REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY) 17-08-2015	2. REPORT TYPE Final Report	3. DATES COVERED (From - To) 14-Jan-2011 - 13-Jul-2011

4. TITLE AND SUBTITLE Final Report: Enforcing Hardware-Assisted Integrity for Secure Transactions from Commodity Operating Systems	5a. CONTRACT NUMBER
	5b. GRANT NUMBER W911NF-11-C-0048
	5c. PROGRAM ELEMENT NUMBER 606055
6. AUTHORS Dr. Anup Ghosh, Dr. Kun Sun, Christopher Greamo	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES Secure Command, LLC 4972 Marshall Crown Rd. Centreville, VA 20120 -6425	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSOR/MONITOR'S ACRONYM(S) ARO
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) 58890-CS-SB1.1

12. DISTRIBUTION AVAILIBILITY STATEMENT

Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

In this project, we tried to solve the isolation problem from a different perspective. We still set up two OSes for the user. One is the trusted OS for secure transactions; the other is the untrusted OS for normal transactions. To overcome the drawbacks of the VMMs, we provide a firmware-assisted system, referred to as secure switching system, which allows users to switch between a trusted operating system and an untrusted operating system on the same machine with a short switching time. In our solution, we put a small number of relatively trusted applications in the trusted OS, and a large number of untrusted applications in another untrusted OS. Even if the untrusted OS

15. SUBJECT TERMS

Tailor Trusted Spaces, Hardware-Assisted Security

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Anup Ghosh
a. REPORT	b. ABSTRACT	c. THIS PAGE			
UU	UU	UU	UU		19b. TELEPHONE NUMBER 703-993-4776

Standard Form 298 (Rev 8/98)
Prescribed by ANSI Std. Z39.18

Report Title

Final Report: Enforcing Hardware-Assisted Integrity for Secure Transactions from Commodity Operating Systems

ABSTRACT

In this project, we tried to solve the isolation problem from a different perspective. We still set up two OSes for the user. One is the trusted OS for secure transactions; the other is the untrusted OS for normal transactions. To overcome the drawbacks of the VMMs, we provide a firmware-assisted system, referred to as secure switching system, which allows users to switch between a trusted operating system and an untrusted operating system on the same machine with a short switching time. In our solution, we put a small number of relatively trusted applications in the trusted OS, and a large number of untrusted applications in another untrusted OS. Even if the untrusted OS has been compromised, it cannot affect the applications in the trusted OS. Our solution reduces the attack surface for secure transactions by establishing a tailored trustworthy space and enables secure transactions with very low switching time on commodity hardware platforms.

Enter List of papers submitted or published that acknowledge ARO support from the start of the project to the date of this printing. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in peer-reviewed journals:

(b) Papers published in non-peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in non peer-reviewed journals:

(c) Presentations

Number of Presentations: 0.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received	Paper

TOTAL:

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received	Paper

TOTAL:

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts):

(d) Manuscripts

Received	Paper

TOTAL:

Number of Manuscripts:

Books

Received	Book

TOTAL:

Received	Book Chapter

TOTAL:

Patents Submitted

Patents Awarded

Awards

Graduate Students

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Names of Post Doctorates

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Names of Faculty Supported

NAME	PERCENT_SUPPORTED	National Academy Member
Dr. Kun Sun	0.20	
FTE Equivalent:	**0.20**	
Total Number:	**1**	

Names of Under Graduate students supported

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Student Metrics

This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period:	0.00
The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields:......	0.00
The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields:......	0.00
Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale):......	0.00
Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering:......	0.00
The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense	0.00
The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields:......	0.00

Names of Personnel receiving masters degrees

NAME
Total Number:

Names of personnel receiving PHDs

NAME
Total Number:

Names of other research staff

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Sub Contractors (DD882)

1 a. George Mason University	1 b. 4400 University Drive, MS 4C6
	Fairfax VA 220304422
Sub Contractor Numbers (c):	
Patent Clause Number (d-1):	
Patent Date (d-2):	
Work Description (e):	Research in bios hooks to support switching between operating systems.
Sub Contract Award Date (f-1):	3/3/11 12:00AM
Sub Contract Est Completion Date(f-2):	7/11/11 12:00AM

1 a. George Mason University	1 b. 4400 University Drive, MSN 4C6
	Fairfax VA 220304422
Sub Contractor Numbers (c):	
Patent Clause Number (d-1):	
Patent Date (d-2):	
Work Description (e):	Research in bios hooks to support switching between operating systems.
Sub Contract Award Date (f-1):	3/3/11 12:00AM
Sub Contract Est Completion Date(f-2):	7/11/11 12:00AM

Inventions (DD882)

Scientific Progress

See attachment

Technology Transfer

"Enforcing Hardware-Assisted Integrity for Secure Transaction from Commodity Operating Systems"

PROJECT: OS10-IA4

Contract #: W911NF-C-0048

Final Technical Report (0001AF)

14 July 2011

Sponsored by

Army Research Office

Issued by U.S. Army Aviation and Missile Command Under

By

Secure Command Government Solutions, LLC
3975 University Drive, Suite 460
Fairfax, VA 22030

anup.ghosh@securecommand.com

Effective Date of Contract: 14 January 2011

Reporting Period: 14 June 2011 – 13 July 2011
CLIN: 0001

Problem Summary

Nowadays, the end user's computer is used for handling various tasks and transactions, which can be divided into two categories: normal transactions and secure ones. Web browsing, online gaming, and online socialization are the examples of the former category; online banking and online shopping are the examples of the latter category. The current operating system (OS) can provide user and processes level isolations between the secure transactions and normal transactions; however, these isolations can be easily bypassed by the malware through privilege escalation or other techniques. When one of the applications is compromised, the whole operating system may be compromised too. Considering the growing number of vulnerabilities in the applications and the OS kernels, end users ask for an effective way to isolate secure transactions from normal transactions.

Researchers have tried to use virtual machine monitors (VMMs, also referred to as hypervisors) to isolate different transactions. They dedicate one virtual machine (VM) for normal transactions, and one or more VMs for secure transactions. As long as the virtual machine monitor is not compromised, the compromised applications in the normal VM cannot affect the applications in the secure VM. To reduce the virtualization overhead, other virtualization technologies, such as OS level virtualization, are also used. In addition, VMMs are increasingly employed as components to enforce system security and resilience. We deem that the separation of secure transactions from normal transactions is a good idea, but current isolation for these two categories of transactions still has some problems. Although VMMs have raised the bar, their widespread adoption has attracted the attention of the attackers towards VMM vulnerabilities. Once the VMM has been compromised, all the VMs cannot be trusted. There has been a surge in the reported vulnerabilities for commercial and open source hypervisors. Moreover, the number and nature of attacks against the hypervisors are poised to grow.

Technical Approach

In this project, we tried to solve the isolation problem from a different perspective. We still set up two OSes for the user. One is the trusted OS for secure transactions; the other is the untrusted OS for normal transactions. To overcome the drawbacks of the VMMs, we provide a firmware-assisted system, referred to as secure switching system, which allows users to switch between a trusted operating system and an untrusted operating system on the same machine with a short switching time. In our solution, we put a small number of relatively trusted applications in the trusted OS, and a large number of untrusted applications in another untrusted OS. Even if the untrusted OS has been compromised, it cannot affect the applications in the trusted OS. Our solution reduces the attack surface for secure transactions by establishing a tailored trustworthy space and enables secure transactions with very low switching time on commodity hardware platforms.

In our system, one trusted OS and one untrusted OS are installed on the same computer. Although loaded into the computer memory at the same time, only one of the two OSes is running for a given time; another OS is in sleep state. To switch, the system sleeps one OS and wakes up another OS. We harness the BIOS and sleep states of Advanced

Configuration and Power Interface (ACPI) to control the switching between OSes. We employ Trusted Platform Module (TPM) during boot-up to ensure the integrity of the BIOS. There is no reliance on the TPM after the system boot-up process is complete. The combination of BIOS and TPM provides the Trusted Computing Base (TCB) of our system.

Our system guarantees the integrity of the trusted operating system by operating at BIOS level and providing a thorough isolation between the trusted and untrusted OSes, without sharing any software code that could compromise the integrity of both OSes. Therefore, there is no avenue for a vulnerable or compromised untrusted OS to compromise the trusted OS or its applications, even though the two systems are installed and loaded on the same machine.

To guarantee that a compromised untrusted OS cannot lead to the compromise of the trusted OS, our system uses firmware-assisted isolation to achieve this goal. More specifically, we isolate the following components:

- *Memory Isolation*: Two OS images will run in separate physical memory space. Either OS cannot access the other OS's memory space after boot-up. Therefore, a process running in one OS cannot read, write, or execute memory space allocated to another OS.
- *CPU Isolation*: Two operating systems never run concurrently. When one OS is switched off, all contents in CPU registers and caches are saved in the memory or hard disk and then flushed. There are no hidden channels between the two OSes through the CPU registers or caches.
- *Hard Disk Isolation*: The trusted OS's hard disk is isolated from the untrusted OS. First, we dedicate one hard disk to each OS. A System Management Mode (SMM)-based monitoring module monitors if an OS is accessing another hard disk not belonging to itself or not. Second, a RAM disk is used for saving temporary user data to add an extra layer of protection. Since these data is actually stored in the RAM, it is not accessible in case the hard disk isolation fails.
- *Other I/O Isolation*: When one OS is switched off, all contents maintained by the device driver (e.g., graphic card, network card) are saved and then the devices are powered off. At any time, at most one OS has the control of the devices. This guarantees there is no hidden channel between the two OSes through the I/O devices.

Since there is no running hypervisor or other middleware software that can connect and control the two OSes, the attack surface in our system is much smaller than hypervisor-based systems. An adversary can only target the BIOS-anchored SMM code, which is tiny, and without any need for foreign code (i.e. third party device drivers). Moreover, the BIOS code can be set to read-only at boot-up using TPM or other hardware lock and thus protected from being modified by adversaries.

Summary of Progress for Current Reporting Period

Reporting Period: 14 June – 13 July, 2011

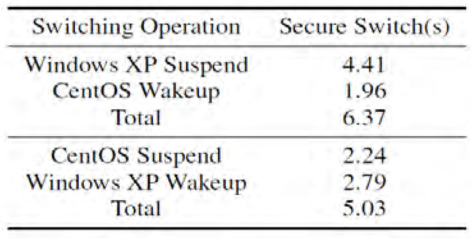
During this reporting period, the research team focused on measuring performance on the secure switching prototype. We measured two latencies: system loading time and switching latency. System loading time is the time duration for loading two OSes into the memory. OS Switching latency measures the time duration when switching from one OS to another.

OS Loading and Switching Latency

We measure two latencies during SecureSwitch: system loading time and switching latency. System loading time is the time duration for loading two OSes into the memory. We use the real-time clock (RTC) to measure it. To record the beginning time, we print out the RTC time through the serial port console at the beginning of the BIOS code. For the ending time, we record the time when the "rc.local" file is executed in CentOS or when a startup application is called in Windows XP. The loading times for both OSes are very close within our system: 74 seconds for loading CentOS and 79 seconds for loading Windows XP. The total loading time is 153 seconds. Though the loading time is relatively long, it only occurs once when the user boots up the system. Moreover, the loading time may be further reduced by using solid-state drive.

OS Switching latency measures the time duration when switching from one OS to another. It consists of two parts: the time to suspend the current OS and the time to wake up another OS. We use the system's Time Stamp Counter (TSC) to measure the OS wakeup time. TSC is a 64-bit register that is present on all x86 processors since the Pentium, and it counts the number of ticks since reset. After pressing the power button, the TSC is reset to 0. We write a user-level program to obtain the current TSC value continuously. We then calculate the wakeup time as TSC*(1/CPU frequency). TSC can be used to measure the wakeup time for both CentOS and Windows XP. However, it is difficult to use TSC to measure Windows XP's suspension time without the Windows source code. Since the OS suspend does not involve the BIOS, we cannot use the BIOS to read the TSC value either.

Table 1. Switching Time



Switching Operation	Secure Switch(s)
Windows XP Suspend	4.41
CentOS Wakeup	1.96
Total	6.37
CentOS Suspend	2.24
Windows XP Wakeup	2.79
Total	5.03

To solve this problem, we use an Oscilloscope, Tektronix TDS 220, to measure the suspension time. We connect the oscilloscope to the serial port on the motherboard. When we initiate the ACPI S3 sleep, a customized program sends an electrical signal to the serial port to indicate the start of S3 sleep. When the system finishes S3 sleep, the oscilloscope receives a power-off electrical signal from the serial port. We use this method to measure the suspension delay for both CentOS and Windows XP.

The latency when the system switches from the trusted OS to the untrusted one is different from the latency when the system switches back, as shown in Table 1. We can see that switching from Windows XP to CentOS requires 5.03 seconds, which is a little faster than switching from CentOS to the Windows XP. For both OSes, the suspend time is longer than the wakeup time. Windows XP's suspend and wakeup times are longer than those of CentOS.

Table 1 only provides a rough latency measurement that is constrained to the specific hardware and software used in our prototype system. For instance, these measurements will change when we use an external VGA card or execute a large number of processes in the OS. The ASUS motherboard has one integrated VGA card with VIA chip and 256 MB video memory. When we insert an external VGA card with S3 chip and 64 MB memory, the external VGA card needs less suspension time than the integrated one due to a smaller video memory size. To our surprise, the external VGA card requires a wakeup time that is three times longer than the integrated one due to the fact that coreboot needs to call the option ROM of the external video card, but it encounters a computability problem and dramatically delays the wakeup.

In addition, we run multiple while(1) programs on the Linux to see how the CPU intensive processes affect the switching time. When we run five while(1) programs at the same time, the switching time is about three times longer. We deduced that most of the increasing is due to the user space suspend and wakeup, while the delay in kernel space does not change much. This leads us to breakdown the operations in BIOS, user space, and kernel space to understand the major contributors for the time delay. Due to the closedsource nature of Windows XP, we only break down the operations on the CentOS 5.5 with Coreboot V4.

Linux Suspend Breakdown

We use *Ftrace* to trace the suspension function calls in Linux S3 sleep. According to the function call graph generated by Ftrace, we divide the suspend operations into two phases: user space suspend and kernel space suspend. We use the pm-suspend script provided by the OS to trigger the suspend. The script basically notifies the Network Manager to shut down networking and uses vbetool to call functions at video option ROM to save VGA states. It then echoes string "mem" to /sys/power/state. This jumps to the kernel space and stops the user space. In the kernel space, the suspend code goes through the device tree and calls the device suspend function in each driver. The kernel then powers off these devices. To measure the user space suspend, we record the TSC time stamp in file /var/log/pm/suspend.log. For kernel time measurement, we add "printk" statements between various components of the kernel.

Figure 1 shows the time breakdown for user space suspend, and Figure 2 shows the time breakdown for kernel space suspend. The total suspend times for user space and kernel space are 1517.33 ms and 722.79 ms respectively. In the user space, by running command chvt 63, the monitor changes the GUI terminal to /dev/tty63 as the foreground virtual terminal. In the clock operation, the OS stops the Network Time Protocol Daemon and writes the current system time to RTC time in CMOS. For the video operation, the OS uses *vbetool* to save current video state to the /var/run directory in memory. Other events include stopping network manager and saving the state of CPU frequency governors, etc. In the kernel space, the most time is consumed by stopping the keyboard, mouse, and hard disks. We use a PS/2 mouse and keyboard in our system. The suspending functions of the mouse and keyboard drivers reset the devices, which causes the delay. For the hard disk, delay comes from synchronizing the cache. The two hard disks each have 16 MB caches, and cache write is enabled by default for the SATA disk. The OS also needs to stop other devices, such as the USB and serial ports, which takes relatively less time.

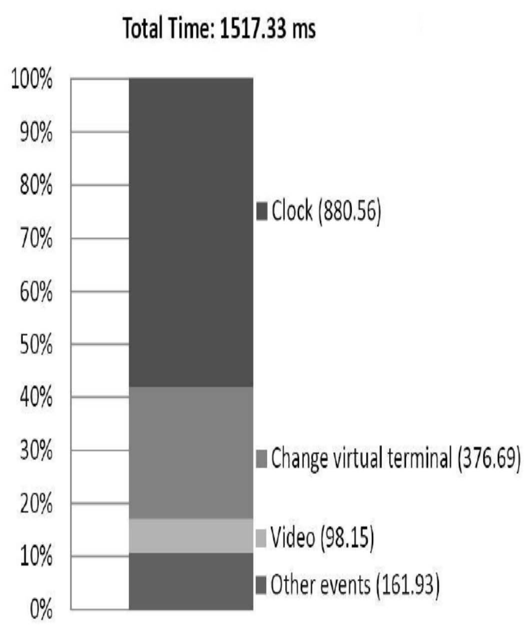


Figure 1. User Space Suspend Breakdown

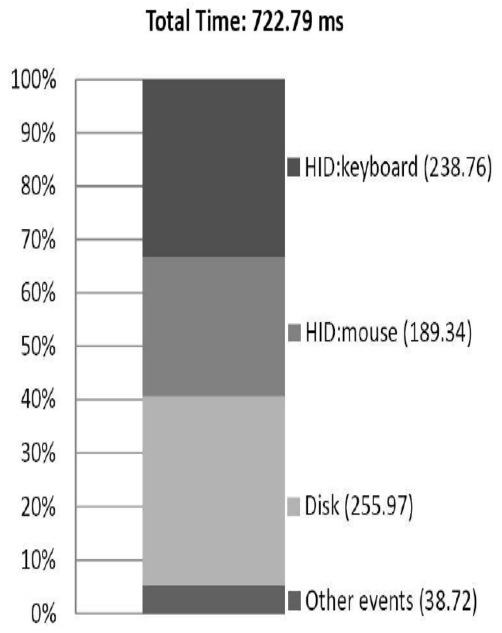


Figure 2. Kernel Space Suspend Breakdown

Linux Wakeup Breakdown

Unlike S3 suspend, S3 wakeup operations are handled by both the BIOS and the OS. The wakeup process starts from a hardware reset. The system enters the BIOS first, then jumps to the OS wakeup vector. The total latency time in BIOS is almost constant and equal to 1259.25 ms. Again, the OS wakeup operations can be divided into two parts: kernel space wakeup and user space wakeup. The wakeup latency in kernel space and user space are 698.74 ms and 612.04 ms respectively. Figure 3 shows the time breakdown for the major components in the kernel space. The major delay contributors in kernel space are the USB and the mouse. There are four USB ports on the motherboard. Since coreboot doesn’t provide an optimized support for the USB, OS needs to initialize all four of the USB ports. The BIOS must initialize the keyboard, but not necessarily the mouse. We discovered that the mouse takes more time than the keyboard in kernel-space wakeup due to the OS initialization of the mouse. Figure 4 shows the time breakdown for wakeup in the user space. We can see that cleaning up the files and changing the foreground’s virtual terminal (chvt 1) take up most of the time.

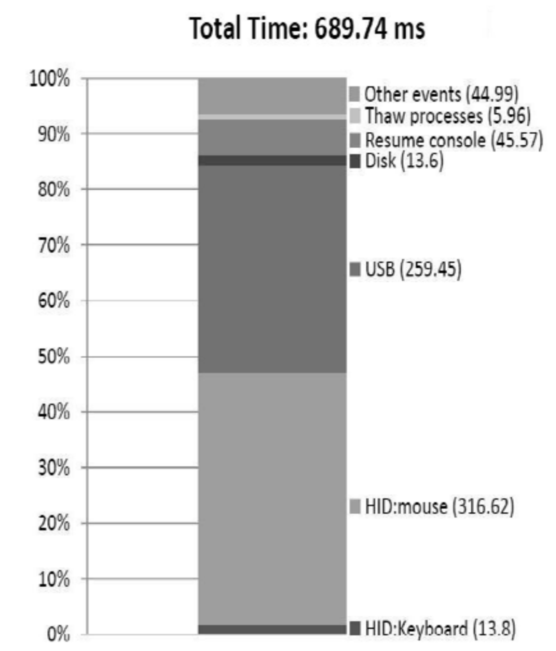


Figure 3. User Space Wakeup Breakdown

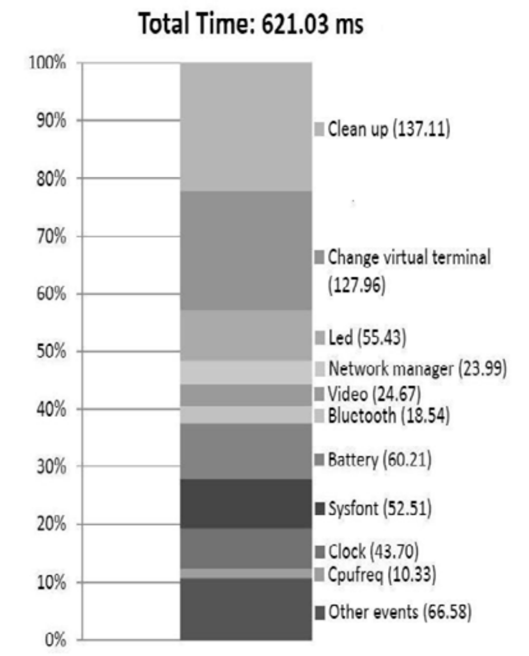


Figure 4. Kernel Space Wakeup Breakdown

Plans for Next Reporting Period

This report completes the Phase I effort.

Issues / Concerns

No issues to report for this period.

Financial Status

As of July 13, 2011:

- Total cumulative expenditures are **$99,997**.
- Total funding received to date is **$99,997**.
- Remaining funding backlog is **$0**.

Projected Spend Profile:

	2011					
	Jan 14 – Feb 13	*Feb 14 – Mar 13*	*Mar 14 – Apr 13*	*Apr 14 – May 13*	*May 14 – Jun 13*	*Jun 14 – July 13*
Expenditures	$16,663	$16,663	$16,663	$16,663	$16,663	$16,663
Cumulative	$16,663	$33,326	$49,989	$66,651	$83,314	$99,997
Funds Remaining	$83,314	$66,651	$49,989	$33,326	$16,663	$0