Type Enforcement Rules and Macros

Security Policy Development Primer for Security Enhanced Linux

(Module 7)

Tresys Technology

Security Enhanced
Linux

Overview of Type Enforcement Rules

- Set of policy rules that specify relationships between types
  - i.e. the type enforcement policy
- Several different rules
  - somewhat evolved and changed over past year
  - challenge for those working with older systems
- TE rules in a policy can be numerous
  - for example in sample policy:
    - > 27,000 type `allow' rules
    - > 1,000 `type_transition' rules

Security Enhanced
Linux

A Primary Goal of a TE Policy

- Define access control for given programs (i.e., a domain)
- Some of the motivations/reasons governing decisions
    - program protection
        - prevent interference/modification of program's resources
    - least privilege
        - limit program to minimal access rights necessary
    - limit error propagation
        - side effects of errors contained within the domain access rights
    - all of which leads to greater security assurance
        - significantly less chance of exposure to vulnerability
- Roles associate users with domains (not the TE policy)
    - some domain types designed for users rather than programs

Security Enhanced Linux

Other Goals for a TE Policy

- ## Self-protection
  - kernel protects itself and its resources
  - protect the policy itself

- ## Enforce other Mandatory policies
  - information flow
  - domain isolation
  - guard applications (controlled information flow)

- ## All focused on domain (program) access
  - not users!

Security Enhanced
Linux

TE Access Vector Rules Syntax

- rule_name src_types tgt_types : classes permissions ;
  - access vector (AV) rules
    - allow   grant access
    - neverallow   TE assertions
    - auditallow   log when access granted
    - dontaudit   (NEW) don't log access denied
    - auditdeny   (replaced by dontaudit)
  - types (source and target)
    - one or more type <u>or</u> type attribute identifiers, or
      - `*` means all types
      - keyword `self' in target (same as source, including multiples)
    - `~`   can be used for complement of specified type/attrib set
    - with more than one identifier, list enclosed in braces `{ }'
      - { type1_t type2_t typeN_t attribute }

Tresys Technology

Security Enhanced Linux

TE Access Vector Rules Syntax

- rule_name src_types tgt_types : classes permissions ;
  - classes
    - one or more defined object classes
    - `*' and `~' may be used
    - multiple classes enclosed in braces `{ }'
  - permissions
    - one or more permissions defined for the specified class(es)
    - all permissions must be valid for all object classes specified
    - `*' and `~' may be used
    - multiple permissions enclosed in braces `{ }'
    - if multiple rules specify same source-target-class, then
      - allow, auditallow, dontaudit, auditdeny (old): union of all permissions used

Tresys Technology

Security Enhanced
Linux

Type Allow Rule

- Grants source type(s) access to target type(s)
  - no access granted by default
  - granular access specification
    - object classes & permissions

allow user_t bin_t:file {read getattr lock execute ioctl execute_no_trans };
- allow user_t domain type read and execute access to bin_t files
- with or without a transition

allow user_t self : process * ;
- allow user_t domain types all access to itself

allow userdomain shell_exec_t : file { read getattr lock execute ioctl };
- allow types with userdomain attribute read/execute to shell_exec_t files
- but only with a domain transition (i.e., no exec_no_trans access)

Neverallow Rule

- States invariants for the policy
  - no allow rule may violate any invariant
  - if so policy will not compile
- Not included in running system
  - enforced by checkpolicy when compiling policy

neverallow passwd_t ~{ bin_t sbin_t ld_so_t } : file execute_no_trans ;

  - passwd_t domain may never execute without a domain transition, files of any types other than bin_t, sbin_t and ld_so_t

neverallow domain ~domain : process transition ;

  - no domain type (`domain' is an attribute) may transition to a new type unless the new type is also a domain type

Security Enhanced
Linux

A Look at Macros

- Sample policy uses m4 macros
  - provides easier-to-use abstractions
  - not intrinsic to SE Linux policy language
- Global macros: ./policy/macros/global_macros.te
- Object class macro examples

file_class_set   { file lnk_file sock_file fifo_file chr_file blk_file }

notdevfile_class_set   { file lnk_file sock_file fifo_file }

- be careful! you might include objects not intended (e.g., devices)

- Permission macro examples

rx_file_perms {read getattr lock execute ioctl }

r_dir_perms {read getattr lock search ioctl }

Tresys Technology

Security Enhanced
Linux

Warning on Using Macros

- **Be careful not to overuse macros**
  - may provide more access than intended
- **every_domain macro**
  - name implies required for every domain
  - does provide a pragmatic set of access
  - but may be too permissive for some domains
    - allows network access (can_network macro)
    - read access to many, many types
    - execute shared libraries

Type Transition Rule

- Specified default type for new object; two forms:
  - default process transition
  - default type for new file objects
- Syntax

  type_transition src_types tgt_types : class default_type ;
  - src_type & tgt_types: may use `*' and `~', and sets of types
  - default_type: single type
  - class governs which rule form
    - process → domain transition
    - file related object → default object type

Security Enhanced
Linux

Type Transition Rule

type_transition src_type tgt_type : process default_type ;

- default transition form
- unless otherwise requested, when process with src_type executes file with tgt_type, the process will have default_type domain
  - if allowed by TE policy

type_transition src_type tgt_type : file-related default_type ;

- default object type form
- unless otherwise requested, when process with src_type creates new file related object (e.g., file, dir) in a directory of tgt_type, the new object will have default_type
  - if allowed by TE policy

Security Enhanced Linux

Type Transition Rule Examples

type_transition userdomain passwd_exec_t:process passwd_t;

- domain transition
- causes domains with userdomain attribute to transition to passwd_t when executing passwd_exec_t programs by default

type_transition passwd_t tmp_t :
{ file lnk_file sock_file fifo_file } passwd_tmp_t;

- default file type
- when passwd_t process creates new file system objects in a tmp_t directory (e.g., /tmp), those new files will have passwd_tmp_t type
- common technique to protect a domain's temporary files

Security Enhanced
Linux

Walk-through Example

Creating a Policy Module For 'who'

- Only allow sysadm_r to run the `who' command
- Policy requirements
  - create who_t domain/type
  - only allow sysadm_r access to who_t domain
  - allow sysadm_t to transition to who_t
  - protect system resources `who' requires

'who' Module: the Beginning

- ## Create the module files (.te & .fc files)
- ## Create the types

```
# who.te
#DESC who command
type who_t, domain;
role sysadm_r types who_t;
type who_exec_t, file_type, exec_type;
```

- ## Assign labeling in the .fc file

```
# who.fc
/usr/bin/who   system_u:object_r:who_exec_t
```

'who' Module: Next step

- ## Add a domain transition for sysadm_t

```
type who_t, domain;
role sysadm_r types who_t;
type who_exec_t, file_type, exec_type;

domain_auto_trans(sysadm_t, who_exec_t,
     who_t)
```

- ## Build, load and test
  - chcon /usr/bin/who after loading policy
    chcon system_u:object_r:who_exec_t /usr/bin/who

Security Enhanced Linux

`who' Module: part 3

- ## Allow common access permissions

```
type who_t, domain;
role sysadm_r types who_t;
type who_exec_t, file_type, exec_type;
domain_auto_trans(sysadm_t,
  who_exec_t,who_t)
```

every_domain(who_t)

- ## Build, load and test

`who' Module: part 4

- **Access to tty**

```
type who_t, domain;

role sysadm_r types who_t;

type who_exec_t, file_type, exec_type;

domain_auto_trans(sysadm_t, who_exec_t,
      who_t)

every_domain(who_t)

allow who_t admin_tty_type:chr_file
  { rw_file_perms };
```

Other who.te issues

- Restrict access to
  - /var/run/utmp
  - /var/log/wtmp

- Difficult to determine what domains also require access to these files.
- Exercise for the student! ☺

Security Enhanced
Linux

QUESTIONS?

Tresys Technology

Security Enhanced
Linux