XEBHRA: A Virtualized Platform for Cross Domain Information Sharing

Extended Abstract

Charles Payne, Jr., and Jason Sonnek and Steven A. Harp
Adventium Enterprises, LLC
Minneapolis, MN USA
{charles.payne, jason.sonnek, steven.harp}@adventiumlabs.com

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Information Flow Controls; J.7 [**Computers in Other Systems**]: Military

General Terms

Security, Design, Measurement

Keywords

information sharing, cross domain solution, virtualization

1. INTRODUCTION

The Unified Cross Domain Management Office (UCDMO) states that its mission is to provide coordination and oversight for the cross domain community's vision of "secur[ing] cross domain access to and sharing of timely and trusted information, creating a seamless Enterprise that enables decision advantage." [14] The UCDMO defines three types of cross domain solution (CDS) — transfer, access and multilevel — to satisfy this vision [15].

The transfer CDS, or *guard*, moves information securely between software applications running in different information security domains. Since the guard must approve all information flows between domains, it is traditionally deployed on a standalone computer host that provides the only physical link between the domains' networks. This deployment strategy ensures that the guard cannot be bypassed. Unfortunately, as the demand for sharing increases, this strategy can prove costly. Data centers, for example, may charge more for custom guard hardware that cannot be reallocated easily for other uses.

To address rising deployment costs, the UCDMO has proposed to implement the guard as a software service that can be installed and managed from a central location [13]. The guard would be available via the network to any cross domain application that requires it. It may even be deployed on a virtual machine (VM) in a data center. Unfortunately, this deployment strategy also has shortcomings. For example, in tactical environments, access to the network — and thus to a remotely hosted guard service — can be unreliable. The tactical user needs to operate through network loss, so this user would benefit from a local guard deployed on a small hardware footprint for a resource-constrained environment. In other situations, a local guard may be required. A network management application, for example, may use a guard to control many networks at different security levels. In this case, network availability depends on the availability of the guard rather than the reverse. In summary, these use cases encourage deploying the guard as close as possible to its cross domain applications.

In this paper, we introduce XEBHRA (Xen-Based, Host-Resident, Assurance), a layered assurance architecture for virtualized cross domain information sharing. XEBHRA hosts the guard and its domain applications as separate VMs on a trustworthy virtual machine monitor (VMM). XEBHRA joins the domain application VMs to the guard VM using virtual networks, and XEBHRA configures those virtual networks so that the guard VM cannot be bypassed.

The XEBHRA architecture enables the user to not only interact with each domain like an access CDS but also to initiate approved transfers between domains using the local guard. XEBHRA achieves a size, weight and power (SWAP) reduction of $(n+1)$-to-1 for n domains and a single guard. XEBHRA eliminates physical network disruption as an impediment to local information sharing and, since the guard is accessed only from the domain application VMs, it also reduces the exposure of the guard VM to attacks from the physical network.

The next section describes challenges that the XEBHRA architecture must address.

2. CHALLENGES

Figure 1 illustrates a conceptual view of XEBHRA. The notional domains HI and LO and the guard G are VMs. XEBHRA can host more than two domains if needed. HI and LO send data over virtual networks N_{HI} and N_{LO}, respectively, that are linked by G. The Trustworthy VMM controls all access by the VMs to the Hardware, and it assigns the network interfaces for the physical domain networks, HInet and LOnet, to HI and LO, respectively. Hardware emulation, where needed, is provided by Privileged VM, also known as `dom0` in Xen. The XEBHRA architecture must provide this functional behavior while addressing two key challenges.

CSIIRW'12 October 30 – November 2, Oak Ridge, Tennesee, USA

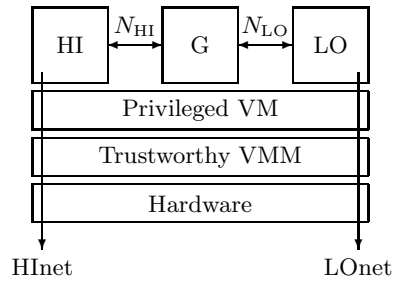

Figure 1: An abstract view of XEBHRA

The first challenge is for XEBHRA to reuse certified components. The guard software running in G undergoes an expensive certification (which can approach six figures) to assure its trustworthiness. Cross domain applications running in HI and LO may undergo their own certifications. If any of those components require modification to install and operate on XEBHRA, then the cost savings from SWAP reductions may be lost in certification costs. A practical solution, then, is for XEBHRA's VMM to export the standard device interfaces expected by — and *already supported by* — those components.

The second challenge is for XEBHRA to achieve reuse with an appropriate level of assurance. There are many ways to engineer the system in Figure 1 to enable the reuse of certified components; however, many of those solutions may not provide the level of assurance needed to accredit XEBHRA for operational use. XEBHRA's assurance requirements are driven by the assurance requirements for the guard software. That software was certified under the assumptions that its own processing and hardware resources are completely separate from either HI or LO and that N_{HI} and N_{LO} are physically separate networks. Clearly a virtualized platform like XEBHRA, where VMs share the processing and hardware resources exported by the Trustworthy VMM, will challenge these assumptions. In order to satisfy the conditions of the guard software's certification, XEBHRA must not only demonstrably isolate VMs G, HI and LO but also the virtual networks N_{HI} and N_{LO} that connect those VMs.

In the next section, we present an architecture for XEBHRA to address these challenges.

3. ARCHITECTURE

Figure 2 illustrates the XEBHRA's architecture using layered assurance. Our assurance strategy for XEBHRA is to certify and compose the lower layers so we can satisfy the certification assumptions of the higher layers. We describe each layer briefly below.

- The bottom layer is a certified, Xen-based, Type 1 VMM that controls the hardware and isolates each VM's use of that hardware. The VMM and its Privileged VM must be certified to provide VM isolation sufficient for an access CDS [15].
- At the middle layer, we introduce *bridge VMs* to create communication channels between `High` and `Guard` and between `Low` and `Guard`. Each bridge VM (labelled `High Bridge` and `Low Bridge` in Figure 2) implements a complete virtual network. Because each virtual network is implemented entirely within its bridge VM, and because the VMM guarantees that each VM is isolated from all other VMs (except for interactions authorized by the VMM), then each virtual network is isolated from other virtual networks. This layer composes with the bottom layer to assure that `Guard` is nonbypassable.
- At the top layer, we deploy certified guard software in `Guard`. This layer composes with the lower layers to assure that all information flows between `High` and `Low` are approved by the guard software.

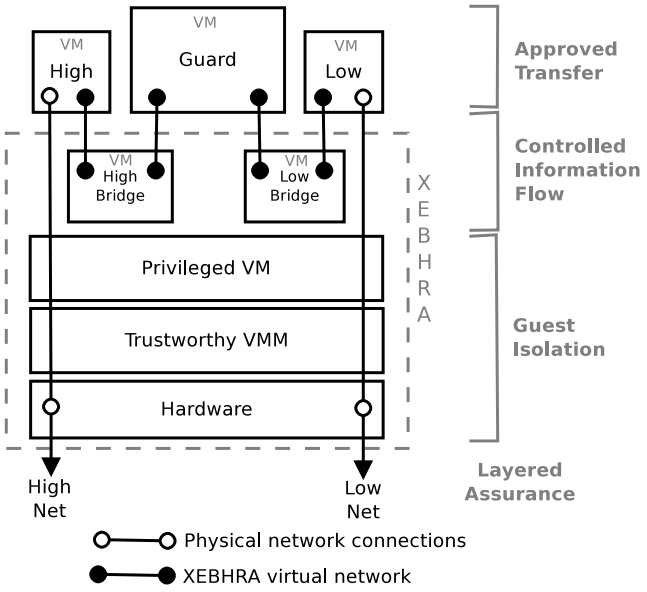

Figure 2: Layered Assurance in XEBHRA

In the following sections, we describe each layer in more detail.

3.1 Layer 1: Trustworthy VMM to isolate VMs

The guard's certification requires that `High` and `Low` be isolated, both in their processing and in their connections to physical networks. While the open source Xen VMM provides some VM isolation, the complexity and structure of its code base inhibits convincing analysis for medium-to-high assurance applications. In addition, documented attacks expose the limits of that protection [17, 10]. The VMM and its Privileged VM may need to be hardened for higher assurance, e.g., using technologies such as XSM [3], sHype (now ACM) [11] or SELinux. In addition, a VMM trusted to operate across multiple levels of security should provide certain features, such as a secure shared file store [5]. Therefore, a more trustworthy VMM is required for XEBHRA's foundation.

McDermott[7] defines a *separation hypervisor* as a Type 1 VMM that offers assurances similar to a separation kernel but that also includes hardware emulation for its VMs. The separation hypervisor includes a privileged VM to perform hardware emulation for all other (non-privileged) VMs. The

privileged VM, which is included in the certification of the separation hypervisor, must be hardened to prevent unauthorized information flows within it.

It is not our intent to construct a separation hypervisor for XEBHRA but to build on existing work. Candidates include the NRL Xenon hypervisor [9, 6, 8], which is still in development, and the commercially available Citrix XenClient XT [2]. Announced in May 2011, XenClient XT leverages NSA's Xen Security Modules (XSM) [3] and SELinux, Intel's Virtualization Technology for Directed I/O (VT-d), Extended Page Tables and Trust Platform Module (TPM)/Trusted Execution Technologies (TXT) to improve VM isolation and to provide a root of trust. The AFRL SecureView workstation, which is based on XenClient XT, has been certified for multi-level operation [4] and deployed by the Intelligence Community.

3.2 Layer 2: Bridge VMs to control information flow

As we consider implementation alternatives for XEBHRA's virtual networks, we must assess the level of effort that will be required to certify that the high and low virtual networks are isolated and that `Guard` cannot be bypassed. For example, emulating both virtual networks within the Privileged VM will require demonstrating that the Privileged VM adequately separates the processing associated with each network. Since the Privileged VM is typically a full operating system, a convincing demonstration may be intractable.

To minimize trust in the Privileged VM, we introduce the *bridge VM*. A bridge VM implements a dedicated communication channel between a domain VM and the guard VM. It removes all virtual network emulation from the Privileged VM while providing the standard networking interface that the domain VMs and the guard VM support.

The bridge VM is based on the concept of a Xen device driver VM. Like a device driver VM, the bridge VM supports Xen's split driver model. The network driver's frontend is implemented in the domain (or guard) VM and the network driver's backend is implemented in the bridge VM. Unlike a device driver VM, a bridge VM does not join the backend to a physical network device. Instead, the bridge VM joins the backend to a virtual bridge that is implemented entirely within that bridge VM.

To use bridge VMs, `High` and `Low` must connect to their virtual networks using paravirtualized network drivers (illustrated as closed circles in Figure 2). Fortunately, even commercial operating systems that must run as fully virtualized VMs, or hardware virtual machines (HVMs), often include support for paravirtualized network drivers [18, 12, 16].

Figure 3 illustrates the use of a bridge VM to create a communication channel between `High` and `Guard`:

- The virtual frontend interface in `High` is connected to a corresponding backend interface in `High Bridge` via a ring buffer implemented on top of shared memory [1].
- `High` places data destined for `Guard` in the buffer, where it is picked up by `High Bridge`.
- `High Bridge` copies the data to its private memory, and the backend interface forwards the data on the virtual bridge hosted in `High Bridge`, where it is seen by the backend interface for `Guard`.

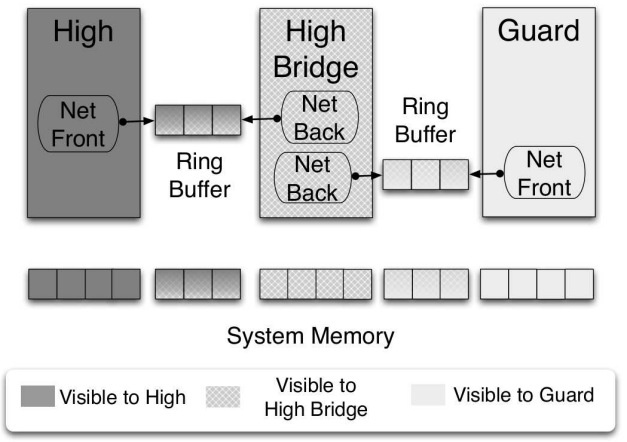

Figure 3: Communication using a XEBHRA Bridge VM

- A similar process transfers the data from `Guard`'s backend interface to `Guard`'s frontend interface.

Using bridge VMs, XEBHRA achieves virtual network isolation because the data is always either in a VM's private memory or in memory that has been explicitly shared between a domain (or guard) VM and its associated bridge VM.

Figure 4 illustrates the use of bridge VMs to enable approved information flow from `High` through `Guard` to `Low`. Each domain VM, `High` and `Low`, requires access to both a physical network interface (labelled "Net" in Figure 4) for accessing its domain network, and a paravirtualized network interface (labelled "Net Front") for accessing `Guard`. `Guard` requires only two paravirtualized network interfaces. Rather than relying on the Privileged VM to emulate the physical network ("Net") interfaces, XEBHRA leverages the chipset's Input/Output Memory Management Unit (IOMMU) support (illustrated as open circles in Figure 2) to safely pass control of those physical interfaces directly to the domain VMs.

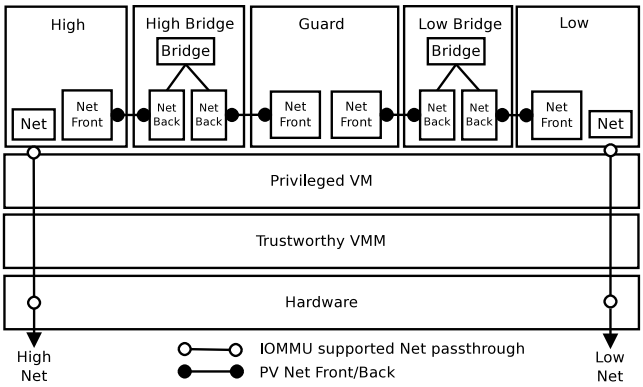

Figure 4: Bridged information flow in XEBHRA

3.3 Layer 3: Certified Guard to approve information transfer

The final layer of our assurance architecture introduces the certified guard software to `Guard` and suitable cross domain applications to `High` and `Low`. Through the lower layers of its security architecture, XEBHRA validates the guard's certification assumption of physically separate networks. XEBHRA can be viewed as a new hardware platform on which to deploy certified guards and their cross domain applications. Like any new platform for the guard software, some certification may still be required, but the effort hopefully will be much less than starting from scratch.

4. STATUS AND NEXT STEPS

We developed a prototype of XEBHRA on Xen 4.0 and evaluated its performance. Our results show Xen's intra-machine virtual network performance to equivalent to a 1 Gbps Ethernet link. While an inter-VM shared-memory transport, such XenLoop or XenSocket [19], could improve on that performance, those solutions could also require modifications to the guard software, which could require recertification. We also installed a certified guard on this prototype, and it installed easily and without alteration.

Our next step will be to pursue implementation of XEBHRA on a trustworthy VMM, because the protections claimed by XEBHRA can only be realized by building on a VMM that provides the necessary isolation guarantees. Our initial analysis of Citrix XenClient XT confirms that it will easily support XEBHRA's bridge VM concept.

Although XEBHRA was designed to address cross domain information sharing requirements, the XEBHRA architecture can also be applied in a straightforward manner to other environments with similar information flow requirements. For example, the Guard could be replaced with filtering software to detect malware moving from less sensitive to more sensitive VMs. In this case, XEBHRA would guarantee that the filters cannot be bypassed.

5. ACKNOWLEDGMENTS

The authors wish to thank the Office of Naval Research and the Naval Research Laboratory for their support of this work.

6. REFERENCES

[1] D. Chisnall. *The Definitive Guide to the Xen Hypervisor*. Prentice Hall, 2007.

[2] Citrix. Citrix announces XenClient2 and XenClient XT. `http://www.citrix.com/English/NE/news/news.asp?newsID=2311981`.

[3] G. Coker. Xen Security Modules, July 2009. Available at `xen.xensource.com/files/summit_3/coker-xsm-summit-090706.pdf`.

[4] R. Durante. Secureview and XenClient XT security. Presentation at NSA Trusted Computing Conference and Exposition, September 2011.

[5] P. A. Karger. Multi-level security requirements for hypervisors. In *Annual Computer Security Applications Conference (ACSAC)*. IEEE Computer Society Press, December 2005.

[6] J. McDermott. Xenon: High-assurance Xen, 2007. `http://www.xen.org/files/xensummit_4/XenSummitSpring07_McDermott.pdf`.

[7] J. McDermott and L. Freitas. A formal security policy for Xenon. In *Workshop on Formal Methods in Software Engineering (FMSE)*. Association for Computing Machinery, October 2008. Held in conjunction with the ACM Conference on Computer and Communications Security.

[8] J. McDermott, B. Montrose, and M. Kang. Separation virtual machine monitors. In *Annual Computer Security Applications Conference (ACSAC)*, Orlando, FL, December 2012. ACM Press. To Appear.

[9] J. P. McDermott and M. Kang. An open source high robustness VMM. In *22nd Annual Computer Security Applications Conference (ACSAC '06)*. IEEE, December 2006.

[10] J. Rutkowska. Owning xen in vegas! blog entry, July 2008. `http://theinvisiblethings.blogspot.com/2008/07/0wning-xen-in-vegas.html` Slides available at `http://invisiblethingslab.com/bh08/`.

[11] R. Sailer, T. Jaeger, E. Valdez, R. Cáceres, R. Perez, S. Berger, J. L. Griffin, and L. van Doorn. Building a MAC-based security architecture for the Xen opensource hypervisor. In *Annual Computer Security Applications Conference (ACSAC)*. IEEE Computer Society Press, December 2005.

[12] H. Su. Installing Solaris 10 virtual machine with Oracle VM manager. Available at `http://blogs.oracle.com/virtualization/2010/02/installing_solaris_10_virtual.html`.

[13] Unified Cross Domain Management Office. Cross domain community roadmap, June 2008. Available at `http://www.ucdmo.gov/CDCommunityRoadmapExecOverviewv7.pdf`.

[14] Unified Cross Domain Management Office. About the UCDMO, June 2011. Available at `http://www.ucdmo.gov/about.html`.

[15] Unified Cross Domain Management Office. Baseline version 3.9.0, July 2011. Available at `http://www.ucdmo.gov/Baseline.v.3.9.0.pdf`.

[16] J. Williams. Xen Windows GPLPV drivers. Available at `http://wherethebitsroam.com/content/gplpv`.

[17] R. Wojtczuk and J. Rutkowska. Following the white rabbit: Software attacks against Intel VT-d, May 2011. `http://www.invisiblethingslab.com/resources/2011/Software\%20Attacks\%20on\%20Intel\%20VT-d.pdf`.

[18] Xen.org. Xen PV-on-HVM drivers for Linux HVM guests. Available at `http://wiki.xensource.com/xenwiki/XenLinuxPVonHVMdrivers`.

[19] X. Zhang, S. McIntosh, P. Rohatgi, and J. Griffin. XenSocket: a high-throughput interdomain transport for virtual machines. In *Proc. of the ACM/IFIP/USENIX 2007 International Conference on Middleware*, pages 184–203, 2007.