Research on Security Algorithm of Virtual Machine Live Migration for KVM Virtualization System

Wei Fan[1], Zhujun Zhang[1](✉), Tingting Wang[1], Bo Hu[1], Sihan Qing[1,2,3], and Degang Sun[1]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{fanwei, zhangzhujun, wangtingting9071, hubo, qsihan, sundegang}@iie.ac.cn
[2] Institute of Software, Chinese Academy of Sciences, Beijing, China
[3] School of Software and Microelectronics, Peking University, Beijing, China

Abstract. Live migration of virtual machine is the process of moving VMs from one physical server to another server keeping services running in VMs, and facilitates load balancing, energy saving, hardware dependent, remote migration and so on. This novel technology brings a huge convenience, and also presents new security challenges that the security concern is the major factor effecting this technology widely adopted in IT industry. Live migration exposes VM's data as plaintext to the network as a result of vulnerabilities in the migration protocol. The traditional protection way is using the SSL protocol, but that consume too much time and not as safe as it used to be, few users adopt this way. So we design a security algorithm based original migration algorithm making up for the lack of security. In this paper, firstly, we analyze and verify security threats to live migration. Secondly, through the analysis on the live migration mechanism, the bottom driver, and the source code of KVM virtualization system, we design a security algorithm for live migration to meet the security needs of different users. Thirdly, the new security algorithm which we innovatively add three functions to the original algorithm to ensure migration data to remain confidential and unmodified during the transmission. The security algorithm make up the security vulnerabilities of original migration mechanism and take less time than the SSL. Finally, a series of experiments validate the algorithm that could solve the balance of the security and performance in live migration process.

Keywords: Live migration · Security threats · Security algorithm · KVM virtualization system

1 Introduction

Cloud computing is increasingly assuming a prominent and leading role in businesses for the purpose of operational efficiency and cost reduction. As the foundations of cloud computing, virtualization allows many OS instances to run concurrently on a single physical machine with high performance, providing better use of physical resources and isolating individual OS instances [1]. It has attracted considerable interest in recent years, particularity from the data center and cluster computing communities [2]. It consolidates many physical servers into a single physical server saving the hardware resources, physical space, power consumption, air conditioning capacity and man power to manage the servers [3, 4]. VM (Virtual machine) migration means to move a VM from one host to another. The migration of virtual machine is divided into two types, Static Migration and Live Migration.

Static Migration is the process of virtual machine in shutdown or suspended state from one physical server to another physical server.

Live Migration is the transition of a running VM from one physical server to another without halting the VM. Provided the service uninterruptedly is a key requirement to many applications, live migration is usually used to achieve load balancing, energy efficiency, and easy hardware maintenances. In spite of the numerous benefits, users remain anxious about migration security and data protection over time [5]. There are many security problems in live migration process, one of which is that the VM data as plaintext could be sniffed easily during the migration [6, 7]. Because of security concerns, banking, government and national defense hesitate to make use of live migration. The generally way to solve this problem is using the SSL protocol, in addition to that takes too much time and is not as safe as it used to be, then users seldom adopt this way. So how to ensure data security of VM during live migration is the main topic of this paper.

We innovatively propose a security algorithm to guarantee the security of live migration for the KVM (Kernel-based Virtual Machine) virtualization platform. It has following four characteristics:

(1) It is designed based on KVM source code, instead of using cryptographic protocols to provide communication security over network, promoting the development of secure live migration mechanism.
(2) It narrows the encoded data range, only encoding users' sensitive data, reducing the consumption in secure live migration process.
(3) It is implemented by three functions, the special highlight is that the security function using different encoding algorithms to meet the security and performance requirements for different users.
(4) It is satisfactory to security and performance.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 presents the security algorithm in detail. Section 4 verifies the security algorithm from security and performance. Section 5 concludes our work.

2 Related Work

2.1 Security Threats in Live Migration

Most of the virtualization technologies now support live migration, such as Xen, KVM, VMware's VMotion. Unfortunately, they all have vulnerabilities in live migration process. There are approaches from both academia and industry that cover insecure live migration from different perceptions. Jon Oberheide et al. [8] in 2008 analyzed live migration threats from three layers, VMM (Virtual Machine Monitor) control layer, data layer and migration module layer. For example, an attacker could gain access to the transmission channel using techniques such as ARP/DHCP poisoning, DNS poisoning and IP/route hijaking to perform passive or active attacks. He designed the Xensploit tool which could automatically or manually to manipulate data in the Man-in-the-middle attack during a live VM migration process. This tool tampered with system's memory data of migrated VM, verifying that these attack strategies could be exploited in the XEN and VMware virtualization platform. Ms. Yamuna Devi. L researched on security in VM live migration and implemented live migration experiments in the KVM virtualization platform [9], which was also easy for attackers to hijack the live migrate process or hypervisor where these migrations occur. But there is no concrete implementation process to confirm this conclusion. Fan Wei [10] in 2014 ever captured memory data on live migration further proves the existence of security threats only in XEN and VMware Virtualization platform. As a kind of special information assets in the computer system, VM in security problems of live migration could be summarized as the following three aspects:

- Insecure communication channel

One of the VM migration protocol vulnerabilities is that migration data is plaintext over the network. If attacker was monitoring transmission channel, migration data would be accessed or even modified. By listening to the network between the source and the target server, the attacker could get the user's application data, user's password and other sensitive information [11]. Attackers also could modify the VM memory to specific data making the virtual machine under their control [12].

- Lack of access control strategies

An inappropriate access control strategy allows an unauthorized user to initiate, migrate and terminate a virtual machine. The attacker could initiate large numbers of outgoing migrations onto a legitimate virtualized host server [13], consuming server resources, decreasing its performance and even causing denial of service. Attackers also could transfer a VM with malware, Trojan horses or malicious code, to attack target server or other VMs on the target server. Attackers also could cause VMs to migrate from one server to another affecting the normal operation of VM or transfer a VM to an unauthentic host [14, 15].

- Vulnerabilities in virtualization software

There are vulnerabilities as stack, heap and integer overflows in the virtualization platform as common software [16–18]. Such vulnerabilities provide attackers the opportunity to inject malicious code breaking confidentiality, integrity and availability of other VM's code or data. Once the attacker successfully gains access to hypervisor through exploiting vulnerabilities, then the attacker will take control of the migration of VM.

2.2 Simulation Attack Experiments

- The attacker model

We assume a realistic attacker model where an attacker not only has access to network data but could also modify or inject messages into the network. We also assume that he is computationally bounded and hence, brute force attacks on cryptographic schemes are difficult. We still assume that the attacker does not have physical access to platforms between which the migration occurs.

- Attack Principle

The VM data must be transferred through network, and as this paper mentioned before, the transmission channel is insecurely, the migration functionality exposes the entire machine state of VM to device module which listens to the incoming live migration requests from remote platforms. So we assume that attackers could monitor the transmission channel to get sensitive data and modify the transmitted data. This way is not only easy to implement but also the attacker frequently used, so we set port mirroring on the switch which is the bridge of the two connected host to eavesdrop on sensitive data of VM, simulating attack process and verifying the threat of leaking sensitive information during live migration process.

- Experimental Design

The experiment is designed as Fig. 1, the same CPU type hosts both running QEMU-KVM released 1.5.3. Guest 1, Guest 2 and Guest 3 are the VM. Host1 and Host2 both are the Lenovo K4450 with Intel(R) Core i7 CPU and 8G RAM. The switch is H3C S1526. Because storage migration requires a lot of time, we migrate Guest 2 based on NFS shared storage. Only memory data and CPU status need to be transferred from the host1 to the host2. We use the software to sniff the whole transport channel.

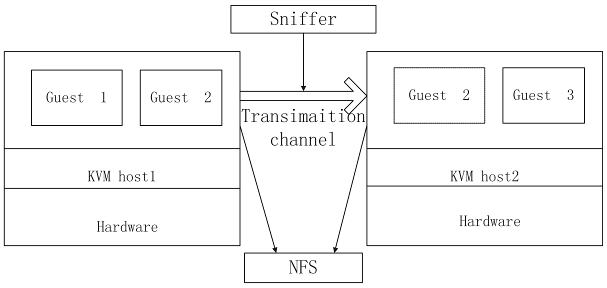


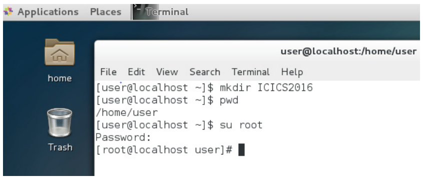
Fig. 1. Experimental structure.

- Results and Analysis

Scenario 1: Start a virtual machine configured for Linux OS, log into the system with the ordinary user "iie", and switch the user of the system with the "root" user, the password is “centospassword”. Then migrating this VM to another host. Figure 2 shows the state of VM before migration. Figure 3 shows the analysis of the captured data packets.

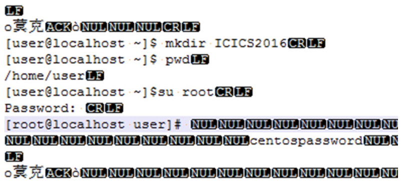
Figures 2 and 3 show that the Linux root user’s password is quite dangerous on the condition of VM being migrated, which means that attackers could get the root password of the Linux system during the live migration as well as the general user’s password. Attackers could use password to do illegal things causing more serious security risks.

Scenario 2: Open the Notepad software of Windows OS, input some characters and numbers, and save the file and start migrating.


Applications
Places
Terminal
home
Trash
user@localhost:/home/user
File Edit View Search Terminal Help

```
[user@localhost ~]$ mkdir ICICS2016
[user@localhost ~]$ pwd
/home/user
[user@localhost ~]$ su root
Password:
[root@localhost user]#
```

Fig. 2. The VM state no. 1

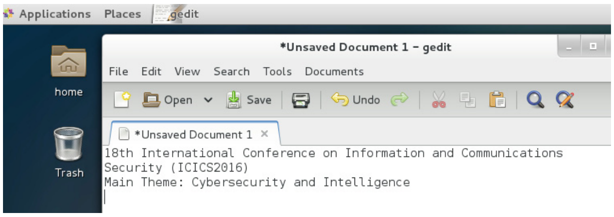


```
LF
o莫克ACKòNULNULNULCRLF
[user@localhost ~]$ mkdir ICICS2016CRLF
[user@localhost ~]$ pwdLF
/home/userLF
[user@localhost ~]$su rootCRLF
Password: CRLF
[root@localhost user]# NULNULNULNULNULNULNULNU
NULNULNULNULNULNULNULNULNULcentospasswordNULN
LF
o莫克ACKòNULNULNULNULNULNULNULNULNULNULNULNU
```

Fig. 3. The analysis of the captured data packets no. 1.

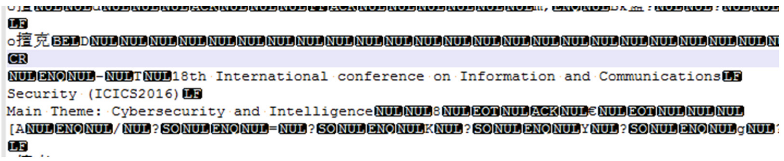
From Figs. 4 and 5, we could see that attackers could get all the data being used by applications of VM during the live migration.

The analyses imply that the vulnerability of live migration do exist which attackers could get sensitive data of VM during live migration process whatever the OS of VM.


Applications
Places
gedit
home
Trash
*Unsaved Document 1 – gedit
File Edit View Search Tools Documents
Open
Save
Undo
*Unsaved Document 1

```
18th International Conference on Information and Communications
Security (ICICS2016)
Main Theme: Cybersecurity and Intelligence
```

Fig. 4. The VM state no. 2.



```
LF
o揎克BELDNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULN
CR
NULENQNUL-NULTNUL18th International conference on Information and CommunicationsLF
Security (ICICS2016)LF
Main Theme: Cybersecurity and IntelligenceNULNUL8NULEOTNULACKNUL€NULEOTNULNULNUL
[ANULENQNUL/NUL?SONULENQNUL=NUL?SONULENQNULKNUL?SONULENQNULYNUL?SONULENQNULgNUL
LF
```

Fig. 5. The analysis of the captured data packets no. 2.

3 Proposed the Security Algorithm

The SSL protocol is adopted to protect VM data security during live migration process. But this way requires a lot of time and not applied to high service requirements conditions. Besides, many vulnerabilities in SSL have been found [19], which means that it is no longer secure as before. In order to improve the security of migration mechanism, and achieve the balance of performance and security, we extraordinarily present a security algorithm based on the source code.

We analyze codes and algorithm about live migration for QEMU-KVM released 1.5.3., which could be summarized as follows:

Stage 0: Pre-migration. There is an active VM on source physical host A. The source informs destination B to start reserving resources.
Stage 1: Iterative Pre-copy. During the first iteration, all pages are set dirty and transferred from A to B. Subsequent iterations only copy those pages dirtied during the previous transfer phase.
Stage 2: Stop-and-Copy. In this phase, CPU state and any remaining inconsistent memory pages are then transferred. At the end of this stage there is a consistent suspended copy of the VM at both A and B. The VM related network is redirected to target B through unsolicited ARP reply adverting.

3.1 Security Algorithms Design

- Performance improvement

The measurement of virtual machine migration efficiency is generally from the following several aspects:

(1) Total time: the time required of VM migrating from the source host to the destination host and resuming it.
(2) Service downtime: VM on the source host or on the destination host is out of service in the migration process. During this time VM on the source host has to stop the service, while VM on the destination host has not been restored.
(3) The impact on the performance of services: contains the performance of the VM application, and the performance of other services (or other VMs) on the host server during the migration process.

The most fundamental factor influencing migration efficiency is the amount of data to be transmitted. The SSL protocol is the only existing way for migration protection that all the VM data must be encrypted, which greatly increases the migration time.

Memory data is the most important part of transferred data. In order to resist the threat of insecure communication channel and reduce time cost, we only choose to protect VMs' sensitive memory data. The memory of Linux operating system includes kernel space and user space, while the memory of windows operating system includes user space and system space. It can be distinguishing from each other by the characteristic of memory. The contents of kernel memory are always the same system information in different VM. However, privacy data currently being processed by the user is within the user memory space, so we especially focus on the confidentiality of the user space data and the integrity of the system space data, reducing the time cost in secure live migration, ensuring the security of the private data at the same time.

Hypervisor, also known as VMM, manages the several virtual machines placed on a single hardware [20], could access all physical devices on the server. It is responsible for creating the virtual environment on which the guest virtual machines operate. It supervises the guest systems and makes sure resources are allocated to the guests as necessary [21]. All guest software (including the guest OS) runs in user mode; only the VMM runs in the most privileged level (kernel mode) [22]. Due to the particularity of its role, it is not only the favorite attack target, but also the important part that security professionals adopt a lot of defensive measures to protect. If this part was not secure, the security of VMs or other service could not be guaranteed. In another word, VMM could be considered the most secure part of virtual system. All the VM relevant data must go through VMM during the VM migration. If encryption is done at the VMM level, there would be less overhead, less downtime [23]. So the design of security algorithms is based on the secure VMM that security professionals have guaranteed its security from other aspects. The monitoring function is designed to monitor the memory data needing to be protected, then call the security function to encode.

- Security improvement

This part is the most important part of the security algorithm. In order to prevent attackers from illegally getting plaintext of VM and modifying on purpose, we present security algorithm applied in source host and destination host from confidentiality and integrity:

- **Algorithm principle**

In order to allow the attacker not to get sensitive data, and ensure the speed of the migration, we design an encoding algorithms for VM user space and also adopt timestamp mechanism to avoid reply attack.

The RC4 algorithm is a typical stream cipher based on nonlinear transform of array, applied on the SSL protocol to protect Internet information flow. It is based on Key-scheduling algorithm (KSA) and Pseudo-random generation algorithm (PRGA). The secret key length is within 1–256 bytes, then the possibility of the secret key is $256 + 256^2 + 256^3 + \ldots\ 256^{256} \approx 256^{256}$ kind's possibilities more than 10^{600}. Its simplicity and speed make it more suitable for live migration. However, the fatal weakness of RC4, which is that secret key being used for a long time, may cause attackers frequency analysis and crack, so we propose this new algorithm combined with the RC4 using different seed secret keys in each iteration of live migration. Assuming that each VM migration needs n times memory iterations, this algorithm makes the source data have 2^{2048n} conversion forms. As Fig. 6 shows the migration iteration timeline, Round n is the nth of migration iteration, State 1 is the memory state of Round n, Round n + 1 is the (n + 1)th of migration iteration, State 2 is the memory state of Round n + 1, ③page and ④page are the new dirty pages during Round nth iteration. At this moment, this algorithm checks whether ③page and ④page belong to user's space. If any pages belong to user's space, secret key function we designed will generate seed key to encode these pages with RC4 as Fig. 7 shows. So are the other iterations. The Key in Fig. 7 is the seed secret key generated in each iterations by secret key function. This algorithm make use of IV (Initialization Vector) and Key generated Stream by PRNG (Pseudo Random Noise Generation). Using multiple keys in each migration destroys the law of encoded data, and makes the attacker analyze and crack more difficult. Moreover it narrows down the field of data need to be encoded, making it is remarkable that this algorithm reaches a high level of security and takes shorter time than using SSL.

The encryption process can be concluded in three steps as follows:

a. Calculating the checksum: set message as M, CRC (Cyclical Redundancy Check) checksum for M as C (M), get the plaintext P = <M, C (M)>;
b. Data encryption: set the initial vector as v, key as k, the key sequence as RC4 (v, k), get cipher text C = P ⊕ RC4 (v, k);
c. Data transmission: in the end, the IV and cipher text transmitted through net. Figure 8 shows the specific encryption process.

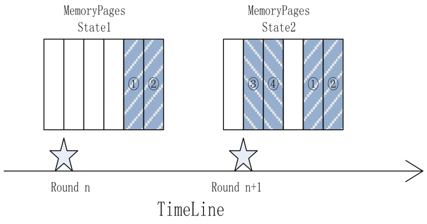


Fig. 6. The migration iteration TimeLine.

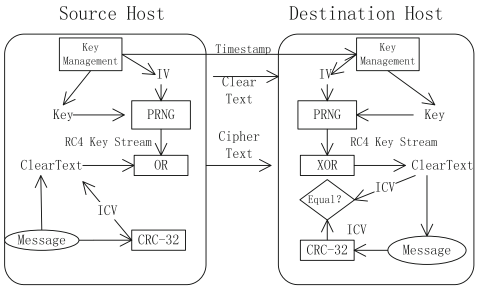


Fig. 7. The entire algorithm implementation process.

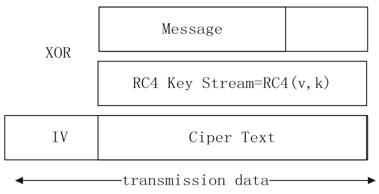


Fig. 8. The specific encryption process.

The decryption process can be concluded in two steps as follows:

a. The decryption process is the reverse of encryption process. The receiver generates key according to the received Timestamp. Then, the receiver could generate the same key sequence RC4(v,k). The cipher text make XOR operation with RC4(v,k) getting the plaintext.

b. Integrity checking. Decomposing P into <M, C>, recalculate the checksum C (M)' compared with the received C(M). Only are they equal to each other, this data frame can be regarded as effective, therefore ensure the integrity of the data frame.

- **Confidentiality**

To ensure the receiver to generate the same seed secret key, the secret key function in source host generates random numbers and public key according to the current time firstly. Random numbers are used as the seed secret key for encoding and public key is used for encoding the hash value. Then source host sends this time as Timestamp to the destination host, if the time is acceptable, the destination host would generate random numbers and public key based on the received time, because of the same secret key generation mechanism, destination host could generate same random numbers and public key for decoding. Similarly, for the next iteration, source host and destination host generates the same secret key according to their current time. Therefore, live migration uses different key in each iteration ensuring VM data to have higher confidentiality.

- **Integrity**

To prevent the VM data including user space and system space from being modified, we add CRC to prevent the migration data from being modified. The dirty bitmap is used to mark dirty page in each iteration of live migration. In the integrity mechanism, we present the mistake bitmap which mainly be used to mark the memory pages that the attacker has changed or destroyed in live migration. Source host calculates data hash value, uses public key to encode, and the destination host uses public key to decode the hash value confirming data integrity, if data had been modified, destination host would send to the source host the mistake signal including the modified data position. Once source host received the mistake signal, the source host would mark on relative position of mistake bitmap. The VM memory, often be rewriting and destroyed by the attacker may be dirty in the next iteration. The source host resending such dirty memory pages will become meaningless and waste time, so we design that all the destroyed memory pages are send to destination host at last round. At stop-and-copy phase, the source host migrates dirty pages according to the result of the dirty bitmap making OR operation with the mistake bitmap. If these memory pages are still tampered in the final round, VMM will inform the administrator that someone is trying to modify migration data, waiting administrator determine the next step action.

3.2 Main Algorithm Functions

This secure algorithm is mainly implemented through adding three functions to the traditional migration mechanism, called monitor function, secret key function and security function.

- Monitor Function

This function mainly narrows the range of VM data needed to encode or decode by monitoring the transmitted data. In source host, this function calculate the VM data including user space and system space, and monitor the data belonging to the user space. Once the data belong to the user space, it calls the security function. In destination host, this function monitors the received VM data whether have been encoded. When the data has been encoded, it calls the security function.

- Secret Key Function

This function mainly generates seed keys and public keys in each iteration. Time is one of the influencing factors for generating the seed secret key. The seed secret key is the core for encode and decode. In source host, security function uses seed secret keys for encoding VM data and its hash value. In destination host, security function uses this seed secret for decoding VM data and its hash value.

- Security Function

This function mainly uses optimized RC4 for encoding or decoding. In source host, this function encodes the calculated hash value and VM data, then sends encoded data and timestamp to destination. In destination host, this function compares the time of timestamp with current time and judges whether the received time belongs to the acceptable range. If time was illegal, destination host would stop migration process and send attacked signals. If time was legal, destination host would decode received data of user space and calculate all the received data hash value. If hash values were different, destination host would skip this memory page and send mistake signal, if hash values were same, destination host would put these data in right place.

3.3 Specific Algorithm Process

Although the security algorithm is based on the original algorithm, its process is more or less different from the original algorithm. Specific process is shown in Fig. 9.

The algorithm process at source host is summarized as follows:

(1) After the source host establishes connection with destination, initials mistake bitmap and other migration related parameters.
(2) Secret key function generates the seed secret key, and saves this time as timestamp.
(3) Monitor function monitors VM memory data. If this memory page was sensitive, call the security function, if not, its contents would be copied to the QEMU file waiting to be sent.
(4) Security function calculates the hash values of memory data, and encodes sensitive data and its hash value using the seed secret key. Then the encoded data will be copied to QEMU file waiting to be sent.
(5) If source host received the mistake signal during live migration process, source host would make marks on the mistake bitmap in the corresponding position. In the last stage, source host transmits dirty memory pages according to the result of

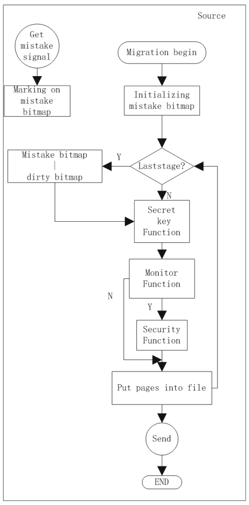


Fig. 9. Algorithm process at source.

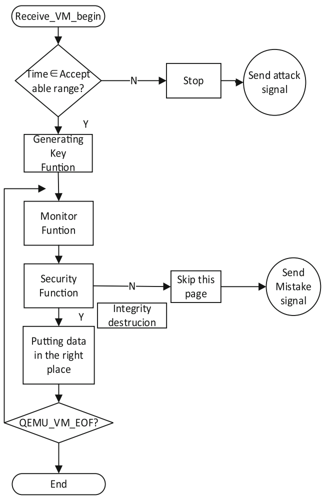


Fig. 10. Algorithm process at destination

dirty bitmap making OR operation with mistake bitmap. If the last round migration data continued being destroyed, source host would stop migration, report that someone is trying to destroy this migration.

The algorithm process at destination host is shown in Fig. 10. The destination does the opposite operation to the source host. When destination host finds the page modified, host will skip this page and send mistake signal to source host. If the received time was unacceptable, host would stop this migration and report to the VM user.

3.4 Expectation

The security algorithm of this paper mainly improve the traditional migration mechanism from two aspects, one is the protection object, and the other one is the protection algorithm.

- Time complexity

Traditional protection method encrypts all of the VM data in transport layer or network layer to prevent attackers from eavesdropping or modifying, and this security algorithm narrows the range of data needing to be protected, reducing the time of security migration in the same time. Table 1 summarizes the parameters and notations used throughout this paper. Assume the migrating algorithm proceeds in n ($n <= N$) rounds. Let v_i ($0 <= i <= n$) denote the data volume transmitted at each pre-copying round, and t_i ($0 <= i <= n$) denote the elapsed time at each round. The data transmitted in round i is calculated in Eq. (1).

Table 1. Parameters for VM live migration modeling.

Symbol	Description
M	Size of VM memory image
M_u	Size of VM users' memory image
r	Network transmission rate during migration
d	Memory dirtying rate during migration
V_{thd}	Threshold of the remaining dirty memory that should be transferred during the stop-and-copy phase
N	The pre-defined maximum number of rounds for iterative pre-copying in migration algorithm
t_t	The time of generating Timestamp
t_k	The time of generating secret key
k_1	The factor of encryption speed of the new security algorithm

$$v_i = \begin{cases} M, & \textit{if } i = 0; \\ \mathrm{d} \cdot \mathrm{t}_{i-1}, & \textit{otherwise}. \end{cases} \tag{1}$$

The elapsed time at round i is calculated in Eq. (2).

$$t_i = \begin{cases} \frac{M}{\mathrm{r}}, & \textit{if } i = 0; \\ \frac{d \cdot \mathrm{t}_{i-1}}{r}, & \textit{otherwise}. \end{cases} \tag{2}$$

To evaluate the convergence rate of VM migration algorithm

We calculate the total number of rounds by the inequality $v_n < V_{thd}$. It is the condition to terminate the iterative pre-copying and to start the stop-and-copy phase. Furthermore, it should not be larger than the pre-defined parameter N. As a result, the number of pre-copying iterations becomes:

$$n = \min\left\{ \left\lceil \log \frac{V_{thd}}{M} \Big/ \log \frac{d}{r} \right\rceil | N \right\}. \tag{3}$$

For a given VM, M and V_{thd} (determined by migration algorithms) can be viewed as constants. Consequently, the iterative pre-copying would converge faster if d/r was smaller. Then define d/r as the convergence coefficient of VM live migration.

The data of the new security algorithm in round i is calculated in Eq. (4). M_u is smaller than M. The elapsed time of the new security algorithm at round i is calculated in Eq. (5).

$$v_i = \begin{cases} M_{\mathrm{u}}, & \textit{if } i = 0; \\ \mathrm{d} \cdot \mathrm{t}_{i-1}, & \textit{otherwise}. \end{cases} \tag{4}$$

$$t_i = \begin{cases} \frac{M_{\mathrm{u}}}{\mathrm{r}} + t_t, & \textit{if } i = 0; \\ k_1 \cdot \frac{d \cdot \mathrm{t}_{i-1}}{r} + t_k, & \textit{otherwise}. \end{cases} \tag{5}$$

- Anti-attack capability

  The new security algorithm could resist security attack types as follows:

  - Anti-Replay Attacks: Time is one of the influencing factors for generating the seed secret key. Once the received timestamp was illegal, destination host would refuse to accept the VM data leading attacker to fail.
  - Anti-Eavesdrop: The insecure and unprotected transmission channel is the result from vulnerabilities of migration protocol. The migration protocol does not encrypt the data as it travels over the network, susceptible to Eavesdrop attack. This security algorithm makes attackers could not get plaintext of VM data, and uses a number of different keys leading to attackers harder to frequency analysis and crack.
  - Anti-modification: This security algorithm adds the integrity verification to the process of migration. Destination host checks the integrity of received data. If integrity was destroyed, destination host would notify the source host to send again, therefore modified data is useless.

The new security algorithm is designed based on KVM source code. In spite of this algorithm implemented based on RC4, it should be noted that the new algorithm use a number of different keys in every migration and never let keys exposure in the network. Hence it is impossible to crack unless the attacker knows the seed secret key generation mechanism. Besides it narrows down the field of data need to be encode, and add timestamp and integrity verification, ensuring the security of migrating data, meanwhile, guaranteeing the efficiency of migration. The security algorithm is a part of migration module in VMM layer which any users couldn't get into, thus ensuring the algorithm is secure. It only encodes user space data reducing the consumption of the secure live migration process. So the security algorithm guarantee the security of migrating VM and the acceptable migration time.

The focus of this article is the protection of vulnerabilities in live migration mechanism, which is based on the version of the VMM has owned the official security certification. Therefore, we put the monitoring module and security module on the VMM layer to ensure the security of the algorithm itself. The security problems of the VMM layer also is our research direction in the future.

4 Implementation and Evaluations

In order to evaluate the security algorithm, we verify it from perspective of functionality and performance by applying it to the KVM virtualization platform.

4.1 Function Verification

We have made a lot of experiments to test the security of VM data. One of them is that the user enters any characters through the Gedit editor in the Linux system, then migrates VM from the source host to the destination host.

The attacker could get the encoded data during normal migration, but attackers could not find the sensitive information when migrated VM protected by applying the new security algorithm. Due to the limited essay space, we don't show the results of these experiments one by one. As the new algorithm encodes the sensitive data, the sensitive data is messy code in transmit channel. The attacker is hard to distinguish the meaningful messy code from the messy code. The new algorithm fully achieves the expected security features.

4.2 Performance Verification

Migrations have executed a lot of times for evaluation purpose under the condition of the same load state. Performance evaluation is divided into three categories which are live migration without change (Normal), live migration with security algorithm based on improved RC4 (The new algorithm) and live migration with SSL implementation. As mentioned above, the measurement of virtual machine migration efficiency is generally from several aspects, total time, downtime and the impact on the performance of other services. So we make a comparison from three aspects, total time, downtime and peak CPU usage.

When migrating with the new security algorithm, it costs more time than normal migration, and less time than live migration with SSL implementation.

The usage of the new security algorithm makes downtime longer than normal, but its downtime is shorter than the way using the SSL protocol at the same time.

The higher CPU usage, the greater impact on other services' (or other VMs) performance on the host server. How much CPU overhead occurs is observed by measuring the peak CPU usage (%) during the live migration. In this experiment, the peak CPU usage is measured using the top command because it provides an ongoing look at the activities of processor in real time which is suitable as to measure and calculate the peak CPU usage during the migration. The peak CPU usage of the source host machine is calculated during the migration process. It is understood that security needs performance penalty. Live migration with SSL implementation requires the greatest CPU resource. By contrast, the new security algorithm is more acceptable.

Based on the results, SSL implementation probably meets the requirement for the secure live migration of virtual machines to some extent, consuming more CPU usage and longer migration time. We also test the traditional RC4 for migration encryption. RC4 takes less time than the new protection way. However, for the widespread of RC4, its cracked difficulty is far less than the theoretical value. Live migration based on the new security algorithm makes the balance of security and performance come true. It only takes very little time but increase index cracked difficulty than RC4.

4.3 Contrast and Summary

- Performance

The traditional protection way SSL have a significant impact on the total migration time [24]. The new security algorithm of this article is implemented by three functions and not brings as much burden as SSL to living migration. Compared to SSL, the security algorithm only encodes user space data, making migration more efficient. The new algorithm could guarantee the integrity and confidentiality of data which are based on the original migration mechanism.

- Security mechanism

SSL sessions consist of two phases, the SSL Handshaking Protocol and the Record Protocol, and the client and server agree on various parameters used to establish the connection's security. One of the SSL protocol weaknesses is that the selected encryption algorithm and key are transport through network in plain text. The attacker could modify encryption algorithm to weak one at this time. Leading to transported encrypted packets may be cracked easily. OpenSSL is an open source project that provides a robust, commercial-grade, and full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. Lately, the vulnerabilities (CVE-2016-0701) were released by OpenSSL official. In OpenSSL1.0.2, due to the program didn't correctly generate the prime number for the Diffie-Hellman protocol, remote attacker could use the vulnerability to obtain the encryption key and sensitive information. Besides, because the SSL protocol is widely adopted, more and more holes are discovered, its security level is no longer as high as the past. As stated earlier, the security algorithm is based on the migration algorithm, so source host firstly needs to transmit timestamp to destination host to notice of the secret key in this migration, the destination host timestamp verification mechanism could ensure the legitimacy of this migration preventing illegal tampering with the secret key or implementing replay attacks. Thus in this sense, the new security algorithm has higher security than SSL.

5 Conclusion

In this paper, we innovatively propose a security algorithm, using different seed secret keys in each iteration and checking the modified migration data at last round iteration, to strengthen the protection of user space data, and making up for the vulnerabilities in live migration protocol of KVM virtualization. The simulation experimental results demonstrate that the proposed algorithm ensures the confidentiality and integrity of migrated VM's data and cost less time than SSL implementation. In the future, we plan to develop a compression function to reduce the migration time in the security algorithm. Furthermore, we also intend to implement our approach to different versions of KVM virtualization systems to generalize findings and refinement of the work.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (Grant No. 61502486; 61170282).

References

1. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (SOSP19), pp. 164–177. ACM Press (2003)
2. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: Proceedings of NSDI, pp. 273–286. USENIX Association, Berkely (2005)
3. Padala, P., Zhu, X., Wang, Z., et al.: Performance evaluation of virtualization technologies for server consolidation. Virtualiz. VMware ESX Serv. **9**, 161–196 (2007)
4. Murugesan, S.: Harnessing green IT: principles and practices. In: Proceeding of IT Professional, vol. 10, pp. 24–33. IEEE Computer Society (2008)
5. Djenna, A., Batouche, M.: Security problems in cloud infrastructure. In: The 2014 International Symposium on Networks, Computers and Communications, pp. 1–6. IEEE (2014)
6. Ristenpart, T., Tromer, E., Shacham, H., et al.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: CCS Conference, pp. 199–212 (2009)
7. Fan, W., Kong, B., Zhang, Z.J., Wang, T.T., Zhang, J., Huang, W.Q.: Security protection model on live migration for KVM virtualization. J. Softw. **27**(6), 1402–1416 (2016). (in Chinese)
8. Oberheide, J., Cooke, E., Jahanian, F.: Empirical exploitation of live migration of virtual machines. In: Black Hat DC Briefings, Westin Washington DC City Center (2008)
9. Yamunadevi, L., Aruna, P., Sudha, D.D., et al.: Security in virtual machine live migration for KVM. In: 2011 International Conference on Process Automation, Control and Computing (PACC), pp. 1–6. IEEE (2011)
10. Fan, W., Huang, W.Q., Jiang, F., Liu, C., Lv, B., Wang, R.R.: Research on security of memory leakage in live migration based virtualization. In: Twenty-Fourth National Conference on Information Security (IS 2014), vol. 09, pp. 12–17 (2014)
11. Dawoud, W., Takouna, I., Meinel, C.: Infrastructure as a service security: challenges and solutions. In: The 7th International Conference on Informatics and Systems (INFOS), pp. 1–8 (2010)
12. Anala, M.R., Shetty, J., Shobha, G.: A framework for secure live migration of virtual machines. In: 2013 International Conference on IEEE Advances in Computing, Communications and Informatics (ICACCI), pp. 243–248 (2013)
13. Aiash, M., Mapp, G., Gemikonakli, O.: Secure live virtual machines migration: issues and solutions. In: 2014 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 160–165. IEEE Computer Society (2014)
14. Garfinkel, T., Rosenblum, M.: When virtual is harder than real: security challenges in virtual machine based computing environments. In: Workshop on Hot Topics in Operating Systems (2005)
15. Sun, D., Zhang, J., Fan, W., et al.: SPLM: security protection of live virtual machine migration in cloud computing. In: Proceedings of the 4th ACM International Workshop on Security in Cloud Computing, pp. 2–9. ACM (2016)
16. Ballani, H., Francis, P., Zhang, X.: A study of prefix hijacking and interception in the internet. ACM SIGCOMM Comput. Commun. Rev. **37**(4), 265–276 (2007)
17. Zargar, S.T., Joshi, J., Tipper, D.: A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. IEEE Commun. Surv. Tutor. **15**(4), 2046–2069 (2013)
18. Cowan, C., Wagle, F., Pu, C., et al.: Buffer overflows: attacks and defenses for the vulnerability of the decade. In: Information Survivability Conference and Exposition (2000)

19. Wang, J., Yang, Y., Chen, L., Yang, G., Chen, Z., Wen, L.: A combination of timing attack and statistical method to reduce computational complexities of SSL/TLSside-channel attacks. In: 2015 11th International Conference on Computational Intelligence and Security (CIS) (2015)
20. Awasthi, A., Gupta, R.: Multiple hypervisor based open stack cloud and VM migration. In: 2016 6th International Conference - Cloud System and Big Data Engineering (Confluence), Noida, pp. 130–134 (2016)
21. Graziano, C.D.: A performance analysis of Xen and KVM hypervisors for hosting the Xen Worlds Project. Graduate Theses and Dissertations, Paper 12215 (2011)
22. King, S.T., Chen, P.M.: SubVirt: implementing malware with virtual machines. In: IEEE Symposium on Security & Privacy, pp. 314–327. IEEE (2006)
23. Ravi, P., Shah, P.H.: Security in live virtual machine migration. Wichita State Univ. **5**(5), 31 (2011)
24. Hu, Y., et al.: Performance analysis of encryption in securing the live migration of virtual machines. In: 2015 IEEE 8th International Conference on Cloud Computing, New York City, NY, pp. 613–620 (2015)