A Generic Architecture and Extension of eCryptfs: Secret Sharing Scheme, Smartcard Integration and a new Linux Security Module

Daniel Bußmeyer\(^2\), Benedikt Driessen\(^1\), André Osterhues\(^1\), Jan Pelzl\(^1\), Volker Reiß\(^1\), Jörg Schwenk\(^2\), Christoph Wegener\(^2\)

\(^1\)escrypt GmbH – Embedded Security, Bochum
\(^2\)Chair for Network and Data Security (NDS), Ruhr-University Bochum
Overview

- Introduction
- Generic security architecture
- Linux Security Module – esCAP
- Integration of eCryptfs
- Secret Sharing Scheme
- Smartcard Integration
- Implementation Details
- Conclusion
Goal: handle security-sensitive data in Linux environments

Encryption systems:

- File encryption systems:
  - GnuPG

- Device/partition encryption systems:
  - DM-Crypt, TrueCrypt, eCryptfs

Security often depends on strength of chosen password
Introduction – Problems

- Weak passwords -> vulnerable to dictionary and/or social engineering attacks
- Single key for single user -> single point of failure
- User can (accidentally) write data to insecure places (USB sticks, email)
- System admin (superuser) can access keys (using exploits, tracing/debugging processes etc.)
- Weak passwords: use **smartcards** instead

- Single key for single user: **secret sharing scheme**

- User can (accidentally) write data to insecure places (USB sticks, email): restrict user by applying access control using a **Linux Security Module**

- System admin (superuser) can access keys: restrict root user by applying access control using a **Linux Security Module**
Generic Security Architecture

- Smartcard 1
- User 1
- Client 1
- Server
- Smartcard 2
- User 2
- Client 2
- User 3
- Smartcard 3
- Server
- Admin
Generic Security Architecture

- Security-critical data is stored in encrypted form on a central server

- Limit superuser:
  - Still can administer most services and infrastructure (e.g., backups)
  - No access to security-critical data, keys or configuration files
  - Mandatory Access Control (MAC) mechanism called “esCAP”

- Device encryption (eCryptfs) using symmetric keys
  - FEK: File encryption key, per file
  - FEKEK: FEK encryption key, per device/partition

- Asymmetric cryptography (RSA)
  - Encrypt symmetric keys (FEKEKs)
  - Based on smartcards, RSA private key never leaves smartcard
Linux Security Module – esCAP

- Mandatory Access Control system, in-kernel
- Subjects: tasks or processes
- Objects: tasks, keys or inodes
- Association: read/write access
- Rule: Subject $S$ may or may not read/write an object $O$
- Rules are set using esCAP’s procfs interface
- Fast interpretation of rules, virtually no performance penalty
Object-specific rules
- Defined at run-time by giving subject, object and association
- Control read/write access, signals, debugging
- Limit access (read/write/search) to kernel keyrings

Special case: File “firewall”
- Notification on file access
- Applet forwards notification to user
- Generation of dynamic rule depending on user decision

Global rules
- Defined at startup
- Enable/disable module loading
- Enable/disable raw sockets
Integration of eCryptfs

- Wrapper library
  - Attach symmetric key to user’s keyring
  - Remove a key from user’s keyring
  - Mount a directory
  - Unmount a directory

- PKI module for eCryptfs
  - AES Key Wrap algorithm [NIST 2001]
  - Encrypt/decrypt symmetric file keys (FEKs) using a symmetric directory key (FEKEK)
  - FEKEK is encrypted with the smartcard’s public key (RSA-2048)
  - FEKEK can only be decrypted using the private key, which remains on the smartcard
Secret Sharing Scheme

- Idea: distribute a secret (key) among a group of $n$ users
- Secret is split into $n$ parts
- Threshold $k$ with $2 \leq k \leq n$: amount of users required to reconstruct the secret
- Used for emergency file access in our system:
  - For each new directory, a secret sharing group and threshold $k$ is defined
  - The directory’s FEKEK is split among the secret sharing users
  - In an emergency case, $k$ of the users can reconstruct the secret and access the directory
Smartcard Integration

- **Generic smartcard interface:**
  - Decryption
  - Signature generation
  - Read bytes from random number generator (RNG)
  - Read/write files

- **Smartcard requirements:**
  - Asymmetric cryptography (RSA decryption and signing)
  - Secure storage (for private key)
  - Minimal filesystem (for the public key and certificates)
  - True random number generator (TRNG)

- **Supported smartcards:**
  - eDA (elektronischer Dienstausweis, “electronic office ID card”)
Implementation Details – Overview

Client
- SmartCard
- GUI
- DirCache

Control

Server
- CertStore
- DirHeader
- FileSystem
- Secret Sharing
- FireWall
- esCAP

Daemon

File Server
- eCryptfs
- eCryptfs PKI Module

TCP/IP

NFS / local

procfs

procfs

esCAP Kernel Module

esCAP Kernel Module
Implementation Details – Client-side Modules

- **Control:** Central module, message passing
- **GUI:** Graphical User Interface
- **DirCache:** Cache of directory-related information from DirHeader
- **Smartcard:** Smartcard interface
- **esCAP Applet:** GUI for “file firewall” function of esCAP
Implementation Details – Server-side Modules

- Daemon: Central module, XML script interpretation and message passing
- CertStore: User certificate storage
- DirHeader: Information about each directory
- DataBase: List of user’s directories
- Secret Sharing: Secret sharing implementation
- FireWall: Netfilter configuration, dynamic rules
- esCAP: Interface to esCAP kernel module
- FileSystem: Interface to eCryptfs
Implementation Details

- **Programming language:**
  - C++ for framework and modules
  - C for esCAP and eCryptfs PKI module

- **Message flow:**
  - Definition of use cases
  - UML 2.0 sequence diagrams
  - XML scripts
  - Small and simple XML parser
  - Command interpreter in Daemon
  - Easy adoption of use cases by changing the XML script

- **Hardware:**
  - Standard smartcard readers
Conclusion

- **Software suite:**
  - User-friendly GUI
  - Command-line tools for security administrator and system administrator
  - Based on a Fedora 10 distribution
  - Slightly modified Linux 2.6.26 kernel (patches include esCAP and eCryptfs modifications)

- Demonstrator already available, prototype in near future

- Project homepage: [http://sourceforge.net/projects/esosi](http://sourceforge.net/projects/esosi)

- License: LGPL