ABSTRACT
Virtual machines are increasing in popularity and are being widely adopted. The concept of a virtual machine is however not new. Hypervisors are also popular from the perspective of security as a means of isolation and inspection. This paper attempts to understand the functioning of a hypervisor and how it can be leveraged for security. First, an overview of different hypervisor architectures is presented through articles and online resources. Next, the paper discusses the Hyper-V hypervisor architecture and its security guidelines are analyzed. Finally, recent research in the area of hypervisor security is analyzed. Research into the security of hypervisors is limited. Most security research is concentrated in the cloud platform where hypervisors play a prominent role. As an evaluation of security, some security bugs reported for the Hyper-V system and the corresponding mitigation strategies used are also discussed.

1. INTRODUCTION
Virtual Machines have re-emerged and have become a primary in most data center, server and business environments. Rosenblum and Garfinkel [7] explain that the very reasons that drove out virtualization in the 1980s, complex Operating Systems (OS) and emergence of highly efficient and scalable hardware, are also the main reason for the increasing adoption of virtual machines in the twenty first century. While OSes have improved from the early 1960s when virtual machines and hypervisors were proposed, they have also become more vulnerable due to the diversity of operations and applications that run today. In addition, advances in hardware have resulted in efficient performance but low utilization of hardware resources.

Virtualization is seen as an efficient solution for optimum use of hardware, improved reliability and security through the use of hypervisors monitoring virtual machines, cost reduction and lower space requirements. Server consolidation which refers to ability for multiple systems to be multiplexed over the same hardware is a major advantage driving virtualization in industry environments. Apart from scalability and multiplexing, the fact that virtual machines provide a solution for legacy systems and applications is helpful. Replication and migration of virtual machines becomes easier without having to terminate any instances [7].

A Virtual Machine Monitor (VMM) also known as a hypervisor is the layer of software that sits between the hardware and operating systems monitoring and maintain control of the hardware resources while interfaces OSes to the hardware drivers. Different models of virtualization depending on the hypervisor’s role have been developed. An overview of the types of virtualization techniques is discussed in the next section. It should be noted the hypervisors techniques discussed in here mainly address the x86 virtualization issues which may or may not apply to proprietary server level virtualization systems.

The use of a VMM for security is a formidable option. A VMM allows transparency of access and isolation of low-level resources. Wang et al [9] propose the HookSafe system based on a hypervisor to protect against kernel rootkits. The architecture and methodology adopted by the authors as well as their results are presented in Section 4 along with another security hypervisor sHype. Section 2 provides an overview on hypervisor types. Section 3 is an analysis on real-world deployed hypervisor - Hyper-V. Section 6 is the conclusion.

2. BACKGROUND
Virtualization allows multiple applications or operations to access and use the same resource while oblivious to accesses to the same resources by others. Virtualization provides access transparency and is a layer between the operating systems and the hardware. The operating systems or higher level applications are managed by a hypervisor or VMM. The VMM or hypervisor creates isolation paths as virtual machines where different operating systems run in virtual machines on top of the hypervisor. The hypervisor manages requests by virtual machines to access to the hardware. According to Wikipedia, virtual machines originated from the PR/SM hypervisor built for the IBM 370 mainframe system in 1970. Since then hypervisors have come a long way and are increasingly more complex and often similar to a complete operating system 1. In the paper [7], the authors describe the fundamental characteristics for a VMM. First,

1http://en.wikipedia.org/wiki/Hypervisor
the VMM creates an environment similar to the real environment for the virtual machines. Second, the slow down in performance is minor and thirdly, all the hardware is managed by the VMM with the VMM capable of re-allocating or re-occupying its control on existing resource allocations. The trap-and-emulate method for virtualization proposed by Popek and Goldberg is considered a classical virtualization technique although it is still prominent today. Variations of the trap-and-emulate model are currently used.

Hypervisors can be classified into two types - Type 1 and Type 2 hypervisors. Type 1 hypervisors run directly above the host hardware and monitor operating systems that run above the hypervisor. These are also known as bare-metal, embedded or host hypervisors. Bare metal hypervisors currently offer better performance. The hypervisor is small as its main task is sharing and managing hardware resources between different operating systems. A major advantage is that any problems in one virtual machine or guest operating system do not affect the other guest operating systems running on the hypervisor. In the case of Type 2 hypervisors, the hypervisor is installed on an operating system and then supports other operating systems above it. While having a base operating system allows better specification of policies; any problems in the base operating system affects the entire system as well even if the hypervisor running above the base OS is secure.

While hypervisors originally were developed for server platforms, virtualization for PC and Desktop OSes was soon realized. A major challenge facing Desktop OS virtualization was the virtualization of the x86 CPU architecture. The white paper [6] presents a detailed difference of the types of virtualization techniques explored by different vendors. The x86 architecture is often viewed as having four privilege levels as shown in the figure -

Virtualizing the x86 architecture requires a VMM blow the OS at privilege level 0. In addition, since some instructions directly execute on the hardware, interfacing would disrupt the working of several applications. One technique of virtualization is using binary translation and direct execution techniques. That is, the kernel instructions are changed to virtualized instructions to execute on the hardware. User applications with hardware instructions can directly execute on the hardware. In this case, the operating system is virtualized. This form of virtualization is known as full virtualization. Full Virtualization is a virtualization technique allowing the guest OS to directly run instructions through a modified VMM. Full virtualization therefore requires no assistant technologies and is often considered more secure and provides more performance. Another form of virtualization called Para-Virtualization works by modifying the guest OS to work along with the VMM to interface the virtualized hardware. This form of virtualization requires modification of the guest OS which is easier than creating binary instruction translation support. However, older unmodified operating systems such as Windows XP cannot be Para-Virtualized. A third form of virtualization is based on modified hardware which directly supports virtualized instructions without translation. Specifically, the Intel Virtualization Technology and the AMD-V virtualization engine allow the VMM to run in a root privilege mode below the OS at root 0 which translates any ‘unusual’ instructions which are caught by the VM trap. Hardware assisted virtualization has the advantage of reducing the overhead of modifying the OS as in para-virtualization. However, the disadvantage is the modification of host processors which may be cumbersome and difficult to implement for all vendor products.

Memory virtualization is achieved through the use of shadow page table. The OS page table maps to the VMM shadow page table which performs the translation to hardware address. One of the problems of memory virtualization is reclaiming memory. In para-virtualization adopted systems like VMware ESX this problem is solved through the use of a balloon process in the OS that communicates with the VMM [7]. In x86 virtualization systems, a Memory Management Unit (MMU) along with a Translation Lookaside Buffer (TLB) are used.

Device and I/O virtualization is addressed through the use of management operating systems or host operating systems that either lie at the same level as the hypervisor or above.
the hypervisor and have superior privileges compared to the guest OSes. The hardware drivers are loaded into this host operating system. The host OS interfaces I/O device requests from guest OSes and communicates with the VMM to access the hardware.

In the paper [5], the authors give a detailed overview of the functioning of software assisted VMs and hardware assisted VMs. The authors perform several benchmarking performance analysis tests. The hardware assisted VMM consisted of an in-house developed system running on HP xw4300 workstation with a virtualization technique enables 3.8 GHz Intel Pentium 4 672 processor with hyper-threading disabled. The software assisted VMM is a general virtualization commercial product. The first test measuring small user process saw a slowdown of approximately 4% for the software VMM and 5% for the hardware VMM. This however might change with processor speed and the hardware assisted VMM can easily overtake a software assisted VMM. To measure performance under server workload, the authors used Apache ab server benchmarking tool against Linus and Windows installations of Apache with 128 clients. In this case, the software assisted VMMs greatly outdistanced hardware assisted VMMs with 38% and 46% slow down respectively. The authors also find in/out instructions to dominate execution time for a Windows XP 64-bit boot/halt.

3. HYPER-V
Hyper-V is a popular virtualization platform. It is available for x64 versions of Windows Server 2008. Some of the major features highlighting hyper-V’s capabilities are its reliability through a micro-kernlized architecture which removes the need for multiple emulators, scalability and performance gain through server core that allows running as a low maintenance-limited functionality environment, strong isolation through the use of partitions for VMs, virtualization aware hardware which controls the resources available to each entity, and several security features.

3.1 Architecture
The hyper-V architecture is based on micro-kernlized hypervisors [1]. Micro-kernlized hypervisors are different from monolithic hypervisors that depend on the single Virtual Machine Monitor which is code heavy. The VMM emulates and handles all lower level accesses and therefore is also slower. This is the traditional or classical hypervisor similar to the emulate-and-trap model by [7]. The guest OSes need to request hardware access from the hypervisor as the device drivers are integrated into the hypervisor. The hypervisor in this case provides transparency. A micro-kernel provides limited functionality such as managing memory address space, process communication and management while other hardware management tasks are typically managed by processes independent of the kernel.

Figure 4: Hyper-V top level architecture

The micro-kernlized hypervisor does not contain device drivers. The device drivers are present in the OS in the parent or root partition. In addition, since hyper-v uses virtualization aware hardware, the hypervisor code base is reduced while improving performance since the hardware can directly handle virtualized instructions. The virtualization-aware processors such as Intel’s VT and AMD-V allow each VM to 4 processors per VM.

In hyper-V virtualization is achieved through partitions. Each virtual machine is run in a separate logical partition. A root or parent partition contains the Windows 2008 x64 OS which is responsible for creating the other partitions that contain different guest OSes. The parent partition also contains a Virtualization Stack which is combination of user and kernel level code and interfaces with the hypervisor and the child partitions. In addition, the virtualization stack manages memory for child partitions. The virtualization stack in the parent partition has direct access to the hardware resources and creates partitions using calls to the hypervisor application programming interface. The hypervisor monitors processor interrupts and guest OSes do not have access to the processor. Translation of guest virtual address spaces to physical addresses is done using the Input Output Memory Management Unit which operates independent of the memory management unit in the CPU. IOMMU uses hardware acceleration features for faster address translation. The hypervisor is a thin software layer which enforces memory and CPU usage and access specifications. Partitions can share hardware resources but all access is governed by the hypervisor.

The parent partition manages the input-output devices such as keyboard, mouse, printer and other devices connected to the hypervisor. It handles access to devices through virtual devices. Device virtualization is achieved through the use of a provider-consumer model. The child partitions containing guest OSes request the virtual devices through the virtual memory bus. Access to the physical devices is controlled by the hypervisor. According to the hyper-V documentation, the virtual memory bus is a logical inter-partition communication channel. The child partitions communicate over the VMBus to the Virtualization Service Provider which is in the
3.2 Security

According to the hyper-V documentation [1] and security guide [2], security features developed for hyper-v include:

- Server hardware containing virtualization features can improve security by providing the ability to disable the execution bit preventing viruses and worms from executing.

- The micro-kernelized architecture improves security by avoiding the need to load third-party drivers in the hypervisor.

- The ability to provide role-based access control mechanisms which are necessary in a server coalition environment. Streamlined, lightweight architecture with automatic network address translation and network access policy protection features. Microsoft in their security guide (give citation), recommends using the server core model. As explained before, enabling server core reduces attack surface as it has a limited operation environment. It also adds to performance gains. However, the downside of this approach is that several drivers and units may not be compatible with the server core system. In addition, .NET support is lost for the management operating system. However, as we will see in the next section, some security vulnerabilities bypass even the server core system.

In addition, it is better to have an additional network connection reserved for the management operating system as it provides an administrator with timely access and separates the VM network from the management network. Enabling the hardware enforced Data Execution Prevention feature in the BIOS prevents execution of memory reserved for data. Hyper-V does not impose any limits on the processor usage by a virtual machine but Microsoft recommends specifying limits for different VMs depending on their workload. This can prevent malicious VMs from providing Denial of Service to other VMs.

The Hyper-V configuration files are stored in the directory %programdata%\Microsoft\Windows\Hyper-V\. Administrators have full access to this directory. Services and interactive processes may have read-write access while batch processes have only delete access. System processes and user-defined process accesses to virtual hard disk files should be monitored through access control lists. In such a case even if a rootkit installs on an OS, it should not be able to rewrite system files. In addition, encryption systems can be used to protect system files and disk drives.

Firewall and anti-virus need to be enabled separately on each VM and the hypervisor is not responsible for any malware or rootkits affecting the guest OSes.

3.3 Reported Vulnerabilities

**MS10-010 Vulnerability - Hyper-V Instruction Set Validation Vulnerability [3]**

This vulnerability allows an attacker with privileges to one of the guest OSes running on Hyper-V to manipulate a set of machine instructions in the guest-OS. This vulnerability can allow an attacker to perform a Denial Of Service (DoS) attack. The attack can be remotely managed by an anonymous user. The vulnerability affects all Microsoft Windows 2008 products that even run in the server core mode.

According to CWE(Common Weakness Enumeration) [4], the vulnerability is due to improper validation of input data corrupting the control flow or control data i.e. in other words the kernel hooks. Mitigation: The only reported mitigating factor is that the attacker must have valid local logon credentials with sufficient privileges. Patch: A patch eliminating the vulnerability was released by Microsoft.

**MS10-102 Vulnerability [4]**

This vulnerability is exploited by sending a special, carefully crafted malicious packet to the VMBus in Hyper-V from any guest OS. The attacker needs to have local logon credentials to one of the guest-OSes and cannot remotely exploit the vulnerability. The exploit results in the Hyper-V server from stopping to respond causing a Denial of Service attack(Dos). It affects Microsoft Windows Server 2008 and Microsoft Windows Server 2008 R2 even with the server core option installed.

4. RELATED WORK ON HYPERVISOR SECURITY

**HookSafe** is hypervisor based kernel rootkit protection system proposed by Wang et al [9]. Kernel rootkits due to their stealthy nature are hard to detect as they directly subvert the control data in the kernel space. Protection against such rootkits requires not only code integrity but also integrity of control data. Control data refers to any kernel data that is uploaded to a program counter at some point in kernel execution. Kernel control data can be classified into return addresses and function pointers. Return addresses are mostly used in the case of buffer overflow attacks. Kernel rootkits themselves can be divided into Kernel Object Hooking (KOH) and Dynamic Kernel Object Manipulation (DKOM). KOH rootkits hijack the entry points of a control flow. They typically hijack the code hooks or data hooks. Code hooks are static and any modification is easy to inspect. However, hijacking data hooks that point to physical memory are hard to eliminate. Control flow rootkits reside both in the pre-allocated memory areas as well as dynamic memory areas. Approaches based on hardware-based page-level protection

work as long as the kernel hooks are co-located to each other and present in small number which is usually in contrast to the widely scattered hooks in Windows and Linux OSes. Furthermore, the authors point out that dynamic allocation of hooks from the kernel heap can result in page faults due to accesses to data that may have existed or does not exist.

The basic principle of Hooksafe is simple. A write access to the hooks is trapped by the indirection layer and if authenticated, the hypervisor updates the kernel hook on behalf of the guest OS. Read accesses are redirected through code in the guest OS to fetch the kernel hook thus avoid costly switching between the hypervisor and guest OS. The authors analyzed all hooks in the Ubuntu 8.04 OS and found that sometimes pages contains less than 15 hooks. Marking such pages as write protected would cause page faults if the page also contains dynamic data. The authors tested this premise by creating write-protected pages for all hooks which cause 1% page faults.

The Hooksafe architecture contains two main components - an offline hook profiler and an online hook protector. All hooks in a guest OS kernel are collected by the offline hook profiler into Hook Access Points (HAPs). Dynamic analysis of the guest OS is done through an emulator such as QEMU to determine all memory accesses to determine hook access instructions placed in HAPs. HAP contains read/write instructions for the particular hook and its values. The online hook provider then creates a shadow copy and serves as the indirection layer for the HAPs.

The online hook provider proceeds in two steps. First, the guest kernel module or in-guest kernel module is loaded before the OS and allocates memory pages. Kernel hooks are copied to these memory pages following which the in-guest kernel module loads the indirection code for accessing the memory pages. After completion of this operation, the hypervisor is notified of the memory staging address and size following which the hypervisor constructs the HAPs based on the shadow page table. Write accesses are validated by checking whether the new hook value has been listed by the offline profiler as a valid value. Values are updated only by the hypervisor by shifting control temporarily from the guest-OS to the hypervisor. To also include dynamically added hooks, HookSafe considers all memory allocation and de-allocation operations to determine if a kernel hook is being allocated in which case a hypercall is issued to create a shadow copy. Hardware registers are listed and protected from invalid modification by rootkits through the use of secure descriptor tables in virtualized hardware implementations. In addition HookSafe uses the Input Output Memory Management Unit of the Xen hypervisor to limit dynamic memory access.

The system was evaluated against nine Linux kernel rootkits and defended against each. To measure performance, the authors used a Dell Optiplex 740 system with an AMD64 X2 5200+ CPU and 2GB memory. Version 3.30 of the Xen hypervisor was used with a default guest installation of Ubuntu 8.04. The test was conducted using an Apache web server serving web pages of size 8KB. The main overhead is during the gunzip operation of the linux 2.6.18-8 package which is about 6.5%. In the case of the Apache Bench test, the degradation in speed was about 6%.

Hooksafe is a secure hypervisor system developed by IBM research [8]. The main problem the sHype addresses is controlling information flow between virtual machines. While virtual machine monitors isolate virtual machines from resource access, communication between virtual machines through side channels using shared virtual resources is still a possibility. The authors point out that information flow between virtual machines is possible whenever a shared resource such as virtual disk or virtual memory is accessed which is not correctly isolated for each virtual machine. Virtual LANs also can be used a resource for information flow.

sHype has four main policies - non-exclusive access to resources is controlled and monitored through mandatory access control policies, exclusive assignment of resources is guarded and controlled by a security policy, direct information flow between virtual resources should be prevented and information flow between real resources that are mapped to virtual resources should also be prevented [8]. To achieve these goals, sHype uses a reference monitor as part of the hypervisor for enforcing mandatory access control policies. The policy specification and enforcement are separated allowing greater flexibility and control.

In essence, whenever a virtual resource or object is requested, an enforcement hook maps the request to an access control module which retrieves and applies the policy from the security manager. The result is returned to the hook. Based on the result returned the access is granted and controlled or denied.

While the system presents an excellent security solution there are a few caveats. Covert channels are not prevented by sHype and more research in this direction is needed.

5. CONCLUSION

This paper provides a comprehensive overview of the different types of hypervisors. A view of the architecture development of commercial hypervisors is presented through the analysis of the Hyper-V hypervisor from Microsoft. In addition, the security recommendations from Microsoft for Hyper-V are discussed. Two reported security vulnerabilities provide a basic idea of the type of security breaches that need to be handled. Finally, the paper looks at two secure hypervisor techniques in the research literature as a guide to help towards developing more secure hypervisors.

6. REFERENCES


APPENDIX

A. FIGURES

1. Figure 2, http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf/

2. Figure 3, http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf/

3. Figure 4, http://msdn.microsoft.com/en-us/library/cc768520(v=bts.10).aspx/