SELinux

Security Enhanced Linux

Introduction and brief overview.

Agenda

- DAC – Discretionary Access Control
- ACL – Access Control Lists
- MAC – Mandatory Access Control
- SELinux
  - History
  - FLASK
  - Details and implementation
- Examples

Discretionary Access Control

- Benefits
  - Fast
  - Robust
  - Well known
- Limitations
  - Risky control over the permissions
    - Error prone
  - Power-users vs. normal users

DAC - continued...

- Examples
  - chmod 777 /etc/shadow
  - Binding to protected ports (<1024)
  - Full control over user's files
    - Compromised applications
    - setuid/setgid

Access Control Lists

- Supersedes DAC in the area of FS permissions
- Imposes overhead
- More complicated than DAC
- Applies to FS permissions only

Mandatory Access Control

- Least privilege approach (opt-in)
- All available information is concerned

SELinux

- Security Enhanced Linux
- Originally developed by the NSA
- LSM – Linux Security Modules
  - Object oriented security
  - Present in 2.6 Linux kernel tree
  - SELinux – inspiration and the main reason
- Type Enforcement™ (TE) & RBAC

SELinux – data storage

- Persistent Security IDs (PSIDs)
  - Unused part of an inode in the ext2 FS
  - Flat-file storage
- LSM xattrs (extended attributes)
  - getfattr
  - ext3, xfs, ReiserFS
  - Coexistence of multiple security modules
  - SELinux being reference implementation

Fundamentals

- Subjects
  - Processes
- Objects
  - Resources
    - Files
    - Devices
    - Sockets
    - Ports
    - Processes
    - Etc.

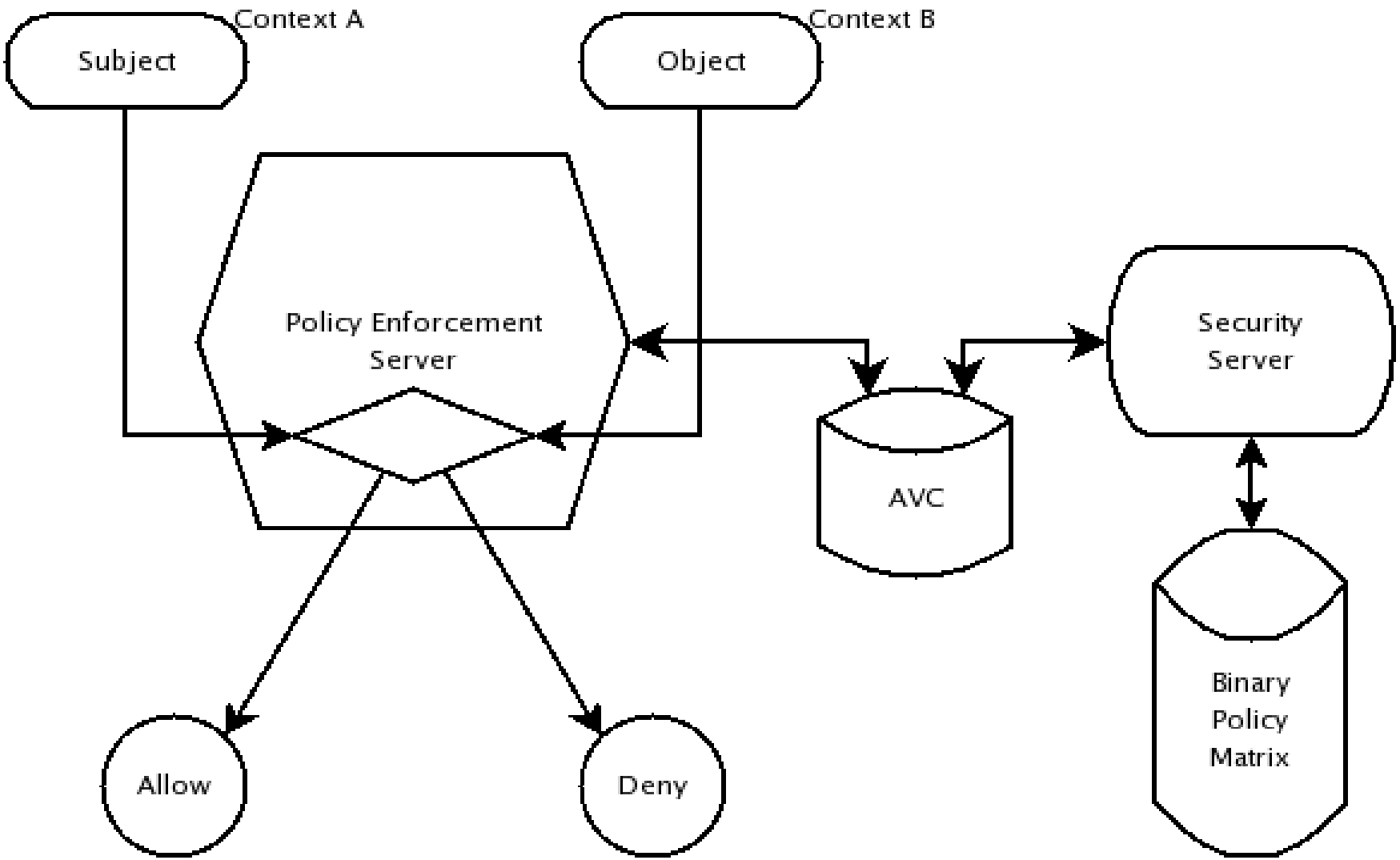
MLS

- Multi Level Security
  - No data integrity
  - No least privilege
  - No processes and object duty separation

FLASK

- Security Server
  - Security policy logic
  - Security contexts
- Access Vector Cache

FLASK – general principles



FLASK – operation

- Considered at the operation attempt
- Security context are sent to the AVC
- AVC check
  - Cache driven
  - Misses relayed to the SS
- Enforcement Server (kernel) receives the decision and allows or denies the operation
- Populating audit log (if applicable)

FLASK vs. pure MLS

- No rigidly defined lattice of relationships
- Defining security labels based on
  - user identity (UID)
  - role attributes
  - domain or type attributes
  - MLS levels
  - ...

Security contexts

- Also known as security labels
- General
  - <user>:<role>:<type>
- Example
  - system_u:system_r:crond_t

SELinux and FLASK

- No distinction between a type and a domain
  - Domains have the process attribute
- Security server, AVC and the policy engine are incorporated into the kernel
- Domain-type access control w/ role-based limiting

Policies

- Set of rules that guide the security engine
- Defines types (resources) and domains (processes)
- Uses roles to limit domain transients
- A domain is akin to a type whenever we consider processes

Types

- Groups together connected resources
- Abstraction layer for the functionality
  - etc_t

Boot up process - 1

- Kernel load
  - Initial process gets predefined SID (kernel)
    - No policy loaded yet!
- Mounting /proc
  - Checks /proc/filesystems for selinuxfs
- Mounting /selinux
- Check /selinux/policyvers
- Check /etc/selinux/config for the policy flavour

Boot up process - 2

- In case of troubles – fall back to old policy
- Remap SIDs into contexts
- /sbin/init re-executes itself
- Normal bootup

TE Rules – Access Vectors

- <av_kind> <source_type(s)> <target_type(s)>:<class(es)> <permission(s)>
- allow named_t sbin_t:dir search;

AVC denied messages

- type=AVC msg=audit(1133209488.535:344): avc: denied { getattr } for pid=4198 comm="httpd" name="index.html" dev=dm-0 ino=3438923 scontext=root:system_r:httpd_t tcontext=system_u:object_r:httpd_private_content_t tclass=file

AVC – continued...

- type=AVC
- msg=audit(1133209488.535:344):
- avc: denied { getattr }
- for pid=4198
- comm="httpd"

AVC – continued...

- name="index.html"
- dev=dm-0
- ino=3438923 scontext=root:system_r:httpd_t
- tcontext=system_u:object_r:httpd_private_content_t tclass=file

File contexts

- regexp <-type> (<file_label> | <<none>>)
- /bin(/.*)? system_u:object_r:bin_t
- /etc/shadow.* -- system_u:object_r:shadow_t

Targeted vs. Strict

- Only selected subjects are concerned
- Easy to implement
- Non-standard applications

Examples

- Accidental chmod usage
  - /etc/shadow
  - user directory
- Compromised program
  - Port binding
  - Port connection

Bibliography

- Security-Enhanced Linux
  - http://www.nsa.gov/selinux/
- Red Hat SELinux Guide
  - http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/selinux-guide/
- Fedora Core 3 SELinux FAQ
  - http://fedora.redhat.com/docs/selinux-faq-fc3/
- The UnOfficial SELinux FAQ
  - http://www.crypt.gen.nz/selinux/faq.html
- Getting Started with SE Linux HOWTO: the new SE Linux
  - https://sourceforge.net/docman/display_doc.php?docid=20372&group_id=21266
- Writing SE Linux policy HOWTO
  - https://sourceforge.net/docman/display_doc.php?docid=21959&group_id=21266
- SELinux, Kerry Thompson
  - http://www.samag.com/documents/s=7835/sam0303a/0303a.htm

Questions?

The End!

Legal statement

You may not publish this document in any form possible without a written permission of the author.

The most recent version of this presentation is always available at the following address:

http://pjs.name/papers/

Author: Paweł J. Sawicki <pjs@pjs.name>