Multi-level VM Replication based Survivability for Mission-critical Cloud Computing

Ming Zhao[1], Francois D'Ugard[1], Kevin A. Kwiat[2], Charles A. Kamhoua[2]
[1] Florida International University, School of Computing and Information Sciences, Miami, FL
[2] Air Force Research Laboratory, Information Directorate, Cyber Assurance Branch, Rome, NY

***Abstract*—The elasticity and economics of cloud computing offer significant benefits to mission-critical applications which are increasingly complex and resource demanding. Cloud systems also provide powerful tools such as virtual machine (VM) based replication for defending mission-critical applications. However, cloud-based mission-critical computing raises serious challenges to mission assurance. VM-based consolidation brings different applications to the same set of physical resources, increasing the risk of one user compromising the mission of another. The mission-critical application in a VM lacks the visibility and control to detect and stop outside malicious attacks, whereas the support for security isolation from existing cloud systems is also limited. The objective of the research presented in this paper is to address these challenges and improve the survivability of mission-critical applications through the novel use of VM replication. Specifically, this paper presents a new multi-level VM replication approach which uses different types of VM clones to provide a variety of protections to mission-critical applications, and improve the survivability of the applications under accidental faults and malicious attacks. In this approach, full VM clones are employed to provide tolerance of attacks, decoy clones are created to divert attacks, and honeypot clones are used to analyze attacks. The paper also presents the prototypes of the proposed approach implemented for the widely used OpenStack-based private cloud systems and Amazon-EC2-based public cloud systems.**

I. INTRODUCTION

Cloud computing systems emerge as important computing platforms because of their elasticity—the ability to dynamically grow and shrink the resources provisioned to an application on demand, and economics—the ability for users to run their applications at scale without up-front or long-term commitment to the resources. Public clouds allow public users to rent resources and run a wide variety of applications (e.g., [1][2]); private clouds allow users from the same organization to run their internal applications on shared resources (e.g., [3]). Like many other applications, mission-critical applications can also benefit from the elasticity and economics of cloud systems. Modern mission-critical applications are increasingly complex and demanding. Examples such as command and control (C2) applications are often required to process large volumes of data and make precise decisions with a stringent performance requirement. As the cost of building and operating dedicated computing platforms becomes prohibitive, it will be inevitable to run mission-critical applications on shared resources from private and public clouds.

To mission-critical applications, the benefits of clouds are beyond just scalability and cost-effectiveness. Cyber-attackers have already been known to use cloud resources to spread malwares and launch attacks (e.g., [4][5]). It will be to the disadvantage of users of mission-critical applications if they cannot effectively use the power of cloud systems to defend themselves—the game between defenders and attackers will be asymmetric and the balance will tip towards the attackers. Cloud systems in fact offer unique, powerful tools for defending mission-critical applications. For example, a virtual machine (VM) hosting the mission-critical application can be replicated conveniently to tolerate attacks; a VM can be transparently checkpointed and recovered in case of failure; and a VM can also be dynamically migrated across cloud systems to avoid attacks from affecting the application.

Cloud-based mission-critical computing is, however, a double-edged sword. Because the VMs hosting different applications share the same set of physical resources, they may interfere with each other in complex and unpredictable ways. Having VMs from different users computing and storing data on the same hardware increases the risk of one user stealing information or compromising the computing or data of another. Further, in a public cloud, the VMs may belong to users from different organizations, which worsens the security risks. Nonetheless, the mission-critical application running in a VM lacks the visibility and control outside of its VM to detect and stop the malicious attacks, whereas the support for security isolation from existing cloud systems is also quite limited.

The overarching goal of our research is to address the above mentioned challenges for delivering mission assurance to mission-critical applications on cloud systems, thereby allowing such applications to benefit from cloud computing and meet their demanding scalability and security requirements. As a first step towards achieving this long-term goal, in this paper, we study new VM-replication-based techniques for mission-critical applications to survive attacks and accomplish missions in cloud systems. We propose a multi-level VM replication approach using different types of VM clones, including full clones, decoy clones, and honeypot clones, to provide a variety of protections to mission-critical applications. We also create a VM replica management system that automatically manages the creation and regeneration of these VM clones in cloud systems. Finally, we implement proof-of-concept prototypes of this multi-level VM replication approach on both typical public cloud systems, based on Amazon EC2 [1], and private cloud systems, based on OpenStack [3].

The rest of this paper is organized as follows. Section 2 discusses the background and related work. Section 3 presents the multi-level VM replication approach. Section 4 discusses

Approved for Public Release; Distribution Unlimited: 88ABW-2014-3491, dated 25 Jul 14. This research is sponsored by an Air Force Summer Faculty Fellowship, an Air Force summer extension grant, the Department of Defense award W911NF-13-1-0157, and the National Science Foundation CAREER award CNS-125394.

the prototypes. Section 5 concludes the paper and outlines the future work.

II. BACKGROUND AND RELATED WORK

A. Virtualization and Cloud Computing

The information technology landscape has been deeply transformed by the resurgence of virtualization [6]. Initially motivated by the need to support legacy systems and improve the utilization of expensive mainframe systems in the early 70s, modern virtualization technologies in the Internet era have enabled resource consolidation, flexible resource management, and on-demand provisioning of infrastructure-as-a-service—in essence, providing the core technology that has paved the way to the implementation and wide adoption of cloud computing. The VMs considered in this paper are system-level VMs, which virtualize an entire physical host's resources, including CPU, memory, and IO devices, and present virtual resources to the guest OSes and applications. System virtualization is enabled by the layer of software called virtual machine monitor (VMM, or hypervisor), which is responsible for multiplexing physical resources among the VMs (e.g., [7][8][9]).

Cloud computing is an emerging paradigm which has the potential to finally realizing the society's long-held dream of delivering computing as a utility. Depending on the level of delivered services, cloud offerings are often classified as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service. An IaaS cloud (e.g., [1]) provides a user access to virtualized hardware, presented by a VMM and encapsulated in a VM, where the user is able to deploy and run arbitrary software including operating systems and applications on the underlying shared hardware. A PaaS cloud (e.g., [2]) provides a user a language-specific platform (e.g., JVM, .Net) to deploy and run arbitrary applications developed using the given language on the underlying shared platform. A SaaS cloud provides a user access to a particular application (e.g., web-based email, document editor) where the user can use the functionality provided by the underlying shared application. Depending on the model of deployment, cloud systems can also be classified as public or private. Public clouds allow public users to rent resources and run a wide variety of applications [1][2]; private clouds allow users from the same organization to run their internal applications on shared resources [3].

Virtualization is the key enabling technology of cloud computing. VMs are directly exposed to users in an IaaS cloud; they are also often used internally in PaaS and SaaS systems for hosting the underlying platform and software and benefiting from the use of virtualization. Although these different levels of cloud services can be built separately, it is increasingly common to build a high-level cloud service using resources provided by a lower-level one (e.g., build a SaaS on resources from PaaS and a PaaS on resources from IaaS), so that the former can benefit from the elasticity and economics provided by the latter. Therefore, although this paper focuses on VM-based hosting of mission-critical applications in an IaaS setting, its outcomes will also generate an impact to other models of cloud computing.

Virtualization and cloud computing are increasingly perceived as valuable to defense applications—from the perspective of resource consolidation, increased efficiencies, reduced power consumption, increased flexibility and agility in deploying applications and workloads. Private clouds can be custom-built to meet the stringent requirements of defense applications (e.g., [10]). Public clouds can also be valuable to defense applications because of their much cheaper cost and better scalability. An analogy can be drawn from the use of private versus public networks which are both found important for defense communications. However, the use of cloud systems for mission-critical applications also raises serious challenges to mission assurance. A public cloud can be assumed to be unsafe and insecure because of its multi-tenancy. Even in a private cloud, virtualization-enabled consolidation brings applications to the same set of shared physical resources and makes them more tightly coupled and consequently more vulnerable, compared to a non-virtualized environment where each application runs on a dedicated host.

The goal of this paper is to study and develop novel techniques that will enable virtualization technologies that support mission-critical applications. The focus is on modern system virtualization technologies available for commodity off-the-shelf computing platforms, but the approaches investigated in this paper are of general applicability. The proposed techniques do not assume special support from the cloud infrastructure and are hence applicable to both public and private cloud systems.

B. Mission Assurance in Cloud Systems

Virtualization can be a powerful tool for enhancing mission assurance, and it has been explored in different ways in the literature. Intrusion detection can be realized at the VMM layer with both good visibility into and isolation from the monitored VMs because VMM can directly introspect the state of the VMs while being protected from the VMs [11][12]. VMM can also be used to log the executions of VMs and replay them for analyzing intrusions to the VMs [13]. These solutions are complementary to this paper's proposed VM-replication-based mission assurance techniques.

VMs can be used as honeypots which are carefully monitored systems that allow attackers to interact so that their behaviors can be analyzed for future security enhancement. VM-based honeypots simplify the containment and isolation of attacks, the reverting of a comprised honeypot to a clean state, and the real-time monitoring of the honeypot. This paper considers honeypot VMs as part of its proposed multi-level VM replication based survivability approach. Related work in the literature has studied techniques to reduce the overhead of creating a large number of honeypots [14][15]. These techniques are complementary to this paper's focus on creating realistic-looking and safe honeypots from mission-critical VMs. Biedermann *et al*. proposed an approach to dynamically extract a honeypot from an attacked VM via live cloning [16], which however is difficult to ensure that all sensitive information is stripped out. In contrast, this paper's approach ensures that the honeypots only look the same as the mission-critical VMs from the resource usage perspective but do not possess any sensitive code or data from the mission-critical applications.

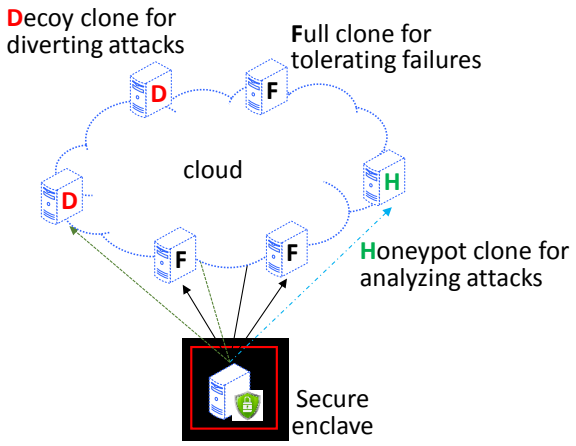

Figure 1. Multi-level VM replication based survivability for mission-critical applications in cloud computing systems

The use of virtualization and cloud computing is also a double-edged sword to mission assurance. From the perspective of a cloud user, there is no guarantee whether the underlying VMM or the co-resident VMs are trustworthy. Various solutions are proposed to address these threats. Li *et al.* proposed several techniques to protect VMs from a untrusted management VM, including modifying the VMM to restrict access to the memory mappings of a VM by the management VM, encrypting the VM's memory pages and vCPU registers, and verifying the hash value of the VM's kernel image [17]. HyperSentry enables stealthy in-context measurement of hypervisor integrity using a hardware channel to trigger the measurement and using the system management mode to protect the measurement agent's base code and critical data [18]. These solutions are complementary to this paper's VM-replication-based protection techniques which do not require any change to the VMM or hardware.

A mission-critical VM may leak sensitive information to its co-resident VMs through side channels caused by imperfect isolation of shared resources among VMs. For example, Ristenpart *et al*. implemented a side channel using shared L1 or L2 cache to detect a co-resident VM's keystroke activities by measuring the load on the shared cache [19]. HomeAlone instead uses the side channel to ensure that all co-resident VMs are friendly VMs which coordinate to silence their activity in a selected cache region so that non-friendly VMs reveal themselves due to their cache activities [20]. This solution, however, requires that all the friendly VMs to be co-resident which is often difficult to achieve and makes them more vulnerable to failures on the hardware and VMM that they all share. In contrast, this paper proposes the use of decoy VM clones that can mimic the behaviors of mission-critical VMs and fool attackers who exploit the side channels.

III. MULTI-LEVEL VM REPLICATION BASED SURVIVABILITY

Virtualization allows the computing and its execution environment to be seamlessly encapsulated in a VM which can be transparently replicated to tolerate failures caused by malicious attacks. This unique capability is employed in our proposed approach to improve the survivability of mission-critical applications. Replication-based fault tolerance is well understood and can be employed to improve the survivability of a mission-critical application in cloud systems via VM cloning. However, the use of full VM clones provides only passive defense for surviving attacks, and it is expensive as its resource cost grows linearly to the number of clones employed. To address these limitations, we propose a multi-level VM replication approach in this paper, which replicates a mission-critical VM using different types of VM clones, including full clones, decoy clones, and honeypot clones, and provides a variety of protection to improve the application's survivability in clouds while making efficient use of resources (Figure 1).

A. *Full VM Clones*

In this approach, a full VM clone is a complete VM replica containing the mission-critical application and its execution environment. It is kept in sync with the other full clones so that it can actually run the application and provide tolerance of attacks. Voting algorithms (e.g., [21]) are employed to derive the correct computing result from the results given by different full clones, assuming attackers cannot compromise the majority of the full clones without being detected. Synchronization across the full clones is used to detect failed or compromised clones, when they become unresponsive (detected by heartbeats) or give wrong results (detected by voting). The failed or compromised full clones will be automatically regenerated to sustain the protection of the mission-critical application. If the mission-critical application is stateless, the regenerated clone can immediately join the computing given the new input data. If the application is stateful, the regenerated clone needs to be synchronized with the other clones before the computing can resume. The new clone can be regenerated from the existing full clones that are still functioning properly and correctly. VM checkpointing can be employed to reduce recovery time by periodically checkpointing the full clones so that when a failure happens the regenerated clone and the remaining clones can be rolled back to the most recent checkpoint and resumed from there.

B. *Decoy VM Clones*

A decoy VM clone is a lean VM replica containing only the necessary software to mimic the behaviors of running the mission-critical application. It is used solely to divert attacks from the real targets, the full VM clones, and waste attackers' time and resources. It can be safely discarded if compromised. Fake data are fed to a decoy clone to also mimic the communication patterns of a full clone. An analogy to the use of decoy VMs can be drawn from the chaff cloud spread by a military aircraft, which are made of cheap materials but can effectively fool the enemy's radar and avoid the attack on the aircraft. Although the use of false-target-based defense has also been studied in the related work [22][23], this paper is the first to study its real use in cloud-based mission-critical computing for survivability.

A key challenge to creating decoy VMs is how to make them realistic-looking enough to fool attackers while at the same time cheap enough (in terms of resource usages) to be employed at scale. From outside of the decoy VMs, they should

be indistinguishable from a real VM to attackers who are in the co-resident VMs. But the inside software stack of the decoy VMs can be reduced to save resource costs, because their purposes are served when they are compromised. Therefore, the key to decoy VMs is that they should look the same from the attacker's perspective while using minimum amount of resources. Because the interfaces that an attacker can use to observe a co-resident mission-critical VM are the side channels provided by shared non-partitionable resources (e.g., CPU caches), the decoy VMs need to behave in the same way only in terms of the use of these resources, but not the partitionable resources (e.g., CPU cycles, memory capacity). For example, to fool an attacker who is monitoring the cache-based side channel, the decoy VM only needs to load the cache, but not the memory, in the same fashion as the mission-critical VM. As a result, the decoys can use much less resources than the full clones because the capacity of non-partitionable resources is typically small and cheap to manipulate.

Specifically, this paper's approach to generating realistic-looking decoy clones is to create a customized cache stub which can be tuned flexibly according to the cache profile observed from the mission-critical application, thereby mimicking the application's cache behavior. Note that the cache stub needs to mimic the cache behavior only statistically but not exactly so that it will not leak any meaningful information that might reveal the application's true activities. To create such a cache stub, we employ the techniques developed by the computer architecture community (e.g., [24]) for profiling application cache behaviors.

C. Honeypot VM Clones

A honeypot VM clone is a partial VM replica containing a sanitized copy of the application and a hidden layer to monitor attacks stealthily. Different from decoy VMs, the purpose of a honeypot VM is to allow attackers to interact with it so that its behaviors can be monitored and analyzed [25][26][27]. The monitoring layer observes the behaviors of attackers and saves such information for vulnerability analysis and protection improvement. Therefore, a honeypot needs to continue functioning even after it is intruded by an attacker. A key question that needs to be addressed is how to create a sanitized version of the mission-critical application, which still behaves like the original application. The application in a honeypot VM is stripped away of all sensitive information so it does not cause any damage to the mission if the honeypot VM is compromised. If the application's code itself is not sensitive, it is possible to simply use the same code in the honeypot while feeding it fake data.

However, a mission-critical application often contains sensitive logic such as the intelligence algorithm for processing data, and thus cannot be used directly in the honeypot. To address this requirement, we employ a fuzzy-logic-based VM performance modeling technique [28] to create a stub program for a honeypot VM that can mimic the resource usage behavior of the mission-critical application without exposing the application's actual code to attackers. The fuzzy modeling technique is able to create an accurate resource usage profile for the mission-critical application, which will be used to customize the stub so that it can generate the same, statistically speaking, patterns in the honeypot VM's use of CPU, memory, and IOs. As a result, the attacker who is inside of the VM cannot recognize the stub and cannot steal any sensitive information while interacting with the honeypots.

As argued by Schneider and Birman [29], having homogeneous clones of a mission-critical VM helps defend against configuration attacks which exploit the errors in the configuration of the mission-critical application and its execution environment. It is much easier to thoroughly examine a configuration and harden it compared to using many different configurations for the same mission-critical application. However, having a "monoculture" in VM clones makes the mission-critical application more vulnerable to technology attacks which exploit the programming or design errors in the application. A monoculture also allows attackers to quickly spread the damage and compromise the entire mission because attacks that succeed on one clone are likely to succeed on all. The use of multi-level VM clones studied in this paper helps defend against such technology attacks as it employs diverse types of clones which make it harder for attacks to spread and compromise the mission, thereby providing support of both monoculture and diversity for defending various types of attacks in cloud systems.

IV. PROTOTYPES

As a proof-of-concept, we have implemented prototypes of the proposed multi-level VM replication approach upon representative private cloud and public cloud systems. In this section, we summarize the key aspects of our prototypes.

A. OpenStack-based Private Cloud

OpenStack [3] is an open-source cloud computing platform commonly used by various organizations for building private cloud systems. OpenStack's Service-Oriented Architecture consists of a series of modules that manage processing, storage, and networking resources of a cloud computing environment. Integration among component services is achieved through application programming interfaces (APIs); each service exposes its API to other services and likewise each service can consume from other services' APIs. Our prototype utilizes the APIs of mainly three OpenStack components, Nova, Glance, and Keystone for creating and managing the different types of replicas of mission-critical VMs in a private cloud. The Nova compute service creates and manages compute resources; the Glance service provides the interface for image discovery, image retrieval, and image storage, and the Keystone Identity service manages the authentication among each component service. Our prototype uses the Grizzly release of OpenStack, and it consists of a series of scripts written in Python which interact with OpenStack via the aforementioned interfaces.

Given a user who wants to execute his/her application in the cloud with multi-level replication in order to avoid and survive attacks, our prototype works as follows. It starts by authenticating against Keystone using the user's credentials in order to manage the VM replicas on behalf of the user. It then requests Nova to create the instances of full clones, decoy clones, and honeypot clones based on the VM images prepared offline. The numbers of instances for each type of clones can be

specified by the user. A unique ID is assigned to each instance. When creating the instances, the prototype also applies the SSH key supplied by the user to the instances. If the key is not supplied, a new key is generated automatically for the user. Once the instances are started, the prototype can log into them using the key to start the application and its stubs.

While the above created clone instances are executing, some of them may fail due to malicious attacks or incidental hardware or software faults. A failure is detected via the heartbeats maintained across the instances on a regular basis or the voting algorithm employed by the full clones. Note that we do not assume that a compromised instance can always be detected by the heartbeat mechanism. Nonetheless, the redundant computing provided by the full clones improves the application's survivability against such attacks.

When a failed instance is detected, the prototype destroys the failed instance and creates a new one in order to restore the user-specified numbers of different types of replicas. However, current OpenStack implementation does not provide the interface to clone an existing VM instance—it allows only the creation of a fresh instance which is then started and booted up from scratch. Replication of a live instance is required for restoring a full clone, because an instance started from scratch cannot catch up with the progress that the rest of the clones have already made in computing. Therefore, the restored instance must be created from the remaining full clone instances. In order to support such live replication in our prototype, we have added a new API to OpenStack, which works as follows.

The live replication method first suspends all the remaining instances to temporarily pause the computing and synchronize their in-memory state to the host's storage. It then replicates one of the full clones by copying its memory state and disk state stored on its host. It also provisions the replica with CPU, memory, and I/O resources identical to the target instance being replicated. Before resuming the new replica, it needs to do the necessary bookkeeping. It assigns the new replica a unique ID in order to identify it in OpenStack. Because OpenStack saves an instance's configuration file as part of its memory state when suspending the instance, this configuration file needs to be extracted out of the memory state, modified to reflect the new instance's identify, and injected back into the memory state, before the new instance can be resumed and recognized by OpenStack. Moreover, it needs to change the new replica's network configuration to avoid conflict with the existing one. To do so, it assigns the new replica a unique MAC address in its configuration, and log into the instance to modify the MAC and apply the change by restarting the network. Finally, once the bookkeeping is done, all the other replica instances are resumed and the computing proceeds as normal.

B. Amazon EC2 based Public Cloud

We have also implemented a proof-of-concept prototype for the proposed multi-level VM replication based survivability approach on Amazon EC2 [1]. EC2 is one of the most widely used public IaaS providers. Users can rent VM instances from EC2 of different sizes, ranging from micro to extra-large, and with different costs. These instances can be created from user-uploaded VM images or customized based on EC2-provided templates. The VMs can be stored on either Amazon's EBS [30] or S3 [31] storage service.

Because the interface exposed by Amazon EC2 is quite similar to the interface of OpenStack, we omit most of the discussion of our EC2-based prototype for brevity. One limitation of applying our multi-level replication approach to a public cloud is that we cannot change its interface to implement the live replication discussed in the previous section. As a result, a lost full clone instance can only be recovered by starting a new instance from scratch. This scheme supports only applications that are stateless, in which case the restored instance can still join the remaining full clone instances for computing.

V. CONCLUSIONS AND FUTURE WORK

Cloud computing offers serious challenges and at the same time great opportunities to mission assurance. In this paper, we have studied the use of VM replication to improve the survivability of mission-critical applications in cloud systems through a new multi-level VM replication approach. In this approach, different types of VM clones, including full clones, decoy clones, and honeypot clones are created to provide a variety of protections to a mission-critical application. We have implemented proof-of-concept prototypes on the widely used OpenStack-based private cloud and Amazon EC2 based public cloud. Although the prototypes are demonstrated to be working in our preliminary testing, we will conduct more rigorous evaluation on their performance and survivability in our future work. In the long term, we will also consider how to leverage other unique capabilities of VMs, such as live migration, to further improve the survivability of mission-critical applications in cloud computing.

The benefits of virtualization and cloud computing are already well understood for many endeavors, but these emerging technologies are still largely underexplored for mission-critical applications. The outcome of our research will enable a broad range of mission-critical applications on cloud systems with mission assurance while under a variety of malicious attacks. Our proposed approach does not make assumptions on the trustworthiness of the cloud environment and are hence applicable to both public clouds built for general purposes and private clouds specially built for mission-critical computing (e.g., [10]).

VI. REFERENCES

[1] Amazon Elastic Compute Cloud (Amazon EC2), URL: http://aws.amazon.com/ec2/.

[2] Windows Azure Platform, URL: http://www.microsoft.com/windowsazure/.

[3] OpenStack Open Source Cloud Computing Software. URL: https://www.openstack.org/.

[4] "Hackers using cloud networks to launch powerful attacks," Homeland Security News Wire, URL: http://www.homelandsecuritynewswire.com/hackers-using-cloud-networks-launch-powerful-attacks.

[5] "Amazon.com Server Said to Have Been Used in Sony Attack", Bloomberg, URL:

http://www.bloomberg.com/news/2011-05-13/sony-network-said-to-have-been-invaded-by-hackers-using-amazon-com-server.html.

[6] R. Figueiredo, P. Dinda, J. Fortes, "Resource Virtualization Renaissance", IEEE Computer Magazine 38(5), Special Issue on Virtualization, pp. 28-31, May 2005.

[7] VMware Inc., URL: http://www.vmware.com.

[8] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the Art of Virtualization", in Proc. of the ACM Symposium on Operating Systems Principles (SOSP), October 2003.

[9] Kernel Based Virtual Machine, URL: http://www.linux-kvm.org/page/Main_Page.

[10] "Amazon Building Private Cloud for CIA", Business Insider, URL: http://www.businessinsider.com/amazon-building-private-cloud-for-cia-2013

[11] Garfinkel, Tal, and Mendel Rosenblum, "A Virtual Machine Introspection Based Architecture for Intrusion Detection," NDSS, 2003.

[12] Jiang, Xuxian, Xinyuan Wang, and Dongyan Xu, "Stealthy Malware Detection through VMM-based Out-Of-The-Box Semantic View Reconstruction," Proceedings of the 14th ACM Conference on Computer and Communications Security," 2007.

[13] Dunlap, George W., et al., "ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay," ACM SIGOPS Operating Systems Review 36.SI (2002): 211-224.

[14] Vrable, Michael, et al., "Scalability, Fidelity, and Containment in the Potemkin Virtual Honeyfarm," ACM SIGOPS Operating Systems Review, Vol. 39, No. 5, ACM, 2005.

[15] Lengyel, Tamas K., Justin Neumann, Steve Maresca, and Aggelos Kiayias, "Towards Hybrid Honeynets via Virtual Machine Introspection and Cloning," The 7th International Conference on Network and System Security (NSS'13), pp.164-177, 2-3 June 2013.

[16] Sebastian Biedermann, Martin Mink, and Stefan Katzenbeisser. 2012. "Fast Dynamic Extracted Honeypots in Cloud Computing." In Proceedings of the 2012 ACM Workshop on Cloud computing security workshop (CCSW '12). ACM, New York, NY, USA, 13-18.

[17] Chunxiao Li, Anand Raghunathan, Niraj K. Jha, "A Trusted Virtual Machine in an Untrusted Management Environment," IEEE Transactions on Services Computing, vol. 5, no. 4, pp. 472-483, Fourth Quarter 2012.

[18] Ahmed M. Azab, Peng Ning, Zhi Wang, Xuxian Jiang, Xiaolan Zhang, and Nathan C. Skalsky. 2010. "HyperSentry: enabling stealthy in-context measurement of hypervisor integrity." In Proceedings of the 17th ACM conference on Computer and communications security (CCS '10). ACM, New York, NY, USA, 38-49.

[19] Ristenpart, Thomas, et al., "Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds," Proceedings of the 16th ACM conference on Computer and communications security. ACM, 2009.

[20] Yinqian Zhang, Ari Juels, Alina Oprea, and Michael K. Reiter, "HomeAlone: Co-residency Detection in the Cloud via Side-Channel Analysis," In Proceedings of the 2011 IEEE Symposium on Security and Privacy (SP'11). IEEE Computer Society, Washington, DC, USA, 313-328.

[21] Wang, Li, et al., "Optimal Voting Strategy against Rational Attackers," 6th International Conference on Risk and Security of Internet and Systems (CRiSIS), 2011.

[22] Levitin, Gregory, and Kjell Hausken, "False Targets Efficiency in Defense Strategy," European Journal of Operational Research, 2009.

[23] Wang, Li, et al., "Optimal resource allocation for protecting system availability against random cyber attacks," 3rd IEEE International Conference on Computer Research and Development (ICCRD), 2011.

[24] Govindan, Sriram, et al., "Cuanta: Quantifying Effects of Shared On-Chip Resource Interference for Consolidated Virtual Machines," Proceedings of the 2nd ACM Symposium on Cloud Computing.

[25] Vrable, Michael, et al., "Scalability, Fidelity, and Containment in the Potemkin Virtual Honeyfarm," ACM SIGOPS Operating Systems Review, Vol. 39, No. 5, ACM, 2005.

[26] Lengyel, Tamas K., Justin Neumann, Steve Maresca, and Aggelos Kiayias, "Towards Hybrid Honeynets via Virtual Machine Introspection and Cloning," The 7th International Conference on Network and System Security (NSS'13), pp.164-177, 2-3 June 2013.

[27] Sebastian Biedermann, Martin Mink, and Stefan Katzenbeisser. 2012. "Fast Dynamic Extracted Honeypots in Cloud Computing." In Proceedings of the 2012 ACM Workshop on Cloud computing security workshop (CCSW'12).

[28] L. Wang, J. Xu, M. Zhao, Y. Tu, and J. Fortes, "Fuzzy Modeling based Resource Management for Virtualized Database Systems," Proceedings of the 19th Annual Meeting of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2011), July 2011.

[29] Birman, Kenneth P., and Fred B. Schneider, "The Monoculture Risk Put into Context," IEEE Security & Privacy, 2009.

[30] Amazon Elastic Block Store (EBS), URL: http://aws.amazon.com/ebs/.

[31] Amazon Simple Storage Service (Amazon S3), URL: http://aws.amazon.com/s3/.