Muen
An x86/64 Separation Kernel for High Assurance

Reto Buerki
Adrian-Ken Rueegsegger
Institute for Networked Solutions
21.09.2017
Overview

- Introduction
- Security of Complex Software
- Low Complexity Kernel
- SPARK for Operating Systems
- Conclusion
Who are we?

• MSc in Engineering from HSR
• 9+ years in IT security industry
• Researchers @ INS
• Focus on
  • Development of trustworthy systems
  • Platform security
  • Component-based systems
What are we currently doing?

Muen Separation Kernel
https://muen.sk
Examples of Software Failures

- RCE: Symantec Anti-Virus Engine (kernel) (CVE-2016-2208)
- RCE: Cisco ASA (CVE-2016-1287)
- RCE: glibc (CVE-2015-7547)
- Authentication Bypass: Cisco (CVE-2016-1329)
- VM escape: MS Hyper-V (CVE-2016-0088)
- Information disclosure: Heartbleed (CVE-2014-0160)
Getting Software right is hard

• Software is inherently complex
  • Accidental difficulty
  • Essential difficulty

• Real world problems are hard to solve
• There is no silver bullet [3]
\[ P(\text{Program\_Correct}) = P(\text{Line\_Correct})^{SLOC} \]
Security of Complex Software

![Graph showing the relationship between kSLOC and the probability of a defective program.](image)
Assumptions (e.g.):
10% security defects,
20% exploitable
Secure Software

- Tiny size
- *Very* low defect rate
- Low security defect ratio
SLOC

Linux kernel

Separation Kernel I

- Kernel must guarantee component separation
- Static partitioning and isolation of resources
- Static configuration during integration
- Only includes necessary features
- Well suited for formal verification
Separation Kernel II

Native Subject

Linux VM

Genode VM

5,500 SLOC

Muen Separation Kernel

Subject base-hw kernel

Linux kernel

base-hw kernel

non-root mode

root mode

Non-root mode

Root mode
Low Complexity

• In trusted code, every line counts and should be avoided
• Shift complexity out of trusted components
• Design system in order to minimize overall TCB
• KISS
  • Greatly improves readability of code
  • Facilitates effective code review by third party
Low Kernel Complexity

- Init
- Signaling
- Scheduler
- Page Tables
- Caps/Perms
- VT-x
- VT-d
- Message Passing
- Schedule Planning
- Memory Allocator
- Device Allocator
- Device Drivers
- User Interface
- File System
- VM Monitor
- Posix Interface
Low Kernel Complexity

- Init
- Signaling
- Scheduler
- Page Tables
- Caps/Perms
- VT-x
- VT-d
- Message Passing
- Schedule Planning
- Memory Allocator
- Device Allocator
- Device Drivers
- User Interface
- File System
- VM Monitor
- Posix Interface
Static Resource Allocation

Init

Signaling

Scheduler

Page Tables

Caps/Perms

VT-x

VT-d

Schedule Planning

Memory Allocator

Device Allocator

Device Drivers

User Interface

File System

VM Monitor

Posix Interface
SPARK 2014 for Operating Systems

• No pointers
• No dynamic memory allocation
• No concurrency
SPARK 2014 for Operating Systems

- No pointers
- No dynamic memory allocation
- No concurrency
- Fixed structures
- Static resource allocation
- One kernel instance / CPU
- Abort on host interrupts

Greatly simplified verification
SPARK 2014 for Operating Systems

• No pointers
• No dynamic memory allocation
• No concurrency

• Fixed structures
• Static resource allocation
• One kernel instance / CPU Abort on host interrupts

→ Greatly simplified verification
Lean verification

- Proof annotations are part of the language
- Implicit generation of VCs for integrity preservation (Absence of runtime errors)
- All ARTE VCs proven automatically
- Integration of theorem provers possible when needed
- Speed allows proofs to be part of build process
This presentation is given on a system running on Muen
Summary

- Secure software is limited in complexity
- Separation of untrusted components essential
- Muen provides a solid foundation for high assurance systems
- Muen is the base of complex high security solutions in development
- SPARK 2014 enables lean verification
- Formal verification can be done under commercial constraints
Discussion

Get Muen at https://muen.sk/

Getting started with SPARK 2014
http://university.adacore.com/courses/spark-2014/
AdaCore and Altran UK Ltd.  

Alex Beal.  
http://www.usrsb.in/probability-of-a-correct-program.html.

Frederick P. Brooks, Jr.  
No Silver Bullet Essence and Accidents of Software Engineering.  

J. M. Rushby.
Design and Verification of Secure Systems.