CentOS

Managing Selinux on CentOS with your cfgmgmt solution

(puppet and ansible covered)

Fabian Arrotin

arrfab@centos.org

CentOS

/whois arrfab

- Belgian guy
- SysAdmin by choice
- CentOS (ab)user for a long time
- CentOS Project member

CentOS

Agenda

- Selinux overview
- Contexts modification
- Booleans
- Building and distributing custom selinux policies

CentOS

Why selinux ?

“ security is a chain; it’s only as secure as the weakest link ”

CentOS

Selinux in some keywords

- DAC/ACLs vs MAC
- All about contexts !
- Disabled/permissive/enabled
  - Targeted (only identified services are confined - default)
  - Strict/MLS (use at your own risk :-))
- In almost all distributions now

CentOS

Selinux useful commands

- {get,set}enforce , sestatus
- Traditional commands with -Z (ls -Z, ps -Z, ...)
- chcon/restorecon (needs semanage fcontext -a)
- semanage (swiss knife)
- sealert, audit2why, audit2allow
- Selinux-policy-devel ; man $topic_selinux

CentOS

But before the 'deploy step', test !

(rule applies also to selinux changes)

CentOS

Managing selinux state – puppet

No puppet selinux resource type, but:

- Erb template
- Augeas resource type :

```
augeas {"/etc/sysconfig/selinux" :
  context => "/files/etc/sysconfig/selinux",
  changes => "set SELINUX enforcing",
}
```

- Exec resource type :

```
exec { "Selinux in enforcing mode":
 command => "/usr/sbin/setenforce 1",
 unless => "/usr/sbin/getenforce |grep Enforcing",
}
```

CentOS

Managing selinux state – ansible

- Important remark *(not needed anymore on 7)* :

```
ansible -m yum -a "pkg=libselinux-python state=installed" all
```

- selinux module (listed in the system modules)

```
- name: Ensuring selinux is enforced
  selinux: policy=targeted state=enforcing
```

- shell/command module (with the register: feature from previous tasks)

CentOS

Contexts – puppet

- File resource type can handle it :

```
file {'/var/www/html/vhost1/cgi-bin/blabla.pl':
  source => 'puppet:///modules/bla/blabla.pl',
  mode => '0755',
  owner => apache,
  group => apache,
  seltype => "httpd_user_script_exec_t",
}
```

- By default puppet will try to use the correct context
- Doesn't add it to defaults ! (so “`semanage fcontext`”)
- You can ignore that (to avoid matchpathcon):

```
selinux_ignore_defaults => on,
```

CentOS

Contexts – ansible

- By default ansible will try to use the correct context
- Doesn't add it to defaults ! (so “`semanage fcontext`”)
- The 'file' module (and all derived modules) can handle selinux contexts just “fine” :

```
- name: Creating the correct incoming folder
      file: |
        path=/incoming
        owner=root group=sftpusers
        mode=0750
        setype=public_content_rw_t
        state=directory
```

CentOS

Booleans – puppet

- Through the 'selboolean' resource :

```
if $selinux == 'true' {
  selboolean {'httpd_enable_homedirs':
    value => on,
    persistent => true,
  }
}
```

CentOS

Booleans – ansible

- Through the 'seboolean' module :

```
- name: Ensuring httpd can reach network ports
  seboolean: name=httpd_can_network_connect state=yes persistent=yes
```

CentOS

Note about ports - puppet

- No seport puppet resource but :
- Policycoreutils-python (to provide semanage)
- Exec resource (through defined types)

```
exec { "add_${context}_${port}":
  Command => "/usr/sbin/semanage port -a -t ${context} ${protocol_switch}${port}",
  Unless => "/usr/sbin/semanage port -l|grep \"^${context}.*${protocol}.*${port}\"",
}
```

CentOS

Note about ports - ansible

- No seport ansible module but :
- Policycoreutils-python (to provide semanage)
- Register output and using the when: feature

```
- name: Checking if selinux authorizes http_port_t to tcp 8082
  shell: /usr/sbin/semanage port --list|grep "^http_port_t.*tcp.*8082"
  register: selinux_port_check
  ignore_errors: true
- name: Adding the port to selinux managed port if needed
  shell: /usr/sbin/semanage port -a -t http_port_t -p tcp 8082
  when: selinux_port_check|failed
```

CentOS

What if that's not enough ?

CentOS

What to do (and not)

- Disable selinux => no
- Permissive mode => yes/no
- Permissive mode for *only* the concerned domain => yes

```
semanage permissive -a zabbix_agent_t
```

- Audit/analyze/compile/test new policy

CentOS

Building custom selinux policies

- Required when no context/boolean can solve it
- When a new policy blocks your application when you're sure it would have to be allowed (exemple zabbix_agent_t)
- Clean machine (dev environment) with selinux-policy-targeted
- Audit2allow, audit2why
- Produce a .te (and/or .fc) and not directly a .pp (store it in your VCS)
- Build the policy .pp
- Test, test, test, test, ... rinse/repeat

CentOS

Building the selinux .pp file

- Put {clean,isolated} machine in permissive mode
- Launch your application
- Analyze audit.log
  - `grep denied /var/log/audit/audit.log|audit2allow -m mypolicy`
- Review the .te file, rinse/repeat
- Build the pp file
  - `make -f /usr/share/selinux/devel/Makefile mypolicy.pp`

CentOS

Distributing policies : overview

- Use a dedicated folder to store your .pp files
- Not under /etc/selinux/targeted (deleted/dynamic !)
- Distribute your policies
- ! with latest selinux-policy-targeted !
- Load them, enjoy

CentOS

Distributing policies : puppet

- Distribute files / load policies

```
file {"/etc/selinux/local-policies/custom-policy1.pp":
  ensure => file,
  owner => root,
  group => root,
  require => File['/etc/selinux/local-policies'],
  source => "puppet:///modules/selinux/$os_major_ver/custom-policy1.pp",
}

selmodule {"custom-policy1":
  ensure => present,
  selmoduledir => "/etc/selinux/local-policies/",
  syncversion => true,
  require => File['/etc/selinux/local-policies/custom-policy1.pp'],
}
```

CentOS

Distributing policies : ansible

- Distribute files / register diff / load :

```
- name: Creating selinux custom policy drop folder
  file: path={{ custom_selinux_dir }} state=directory owner=root group=root mode=0750

- name: Distributing custom selinux policies
  copy: src=../files/selinux/policies/{{ ansible_distribution_version[0] }}/{{ item }}
dest={{ custom_selinux_dir}}/{{ item }}
  with_items:
    - custom-policy1.pp
    - custom-policy2.pp
  register: custom_policies_output

- name: Reloading custom selinux policy files
  shell:  /usr/sbin/semodule -u {{custom_selinux_dir}}/{{ item.item }}
  with_items: custom_policies_output.results
  when: item.changed
```

CentOS

Q&A

Questions ?
Thank you !

CentOS