An Improved SELinux Policy Infrastructure

James Carter
jwcart2@tycho.nsa.gov
National Security Agency
National Information Assurance Research Laboratory
(NIARL)

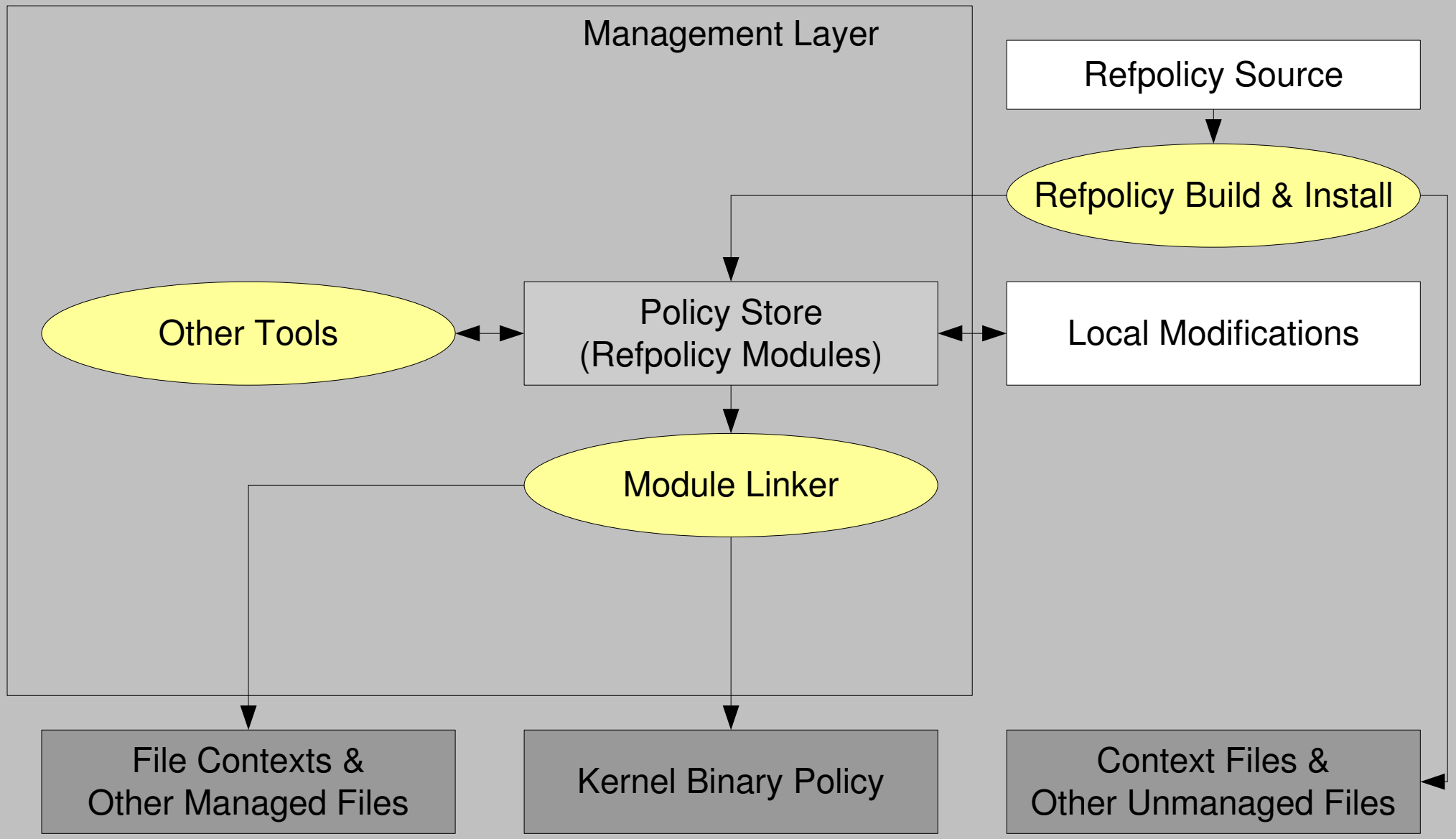
Outline

- Goals
- Current Architecture
- New Policy Build and Management Architecture
- Common Intermediate Language (CIL)
- Plan

Goals

- Short-term
  - Enhanced higher-level language support
  - More scalable and usable policy infrastructure
  - Better support for customizing the policy
- Long-term
  - Higher-level policy languages
  - Less complex policies

Current Architecture



NATIONAL INFORMATION ASSURANCE RESEARCH LABORATORY

Policy Build and Management Architecture

Problems with the Management Layer

- Removing permissions requires the modification of policy modules
- Custom policy distribution is hard
- Not clear where to store local module sources
- Binary module format is a hindrance to extensibility
- Mixture of managed and unmanaged policy in the policy store
- Build process is brittle
- Multiple policy file formats exist
- No common abstraction that can contain any policy statement

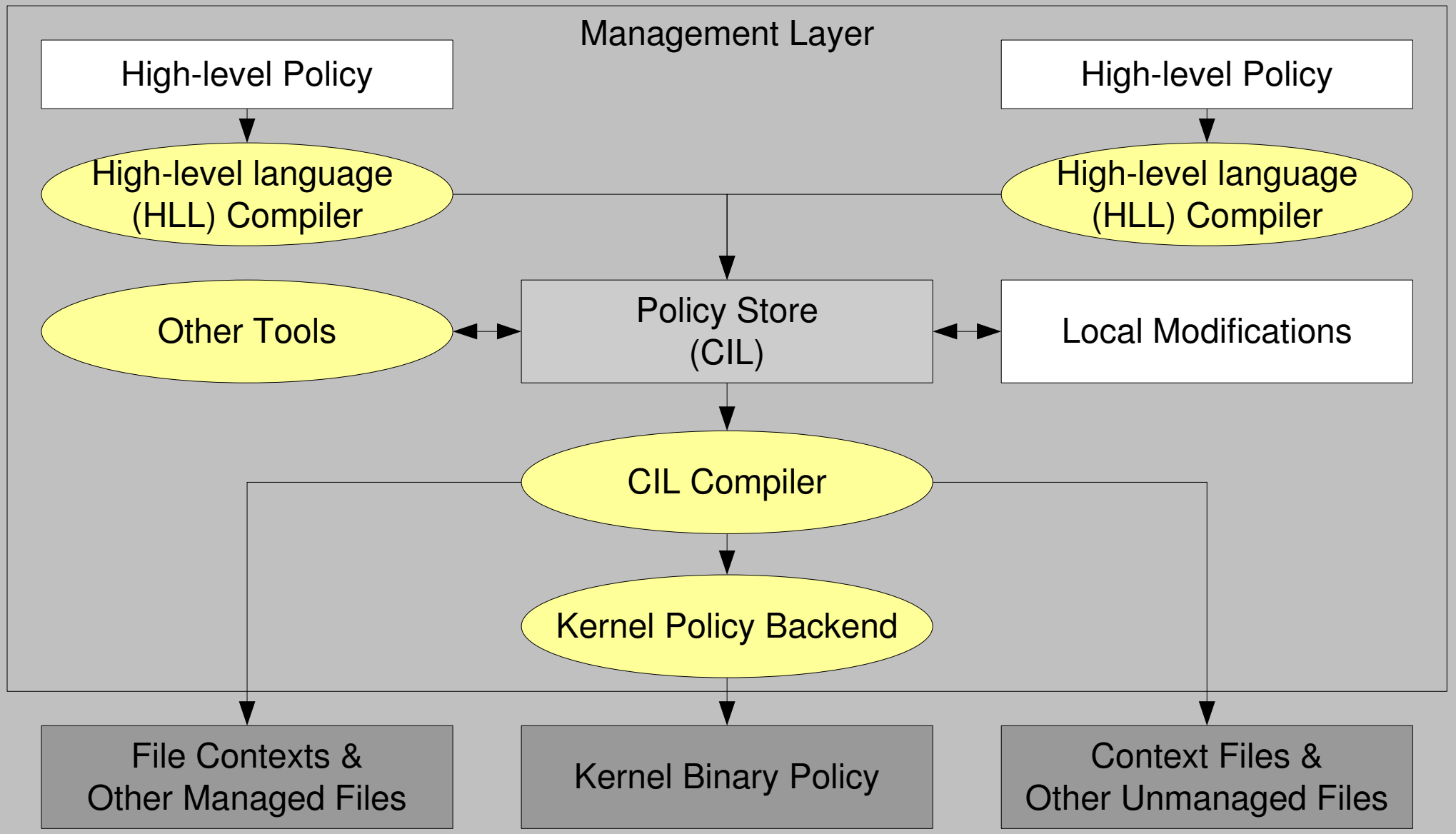
Requirements of the Management Layer

- Must have the ability to add and remove policy rules
- Must be able to import/export customizations
- Must keep the distribution policy and local customizations separate
- Should store local customizations in the policy store
- Should use a source policy format

Requirements of the Management Layer (cont)

- Should not mix managed and unmanaged policy files
- Should keep the policy build as simple, flexible, and resource friendly as possible
- May eliminate some policy file formats

Proposed Architecture



Changes to the Management Layer

- Will be able to retrieve the source for a module

```
semanage module --get <module>
```

- Every command will create a log entry
  - New global option `--message`
  - Format:

```
COM="<command+options>" /
ID=<login id> CONTEXT=<context> /
TIME=<time stamp> MSG="<message>"
```

- Priority Levels
  - Management -> Local -> Distribution (Higher to Lower)

Changes to the Management Layer

- Source Control Management
  - basic
    - Does not support reverting back more than one version.
  - git

Policy Build and Load Sequence

- Build CIL tree by traversing the policy store from highest to lowest priority
- Execute CIL transforms on the tree
- Remove disabled modules from the tree
- Convert CIL tree to policydb and managed files
- Serialize the policy to disk in a temporary location

Policy Build and Load Sequence (cont)

- Sanity check the policy files
- Copy the policy files to the destination in the policy store
- Load policy

Common Intermediate Language (CIL)

Problems with the Modular Policy Language

- Lack of abstraction
  - Not a good target for high-level languages
  - Modules are inflexible
  - Not easy to create new types based on an old type
- Gaps in features
- Confusing semantics
- Inconsistencies in the syntax
- Inadequate debugging support
- Ordering dependencies for portcon, category, sid, and class

Requirements of the Language

- Support the use of high-level languages
- Provide a comprehensive and unambiguous representation of the policy
- Support programmatic introspection and manipulation of the policy
- Provide detailed debugging information
- Be order independent
- Support policy modification without changing the original sources
- Support policy access control

New Features

- Transformation Language
- Selection
- Generic blocks
- Generic ifs
- Defined sets

Transformation Language

- Provides the ability to do policy manipulation at a semantic level.
- Three basic operations
  - `add <target>`
  - `del <target>`
  - `copy <source> => <target>`
    `except <items from source>`
- Example

```
add BLOCK foo {
    TYPE bar
  }
```

Selection

- Syntax

  / Absolute path

  ./ Relative path

  .. Parent

  * Match zero or more of anything

  0-9 Ranged matching on number

  <path>/ Child of <path> - Only valid in a target

- Example
  - To select all allow rules with a target of bar:

    /*/(ALLOW:TYPE * (TYPE bar) *)

Other Features

- Generic Block
  - Replaces interfaces and templates
  - Used for selection
- Generic if
  - Replaces tunables, booleans, optionals, and ifdefs
- Defined Sets
  - Ability to define sets now built in

Plan

- Send the new architecture and CIL to the SELinux mailing list for discussion
- Modify libraries to support the new architecture
- Modify SELinux tools to support the new architecture
- Develop the CIL compiler
- Modify Reference policy build to target CIL