Writing Security Enhanced Linux Policies for your Applications

Dr. Charles J. Antonelli
Civil & Environmental Engineering

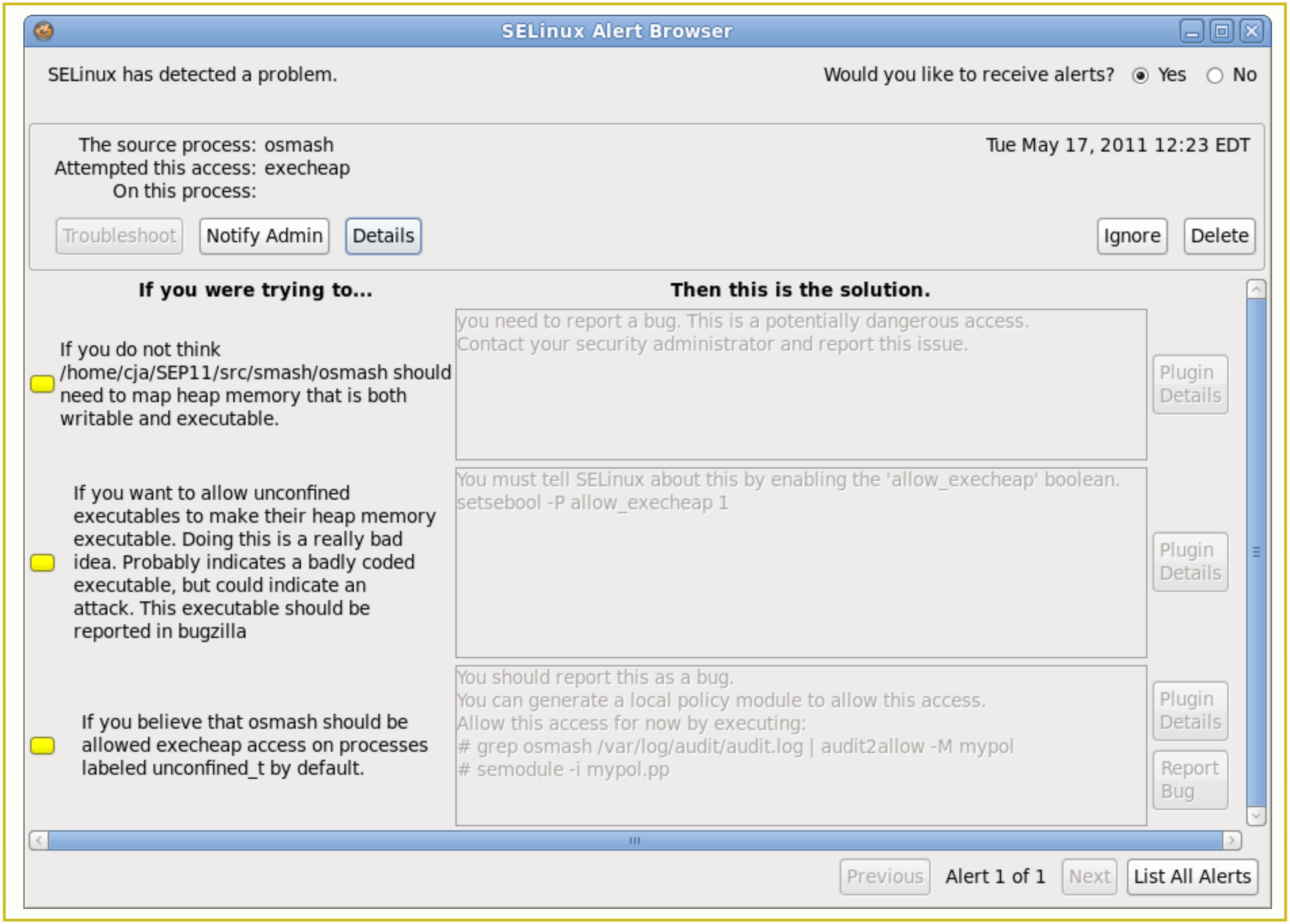
The University of Michigan

May 18, 2011

Roadmap

- Quick SELinux Summary
  - SELinux Permissive Domains
  - SELinux Booleans
- SELinux Policy Theory
- SELinux audit2allow

Quick SELinux Summary


SELinux Alert Browser
SELinux has detected a problem.
Would you like to receive alerts?
Yes
No
The source process: osmash
Attempted this access: execheap
On this process:
Tue May 17, 2011 12:23 EDT
Troubleshoot
Notify Admin
Details
Ignore
Delete
If you were trying to...
Then this is the solution.
If you do not think /home/cja/SEP11/src/smash/osmash should need to map heap memory that is both writable and executable.
you need to report a bug. This is a potentially dangerous access.
Contact your security administrator and report this issue.
Plugin Details
If you want to allow unconfined executables to make their heap memory executable. Doing this is a really bad idea. Probably indicates a badly coded executable, but could indicate an attack. This executable should be reported in bugzilla
You must tell SELinux about this by enabling the 'allow_execheap' boolean.
setsebool -P allow_execheap 1
Plugin Details
If you believe that osmash should be allowed execheap access on processes labeled unconfined_t by default.
You should report this as a bug.
You can generate a local policy module to allow this access.
Allow this access for now by executing:
grep osmash /var/log/audit/audit.log | audit2allow -M mypol
semodule -i mypol.pp
Plugin Details
Report Bug
Previous
Alert 1 of 1
Next
List All Alerts

SETroubleshoot Details Window

Additional Information:

Source Context	unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
Target Context	unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
Target Objects	Unknown [process]
Source	osmash
Source Path	/home/cja/SEP11/src/smash/osmash
Port	<Unknown>
Host	albedo.engin.umich.edu
Source RPM Packages	
Target RPM Packages	
Policy RPM	selinux-policy-3.9.7-40.fc14
Selinux Enabled	True
Policy Type	targeted
Enforcing Mode	Enforcing
Host Name	albedo.engin.umich.edu
Platform	Linux albedo.engin.umich.edu 2.6.35.6-45.fc14.x86_64 #1 SMP Mon Oct 18 23:57:44 UTC 2010 x86_64 x86_64
Alert Count	1
First Seen	Tue 17 May 2011 12:23:50 PM EDT
Last Seen	Tue 17 May 2011 12:23:50 PM EDT
Local ID	da2a0e57-a673-48fd-ad9c-474318fd37c4

Raw Audit Messages

```
type=AVC msg=audit(1305649430.780:4122): avc:  denied  { execheap } for
pid=28645 comm="osmash" scontext=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023 tcontext=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
tclass=process


type=SYSCALL msg=audit(1305649430.780:4122): arch=x86_64 syscall=mprotect
success=no exit=EACCES a0=1e9f000 a1=1000 a2=7 a3=7fffd4b20d90 items=0
ppid=28412 pid=28645 auid=1122 uid=1122 gid=1122 euid=1122 suid=1122
fsuid=1122 egid=1122 sgid=1122 fsgid=1122 tty=pts1 ses=165 comm=osmash
exe=/home/cja/SEP11/src/smash/osmash
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)

Hash: osmash,unconfined_t,unconfined_t,process,execheap

audit2allow

#============= unconfined_t ==============
#!!!! This avc can be allowed using the boolean 'allow_execheap'
```

Why SELinux?

Discretionary Access Control (DAC)

- Linux and UNIX systems provide only Discretionary Access Control
  - Users determine access control settings of their objects
  - Improper access control settings expose data
  - Superusers can access everything
    - Access checks disabled

Why SELinux?

No Compartmentalization

- Linux and UNIX processes have extensive access to system objects
  - e.g. /tmp, /proc, libc, syscalls
  - Process can change DACs
  - Process inherit's parent's rights
  - A subverted root process can access everything

Why SELinux?

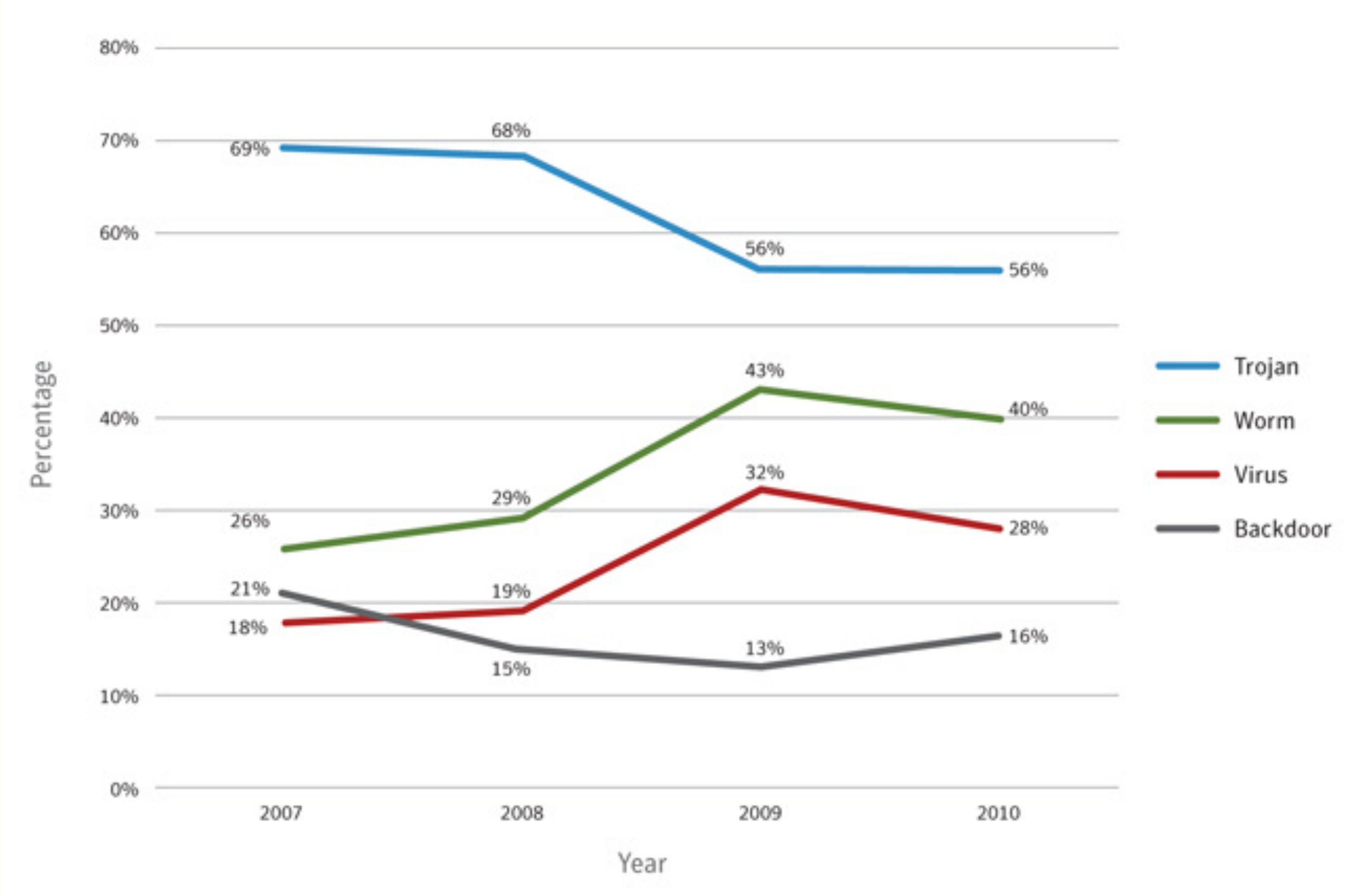


Figure 17. Prevalence of malicious code types by potential infections, 2007–2010
Source: Symantec Global Internet Security Threat Report, Vol. XVI, April 2011

SELinux

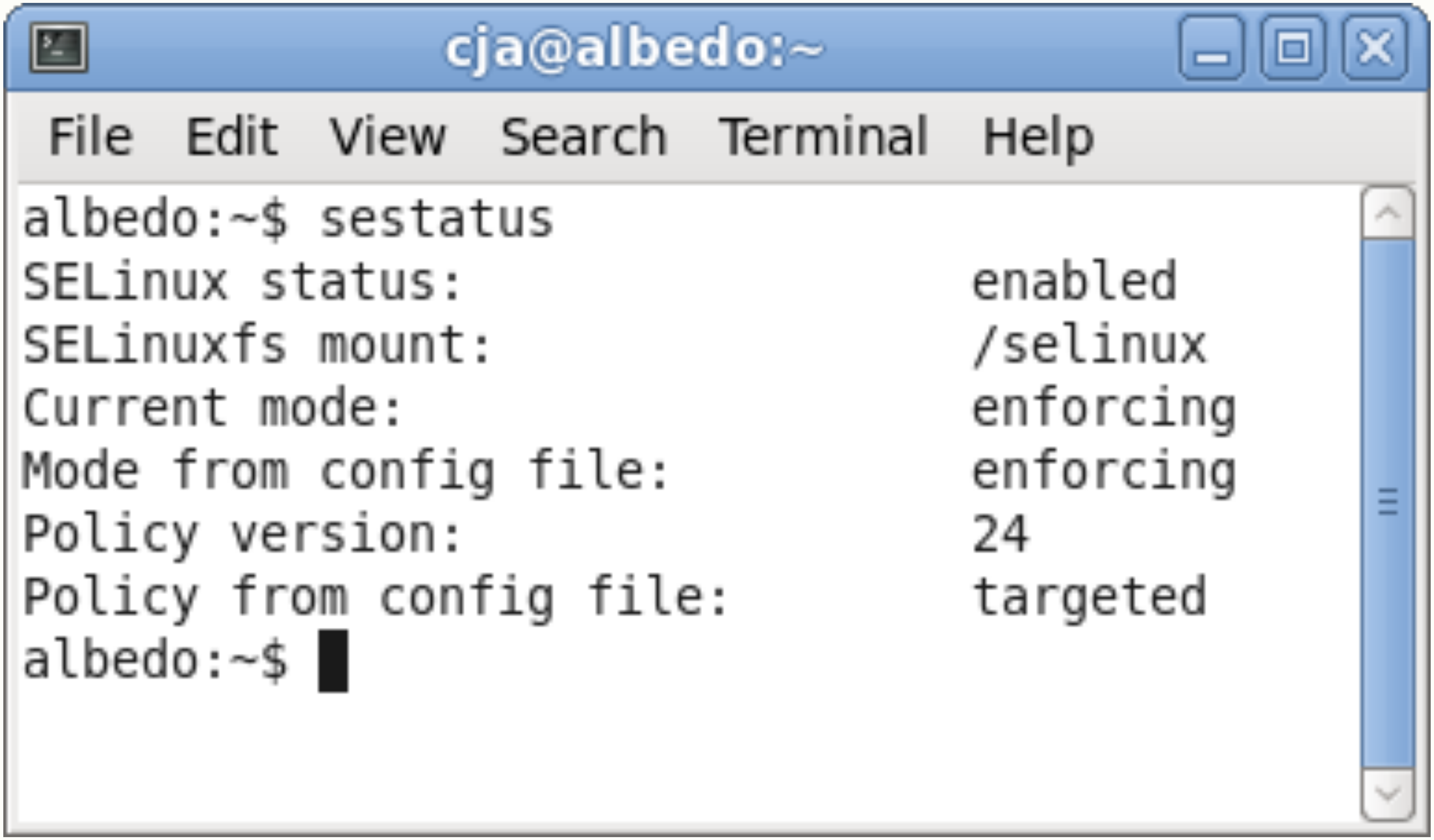
- Three forms of access control
  - Type Enforcement (TE)
    Access to objects controlled by *types* attached to objects
  - Role-Based Access Control (RBAC)
    Subjects are controlled by *roles* attached to them
  - Multi-Level Security (MLS)
    Bell-LaPadula security levels & controls

SELinux

- Three kinds of policies
  - Targeted
    Only key system processes are protected, all other processes are *unconfined*
  - Strict
    Everything is denied by default, policies specify minimal access required for each process
  - MLS
    Not fully implemented

SELinux Modes

- Three operating *modes*:
  - Enforcing – performs and logs access checks, and enforces access decisions
    - Default mode
  - Permissive – performs and logs access checks, but does not enforce access decisions
  - Disabled– doesn't do anything


cja@albedo:~
File Edit View Search Terminal Help
albedo:~$ sestatus
SELinux status: enabled
SELinuxfs mount: /selinux
Current mode: enforcing
Mode from config file: enforcing
Policy version: 24
Policy from config file: targeted
albedo:~$

SELinux Limits

- SELinux is thus a security layer
  - Targeted type-enforcement
  - Not a replacement for firewalls, passwords, encryption, …
  - Not a complete security solution
  - How to defeat SELinux
    - `$ smash`
    - `# setenforce 0`

SELinux Mechanics

- An **Identity** identifies the user
- A **Role** determines in which domains a process runs
- A **Type** is assigned to an object and determines access to the object
- A **Domain** is assigned to a subject and determines what that subject may do
  - So a domain is a *capability*
  - “Domain” and “Type” are synonymous

Example

```
$ ls -ldZ .
drwx——  cja cja system_u:object_r:user_home_dir_t:s0 .
$ ls -lZ .bashrc
-rw-r—r—  cja cja system_u:object_r:user_home_t:s0 .bashrc
$ ps -Z
LABEL                              PID TTY          TIME CMD
unconfined_u:unconfined_r:unconfined_t:s0 3581 pts/0 00:00:00 bash
unconfined_u:unconfined_r:unconfined_t:s0 3732 pts/0 00:00:00 ps
$ ps axZ | grep sendmail:\ accepting
system_u:system_r:sendmail_t:s0  2756 ?        Ss     0:00 sendmail:
   accepting connections
$ ps axZ|wc -l
203
$ ps axZ|grep unconfined|wc -l
55
$ ps axZ|grep -v unconfined|wc -l
149
```

SELinux Logging

- /var/log/audit/audit.log if `auditd` is running
- /var/log/messages otherwise
- Failure audits include
  - Failing operation (read, etc.)
  - Process ID of executable
  - Name of executable
  - Mount point and path to object accessed
  - Linux inode of object accessed

SELinux GUI

- Tools
  - Configure SELinux
    sudo /usr/bin/system-config-selinux
    System | Administration | SELinux Management
  - Interpreting SELinux log errors
    /usr/bin/sealert
    Applications | System Tools | SELinux Troubleshooter

Lab – stopping buffer overflows

1. Get smash.tgz
   - `wget http://www-personal.umich.edu/~cja/mmc11/smash.tgz`
   - `tar zxf smash.tgz`
   - `cd ~/smash`
   - `make`
2. Run the executable
   - What happened?
   - Examine the SELinux audit
3. Change SELinux to permissive mode
   - System | Administration | SELinux management
   - Set current enforcing mode to permissive
4. Rerun the executable
   - What happened this time?

Permissive domains

- Can set a single domain to be permissive
  - Investigate a problem with a single process
  - Define policies for new applications
  - Greatly reduces need for permissive mode

```
semanage permissive -a httpd_t
semodule -l | grep permissive
semanage permissive -d httpd_t
```

Booleans

- Allow policies to be changed at runtime
  - Fine-tune service access
  - Change service port numbers
    - Must be pre-defined
  - Greatly reduces need for new policy modules

```
getsebool -a
setsebool -P httpd_can_network_connect_db on
semanage boolean -l
semanage port -l
semanage port -a -t http_port_t -p tcp 1234
```

SELinux Policy Theory

SELinux policy

Overview

- Behavior of processes is controlled by policy
- A base set of policy files define the system policy
- Additional installed software may specify additional policy
  - This policy is added to the system policy on installation

SELinux policy
Six easy pieces

- Type enforcement (TE) attributes
- TE type declarations
- TE transition rules
- TE change rules (not used much)
- TE access vector rules
- File context specifications

TE attributes

- Files named *.te
- Attributes identify sets of types with similar properties
  - SELinux does not interpret attributes
- Format:
  - <attribute> <name>
- Examples:
  - `attribute logfile;`
  - `attribute privuser;`

TE type declarations

- Files named *.te
- Defines type names, with optional aliases and attributes
- Format:
  - type <name> [alias <aliases>] [atttributes]
- Examples:
  - `type mailman_log_t, file_type, sysadmfile, logfile;`

TE transition rules

- Files named *.te
- Specifies allowed type transitions
- Format:
  - type_transition <source> <action> <target>
- Examples:
  - `type_transition inetd_t ftpd_exec_t:process ftpd_t;`
    When running in the inetd_t domain, transition to the ftpd_t domain when executing a program of type ftpd_exec_t
  - `type_transition sshd_t tmp_t:{ dir file lnk_file sock_file fifo_file } sshd_tmp_t;`
    When a process running in the sshd_t domain creates a file in a directory of type tmp_t, the new file should be labeled with the sshd_tmp_t type

TE change rules

- Files named *.te
- Specifies the new type to use when relabeling, based on process domain, object type, and object class
- Format:
  - type_change <source> <action> <target>
- Examples:
  - type_change user_t tty_device_t:chr_file user_tty_device_t;
    When running in the user_t domain, relabel the associated terminal device as a user terminal

TE access vector rules

- Files named *.te
- Specifies the set of permissions based on a type pair and an object security class.
- Format:
  - <kind> <source> <target> <securityclass>
    <kind> is one of:
    - allow – allow requested access
    - auditallow – allow and log access
    - dontaudit – don’t allow and don’t log
    - neverallow – stop compilation of policy

TE access vector rules

- Examples
  - allow initrc_t acct_exec_t:file { getattr read execute };
    Processes running in the initrc_t domain have get-attribute, read, and execute access to files of type account_exec_t
  - dontaudit traceroute_t { port_type -port_t }:tcp_socket name_bind;
    Processes running in the traceroute_t domain do not log the denial of a request for name_bind permission on a tcp_socket for all types associated to the port_type attribute (except port_t)
  - auditallow ada_t self:process execstack;
    Processes runnin the the ada_t domain logs the granting of a request to execute code located on the process stack. Note*: a separate rule must exist to grant this permission.*
  - neverallow ~can_read_shadow_passwords shadow_t:file read;
    No subsequent allow rule can permit the shadow password file to be read, except for those rules associated with the can_read_shadow_passwords attribute. Note*: this rule is intended to be used during the compilation of policy files, not to protect a running system.*

File context specifications

- Files named *.fc
- Defines default contexts for files
- Format:
  - <name-re> [file-type][security-context]
- Examples:

```
/bin/login            --   system_u:object_r:login_exec_t:s0
/var/tmp/logcheck     -d   system_u:object_r:logrotate_tmp_t
/etc/tripwire(/.*)?        system_u:object_r:tripwire_etc_t
```

Lab – examine policy sources

- Get sample policy sources
  - `wget` [`http://www-personal.umich.edu/~cja/mmc11/selinux-3012-targeted-sources.tgz`](http://www-personal.umich.edu/~cja/mmc11/selinux-3012-targeted-sources.tgz)
  - `tar zxf selinux-3012-targeted-sources.tgz`
  - `cd targeted`

Lab – examine policy sources

- Raw Audit Messages :

  node=localhost.localdomain type=AVC msg=audit(1295914760.945:51): avc: denied { read } for pid=3317 comm="ifconfig" path="/var/run/vmware-active-nics" dev=dm-0 ino=929018 scontext=system_u:system_r:ifconfig_t:s0 tcontext=system_u:object_r:init_var_run_t:s0 tclass=file

  node=localhost.localdomain type=SYSCALL msg=audit (1295914760.945:51): arch=40000003 syscall=11 success=yes exit=0 a0=85cd3c0 a1=85cd4e8 a2=85cc480 a3=85cd4e8 items=0 ppid=2939 pid=3317 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="ifconfig" exe="/sbin/ifconfig" subj=system_u:system_r:ifconfig_t:s0 key=(null)

Lab – examine policy sources

- Let's examine the ifconfig policy source:
  - `find . -name ifconfig\*`
    Output:
    ./domains/program/ifconfig.te
    ./file_contexts/program/ifconfig.fc
- Type enforcement:
  - `less ./domains/program/ifconfig.te`
  - `find . -print | xargs grep general_domain_access`
    `less./macros/core_macros.te`
  - `find . -print | xargs grep setfscreate`
    `less ./flask/access_vectors`
- File contexts:
  - `less ./file_contexts/program/ifconfig.c`

audit2allow

- Generates SELinux policy “allow” rules from logs of denied operations
  - Creates installable policy modules
    - These modules may not be correct!
  - Warns if Booleans already exist

audit2allow

- Generate policy source for examination:
  - `audit2allow -a -m localpol >localpol`
- Generate policy object and install:
  - `audit2allow -a -M localpol`
  - `sudo semodule -i localpol.pp`
- Remove a policy object:
  - `sudo semodule -r localpol.pp`

References

- P. A. Loscocco, S. D. Smalley, P. A. Muckelbauer, R. C. Taylor, S. J. Turner, and J. F. Farrell, "The inevitability of failure: the flawed assumption of security in modern computing environments," *Proceedings of the 21st National Information Systems Security Conference, pp 303–314, Oct. 1998. http://csrc.nist.gov/nissc/1998/proceedings/paperF1.pdf*
- Ray Spencer, Stephen Smalley, Peter Loscocco, Mike Hibler, Dave Andersen, and Jay Lepreau, "The Flask Security Architecture: System Support for Diverse Security Policies," Proceedings of the 8th USENIX Security Symposium, Washington D.C., August 1999.
- Loscocco, P. and S. Smalley, "Integrating Flexible Support for Security Policies into the Linux Operating System," Proceedings of the FREENIX Track, Usenix Technical Conference, June 2001.
- Trent Jaeger, Reiner Sailer, and Xiaolan Zhang, "Analyzing Integrity Protection in the SELinux Example Policy," Proc. 12th Usenix Security Symposium, Washington DC, August 2003.
- D. E. Bell and L. J. La Padula, "Secure computer systems: Mathematical foundations and model," Technical Report M74-244, MITRE Corporation, Bedford, MA, May 1973.
- Richard Petersen, "Fedora 14 Administration and Security", Surfing Turtle Press, 2010. ISBN 1-936280-22-1.
- Bill McCarty, "SELinux: NSA's Open Source Security Enhanced Linux," O'Reilly Media Inc., 2005.
- http://wiki.centos.org/HowTos/SELinux. Accessed 17 May 2011.