

SELinux

Thorsten Scherf

Red Hat EMEA

October 2015



redhat.

What is wrong with UNIX security?

- Programs have full control over the access given to files they create (Discretionary Access Control - DAC)
- Therefore no protection against malicious software, “social engineering” and bugs in privileged software which may result in the software granting inappropriate access to files (eg, creating a mode 777 file in /tmp)
- No protection against 0-Day exploits
- No Isolation in **Cloud** environments

SELinux is a LABELING system

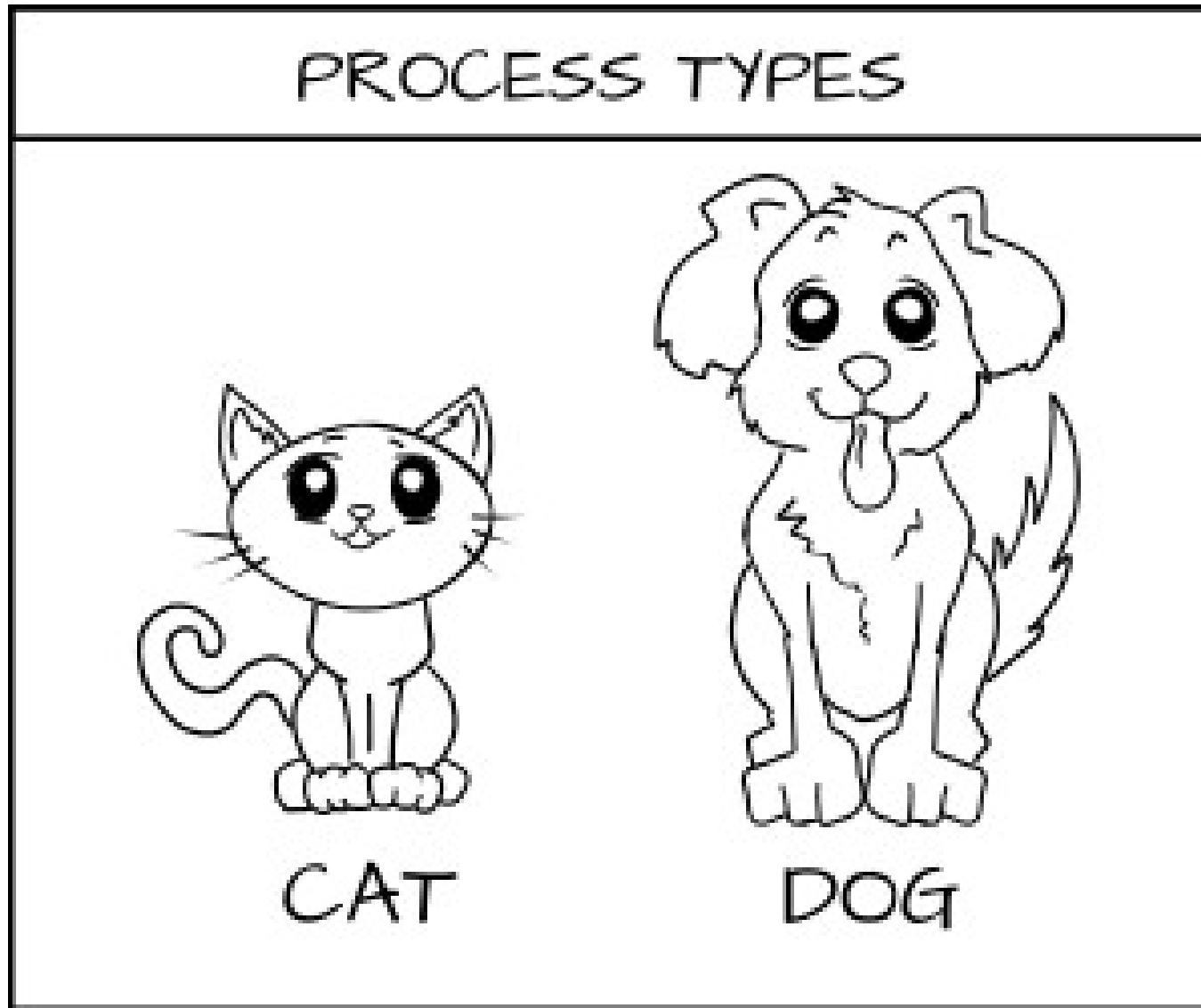
Every Process has a LABEL

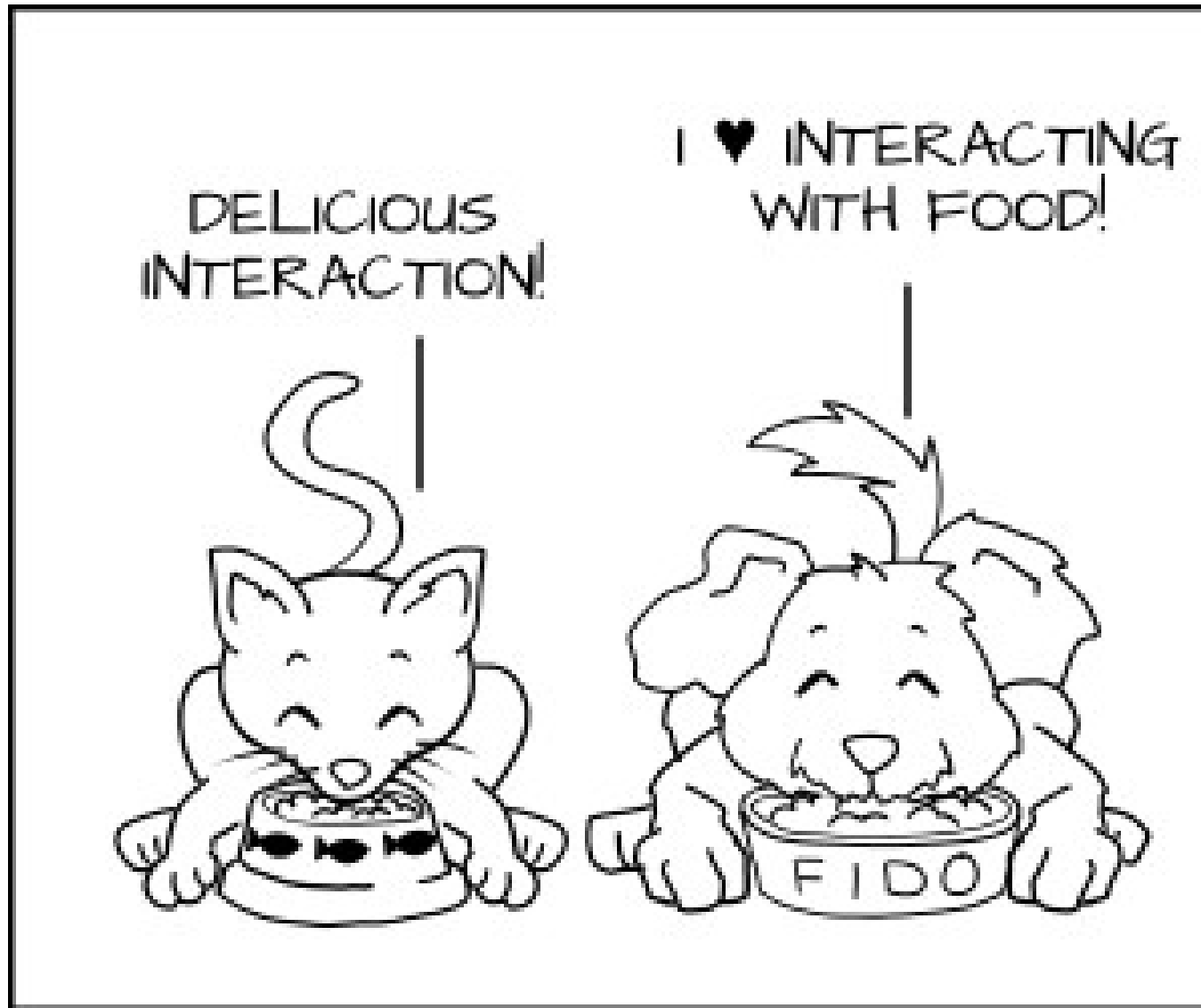
Every File, Directory, System object has a
LABEL

Policy rules control access between
labeled processes and labeled objects

The Kernel enforces the rules

How do Type Enforcement work?





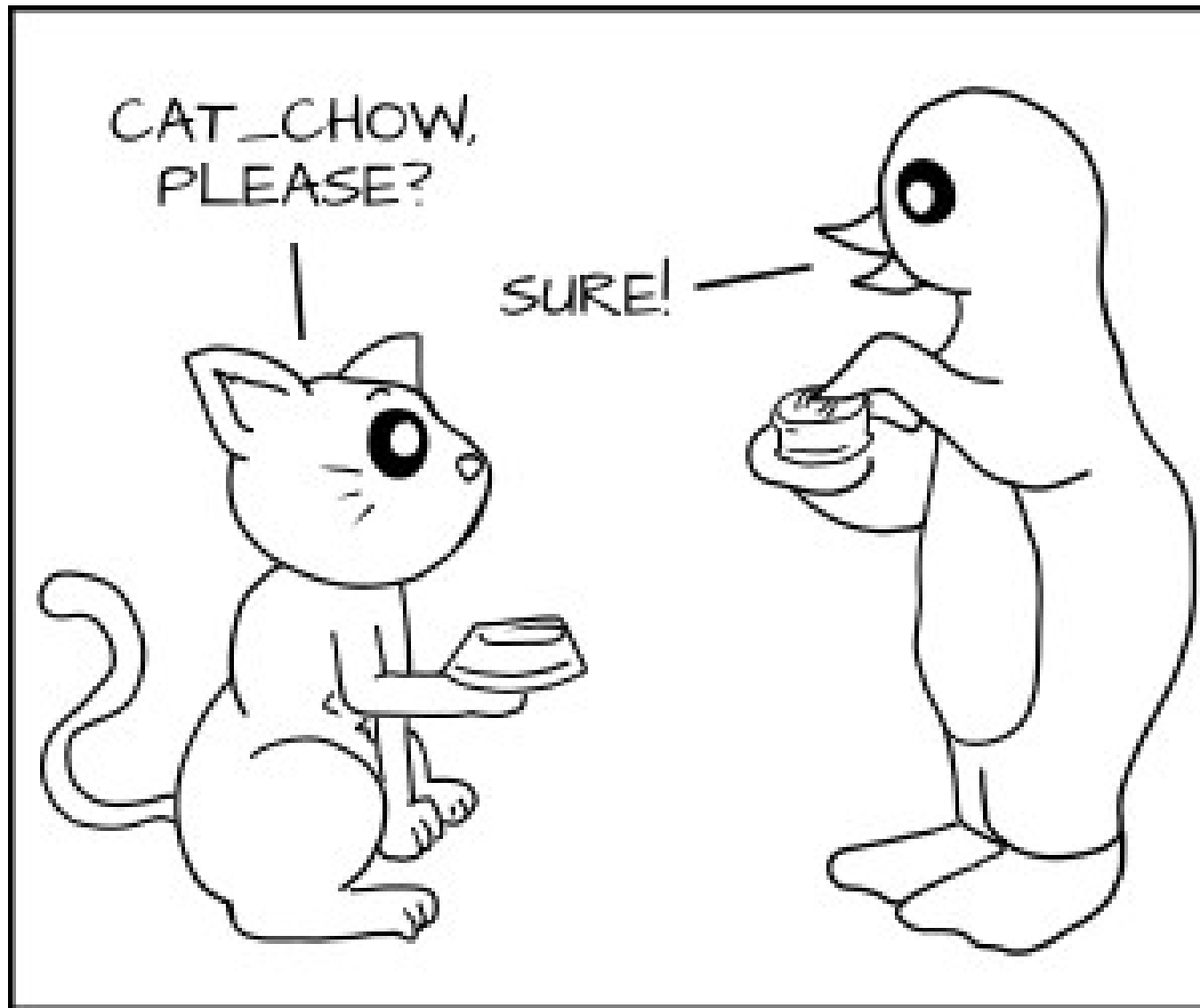
OBJECT TYPES

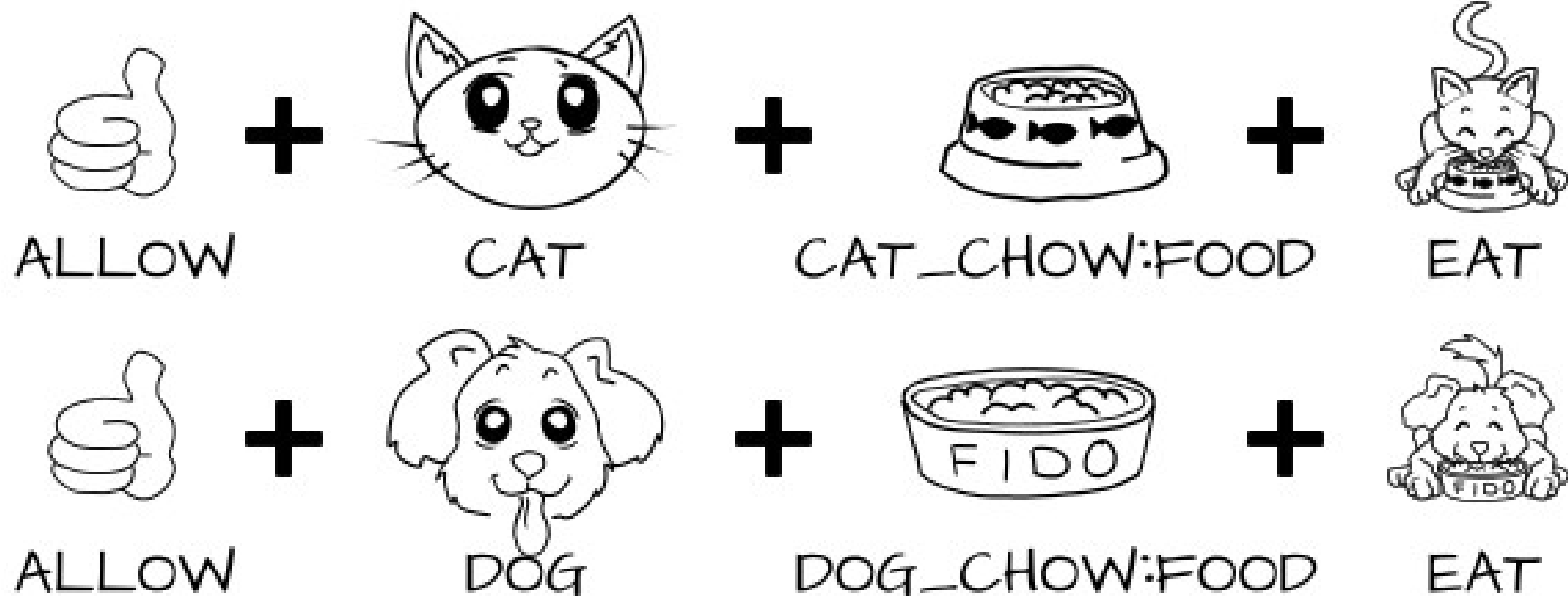


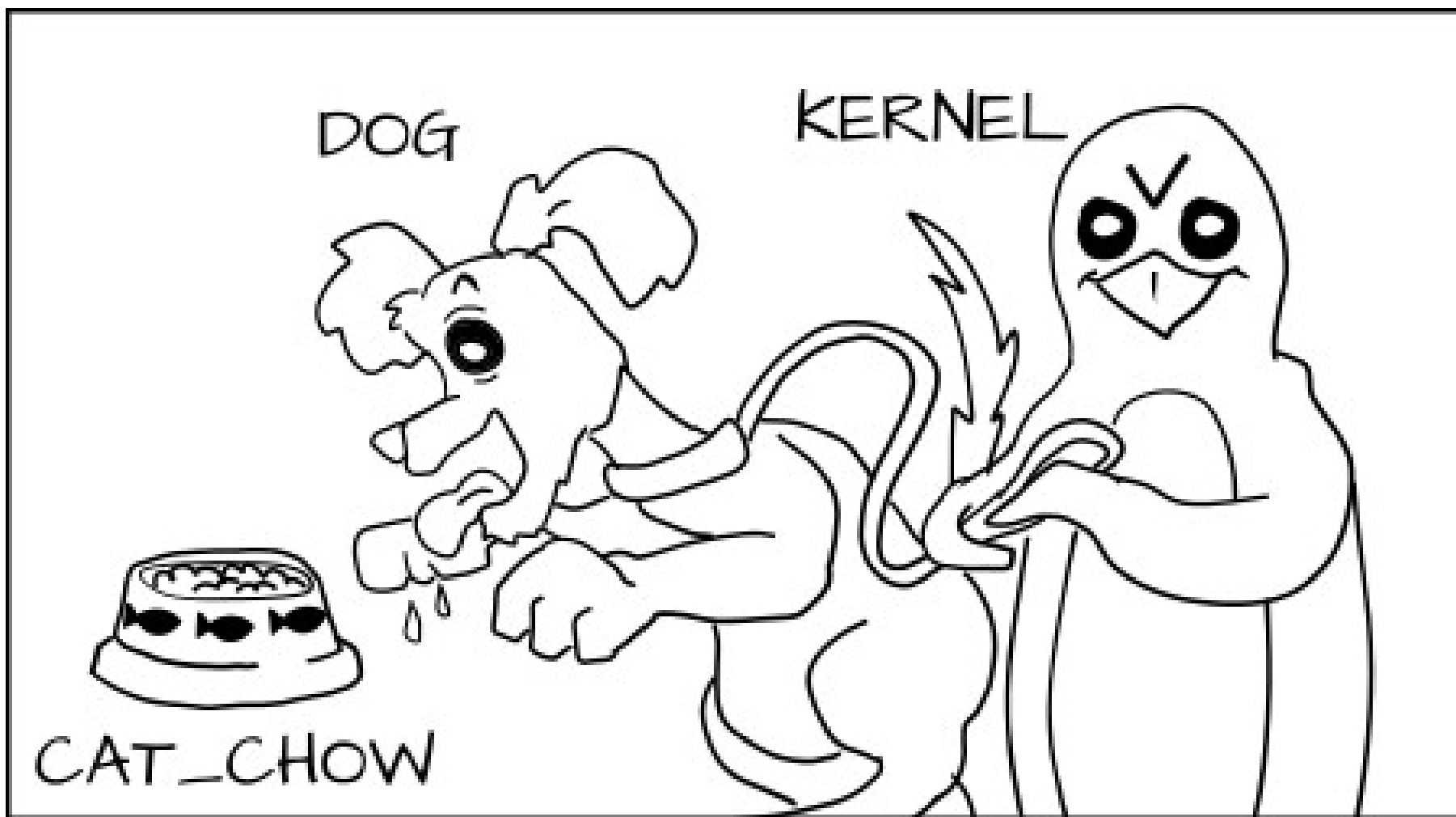
CAT_CHOW



DOG_CHOW







How does Multi Category Security work?



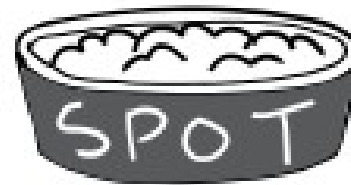
FIDO



SPOT



DOG_CHOW:FIDO

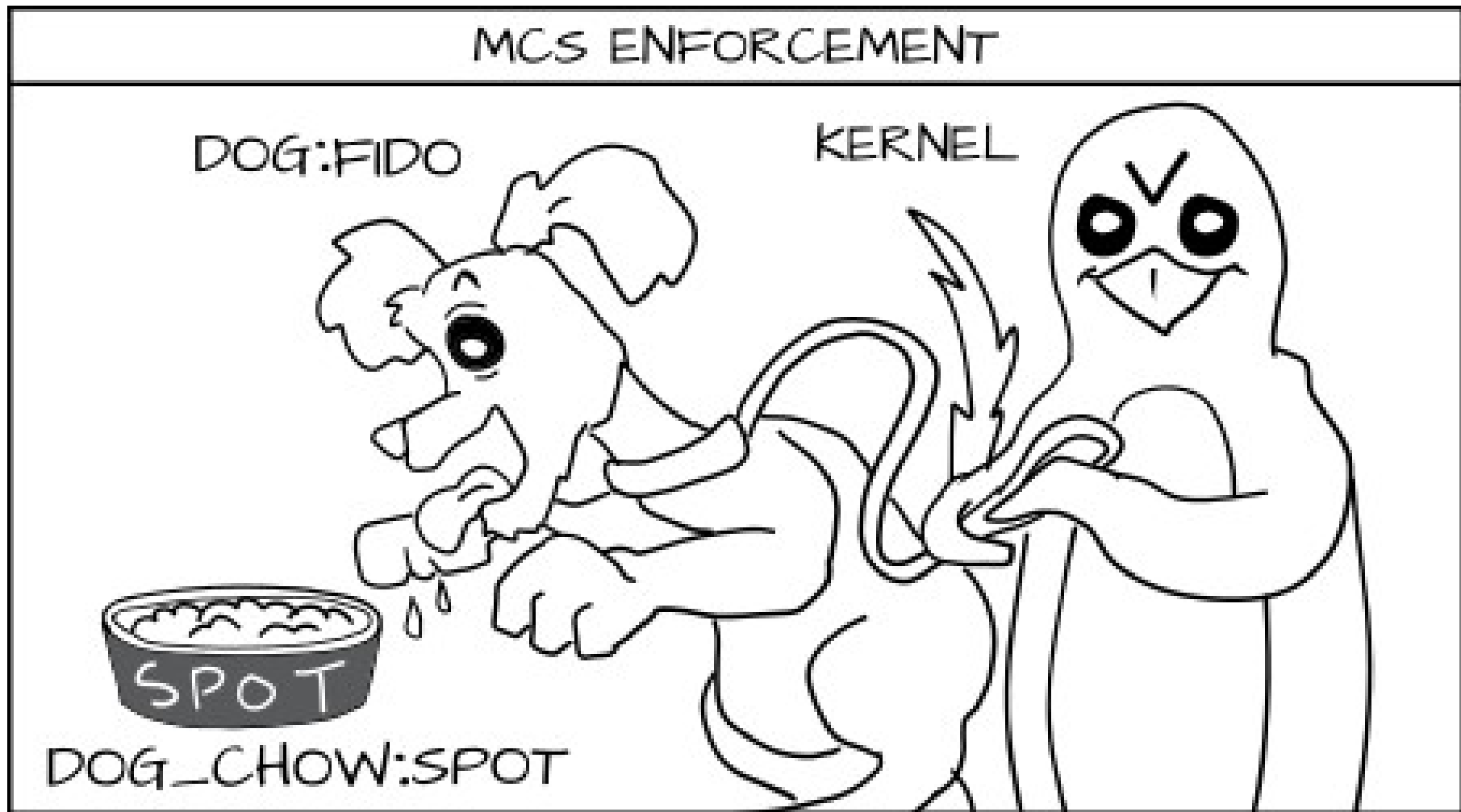


DOG_CHOW:SPOT

DOG:FIDO



DOG_CHOW:FIDO



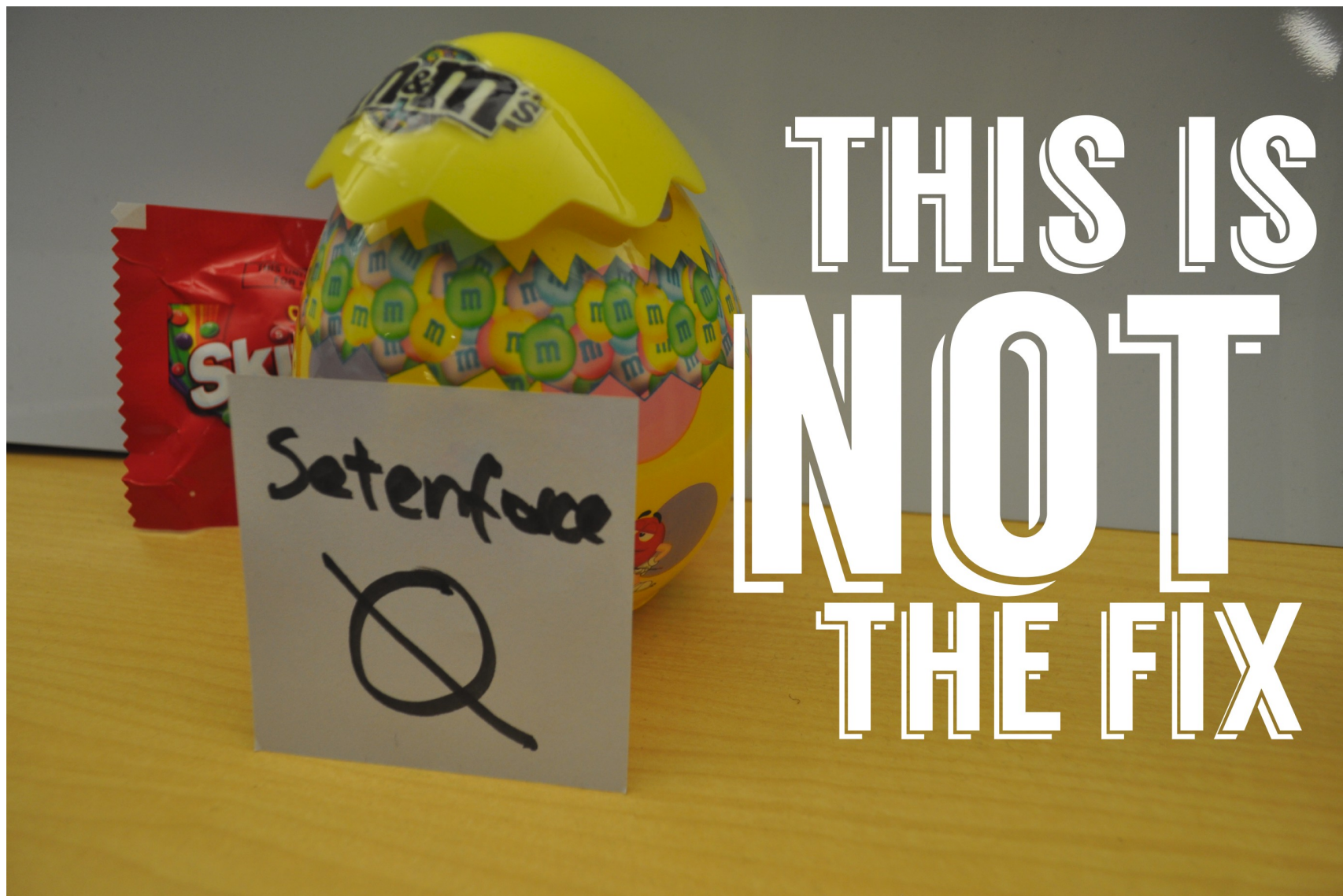
I TOLD YOU

-

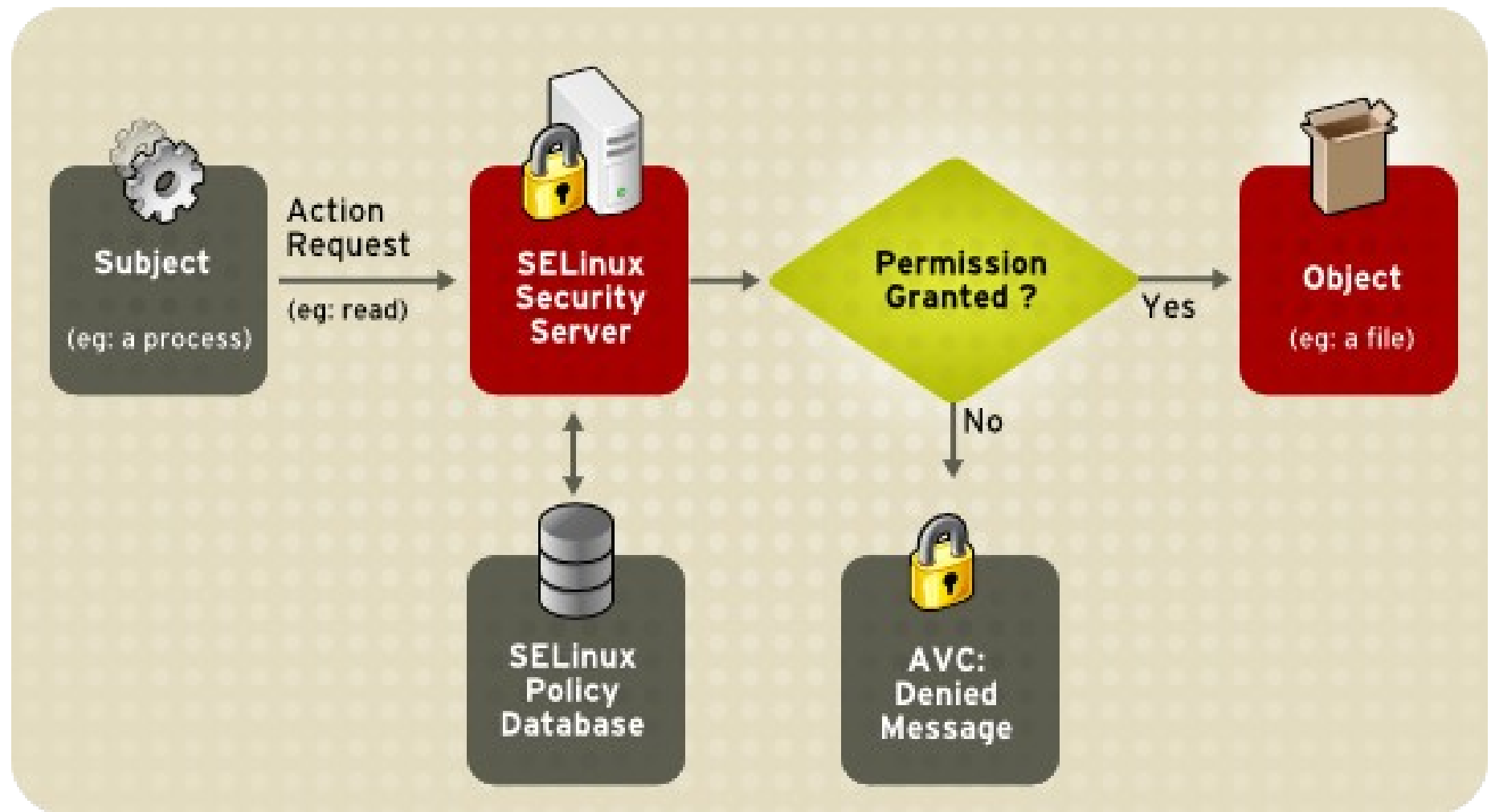
IT'S EASY

AND IT'S ALL ABOUT LABELING





Architecture



SELinux Key Components

Security Context

- Basic labels used in SELinux
 - `system_u:object_r:httpd_exec_t[:s0-s0:c0]`
 - `system_u:system_r:httpd_t[:s0-s0:c0]`
- All subjects/objects have an associated security context
- Called a domain when used on a process (TE)
- Called a file_context when associated with a file (TE)
 - File Context are stored as extended attributes with the inode on the file system
 - On some file systems the kernel that do not support extended attributes the kernel provides the file context.

Example: Confined process

```
# echo "Hello World..." > /var/www/html/foo
```

```
# ls -lZ /var/www/html/foo
```

```
-rw-r--r--. root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html/foo
```

```
# wget -nv -O - http://localhost/foo
```

```
Hello World...
```

```
# chcon -t admin_home_t /var/www/html/foo
```

```
# wget -nv -O - http://localhost/foo
```

```
2011-11-28 20:03:59 ERROR 403: Forbidden.
```

```
# ausearch -m avc -ts today 08:00:00|grep httpd
```

```
type=AVC msg=audit(1346133661.562:869): avc: denied { getattr } for p
```

```
comm="httpd" path="/var/www/html/foo" dev=sda3 ino=9344
```

```
scontext=unconfined_u:system_r:httpd_t:s0
```

```
tcontext=system_u:object_r:admin_home_t:s0
```

Example: Confined user

```
# id -Z
```

```
user_u:user_r:user_t:s0
```

```
# ping -c1 www.redhat.de
```

```
PING www.redhat.de (209.132.183.88) 56(84)  
bytes of data.
```

```
# semanage login -m -s guest_u tscherf
```

```
# id -Z
```

```
guest_u:guest_r:guest_t:s0
```

```
# ping -c1 www.redhat.de
```

```
ping: icmp open socket: Permission denied
```

How does this work?

- Or - How do we transition from an unconfined to a confined process?

id -Z

unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

ls -lZ /etc/init.d/httpd

-rwxr-xr-x. root root system_u:object_r:httpd_initrc_exec_t:s0
/etc/init.d/httpd

sestatus -T -s unconfined_t -t initrc_exec_t

type_transition unconfined_t httpd_initrc_exec_t : process
initrc_t;

sestatus -T -s initrc_t -t httpd_exec_t

type_transition initrc_t httpd_exec_t : process httpd_t;

ps -efZ | grep httpd

unconfined_u:system_r:httpd_t:s0 root root 00:00:00
/usr/sbin/httpd

Check the policy

- httpd_t is allowed to access httpd_sys_content_t....

```
# sesearch --allow --direct -c file -s httpd_t -t  
httpd_sys_content_t
```

```
allow httpd_t httpd_sys_content_t : file { ioctl read getattr lock  
open } ;
```

- ...but not admin_home_t

```
# sesearch --allow --direct -c file -s httpd_t -t  
admin_home_t
```

Policy customization

- Modular instead of monolithic
- No need to access policy source anymore...
- ...semanage can do the job for you

```
# semanage port -l|grep http
```

```
http_port_t          tcp      80, 443, 488, 8008, 8009, 8443
```

```
# semanage port -a -t http_port_t -p tcp 8888
```

```
# semanage port -l|grep http
```

```
http_port_t          tcp      8888, 80, 443, 488, 8008, 8009,  
8443
```

```
# semanage fcontext --add --type 'public_content_rw_t'  
'/www(/.*)?'
```

```
# restorecon -Rv /www
```

Policy customization II

- Check audit.log for AVC deny messages
 - Raw log messages
- Setroubleshoot daemon
 - Easy to read log messages
- Disable don't audit rules
 - # semanage donaudit on|off
- If you have a new file-type, use semanage to add it to the policy and use restorecon afterwards
- Use restorecon over chcon whenever possible
 - # restorecon -v /var/www/html/foo

Policy customization III

- Create a local module to fix problems with the policy
 - Use audit2allow to create a policy file
 - Make yourself familiar with interfaces
 - Use interfaces

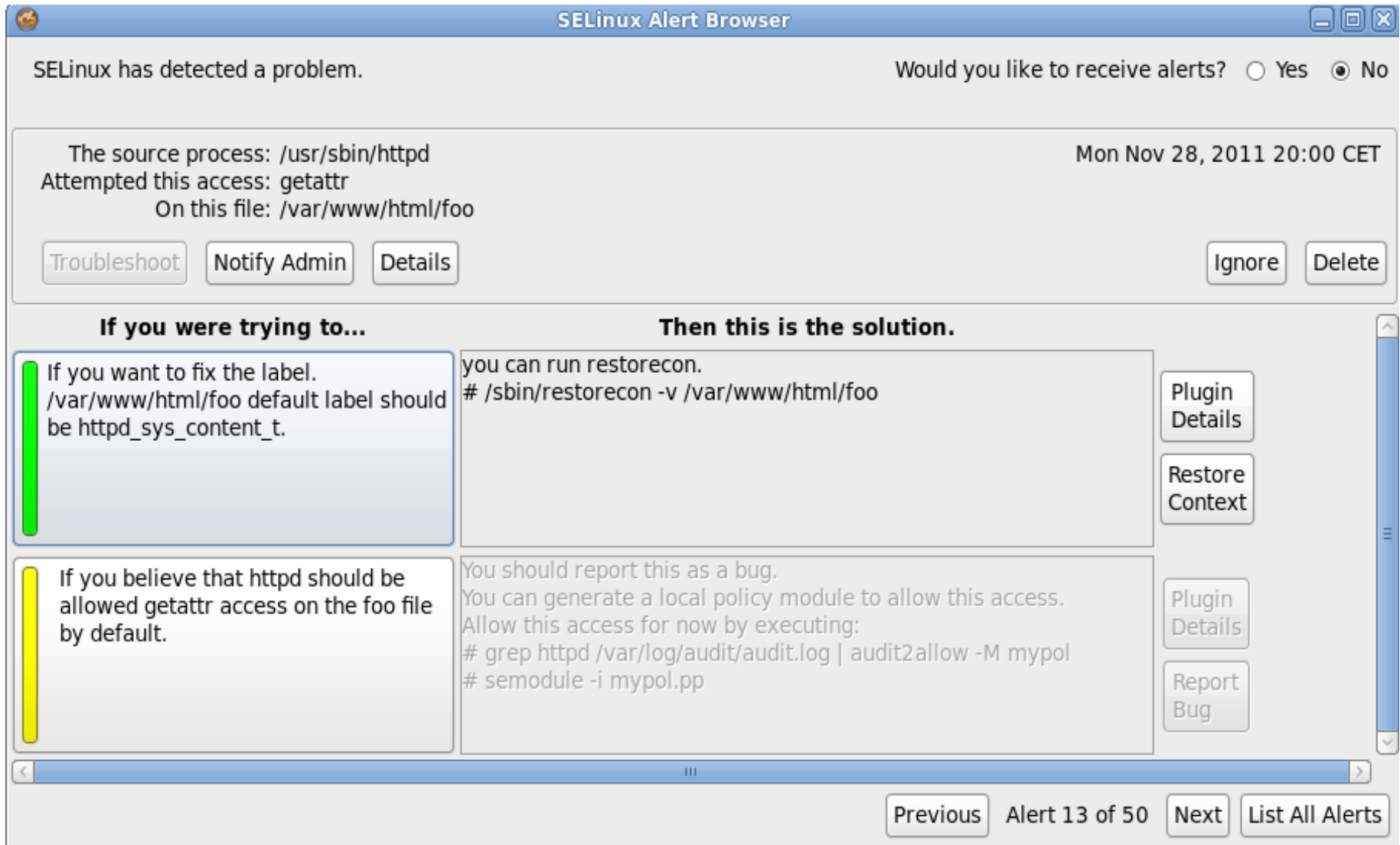
```
# cat /var/log/audit/audit.log | audit2allow -R -m local > local.te
```

```
# cat local.te
```

```
policy_module(local, 1.0)  
    gen_require(  
        type myapp_t;  
        type etc_t;  
    );  
files_read_etc_files(myapp_t)
```

- Review local.te and customize as necessary

Graphical log analyzer



SELinux Alert Browser

SELinux has detected a problem. Would you like to receive alerts? ☐ Yes ☒ No

The source process: /usr/sbin/httpd Mon Nov 28, 2011 20:00 CET
Attempted this access: getattr
On this file: /var/www/html/foo

[Troubleshoot](#) [Notify Admin](#) [Details](#) [Ignore](#) [Delete](#)

If you were trying to...	Then this is the solution.
<div><div></div><div>If you want to fix the label. /var/www/html/foo default label should be httpd_sys_content_t.</div></div>	<div><div>you can run restorecon. # /sbin/restorecon -v /var/www/html/foo</div><div>Plugin Details Restore Context</div></div>
<div><div></div><div>If you believe that httpd should be allowed getattr access on the foo file by default.</div></div>	<div><div>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 httpd /var/log/audit/audit.log audit2allow -M mypol # semodule -i mypol.pp</div><div>Plugin Details Report Bug</div></div>

[Previous](#) Alert 13 of 50 [Next](#) [List All Alerts](#)

Booleans

- Can also be used to change policy “on the fly”
- Don't trust your users?
- Simply put them into user_r role and deny content execution in /home and /tmp

```
# getsebool -a |grep user_exec
```

```
allow_user_exec_content --> on
```

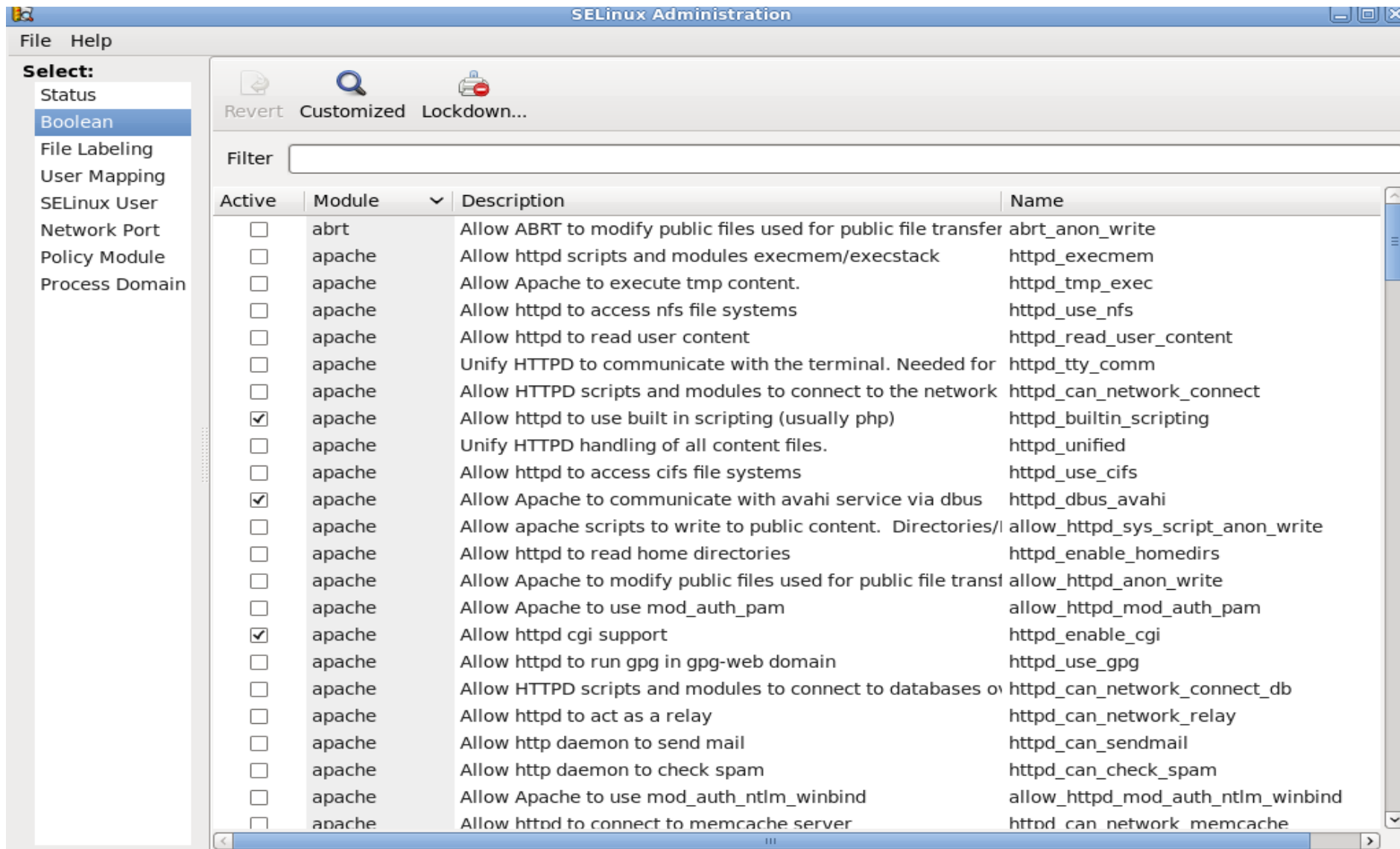
```
# setsebool -P allow_user_exec_content off
```

```
# cd ~
```

```
# ./virus
```

```
./virus: Permission denied
```

Graphical management tool



The screenshot shows the SELinux Administration window. The 'Boolean' tab is selected in the left sidebar. The main area displays a table of SELinux booleans for the 'apache' module. The table has columns for 'Active', 'Module', 'Description', and 'Name'. The 'Active' column contains checkboxes, some of which are checked. The 'Module' column lists 'abrt' and 'apache'. The 'Description' column provides details for each boolean. The 'Name' column lists the corresponding SELinux boolean names.

Active	Module	Description	Name
<input type="checkbox"/>	abrt	Allow ABRT to modify public files used for public file transfer	abrt_anon_write
<input type="checkbox"/>	apache	Allow httpd scripts and modules execmem/execstack	httpd_execmem
<input type="checkbox"/>	apache	Allow Apache to execute tmp content.	httpd_tmp_exec
<input type="checkbox"/>	apache	Allow httpd to access nfs file systems	httpd_use_nfs
<input type="checkbox"/>	apache	Allow httpd to read user content	httpd_read_user_content
<input type="checkbox"/>	apache	Unify HTTPD to communicate with the terminal. Needed for	httpd_tty_comm
<input type="checkbox"/>	apache	Allow HTTPD scripts and modules to connect to the network	httpd_can_network_connect
<input checked="" type="checkbox"/>	apache	Allow httpd to use built in scripting (usually php)	httpd_builtin_scripting
<input type="checkbox"/>	apache	Unify HTTPD handling of all content files.	httpd_unified
<input type="checkbox"/>	apache	Allow httpd to access cifs file systems	httpd_use_cifs
<input checked="" type="checkbox"/>	apache	Allow Apache to communicate with avahi service via dbus	httpd_dbus_avahi
<input type="checkbox"/>	apache	Allow apache scripts to write to public content. Directories/	allow_httpd_sys_script_anon_write
<input type="checkbox"/>	apache	Allow httpd to read home directories	httpd_enable_homedirs
<input type="checkbox"/>	apache	Allow Apache to modify public files used for public file trans	allow_httpd_anon_write
<input type="checkbox"/>	apache	Allow Apache to use mod_auth_pam	allow_httpd_mod_auth_pam
<input checked="" type="checkbox"/>	apache	Allow httpd cgi support	httpd_enable_cgi
<input type="checkbox"/>	apache	Allow httpd to run gpg in gpg-web domain	httpd_use_gpg
<input type="checkbox"/>	apache	Allow HTTPD scripts and modules to connect to databases o	httpd_can_network_connect_db
<input type="checkbox"/>	apache	Allow httpd to act as a relay	httpd_can_network_relay
<input type="checkbox"/>	apache	Allow http daemon to send mail	httpd_can_sendmail
<input type="checkbox"/>	apache	Allow http daemon to check spam	httpd_can_check_spam
<input type="checkbox"/>	apache	Allow Apache to use mod_auth_ntlm_winbind	allow_httpd_mod_auth_ntlm_winbind
<input type="checkbox"/>	apache	Allow httpd to connect to memcache server	httpd_can_network_memcache

Policy development

- Easy to build new modules
- Again, no need to access existing policy source anymore...
- ...just create a new module for your own application

```
# /usr/bin/sepolgen -t 3 /usr/bin/foo foo
```

Created the following files in ./ :

foo.te: Type Enforcement file

foo.if: Interface file

foo.fc: File Contexts file

foo.sh: Setup Script

Policy development II

- There is also a graphical tool available
 - selinux-polgengui



The screenshot shows a window titled "SELinux Policy Generation Tool". Inside the window, there is a heading "Select the policy type for the application or user role you want to confine:". Below this heading, there are three columns of radio button options: "Applications", "Login Users", and "Root Users".

Applications	Login Users	Root Users
<input checked="" type="radio"/> Standard Init Daemon	<input type="radio"/> Existing User Roles	<input type="radio"/> Root Admin User Role
<input type="radio"/> DBUS System Daemon	<input type="radio"/> Minimal Terminal User Role	
<input type="radio"/> Internet Services Daemon (inetd)	<input type="radio"/> Minimal X Windows User Role	
<input type="radio"/> Web Application/Script (CGI)	<input type="radio"/> User Role	
<input type="radio"/> User Application	<input type="radio"/> Admin User Role	
<input type="radio"/> Sandbox		

At the bottom of the window, there are three buttons: "Cancel", "Back", and "Forward". A mouse cursor is pointing at the "Back" button.

sVirt - Securing your virtual machines and containers

- KVM processes have a unique security label (svirt_t:c1,c2)
- Container processes also have a unique label (svirt_lxc_net_t:c1,c2)
- Isolate virtual guests and container processes using SELinux security policy
- MCS Categories are used to define access control on objects
- Integrated into libvirt tools (virt-install, virt-manager)

SELinux and Networking

- Policy Based packet filtering
 - Netfilter framework “tags” IP packets with security context
 - SELinux policy is used for access control
 - Example: `http_server_packet_t` (port 443) is only readable by `httpd_t` but not from `sshd_t`
- IPSec based Labeling
 - Implements access control between local and remote processes
 - Needs IPSec
 - Security Policy Database (SPD) contains SELinux label for established Security Associations (SA)

Memory protection

- The following error sounds familiar to you?
 - **error while loading shared libraries:
/usr/lib/libfoo.so.42:
cannot restore segment prot after reloc: Permission denied**
- SELinux does memory checks also on unconfined processes
- Bad libraries try bad things like text relocations – SELinux prevents this
- Best to file a bug report against the software
- If you need to workaround the problem

```
# /usr/sbin/semanage fcontext -a -t textrel_shlib_t  
'/usr/lib/libfoo.so.42'  
# restorecon -v /usr/lib/libfoo.so.42
```

