

Asynchronous pseudo physical memory snapshot and forensics on paravirtualized VMM using split kernel module

Ruo Ando, Youki Kadobayashi and Youichi Shinoda

National Institute of Information and Communication Technology,
4-2-1 Nukui-Kitamachi, Koganei,
Tokyo 184-8795 Japan
ruo@nict.go.jp
http://www2.nict.go.jp/y/y212/index_en.html

Abstract. VMM (virtual machine monitor) based system provides the useful inspection and interposition of guest OS. With proper modification of guest OS, we can obtain event-driven memory snapshot for malicious code forensics. In this paper we propose an asynchronous memory snapshot and forensics using split kernel module. Our split kernel module works for virtualized interruption handling, which notifies malicious fault, illegal system call and file access. On frontend, we insert virtualized interruption into source code of MAC (mandatory access control) module, fault handler and gcc-extension. Then, Backend kernel module receives the asynchronous incident notification. In experiment, we take RAM snapshot of LKM-rootkit installation using system call extension. Frequently appeared n-grams is extracted by weighted resolution in order to find memory blocks which is used by LKM-rootkit. Proposed system can detect unknown malware (malicious software) of which name is not matched by signature. Also, it is showed that we can find evidence of malware from memory snapshot by linear extraction algorithm.

Keywords: Asynchronous snapshot, paravirtualized VMM, memory forensics, virtualized interruption, split kernel module.

1 Introduction

1.1 Virtual machine monitor

VMM (virtual machine monitor) is a thin layer of software between the physical hardware and the guest operating system. The rapid increase of CPU performance enables VMM to run several operating system as virtual machine, multiplexing CPU, memory and I/O devices in reasonable processing time. Recent VMM is a successful implementation of microkernels. Under the guest OS, VMM runs directly on the hardware of a machine which means that VMM can provides the useful inspection and interposition of guest OS.

1.2 HIDS, NIDS and VMM based IDS

IDS (Intrusion detection system) is classified into HIDS (Host-based IDS) and NIDS (Network-based IDS). Previously, there is a tradeoffs between NIDS and HIDS about visibility and attack residence. HIDS provides good visibility. However, HIDS has weaker isolation than NIDS. That is, once the operating system running HIDS is compromised, attacker can get down HIDS. On the other hand, NIDS offer higher attack residence instead of the cost of visibility. VMM-based IDS can provide good visibility while maintaining secure isolation for the IDS. VMM-based IDS takes advantages in three points, isolation, inspection and interposition. Besides, the utilities of VMM makes it possible to take snapshot of the pseudo physical (virtualized) memory of the guest OS.

1.3 Asynchronous memory snapshot

VMM has access to all the state of a virtual machine. VMM enables us to access all events of CPU, memory and I/O devices of the guest (virtualized) OS. In previous operating system, it is difficult to take snapshot of system itself. This is partly possible by using debuggers, but it is still difficult to capture all the state of system. It is important for us IDS designers that with proper modification of VMM and guest OS, automatic snapshot triggered by some incidents is possible. In this paper we propose an asynchronous pseudo physical memory snapshot and forensics on paravirtualized VMM using split kernel module. We modify VMM and guest OS by implementing split kernel module. Timely memory snapshot is useful for detecting and inspecting malicious software attacks on the guest OS.

Asynchronous memory snapshot enables us to extract evidence memory blocks by simple string analysis. By frequently appeared string analysis, we can know malware of which name is unknown and installer's behavior from our memory snapshot. Previous malware detector is basically signature-based, which means it cannot detect the malware of which name is not matched. Attackers can easily avoid black-list based detection by changing or obfuscating the signature of their malware.

Proposed method is frequency analysis based. We can discover and sort the parts of memory snapshot including frequently appeared strings. By doing this, we can detect which blocks was used by attacker (assigned for malware operation). Then, inspecting these blocks provides detailed information for forensics. Let us discuss the example:

```
72 65 5f 72 6f 6f 74 5f 66 69 6c 6c 64 69 72 20 re_root_filldir
6e 61 6d 65 20 2d 3e 20 74 65 73 74 c0 bc 5e 47 name -> test..^G
20 0a 4d 61 79 20 33 31 20 30 35 3a 31 39 3a 31 .May 31 05:19:1
32 20 6c 6f 63 61 6c 68 6f 73 74 20 6b 65 72 6e 2 localhost kern
65 6c 3a 20 61 64 6f 72 65 5f 72 6f 6f 74 5f 66 e1: adore_root_f
69 6c 6c 64 69 72 20 6e 61 6d 65 20 2d 3e 20 6f illdir name -> o
75 74 6f c7 bc 5e 47 20 0a 4d 61 79 20 33 31 20 uto..^G .May 31
```

name	grep hit	name	grep hit
kernel	4699	usr	1651
driver	4937	xen	7544
module	3476	adore	1780

Table 1. Frequently-appeared strings in memory snapshot. Snapshot is taken on XEN kernel 2.6.18. LKM-rootkit "adore" is installed in this kernel.

This is the part of memory dump (snapshot), which is used by malware installation. These strings means that function "filldir" is changed to adore_root_filldir on May 31. Adore is the name of malware which is appeared frequently in memory dump. Memory snapshot is taken asynchronously in time of malware installation. To extract such a information, memory snapshot needs to be obtained during the execution of some system calls on the guest OS. In previous signature-based system, if the word of "adore" is not in database, we cannot detect it. In proposed system, once the memory snapshot is taken timely, this part of memory dump is extracted automatically by simple n-gram analysis.

2 Proposed method

Proposed method is divided into two operations, architecture modification and memory inspection. First, we append split kernel module to paravirtualized VMM. Second, we insert probe code into the points where exploitation could be occurred. For example, fault handling, illegal file access could be probed. Also, important system call (such as create_module) need to be modified (see section 7.2). Third, we extract frequently appeared n-gram from RAM snapshot. In this process, we divided RAM snapshot into M blocks. A string of unknown malware could be found because proposed system has taken snapshot just after the malware was installed. Finally, we sort strings according to frequency and choose top N. We can find block which is used by malware as follows.

```

for i to N
  for j to M
    grep string[i] EVENT_DRIVEN_SNAPSHOT_BLOCKS[j]
  next
next

```

Then the evidence block introduced in section 1 is extracted automatically. Motif in Figure 1 means a string that appears in memory snapshot frequently. Table 1 shows example of motif on kernel memory. In Linux, kernel code is memory resident. So the string such as kernel, driver and module appears frequently. In this case of Table 1, snapshot is taken just when LKM-rootkit "adore" is installed. Adore appears 1780 times. Even if attacker installs unknown malware, the name of the malware appears as frequently as other frequent strings. In extraction,, We use weighted resolution (see section 7.4).

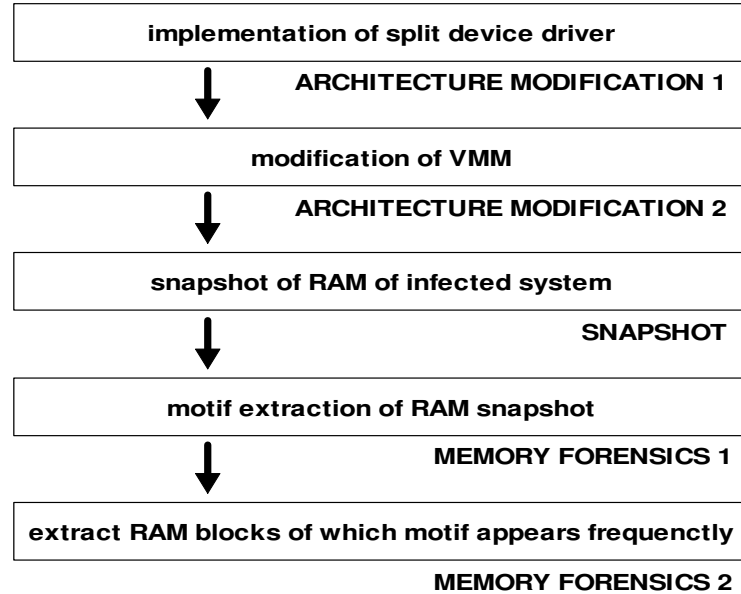


Fig. 1. Flow chart of proposed method. Proposed method is divided into architecture modification and memory forensics.

3 Related work

VMM is also called hypervisor. Hypervisor is old technology, originated from IBM mainframe in 1960s. However, VMM now offers huge potential to generic servers about consolidation and secure isolation [1]. Classical aspects of secure kernel for VMM is discussed in [2][3]. Recently, XEN [4][5] and KVM (kernel virtual machine) [6] is going to mainline of linux kernel.

To obtain system control, security module need to be at lower-level of system. Secure OS is a kernel module as mediator between system call execution and I/O access. With the deployment of VMM under operating system, some security applications of VMM has been proposed. The design and deployment of IDS on VMM is discussed in [7]. They propose VMI (virtual machine introspection) applied for IDS policy management. Some of the LKM-rootkits is tested to validate their system. In [10][11], moving functionality of secure linux module to outside guest VM is discussed. However, in these papers memory forensics to extract further information of malware is not discussed.

VMM provides new concepts for verification and debugging. These are called as deterministic replay or time-travel debugging which is also useful for security. Security application of logging and replay using VMM are discussed in [6][7]. However, their paper is not specified for the detection of kernel mode malware such as LKM-rootkit. Computer forensics is much concerned with rootkit detection and inspection. VMM based rootkit is discussed on [6][7]. They do not

mention how to detect the event of installation of VMR. The thrust of this paper is the specific modification of VMM architecture in order to enhance memory snapshot for malware forensics. Besides detection, proposed system enables us to extract further information of exploitation of kernel based malware. Once the split kernel module is implemented and registered correctly, modification of guest is simple. Also, evidence memory block is extracted by a simple string analysis.

4 Paravirtualized VMM of XEN

In this paper we implement the proposed system on virtual machine monitor XEN[1]. Virtualization technology is divided into two categories: full-virtualization and para-virtualization. Para-virtualization needs kernel-modification in order to run guest OS on ring 1. Full-virtualization uses new CPU mode such as Intel VT and AMD Pacifica, which makes it possible to run VMM on specified mode. Para-virtualization is more lightweight mainly about I/O performance. Another advantage of para-virtualization is that we can insert hypervisor call into source code of guest kernel. This enables us to construct new (specified for security) notification channel between hosted OS and VMM.

Figure 3 illustrates architecture of XEN. Domain U (guest OS) has "virtualized" frontend-driver, like proxy for I/O access of "real" device driver under Domain 0 (host OS). I/O request of process on guest OS is notified by split device driver (backend and frontend driver).

4.1 Event-channel of XEN

In VMM, there is asynchronous notification mechanism between guest OS and privileged OS. In XEN, guest domain access hardware through backend driver of domain 0 (privileged domain) using event-channel. Event channel is a virtualized interruption mechanism from guest domain (U) to host domain(0). In proposed system, we newly construct event channel for the notification of security event on guest OS.

4.2 Split kernel module

Figure 2 shows generic framework of split kernel module of virtual machine monitor. Guest OS and device driver of host OS share virtualized interruption and memory using this mechanism. In XEN, virtualized interruption is called event channel. Shared memory is called grant table. I/O request is triggered by event channel hypervisor call.

5 Asynchronous notification channel using split kernel module

In proposal method, memory snapshot need to be taken timely. Polling or sequential mechanism is not proper to cope with the security incidents. Particularly, on-line or asynchronous mechanism is necessary for inspecting malware on

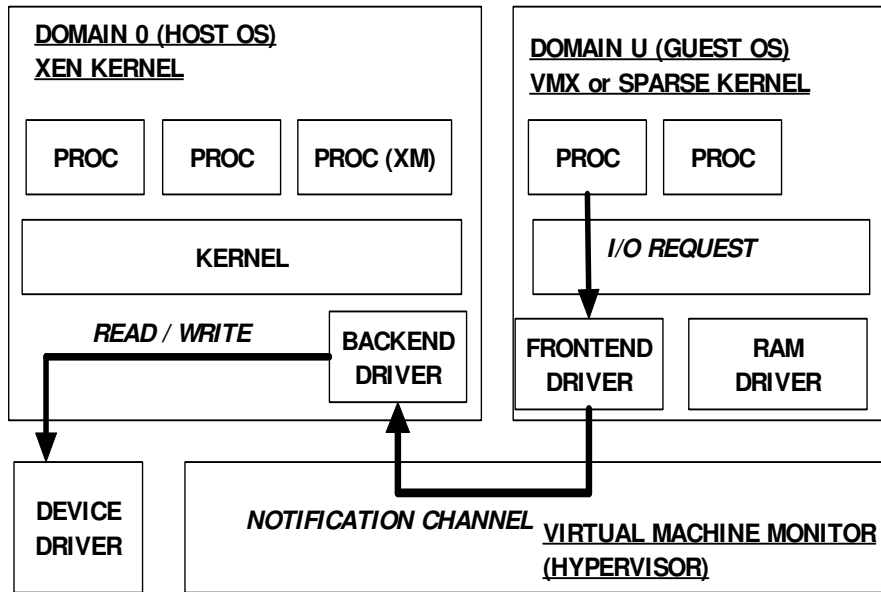


Fig. 2. Asynchronous notification. IRQ go through frontend and backend kernel module.

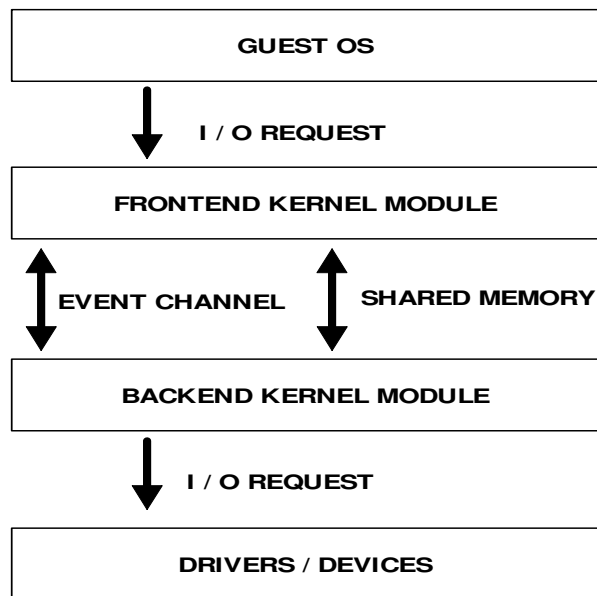


Fig. 3. Split kernel module consists of frontend and backend module. These two modules communicate by event-channel and shared memory. I/O request of guest OS go through split kernel module to "real" device driver on host OS.

memory. In proposal system we construct an asynchronous notification channel using split kernel module (device driver). When security incidents such as buffer overflow exploitation, malware installation and malicious file access has been occurred on the guest OS, it is notified to our frontend device driver. Then, the notification is transferred to backend device driver. Device driver is more suitable (than application) for coping with the asynchronous events.

In modified VMM, the detected incidents are equivalent of a hardware interrupt. When malicious behavior is detected, it is notified to our kernel module as IRQ (interrupt request). Let us show the list of the guest OS.

```
256: 1782 Dynamic-irq timer0
257: 0    Dynamic-irq resched0
258: 0    Dynamic-irq callfunc0
259: 227  Dynamic-irq xenbus
260: 187  Dynamic-irq xencons
261: 891  Dynamic-irq blkif
262: 0    Dynamic-irq blkif
263: 0    Dynamic-irq eth0
264: 0    Dynamic-irq sec-notify
```

This is the list of dynamic (virtualized) IRQs of the guest OS. We have obtained this list by "cat /proc/interrupt". 261 is interrupt handler of block device. 262 is for swap. 263 is for network interface. 264 sec-notify is our frontend driver of split kernel module. We append dynamic-irq sec-notify and by using this, the incident notification is transferred to VMM.

Once our driver is registered correctly, modification of guest OS is very simple. All we need to do is inserting this code into probes of guest OS.

```
int port=9;

    evtchn_op_t op;
    op.cmd      = EVTCHNOP_send,
    op.u.send.port = port;

(void)HYPERVISOR_event_channel_op(&op);
```

This code is activating event-channel and sending the interruption signal to our frontend-drivers in port 9. In this case, port number 9 is assigned to Dynamic-IRQ 264. By inserting this code, the incident notification is transferred to frontend driver, VMM, backend driver and finally the host OS.

The deployment of proposed system with split kernel module is completed in these steps.

- (1) Implementing front/backend driver (split kernel module)
- (2) Registering driver to Xenstore
- (3) System call extension: inserting hypervisor call into system call (create_module)

- (4) MAC extension: inserting hypervisor call into i-node permission checker
- (5) Buffer overflow handling: inserting hypervisor call into GCC-extension or fault handler

Xenstore in (2) is the utility of XEN, device database of the guest OS. By step (3)(4) and (5), secure OS and other protection system can communicate with our split device driver. We discuss the step (3) and (4) in the next section. We discuss step (5) in appendix section. These steps are completed only by inserting five lines of code above.

6 Memory snapshot and forensics

Compared with other operating systems, linux has not paid much attention for debugging facilities inside kernel. Until linux 2.6.16 (or later) is modified to be able to be run on VMM, memory dump needs additional implementation. Fortunately in XEN, command "xm save" provides snapshot of memory. Memory snapshot is useful for malware profiling and forensics. In this section we discuss the way to take a snapshot of malicious behavior and analyze it.

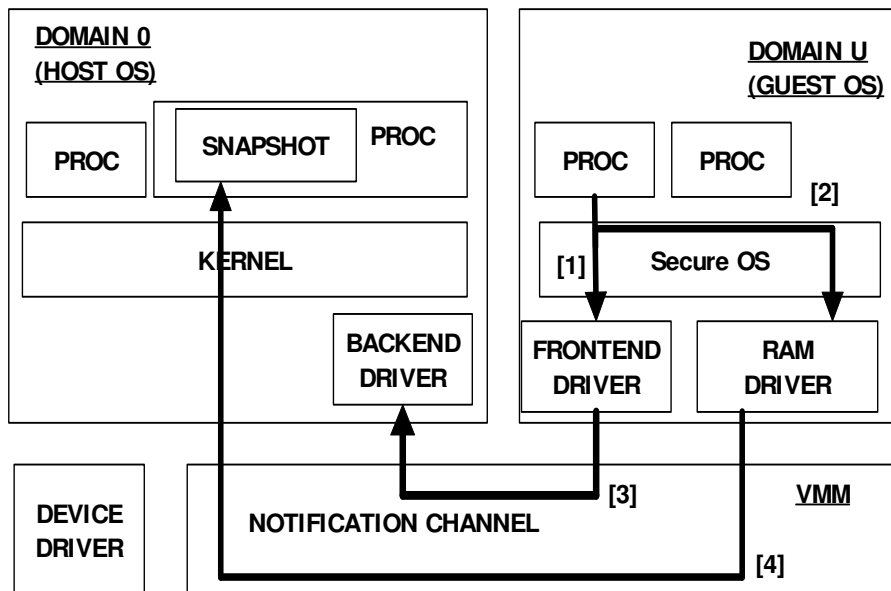


Fig. 4. Event-driven memory snapshot.

Figure 4 shows proposed snapshot system. Our method consists of four steps. Step [1] and [2]: buffer overflow, LKM installation and illegal file access

are detected the modification of exception handler, secure OS module and gcc-extension. Step [3]: notification of incidents goes from frontend driver to backend driver (interruption is activated). Step [4]: domain management tools (XM) takes snapshot of RAM.

6.1 LKM-rootkit

LKM (Loadable kernel module) is a kind of module which is plugged (embedded) dynamically into kernel. Unfortunately, this mechanism is also used for stealth rootkit implementation. LKM-rootkit changes core system (system call table, specific file system) of kernel. Therefore, once LKM-rootkit is installed, it is difficult to detect or profile it by application of user space. VMM takes advantages in coping with this malware because VMM is deployed "below LKM-rootkit" and can hook illegal changes on hosted domain.

6.2 System call extension

We insert hypervisor call into the source code of `create_module`. `create_module` is the system call when new module is installed into kernel. By doing this, we can obtain memory snapshot just in time LKM rootkit is installed. Also, the system calls `chown` or `lchown` are modified. After the installation of LKM rootkit, these system calls are used to hide file and process.

6.3 Behavior detect using MAC

Another extension of proposed system is inserting hypervisor call into LSM (linux security module) code. In this paper we apply MAC extension for LIDS (linux intrusion detection system). LIDS is security patch and admin tools for linux kernel to achieve MAC framework. We insert hypervisor call `EVTCHN_send` into i-node permission routine of LIDS as follows:

```
static int
lids_inode_permission(struct inode *inode, int mask,
struct nameidata *nd)
```

When READ/WRITE request is hooked, we can get information from `inode.i_lino`, `inode.i_sb.s_dev` and `d.d_iname`. By using MAC module, we can obtain memory snapshot when illegal file access (such as `/proc` and `/tmp`) is occurred.

7 Experimental result

In experiment, we count frequency of 5-gram (5-characters strings) on 128MB snapshot of RAM. We divide 128MB RAM into 1MB blocks. Figure 5 and 6 shows the number of times 5-gram appears on 1-64th and 65-128th blocks. Also, the frequency of the string "adore" is plotted. The number of blocks where the

string `adore` is found is 27. The total number of 5-gram appeared is 47618 with average 372.01 per one block. The total number of the string `adore` appeared is 7842 with average 245.06 per one 32 blocks. In general, we can conclude the frequency of the string `adore` is relatively high compared with other five strings.

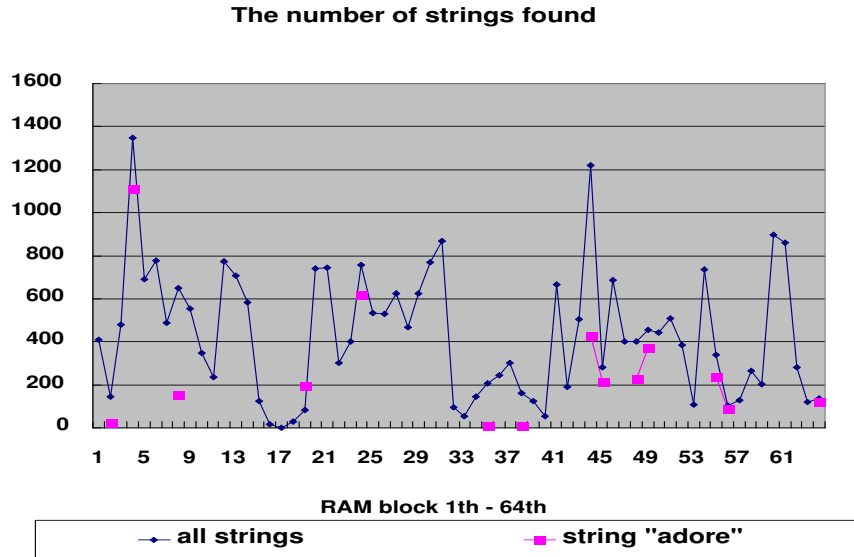


Fig. 5. The number of five character strings (5-gram) found. The string `adore` is found in 14 blocks. Perhaps the blocks of which the frequency is relatively small including `adore` are important (34 and 38).

Table 3 shows the number of strings found and grep time of 1-64th and 65-128th block. Totally, we can search (grep) all the strings found within 45 seconds. Also, we can search with the pipe option `grep -C N adore`. With N 10, we can search all blocks for about 40 seconds. We use weighted resolution. Proposed method is linear algorithm based. As a result, computing time is constant regardless of the number of strings found. We can conclude that we have obtained the expected result.

8 Conclusion and further work

VMM (virtual machine monitor) provides the useful inspection and interposition of guest OS. With proper modification of guest OS, we can obtain event-driven RAM snapshot of guest OS for unknown malware forensics. In this paper we propose the implementation of split kernel module for asynchronous memory

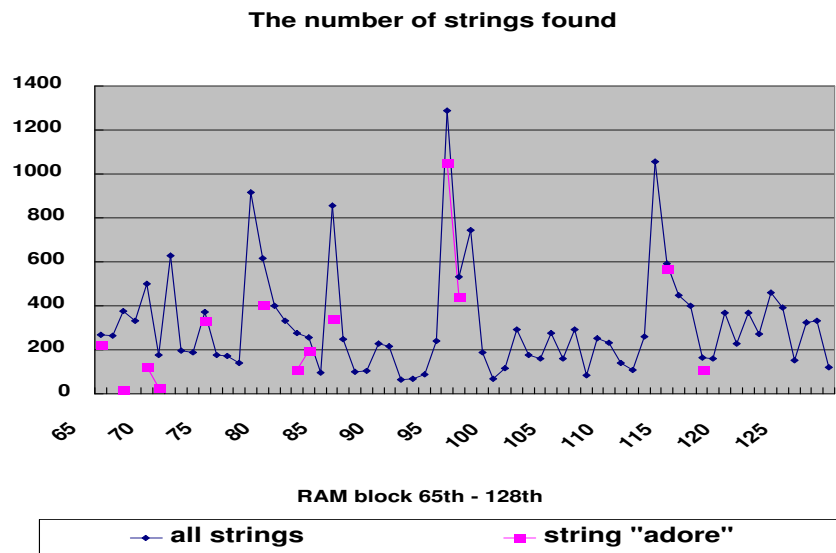


Fig. 6. The number of five character strings (5-gram) found. The string adore is found in 13 blocks. The blocks of which the frequency is relatively small including adore are important (66 and 121).

snapshot and forensics. Our split kernel module works for new virtualized interruption handler which can be combined with fault handler and security module (MAC, stackguard and etc). In proposed system we insert send code of event-channel (evchn_send) for activating virtualized IRQ of frontend driver. Then, backend kernel module receives the asynchronous notification and VMM take snapshot. In this paper, proposal system is applied for for buffer overflow handling and malware installation (including illegal resource access). In experiment, we take RAM snapshot of LKM-rootkit installation using system call extension. We extracted frequently appeared n-gram using weighted resolution. It is showed that we can find evidence blocks of RAM within 40-45 seconds by grep command. Proposal method is linear algorithm based, which makes computing time constant regardless of the number of strings found and grep hits. For further work, grant table (shared memory) manipulation can improve proposed system.

References

1. Greg Goth, "Virtualization: Old Technology Offers Huge New Potential," IEEE Distributed Systems Online, vol. 8, no. 2, 2007
2. Paul A. Karger, Mary Ellen Zurko, Douglas W. Bonin, Andrew H. Mason, Clifford E. Kahn, "A Retrospective on the VAX VMM Security Kernel", IEEE Trans. Software Eng. 17(11): 1147-1165, 1991

the number of strings found	47618 strings
grep hits (1-64) with strings found	51908 blocks
grep hits (65-128) with strings found	36691 blocks
the number of blocks with "adore" found	27 blocks
grep time (1-64)	22.198s
grep time (65-128)	23.433s
grep time with grep -C N adore (1-64)	20.681s
grep time with grep -C N adore (65-128)	20.229s

Table 2. Numerical result of extracting the string adore from memory snapshot. The number of blocks where adore was found is 27. Proposed method is linear-algorithm based. Computing time is constant regardless of the number of strings found and grep hits.

- Paul A. Karger, Mary Ellen Zurko, Douglas W. Bonin, Andrew H. Mason, Clifford E. Kahn, "A Retrospective on the VAX VMM Security Kernel", IEEE Trans. Software Eng. 17(11): 1147-1165, 1991
- XEN virtual machine monitor, <http://www.cl.cam.ac.uk/Research/SRG/netos/xen/>
- Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In Proceedings of the 19th Symposium on Operating System Principles(SOSP 2003), Bolton Landing, NY, October 2003.
- KVM: Kernel-based virtualization driver available at: <http://kvm.qumranet.com/>
- A Virtual Machine Introspection Based Architecture for Intrusion Detection Tal Garfinkel and Mendel Rosenblum In the Internet Society's 2003 Symposium on Network and Distributed System Security (NDSS), pages 191–206, February 2003.
- Nguyen Anh Quynh, Ruo Ando, and Yoshiyasu Takefuji : "Centralized Security Policy Support for Virtual Machine", USENIX, 20th Large Installation System Administration Conference, December 2006.
- Reiner Sailer and Trent Jaeger and Enriquillo Valdez and Ramon Caceres and Ronald Perez and Stefan Berger and John L. Griffin and Leendert van Doorn, "Building a MAC-Based Security Architecture for the Xen Opensource Hypervisor", in Proceedings of the 2005 Annual Computer Security Applications Conference (ACSAC), December 2005.
- ReTrace: Collecting Execution Trace with Virtual Machine Deterministic Replay, Min Xu, Vyacheslav Malyugin, Jeffrey Sheldon, Ganesh Venkitachalam and Boris Weissman, MoBS2007, June 2007.
- Sanjay Bhansali, Wen-Ke Chen, Stuart De Jong, Andrew Edwards, and Milenko Drinic, "Framework for Instruction-level Tracing and Analysis of Programs", Second International Conference on Virtual Execution Environments (VEE06), June 2006.
- George W. Dunlap, Samuel T. King, Sukru Cinar, Murtaza Basrai, and Peter M. Chen. ReVirt: Enabling intrusion analysis through virtual-machine logging and replay. In Proceedings of the 2002 Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA, December 2002.
- Samuel T. King, Peter M. Chen, Yi-Min Wang, Chad Verbowski, Helen J. Wang, Jacob R. Lorch, "SubVirt: Implementing malware with virtual machines", in Proc. IEEE Symp. on Security and Privacy (the Oakland Conference), May 2006.

14. C.Cowan, C.Pu, D.Maier, J.Walpole, P.Bakke, S.Beattie, A.Grier, P.Wagle, Q.Zhang, and H.Hinton, "StackGuard Automatic adaptive detection and prevention of buffer-overflow attacks", in Proceedings of 7th USENIX Security Conference, 1998.
15. Intel Corporation: IA-32 IntelR Architecture Software Developer's Manual, Volume 2A: Instruction Set Reference A-L,2004.
16. Intel Corporation: IA-32 IntelR Architecture Software Developer's Manual, Volume 2B: Instruction Set Reference N-Z,2004.
17. Uhlig, R.; Neiger, G, Rodgers, D, Santoni, A.L, Martins, F.C.M, Anderson, A.V, Bennett, S.M, Kagi, A, Leung, F.H, Smith, L, "Intel Virtualization Technology," IEEE Computer Volume 38, Issue 5, pp. 48–56, May 2005.

Appendix A: Modification for full-virtualization

Full virtualization is possible in recent processors such as Intel(R) Virtualization Technology (Intel-VT). In previous processors, ring 0 is assigned to VMM and OS need to be modified. Figure shows the full-virtualization system where VMX non-root mode and VMX root mode is added. When the control switches from VMX non-root mode to VMX root mode, the context of CPU is saved. We can detect the changes of register on guest OS by VMM and notify it to host OS. Instead of implementing the split device driver, we can construct the same system by inserting code of changing registers (such as DR, TR and MSR) into probes.

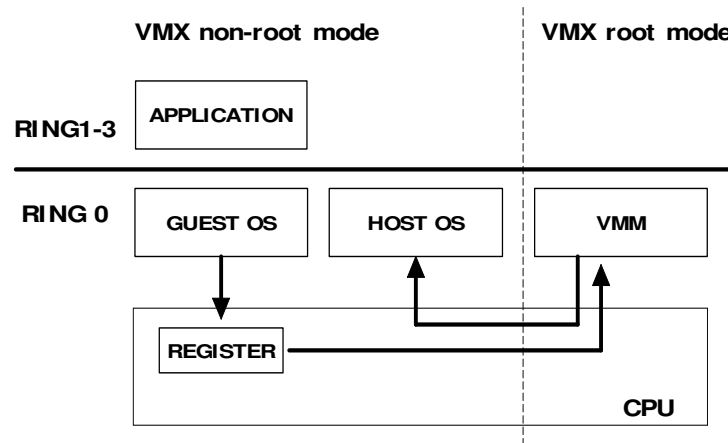


Fig. 7. Full virtualization. Ring 0 is available for both OS and VMM. By changing special registers such as DR, TR and MSR, notification can be transferred by the modification of register handling function of host OS.