Madison – A New Approach to Policy Generation

Karl MacMillan
kmacmill@redhat.com
Principal Software Engineer

2007 SELinux Symposium

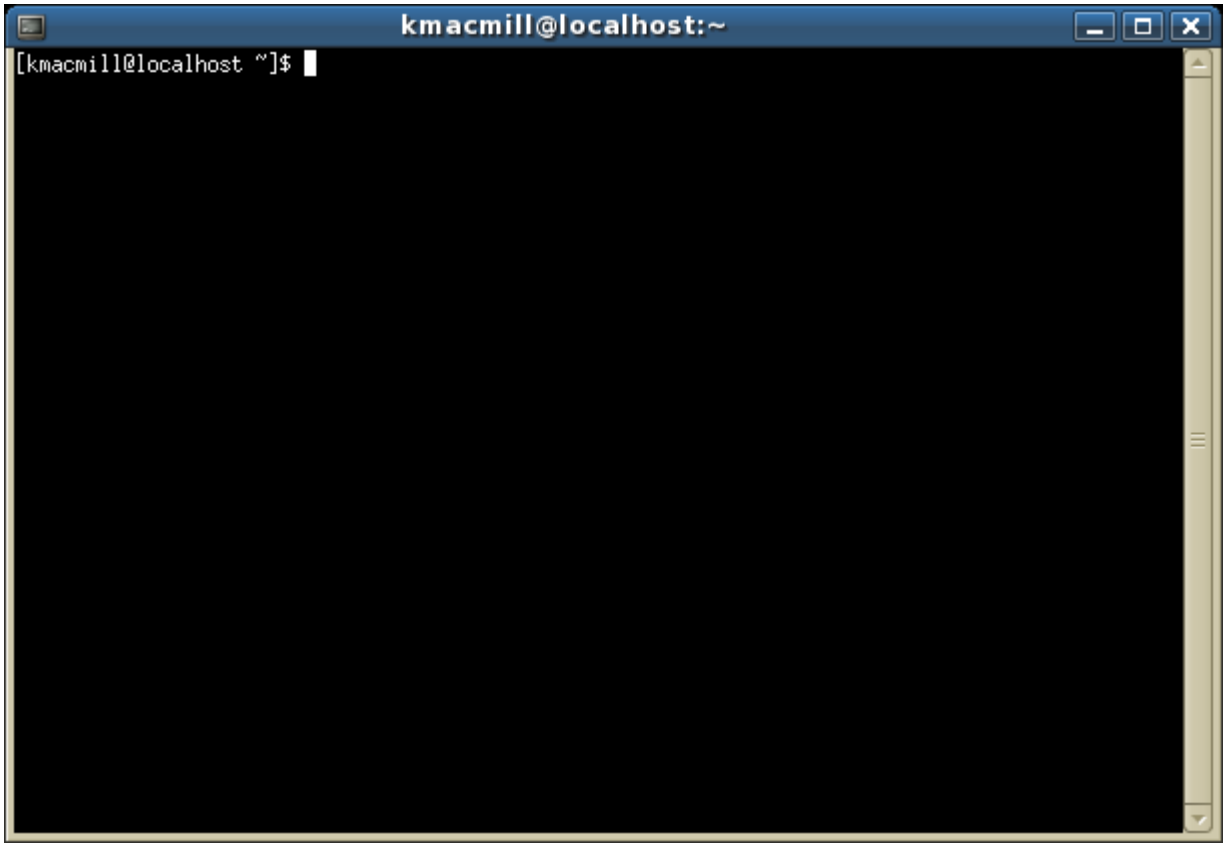
Sepolgen
~~Madison~~ – A New Approach to Policy Generation

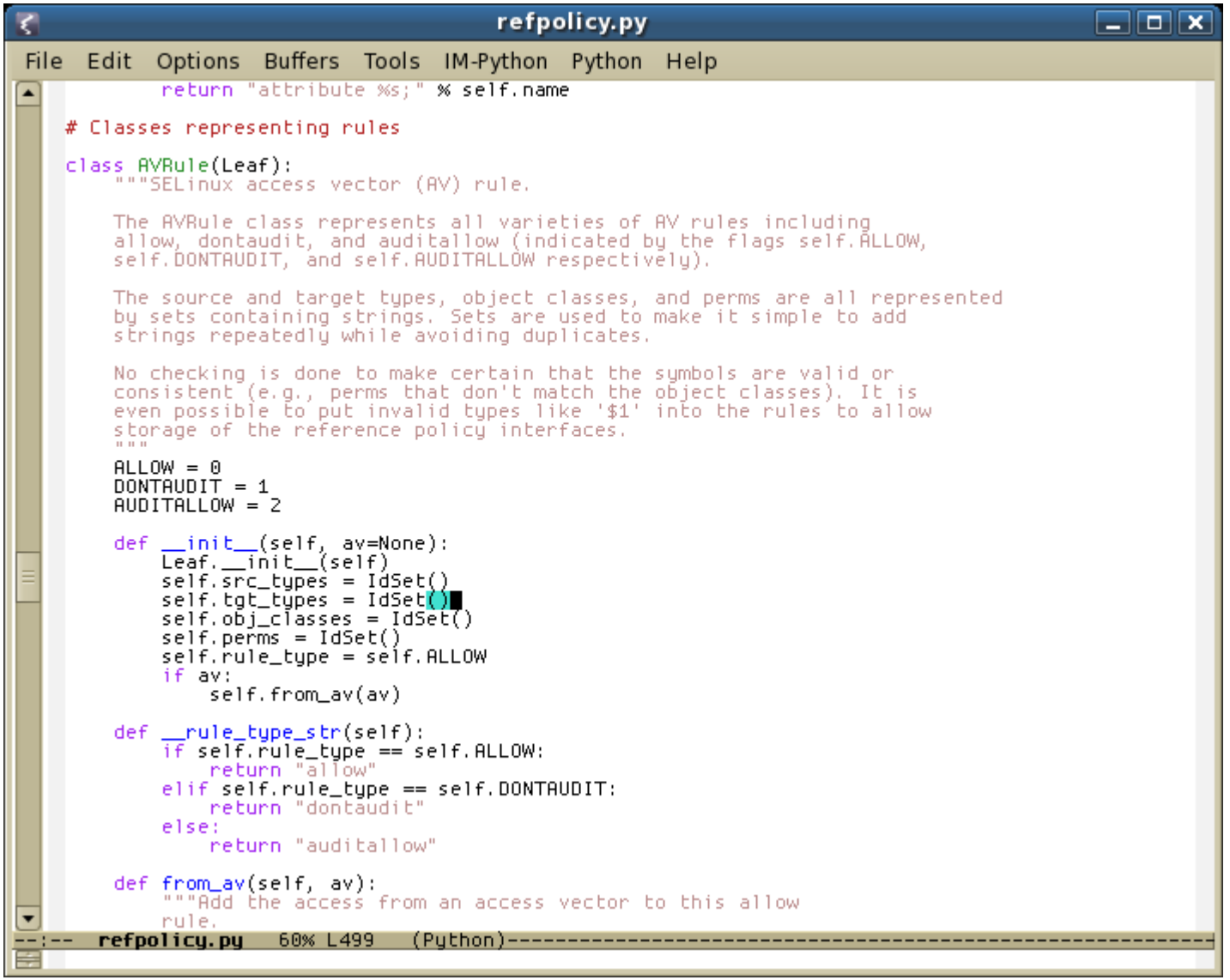
Karl MacMillan
kmacmill@redhat.com
Principal Software Engineer

2007 SELinux Symposium

Introduction

- **Sepolgen mandate: improve SELinux usability**
  - but what is usability?
- **Usability is a single metric for a multi-faceted problem**
  - often shortened as “make it simple to use”
  - if simplicity were the only goal, software usability would be solved
  - sad results: “clippy” the office assistant
- **Inherit tension between expressiveness and simplicity**
  - or perhaps expressiveness and initial learning
- **Simple applications fail when faced with new situation**
  - often caused by “leaky abstractions


kmacmill@localhost:~
[kmacmill@localhost ~]$


refpolicy.py
File Edit Options Buffers Tools IM-Python Python Help
return "attribute %s;" % self.name
# Classes representing rules
class AVRule(Leaf):
"""SELinux access vector (AV) rule.
The AVRule class represents all varieties of AV rules including
allow, dontaudit, and auditallow (indicated by the flags self.ALLOW,
self.DONTAUDIT, and self.AUDITALLOW respectively).
The source and target types, object classes, and perms are all represented
by sets containing strings. Sets are used to make it simple to add
strings repeatedly while avoiding duplicates.
No checking is done to make certain that the symbols are valid or
consistent (e.g., perms that don't match the object classes). It is
even possible to put invalid types like '$1' into the rules to allow
storage of the reference policy interfaces.
"""
ALLOW = 0
DONTAUDIT = 1
AUDITALLOW = 2
def __init__(self, av=None):
Leaf.__init__(self)
self.src_types = IdSet()
self.tgt_types = IdSet()
self.obj_classes = IdSet()
self.perms = IdSet()
self.rule_type = self.ALLOW
if av:
self.from_av(av)
def __rule_type_str(self):
if self.rule_type == self.ALLOW:
return "allow"
elif self.rule_type == self.DONTAUDIT:
return "dontaudit"
else:
return "auditallow"
def from_av(self, av):
"""Add the access from an access vector to this allow
rule.
--:-- refpolicy.py 60% L499 (Python)

redhat

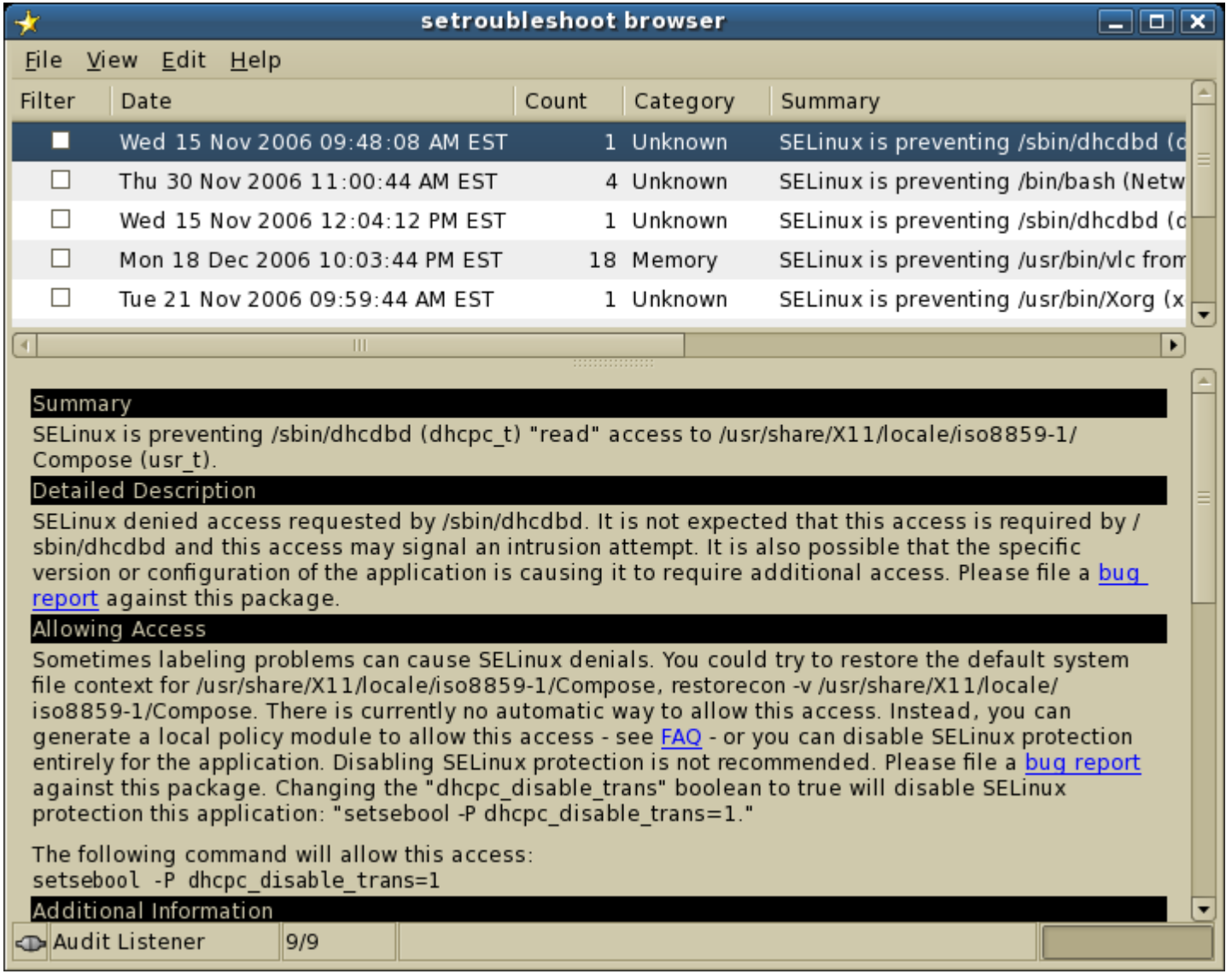
SELinux Usability

- **Linux success based on expressiveness and power**
  - yields flexibility and simplicity
  - however, the trade-off is not always acceptable
- **Microsoft is copying**
  - re-introducing the command line for administration
- **Alternate goal: enable users to accomplish goals**
  - what users?
  - what goals?
- **One possible answer:**
  - users: admins disabling SELinux
  - goal: make their systems function correctly

Why Do Admins Disable SELinux?

type=AVC msg=audit(1173876205.535:130): avc: denied { read } for pid=8266
comm="firefox" name="ld.so.cache" dev=dm-0 ino=18874522
scontext=user_u:system_r:firefox_t:s0 tcontext=user_u:object_r:ld_so_cache_t:s0 tclass=file
type=SYSCALL msg=audit(1173876205.535:130): arch=40000003 syscall=5 success=yes exit=3
a0=4a569217 a1=0 a2=0 a3=ffffffff items=0 ppid=7967 pid=8266 auid=500 uid=500 gid=500
euid=500 suid=500 fsuid=500 egid=500 sgid=500 fsgid=500 tty=pts2 comm="firefox"
exe="/bin/bash" subj=user_u:system_r:firefox_t:s0 key=(null)
type=AVC msg=audit(1173876205.535:131): avc: denied { getattr } for pid=8266
comm="firefox" name="ld.so.cache" dev=dm-0 ino=18874522
scontext=user_u:system_r:firefox_t:s0 tcontext=user_u:object_r:ld_so_cache_t:s0 tclass=file
type=SYSCALL msg=audit(1173876205.535:131): arch=40000003 syscall=197 success=yes
exit=0 a0=3 a1=bfe03330 a2=4a56dfc0 a3=ffffffff items=0 ppid=7967 pid=8266 auid=500
uid=500 gid=500 euid=500 suid=500 fsuid=500 egid=500 sgid=500 fsgid=500 tty=pts2
comm="firefox" exe="/bin/bash" subj=user_u:system_r:firefox_t:s0 key=(null)
type=AVC_PATH msg=audit(1173876205.535:131): path="/etc/ld.so.cache"
type=AVC msg=audit(1173876205.535:132): avc: denied { search } for pid=8266
comm="firefox" name="lib" dev=dm-0 ino=5308417 scontext=user_u:system_r:firefox_t:s0
tcontext=system_u:object_r:lib_t:s0 tclass=dir
type=AVC msg=audit(1173876205.535:132): avc: denied { read } for pid=8266
comm="firefox" name="libtinfo.so.5" dev=dm-0 ino=5308676
scontext=user_u:system_r:firefox_t:s0 tcontext=system_u:object_r:lib_t:s0 tclass=lnk_file
type=AVC msg=audit(1173876205.535:132): avc: denied { read } for pid=8266
comm="firefox" name="libtinfo.so.5.6" dev=dm-0 ino=5309432
scontext=user_u:system_r:firefox_t:s0 tcontext=system_u:object_r:lib_t:s0 tclass=file
type=SYSCALL msg=audit(1173876205.535:132): arch=40000003 syscall=5 success=yes exit=3
a0=b7fdf4c7 a1=0 a2=47 a3=b7fdf4c7 items=0 ppid=7967 pid=8266 auid=500 uid=500
gid=500 euid=500 suid=500 fsuid=500 egid=500 sgid=500 fsgid=500 tty=pts2 comm="firefox"
exe="/bin/bash" subj=user_u:system_r:firefox_t:s0 key=(null)

Partial Solution: Setroubleshoot



Enabling Administrators

- **What happens when setroubleshoot has no answer?**
  - “It is not expected that this access is required . . . “
- **This is one starting point for sepolgen**
- **Improved version of audit messages**

```
# src="firefox_t" tgt="lib_t" class="dir", perms="{ read search getattr }"
# comm="firefox" exe="" path=""
allow firefox_t lib_t:dir { read search getattr };
# src="firefox_t" tgt="lib_t" class="file", perms="{ read getattr execute }"
# comm="firefox" exe="" path=""
allow firefox_t lib_t:file { read getattr execute };
# src="firefox_t" tgt="lib_t" class="lnk_file", perms="read"
# comm="firefox" exe="" path=""
allow firefox_t lib_t:lnk_file read;
```

Understanding Audit Messages

- **What is difficult about translated audit messages?**
  - types? object classes and permissions? policy language syntax?
- **Type enforcement concepts and syntax are *not* the problem**
  - at least not the most pressing problem
- **Evidence:**

```
-A RH-Firewall-1-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
```

redhat

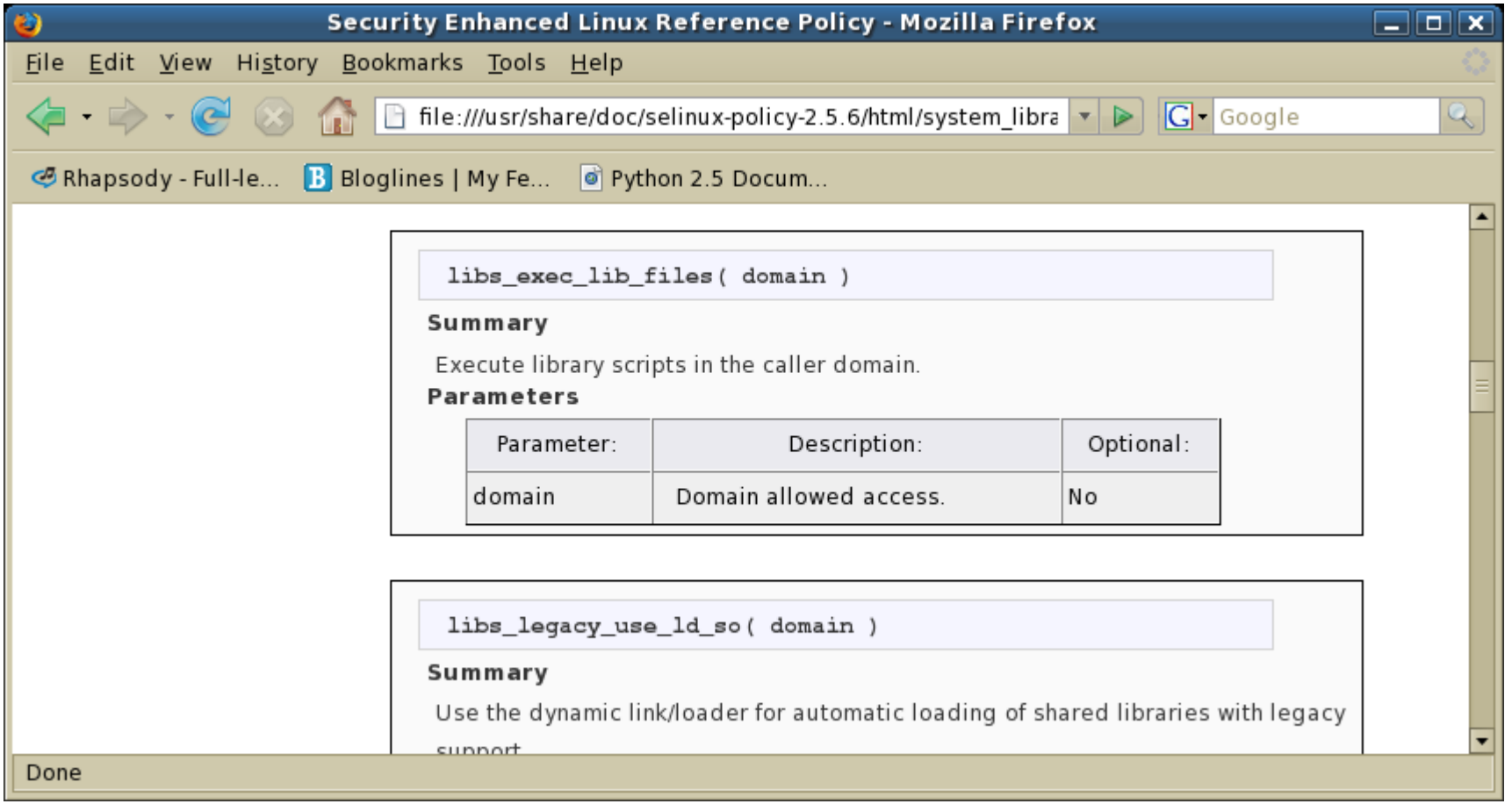
Iptables is Simple?

- **Iptables is accepted and used by administrators**
  - at least professional administrators
- **What is the difference between Iptables and SELinux?**
  - SELinux has simpler policy syntax?
  - Iptables problems are more difficult to diagnose?
- **Administrators understand network security**
  - have some idea of what network traffic should be allowed
  - understand the access requirements of applications
    - “did you open port 22 for ssh?”
- **SELinux access for applications is more challenging . . .**

redhat

Deciphering Application Access

- **Example: allow firefox_t lib_t:file { read getattr execute };**
- **Types and object classes are somewhat challenging**
  - but not too bad: firefox_t == firefox
  - some object classes and types are worse than others
- **Real problem: what does the access *mean*?**
  - and is it dangerous?
- **Reference policy helps:**
  - libs_exec_lib_files(firefox_t)
- **There is even documentation**


Security Enhanced Linux Reference Policy - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///usr/share/doc/selinux-policy-2.5.6/html/system_libra
Google
Rhapsody - Full-le...
Bloglines | My Fe...
Python 2.5 Docum...
libs_exec_lib_files(domain)
Summary
Execute library scripts in the caller domain.
Parameters
Parameter:
Description:
Optional:
domain
Domain allowed access.
No
libs_legacy_use_ld_so(domain)
Summary
Use the dynamic link/loader for automatic loading of shared libraries with legacy
Done

redhat

Discovering Reference Policy Interfaces

- **Refpolicy interfaces are great**
  - there are so many to choose from!
- **How do we discover interfaces?**
  - setroubleshoot approach doesn't scale
- **Sepolgen approach: automated matching**
  - summary: somewhat hard but doable
  - requires some infrastructure
- **First step: parsing reference policy**
  - M4 is evil
  - the Chris factor makes it possible

Sepolgen Approach to Interface Matching

- **Basic steps:**
  - Parse reference policy into syntax tree
  - Extract access from each interface
  - Match requested access interfaces
- **Requires simplifying assumptions**
  - interfaces allow related sets of access
  - related interfaces form set relations
    - files_read_etc_files is a subset of files_rw_etc_files
- **Information flow helps with relevance**
  - don't allow write if read is requested

Example Output

corecmd_search_bin(firefox_t)
corenet_udp_sendrecv_dns_port(firefox_t)
dev_read_urand(firefox_t)
files_list_etc(firefox_t)
files_list_usr(firefox_t)
files_read_etc_files(firefox_t)
files_read_etc_runtime_files(firefox_t)
files_read_etc_symlinks(firefox_t)
files_read_generic_tmp_files(firefox_t)
files_read_usr_files(firefox_t)
files_read_usr_symlinks(firefox_t)
files_read_var_files(firefox_t)
files_search_home(firefox_t)
fs_rw_tmpfs_files(firefox_t)
xserver(firefox_t)
fs_search_inotifyfs(firefox_t)
kernel_read_all_sysctls(firefox_t)
kernel_read_system_state(firefox_t)
kernel_search_network_sysctl(firefox_t)
libs_exec_ld_so(firefox_t)
libs_exec_lib_files(firefox_t)
libs_read_lib_files(firefox_t)
nscd_read_pid(firefox_t)
sysnet_read_config(firefox_t)
term_search_ptys(firefox_t)
unconfined_stream_connect(firefox_t)
userdom_manage_generic_user_home_co
ntent_dirs(firefox_t)
userdom_search_generic_user_home_dirs
(firefox_t)
xserver_read_xdm_tmp_files(firefox_t)
xserver_stream_connect_xdm(firefox_t)
xserver_stream_connect_xdm_

Future Work

- **Local policy modification tool**
  - address most common workflow for admins
- **More complete policy analysis (e.g., access through attributes)**
  - libsepol work will help
- **Round-trip policy modifications**
  - needed for new module development
  - updating existing modules
- **Policy searching (e.g., which interfaces reference type lib_t)**
- **Graphical tools**
- **Integration with other applications (SLIDE, setroubleshoot)**

Questions?

Karl MacMillan
kmacmill@redhat.com