

---

# Example Trusted OS

CS498SH – Information  
Assurance

Fall 2006

Susan Hinrichs

# Overview

---

- Examine several Trusted Operating Systems
- Compare MAC mechanisms
- Computer Integrity mechanisms
- Identify tricky bits that arise in real systems

# Reading

---

- Pitbull Foundation – Traditional BLP
  - [http://www.argus-systems.com/public/docs/AIX/foundation\\_admin\\_guide.pdf](http://www.argus-systems.com/public/docs/AIX/foundation_admin_guide.pdf)
- SE Linux Type Enforcement
  - <http://www.nsa.gov/selinux/papers/policy2.pdf>
- SE Linux MLS – BLP mechanism
  - <http://selinux-symposium.org/2006/papers/03-SELinux-and-MLS.pdf>
- SE Linux – Multi-Category Security (MCS)
  - <http://james-morris.livejournal.com/5583.html>
- Pitbull LX – Compartment oriented MAC
  - [http://www.argus-systems.com/public/docs/AIX/PBLX20/lx\\_admin\\_guide.pdf](http://www.argus-systems.com/public/docs/AIX/PBLX20/lx_admin_guide.pdf)
- Data General B2 Unix System
  - Computer Security text 5.2.2

# Common points

---

- All implement kernel level access control
- All implement MAC
  - Plus additional DAC mechanisms (MCS)
- All keep base OS DAC

# Pitbull Foundation

---

- Two types of Mandatory Controls
  - Mandatory Access Control (MAC)
    - implemented as BLP
  - Also have Mandatory Integrity Controls (MIC)
    - implemented as strict Biba
- Subjects have 6 labels each
  - Max, Min, effective Sensitivity Label (SL)
  - Max, Min, effective Trust Label (TL)
  - Max must dominate min
- Directories have range of labels

# Pitbull Foundations

---

- Initial process and file labels must be explicitly enumerated
- Assume an Information System Security Officer (ISSO) to configure

# SE Linux MLS

---

- Implemented in addition to type enforcement
- Supports MAC as BLP
- Many of the same issues as Pitbull Foundations

# Problem 1 – common directories

---

- Programs running at TopSecret and Unclassified both need to write to /tmp
  - Could re-write all programs...
- Introduce partitioned or poly-instantiated or multi-level directories

# Multi-level or Poly-instantiated Directories

---

- Can be implemented as a nested directory
  - Special top level multi-level directory
  - Sub directory for each level
- Normal users see only one directory with the contents at their own level
- SE Linux plays with filesystem namespace
  - Hooks with PAM to unmap “wrong” versions from users' namespace

# Problem 2 – exceptions to BLP

---

- Normal users should follow BLP
- Need special users and programs
  - Administer labels and labeling
  - Declassify data
  - Change effective sensitivity labels
  - Backup and restore data
- Special user in vanilla Unix is root
  - Root bypasses all restrictions

# Privileges for Exceptions

---

- Foundation uses privileges
  - Essentially splits root's bypasses into more refined elements
  - Least privilege
  - Privilege hierarchy
  - PV\_ROOT, PV\_DAC, PV\_DAC\_R
- SELinux uses type attributes and constraint rules to model privileged exceptions

# Problem 3 – Constraining Privilege

---

- What privilege does a new process inherit?
  - Same privilege as calling process?
  - More privileges?
  - Fewer?

# Additional Labels to Constrain Process Privilege

---

- Label process with three sets
  - Effective Privilege – set of active privileges
  - Maximum privilege – set current process can activate
  - Limiting Privilege – Upper limit on privileges that can be inherited by child process. May include privileges not in maximum set.
- Label executable with three sets
  - Innate Privilege – Intersected with parent's limiting privilege set to get maximum set of new process
  - Proxy Privilege – Intersected with parent's maximum set to add to maximum set of new process
  - Authorized Privilege – Added to new process maximum set only if user is authorized

# Problem 4 – What about network data?

---

- What is the granularity of labeling network data?
- Can you exchange labels between trusted systems?
- How is the label of network data tracked between systems?
  - Label can be encoded in IP Options
    - Encoded using CIPSO
    - IP Options have routing problems
  - SE Linux has patch to label IPSec Security Association
    - Leverage SA negotiation to exchange labels.
  - Netrules refer to protocol port and interface
    - define label of outgoing traffic

# Labelled Network Data

---

- Label can be encoded in IP Options
  - Encoded using CIPSO
  - IP Options have routing problems
- SE Linux has patch to label IPSec Security Association
  - Leverage SA negotiation to exchange labels.
- Netrules refer to protocol port and interface
  - define label of outgoing traffic
  - Define label of incoming traffic for unlabeled network

# Problem 5: When are Checks Performed?

---

- Ideally the mandatory checks is performed on each object access
- Normally, checks correspond to file opens
  - If subject changes effective level some illegal accesses may be allowed

# Example Scenario

<b>Role</b>	<b>User</b>	<b>Clearance</b>	<b>Projects</b>
Project Manager	Alice	High	Proj1,Proj2, Proj3
Intern	Bob	Low	Proj1,Proj2
Dev Manager	Charles	High	Proj1

# Foundation Sensitivity Labels

<b>User</b>	<b>Sensitivity Label</b>
Alice	High:Proj1,Proj2,Proj3
Bob	Low:Proj1,Proj2
Charles	High:Proj1

# Operations

---

- What is the highest Proj1 file label such that
  - Alice and Bob can both read?
  - Alice and Charles can both read?
  - All three can read
- What about write?

# Pitbull LX

---

- Compartment only labels
- Less restrictive rules more oriented to collaborating on multiple projects
- Each user and file is labeled with a domain set for read, write, execute
  - $\text{dom1(r-x)} = \text{read:dom1, write:, execute:dom1}$

# LX Access Rules

---

- Dominator operator is superset
  - DomSet1 dominates DomSet2 if DomSet1 is a superset of DomSet2
- For read, write, and execute, access is allowed if:
  - Process DomSet for the right dominates file DomSet for the right
  - E.g., process has domset D1(r-x),D2(rwx) and file has DomSet D1(rwx)
    - Process dominates for read and execute but not for write
- Network access is controlled by intersect

# LX Example Domain Sets

<b>User</b>	<b>Domain Sets</b>
Alice	Proj1(rwx), Proj2(rwx), Proj3(rwx), SecretProj1(rwx), SecretProj2(rwx)
Bob	Proj1(r-x), Proj2(r-x)
Charles	Proj1(rwx), SecretProj1(rwx)

# Operations

---

- What should be the file label to enable Alice, Bob, and Charles to operate on Proj1 files?
- How can Alice and Charles keep Proj1 information from Charles?
- What is the security impact of the symmetric read, write, and execute constraints?

# LX Opt Out Model

---

- Files and Processes must be tagged with AGS\_AWARE flag for LX MAC to apply
  - AGS\_AWARE processes can add further tags to avoid working with unlabeled data
- Goal is to concentrate security efforts on most critical applications

# LX New Process Labeling

---

- Files have two domain sets
  - Access Domain Sets – Control access to the file
  - Execute Domain Sets – Control access domain for process created by executing file
- Details controlled by executable file tags
  - AGS\_COMB\_EXEC – new process domain set is union of parent access domain set and file execute domain set
  - AGS\_MASK\_EXEC – new process domain set is intersection of parent access domain set and file execute domain set
- Similar controls for creating domain set for new file

# SE Linux Type Enforcement (TE)

---

- Access controlled by unstructured label called a type
  - When labeling a process the type is sometimes called a domain
- Policy defines access rules in terms of process and file types
  - `allow <subject type>`  
`<target type>:<class set> <permission set>`
  - `allow httpd_t http_config_t:file`  
`{ read, write };`

# Example TE mapping

User	Domain or type
Alice	Proj1, Proj2, Proj3, SecretProj1, SecretProj2
Bob	ROProj1, ROProj2
Charles	Proj1, SecretProj1

# TE Rules

---

- `allow Proj1 ProjData:file  
                                  { read, write, execute };`
- `allow ROProj1 ProjData:file { read, execute };`
- `allow SecretProj1 SecretProj1Data: file  
                                  { read, write, execute };`

# Operations

---

- How must data be labeled for Alice, Bob, and Charles to coordinate on Proj1?
- How must sensitive Proj1 data be labeled?
- Can Bob write any Proj1 data?

# Domain Transitions

---

- By default new process inherits domain of creating process
- Can create additional rules to enable a domain transition
  - `type_transition d1 d2:process f1`
  - Plus three all rules to permit execute access between the three types

# Attributes

---

- Attributes are arbitrary tags associated with types at definition type
- In many places in policy attributes can be used in place of direct types
  - `allow domain unlabeled_t:file { read, write, execute };`

# Attributes Use

---

- SE Linux used attribute to define privileges and associate them with types
- SE Linux MLS uses constraints to explicitly encode BLP requirements
- ```
mlsconstrain { dir file lnk_file chr_file blk_file
               sock_file fifo_file } { read getattr execute }
(( l1 dom l2) or
 ( ( t1 == mlsfilereadtoclr ) and ( h1 dom l2 )) or
 ( t1 == mlsfileread ) or
 ( t2 == mlstrustedobject ));
```

# TE Policy Problems

---

- Explicit rule base policy gives expressibility, but...
- Policies become very large
  - 150,000 rules in “targeted” SE Linux policy (after macro expansion)
- Lacks modularity
  - Create new application. Would like to install good policy for application
  - Modularity mechanism being worked on, but not there yet
- Policy language is powerful, but very low level
  - Macros used to approximate program modularity
  - Analysis tools work post macro

# SE Linux Multi-Category Security

---

- Uses MLS mechanism to implement category-only labeling
  - Security Level pinned to S0
- Trusted operator associates categories with users
- Normal user can associate categories with files
- File access determined by intersecting file categories with user categories

# SE Linux MCS

---

- Discretionary Control
- Evaluated after Unix DAC and Type Enforcement
- Logically no different than normal ACL's
  - Stylistic differences
- Goal is to avoid user errors rather than foil malicious users

# Key Points

---

- Different MAC models are implemented
- While the model may seem simple, real implementations have complexity
- Know where to look when evaluating the usefulness of a system
- Understand the target deployment