IEICE TRANS. INF. & SYST., VOL.E97–D, NO.4 APRIL 2014

LETTER

A Novel Intrusion Tolerant System Using Live Migration

Yongjoo SHIN[†a)], ***Student Member***, **Sihu SONG**[†b)], **Yunho LEE**[††c)], ***and*** **Hyunsoo YOON**[†d)], ***Nonmembers***

SUMMARY This letter proposes a novel intrusion tolerant system consisting of several virtual machines (VMs) that refresh the target system periodically and by *live migration*, which monitors the many features of the VMs to identify and replace exhausted VMs. The proposed scheme provides adequate performance and dependability against denial of service (DoS) attacks. To show its efficiency and security, we conduct experiments on the CSIM20 simulator, which showed 22% improvement in a normal situation and approximately 77.83% improvement in heavy traffic in terms of the response time compared to that reported in the literature. We measure and compare the response time. The result show that the proposed scheme has shorter response time and maintains than other systems and supports services during the heavy traffic.

key words: *intrusion tolerant system (ITS), proactive & reactive recovery, live migration, denial of service (DoS)*

1. Introduction

The concept of intrusion tolerance was invented to cope with attacks that exploit unknown vulnerabilities. An intrusion tolerant system (ITS) aims to maintain continuous and correct services of the system under attacks or intrusions. ITSs are concerned more with the effects of attacks on the target system than the causes of the attacks because they mainly support the availabilities and do not detect or prevent attacks. ITS architectures are classified into four categories [1]: detection triggered [2]–[4], algorithm driven [5], [6], recovery based [7]–[20], and hybrid [21]. This study focuses on recovery-based methods because the other types of ITSs employ intrusion detection methods and the aim was to build ITSs to cope with situations where the intrusion cannot be detected.

Proactive-based recovery constitutes a major part of research in the recovery-based architecture [7], [9]–[11], such as Self-Cleansing Intrusion Tolerance (SCIT) [7] which utilizes virtualization technology [16], [17]. This is comprised of server virtual machines (VMs). In SCIT, the target system is refreshed at the end of every period, which is known as the exposure time, during which the online VMs provide their services. The exposure time is defined as the time that the server is continuously connected to the internet [22]. Generally, this time means the intruder residence time, which is the duration for attackers to stay in a system for a successful intrusion. The short intruder residence time means a low probability that the system is contaminated by attackers. In order to reduce this time, the system is rejuvenated regularly in SCIT. The refreshing process involves rotating the state of VMs periodically to maintain the pristine state against attacks. Because attackers can have only a bounded opportunity to access the system due to relatively short exposure time, it is difficult to obtain sufficient information to intrude or penetrate. Moreover, although an attack succeeds during the exposure time and the system is modified or damaged, it is possible to provide initially intended services through the periodical refreshing of VMs. The service in SCIT system is related with the high-availability such as the seamless server transitions and sharing of server identities (IP and/or hardware addresses). Examples of existing high-availability systems include DNS servers, NFS servers, web services, authentication services, firewalls, IPsec gateways, and virtual private network (VPN) gateways [23].

On the other hand, the proactive recovery methods have two problems. First, they do not consider the inner status of each VM for refreshing. Therefore, they cannot cope if a VM is exhausted very early in an exposure-time period. Second, they cannot deal with the occurrence of heavy traffic in a short time, such as when a denial of service (DoS) attack occurs. Refreshing VMs periodically can remove the vulnerabilities caused by faults or intrusions but it cannot help process the massive packets because it does not affect the capacity to process incoming packets.

A few studies [11]–[13] reported reactive methods, which refresh the system based on events, not on the time period. On the other hand, they cannot preserve the availability on the intensive short-term heavy traffic. Moreover, some of them [19]–[21] require intrusion detection algorithms, which cannot cope with unknown attacks. They also do not address the issue of managing heavy packets except for the study reported by Lim et al. [24], which adaptively alters the size of a cluster, which is the number of active clusters based on the incoming packets. Nevertheless, because no refreshing process is contained in this approach, it does not provide protection against other attacks with malicious codes.

To maintain the services on the system, even with

Manuscript received July 5, 2013.
Manuscript revised December 6, 2013.

[†]The authors are with Computer Science Department, Korea Advanced Institute of Science and Technology, 291 Daehak-ro, Yuseong-gu, Daejeon, Repubic of Korea.
[††]The author is with ITM Programme, Department of Industrial and Systems Engineering, SeoulTech, 230 Gongneung-ro, Nowon-gu, Seoul, Republic of Korea.
a) E-mail: yj_shin@nslab.kaist.ac.kr
b) E-mail: shsong@nslab.kaist.ac.kr
c) E-mail: younholee@seoultech.ac.kr (Corresponding author)
d) E-mail: hyoon@nslab.kaist.ac.kr

DOI: 10.1587/transinf.E97.D.984

heavy traffic, and improve the performance of the system, this letter proposes a novel ITS using reactive recovery as well as proactive recovery. The proposed scheme is operated in a virtualization environment and has four states (active, grace, inactive, and live spare) like an existing SCIT approach. In contrast to the existing approach, the proposed scheme refreshes the VMs not only at the expiration of the exposure time but also based on the inner status of the VMs through live migration [14], [15]. Generally, VMs are exhausted by the computational overheads which can be produced by plenty of incoming requests as well as the effect of intrusion such as various viruses, worm, and malicious codes. Therefore, the proposed scheme inspects the CPU usage and the queue length in real time in order to discover exhausted VMs suffering from computational overheads. With live migration, which supports the ability to monitor the utilization of each VM to determine exhausted VMs and exchange the exhausted VM to a refreshed VM in real time, the exhausted VMs can be refreshed before expiration of the exposure time. This gives the target system more opportunity to provide services on healthy VMs, which can enhance the quality of service (QoS). Moreover, the proposed scheme can deal with heavy traffic by expanding the size of a cluster. If the amount of incoming packets exceeds the capacity that a cluster can support, the system recognizes it immediately and increases the number of VMs refreshed through reactive recovery using live migration. At that time, the system increases the number of online VMs so that it can obtain sufficient capacity to continue correct services. Unlike the work reported in reference [24], the proposed ITS can adjust the size of a cluster more flexibly and adaptively due to live migration, which results in lowering the cost of constructing secure systems because limited resources can be used efficiently to obtain improved performance and survivability against a DoS attack.

The proposed scheme was implemented, and its efficient performance and capability to fend off a DoS attack were verified using a simulator, CSIM 20, which is a process-oriented, discrete-event simulation model. The response time was measured in the existing system and proposed system. The combination of proactive and reactive recovery improved the performance by more than 22% in a normal situation, and the expansion/reduction mechanism produced more than 77.83% improvement in heavy traffic: it could continue providing services with a lower response time during heavy incoming packets, compared to the system reported elsewhere [24] which adjusts the cluster based on the response time.

2. Proposed scheme

The proposed scheme consists of three components, such as time-triggered recovery, event-triggered recovery, and expansion/reduction of the cluster.

Figure 1 shows the basic framework of the proposed scheme. Similar to SCIT [7], the proposed scheme includes the same cycle with four states, such as Active, Grace-Period, Inactive, and Live-Spare. The refreshing process involves rotating the state of VMs periodically to maintain the pristine state, which is the initial state cleaned, against attacks. The server to provide services must be periodically cleansed to restore itself into a pristine state, regardless of whether an intrusion occurs or not. Although there are undetected successful breaches in a system, they are easily removed before causing serious damages to the system by frequent server rotations. There are four state of servers in a SCIT system such as Active, Grace-Period, Inactive, and Live-Spare state. The services are provided during the Active state. After the exposure time, the VM enters into the Grace-Period state and no longer receives requests from clients. Before refreshing process, all remaining requests in the VM are processed. After accomplishing the process of the remaining requests, the VM enters into the Inactive state and is disconnected with external networks completely to prevent contaminating the pristine image. The pristine image is then mounted to the target VM by the virtualization hypervisor. If above all processes are completed, the VM is ready to do services in the Live-Spare state. The system provides services on virtualization technology. Each VM operates like that reported in reference [22]. In contrast to the existing SCIT approach, it implements a hybrid recovery in which the system is rejuvenated by the given time or inner status of the online VMs. In addition, it increases and decreases the number of online VMs. The virtualization hypervisor plays the role of a central controller, which is responsible for the rotation of each VM according to the time or event triggered recovery and for adjusting the cluster based on the number of incoming packets.

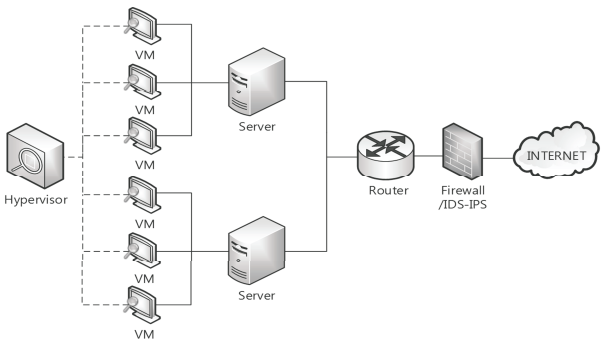

Fig. 1 Basic framework.

Algorithm 1 shows how the proposed scheme works; it basically executes the proactive recovery. Regardless of the occurrence of intrusions or failures, VMs provide services during every exposure time period. Whenever an exposure time period is over, VMs are refreshed as the existing SCIT method [22].

In addition, the proposed scheme carries out the reactive recovery by live migration, which is a technique used to monitor the inner status of the system and identify misbehavior in the system. The virtualization hypervisor inspects the features of each VM, such as CPU usage and queue length for measuring the healthy degree of each VM.

Algorithm 1 ITS with Live Migration

Require:
$hi\_cpu[i]$: the value of the CPU usage in i^{th} VM
$hi\_mem[i]$: the value of the queue length in i^{th} VM
$iv\_cpu[i]$: the ideal value of the CPU usage
$iv\_mem[i]$: the ideal value of the queue length
$time\_exp$: the exposure time
N_act: the # of VMs in the active state.
N_gp: the # of VMs during the grace period state.
$VM[i]$: i^{th} VM in the active state

Periodic Check for Reactive Recovery:
for $i = 1$ to N_act **do**
  $hi\_cpu[i]$ = the usage of CPU in each $VM[i]$
  $hi\_mem[i]$ = the length of queue in each $VM[i]$
end for
Proactive and Reactive Recovery:
for $i = 1$ to N_act **do**
  if $hi\_cpu[i] > iv\_cpu$ or $hi\_mem[i] > iv\_mem$ **then**
    Execute the recovery of $VM[i]$ {Reactive}
  else if $time\_exp$ is expired **then**
    Execute the recovery of $VM[i]$ {Proactive}
  else
    Keep providing a service.
  end if
end for
Expansion and Reduction Mechanism:
if $\frac{N\_gp}{N\_act} > 0.8$ **then**
  $N\_act = N\_act + 1$ {Expansion}
else if $\frac{N\_gp}{N\_act} < 0.2$ **then**
  $N\_act = N\_act - 1$ {Reduction}
else
  Keep providing a service.
end if

The queue is a collection in which the incoming packets in the collection are kept in order. In our scheme, the queue is a buffer queue where the packets forwarding to servers are temporary stored before being processed by service processes. The queue length is the number of packets to be waiting for processing during processing another packet in a VM, where arrivals are determined by a Poisson process and job service times have an exponential distribution. The virtualization hypervisor discovers the abnormal VMs by the ideal values, which were obtained in a normal situation from reference [24]. These represent the values of CPU usage and the queue length during normal service. If the features of a VM exceed the ideal values, the VM suffers from computational overheads and needs to be refreshed. In this case, the system performs the refreshing process of the exhausted VM by immediately ignoring the exposure time period. Therefore, it increases the rate of time when the services are provided on healthy VMs and the entire performance of the system is improved.

Although the recovery solutions are effective in defending against attacks that falsify and modify the inner environment of the target system, it is helpless to sustain the seamless services under a DoS attack because the recovery process does not increase the capacity to process heavy packets. If many incoming packets are brought into the system during a short time, it is inevitable to increase the number of VMs to continue providing services, because heavy packets increase CPU usage and the queue length in the system. In Grace-Period state, virtual machine processes any existing requests, but does not accept any new requests [22]. Because VMs are disconnected with external networks during refreshing process, the online VM to refresh must complete existing requests and deny receiving new requests before beginning the refreshing process. Thus, in order to response requests completely in SCIT, the Grace-Period state is necessary. If the number of exhausted VMs is increasing, they should enter into the Grace-Period state for refreshing by the reactive recovery. The proposed scheme monitors the number of VMs in the Grace-Period state. If the ratio of the number of VMs in the Grace-Period state to that in the Active state is more than 80%, the current number of online VMs is not sufficient to keep providing normal services. If so, additional VMs are allocated to process incoming packets. The parameter, 80%, was chosen to handle the situation appropriately without much performance degradation on the system. This parameter can be made smaller, but it induces over-redundant resource usage.

On the other hand, if the number of incoming packets is decreased, the values of the CPU usage and queue length are restored to the normal state. In this situation, a few VMs are refreshed and enter into the Grace-Period state. If the ratio is less than 20%, there are too many VMs to process the incoming packets. In this case, the number of online VMs is reduced to utilize efficiently the restricted resources of the physical server. The parameter, 20%, is chosen because reducing the number of online VMs does not affect the overall system performance significantly when the value is down to approximately 20%.

3. Experiment and Analysis

The configuration for the proposed scheme is as follows. The initial numbers of online and reserved VMs are three and six, respectively. The number of total VMs used is 20. The exposure time is 10 seconds during which the VM can provide services. The cleansing time is the duration to take to mount an initial pristine image to the target VM, which is believed to take 6 seconds regularly in this experiment.

The environmental assumptions were as follows, which were obtained from references [24], [25] for a fair comparison: the mean incoming packet interval was 11.5ms, and the mean process time of an incoming packet was 11.5ms $\times$ 0.99. The exponential distribution with the mean value was used to generate incoming packets and the processing time relatively, which reflects their variety. The ideal values were used to determine if a VM is exhausted, which are as follows: the CPU usage was 0.907 and queue length was 10.58. These values were obtained under environment assumptions. The simulator used was CSIM 20, which is a library of routines to create process-oriented, discrete-event simulators [26].

The response times of the proposed scheme and the existing system employing only proactive recovery approach [7] were measured in the first experiment. During

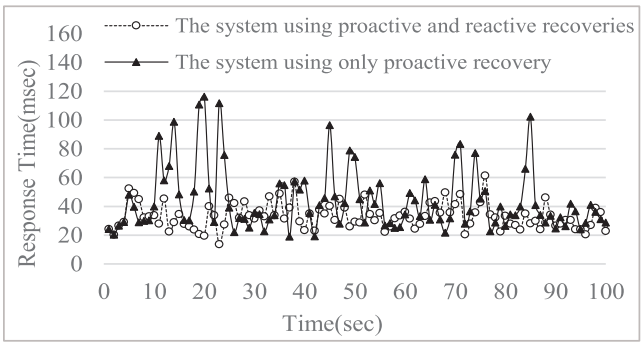

Fig. 2 The results of efficiency.

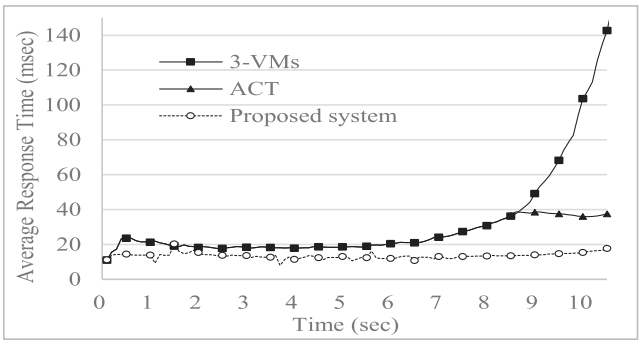

Fig. 3 The results of security against DoS.

a given time, the existing system performed the refreshing process 24 times and the mean response time was 43.577 msec. On the other hand, the proposed system performed the refreshing process 46 times with a mean response time of 33.905msec. Because the proposed system executed almost double the number of refreshing processes due to combination of proactive and reactive recovery, the mean response time was approximately 22% lower than that of the existing system. Additional refreshing processes do not affect the entire performance of the system due to the advanced virtualization technology [27]. Figure 2 shows the results for efficiency.

The purpose of the second experiment is to show whether the proposed scheme continues to provide a service under a DoS attack better, compared to the existing scheme. To simulate a DoS attack, the mean incoming-packet interval time is decreased after processing each packet (Interval = Interval×0.99), as reported in reference [22]. In the experiment, the interval is 3,000 times as short as the initial interval after 10 seconds: the interval becomes 9.253×10^{-13}, which means that 10^{12} packets per second are incoming. In general, this situation is much tougher than a real DoS situation. Figure 3 shows the results for security against a DoS attack.

Under the assumption that the VMs provide the same function and have the same capacities as each other, three types of systems were compared in this experiment. 1) A 3-VMs system consists of three VMs with no recovery and no cluster expansion/reduction mechanism. 2) An ACT system is an existing system with only a cluster expansion/reduction mechanism [24]. 3) The proposed system has a recovery and cluster expansion/reduction mechanism. As the number of incoming packets increases, the response time of the 3-VMs system was increased so gradually that the system could not provide normal services. The ACT and the proposed systems kept providing services in heavy traffic. On the other hand, because the proposed system employs a combination of proactive and reactive recovery by live migration, there were more cluster expansions and a smaller average response time compared to the ACT system. The mean response time of the ACT system was 23.667 msec, whereas that of the proposed system was 13.309 msec, showing 77.83% performance improvement. Because advanced virtualization technology allocates efficiently the restricted resources of the system for creating VMs, many physical servers are not required to implement the proposed scheme, compared to the existing schemes.

4. Conclusion

This letter proposed a novel ITS using live migration, which employs a combination of proactive and reactive recovery. This makes the system more efficient. Moreover, it provides a solution against heavy packet influxes within a short term, such as a DoS attack. Experiments on CISM20 showed that the proposed scheme is more efficient than the conventional system with only proactive recovery and continues the service under a DoS attack.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MEST). (No. 2011-0016584)

References

[1] Q.L. Nguyen and A. Sood, “A comparison of intrusion-tolerant system architectures,” IEEE Security & Privacy, vol.9, no.4, pp.24–31, 2011.
[2] F. Wang, F. Gong, C. Sargor, K. Goseva-Popstojanova, K. Trivedi, and F. Jou, “SITAR: A scalable intrusion-tolerant architecture for distributed services,” Workshop on Information Assurance and Security, vol.1, p.1100, 2003.
[3] P. Pal, F. Webber, and R. Schantz, “The DPASA survivable JBI-a high-water mark in intrusion-tolerant systems,” Proc. 2007 Workshop Recent Advances in Intrusion Tolerant Systems, 2007.
[4] A. Valdes, M. Almgren, S. Cheung, Y. Deswarte, B. Dutertre, J. Levy, H. Saidi, V. Stavridou, and T.E. Uribe, “An architecture for an adaptive intrusion-tolerant server,” Security Protocols, 2004.
[5] P.E. Verissimo, N.F. Neves, C. Cachin, J. Poritz, D. Powell, Y. Deswarte, R. Stroud, and I. Welch, “Intrusion-tolerant middleware: The road to automatic security,” IEEE Security & Privacy, vol.4, no.4, pp.54–62, 2006.
[6] P. Pal, P. Rubel, M. Atighetchi, F. Webber, W.H. Sanders, M. Seri, H. Ramasamy, J. Lyons, T. Courtney, and A. Agbaria, “An architecture for adaptive intrusion-tolerant applications,” Software: Practice and Experience, vol.36, no.11-12, pp.1331–1354, 2006.

[7] Y. Huang and A. Sood, "Self-cleansing systems for intrusion containment," Proc. Workshop on Self-Healing, Adaptive, and Self-Managed Systems (SHAMAN), 2002.
[8] D. Arsenault, A. Sood, and Y. Huang, "Secure, resilient computing clusters: self-cleansing intrusion tolerance with hardware enforced security (SCIT/HES)," Second International Conference on Availability, Reliability and Security, 2007.
[9] P. Sousa, A.N. Bessani, and R.R. Obelheiro, "The FOREVER service for fault/intrusion removal," Proc. 2nd workshop on Recent advances on intrusiton-tolerant systems, 2008.
[10] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," ACM Trans. Comput. Syst. (TOCS), vol.20, no.4, pp.398–461, 2002.
[11] Y. Huang and C. Kintala, "Software implemented fault tolerance: Technologies and experience," FTCS, vol.23, pp.2–9, 1993.
[12] D. Patterson, A. Brown, P. Broadwell, G. Candea, M. Chen, J. Cutler, P. Enriquez, A. Fox, E. Kiciman, and M. Merzbacher, Recovery-oriented computing (ROC): Motivation, definition, techniques, and case studies, Technical Report UCB//CSD-02-1175, 2002.
[13] K.R. Joshi, M.A. Hiltunen, W.H. Sanders, and R.D. Schlichting, "Automatic model-driven recovery in distributed systems," 24th IEEE Symposium on Reliable Distributed Systems, 2005.
[14] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "Live migration of virtual machine based on full system trace and replay," Proc. 18th ACM International Symposium on High Performance Distributed Computing, 2009.
[15] B. Jiang, B. Ravindran, and C. Kim, "Lightweight live migration for high availability cluster service," Stabilization, Safety, and Security of Distributed Systems, 2010.
[16] R. Sindoori, V.P. Pallavi, and P. Abinaya, "An Overview of Disaster Recovery in Virtualization Technology," J. Artificial Intelligence, vol.6, pp.60–67, 2012.
[17] J.-H. Huang, J. Huang, R. Li, and X.-M. Li, "Virtualization-based recovery approach for intrusion tolerance," Information Technology Journal, vol.12, pp.385–390, 2012.
[18] A. Nagarajan, Q. Nguyen, R. Banks, and A. Sood, "Combining intrusion detection and recovery for enhancing system dependability," 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W), 2011.
[19] J.-H. Huang and F.-F. Wang, "The strategy of proactive-reactive intrusion tolerance recovery based on hierarchical model," Web Information Systems and Mining, 2011.
[20] L. Zheng, X. Nie, Y. Tao, and L. Li, "Applications of intrusion-tolerance pre-response in the grid enterprises," International Conference on Computational and Information Sciences (ICCIS), 2012.
[21] P. Sousa, A.N. Bessani, M. Correia, N.F. Neves, and P. Verissimo, "Highly available intrusion-tolerant services with proactive-reactive recovery," IEEE Trans. Parallel Distrib. Syst., vol.21, no.4, pp.452–465, 2010.
[22] A.K. Bangalore and A.K. Sood, "Securing web servers using self cleansing intrusion tolerance (SCIT)," Second International Conference on Dependability, 2009. DEPEND'09. 2009.
[23] Y. Huang, D. Arsenault, and A. Sood, "Incorruptible system self-cleansing for intrusion tolerance," 25th IEEE International Performance, Computing, and Communications Conference, 2006. IPCCC 2006. IEEE, 2006.
[24] J. Lim, Y. Kim, D. Koo, S. Lee, S. Doo, and H. Yoon, "A novel adaptive cluster transformation (ACT)-based intrusion tolerant architecture for hybrid information technology," J. Supercomputing, pp.1–18, 2013.
[25] A. Saidane, V. Nicomette, and Y. Deswarte, "The design of a generic intrusion-tolerant architecture for web servers," Dependable and Secure Computing, vol.6, no.1, pp.45–58, 2009.
[26] H. Schwetman, "CSIM19: CSIM19: a powerful tool for building system models," 33rd Conference on Winter Simulation, 2001.
[27] T. Distler, R. Kapitza, and H.P. Reiser, "Efficient state transfer for hypervisor-based proactive recovery," Proc. 2nd Workshop on Recent Advances on Intrusiton-Tolerant Systems, 2008.