VMPmonitor：An Efficient Modularity Approach for Hidden Process Detection

Chaoyuan Cui
Institute of Intelligent Machines
Chinese Academy of Sciences
Hefei, China
cycui@iim.ac.cn

Yun Wu
Anhui Xunhuan Jingji Jishu GongCheng Yuan
Chinese Academy of Sciences
Hefei, China
wuyun@iim.ac.cn

Ping Li
Department of Automation
University of Science and Technology of China
Hefei, China
liping89@mail.ustc.edu.cn

Rujing Wang
Institute of Intelligent Machines
Chinese Academy of Sciences
Hefei, China
rjwang@iim.ac.cn

***Abstract*—With the development of the Cloud computing, more and more people are accustomed to resource sharing or online shopping. And malware has become a major threat to the Cloud safety. Process hiding is a powerful technique commonly used by stealthy malware to evade detection by anti-malware. In this paper, we present a novel approach called VMPmonitor-an efficient modularity approach for hidden process detection. With the help of the guest OS register information (mainly the ESP) collected by virtual machine monitor, VMPmonitor can implicitly capture the hidden process information of target guest OS. Compared to other approaches, VMPmonitor obtains guest process information implicitly. Using implicit information reduces its susceptibility to guest evasion attack. Experimental result shows that VMPmonitor has better reliability and accuracy.**

Keywords-security; malware; hidden process detection; virtual machine monitor

I. INTRODUCTION

With the rapid development of the Internet technology, computer network makes people's life more and more convenient. People can shop or share resource through the network, so computer network security has become the focus of attention. The data of Kingsoft cloud security center shows: fishing website grows quickly in 2011, and outbreak later this year. The new fishing website number is 450000. The new fishing website of December is more than two times that of January. Trojans, hacking software and backdoor occupy most part of the viruses. Malware such as Rootkit uses various techniques to make system tools can't detect its existence. Process hiding is a powerful technique commonly used by stealthy malware to evade detection by anti-malware. It has posed a great threat to the safety of the computer system and network. Therefore we propose VMPmonitor, a Xen-based front-end and back-end model to detect hidden process and protect the security of the system. VMPmonitor is based on Xen Hypervisor.

The rest of this paper is as follows: Section II discusses related research in the area of detecting process. Section III presents the design of VMPmonitor, followed by the implementation details in Section IV. We then present evaluation results in Section V. Finally, Section VI concludes this paper.

II. RELATED WORK

Classical virtual machine security has been widely studied. In order to enhance system security, many researchers have adopted VMs to detect intrusions [1, 2] and diagnose system problems [3, 4]. Virtual machine monitor (VMM) has become an important platform for building honeypots.

From the point of the implementation technology of secure monitor, the related works can be classified into internal monitoring and external monitoring. Lares [5] and SIM [6] are the typical system of internal monitoring. The internal monitoring needs to insert kernel module to the guest operating system, and it doesn't have transparency. The kernel protection module and jump module is closely related to the virtual machine, so the internal monitoring don't have versatility. External monitoring has an advantage over internal monitoring. Livewire [7] is a typical system of external monitoring, which does not affect the behavior of target virtual machine and is reliability. Livewire has taken an extremely conservative approach to introspection by primarily engaging in passive checks that incur no visible impact on system performance. However, the cost of this was that monitoring frequent asynchronous events, e.g. all system calls, may be quite performance intensive.

In order to evade detection by anti-malware, an attacker often attempt to hide their malicious processes by modifying some aspect of the system using Rootkit. Hidden process is the most common security threat, so the system administrator must face this problem. VMwatcher [8] and Lycosid [9] can detect hidden process. Lycosid detects the processes by using cross-view validation techniques and least squares regression

analysis. However, Lycosid recognizes the processes with probability which will lead to false positives or false negatives. Compared to Lycosid, VMPmonitor does not use cross-view and do little data processing which make VMPmonitor relatively lightweight. VMPmonitor is built with the merits of both high resistance and accuracy.

III. DESIGN

A. Background

1) Xen and virtualization

Xen [10] is a hypervisor providing services that allow multiple computer operating systems to execute on the same computer hardware concurrently. Xen supports para-virtualization and hardware-assisted virtualization to run the guest operating system. In computing, a hypervisor or virtual machine monitor is a piece of computer software, firmware or hardware that creates and runs virtual machine. A computer on which a hypervisor is running one or more virtual machines is defined as a host machine. Each virtual machine is called a guest machine.

Shadow page table (SPT) is often used in simulating more than one operating system on a single set of memory and processor. An operating system uses the SPT to map the virtual memory to its location on the physical memory. The introduction of the SPT makes memory virtualization is completely transparent for the guest OS.

Since the introduction of the guest physical address, memory virtualization needs two address translation to support virtual address. The guest software determines the translation from guest virtual address (GVA) to guest physical address (GPA). The VMM determines the translation from guest physical address (GPA) to host physical address (HPA). The SPT can eliminate additional address mapping and achieve the translation from GVA to HMA. It improves the access efficiency.

2) Linux kernel

A process is an instance of a computer program that is being executed. Specifically, every process in Linux is represented by a process control block (define as task_struct). The task_struct data structure contains almost all of the information (e.g. state, stack, prio, tasks, mm, and children etc.). This structure is defined in linux-source-3.2.0/include/linux/sched.h. The Linux kernel allocates a task_struct for each process through the slab allocator. When the slab allocator dynamically generates and stores the task_struct, the kernel creates a new data structure thread_info at the bottom of the kernel stack or the top of the kernel stack.

The first member of the thread_info data structure is task. It stores a pointer, which points to the task_struct data structure. And stack is the second member of the task_struct data structure. It is a pointer, which points to the thread_info data structure. Figure 1 shows the relation between the task_struct data structure and the thread_info data structure.

All running processes are linked by a doubly linked list. The linked list implements through the tasks member of the task_struct. Every tasks has a next pointer, which points to the next tasks of the task_struct data structure in the linked

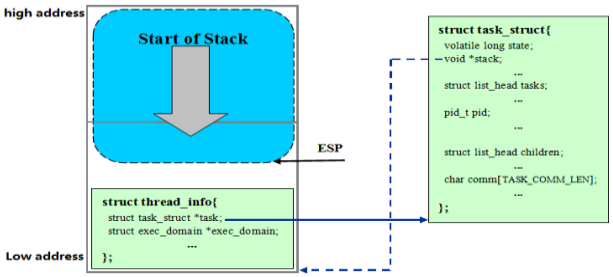

Figure 1. The relation between the task_struct data structure and the thread_info data stracture.

list. And every tasks has a prev pointer, which points to the previous tasks of the task_struct data structure in the linked list. We can traverse the entire process list through any one process. Figure 2 shows the connection between these processes through the linked list.

B. System architecture

Virtual machine manager is a lightweight software layer in Xen system. It is also called Xen Hypervisor. And the virtual machine is called a Domain. Xen Hypervisor lies between operating systems and hardware. It provides the virtual hardware environment for operating system. Xen uses hardware-assisted virtualization. Figure 3 shows a typical Xen model. The virtual machine manager can use I/O device drivers of the exiting operating system and directly control physical resource in this model. Thus the efficiency of virtualization is improved than before.

Privileged Domain (Dom0) is a virtual domain running on Xen. Dom0 has a native device driver and can direct access to hardware devices. It can control and manage other Domain through the interface provided by Xen. The guest virtual machine is Unprivileged Domain (DomU). DomU can migrate between different machines.

In our approach, VMPmonitor is consist of front-end module and back-end module. The former is located in Dom0 and is used to analysis physical memroy, while the latter is in DomU and is a function library that contains detail information of process.

IV. IMPLEMENTATION

We have implemented VMPmonitor on Xen 4.1.2 and Linux 3.2.16. The VMPmonitor technology centers on locating the task_struct data structure of the guest machine. We can get useful information from this structure.

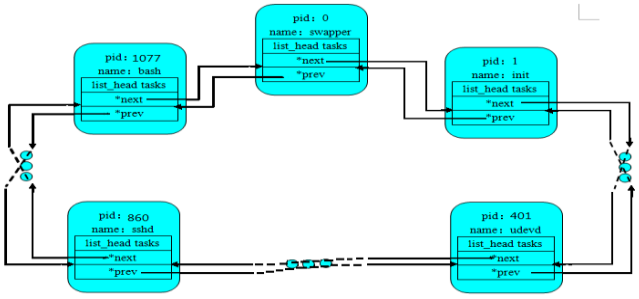

Figure 2. The connection between these processes through the linked list.

In theory, if we want to read the members in task_struct data structure, the related head files need to be included in the VMPmonitor source code. Therefore we design the front-end and back-end model. And this model is easy to extend to different virtual machine.

We design a modularized method to implement this function. VMPmonitor is divided into two modules: front-end module and back-end module. The function of the front-end module is locating the HMA of the task_struct data structure by reading ESP locater and request the process information from back-end. The back-end is a function library, which includes a series of process information interface functions. Two modules are separately compiled into object files in Xen source code and Linux source code. Then we put them together to compile and produce the final executable file.

A. *Front-end module*

1) *task_struct data structure*

The task_struct is the process descriptor of the Linux kernel. Every process has a task_struct data structure, which includes all information about the process.

2) *HMA of guest machine*

The VMPmonitor technology uses the ESP register to indicate the current stack's location. When we start using the stack which is the same size as the THREAD_SIZE, the pseudo-code is as follows:

```
1 struct task_struct* current_task_address()
2 {
3    struct task_struct *p;
4    p=~(THREAD_SIZE-1)&esp;
5    p=map_page(ctx,vcpu,p->task);
6    return p;
7 }
```

The 4th line locates the thread_info data structure of the current process. What must do next is calling map_page() to translate the GVA to HMA, and finally the return value is just a pointer which points to task_struct with HMA.

B. *Back-end module*

After locating the HMA of the tasks_truct data structure, we can obtain the process information, such as pid, comm, stack, and children etc. For example, we can acquire process identification by the operation of reading p->pid as follows:

```
1 int returnpid(guest_word_t *addr)
2 {
3   struct task_struct *p = (struct task_struct *) addr;
4   return p->pid;
5 }
```

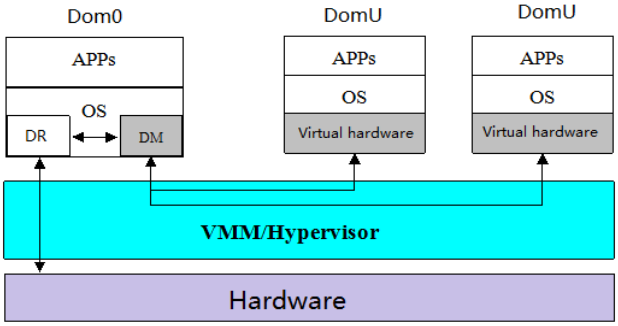

Figure 3. A typical Xen model.

V. EVALUTATION

A. *Experimental environment*

We test the performance of VMPmonitor on Xen 4.1.2 and Linux 3.2.16. The specific configuration of experimental environment is shown in table 1.

B. *Experimental content*

.We have implemented VMPmonitor for Centos 6.4 (64 bits) with Linux 2.6.32 (Dom1). VMPmonitor runs in the kernel mode of Dom0, thus does not affect the normal execution of Dom1. And this paper also can detect process in web page through JSP technology. Therefore we can detect the hidden process away from the real machine.

1) *Detect process*

At first, we use VMPmonitor to detect the process and display the process list in web page outside of Dom1. Domain identification, virtual CPU and OS type are the arguments requested by the executable file. Then we use "ps -e" command to obtain the process list in Dom1. In order to verify the accuracy of the result, we compare two results. Figure 4 shows the result of detecting process in two methods. The result demonstrates all processes in Dom1 can be displayed by VMPmonitor.

2) *Detect new process*

Then, we use VMPmonitor to detect the real-time process and display the process list in web page outside of Dom1. We start a new command (e.g. ping www.baidu.com) in Dom1. At the same time we start VMPmonitor program to detect this process. Figure 5 shows the results and we find the ping process in the process list. In order to guarantee the stability of this approach, we use vim command to try again. And VMPmonitor can detect the vim process quickly.

3) *Detect hidden process*

At last, we use VMPmonitor to detect the hidden process. We use adore-ng as a sample, which is an advanced Linux kernel rootkit that can hide files and processes. In our test, adore-ng.ko is firstly loaded into Dom1. Then we execute "./ava i 1202" command to hide sshd process with PID 1202.

And we start VMPmonitor to detect this hidden process. Figure 6 shows the results and we find the sshd process in the list. The outputs from the command ps is manipulated to conceal the existence of any process with PID 1202. Thus we can't find ssh process in the Dom1 through "ps -e" command. We have made several experiments by hiding different processes. But VMPmonitor can detect the hidden process accurately. Therefore, the result shows the effectiveness of this approach through the comparison between internal and external process list.

TABLE I. CONFIGURATION OF EXPERIMENTAL ENVIRONMENT

	Xen Hypervisor	Guest OS
OS	Ubuntu12.04(desktop, 64bits)	Centos 6.4 (server, 64bits)
Linux kernel	3.2.16	2.6.32-358.14.1.e16.x86_64
Xen	Xen_4.1.2	
others	CPU: 4; RAM: 4G	CPU: 1; RAM: 1G

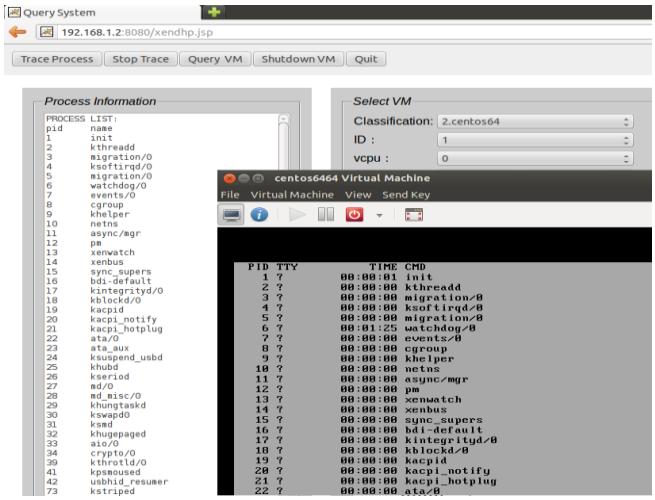

Figure 4. Result of detecting process.

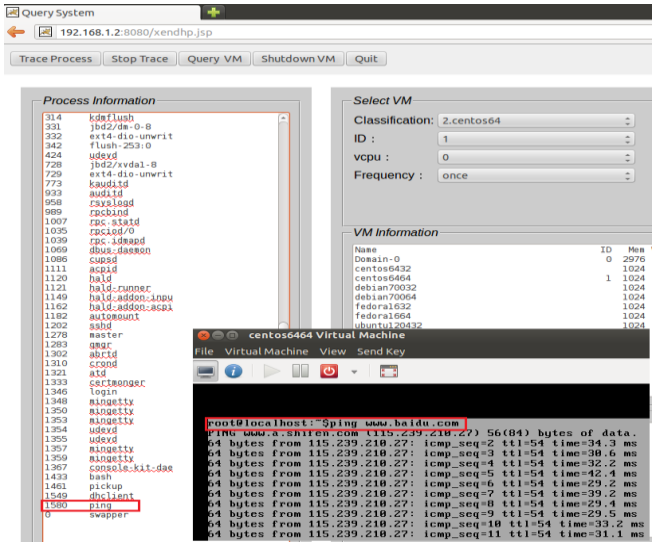

Figure 5. Result of detecting new process.

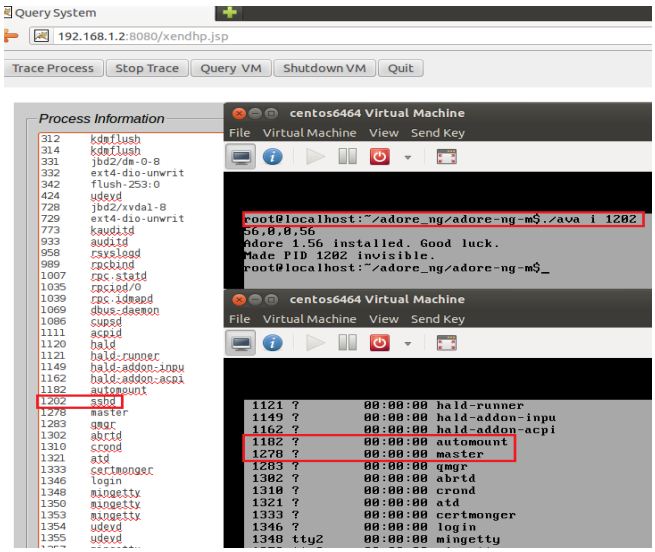

Figure 6. Result of detecting hidden process.

VI. Conclusion

In this paper, we present a novel approach called VMPmonitor to detect hidden process accurately. VMPmonitor can obtain the real-time and hidden process list outside of the virtual machine. The experiments prove that this approach is accurate and effective. And the modular mechanism has strong portability and reusability.

VMPmonitor can only achieve the basic hidden process detection, so we want to implement hidden file detection, hidden port detection and hidden registry detection in the future. Finally, we must combine all kinds of detection information to judge whether there is any malware in the system.

References

[1] A. Joshi, S. T. King, G. W. Dunlap, and P. M. Chen. Detecting Past and Present Intrusions through Vulnerability-Specific Predicates. Proc. the 20th ACM Symposium on Operating System Principles, Oct. 2005, pp. 91-104.

[2] K. Kourai and S. Chiba. HyperSpector: Virtual Distributed Monitoring Environments for Secure Intrusion Detection. Proc. the 1st ACM/USENIX International Conference on Virtual Execution Environments, June 2005, pp. 197-207.

[3] S. T. King, G. W. Dunlap, and P. M. Chen. Debugging operating systems with time-traveling virtual machine. Proc. the 2005 USENIX Annual Technical Conference, Apr. 2005.

[4] A. Whitaker, R. S. Cox, and S. D. Gribble. Configuration Debugging as Search: Finding the Needle in the Haystack. Proc. the 6th Symposium on Operating Systems Design and Implementation, NOV. 2004.

[5] B. D. Payne, M. Carbone, M. Sharif, W. Lee. Lares: A Architecture for Secure Active Monitoring Using Virtualization. Proc. IEEE Symposium on Security and Privacy, 2008, pp. 233-247.

[6] M. I. Sharif, W. Lee, W. Cui, A. Lanzi. Secure in-VM Monitoring Using Hardware Virtualization. Proc. the 16th ACM Conference on Conputer and Communications Security, 2009, pp. 447-487.

[7] T. Garfinkel and M. Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. Proc. the 10th Annual Network and Distributed System Security Symposium, Feb. 2003.

[8] X. Jiang, X. Wang, D. Xu. Stealthy Malware Detection Through VMM-Based "Out-of-the-Box" Semantic View Reconstruction. Proc. the 14th ACM Conference on Computer and Communications Security , 2007, pp. 128-138.

[9] S. T. Jones, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau. VMM-based Hidden ProcessDetection and Identification using Lycosid. Proc. the 4th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, 2008, pp. 91-100.

[10] 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. Proc. the 19th ACM Symposium on Operating Systems Principles, Dec. 2003,pp.164-177.