Genode - OS Security By Design

Dr.-Ing. Norman Feske
<norman.feske@genode-labs.com>
Outline

1. Introduction
2. Architectural Principles
3. Showcases
4. Current Topics
Outline

1. Introduction

2. Architectural Principles

3. Showcases

4. Current Topics
Universal Truths

Assurance ↔ Scalability

Accountability ↔ Utilization

Security ↔ Ease of use
Problem: Complexity

Today’s commodity OSes  Exceedingly complex trusted computing base (TCB)

TCB of an application on Linux:

- Kernel + loaded kernel modules
- Daemons
- X Server + window manager
- Desktop environment
- All running processes of the user

→ User credentials are exposed to millions of lines of code
Implications:

- High likelihood for bugs (need for frequent security updates)
- Huge attack surface for directed attacks
- Zero-day exploits
Universal Truths

Assurance  Scalability

Accountability  Utilization

Security  Ease of use
Problem: Resource management

- Pretension of unlimited resources
- Lack of accounting
  → Largely indeterministic behavior
  → Need for complex heuristics, schedulers
Universal Truths

Assurance ↔ Scalability

Accountability ↔ Utilization

Security ↔ Ease of use
Key technologies

- Microkernels
- Componentization, kernelization
- Capability-based security
- Virtualization

…but how to compose those?
Genode architecture

→ Application-specific TCB
Combined with virtualization
Components

- OKL4
- NOVA
- Base-HW
- FIASCO
- FIASCO.OE
- CODEZERO
Components

- Genode - OS Security By Design

Diagram showing various components and their connections:
- Noux
- GDB Monitor
- CLI Monitor
- Init
- VirtualBox
- Seoul
- NOVA Microhypervisor
- FIASCO
- FIASCO OC
- L4Linux
- L4Ka
- FB SDL
- PCI
- SATA
- Audio
- UART
- OMAP4
- PS2
- NIC
- VESA
- USB
- GPIO
- Exynos-5
- Rpi
- ACPI
- ATA
- Timer
- SD-card
- i.MX
Components
Components
1. Introduction

2. Architectural Principles

3. Showcases

4. Current Topics
Object capabilities

Delegation of authority between components

- Each component lives in a virtual environment
- A component that possesses a capability can
  - Use it (*invoke*)
  - Delegate it to acquainted components
Session creation

GUI

User Application

session("GUI", "input:read label:terminal")

User Session

session("GUI", "label:olaf.xterm; input:read")

Init

Core
Session creation

1. User Application
   
2. User Session
      
3. Init
      
4. GUI

- session("GUI", "input:read label:terminal")
- session("input:read label:olaf.xterm")
- session("GUI", "label:olaf.xterm; input:read")
- session("input:none; label:olaf.xterm")
Resource management

Explicit assignment of physical resources to components
Resources can be attached to sessions
1. Introduction

2. Architectural Principles

3. Showcases

4. Current Topics
OS-level Virtualization

Recompiled UNIX program

FreeBSD libc

libc plugin

Noux

Terminal

VFS

TarFS

I/O channels

ROM session

Terminal session

Init

Core

open read write
select ioctl
stat readdir

Noux session
1. Introduction

2. Architectural Principles

3. Showcases

4. Current Topics
Current Topics

- Eating our own dog food
  - Noux (GCC, VIM, bash, coreutils…)
  - Wireless networking

- Capability-based user interface

- seL4 kernel as base platform

- ARM Virtualization

- Package management
Thank you

Genode OS Framework
http://genode.org

Genode Labs GmbH
http://www.genode-labs.com

Source code at GitHub
http://github.com/genodelabs/genode