Guided Policy Generation for Application Authors

David R. Harris
John D. Ramsdell
Brian T. Sniffen
SELinux Policy Management

Goals of MITRE SELinux effort

- Provide new capabilities to existing SELinux users
- Lower barrier-to-entry for new types of SELinux installations

Sample scenarios:

- Enterprise wants to use SELinux on internet-facing systems
  - Catch: Enterprise-specific applications must be supported
- Government organization wants to use SELinux as trusted base for particular application, system
  - Require provable security, need policy as quickly as possible
One Solution

- Build flexible, extensible policy management tools
  - Capture nuances of underlying system with appropriate model
  - Determine relevant security goals, policy design goals
  - Solve core problems while allowing for future expansion

- Our Method: Information flow policy management tools
  - Focus on the creation and restriction of information flow through the system
  - Support desirable flows, highlight undesirable flows

- Currently Existing:
  - Policy analysis (with SLAT)
  - Policy generation (with polgen)
Sample Security Goals

- “I need to support this FTP server, but I want to make sure nobody can use it to write data to my disk.”
- “My environment requires segregation of various types of information.”
- “I want to check that all mail flows directly through the virus-checking program.”

Information Flow Problems
Writing programs with Polgen
Approach: Guided Automation

- Focus on least privilege
  - Give the program only the privileges it actually needs *in use*

- Discover new types

- Basic idea: Program behavior pattern-based generation
  - What is the program trying to do?
  - What component interactions can we recognize?
  - What reasons might it do that?
  - What constitutes sensible policy for those interactions?
Polgen Walkthrough: Goal

- Allow Jabber to run in a reasonable way on SELinux server
  - Open-source instant messaging server
  - Implemented by several processes, communicating by sockets
  - Clients run in user-space, server requires protection

- Have some idea about goals ahead of time
  - Protect Jabber configuration files

- Viewpoint of Jabber’s authors

- Old support: viewpoint of admins
Polgen Walkthrough: Overview

- Define components with Polgen Specification Language (PSL)
- Observe information flow between components
  - strace now, autrace soon, other methods later
- Detect patterns in information flow graph
- Suggest policy additions to the user
  - User decides which additions/changes make sense
- Produce policy based on user input
  - Standard M4 macros (hierarchical support coming)
Polgen Preparation

- Initial type generation accomplished with PSL
  - Seed configuration file, install new policy
  - Tags dynamic traces of program for later observation
  - Later integration

- Jabberd PSL file excerpt:

```c
component router {
  type process
  parent jabberd
  reads {usr/lib, router.xml}
  writes {$prefix/router.log,$prefix/router.pid}
  listens_at {resolver, c2s}
}
```
Guided Observation

Exercise program with SELinux-aware `strace`
- Functions only in permissive mode
- Output from `strace` expansive, must be filtered
  - Jabber output ~2MB of text

Filter `strace` output with pre-processing script
- Tracks read/write system calls to file descriptors created using `open()`, `socket()`, `pipe()`
- Tracks file descriptors through different processes, threads
- Each output line contains PID, system resource, resource/directory/domain contexts, system call, etc.

Filtered Jabber output 48KB of text. Sample line:

```
335 Open "'/usr/local/lib/libldap-2.2.so.7"
   root:object_r:lib_t FILE R root:object_r:lib_t
3 119 user_u:system_r:polgen_temp_jabberd_t
```
Automated Recognition

- Recognize patterns in PSL, dynamic traces

- Recognized a round two dozen patterns
  - Temporary/configuration file use, shared library use, pipe use
  - Pipeline, mediator, proxy, hub-and-spoke

- Some excerpts from Jabber traces:
  - Configuration file instances
    - General files in /etc/, /usr/local/etc/jabberd/ files, etc.
    - Also finds instance of “hub and spoke” pattern
      - A collection of processes (“spokes”) communicate with one another through a central “hub”
      - In this case, router process acts as the hub
Pattern Discovery

- Polgen can also help users understand programs
  - Patterns provide view into how the program operates
  - These can suggest new information flow goals for programs

- Example: Hub-and-spoke pattern in Jabber server
  - This communication pattern also found in the Jabber documentation
  - New information flow goal: Spoke processes communicate with one another only through hub
User Interaction: Seeking Guidance

- User presented with GUI
  - Confirm real patterns
  - Reject false positives or bad goals
  - Unify types or groups of types with PSL components
  - Confirm or reject all file type re-labeling

- Contexts suggested depend upon patterns
  - If no pattern detected, existing context shown
  - For selected patterns, new type suggestions listed
Web Pages: Config pattern Instance

pa241:polgen_config(
jabberd2_polgen_temp_jabberd_t,
jabberd2_polgen_temp_jabberd/etc_dir_t)

( 7 Instances Found )
Return to top

----1.) application:jabberd_5318
config_file:/usr/local/etc/jabberd/c2s.xml

----2.) application:jabberd_5316
config_file:/usr/local/etc/jabberd/sm.xml

----3.) application:jabberd_5317
config_file:/usr/local/etc/jabberd/s2s.xml
Web Pages: Resources Accessed

jabberd_5318 a Process

with security context = user_u:system_r:jabberd2_polgen_temp_jabberdt

Playing client role for a Client
Playing client role for a ClientServer
Playing application role for a SharedLibrary
Playing application role for a Config
Playing proxy role for a Proxy
Playing client role for a Proxy
Playing client role for a Proxy
Playing client role for a Proxy
Reading from:
   ---/usr/lib/libldap-2.2.so.7
   ---/usr/lib/liblber-2.2.so.7
   ---/usr/local/mysql/lib/mysql/libmysqlclient.so.12
   ---/usr/lib/libz.so.1
   ---/lib/libc8.so.1
   ---/lib/libnsl.so.1
   ---/lib/tls/libm.so.6
   ---/lib/libssl.so.4
   ---/lib/libcrypto.so.4
   ---/usr/local/lib/libidn.so.11
Summary

- Important security goals are the information flow goals

- Information flow builds powerful tools

- Authors can easily amplify design knowledge with Polgen

- Get software, share suggestions, for more information:
  - http://www.mitre.org/tech/selinux/
    - Source tarball
    - FC4, Rawhide RPMs
  - http://sourceforge.net/projects/polgen
    - Continued development
    - Polgen-devel mailing list

- Questions?
Future Steps

- Planned tool improvements:
  - Syscall Audit
    - Better multi-program, multi-threaded support
  - Hierarchy / Reference Policy
    - Free structural information

- Welcomed tool improvements:
  - Expand collection of patterns recognized (“local” and more abstract patterns)
  - Improve policy suggestions for all patterns
  - Enhance GUI usability
  - Other analysis methods

- Future tools planned: Compression, multi-policy analysis, composite analysis, ??
Further Information
Backups
Mailman “qrunner” PSL

component qrunner {
    type process
    parent bin
    requires {Logging, Queue}
    execs {ArchRunner, BounceRunner, CommandRunner, 
          IncomingRunner, NewsRunner, OutgoingRunner, 
          VirginRunner}
}


component Mailman {
    type application
    pipeline {'SpamDetect', 'Approve', 'Replybot', 'Moderate',
              'Hold', 'MimeDel', 'Emergency', 'Tagger', 'CalcRecips',
              'AvoidDuplicates', 'Cleanse', 'CookHeaders',
              'ToDigest', 'ToArchive', 'ToUsenet',
              'AfterDelivery', 'Acknowledge', 'ToOutgoing'}
    requires {python_module, apache, Exim, Postfix } 
}