SELinux

Igor Vuk

What are we going to talk about?

- Overview
- Command line tools
- GUI tools

Overview (1/3)

- Security enhancement to the GNU/Linux OS
- Mandatory Access Control(MAC) framework
- Shipped by Fedora, RHEL{4,5,6}, Debian, …
- Provides the mechanism for supporting access control security policies, including US DoD mandatory access controls, through the use of LSM in the Linux kernel

Overview (2/3)

- The goal is to create a better form of system security
  - Tries to protect you from bugs in applications
- The restrictions SELinux imposes are mandatory
  - Default policy is deny
  - There is no equivalent of a root user
  - Access rules depend on attributes given to a certain subject and object pair
- The protection stacks with DAC
  - Both are required for an action to be allowed

Overview (3/3)

- Relies on several basic concepts
  - Subjects (i.e. processes)
  - Objects (i.e. files, folders, sockets...)
  - Access vectors (rules)
- Attributes of subjects and objects are called security contexts
- A combination of kernel modules and user-space tools
  - Don't forget about the reference policy
- Licensed under GPL licence

The console (1/18)

- /sys/fs/selinux/
  - policyvers, load, disable
- `cat /proc/<pid>/attr/current`
  - prev
- Basic system tools are patched
  - -Z parameter
  - `ps, id, ls`
- Remember that the security context is stored in extended attributes
  - archiving
  - `tar, star`

The console (2/18)

- `getenforce`
  - http://manpg.es/getenforce
  - Shows the current SELinux operating mode
- `setenforce`
  - http://manpg.es/setenforce
  - Sets the current SELinux operating mode until reboot
  - Enforcing, Permissive (1 or 0)
- `cat /etc/selinux/config`
- selinux=0 as boot parameter
  - `cat /proc/cmdline`

The console (3/18)

- Remember the AVC?
- You don’t want to trigger the security server every time
  - slow
  - large number of queries
- `avcstat`
  - http://manpg.es/avcstat
  - Cache hit rate ~99.91% on my box last time I checked
  - Reads from /sys/fs/selinux/avc/cache_stats

The console (4/18)

- /etc/selinux/
  - Everything important is here
  - `ls /etc/selinux/targeted/policy/`
- `seinfo`
  - http://manpg.es/seinfo
  - Shows information about currently loaded policy
- `sesearch`
  - http://manpg.es/sesearch
  - Searches the currently loaded policy
  - `sesearch -A -t httpd_t`
  - `sesearch -A`

The console (5/18)

- `secon`
  - http://manpg.es/secon
- `restorecon`
  - http://manpg.es/restorecon
  - The command you should absolutely know about
  - Relabels files and folders with proper security context defined in SELinux config
  - `restorecon -Rv ~/VirtualMachines`
- `chcon`
  - http://manpg.es/chcon
  - Changes the security context of a file or folder
  - `chcon -t httpd_log_t ~/apachelogs`

The console (6/18)

- There’s a funny moment now
- If you noticed, we can change the security context of a file or a folder by using `chcon`
- But `restorecon` changes it back to match the data SELinux has in the policy
- Unless the type you chose was “remembered”, it will get trashed
- Consider autorelabeling!
- Use `semanage`

The console (7/18)

- `restorecon` will check two sources of data
  - The policy
  - Local file contexts created by `semanage`
  - http://etbe.coker.com.au/2007/11/13/restorecon-equivalent-for-unix-permissions/
- `semanage`
  - http://manpg.es/semanage
  - Your new best friend
  - `semanage fcontext -a -t httpd_log_t apachelogs/`
  - `restorecon -Rv apachelogs/`

The console (8/18)

- `cat /etc/selinux/targeted/contexts/files/file_contexts.local`
- Note the file_contexts.local.bin file
- `semanage fcontext -d /home/<user>/apachelogs/`
- And it’s gone!

The console (9/18)

- `semanage port -l`
- `semanage port -a -t http_port_t -p udp 81`
- `semanage port -l  | grep 81`
- `semanage port -d -p udp 81`
- `vi /usr/sbin/semanage`
  - It’s Python <3
  - Consider this as an example how to use Python with SELinux

The console (10/18)

- `semanage login -l`
  - Shows the SELinux user to GNU/Linux (local) user mapping
- `semanage user -l`
  - Shows the SELinux users
- `adduser testuser; passwd testuser`
- `semanage login -a -s xguest_u testuser`
- Logout, login as testuser
- `id -Z`
- `semanage login -d testuser`

The console (11/18)

- `fixfiles`
  - http://manpg.es/fixfiles
  - It will relabel all (supported) mounted filesystems by default
  - If you run `fixfiles check`, there's an error for you in it :)
  - `fixfiles verify`
  - `fixfiles onboot`
- `setfiles`
  - http://manpg.es/setfiles
  - Initializes the security context fields
  - Usually run at SELinux installation time

The console (12/18)

- auditd
  - http://manpg.es/auditd
  - You want this one running
  - As far as SELinux is concerned, it will log access violations
  - ‘avc: denied’ etc. (they chose two spaces just so I’d get it wrong every time)
  - `grep ‘avc:  denied’ /var/log/audit/audit.log`
- It logs to /var/log/audit/audit.log
- `aureport`
  - http://manpg.es/aureport

The console (13/18)

- There are some other tools that can parse /var/log/audit/audit.log
- `audit2why`
  - http://manpg.es/audit2why
  - It should show you why something qualifies as an access violation
- `audit2allow`
  - http://manpg.es/audit2allow
  - It should help you generate an adequate SELinux policy module for an application that was forbidden some type of access

The console (14/18)

- If you're not running auditd, the SELinux messages will end up in /var/log/messages
- `ausearch -m avc`
- `audit2why -a`
- `audit2why -wa`
- `audit2why -wave`
- `audit2allow`
  - We'll just check the man for this one

The console (15/18)

- `getsebool`
  - http://manpg.es/getsebool
  - `getsebool -a`
  - `getsebool xguest_exec_content`
- `setsebool`
  - http://manpg.es/setesebool
  - `setsebool xguest_exec_content false`
  - `setsebool xguest_exec content true`
  - `setsebool -P xguest_exec_content true`
- You'll be using this
- They make your work easier

The console (16/18)

- `sandbox`
  - http://manpg.es/sandbox
  - A great tool
  - Enables you to run apps in a sandboxed environment
  - Great for checking out apps which you don't trust (all of them, until proven different)
  - `sandbox elinks http://www.google.com`
  - `sandbox -X firefox`
  - `sandbox -t sandbox_web_t -X firefox`

The console (17/18)

- There’s a lot of stuff you can do without ever messing with the policy
- No m4 involved so far :)
- The compiled policy is located in /etc/selinux/targeted/policy/
- `semodule`
  - http://manpg.es/semodule
  - `semodule -l`
  - `semodule -B`
  - `semodule -R`

The console (18/18)

- `sepolgen`
  - http://danwalsh.livejournal.com/61107.html
  - A tool for generating the inital SELinux policy module
  - A starting point for writing your own SELinux policy modules
  - Added in Fedora 18
  - `sepolgen --init <app>`
  - `sepolgen --application <app>`
- `runcon`
  - http://manpg.es/runcon

The GUI (1/2)

- SLIDE
  - http://oss.tresys.com/projects/slide
  - IDE for SELinux policy writing
  - Developed as a plugin for Eclipse
- `apol`
  - http://oss.tresys.com/projects/setools
  - Tool for analyzing SELinux security policy
- `seaudit`
  - http://oss.tresys.com/projects/setools
  - View audit messages

The GUI (2/2)

- `system-config-selinux`
  - Can manage the state of SELinux, force a relabel on the next reboot, ...

The end

- Thank you for listening :)
- Questions?