An Independent Component Recovery Approach for Intrusion Tolerance

Jianhua Huang[1, a], Yang Liu[1, b], Yi Jin[1, c]and Shalin Huang[2, d]

[1]East China University of Science and Technology, Institute of information science and engineering, Shanghai200237, PRC

[2]Wangsu Science & Technology Co.,Ltd., Shanghai 200020, PRC

[a]jhhuang@ecust.edu.cn, [b]412485949@qq.com, [c]jinyi@ecust.edu.cn, [d]sallyhuang@chinanetcenter.com

Keywords: Intrusion Tolerance, Independent Component, Recovery, State Synchronization

Abstract. An independent component recovery approach for intrusion tolerance is proposed in this paper. Virtual Machines (VMs) are used to provide each online replica with an independent backup replica. The recovery of each component depends on the actual service state. A strict execution order is not needed to create VMs, destroy VMs and synchronize state data. The approach improves the flexibility of components of the system. In addition, for preventing the state data from losing during the recovery of replicas, a set of virtual storages, which can be connected or disconnected with online servers quickly, are used for storing the state data. A fixed cycle is set for synchronizing the state data independently. A prototype is implemented in the OpenStack cloud platform. Experimental results show that the approach improves the continuity and security of the services.

Introduction

Modern life is becoming more and more dependent on network applications. It is very important for us to protect the applications from being intruded. At present, perimeter protection techniques such as firewalls, access control, encryption and intrusion detection [1] are the major methods for protecting the application systems. However, studies have shown that it is difficult to eliminate vulnerabilities in an application system, and it is not enough to only use these techniques for protecting applications [2]. In the past few years, a novel technique called intrusion tolerance has been proposed to address network security problems. Its goal is to guarantee the correct behavior of a system even if some of its components are compromised by an intelligent adversary [3]. Intrusion tolerance uses redundancy, diversity, recovery and other defense mechanisms to build survivable systems. So far, significant progress has been made in the research about intrusion tolerance, but there are still some problems to be solved. For examples, transferring state information effectively between replicas of an intrusion tolerance system is a fatal problem for the continuity of services. A better solution for recovery is required to improve the flexibility of application systems.

An intrusion tolerance recovery approach based on independent components is proposed to address above problems. By calculating recovery time of each component respectively, the dependencies between different components are reduced. In addition, a set of independent virtual storages are used to store the state data. These virtual storages prevent state data from losing during the recovery of VMs by connecting and disconnecting them with online VMs.

Related works

Intrusion tolerance has attracted great attention over the last decade. SITAR is a classic intrusion tolerance model which uses redundancy, diversity and voting to protect the security of the system [4]. MAFTIA [5] is a middleware-based intrusion tolerance solution that helps survivability. Common intrusion-tolerant systems are distributed architectures based on Byzantine fault-tolerant replication algorithms. A problem with Byzantine fault-tolerant replication is the assumption that the system operates correctly only if at most f out of 3f +1 replicas are compromised. Given enough time, however, an adversary might still be able to compromise f+1 replicas. Some works show that the problem can be solved by using a technique called proactive recovery if replicas are rejuvenated periodically [6]. Proactive recovery can remove potentially undetected intrusions as long as the recovery time of servers is faster than an upper-bound on fault production assumed at system deployment time. Castro [7] was the first to present a proactive recovery protocol for BFT systems. Authors in [8] proposed splitting the system into synchronous components that activates periodic rejuvenation and any-synchronous subsystems that include the payload application. This approach was adopted later in other papers [9,10]. In particular, authors in [9] enhanced proactive recovery with reactive recovery, which lets replicas to be recovered as soon as they are detected or suspected of being compromised. Yih Huang proposed a SCIT model which eliminates possible invasion by recovering the online server regularly [11], but it could not stop the malicious behavior of the adversary before the recovery is triggered. PR-SCIT added intrusion detection components to SCIT to solve the problem [12], which takes them offline as soon as possible. However, all these methods do not have a better solution on eliminating the restriction relationship of components in recovery.

Virtualization technology is used for implementing intrusion tolerant systems to cope with the high cost brought by multiple servers as redundancy [13]. VM-FIT deploys multiple VMs on an entity server and provides services in parallel [14], which reduces the waiting time of reloading the image of the entity server. TwinBFT in [15] enhances the security of system by deploying multiple sets of twinVMs to check the consistency of its' response. However, these intrusion tolerant service systems do not have a good solution of synchronizing state information.

ICRIT Architecture

Cloud Computing provides us with enough flexibility for dynamic allocation of Virtual Machines (VMs). In this paper, we propose a novel cloud-based architecture named Independent Component Recovery for Intrusion Tolerance (ICRIT). ICRIT stores state data independent with the recovery of VMs. It has the following main subsystems: Proxy, VM group, PSS group, Voter, VMMC, and RSS. The logical view of the ICRIT architecture is shown in Fig. 1. A set of redundant VMs in the VM group are used for implementing Byzantine fault-tolerant replication. Given enough time, however, an adversary might still be able to compromise f+1 replicas. But ICRIT only creates VM image's instances as replicas to provide online services. After a replica is running for a while, it is taken offline and a new, pristine virtual server replaces the prior one. Periodically replacing replicas with new backup replicas makes it hard for adversaries to maintain control over more than f compromised replicas.

In the ICRIT architecture, Proxy is the public access interface of the system. During normal operation, a client request is first routed to Proxy for preliminary screening, and then it is forwarded to VM group for processing. After the request is processed, the replicas propagates the state changes together with the responses to a set of private state storages (PSS) located in PSS group. The responses are then passed to Voter for voting. The agreement process result is subsequently forwarded to Proxy for sending back to the client.

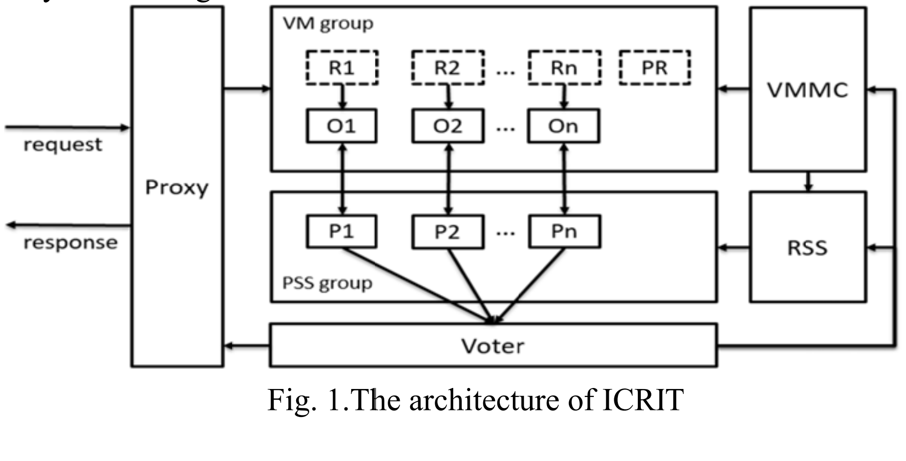

Fig. 1.The architecture of ICRIT

In VM group, O_1, O_2, ..., O_n represent a group of online VM replicas. R_1, R_2, ..., R_n represent a group of backup VM replicas. It should be noted that each online VM independently connects with its own PSS. After processing the request, online VMs return the state changes to PSSs. Instead of single replica in other intrusion tolerance systems, a common replica (PR block) and more than one backup replica are provided in our architecture. When PSSs execute the updating command of the state data, it is first processed by Voter. Voter obtains the agreement state data through majority voting, marks anomaly VMs, and sends the agreement state data to RSS and VMMC.

VM Management Center (VMMC) is the core control component which is responsible for creating and destroying VMs according to the recovery policies. Besides, VMMC also informs RSS to synchronize state data to PSSs and control the connection between PSSs and online VMs.

Referential State Storage (RSS) is the control point of synchronization of PSSs, it sends update instructions to the PSSs to be synchronized. The update instructions are generated by a log which records all the voting information and contrasting results of state data.

The recovery of replicas

Recovery strategy of replicas. This paper proposes a solution which combines proactive recovery with reactive recovery in order to increase the overall performance and resilience of intrusion-tolerant systems. To remove potentially undetected intrusions, all online VMs will be rejuvenated periodically. Reactive recovery means that the recovery of replicas will be accelerated if some replicas are detected or suspected to be faulty.

Commonly, a new VM is created before an online replica is taken offline. It replaces services as soon as the original replica is taken offline. Thinking of the number limitation of replicas in other systems, the online replicas in ICRIT have their own backup VMs. The substitute relationship shown in Fig. 2 enables the system to have enough replicas even more than one online replica are compromised simultaneously. In other one-replica systems, when a replica is compromised, a backup replica will be taken online, and there is no other replica as replacement if another online replica is compromised. However, ICRIT independently creates specific backup VMs for each online replica. We can obtain more flexible and secure systems.

Considering the situation that the backup VM of an online replica hasn't be created, except these normal replicas, a replica named public replica (PR) is always reserved in the system. The PR is taken online when the above situation occurs, and the system creates a new replica as a PR again.

Recovery time of replicas. In ICRIT, each VM replica may be in one of three states: normal service, suspected compromise and confirmed compromise. We define three corresponding recoveries for VMs: N (Normal) recovery, S (Shrink) recovery and D (Direct) recovery. N recovery means that a VM is running well and recovered only if the recovery timer goes off. S recovery means that a VM is suspected to be compromised and need to shrink its remaining run time. D recovery means that a VM is confirmed to be compromised and must be taken offline at once.

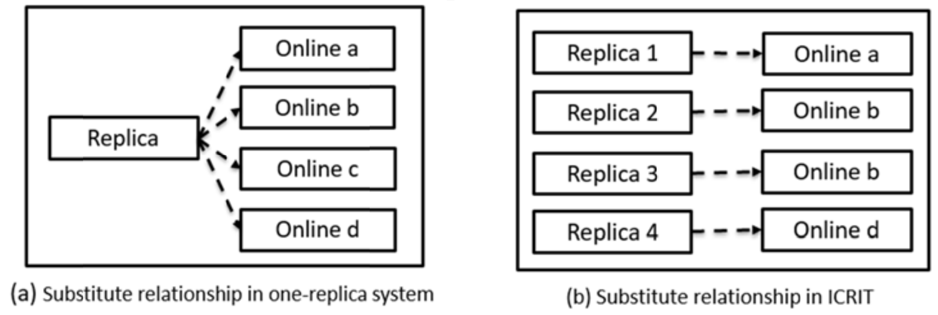

Fig. 2. The contrast of substitute relationship in ICRIT and other ITS

Though ICRIT removes the competitive relationship of online replicas to backup VMs, for the reason that keeps the continuity of the system, the time of creating new replicas and taking VMs offline is related. Therefore, we discuss the time of creating replicas and taking replicas offline together. Table 1 lists all the parameters which are used in our algorithm. Specially, the values of Θ and n are decided by the demands of security and performance.

Table. 1. Parameters of the recovery time algorithm

[Parameter]	[Description]
T	Proactive recovery cycle of VMs
L	Maximum time of creating a new replica
Θ	Time threshold of shrinking the remaining run time of VMs
n	Maximum number of triggering S recovery
T_r	Remaining run time of VMs

The process of calculating the recovery time of replicas is as follows:

a) If no anomaly is detected, a backup VM is firstly created for the online replica (L ahead of time). The online replica will be taken offline and destroyed after its running cycle T expires.

b) If a S recovery is triggered and the remaining run time of the replica is greater than the threshold Θ, the T_r of the VM will be updated to:

$$T_r = T_r * (n - 1 / n),\ n = n - 1. \qquad (1)$$

In equation 1, for reducing more time in the next triggering, n is reduced by 1 after triggered every time. Obviously, if S recovery is triggered n times, the system starts a process to take the replica offline directly.

While if T_r is less than Θ, the remaining run time of the replica is not reduced until the remaining run time goes off. Therefore, the system has enough time to create a new replica.

c) If a D recovery is triggered and the backup VM of the replica hasn't been created. The remaining run time of the replica is set to L. Otherwise, the VM is taken offline once the remaining run time goes off.

When the number of triggering D recovery is less than the number of triggering S recovery, the performance of the system will be improved by shrinking the remaining run time of the replica instead of taking the replica offline directly. Creating more replicas in a short time increases the overload of the physical server, but better security is gotten.

The recovery of state data

Mechanism of Independent state storage. Commonly, the state data stored in VMs is lost during the recovery of VMs. A replica to be taken online newly has to synchronize its state data from another online VM. Services may be interrupted during the synchronization. To cope with the problem, ICRIT separates the storage of state data from VMs. A set of virtual storages are used for storing state data. They provide online VMs with the state data and accept the results processed by online VMs. As shown in Fig.3, when a VM is going to be destroyed, its virtual storage will be disconnected with the VM and be connected with the new backup replica of the VM again. Thus, the time of synchronization is greatly reduced.

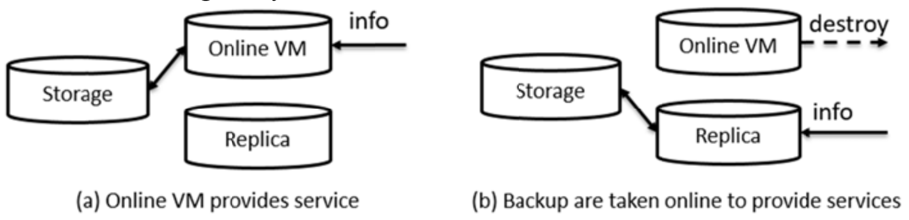
info
Online VM
Storage
Replica
(a) Online VM provides service
destroy
Online VM
Storage
info
Replica
(b) Backup are taken online to provide services

Fig.3.The service mode of recovery process

State synchronization mechanism. A compromised VM always leads data abnormally to a virtual storage. Instead of synchronizing the state data when a replica is taken online and all VMs have to attend synchronization, ICRIT synchronizes state data in the following two situations:

a) Proactive state synchronization. Periodical synchronization of state data is taken to ensure the consistency of state data. An independent state synchronization cycle Q is set to achieve this function. We suggest that the state synchronization cycle should be below the recovery cycle of VMs. Every Q time, each PSS has to be synchronized by RSS.

b) Reactive state synchronization. Drawing lessons from the reactive recovery of replicas, ICRIT imports the reactive policy into state data synchronization. ICRIT accepts the abnormal state data of the storage in a short time, which takes the performance and security of the system into consideration. Frequent state synchronization is useless for restoring the system back to the normal state when a replica is compromised, and less synchronization is not conducive to the security of the system. We assume that R is the anomaly number that a PSS can tolerant. RSS records all updating commands of replicas and the updating results of the PSS in a log, then marks the anomaly number of each PSS. Once a PSS's anomaly number reaches R, the PSS will be synchronized by RSS.

Table. 2. Logs in RSS

[ID]	[Updating command]	[S1]	[S2]	[S3]	[S4]
1	"SELECT* FROM USERWHERE …"	1	1	0	1
2	"DELETE STU WHERE id=8"	1	1	1	0
…	……	…	…	…	…

State synchronization algorithm. The logs recorded by RSS determinate which part of the state data of a PSS should be synchronized. RSS records updating commands and each PSS's updating results. If a PSS's updating result to an updating command is right, RSS will mark the result of the PSS to the updating command as 1. Otherwise, it marks the result of the PSS to the updating command as 0. Table 2 is samples of the RSS's logs. We assume that there are four storages named {S1, S2, S3, S4}. The updating results of updating command where ID equals 1 are {1, 1, 0, 1}.It means that the updating result of S3 is anomaly. Therefore, the synchronization of the updating command should only occurs between RSS and S3.

For a long-lived system, its state data should be very large. Synchronizing all the data during every recovery may impact the performance and continuity of the system. Therefore, a timestamp is used for limiting the size of the state data which is synchronized. The timestamp is recorded by RSS every certain time, and RSS ensures that all state synchronizations have been executed.

Analysis and Experiment

Security Analysis. The recovery of replicas in ICRIT is decided by the voting result of Voter. Assuming that there are N replicas(N meets 3f+1 of the Byzantine protocol). If only the number of anomaly replicas is less than N/3, the voted results will keep right.

The recovery mechanism of ICRIT is proactive-reactive. Therefore, the time of invading the system is limited. Replicas will be recovered after running for a period T to eliminate unconscious invasion, which makes the work done by an adversary useless. Besides, by recovering periodically, the recovery time of all replicas is distributed in the time T, and one replica will be taken offline in each T/N time. Therefore, the adversary must invade at least N/3+1 replicas in (N/3) * (T/N) time. Otherwise one of replicas that he invaded is sure to be taken offline.

If the system can't be confirmed whether a replica is compromised and S recovery is triggered m times according to the reactive recovery algorithm proposed in the paper, the time for the adversary will be reduced to (N/3)*(T*(n-m)/n). It is clear that the more S recovery is triggered, the less time for the adversary is remained. If an intrusion can be detected, a D recovery will be triggered and a process will be activated to take the online replica offline directly.

The analysis above shows that the number of replicas in ICRIT is proportional to the system security. When there are only less VMs, the number of triggering S recovery is set to 1. The reactive recovery process is the same as other systems, which avoids the adversary to accomplish his goal of compromising the system if the running time of the VM is shrunk instead of taken offline.

The usage of CPU and time. The experimental setup was composed by four Dell servers, with the Intel Xeon (R) E5-2407 CPU and 32GB memory. Ubuntu Server 14.04 and the OpenStack cloud platform are installed in these servers. One of the servers is selected as the core control node, the other 3 servers are the compute nodes. In addition, we used a 1.5GB Ubuntu image with 2.0GB RAM and 20GB disk space to create replicas. We focus on the usage of CPU and the recovery time of replicas in our experiment. Tests were executed 60 times and the results are as follows:

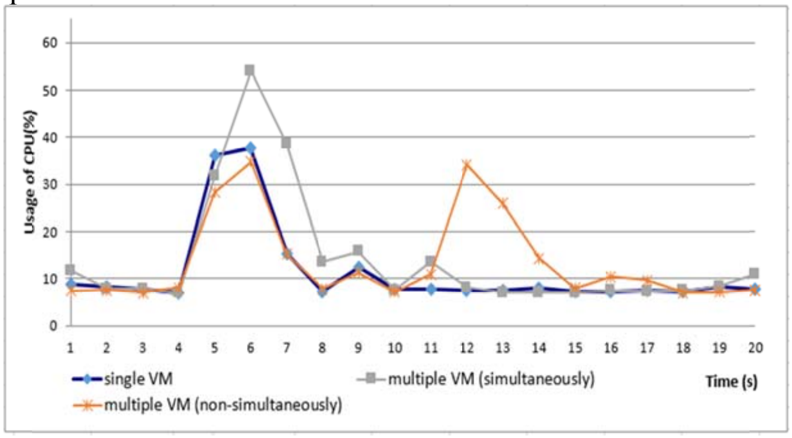

Fig.4. The usage of CPU in different situation

1. For creating a single VM. As shown in Fig.4, the average time for the system to create a new replica is 14.327 seconds, the max usage of CPU is 37%, and the average usage is 8.5%.

2. For creating more than one VM (2 in our test) simultaneously. The average time of creating two replicas is19.227seconds， the max usage of CPU is 52.1%, and the average usage is 9.4%.

3. For creating more than one VM at a period of time. A random number between 1 and 5 was generated to control the time difference of creating two VMs. The max usage of CPU is 44.6%, and the average usage is 8.9%.

In addition, we tested the time for connecting a virtual disk to a VM. The test was executed 50 times. The average connecting time is 0.784 second, which has little influence on the system.

Time consuming of synchronization. A Java program was used for testing the time consuming of state synchronization. The tested data only have one column. We did the test 30 times, the average time of state synchronization that we calculate is shown in Fig.5 (The curve denoted by square). Obviously, the time of state **synchronization** is proportional to the quantities of records.

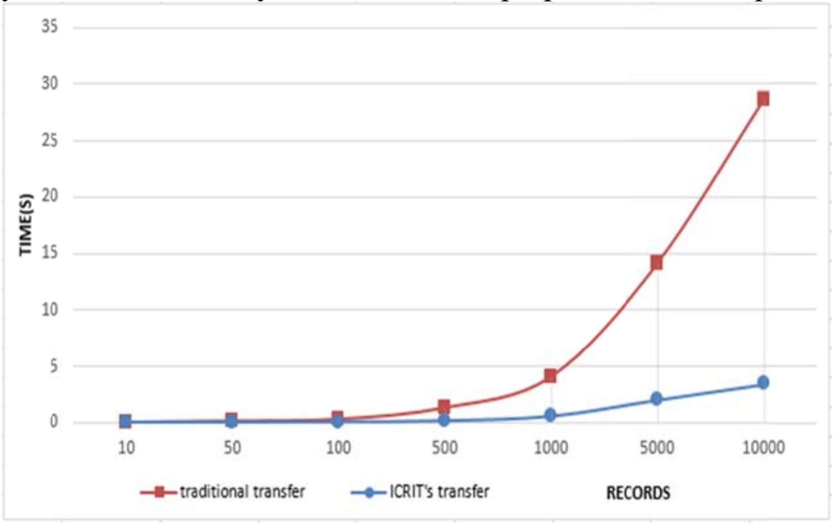

Fig. 5.The contrast of transferring information

While the timestamp-based transfer way of records has the less time. The interval to set timestamp is 60 seconds. 10000 data sets are logged in order to observe the effects of different scale anomalies, 50 to 5000 abnormal data sets are marked in a virtual disk. ICRIT’s transfer average time for testing 30 times is shown in Fig.5. It is clear that ICRIT's synchronization greatly limits the data size in the process of state transition and significantly reduces the time of data synchronization.

In addition, an experiment was done to test the influence of voting. The Voter uses MD5 code of received data to compare analysis. The average time of voting is 22ms. So the performance of the system almost isn't influenced by voting.

Conclusion

This paper proposed an independent component recovery approach for intrusion tolerance. The combination of proactive and reactive recovery in the cloud environment was designed in order to increase the overall performance and resilience of intrusion-tolerant systems. Virtual machines are used to provide each online replica with an independent backup. Replicas' state data is stored in independent PSSs which can be quickly connected or disconnected with replicas. Therefore, the independence and security of the state data are improved. At the same time, the approach takes the state data of updated results as the basis of triggering reactive recovery, and further improves the security of the system. In addition, because our approach does not need to synchronize state data every time an abnormity occurs, the CPU usage and the time cost for data synchronization are reduced. In the future work, adjusting the number of servers dynamically may be considered on the basis of the ICRIT architecture, and the better algorithm of optimizing reactive recovery is needed.

Acknowledgments

We are pleased to acknowledge the National Natural Science Foundation of China under Grant 61472139. The authors are also grateful to the anonymous referees for their insightful and valuable comments and suggestions.

Reference

[1] Nguyen Q L, Sood A, A Comparison of Intrusion-Tolerant System Architectures[J], IEEE Security & Privacy, 2011, 9(4):24-31.
[2] G. Jakobson, Mission-Centricity in Cyber Security: Architecting cyber attackresilient missions, 2013 5th International Conference on Cyber Conflict, 2013:1-18.
[3] Kim Y, Lim J, Doo S, et al., The design of adaptive intrusion tolerant system(ITS) based on historical data[C], Internet Technology And Secured Transactions, 2012 International Conference for. IEEE, 2012:662-667.
[4] Wang F, Upppalli R, SITAR: A Scalable Intrusion-Tolerant Architecture for Distributed Services-A Technology Summary[C], IEEE Computer Society, 2003:153-153.
[5] Adelsbach A, Alessandri D, Cachin C, et al., Conceptual Model and Architecture of MAFTIA[J], Technical Report, 2003.
[6] Distler T, Kapitza R, Reiser H P, Efficient state transfer for hypervisor-based proactive recovery[J], 2nd Workshop in Recent Advances on Intrusion-Tolerant Systems - WRAITS'08, 2008:1-6.
[7] Castro M, Liskov B, Practical byzantine fault tolerance and proactive recovery[J], Acm Transactions on Computer Systems, 2002, 20(4):398-461.
[8] Sousa P, Neves N F, Verissimo P, How resilient are distributed f fault/intrusion-tolerant systems?[J]. Department of Informatics University of Lisbon, 2005:98-107.
[9] Sousa P, Bessani A N, Correia M, et al., Highly Available Intrusion-Tolerant Services with Proactive-Reactive Recovery[J], IEEE Transactions on Parallel & Distributed Systems, 2010, 21(4):452-465.
[10] Zhao F, Li M, Qiang W, et al., Proactive recovery approach for intrusion tolerance with dynamic configuration of physical and virtual replicas[J], Security & Communication Networks, 2012, 5(10):1169–1180.
[11] Huang Y, Sood A, Self-cleansing systems for intrusion containment[J]. Workshop on Self Healing Adaptive & Self Managed Systems, 2002.
[12] Sousa P, Bessani A N, Correia M, et al., Resilient Intrusion Tolerance through Proactive and

Reactive Recovery, Dependable Computing, 2007. 13th Pacific Rim International Symposium on, 17-19 Dec (2007):373-380.
[13] Huang J, Ai Q, Analysis of a Virtualization-based Recovery Approach for Intrusion Tolerance Systems[C], Computer Sciences and Applications (CSA), 2013 International Conference on. IEEE, 2013:41-46.
[14] Ficco M, Rak M, Intrusion Tolerance in Cloud Applications: The mOSAIC Approach[C], Proceedings of the 2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS). IEEE Computer Society, 2012:170-176.
[15] Dettoni F, Lung L C, Correia M, et al., Byzantine fault-tolerant state machine replication with twin virtual machines[C], Computers and Communications (ISCC), 2013 IEEE Symposium on. IEEE, 2013:000398 - 000403.