2010 Second Vaagdevi International Conference on Information Technology for Real World Problems

Configurable and Secure System Architectures

Dipankar Dasgupta, Ph.D.
Director, Center for Information Assurance
Professor of Computer Science
The University of Memphis
Memphis, TN. USA
dasgupta@memphis.edu

Marco Carvalho, Ph.D.
Research Scientist
Florida Institute for Human and Machine Cognition
5 SE Osceola Ave.
Ocala, FL. USA
mcarvalho@ihmc.us

Abstract— **In this paper we introduce a novel approach for host protection based on a security subsystem for continuous monitoring and control of user applications. To enable a secure monitoring capability, a strict hardware separation is proposed, in combination with a signaling layer for monitoring and control. The paper presents our preliminary work, introducing some of the core ideas and discussing a few scenarios and applications.**

Keywords – resource partitioning, message passing, muticore system, security subsystem, self monitoring.

I. INTRODUCTION

Computing systems are increasingly under sophisticated attacks which are difficult to prevent. Recognizing that most computational systems are likely to be compromised at some point, there is a growing tendency to building systems and architectures that are resilient to attacks and able to gracefully degrade, or recover. As the software only security solutions are not adequate, hardware manufacturers are also extending processor capabilities to provide secure execution environment [15]. The important task is to build secure and resilient mission critical systems that are capable to maintain mission execution while isolating, recovering from compromised components [17].

In this paper, we introduce a proposal for a resilient architecture that can provide isolated execution environment primarily for monitoring and control the system and its applications, along with distributed coordination strategies for mission management and immunization. Our approach is inspired by the way in which biological systems survive in response to environmental changes, adversaries and peer pressures. In particular, our approach relies on systems being able to, at the local level, detect and identify potential attacks in order to properly respond and survive [13, 14].

The detection and identification of attacks, however, is not a simple task, and for most practical cases, it requires a monitoring component capable to observe the operational system and infer anomalies, attacks or compromises. Existing works (summarized in section IV) have adopted different approaches for security monitoring, but most rely on software-based solutions such as secure virtualization, fixed hardware configuration, and often requiring additional hardware integration.

Virtualization-based computing platforms provide efficient mechanisms for resource sharing, monitoring and control of operational systems. However, by design, they rely on the host operating system (or equivalent software layer) for support, which could in itself be compromised, defeating the purpose of hypervisor-based monitoring [5, 6].

Other approaches have relies on external monitoring, using Trusted Computing Hardware components, or additional hardware components (crypto-processors, embedded chips, USB-keys, and others) for security. While more effective than software based solutions, these approaches require a fixed allocation of resources for security monitoring and cannot adapt to the demands or resource availability of the system.

There is a need for configurable monitoring capabilities that can provide the necessary hardware separation for system to application-level monitoring and control. In this paper, we will introduce a new concept for a self-monitoring security system that is configurable and adaptive to system resources and security requirements.

In virtualization, resources are logically partitioned to have multiple operating environments and applications, where host OS usually manage and control such resource partitioning and sharing. However, if the host OS gets compromised, integrity and security of guest OSs are hard to maintain and guarantee. Monolithic nature of guest VMs make them easy target for attack and spread infections.

II. AN ARCHITECTURE FOR SYSTEM RESILIENCE

The proposed architecture incorporates a self-monitoring mechanism in a multi-core system such that the "health" of the entire system can be monitored securely and in an implicit manner. Accordingly, a multi-core system will be vertically partitioned (from hardware to application level) so that a small set of processors (security core) be separated from other user-data processing cores to build a security subsystem. Then some security handling tasks are delegated to this isolated subsystem, which uses a set of privileged instructions to monitor the other part (user side) and interact using a secure message exchange protocol.

Figure 1 illustrates this conceptual architecture, a multi-core (a quart core processor) system is logically partitioned to build two separate unequal systems (with no shared resources) where the smaller partition (having a single core, small RAM and

Disk spaces as shown in the right side oval) works as security subsystem to monitor the user-side system (as shown in the left side oval). The user side OS sees and schedules 3 cores and manages its partition of the memory space. Implicit monitoring in this case can be defined as the ability to partition system resources efficiently and effectively such that one partition can monitor the other for enhancing the system's overall security.

While still a topic for research, we have preliminarily identified some types of information that would be exchanged between the monitored user environment and the security subsystem. Table 1 illustrates the message types providing information from different levels to the security subsystem (SS) for analysis and receiving corrective or preventive measures. However, further research needs to be conducted to determine what to be monitored and specific security tools and techniques to be used. Since the security subsystem is under the different operating environment, a compromise on the user operating system should not affect the monitoring capabilities of the security subsystem and thus help improve the overall system security and resiliency.

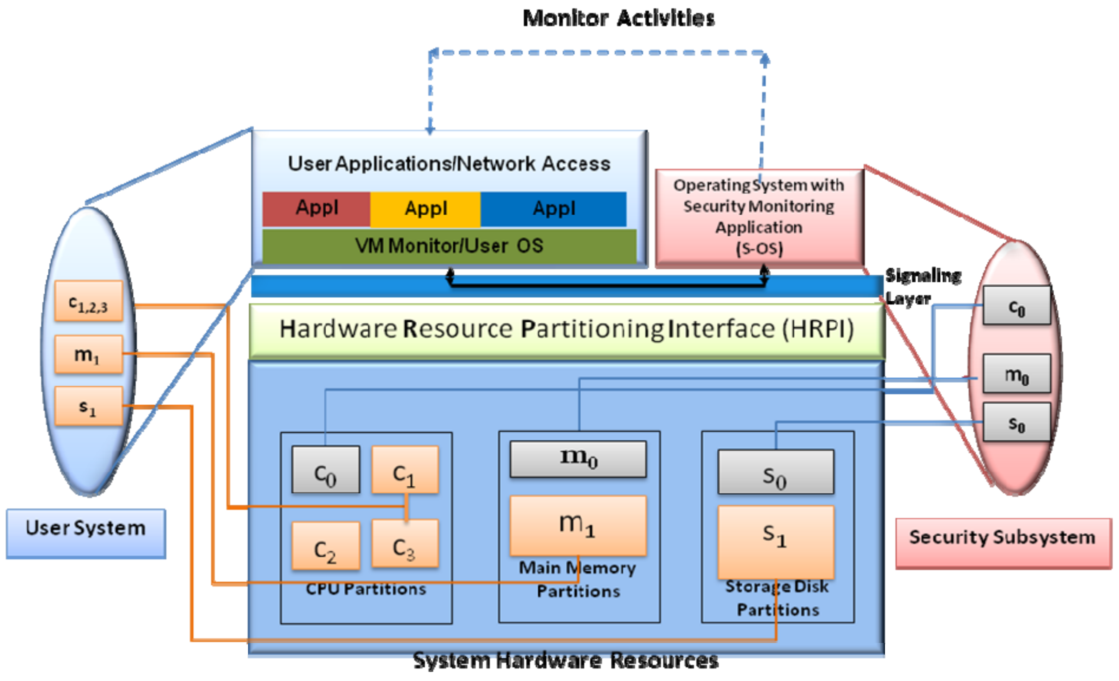

Figure 1. A Configurable Resilient System Architecture

Accordingly, a Hardware Resource Partitioning Interface (HRPI) will be designed and implemented which can create two partitions of system hardware resources; one is the Security Subsystem (SS) with unique ability to monitor the user-side operating environment (other partition).

As illustrated in Figure1, two operating systems run independently above HRPI layer; each using non-sharable hardware/software resources. Importantly, the security subsystem should use a different OS with a minimal privileged instruction set and independent of the user-side OS. In general, user-side system is where virtualization and networking will take place and will run user applications.

In principle, the HRPI should be designed and implemented as a thin firmware layer whose purpose is to partition hardware resources and is independent of OSs that are installed on two partitions while providing services to upper layers. It also should provide mechanism so that the security subsystem can uniquely monitor the user system while not being detected by the same. This ability of the security subsystem is termed as one-sided monitoring where asynchronous signaling and message-passing between two-parts occur through a secure signaling layer on HRPI.

The proposed architecture enables the creation of a secure domain that can closely monitor the execution of applications and operating system behavior of the user domain. From a digital immunity perspective, the architecture enables the realization of the following capabilities:

Innate Defense: Innate (protective) defense techniques can be implemented in the SS to monitor, classify and respond to known threats and attacks. The proposed space separation at all levels minimizes the chances for compromise and damage to the isolated security domain.

Process Recovery: In addition to detection and response, the SS is also capable to recover compromised processes by re-instantiating recent check-pointed states of compromised VMs in the user domain. This capability allows the SS to temporarily maintain system functionally from a mission perspective while

TABLE I. TYPES OF INTER-PARTITION MESSAGING

Type	Description	Remarks
Application-level Behavioral Information	Observed behavior of the monitored application/user.	Generated periodically and send to the SS.
System-level Anomaly detection	Statistical measures on resource utilization such as memory, cpu, etc.	Messages may be piggy-backed and exchanged when needed.
Check for Specific Attacks	Compared with known attacks and virus signatures	According to the scheduling
Low-level Security Check	Analysis of data and control flow at the process and thread level	Priority signaling to the security subsystem.

addressing issues of the user domain. Process recovery is one of the key capabilities to enable mission survivability.

Adaptive Defense: Adaptive defense (as a part of self healing) can also be supported through the security subsystem, where it can sense and learn from events on the user domain either though observation or proactive probing in order to adapt and survive future attacks.

System and Mission Survivability: Another key capability enabled by the proposed architecture is system and mission resilience. Learning algorithms (active or passive) which will be used for process recovery and adaptive defense can also provide predictive mechanisms for graceful mission degradation.

III. APPLICATION SCENARIOS

In this section, we present some illustrative scenarios of the capabilities provided by the proposed security subsystem.

A. *Recovering from a Compromised Environment*

An application scenario for the proposed infrastructure considers the case illustrate in Figure 2. In this scenario, an application executing at the user-side system is monitored implicitly by the SS through HRPI signal messages. Resources and system calls triggered by the application (or application supporting services) are monitored, captured and reported to the security subsystem.

In the event of a failure or compromise of a user application and/or the operating system, the SS will detect or will be notified of the event and will take the necessary corrective actions. However, further research needs to be conducted to determine where to put the sensors effectively.

An attack or compromise detection in this example could be based on different measurement and evidence provided by the virtualization layer (which could also be compromised), network traffic monitors (IDS), and anomalous application behavior based, for example, on related operating system call patterns or performance degradation. Once detected, the SS

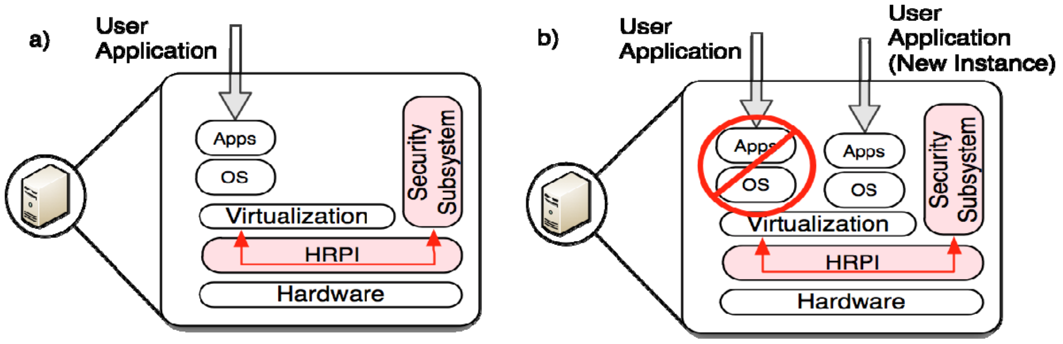

Figure 2. An Illustrative Example: The Secure Subsystem Recreating a Compromised Environment

could kill the compromised VM and recreate the environment from a read-only copy, re-instantiating a copy of the application to bring it back to a known clean state (2(b)). In this example, the state of the application may be lost, but the service remains functional, and restored to a prior clean state.

In this context, the proposed approach focuses on application resilience, favoring recovery in lieu of protection against future attacks. This capability, however, as described later, can be combined with distributed coordination from multiple nodes to enable adaptive diversity and the potential immunization of the mission-critical system.

B. *Local Response to Denial or Disruption of Service*

The detection of a damaged or compromised environment could also be based on functionality and QoS requirements. For example, service degradation due to denial of service attack could be detected and mitigated by the secure subsystem.

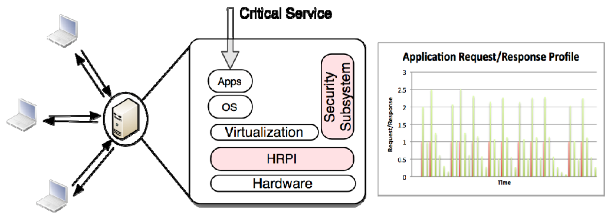

Figure 3. Monitoring and Profiling of Critical Application

Let us consider the scenario of a node providing a time-sensitive critical service to other nodes. The critical service shown in Figure 3 is instantiated as part of a virtual machine that is fully monitored by the security subsystem. Network requests and responses to the applications as well operating system behavior associated with network traffic are profiled by the security subsystem, which also monitors for known signatures of requests.

As illustrated in Figure 3, a response pattern is generated by the application for each client request. The security subsystem will use these learned patterns, as well as explicit QoS requirements (not shown in the figure) to infer a potential anomaly to the system

In the event of a failure or compromise of the critical application, the security subsystem will receive multiple queues associated with the disruption: *i*) possible signaling from the virtualization monitor, *ii*) variations in the traffic pattern due to application failure or degradation (Figure 4), *iii*) changes in system call patterns reported by the VM Monitor due to service failure. Failure or disruption of critical application detected by the security subsystem, and triggering a response for service continuity

When disruption is detected (Figure 4), the SS will initiate the steps to ensure mission continuity. At the network level, the information inferred by the secure subsystem is shared with peer nodes (i.e. the SS on peer nodes) to allow for a proactive preparation of other functionally similar nodes.

At the local level, the SS will immediately instantiate a secondary replacement service on a second virtual machine. New VM configurations may include different operating systems or application settings while maintaining the service functionality.

In the example illustrated in Figure 4, the secondary service is created with small variations in from the original service, possibly in the choice of operating system or configuration. The goal is to immediately restore functionality while trying to mitigate a similar attack to the recently created copies of the server.

In parallel to the new VM instantiation, the system will also monitor the utilization of the new service to verify if the attack (or failure) has been mitigated. If the new service failures under similar conditions, the process is continued based on information gained from each attack to develop safe configuration.

Alternative defense strategies may also be adopted by the secure subsystem including the temporary blocking of traffic from clients directly correlated with the system failures, or the delegation of service provisioning to other peer-nodes, through negotiations between secure subsystems.

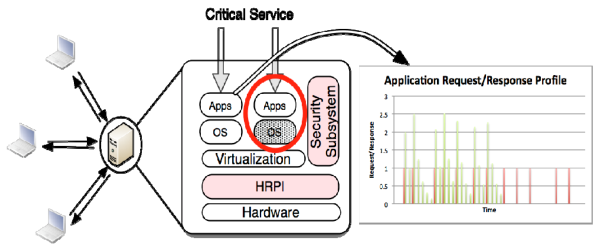

Figure 4. Damage Detection Triggering Re-instantiation of the Critical Application

Other types of attacks such as data exfiltration or corruption will also create variations of the behavioral pattern of the OS, which may be more subtle and difficult to detect and identify. For our purposes, however, we assume that the combination of inference queues is likely to provide a relatively accurate aggregate estimate of the anomaly.

IV. RELATED WORKS

Multi-core systems have been used in many ways including concurrency to exploit parallelism [3] for performance gains, secure execution, etc. For example, Green Hills Software's INTEGRITY Multivisor [10] is a multi-core hypervisor, a version of INTEGRITY Secure Virtualization (ISV) solutions, allows safe execution of trusted real-time critical software in parallel with untrusted applications enabling virtual machines and applications to safely co-exist with guaranteed memory, CPU time resources, and strictly enforced device access control.

The Secure Execution Architecture (SEA) [2] was developed to run the security-sensitive code of an application (called Piece of Application Logic, PAL) in isolation from other software concurrently. The hardware resource isolation is achieved using AMD's Secure Virtual Machine (SVM) and Intel's Trusted Execution Technology (TXT), while Trusted Platform Module (TPM) was added for SEA's TCB to provide sealed storage and attestations of PALs while running in multiple CPUs.

IBM developed hypervisor security architecture (sHype) [1] to govern the control and mediated sharing of resources among virtual machines. sHype extends the existing resource-level isolation hypervisors by implementing mandatory access control on VMs [6].

The configurable isolation architecture [4] proposed a method of configurable partitioning multi-core processors for fault containment and to support redundancy. The primary focus of this work is to contain hardware faults by isolating faulty hardware components through dynamic reconfiguration in order to perform load balancing with gradual degradation of performance. Though configurable isolation approach exhibits how to partition hardware resources to address hardware fault problems, this work did not consider security issues involving OS, applications and user interactions.

Garfinkel and Rosenblum [11] developed an introspection (VMM or VMI) technique for inspecting virtual machine environment from the outside providing strong isolation of IDS from the monitoring host. They built a prototype of MMI-based IDS, called Livewire and tested the same on different attack scenarios. While this approach uses separate VM for monitoring, (but the use of same underlying shared hardware resources for instantiating multiple) VMs may take over the host by exploiting vulnerabilities in the hypervisor. Recently, a lightweight MAVMM [8] architecture is proposed for malware analysis by monitoring guest VMs.

Virtual machines running on the same physical hardware could use covert channels to transfer sensitive information, resources like virtual shared discs and virtual shared networks (features provided by majority of hypervisors) can also be used to share information explicitly by malicious users. This is mainly because hardware resources are not as often monitored as network traffic; therefore low level attacks are possible. A hypervisor that maintains and runs a number of virtual machines (VM) becomes a major target for attackers. Therefore it is in the best interest of the concerned to keep the hypervisor itself as secure as possible. Recent reports indicate that Bluepill and Vitriol hacking tools try to install a malicious hypervisor by adding stealth backdoor functionality in legal hypervisor that is already present. Some vulnerability has been reported in Xen[12], which can be exploited by malicious, local users bypassing certain security restrictions or gain escalated privileges. Also flaws in VMware software that allow malicious code to run in a virtual machine can take over the host operating system.

In another research, a protected zone was created inside a reconfigurable and extensible embedded RISC processor for secure and isolated execution of cryptographic algorithms [15]. The protected zone was implemented with some processor subsystems such as functional units optimized for high-speed execution of integer operations, a small amount of local memory, and general- and special-purpose registers.

The NoHype architecture [16] extended currently available multiprocessor and I/O device features to realize the virtualization layer: arbitrating access to CPU, memory, and I/O devices, acting as a network device (e.g., Ethernet switch), and managing the starting and stopping of guest virtual machines. However, this approach ignores the advantages of virtualization and multi-core processing power by strict partitioning of cores to attach individual application.

Kaspersky Labs [9] reported their patented technology, which comprises of a hardware-based antivirus system to combat malicious programs. This hardware device is installed between the system hard drive and the computing unit, i.e. the system's CPU and the main memory and is connected to the system bus or integrated into the disk controller. They claim that the device can therefore effectively combat malicious programs like rootkits that elevate their privileges in the system. We believe that our proposed approach is along similar lines but more generic and flexible to secure mission critical systems with high reliability, efficiency and self dependency.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have introduced a novel approach for the design of mission critical systems on multi-core computational platforms. Our approach relies on a hardware separation layer that enables the physical/logical isolation of a secure computational environment capable to monitor, and control, the user space. The major differences between the virtualization (of multiple guest OSs) and the proposed approach are as follows:

- the role of HRPI to provide implicit partitioning of hardware resources in two unequal independent subsystems while user-side OSs cannot see other side (SS) resources while reverse may not be true
- the HRPI should not part of any particular OS environment while provide services to multiple OSs
- two sides should be able to reboot independent of each other
- virtualization is allowed on the user-side, while smaller partition can have tiny OS for security purpose only

Accordingly, User-level computational environments may be virtualized, so these can be monitor, terminated and recreated by the security subsystem as necessary. We have also presented a few illustrated scenarios to introduce and discuss some of the capabilities of the proposed framework in different operational conditions and attack scenarios.

We are currently developing a proof-of-concept implementation of the proposed framework using hardware emulation environments. The goal is to further define the requirements, performance and effectiveness of the proposed architecture in handling different attack and failure conditions.

REFERENCES

[1] R. Sailer, E. Valdez, T. Jaeger, R. Perez, L. V. Doorn, J. L. Griffin and S. Berger. SHype: Secure Hypervisor Approach to Trusted Virtualized Systems. IBM Research Report, February 2, 2005.

[2] J. M. McCune, B. Parno, A. Perrig, M. Reiter, A. Seshadri. "How Low Can You Go? Recommendations for Hardware-Supported Minimal TCB Code Execution". ACM Conference on Architectural Support for Programming Languages and Operating Systems, 2008.

[3] A. Baumann, P. Barham, P. E. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schupbach, and A. Singhania. The Multikernel: A new OS architecture for scalable multicore systems, In the proceedings of 22nd Symposium on Operating Systems Principles (SOSP), 2009.

[4] N. Aggarwal, P. Ranganathan, N. P. Jouppi, J. E. Smith. Configurable isolation: building high availability systems with commodity multi-core processors. In ACM Proceedings of the 34th annual international symposium on Computer architecture, 2007

[5] V. D. Gligor. A guide to understanding covert channel analysis of trusted systems. Technical report, National Computer Security Center, November 1993.

[6] J. Kirch. Virtual Machine Security Guidelines. The Center for Internet Security, September 2007

[7] A. M. Nguyen, N. Schear, H. D Jung, A. Godiyal, S. T. King, H. D. Nguyen. MAVMM: Lightweight and Purpose Built VMM for Malware Analysis. In Annual Computer Security Applications Conference, Honolulu, Hawaii, December 2009.

[8] D. Dasgupta, H. Bedi, D. Garrett. A Conceptual Model of Self-Monitoring Multi-core Systems. In Sixth Cyber Security and Information Intelligence Research Workshop. Oak Ridge, April, 2010.

[9] Kaspersky Labs Business News: http://www.kaspersky.com/news?id=207576021 (accessed on March 13, 2010)

[10] [10] INTEGRITY Multivisor, Green Hills Software at www.ghs.com.

[11] T. Garfinkel and M. Rosenblum. A virtual machine introspection based architecture for intrusion detection. Proc. In Network and Distributed Systems Security Symposium. Pages: 253-285. 2003.

[12] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, and R. Neugebauer, I. Pratt and A. Warfield. Xen and the art of virtualization. In Proceedings of the nineteenth ACM symposium on Operating systems principles, 2003.

[13] M. Carvalho. A Distributed Reinforcement Learning Approach to Mission Survivability in Tactical MANETs, Cyber Security and Information Intelligence Research Workshop, CSIIRW 2009, Oak Ridge National Laboratory, April 2009.

[14] M. Carvalho, Tom Lamkin, Carlos Perez, Organic Resilience for Tactical Environments. 5th International ICST Confernece on Bio-Inspired Models of Network, Information, and Computing Systems (Bionetics). Boston, MA, December, 2010.

[15] A. O. Durahim, E. Savas, K. Yumbul. Implementing a Protected Zone in a Reconfigurable Processor for Isolated Execution of Cryptographic Algorithms. In International Conference on Reconfigurable Computing and FPGAs, December 2009.

[16] E. Keller, J. Szefer, J. Rexford, R. B. Lee. NoHype: Virtualized Cloud Infrastructure without the Virtualization. In Proceedings of the 37th annual international symposium on Computer architecture, June 2010

[17] H. G. Goldman. Building Secure, Resilient Architectures for Cyber Mission Assurance. Presented at Secure & Resilient Cyber Architecture Conference, MITRE, McLean, VA, October 29, 2010.