Virtualization Based Security Framework (vBASE)

Srujan Das Kotikela
CSE Department
University of North Texas
SrujanKotikela@my.unt.edu

Satyajeet Nimgaonkar
CSE Department
University of North Texas
SatyajeetNimgaonkar@my.unt.edu

Mahadevan Gomathisankaran
CSE Department
University of North Texas
mgomathi@unt.edu

I. Introduction

In general, computer security aims at providing confidentiality, integrity and availability to computing systems. Traditionally, researchers in the fields of computer security have used software and hardware mechanisms for implementing security in computing systems. Software only security approaches typically deal with application level and Operating System (OS) level security mechanisms. Though, software-only mechanisms are easy to implement and patch, they suffer from heavy false-positives and false-negatives, thus making them vulnerable to software attacks and untrustworthy.

Hardware security solutions, such as ABYSS [10], AEGIS [9], Arc3D [5], Hide [12] and XOM [13], on the contrary, involve changes to the micro-architecture (typically adding new instruction sets, privilege levels etc.). These modified secure architectures deploy hardware mechanisms like memory encryption to provide confidentiality and memory authentication to ensure the integrity of the applications. However, adopting these secure architectures involves changes to the micro-architecture. Also, since the fabrication cycle is long (usually 5 years), testing these architectures is a time consuming process and therefore adopting these architectures as a security solution becomes infeasible.

Virtualization Technology introduces a software abstraction layer or virtualization layer (virtualization software) between the hardware and the operating system. This software abstraction layer is known as a Virtual Machine Monitor (VMM) [2] or the hypervisor. Virtualization has the power to emulate any required hardware features and project it to the OS. This feature of virtualization makes it much easier to incorporate security mechanisms within the virtualization layer. Also virtualization is supported by almost all the hardware vendors. Thus the security of the system can be increased without incurring excessive costs and performance overheads. Hence with this motivation, we propose a virtualization based security framework (vBASE).

II. vBASE Framework

The primary goal of vBASE is to aid the testing of security architecures and to enhance the existing security features provided by virtualization to provide a framework that can be deployed in Cloud Computing environments. The overall framework is shown in Figure 1. The vBASE framework is developed by modifying the Xen VMM [20]. The various component of the vBASE framework are as follows.

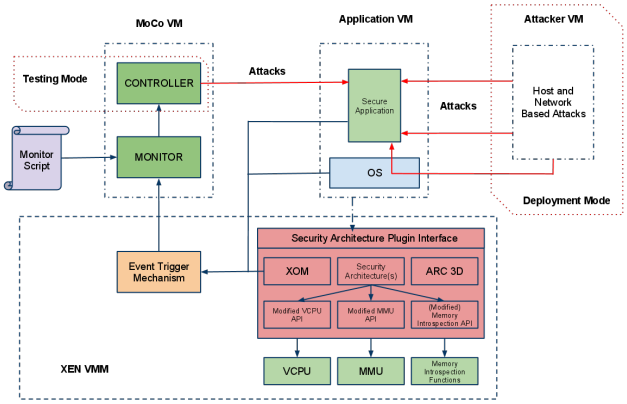

Fig. 1. vBASE Framework

The MoCo VM: This is a DOM 0 virtual machine in which the Monitor and/or Controller is running.

Application VM: This is a DOM U virtual machine in which the secure process/application is running.

Monitor: The Monitor is responsible for monitoring all the events at various levels pertaining to the secure processes/applications running on the security architecture.

Controller: The Controller is only used in testing mode of operation. Here, it has complete control of the memory, network and I/O etc of a secure process/application running on the Application VM. This is used to mount attacks on such a secure process/application. These attacks test the effectiveness of security measures provided by a security architecture.

Events: Events are used to notify various actions from application, operating system and SAPI to the monitor and other components of the vBASE. Events are implemented using Hypercalls [21] (for application, OS to SAPI communications) and VIRQS: Virtual Interrupts (for SAPI/VMM to Monitor communication). We have implemented custom hypercall VBASE_OP hypercall with commands like ENTER, EXIT, ENCRYPT, DECRYPT etc. and Virtual intterupts like VIRQ_VBASE for implementing events. For ease, we will refer to VBASE_OP hypercall with ENTER command as ***enter-vbase*** and likewise.

SAPI: The Secure Architecture Plugin Interface (SAPI) enables the VMM to plugin security architectures for testing and deployment purposes. A domain is associated with a security architecture through its configuration file. The secure

processes/applications running on these architectures by using a new set of instructions provided by SAPI e.g. ***enter-vbase*** and ***exit-vbase*** instructions. In testing mode, the plugged in security architecture reports the MoCo VM about the protected processes that it wants to be monitored.

III. Implementation

The example implementation shows how process data can be protected using virtualization. All the data to be protected is saved as one big chunk of memory.

The working of the secure process in conjunction with the security architecture is described in Algorithm 1

Algorithm 1 Secure Process Execution

1: Secure Process Begins
2: Allocate Secure Memory
3: The virtual address of secure memory is passed through a system call
4: Virtual address of secure memory is received in kernel module
5: Physical Frame Number (PFN) is computed for the received virtual address
6: Hypercall is used to pass the PFN to the hypervisor
7: PFN is received in the hypercall handler
8: Machine Frame Number (MFN) is computed for the received PFN
9: MFN is made read-only
10: Virtual Interrupt (VIRQ) is sent to Dom0
11: VIRQ is received in the Dom0 kernel
12: Dom0 kernel sends signal to the Monitor
13: Monitor prints messages to user

A. Source Code

```
 char secure_mem[PAGE_SIZE] secure_memory
     __attribute__((__aligned__(PAGE_SIZE
    )));
 int * secret_key = (int *) secret_mem;
 sys_vbase(secret_key, PROTECT, 0);
```

Listing 1. Secure Memory Allocation

```
pfn = vmalloc_to_pfn(secret);
HYPERVISOR_vbase_op(pfn, PROTECT);
```

Listing 2. Translate VA to PFN

```
mfn = gfn_to_mfn(v->domain, gfn , &
    old_type);
p2m_change_type(v->domain, gfn, old_type,
     p2m_ram_ro);
```

Listing 3. Translate PFN to MFN

```
//In xen
send_guest_global_virq(dom0, VIRQ_VBASE);

//In Dom0 kernel
bind_virq_to_irqhandler(VIRQ_VBASE,0,
    vbase_handler,NULL,NULL, 0);
```

Listing 4. Dispatch VIRQ from XEN

```
//In DomU
sec\_key = 654321;
sys_vbase(secret_key, WRITE, 654321);

//In DomU kernel
HYPERVISOR\_vbase\_op(pfn, WRITE,
    received_data);

//In VMM
secret_data = encrypt(received_data);
mfn = gfn_to_mfn(d, gfn , &old_type);
p2m_change_type(d, gfn, ot, p2m_ram_rw);
secure_write(mfn, secret_data);
p2m_change_type(d, gfn, p2m_ram_rw,
    p2m_ram_ro);
```

Listing 5. Secure WRITE

```
//In DomU
int secret\_data = 0;
secret\_data = sys\_vbase(secret_key,
    READ, 0);

//In DomU kernel
pfn = vmalloc\_to\_pfn(secret\_key);
return HYPERVISOR\_vbase\_op(pfn, READ);

//In VMM
mfn = gfn_to_mfn(d, gfn , &old_type);
read_data = secure_read(mfn);

secret_data = decrypt(read_data);

return secret_data;
```

Listing 6. Secure READ

IV. Future Work

We have implemented hypercalls, monitor, and event trigger mechanism of vBASE framework. We have implemented secure memory aspects of the SAPI. Our future work involves completing the SAPI for one of the secure architectures and implementing the controller.

REFERENCES

[1] Hai Jin. Virtualization Technology for Computing System: Opportunities and Challenges. High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference. 10.1109/HPCC.2008.167.

[2] Jyotiprakash Sahoo,Subasish Mohapatra and Radha Lath. Virtualization: A Survey On Concepts,Taxonomy And Associated Security Issues. Second International Conference on Computer and Network Technology, 2010.

[3] Donghai Tian, Deguang Kong, Hu Changzhen, Peng Liu. Protecting Kernel Data through Virtualization Technology. 2010 Fourth International Conference on Emerging Security Information, Systems and Technologies, 2010.

[4] Jianhua Che, Qinming He, Qinghua Gao, Dawei Huang. Performance Measuring and Comparing of Virtual Machine Monitors. 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2008.

[5] M. Gomathisankaran and A. Tyagi. Arc3D : A 3D Obfuscation Architecture, in High Performance Embedded Architectures and Compilers (HiPEAC). pp. 184199, Springer, 2005.

[6] Thomas Ristenpart, Eran Tromer, Hovav Shacham and Stefan Savage. Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds. CCS09, November 913, 2009, Chicago, Illinois, USA.

[7] Top Threats to Cloud Computing V1.0. Cloud Security Alliance. March 2010.

[8] M. Rosenblum and T. Garfinkel. Virtual machine monitors: current technology and future trends. IEEE Computer, vol. 38, no. 5, pp. 3947, 2005.

[9] Suh, G.E. O'Donnell, C.W., Devadas, S. Aegis: A Single-Chip Secure Processor. 2008 International Symposium on Information Science and Engieering, 2008.

[10] Steve R. White and Liam Comerford. ABYSS: An Architecture for Software Protection. IEEE Transactions on Software Engineering, vol. 16. no. 6, June 1990.

[11] Xianghua Xu, Peipei Shan, Jian Wan, Yucheng Jiang. Performance Evaluation of the CPU Scheduler in XEN. 2008 International Symposium on Information Science and Engieering, 2008.

[12] Xiaotong Zhuang,Tao Zhang,Santosh Pande. HIDE: An Infrastructure for Efficiently Protecting Information Leakage on the Address Bus. ASPLOS04, October 913, 2004, Boston, Massachusetts, USA.

[13] Jun Yang, Youtao Zhang, Lan Gao. Fast Secure Processor for Inhibiting Software Piracy and Tampering. Proceedings of the 36th International Symposium on Microarchitecture (MICRO-3603), 2003.

[14] Jeffrey Dwoskin, Mahadevan Gomathisankaran, Ruby Lee. Framework for Design Validation of Security Architectures. Princeton University Department of Electrical Engineering Technical Report CE- L2008-013, November 2008.

[15] Aiken, Mark, Fhndrich, Manuel, Hawblitzel, Chris, Hunt, Galen, Larus, and James R. Deconstructing Process Isolation. ACM SIGPLAN Workshop on Memory Systems Performance and Correctness, Oct 2006.

[16] Matt Piotrowski, Anthony D. Joseph Virtics: A System for Privilege Separation of Legacy Desktop Applications. http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-70.html.

[17] Intel Corporation. Intel Virtualization Technology . IEEE Publication,Volume: 38 , Issue: 5,Publication Year: 2005, Page(s): 48 - 56.

[18] Xen Split Drivers http://wiki.xensource.com/xenwiki/XenSplitDrivers. xen.org.

[19] Karen Scarfone, Murugiah Souppaya, Paul Hoffman, National Istitute of Standards and Technology (NIST). Guide to Security for Full Virtualization Technologies. Special Publication 800-125 (Draft)Natl. Inst. Stand. Technol. Spec. Publ. 800-125, 35 pages (July 2010).

[20] Xen Hypervisor. Xen Architecture Manual. www.xen.org.

[21] Xen Hypercall. xen.org. http://wiki.xensource.com/xenwiki/hypercall.

[22] VM Ware Workstation. VMware Inc. http://www.vmware.com.

[23] WikiLeaks. http://en.wikipedia.org/wiki/WikiLeaks.