ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 8, Issue 2, February 2020

A Survey on Securing the Virtual Machines in Cloud Computing

Deepesh Shrivas[1], Prof. Ajeet Singh[2]

M.Tech. Scholar, Department of Computer Science & Engineering, Global Engineering College, Jabalpur, Madhya Pradesh, India [1]

Assistant Professor, Department of Computer Science & Engineering, Global Engineering College, Jabalpur, Madhya Pradesh, India [2]

ABSTRACT: Cloud computing may be defined as management and provision of resources, software, applications and information as services over the cloud (internet) on demand. Cloud computing is a model for enabling convenient, on demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. With its ability to provide users dynamically scalable, shared resources over the Internet and avoid large upfront fixed costs, cloud computing has recently emerged as a promising hosting platform that performs an intelligent usage of a collection of services, applications, information and infrastructure comprised of pools of computer, network, information and storage resources. However along with these advantages, storing a large amount of data including critical information on the cloud motivates highly skilled hackers thus creating a need for the security to be considered as one of the top issues while considering Cloud Computing. In this paper we explain the cloud computing along with its open secure architecture advantages in brief and emphasize on various security threats in cloud computing also the existing methods to control them along with their pros and cons.

I. INTRODUCTION

Virtualization of servers in the cloud operates by adding a new layer to the software stackknown as the hypervisor [1] or Virtual Machine Monitor (VMM) [2]. The hypervisor encapsulates the hardware, allowing it to be used by multiple operating system instances concurrently. This flexibility, coupled with the cost and performance advantages of sharing the underlying hardware, has revolutionized the computing industry: large numbers (i.e. hundreds of thousands) of generic hardware platforms, using multi- core blade technology, are now coupled through high-performance networking to produce a generic computing surface. Any subset of this collection can be combined to operate in tandem for a particular application using a multitude of operating systems.

Conceptually, the hypervisor presents a virtual machine abstraction that restricts malicious code, executing within one instance of an operating system, from affecting a different instance. Unfortunately, hypervisors have introduced their own new security challenges: Adversaries now actively attempt to detect the presence of an operating hypervisor in order to tailor attacks accordingly [3]. A wide range of hypervisor detection techniques have already appeared against popular systems such as VMWare, VirtualPC, Bochs, Hydra, Xen, and QEMU [4]. Often, these techniques operate by exploiting timing differences between virtualized and non-virtualized operations [5]. Alternatively, they detect unusual memory locations associated with key operating system data structures [6]. For example, the Red Pill technique works by using the SIDT X-86 instruction to determine the location in memory of the interrupt descriptor table; a machine running above a hypervisor will return a location much higher in memory than one that is not [7]. Following hypervisor detection, the adversary then attacks either the operating system, the virtual switch (vSwitch) sharing network connectivity bet- ween virtual machines, or the hypervisor itself [8].

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 8, Issue 2, February 2020

The presence of a hypervisor has no impact on the vulnerabilities associated with the operating system. As a result, any exploit that leverages a known vulnerability will still operate successfully [9]. Although, a remote exploit gives the adversary control of a single virtual machine, by using the exploit in a virus the entire cloud could be compromised. It is this vulnerability amplification that poses the most significant threat to the future of cloud computing.

Direct attacks against a vSwitch may undermine the operation of multiple virtual machines on a single host by denying connectivity to all of them simultaneously. The vSwitch provides the same functionality as a phy- sical switch and in consequence exhibits the same vulnerabilities, enabling the same exploits [10]. For example, Address Resolution Protocol (ARP) spoofing, involves the interception of valid network packets by sending fake ARP packets to a switch [11].

Hypervisor attacks involve the direct exploitation of vulnerabilities in the hypervisor. All virtual machines executing on a hypervisor have distinct data structures, separated in hardware. This separation forms a semantic gap [12] that prevents virtual machines from having visibility or impact upon each other's data structures [13]. Direct Kernel Structure Manipulation (DKSM) bridges the semantic gap by patching virtual machine data structures and redirecting hypervisor accesses to shadow copies. This allows the virtual machine to present false information to the hypervisor regarding the virtual machine state, allowing implants, such as rootkits [14], to persist without detection.

Virtualization provides inherent redundancy and ap- pears to provide robust, large-scale, cost-effective avail- ability of shared resources [15]. However, this perception is tempered by the known risk of vulnerability amplification and the paucity of knowledge regarding zero-day exploitation in clouds: history has shown that lack of detection does not imply lack of infection. Current mitigation techniques reviewed by this paper have already evolved based on malware detection and prevention, se- cure virtual machine managers, and cloud resilience. These three categories and their roles in preventing an attacker from gaining access to the cloud is illustrated in Figure 1. Omitted from Figure 1 are cloud services that provide authentication such as lightweight active directory protocol servers and trusted computing techniques as they are outside the scope of this survey. Initially, the attacker has to overcome or bypass the intrusion detection and prevention systems typically employed at the cloud boundary. They are then faced with a secure hypervisor usually installed on a single host; whose purpose is to restrict access to kernel and hypervisor data structures. Finally, cloud resilience is used by a host to restore a single compromised or failed virtual ma- chine to a known good state. Although not currently prevalent throughout the industry, hypervisors offer the opportunity to restrict the attacker's access to the base of the software stack. Since typically the number of vulnerabilities is directly related to the number of source

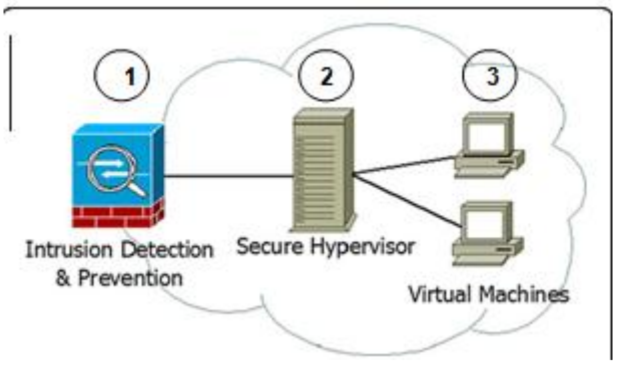

Figure 1 The three cloud security techniques reviewed by this paper: intrusion detection & prevention, secure hypervisors, and virtual machines.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 8, Issue 2, February 2020

lines of code [16], this would allow tight control of the hardware and allow operating system designers to build successive layers on a secure base of trust. The small size of the hypervisor also opens the door to formal reasoning concerning its security properties [17]. Unfortunately, these ideas have yet to be cohesively integrated and their impact upon security quantified. In the sections that follow we explore the building blocks that are available for improving cloud security and assess them on the basis of their performance impact, ability to reduce the attack sur- face, detect known and zero-day threats, resolve detected threats, and increase attacker workload by denying either surveillance or persistence.

III. THREAT MODEL

The security implementation analyzed in this survey address the threat model for intrusions employing remote control outlined in Figure 2. It may involve several steps including surveillance to determine if a vulnerability exists [18], use of an appropriate exploit or other access method [18], privilege escalation [19], removing exploit artifacts, and hiding behavior [14]. Surveillance may involve obtain- ing a copy of the binary code and using reverse engineering [20,21] or fuzzing [22] to facilitate a broad range of attack vectors including return oriented programming [23]. The implant then persists for a time sufficient enough to carry out some malicious effect, obtain useful information, or propagate intrusion to other systems.
Unlike the time to execute an exploit, the time spent in surveillance and persistence may range from minutes to months or even years depending upon the intended effect. Moreover, the presence of an intrusion may never be detected by network defenses but instead may be recognized indirectly due to either a deviation from expected behavior, or may be derived from intelligence sources.

Nevertheless, each cloud security technique represents an integral building block in the multilayered defense of

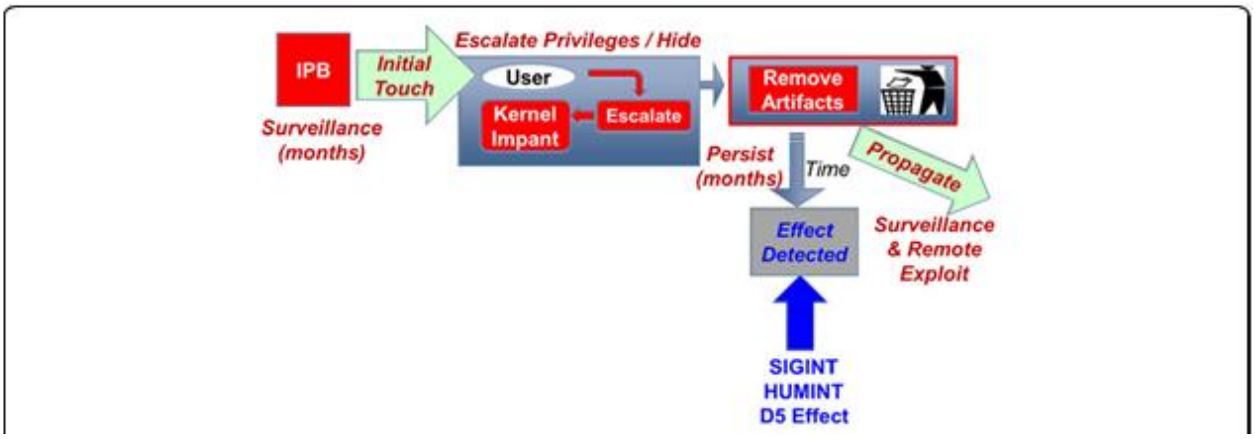

Figure 2 The threat model, detailing the process from surveillance to exploitation in the cloud.

the cloud. Malware detection and prevention systems are the initial line of defense in preventing an attacker from gaining a foothold on a cloud. The secure hypervi- sors present a hardened code base that restricts access to hardware to all, but the most privileged operations. Lastly, cloud resilient solutions are present to protect against the unknown exploits, which may allow an at- tacker to operate on a cloud indefinitely.

Malware detection and prevention
Malware detection was one of the first techniques implemented after the introduction of hypervisors. To achieve this, researchers paired the proven technology of Intrusion Detection Systems (IDS) with the ability to hide in a virtual machine. In this scenario, the IDS still performs the same function of identifying patterns of malicious behavior on a system that may be compromised [24]; for example a proof of concept based on the Snort IDS successfully

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 8, Issue 2, February 2020

prevented a Distributed Denial of Service (DDoS) attack [25]. This implementation installed a virtual machine that ran Snort on top of the VMware hypervisor to monitor network traffic to all guest virtual machines attached to a virtual switch. Once running, the IDS dealt with DDoS attacks in two steps: Initially, attacking computers were blocked by Snort; subsequently, the virtual server automatically moved the application under attack to a new location in the cloud. This demonstrated that an IDS can function inside the cloud; however, the implementation was just as vulnerable to zero-day attacks as non-virtualized IDS's [26]: attacks were missed due to IDS configuration and the failure of signatures to detect new attacks.

The Hybrid Virtual IDS is a solution that leverages the strengths of the cloud and improves upon the previous Snort implementation [27]. The approach combines resilience of a virtual IDS and the versatility offered by a host based IDS. This is possible through the use of integrity checking [28] and system call trace analysis
[29]. Integrity checking is a static detection process in which a changed file is compared to a gold standard to determine if the change is malicious. System call trace analysis dynamically flags anomalous system call behavior as potentially dangerous. These two approaches are implemented inside of a virtual machine to provide an isolated environment. A custom hypervisor is then used to ensure the isolation between all virtual machines. To provide functionality to the IDS, the hypervisor has hooks that allow the inspection of other guest virtual machines running on the hypervisor. This allows the hybrid virtual IDS to remain isolated from other running virtual machines, while still allowing it to access data from the virtual machines it is monitoring. This technique performed well in testing conducted by the authors of the Hybrid Virtual IDS, but returned unexpected performance results: as the IDS decreases the length of time between inspecting of the monitored virtual machine, the workload processing time did not increase linearly as to be expected and instead became erratic. The cause of this erratic performance is open to additional research.

With the introduction of a hypervisor and a virtualized IDS, it was only a matter of time before firewalls were moved into the cloud. One of these virtual firewall implementations is VMwall [30], which runs in the privileged virtual machine that controls the Xen hypervisor and uses virtual machine introspection [31]. This is the process of inspecting the data structures of a separate virtual machine. To enable this functionality, the Xen hypervisor has added hooks that capture all network connections created by a process. The data pertaining to these connections is then passed to VMwall for analysis. The connection is either allowed or blocked by using a whitelist (a list of approved processes and connection types). To deter false data during introspection, kernel integrity checking [32] is used to verify the state of kernel data structures in the guest virtual machines. This is necessary, as the primary method of inspecting trafficis through these data structures; malicious modification may compromise the monitoring of traffic. However, VMwall may be vulnerable to hijacking of a whitelisted process or an already established connection. The only method of detection against the compromise of an approved process is through the checksumming of the in-memory image of that process. This is performed by ensuring that the hash of a process has not changed from that of one contained in the whitelist. Due to the performance impact of hash analysis, this method is generally not implemented. Hijacking an established connection can be partially prevented through time outs associated with kernel rules contained in the whitelist. To fully prevent this type of compromise, deep packet inspection could be used, but is not currently employed by VMwall. Importantly, the employed introspection techniques cause a minimal performance impact: the add- itional overhead is 7% for file transfers from hypervisor to guest and 1% for file transfers from a guest to the hypervisor. Added overhead for Transmission Control Protocol (TCP) and User Data Protocol (UDP) connections are negligible; increases are measured in microseconds.

An alternative approach to detection techniques, like VMwall and hybrid IDS, are prevention methods. One security appliance that performs prevention is Malaware, which is designed to prevent malware that tailors attacks upon detection of a hypervisor [33]. To deter this initial identification of a virtual environment, a signature based method is used. In this instance, a signature is an in- struction that should not be executed by an unprivileged process. As an example, when a process such as Red Pill attempts to run the SIDT instruction, it will be flagged as malicious. However, as the authors of Malaware have stated, a signature based approach is only effective against known

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 8, Issue 2, February 2020

types of malware. To combat zero-day threats, two behavior based approaches that utilize dy- namic analysis are proposed [34]. This could be accomplished by first learning the current process and its page table base address. With this, it is possible to check if the current instruction register belongs to the process' code pages. If this mapping does not exist, Malaware could flag the process as malicious. The second dynamic analysis method suggested is taint tracking. Changes to the system, otherwise known as taint, are created, when a process modifies any code or memory location. Ac- cordingly, when taint is created in monitored locations, the offending process is immediately flagged as malicious. An added benefit of taint tracking is it defeats malicious code that has been transformed to look harm- less, also known as code obfuscation [35]. Once loaded into a monitored region, the obfuscated code is immediately marked as tainted and the associated process is flagged as malicious. Unfortunately, only the signature based piece of the detection has been implemented andno data relating to added overhead has been collected. However, the initial detection results were promising with a malware detection rate of 76%. Lastly, it is important to note that techniques that alter known memory states, such as address space layout randomization (ASLR) may increase the difficulty of this type of taint tracking [36].

Another prevention method, guest view casting [37], moves malware prevention from the guest virtual machine to the hypervisor. This approach reconstructs the data structures of the guest for analysis at the hypervisor level. This is achieved by translating guest virtual memory addresses to physical memory addresses, then reading the raw data from the guest's virtual hard drive. The reassembled state in the hypervisor can then be compared to the guest's state using viewing tools such as Windows Task Manager and memory dump to display all processes in memory. The presence of discrepancies between the two states may indicate the existence of malware in the guest. The authors have labeled this method of searching for discrepancies between states as view comparison-based malware detection. An outgrowth of this method is to use anti-virus software to scan the guest's state from inside of the hypervisor. The use of anti-virus outside of the guest shows that it identifies malware more effectively than anti-virus running inside a virtual machine. Additionally, performance of anti-virus is improved outside of the virtual machine. The primary drawback to this approach is the assumption that the hypervisor has not been compromised. The authors agree that malicious code that targets the hypervisor [38] can compromise their approach.
Although detection and prevention are important, the last two decades have demonstrated that it is unlikely that malware can be eliminated completely [39]. Security researchers in an attempt to understand these attacks have to rely on system logs that lack integrity [40] and are often incomplete [41]. The ReVirt IDS [42], which runs on UMLinux [43]; was created in an attempt to improve upon these inadequacies. This is accomplished by creating logs for all of the relevant system level information needed to replay what transpired at an instruction by instruction level for a specific virtual machine. This allows administrators to determine all the relevant information pertaining to an attack. The overhead of performing these functions is 13-58% for kernel tasks and up to 8% for logging tasks.

Secure virtual machine managers

Hypervisors have afforded researchers with new security capabilities. However, the hypervisor itself has come under attack as a way of gaining control of a system [44]. This has led to the introduction of Secure Hypervisors that reduce the attack surface and increase reliability by reducing the number of lines of code [16]. sHype [45],designed by IBM, increases security by taking the idea of control flow enforcement first seen in SELinux [46] and applying those controls on information flows between virtual machines through a mandatory access control model. Using intricate security policies; unfortunately, these make it difficult to guarantee security and can be over 50,000 lines of code [47]. To remove this level of complexity, sHype affords the same control flow protections, but at the hypervisor level and without the need of a policy administrator. These information flows are maintained through the use of a reference monitor that decides what connections to accept and deny between virtual machines. The sHypeapproach creates a flexible architecture, which allows it to support many different security modules [48]. This is accomplished in around 11,000 lines of code; SELinux alone is over 85,000 lines of code.

The performance impact ofsHype enforcement policies is less than 1% [45]. However, sHype's primary shortfall is that it does not completely protect against unauthorized transfer of information between two virtual machines that are not allowed to share information. Figure 3 illustrates the problem: nodes A, B, and C represent three different virtual machines and all are connected to a reference monitor. Virtual machines A and B are not allowed to share information, but both are allowed to share information with virtual machine C. A covert channel is created, when virtual machine C acts as an intermediary and passes information between A and B. In this case the reference monitor would not intervene, as it only sees information being transferred from A to C and from C to B. Fortunately, the addition of a Chinese wall (communication rules) can be added to sHype to protect against this covert channel [49]. In this case, the rule would only allow two of the three virtual machines to run at any one time. However, this method has the drawback of causing a decrease in performance of up to 9.1% [50].

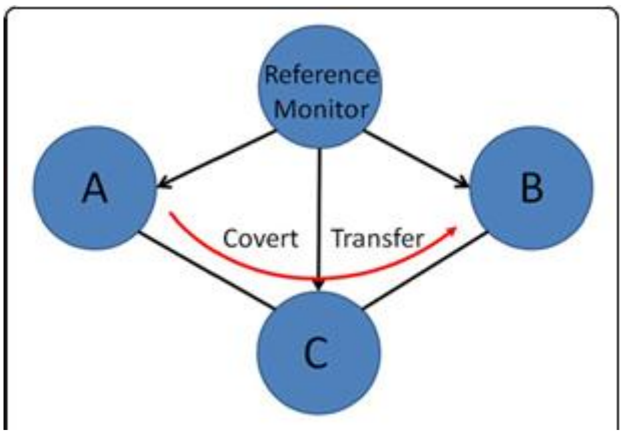

Figure 3 An example of a covert channel, where node A transfers information to node B, through the intermediary node C.

This performance impact can be mitigated by perfor- ming Chinese wall policy checks at virtual machine creation and then caching these decisions. Since, pol- icy changes are infrequent, this configuration reduces the performance impact to less than 1% [51].
A different direction from control flow enforcement is used in the noHype hypervisor [52]. This minimalist approach removes as much as possible from the hyper- visor; unfortunately, no published numbers for lines of code are available. However, the first prototype was based on a stripped down version of Xen 4.0; implying that it falls somewhere less than 1.6 million lines of code [53]. The code count was reduced by shrinking the size of the hypervisor by following four rules. First, noHype pre-allocates processor cores and memory to virtual ma- chines. This allows the virtual machine to control its own hardware, which improves performance. Second, each virtual machine is assigned its own I/O device. Being in the cloud, it is assumed that these virtual machines only need network interface cards (NIC). The issue here is that servers have a limited number of NICs. Thankfully, newer NICs take advantage of Single-Root I/ O Virtualization [54], which allows them to present themselves as multiple NICs. Thus, each virtual machine on a server is able to receive its own NIC, even if there are more virtual machines than NICs. Third, noHypeprovides the user with a predefined guest virtual ma- chine in order to control the discovery of hardware. This also prevents a user from uploading a malicious guest virtual machine, which could attack the hypervisor. Lastly, noHype avoids indirection that occurs through the creation of virtual cores and memory, since cores and memory are assigned directly to each virtual machine. These four principles were tested against a standard Xen 4.0 install and startup time was reduced by 1% in the noHype implementation. However, noHypeloses the ability to perform any introspection of the guest virtual machines as the hypervisor is limited in functionality. Thus, a virtual machine in the noHype cloud could become infected without noHype being aware of the infection.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 8, Issue 2, February 2020

Another popular feature of the cloud is live migration of virtual machines [55]. This can be seamlessly accomplished with little downtime thanks to virtualization. However, migrations lose the states maintained by stateful firewalls [56] and IDS' [57]. These states can be maintained using a network security enabled hypervisor (NSE-H) designed on top of the Xen hypervisor [58]. This builds on the concepts used in secure hypervisors, but adds support for secure file transfers. The perform- ance impact of this method is measured in downtime, which is the time a virtual machine is not available during transfer. The cost of securing these migrations is up to a 15% increase in downtime versus downtime of
non-secure transfers [58]. This downtime occurs for two reasons when maintaining the security context of the virtual machines being migrated. The first is the add- itional time needed to securely copy a virtual machine's memory space from one host to another. The second is the NSE-H security additions, as they are using additional resources on the system.

Cloud resilience
An often over-looked aspect of cloud computing is Resilience, defined as the ability for a system to recover and continue to provide services when a loss of hard- ware or software occurs [59]. One such system, Cloud Resilience for Windows (CReW) [60], expands the idea of resilience to the security domain through the use of strong security in guest virtual machines [61], and intro- spection [62]. Implementation is on top of the 270,000 plus lines of code that comprise the kernel-based virtual machine hypervisor [63]. This has enabled CReW to effectively prevent attacks from some rootkits and repair any damage they may have caused, but at a cost to performance as the number of virtual machines increases or security level is raised. At a strict level with three virtual machines, CReW adds ~48% increase in time needed for CPU tasks and ~279% increase in time required for I/O related tasks. For the paranoid setting, CReW adds ~116% increase in time for CPU related tasks and adds ~347% increase in time for I/O related tasks [60].

A technique that builds upon the ideas presented in CReW and supports other operating systems is that of hypervisor-based efficient proactive recovery [64]. This approach makes the assumption that no matter what defense is implemented on the cloud, a machine will eventually be maliciously compromised or taken offline. Thus, after particular failure conditions are met, the guest virtual machine is refreshed from a gold standard. A prototype of these concepts was developed using a modified Xen hypervisor [65]. Testing has shown there is a balance between throughput and availability. Thus, a user of this method can choose between lower through- put and higher availability or higher throughput and lower availability when faults occur.

The Bear operating system is a minimalist implemen- tation that builds resiliency on top of a secure hypervisor [66]. A key design choice is the strong enforcement of separating core functionality into four layers, which is typical of modern micro kernels, like the MINIX opera- ting system [67]. Importantly, the attack surface is reduced with a shared code base (>50%) of 10,903 lines of code shared between the Bear Hypervisor and Kernel. The size is attributable to a small custom hypervisor and small custom kernel. Resiliency is derived from non- deterministically refreshing the virtual machines on thehypervisor to a gold standard after a period of time. This refresh is done by starting a second virtual machine from the known valid state and then transferring func- tionality to it, all while simultaneously tearing down the first virtual machine. By using this method, control is seamlessly transferred between virtual machines and without an impact to performance. Also, any known or zero-day malware present on the torn down virtual ma- chine will not be present on the newly started virtual machine.

Comparative analysis
Table 1 presents a summary comparison, of the approa- ches based on reduction of the attack surface, prevention of zero-day threats, and overhead. The "Reduces Attack Surface" category shows that all of the technologies other than sHype and Bear rely on a large code base. This poses a concern, as demonstrated by the authors of "Reliability Issues in Open Source Software", who have shown that errors occur at a rate of .09 defects per thousand lines of open source code. This problem is worse for closed source systems, with .57 defects per thousand lines of code. Although the numbers will vary with code base naturally, this result that indicates Xen will have144

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 8, Issue 2, February 2020

defects, KVM 25, UMLinux 162, sHype and Bear each present a single defect. An interesting comparison was provided between open source software and closed source software. Due to the partial unintended release of300,000 lines of VMware kernel code; the code could contain up to 171 defects, which is more defects then a full install of UMLinux. Obviously, sHypeand Bear sys- tems are a bare minimum install and have less function- ality when compared to the other hypervisors. This has led to the sHype architecture being ported to the Xen hypervisor by the authors of "Building a MAC-Based Security Architecture for the Xen Open-Source Hypervisor", which has the net effect of increasing functionality and potential number of defects. The key takeaway is that a small code size and open source distribution are desir- able to prove a system to be reliable and secure. How- ever, closed source systems, which are outside of the purview of this article, do exist and provide similar secu- rity features. Two such commercial hypervisors not reviewed are Citrix XenClient andHyTrust.
After evaluating each system on its abilities to perform "Malware Detection" and "Prevents Zero-Days"; there were two clear outliers. Malware detection and prevention methods primarily protect against known threats, because of their use of whitelists and signatures. However, ReVirt is the outlier in this category, as it provides capabilities to remove zero-days; unlike its counterparts, it has no ability to detect malware. Secure hypervisors restrict access to the hypervisor but generally provide no malware detection abilities or zero-day prevention. Lastly, resilient systems

Cloud security implementation	Reduces attack surface	Malware detection	Mitigates zero-day threats	Added overhead (%)
	(lines of code)			
Malaware	> 725 K	Yes	No	No data
Guest view casting	> 1,600 K	Yes	No	Reduced up to 70%
Virtual snort	> 300 K	Yes	No	No data
Hybrid IDS	> 300 K	Yes	No	~4-36%
VMwall	~ 1,600 K	Yes	No	1-7%
ReVirt	~ 1,800 K	No	Yes	8-58%
NSE-H	> 1,600 K	No	No	15%
Shype	~ 11 k	No	No	< 1%
Shype with Chinese wall in critical path	> 1,600 K	No	No	9.10%
Shype with Chinese wall outside critical path	> 1,600 k	No	No	< 1%
NoHype	< 1,600 K	No	No	Reduced up to 1%
CReW	> 270 K	Yes	Yes	~48-347%
Hypervisor-based proactive recovery	~ 1,600 K	Yes	Yes	~8-12.7%
Bear	~ 11 k	Not applicable	Yes	< 1%

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 8, Issue 2, February 2020

such as CReW and hypervisor based proactive recovery have shown promising results in both categories. The model of whitelists and signatures is replaced with restor- ation upon detection of anomalous system behavior. Thus, both known malware and zero-days are removed from the system when it is restored to a valid state. Resilient sys- tems do not prevent the initial compromise from known threats, unlike malware prevention and detection systems. The outlier in this group is Bear, which makes no attempt to check for anomalous behavior. Instead, it assumes the system will eventually be compromised and therefore refreshes the system non-deterministically. This has the same end result of removing any known or zero-day at- tacks that may be present, but also invalidates surveillance and prevents persistence. Nevertheless, the effectiveness of resilient systems warrants further research. The final category of "Added Overhead" is important, as no technique should overly impact system performance. Both secure hypervisors and malware prevention and detection schemes can minimally impact and in some cases improve performance. The larger resilient prototypes such as CReW and hypervisor proactive recovery have not yet reached this level of performance. Bear however, has had a negligible impact on perform- ance when refreshing virtual machines. Research into future resilient system implementations should aim to maintain the performance levels set by intrusion detection and prevention systems, secure hypervisors, and the Bear operating system. This can be achieved by leveraging the proven practices of either adding functionality to the hypervisor as seen in Guest View Casting or reducing the hypervisor foot print as accomplished by NoHype and Bear. Once this performance requirement is met, further capabilities can be added to resilient systems, which allow for the creation of a new cloud security architecture.

IV. RELATED FIELDS OF WORK

One field of study that has not been included in this sur- vey is the idea of trust [68] in regards to the unauthorized access of data. One approach to handle trust in data secur- ity is that of security labels in the cloud [69]. The goal of this approach is to isolate customer virtual machines from each other to prevent data leakage across virtual machi- nes. This work is an enhancement of a trusted hypervisor that extends trust to network storage [70]. In regards to privacy, customers are concerned that their personal information will be leaked to those who should not have access to it. A current solution to this problem is the use of encryption with access control [71]. Using public key cryptography in the cloud, the user can be sure that their data is safe and only they have access to it.

V. CONCLUSION

All of the techniques reviewed in this paper have produced gains in making cloud computing more secure. Most of the solutions strive to race to the bottom of the software stack to combat known risks, rather than un- known zero-day risks. Moreover, it is currently left up to the cloud provider to pick from a grab bag of techniques to secure their infrastructure. This has led to a diverse set of approaches in cloud security, each with its own goals. The most successful approaches could be combined to build new cloud infrastructure. A starting point would be to begin with the idea of resilience as discussed in this paper. Non-determinism could then be added through process specific virtual machines. Multiple copies of these machines could refresh some processes in a non-deter- ministic manner. Lastly, secure migrations of processes and whole virtual machines can be added. Combining all these techniques could provide a cloud computing environment that drastically increases attacker workload.

REFERENCES

1. Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A (2003) "Xen and the Art of Virtualization. In: Proceedings of the nineteenth ACM symposium on Operating systems principles, pp 164–177
2. Goldberg RP (1974) A survey of virtual machine research. In: Proceedings of Computer, 7th edn., pp 34–45
3. Paleari R, Martignoni L, Roglia GF, Bruschi D (2009) A fistful of red-pills: how to automatically generate procedures to detect CPU emulators. In: Proceed- ings of the 3rd USENIX conference on Offensive technologies
4. Ferrie P (2006) "Attacks on Virtual Machine Emulators". Symantec Advanced Threat Research
5. Fitzgibbon N, Wood M, Conficker C (2009) A Technical Analysis. SophosLabs, Sophos Inc

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 8, Issue 2, February 2020

6. Quist D, Smith V (2006) "Detecting the Presence of Virtual Machines Using the Local Data Table". http://www.offensivecomputing.net/files/active/0/vm.pdf
7. Rutkowska J (2006) "Red Pill"., http://www.hackerzvoice.net/ouah/Red_% 20Pill.html
8. Ibrahim AS, Hamlyn-Harris J, Grundy J (2010) "Emerging Security Challenges of Cloud Virtual Infrastructure". In: Proceedings of APSEC Cloud Workshop
9. Corregedor M, Von Solms S (2011) "Implementing Rootkits to Address Operating System Vulnerabilities". Proceeding of Information Security South Africa, pp 1–8
10. Cabuk S, Dalton CI, Edwards A, Fischer A (2008) "A Comparative Study on Secure Network Virtualization". Technical Report HPL-2008-57, HP Laboratories
11. De Vivo M, De Vivo GO, Isern G (1998) "Internet Security Attacks at the Basic Levels", 32nd edn. ACM SIGOPS Operating Systems Review, pp 4–15
12. Chen PM, Noble BD (2001) "When virtual is better than real". Proceedings of the Eighth Workshop on Hot Topics in Operating Systems, pp 133–138
13. Bahram S, Jiang X, Zi W, Grace M, Li J, Srinivasan D, Rhee J, Xu D (2010) "DKSM: subverting virtual machine introspection for fun and profit". 29th IEEE International Symposium on Reliable Distributed Systems, pp 82–91
14. Hoglund G, Butler J (2005) Rootkits: subverting the windows Kernel. Addison-Wesley Professional, USA
15. Neal L (2009) Is Cloud Computing Really Ready for Prime Time? 42nd edn, of Computer 42:15–20
16. Pandey RK, Tiwari V (2011) "Reliability issues in open source software". In: Proceedings of the International Journal of Computer Applications, 34th edn., p 1
17. Klein G, Elphinstone K, Heiser G, Andronick J, Cock D, Derrin P, Elkaduwe D, Engelhardt K, Kolanski R, Norrish M, Sewell T, Tuch H, Winwood S (2009) "seL4: formal verication of an OS Kernel". In: Proceedings of 22nd ACM Symposium on Operating Systems Principles
18. Kennedy D, O'Gorman J, Kearns D, Aharoni M (2011) Metasploit: the penetration testers guide. No Starch Press
19. Davi L, Dmitrienko A, Sadeghi AR, Winandy M (2011) Privilage escalation attacks on android. Information Security, Springer
20. Eagle C (2011) The IDA Pro Book. No Starch Press, San francisco, USA
21. Eilam E (2005) Reversing. Wiley, New York, USA
22. Forresterm JE, Miller BP (2000) "An Empirical Study of the Robustness of Windows NT Applications Using Random Testing". 4th USENIX Windows Systems Symposium, Seattle, Appears (in German translation) as "EmpirischeStudiezurStabilität von NT-Anwendungen", iX, September 2000
23. Checkoway S, Halderman JA, Feldman AJ, Felten EW, Kantor B, Shacham H (2009) "Can DREs provide long-lasting security? The case of return-oriented programming and the AVC advantage". In: Proceedings of the USENIX/AC- CURATE/IAVoSS Electronic Voting Technology Workshop., August 2009
24. Denning D (1987) "An intrusion-detection model". Proc IEEE Trans SoftwEng SE-13(2):222–232
25. Bakshi A, Yogesh B (2010) "Securing cloud from DDOS Attacks using Intrusion Detection System in Virtual Machine". Second International Conference on Communication Software and Networks, pp 26–264
26. Lippmann R, Haines JW, Fried DJ, Korba J, Das K (2000) "The 1999 DARPA Off-Line Intrusion Detection Evaluation". In: Proceedings The International Journal of Computer and Telecommunications Networking - Special issue on recent advances in intrusion detection systems, vol 4, 34th edn., pp 579–595
27. Garfinkel T, Rosenblum M (2003) "A Virtual Machine Introspection Based Architecture for Intrusion Detection". In: Proceedings of Network and Distributed Systems Security Symposium
28. Kim GH, Spafford EH (1994) "The design and implementation of tripwire: a file system integrity checker". In: Proceedings of the 2nd ACM Conference on Computer and communications security., pp 18–29
29. Hofmeyr SA, Forrest S, Somayaji A (1998) Intrusion detection using sequences of system calls. J ComputSecur 6:151–180
30. Srivastava A, Giffin J (2008) "Tamper-Resistant, Application-Aware Blocking of Malicious Network Connections". In: Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection., pp 39–58
31. Pfoh J, Schneider C, Eckert C (2009) "A Formal Model for Virtual Machine Introspection". In: Proceedings of the 1st ACM workshop on Virtual machine security., pp 1–10
32. Loscocco PA, Wilson PW, Pendergrass JA, McDonell CD (2007) "Linux Kernel Integrity Measurement Using Contextual Inspection". In: Proceedings of the ACM workshop on Scalable trusted computing., pp 21–29
33. Zhu D, Chin E (2007) "Detection of vm-aware malware"
34. Egele M, Kruegel C, Kirda E, Yin H, Song D (2007) "Dynamic Spyware Analysis". In: Proceedings USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference, 18th edn
35. You I, Yim K (2010) "Malware obfuscation techniques: a brief survey". In: Proceedings of International Conference on Broadband, Wireless Computing, Communication and Application
36. Livshits B (2012) "Dynamic taint tracking in managed runtimes". Microsoft Research Technical Report, MSR-TR-2012-114
37. Jiang X, Wang X, Xu D (2007) "Stealthy Malware Detection Through VMM- Based "Out-of-the-Box" Semantic View Reconstruction". In: Proceedings of the 14th ACM conference on Computer and communications security., pp 128–138
38. Klein T (2003) "Scooby Doo-VMware Fingerprint Suite", http://www.trapkit. de/research/vmm/scoopydoo/index.html
39. Giffin J (2010) "The Next Malware Battleground Recovery after Unknown Infection". In: Proceedings of IEEE Journal on Security and Privacy, pp 74–76
40. CERT Coordination Center (2001) "CERT/CC security improvement modules: analyze all available information to characterize an intrusion"
41. Dittrich D (2000) "Report on the Linux Honeypot Compromise", http://project.honeynet.org/challenge/results/dittrich/evidence.txt
42. Dunlap GW, King ST, Cinar S, Basrai MA, Chen PM (2002) "ReVirt: enabling intrusion analysis through virtual-machine logging and replay". In: Proceedings of the 5th symposium on Operating systems design and implementation., pp 211–224

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 8, Issue 2, February 2020

43. Buchacker K, Buchacker K, Sieh V, Sieh V, Alexander F, Universität Erlangen-nürnberg (2001) "Framework for testing the fault-tolerance of systems including OS and network aspects". In: Proceedings IEEE High-Assurance System Engineering Symposium
44. Rutkowska J, Tereshkin A (2008) "Bluepilling the Xen Hypervisor". Black Hat, USA
45. Sailer R, Valdez E, Jaeger T, Perez R, Van Doorn L, Griffin JL, Berger S, Sailer R, Valdez E, Jaeger T, Perez R, Doorn L, Linwood J, Berger GS (2005) "sHype: Secure Hypervisor Approach to Trusted Virtualized Systems". IBM Research Report RC23511
46. Loscocco P, Smalley S (2001) "Integrating Flexible Support for Security Policies into the Linux Operating System". In: Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference., pp 29–42
47. Jaeger T, Sailer R, Zhang X (2003) "Analyzing integrity protection in the SELinux example policy". In: Proceedings of the 12th conference on USENIX Security Symposium, 12th edn., p 59, 74
48. Vogl S (2010) "Secure hypervisors". In: Proceedings of 12th International Conference on Enterprise Information System
49. Cheng G, Jin H, Zou D, Ohoussou AK, Zhao F (2008) "A Prioritized Chinese Wall Model for Managing the Covert Information Flows in Virtual Machine Systems". In: Proceedings of The 9th International Conference for Young Computer Scientists., pp 1481–1487
50. Wang G, Li M, Weng C (2010) "Chinese Wall Isolation Mechanism and Its Implementation on VMM". In: Proceedings of Systems and Virtualization Management: standards and the cloud, 71st edn.,pp 13–18
51. Sailer R, Jaeger T, Valdez E, Cáceres R, Perez R, Berger S, Griffin JL, Van Doorn L (2005) "Building a MAC-Based Security Architecture for the Xen Open-Source Hypervisor". In: Proceedings of Computer Security Applications Conference, 21st Annual
52. Szefer J, Keller E, Lee RB, Rexford J (2011) "Eliminating the Hypervisor Attack Surface for a More Secure Cloud". In: Proceedings of the 18th ACM conference on Computer and communications security., pp 401–412
53. Murray D, Milos G, Hand S (2008) "Improving Xen Security through Disaggregation". In: Proceedings of the Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments., pp 151–160
54. Dong Y, Yu Z, Rose G (2008) "SR-IOV networking in Xen: architecture, design and implementation". In: Proceedings of the First conference on I/O virtualization
55. Clark C, Fraser K, Hand S, Hansen JG, July E, Limpach C, Pratt I, Warfield A (2005) "Live Migration of Virtual Machines". In: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, 2nd edn., pp 273–286
56. Gouda MG, Liu AX (2005) "A Model of Stateful Firewalls and its Properties". In: Proceedings of the IEEE International Conference on Dependable Systems and Networks
57. Kruegel C, Valeur F, Vigna G, Kemmerer R (2002) "Stateful Intrusion Detection for High-Speed Networks". In: Proceedings of the 2002 IEEE Symposium on Security and Privacy
58. Xianqin C, Han W, Sumei W, Xiang L (2009) "Seamless Virtual Machine Live Migration on Network Security Enhanced Hypervisor". In: Proceedings of Broadband Network & Multimedia Technology., pp 847–853
59. Laprie J, T LAAS-CNRS (2005) "Resilience for the scalability of dependability". In: Proceedings of ISNCA., pp 5–6
60. Lombardi F, Di Pietro R, Soriente C (2010) "CReW: cloud resilience forwindows guests through monitored virtualization". In: Proceedings of the 29th IEEE Symposium on Reliable Distributed Systems., pp 338–342
61. Lombardi F, Di Pietro R (2009) "Kvmsec: a security extension for linux kernel virtual machines". In: Proceedings of the ACM symposium on Applied Computing., pp 2029–2034
62. Lombardi F, Di Pietro R (2011) "Secure virtualization for cloud computing". In: Proceedings of the Journal of Network and Computer Applications., pp 1113–1122
63. Russel R (2007) "lguest: implementing the little Linux hypervisor". In: Proceedings of the Linux Symposium, 2nd edn., pp 173–178
64. Reiser HP, Kapitza R (2007) "Hypervisor-Based Efficient Proactive Recovery". In: Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems., pp 83–92
65. Reiser HP, Kapitza R (2007) "VM-FIT: supporting intrusion tolerance with virtualisation technology". In: Proceedings of the 1st Workshop on Recent Advances on Intrusion-Tolerant Systems., pp 18–22
66. Taylor S, Henson M, Kanter M, Kuhn S, McGill K, Nichols C (2011) "Bear–a resilient operating system for scalable multi-processors". Submitted for publication in IEEE Security and Privacy, Nov/Dec 2011
67. Tanenbaum A, Woodhull A (2006) "Operating systems: design and implementation. Prentice Hall, Upper Saddle River, USA 68. Lehtinen R, Russell D, Gangemi GTSr (2006) "Computer security basics", 2nd edn. O'Reilly Media, Sebastopol, USA
69. Berger S, Cáceres R, Goldman K, Pendarakis D, Perez R, Rao JR, Rom E, Sailer R, Schildhauer W, Srinivasan D, Tal S, Valdez E (2009) "Security for the cloud infrastructure: trusted virtual data center implementation". In: Proceedings of the IBM Journal of Research and Development, 53rd edn.,pp 560–571
70. Berger S, Cáceres R, Pendarakis D, Sailer R, Valdez E (2008) "TVDc: managing security in the trusted virtual datacenter". In: Proceedings of ACM SIGOPS Operating Systems Review, 42nd edn., pp 40–47
71. Yu S, Wang C, Ren K, Lou W (2010) "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing". In: Proceedings of the 29th conference on Information communications., pp 534–542