NPS-CS-07-021

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

SecureCore Software Architecture:
Trusted Management Layer (TML)
Kernel Extension Module Interface Specification

by

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

January 2008

Approved for public release; distribution is unlimited

This page intentionally left blank

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

Vice Admiral Daniel T. Oliver (Retired)
President

Leonard Ferrari
Provost

This material is based upon work supported by the National Science Foundation (NSF). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of that agency.

This report was prepared by:

David J. Shifflett
Research Associate

Paul C. Clark
Research Associate

Cynthia E. Irvine
Professor

Thuy D. Nguyen
Research Associate

Timothy M. Vidas
Research Associate

Timothy E. Levin
Research Associate Professor

Reviewed by:

Peter J. Denning, Chair
Department of Computer Science

Released by:

Dan C. Boger
Interim Associate Provost
and Dean of Research

This page intentionally left blank

REPORT DOCUMENTATION PAGE

Form approved

OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	29 January 2008	Research; September 2006 - January 2008

4. TITLE AND SUBTITLE

SecureCore Software Architecture: Trusted Management Layer (TML)
Kernel Extension Module Interface Specification

5. FUNDING

CNS-0430566

6. AUTHOR(S)

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

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Center for Information Systems Security Studies and Research (CISR)
1411 Cunningham Road, Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

NPS-CS-07-021

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Science Foundation (NSF)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

This material is based upon work supported by the National Science Foundation (NSF). 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 NSF.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words.)

A mobile computing device has more inherent risk than desktops or most other stationary computing devices. Such mobile devices are typically carried outside of a controlled physical environment, and they must communicate over an insecure medium. The risk is even greater if the data being stored, processed and transmitted by the mobile device is classified. The purpose of the SecureCore research project is to investigate fundamental architectural features required for the trusted operation of mobile computing devices so the security is built-in, transparent and flexible. A building block for the SecureCore project is a Least Privilege Separation Kernel (LPSK). The LPSK together with extension modules provides the security base. Detailed functional interfaces between the LPSK and extension modules are described, as well as usage scenarios.

14. SUBJECT TERMS

High Assurance, Security Kernel

15. NUMBER OF PAGES

20

16. PRICE CODE

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

NSN 7540-01-280-5800 Standard Form 298 (Rev. 2-89)

This page left intentionally blank

NPS-CS-07-021

SecureCore

Trustworthy Commodity Computation and Communication

SecureCore Technical Report

SecureCore Software Architecture:

Trusted Management Layer (TML) Kernel Extension Module Interface Specification

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

January 29, 2008

The Center for Information Systems Security Studies and Research

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0430566 and CNS-0430598 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 Affiliation:

Center for Information Systems Security Studies and Research
Computer Science Department
Naval Postgraduate School
Monterey, California 93943

Table of Contents

1 Introduction 1
  1.1 Background 1
2 Core Kernel Interfaces for kernel extension modules 1
  2.1 kio_printf 2
  2.2 kio_printf_str 3
  2.3 kio_printf_int 4
  2.4 kio_printf_char 5
3 SP Emulation Module Interfaces for LPSK 6
References 7

[THIS PAGE IS INTENTIONALLY BLANK]

1 Introduction

1.1 Background

SecureCore is a research project funded by the National Science Foundation (NSF) to investigate the fundamental architectural features required for trustworthy operation of mobile computing devices such as smart cards, embedded controllers and hand-held computers. The goal is to provide secure processing and communication features for resource-constrained platforms, without compromise of performance, size, cost or energy consumption. In this environment, the security must also be built-in, transparent and flexible.

This document describes the interfaces for kernel extension modules that may be incorporated into the Trusted Management Layer (TML), specifically the Least Privilege Separation Kernel (LPSK). The LPSK is composed of modules which are used as the building blocks of the kernel implementation, these modules are referred to as core kernel modules. Kernel extension modules are separate from the core LPSK modules, providing additional functionality. Included in this document are interfaces that the LPSK provides for the kernel extension modules to call, as well as interfaces the kernel extension modules present for the LPSK to call under certain circumstances.

A description of the software architecture and definitions can be found elsewhere [1]. This document assumes the reader is familiar with the architecture and terminology of the SecureCore project.

2 Core Kernel Interfaces for kernel extension modules

The ‘printf’ interfaces function similar to the C library ‘printf’ call, with the following limitations.

- Only certain escape characters (e.g. \n, \r) are recognized. The allowed escape characters are ‘\r’ and ‘\n’. These escape characters are used in the same manner as the C library ‘printf’ call.
- Only certain format specifiers (e.g. %d, %s) are recognized. The allowed format specifiers are %s, %c, %d, %u, and %x. These format specifiers are used in the same manner as the C library ‘printf’ call.
- It is assumed that string inputs to the ‘printf’ functions will be NULL (‘\0’) terminated and contain only ASCII printable characters.

The LPSK does not support dynamic allocation of memory, therefore there is no ‘malloc’ interface. Memory required by kernel extension modules must be compiled into the kernel extension module, via data declarations, as described in ‘Kernel Extension Module Integration Guide’ [3].

2.1 kio_printf

This call is used to display a string to the screen.

2.1.1 Prototype

void kio_printf(const char * const buffer);

2.1.2 Inputs

- buffer
  The string to be displayed.

2.1.3 Outputs

- None

2.1.4 Effects

- None

2.1.5 Errors

- None

2.2 kio_printf_str

This call is used to display a formatted string to the screen.

2.2.1 Prototype

```
void kio_printf_str(
    const char * const format,
    const char * const buffer);
```

2.2.2 Inputs

- format

  The string containing the format specifier. The format specifier (%s) will be replaced by the input buffer.

- buffer

  The string to be displayed, according to the format specifier.

2.2.3 Outputs

- None

2.2.4 Effects

- None

2.2.5 Errors

- None

2.3 *kio_printf_int*

This call is used to display a formatted number to the screen

2.3.1 Prototype

```
void kio_printf_int(
    const char * const format,
    const int value);
```

2.3.2 Inputs

- format

  The string containing the format specifier. The format specifier (%d, %x, or %u) will be replaced by the string representation of the input value.

- value

  The numeric value to be displayed, according to the format specifier.

2.3.3 Outputs

- None

2.3.4 Effects

- None

2.3.5 Errors

- None

2.4 kio_printf_char

This call is used to display a formatted character to the screen.

2.4.1 Prototype

```
void kio_printf_char(
    const char * const format,
    const char  value);
```

2.4.2 Inputs

- format

  The string containing the format specifier. The format specifier (%c) will be replaced by the input character.

- value

  The character to be displayed, according to the format specifier.

2.4.3 Outputs

- None

2.4.4 Effects

- None

2.4.5 Errors

- None

3 SP Emulation Module Interfaces for LPSK

```
/* This structure defines the register state passed to the CEMInterrupt calls */
typedef struct {
  unsigned int gs;           /* the GS register */
  unsigned int fs;           /* the FS register */
  unsigned int es;           /* the ES register */
  unsigned int ds;           /* the DS register */
  unsigned int cspl0;        /* the CS register in the interrupt handler */
  unsigned int sspl0;        /* the SS register in the interrupt handler */
  unsigned int edi;          /* the EDI register */
  unsigned int esi;          /* the ESI register */
  unsigned int ebp;          /* the EBP register */
  unsigned int esp;          /* the ESP register */
  unsigned int ebx;          /* the EBX register */
  unsigned int edx;          /* the EDX register */
  unsigned int ecx;          /* the ECX register */
  unsigned int eax;          /* the EAX register */
  unsigned int int_num;      /* the interrupt number */
  unsigned int error_code;   /* the error code that caused the interrupt,
                                only supported for interrupts
                                0x08, 0x0A, - 0x0E, and 0x10,
                                all other interrupts have 0 in this field */
   /* The ‘plx’ fields below refer to the register state at the time the
      interrupt occurred. If the interrupt occurred inside PL0 the
     ‘ssplx’ and ‘espplx’ fields will contain 0 */
  unsigned int eipplx;       /* the IP register at the time of the interrupt */
  unsigned int csplx;        /* the CS register at the time of the interrupt */
  unsigned int eflags;       /* the flags register */
  unsigned int espplx;       /* the ESP register at the time of the interrupt */
  unsigned int ssplx;        /* the SS register at the time of the interrupt */
}registers_struct;
```

These following interfaces are defined elsewhere. [2]

This function will be called during LPSK initialization.

```
int SPHW_PowerOn (void *initdata);
```

This function will be called during LPSK shutdown, or halt.

```
int SPHW_PowerOff (void *initdata);
```

This function will be called upon receipt of interrupt number 200 (0xC8).

```
SPFault SPHW_CEMInterrupt_Suspend (
    void *regs,
    size_t regslen,
    void *returnip,
            const unsigned int partitionid,
```

```
        const unsigned int processid);
```

This function will be called prior to returning from the handler for interrupt number 200 (0xC8).

```
    SPFault SPHW_CEMInterrupt_Resume (
        void *regs,
        size_t regslen,
        void *returnip,
        const unsigned int partitionid,
        const unsigned int processid);
```

This function will be called after calling an emulated SP instruction to determine if a hardware fault was generated.

```
    int SPHW_CheckFault (SPFault fault);
```

References

[1] Clark, Paul C., Irvine, Cynthia E., Levin, Timothy E., Nguyen, Thuy D., Vidas, Timothy M., “SecureCore Software Architecture: Trusted Path Application (TPA) Requirements”, NPS-CS-07-001, December 2007.

[2] Dwoskin, Jeffrey, Bhaskara, Ganesha, Lee, Ruby, “SecureCore Prototype/Demo Manual: Definition & Concept of Operations, SP TSM Interfaces, TML & SCOS Interface, and SP Emulation Module Interface”, Version 0.7, January 20, 2008.

[3] Shifflett, David J., Clark, Paul C., Irvine, Cynthia E., Nguyen, Thuy D., Vidas, Timothy M., Levin, Timothy E., “SecureCore Software Architecture: Trusted Management Layer (TML) Kernel Extension Module Integration Guide”, NPS-CS-07-022, December 2007

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 93943-5100	2
3.	Research Office, Code 09 Naval Postgraduate School Monterey, CA 93943-5138	1
4.	Karl Levitt National Science Foundation 4201 Wilson Blvd. Arlington, VA 22230	1
5.	Lee Badger DARPA 3701 Fairfax Drive Arlington, VA 22203	1
6.	David J. Shifflett Code CS Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5118	1
7.	Paul C. Clark Code CS/Cp Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5118	1
8.	Cynthia E. Irvine Code CS/Ic Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5118	2

9. Timothy E. Levin 1
Code CS/Tl
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943-5118

10. Thuy D. Nguyen 1
Code CS/Tn
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943-5118

11. Timothy M. Vidas 1
Code CS
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943-5118

[THIS PAGE IS INTENTIONALLY BLANK]