International Journal of Engineering Research and Development
eISSN : 2278-067X, pISSN : 2278-800X, www.ijerd.com
Volume 2, Issue 3 (July 2012), PP. 40-45

Cutting-Edge Perspective of Security Analysis for Xen Virtual Machines

Omkar Kulkarni, Niu Xinli, Pamu Kumar Swamy

Abstract—Virtualization and cloud computing is rapidly changing the enterprise landscape with enterprises and even individual users opting for cloud based solutions. With the large number of benefits that it provides it also increases security risk and needs immediate attention owing to its wide adoptability. We present in this paper an analysis of two attacks on virtual machines which run on xen hypervisor concentrating on insider as well as outsider intruder. In xen, pygrub is the commonly used bootloader for domU, but posses most of the security breaches. We present a DoS attack that occurs due to the inherent behavior of pygrub. We also present another attack where it is possible to compromise passwords of guest domain either through the control domains root user or through an external user that gains these root privileges. We present some of the architectural pitfalls in xen that leads to these security breaches and suggest the importance of utilizing stub-domains that separates the functionality of control domain (dom0) to make xen a more secure hypervisor.

I. INTRODUCTION

Enterprises today are increasingly inclined towards utilizing a dynamic IT infrastructure as enabled by Cloud computing and Virtualization technology. Cloud computing enables enterprises to deploy and manage their services such as enterprise applications, e-commerce, content delivering and web hosting at a minimal expense of actual resources utilized. Thus, the enterprises data and services will be managed by some “trusted” corporation’s data center.

Virtualization is a key technology adopted for enabling cloud computing at data centers, where the applications or services run inside virtual machines mapped onto physical servers. Multiple virtual machines can run on one physical server depending on the need and resource utilizations of the applications. A software layer, virtual machine monitor (VMM) provides a virtual hardware abstraction that reflects the underlying physical machine. In some architectures such as Xen**[1]** there is an additional management domain that manages other virtual machines. Virtualization provides several advantages such as performance isolation, security, and easy management of resources.

Given the constant increase in the tendency of enterprises and individual users, offloading their data and computing services to the data center for number of usage scenarios, security of user’s sensitive data has emerged as a major concern. The data present in data centers is beyond direct control of individual users and enterprises, is constantly under attack from internal and external attacks. Either intended data access or accidentally employees of cloud providers can malice user’s sensitive data. External attacks such as Denial of services can cause service breakdowns for the enterprises cloud hosting services. This fact of security threats have been well studied by researchers [6,7,4] and brings out the significance of the need for improved security at these data centers.

Taking a closer look at the virtual machines based architecture that enables these data center, a major question arises around the trustworthiness of the VMM or the hypervisor together with the management domains if present. We can identify two major security loopholes, first one being the dependence on the additional privileged domain for drivers for physical devices and administrative tools which manages access controls along with the hypervisor but fails to manage implicit information flows [4]. The second one which emerges as a subsidiary of this is the higher privileges that the user of this domain enjoys leading to a potential internal threat. Also compromising this domain leads to the compromise of all the VMs running on that physical system. This enforces the need for a better and secure architecture for the virtualized architectural environments.

In this paper, we first discuss xen’s architecture and the role of management domain. We then showcase a typical attack scenario that presents a potential for Denial of Service attack. We then discuss another approach to circumvent this attack and in the following section we present need for stub domain based architecture.

II. RELATED WORK

Virtualization technology and virtual machines provides numerous technical and cost advantages. But these advantages doesn't come for free, it also brings some risks with it too. The work by Keisuke Okamura[4] shows a technique to create load based covert channels between xen virtual machines. Here they use cpu load as covert channels. They have developed CCCV, a system that creates a covert channel and communicates data secretly using CPU loads.The work by Rafal Woitczuk[7] outlines the recent work by the author to design and develop a backdoor for machines running the Xen hypervisor. An attacker can gain backdoor control over the host by overwriting xen code and datastructures. The work by Ristenpart et al. [8] shows an attack that identifies whether a particular virtual machine is likely to reside on the same physical server of a cloud infrastructure. It also demonstrates a side-channel attack that causes information leakage across virtual machines. These different attacks demonstrated in the above mentioned research work alleviate the security concerns. In our work we present analysis of a denial of service (DoS) attack and other attack scenarios to demonstrate the need for better security architecture for Xen.

The work by Fengzhe Zhang et al[9] shows the design and working of cloudvisor, a retrofitting protection of virtual machines in multi-tanent cloud with nested virtualization. This approach protects the privacy and integrity of customers virtual machines on virtualized infrastructures. sHype [10] is a MAC-based security extension to Xen. sHype enables administrators to apply various security policies to virtual machines running on an sHype-extended hypervisor. The work by Jon Oberheide et al [11] describes the Empirical Exploitation of Live Virtual Machine Migration. They defined and investigated three classes of threats to virtual machine migration: control plane, data plane, and migration module threats. Then showed how a malicious party using these attack strategies can exploit the latest versions of the popular Xen and VMware virtual machine monitors and presented a tool to automate the manipulation of a guest operating system's memory during a live virtual machine migration. Patrick Colp et al [12] presented Xoar, a modified version of Xen that retrofit the modularity and isolation principles used in xen. Xoar breaks the control domain into single purpose components and showed numerous advantages of this componentization. With the study of these security architectures, in this paper we present our thoughts for making xen less vulnerable.

III. XEN AT A GLANCE

Xen, a virtual machine monitor(VMM) known as hypervisor too, is the basic abstraction layer of software which directly sits on the hardware (i.e. bare-metal or type1 hypervisor) that lies below any operating system and allows execution of multiple virtual guest operating systems simultaneously on a single physical machine.

The architecture diagram is shown in figure1. Xen is responsible for CPU scheduling and memory partitioning of various virtual machines running on hardware device. It also controls the execution of virtual machines as they share the common processing environment.

3.1.Domain0(Dom0): Xen runs guests in environments known as domains, which encapsulate a complete running virtual environment. When Xen boots, it does load a Domain0 (dom0) guest kernel. Domain 0 is the first guest to run, and has elevated privileges. All Xen virtualization environments require Domain 0 to be running before any other virtual machines can be started. Xen does not include any device drivers by itself, nor a user interface. These are all provided by the operating system and userspace tools running in the dom0 guest. Domain 0 has special rights to access physical I/O resources as well as it interact with the other virtual machines (Domain U: PV and HVM Guests) running on the system.

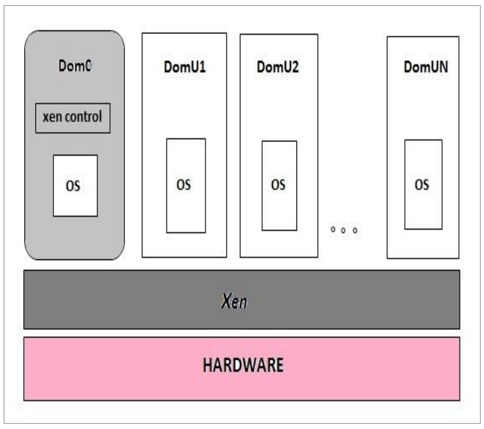


Figure 1: Xen architecture

3.2.DomainU(DomU): An unprivileged domain (domU) guest is more restricted. DomainU guests have no direct access to physical hardware on the machine. Unlike dom0 guests, you can have an arbitrary number of domU guests on a single machine, and they may be able to be migrated. Xen supports Para and full virtualization. All paravirtualized virtual machines running on a Xen hypervisor are referred to as Domain U PV Guests and are modified Linux operating systems, Solaris, FreeBSD, and other UNIX operating systems. All fully virtualized machines running on a Xen hypervisor are referred to as Domain U HVM Guests and run standard Windows or any other unchanged operating system. The Domain U PV Guest virtual machine is aware that it does not have direct access to the hardware and recognizes that other virtual machines are running on the same machine. The Domain U HVM Guest virtual machine is not aware that it is sharing processing time on the hardware and that other virtual machines are present. For each HVM a special daemon is started in domain0, Qemu-dm. Qemu-dm supports the DomainU HVM Guest for networking and disk access requests.

IV. PYGRUB

The bootloader most commonly used to boot domU is PyGrub or python grub. PyGrub is a part of xen-utils package. DomU's boot configurations are specified in config file using kernel, ramdisk and some extra lines. PyGrub is the alternate method which specifies a 'bootloader' line in config file and uses that to load a kernel from domU's filesystem.

PyGrub enable us to start linux domUs with a kernel inside domU’s filesystem instead of a kernel that lies in the filesystem of dom0. This means easier management i.e. domU manages its own kernel and initrd. This enables easier kernel update from domU or migration of HVMed linuxes.

When we try to load/start a domU(PV Guest), a request is sent to domain loader from xen to load the required domain. Domain loader runs pyGrub Utility. PyGrub itself runs in dom0 root privileges. It opens PV guest disk image and reads grub menu.lst from disk. PyGrub copies the required kernel image and ramdisk image in /tmp i.e. dom0 filesystem and closes the disk. The domain loader reads /tmp and loads particular kernel and domU get started. The architecture diagram is shown as below in figure2.

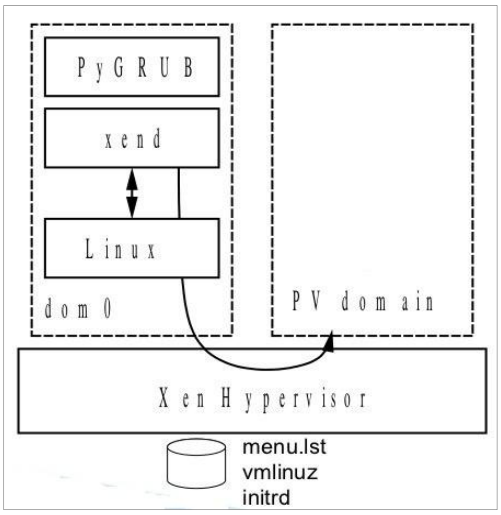


Figure2: PyGrub

Problems with pygrub:

1. PyGrub runs in root privileges of dom0. It reads the data directly provided by domU user. Such a dom0 root process that parses user-provided data is a potential security breach.
2. pyGrub bootloader used to boot Xen para-virtualized guests did not have support for password command as supported by normal grub. If this option was used in grub.conf, it did not restrict users with access to para-virtualized guest's console from booting guest or changing kernel boot parameters without providing configured password. This could allow user with access to VMs console to get root privileges on the guest's operating system.[16]

3. pyGrub does not check the size of bzip2-compressed kernel. This can be a security breach and can be used as a DoS to guest PV domains
4. PyGrub (the version we use anyway) is unfortunately not a complete re-implementation of GRUB, so functionality such as the menu does not readily exist. It serves mainly to allow specification of the kernel, initial ramdisk and boot-time parameters.
5. It does not provide network boot functionality.

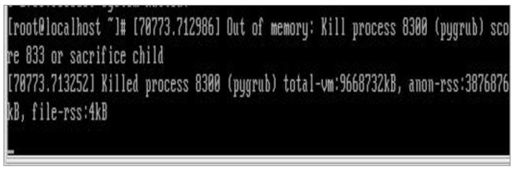
V. DENIAL OF SERVICE ATTACK(DoS)

A denial of service attack (DoS) is an incident in which a user is deprived of the service of resource they would normally expect to have. Here we present a scenario that shows a DoS type attack on guest virtual machines hosted by an hypervisor. Let a hypervisor is hosting n-domUs, say domain-A, domain-B and so on. Each domain is provided with its own separate disk space. All domains are given equal weightage. Required resources are provided and VCPUs are mapped to core CPU in fair scheduling policy. Let user of domain-A is an attacker/intruder. The pygrub i.e. pv bootloader [python bootloader] doesn't check the size of a bzip2 or lzma compressed kernel image. Denial of service can be caused by padding a large file at the end of the kernel image, which can be used by a malicious domU root i.e. by domain-A user. Attacker takes advantage of this breach. What an attacker actually does is, he compiles a bzip2 compressed kernel, attaches a large file of size approximately 10GB to kernel image and boots through the newly compiled kernel. To start this heavy kernel, most of the resources are occupied by domain *A* denying rest of the domains from those resources for particular time till that domain get booted. These resources are occupied by domain a till it gets started, then the resources are released. The attacker will reboots the domain a in tolerant time interval so that resources are occupied by this domain a most of the time, denying other domains to use those resources.

VI. EVALUATION OF DoS

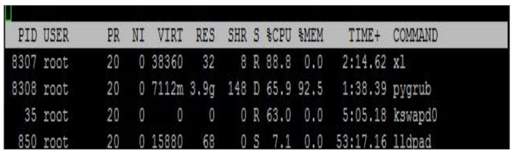
The test environment platform was a single desktop machine with an Intel core2 duo processor and 4GB main memory(RAM). The cloud infrastructure was emulated by xen 4.1-amd64 hypervisor. The management VM was dom0 running ubuntu server 11.10 with inbuilt xen enabled kernel 3.0.0-12-generic. We ran ubuntu 10.10 with kernel 2.6.35-22-generic as domU.

First to compress a kernel in bzip2 and to compile the bzip2 compressed kernel we downloaded a kernel which has domU support. For our experiments we used Linux-3.1. We compiled this bzip2-compressed kernel in domU. Then we padded a file of large size approximately of size 2GB(generated by dd from /dev/zero) to this compiled kernel image. After trying for booting the system through this heavy kernel we got actual results. When we allocated 2GB memory to dom0 with 4.3GB swap space, the domU can't be started because of lack of memory. The system was busy in swapping but eventually ran out of memory and swap space. See figures 3 and 4 for references.



```
[root@localhost ~]# [70773.712986] Out of memory: Kill process 8300 (pygrub) sco
re 833 or sacrifice child
[70773.713252] Killed process 8300 (pygrub) total-vm:9668732kB, anon-rss:3876876
kB, file-rss:4kB
_
```

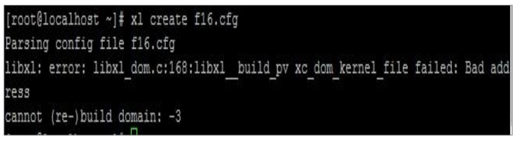
Figure3: system out of memory



```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 8307 root      20   0 38360   32    8 R 88.8  0.0   2:14.62 xl
 8308 root      20   0 7112m 3.9g  148 D 65.9 92.5   1:38.39 pygrub
   35 root      20   0     0    0    0 R 63.0  0.0   5:05.18 kswapd0
  850 root      20   0 15880   68    0 S  7.1  0.0  53:17.16 lldpad
```

Figure4: xen related processes occupy the cpus, and kswapd is busying swapping

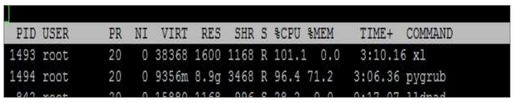
The most of the resources were occupied by this booting process to boot domU. It took nearly 10 minutes and pygrub got crashed due to lack of memory, shown in figure5.



```
[root@localhost ~]# xl create f16.cfg
Parsing config file f16.cfg
libxl: error: libxl_dom.c:168:libxl__build_pv xc_dom_kernel_file failed: Bad add
ress
cannot (re-)build domain: -3
```

Figure5: PyGrub crashed

Then we allocated **4GB** memory to dom0, now domU can be started but with a great cost, as shown in figure6.



```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1493 root      20   0 38368 1600 1168 R 101.1  0.0   3:10.16 xl
 1494 root      20   0 9356m 8.9g 3468 R 96.4 71.2   3:06.36 pygrub
```

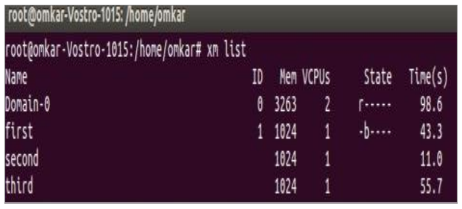
Figure6: domU boots successfully at great cost

At the same time when the booting was going on we tried to use some resources from other domUs running but got denied. If the domU with heavy kernel is rebooted in some xen tolerant interval, the most of the resources are occupied by malicious domU all the time.

VII. CLERATEXT PASSWORDS IN MEMORY SNAPSHOTS

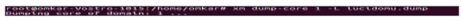
In this section we are going to discuss a problem of a malicious insider in a cloud which is based on Infrastructure as a service(IaaS) model. The service availing user mostly trusts the service provider for the storage and safety of his data. The attack demonstrated below shows that how a malicious insider can steal confidential data of a cloud user. Thus, concluding that how unwise it is to trust a cloud service provider mostly for safety of stored data.

[15]What an intruder does is, he just simply issues a dump-core command, which is a part of xen management(xm) user interface. By issuing this command on a target VM, malicious insider takes a snapshot/dump of a target VM. The dump-core command performs a dump of the memory region reserved to the targeted VM. The attacker only needs to identify the VM in the command. In our case, to confirm the presence of passwords in the dump file, we used the cat command to send the dump into the standard output that was redirected to the strings and grep commands in order to locate the known password. See the figures, 7,8 and 9 for more details.



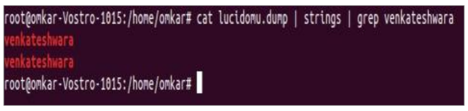
```
root@omkar-Vostro-1015: /home/omkar
root@omkar-Vostro-1015:/home/omkar# xm list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0  3263     2     r-----      98.6
first                                        1  1024     1     -b----      43.3
second                                          1024     1                 11.0
third                                           1024     1                 55.7
```

Figure7: xm list



```
root@omkar-Vostro-1015:/home/omkar# xm dump-core 1 -L lucidomu.dump
Dumping core of domain: 1 ...
```

Figure8: dump-core command



```
root@omkar-Vostro-1015:/home/omkar# cat lucidomu.dump | strings | grep venkateshwara
venkateshwara
venkateshwara
root@omkar-Vostro-1015:/home/omkar#
```

Figure9: Analyzing the dump for presence of password

VIII. THE ANALYSIS AND SOLUTION PHILOSOPHY FOR DoS AND PASSWORD STEALING ATTACK

An in depth analysis for finding the root cause of the DoS attack and the determining of passwords in memory dump and other similar attacks demonstrated is the inappropriate architecture of the control domain i.e. the dom0 in security point of view. This control VM enjoys highly elevated priveldges and is responsible for all major tasks including the booting of VM, enabling the transfer of network data, disk access, migration of VM etc. This poses us with an important question, is this domain worth of these highly elevated privileges? From the functional working of Xen architecture point of view, this domain actually requires such high privileges. If this is required then how do we solve the root cause for the security breaches presented? Let us first analyze a possible solution for the DoS attack.

Pv-Grub:

PV-GRUB is a real GRUB source code recompiled against Mini-OS, and works more like a usual, full fledged bootloader so overcoming the drawbacks of pygrub, which is not a usual bootloader. In PV domU it is just required to provide a path to PV-GRUB kernel in configuration file. PV-GRUB replaces PyGrub to boot domU image safely. It runs regular grub inside created domain itself and uses regular domU facilities to read the disk, so it reduces the overload of tasks of dom0. It can fetch from network too i.e. provide network boot facility. It reads domU disk and network interface of domain and access menu.lst of PV guest. Then uses the regular PV console to show grub menu and loads the required kernel image. The actual architecture and working of PV-GRUB is shown in figure10 below.

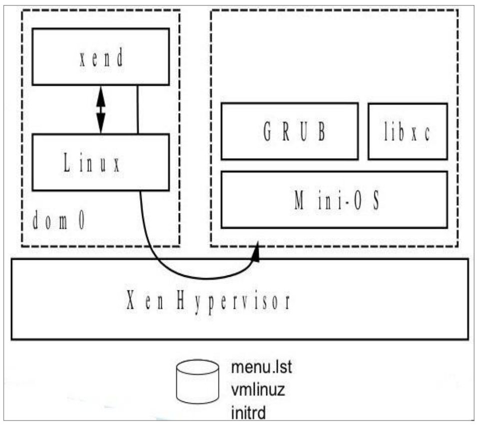


Figure10: PV-Grub

pv-grub lies in philosophy with breaking the control VM into parts[12]. The functionality of dom0 is broken into different stub-domains those run as normal domains on xen hypervisor. This breaking is useful in management as well as security point of view.

Looking at the password attack, there is no necessity of allowing dom0 a full control of guest domain. Activities such as guest OS dump must not be allowed at all from dom0. If at all required for all such tasks like taking memory dumps etc. must be delegated to a new VM that is executed only on active involvement of Enterprises or user whose data lies on the cloud.

Thus breaking up the functionality of control VM can reduce the security risks. To break a management domain(dom0), we need to define the classes of interaction where dom0 is required for domU and this can be used as a basis for breaking.

IX. CONCLUSION

The Cloud computing and virtualization is a most promising paradigm with acceptance worldwide. The paper focused on the malicious insider as well as outsider threat. We also discussed the detailed analysis of these threats and a philosophical solution for these attacks. The Py-grub is responsible for DoS based attack. The dom0 is provided with all privileges and is responsible for all important tasks. To avoid internal threats, the stub-domains concept must be implemented and dom0 should not have the privileges alone like taking a dump of guest domU.

REFERENCES

[1]. Paul Rarham, Boris Dragovic, Keir Fraser, ‘Xen and the Art of Virtualization’, ACM2003

[2]. Reiner Sailer, Enriquillo Valdez, Trent Jaeger, ‘sHype: Secure Hypervisor Approach to Trusted Virtualized Systems’, IBM Research Report, February 2, 2005

[3]. Flavio Lombardi, Roberto Di Pietro, ‘KvmSec: A Security Extension for Linux Kernel Virtual Machines’, ACM 2009

[4]. Keisuke Okamaura, Yoshihiro Oyama, ‘Load based Covert Channels between Xen Virtual Machines’, ACM 2010

[5]. David Chisnall, ‘The Definitive Guide to the Xen Hypervisor’, Prentice Hall Publications, 2008

[6]. Understanding Cloud Computing Vulnerabilities, Article in 2011 IEEE journal

[7]. Rafal Wojtczuk, ‘Subverting the Xen hypervisor’, Black Hat USA 2008

[8]. T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS 2009),* 2009.

[9]. Fengzhe Zhang, Jin Chen, Haibo Chen and Binyu Zang, ‘Cloudvisor: Retrofitting Protection of Virtual Machines in Multi-tanent cloud with nested Virtualization’

[10]. R. Sailer, T. Jaeger, E. Valdez, R. C´aceres, R. Perez,S. Berger, J. L. Griffin, and L. van Doorn. Building a MAC-based Security Architecture for the Xen Opensource Hypervisor. In *Proceedings of the 21st Annual Computer Security Applications Conference*, pages 276–285, 2005.

[11]. Jon Oberheide, Evan Cooke, Farnam Jahanian, ‘Empirical Exploitation of Live Virtual Machine Migration’, ACM

[12]. Patrick Colp, Mihir Nanavati, Jun Zhu, William Aiello, ‘Breaking Up is Hard to Do: Security and Functionality in a Commodity Hypervisor’, *SOSP ’11*, October 23-26, 2011, Cascais, Portugal. ACM 2011

[13]. Xen 4.1, http://bits.xensource.com/oss-xen/release/4.1.0/xen-4.1.0.tar.gz

[14]. *VMware Workstation*. http://www.vmware.com/products/ws/

[15]. Francisco Rocha, Miguel Correia, 'Lucy In The Sky without diamonds: Stealing confidential data in the cloud'

[16]. www.google.co.in

[17]. https://bugzilla.redhat.com/show_bug.cgi?id=525740