SELinux Policy Within Package Managers

Why policy is special

TRESYS TECHNOLOGY

SELinux? Policy? What?

- SELinux is a MAC system for Linux
  - Enabled by default on Fedora, RHEL
  - Available on Ubuntu, Gentoo, Debian, etc
- Policies are available for 290+ apps
  - Typically distributed by distro
  - Customized by distro’s, end users
  - Based on upstream Reference Policy
  - Currently packaged like applications

State of the Art

- Policy distributed by distro’s
  - The vast majority of policy in a single package
- Loading policy via post-script kludges
  - Failures in post-scripts have no rollback
- User intervention sometimes required
  - On application upgrades, policy load failures
- Third parties have few options
  - Separate packages
  - Combined packages with aforementioned hacks

State of the Art

- Ordering issues very common
  - Security labels must be available before install
- Multiple policies can take a long time to load
  - If they are installed from separate packages
- Dependency issues
  - Co-dependent policies must be installed together
- Numerous hacks
  - policy renames, moves between packages

Why policy is different

- Potentially affects entire system
- Must be installed first
  - before affected applications are installed
- Needs to control access to data
  - Even after applications have been removed
    - (e.g., database files may contain sensitive data)
  - Data shared by multiple applications
- Controls interaction between applications
  - IPC, network access, shared data

TRESYS TECHNOLOGY

What we want to do

- Include Policy in distro packages
  - Without hacks
  - Natively support in package manager
- Install policy before affected apps
  - At the beginning of a transaction
  - All policy installed together
  - Be able to back out in case of failure

What we want to do

- Gracefully support corner cases
  - Policy renames
  - Bootstrapping
  - Installing in clean chroot
  - Cross-installs
- Help third parties distribute policy
  - Support multiple policies
    - For different distros, releases, policy types
- Make life with SELinux easier

TRESYS TECHNOLOGY

Work in Progress

- Targeting RPM
  - Since Fedora/RHEL use SELinux by default
  - Already had minimal support
  - Hopefully more open to support
- %Policy directive already present
  - Stores it in RPM header
  - Only supports one policy, no parameters
  - Doesn’t actually do anything with policy

Initial patch set

- Adds policy loading support
- Adds --no-policy flag
- Installs all policy before %pre-trans
- Aborts transaction if policy load fails
  - Policy install also reverted
- Does not uninstall policies with app
  - Remaining data may be sensitive
  - Do not want other apps losing access

TRESYS TECHNOLOGY

Second patch set

- Changes to %Policy directive
  - Policy section in spec with key-value pairs
    - Policy type (MLS, strict, targeted)
    - Obsoletes (for policy renames)
    - Base policy
  - Still stored in header
- Policy rename support
  - Allow policies to obsolete one another

New %Policy Directive

```
%policy
%module poltest-policy-%{version}/foo.pp
  Name: foo
  Types: default
  Obsoletes: bar baz
%module poltest-policy-%{version}/bar.pp
  Name: bar
  Types: mls targeted
  Obsoletes: baz qux
```

Upcoming patch sets

- Chroot installation
  - Cross-install support
  - Falls back to libsemanage interfaces
- Bootstrap support
  - Package declares itself policybootstrap
    - If it is required for policy installation
    - For example, policycoreutils, libsepol
    - Will delay policy installation until the end
  - Only if packages not already present

TRESYS TECHNOLOGY

Upcoming patch sets

- Store policies in RPM database
  - Used for policy renaming
  - Also used for policy-type switching
    - If user wants to switch from targeted to MLS
    - RPM installs MLS policies onto system
- Base module support

Future Work

- Split out functions of a package manager
  - Multiple processes (and security domains)
  - Move vulnerable parts into more strict domains
    - Network-facing components
    - Package parsing
  - Isolate trusted processes from bad input
- Inform admin of what an app can do
  - Based on the policy being installed with it

Future Work

- Various levels of trust
  - Enforce restrictions on package manager
    - Based on who is running it
    - Where the package came from
    - Whether the package is signed
  - End user can specify restrictions
    - Only let a package install in /opt
    - Whether or not it can add users
    - How it can label its application data

Conclusion

- Policy distribution is currently adhoc
  - Full of hacks, inadequacies, etc
- Package managers can help
  - Already have transaction capabilities
  - Already store package metadata
- Some support being sent upstream already
- Lots of corner cases to cover
- Eventually want to raise assurance
- Allow users more control over packages