Improving Linux Security

M.B.G. Suranga De Silva
Information Security Specialist
TECHCERT
c/o Department of Computer Science and Engineering
University of Moratuwa.

Founder of http://www.ceylonlinux.com

Agenda

- How to secure Linux
- Hardening Linux
- Disable Unwanted Services
- Memory Protection
- DAC and MAC
- LSM Architecture
- SELinux – what is it?
- Processes and Domains
- Security Server
- Type Enforcement
- Role Base Access Control
- Access Vector Cache
- kernel & File System Integrity
- process separation
- Holistic View of a secure Linux OS

How to secure Linux OS

- Only enabling and configuring just wanted services, and patching those services accordingly. This is known as operating system hardening.
- Improve access control mechanism
- Process separation– Vulnerability in one should not lead to compromise of all

Hardening Linux

- Remove or disable all unnecessary software
- Patch Linux System against vulnerabilities
- Close all unnecessary network ports and disable all unnecessary runlevel system services and xinetd services if there are any
- Delete or disable all unnecessary user accounts
- Review inittab and boot scripts
- Configure to run each public access services in a chrooted filesystem
- Enable password aging, enforce strong passwords, restrict the use of previous passwords

Hardening Linux (cont.)

- Lock user accounts after many failure login attempts
- Identify and eliminate needless SUID bit settings
- Restrict direct login access for system and shared accounts
- Restrict root login over the network
- Restrict su access to system and shared accounts
- Configure logging and check logs regularly
- Configure a host-based firewall
- Mount Linux partitions where executables are residing in read-only mode under normal operation
- Enable Memory Protection

Disable Unwanted Services

- find / -perm +4000 -user root -type f -print
- find / -perm +2000 -group root -type f –print
- chkconfig --list | awk '/3:on/ { print $1 }'
- for DISABLE_SERV in gpm kudzu netfs atd apmd pcmcia nfslock isdn portmap rhnsd

  do

  /sbin/chkconfig $ DISABLE_SERV off

  /sbin/service $ DISABLE_SERV stop

  done
- rpm –e <package name> or yum remove <package name>
- Finally make sure to execute yum update

DAC and MAC

- Most operating systems have a built-in security mechanism known as access control. We can consider them as Discretionary Access Control (DAC) and Mandatory Access Control (MAC).

DAC

- DAC – Discretionary Access Control

DAC is used to control access by restricting a subject's access to an object. It is generally used to limit a user's access to a file. **In this type of access control it is the owner of the file who controls other users' accesses to the file.**”

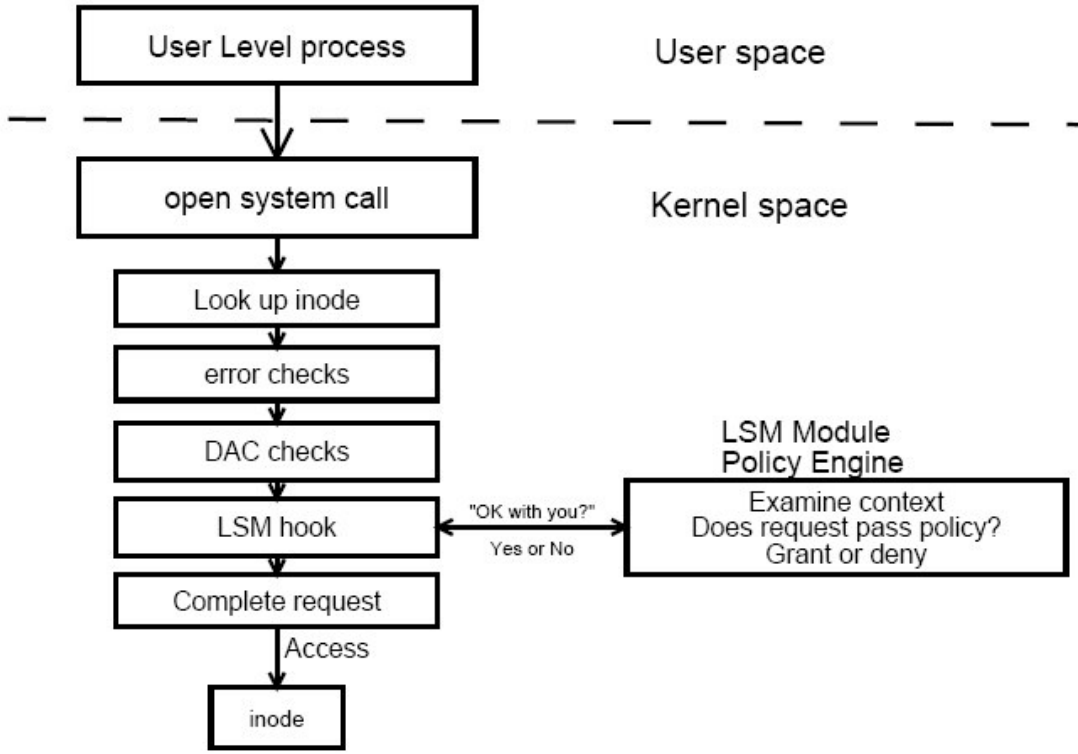
Ex: ls –l

-rw-rw-r– 1 suranga suranga 2645 Feb 12 08:48 personnel.txt

MAC

- MAC is much more effective than DAC because MACs are often applied to agents other than users.
- MACs cannot be overridden by the owner of the object.
- MACs may be applied to objects not protected by ordinary unix style DACs such as network sockets and processes.
- The other advantage is it makes data flow control possible which was impossible in DAC.

LSM Architecture (Linux Security Module)



SELinux – What is it ?

- Released by NSA September, 2001
- Based on previous research projects (FLASK OS)
- Integrated with Linux Security Module (LSM)
- Adopted into 2.6 kernel series.
- Type Enforcement (TE) rules – which subjects can access which objects.
- Role-Based Access Control (RBAC) – which roles users can adopt and what they can do.
- Provides fine-grained controls and operation on files, sockets and processes.

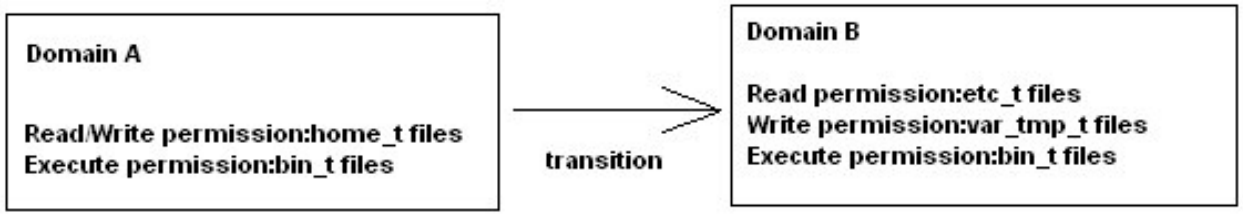
Processes and Domains

- A process running with a specific security context is said to be running within a domain (process with a sandbox)
- Each domain is assigned only sufficient permissions to properly function but do nothing else.

- Rules are configured to:

– Specify which objects a domain can access, and how

– Specify which roles a domain can transition to



Security Server (SS)

- The security policy decision logic is embedded to a new kernel component known as *Security Server* (SS).
- SS makes labeling, access and transition decisions.
- Each file is labeled with information called *security context*.
- S*ecurity context* is a data type and it can only be interpreted by the *Security Server.*
- SS maintains the *Security Context* with three security attributes known as *identity, role* and *type*.

Type Enforcement (TE)

- Clearly define which subjects can access which objects, and how
- Define domain transitions.

- ex: init run-control processes are in initrc_t domain. When init starts web server process it shouldn't be in that domain but http_t domain.

- Permissions are encoded as access vectors.
- Written in plain text, processed by the m4 macro processor.

Role Based Access Control (RBAC)

- Which roles users can adopt and what they can do.
- Works along with Type Enforcement
- Users are assigned roles by user statement

  -ex: user fossed roles { staff_r sysadm_r};
- Transition between roles are governed by allow statement

  -ex: allow staff_r sysadm_r;
- Roles are authorized to enter domains by the role statement.

  -ex: role sysadm_r types ifconfig_t

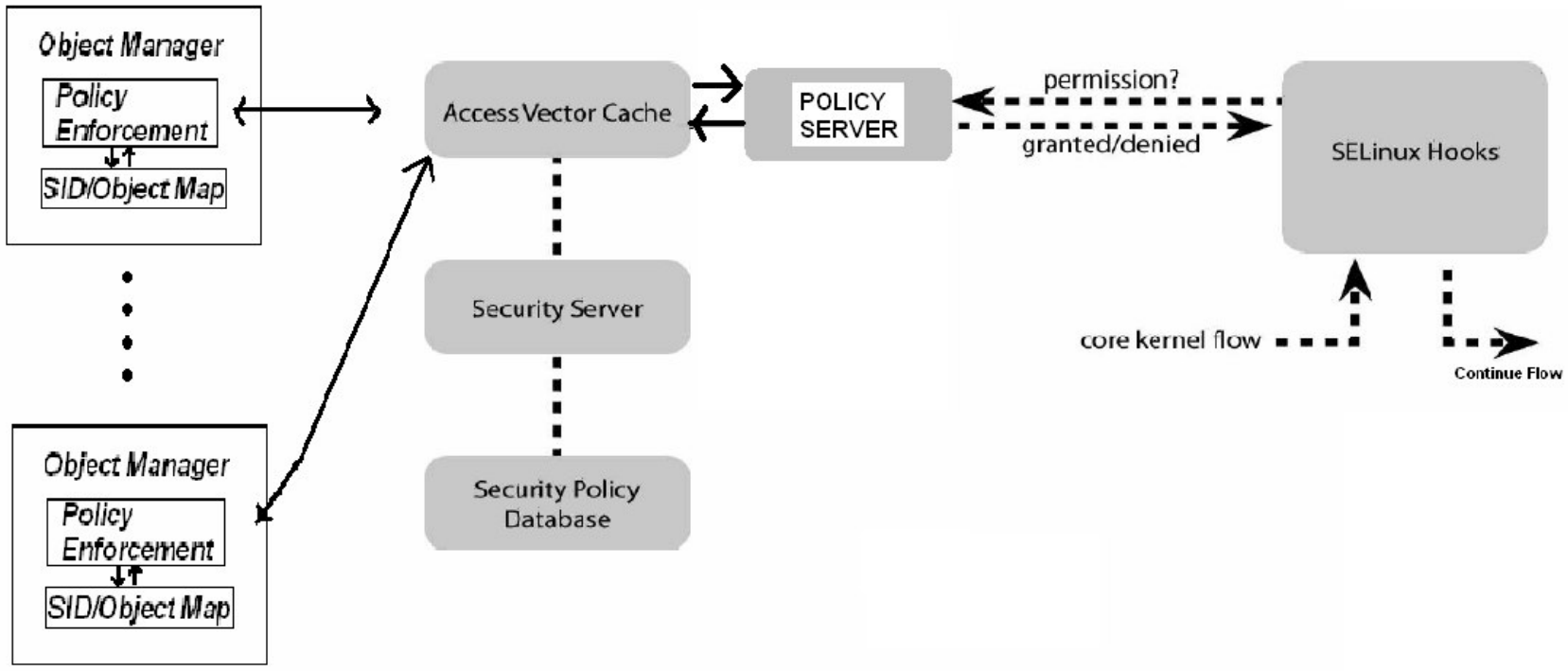
Access Vector Cache (AVC)

- To improve the efficiency of SELinux operation, the Security Server caches access vectors in a data structure called *Access Vector Cache*.
- The *Access Vector Cache* stores past SS policy requests/responses.

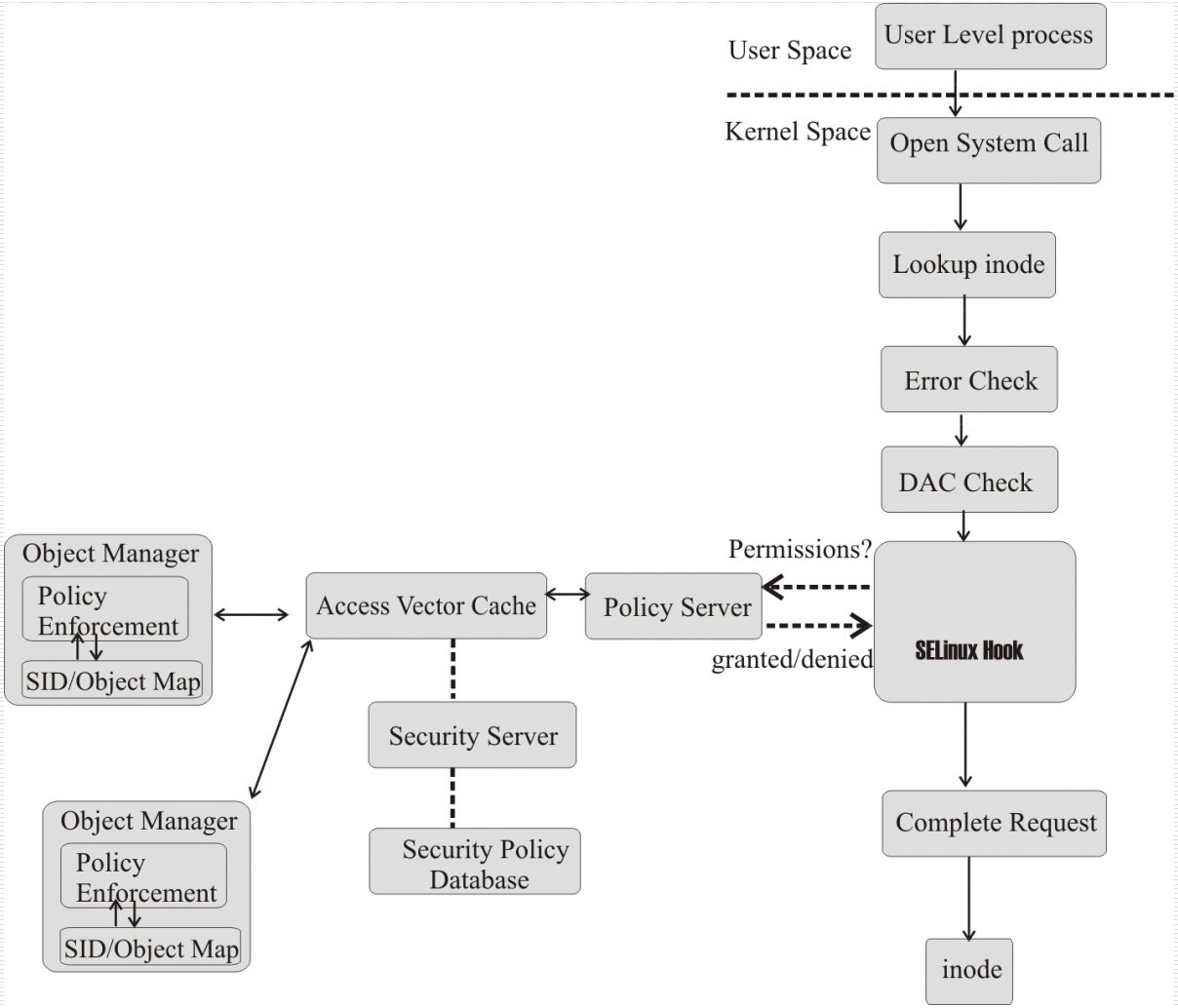
SELinux's Object Managers

- Object management includes labeling objects with a security context, managing object labels in memory.
- Object managers are there to obtain security policy decisions from the security server and to apply the decisions to label and control access to their objects.

SELinux in a Diagram



SELinux Complete Diagram



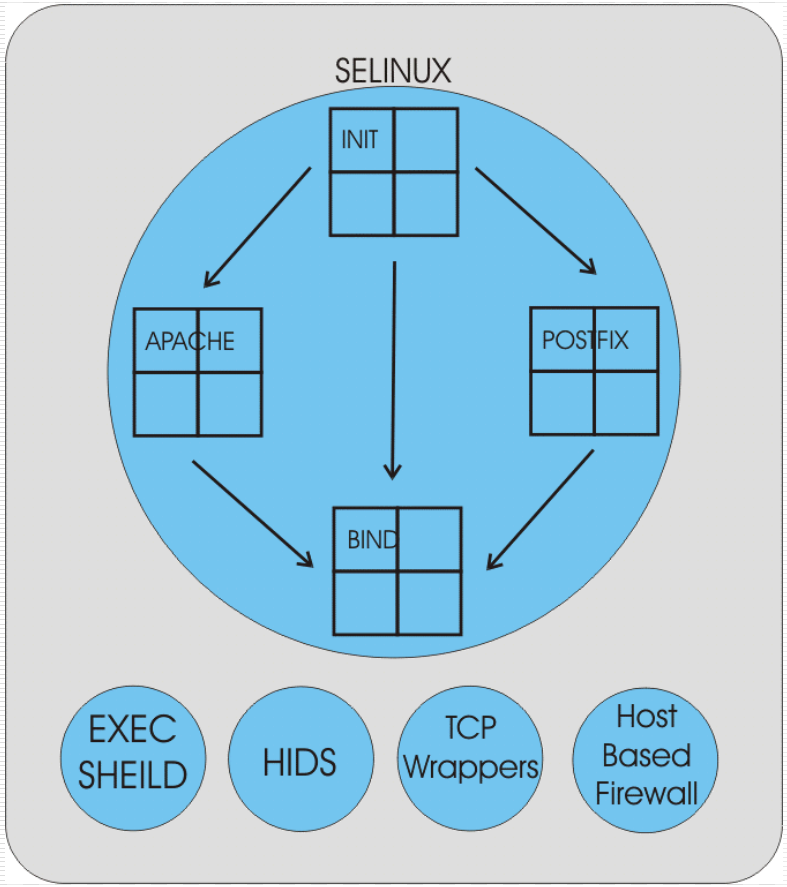
SELinux Operation

1. The policy server gathers the security context from the subject and object, and sends the pair of labels to the security server, which is responsible for policy decision making.
2. The policy server first checks the AVC, and returns a decision to the enforcement server.
3. If the AVC does not have a policy decision cached, it turns to the security server, which uses the binary policy that is loaded into the kernel during initialization. The AVC caches the decision, and returns the decision to the policy server.
4. If the policy permits the subject to perform the desired operation on the object, the operation is allowed to proceed.
5. If the policy does not permit the subject to perform the desired operation, the action is denied, and one or more avc: denied messages are logged to $AUDIT_LOG, which is typically /var/log/messages.

Process Separation

- Policy configuration can restrict the interference by a process in one domain to a process in the other domain.
- SELinux has the ability to trace other processes or send signals to other processes in the same domain.
- Ex: Sending SIGCHILD to notify the parent of the completion of the child process is ALLOWED
- BUT when a process signals SIGKILL on all other processes is NOT ALLOWED

Complete View of a secure Linux OS



Further Reading

- For a description of the policy language syntax as well as an example policy refer to [1]
- For a set of some object classes and permissions refer to [2]

References

- *[1] C. Wright, et al., "Linux Security Module Framework,"presented at The Ottawa Linux Symposium 2002, June 2002.*
- *[2] NSA Security-Enhanced Linux http://www.nsa.gov/selinux/*
- *[3] P. Loscocco and S. Smalley, "Integrating Flexible Support for Security Policies into the Linux Operating System", 2001 USENIX Annual Technical Conference*
- *[4] "Common Vulnerability and Exposure", available at:http://cve.mitre.org, [last access: Nov 27,2006]*
- *[5] M.D.Bauer., "Building secure servers with Linux", 2003 O'Reilly & Associates.*
- *[6] Mehmet Sahinoglu, "Security Meter: A Practical Decision-Tree Model to Quantify Risk," IEEE Security and Privacy, vol. 03, no. 3, pp. 18-24, May/Jun, 2005.*
- *[7] M. Schiffman et al., Common Vulnerability Scoring System, tech. report, US Nat'l Infrastructure Advisory Council, 2004; www.dhs.gov/interweb/assetlibrary/NIAC_CVSS_FinalRpt_12-2-04.pdf.*
- *[8] Vladimir Kiriansky, Derek Bruening, and Saman Amarasinghe. Secure Execution Via Program Shepherding In Proceedings of the 11th USENIX Security Symposium, San Francisco, August 5, 2002.*

THANK YOU!!!

- Any Questions?

Kernel and File System integrity

- **Protecting Kernel Integrity**

Most of /boot files are labeled with boot_t type and can only be modified by an administrator.

- **Protecting System File Integrity**

Separate types are defined and assigned to system files

Ex: The Dynamic Linker is labeled with the ld_so_t type

System programs are labeled with type bin_t

System Administration Programs are labeled with sbin_t

- Write access to these types is limited to administrator.

Security Context and SID

- Two security label data types (used in SS):

1. Security ID (SID) – An integer mapped to security context
2. Security context – A string that represents the security level

- *Security context* contains all of the security attributes associated with a particular labeled object.
- *Security Identifier* (SID) is directly bound to the object.
- SID is mapped with *Security context.*
- The mapping is created at run time and maintained by the S*ecurity server*.
- SIDs associated to the new file is send to SS.
- The Object Managers are responsible for associating SIDs to objects.

SELinux-aware Applications

- Many basic Linux commands have been modified to be SELinux-aware

– login, ls, ps, id, cron

Ex:su - root

id -Z

root:system_r:unconfined_t

useradd shantha

ls -Z /home

drwx------ shantha shantha root:object_r:user_home_dir_t /home/shantha

- Other applications patched for SELinux

– OpenSSH

- Additional commands added to perform SELinux functions

– chcon, restorecon , fixfiles etc.

- Tresys GUI tools for managing policies
- Backup be careful !!!

Examples

- **ls -aZ /home/suranga**

  drwx------ suranga suranga root:object_r:user_home_dir_t .
  drwxr-xr-x root root system_u:object_r:home_root_t ..
  -rw-r--r-- suranga suranga user_u:object_r:user_home_t anaconda-ks.cfg-
  rwxr-xr-x suranga suranga user_u:object_r:user_home_t anaconda.log...

- **sudo mv /home/suranga/about.html /var/www/html**
- **ls -aZ /var/www/html/**

  drwxr-xr-x root root system_u:object_r:httpd_sys_content_t .
  drwxr-xr-x root root system_u:object_r:httpd_sys_content_t ..
  -rw-r--r-- root root system_u:object_r:httpd_sys_content_t index.php
  -rw-r--r-- suranga suranga system_u:object_r:user_home_t about.html

- Oct 19 17:54:59 hostname kernel: audit(1098222899.827:0): avc: \denied { getattr } for pid=19029 exe=/usr/sbin/httpd \path=/var/www/html/about.html dev=dm-0 ino=373900 \scontext=root:system_r:httpd_t tcontext=user_u:object_r:user_home_t \tclass=file

- **chcon -t httpd_sys_content_t /var/www/html/about.html**

- **ls -aZ /var/www/html/**

  drwxr-xr-x root root system_u:object_r:httpd_sys_content_t .
  drwxr-xr-x root root system_u:object_r:httpd_sys_content_t ..
  -rw-r--r-- root root system_u:object_r:httpd_sys_content_t index.php
  -rw-r--r-- suranga suranga system_u:object_r:httpd_sys_content_t about.html