup H_{2c}\cup\ldots\cup H_{nc}$.

According to the definition of division, there is no overlap between these address space to each other, there is no public read/write address space, so the intersection between any two address space among all the address space is an empty set.

That is $\forall l_1,l_2\in L,l_1\neq l_2\Rightarrow l_1\cap l_2=\varnothing$.

$$
\begin{aligned}
&R(Co_i)\cap W(Co_j)\\
&=(C_i\cup H_{ro}\cup H_{ic})\cap(C_j\cup H_{jc})\\
&=(C_i\cap C_j)\cup(C_i\cap H_{jc})\cup(H_{ro}\cap C_j)\\
&\cup(H_{ro}\cap H_{jc})\cup(H_{ic}\cap C_j)\cup(H_{ic}\cap H_{jc})\\
&=\varnothing\cup\varnothing\cup\varnothing\cup\varnothing\cup\varnothing\cup\varnothing\\
&=\varnothing
\end{aligned}
$$

To sum up, for $\forall i,j,1\le i\le n,1\le j\le n,i\neq j,\forall s\in S$, we have: $R(Co_i)\cap W(Co_j)=\varnothing$.

The above formally proofs lightweight security isolation strategy is credible. At the same time, the Trusted Platform technologies such as the Core Root of Trust for Measurement (CRTM) and Trusted Platform Module (TPM), etc., can guarantee the host machine itself runtime to be trusted.

4.2 Safety of isolation strategy mechanism at run time

In order to implement the isolation strategy mechanism, and guarantee mechanism not been tampered, which means the container subject can only access the address space in permissions. In this paper, the idea of the trusted application white List technology (Trusted Software List, TSL) in trusted computing platform is used to deal with the problem, the TSL in this paper including: all safe executable files in system and its integrity check values (hash value).

Once a subject launch the system call, according the system call information, the first step is to determine whether this executable file in TSL: if in, then go to the next step; Otherwise, return to ban. Second step is to compute the hash value of the executable file, and comparing with the credible hash value in the TSL: if same, return to allow, continue system call to normal execution; Otherwise, return to ban, system call failed.

TSL = {E, H}denotes a credible white list in the host machine system, of which E = {e_1, e_2,..., e_n} said executable file collection in TSL, H = {h_1, h_2,..., h_n} said the integrity of the calibration value of collection corresponding to TSL executable file. With function In (E) = {1, 0} said whether an executable file in TSL: if in, then return to 1 on the result, can go to the next phase of judgment; If not, then the result return 0, calls are forbidden to perform. Function (E, H) = {1, 0} said whether an executable file hash value is consistently with the hash values corresponding to TSL, if can the result return 1, the system calls continue normal execution; if can't the result return zero, the system call failed.

The process discussed above is shown as follows:

if:

$\forall e_1 \in E,\ In(e_1) = 0 \Rightarrow return\ 0;$

else if:

$\forall e_1 \in E,\ \forall h_1 \in H,\ Equal(e_1, h_1) = 0 \Rightarrow return\ 0;$

else:

return 1.

This method can realize the effective operation of isolation strategy mechanism, on the premise of security, and ensure the isolation strategy mechanism of unity between safety and efficacy, to enhance the user's confidence in lightweight virtualization technology.

5 Lightweight Virtualization Security Isolation Strategy Mechanism Feasibility

According to the characteristics of the lightweight virtualization, security isolation strategy mechanism proposed in this paper can be realized by Namespace and Copy-On-Wirte(COW) technology. Namespace is an operating system level environmental isolation method provided by Linux, in order to realize the lightweight virtualization services. COW is only in the process of paragraphs in the space content will change, will be a copy of the contents of the parent for the child process technology.

Of which PID is used to isolate the process of different containers; NET makes the container has an independent Network Devices, IP Addresses, IP Routing Tables, /proc/net directory, that each container network can be isolated; MNT makes the process of different Namespace see file system is different, so that the file system saw by process in different Namespace is isolated from each other.

The biggest difference between COW file system and other systems is that: COW never overwrite the existing content of the file system. COW makes container to read or write its own address space rather than to read-only storage address space of a host computer, can be used to separate the container storage address space and the hosting read-only storage address space. COW will merge two address space (the hosting read-only storage address space and the container storage address space) into one, therefore the angle of view on the container is all content of combined address space. If the container needs to update its perspective /etc/hosts file, the file happens to be contents of the hosting read-only storage address space, COW will firstly not overwrite /etc/hosts file on the read-only storage address space in the host machine, secondly it will copy the file to the container storage address space, namely copy to/etc/hosts file on the container storage address space , and then conduct update operation on the latter .In this way, even if the host machine read-only storage address space and the container storage address space both have /etc/hosts file ,and COW can ensure that container only read or write contents of /etc/hosts file on the container storage address space , namely the updated content.

Acknowledgement

In this paper, the research on security isolation under lightweight virtualization environment involves multiple levels, on the basis of the lightweight virtualization environment storage address space division, proposed lightweight virtualization security isolation definition, formally described security isolation strategy, proved that lightweight virtualization security isolation strategy for security isolation between the container is credible, and combined with related technical to verify lightweight virtualization security isolation strategy mechanism this paper proposed is feasible. The significance of this paper is to provide system with an isolation strategy mechanism, deploy a system without this isolation strategy mechanism may exist a potential safety hazard like covert channel. And the method

of translating the abstract container isolation credible and feasible requirements into specific formal rules is a kind of innovation, provides a reference for future similar research.

References

1. Perez R, Sailer R, van Doorn L. vTPM: virtualizing the trusted platform module[C]. Proc 15th Conf on USENIX Security Symposium; 2006; 2006. p. 305-20.
2. England P, Loeser J. Para-virtualized TPM sharing[C]. Trusted Computing-Challenges and Applications: Springer; 2008: 119-32.
3. Stumpf F, Eckert C. Enhancing trusted platform modules with hardware-based virtualization techniques[C]. Emerging Security Information, Systems and Technologies, 2008 SECURWARE'08 Second International Conference on; 2008: IEEE; 2008. p. 1-9.
4. Greve D, Wilding M, Vanfleet WM. A separation kernel formal security policy[C]. Proc Fourth International Workshop on the ACL2 Theorem Prover and Its Applications; 2003: Citeseer; 2003.
5. Pauley WA. Cloud provider transparency: an empirical evaluation [J]. Security & Privacy, IEEE 2010; 8(6): 32-9.
6. Whaiduzzaman M, Gani A. Measuring security for cloud service provider: A Third Party approach[C]. Electrical Information and Communication Technology (EICT), 2013 International Conference on; 2014: IEEE; 2014. p. 1-6.
7. Wüllenweber K, Weitzel T. An empirical exploration of how process standardization reduces outsourcing risks[C]. System Sciences, 2007 HICSS 2007 40th Annual Hawaii International Conference on; 2007: IEEE; 2007. p. 240c-c.
8. Chakraborty S, Roy K. An SLA-based framework for estimating trustworthiness of a cloud[C]. Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on; 2012: IEEE; 2012. p. 937-42.
9. Zadeh LA. Probability measures of fuzzy events [J]. Journal of mathematical analysis and applications 1968; 23(2): 421-7.
10. Shapley LS. A value for n-person games[R]: DTIC Document, 1952.
11. Anderson JP. Computer Security Technology Planning Study. Volume 2[R]: DTIC Document, 1972.
12. Peter Loscocco N. Integrating flexible support for security policies into the Linux operating system[C]. Proceedings of the FREENIX track: USENIX Annual Technical Conference; 2001; 2001.
13. Rushby JM. Design and verification of secure systems [M]: ACM; 1981.
14. Rushby J. Separation and integration in mils (the mils constitution)[J]. Computer Science Laboratory SRI International, Technical Report 2008.
15. Zhang F, Chen J, Chen H, Zang B. CloudVisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization[C]. Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles; 2011: ACM; 2011. p. 203-16.
16. Lie D, Thekkath C, Mitchell M, et al. Architectural support for copy and tamper resistant software [J]. ACM SIGPLAN Notices 2000; 35(11): 168-77.
17. Sahai A, Waters B. Fuzzy identity-based encryption[C]. Advances in Cryptology–EUROCRYPT 2005: Springer; 2005: 457-73.
18. Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption[C]. Security and Privacy, 2007 SP'07 IEEE Symposium on; 2007: IEEE; 2007. p. 321-34.
19. Pike R, Presotto D, Thompson K, Trickey H, Winterbottom P. The use of name spaces in Plan 9[C]. Proceedings of the 5th workshop on ACM SIGOPS European workshop: Models and paradigms for distributed systems structuring; 1992: ACM; 1992. p. 1-5.