

As a security researcher, my passion and goal has always been to enable effective security that does not impede the intended uses of a computer. To this end, my research style combines proven classic security design techniques with modern ideas that advance the state-of-the-art for security in today's computing-centric world. I strongly believe that security ideas must be implemented and evaluated to fully understand their utility. Furthermore, security must be carefully integrated into each level of a system from the hardware up to the user interface. My long-term research goals involve working at each of these levels to create unobtrusive secure systems.

Throughout the past several years, I have been laying the foundation for my long-term research goals by designing a system security architecture that enables secure host-based monitoring. A key functionality of this architecture is the ability to perform **secure active monitoring**, which involves sending uncircumventable notifications of software events (e.g., system calls, critical memory accesses, etc.) to a protected security application. We published this important technique at the IEEE Symposium on Security and Privacy, the top security conference [4]. In a subsequent paper showing how this technique can be used to prevent a new class of kernel control flow attacks [8], we received the Outstanding Paper Award. I strive to keep my work both practical and relevant by maintaining close interactions with companies deploying technologies related to my work. Leveraging one such relationship, I secured a grant from IBM that funded a portion of my doctoral research.

CURRENT WORK: SYSTEM SECURITY FOUNDATIONS

The goals of system security research have fundamentally changed over the past thirty years. Previously, researchers worked to build secure operating systems from the ground up. Today, we generally assume that existing commodity operating systems (e.g., Windows or Linux) are too entrenched to be replaced. Therefore, the modern system security problem revolves around providing effective security to these innately insecure environments. My approach is to insert a small, trusted layer of software (e.g., a hypervisor) between the commodity operating system and the hardware. This layer enables the creation of an isolated execution environment, along with the complete mediation of all hardware events.

Virtualization is a natural fit for implementing this software architecture. In this setting, the hypervisor performs hardware mediation and virtual machines (VMs) are used as isolated execution environments. For desktop systems, we run the security software in one VM, called the Security VM, and the user's commodity operating system in another VM, called the User VM. Figure 1 shows an overview of this architecture. More complex environments, such as server deployments, may have multiple User VMs. In either case, we rely on the hypervisor to provide strong isolation between the VMs. After deciding on a virtualization-based architecture, **my research focused on the foundational capabilities necessary to support secure monitoring in this environment.**

In my first summer as a doctoral student, I worked at the IBM T. J. Watson Research Center on improving the mandatory access control (MAC) capabilities in Xen, an open

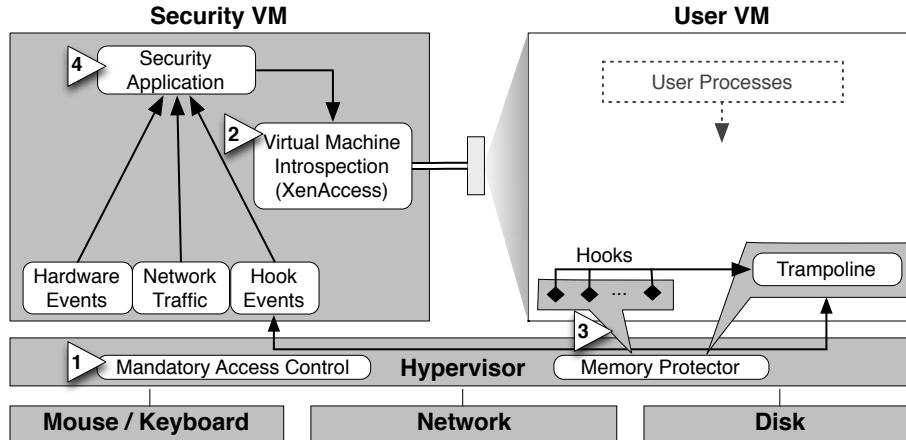


Figure 1: An overview of the system architecture used in my research. My primary contributions are highlighted by number: (1) layered mandatory access control, (2) virtual machine introspection, (3) secure active monitoring, and (4) security applications that leverage the architecture.

source hypervisor. In Xen, MAC can be used, along with a well-defined security policy, to enforce isolation between the VMs. I implemented **software to enforce MAC policies for disks attached to VMs** that is available today through the Xen project [1]. We also developed **ideas for layering MAC policies between the hypervisor and operating systems** in the VMs to reduce complexity while enforcing both complete isolation and information flow policies between VMs [7]. Using MAC within the hypervisor provides the inter-VM isolation needed for our architecture, but it also makes it difficult for any software in the Security VM to monitor the User VMs.

Virtual machine introspection (VMI), a technique for externally monitoring the runtime state of a VM, is the way to enable monitoring from the Security VM. However, deploying VMI securely is non-trivial because it requires carefully breaking the inter-VM isolation to enable the security monitoring without creating an attack vector for malicious software. Recognizing that VMI would become a fundamental building block for security applications, we researched **techniques to ensure secure deployment of VMI monitoring applications** [3]. I also designed and implemented an **open source VMI software library** [2] that has subsequently been used to enable research and production projects at universities, industrial labs, and government labs worldwide.

Traditionally, VMI has been a passive monitoring technique because it provides a mechanism to view what is happening within the User VM, but it cannot tell you when a specific event occurs. Active monitoring complements VMI by providing event notifications when specified code is executed or memory regions are modified. Active monitoring typically involves placing hooks within the code being monitored; in this case within the User VM. This leaves the hooks vulnerable to circumvention or removal by malicious software. We addressed this problem by extending VMI with **secure active monitoring** that allows for placement of protected, kernel-level hooks within the User VM [4].

Using our secure monitoring architecture we designed, implemented, and evaluated two new security applications. The first application **stops a new class of attack that we identified and named soft-timer driven transient kernel control flow attacks**.

This class of attack utilizes the soft timer mechanism in an operating system kernel to schedule its own execution, which is hard to detect because the code normally scheduled through this mechanism is dynamic and does not conform to a predefined signature. Using VMI and our secure active monitoring capability, we initiated a security check before each soft-timer execution that verified all possible code execution against information obtained through static analysis of the kernel source code [8].

Our second application **stops malicious software designed to send network traffic that mimics normal human activity** [5]. Using our monitoring architecture, we securely monitor all keyboard and mouse input events in route to the User VM. Next, we use VMI to view the User VM's operating system and application states and determine if the user input event will trigger any outgoing network traffic. Finally, we monitor outgoing network traffic to stop any traffic that was not intended by the user. This technique, when combined with existing network filtering tools such as application firewalls, provides a powerful defense against malicious software that use hosts to send spam or execute click fraud attacks. Both of these applications stop important classes of attacks, and neither could be implemented securely without our monitoring architecture.

FUTURE WORK: TRUSTED, RELIABLE, & FLEXIBLE SECURITY APPLICATIONS

My doctoral research focused on designing and building an architecture that enables secure passive and active monitoring. This systems building work involved significant software engineering to fully demonstrate and evaluate each portion of the architecture. Looking ahead, I intend to leverage this architecture as a foundation to support a variety of security research ideas ranging from low-level hardware interactions up to the user interface. I provide an overview below of three research directions that I plan to pursue, each benefiting from my current work.

My primary research direction will be to secure the hypervisor and the Security VM. Our existing security architecture provides security, in part, by communicating directly with the hardware. If an attacker can insert a software layer between the hypervisor and the hardware, then we lose some important security properties such as the ability to trust the source of hardware-level events. More generally, the **security must be rooted in the hardware**. Leveraging existing hardware security mechanisms such as the trusted platform module (TPM) and late launch processor instructions, I plan to explore techniques for both detecting and mitigating runtime security breaches in the hypervisor and the Security VM. In addition, I plan to research and develop a small, security-focused hypervisor that is sufficient to support our existing security architecture and be formally verified for correctness. Formal security verification has only recently reached sufficient maturity to verify software of this scale, resulting in exciting possibilities for creating a secure foundation for our architecture.

My second research direction will be to give applications in the Security VM greater semantic understanding of the software running in the User VM. While our monitoring techniques benefit from strong isolation from the User VM, this isolation means that we lose access to the software interfaces typically used to extract information about the running system. Instead, our monitors must **rebuild semantically meaningful information** from a physical view of system memory. Drawing from a variety of techniques such as forensic memory analysis, software analysis, automated reverse engineering, and

code injection, I plan to develop general techniques for extracting both application and kernel level information from the User VM. While current research has focused on related techniques for a single software version, my more general techniques would be designed to work across a wide variety of software versions. This information can then be made available to monitoring applications such that each monitor can be built once to work effectively across a large variety of software in the User VM.

My third research direction will be to design new security applications to work with our architecture. My goal here will be to consider how existing security applications can be used together, and then identify the remaining gaps as potential research directions. These gaps may include unaddressed security problems, or areas where existing tools do not work well together (e.g., due to performance, required architecture, etc). In addition, building on my previous experience with usable security [6], I plan to explore techniques to allow these security applications to securely display information to and receive input from the user. This critical capability is necessary for any desktop security system.

Each of these research directions will advance my current work toward a system capable of providing robust security for laptop, desktop, and server computing platforms. This is an ambitious, long-term goal that will bring a level of security to commodity operating systems that was previously thought to be impossible. However, my passion for security does not end with this specific goal. I envision working collaboratively on security-related research projects ranging from embedded systems to network security and look forward to exploring new research ideas as computers continue their push into ubiquity.

REFERENCES

- [1] Citrix Systems Incorporated. Xen hypervisor. <http://xen.org/>.
- [2] **Bryan D. Payne**. XenAccess: A virtual machine introspection library for Xen. <http://xenaccess.org/>.
- [3] **Bryan D. Payne**, Martim Carbone, and Wenke Lee. Secure and flexible monitoring of virtual machines. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, December 2007.
- [4] **Bryan D. Payne**, Martim Carbone, Monirul Sharif, and Wenke Lee. Lares: An architecture for secure active monitoring using virtualization. In *Proceedings of the IEEE Symposium on Security and Privacy (Oakland)*, May 2008.
- [5] **Bryan D. Payne**, Brendan Dolan-Gavitt, and Wenke Lee. Gyrus: A host-based security framework for distinguishing user-intended and malicious network traffic. In *Submission*, 2009.
- [6] **Bryan D. Payne** and W. Keith Edwards. A brief introduction to usable security. *IEEE Internet Computing*, 12(3):30 – 38, May 2008.
- [7] **Bryan D. Payne**, Reiner Sailer, Ramon Caceres, Ronald Perez, and Wenke Lee. A layered approach to simplified access control in virtualized systems. *ACM SIGOPS Operating Systems Review*, 41(4):12 – 19, July 2007.
- [8] Jinpeng Wei, **Bryan D. Payne**, Jonathon Giffin, and Calton Pu. Soft-timer driven transient kernel control flow attacks and defense, **Outstanding Paper Award**. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, December 2008.