Linux Security
Tools & Technologies

Hadi Nahari
Chief Security Architect
MontaVista Software, Inc.
Hadi’s Background

- Security, Cryptography, Complex System Analysis
- Identity Management, Asset Protection
- Vulnerability Assessment, Threat Analysis
- Information Assurance Certifications (FIPS, CC, NSA)
- Enterprise & Embedded (Netscape, Sun Microsystems, United States Government, Motorola, ...)

Agenda

- General Introduction, Definitions
- Embedded Security Infrastructure
- Discussion, Questions & Answers
Bad Security
Worse Than
No Security!
• **Linux Is Highly Active In Embedded World**
  • **Embedded Linux Developers' Facts:**
    – >75% Of New Semiconductor Devices Are Linux-enabled
    – Emerging Professionals Are Linux-comfortable And -savvy
    – 100,000~150,000 Embedded Linux Developers
• Security Means Different Things To Different People
• Closed Source More Secure Than Open Source
• Security Could Be Achieved By Obscurity
• Software-Only Security Is Good (Enough)
• Security Staff Are Pain In The “Neck”
• Security Is A Set Of Components
• Can Protect Against All Attacks
• Encryption Equals Security
• Can Add Security Later
• Hackers Are Clueless
What’s Needed

- **Security Infrastructure Should Provide**
  - Static/Dynamic Security Asset Protection
  - Strong Authentication Mechanisms *(e.g. Secure Key Management)*
  - Access Control, Effective Containment *(Jailhouse)*
  - Secure Update Mechanism *(i.e. Verification Prior To Installation)*
  - Secure-Vault, Encrypted Filesystem
  - Virtualization/Container Security
  - Distributed Security Infrastructure

- **And Be**
  - Light-weight & High-performance
  - Simple, Flexible, Extensible
  - Layered & Scalable
Security Should Cover (@least!)

- **Privacy**
  - Confidentiality & Integrity
  - Message Security, Hash, MAC
  - Data Protection
  - Unique Device ID (TPM, HSM, etc.)
- **Networked- and Connected-devices Security**
  - USB, Firewire, BlueTooth, etc.
- **Protocol Security**
- **Threats**
  - Threat Model
- **Cryptography**
• VATA (Vulnerability Assessment & Threat Analysis)
  – VATA: 3W Rule
    • What Are The Security Assets?
    • What Threats To Protect Against?
    • What Defense Mechanisms Are Already In Place?

• Embedded Systems’ Threat Model
  – Less or No Control Over TOE (Target Of Evaluation)
  – Conceptually Similar To Non Embedded
  – Case-By-Case Evaluation Needed
• Secure Channel
• Key Negotiation & Exchange
  – DH
• Mutual Authentication
  – Zero Knowledge
  – RSA
• Key Management
• Prime Numbers
• Generating Randomness
  – RNG
  – PRNG
On Cryptography

- The Role Of Cryptography
  - Cryptography != Security
- The Weakest Link Property
- The Adversarial Setting
  - Practical Paranoia
  - Attacks
  - Threat Model
- Cryptography Is Not The Solution
- Cryptography Is Very Difficult
- Cryptography Is The Easy Part
Agenda

- General Introduction, Definitions
- Embedded Security Infrastructure
- Discussion, Questions & Answers
Embedded Security Infrastructure

- High Assurance Computing Environment
- Flexible Mandatory Access Control
- Trusted Secure Isolation
- Security Governance
High Assurance Computing Environment

• **Trusted Computing Environment Via**
  – Chain Of Trust Based On HW Root Of Trust
  – Trusted Platform Module (TPM)
  – Hardware Security Module (HSM)
• **Different Designs Defend Against Different Attacks**

• **Access Control Is Necessary At Runtime**

• **Continuous Protection Is The Key**
  – Hardware-rooted Security Necessary

• **Challenges**
  – No Standardized H/W Implementation (due to IP?)
  – No Standard API For Applications/Frameworks
Establishing Trust (cont’d)

Chain of Trust

- Hardware Security Device (HSM, TPM, etc.)
- SecureBoot Code
- Integrity Checking Of Basic Parameters (e.g. Public Keys)
- Validating System Images (kernel, libs, etc.)
- Applications
The “Vegas” Principle

Applications

Validating System Images (kernel, libs, etc.)

Integrity Checking Of Basic Parameters (e.g. Public Keys)

SecureBoot Code

Chain of Trust

What Happens In TPM, Remains In TPM

Trusted Platform Module
Processes Runtime Confinement

Applications

Validating System Images (kernel, libs, etc.)

Integrity Checking Of Basic Parameters (e.g. Public Keys)

SecureBoot Code

Hardware Security Device (HSM, TPM, etc.)

Access Control

Additional Services?

TPM Services

Hardware Root of Trust

Processes Runtime Confinement

Chain of Trust
High Assurance Computing Environment
Flexible Mandatory Access Control
Trusted Secure Isolation
Security Governance
• **Security Enhanced Linux**
  – SELinux Overview
• **Effective Confinement**
  – Isolation Via Security Domains
• **Mandatory Access Control**
  – As Opposed To What?
• SELinux: Security Enhanced Linux
• Developed by NSA, Became Open Source in 2000
• Based on FLASK Security Architecture
  – FLASK: Flux Advanced Security Kernel
  – Flexibly Controlling Security Policy & Propagation of Access Rights
• Based on TE (Type Enforcement) & RBAC (Role Based Access Control)
• Uses *xattrs* To Tag Objects & Make Security Decisions
• Solid Design; Heavily Scrutinized By The Community
• Provides Isolation via **Security Domains**
• Network-aware
• Configuration Non-trivial: Devising A Correct **Policy** is key
Separation Of “Policy” & “Mechanism”
SELinux Protection Mechanisms

- Implements Isolation via **Security Domains**
  - Permissions & Operations Define a Security Domain
  - "Read/Write permission:home_t files" is a Security Domain

- Includes TE & RBAC Mechanisms
  - TE (Type Enforcement)
    - Mechanism to Enforce Security Domain Rules
  - RBAC (Role Based Access Control)
    - Limits Users’ Access to Security Domains

- Defines **Security Contexts & Domain Transitions**
  - Security Context: Specification of Object’s Permissions
  - Transition: Performing an Operation **On-behalf** of an Object
Isolation (aka *Sandboxing*) via Security Domains

Without SE Linux all server programs run with ultimate access to the system. Only the Administrator needs ultimate access, but all servers get it.

With SE Linux each server program gets limited access to the system. Servers are isolated from each other and prevented from harming the rest of the system. Only the Administrator has ultimate access.
SELinux Implements RBAC

Role 1

Permission 1

Role 2

Permission 2

Role 3

Permission 3
SELinux Component Diagram

- Coreutils (SELinux-aware)
- SELinux LSM
- LSM Framework
- SELinux Policy Manager
- Security Policy
- Other Kernel Modules
- Framework / Middleware
- Access Mediation
- Stand-alone App.
- App. 1
- App. n

Security Governance Tools

Hardware Architecture

MontaVista

Existing

SELinux Mandatory Access Control Perimeter
• Security Enhanced Linux
  – SELinux Overview
• Effective Confinement
  – Isolation Via Security Domains
• Mandatory Access Control
  – As Opposed To What?
Unix Permissions: Not Good Enough

• Not Fine Grained Enough
• Permissions Are Primarily File Based
• Permissions Are Controlled By Users
  – As Opposed To Administrator
• Hard To Set Up Access To The Same File:
  – For Multiple Users
  – For Multiple Groups
**DAC vs. MAC**

- **Discretionary Access Control (DAC)**
  - Defines Basic Access Control On *Filesystem Objects*
  - Used & Enforced At The Discretion Of Object Owner
  - Applies to *Users*
  - Implements “*Allow Unless Explicitly Denied*” Access Concept
  - Users Entirely Determine Access Rights Of Object They Create

- **Mandatory Access Control (MAC)**
  - Defines System Security Policy For *All Objects*
  - Applies to *Agents* (Programs, Processes, etc.) Not *Users*
  - Applies to Objects That DAC Doesn’t
    - *(e.g. Network Sockets)*
  - Enforcement Of Security Policy Is Mandatory
  - Implements “*Deny Unless Explicitly Allowed*”
  - Administrator Determines Access Rights
• Modular
• Can Be Compiled As:
  – Strict
  – Targeted
  – MLS (Multi Level Security)
  – MCS (Multi Category Security)
• Can Be Reconfigured At Runtime Via Booleans (partly)
• Administrators May Be Allowed Access To Update:
  – All Policy
  – Only Parts Of Policy
• SELinux Management Requires New Tools
• Governance Is The Focus Area
  – MontaVista Security Governance Tools
• GUI Tools To Help Troubleshoot Denials
  – SEtroubleshooter
• GUI Tools To Help Write Policies
  – SLIDE, Bedrock
• GUI Tools For Analyzing Policies
  – Apol
• High Assurance Computing Environment
• Flexible Mandatory Access Control
• Trusted Secure Isolation
• Security Governance
Trusted Secure Isolation Requirements

- **Trusted Secure Isolation Should Provide**
  - Execution Segregation: Running Trusted Code
    - Along With Untrusted Code
    - Inside Untrusted Environment
  - Security Controls *Within* VMM
    - Fine Grained Enough To Guarantee Isolation
    - Coarse Grained Enough To Not Affect Performance
  - GPL Jailhouse
    - Adoption By Close Source
• `security_context(Dom_n_id)`
  – Lacks Individual Application Identification Within a Domain

• `security_context(Dom_n_id, App_m_id)`
  – Individual Applications Within a Domain Identified
  – But Who Handles
    • Identity Management?
    • Access Control Definition & Enforcement?
  – What's The Mediation Mechanism Across Domains??
    • Who Arbitrates & Attests The Identities?
    • Hypervisor? Could It Still Be Considered “Thin”?
Hardware Architecture

Virtual Machine Monitor (aka “Hypervisor”)

Isolation: What’s Missing?

- Minimal Security: Only MMU
- No Secure Isolation
- No VMM Access Control
- No Secure Communication
- No Secure Services
- No VM Mediated Sharing
- No Attestations by VM
- No Integrity Guarantees
Virtualized Chain Of Trust ;-)
• Access Control Granularity Is Important
• IBM's sHype
  – A Step In The Right Direction
  – Available On Xen
  – VMWare ESX & MS Viridian Likely To Adopt Same Style
  – Not Fine-grained Enough
  – More Work Needed: (Mainline?)
• XSM (Xen Security Modules)
  – By NSA/NIARL
  – Includes: FLASK, ACM (sHype), dummy (default)
  – FLASK Module: Fine-grained, SELinux-like MAC
  – No Precedence in Embedded (Yet!)
• High Assurance Computing Environment
• Flexible Mandatory Access Control
• Trusted Secure Isolation
• Security Governance
Security Governance: Idea & Tools

- Manage SELinux Infrastructure
- Cross-development Environment (Host ↔ Target)
- Develop, Debug, Analyze & Provision Policy Remotely
- SELinux API Integration With IDE
- Unified Governance For TPM/HSM
• Discussion
• Q&A

Thank You!