SELinux

It is all about the labels.

Who am I?

Open Source Advocate
Instructor
Consultant
Student, Tester, Volunteer

laubersm (twitter, freenode, fedoraproject)
sml@laubersolutions.com
https://github.com/laubersm/LauberSolutions

Where is SELinux

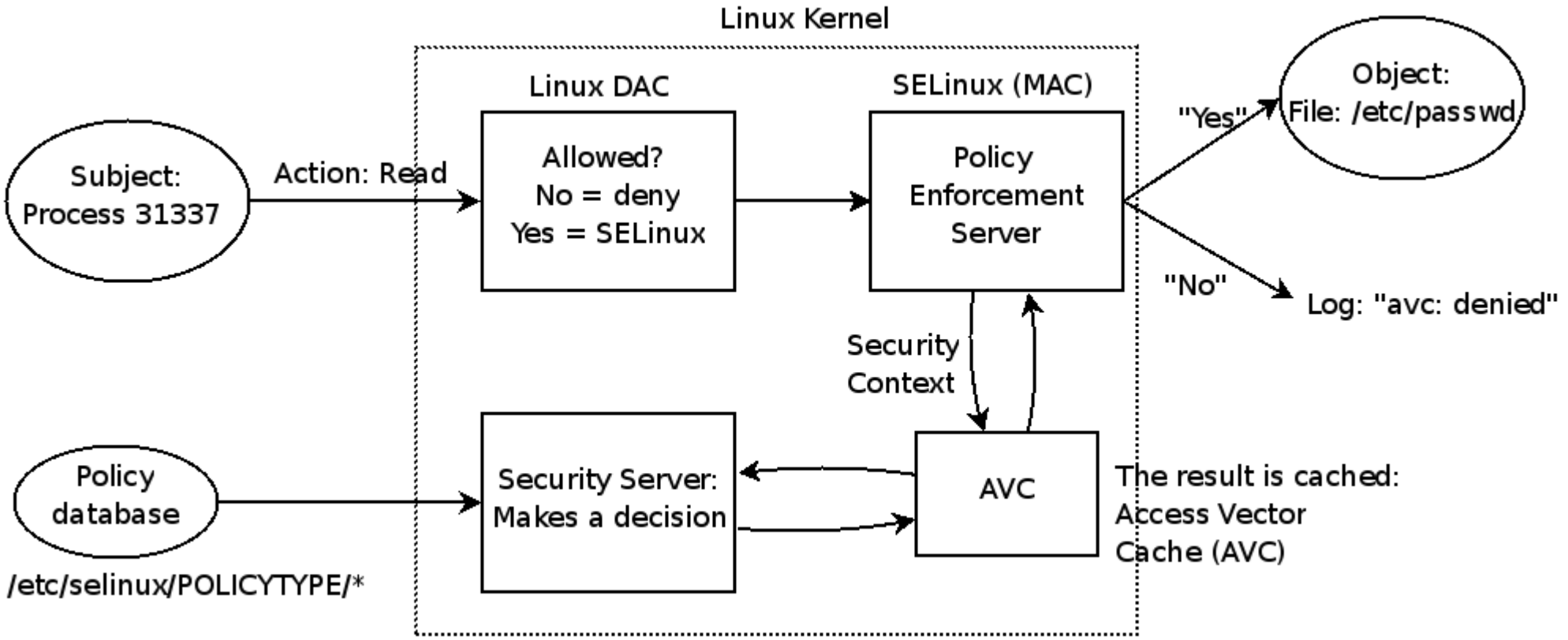
CentOS

redhat.

DAC vs MAC

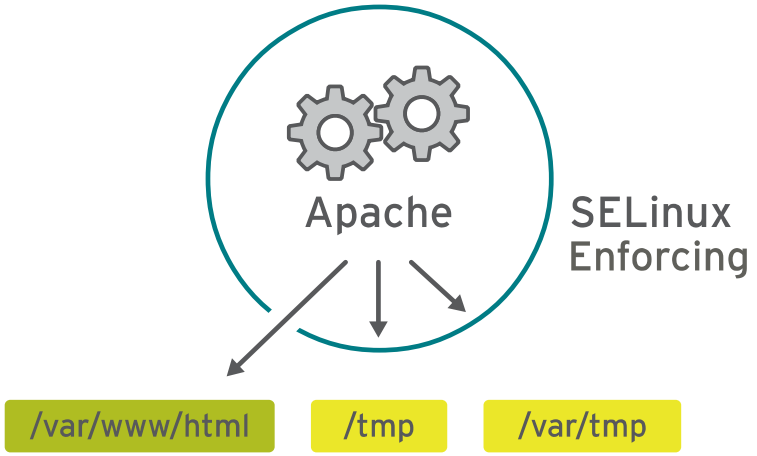
$ chmod -R 777 $HOME

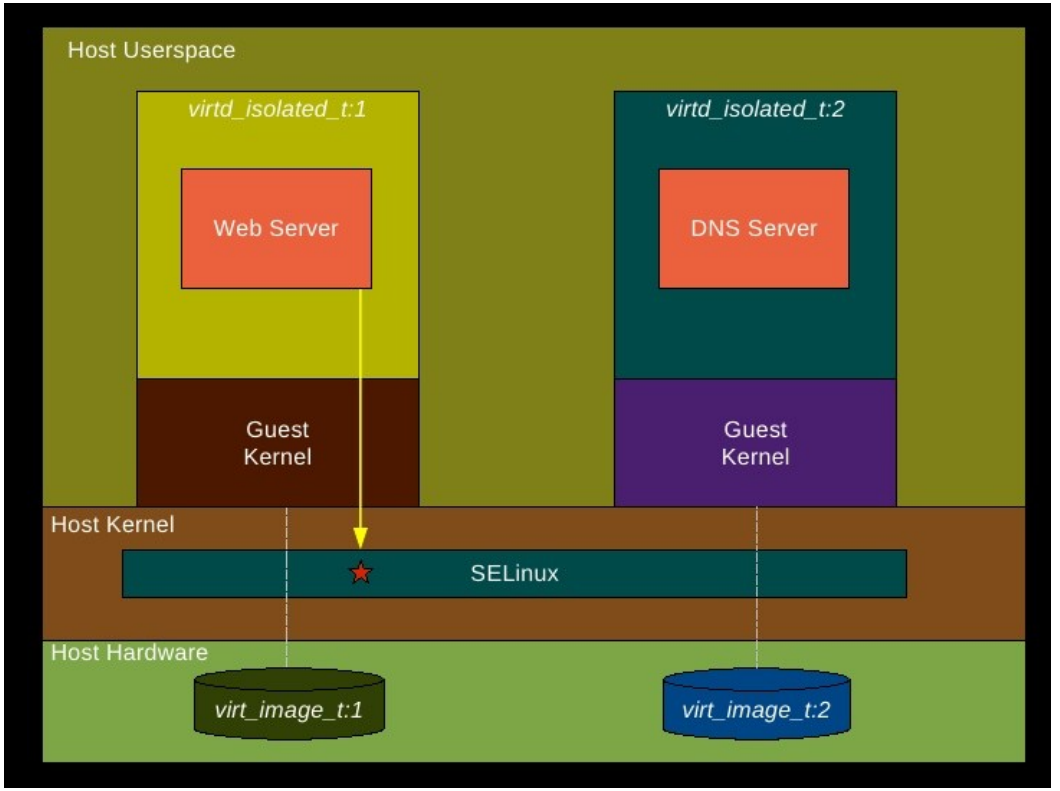
Both DAC and MAC

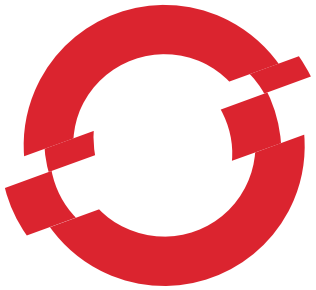


Source: docs.openstack.org

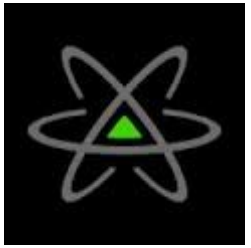
Not just “The Government”



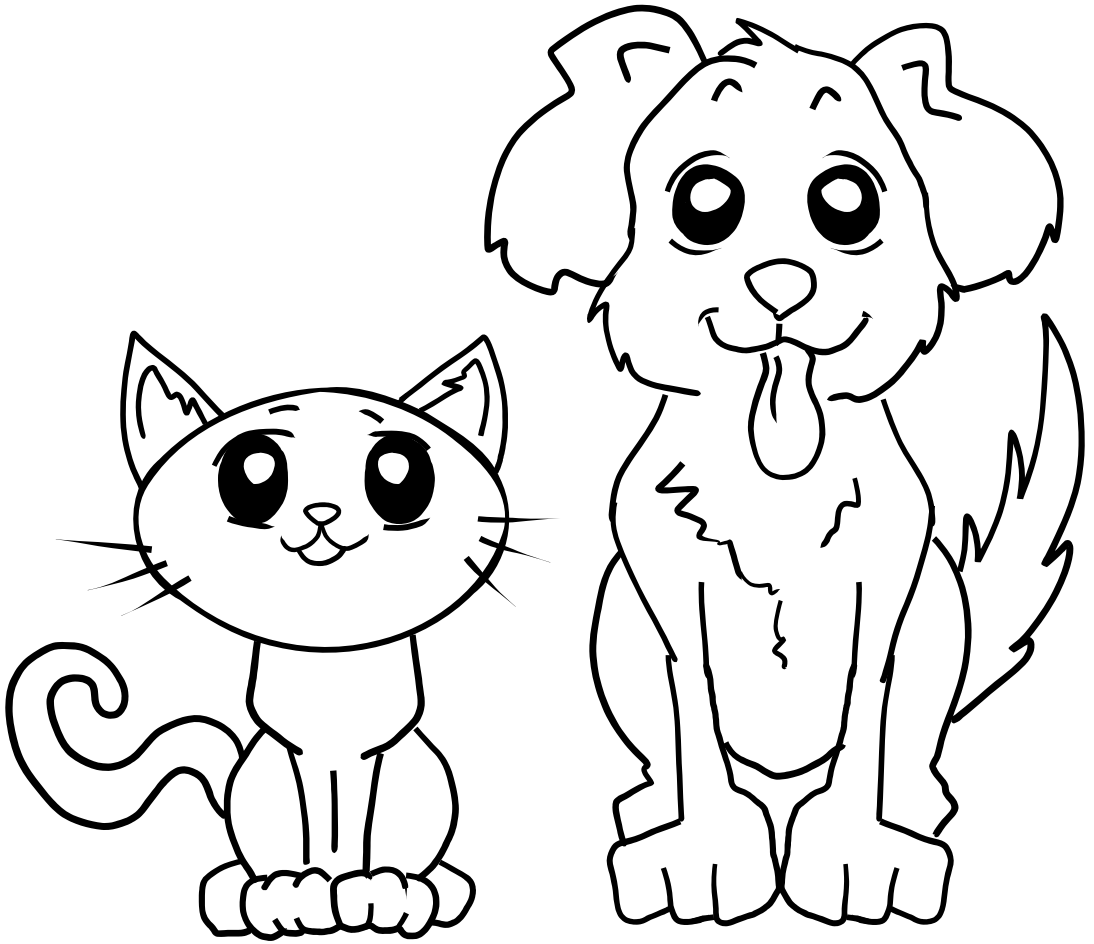


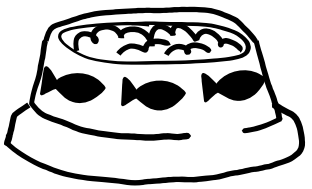


OPENSHIFT

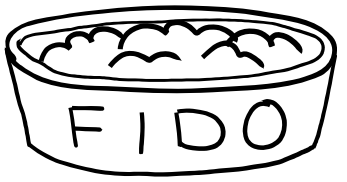


Type Enforcement





CAT_CHOW



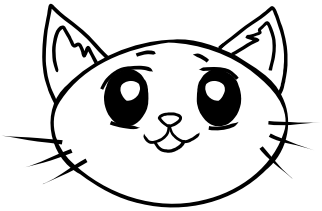
DOG_CHOW

ALLOW

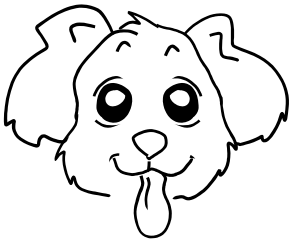
ALLOW

+

+



CAT

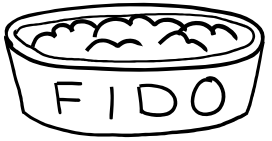


DOG

+

+

CAT_CHOW : FOOD



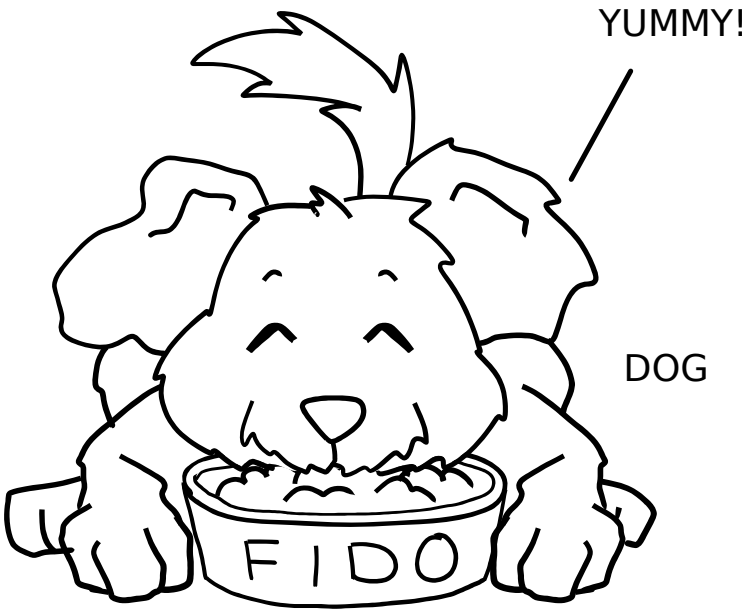
DOG_CHOW : FOO

+

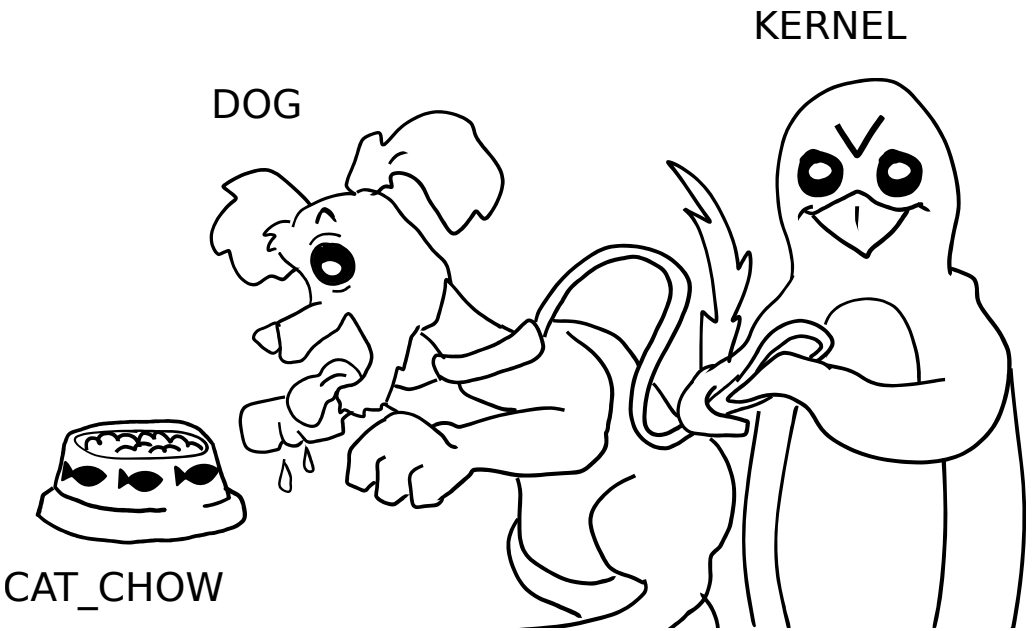
+

EAT

EAT



not allowed = DENIED



What does this look like on a system?

NOTE: These examples are from Fedora and Red Hat Enterprise Linux. Other distributions have different default policies.

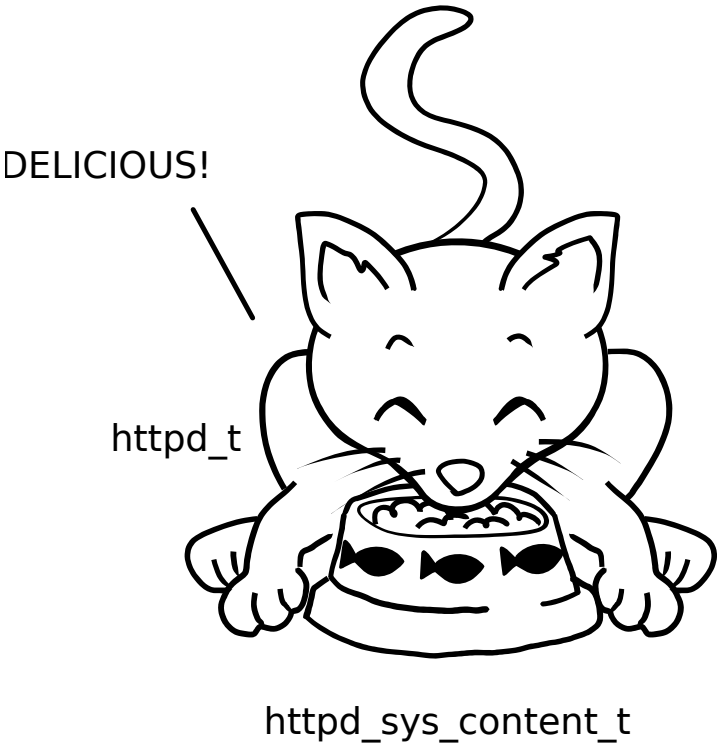
$ sesearch --allow -s httpd_t -t httpd_sys_content_t

Found 15 semantic av rules:
allow httpd_t file_type : filesystem getattr ;
allow httpd_t file_type : dir { getattr search open } ;
allow daemon httpd_sys_content_t : dir { getattr search open } ;

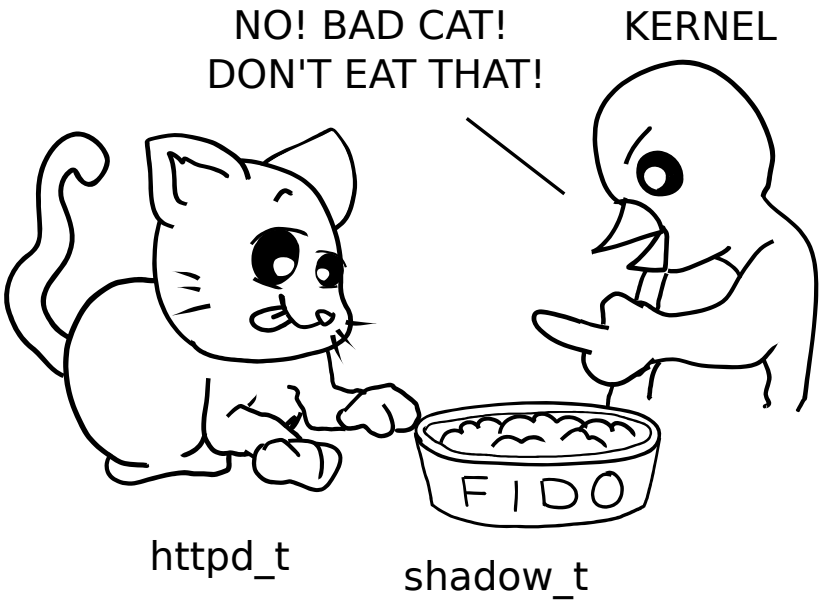
allow httpd_t httpd_sys_content_t : file { ioctl read getattr lock open } ;

allow httpd_t httpd_sys_content_t : dir { ioctl read getattr lock search open } ;
allow httpd_t httpd_sys_content_t : lnk_file { read getattr } ;
allow httpd_t httpd_content_type : file { ioctl read getattr lock open } ;
allow httpd_t httpd_content_type : dir { getattr search open } ;
allow httpd_t httpd_sys_content_t : dir { ioctl read write getattr lock add_name remove_name search open } ;

```
$ ps -ZU apache
LABEL                             PID TTY          TIME CMD
system_u:system_r:httpd_t:s0     4876 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0     4877 ?        00:00:00 httpd

$ ls -Z /var/www/html/
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/index.html
```

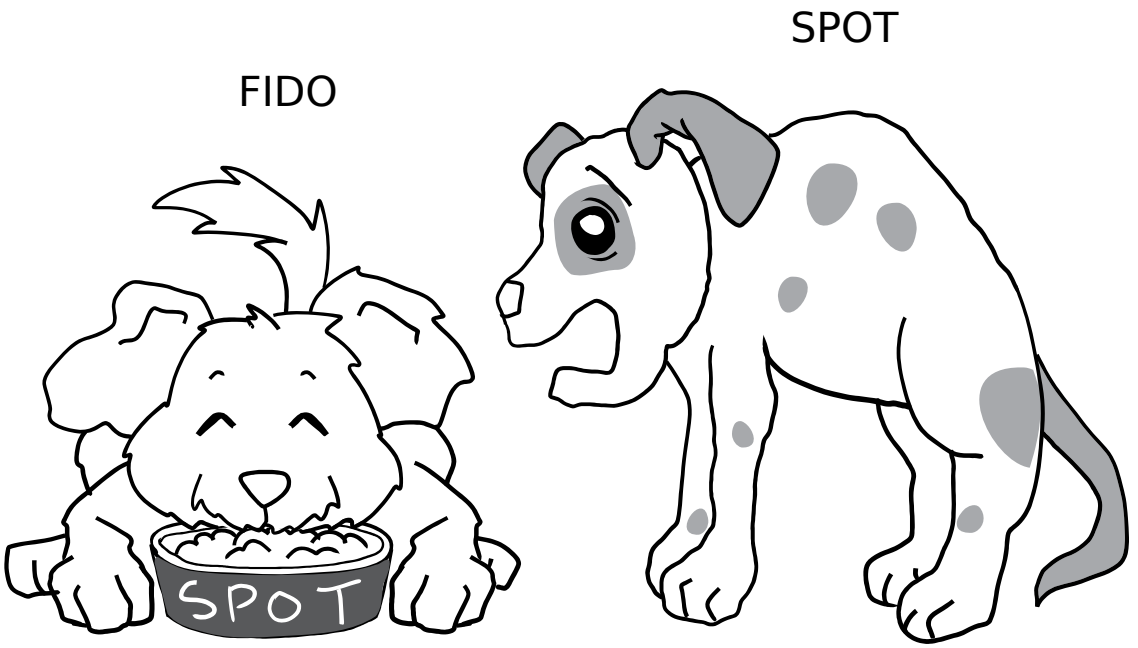


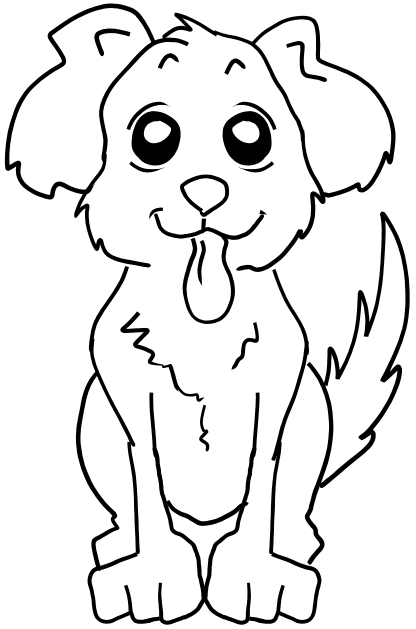
```
$ sesearch --allow -s httpd_t -t shadow_t
Found 2 semantic av rules:
   allow httpd_t file_type : filesystem getattr ;
   allow httpd_t file_type : dir { getattr search open } ;

$ ls -Z /var/www/html/
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/index.html

$ ls -Z /etc/shadow
system_u:object_r:shadow_t:s0 /etc/shadow
```



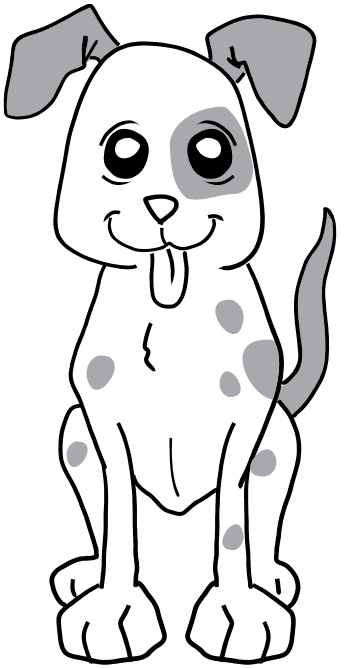
Wait! There's more.

Multi Category Security (MCS)

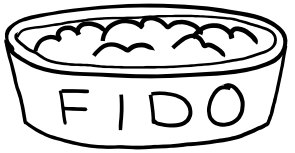




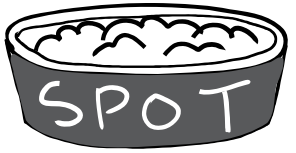
DOG:RANDOM1



DOG:RANDOM2



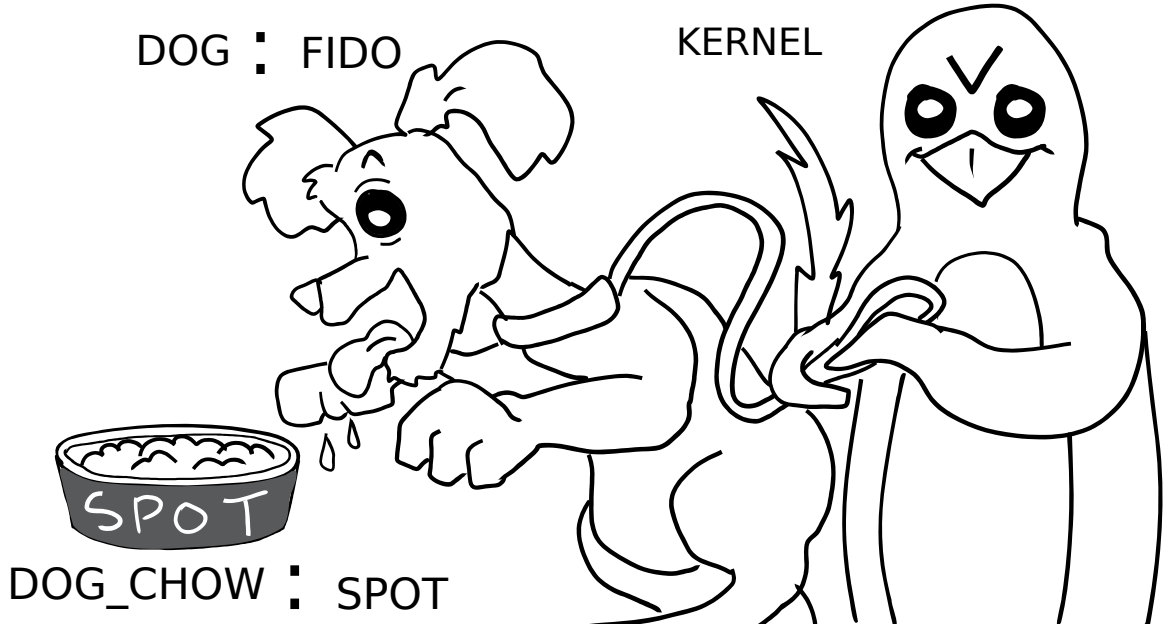
DOG_CHOW:
RANDOM1

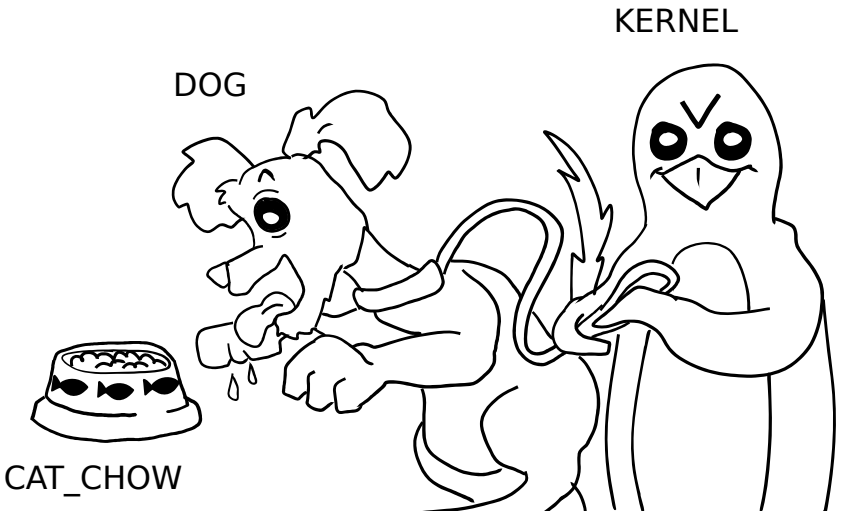


DOG_CHOW:
RANDOM2

DOG : RANDOM1

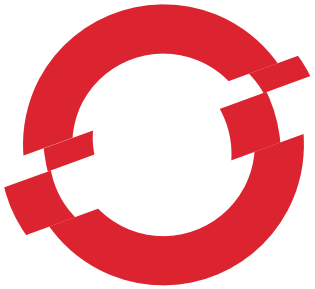
DOG_CHOW : RANDOM


DOG : FIDO
KERNEL
SPOT
DOG_CHOW : SPOT


KERNEL
DOG
CAT_CHOW

What does MCS look like on a system?

sandbox



A VM can only access its own disk

$ ps -ef -Z | grep qemu
system_u:system_r:svirt_t:s0:c189,c390 qemu 27671 1 99 17:36 ?
    00:02:44 /usr/bin/qemu-system-x86_64 -machine accel=kvm -name
AdminLocal … file=AdminLocal.raw ...

$ ls -Z AdminLocal.raw
system_u:object_r:svirt_image_t:s0:c189,c390 AdminLocal.raw

$ ls -Z RHcurr.img
system_u:object_r:svirt_image_t:s0:c290,c831 RHcurr.img

Network connections have context too!

$ ps -ef -Z | grep qemu

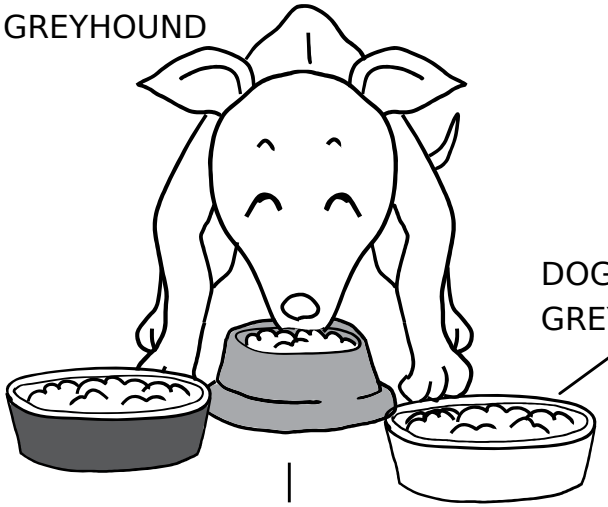
system_u:system_r:svirt_t:s0:c189,c390 qemu 27671 1 99 17:36 ?
00:02:44 /usr/bin/qemu-system-x86_64 -machine accel=kvm -name
AdminLocal … file=AdminLocal.raw ...

netstat -tZ | egrep '5901|5900'

tcp 0 0 127.0.0.1:5901 127.0.0.1:49865
ESTABLISHED 27671/qemu-system-x
system_u:system_r:svirt_t:s0:c189,c390

tcp 0 0 127.0.0.1:5900 127.0.0.1:36699
ESTABLISHED 3672/qemu-system-x8
system_u:system_r:svirt_t:s0:c290,c831

Multi Level Security (MLS)



DOG : CHIHUAHUA

DOG_CHOW: CHIHUAHU

Administration Scenario

Policy: default “targeted”

Content: custom location

Administration Scenario

Add file context for everything under /web

```
# semanage fcontext -a -t httpd_sys_content_t "/web(/.*)?"
# restorecon -R -v /web
```

OR match an existing file context.

```
# semanage fcontext -a -e /var/www /web
# restorecon -R -v /web
```

$ man semanage-fcontext

Administration Scenario

Policy: default “targeted”

Content: allow ftp dropbox

Administration Scenario

Add a read/write file-context for everything under /upload

```
# semanage fcontext -a -t public_content_rw_t "/upload(/.*)?"
```

```
# restorecon -R -v /upload
```

Enable allow rules with boolean

```
# semanage boolean -m --on ftpd_anon_write
```

$ man semanage-boolean

Administration Scenario

Policy: default “targeted”

Content: run service on custom port

Administration Scenario

Allow sshd to listen on tcp port 8991

```
# semanage port -a -t ssh_port_t -p tcp 8991
```

```
$ man semanage-port
```

Administration Scenario

Policy: default “targeted”

Place a particular module into permissive mode.

Administration Scenario

List all permissive modules

semanage permissive -l

Make httpd_t (Web Server) a permissive domain

semanage permissive -a httpd_t

$ man semanage-permissive

Logs, logs, logs (and a bit of troubleshooting)

```
# grep avc /var/log/audit/audit.log

type=AVC msg=audit(1439679445.509:7156):
avc:  denied  { read } for  pid=4878
comm="httpd" name="index.html" dev="dm-1"
ino=57673974
scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:user_home_t:
s0 tclass=file permissive=0
```

Got setroubleshoot?

/var/log/messages or journalctl will have a message similar to:

setroubleshoot[5564]: SELinux is preventing /usr/sbin/httpd from read access on the file index.html. For complete SELinux messages. run sealert -l ad4801f3-7f10-4191-a85f-b95d9de40ac1

python[5564]: SELinux is preventing /usr/sbin/httpd from read access on the file index.html.

***** Plugin catchall_boolean (89.3 confidence) suggests *********

If you want to allow httpd to read user content
Then you must tell SELinux about this by enabling the 'httpd_read_user_content' boolean.

Do
setsebool -P httpd_read_user_content 1

When all else fails...

getenforce
setenforce 0
<test, test, test>
setenforce 1

man audit2allow

Does it really protect a system?

Shellshock

“SELinux does not block the exploit but it would prevent escalation of confined domains”

-Dan Walsh http://danwalsh.livejournal.com/71122.html

https://securityblog.redhat.com/2014/09/26/frequently-asked-questions-about-the-shellshock-bash-flaws/

http://cybermatters.info/2014/10/09/shellshock-selinux/ (interview with Dan)

https://blog.hqcodeshop.fi/archives/243-SElinux-and-Shellshock.html

CVE-2014-6271 and CVE-2014-7169 and CVE-2014-7186 and CVE-2014-7187

Vemon

“sVirt provides good anti-venom for this flaw”
-Dan Berrange (creater of libvirt)

http://danwalsh.livejournal.com/71489.html

https://securityblog.redhat.com/2015/05/13/venom-dont-get-bitten/

CVE-2015-3456

Svirt to the rescue is not new!

CVE-2011-1751 2011
http://danwalsh.livejournal.com/45194.html

PDFs, Browsers, and more

“Why we don't confine Firefox with SELinux”
By Dan Walsh
http://danwalsh.livejournal.com/72697.html

CVE-2015-4495 Aug 2015

Consider sandbox tools to confine any application as needed.

More:

http://selinuxproject.org/page/NewUsers

https://www.nsa.gov/research/selinux/

http://oss.tresys.com/
https://github.com/TresysTechnology/refpolicy/wiki
https://github.com/TresysTechnology/setools3/wiki

Thanks!

Máirín Duffy

Dan Walsh

https://github.com/mairin/selinux-coloring-book

http://stopdisablingselinux.com

Slides available at:
http://github.com/laubersm/LauberSolutions