VIRTUAL MACHINE INTROSPECTION FOR DETECTION AND HANDLING OF ROOTKIT ATTACK

[1]Ms. Pallavi Baviskar, [2] Gauri Kasat, [3]Vijay Pardeshi, [4]Vishesh Neve
[1]Asst. Professor, [2]Student, [3]Student, [4]Student
[1,2,3,4]Dept. Computer Engineering,
[1,2,3,4]Modern College of Engineering, Pune, India

Abstract: Virtual machine introspection (VMI) is a mechanism that allows indirect inspection and management of the status of virtual machines. The indirection of this approach offers attractive isolation properties that have resulted in a variety of VMI-based applications dealing with security, performance, and debugging in virtual machine environments. As it gives the access to the virtual machine monitor, VMI functionality is unfortunately not available to cloud users on public cloud platforms.

In this paper, we present our work on the CloudVMI architecture to address this concern. CloudVMI virtualizes the VMI interface and makes it available as-a-service in a cloud environment. Because it allows introspection of users' VMs running on arbitrary physical machines in a cloud environment, our VMI-as-a-service abstraction allows a new class of cloud-centric VMI applications to be developed.

In this paper we are proposing a rootkit recognition scheme that takes cloud computing environments into contemplation. The method utilizes vIPS platform to gain many beneficial traits including hypervisor-independency, agentless virtual security appliance structure, and usability. We are monitoring VM statically and dynamically with the help of SHA-512 and Feed forward neural network respectively. The runtime status of the VM will be monitored with the help of QEMU hypervisor. Therefore the method provides effective protection against rootkits in cloud computing environments.

***Index Terms* - Rootkit Attack, CloudVMI, Virtual Machine Introspection (VMI), QEMU, Feed Forward Neural Network algorithm, SHA512.**

I. Introduction

Virtual machine introspection is a technique used to inspect and analyze the code running on a given virtual machine. Virtual machine introspection has put on significant attention in the computer security research section. Virtual Machines are a prime target for an adversary to take control by exploiting the identified vulnerability present in it. Due to increasing number of Advanced Persistent Attacks such as malware, rootkit, spyware etc., virtual machine protection is a highly challenging task. The key element of Advanced Persistent Threat is a rootkit that provides stealthy control of underlining Operating System (kernel). Protecting individual guest operating system by using antivirus and commercial security defense mechanism is cost-effective and ineffective for a virtualized environment.
Virtual Machine Introspection has emerged as one of the promising approaches to secure the state of the virtual machine. Virtual Machine Introspection inspects the state of multiple virtual machines by operating outside the virtual machine i.e. at the hypervisor level. Virtual Machine Introspection is a fine-grained approach. Isolation property of VMI allows inspecting Memory, Disk, CPU registers, Network connections and other related hardware events of the virtual machine during run state from hypervisor level. Securing virtual machine using VMI is an active research area.

Related Work:

In recent years, it has been applied in various areas, ranging from intrusion detection and malware analysis to complete cloud monitoring platform. The present virtual machine introspection tools are necessary to deal with a variety of probable research gaps and to focus on key features required for wide application of virtual machine introspection techniques. In this paper, we focus on the evolution of virtual machine introspection tools and their ability to address the semantic gap problem. VMI demonstrate the method of observing and analyzing the status of a virtual machine from the hypervisor level. This lends itself well to security applications, though the hardware virtualization support from Intel and AMD was not designed with VMI in mind. This results in many challenges for developers of hardware-supported VMI systems.

Name Of Paper	Author	Published In	Year	Remarks
Securing KVM-based Cloud Systems via Virtualization Introspection.	Sheng-Wei Lee,Fang Yu.	IEEE	2014	Controlling and monitoring attack.
Virtual Machine Introspection: Observation or interference?	Nance,Kara, Matt Bishop and Brian Hay.	IEEE	2008	VM Behavior.
An Intelligent Virtual Machine Monitoring System Using KVM for Reliable And Secure Environment in Cloud.	Mrs. Swarupa Mahesh Deshpande, Prof. Mrs. Bharati Ainapure.	IEEE	2016	Working Of VMI System.

Table 1:- Literature Survey

Paper [1] Lets users have "one machine, multiple operating systems, multiple applications" and switch between them at will. This not only lets developers easily test their programs on multiple OSs and enterprise users more effectively utilize hardware through server consolidation, it's also useful to computer users in general. When virtual machines are distributed with a set of preconfigured applications, users can easily utilize complex applications. Further, the isolation offered by VMs provides some security benefit, such as allowing general Web browsing while reducing the risk of compromise to the underlying physical system.
Although virtualization isn't new, the recent development of x86 virtualization products has revived interest in the virtualization market. This has led to the evolution of virtual machine introspection (VMI) techniques and tools to monitor VM behavior. VMI tools inspect a VM from the outside to assess what's happening on the inside. 1 This makes it possible for security tools—such as virus scanners and intrusion detection systems—to observe and respond to VM events from a "safe" location outside the monitored machine. Here, we survey and categorize the current crop of VMI technologies, then offer a detailed description of the Virtual Introspection for Xen (VIX) tool suite, which addresses key VMI requirements.
Paper [2] gives VMI framework, Nitro, for hardware-based system call tracing and monitoring.Nitro is the first VMI-based system that supports all three system call mechanisms provided by the Intel x86 architecture and has been proven to work for Windows, Linux, 32-bit, and 64-bit guests. This framework is flexible enough to feasibly support almost any OS built upon the x86 architecture.
Virtual machine introspection (VMI) describes [3] malware detection that makes use of a support vector machine (SVM) to classify system call traces. It uses system call traces for malware detection, a string kernel to make better use of the sequential information inherent in a system call trace. By classifying system call traces in small sections and keeping a moving average over the probability estimates produced by the SVM, This is capable of detecting malicious behavior online and achieves a very high accuracy.
By abstracting the original machines Cloud computing is able to contribute the resources between several mutually distrusting clients. While there are numerous practical benefits to this system, this kind of resource sharing enables new forms of information leakage such as hardware side-channels.

II. Proposed system

Initially user has to set up centos on the machine. The First step of work is the installation of the hypervisor. We are installing QEMU on our Linux environment. The figure shows the architecture of the rootkit detection system. Following is the command for hypervisor installation :

```
[sudo apt-get install qemu-KVM libvirt-bin bridge-utils virt-manager]
```

Once we are done with the hypervisor installation then our next step is installing virtual machines on the hypervisor. We are using QEMU-KVM hypervisor. We may install VM using Virt Manager GUI. The Basics steps are as follows:

- Setting up VM configuration.
- Sharing an NFS Directory.
- Exporting the Directory
- Mounting the NFS Directory.

Rootkit Attack:

A rootkit is a collection of programs that allow administrator-level access to a computer or computer network. Normally, a cracker installs a rootkit on a computer after first getting the user-level access, either by exploiting a known vulnerability or cracking a password. Once the rootkit is installed, it permits the attacker to mask interruption and increase root or privileged access to the computer and, possibly, other machines on the network:

1) Application Level
2) Kernel Level

Monitoring Rootkit Attacks:

Application Level: First generate HashCode of Each File using SHA512 Algorithm. Write all hashCode in one file and name it as HashCode. Then send that HashCode file to Physical Machine.
CentOS machine receives HashCode file from VM. CentOS machine sends Iterative to VM each 30sec for a file. Machine compares hashcode file with original and gives result.

Kernel Level: First extract features of VM like BootLoader, process Ids, rc0.d, Runit, Systemd, gcc, gpg, keytool, apache build. Send this information to Physical Machine Server. CentOS validate this information using FeedForward Neural Network algorithm and gives a result.

III. PROPOSED SYSTEM

Algorithm Used:

1. Feed Forward Neural Network Algorithm

A multilayer perceptron (MLP) is a feed-forward artificial neural network representation that maps sets of input data onto a set of appropriate outputs. An MLP consists of numerous layers of nodes intended for a graph, with every layer is entirely linked to the next one. Apart from the input nodes, each node is a processing element with a nonlinear activation function.

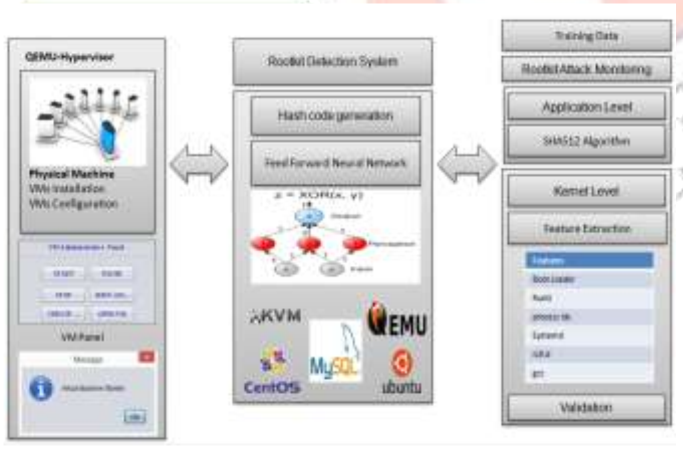

Fig.1-architechture diagram of proposed system

MLP utilizes a supervised learning technique called backpropagation for training the network. MLP is an alteration of the normal linear perceptron and can differentiate data which is not linearly distinguishable.

a. Activation Function:

- If a multilayer perceptron has a linear activation function in all neurons, that is, a linear function that maps the weighted inputs to the output of each neuron, then it is simply verified with linear algebra that any number of layers can be diminished to the typical two-layer input-output model.

$$y(v_i) = \tanh(v_i) \quad \text{and} \quad y(v_i) = (1 + e^{-v_i})^{-1},$$ [8]………..equation 1.

- We represent the error in output node j in the nth data point by,

$$e_j(n) = d_j(n) - y_j(n)$$ [8]………equation 2.

- We subsequently formulate the modification to the weights of the nodes based on those corrections which minimize the error in the entire output, given by

$$\mathcal{E}(n) = \frac{1}{2}\sum_j e_j^2(n)$$

[8]………..equation 3.

- Using gradient descent, we find our change in each weight to be

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial v_j(n)} y_i(n)$$

[8]…………equation 4.

Where $y_i$is the output of the previous neuron and $\eta$is the learning rate.

- The derivative to be calculated depends on the induced restricted field, v_j which itself shows a discrepancy. It is easy to prove that for an output node this derivative can be simplified to,

$$-\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} = e_j(n)\phi'(v_j(n))$$

[8]………..equation 5.

Where ϕ' is the derivative of the activation function described above.

2. **SHA512 Algorithm:**

SHA (Secure Hash Algorithm) is a set of cryptographic hash functions designed by the National Security Agency (NSA). Cryptographic hash functions are mathematical operations run on digital data; by comparing the computed "hash" (the output from execution of the algorithm) to a known and expected hash value, a person can determine the data's integrity. For example, calculate the hash of a downloaded file and evaluating the effect to a previously published hash result can show whether the download has been modified or tampered with. A key aspect of cryptographic hash functions is their collision resistance: nobody should be able to find two different input values that result in the same hash output.

SHA-2 comprises of major changes from its predecessor, SHA-1. The SHA-2 family includes six hash functions with hash values that are 224, 256, 384 or 512 bits: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256.

SHA-512 is novel hash functions computed with 32-bit and 64-bit words, respectively. They use different move amounts and preservative constants, but their structures are almost the same, differing only in the number of rounds.

IV. CONCLUSION

We propose an efficient Virtual Machine Introspection as a Cloud Service, Rootkit Detection System.An effective rootkit detection method for detecting rootkits inside VMs in cloud environments. It builds upon vIPS platform, an effective virtualized system security protection platform, to create a hypervisor independent, agentless virtualized security appliance. This system will be useful in public cloud for Security and safety. The system will monitor the runtime state of the VM and will detect that it is attacked or not.

REFERENCES

[1] Lee, Sheng-Wei and Fang Yu. 2014. Securing KVM-Based Cloud System via Virtualization Introspection. 47th Hawaii International Conference on IEEE.

[2] Kara, Nance, Matt Bishop and Brian Hay, 2008. Virtual Machine Introspection: Observation or interference? .IEEE Security & Privacy, 5: 32-37.

[3] Mrs. Swarupa Mahesh Deshpande, Prof. Mrs. Bharati Ainapure. 2016. An Intelligent Virtual Machine Monitoring System Using KVM for Reliable and Secure Environment in Cloud. IEEE International Conference on Advances in Electronics, Communication and Computer Technology (ICAECCT) Rajarshi Shahu College of Engineering, Pune India. Dec 2-3.

[4] Pfoh, Jonas, Christian Schneider and Claudia Eckert. 2011. Nitro: Hardware-based system call tracing for virtual machine. Advances in Information and Computer Security. Springer Berlin Heidelberg: 96-112.

[5] Thu Yein Win, Huaglory Tianfield and Quentin Mair. 2015. Detection of Malware and Kernel-level Rootkits in Cloud Computing Environments. IEEE 2nd International Conference on Cyber Security and Cloud Computing.

[6] Chuliang Weng, Qian Liu, Kenli Li, and Deqing Zou. 2016. CloudMon: Monitoring Virtual Machines in Clouds. IEEE Transaction.
[7] Borisaniya, Bhavesh and Dhiren Patil. 2014. Evasion Resistant Intrusion Detection Framework at Hypervisor Layer in Cloud. International Conference on IEEE.
[8] https://en.wikipedia.org/wiki/Feedforward_neural_network.