NPS-CS-08-012

NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

Use of Trusted Software Modules for High Integrity Data Display

by

Timothy E. Levin, Thuy N. Nguyen, Paul C. Clark, Cynthia E. Irvine,
David J. Shifflett, Timothy M. Vidas

June 2008

Prepared for NSF and DARPA

This report was prepared for and funded by National Science Foundation and the Defense Advanced Research Projects Agency.

This report was prepared by:

Timothy E. Levin
Research Associate Professor

Reviewed by:

Peter J. Denning
Department of Computer Science

Released by:

Dan C. Boger
Interim Vice President and
Dean of Research

REPORT DOCUMENTATION PAGE		Form approved OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** 21 November 2007	**3. REPORT TYPE AND DATES COVERED** Research; 7/1/07 – 7/1/08

4. TITLE AND SUBTITLE Use of Trusted Software Modules for High Integrity Data Display	**5. FUNDING** Grant number: CNS-0430566 and CNS-0430598
6. AUTHOR(S) Timothy E. Levin, Thuy N. Nguyen, Paul C. Clark, Cynthia E. Irvine, P. David J. Shifflett, Timothy M. Vidas	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Center for Information Systems Security Studies and Research (NPS CISR) 1411 Cunningham Rd., Monterey, CA 93943	**8. PERFORMING ORGANIZATION REPORT NUMBER** NPS-CS-08-012
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Science Foundation, 4201 Wilson Blvd. 1175 N. ArlingtonVA22230 DARPA, 3701 Fairfax Drive, Arlington, VA 22203	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** Not applicable

11. SUPPLEMENTARY NOTES

The views expressed in this report are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT (Maximum 200 words.)

This report provides summary of the interface, mechanisms and semantics for high integrity display of information in a secure computer system, based on the use of a high assurance separation kernel and trusted software modules in both the application domain and the trusted software domain.

14. SUBJECT TERMS Operating systems: Separation Kernel; secure display; trusted software module; security; security architecture	**15. NUMBER OF PAGES** 16
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

Naval Postgraduate School
Center for Information Systems Security Studies and Research

Center of Academic Excellence

SecureCore
Trustworthy Commodity Computation and Communication

Technical Report: NPS-CS-08-012

Use of Trusted Software Modules for High Integrity Data Display

Timothy E. Levin, Thuy D. Nguyen, Paul C. Clark, Cynthia E. Irvine, David J. Shifflett and Timothy M. Vidas

June, 2008

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0430566 with support from DARPA ATO. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or of DARPA ATO.

Author Affiliations

Timothy E. Levin, Thuy D. Nguyen, Paul C. Clark, Cynthia E. Irvine, David J. Shifflett and Timothy M. Vidas:

Naval Postgraduate School

Center for Information Systems Security Studies and Research

Computer Science Department

Naval Postgraduate School

Monterey, California 93943

Abstract

This report provides summary of the interface, mechanisms and semantics for high integrity display of information in a secure computer system, based on the use of a high assurance separation kernel and trusted software modules in both the application domain and the trusted software domain.

This page is intentionally blank.

Use of Trusted Software Modules for High Integrity Data Display

Timothy E. Levin, Thuy D. Nguyen, Paul C. Clark, Cynthia E. Irvine,
David J. Shifflett and Timothy M. Vidas

This document describes an on-going research effort of the SecureCore project.[4] The intended audience is the SecureCore research team whose members are familiar with the SP hardware [5] and SecureCore software architecture [1].

Introduction

High assurance computer systems provide a trusted path for users to securely interact with the trusted computing base (TCB), including a keyboard for user input and a display device for the TCB to output information to the user. The TCB may also provide interfaces for other programs to write to an attached display device such that there is high assurance that data written to the device is displayed to the user without modification. Usually this assurance is established through the program's direct access to a securely designed interface, along with the ability of the TCB to protect itself.

Despite these capabilities, there is no guarantee that high integrity information is written to the screen by untrusted applications supported by untrusted operating systems. During emergencies, there may be a need for applications, which do not have direct access to the TCB interface, to write to the screen with high assurance that the data is displayed to the user. The SP trusted software module (TSM) provides a context for high integrity code execution in the application domain, but requires support for a trusted display. This document describes an architecture that supports a high integrity conduit between an application-level TSM and a high integrity display. Before describing the design of this capability, this report briefly reviews the SecureCore architecture, including the management of devices and communication channels.

SecureCore Overview

In the SecureCore security architecture, a *trusted software module* (TSM) hosted by an untrusted commercial operating system may communicate with a remote *central authority* (the Authority) through a channel that is encrypted with the SP [5] hardware-based *Device Root Key* (DRK). [6] Using a mechanism described in this paper, the TSM can communicate directly with the TCB – the Trusted Management Layer (TML) – ensuring that data from the TSM is displayed to the user, without potential interference by the commercial OS.

The TML is composed of the Least Privilege Separation Kernel (LPSK) [7][9] and the Secure Core Security Services (SCSS) layer. The TML provides a processing environment in which a separate OS in each partition can manage its own applications without interference from other partitions. In this environment there are three types of partitions: trusted, emergency, and normal.

The TML provides a high integrity display mechanism that is available to both applications hosted by untrusted operating systems in untrusted partitions and applications hosted by the SecureCore Operating System (SCOS) in the trusted partition. This mechanism ensures that information can be displayed to the user with high integrity, without requiring the user to explicitly change the *focus* (see below) of the system to the trusted partition. The TML manages small, high integrity display areas such that critical messages from different applications can be seen simultaneously by the user. The limited screen area allocated for these critical messages as well as the simple mechanisms required in high assurance TCB development result in the requirement that each message must be short and non-graphical.

Management of Devices and Communication Channels

The TML provides synchronous and asynchronous device I/O interfaces and services. Device management is characterized in three ways. First, certain devices may be virtualized by the TML. In this case, the TML presents the OSes with a virtual hardware device, such that undesirable interference between partitions is prevented, while the client operating systems and SCOS may simultaneously share the services of the actual underlying device.

Second, a device may be assigned solely to a particular partition. For example, memory-mapped devices may be used so that they interact with processes through dedicated memory regions. Here device management is vectored by the LPSK to the client OS for its exclusive use.

Finally, the SCSS supports focus management by allowing a particular partition temporary exclusive use of a device until the user, via the TPA, chooses to associate the device with a different partition. The keyboard, mouse, and screen require focus management. While keyboard and mouse input is vectored to the selected partition, all partitions continue to update their screen buffers. The system supports configurations with tiled windowing, so that output from partitions with sensitivity levels dominated by the user's current session level can be simultaneously displayed. For partitions that write to the display at a level higher than the current session level, the SCSS will maintain their screen output until the user establishes a session at a high enough level. Thus, instead of having to shut down activity at a particular session level to make the device available at a different level, e.g. [2][3], all partitions can continue to execute according to the predefined schedule.

The TML divides the physical display screen into two regions. One region is reserved by the TML as a high-integrity display area, for example, for critical system messages. The other region is available to partition applications, as normal.

Each exported device has two labels associated with it: a read-class and a write class. [8] A device where both labels are equal is a single-level device; a device for which the two labels are different is a multilevel device. For single-level devices, the SCSS exports an interface for administrative change of the device level during runtime, thereafter restricting access to the device to that security level. A multilevel device may be assigned to an untrusted partition by the SCSS, which then labels all of its I/O at the level of the partition. Finally, a multilevel device can be used by a trusted application, which is trusted to apply the right labels to its data.

The secure attention key (SAK), which is currently a keystroke sequence, allows users to invoke the Trusted Path Application (TPA). The LPSK services the SAK interrupt and passes control to the SCSS, which manages the focus change to the Trusted Path Application in the trusted partition.

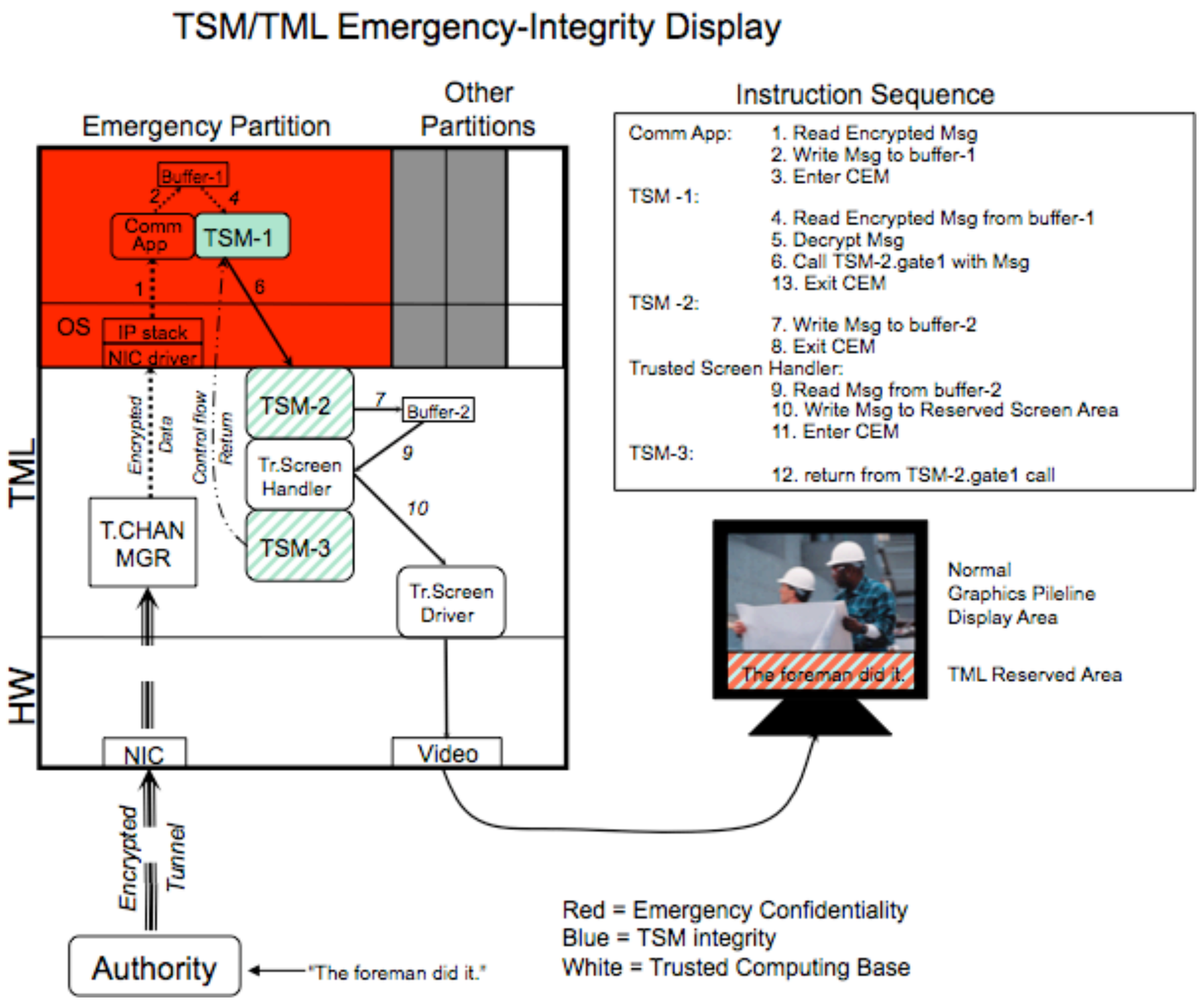
The SCSS is responsible for establishing trusted communication channels between the device and the central authority or trusted third parties. It detects and verifies emergency signals from the central authority, restricts or allows access to the emergency partition accordingly, and raises or lowers alarms to other programs within the device, as configured to do so.

Partitions may host network-capable client operating systems. The TML multiplexes communications among its partitions and to external nodes while ensuring enforcement of the system's information flow policy. It associates implicit or explicit labels with information as needed. It also provides support for the establishment of secure communication channels, e.g. VPNs, with external nodes. For example, since network communication protocols are bidirectional, in a partition-to-partition communication channel, both partitions must be the same sensitivity level.

High Integrity Display of Data

The Trusted Application Display mechanism allows an application TSM or the TPA to send data directly to a reserved region of the display device. Figure 1 illustrates how this would be organized in an untrusted partition, in this case, the Emergency Partition.

In the case of the TSM, the text for display may be encrypted, e.g., if it has come from the Authority. Figure 1 shows this information stored in *buffer 1*.The TSM decrypts the data using SP-protected keys, and then stores the clear text in SP-protected memory using SP Secure_Store instructions, to prevent other applications from viewing or modifying it. The application TSM then invokes a TML TSM module, through a TML call gate. The TML TSM reads the clear text using SP Secure_Load and then uses regular store instructions to write the data to a TML buffer (*buffer 2*). The TML TSM then exits CEM mode and invokes the TML's Trusted Screen Handler, which sends the data to the Trusted Screen Driver for display in the restricted display region, labeling the data as appropriate. Finally, the TML TSM code re-enters CEM mode and returns execution control to the application-TSM, with a return value or similar mechanism indicating the success or failure of the display operation. As the data traverses through different software components, its integrity is protected by SP hardware mechanisms and the TML.



Comm App:	1. Read Encrypted Msg
	2. Write Msg to buffer-1
	3. Enter CEM
TSM -1:	
	4. Read Encrypted Msg from buffer-1
	5. Decrypt Msg
	6. Call TSM-2.gate1 with Msg
	13. Exit CEM
TSM -2:	
	7. Write Msg to buffer-2
	8. Exit CEM
Trusted Screen Handler:	
	9. Read Msg from buffer-2
	10. Write Msg to Reserved Screen Area
	11. Enter CEM
TSM-3:	
	12. return from TSM-2.gate1 call

Figure 1. Trusted Application Display

Summary

This report presents a high level design of a trusted display capability that utilizes both hardware and software protection mechanisms to preserve data integrity. This capability affords users a high level of confidence that the data seen on the display device has not been tampered with by untrusted programs executing in the application domain.

REFERENCES

[1] Paul C. Clark, Cynthia E. Irvine, Timothy E. Levin, Thuy D. Nguyen and Timothy M. Vidas, "SecureCore Software Architecture: Trusted Path Application (TPA) Requirements," NPS Technical Report NPS-CS-07-001, Naval Postgraduate School, Monterey, CA, December 2007.

[2] Final Evaluation Report, Wang Federal Incorporated, XTS- 300, National Computer Security Center, CSC-EPL- 92/003.B, July 11, 1995.

[3] Final Evaluation Report, Gemini Computers, Incorporated, Gemini Trusted Network Processor, National Computer Security Center, 34-94, June 28, 1995.

[4] Irvine, C., "Collaborative Research: SecureCore for Trustworthy Commodity Computing and Communications," www.fastlane.nsf.gov, Award 0430566. 31 Mar. 2005.

[5] R. Lee, P. Kwan, J. McGregor, J. Dowskin, and Z. Wang, "Architecture for protecting critical secrets in microprocessors," in *Proc. 32nd International Symposium on Computer Architecture*, (Madison, Wisconsin), pp. 2–13, IEEE Computer Society, June 2005.

[6] T. E. Levin, et al., "Trusted Emergency Management," June 2008. In submission.

[7] T. E. Levin, C. E. Irvine, C. Weissman and T. D. Nguyen, "Analysis of Three Multilevel Security Architectures." in *Proc. Computer Security Architecture Workshop*, Fairfax, Virginia, USA November 2007.

[8] T. F. Lunt, P. G. Neumann, D. E. Denning, R. R. Schell, M. Heckman, and W. R. Shockley, "Secure Distributed Data Views Security Policy and Interpretation for DMBS for a Class A1 DBMS," Tech. Rep. RADC-TR-89-313, Vol I, Rome Air Development Center, Griffiss, Air Force Base, NY, December 1989.

[9] U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness. National Information Assurance Partnership, Version 1.03 ed., 29 June 2007.

This page left intentionally blank

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	2
2.	Dudley Knox Library, Code 013 Naval Postgraduate School Monterey, CA 93943	1
3.	Research Office Naval Postgraduate School Monterey, CA 93943	1
4.	Paul C. Clark Department of Computer Science Naval Postgraduate School Monterey, CA 93943	2
5.	Cynthia Irvine Department of Computer Science Naval Postgraduate School Monterey, CA 93943	2
6.	Timothy Levin Department of Computer Science Naval Postgraduate School Monterey, CA 93943	2
7.	Karl Levitt National Science Foundation 4201 Wilson Blvd. Arlington, VA 22230	1
8.	Thuy Nguyen Department of Computer Science Naval Postgraduate School Monterey, CA 93943	2
9.	David J. Shifflett Department of Computer Science Naval Postgraduate School Monterey, CA 93943	2

10. Timothy M. Vidas 2
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943