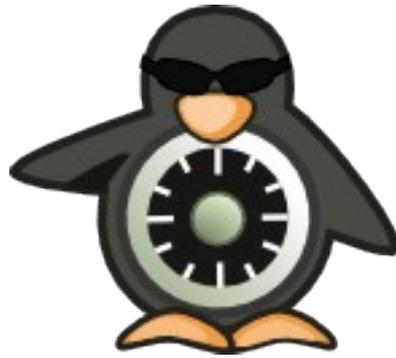


# The SELinux Notebook



## Volume 2 Sample Policy Source (2<sup>nd</sup> Edition)

## 0. Notebook Information

### 0.1 Copyright Information

Copyright © 2010 [Richard Haines](#).

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled "[GNUFree Documentation License](#)".

The scripts and source code in this Notebook are covered by the GNU General Public License. The scripts and code are free source: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or any later version.

These are distributed in the hope that they will be useful in researching SELinux, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with scripts and source code. If not, see <http://www.gnu.org/licenses/>.

### 0.2 Revision History

| Edition | Date                     | Changes   |
|---------|--------------------------|---|
| 1.0     | 20 <sup>th</sup> Nov '09 | First released.   |
| 2.0     | 9 <sup>th</sup> May '10  | Split the Notebook into two volumes: <ol style="list-style-type: none"><li>1. The Foundations - covers SELinux and its supporting services.</li><li>2. Sample Policy Source - contains sample application and policy source code to build a simple message filter and experiment with X-Windows.</li></ol> In this volume: <ul style="list-style-type: none"><li>• Updated all relevant sections to reflect Fedora 12 release and correct errors.</li><li>• Modified sample policies to work with F-12 (signal handling changed).</li><li>• Added sections on: Experimenting with X-Windows polyinstantiation copy &amp; paste code and policy; A test module for XSELinux functions.</li></ul> |

## 0.3 Acknowledgements

Logo designed by [Máirín Duffy](#)

## 0.4 Abbreviations

| Term           | Definition                           |
|----------------|--------------------------------------|
| <b>apol</b>    | Policy analysis tool                 |
| <b>AV</b>      | Access Vector                        |
| <b>AVC</b>     | Access Vector Cache                  |
| <b>F-12</b>    | Fedora 12                            |
| <b>MAC</b>     | Mandatory Access Control             |
| <b>OM</b>      | Object Manager                       |
| <b>RBAC</b>    | Role-based Access Control            |
| <b>SELinux</b> | Security-Enhanced Linux              |
| <b>SID</b>     | Security Identifier                  |
| <b>SL</b>      | Security Level                       |
| <b>TE</b>      | Type Enforcement                     |
| <b>UID</b>     | User Identifier                      |
| <b>XACE</b>    | X (windows) Access Control Extension |

## 0.5 Index

|  |                 |
|--|-----------------|
| <b><u>0. NOTEBOOK INFORMATION.....</u></b>                 | <b><u>2</u></b> |
| <u>0.1 COPYRIGHT INFORMATION.....</u>                      | <u>2</u>        |
| <u>0.2 REVISION HISTORY.....</u>                           | <u>2</u>        |
| <u>0.3 ACKNOWLEDGEMENTS.....</u>                           | <u>3</u>        |
| <u>0.4 ABBREVIATIONS.....</u>                              | <u>3</u>        |
| <u>0.5 INDEX.....</u>                                      | <u>3</u>        |
| <b><u>1. THE SELINUX NOTEBOOK.....</u></b>                 | <b><u>6</u></b> |
| <u>1.1 INTRODUCTION.....</u>                               | <u>6</u>        |
| <u>1.2 VOLUME 1 - THE FOUNDATIONS OVERVIEW.....</u>        | <u>6</u>        |
| <u>1.3 VOLUME 2 - SAMPLE POLICY SOURCE OVERVIEW.....</u>   | <u>7</u>        |
| <u>1.3.1 Sample Policy Source Sections.....</u>            | <u>7</u>        |
| <u>1.4 RELEVANT F-12 PACKAGES.....</u>                     | <u>8</u>        |
| <b><u>2. BUILDING A BASIC POLICY.....</u></b>              | <b><u>9</u></b> |
| <u>2.1 INTRODUCTION.....</u>                               | <u>9</u>        |
| <u>2.1.1 Overall Objectives.....</u>                       | <u>9</u>        |
| <u>2.1.2 Build Requirements.....</u>                       | <u>9</u>        |
| <u>2.1.3 The Test Policies.....</u>                        | <u>10</u>       |
| <u>2.2 BUILDING THE POLICY SOURCE FILES.....</u>           | <u>10</u>       |
| <u>2.2.1 Policy Source Files.....</u>                      | <u>13</u>       |
| <u>2.2.1.1 Problem Resolution.....</u>                     | <u>14</u>       |
| <u>2.2.1.2 Monolithic and Base Policy Source File.....</u> | <u>14</u>       |

|  |            |
|--|------------|
| 2.2.1.3 file_contexts File.....  | 20         |
| 2.2.1.4 default_contexts File.....                                     | 20         |
| 2.2.1.5 seusers File.....  | 20         |
| 2.2.1.6 dbus_contexts File.....  | 20         |
| 2.2.1.7 x_contexts File.....   | 21         |
| 2.3 BUILDING THE MONOLITHIC POLICY.....                                | 21         |
| 2.3.1 <i>Checking the Build</i> .....                                  | 23         |
| 2.4 BUILDING THE BASE POLICY MODULE.....                               | 24         |
| 2.4.1 <i>Checking the Base Policy Build</i> .....                      | 26         |
| <b>3. BUILDING THE MESSAGE FILTER LOADABLE MODULES.....</b>            | <b>27</b>  |
| 3.1 OVERVIEW OF MODULES.....   | 27         |
| 3.2 BUILDING THE SECMARK TEST LOADABLE MODULE.....                     | 28         |
| 3.2.1 <i>Testing the Module</i> .....                                  | 42         |
| 3.2.1.1 <i>Running the Tests</i> .....                                 | 43         |
| 3.2.2 <i>Points to Note</i> .....                                      | 46         |
| 3.2.2.1 <i>Importance of Loading the iptables</i> .....                | 46         |
| 3.2.2.2 <i>Running tests out of sequence</i> .....                     | 46         |
| 3.3 BUILDING THE NETLABEL LOADABLE MODULE.....                         | 47         |
| 3.4 BUILDING THE REMAINING MESSAGE FILTER SERVICE.....                 | 49         |
| 3.4.1 <i>Internal Gateway Loadable Policy Module</i> .....             | 49         |
| 3.4.2 <i>File Move Application</i> .....                               | 52         |
| 3.4.3 <i>File Mover Loadable Policy Module</i> .....                   | 55         |
| 3.4.4 <i>Testing the Message Filter Build</i> .....                    | 58         |
| <b>4. EXPERIMENTING WITH X-WINDOWS.....</b>                            | <b>61</b>  |
| 4.1 SECTION OVERVIEW.....  | 61         |
| 4.2 OVERVIEW OF MODULES AND APPLICATIONS.....                          | 61         |
| 4.2.1 <i>The x_contexts Files and Supporting Loadable Module</i> ..... | 61         |
| 4.2.2 <i>The Select - Paste Applications and Loadable Module</i> ..... | 63         |
| 4.2.2.1 <i>Test Conclusions</i> .....                                  | 65         |
| 4.2.2.2 <i>Calling the XSELinux Functions</i> .....                    | 67         |
| 4.3 BUILDING THE X-WINDOWS SELECT AND PASTE EXAMPLES.....              | 67         |
| 4.3.1 <i>Building the x_contexts Files and Loadable Module</i> .....   | 68         |
| 4.3.2 <i>Building the X-select and X-paste Applications</i> .....      | 79         |
| 4.3.3 <i>Building the X-select and X-paste Loadable Module</i> .....   | 107        |
| 4.3.4 <i>Testing Derived Labels</i> .....                              | 113        |
| 4.3.4.1 <i>Derived Object Test Conclusions</i> .....                   | 116        |
| 4.3.5 <i>Testing Polyinstantiated Labels</i> .....                     | 117        |
| 4.3.5.1 <i>Polyinstantiated Object Test Conclusions</i> .....          | 122        |
| 4.4 BUILDING THE XSELINUX FUNCTION TEST APPLICATION.....               | 122        |
| <b>5. APPENDIX A - POLICY INVESTIGATION TOOLS.....</b>                 | <b>131</b> |
| 5.1 INTRODUCTION.....  | 131        |
| 5.2 USING AUDIT2ALLOW AND AUDIT2WHY.....                               | 131        |
| 5.3 USING SEAUDIT AND SETROUBLESHOOT.....                              | 132        |
| 5.4 USING SEDIFFX.....   | 132        |
| 5.5 USING SECHECKER.....   | 134        |
| 5.5.1 <i>Testing the Policy</i> .....                                  | 134        |
| 5.5.2 <i>The Results</i> .....   | 136        |

|  |                            |
|--|----------------------------|
| <a href="#">5.6 USING APOL .....</a>   | <a href="#">145</a>        |
| <a href="#">5.6.1 General Information.....</a>   | <a href="#">145</a>        |
| <a href="#">5.6.2 Type Enforcement Rules.....</a>  | <a href="#">147</a>        |
| <a href="#">5.6.3 Direct Relabel.....</a>  | <a href="#">149</a>        |
| <a href="#">5.6.3.1 apol Direct Relabel Analysis.....</a>                                      | <a href="#">149</a>        |
| <a href="#">5.6.4 Transitive Information Flows.....</a>  | <a href="#">150</a>        |
| <a href="#">5.6.4.1 apol Transitive Information Flows Analysis.....</a>                        | <a href="#">150</a>        |
| <b><a href="#">6. APPENDIX B – NETLABEL MODULE SUPPORT FOR NETWORK_PEER_CONTROLS .....</a></b> | <b><a href="#">153</a></b> |
| <a href="#">6.1 INTRODUCTION.....</a>  | <a href="#">153</a>        |
| <a href="#">6.2 CONFIGURATION.....</a>   | <a href="#">153</a>        |
| <b><a href="#">7. APPENDIX C – LABELED IPSEC MODULE EXAMPLE.....</a></b>                       | <b><a href="#">156</a></b> |
| <a href="#">7.1 INTRODUCTION.....</a>  | <a href="#">156</a>        |
| <a href="#">7.2 MANUAL IPSEC CONFIGURATION.....</a>  | <a href="#">156</a>        |
| <a href="#">7.3 KEY EXCHANGE IPSEC CONFIGURATION.....</a>                                      | <a href="#">160</a>        |
| <b><a href="#">8. APPENDIX D – IMPLEMENTING A CONSTRAINT .....</a></b>                         | <b><a href="#">163</a></b> |
| <a href="#">8.1 INTRODUCTION.....</a>  | <a href="#">163</a>        |
| <a href="#">8.2 CONFIGURATION.....</a>   | <a href="#">163</a>        |
| <a href="#">8.3 REFERENCE POLICY CONSTRAINTS INFORMATION.....</a>                              | <a href="#">164</a>        |
| <b><a href="#">9. APPENDIX E - GNU FREE DOCUMENTATION LICENSE.....</a></b>                     | <b><a href="#">166</a></b> |

## 1. The SELinux Notebook

### 1.1 Introduction

This Notebook is split into two volumes:

1. **The Foundations** - that describes Security-Enhanced Linux (SELinux) services as built into the Fedora 12 release<sup>1</sup> of GNU / Linux.
2. **Sample Policy Source** - that contains sample policy and code to build a simple policy to experiment with a message filter and with X-Windows polyinstantiation.

These should help with explaining:

- a) SELinux and its purpose in life.
- b) The LSM / SELinux architecture, its supporting services and how they are implemented within GNU / Linux.
- c) The Virtual Machine, X-Windows, SE-PostgreSQL and Apache/SELinux-Plus SELinux-aware capabilities.
- d) The core SELinux policy language and how basic policy modules can be constructed for instructional purposes.
- e) The core SELinux policy management tools with examples of usage.
- f) The Reference Policy architecture, its supporting services and how it is implemented.

### 1.2 Volume 1 - The Foundations Overview

For reference Volume 1 - The Foundations has sections covering:

**SELinux Overview** - Gives a high level description of SELinux and its major components to provide Mandatory Access Control services for GNU / Linux. Hopefully it will show how all the SELinux components link together and how SELinux-aware applications and their object managers have been implemented (such as X-Windows, SE-PostgreSQL and virtual machines).

**SELinux Configuration Files** - Describes all the known SELinux configuration files in F-12 with samples. Also lists any specific commands or `libselinux` APIs used to manage them.

**SELinux Policy Language** - Gives a brief description of each policy language statement, with supporting examples taken from the Reference Policy source.

**The Reference Policy** - Describes the Reference Policy and its supporting macros.

**Object Classes and Permissions** - Describes the SELinux object classes and permissions. These have been updated to reflect those in the 20091117 Reference Policy build.

**SELinux Commands** - Describes each of the core SELinux commands.

---

<sup>1</sup> This Notebook uses Fedora 12 simply because that is what is installed on the authors test system.

**API Summary for libselinux** - Contains a sorted alphabetical list of `libselinux` library functions with comments extracted from the header files.

**SE-PostgreSQL Database Example** - Walks through setting up a simple database with each object created having a unique security context to demonstrate how they are implemented. Also shows the additional SE-PostgreSQL functions.

**General Information** - This section contains information about some minor problems encountered and information that could be useful.

**References** - List of references used within this Notebook.

### 1.3 Volume 2 - Sample Policy Source Overview

This volume contains a number of sample policy source files that have been written by the author to better understand SELinux. These do not use the Reference Policy but are built using SELinux policy language statements to form a very simple message filter that is then investigated using various SELinux tools. There are also sample X-Windows applications to demonstrate the XSELinux object manager that is now operational under F-12.

The Message Filter demonstrates:

- Building base and loadable policy modules.
- Using SECMARK, NetLabel and IPsec networking (via loop-back so no additional systems are required).

The X-windows applications demonstrate:

- Building base and loadable policy modules.
- Adding additional entries in the x-contexts configuration file.
- Simple copy and paste applications to show the difference between standard and polyinstantiated selections.
- Using the built-in XSELinux functions that Get/Set X-windows security context information.

The source software is available, however it is possible to copy and paste the code from the relevant sections of this Notebook into an editor such as `gedit`.

#### 1.3.1 Sample Policy Source Sections

This volume has the following sections:

**Building a Basic Policy** - Describes how to build monolithic, base and loadable policy modules using core policy language statements and SELinux commands. Note that these policies should not to be used in a live environment, they are examples to show simple policy construction.

**Building the Message Filter Loadable Modules** - Describes how to build a simple network and file handling application with policy using SECMARK and NetLabel services.

**Experimenting with X-Windows** - Builds sample copy and paste application and policy to demonstrate polyinstantiated selections. Also has a simple test application for the XSELinux extension Get/Set functions.

**Policy Investigation Tools** - Investigate the sample message filter application policy using the Tresys SETools apol, sechecker and sediff.

**NetLabel Module Support for network\_peer\_controls** - This builds on the modules developed in the Building the Message Filter section to implement an enhanced module to support the network peer controls.

**Labeled IPSec Module Example** - This builds on the modules developed in the Building the Message Filter section to implement Labeled IPSec.

**Implementing a constraint** - This builds on the modules developed in the Building a Basic Policy section to show a simple constraint statement and its impact on the policy.

### 1.4 Relevant F-12 Packages

The following are the relevant rpm packages installed on the test machine and used for all examples:

```
checkpolicy-2.0.19-3.fc12.i686
ipsec-tools-0.7.3-4.fc12.i686
kernel-2.6.31.5-127.fc12.i686
libselinux-2.0.90-5.fc12.i686
libsemanage-2.0.45-1.fc12.i686
libsepol-2.0.41-3.fc12.i686
netlabel_tools-0.19-3.fc12.i686

policycoreutils-2.0.79-1.fc12.i686
policycoreutils-gui-2.0.79-1.fc12.i686
policycoreutils-sandbox-2.0.79-1.fc12.i686
policycoreutils-python-2.0.79-1.fc12.i686
policycoreutils-newrole-2.0.79-1.fc12.i686

setools-3.3.6-4.fc12.i686
setools-console-3.3.6-4.fc12.i686
setools-gui-3.3.6-4.fc12.i686
setools-libs-3.3.6-4.fc12.i686
setools-libs-java-3.3.6-4.fc12.i686
setools-libs-tcl-3.3.6-4.fc12.i686
```

The gcc tools will be required to compile and link the test 'C' applications used in some of the scenarios (gcc-4.4.2-20.i686 and libgcc-4.4.2-20.i686 rpms are installed on the test machine that is using the kernel-2.6.31.5-127.fc12.i686 rpm).

## 2. Building a Basic Policy

### 2.1 Introduction

The objective of this section is to show how policy files are constructed, compiled and loaded using the SELinux command line tools and editors such as `vi` or `gedit` to produce a usable policy for instructional use only.

A monolithic and modular (with loadable modules) policy are built without the use of any support macros or make files from the Reference Policy source.

Important Note: While these are simple policies, they are built to support X-Windows therefore an `x_contexts` file must be installed. This file is required by the X-Windows SELinux object manager.

It is recommended that the `notebook-source-1.1.0-1.tar.gz` file is installed in `$HOME` as this contains all the configuration files and source code required to produce the required modules. It also contains README and a simple Makefile for each section.

#### 2.1.1 Overall Objectives

The main objectives of the sections that follow are to:

1. Show how to construct and build a simple monolithic and base policy.
2. Show how to construct and build a series of loadable modules for use with the base module. This builds into a very [simple message filter](#) using a network client / server application and file moving (filter) application. To examine the message filter application policy, some very minor policy errors have been introduced into the modules that will then be investigated using the SETools package in [Appendix A - Policy Investigation Tools](#).
3. Demonstrate simple [X-windows select and paste](#) applications using customised `x_contexts` files to show the different between standard (as used by the Reference Policy) and polyinstantiation selections using the XSELinux object manager / XACE services.
4. Build a test application that allows the XSELinux `SELinuxGet..` and `SELinuxSet..` functions to be tested.

#### 2.1.2 Build Requirements

To be able to build the policy files only standard SELinux utilities are required. However to build the test 'C' programs development tools will be required, therefore ensure that the following are installed:

- `gcc` tools to compile and link the test applications (`gcc` and `libgcc` packages).
- The `libselinux` library and `libselinux-devel` packages.
- For the X-Windows examples the the following will also be required:

- The Xlibpackages `libX11`, `libX11-common` and `libX11-devel`.
- For retrieving Xdevice information `libXi` and `libXi-devel`.

If the NetLabel module is being built, the NetLabel tools will need to be installed as they are not part of the standard F-12 installation (`netlabel_tools`).

If the Tresys utilities are used (`apol`, `sechecker` etc.), then it is recommended that the policies are built uncompressed by adding the following entry in the `semanage.conf` file:

```
bzip-blocksize=0
```

### 2.1.3 The Test Policies

Normally SELinux policies are built to deny everything by default, and then enable access as required, however the example policies in this section grant access to everything and then run the test applications in their own domains to isolate them.

The policies built in this section have been tested using the follow sequence:

1. Will the system load, allow users to logon and run applications in permissive mode – If yes then:
2. Set the system to enforcing mode by `setenforce 1`, if still okay then:
3. Log out users and log in again (as now in enforcing mode, the login may fail), if okay then:
4. Edit the `config` file and set `SELINUX=enforcing`, then reboot the system, if okay then:
5. Log in users and run applications, if okay then:
6. Test that the policy meets the security requirements.

If at any stage the load fails, then the repair CD/DVD may have to be used to investigate the cause. Setting the `config` file `SELINUX` entry to `permissive` and investigating the messages and audit logs can be helpful (but not always).

## 2.2 Building The Policy Source Files

There are at least three ways to build a monolithic or base policy source file to experiment with:

1. Use the samples shown in this section that are valid for F-12. However as SELinux gets updated, the object classes and their associated permissions do change, therefore these samples may not be correct for other versions or distributions.
2. Rebuild the source files using the `flask` `security_classes`, `initial_sids` and `access_vectors` files from the Reference Policy source appropriate to the GNU / Linux distribution being used. The policy statements must then be added as necessary.

The buildpolicy script shown below can be used to produce the complete policy automatically from the Reference Policy source<sup>2</sup> using the flask security\_classes, initial\_sids and access\_vectors files.

```
#!/bin/sh
#
#####
#
# This script will build an SELinux monolithic or base policy file suitable
# for building test policy for use in the SELinux Notebook. The Reference
# Policy must be available for this script to build the policy.
#
# A full description of its use is in the SELinux Notebook
#
# Copyright (C) 2010 Richard Haines
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <http://www.gnu.org/licenses/>.
#
#####
#

usage() {
    echo "Command format is: ./buildpolicy <policy_name>
<ref_policy_root_dir>"
    echo "Examples:"
    echo "buildpolicy base.conf ."
    echo "Or:"
    echo "./buildpolicy policy.conf $HOME/rpmbuild/SOURCES/serefpolicy-3.5.13"
    exit 1
}

if test "$1" = ""
then echo "Need policy file name"
    usage
fi

if test ! -f "$2/policy/flask/security_classes"
then echo "Not a valid Reference Policy source tree"
    usage
fi

echo -e "#\n# ***** WARNING - THIS POLICY MUST NOT BE USED IN LIVE
*****\n# ***** IT IS FOR TESTING ONLY
*****\n#" > $1

echo -e "##### START OF POLICY BUILD
#####\n#" >> $1

echo -e "#\n##### Start of FLASK Entries
#####\n#" >> $1
echo -e "#\n# ./policy/flask/security_classes file entries\n#" >> $1
cat "$2/policy/flask/security_classes" >> $1

echo -e "#\n# ./policy/flask/initial_sids file entries\n#" >> $1
cat "$2/policy/flask/initial_sids" >> $1

echo -e "#\n# ./policy/flask/access_vectors file entries\n#" >> $1
cat "$2/policy/flask/access_vectors" >> $1

echo -e "\n#\n##### End of FLASK Entries
#####\n#\n#" >> $1
```

---

<sup>2</sup> The X-Windows to work the minimum reference policy build is 20091117.

```
echo -e "#\n# This polycyap statement will be used in a netlabel module
exercise\n# to show network_peer_controls. For now comment out:\n#
polycyap network_peer_controls;\n" >> $1

echo -e "# The only type defined for this policy:" >> $1
echo -e "type unconfined_t;\n" >> $1

echo -e "# The only role defined for this policy:" >> $1
echo -e "role unconfined_r types { unconfined_t };\n" >> $1

echo -e "#\n# These allow rules enable all of the objects to access all of
their\n# permissions. This effectively gives access to everything.\n#" >>
$1
awk '$1 == "class" {print "allow unconfined_t self:"$2 " *;"}'
"$2/policy/flask/security_classes" >> $1

echo -e "\n# The only real SELinux user defined for this policy:" >> $1
echo -e "user user_u roles { unconfined_r };\n" >> $1

echo -e "#\n# The system_u user is defined so that objects can be labeled
with" >> $1
echo -e "# system_u:object_r as in standard policies, also so that semanage
can add" >> $1
echo -e "# ports etc. as it requires a system_u user for adding these type of
objects." >> $1
echo -e "user system_u roles { unconfined_r };\n" >> $1

echo -e "#\n# This role constraint statement will be used to show
limiting\n# a role transition in the external gateway. For now comment
out:\n# constrain process transition ( r1 == r2 );\n" >> $1

echo -e "#\n# These are the default labeling operations for these
objects.\n# Note that the kernel entry is unconfined_r not object_r\n#" >>
$1
awk '$1 == "sid" {if ($2 == "kernel") print $1 " " $2 "
system_u:unconfined_r:unconfined_t"}' "$2/policy/flask/initial_sids" >> $1
awk '$1 == "sid" {if ($2 != "kernel") print $1 " " $2 "
system_u:object_r:unconfined_t"}' "$2/policy/flask/initial_sids" >> $1

echo -e "\n#\n# These are the default file labeling routines.\n#" >> $1
echo "fs_use_xattr ext3 system_u:object_r:unconfined_t;" >> $1
echo "fs_use_xattr ext4 system_u:object_r:unconfined_t;" >> $1

echo "fs_use_task eventpollfs system_u:object_r:unconfined_t;" >> $1
echo "fs_use_task pipefs system_u:object_r:unconfined_t;" >> $1
echo "fs_use_task sockfs system_u:object_r:unconfined_t;" >> $1

echo "fs_use_trans mqueue system_u:object_r:unconfined_t;" >> $1
echo "fs_use_trans shm system_u:object_r:unconfined_t;" >> $1
echo "fs_use_trans tmpfs system_u:object_r:unconfined_t;" >> $1
echo "fs_use_trans devpts system_u:object_r:unconfined_t;" >> $1

echo "genfscon proc / system_u:object_r:unconfined_t" >> $1
echo "genfscon sysfs / system_u:object_r:unconfined_t" >> $1
echo "genfscon selinuxfs / system_u:object_r:unconfined_t" >> $1
echo "genfscon securityfs / system_u:object_r:unconfined_t" >> $1

echo -e "\n#\n# END OF POLICY BUILD
#####\n#\n" >> $1
```

3. There is an article "[SELinux From Scratch](#)" [Ref. 15] that describes a process using a C program and some scripts for building a test policy from the GNU / Linux kernel source tree. This process has since been enhanced and built into the kernel source tree from version 2.6.28, where the following files can be found that describe and build the 'make dummy policy'(mdp):

```
Documentation/SELinux.txt
scripts/Makefile
scripts/selinux/Makefile
```

```
scripts/selinux/README
scripts/selinux/install_policy.sh
scripts/selinux/mdp/Makefile
scripts/selinux/mdp/dbus_contexts
scripts/selinux/mdp/mdp.c
```

### 2.2.1 Policy Source Files

The policies built in this section make use of a common `policy.conf` source file to demonstrate a monolithic build and a base loadable policy build (traditionally called `base.conf`). The source is shown in the [Policy Source File](#) section with [Table 2-1](#) describing the core policy components.

| <i>Entry</i>                                   | <i>Comments</i>   |
|--|---|
| Security Classes ( <code>class</code> )        | These are from the Reference Policy (build 20091117) files as they have the correct X-Windows classes:<br><pre>./policy/flask/security_classes ./policy/flask/access_vectors ./policy/flask/initial_sids</pre>  |
| Access Vectors (permissions)                   |   |
| Initial SIDs                                   |   |
| MLS Sensitivity, category and level Statements | There are no MLS security level information in the sample policy.   |
| MLS Constraints                                | There are no MLS constraints in the sample policy.  |
| Policy Capability Statements                   | There are no <code>policycap</code> statements in the sample policy, however one is added later for a NetLabel exercise using <code>network_peer_controls</code> .  |
| Attributes                                     | There are no <code>attributes</code> in the sample policy.  |
| Booleans                                       | There are no <code>bool</code> statements in the sample policy.   |
| Type / Type Alias                              | There is only one <code>type:unconfined_t</code> . There are no <code>typealias</code> statements.  |
| Roles  | There is only one <code>role:unconfined_r</code> .  |
| Policy Rules                                   | There is one <code>allow</code> rule for each object class (taken from the <code>security_classes</code> file) in the policy that allows unrestricted access to all its permissions as follows:<br><pre>allow unconfined_t self : class_name *;</pre>   |
| Users  | There is one user: <code>user_u</code> , for logging on. The <code>system_u</code> user is there to allow objects to be labeled <code>system_u:object_r</code> as in the standard Reference Policy. The <code>system_u</code> user is also required by <code>semanage(8)</code> to add network objects. |
| Constraints                                    | There are no constraints in the sample policy, however one is added later to show role constraints.   |

| <i>Entry</i>   | <i>Comments</i>  |
|--|--|
| Default SID labeling   | These have been taken from the standard Reference Policy build with the security contexts updated. Note that the kernel is labeled <code>unconfined_r</code> and not <code>object_r</code> . |
| <code>fs_use_xattr</code> Statement  | Only the <code>ext3</code> and <code>ext4</code> filesystems have been added. If the system being built supports other filesystems then these will need to be added.                         |
| <code>fs_use_task</code> and <code>fs_use_trans</code> Statements                | These have been taken from the standard Reference Policy build.  |
| <code>genfscon</code> Statements   | Only a selection have been taken from the standard Reference Policy build.   |
| <code>portcon</code> , <code>netifcon</code> and <code>nodecon</code> Statements | There are none of these statements in the policy.  |

**Table 2-1: Policy Components** - *for the `policy.conf` and `base.conf` source file.*

### 2.2.1.1 Problem Resolution

The following may help with resolving issues when building the examples:

1. If the files are cut from this document and then pasted into a GNU / Linux editor (such as `vi` or `gedit`) as a text file, then there could be a `cr` at the end of each line. This can cause problems with some compilers such as `checkpolicy` and `checkmodule`. To remove the `cr` use the following command:

```
cat <file_name> | tr -d \\r >new_file_name
```

2. Once the policies etc. have been built and all goes well, the filesystem will relabeled and the new policy loaded during the reboot process, however any errors encountered will probably result in either:
  - a. GNU / Linux hanging, in which case the repair disk will be required. To allow GNU / Linux to load, the `/etc/selinux/config` file should be edited to set either `SELINUX=disabled` or the `SELINUXTYPE=` to a known working policy. The reason for the hang can then be investigated (such as correcting the policy source files and/or re-running the build commands).
  - b. The policy will be rejected by the kernel and not loaded, GNU / Linux will then load with no policy enabled, giving another chance at fixing the problem (the screen messages will generally give the reason for the rejection).

### 2.2.1.2 Monolithic and Base Policy Source File

The policy source file for monolithic and base loadable module is as follows:

## The SELinux Notebook - Sample Policy Source

```
#
# ***** WARNING - THIS POLICY MUST NOT BE USED IN LIVE *****
# ***** IT IS FOR TESTING ONLY *****
#
##### START OF POLICY BUILD #####
#
#
##### Start of FLASK Entries #####
#
#
# ./policy/flask/security_classes file entries
#
# FLASK
class security
class process
class system
class capability
class filesystem
class file
class dir
class fd
class lnk_file
class chr_file
class blk_file
class sock_file
class fifo_file
class socket
class tcp_socket
class udp_socket
class rawip_socket
class node
class netif
class netlink_socket
class packet_socket
class key_socket
class unix_stream_socket
class unix_dgram_socket
class sem
class msg
class msgq
class shm
class ipc
class passwd          # userspace
class x_drawable      # userspace
class x_screen        # userspace
class x_gc            # userspace
class x_font          # userspace
class x_colormap      # userspace
class x_property      # userspace
class x_selection     # userspace
class x_cursor        # userspace
class x_client        # userspace
class x_device        # userspace
class x_server        # userspace
class x_extension     # userspace
class netlink_route_socket
class netlink_firewall_socket
class netlink_tcpdiag_socket
class netlink_nflog_socket
class netlink_xfrm_socket
class netlink_selinux_socket
class netlink_audit_socket
class netlink_ip6fw_socket
class netlink_dnrt_socket
class dbus            # userspace
class nscd            # userspace
class association
class netlink_kobject_uevent_socket
class appletalk_socket
class packet
class key
class context         # userspace
class dccp_socket
class memprotect
class db_database     # userspace
```

```
class db_table          # userspace
class db_procedure     # userspace
class db_column        # userspace
class db_tuple         # userspace
class db_blob          # userspace
class peer
class capability2
class x_resource       # userspace
class x_event          # userspace
class x_synthetic_event # userspace
class x_application_data # userspace
class kernel_service
class tun_socket
class x_pointer        # userspace
class x_keyboard       # userspace
#
# ./policy/flask/initial_sids file entries
#
sid kernel
sid security
sid unlabeled
sid fs
sid file
sid file_labels
sid init
sid any_socket
sid port
sid netif
sid netmsg
sid node
sid igmp_packet
sid icmp_socket
sid tcp_socket
sid sysctl_modprobe
sid sysctl
sid sysctl_fs
sid sysctl_kernel
sid sysctl_net
sid sysctl_net_unix
sid sysctl_vm
sid sysctl_dev
sid kmod
sid policy
sid scmp_packet
sid devnull
#
# ./policy/flask/access_vectors file entries
#
common file { ioctl read write create getattr setattr lock relabelfrom relabelto
append unlink link rename execute swapon quotaon mounton }
common socket { ioctl read write create getattr setattr lock relabelfrom
relabelto append bind connect listen accept getopt setopt shutdown recvfrom
sendto recv_msg send_msg name_bind }
common ipc { create destroy getattr setattr read write associate unix_read
unix_write }
common database { create drop getattr setattr relabelfrom relabelto }
common x_device { getattr setattr use read write getfocus setfocus bell
force_cursor freeze grab manage list_property get_property set_property add
remove create destroy }
# Define the access vectors.
class filesystem { mount remount unmount getattr relabelfrom relabelto
transition associate quotamod quotaget }
class dir inherits file { add_name remove_name reparent search rmdir open }
class file inherits file { execute_no_trans entrypoint execmod open }
class lnk_file inherits file
class chr_file inherits file { execute_no_trans entrypoint execmod open }
class blk_file inherits file { open }
class sock_file inherits file { open }
class fifo_file inherits file { open }
class fd { use }
class socket inherits socket
class tcp_socket inherits socket { connectto newconn acceptfrom node_bind
name_connect }
class udp_socket inherits socket { node_bind }
class rawip_socket inherits socket { node_bind }
```

```
class node { tcp_rcv tcp_snd udp_rcv udp_snd rawip_rcv rawip_snd
enforce_dest dccp_rcv dccp_snd rcvfrom sendto }
class netif { tcp_rcv tcp_snd udp_rcv udp_snd rawip_rcv rawip_snd
dccp_rcv dccp_snd ingress egress }
class netlink_socket inherits socket
class packet_socket inherits socket
class key_socket inherits socket
class unix_stream_socket inherits socket { connectto newconn acceptfrom }
class unix_dgram_socket inherits socket
class process { fork transition sigchld sigkill sigstop signull signal ptrace
getsched setsched getsession getpgid setpgid getcap setcap share getattr setexec
setfscreate noatsecure siginh setrlimit rlimitinh dyntransition setcurrent
execmem execstack execheap setkeycreate setsockcreate }
class ipc inherits ipc
class sem inherits ipc
class msgq inherits ipc { enqueue }
class msg { send receive }
class shm inherits ipc { lock }
class security { compute_av compute_create compute_member check_context
load_policy compute_relabel compute_user setenforce setbool setseccparam
setcheckreqprot }
class system { ipc_info syslog_read syslog_mod syslog_console module_request }
class capability { chown dac_override dac_read_search fowner fsetid kill setgid
setuid setpcap linux_immutable net_bind_service net_broadcast net_admin net_raw
ipc_lock ipc_owner sys_module sys_rawio sys_chroot sys_ptrace sys_pacct
sys_admin sys_boot sys_nice sys_resource sys_time sys_tty_config mknod lease
audit_write audit_control setfcap }
class capability2 { mac_override mac_admin }
class passwd { passwd chfn chsh rootok crontab }
class x_drawable { create destroy read write blend getattr setattr list_child
add_child remove_child list_property get_property set_property manage override
show hide send receive }
class x_screen { getattr setattr hide_cursor show_cursor saver_getattr
saver_setattr saver_hide saver_show }
class x_gc { create destroy getattr setattr use }
class x_font { create destroy getattr add_glyph remove_glyph use }
class x_colormap { create destroy read write getattr add_color remove_color
install uninstall use }
class x_property { create destroy read write append getattr setattr }
class x_selection { read write getattr setattr }
class x_cursor { create destroy read write getattr setattr use }
class x_client { destroy getattr setattr manage }
class x_device inherits x_device
class x_server { getattr setattr record debug grab manage }
class x_extension { query use }
class x_resource { read write }
class x_event { send receive }
class x_synthetic_event { send receive }
class netlink_route_socket inherits socket { nlmsg_read nlmsg_write }
class netlink_firewall_socket inherits socket { nlmsg_read nlmsg_write }
class netlink_tcpdiag_socket inherits socket { nlmsg_read nlmsg_write }
class netlink_nflog_socket inherits socket
class netlink_xfrm_socket inherits socket { nlmsg_read nlmsg_write }
class netlink_selinux_socket inherits socket
class netlink_audit_socket inherits socket { nlmsg_read nlmsg_write nlmsg_relay
nlmsg_readpriv nlmsg_tty_audit }
class netlink_ip6fw_socket inherits socket { nlmsg_read nlmsg_write }
class netlink_dnrt_socket inherits socket
class dbus { acquire_svc send_msg }
class nscd { getpwd getgrp gethost getstat admin shmempwd shmegrp shmemhost
getserv shmempwd }
class association { sendto rcvfrom setcontext polmatch }
class netlink_kobject_uevent_socket inherits socket
class appletalk_socket inherits socket
class packet { send rcv relabelto flow_in flow_out forward_in forward_out }
class key { view read write search link setattr create }
class context { translate contains }
class dccp_socket inherits socket { node_bind name_connect }
class memprotect { mmap_zero }
class db_database inherits database { access install_module load_module
get_param set_param }
class db_table inherits database { use select update insert delete lock }
class db_procedure inherits database { execute entrypoint install }
class db_column inherits database { use select update insert }
class db_tuple { relabelfrom relabelto use select update insert delete }
```

## The SELinux Notebook - Sample Policy Source

```
class db_blob inherits database { read write import export }
class peer { recv }
class x_application_data { paste paste_after_confirm copy }
class kernel_service { use_as override create_files_as }
class tun_socket inherits socket
class x_pointer inherits x_device
class x_keyboard inherits x_device
#
##### End of FLASK Entries #####
#
#
# This polycyap statement will be used in a netlabel module exercise
# to show network_peer_controls. For now comment out:
# polycyap network_peer_controls;
#
# The only type defined for this policy:
type unconfined_t;
#
# The only role defined for this policy:
role unconfined_r types { unconfined_t };
#
# These allow rules enable all of the objects to access all of their
# permissions. This effectively gives access to everything.
#
allow unconfined_t self:security *;
allow unconfined_t self:process *;
allow unconfined_t self:system *;
allow unconfined_t self:capability *;
allow unconfined_t self:filesystem *;
allow unconfined_t self:file *;
allow unconfined_t self:dir *;
allow unconfined_t self:fd *;
allow unconfined_t self:lnk_file *;
allow unconfined_t self:chr_file *;
allow unconfined_t self:blk_file *;
allow unconfined_t self:sock_file *;
allow unconfined_t self:fifo_file *;
allow unconfined_t self:socket *;
allow unconfined_t self:tcp_socket *;
allow unconfined_t self:udp_socket *;
allow unconfined_t self:rawip_socket *;
allow unconfined_t self:node *;
allow unconfined_t self:netif *;
allow unconfined_t self:netlink_socket *;
allow unconfined_t self:packet_socket *;
allow unconfined_t self:key_socket *;
allow unconfined_t self:unix_stream_socket *;
allow unconfined_t self:unix_dgram_socket *;
allow unconfined_t self:sem *;
allow unconfined_t self:msg *;
allow unconfined_t self:msgq *;
allow unconfined_t self:shm *;
allow unconfined_t self:ipc *;
allow unconfined_t self:passwd *;
allow unconfined_t self:x_drawable *;
allow unconfined_t self:x_screen *;
allow unconfined_t self:x_gc *;
allow unconfined_t self:x_font *;
allow unconfined_t self:x_colormap *;
allow unconfined_t self:x_property *;
allow unconfined_t self:x_selection *;
allow unconfined_t self:x_cursor *;
allow unconfined_t self:x_client *;
allow unconfined_t self:x_device *;
allow unconfined_t self:x_server *;
allow unconfined_t self:x_extension *;
allow unconfined_t self:netlink_route_socket *;
allow unconfined_t self:netlink_firewall_socket *;
allow unconfined_t self:netlink_tcpdiag_socket *;
allow unconfined_t self:netlink_nflog_socket *;
allow unconfined_t self:netlink_xfrm_socket *;
allow unconfined_t self:netlink_selinux_socket *;
allow unconfined_t self:netlink_audit_socket *;
```

## The SELinux Notebook - Sample Policy Source

```
allow unconfined_t self:netlink_ip6fw_socket *;
allow unconfined_t self:netlink_dnrt_socket *;
allow unconfined_t self:dbus *;
allow unconfined_t self:nscd *;
allow unconfined_t self:association *;
allow unconfined_t self:netlink_kobject_uevent_socket *;
allow unconfined_t self:appletalk_socket *;
allow unconfined_t self:packet *;
allow unconfined_t self:key *;
allow unconfined_t self:context *;
allow unconfined_t self:dccp_socket *;
allow unconfined_t self:memprotect *;
allow unconfined_t self:db_database *;
allow unconfined_t self:db_table *;
allow unconfined_t self:db_procedure *;
allow unconfined_t self:db_column *;
allow unconfined_t self:db_tuple *;
allow unconfined_t self:db_blob *;
allow unconfined_t self:peer *;
allow unconfined_t self:capability2 *;
allow unconfined_t self:x_resource *;
allow unconfined_t self:x_event *;
allow unconfined_t self:x_synthetic_event *;
allow unconfined_t self:x_application_data *;
allow unconfined_t self:kernel_service *;
allow unconfined_t self:tun_socket *;
allow unconfined_t self:x_pointer *;
allow unconfined_t self:x_keyboard *;

# The only real SELinux user defined for this policy:
user user_u roles { unconfined_r };

#
# The system_u user is defined so that objects can be labeled with
# system_u:object_r as in standard policies, also so that semanage can add
# ports etc. as it requires a system_u user for adding these type of objects.
user system_u roles { unconfined_r };

#
# This role constraint statement will be used to show limiting
# a role transition in the external gateway. For now comment out:
# constrain process transition ( r1 == r2 );

#
# These are the default labeling operations for these objects.
# Note that the kernel entry is unconfined_r not object_r
#
sid kernel system_u:unconfined_r:unconfined_t
sid security system_u:object_r:unconfined_t
sid unlabeled system_u:object_r:unconfined_t
sid fs system_u:object_r:unconfined_t
sid file system_u:object_r:unconfined_t
sid file_labels system_u:object_r:unconfined_t
sid init system_u:object_r:unconfined_t
sid any_socket system_u:object_r:unconfined_t
sid port system_u:object_r:unconfined_t
sid netif system_u:object_r:unconfined_t
sid netmsg system_u:object_r:unconfined_t
sid node system_u:object_r:unconfined_t
sid igmp_packet system_u:object_r:unconfined_t
sid icmp_socket system_u:object_r:unconfined_t
sid tcp_socket system_u:object_r:unconfined_t
sid sysctl_modprobe system_u:object_r:unconfined_t
sid sysctl system_u:object_r:unconfined_t
sid sysctl_fs system_u:object_r:unconfined_t
sid sysctl_kernel system_u:object_r:unconfined_t
sid sysctl_net system_u:object_r:unconfined_t
sid sysctl_net_unix system_u:object_r:unconfined_t
sid sysctl_vm system_u:object_r:unconfined_t
sid sysctl_dev system_u:object_r:unconfined_t
sid kmod system_u:object_r:unconfined_t
sid policy system_u:object_r:unconfined_t
sid scmp_packet system_u:object_r:unconfined_t
sid devnull system_u:object_r:unconfined_t
```

```
#
# These are the default file labeling routines.
#
fs_use_xattr ext3 system_u:object_r:unconfined_t;
fs_use_xattr ext4 system_u:object_r:unconfined_t;
fs_use_task eventpollfs system_u:object_r:unconfined_t;
fs_use_task pipefs system_u:object_r:unconfined_t;
fs_use_task sockfs system_u:object_r:unconfined_t;
fs_use_trans mqueue system_u:object_r:unconfined_t;
fs_use_trans shm system_u:object_r:unconfined_t;
fs_use_trans tmpfs system_u:object_r:unconfined_t;
fs_use_trans devpts system_u:object_r:unconfined_t;
genfscon proc / system_u:object_r:unconfined_t
genfscon sysfs / system_u:object_r:unconfined_t
genfscon selinuxfs / system_u:object_r:unconfined_t
genfscon securityfs / system_u:object_r:unconfined_t

#
##### END OF POLICY BUILD #####
#
```

### 2.2.1.3 file\_contexts File

The file\_contexts file for the build is as follows:

```
/ system_u:object_r:unconfined_t
/* system_u:object_r:unconfined_t
```

### 2.2.1.4 default\_contexts File

The default\_contexts file is to ensure that the initial logon process uses the unconfined\_r:unconfined\_t role / type pair and is as follows:

```
unconfined_r:unconfined_t unconfined_r:unconfined_t
```

Note that this file will only be required when the additional loadable modules are built as they contain multiple types associated to a single role (therefore the logon process needs to know which of the types to use for the users user:role:type security context).

### 2.2.1.5 seusers File

The seusers file is not mandatory, however one is added as all policies tend to have one, also when adding additional users via semanage, one will be required.

```
system_u:system_u
user_u:user_u
__default__:user_u
```

### 2.2.1.6 dbus\_contexts File

The dbus\_contexts file is required to allow X-Windows to run and is as follows:

```
<!DOCTYPE busconfig PUBLIC "-//freedesktop//DTD D-BUS Bus Configuration 1.0//EN"
"http://www.freedesktop.org/standards/dbus/1.0/busconfig.dtd">
<busconfig>
  <selinux>
```

```
</selinux>  
</busconfig>
```

### 2.2.1.7 x\_contexts File

The x\_contexts file is required to allow X-Windows to run and is as follows (this is a modified version taken from Reference Policy 20091117):

```
client *                system_u:object_r:unconfined_t  
### Rules for X Properties  
property _SELINUX_*    system_u:object_r:unconfined_t  
property CUT_BUFFER?  system_u:object_r:unconfined_t  
property *             system_u:object_r:unconfined_t  
### Rules for X Extensions  
extension SELinux     system_u:object_r:unconfined_t  
extension *           system_u:object_r:unconfined_t  
### Rules for X Selections  
selection PRIMARY    system_u:object_r:unconfined_t  
selection CLIPBOARD  system_u:object_r:unconfined_t  
selection *          system_u:object_r:unconfined_t  
### Rules for X Events  
event X11:KeyPress    system_u:object_r:unconfined_t  
event X11:KeyRelease  system_u:object_r:unconfined_t  
event X11:ButtonPress system_u:object_r:unconfined_t  
event X11:ButtonRelease system_u:object_r:unconfined_t  
event X11:MotionNotify system_u:object_r:unconfined_t  
event XInputExtension:DeviceKeyPress system_u:object_r:unconfined_t  
event XInputExtension:DeviceKeyRelease system_u:object_r:unconfined_t  
event XInputExtension:DeviceButtonPress system_u:object_r:unconfined_t  
event XInputExtension:DeviceButtonRelease system_u:object_r:unconfined_t  
event XInputExtension:DeviceMotionNotify system_u:object_r:unconfined_t  
event XInputExtension:DeviceValuator system_u:object_r:unconfined_t  
event XInputExtension:ProximityIn system_u:object_r:unconfined_t  
event XInputExtension:ProximityOut system_u:object_r:unconfined_t  
event X11:ClientMessage system_u:object_r:unconfined_t  
event X11:SelectionNotify system_u:object_r:unconfined_t  
event X11:UnmapNotify system_u:object_r:unconfined_t  
event X11:ConfigureNotify system_u:object_r:unconfined_t  
event *              system_u:object_r:unconfined_t
```

## 2.3 Building the Monolithic Policy

The basic steps to produce a simple monolithic test policy are:

- 1) Ensure you are logged on as 'root' and SELinux is running in permissive mode (`setenforce 0`) to perform the build process. It is assumed that the files are built in the `./notebook-source/monolithic-policy` directory.
- 2) Produce a `policy.conf` file with a text editor (such as `vi` or `edit`) containing the contents shown in the [Policy Source File](#) section.
- 3) Produce a `file_contexts` file with the contents shown in the [file\\_contexts file](#) section. This will be used to relabel the file system.
- 4) Produce a `dbus_contexts` file with the contents shown in the [dbus\\_contexts file](#) section. This is required for X-Windows to load as it uses the dbus messaging service that has a SELinux user space object manager.

- 5) Produce a `x_contexts` file with the contents shown in the [x\\_contexts File](#) section. This is required for the X-Windows object manager.
- 6) Find the maximum policy version the SELinux kernel will support by executing the following command:

```
cat /selinux/policyvers
24
```

The output for the F-12 kernel should be '24' depending on any package updates that have been added.

- 7) Compile the policy with `checkpolicy` to produce the binary policy file:

```
checkpolicy -c24 -o policy.24 policy.conf
```

The output from the compilation should be:

```
checkpolicy: loading policy configuration from policy.conf
checkpolicy: policy configuration loaded
checkpolicy: writing binary representation (version 24) to
policy.24
```

- 8) Make the following directories to store the policy:

```
mkdir /etc/selinux/monolithic-test/policy
mkdir -p /etc/selinux/monolithic-test/contexts/files
```

- 9) Copy the following files to SELinux policy area:

```
cp policy.24 /etc/selinux/monolithic-test/policy
cp seusers /etc/selinux/monolithic-test/seusers
cp dbus_contexts /etc/selinux/monolithic-test/contexts
cp x_contexts /etc/selinux/monolithic-test/contexts
cp default_contexts /etc/selinux/monolithic-test/contexts
cp file_contexts /etc/selinux/monolithic-
test/contexts/files
```

- 10) The file and directory list in the `/etc/selinux/monolithic-test` directory area should now consist of the following:

```
monolithic-test:
drwxr-xr-x 3 root root 4096 2010-02-25 13:04 contexts
drwxr-xr-x 2 root root 4096 2010-02-25 13:59 policy
-rw-r--r-- 1 root root 51 2010-02-25 14:00 seusers

monolithic-test/contexts:
-rw-r--r-- 1 root root 195 2010-02-25 13:04 dbus_contexts
-rw-r--r-- 1 root root 53 2010-02-25 13:04 default_contexts
drwxr-xr-x 2 root root 4096 2010-02-25 14:00 files
-rw-r--r-- 1 root root 2764 2010-02-25 13:04 x_contexts

monolithic-test/contexts/files:
-rw-r--r-- 1 root root 69 2010-02-25 14:00 file_contexts

monolithic-test/policy:
-rw-r--r-- 1 root root 12457 2010-02-25 13:59 policy.24
```

- 11) Edit the `/etc/selinux/config` file and change the entries shown. This will set permissive mode and the location of the policy that will be loaded on the next re-boot. Note - do not put any spaces after these entries.

```
SELINUX=permissive
SELINUXTYPE=monolithic-test
```

- 12) To allow file system relabeling to be actioned on reboot execute the following command:

```
touch /.autorelabel
```

- 13) Optionally clear the log files so that they are clear for easier reading after the reboot:

```
> /var/log/messages
> /var/log/audit/audit.log
```

- 14) Reboot the system. During the boot process, the file system should be re-labeled.

```
reboot
```

### 2.3.1 Checking the Build

Once the system has reloaded, SELinux will be running in ‘permissive’ mode. Logon as root and use either `seaudit`, `troubleshooter` or simply `tail` in a couple of ‘terminal windows’ to view the logs:

```
# In one terminal window run:
tail -f /var/log/messages

# In another terminal window run:
tail -f /var/log/audit/audit.log
```

There should be entries for the boot process in the `/var/log/messages` file, however the `/var/log/audit/audit.log` file should only contain entries for the audit daemon, user logon and role change for the logon process.

If the system is ‘working’ (i.e. it should be stable, load the desktop and allow utilities to be loaded from the menus), then SELinux can be set to enforcing mode by:

```
setenforce 1
```

The new policy will be enforced and the only entries in the logs should be about setting enforcing mode.

If the system is unstable when rebooted, then see the [Problem Resolution](#) section for a possible resolution.

## 2.4 Building the Base Policy Module

This exercise will build the mandatory base policy module that uses the same policy source file as the monolithic policy discussed above.

The basic steps to produce a simple base test policy are:

1. Ensure you are logged on as 'root' and SELinux is running in permissive mode (`setenforce 0`) to perform the build process. It is assumed that the files are built in the `./notebook-source/modular-base-policy` directory.
2. Produce a `base.conf` file with a text editor (such as `vi` or `gedit`) containing the contents shown in the [Policy Source File](#) section.
3. Produce a `base.fc` file with the contents shown in the [file\\_contexts](#) file section. This will be used to relabel the file system.
4. Produce a `default_contexts` file with the contents shown in the [default\\_contexts](#) file section. This will be used to ensure that the correct context is used for the logon process (only really needed when the additional loadable modules are built).
5. Produce an `seusers` file with the contents shown in the [seusers](#) file section.
6. Produce a `dbus_contexts` file with the contents shown in the [dbus\\_contexts](#) file section. This is required for X-Windows to load as it uses the `dbus` messaging service that has a SELinux user space object manager.
- 15) Produce a `x_contexts` file with the contents shown in the [x\\_contexts File](#) section. This is required for the X-Windows object manager.
7. Compile the policy with `checkmodule` to produce an intermediate binary policy file:

```
checkmodule -o base.mod base.conf
```

The output from the compilation should be:

```
checkmodule: loading policy configuration from base.conf
checkmodule: policy configuration loaded
checkmodule: writing binary representation (version 10) to base.mod
```

8. Package the policy with `semodule_package`, this will produce a base policy module file (note – if successful there are no output messages):

```
semodule_package -o base.pp -m base.mod -f base.fc -s
seusers
```

9. Make the following directories to store the policy:

```
mkdir /etc/selinux/modular-test/policy
mkdir -p /etc/selinux/modular-test/contexts/files
mkdir -p /etc/selinux/modular-test/modules/active/modules
```

10. Copy the following files to SELinux policy area:

```
cp seusers /etc/selinux/modular-test
cp dbus_contexts /etc/selinux/modular-test/contexts
cp default_contexts /etc/selinux/modular-test/contexts
cp x_contexts /etc/selinux/modular-test/contexts
```

11. Install the base policy with `semodule`. This will produce a base policy and a number of files, some of which will be empty (note – if successful there are no output messages):

```
semodule -s modular-test -b base.pp
```

12. The file and directory list in the `/etc/selinux/modular-test` directory area should now consist of the following:

```
/etc/selinux/modular-test:
drwxr-xr-x. 3 root root 4096 2010-02-28 15:57 contexts
drwxr-xr-x. 3 root root 4096 2010-02-28 15:57 modules
drwxr-xr-x. 2 root root 4096 2010-02-28 15:57 policy
-rw-r--r--. 1 root root  51 2010-02-28 15:57 seusers

/etc/selinux/modular-test/contexts:
-rw-r--r--. 1 root root  195 2010-02-28 15:57 dbus_contexts
-rw-r--r--. 1 root root   53 2010-02-28 15:57 default_contexts
drwxr-xr-x. 2 root root 4096 2010-02-28 15:57 files
-rw-r--r--. 1 root root    0 2010-02-28 15:57 netfilter_contexts
-rw-r--r--. 1 root root 2764 2010-02-28 15:57 x_contexts

/etc/selinux/modular-test/contexts/files:
-rw-r--r--. 1 root root  68 2010-02-28 15:57 file_contexts
-rw-r--r--. 1 root root   0 2010-02-28 15:57 file_contexts.homedirs

/etc/selinux/modular-test/modules:
drwx-----. 3 root root 4096 2010-02-28 15:57 active
-rw-----. 1 root root    0 2010-02-28 15:57 semanage.read.LOCK
-rw-----. 1 root root    0 2010-02-28 15:57 semanage.trans.LOCK

/etc/selinux/modular-test/modules/active:
-rw-r--r--. 1 root root 22625 2010-02-28 15:57 base.pp
-rw-----. 1 root root   32 2010-02-28 15:57 commit_num
-rw-----. 1 root root   68 2010-02-28 15:57 file_contexts
-rw-r--r--. 1 root root    0 2010-02-28 15:57 file_contexts.homedirs
-rw-----. 1 root root   68 2010-02-28 15:57 file_contexts.template
-rw-----. 1 root root    0 2010-02-28 15:57 homedir_template
drwx-----. 2 root root 4096 2010-02-28 15:57 modules
-rw-----. 1 root root    0 2010-02-28 15:57 netfilter_contexts
-rw-r--r--. 1 root root 12457 2010-02-28 15:57 policy.kern
-rw-----. 1 root root   51 2010-02-28 15:57 seusers.final
-rw-----. 1 root root   25 2010-02-28 15:57 users_extra

/etc/selinux/modular-test/modules/active/modules:

/etc/selinux/modular-test/policy:
-rw-r--r--. 1 root root 12457 2010-02-28 15:57 policy.24
```

13. Edit the `/etc/selinux/config` file and change the entries shown. This will set permissive mode and the policy location that will be loaded on the next re-boot. Note - do not put any spaces after these entries.

```
SELINUX=permissive
SELINUXTYPE=modular-test
```

This will set permissive mode so if the policy is too restrictive it will still allow a login at least. The SELinux policy name/location is also added (`modular-test`). Note do not put any spaces after the entries.

14. To allow a file system relabeling to be actioned on reboot execute the following command:

```
touch /.autorelabel
```

15. Optionally clear the log files so that they are clear for easier reading:

```
> /var/log/messages  
> /var/log/audit/audit.log
```

16. Reboot the system. During the boot process, the file system should be re-labeled.

```
reboot
```

### 2.4.1 Checking the Base Policy Build

Once the system has reloaded, SELinux will be running in 'permissive' mode. Logon as root and follow the same routine as defined in the [Checking The Build](#) section.

## 3. Building the Message Filter Loadable Modules

### 3.1 Overview of modules

In the sections that follow there are a number of loadable modules built with supporting 'C' programs that form a very simple message filter service as shown in [Figure 3.1](#). The external and internal gateways are client / server applications making use of SEMARK services that are built into `iptables` as discussed in SELinux Networking section of 'The Foundations' volume. The server component can also read and write files to / from a protected directory area (or message queues). The message filter itself is a simple file mover application that moves files from one queue to another.

The modules attempt to use as many SELinux statements and rules as possible that are then analysed in [Appendix A - Policy Investigation Tools](#).

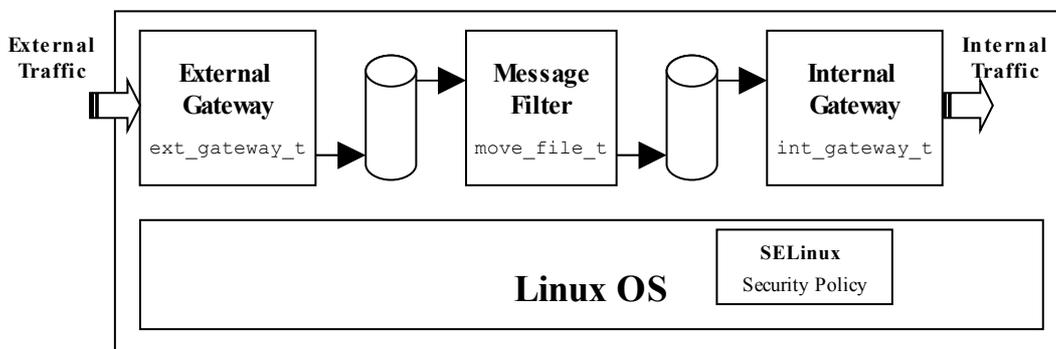


Figure 3.1: Message Filter Components

The components that form the message filter are:

**External Gateway** – This has a loadable module `ext_gateway.conf` that defines the policy for the external gateway, it also includes an `optional` section that is loaded when other message filter modules are loaded. The gateway requires client / server applications (`client.c` and `server.c`) to be compiled for testing and `iptables` (the `mangle` table) to be loaded for `SECMARK` testing.

**NetLabel Service** – This is a simple `netlabel.conf` module that just adds a label at peer level. As F-12 does not have NetLabel services installed as default, `yum` is used to install the service.

**Internal Gateway** – This is a version of the external gateway module that has been modified to handle internal processing permissions (`int_gateway.conf`). It requires additional entries in the `iptables` as it uses a different network port. The same client / server applications are used as for the external gateway.

**File Mover** - This has a loadable module `move_file.conf` that defines the policy for moving files between the external and internal gateways. There is also an application (`move_file.c`) that copies files from one message queue to another (but controlled by the policy).

The security policy for the message filter is simply:

1. No other application must use the secured ports configured in the `iptables` and allocated to the gateways. The secure ports are:

`port 1111 and labeled int_gateway_packet_t`

`port 9999 and labeled ext_gateway_packet_t`

All other ports are labeled: `default_secmark_packet_t`

2. The message queues and files must be protected from all possible access (read, write, delete etc.) by other domains.

The assumptions are:

1. The SELinux policy will always be in enforcing mode while the message filter is active.
2. The SELinux message filter modules may be in permissive mode for the initial file and directory configuration / initialisation via `restorecon` (this is so that permissions such as `relabelto` / `relabelfrom` are limited to the absolute minimum, in fact only the `iptables` need relabeling permissions as they are loaded under the `unconfined_t` domain).

The modules are built and tested in the following sequence:

1. The external gateway is built along with the client / server applications. This is used to demonstrate the basic `secmark` functionality using the `iptables`.
2. The `NetLabel` module is then built to demonstrate adding a `netlabel` to the peer network service.
3. The internal gateway and the file mover application and module are finally built to demonstrate the overall message filter as shown in [Figure 3.1](#).

Once these have been built and tested, the policy will be reviewed in [Appendix A - Policy Investigation Tools](#).

Any comments or views on the modules, applications and their testing are welcome.

## **3.2 Building the SECMARK Test Loadable Module**

The SECMARK tests make use of the external gateway loadable module. The objective of this module is to prove that SECMARK labels can be added to packets, and that depending on the label assigned, those packets can be granted access to the correct domain and denied access other domains using SELinux enforcement.

The tests will use various client / server configurations using the network loop back (`lo`) interface (see [Figure 3.2](#)) as follows:

1. Use a 'secure' client / server running in the `ext_gateway_t` domain that will show that packets labeled:

`system_u:object_r:ext_gateway_packet_t`

on ports 9999 will get through, while other ports will NOT get through, as they would be labeled:

`system_u:object_r:default_secmark_packet_t`

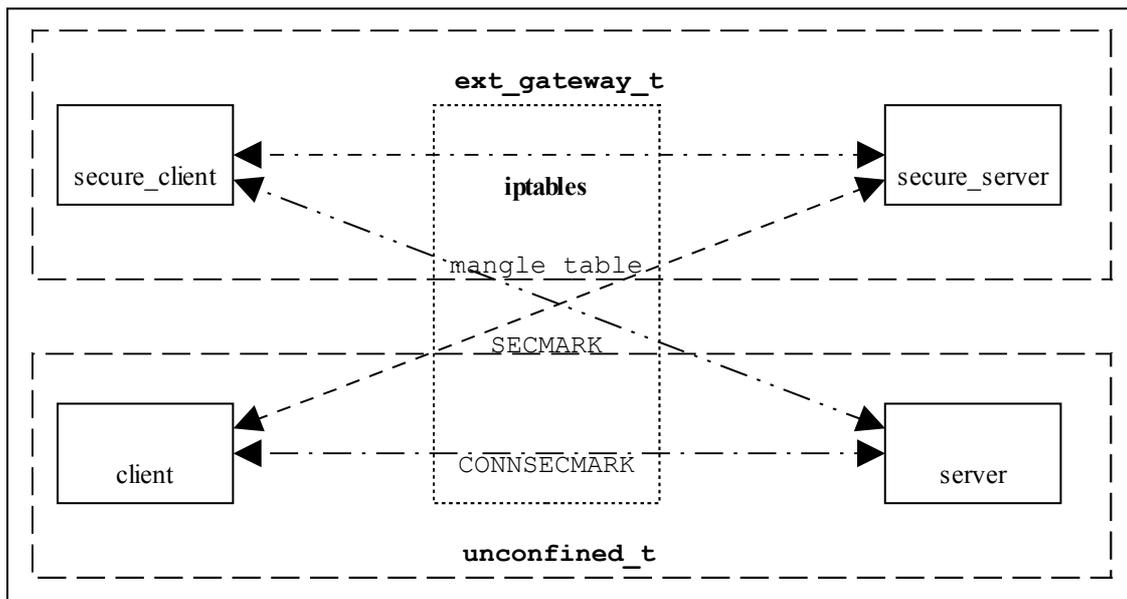
- Use an 'insecure' client / server running in the `unconfined_t` domain that will show that packets labeled:

```
system_u:object_r: ext_gateway_packet_t
```

will NOT get through, while other ports will get through, as they would be labeled:

```
system_u:object_r:default_secmark_packet_t
```

- A mixture of secure and insecure client / server configurations to show access is denied by SELinux unless both services are running in the `ext_gateway_t` domain using port 9999 on the `lo` network.



**Figure 3.2: SECMARK Testing** – The scenarios for testing the access allowed for SECMARK packets. Note that not all of these tests will be described.

The SECMARK test loadable module (`ext_gateway.conf`) has a boolean called `ext_gateway_audit` that by default enables the transition, send and receive audit events to be logged when successful, these events are shown in the test results below. The `auditallow` statements can be disabled by using the following command:

```
setsebool -P ext_gateway_audit=false
```

To test SECMARK functionality the following will need to be built and installed:

- The base loadable module built in the [Building a Base Loadable Module Policy](#) section is installed and active.
- A loadable policy module (`ext_gateway`) that will enforce the SECMARK policy configured via `iptables`. Note the following points:
  - The `ext_gateway` module requires a new role of `message_filter_r` to be added to SELinux. This has only been added to demonstrate a `role_transition` rule.

- b. The `ext_gateway` module has an optional section that is only enabled when other modules are loaded as a further exercise for the message filter service.
- An iptables configuration file that will set-up the mangle table to mark packets with `SECMARK` and `CONNSECMARK` labels.
- Two executable clients (`secure_client` and `client`) and two executable servers (`secure_server` and `server`) that will be used to test the `SECMARK` functionality. Note that the clients and servers are built using common source code, and are only labeled differently to allow testing (the secure executables are labeled `secure_services_exec_t` while the others are labeled by default with `unconfined_t`).

The following steps need to be followed to build the test services. It is assumed that the services are installed in `./notebook-source/message-filter/gateway`:

1. Ensure you are logged on as 'root' and SELinux is running in permissive mode (`setenforce 0`) to perform the build process.
2. Produce a `ext_gateway.conf` loadable module file with a text editor (such as `vi` or `gedit`) containing the contents shown below:

```
module ext_gateway 1.1.0;

#####
#
# This Loadable Module will allow SECMARK, NetLabel and a simple
# Message Filter to be tested with the simple client / server 'C'
# programs (client.c and server.c).
#
# The module is used with the base.conf that sets up the unconfined_t
# space. This module can be built by:
#   checkmodule -m ext_gateway.conf -o ext_gateway.mod
#   semodule_package -o ext_gateway.pp -m ext_gateway.mod
#   -f gateway.fc
#   semodule -v -s modular-test -i ext_gateway.pp
#
# The gateway.fc file can be modified to reflect where the client /
# binaries will located (currently /usr/local/bin) and once they have
# been compiled the exe's need to be labeled by restorecon:
#   restorecon -f restorefiles_gateway
#
# The test requires the client.c and server.c programs compiled by:
#   gcc -o secure_server server.c -lselinux
#   gcc -o secure_client client.c -lselinux
# The executables will be labeled secure_services_exec_t
#
#   gcc -o server server.c -lselinux
#   gcc -o client client.c -lselinux
# The executables will be labeled the default of unconfined_t
#
# The secure port for this external gateway is 9999 and can only be
# read/write by the secure client / server. The internal gateway will
# use port 1111 and can only be read/write by the secure client / server.
# Any other port can be read / write by the standard client / server.
#
# The iptables_secmark script can be modified to other ports if required.
# WARNING - If the iptables are not loaded to label packets, ports etc.
# then the standard client / server can use the secure ports.
#
# Run setenforce 1, the policy can be tested using combinations of:
#   ./server <port>
#   ./secure_server <port>
#
```

## The SELinux Notebook - Sample Policy Source

```
# ./client <host> <port> #
# ./secure_client <host> <port> #
# #
# There is a boolean that can turn off the auditallow statements: #
# setsebool [-P] ext_gateway_audit off #
# #
# The module transitions to a role of message_filter_r simply to show #
# a role transition. To add the role to user_u the semanage command is #
# used as follows: #
# semanage user -m -R "message_filter_r unconfined_r" user_u #
# Note: Need to put in the unconfined_r role as semanage will remove it #
# from the current policy otherwise, causing much grief. #
# #
# To allow the message filter to be tested with the move_file.conf #
# (and int_gateway.conf) modules, a set of optional statements are #
# added that will be enabled once the move_file.conf module is loaded #
# for testing. This allows the server application to either read or #
# write files to the message queues as described in the server.c #
# comments section (and the SELinux Notebook). #
# #
#####
require {
    type unconfined_t;
    role unconfined_r;
    class packet { send recv relabelto };
    class process { fork sigchld transition siginh rlimitinh noatsecure signal };
    class file { entrypoint read getattr execute relabelto unlink write create };
    class filesystem { getattr associate };
    class chr_file { read write getattr };
    class dir { read search getattr write add_name remove_name };
    class fd use;
    class lnk_file read;
    class tcp_socket { write listen node_bind name_bind accept bind read name_connect connect create getopt };
    class association recvfrom;
    class unix_stream_socket { create connect };
}

attribute message_filter_domains;

# All iptables SECMARK packets other than ports 9999 and 1111 are marked
# with this label:
type default_secmark_packet_t;

# The external gateway will have a SECMARK in the iptables of this label:
type ext_gateway_packet_t;

# The external gateway will run in this domain:
type ext_gateway_t;
# The binaries will be labeled:
type secure_services_exec_t;

# Add the gateway domain to the attribute:
typeattribute ext_gateway_t message_filter_domains;

# Use message_filter_r role and role transition for the gateway:
role message_filter_r types ext_gateway_t;
allow unconfined_r message_filter_r;
role_transition unconfined_r secure_services_exec_t message_filter_r;

# boolean to enable / disable audit events
bool ext_gateway_audit true;

if ( ext_gateway_audit ) {
    # Audit the send and recv events:
    auditallow unconfined_t default_secmark_packet_t : packet { send
recv };
    # Audit the send and recv events:
    auditallow ext_gateway_t ext_gateway_packet_t : packet { send recv };
    # Audit the transition:
    auditallow unconfined_t ext_gateway_t : process { transition };
}
}
```

## The SELinux Notebook - Sample Policy Source

```
# End of conditional statements

# Allow all ports except 1111 & 9999 to be handled by unconfined_t:
allow unconfined_t default_secmark_packet_t : packet { send recv };

# Allow unconfined_t to relabel the secure ports. This is needed so that
# iptables can be updated easily. Note: Against security policy, however
# these need to be loaded at boot time when the policy is in enforcing mode
# so no choice !!
allow unconfined_t ext_gateway_packet_t : packet relabelto;
allow unconfined_t default_secmark_packet_t : packet relabelto;

# Allow gateway access only to secure ports:
allow ext_gateway_t ext_gateway_packet_t : packet { send recv };

# Allow the external gateway to transition to ext_gateway_t by using the
# type_transition statement (note that the internal gateway does not use
# this method but transitions via runcon instead):
allow unconfined_t ext_gateway_t : process { transition };
allow unconfined_t secure_services_exec_t : file { read execute getattr };
allow ext_gateway_t secure_services_exec_t : file { entrypoint };
type_transition unconfined_t secure_services_exec_t : process
ext_gateway_t;
#

# Stop segmentation faults
allow ext_gateway_t unconfined_t : filesystem associate;
allow unconfined_t ext_gateway_t : process noatsecure;
dontaudit unconfined_t ext_gateway_t : process { siginh rlimitinh };
#
# Need this in F-12 build to allow the client / server apps to exit:
allow unconfined_t ext_gateway_t : process signal;

# Allow the ext_gateway_t access to areas under unconfined_t domain:
allow ext_gateway_t unconfined_t : packet { recv send };
allow ext_gateway_t unconfined_t : chr_file { read write getattr };
allow ext_gateway_t unconfined_t : dir search;
allow ext_gateway_t unconfined_t : fd use;
allow ext_gateway_t unconfined_t : filesystem getattr;
allow ext_gateway_t unconfined_t : tcp_socket name_connect;
allow ext_gateway_t unconfined_t : association recvfrom;
allow ext_gateway_t self : dir search;
allow ext_gateway_t self : tcp_socket { read create connect };

# Need this in F-12 build to allow the client / server apps to exit:
allow ext_gateway_t unconfined_t : process sigchld;
# This was the F-10 statement:
# dontaudit ext_gateway_t unconfined_t : process sigchld;

# For client and server to access the shared libc:
allow ext_gateway_t unconfined_t : file { read getattr execute };
dontaudit ext_gateway_t unconfined_t : dir { getattr };
allow ext_gateway_t unconfined_t : lnk_file read;
#

# Required if use host name instead of the IP address (e.g. localhost
# instead of 127.0.0.1) in the client command line:
dontaudit ext_gateway_t self : unix_stream_socket { create connect };

# Required to get context information when using the libselinux api calls
# getcon() and getpeercon():
allow ext_gateway_t self : file read;
allow ext_gateway_t self : tcp_socket getopt;

# Required to allow setfiles to relabel the secure client/server binaries:
# Note: Against security policy so commented out as can do this at
# system build time with setenforce 0
# allow unconfined_t secure_services_exec_t : file { write relabelto };
# allow secure_services_exec_t unconfined_t : filesystem associate;

# These entries are for the server only:
allow ext_gateway_t self : tcp_socket { listen write accept bind };
allow ext_gateway_t unconfined_t : tcp_socket { name_bind node_bind };
#
```

```
#
##### START OPTIONAL SECTION #####
#
optional {
#
#####
#
# These entries are for the message filter part of the exercise #
# where files are moved from the in_queue to the out_queue by the #
# message filter (move_file.c) application. #
# #
# These rules allow ext_gateway_t to write files to the #
# in_queue. The int_gateway_t is allowed to read and remove #
# files from the out_queue. #
# #
#####
#
require {
# These are defined in the move_file.conf module:
type in_queue_t, out_queue_t, in_file_t, out_file_t;
# This is defined in the int_gateway.conf module:
type int_gateway_t;
}
# Allow the external gateway access to in_queue rules:
# The server application then writes the file to the in_queue:
type_transition ext_gateway_t in_queue_t : file in_file_t;
allow ext_gateway_t in_queue_t : dir { read getattr write search
add_name };
allow ext_gateway_t in_file_t : file { write create getattr };
allow in_file_t unconfined_t : filesystem associate;
dontaudit ext_gateway_t unconfined_t : filesystem getattr;
dontaudit ext_gateway_t self : file getattr;

# Allow the internal gateway access to out_queue rules:
type_transition int_gateway_t out_queue_t : file out_file_t;
allow int_gateway_t out_queue_t : dir { read getattr write remove_name
search };
allow int_gateway_t out_file_t : file { read getattr unlink };
}
#
##### END OPTIONAL SECTION #####
#
```

3. Produce a `gateway.fc` file (a segment that will be added to the `file_contexts` file during the build) with the contents shown below. This will be used to relabel application files and directories.

```
/usr/local/bin/secure_client system_u:object_r:secure_services_exec_t
/usr/local/bin/secure_server system_u:object_r:secure_services_exec_t
/usr/local/bin/client system_u:object_r:unconfined_t
/usr/local/bin/server system_u:object_r:unconfined_t
```

4. Compile the policy with `checkmodule` to produce an intermediate binary policy file:

```
checkmodule -m ext_gateway.conf -o ext_gateway.mod
```

5. Package the policy with `semodule_package`, this will produce a policy module file:

```
semodule_package -o ext_gateway.pp -m ext_gateway.mod -f gateway.fc
```

6. Install the loadable module with `semodule` (note – if successful there are no output messages):

```
semodule -v -s modular-test -i ext_gateway.pp
```

7. If there are no errors reported, then the loadable module has been added to the policy store and loaded as a part of the policy. The policy module can be checked by:

```
semodule -s modular-test -l
```

8. Produce a 'C' application called `client.c` with the contents shown below:

```
/*
/* *****
/* This is the client component for the Notebook demo modular policy. It
/* will be used to demonstrate SECMARK, NetLabel and Message Filter
/* loadable modules.
/*
/* Copyright (C) 2009 Richard Haines
/*
/* This program is free software: you can redistribute it and/or modify
/* it under the terms of the GNU General Public License as published by
/* the Free Software Foundation, either version 3 of the License, or
/* (at your option) any later version.
/*
/* This program is distributed in the hope that it will be useful,
/* but WITHOUT ANY WARRANTY; without even the implied warranty of
/* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
/* GNU General Public License for more details.
/*
/* You should have received a copy of the GNU General Public License
/* along with this program. If not, see <http://www.gnu.org/licenses/>.
/*
/* *****
/* This client connects to the server that builds a buffer containing
/* information on ports and contexts used by the server, returning this
/* to the client.
/*
/* The client is compiled as follows:
/* gcc client.c -o client -lselinux
/* (This is labeled system_u:object:unconfined_t)
/* gcc client.c -o secure_client -lselinux
/* (This is labeled system_u:object:secure_services_exec_t)
/*
/* For the tests, the binaries should be installed in /usr/local/bin and
/* then the restorecon -f restorefiles_gateway run once the
/* external_gateway loadable module has been installed.
/* *****
/* For SECMARK, NetLabel and Message Filter demos the clients are called
/* as follows:
/* ./client <port> - Where the port is any you like (e.g. 1234)
/* ./secure_client <port> - Where for the demo thes ports are 1111
/* and 9999 as the iptables mangle table has been configured for these.
/*
/* *****
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <selinux/selinux.h>

#define MAXBUFFERSIZE 256
#define ENFORCING 1

#define ESC 0x1B

char red [] = "0;31";
char green [] = "0;32";
```

```
char reset [] = "0";

int main(int argc, char *argv [])
{
    int rc, sock_fd, bytes_received;
    char buffer [MAXBUFFERSIZE];
    struct hostent *server_info;
    struct sockaddr_in server_addr;
    short client_port;
    security_context_t context, peer_context;
    char *peer_context_str;

    if (argc != 3) {
        fprintf (stderr, "usage: %s <hostname> <port> (Use port 9999 for secure
port test)\n", argv[0]);
        exit (1);
    }

    client_port = atoi (argv [2]);

    if ((server_info = gethostbyname (argv [1])) == NULL) {
        perror ("Client gethostbyname");
        exit (1);
    }

    if (rc = security_getenforce () != ENFORCING)
        printf ("Should be in enforcing mode for valid testing\n");

    if ((sock_fd = socket (PF_INET, SOCK_STREAM, 0)) == -1) {
        perror ("Client Socket");
        exit (1);
    }

    bzero((char *) &server_addr, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(client_port);
    server_addr.sin_addr = *((struct in_addr *)server_info->h_addr);

    if (connect (sock_fd, (struct sockaddr *)&server_addr, sizeof(struct
sockaddr)) == -1) {
        perror ("Client connect");
        exit (1);
    }

    // clear the buffer
    memset (buffer, 0, sizeof(buffer));
    if ((bytes_received = recv (sock_fd, buffer, MAXBUFFERSIZE-1, 0)) == -1) {
        perror ("Client recv");
        exit (1);
    }

    buffer [bytes_received] = '\0'; // Add null at end of line.
    printf ("\033[%smServer Information in RED:\n", red);
    printf ("%s \n", buffer);

    // Print the Clients context information
    if (rc = getcon (&context) < 0) {
        perror ("Client context");
        exit (1);
    }

    if (rc = getpeercon (sock_fd, &peer_context) < 0)
        peer_context_str = strdup ("No Peer Context Available");
    else {
        peer_context_str = strdup (peer_context);
        freecon (peer_context);
    }
    printf ("\033[%smClient Information in GREEN:\n", green);
    printf ("Client Context: %s \nClient Peer Context: %s \n", context,
peer_context_str);

    freecon (context);
    close (sock_fd);
    printf ("\033[%sm\n", reset);
    return 0;
}
```

```
}
```

### 9. Compile two versions of the client by running:

```
gcc -o secure_client client.c -lselinux
gcc -o client client.c -lselinux
```

### 10. Produce a 'C' application called `server.c` with the contents shown below:

```
/*
/*
/* *****
/* This is the server component for the Notebook demo modular policy. It
/* will be used to demonstrate SECMARK and NetLabel network functionality
/* and also creates and reads files for the Message Filter example.
/*
/*
/* Copyright (C) 2009 Richard Haines
/*
/*
/* This program is free software: you can redistribute it and/or modify
/* it under the terms of the GNU General Public License as published by
/* the Free Software Foundation, either version 3 of the License, or
/* (at your option) any later version.
/*
/*
/* This program is distributed in the hope that it will be useful,
/* but WITHOUT ANY WARRANTY; without even the implied warranty of
/* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
/* GNU General Public License for more details.
/*
/*
/* You should have received a copy of the GNU General Public License
/* along with this program. If not, see <http://www.gnu.org/licenses/>.
/*
/*
/* *****
/*
/* The server is compiled as follows:
/* gcc server.c -o server -lselinux
/* (This is labeled system_u:object:unconfined_t)
/* gcc server.c -o secure_server -lselinux
/* (This is labeled system_u:object:secure_services_exec_t)
/*
/*
/* For the tests, the binaries should be installed in /usr/local/bin and
/* then the restorecon -f restorefiles_gateway run once the
/* external_gateway loadable module has been installed.
/*
/*
/* *****
/* The server receives a connection from a client (but no data) and then
/* builds a buffer containing information on ports and contexts used,
/* returning this to the client (the default action).
/*
/*
/* *****
/* For the SECMARK and NetLabel demo the servers are called as follows:
/* server <port> - Where the port is any you like (e.g. 1234)
/* secure_server <port> - Where for the demo are ports 9999 and 1111
/* as the test iptables mangle table has been configured for these.
/* ports.
/*
/* *****
/* For the Message Filter demo the servers are called as follows:
/* -----
/* To queue messages to the Message Filter's IN queue:
/* secure_server <port> in - Where the port is 1111
/* And then run the secure_client in another terminal session:
/* secure_client 127.0.0.1 9999
/* Note - This is using the external_gateway module for controlling
/* network access and the move_file module for controlling file access.
/*
/*
/* The [in] command line option writes the buffer to a file named
/* Message-<message_number> in the in_path directory. If the server is
/* restarted, then the <message number> just starts from 1 again
/* These files will be removed by the Message Filter move_file
/* application and moved to the out_path directory.
/* -----
/* To read messages from the Message Filter's OUT queue (after they have
/* been move by the move_file application):
/*
```

```
/*      runcon -t int_gateway_t -r message_filter_r secure_server 9999      */
/*
/* And then run the secure_client in another terminal session:
/*      runcon -t int_gateway_t -r message_filter_r secure_client \
/*                                     27.0.0.1 9999
/*
/*
/** Note - This is using the internal_gateway module for controlling
/* network access and the move_file module for controlling file access.
/*
/*
/* The [out] command line option reads files from the out_path directory
/* and sends them to the client. The file is then unlinked.
/*
/*
/*****
/*
/* NOTE: unconfined_t is not allowed to read/write files in the [in] or
/* [out] directories, if it tries, the context is displayed and the
/* server will exit (e.g 'server 1234 in' and 'client 127.0.0.1 1234').
/*
/*
/* Note when tested, the fopen function call on the [in] queue processing
/* caused a segmentation fault (a feature ??). The only way found to stop
/* this was to add an 'opendir' function call to the code, the server can
/* then exit gracefully displaying the context.
/*
/*
/*****

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/param.h>
#include <sys/types.h>
#include <sys/param.h>
#include <dirent.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <selinux/selinux.h>

#define MAXBUFFERSIZE 256

// variable to store current path
char in_path[] = "/usr/message_queue/in_queue";
char out_path[] = "/usr/message_queue/out_queue";

int main(int argc, char *argv [])
{
    short    server_port;
    int count, i, rc, sock_fd, new_sock_fd, message_number, option;
    struct sockaddr_in server_addr;
    struct sockaddr_in client_addr;
    socklen_t  sin_size;
    char      buffer [MAXBUFFERSIZE];
    security_context_t context, peer_context, dir_context;
    char      *peer_context_str;
    FILE      *fp;
    char      file_name [MAXPATHLEN];
    char      in[] = "in";
    char      out[] = "out";
    DIR *dp;
    struct dirent *ep;

    if (argc < 2) {
        fprintf (stderr, "Usage: %s <port>\n", argv[0]);
        exit (1);
    }

    if ((server_port = atoi (argv [1])) == 0) {
        fprintf (stderr, "Usage: %s <port>\n", argv[0]);
        exit (1);
    }

    option = 0; // Set to default (i.e. no in or out queue parameter)

    // Display default message about port, but alter if other options
    selected.
    sprintf (buffer, "Listening on port %d", server_port);
```

```
if (argc == 3) {
    if (strcmp(argv [2], in) == 0) {
        option = 1;
        sprintf (buffer, "Listening on port %d. Information sent to client
will be written to files in %s", server_port, in_path);
    }
    else if (strcmp(argv [2], out) == 0) {
        option = 2;
        sprintf (buffer, "Listening on port %d. Files will be read from %s
and the contents sent to the client", server_port, out_path);
    }
    else {
        fprintf (stderr, "Usage: %s <port> [in | out]\n", argv[0]);
        exit (1);
    }
}

printf ("%s\n", buffer);

if ((sock_fd = socket(PF_INET, SOCK_STREAM, 0)) == -1) {
    perror ("Server socket");
    exit (1);
}

// Set the message number to 1 so a "Message[message_number] is generated.
message_number = 1;

bzero((char *) &server_addr, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(server_port);
server_addr.sin_addr.s_addr = INADDR_ANY;

if (bind (sock_fd, (struct sockaddr *)&server_addr, sizeof(struct
sockaddr)) == -1) {
    perror ("Server bind");
    exit (1);
}

if (listen (sock_fd, 5) == -1) {
    perror ("Server listen");
    exit (1);
}

while (1) {
    sin_size = sizeof(struct sockaddr_in);
    if ((new_sock_fd = accept (sock_fd, (struct sockaddr *)&client_addr,
&sin_size)) == -1) {
        perror ("Server accept");
        continue;
    }

    // Get Server context information
    if (rc = getcon (&context) < 0) {
        perror ("Server context");
        exit (1);
    }

    if (rc = getpeercon (new_sock_fd, &peer_context) < 0)
        peer_context_str = strdup ("No Peer Context Available");
    else {
        peer_context_str = strdup (peer_context);
        freecon (peer_context);
    }

    // Clear the buffer of rubbish
    memset (buffer, 0, sizeof(buffer));

    switch (option) {
    case 1:
        // This option sends the buffer to client,
        // and then writes it to a file in the in_que
        // Make up a file name
        sprintf (file_name, "Message-%d", message_number);

        // Build buffer with Message Number at start.
```

```
    // The Message number will also be the file name
    sprintf (buffer, "This is %s from the server listening on port: %d
\nClient source port: %d \nServer Context: %s \nServer Peer Context: %s \n",
file_name, ntohs (server_addr.sin_port), ntohs (client_addr.sin_port),
context, peer_context_str);

    if (send (new_sock_fd, buffer, strlen (buffer), 0) == -1)
        perror ("Server send");
    // Now write buffer to file as well

    // This opendir has been put here as get Segmentation
    // fault if just do the fopen and its
    // unconfined_t trying to write a file here
    if ((dp = opendir (in_path)) == 0) {
    // Could be that unconfined_t is trying this, if so exit showing
context:
        getcon (&dir_context);
        printf ("Open Directory error %s Context is: %s\n", in_path,
dir_context);
        exit (1);
    }
    closedir (dp);

    // Make up full path + file name
    sprintf (file_name, "%s/Message-%d", in_path, message_number);
    if ((fp = fopen (file_name, "w")) == 0) {
        perror (fp);
        exit (0);
    }

    count = strlen (buffer);
    if (fwrite (buffer, count, 1, fp) != 1) {
        perror (fp);
        exit (0);
    }
    fclose (fp);
    break;

case 2:
    // This option will read a file from the out_queue,
    // send it to the client and then delete it.
    if ((dp = opendir (out_path)) == 0) {
    // Could be that unconfined_t is trying this on insecure
    // channel ?? If so then exit showing context:
        getcon (&dir_context);
        printf ("Open Directory error %s Context is: %s\n", out_path,
dir_context);
        exit (1);
    }

    do {
        if ((ep = readdir (dp)) == 0) {
            sprintf (buffer, "Server has no files to send\n");
            break;
        }
    } while ((strcmp(ep->d_name, ".") == 0) || (strcmp(ep->d_name,
"..") == 0));

    if (ep != 0) { // There is a file if ep != 0, otherwise send note
to client saying no more files
        sprintf (file_name, "%s/%s", out_path, ep->d_name);

        if ((fp = fopen (file_name, "r")) == 0) {
            perror (fp);
            exit (0);
        }

        // Read Contents of File
        if (fread (buffer, sizeof (buffer), 1, fp) != 0) {
            perror (fp);
            exit (0);
        }
        unlink (file_name);
        fclose (fp);
        closedir (dp);
    }
```

```
    }

    // Now send the buffer to client
    count = strlen (buffer);
    if (send (new_sock_fd, buffer, count, 0) == -1)
        perror ("Server send");
    break;

    default: // There is no in_que or out_que parameter so just send the
buffer.
        // Make up a Message name
        sprintf (file_name, "Message-%d", message_number);

        // Print Server network information
        printf ("Server has connection from client: host = %s destination
port = %d source port = %d\n",
            inet_ntoa(client_addr.sin_addr), ntohs (server_addr.sin_port),
            ntohs (client_addr.sin_port));

        printf ("Server Context: %s \nServer Peer Context: %s \n",
            context, peer_context_str);

        sprintf (buffer, "This is %s from the server listening on port: %d
\nClient source port: %d \nServer Context: %s \nServer Peer Context: %s \n",
            file_name, ntohs (server_addr.sin_port), ntohs (client_addr.sin_port),
            context, peer_context_str);

        if (send (new_sock_fd, buffer, strlen (buffer), 0) == -1)
            perror ("Server send");
    }
    message_number++;
    freecon (context);
    close (new_sock_fd);
}
return 0;
}
```

### 11. Compile two versions of the server by running:

```
gcc -o secure_server server.c -lselinux
gcc -o server server.c -lselinux
```

### 12. Move the binaries to /usr/local/bin:

```
cp client /usr/local/bin
cp secure_client /usr/local/bin
cp server /usr/local/bin
cp secure_server /usr/local/bin
```

### 13. Produce a script called iptables\_secmark with the contents shown below. This will be used to load the iptables (notes: 1. that if the current mangle table has other entries, then they will be lost as this script flushes the table before loading the new contents. 2. The entries for the internal gateway are commented out. This is because the module has not been built yet and leaving these in would produce an error when loading the table with SELinux in enforcing mode).

```
# Flush the mangle table first:
iptables -t mangle -F

#----- INPUT IP Stream -----#

# This INPUT rule sets all packets to default_secmark_packet_t: as it is
# executed first:
```

```
iptables -t mangle -A INPUT -i lo -p tcp -d 127.0.0.0/8 -j SECMARK
--selctx system_u:object_r:default_secmark_packet_t

# These rules that will replace the above context with the internal or
# external gateway if port 9999 or 1111 is found in either the source or
# destination port of the packet:
iptables -t mangle -A INPUT -i lo -p tcp --dport 9999 -j SECMARK --selctx
system_u:object_r:ext_gateway_packet_t
iptables -t mangle -A INPUT -i lo -p tcp --sport 9999 -j SECMARK --selctx
system_u:object_r:ext_gateway_packet_t
#
# These are not required until using the internal gateway:
#iptables -t mangle -A INPUT -i lo -p tcp --dport 1111 -j SECMARK --selctx
system_u:object_r:int_gateway_packet_t
#iptables -t mangle -A INPUT -i lo -p tcp --sport 1111 -j SECMARK --selctx
system_u:object_r:int_gateway_packet_t

iptables -t mangle -A INPUT -m state --state ESTABLISHED,RELATED -j
CONNSECMARK --save

#----- OUTPUT IP Stream -----#

# This OUTPUT rule sets all packets to default_secmark_packet_t: as it is
# executed first:
iptables -t mangle -A OUTPUT -o lo -p tcp -d 127.0.0.0/8 -j SECMARK
--selctx system_u:object_r:default_secmark_packet_t

# These rules that will replace the above context with the internal or
# external gateway if port 9999 or 1111 is found in either the source or
# destination port of the packet:

iptables -t mangle -A OUTPUT -o lo -p tcp --dport 9999 -j SECMARK --selctx
system_u:object_r:ext_gateway_packet_t
iptables -t mangle -A OUTPUT -o lo -p tcp --sport 9999 -j SECMARK --selctx
system_u:object_r:ext_gateway_packet_t
#
# These are not required until using the internal gateway:
#iptables -t mangle -A OUTPUT -o lo -p tcp --dport 1111 -j SECMARK
--selctx system_u:object_r:int_gateway_packet_t
#iptables -t mangle -A OUTPUT -o lo -p tcp --sport 1111 -j SECMARK
--selctx system_u:object_r:int_gateway_packet_t

iptables -t mangle -A OUTPUT -m state --state ESTABLISHED,RELATED -j
CONNSECMARK --save

iptables -t mangle -L
```

14. Produce a `restorefiles_gateway` file with the contents shown below. This will be used by the `restorecon` command to relabel the SECMARK test client / server executables after compilation and if any updates are done later.

```
/usr/local/bin/secure_client
/usr/local/bin/secure_server
/usr/local/bin/client
/usr/local/bin/server
```

15. Run the `restorecon(8)` command to relabel the secure versions of the client / server as follows:

```
restorecon -f restorefiles_gateway
```

16. Check that the secure versions of the client / server are labeled correctly using `ls -Z /usr/local/bin`.

```
user_u:object_r:unconfined_t          client
user_u:object_r:secure_services_exec_t  secure_client
user_u:object_r:secure_services_exec_t  secure_server
user_u:object_r:unconfined_t          server
```

17. Add the `message_filter_r` role by running `semanage` as follows:

```
semanage user -m -R "message_filter_r unconfined_r" user_u
```

Note: Need to add the `unconfined_r` role as `semanage` will remove it from the current policy otherwise, causing much grief (a bug or feature ??). See the [Using sediffx](#) section for more information.

The installation process is now complete, the testing is discussed in the next section.

### 3.2.1 Testing the Module

To test the SECMARK functionality it is recommended that three virtual terminal sessions are opened (as shown in [Figure 3.3](#)) for:

1. Running clients as they will display status messages if successful.
2. Running the servers as they display messages when connections are made with the clients.
3. Viewing the audit log file. Note that the module has `auditallow` rules on `packet { send recv }` so that these events can be seen.

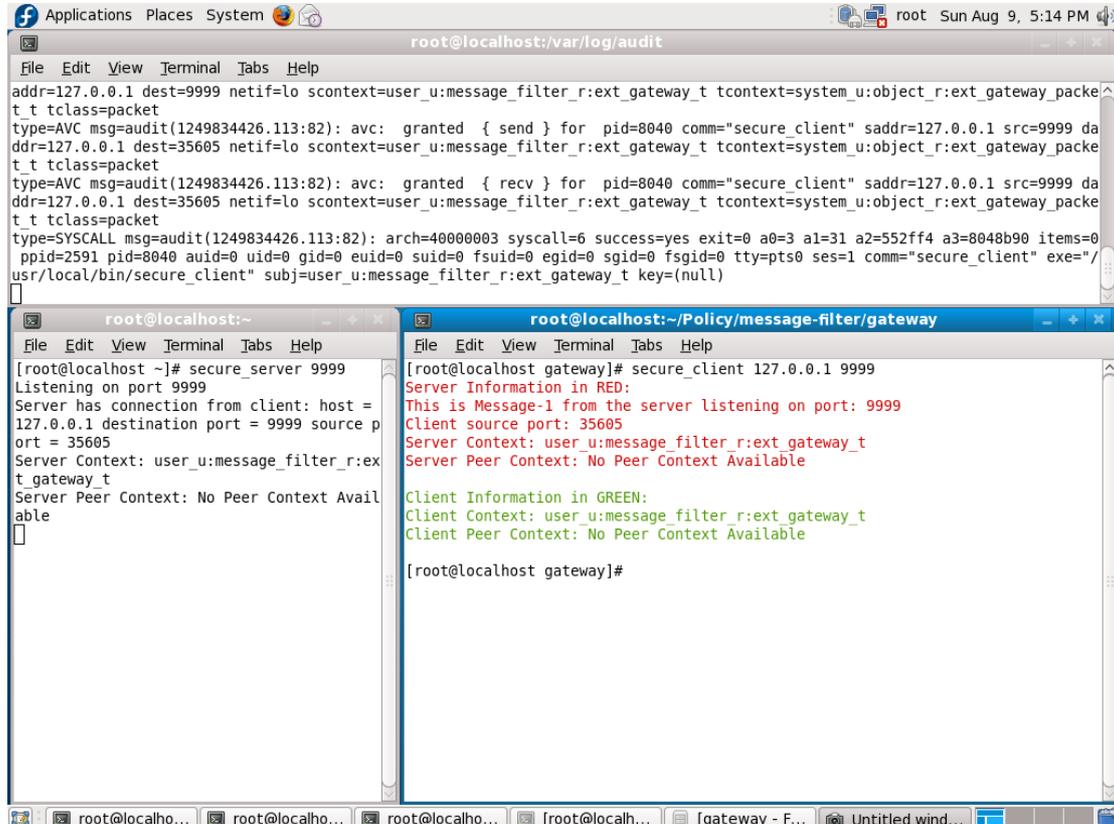


Figure 3.3: Testing using three virtual terminal sessions

### 3.2.1.1 Running the Tests

It is assumed that there are three terminal sessions logged in as root as shown in [Figure 3.3](#)), with the client and server windows both at the directory with the executable secmark code and scripts, and the third window for tailing the `audit.log` file.

Before starting the tests:

1. In the window that will display the audit log, execute the following command:

```
tail -f /var/log/audit/audit.log.
```

2. In a window run the following command to load the iptables:

```
./iptables_secmark
```

Note that it is important to load the iptables as explained in the [Importance of Loading the iptables](#) section below.

3. In a window run the following command to start enforcing policy:

```
setenforce 1
```

Note that the server must be started before the client. To exit any of the server sessions press `ctrl/c`.

**Test 1** – Running secure server and secure client sessions on port 9999 using the loopback interface (127.0.0.1):

1. In a window run the following command to start the secure server:

```
secure_server 9999
```

2. In a window run the following command to start the secure client:

```
secure_client 127.0.0.1 9999
```

The `audit.log` should contain only granted events on transition, send and recv (note that the transition also transitioned the role to `message_filter_r`):

```
type=AVC msg=audit(1249742538.972:23): avc: granted { transition } for
pid=2905 comm="bash" path="/usr/local/bin/secure_server" dev=dm-0 ino=355514
scontext=user_u:unconfined_r:unconfined_t
tcontext=user_u:message_filter_r:ext_gateway_t tclass=process
type=SYSCALL msg=audit(1249742538.972:23): arch=40000003 syscall=11 success=yes
exit=0 a0=858a678 a1=85932c0 a2=858c8e8 a3=0 items=0 ppid=2520 pid=2905 auid=0
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts1 ses=1
comm="secure_server" exe="/usr/local/bin/secure_server"
subj=user_u:message_filter_r:ext_gateway_t key=(null)

type=AVC msg=audit(1249742544.827:24): avc: granted { transition } for
pid=2907 comm="bash" path="/usr/local/bin/secure_client" dev=dm-0 ino=354307
scontext=user_u:unconfined_r:unconfined_t
tcontext=user_u:message_filter_r:ext_gateway_t tclass=process
type=SYSCALL msg=audit(1249742544.827:24): arch=40000003 syscall=11 success=yes
exit=0 a0=87f92d8 a1=87e9ca8 a2=87ee8e8 a3=0 items=0 ppid=2496 pid=2907 auid=0
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=1
comm="secure_client" exe="/usr/local/bin/secure_client"
subj=user_u:message_filter_r:ext_gateway_t key=(null)
```

```
type=AVC msg=audit(1249742544.833:25): avc: granted { send } for pid=2907
comm="secure_client" saddr=127.0.0.1 src=43592 daddr=127.0.0.1 dest=9999
netif=lo scontext=user_u:message_filter_r:ext_gateway_t
tcontext=system_u:object_r:ext_gateway_packet_t tclass=packet

type=AVC msg=audit(1249742544.833:25): avc: granted { recv } for pid=2907
comm="secure_client" saddr=127.0.0.1 src=43592 daddr=127.0.0.1 dest=9999
netif=lo scontext=user_u:message_filter_r:ext_gateway_t
tcontext=system_u:object_r:ext_gateway_packet_t tclass=packet
....
....
type=AVC msg=audit(1249742544.834:26): avc: granted { send } for pid=2905
comm="secure_server" saddr=127.0.0.1 src=9999 daddr=127.0.0.1 dest=43592
netif=lo scontext=user_u:message_filter_r:ext_gateway_t
tcontext=system_u:object_r:ext_gateway_packet_t tclass=packet

type=AVC msg=audit(1249742544.834:26): avc: granted { recv } for pid=2905
comm="secure_server" saddr=127.0.0.1 src=9999 daddr=127.0.0.1 dest=43592
netif=lo scontext=user_u:message_filter_r:ext_gateway_t
tcontext=system_u:object_r:ext_gateway_packet_t tclass=packet
```

**Test 2** – Running the server on port 9999 and the secure client on port 1234 using the loopback interface:

1. In a window run the following command to start the server:

```
server 9999
```

2. In a window run the following command to start the secure client:

```
secure_client 127.0.0.1 1234

Note: ctrl/c will exit the session
```

There should be an AVC audit message where the secure client is granted the transition but denied the send:

```
# Note that the client is allowed to transition:
type=AVC msg=audit(1249742696.572:30): avc: granted { transition } for
pid=2944 comm="bash" path="/usr/local/bin/secure_client" dev=dm-0 ino=354307
scontext=user_u:unconfined_r:unconfined_t
tcontext=user_u:message_filter_r:ext_gateway_t tclass=process
type=SYSCALL msg=audit(1249742696.572:30): arch=40000003 syscall=11 success=yes
exit=0 a0=87f92d8 a1=87f5300 a2=87ee8e8 a3=0 items=0 ppid=2496 pid=2944 auid=0
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=1
comm="secure_client" exe="/usr/local/bin/secure_client"
subj=user_u:message_filter_r:ext_gateway_t key=(null)

# But is not allowed to send message to the server as the packet
# is marked default_secmark_packet_t:
type=AVC msg=audit(1249742696.579:31): avc: denied { send } for pid=2944
comm="secure_client" saddr=127.0.0.1 src=42942 daddr=127.0.0.1 dest=1234
netif=lo scontext=user_u:message_filter_r:ext_gateway_t
tcontext=system_u:object_r:default_secmark_packet_t tclass=packet
```

**Test 3** – Running both client and server sessions using port 1234 on the loopback interface (127.0.0.1):

1. In a window run the following command to start the server:

```
server 1234
```

2. In a window run the following command to start the client:

```
client 127.0.0.1 1234
```

The `audit.log` should contain only granted events on `send` and `recv` (note that there is NO transition and the role remains as `unconfined_r`):

```
type=AVC msg=audit(1249742778.361:34): avc: granted { send } for pid=2964
comm="client" saddr=127.0.0.1 src=42943 daddr=127.0.0.1 dest=1234 netif=lo
scontext=user_u:unconfined_r:unconfined_t
tcontext=system_u:object_r:default_secmark_packet_t tclass=packet

type=AVC msg=audit(1249742778.361:34): avc: granted { recv } for pid=2964
comm="client" saddr=127.0.0.1 src=42943 daddr=127.0.0.1 dest=1234 netif=lo
scontext=user_u:unconfined_r:unconfined_t
tcontext=system_u:object_r:default_secmark_packet_t tclass=packet
....
....
type=AVC msg=audit(1249742778.362:35): avc: granted { send } for pid=2961
comm="server" saddr=127.0.0.1 src=1234 daddr=127.0.0.1 dest=42943 netif=lo
scontext=user_u:unconfined_r:unconfined_t
tcontext=system_u:object_r:default_secmark_packet_t tclass=packet

type=AVC msg=audit(1249742778.362:35): avc: granted { recv } for pid=2961
comm="server" saddr=127.0.0.1 src=1234 daddr=127.0.0.1 dest=42943 netif=lo
scontext=user_u:unconfined_r:unconfined_t
tcontext=system_u:object_r:default_secmark_packet_t tclass=packet
```

**Test 4** – Running the server on port 9999 and the secure client on port 9999 using the loopback interface:

3. In a window run the following command to start the server:

```
server 9999
```

4. In a window run the following command to start the secure client:

```
secure_client 127.0.0.1 9999
```

```
Note: ctrl/c will exit the session
```

The AVC audit messages show that the secure client has been granted the transition and `send` but denied the `recv` from the standard server (but note that the server was allowed to accept the connection):

```
type=AVC msg=audit(1249742873.035:38): avc: granted { transition } for
pid=2987 comm="bash" path="/usr/local/bin/secure_client" dev=dm-0 ino=354307
scontext=user_u:unconfined_r:unconfined_t
tcontext=user_u:message_filter_r:ext_gateway_t tclass=process
type=SYSCALL msg=audit(1249742873.035:38): arch=40000003 syscall=11 success=yes
exit=0 a0=8801cf0 a1=87e9ca8 a2=87ee8e8 a3=0 items=0 ppid=2496 pid=2987 auid=0
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=1
comm="secure_client" exe="/usr/local/bin/secure_client"
subj=user_u:message_filter_r:ext_gateway_t key=(null)

type=AVC msg=audit(1249742873.041:39): avc: granted { send } for pid=2987
comm="secure_client" saddr=127.0.0.1 src=35900 daddr=127.0.0.1 dest=9999
netif=lo scontext=user_u:message_filter_r:ext_gateway_t
tcontext=system_u:object_r:ext_gateway_packet_t tclass=packet

type=AVC msg=audit(1249742873.041:39): avc: denied { recv } for pid=2987
comm="secure_client" saddr=127.0.0.1 src=35900 daddr=127.0.0.1 dest=9999
netif=lo scontext=user_u:unconfined_r:unconfined_t
tcontext=system_u:object_r:ext_gateway_packet_t tclass=packet
```

The reader can experiment with the remaining scenarios to find if there are any holes in the configuration.

### 3.2.2 Points to Note

#### 3.2.2.1 Importance of Loading the `iptables`

The external gateway policy module relies on the fact that the `iptables` are loaded correctly to label the network packets. If they are not loaded, or (for example) the command:

```
iptables -t mangle -F
```

was allowed to be run that removes the mangle table entries, then the network packets would be labeled with the initial SID default of `unconfined_t`. The result is of course that all packets would be allowed. For example, running the `secure_client` and standard server on port 9999 with no `iptables` loaded would have the following `audit.log` entries (as all traffic on all ports would flow, as no policy is being enforced):

```
type=AVC msg=audit(1247241956.542:32): avc: granted { transition } for
pid=2876 comm="bash" path="/usr/local/bin/secure_client" dev=dm-0 ino=354307
scontext=user_u:unconfined_r:unconfined_t
tcontext=user_u:message_filter_r:ext_gateway_t tclass=process
type=SYSCALL msg=audit(1247241956.542:32): arch=40000003 syscall=11 success=yes
exit=0 a0=9474a68 a1=947f460 a2=946d8e8 a3=0 items=0 ppid=2634 pid=2876 auid=0
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=1
comm="secure_client" exe="/usr/local/bin/secure_client"
subj=user_u:message_filter_r:ext_gateway_t key=(null)
```

Compare this `audit.log` trail with those shown in [Test 4](#) that was run using the same scenario except that the `iptables` were loaded, thus denying the `recv`.

#### 3.2.2.2 Running tests out of sequence

The server component allows files to be created in an ‘in queue’, and read / unlinked for the ‘out queue’ when running the message filter test. However should the [message filter](#) tests be run (see the [Testing the Message Filter Build](#) section) before the internal gateway and file mover loadable modules are loaded, the following will be noted:

1. When running the unconfined client / server, files can be written (server 1234 in with client 127.0.0.1 1234), moved (`move_file`) and then read / unlinked (server 1234 out with client 127.0.0.1 1234). This is because the base policy allows `unconfined_t` to do anything it likes.
2. When running the secure client / server, files cannot be written to the ‘in queue’ or read / unlinked from the ‘out queue’. This is because the `ext_gateway_t` process does not have the required allow permissions to do this.

### 3.3 Building the NetLabel Loadable Module

This simple module enables a NetLabel `netlabel_peer_t` label to be added to the network connection to show that additional information at the peer level (as `secmark` handles packet level labeling) can be added.

Because standard F-12 has the Policy Capabilities<sup>3</sup> `network_peer_controls` set to '0', the full peer controls are not enabled, however the legacy implementation will use the `tcp_socket` object class with the `recvfrom` permission to manage peer labeling for this example.

For an example where the `network_peer_controls` is set to '1', allowing the use of the new controls see [Appendix B – NetLabel Module Support for network\\_peer\\_controls](#).

The following steps need to be followed to build the test services (it is assumed that the files are built in the `./notebook-source/message-filter/netlabel` directory):

1. Ensure you are logged on as 'root' and SELinux is running in permissive mode (`setenforce 0`) to perform the build process.
2. Download and install the NetLabel rpm as this is not included in the FC-12 build:

```
yum install netlabel_tools
# yum will then install netlabel_tools
```

3. Produce a `netlabel.conf` loadable module file with a text editor (such as `vi` or `gedit`) containing the contents shown below:

```
module netlabel 1.0.0;
#
#####
#
# This Loadable Module will allow the netlabels to be added and checked #
# within the client / server applications that form part of the SECMARK #
# test examples. #
# #
# The following needs to happen to enable Netlabel to work as it is not #
# installed by default in F-12: #
# #
# (1) Download and install netlabel_tools rpm (or equiv) #
# #
# (2) Install this loadable module. #
# #
# (3) Run the following netlabelctl command: #
#     netlabelctl unlbl add interface:lo address:127.0.0.1 \ #
#         label:system_u:object_r:netlabel_peer_t #
# #
# (4) Run netlabelctl -p unlbl list command to check all is okay. #
# #
# (5) Run the secure and standard client/server that should now display #
#     the netlabel_peer_t as the peer context. #
# #
# Important note: F-12 does not support the latest netlabel services in #
# the kernel as: #
#     /selinux/policy_capabilities/network_peer_controls = 0 #
# #
# The optional section is used when the internal gateway module is #
# loaded. #
#####
```

---

<sup>3</sup> See the SELinux Filesystem section in 'The Foundations' volume.

```
#
require {
    type ext_gateway_t, unconfined_t;
    class tcp_socket { recvfrom };
}
type netlabel_peer_t;
type socket_t;

# These are used when /selinux/policy_capabilities/network_peer_controls =
0
# which is the default for F-12
allow ext_gateway_t netlabel_peer_t : tcp_socket recvfrom;
allow unconfined_t netlabel_peer_t : tcp_socket recvfrom;

#
##### START OPTIONAL SECTION #####
#
optional {
    require {
        # This is defined in the int_gateway.conf module:
        type int_gateway_t;
    }
    allow int_gateway_t netlabel_peer_t : tcp_socket recvfrom;
}
#
##### END OPTIONAL SECTION #####
#
```

4. Compile and install the module as follows:

```
checkmodule -m netlabel.conf -o netlabel.mod
semodule_package -o netlabel.pp -m netlabel.mod
semodule -v -s modular-test -i netlabel.pp
```

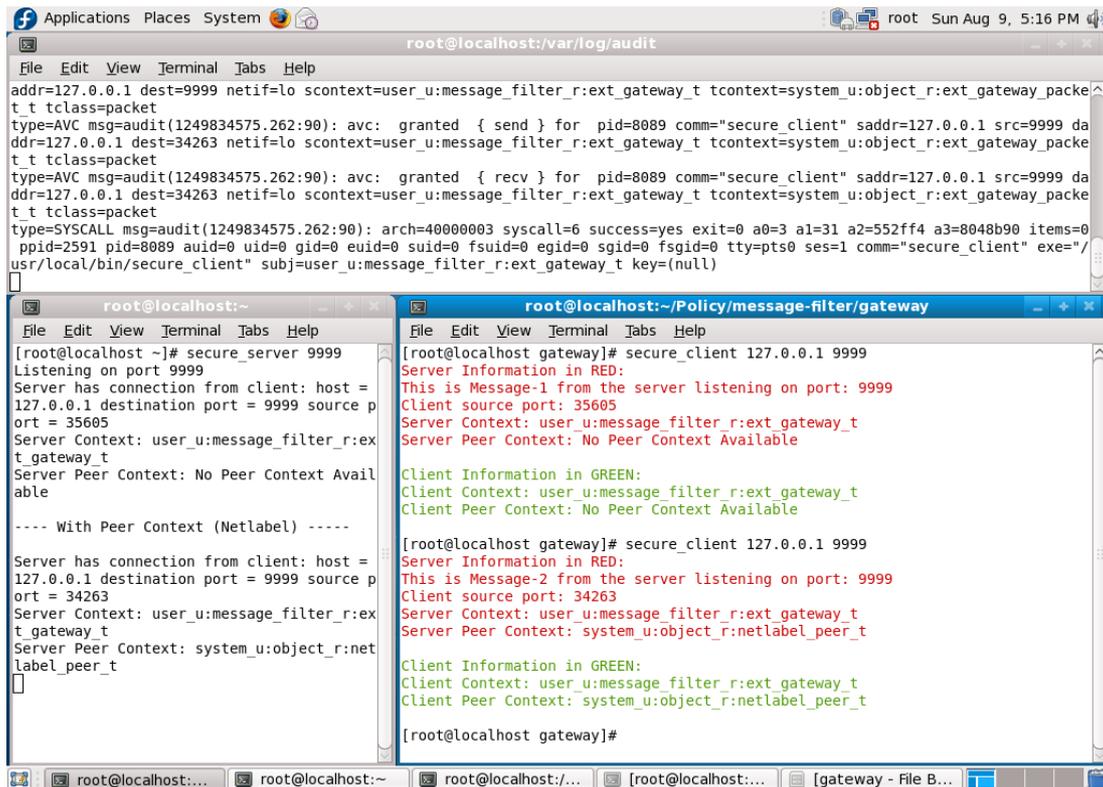
5. Run the following command to add the `netlabel_peer_t` label as follows:

```
netlabelctl unlbl add interface:lo address:127.0.0.1 \
    label:system_u:object_r:netlabel_peer_t
```

6. Run enforcing mode:

```
setenforce 1
```

7. Run either the client / server or `secure_client` / `secure_server` applications as shown in the [SECMARK tests](#). There should now be a peer context displayed as shown in [Figure 3.4](#).



**Figure 3.4: Running the secure client / server with NetLabel enabled**

To remove the NetFilter label, the following command can be run:

```
netlabelctl unlbl del interface:lo address:127.0.0.1 \
label:system_u:object_r:netlabel_peer_t
```

## 3.4 Building the Remaining Message Filter Service

To complete the overall message filter shown in [Figure 3.1](#), the internal gateway and file mover applications and policy modules need to be built. These are explained in this section plus how to test the modules via simple helper scripts. The source and scripts are included in the source code rpm package.

The following will be built in this section:

1. The internal gateway policy module.
2. The file mover application.
3. The file mover policy module.

### 3.4.1 Internal Gateway Loadable Policy Module

This loadable module will apply policy rules for the internal gateway. The policy applies `dontaudit` rules for those permissions known not to cause problems.

## The SELinux Notebook - Sample Policy Source

The following steps need to be followed to build the internal gateway module. It is assumed that the services are installed in `./notebook-source/message-filter/gateway`:

1. Ensure you are logged on as 'root' and SELinux is running in permissive mode (`setenforce 0`) to perform the build process.
2. Produce a `int_gateway.conf` file with a text editor (such as `vi` or `gedit`) containing the contents shown below:

```
module int_gateway 1.1.0;

#####
#
# This Loadable Module will allow a simple Message Filter to be tested.
#
# The module is used with the base.conf that sets up the unconfined_t
# space, the external_gateway.conf that manages the incoming data, and
# the move_file.conf module that copies files from the in queue to the
# out queue as explained in the SELinux Notebook.
# This module can be built by:
#   checkmodule -m int_gateway.conf -o int_gateway.mod
#   semodule_package -o int_gateway.pp -m int_gateway.mod
#   semodule -v -s modular-test -i int_gateway.pp
#
# The secure port for this internal gateway is 1111 and can only be
# read/write by the secure client / server. The external gateway will
# use port 9999 and can only be read/write by the secure client / server.
# Any other port can be read / write by the standard client / server.
#
# The iptables_secmark script can be modified to other ports if required.
# WARNING - If the iptables are not loaded to label packets, ports etc.
# then the standard client / server can use the secure ports.
#
# Run setenforce 1, the policy can be tested using combinations of:
#   ./server <port>
#   ./secure_server <port>
#
#   ./client <host> <port>
#   ./secure_client <host> <port>
#
# The module transitions to a role of message_filter_r simply to show
# a role transition. To add the role to user_u the semanage command is
# used as follows:
#   semanage user -m -R "message_filter_r unconfined_r" user_u
# Note: Need to put in the unconfined_r role as semanage will remove it
# from the current policy otherwise, causing much grief.
#
# Note: To run the internal gateway the runcon command must be used.
# This is because the external gateway has a type_transition statement:
#   type_transition unconfined_t secure_services_exec_t :
#     process ext_gateway_t;
# AND the module linker will not allow two type_transition statements
# using the secure_services_exec_t target with a different process type
# i.e. There cannot be in the overall policy:
#   type_transition unconfined_t secure_services_exec_t :
#     process ext_gateway_t;
#   type_transition unconfined_t secure_services_exec_t :
#     process int_gateway_t;
#
# Therefore run this as follows:
# runcon -t int_gateway_t -r message_filter_r secure_server 9999
# runcon -t int_gateway_t -r message_filter_r secure_client \
#                                     127.0.0.1 9999
#####

require {
    type unconfined_t, secure_services_exec_t;
    role unconfined_r;
    attribute message_filter_domains;
    class packet { send recv relabelto };
}
```

```
class process { fork sigchld transition siginh rlimitinh noatsecure signal
};
class file { entrypoint read getattr execute relabelto unlink write create
};
class filesystem { getattr associate };
class chr_file { read write getattr };
class dir { read search getattr write add_name remove_name };
class fd use;
class lnk_file read;
class tcp_socket { write listen node_bind name_bind accept bind read
name_connect connect create getopt };
class association recvfrom;
class unix_stream_socket { create connect };
}

# The internal gateway will run in this domain:
type int_gateway_t;

# The internal gateway will have a SECMARK in the iptables of this label:
type int_gateway_packet_t;

# Add the gateway domain to the attribute:
typeattribute int_gateway_t message_filter_domains;

# Use message_filter_r role and role transition for the gateway:
role message_filter_r types int_gateway_t;
allow unconfined_r message_filter_r;
role_transition unconfined_r secure_services_exec_t message_filter_r;

# Allow unconfined_t to relabel the secure ports. This is needed so that
# iptables can be updated easily. Note: Against security policy, however
# these need to be loaded at boot time when the policy is in enforcing
# mode so no choice !!
allow unconfined_t int_gateway_packet_t : packet relabelto;

# Allow gateway access only to secure ports:
allow int_gateway_t int_gateway_packet_t : packet { send rcv };

# Allow the internal gateway to transition to int_gateway_t. Note that the
# type_transition statement is commented out and runcon is used to
# transition this gateway (see the "Type Enforcement Rules" section of
# the SELinux Notebook for gory details):
allow unconfined_t secure_services_exec_t : file { read execute getattr };
allow unconfined_t int_gateway_t : process { transition };
allow int_gateway_t secure_services_exec_t : file { entrypoint };
#type_transition unconfined_t secure_services_exec_t : process
int_gateway_t;

# Stop segmentation faults
allow int_gateway_t unconfined_t : filesystem associate;
allow unconfined_t int_gateway_t : process noatsecure;
dontaudit unconfined_t int_gateway_t : process { siginh rlimitinh };

# Need this in F-12 build to allow the client / server apps to exit:
allow unconfined_t int_gateway_t : process signal;

# Allow int_gateway_t access to areas under unconfined_t domain:
allow int_gateway_t unconfined_t : packet { rcv send };
allow int_gateway_t unconfined_t : chr_file { read write getattr };
allow int_gateway_t unconfined_t : dir search;
allow int_gateway_t unconfined_t : fd use;
allow int_gateway_t unconfined_t : filesystem getattr;
allow int_gateway_t unconfined_t : tcp_socket name_connect;
allow int_gateway_t unconfined_t : association recvfrom;
allow int_gateway_t self : dir search;
allow int_gateway_t self : tcp_socket { read create connect };

# Need this in F-12 build to allow the client / server apps to exit:
allow int_gateway_t unconfined_t : process sigchld;
# This was the F-10 statement:
# dontaudit int_gateway_t unconfined_t : process sigchld;

# For client and server to access the shared libc:
allow int_gateway_t unconfined_t : file { read getattr execute };
dontaudit int_gateway_t unconfined_t : dir { getattr };
```

```
allow int_gateway_t unconfined_t : lnk_file read;

# Required if use host name instead of the IP address (e.g. localhost
# instead of 127.0.0.1) in the client command line:
dontaudit int_gateway_t self : unix_stream_socket { create connect };

# Required to get context information when using the libselinux api calls
# getcon() and getpeercon():
allow int_gateway_t self : file read;
allow int_gateway_t self : tcp_socket getopt;

# These entries are for the server only:
allow int_gateway_t self : tcp_socket { listen write accept bind };
allow int_gateway_t unconfined_t : tcp_socket { name_bind node_bind };
```

3. Compile the policy with `checkmodule` to produce an intermediate binary policy file:

```
checkmodule -m int_gateway.conf -o int_gateway.mod
```

The output from the compilation should be:

```
checkmodule: loading policy configuration from base.conf
checkmodule: policy configuration loaded
checkmodule: writing binary representation (version 10) to base.mod
```

4. Package the policy with `semodule_package`, this will produce a policy module file (note – if successful there are no output messages):

```
semodule_package -o int_gateway.pp -m int_gateway.mod
```

5. Install the loadable module with `semodule` (note – if successful there are no output messages):

```
semodule -v -s modular-test -i int_gateway.pp
```

6. If there are no errors reported, then the loadable module has been added to the policy store and loaded as a part of the policy. The policy module can be checked by:

```
semodule -s modular-test -l
```

The results should be:

```
ext_gateway 1.0.0
int_gateway 1.0.0
netlabel    1.0.0
```

The file mover application will now be built.

### 3.4.2 File Move Application

This 'C' program will move files from one directory to another and works in conjunction with the `move_file.conf` loadable module that will apply the policy rules.

The following steps need to be followed to build the file move application and it is assumed that the services are installed in `./notebook-source/message-filter/move_file`:

1. With an editor produce the `move_file.c` program as follows:

```
/*
 *
 * This is the file mover component for the Notebook demo modular policy.
 * It moves the files created as a part of the SECMARK tests from the
 * /usr/message_queue/in_queue to the /usr/message_queue/out_queue.
 * The move_file.conf module ensures that the output queue files are
 * correctly labeled out_file_t by an object type_transition.
 *
 * Copyright (C) 2009 Richard Haines
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 *
 */
/*
 * The move_file program is compiled as:
 * gcc -o move_file move_file.c
 *
 * The move_file application can optionally be called with a timer value
 * (in seconds) so that it can be run forever checking the in_queue
 * every X seconds: move_file [timer]
 *
 * For the tests, the binary should be installed in /usr/local/bin and
 * then the mk-dir script run to create the following directories:
 * mkdir -p /usr/message_queue/in_queue
 * mkdir -p /usr/message_queue/out_queue
 *
 * Install the move_file loadable module by:
 * checkmodule -m move_file.conf -o move_file.mod
 * semodule_package -o move_file.pp -m move_file.mod -f move_file.fc
 * semodule -v -s modular-test -i move_file.pp
 *
 * Finally label the binary and message queue directories by:
 * restorecon -r -f restorecon_move_file
 *
 * The server.c file describes how the files are are created etc.
 */
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <dirent.h>
#include <sys/types.h>
#include <sys/dir.h>
#include <sys/param.h>

#define MAXBUFFERSIZE 256

#define FALSE 0
#define TRUE !FALSE

extern int alphasort();

// variable to store current path
char in_path[] = "/usr/message_queue/in_queue";
```

```
char out_path[] = "/usr/message_queue/out_queue";

main (int argc, char *argv [])
{
    FILE    *fp1;
    FILE    *fp2;
    char    buffer [MAXBUFFERSIZE];
    char    in_file_name [MAXPATHLEN];
    char    out_file_name [MAXPATHLEN];
    int timer, count, length, i;
    struct direct **files;
    int select_file();

    // Use arg as the timer to use
    if (argc == 2)
        timer = atoi (argv [1]);
    else
        timer = 0;

    while (TRUE) {
        count = 0;
        count = scandir (in_path, &files, select_file, alphasort);
        printf ("Count = %d Timer = %d\n", count, timer);
        if ((count <= 0 && timer == 0))
            break;
        else
            sleep (timer);

        for (i=1; i<count+1; ++i) {
            // Build file name and clear the buffer
            sprintf (in_file_name,"%s/%s", in_path, files [i-1]->d_name);
            memset (buffer, 0, sizeof (buffer));

            // Open INQ file
            if ((fp1 = fopen (in_file_name, "r")) == 0) {
                perror (fp1);
                exit (1);
            }

            /* Read Contents of File */
            if (fread (buffer, sizeof (buffer), 1, fp1) != 0) {
                perror (fp1);
                exit (1);
            }

            // Build output file name
            sprintf (out_file_name,"%s/%s", out_path, files [i-1]-
>d_name);

            // Open OUTQ file
            if ((fp2 = fopen (out_file_name, "w")) == 0) {
                perror (fp2);
                exit (1);
            }

            // Get buffer length
            strcat (buffer, "(FILE MOVED TO OUT QUEUE)");
            length = strlen (buffer);

            // Write Contents of File
            if (fwrite (buffer, length, 1, fp2) != 1) {
                perror (fp2);
                exit (1);
            }

            unlink (in_file_name);
            fclose (fp1);
            fclose (fp2);
        }
    }

    int select_file (struct direct *entry)
    {
```

```
    if ((strcmp (entry->d_name, ".") == 0) || (strcmp (entry->d_name, "..") == 0))
        return (FALSE);
    else
        return (TRUE);
}
```

2. Compile the `move_file.c` program:

```
gcc -o move_file move_file.c
```

3. Move the binary to `/usr/local/bin`:

```
mv move_file /usr/local/bin
```

To complete the message filter, the file mover loadable module will now be built.

### 3.4.3 File Mover Loadable Policy Module

This loadable module will allow a file to be moved from one directory to another using the file mover application built above with minimum privileges. The policy applies `dontaudit` rules for those permissions known not to cause problems.

Note that in the policy there is a statement that allows a counter to be displayed on the console for testing purposes.

The following steps need to be followed to build the file mover module and it is assumed that the services are installed in `./notebook-source/message-filter/move_file`:

1. Ensure you are logged on as 'root' and SELinux is running in permissive mode (`setenforce 0`) to perform the build process.
2. Produce a `move_file.conf` file with a text editor (such as `vi` or `gedit`) containing the contents shown below:

```
module move_file 1.1.0;

#####
#
# This Loadable Module will allow files to be moved from one directory #
# (or queue) to another using the move_file 'C' program with minimum #
# permissions. The policy applies dontaudit rules for those permissions #
# known not to cause problems. #
#
#####

require {
    role unconfined_r;
    type unconfined_t;
    attribute message_filter_domains;
    class file { entrypoint getattr execute create read write unlink };
    class dir { read search getattr write add_name remove_name };
    class process { transition siginh noatsecure sigchld rlimitinh signal };
    class fd use;
    class chr_file { read write getattr };
    class lnk_file read;
    class filesystem associate;
}

# Define type identifiers for the process / domain:
type move_file_t;
typeattribute move_file_t message_filter_domains;
```

```
# Define the executable type:
type move_file_exec_t;

# These are the file directory types:
type in_queue_t;
type out_queue_t;

# These are the file types:
type in_file_t;
type out_file_t;

# Use message_filter_r role and then allow role transition
role message_filter_r types { move_file_t };
allow unconfined_r message_filter_r;
role_transition unconfined_r move_file_exec_t message_filter_r;

# Need permission for the program to transition:
allow unconfined_t move_file_t : process transition;
auditallow unconfined_t move_file_t : process transition;
allow unconfined_t move_file_exec_t : file { read execute getattr };
allow move_file_t move_file_exec_t : file { entrypoint };
type_transition unconfined_t move_file_exec_t : process move_file_t;

#
# The move_file application reads then deletes the file:
type_transition move_file_t in_queue_t : file in_file_t;
allow move_file_t in_file_t : file { read unlink };
allow move_file_t in_queue_t : dir { read getattr search write remove_name
};
dontaudit move_file_t in_file_t : file getattr;

# Need these if the files are labeled user_u:object_r:in_queue_t
# This happens if restorecond is not running with setenforce 0 and use
# vi to create the files for testing (as they are not relabeled)
# allow move_file_t in_queue_t:file { read getattr unlink };

# The move_file application then writes the file to the out queue:
type_transition move_file_t out_queue_t : file out_file_t;
allow move_file_t out_file_t : file { create write };
allow move_file_t out_queue_t : dir { search write add_name };
dontaudit move_file_t out_file_t : file getattr;

# Do not need these:
# HOWEVER - The move_file application has a printf with:
#     printf ("Count = %d Timer = %d\n", count, timer);
# that can be seen on the console IF this allow is enabled:
allow move_file_t unconfined_t : chr_file { read write getattr };
# OR it can be disabled from printing this on the console by:
# dontaudit move_file_t unconfined_t : chr_file { read write getattr };

# Need this in F-12 build to allow the app to exit:
allow unconfined_t move_file_t : process signal;
allow move_file_t unconfined_t : process sigchld;
# This was the F-10 statement:
# dontaudit move_file_t unconfined_t : process sigchld;

# Need these as /usr/move_file dir is unconfined_t
allow move_file_t unconfined_t : dir search;
allow move_file_t unconfined_t : fd use;

# Need these to run libc.so shared library
dontaudit move_file_t unconfined_t : dir getattr;
allow move_file_t unconfined_t : lnk_file read;
allow move_file_t unconfined_t : file { read getattr execute };

# Need these to stop Segmentation faults
allow out_file_t unconfined_t : filesystem associate;
allow unconfined_t move_file_t : process noatsecure;
dontaudit unconfined_t move_file_t : process { siginh rlimitinh };

# Don't need these:
dontaudit unconfined_t in_queue_t : dir { read getattr search };
dontaudit unconfined_t out_file_t : file getattr;
```

```
dontaudit unconfined_t out_queue_t : dir { read getattr search };
dontaudit unconfined_t in_file_t : file getattr;
```

3. Produce a `move_file.fc` file (a segment that will be added to `file_contexts` file during the build) with the contents shown below. This will be used to relabel application files and directories.

```
# The Move File process makes use of two directory structures
# (in & out) that are labeled as follows:

/usr/message_queue/in_queue -d system_u:object_r:in_queue_t
/usr/message_queue/out_queue -d system_u:object_r:out_queue_t

# Ensure that any files are also relabeled:
/usr/message_queue/in_queue(/.*)? -- system_u:object_r:in_file_t
/usr/message_queue/out_queue(/.*)? -- system_u:object_r:out_file_t

# The Move File 'C' application is labeled:
/usr/local/bin/move_file -- system_u:object_r:move_file_exec_t
```

4. Produce a `restorecon_files` file with the contents shown below. This will be used by the `restorecon` command to relabel application files and directories after any updates.

```
/usr/message_queue/in_queue
/usr/message_queue/out_queue
/usr/local/bin/move_file
```

5. Compile the policy with `checkmodule` to produce an intermediate binary policy file:

```
checkmodule -m move_file.conf -o move_file.mod
```

The output from the compilation should be:

```
checkmodule: loading policy configuration from base.conf
checkmodule: policy configuration loaded
checkmodule: writing binary representation (version 8) to base.mod
```

6. Package the policy with `semodule_package`, this will produce a policy module file (note – if successful there are no output messages):

```
semodule_package -o move_file.pp -m move_file.mod -f move_file.fc
```

7. Make the directories required by the application. These need to be created because when `semodule` loads the policy, it will run `setfiles` to set the file contexts correctly (using the contents of the `move_file.fc` file produced in step 3).

```
mkdir -p /usr/message_queue/in_queue
mkdir -p /usr/message_queue/out_queue
```

8. Install the loadable module with `semodule` (note – if successful there are no output messages):

```
semodule -v -s modular-test -i move_file.pp
```

9. If there are no errors reported, then the loadable module has been added to the policy store and loaded as a part of the policy. The policy module can be checked by:

```
semodule -s modular-test -l
```

The results should be:

```
ext_gateway 1.0.0
int_gateway 1.0.0
move_file   1.0.0
netlabel    1.0.0
```

10. Uncomment the internal gateway entries in the iptables file (`./notebook-source/message-filter/gateways/iptables_secmark`) that was produced in step 13 of the [Building the SECMARK Test Loadable Module](#) section:

```
....
# These are not required until using the internal gateway:
iptables -t mangle -A INPUT -i lo -p tcp --dport 1111 -j SECMARK --selctx
system_u:object_r:int_gateway_packet_t
iptables -t mangle -A INPUT -i lo -p tcp --sport 1111 -j SECMARK --selctx
system_u:object_r:int_gateway_packet_t
....
....
#----- OUTPUT IP Stream -----#
....
#
# These are not required until using the internal gateway:
iptables -t mangle -A OUTPUT -o lo -p tcp --dport 1111 -j SECMARK --selctx
system_u:object_r:int_gateway_packet_t
iptables -t mangle -A OUTPUT -o lo -p tcp --sport 1111 -j SECMARK --selctx
system_u:object_r:int_gateway_packet_t
....
```

11. Ensure all the files are correctly labeled by running the `restorecon` command using the input file produced in step 4 above:

```
restorecon -r -f restorecon_file
```

12. Run enforcing mode:

```
setenforce 1
```

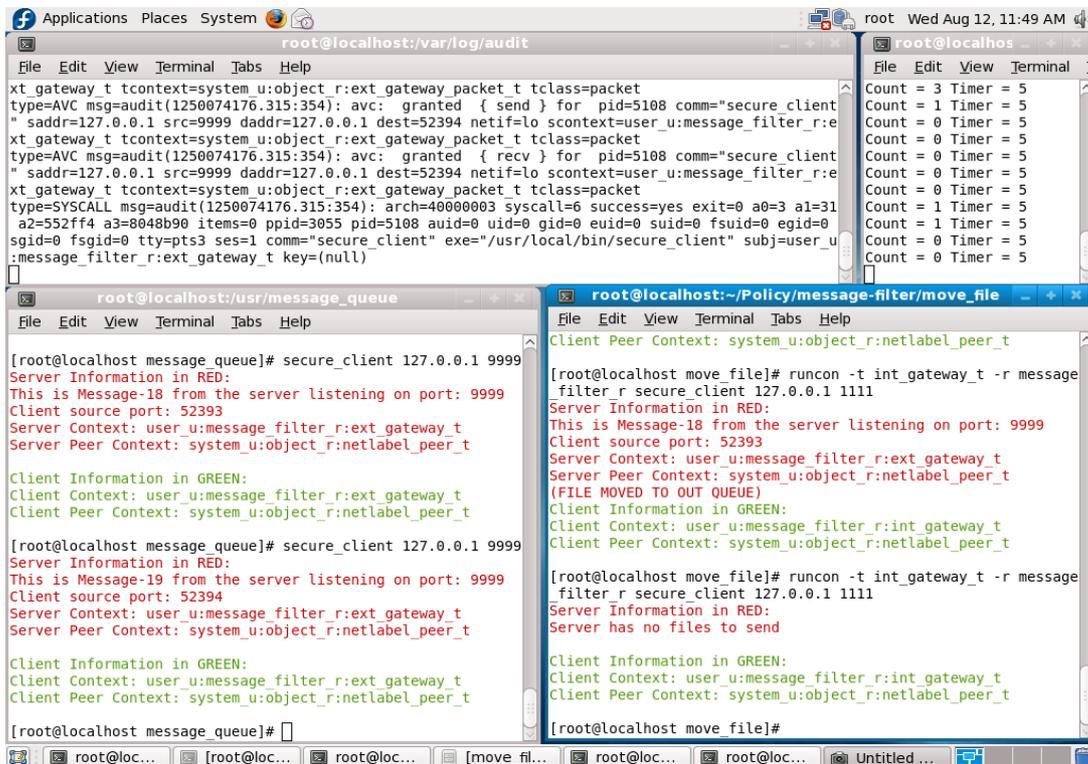
The message filter should now be ready to test.

### 3.4.4 Testing the Message Filter Build

To test the message filter it is recommended that four virtual terminal sessions are opened (as shown in [Figure 3.5](#)) for:

1. Running the external gateway client as it will display status messages if successful. This is shown on bottom left hand side using port 9999. Note that this process is run directly from the command line by `secure_server 9999` as it will automatically transition to the `ext_gateway_t` domain by the policy rules.

1. Running the internal gateway client as it will display status messages if successful.. This is shown on bottom right hand side using port 1111. Note that this process (and the secure server for the internal gateway) has to be run via the `runcon` command because of the type enforcement rules discussed in the Type Enforcement Rules section of 'The Foundations' volume.
2. Running the servers as they display messages when connections are made with the clients.
3. Viewing the audit log file. Note that the module has `auditallow` rules on `packet { send recv }` so that these events can be seen. This is top left.
4. Starting and viewing the file mover application as this will be run to display a count of the files being moved. This is top right.



**Figure 3.5: Testing the message filter service**

If there are four terminal sessions logged in as root as shown in [Figure 3.5](#), then the follow commands will need to be executed to show the message filter is working:

1. In the session that will display the audit log, execute the following command:

```
tail -f /var/log/audit/audit.log.
```

2. In a session run the following command to load the iptables (it is assumed that the current directory is where the file is located):

```
./iptables_secmark
```

3. Each of the server processes for the gateways will be run in background using one of the sessions with the following commands:

```
# Start the external gateway in background with the 'in' argument
# so that files are created in the in_queue with the
communications
# traffic:

secure_server 9999 in &
```

```
# Start the internal gateway in background using the runcon
command # with the 'out' argument so that files are read from the
out_queue:

runcon -t int_gateway_t -r message_filter_r secure_server 1111 out
&
```

4. In a session start the file mover application with a time in seconds argument so that it will loop and display the number of files moved:

```
move_file 5
```

5. In a session start the secure external gateway client:

```
secure_client 127.0.0.1 9999
```

6. In a session start the secure internal gateway client using the runcon command:

```
runcon -t int_gateway_t -r message_filter_r secure_client 127.0.0.1 1111
```

7. Keep repeating the client commands and the messages should be displayed in each window as the clients are run.

If the external gateway client is run a number of times, the messages will be read from the `in_queue` by the file mover and queued to the `out_queue`, the internal gateway client can then be run to read each message off the `out_queue`. The queues can be investigated for their context by using `ls -Z`, however to do this, enforcing mode must be off otherwise `unconfined_t` (that is the logon sessions domain) cannot read these areas.

## 4. Experimenting with X-Windows

### 4.1 Section Overview

The main objectives of this section are to:

1. Demonstrate the use of 'selections' using polyinstantiation and non-polyinstantiation services of the XSELinux Object Manager (OM) with simple Xlib simple select and paste applications.
2. Use the XSELinux OM `SELinuxGet..` series of functions to display various context information that is available while executing the select and paste examples.
3. Build a simple menu driven test application that will allow all the `SELinuxGet/Set..` functions to be called and view the results.

It is recommended that the `notebook-source-1.1.0-1.tar.gz` file is installed in `$HOME` as this contains all the configuration files and source code required to produce the required modules (the file also contains README and simple Makefiles).

This section assumes the following:

- The message filter modules have been removed before starting this exercise, however it is not mandatory.
- SELinux is configured to use the `modular-test` policy in permissive mode initially to build the services. The `modular-test` policy is described in the [Building the Base Policy Module](#) section.

### 4.2 Overview of Modules and Applications

The loadable modules used to support these exercises are built using standard SELinux language statements and rules with customised `x_contexts` files to support the labeling of objects.

The test applications are written in 'C' and use the Xlib function library with Xdevice functions provided by the Xi library. There were a few problems encountered that are discussed in the [Calling the XSELinux Functions](#) section.

#### 4.2.1 The `x_contexts` Files and Supporting Loadable Module

The source files required to build and manage the new `x_contexts` files and supporting loadable module are located in:

```
./notebook-source/x-windows/x-contexts-base-module
```

As the objective of the demonstration is to show how different entries in the `x_contexts` file affect the use of selections it was decided to build two `x_contexts` files based on those in the Reference Policy 20090730 build. To support the new entries created in these `x_contexts` files, required an additional loadable module (`x_context_base.conf`).

The `x_contexts` files are expanded to give each entry a unique label so that it could be detected in the audit log with `audit2allow` when in enforcing mode, a decision could then be made as to whether an `allow` or `dontaudit` rule would be added to the policy. Additional entries were also added just to experiment. A second copy of the file was made that had the `poly_` keyword added to the property and selection entries to test polyinstantiation.

The only entry that caused problems during testing was the:

```
poly_property _XKB_RULES_NAMES .....
```

This entry had to have the `poly_` keyword removed in the polyinstantiated file as it stopped various keys from working (up/down etc. keys) on the keyboard.

The new `x_contexts` files generated are called:

**x\_contexts-file-with-new-labels** - This file is similar to that used by the reference policy. The select and paste policy uses the same method to manage the labeling as the reference policy - called derived labeling as the objects label is derived from an SELinux user name or a prefix (from the 'users\_extra' configuration file), then use a `type_transition` to transition the object to the new label on creation. For example (using standard Refpolicy):

An `x_contexts` entry of:

```
event X11:MapNotify system_u:object_r:manage_xevent_t
```

and the `ssh` policy module (after expansion) having a `type_transition` statement generated by the build process of:

```
type_transition ssh_t manage_xevent_t : x_event
    ssh_manage_xevent_t;
```

will relabel any objects created from `manage_xevent_t` to `ssh_manage_xevent_t`.

**x\_contexts-file-with-new-polylabels** - This is used to support polyinstantiated entries (note - the reference policy does not currently use polyinstantiated entries). With polyinstantiation enabled, the select and paste policy uses the `type_member` rule to enforce the selection to a specific domain (in this example the `x_select_paste_t` domain) as follows:

```
type_member x_select_paste_t primary_xselection_t :
x_selection
    x_select_paste_t;
```

To support these new `x_contexts` file entries an additional policy module was built that defines a `type` for each entry and a corresponding `allow` rule. This module is called `x_context_base.conf` and must be loaded and active when the `modular-test` policy is loaded with either of the new `x_contexts` files. Failure to do this will probably result in the system hanging as it tries to load X-Windows with no defined `type` or `allow` rules for the new `x_contexts` file.

To experiment with additional `x_context` entries:

- 1) Add a new entry in the appropriate `x_contexts` file such as:

```
property WM_ZOOM_HINTS system_u:object_r:wm_zoom_hints_xproperty_t
```

or

```
poly_property WM_ZOOM_HINTS system_u:object_r:wm_zoom_hints_xproperty_t
```

- 2) Add new entries in the `x_context_base.conf` for the type and allow statements:

```
type wm_zoom_hints_xproperty_t;
allow unconfined_t wm_zoom_hints_xproperty_t : x_property
*;
```

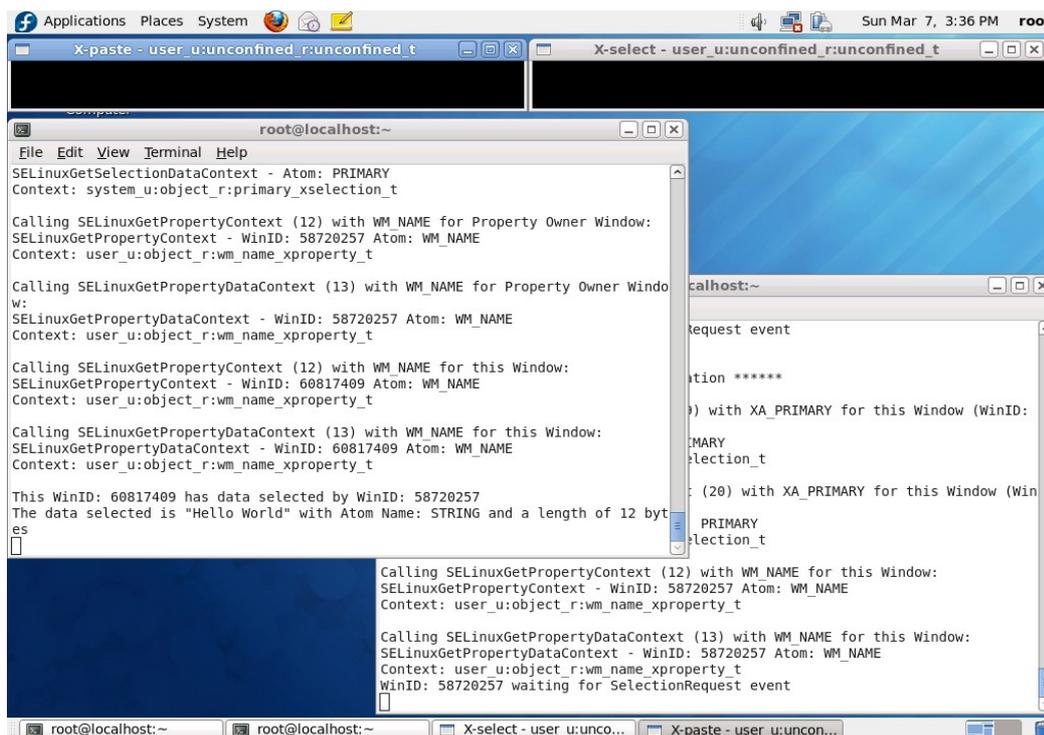
- 3) Run the `make module` command (in the `./x-windows/x-contexts-base-module` directory) and copy over the appropriate `x_contexts` file to `/etc/selinux/modular-test/contexts`.

### 4.2.2 The Select - Paste Applications and Loadable Module

The source files required to build and manage the application and loadable module are located in:

```
./notebook-source/x-windows/x-select+paste
```

There are two simple X-Windows applications that select (X-select) and paste (X-paste) “Hello World” using Xlib selection functions. When they are loaded they show the application name and their context in the title bar as shown in [Figure 4.1](#). Integrated with these applications are calls to the `XSELinuxGet..` functions to return context information as the Xlib functions are executed.



**Figure 4.1: X-select and X-paste running in `unconfined_t`**

The output from the applications can also be captured in a file by adding the capture file name as an argument:

```
X-select poly-demo.txt

# The output will be in poly-demo.txt, with some text also
# displayed on the screen.
```

When the two applications are built they are moved to `/usr/local/bin` and have the default label of `unconfined_t`. When they are both loaded in the `unconfined_t` domain, there are no enforced rules (i.e. there are no restrictions). If the `x_select_paste.conf` module is built and loaded, then when they are run as:

```
runcon -t x_select_paste_t X-select
```

and

```
runcon -t x_select_paste_t X-paste
```

Policy will be enforced as required depending on a boolean that when set to:

```
setsebool -P poly-selection false
```

and the `x_contexts-file-with-new-labels` file is installed as the `x_contexts` file, then the derived policy rules will be enforced.

If the boolean is set to:

```
setsebool -P poly-selection true
```

and the `x_contexts-file-with-new-polylabels` file is installed as the `x_contexts` file, then the polyinstantiated policy rules will be enforced.

### 4.2.2.1 Test Conclusions

After a number of experiments the following conclusions were reached:

- 1) Using the non-polyinstantiated `x_contexts` file (with `poly-selection = FALSE`), resulted in selections being seen across all windows whether running in `unconfined_t` or `x_select_paste_t` domains.
- 2) Using the polyinstantiated `x_contexts` file (with `poly-selection = TRUE`), resulted in selections being restricted to windows running in their own domains (e.g. if running the X-select in the `unconfined_t` domain and X-paste in the `x_select_paste_t` domains, the selected text will not be pasted).
- 3) If the following multiple selection entries are added to the `x_contexts` file, then the non `poly_` entry takes precedence. This means that polyinstantiation for selections will not work (even if a different label is used for each entry).

```
# The poly and non-poly entries cannot be in the x_contexts
# file as the non-poly entry takes precedence:
poly_selection PRIMARY system_u:object_r:primary_xselection_t
selection          PRIMARY system_u:object_r:primary_xselection_t

# Even if different labels are used:
poly_selection PRIMARY system_u:object_r:poly_primary_xselection_t
selection          PRIMARY system_u:object_r:primary_xselection_t
```

Therefore the overall conclusion is that for non-MLS policies, the only effective way to control selections is using polyinstantiation with the `type_member` rule.

The reason for stating non-MLS policy is that the MLS policy uses `mlsconstrain` rules to manage restrictions. Various `constrain` rules were used for non-MLS policy testing but no satisfactory result could be obtained - do you know different !!!

Notes:

- a) When using polyinstantiation the `poly_` keyword must be present in the `x_contexts` file and there must be a corresponding `type_member` rule in the policy.
- b) When analysing the output from the XSELinux function calls between non-polyinstantiated (or derived) and polyinstantiated services when the X-select and X-paste applications are running (apart from their context information), the only major difference was that when calling the `SELinuxListSelections` function, the polyinstantiated service had an additional **PRIMARY** entry (shown in **bold**) as follows:

```
# Non-polyinstantiated (derived) running in
x_select_paste_t:
#
Calling SELinuxListSelections (21) for this display:
```

```
SELinuxListSelections (1 of 10) - Atom: CLIPBOARD
Object Context: system_u:object_r:clipboard_xselection_t
Data Context:   system_u:object_r:clipboard_xselection_t
```

```
SELinuxListSelections (2 of 10) - Atom: PRIMARY
Object Context: system_u:object_r:primary_xselection_t
Data Context:   system_u:object_r:primary_xselection_t
```

```
# Polyinstantiated running in x_select_paste_t:
Calling SELinuxListSelections (21) for this display:

SELinuxListSelections (1 of 11) - Atom: CLIPBOARD
Object Context: system_u:object_r:clipboard_xselection_t
Data Context:   system_u:object_r:clipboard_xselection_t

SELinuxListSelections (2 of 11) - Atom: PRIMARY
Object Context: system_u:object_r:primary_xselection_t
Data Context:   system_u:object_r:primary_xselection_t

SELinuxListSelections (3 of 11) - Atom: PRIMARY
Object Context: system_u:object_r:x_select_paste_t
Data Context:   system_u:object_r:x_select_paste_t
```

- c) The Reference Policy does not use polyinstantiation but supports isolation only with the MLS policy where `mlsconstrain` rules are enforced (see the `mlsconstrain x_selection` entries in the `mls` configuration file).
- d) Various `constrain` rules were tried to limit selections with the non-polyinstantiated `x_contexts` file, but no satisfactory solution was found - any offers !!, therefore when using non-MLS policy, the only way to limit selections is via polyinstantiation. Some example `constrain` rules tried that had the following results:

```
# Add constrain rule to base.conf:
constrain x_selection { read getattr } (t1 == unconfined_t);

# When running "runcon -t x_select_paste_t X-paste" it flags the following
# AVC entry in the Xorg.0.log file:
(WW) avc: denied { getattr } for request=X11:GetSelectionOwner comm=X-
paste selection=PRIMARY scontext=user_u:unconfined_r:x_select_paste_t
tcontext=system_u:object_r:primary_xselection_t tclass=x_selection

# When running X-paste (in unconfined_t) then no errors in log.
```

```
# Add constrain rule to base.conf:
constrain x_selection { read getattr } (t1 == secure_select);
# Where secure_select is an attribute declared in base.conf

# With the following added to x_select_paste.conf:
require { attribute secure_select; ... }
typeattribute x_select_paste_t secure_select;

# When running "runcon -t x_select_paste_t X-paste" there are no errors in
# the log.

# When running X-paste (in unconfined_t) it flags the following AVC entry
# in the Xorg.0.log file:
```

```
(WW) avc: denied { getattr } for request=X11:GetSelectionOwner comm=X-  
paste selection=PRIMARY scontext=user_u:unconfined_r:unconfined_t  
tcontext=system_u:object_r:primary_xselection_t tclass=x_selection
```

### 4.2.2.2 Calling the XSELinux Functions

The X-select, X-paste and X-setest applications call the object manager XSELinuxGet/Set.. functions to get and set contexts as required. To use these functions the standard Xlib GetReq, \_XSend and \_XReply functions need to be called to manage the request / response sequences. As there are 23 functions it was decided to build these into a separate 'C' module called XSELinuxOMFunctions.c that is supported by a header file called Xlib-selinux.h. that are located in the ./x-windows/x-common directory.

The header file is based on the XSELinux extension source header xselinux.h and has been expanded to support the Xlib GetReq macro and associated functions. The only point to note is that the SELinuxQueryVersion request header structure size had to be set to 4 instead of 6 as the client\_major and client\_minor entries were not used and caused errors when added.

The error handling caused much grief (as not an Xlib expert), and it will be seen that there are a number of flags to indicate certain error sequences. The source code has plenty of comments regarding these and if anyone has better methods let the author know.

## 4.3 Building the X-Windows Select and Paste Examples

To build and test the infrastructure to support modified x\_contexts files for the X-Windows object manager, the following will be required:

- a) The Base Module described in the [Building the Base Policy Module](#) section. This will install the base policy module and supporting files in the /etc/selinux/modular-test area.
- b) Two modified x\_contexts files. The Reference Policy sample has been modified to capture additional entries and for each entry allocate its own unique object label. There is one file to support the way the Reference Policy (build 20090730) supports these objects<sup>4</sup>, and the other has the additional 'poly\_' keyword added to support polyinstantiated property and selection entries.

Important note - These sample x\_contexts files must not be used with the reference policy as they are incompatible and will cause the system to hang when X-Windows is being loaded

- c) A loadable module (x\_context\_base.conf) that contains the policy type statements and allow rules of the newly defined x\_contexts file entries described in bullet b). This will allow the X-Windows object manager to load the new x\_contexts file without any errors.

---

<sup>4</sup> Also known as 'derived type' because the objects are assigned labels that are derived from a name based on the SELinux user or a prefix (e.g. from the 'users\_extra' configuration file) and then uses a type\_transition statement to transition the object to the new label on creation.

- d) Two simple X-Windows applications - X-select to automatically select some text (Hello World), and X-paste to paste the text onto the screen. These applications use the minimum Xlib functions possible, however they also contain calls to the SELinux X-Windows functions that are built into the object manager to retrieve context information as the applications execute.
- e) A loadable module (x\_select\_paste.conf) that contains the policy for enforcing the X-select and X-paste applications when running in the x\_select\_paste\_t domain. This policy supports the polyinstantiated x\_contexts file by setting a boolean (poly-selection) to TRUE and the the derived x\_contexts file by setting the boolean to FALSE.

The build and testing will be carried out in the following stages:

- 1) Ensure that the modular-test base module has been built and tested as described in the [Building the Base Policy Module](#) section.
- 2) Build the new x\_contexts files and a loadable module (x\_context\_base.conf). The files to are available in the source file and located in the ./notebook-source/x-windows/x-contexts-base-module directory.
- 3) Build the X-select, X-paste applications and their supporting loadable module for running in the x\_select\_paste\_t domain.
- 4) Install the derived (non-polyinstantiated) x\_contexts file and test using the X-select and X-paste applications in various scenarios using the unconfined\_t and x\_select\_paste\_t domains, recording the results.
- 5) Install the polyinstantiated x\_contexts file and test using the X-select and X-paste applications in various scenarios using the unconfined\_t and x\_select\_paste\_t domains, recording the results.

### 4.3.1 Building the x\_contexts Files and Loadable Module

Before building and installing these, ensure that the modular-test base module has been built, if it has proceed as follows:

- 1) Ensure you are logged on as 'root' and SELinux is running in permissive mode (setenforce 0) to perform the build process. It is assumed that the files are built in the ./notebook-source/x-windows/x-contexts-base-module directory.
- 2) Produce a derived x\_contexts file called x\_contexts-file-with-new-labels with the following entries:

```
#
# Config file for XSELinux extension
#
#####
#
# Each entry in this file has a different label that is based on their
# Object Name for testing a basic policy that is explained in:
#   The SELinux Notebook - The Foundations
#
# This version (x_contexts-base.conf-new-labels) does not have
# polyinstantiated entries. For testing X polyinstantiated objects, use
```

## The SELinux Notebook - Sample Policy Source

```
# the x_contexts-base.conf-new-polylabels file. #
# #
#####
#
### Rules for X Clients
# The default client rule defines a context to be used for all clients
# connecting to the server from a remote host.
#
client *          system_u:object_r:remote_xclient_t

### Rules for X Properties
# Property rules map a property name to a context.  A default property
# rule indicated by an asterisk should follow all other property rules.
#
# Properties that normal clients may only read
property XFree86_VT          system_u:object_r:xfree86_vt_xproperty_t
property XFree86_DDC_EDID1_RAWDATA
    system_u:object_r:xfree86_ddc_edidl_rawdata_xproperty_t
property RESOURCE_MANAGER   system_u:object_r:resource_manager_xproperty_t
property SCREEN_RESOURCES   system_u:object_r:screen_resources_xproperty_t
property MIT_PRIORITY_COLORS
    system_u:object_r:mit_priority_colors_xproperty_t
property AT_SPI_IOR         system_u:object_r:at_spi_ior_xproperty_t
property SELINUX_CLIENT_CONTEXT
    system_u:object_r:selinux_client_context_xproperty_t
property NET_WORKAREA       system_u:object_r:net_workarea_xproperty_t
# Need to remove poly_property from this as it stops some keys working on
# keyboard !!!
property _XKB_RULES_NAMES
    system_u:object_r:xkb_rules_names_xproperty_t

# Clipboard and selection properties
property CUT_BUFFER0        system_u:object_r:cut_buffer0_xproperty_t
property CUT_BUFFER1        system_u:object_r:cut_buffer1_xproperty_t
property CUT_BUFFER2        system_u:object_r:cut_buffer2_xproperty_t
property CUT_BUFFER3        system_u:object_r:cut_buffer3_xproperty_t
property CUT_BUFFER4        system_u:object_r:cut_buffer4_xproperty_t
property CUT_BUFFER5        system_u:object_r:cut_buffer5_xproperty_t
property CUT_BUFFER6        system_u:object_r:cut_buffer6_xproperty_t
property CUT_BUFFER7        system_u:object_r:cut_buffer7_xproperty_t

# Don't really need these as if not defined they will default to the
# Default fallback type below.
# Added these as they are used by the XSetWMProperties function call:
property WM_NAME             system_u:object_r:wm_name_xproperty_t
property WM_ICON_NAME       system_u:object_r:wm_incon_name_xproperty_t
property WM_HINTS           system_u:object_r:wm_hints_xproperty_t
property WM_NORMAL_HINTS    system_u:object_r:wm_normal_hints_xproperty_t
property WM_CLASS           system_u:object_r:wm_class_xproperty_t
property WM_COMMAND         system_u:object_r:wm_command_xproperty_t
property WM_CLIENT_MACHINE  system_u:object_r:wm_client_machine_xproperty_t

# Add XA_STRING:
property STRING              system_u:object_r:string_xproperty_t

# As each Window has its own properties it is important to make sure
# the undefined_xproperty_t is transitioned to the correct type when
# building a module that uses 'derived' types (see x_derived_test.conf).
# Default fallback type
property *                   system_u:object_r:undefined_xproperty_t

### Rules for X Extensions
# Extension rules map an extension name to a context.  A default extension
# rule indicated by an asterisk should follow all other extension rules.
#
# Standard extensions
extension BIG-REQUESTS      system_u:object_r:big-requests_xextension_t
extension SHAPE             system_u:object_r:shape_xextension_t
extension SYNC              system_u:object_r:sync_xextension_t
extension XC-MISC          system_u:object_r:xc-misc_xextension_t
extension XFIXES           system_u:object_r:xfixes_xextension_t
extension XInputExtension  system_u:object_r:xinputextension_xextension_t
extension XKEYBOARD        system_u:object_r:xkeyboard_xextension_t
extension DAMAGE           system_u:object_r:damage_xextension_t
```

```
extension RENDER                system_u:object_r:render_xextension_t
extension XINERAMA              system_u:object_r:xinerama_xextension_t

# Direct hardware access extensions
extension XFree86-DGA          system_u:object_r:xfree86-dga_xextension_t
extension XFree86-VidModeExtension system_u:object_r:xfree86-
vidmodeextension_xextension_t

# Screen management and multihead extensions
extension RANDR                system_u:object_r:randr_xextension_t
extension Composite            system_u:object_r:composite_xextension_t

# Screensaver, power management extensions
extension DPMS                 system_u:object_r:dpms_xextension_t
extension MIT-SCREEN-SAVER     system_u:object_r:mit-screen-saver_xextension_t

# Shared memory extensions
extension MIT-SHM              system_u:object_r:mit-shm_xextension_t
extension XFree86-Bigfont      system_u:object_r:xfree86-bigfont_xextension_t

# Accelerated graphics, OpenGL, direct rendering extensions
extension GLX                  system_u:object_r:glx_xextension_t
extension NV-CONTROL           system_u:object_r:nv-control_xextension_t
extension NV-GLX               system_u:object_r:nv-glx_xextension_t
extension NVIDIA-GLX           system_u:object_r:nvidia-glx_xextension_t

# Debugging, testing, and recording extensions
extension RECORD               system_u:object_r:record_xextension_t
extension X-Resource           system_u:object_r:x-resource_xextension_t
extension XTEST                system_u:object_r:xtest_xextension_t

# Security-related extensions
extension SECURITY              system_u:object_r:security_xextension_t
extension SELinux              system_u:object_r:selinux_xextension_t
extension XAccessControlExtension
    system_u:object_r:xaccesscontrolextension_xextension_t
extension XC-APPGROUP          system_u:object_r:xc-appgroup_xextension_t

# Video extensions
extension XVideo               system_u:object_r:xvideo_xextension_t
extension XVideo-MotionCompensation system_u:object_r:xvideo-
motioncompensation_xextension_t

# Default fallback type
extension *                     system_u:object_r:undefined_xextension_t

### Rules for X Selections
# Selection rules map a selection name to a context.  A default selection
# rule indicated by an asterisk should follow all other selection rules.
#
# Polyinstantiated entries
# Standard selections
selection XA_PRIMARY           system_u:object_r:xa_primary_xselection_t
selection XA_SECONDARY        system_u:object_r:xa_secondary_xselection_t
selection PRIMARY              system_u:object_r:primary_xselection_t
selection CLIPBOARD            system_u:object_r:clipboard_xselection_t

# Default fallback type
selection *                    system_u:object_r:undefined_xselection_t

### Rules for X Events
# Event rules map an event protocol name to a context.  A default event
# rule indicated by an asterisk should follow all other event rules.
#
# Input events
event X11:KeyPress             system_u:object_r:x11_keypress_xevent_t
event X11:KeyRelease           system_u:object_r:x11_keyrelease_xevent_t
event X11:ButtonPress          system_u:object_r:x11_buttonpress_xevent_t
event X11:ButtonRelease        system_u:object_r:x11_buttonrelease_xevent_t
event X11:MotionNotify         system_u:object_r:x11_motionnotify_xevent_t
event X11:SelectionNotify      system_u:object_r:x11_selectionnotify_xevent_t
# Added two additional selection events:
event X11:SelectionRequest     system_u:object_r:x11_selectionrequest_xevent_t
event X11:SelectionClear       system_u:object_r:x11_selectionclear_xevent_t
```

```
event XInputExtension:DeviceKeyPress
system_u:object_r:xinputextension_devicekeypress_xevent_t
event XInputExtension:DeviceKeyRelease
system_u:object_r:xinputextension_devicekeyrelease_xevent_t
event XInputExtension:DeviceButtonPress
system_u:object_r:xinputextension_devicebuttonpress_xevent_t
event XInputExtension:DeviceButtonRelease
system_u:object_r:xinputextension_devicebuttonrelease_xevent_t
event XInputExtension:DeviceMotionNotify
system_u:object_r:xinputextensionext_devicemotionnotify_xevent_t
event XInputExtension:DeviceValuator
system_u:object_r:xinputextension_devicevaluator_xevent_t
event XInputExtension:ProximityIn
system_u:object_r:xinputextension_proximityin_xevent_t
event XInputExtension:ProximityOut
system_u:object_r:xinputextension_proximityout_xevent_t

# Focus events
event X11:FocusIn system_u:object_r:x11_focugin_xevent_t
event X11:FocusOut system_u:object_r:x11_focusout_xevent_t
event X11:EnterNotify system_u:object_r:x11_enternotify_xevent_t
event X11:LeaveNotify system_u:object_r:x11_leavenotify_xevent_t

# Property events
event X11:PropertyNotify system_u:object_r:x11_propertynotify_xevent_t

# Client message events
event X11:ClientMessage system_u:object_r:x11_clientmessage_xevent_t

# Manager events
event X11:ConfigureRequest system_u:object_r:x11_configurerequest_xevent_t
event X11:ResizeRequest system_u:object_r:x11_resizerequest_xevent_t
event X11:MapRequest system_u:object_r:x11_maprequest_xevent_t
event X11:CirculateRequest system_u:object_r:x11_circulaterequest_xevent_t
event X11:CreateNotify system_u:object_r:x11_createnotify_xevent_t
event X11:DestroyNotify system_u:object_r:x11_destroynotify_xevent_t
event X11:MapNotify system_u:object_r:x11_mapnotify_xevent_t
event X11:UnmapNotify system_u:object_r:x11_unmapnotify_xevent_t
event X11:ReparentNotify system_u:object_r:x11_reparentnotify_xevent_t
event X11:ConfigureNotify system_u:object_r:x11_confignotify_xevent_t
event X11:GravityNotify system_u:object_r:x11_gravitynotify_xevent_t
event X11:CirculateNotify system_u:object_r:x11_circulatenotify_xevent_t
event X11:Expose system_u:object_r:x11_expose_xevent_t
event X11:VisibilityNotify system_u:object_r:x11_visibilitynotify_xevent_t

# Unknown events (that are not registered in the X server's name database)
event <unknown> system_u:object_r:unknown_xevent_t

# Default fallback type
event * system_u:object_r:undefined_xevent_t
```

- 3) Produce a polyinstantiated `x_contexts` file called `x_contexts-file-with-new-polylabels`. Not all entries are shown in the file below as the easiest way to produce this is to copy the `x_contexts` file above and add the `'poly_'` keyword to the property and selection entries as follows:

```
#
# Config file for XSELinux extension
#
#####
#
# Each entry in this file has a different label that is based on their
# Object Name for testing a basic policy that is explained in:
#   The SELinux Notebook - The Foundations
#
# This version (x_contexts-base.conf-new-polylabels) does not have
# polyinstantiated entries. For testing X non-polyinstantiated objects,
# use the x_contexts-base.conf-new-labels file.
#
#####
#
#
```

## The SELinux Notebook - Sample Policy Source

```
##
### Rules for X Clients
.....
.....

### Rules for X Properties
# Property rules map a property name to a context. A default property
# rule indicated by an asterisk should follow all other property rules.
#
# Polyinstantiated entries
# Properties that normal clients may only read
poly_property XFree86_VT          system_u:object_r:xfree86_vt_xproperty_t
poly_property XFree86_DDC_EDID1_RAWDATA
system_u:object_r:xfree86_ddc_edidl_rawdata_xproperty_t
poly_property RESOURCE_MANAGER
system_u:object_r:resource_manager_xproperty_t
poly_property SCREEN_RESOURCES
system_u:object_r:screen_resources_xproperty_t
poly_property _MIT_PRIORITY_COLORS
system_u:object_r:mit_priority_colors_xproperty_t
poly_property AT_SPI_IOR          system_u:object_r:at_spi_ior_xproperty_t
poly_property _SELINUX_CLIENT_CONTEXT
system_u:object_r:selinux_client_context_xproperty_t
poly_property _NET_WORKAREA
system_u:object_r:net_workarea_xproperty_t
# Need to remove poly_property from this as it stops some keys working on
keyboard !!!
property _XKB_RULES_NAMES
system_u:object_r:xkb_rules_names_xproperty_t

# Clipboard and selection properties
poly_property CUT_BUFFER0        system_u:object_r:cut_buffer0_xproperty_t
poly_property CUT_BUFFER1        system_u:object_r:cut_buffer1_xproperty_t
poly_property CUT_BUFFER2        system_u:object_r:cut_buffer2_xproperty_t
poly_property CUT_BUFFER3        system_u:object_r:cut_buffer3_xproperty_t
poly_property CUT_BUFFER4        system_u:object_r:cut_buffer4_xproperty_t
poly_property CUT_BUFFER5        system_u:object_r:cut_buffer5_xproperty_t
poly_property CUT_BUFFER6        system_u:object_r:cut_buffer6_xproperty_t
poly_property CUT_BUFFER7        system_u:object_r:cut_buffer7_xproperty_t

# Don't really need these as if not defined they will default to the
# Default fallback type below.
# Added these as they are used by the XSetWMProperties function call:
poly_property WM_NAME            system_u:object_r:wm_name_xproperty_t
poly_property WM_ICON_NAME      system_u:object_r:wm_incon_name_xproperty_t
poly_property WM_HINTS          system_u:object_r:wm_hints_xproperty_t
poly_property WM_NORMAL_HINTS
system_u:object_r:wm_normal_hints_xproperty_t
poly_property WM_CLASS          system_u:object_r:wm_class_xproperty_t
poly_property WM_COMMAND        system_u:object_r:wm_command_xproperty_t
poly_property WM_CLIENT_MACHINE
system_u:object_r:wm_client_machine_xproperty_t

# Add XA_STRING:
poly_property STRING            system_u:object_r:string_xproperty_t

# As each Window has its own properties it is important to make sure
# the undefined_xproperty_t is transitioned to the correct type when
# building a module that uses 'derived' types (see x_derived_test.conf).
# Default fallback type
poly_property *                system_u:object_r:undefined_xproperty_t

### Rules for X Extensions
.....
.....

### Rules for X Selections
# Selection rules map a selection name to a context. A default selection
# rule indicated by an asterisk should follow all other selection rules.
#
# Polyinstantiated entries
# Standard selections
poly_selection XA_PRIMARY        system_u:object_r:xa_primary_xselection_t
poly_selection XA_SECONDARY      system_u:object_r:xa_secondary_xselection_t
poly_selection PRIMARY          system_u:object_r:primary_xselection_t
```

```
poly_selection CLIPBOARD      system_u:object_r:clipboard_xselection_t

# Default fallback type
poly_selection *              system_u:object_r:undefined_xselection_t

### Rules for X Events
....
.....
```

- 4) Produce the `x_context_base.conf` policy file with the following contents:

```
module x_context_base 1.0.0;
#
#####
#
# This Loadable Module will allow the X-Windows OM to label objects      #
# using the sample x_contexts files that form part of the test examples: #
#   x_contexts-file-with-new-labels                                     #
#   x_contexts-file-with-new-polylabels                               #
#                                                                 #
#####
#
require {
    type unconfined_t;
    class x_property { create destroy read write append getattr setattr };
    class x_selection { read write getattr setattr };
    class x_extension { query use };
    class x_event { send receive };
    class x_synthetic_event { send receive };
}

#
##### START TYPES #####
#
# The default client rule defines a context to be used for all clients
# connecting to the server from a remote host.
#
type remote_xclient_t;

### Rules for X Properties
# Property rules map a property name to a context.  A default property
# rule indicated by an asterisk should follow all other property rules.
#
# Properties that normal clients may only read
type xfree86_vt_xproperty_t;
type xfree86_ddc_edid1_rawdata_xproperty_t;
type resource_manager_xproperty_t;
type screen_resources_xproperty_t;
type mit_priority_colors_xproperty_t;
type at_spi_ior_xproperty_t;
type selinux_client_context_xproperty_t;
type net_workarea_xproperty_t;
type xkb_rules_names_xproperty_t;

# Clipboard and selection properties
type cut_buffer0_xproperty_t;
type cut_buffer1_xproperty_t;
type cut_buffer2_xproperty_t;
type cut_buffer3_xproperty_t;
type cut_buffer4_xproperty_t;
type cut_buffer5_xproperty_t;
type cut_buffer6_xproperty_t;
type cut_buffer7_xproperty_t;

# Added these as they are used by the XSetWMPProperties function call:
type wm_name_xproperty_t;
type wm_incon_name_xproperty_t;
type wm_hints_xproperty_t;
type wm_normal_hints_xproperty_t;
type wm_class_xproperty_t;
type wm_command_xproperty_t;
```

```
type wm_client_machine_xproperty_t;

# Add XA_STRING:
type string_xproperty_t;

# Default fallback type
type undefined_xproperty_t;

### Rules for X Extensions
# Extension rules map an extension name to a context. A default extension
# rule indicated by an asterisk should follow all other extension rules.
#
# Standard extensions
type big-requests_xextension_t;
type shape_xextension_t;
type sync_xextension_t;
type xc-misc_xextension_t;
type xfixes_xextension_t;
type xinputextension_xextension_t;
type xkeyboard_xextension_t;
type damage_xextension_t;
type render_xextension_t;
type xinerama_xextension_t;

# Direct hardware access extensions
type xfree86-dga_xextension_t;
type xfree86-vidmodeextension_xextension_t;

# Screen management and multihead extensions
type randr_xextension_t;
type composite_xextension_t;

# Screensaver, power management extensions
type dpms_xextension_t;
type mit-screen-saver_xextension_t;

# Shared memory extensions
type mit-shm_xextension_t;
type xfree86-bigfont_xextension_t;

# Accelerated graphics, OpenGL, direct rendering extensions
type glx_xextension_t;
type nv-control_xextension_t;
type nv-glx_xextension_t;
type nvidia-glx_xextension_t;

# Debugging, testing, and recording extensions
type record_xextension_t;
type x-resource_xextension_t;
type xtest_xextension_t;

# Security-related extensions
type security_xextension_t;
type selinux_xextension_t;
type xaccesscontrolextension_xextension_t;
type xc-appgroup_xextension_t;

# Video extensions
type xvideo_xextension_t;
type xvideo-motioncompensation_xextension_t;

# Default fallback type
type undefined_xextension_t;

### Rules for X Selections
# Selection rules map a selection name to a context. A default selection
# rule indicated by an asterisk should follow all other selection rules.
#
# Standard selections
type xa_primary_xselection_t;
type xa_secondary_xselection_t;
type primary_xselection_t;
type clipboard_xselection_t;

# Default fallback type
```

```
type undefined_xselection_t;

### Rules for X Events
# Event rules map an event protocol name to a context.  A default event
# rule indicated by an asterisk should follow all other event rules.
#
# Input events
type x11_keypress_xevent_t;
type x11_keyrelease_xevent_t;
type x11_buttonpress_xevent_t;
type x11_buttonrelease_xevent_t;
type x11_motionnotify_xevent_t;
type x11_selectionnotify_xevent_t;
type xinputextension_devicekeypress_xevent_t;
type xinputextension_devicekeyrelease_xevent_t;
type xinputextension_devicebuttonpress_xevent_t;
type xinputextension_devicebuttonrelease_xevent_t;
type xinputextensionext_devicemotionnotify_xevent_t;
type xinputextension_devicevaluator_xevent_t;
type xinputextension_proximityin_xevent_t;
type xinputextension_proximityout_xevent_t;

# Focus events
type x11_foucsin_xevent_t;
type x11_focusout_xevent_t;
type x11_enternotify_xevent_t;
type x11_leavenotify_xevent_t;

# Property events
type x11_propertynotify_xevent_t;
# Added two additional selection events:
type x11_selectionrequest_xevent_t;
type x11_selectionclear_xevent_t;

# Client message events
type x11_clientmessage_xevent_t;

# Manager events
type x11_configurerequest_xevent_t;
type x11_resize_request_xevent_t;
type x11_maprequest_xevent_t;
type x11_circulaterequest_xevent_t;
type x11_createnotify_xevent_t;
type x11_destroynotify_xevent_t;
type x11_mapnotify_xevent_t;
type x11_unmapnotify_xevent_t;
type x11_reparentnotify_xevent_t;
type x11_confignotify_xevent_t;
type x11_gravitynotify_xevent_t;
type x11_circulatenotify_xevent_t;
type x11_expose_xevent_t;
type x11_visibilitynotify_xevent_t;

# Unknown events (that are not registered in the X server's name database)
type unknown_xevent_t;

# Default fallback type
type undefined_xevent_t;

#
##### END TYPES #####
#

#
##### START ALLOW RULES #####
#

#
# The default client rule defines a context to be used for all clients
# connecting to the server from a remote host.
#
# Does not need an allow rule as no remote clients
# remote_xclient_t;

### Rules for X Properties
```

## The SELinux Notebook - Sample Policy Source

```
# Property rules map a property name to a context. A default property
# rule indicated by an asterisk should follow all other property rules.
#
# Properties that normal clients may only read
allow unconfined_t xfree86_vt_xproperty_t : x_property *;
allow unconfined_t xfree86_ddc_edid1_rawdata_xproperty_t : x_property *;
allow unconfined_t resource_manager_xproperty_t : x_property *;
allow unconfined_t screen_resources_xproperty_t : x_property *;
allow unconfined_t mit_priority_colors_xproperty_t : x_property *;
allow unconfined_t at_spi_ior_xproperty_t : x_property *;
allow unconfined_t selinux_client_context_xproperty_t : x_property *;
allow unconfined_t net_workarea_xproperty_t : x_property *;
allow unconfined_t xkb_rules_names_xproperty_t : x_property *;

# Clipboard and selection properties
allow unconfined_t cut_buffer0_xproperty_t : x_property *;
allow unconfined_t cut_buffer1_xproperty_t : x_property *;
allow unconfined_t cut_buffer2_xproperty_t : x_property *;
allow unconfined_t cut_buffer3_xproperty_t : x_property *;
allow unconfined_t cut_buffer4_xproperty_t : x_property *;
allow unconfined_t cut_buffer5_xproperty_t : x_property *;
allow unconfined_t cut_buffer6_xproperty_t : x_property *;
allow unconfined_t cut_buffer7_xproperty_t : x_property *;

# Added these as they are used by the XSetWMProperties function call:
allow unconfined_t wm_name_xproperty_t : x_property *;
allow unconfined_t wm_incon_name_xproperty_t : x_property *;
allow unconfined_t wm_hints_xproperty_t : x_property *;
allow unconfined_t wm_normal_hints_xproperty_t : x_property *;
allow unconfined_t wm_class_xproperty_t : x_property *;
allow unconfined_t wm_command_xproperty_t : x_property *;
allow unconfined_t wm_client_machine_xproperty_t : x_property *;

# Add XA_STRING:
allow unconfined_t string_xproperty_t : x_property *;

# Default fallback type
allow unconfined_t undefined_xproperty_t : x_property *;

### Rules for X Extensions
# Extension rules map an extension name to a context. A default extension
# rule indicated by an asterisk should follow all other extension rules.
#
# Standard extensions
allow unconfined_t big-requests_xextension_t : x_extension *;
allow unconfined_t shape_xextension_t : x_extension *;
allow unconfined_t sync_xextension_t : x_extension *;
allow unconfined_t xc-misc_xextension_t : x_extension *;
allow unconfined_t xfixes_xextension_t : x_extension *;
allow unconfined_t xinputextension_xextension_t : x_extension *;
allow unconfined_t xkeyboard_xextension_t : x_extension *;
allow unconfined_t damage_xextension_t : x_extension *;
allow unconfined_t render_xextension_t : x_extension *;
allow unconfined_t xinerama_xextension_t : x_extension *;

# Direct hardware access extensions
allow unconfined_t xfree86-dga_xextension_t : x_extension *;
allow unconfined_t xfree86-vidmodeextension_xextension_t : x_extension *;

# Screen management and multihead extensions
allow unconfined_t randr_xextension_t : x_extension *;
allow unconfined_t composite_xextension_t : x_extension *;

# Screensaver, power management extensions
allow unconfined_t dpms_xextension_t : x_extension *;
allow unconfined_t mit-screen-saver_xextension_t : x_extension *;

# Shared memory extensions
allow unconfined_t mit-shm_xextension_t : x_extension *;
allow unconfined_t xfree86-bigfont_xextension_t : x_extension *;

# Accelerated graphics, OpenGL, direct rendering extensions
allow unconfined_t glx_xextension_t : x_extension *;
allow unconfined_t nv-control_xextension_t : x_extension *;
allow unconfined_t nv-glx_xextension_t : x_extension *;
```

```
allow unconfined_t nvidia-glx_xextension_t : x_extension *;

# Debugging, testing, and recording extensions
allow unconfined_t record_xextension_t : x_extension *;
allow unconfined_t x-resource_xextension_t : x_extension *;
allow unconfined_t xtest_xextension_t : x_extension *;

# Security-related extensions
allow unconfined_t security_xextension_t : x_extension *;
allow unconfined_t selinux_xextension_t : x_extension *;
allow unconfined_t xaccesscontrolexextension_t : x_extension *;
allow unconfined_t xc-appgroup_xextension_t : x_extension *;

# Video extensions
allow unconfined_t xvideo_xextension_t : x_extension *;
allow unconfined_t xvideo-motioncompensation_xextension_t : x_extension *;

# Default fallback type
allow unconfined_t undefined_xextension_t : x_extension *;

### Rules for X Selections
# Selection rules map a selection name to a context. A default selection
# rule indicated by an asterisk should follow all other selection rules.
#
# Standard selections
allow unconfined_t xa_primary_xselection_t : x_selection *;
allow unconfined_t xa_secondary_xselection_t : x_selection *;
allow unconfined_t primary_xselection_t : x_selection *;
allow unconfined_t clipboard_xselection_t : x_selection *;

# Default fallback type
allow unconfined_t undefined_xselection_t : x_selection *;

### Rules for X Events
# Event rules map an event protocol name to a context. A default event
# rule indicated by an asterisk should follow all other event rules.
#
# Input events
allow unconfined_t x11_keypress_xevent_t : x_event *;
allow unconfined_t x11_keyrelease_xevent_t : x_event *;
allow unconfined_t x11_buttonpress_xevent_t : x_event *;
allow unconfined_t x11_buttonrelease_xevent_t : x_event *;
allow unconfined_t x11_motionnotify_xevent_t : x_event *;
allow unconfined_t x11_selectionnotify_xevent_t : x_synthetic_event *;
allow unconfined_t xinputextension_devicekeypress_xevent_t : x_event *;
allow unconfined_t xinputextension_devicekeyrelease_xevent_t : x_event *;
allow unconfined_t xinputextension_devicebuttonpress_xevent_t : x_event *;
allow unconfined_t xinputextension_devicebuttonrelease_xevent_t : x_event *;
allow unconfined_t xinputextensionnext_devicemotionnotify_xevent_t : x_event
*;
allow unconfined_t xinputextension_devicevaluator_xevent_t : x_event *;
allow unconfined_t xinputextension_proximityin_xevent_t : x_event *;
allow unconfined_t xinputextension_proximityout_xevent_t : x_event *;

# Focus events
allow unconfined_t x11_focussin_xevent_t : x_event *;
allow unconfined_t x11_focusout_xevent_t : x_event *;
allow unconfined_t x11_enternotify_xevent_t : x_event *;
allow unconfined_t x11_leavenotify_xevent_t : x_event *;

# Property events
allow unconfined_t x11_propertynotify_xevent_t : x_event *;
# Added two additional selection events:
allow unconfined_t x11_selectionrequest_xevent_t : x_event *;
allow unconfined_t x11_selectionclear_xevent_t : x_event *;

# Client message events
allow unconfined_t x11_clientmessage_xevent_t : x_synthetic_event *;

# Manager events
allow unconfined_t x11_configurerequest_xevent_t : x_event *;
allow unconfined_t x11_resizerequest_xevent_t : x_event *;
allow unconfined_t x11_maprequest_xevent_t : x_event *;
allow unconfined_t x11_circulaterequest_xevent_t : x_event *;
allow unconfined_t x11_createnotify_xevent_t : x_event *;
```

```
allow unconfined_t x11_destroynotify_xevent_t : x_event *;
allow unconfined_t x11_mapnotify_xevent_t : x_event *;
allow unconfined_t x11_unmapnotify_xevent_t : x_synthetic_event *;
allow unconfined_t x11_unmapnotify_xevent_t : x_event *;
allow unconfined_t x11_reparentnotify_xevent_t : x_event *;
allow unconfined_t x11_confignotify_xevent_t : x_synthetic_event *;
allow unconfined_t x11_confignotify_xevent_t : x_event *;
allow unconfined_t x11_gravitynotify_xevent_t : x_event *;
allow unconfined_t x11_circulatenotify_xevent_t : x_event *;
allow unconfined_t x11_expose_xevent_t : x_event *;
allow unconfined_t x11_visibilitynotify_xevent_t : x_event *;

# Unknown events (that are not registered in the X server's name database)
allow unconfined_t unknown_xevent_t : x_event *;

# Default fallback type
allow unconfined_t undefined_xevent_t : x_event *;

#
##### END ALLOW RULES #####
#
```

- 5) Compile, package and load the module as follows:

```
checkmodule -m x_context_base.conf -o x_context_base.mod
semodule_package -o x_context_base.pp -m x_context_base.mod
semodule -v -s modular-test -i x_context_base.pp
```

Use the semodule command to check the module has loaded as follows:

```
semodule -l
x_context_base      1.0.0
```

- 6) Copy the derived `x_contexts-file-with-new-labels` to the `modular-test` policy area as the new `x_contexts` file:

```
cp x_contexts-file-with-new-labels
   /etc/selinux/modular-test/contexts/x_contexts
```

- 7) Optionally clear the log file so that they are clear for easier reading after the reboot:

```
> /var/log/audit/audit.log
```

- 8) Ensure that SELinux is configured to run in permissive mode with the `modular-test` policy enabled, then reboot the system to ensure X-windows loads the new `x_contexts` file entries.

```
reboot
```

The system should reload with no errors, however if the screen should remain blank then the chances are that the `x_contexts` file is incorrect and the repair disk will be required to replace the `x_contexts` file with the one produced in the [Building the Base Policy Module](#) section. Alternatively, reboot with a know good policy and check the `modular-test` policy `x_contexts` entries.

Run the `setenforce 1` command and then check the audit log for `USER_AVC` errors (there should not be any errors).

Note that the `x_contexts` file currently loaded is the standard (non-poly) version.

### 4.3.2 Building the X-select and X-paste Applications

Before building and installing these applications, ensure that the libraries and development packages have been installed.

The easiest way to build these applications is to use the notebook-source files (the X-select and X-paste code is in the `./notebook-source/x-windows/x-select+paste` directory). The code to manage the XSELinux functions is quite long and also requires a header file (these are contained in the `./notebook-source/x-windows/x-common` directory). The source files also contain a pre-compiled set of applications that only need to be copied to `/usr/local/bin`. However to build from scratch proceed as follows:

1. Ensure you are logged on as 'root' and SELinux is running in permissive mode (`setenforce 0`) to perform the build process. It is assumed that the applications will be built in the `./notebook-source/x-windows/x-select+paste` directory, but the XSELinux function call code will be in the `./notebook-source/x-windows/x-common` directory as it is shared by the `X-setest` application as well.
2. In the `./notebook-source/x-windows/x-common` directory, produce the `Xlib-selinux.h` header file with the following entries:

```
/*
/* *****
/*
/* The X_SELinux function headers for the Notebook X-Windows demos.
/*
/* Copyright (C) 2010 Richard Haines
/*
/* Note that the X_SELinux function Request and Reply structure
/* definitions have been taken from the XSELinux object manager source.
/*
/* This program is free software: you can redistribute it and/or modify
/* it under the terms of the GNU General Public License as published by
/* the Free Software Foundation, either version 3 of the License, or
/* (at your option) any later version.
/*
/* This program is distributed in the hope that it will be useful,
/* but WITHOUT ANY WARRANTY; without even the implied warranty of
/* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
/* GNU General Public License for more details.
/*
/* You should have received a copy of the GNU General Public License
/* along with this program. If not, see <http://www.gnu.org/licenses/>.
/*
/* *****

/* Extension protocol IDs (struct entry for req->SELinuxReqType) */
#define X_SELinuxQueryVersion 0
#define X_SELinuxSetDeviceCreateContext 1
#define X_SELinuxGetDeviceCreateContext 2
#define X_SELinuxSetDeviceContext 3
#define X_SELinuxGetDeviceContext 4
#define X_SELinuxSetWindowCreateContext 5
#define X_SELinuxGetWindowCreateContext 6
#define X_SELinuxGetWindowContext 7
#define X_SELinuxSetPropertyCreateContext 8
#define X_SELinuxGetPropertyCreateContext 9
#define X_SELinuxSetPropertyUseContext 10
#define X_SELinuxGetPropertyUseContext 11
#define X_SELinuxGetPropertyContext 12
#define X_SELinuxGetPropertyDataContext 13
```

```
#define X_SELinuxListProperties          14
#define X_SELinuxSetSelectionCreateContext 15
#define X_SELinuxGetSelectionCreateContext 16
#define X_SELinuxSetSelectionUseContext 17
#define X_SELinuxGetSelectionUseContext 18
#define X_SELinuxGetSelectionContext 19
#define X_SELinuxGetSelectionDataContext 20
#define X_SELinuxListSelections        21
#define X_SELinuxGetClientContext       22

/*****
/* Define SELinux structures for Extension requests & responses and the */
/* structure sizes (used by the SIZEOF macro in Xmd.h) */
*****/

// The structure defined in the XSELinux Object Manager
// source (./Xext/xselinux.h) seems wrong as this one works:
typedef struct {
    CARD8    reqType;
    CARD8    SELinuxReqType;
    CARD16   length;
    //    CARD8    client_major;
    //    CARD8    client_minor;
} xSELinuxQueryVersionReq;
// #define sz_xSELinuxQueryVersionReq 6
#define sz_xSELinuxQueryVersionReq 4

typedef struct {
    CARD8    type;
    CARD8    pad1;
    CARD16   sequenceNumber;
    CARD32   length;
    CARD16   server_major;
    CARD16   server_minor;
    CARD32   pad2;
    CARD32   pad3;
    CARD32   pad4;
    CARD32   pad5;
    CARD32   pad6;
} xSELinuxQueryVersionReply;
#define sz_xSELinuxQueryVersionReply 32

typedef struct {
    CARD8    reqType;
    CARD8    SELinuxReqType;
    CARD16   length;
    CARD32   context_len;
} xSELinuxSetCreateContextReq;
#define sz_xSELinuxSetCreateContextReq 8

typedef struct {
    CARD8    reqType;
    CARD8    SELinuxReqType;
    CARD16   length;
} xSELinuxGetCreateContextReq;
#define sz_xSELinuxGetCreateContextReq 4

typedef struct {
    CARD8    reqType;
    CARD8    SELinuxReqType;
    CARD16   length;
    CARD32   id;
    CARD32   context_len;
} xSELinuxSetContextReq;
#define sz_xSELinuxSetContextReq12

typedef struct {
    CARD8    reqType;
    CARD8    SELinuxReqType;
    CARD16   length;
    CARD32   id;
} xSELinuxGetContextReq;
#define sz_xSELinuxGetContextReq8

typedef struct {
```

```
CARD8 reqType;
CARD8 SELinuxReqType;
CARD16 length;
CARD32 window;
CARD32 property;
} xSELinuxGetPropertyContextReq;
#define sz_xSELinuxGetPropertyContextReq 12

typedef struct {
    CARD8 type;
    CARD8 pad1;
    CARD16 sequenceNumber;
    CARD32 length;
    CARD32 context_len;
    CARD32 pad2;
    CARD32 pad3;
    CARD32 pad4;
    CARD32 pad5;
    CARD32 pad6;
} xSELinuxGetContextReply;
#define sz_xSELinuxGetContextReply 32

typedef struct {
    CARD8 type;
    CARD8 pad1;
    CARD16 sequenceNumber;
    CARD32 length;
    CARD32 count;
    CARD32 pad2;
    CARD32 pad3;
    CARD32 pad4;
    CARD32 pad5;
    CARD32 pad6;
} xSELinuxListItemsReply;
#define sz_xSELinuxListItemsReply 32

// These are for Get Selection & Property Lists

typedef struct {
    CARD32 name; // Atom name
    CARD32 object_context_len;
    CARD32 data_context_len; // Context
} xSELinuxListItem;
#define sz_xSELinuxListItem 12

// This one holds the list information but needs to be a link list at some
stage
typedef struct {
    CARD32 atom_name;
    CARD32 object_context_len;
    CARD32 data_context_len;
    char object_context [100];
    char data_context [100];
} xSELinuxListItemEntry;

/*****
/*
/* This section defines for each function call:
/* 1) typedefs to form the structure names in line with Xproto.h
/* rules.
/* 2) Defines structure sizes so that SIZEOF macro (defined in Xmd.h)
/* can work when calling GetReq ();
/*
/*
/*****

// X_SELinuxQueryVersion = 0
// No typedef or sz_ are required as they are defined at the start of this
// header file as they are exclusive to X_SELinuxQueryVersion, whereas the
// rest (functions1 - 22) need typedef and sz_ define's as they use
// common structures.

// X_SELinuxSetDeviceCreateContext = 1
typedef xSELinuxSetCreateContextReq xSELinuxSetDeviceCreateContextReq;
```

```
#define sz_xSELinuxSetDeviceCreateContextReq
    sz_xSELinuxSetCreateContextReq
// the context is in a char buffer that is sent as additional data by _XSend

// X_SELinuxGetDeviceCreateContext = 2
typedef xSELinuxGetCreateContextReq    xSELinuxGetDeviceCreateContextReq;
typedef xSELinuxGetContextReply        xSELinuxGetDeviceCreateContextReply;
#define sz_xSELinuxGetDeviceCreateContextReq
    sz_xSELinuxGetCreateContextReq
#define sz_xSELinuxGetDeviceCreateContextReply sz_xSELinuxGetContextReply

// X_SELinuxSetDeviceContext = 3
typedef xSELinuxSetContextReq          xSELinuxSetDeviceContextReq;
#define sz_xSELinuxSetDeviceContextReq    sz_xSELinuxSetContextReq

// X_SELinuxGetDeviceContext = 4
typedef xSELinuxGetContextReq          xSELinuxGetDeviceContextReq;
typedef xSELinuxGetContextReply        xSELinuxGetDeviceContextReply;
#define sz_xSELinuxGetDeviceContextReq    sz_xSELinuxGetContextReq
#define sz_xSELinuxGetDeviceContextReply  sz_xSELinuxGetContextReply

// X_SELinuxSetWindowCreateContext = 5
typedef xSELinuxSetCreateContextReq    xSELinuxSetWindowCreateContextReq;
#define sz_xSELinuxSetWindowCreateContextReq
    sz_xSELinuxSetCreateContextReq

// X_SELinuxGetWindowCreateContext = 6
typedef xSELinuxGetCreateContextReq    xSELinuxGetWindowCreateContextReq;
typedef xSELinuxGetContextReply        xSELinuxGetWindowCreateContextReply;
#define sz_xSELinuxGetWindowCreateContextReq
    sz_xSELinuxGetCreateContextReq
#define sz_xSELinuxGetWindowCreateContextReply sz_xSELinuxGetContextReply

// X_SELinuxGetWindowContext = 7
typedef xSELinuxGetContextReq          xSELinuxGetWindowContextReq;
typedef xSELinuxGetContextReply        xSELinuxGetWindowContextReply;
#define sz_xSELinuxGetWindowContextReq    sz_xSELinuxGetContextReq
#define sz_xSELinuxGetWindowContextReply  sz_xSELinuxGetContextReply

// X_SELinuxSetPropertyCreateContext = 8
typedef xSELinuxSetCreateContextReq    xSELinuxSetPropertyCreateContextReq;
#define sz_xSELinuxSetPropertyCreateContextReq
    sz_xSELinuxSetCreateContextReq

// X_SELinuxGetPropertyCreateContext = 9
typedef xSELinuxGetCreateContextReq    xSELinuxGetPropertyCreateContextReq;
typedef xSELinuxGetContextReply        xSELinuxGetPropertyCreateContextReply;
#define sz_xSELinuxGetPropertyCreateContextReq
    sz_xSELinuxGetCreateContextReq
#define sz_xSELinuxGetPropertyCreateContextReply
    sz_xSELinuxGetContextReply

// X_SELinuxSetPropertyUseContext = 10
typedef xSELinuxSetCreateContextReq    xSELinuxSetPropertyUseContextReq;
#define sz_xSELinuxSetPropertyUseContextReq sz_xSELinuxSetCreateContextReq

// X_SELinuxGetPropertyUseContext = 11
typedef xSELinuxGetCreateContextReq    xSELinuxGetPropertyUseContextReq;
typedef xSELinuxGetContextReply        xSELinuxGetPropertyUseContextReply;
#define sz_xSELinuxGetPropertyUseContextReq sz_xSELinuxGetCreateContextReq
#define sz_xSELinuxGetPropertyUseContextReply sz_xSELinuxGetContextReply

// X_SELinuxGetPropertyContext = 12 (the req struct has already been
declared)
// typedef xSELinuxGetPropertyContextReq    xSELinuxGetPropertyContextReq;
typedef xSELinuxGetContextReply        xSELinuxGetPropertyContextReply;
#define sz_xSELinuxGetPropertyContextReply sz_xSELinuxGetContextReply

// X_SELinuxGetPropertyDataContext = 13
typedef xSELinuxGetPropertyContextReq    xSELinuxGetPropertyDataContextReq;
typedef xSELinuxGetContextReply        xSELinuxGetPropertyDataContextReply;
```

```

#define sz_xSELinuxGetPropertyDataContextReq
      sz_xSELinuxGetPropertyContextReq
#define sz_xSELinuxGetPropertyDataContextReply sz_xSELinuxGetContextReply

// X_SELinuxListProperties = 14
typedef xSELinuxGetContextReq          xSELinuxListPropertiesReq;
typedef xSELinuxListItemsReply        xSELinuxListPropertiesReply;
#define sz_xSELinuxListPropertiesReq   sz_xSELinuxGetContextReq
#define sz_xSELinuxListPropertiesReply sz_xSELinuxListItemsReply

// X_SELinuxSetSelectionCreateContext = 15
typedef xSELinuxSetCreateContextReq    xSELinuxSetSelectionCreateContextReq;
#define sz_xSELinuxSetSelectionCreateContextReq
      sz_xSELinuxSetCreateContextReq

// X_SELinuxGetSelectionCreateContext = 16
typedef xSELinuxGetCreateContextReq    xSELinuxGetSelectionCreateContextReq;
typedef xSELinuxGetContextReply        xSELinuxGetSelectionCreateContextReply;
#define sz_xSELinuxGetSelectionCreateContextReq
      sz_xSELinuxGetCreateContextReq
#define sz_xSELinuxGetSelectionCreateContextReply
      sz_xSELinuxGetContextReply

// X_SELinuxSetSelectionUseContext = 17
typedef xSELinuxSetCreateContextReq    xSELinuxSetSelectionUseContextReq;
#define sz_xSELinuxSetSelectionUseContextReq
      sz_xSELinuxSetCreateContextReq

// X_SELinuxGetSelectionUseContext = 18
typedef xSELinuxGetCreateContextReq    xSELinuxGetSelectionUseContextReq;
typedef xSELinuxGetContextReply        xSELinuxGetSelectionUseContextReply;
#define sz_xSELinuxGetSelectionUseContextReq
      sz_xSELinuxGetCreateContextReq
#define sz_xSELinuxGetSelectionUseContextReply
      sz_xSELinuxGetContextReply

// X_SELinuxGetSelectionContext = 19
typedef xSELinuxGetContextReq          xSELinuxGetSelectionContextReq;
typedef xSELinuxGetContextReply        xSELinuxGetSelectionContextReply;
#define sz_xSELinuxGetSelectionContextReq
      sz_xSELinuxGetContextReq
#define sz_xSELinuxGetSelectionContextReply
      sz_xSELinuxGetContextReply

// X_SELinuxGetSelectionDataContext = 20
typedef xSELinuxGetContextReq          xSELinuxGetSelectionDataContextReq;
typedef xSELinuxGetContextReply        xSELinuxGetSelectionDataContextReply;
#define sz_xSELinuxGetSelectionDataContextReq
      sz_xSELinuxGetContextReq
#define sz_xSELinuxGetSelectionDataContextReply
      sz_xSELinuxGetContextReply

// X_SELinuxListSelections = 21
typedef xSELinuxGetCreateContextReq    xSELinuxListSelectionsReq;
typedef xSELinuxListItemsReply        xSELinuxListSelectionsReply;
#define sz_xSELinuxListSelectionsReq   sz_xSELinuxGetCreateContextReq
#define sz_xSELinuxListSelectionsReply
      sz_xSELinuxListItemsReply

// X_SELinuxGetClientContext = 22
typedef xSELinuxGetContextReq          xSELinuxGetClientContextReq;
typedef xSELinuxGetContextReply        xSELinuxGetClientContextReply;
#define sz_xSELinuxGetClientContextReq
      sz_xSELinuxGetContextReq
#define sz_xSELinuxGetClientContextReply
      sz_xSELinuxGetContextReply

```

3. In the `./notebook-source/x-windows/x-common` directory, produce the `XSELinuxOMFunctions.c` source file with the following entries:

```

/*****
/*
/* These are the XSELinux functions for the Notebook X-Windows demos.
/* They are used to retrieve contexts and add them as required for the
/* X-Windows Object Manager test examples.
*/

```

```
/* */
/* Note that the XError handling could be improved by using the */
/* appropriate X functions, however the current set-up seems to work ok. */
/* (even though a bit messy). */
/* */
/* Copyright (C) 2010 Richard Haines */
/* */
/* This program is free software: you can redistribute it and/or modify */
/* it under the terms of the GNU General Public License as published by */
/* the Free Software Foundation, either version 3 of the License, or */
/* (at your option) any later version. */
/* */
/* This program is distributed in the hope that it will be useful, */
/* but WITHOUT ANY WARRANTY; without even the implied warranty of */
/* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the */
/* GNU General Public License for more details. */
/* */
/* You should have received a copy of the GNU General Public License */
/* along with this program. If not, see <http://www.gnu.org/licenses/>. */
/* */
/*****/

#include <X11/Xlib.h>
#include <X11/Xlibint.h>

#include <errno.h>

#include <stdio.h>
#include "Xlib-selinux.h"

#define ENFORCING 1

// Declare the function that will plug the _XReply Error handler.
// This would normally be called first when sending X_SELinux..
// functions to the X-server. The CatchXErrorHandler() is then
// called meaning that the error message will be flagged twice.
// The catchXreplyErrorHandlerFlag stops reporting it for the
// second time.
int CatchXreplyErrorHandler (); // Declare the function

// Declare the function that will plug the XError handler when we use
// the X_SELinuxSet... functions to set a security context. Need to plug
// this or the default handler will just exit displaying the generic Xerror.
int CatchXErrorHandler ();

// This flag is used to indicate that an XError has already been detected
// and displayed using the CatchXreplyErrorHandler() function that will
// set it to xTrue. The CatchXErrorHandler() function will ignore the error
// then set this flag to xFalse.
static int catchXreplyErrorHandlerFlag = xFalse;

// Have this flag for functions 12, 13, 19, 20 & 22 to detect "Access
// Denied"
// errors. This is set to xTrue before the _XReply is actioned in these
// functions with the CatchXErrorHandler() and XReplyError() functions
// setting it to xFalse.
//
// An AVC message is generated for "Access Denied", however the 12, 13,
// 19, 20 & 22 functions return an errno with BadAlloc set (as failed
// to allocate a SID ???).
// I think an Access Denied should generate an XError of BadAccess (16).
//
// What seems to happen for Access Denied is that the functions 12, 13, 19,
// 20 & 22 _XReply fails via the "if (! _Xreply ...)" statement with -1
// (even though this has been plugged by the CatchXreplyErrorHandler).
// The failure will call the XReplyError() function that will check various
// flags such as "enforcing mode", then generate an "Access Denied" message
// if required.
//
static int checkAccessDeniedFlag = xFalse;

static const char *functionCodes [] = {
    "SELinuxQueryVersion (0)",
```

```
"SELinuxSetDeviceCreateContext (1)",
"SELinuxGetDeviceCreateContext (2)",
"SELinuxSetDeviceContext (3)",
"SELinuxGetDeviceContext (4)",
"SELinuxSetWindowCreateContext (5)",
"SELinuxGetWindowCreateContext (6)",
"SELinuxGetWindowContext (7)",
"SELinuxSetPropertyCreateContext (8)",
"SELinuxGetPropertyCreateContext (9)",
"SELinuxSetPropertyUseContext (10)",
"SELinuxGetPropertyUseContext (11)",
"SELinuxGetPropertyContext (12)",
"SELinuxGetPropertyDataContext (13)",
"SELinuxListProperties (14)",
"SELinuxSetSelectionCreateContext (15)",
"SELinuxGetSelectionCreateContext (16)",
"SELinuxSetSelectionUseContext (17)",
"SELinuxGetSelectionUseContext (18)",
"SELinuxGetSelectionContext (19)",
"SELinuxGetSelectionDataContext (20)",
"SELinuxListSelections (21)",
"SELinuxGetClientContext (22)"
};

// This is global and used by the various test prgrams
int X_SELinuxExtensionOpcode;

// This global handle will be used to output the information from the
// functions. It would be stdout or a file.
extern FILE *outputPtr;

// The CatchXErrorHandler function captures any errors when any of
// the XLib functions are called. This allows XErrors to be captured
// and displayed, then let the application continue to run (as the
// normal XError handler would just display the error and exit.
// Note that the errors are only displayed if they have not been
// previously displayed by the CatchXreplyErrorHandler() function.
//
int CatchXErrorHandler (Display *dpy, XErrorEvent *error)
{
//printf ("CatchXErrorHandler()\n");
  if (catchXreplyErrorHandlerFlag == xFalse) {
// Got here because of an XLib XError detected with no corresponding
_XReply
// error, so just display a general error message with all the gory details
// and then give some detail on ones that have been seen to give an idea why
// they failed (only seen invalid context so far when using the ..Set..
// functions with invalid context info).

    if (error->request_code == X_SELinuxExtensionOpcode) {
      switch (error->error_code) {

        case  BadValue: // 2
          fprintf (outputPtr, "%s returned BadValue - could be invalid
context\n", functionCodes [error->minor_code]);
          break;

        case  BadLength: // 16
          fprintf (outputPtr, "%s returned BadLength - could be context
= NULL\n", functionCodes [error->minor_code]);
          break;

        default:
          fprintf (outputPtr, "%s returned an XError: %d\n",
functionCodes [error->minor_code], error->error_code);
          break;
      }
    } else
      fprintf (outputPtr, "Detected XError: %d for Major opcode: %d
Minor opcode: %d With ResourceID: %d\n", error->error_code, error-
>request_code, error->minor_code, error->resourceid);
  }
  catchXreplyErrorHandlerFlag = xFalse;
  checkAccessDeniedFlag = xFalse;
}
```

```
    return 1;
}

// This function catches _XReply errors (but NOT the "if (! _XReply ..) -1
// failures) and displays a message. The CatchXErrorHandler() handler is
// always
// called next so the catchXreplyErrorHandlerFlag is set to xTrue so the
// error
// is not reported twice. The xError struct is defined in Xproto.h
//
int CatchXreplyErrorHandler (Display *dpy, xError *err, XExtCodes *codes,
int *ret_code)
{
//printf ("CatchXreplyErrorHandler()\n");
    catchXreplyErrorHandlerFlag = xTrue;

    fprintf (outputPtr, "The %s function returned an _XReply error:\n",
functionCodes [err->minorCode]);
    if (outputPtr != stdout)
        printf ("The %s function returned an _XReply error: %d\n",
functionCodes [err->minorCode], err->errorCode);

    switch (err->errorCode) {
    case    BadRequest: // 1
        fprintf (outputPtr, "BadRequest - An invalid SELinux function
call\n");
        break;

    case    BadValue: // 2
        fprintf (outputPtr, "BadValue - Lookup failed for resourceID: %d
(could also be invalid context)\n", err->resourceID);
        break;

    case    BadWindow: // 3
        fprintf (outputPtr, "BadWindow - Check WindowID: %d\n", err-
>resourceID);
        break;

    case    BadMatch: // 8
        fprintf (outputPtr, "BadMatch - Lookup failed for resourceID: %d\n",
err->resourceID);
        break;

    case    BadAccess: // 10
        fprintf (outputPtr, "BadAccess - Lookup failed for resourceID: %d\n",
err->errorCode, err->resourceID);
        break;

    case    BadAlloc: // 11
        fprintf (outputPtr, "BadAlloc - Generally allocation of resourceID:
%d\n", err->resourceID);
        break;

    case    BadLength: // 16
        fprintf (outputPtr, "BadLength - A context is the wrong length
(resourceID: %d)\n", err->resourceID);
        break;

    default:
        fprintf (outputPtr, "_XReply XError: %d ResourceID: %d\n", err-
>errorCode, err->resourceID);
        break;
    }
    return 0;
}

// Need this function as:
// Always seem to get the BadAlloc message (or is it EAGAIN ?) in errno when
// _XReply fails via the "if (! _Xreply ..)" statement with -1 even though
// this has been plugged by the CatchXreplyErrorHandler.
// However the only case where it seems to be relevant is when obtaining the
// security context of a client (func 22) or Atoms (funcs 12, 13, 19 & 20)
```

```
// when in enforcing mode and access is not allowed. Therefore have this bit
of
// code that checks various bits. See the checkAccessDeniedFlag comments
above.
//
int XReplyError (Display *dpy, int minorOpcode)
{
    if (errno != BadAlloc) { // or is it EAGAIN ???
        fprintf (outputPtr, "The %s function _XReply returned errno: %d\n",
functionCodes [minorOpcode], errno);
        if (outputPtr != stdout)
            printf ("The %s function _XReply returned errno: %d\n",
functionCodes [minorOpcode], errno);
        UnlockDisplay (dpy);
        SyncHandle ();
        return (0);
    }

// Use this to check the error return from an _XReply as BadAlloc means
access
// denied when in enforcing mode for functions 12, 13 & 22.

    if ((security_getenforce () == ENFORCING) && (checkAccessDeniedFlag ==
xTrue)) {
        fprintf (outputPtr, "The BadAlloc error in Enforcing Mode
means \"Access Denied\"\n");
        if (outputPtr != stdout)
            printf ("The BadAlloc error in Enforcing Mode means \"Access
Denied\"\n");
        UnlockDisplay (dpy);
        SyncHandle ();
        checkAccessDeniedFlag = xFalse;
        return (0);
    }
    else {
        //// To stop screen clutter don't display the error for errno == BadAlloc as
        //// already caught by the XError handler anyway.
        // fprintf (outputPtr, "The %s function _XReply returned errno: %d\n",
functionCodes [minorOpcode], errno);
        // if (outputPtr != stdout)
        //     printf ("The %s function _XReply returned errno: %d\n",
functionCodes [minorOpcode], errno);
        UnlockDisplay (dpy);
        SyncHandle ();
        return (0);
    }
}

/*****
/*
/* START XSELinux FUNCTIONS
/* These are the X_SELinux functions that are called by the various test
/* programs. This could be written as a library service with better
/* handling of the X_SELinuxExtensionOpcode and FILE *outputPtr stuff.
/*
/*
*****/

// SELinuxQueryVersion = 0 - This function has the wrong
// xSELinuxQueryVersionReq size. See Xlib-selinux.h
SELinuxQueryVersion (Display *dpy)
{
    xSELinuxQueryVersionReq *req;
    xSELinuxQueryVersionReply rep;

    LockDisplay (dpy);
    GetReq (SELinuxQueryVersion, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxQueryVersion;
    if (!_XReply (dpy, (xReply *)&rep, 0, xTrue)) {
        XReplyError (dpy, X_SELinuxQueryVersion);
        return (-1);
    }
    UnlockDisplay (dpy);
    SyncHandle ();
}
```

```
    fprintf (outputPtr, "SELinuxQueryVersion - Major Version: %d Minor
Version: %d\n", rep.server_major, rep.server_minor);
}

// Get the SELinuxSetDeviceCreateContext = 1
SELinuxSetDeviceCreateContext (Display *dpy, char * buffer)
{
xSELinuxSetDeviceCreateContextReq *req;
long nbytes;

    fprintf (outputPtr, "SELinuxSetDeviceCreateContext - Setting Context:
%s\n", buffer);
    LockDisplay (dpy);
    GetReq (SELinuxSetDeviceCreateContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxSetDeviceCreateContext;
    nbytes = req->context_len = strlen (buffer);
    req->length += (nbytes + 3) >> 2; /* round up to mult of 4 */
    _XSend (dpy, buffer, nbytes); // Can use _XSend or Data as no difference
//Data (dpy, buffer, nbytes);

// This function does not specify a reply, however we need to check that
// the context given is valid, This is done on the next X call, so using
// XSynchronize() to force a return (as _XFlush() does not do this as
// the buffer is clear anyway). If an error is detected, then it goes
// to our CaptureXErrorHandler() function that gives a message and
// allows the function to continue. I'm sure there is a better way
// to do this but !!!
//_XFlush (dpy);
    XSynchronize (dpy, xTrue);
    UnlockDisplay (dpy);
    SyncHandle ();
    XSynchronize (dpy, xFalse);
}

// SELinuxGetDeviceCreateContext = 2
SELinuxGetDeviceCreateContext (Display *dpy)
{
xSELinuxGetDeviceCreateContextReq *req;
xSELinuxGetDeviceCreateContextReply rep;
char deviceContext [100];

    LockDisplay (dpy);
    GetReq (SELinuxGetDeviceCreateContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxGetDeviceCreateContext;
    if (!_XReply (dpy, (xReply *)&rep, 0, xFalse)) {
        XReplyError (dpy, X_SELinuxGetDeviceCreateContext);
        return (-1);
    }

    if (rep.context_len == 0)
        fprintf (outputPtr, "SELinuxGetDeviceCreateContext - No Context
available\n");
    else {
        _XReadPad (dpy, deviceContext, rep.context_len);
        fprintf (outputPtr, "SELinuxGetDeviceCreateContext - Context: %s\n",
deviceContext);
    }
    UnlockDisplay (dpy);
    SyncHandle ();
}

// Get the SELinuxSetDeviceCreateContext = 3
SELinuxSetDeviceContext (Display *dpy, char * buffer, long device_id)
{
xSELinuxSetDeviceContextReq *req;
long nbytes;
```

```
    fprintf (outputPtr, "SELinuxSetDeviceContext - Setting Context: %s for
Device ID %d\n", buffer, device_id);
    LockDisplay (dpy);
    GetReq (SELinuxSetDeviceContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxSetDeviceContext;
    req->id = device_id;
    nbytes = req->context_len = strlen(buffer);
    req->length += (nbytes + 3) >> 2; /* round up to mult of 4 */
//_XSend (dpy, buffer, nbytes);
    Data (dpy, buffer, nbytes);

// See the SELinuxSetDeviceCreateContext() function for the reason why
// the XSynchronize() functions are called.
    XSynchronize (dpy, xTrue);
    UnlockDisplay (dpy);
    SyncHandle ();
    XSynchronize (dpy, xFalse);
}

// SELinuxGetDeviceContext = 4
SELinuxGetDeviceContext (Display *dpy, int deviceID)
{
    xSELinuxGetDeviceContextReq *req;
    xSELinuxGetDeviceContextReply rep;
    char deviceContext [100];

    LockDisplay (dpy);
    GetReq (SELinuxGetDeviceContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxGetDeviceContext;
    req->id = deviceID;
    if (!_XReply (dpy, (xReply *)&rep, 0, xFalse)) {
        XReplyError (dpy, X_SELinuxGetDeviceContext);
        return (-1);
    }

    if (rep.context_len == 0)
        fprintf (outputPtr, "SELinuxGetDeviceContext - No Context
available\n");
    else {
        _XReadPad (dpy, deviceContext, rep.context_len);
        fprintf (outputPtr, "SELinuxGetDeviceContext - DeviceID: %d\nDevice
Context: %s\n", deviceID, deviceContext);
    }
    UnlockDisplay (dpy);
    SyncHandle ();
}

// Get the SELinuxSetWindowCreateContext = 5
SELinuxSetWindowCreateContext (Display *dpy, char * buffer)
{
    xSELinuxSetWindowCreateContextReq *req;
    long nbytes;

    fprintf (outputPtr, "SELinuxSetWindowCreateContext - Setting Context:
%s\n", buffer);
    LockDisplay (dpy);
    GetReq (SELinuxSetWindowCreateContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxSetWindowCreateContext;
    nbytes = req->context_len = strlen(buffer);
    req->length += (nbytes + 3) >> 2; /* round up to mult of 4 */
    _XSend (dpy, buffer, nbytes);
//Data (dpy, buffer, nbytes);

// See the SELinuxSetDeviceCreateContext() function for the reason why
// the XSynchronize() functions are called.
    XSynchronize (dpy, xTrue);
    UnlockDisplay (dpy);
    SyncHandle ();
    XSynchronize (dpy, xFalse);
}
```

```
}

// SELinuxGetWindowCreateContext = 6
SELinuxGetWindowCreateContext (Display *dpy)
{
xSELinuxGetWindowCreateContextReq *req;
xSELinuxGetWindowCreateContextReply rep;
char windowContext [100];

    LockDisplay (dpy);
    GetReq (SELinuxGetWindowCreateContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxGetWindowCreateContext;
    if (!_XReply (dpy, (xReply *)&rep, 0, xFalse)) {
        XReplyError (dpy, X_SELinuxGetWindowCreateContext);
        return (-1);
    }

    if (rep.context_len == 0)
        fprintf (outputPtr, "SELinuxGetWindowCreateContext - No Context
available\n");
    else {
        _XReadPad (dpy, windowContext, rep.context_len);
        fprintf (outputPtr, "SELinuxGetWindowCreateContext - Context: %s\n",
windowContext);
    }
    UnlockDisplay (dpy);
    SyncHandle ();
}

// SELinuxGetWindowContext = 7
SELinuxGetWindowContext (Display *dpy, Window windowID)
{
xSELinuxGetWindowContextReq *req;
xSELinuxGetWindowContextReply rep;
char windowContext [100];

    LockDisplay (dpy);
    GetReq (SELinuxGetWindowContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxGetWindowContext;
    req->id = windowID;
    if (!_XReply (dpy, (xReply *)&rep, 0, xFalse)) {
        XReplyError (dpy, X_SELinuxGetWindowContext);
        return (-1);
    }

    if (rep.context_len == 0)
        fprintf (outputPtr, "SELinuxGetWindowContext - No Context
available\n");
    else {
        _XReadPad (dpy, windowContext, rep.context_len);
        fprintf (outputPtr, "SELinuxGetWindowContext - WinID: %d\nWindow
Context: %s\n", windowID, windowContext);
    }
    UnlockDisplay (dpy);
    SyncHandle ();
}

// Get the SELinuxSetPropertyCreateContext = 8
SELinuxSetPropertyCreateContext (Display *dpy, char * buffer)
{
xSELinuxSetPropertyCreateContextReq *req;
long nbytes;

    fprintf (outputPtr, "SELinuxSetPropertyCreateContext - Setting Context:
%s\n", buffer);
    LockDisplay (dpy);
    GetReq (SELinuxSetPropertyCreateContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
```

```
req->SELinuxReqType = X_SELinuxSetPropertyCreateContext;
nbytes = req->context_len = strlen(buffer);
req->length += (nbytes + 3) >> 2; /* round up to mult of 4 */
_XSend (dpy, buffer, nbytes);
//Data (dpy, buffer, nbytes);

// See the SELinuxSetDeviceCreateContext() function for the reason why
// the XSynchronize() functions are called.
XSynchronize (dpy, xTrue);
UnlockDisplay (dpy);
SyncHandle ();
XSynchronize (dpy, xFalse);
}

// SELinuxGetPropertyCreateContext = 9
SELinuxGetPropertyCreateContext (Display *dpy)
{
xSELinuxGetPropertyCreateContextReq *req;
xSELinuxGetPropertyCreateContextReply rep;
char propertyContext [100];

LockDisplay (dpy);
GetReq (SELinuxGetPropertyCreateContext, req);
req->reqType = X_SELinuxExtensionOpcode;
req->SELinuxReqType = X_SELinuxGetPropertyCreateContext;
if (!_XReply (dpy, (xReply *)&rep, 0, xFalse)) {
_XReplyError (dpy, X_SELinuxGetPropertyCreateContext);
return (-1);
}

if (rep.context_len == 0)
fprintf (outputPtr, "SELinuxGetPropertyCreateContext - No Context
available\n");
else {
_XReadPad (dpy, propertyContext, rep.context_len);
fprintf (outputPtr, "SELinuxGetPropertyCreateContext - Context: %s\n",
propertyContext);
}
UnlockDisplay (dpy);
SyncHandle ();
}

// Get the SELinuxSetPropertyUseContext = 10
SELinuxSetPropertyUseContext (Display *dpy, char * buffer)
{
xSELinuxSetPropertyUseContextReq *req;
long nbytes;

fprintf (outputPtr, "SELinuxSetPropertyUseContext - Setting Context:
%s\n", buffer);
LockDisplay (dpy);
GetReq (SELinuxSetPropertyUseContext, req);
req->reqType = X_SELinuxExtensionOpcode;
req->SELinuxReqType = X_SELinuxSetPropertyUseContext;
nbytes = req->context_len = strlen(buffer);
req->length += (nbytes + 3) >> 2; /* round up to mult of 4 */
//_XSend (dpy, buffer, nbytes);
Data (dpy, buffer, nbytes);

// See the SELinuxSetDeviceCreateContext() function for the reason why
// the XSynchronize() functions are called.
XSynchronize (dpy, xTrue);
UnlockDisplay (dpy);
SyncHandle ();
XSynchronize (dpy, xFalse);
}

// SELinuxGetPropertyUseContext = 11
SELinuxGetPropertyUseContext (Display *dpy)
```

```
{
xSELinuxGetPropertyUseContextReq *req;
xSELinuxGetPropertyUseContextReply rep;
char propertyContext [100];

    LockDisplay (dpy);
    GetReq (SELinuxGetPropertyUseContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxGetPropertyUseContext;
    if (!_XReply (dpy, (xReply *)&rep, 0, xFalse)) {
        XReplyError (dpy, X_SELinuxGetPropertyUseContext);
        return (-1);
    }

    if (rep.context_len == 0)
        fprintf (outputPtr, "SELinuxGetPropertyUseContext - No Context
available\n");
    else {
        _XReadPad (dpy, propertyContext, rep.context_len);
        fprintf (outputPtr, "SELinuxGetPropertyUseContext - Context: %s\n",
propertyContext);
    }
    UnlockDisplay (dpy);
    SyncHandle ();
}

// SELinuxGetPropertyContext = 12
SELinuxGetPropertyContext (Display *dpy, Window windowID, Atom propertyAtom)
{
xSELinuxGetPropertyContextReq *req;
xSELinuxGetPropertyContextReply rep;
char propertyContext [100];

    LockDisplay (dpy);
    GetReq (SELinuxGetPropertyContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxGetPropertyContext;
    req->property = propertyAtom;
    req->window = windowID;
    // Indicate that function 12, 13, 19, 20 or 22 are being called:
    checkAccessDeniedFlag = xTrue;
    if (!_XReply (dpy, (xReply *)&rep, 0, xFalse)) {
        XReplyError (dpy, X_SELinuxGetPropertyContext);
        return (-1);
    }

    _XReadPad (dpy, propertyContext, rep.context_len);
    fprintf (outputPtr, "SELinuxGetPropertyContext - WinID: %d Atom: %s
\nContext: %s\n", windowID, (XGetAtomName (dpy, propertyAtom)),
propertyContext);
    UnlockDisplay (dpy);
    SyncHandle ();
    checkAccessDeniedFlag = xFalse;
}

// SELinuxGetPropertyDataContext = 13
SELinuxGetPropertyDataContext (Display *dpy, Window windowID, Atom
propertyAtom)
{
xSELinuxGetPropertyDataContextReq *req;
xSELinuxGetPropertyDataContextReply rep;
char propertyContext [100];

    LockDisplay (dpy);
    GetReq (SELinuxGetPropertyDataContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxGetPropertyDataContext;
    req->property = propertyAtom;
    req->window = windowID;
    // Indicate that function 12, 13, 19, 20 or 22 are being called:
    checkAccessDeniedFlag = xTrue;
    if (!_XReply (dpy, (xReply *)&rep, 0, xFalse)) {
        XReplyError (dpy, X_SELinuxGetPropertyDataContext);
    }
}
```

```
        return (-1);
    }

    _XReadPad (dpy, propertyContext, rep.context_len);
    fprintf (outputPtr, "SELinuxGetPropertyDataContext - WinID: %d Atom: %s\nContext: %s\n", windowID, (XGetAtomName (dpy, propertyAtom)),
propertyContext);
    UnlockDisplay (dpy);
    SyncHandle ();
    checkAccessDeniedFlag = xFalse;
}

// SELinuxListProperties = 14
SELinuxListProperties (Display *dpy, Window windowID)
{
xSELinuxListPropertiesReq *req;
xSELinuxListPropertiesReply rep;
xSELinuxListItem xlist_item;
xSELinuxListItemEntry *info_struct = NULL;
char propertyList [100];
char context [100];
int i;

    LockDisplay (dpy);
    GetReq (SELinuxListProperties, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxListProperties;
    req->id = windowID;
    if (!XReply (dpy, (xReply *)&rep, 0, xFalse)) {
        XReplyError (dpy, X_SELinuxListProperties);
        return (-1);
    }

    info_struct = Xmalloc((sizeof (xSELinuxListItemEntry) * rep.length) +
sizeof (CARD32));

    if(rep.count) {
        for(i = 0; i < rep.count; i++) {

// Read initial info that has Atom name and context sizes:
            _XReadPad (dpy, (char *)&xlist_item, sz_xSELinuxListItem);

// Add these to the holding structure for later printing:
            info_struct[i].atom_name = (Atom)xlist_item.name;
            info_struct[i].object_context_len = xlist_item.object_context_len;

            info_struct[i].data_context_len = xlist_item.data_context_len;

// Read the object context string and copy to the holding structure:

            _XReadPad (dpy, propertyList, xlist_item.object_context_len);

            if (!strncpy ((char *)&info_struct[i].object_context,
propertyList, (int)xlist_item.object_context_len)) {
                UnlockDisplay (dpy);
                SyncHandle ();
                perror ("strncpy function failed");
                exit (1);
            }

// Read the data context string and copy to the holding structure:
            _XReadPad (dpy, propertyList, xlist_item.data_context_len);

            if (!strncpy ((char *)&info_struct[i].data_context, propertyList,
(int)xlist_item.data_context_len))
                fprintf (outputPtr, "strncpy function failed\n");
            }
        } else
            _XEatData(dpy, rep.length << 2);

    UnlockDisplay (dpy);
```

```
SyncHandle ();

// Done with _XRead processes, so need to print off the information.
fprintf (outputPtr, "SELinuxListProperties found %d properties for WinID:
%d\n", rep.count, windowID);
for (i = 0; i < rep.count; i++) {

    fprintf (outputPtr, "\nSELinuxListProperties (%d of %d) - Atom: %s\n",
i+1, rep.count, (XGetAtomName (dpy, (Atom)info_struct[i].atom_name)));
    fprintf (outputPtr, "Object Context: %s\n", (char
*)&info_struct[i].object_context);
    fprintf (outputPtr, "Data Context: %s\n", (char
*)&info_struct[i].data_context);
}
Xfree (info_struct);
}

// Get the SELinuxSetSelectionCreateContext = 15
SELinuxSetSelectionCreateContext (Display *dpy, char * buffer)
{
xSELinuxSetSelectionCreateContextReq *req;
long nbytes;

    fprintf (outputPtr, "SELinuxSetSelectionCreateContext - Setting Context:
%s\n", buffer);
    LockDisplay (dpy);
    GetReq (SELinuxSetSelectionCreateContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxSetSelectionCreateContext;
    nbytes = req->context_len = strlen (buffer);
    req->length += (nbytes + 3) >> 2; /* round up to mult of 4 */
//_XSend (dpy, buffer, nbytes);
    Data (dpy, buffer, nbytes);

// See the SELinuxSetDeviceCreateContext() function for the reason why
// the XSynchronize() functions are called.
XSynchronize (dpy, xTrue);
UnlockDisplay (dpy);
SyncHandle ();
XSynchronize (dpy, xFalse);
}

// SELinuxGetSelectionCreateContext = 16
SELinuxGetSelectionCreateContext (Display *dpy)
{
xSELinuxGetSelectionCreateContextReq *req;
xSELinuxGetSelectionCreateContextReply rep;
char selectionContext [100];

    LockDisplay (dpy);
    GetReq (SELinuxGetSelectionCreateContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxGetSelectionCreateContext;
    if (!_XReply (dpy, (xReply *)&rep, 0, xFalse)) {
        XReplyError (dpy, X_SELinuxGetSelectionCreateContext);
        return (-1);
    }

    if (rep.context_len == 0)
        fprintf (outputPtr, "SELinuxGetSelectionCreateContext - No Context
available\n");
    else {
        _XReadPad (dpy, selectionContext, rep.context_len);
        fprintf (outputPtr, "SELinuxGetSelectionCreateContext - Context:
%s\n", selectionContext);
    }
    UnlockDisplay (dpy);
    SyncHandle ();
}

// Get the SELinuxSetSelectionUseContext = 17
SELinuxSetSelectionUseContext (Display *dpy, char * buffer)
{
xSELinuxSetSelectionUseContextReq *req;
```

```
long nbytes;

    fprintf (outputPtr, "SELinuxSetSelectionUseContext - Setting Context:
%s\n", buffer);
    LockDisplay (dpy);
    GetReq (SELinuxSetSelectionUseContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxSetSelectionUseContext;
    nbytes = req->context_len = strlen (buffer);
    req->length += (nbytes + 3) >> 2; /* round up to mult of 4 */
//_XSend (dpy, buffer, nbytes);
    Data (dpy, buffer, nbytes);

// See the SELinuxSetDeviceCreateContext() function for the reason why
// the XSynchronize() functions are called.
    XSynchronize (dpy, xTrue);
    UnlockDisplay (dpy);
    SyncHandle ();
    XSynchronize (dpy, xFalse);
}

// Get the SELinuxGetSelectionUseContext = 18
SELinuxGetSelectionUseContext (Display *dpy)
{
    xSELinuxGetSelectionUseContextReq *req;
    xSELinuxGetSelectionUseContextReply rep;
    char selectionContext [100];

    LockDisplay (dpy);
    GetReq (SELinuxGetSelectionUseContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxGetSelectionUseContext;
    if (!_XReply (dpy, (xReply *)&rep, 0, xFalse)) {
        XReplyError (dpy, X_SELinuxGetSelectionUseContext);
        return (-1);
    }

    if (rep.context_len == 0)
        fprintf (outputPtr, "SELinuxGetSelectionUseContext - No Context
available\n");
    else {
        _XReadPad (dpy, selectionContext, rep.context_len);
        fprintf (outputPtr, "SELinuxGetSelectionUseContext - Context: %s\n",
selectionContext);
    }
    UnlockDisplay (dpy);
    SyncHandle ();
}

// Get the SELinuxGetSelectionContext = 19
SELinuxGetSelectionContext (Display *dpy, Atom selectionAtom)
{
    xSELinuxGetSelectionContextReq *req;
    xSELinuxGetSelectionContextReply rep;
    char selectionContext [100];

    LockDisplay (dpy);
    GetReq (SELinuxGetSelectionContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxGetSelectionContext;
    req->id = selectionAtom;
// Indicate that function 12, 13, 19, 20 or 22 are being called:
    checkAccessDeniedFlag = xTrue;
    if (!_XReply (dpy, (xReply *)&rep, 0, xFalse)) {
        XReplyError (dpy, X_SELinuxGetSelectionContext);
        return (-1);
    }

    _XReadPad (dpy, selectionContext, rep.context_len);
    fprintf (outputPtr, "SELinuxGetSelectionContext - Atom: %s\nContext:
%s\n", (XGetAtomName (dpy, selectionAtom)), selectionContext);
//fprintf (outputPtr, "This is the context for x_selection OBJECT\n");
    UnlockDisplay (dpy);
    SyncHandle ();
}
```

```
// Indicate that function 12, 13, 19, 20 or 22 are being called:
checkAccessDeniedFlag = xTrue;
}

// Get the SELinuxGetSelectionDataContext = 20
SELinuxGetSelectionDataContext (Display *dpy, Atom selectionAtom)
{
xSELinuxGetSelectionDataContextReq *req;
xSELinuxGetSelectionDataContextReply rep;
char selectionContext [100];

    LockDisplay (dpy);
    GetReq (SELinuxGetSelectionDataContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxGetSelectionDataContext;
    req->id = selectionAtom;
// Indicate that function 12, 13, 19, 20 or 22 are being called:
checkAccessDeniedFlag = xTrue;
if (!_XReply (dpy, (xReply *)&rep, 0, xFalse)) {
    XReplyError (dpy, X_SELinuxGetSelectionDataContext);
    return (-1);
}

    _XReadPad (dpy, selectionContext, rep.context_len);
    fprintf (outputPtr, "SELinuxGetSelectionDataContext - Atom: %s\nContext:
%s\n", (XGetAtomName (dpy, selectionAtom)), selectionContext);
//fprintf (outputPtr, "This is the context for x_application_data
OBJECT\n");
    UnlockDisplay (dpy);
    SyncHandle ();
// Indicate that function 12, 13, 19, 20 or 22 are being called:
checkAccessDeniedFlag = xTrue;
}

// Get the SELinuxListSelections = 21
SELinuxListSelections (Display *dpy)
{
xSELinuxListSelectionsReq *req;
xSELinuxListSelectionsReply rep;
xSELinuxListItem xlist_item;
xSELinuxListItemEntry *info_struct = NULL;
char selectionList [600];
char context [100];
int i;

    LockDisplay (dpy);
    GetReq (SELinuxListSelections, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxListSelections;
    if (!_XReply (dpy, (xReply *)&rep, 0, xFalse)) {
        XReplyError (dpy, X_SELinuxListSelections);
        return (-1);
    }

    info_struct = Xmalloc((sizeof (xSELinuxListItemEntry) * rep.length) +
sizeof (CARD32));

    if(rep.count) {
        for(i = 0; i < rep.count; i++) {
// Read initial info that has Atom name and context sizes:
        _XReadPad (dpy, (char *)&xlist_item, sz_xSELinuxListItem);

// Add these to the holding structure for later printing:
        info_struct[i].atom_name = (Atom)xlist_item.name;
        info_struct[i].object_context_len = xlist_item.object_context_len;

        info_struct[i].data_context_len = xlist_item.data_context_len;

// Read the object context string and copy to the holding structure:
        _XReadPad (dpy, selectionList, xlist_item.object_context_len);
```

```
        if (!strncpy ((char *)&info_struct[i].object_context,
selectionList, (int)xlist_item.object_context_len))
            fprintf (outputPtr, "strncpy function failed\n");

// Read the data context string and copy to the holding structure:
    _XReadPad (dpy, selectionList, xlist_item.data_context_len);

        if (!strncpy ((char *)&info_struct[i].data_context, selectionList,
(int)xlist_item.data_context_len))
            fprintf (outputPtr, "strncpy function failed\n");
    }
} else
    _XEatData(dpy, rep.length << 2);

UnlockDisplay (dpy);
SyncHandle ();

// Done with _XRead processes, so need to print off the information.
for (i = 0; i < rep.count; i++) {
    fprintf (outputPtr, "\nSELinuxListSelections (%d of %d) - Atom: %s\n",
i+1, rep.count, (XGetAtomName (dpy, (Atom)info_struct[i].atom_name)));

    fprintf (outputPtr, "Object Context: %s\n", (char
*)&info_struct[i].object_context);
    fprintf (outputPtr, "Data Context: %s\n", (char
*)&info_struct[i].data_context);
}
Xfree (info_struct);
}

//
// When reading a client context that this function does not have
// permission to read will not result in an error being detected by
// the _XReply CatchXreplyError() function. Instead the _XReply
// call will fail with 'false' and the XReplyError() function
// will then check the errno return as described in the XReplyError()
// function. Also see the "checkAccessDeniedFlag" comments as
// to why this flag is present.
//
// SELinuxGetClientContext = 22
SELinuxGetClientContext (Display *dpy, Window resourceID)
{
xSELinuxGetClientContextReq *req;
xSELinuxGetClientContextReply rep;
char resourceContext [100];

    LockDisplay (dpy);
    GetReq (SELinuxGetClientContext, req);
    req->reqType = X_SELinuxExtensionOpcode;
    req->SELinuxReqType = X_SELinuxGetClientContext;
    req->id = resourceID;
// Indicate that function 12, 13, 19, 20 or 22 are being called:
    checkAccessDeniedFlag = xTrue;
    if (!_XReply (dpy, (xReply *)&rep, 0, xFalse)) {
        XReplyError (dpy, X_SELinuxGetClientContext);
        return (-1);
    }

    if (rep.context_len == 0)
        fprintf (outputPtr, "SELinuxGetClientContext - No Context
available\n");
    else {
        _XReadPad (dpy, resourceContext, rep.context_len);
        fprintf (outputPtr, "SELinuxGetClientContext - WinID: %d\nClient
Context: %s\n", resourceID, resourceContext);
    }
    checkAccessDeniedFlag = xFalse;
    UnlockDisplay (dpy);
    SyncHandle ();
}
```

4. In the `./notebook-source/x-windows/x-select+paste` directory, produce the `X-select.c` source file with the following entries:

```
/*
 * This is the X-select application for the Notebook X-Windows demos.
 * It retrieves the contexts via the XSELinux OM and the selected item
 * (Hello World) is selected and given to the calling X-paste application.
 * Copyright (C) 2010 Richard Haines
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 */
#include <X11/Xlib.h>
#include <X11/Xatom.h>
#include <X11/Xutil.h>
#include <X11/extensions/XInput2.h>

#include <stdio.h>
#include <stdlib.h>

#include <selinux/selinux.h>

#define ENFORCING 1

// The Error handler functions are in XSELinuxOMFunctions.c
extern int CatchXErrorHandler ();

extern int CatchXreplyErrorHandler ();

//Hold the opcode from the XQueryExtension call in XSELinuxOMFunctions.c
extern int X_SELinuxExtensionOpcode;

// Set output to stdout, but allow output to a file with option 'o'
// Declared here as used by functions in XSELinuxOMFunctions.c to output
info
FILE *outputPtr;

int main (int argc, char **argv)
{
// SELinux Context
security_context_t domainContext;
int result;
unsigned char buffer [] = "Hello World";
int event, error;
Window Sown;
XEvent xevent;
XTextProperty windowName;
XSelectionRequestEvent *request_event;
char windowNameString [100] = "";
char *windowNamePtr;

// Set output to stdout but allow output to a file with command line option
outputPtr = stdout;

// Use argv [1] as the filename for output
if (argc == 2) {
if ((outputPtr = fopen (argv [1], "w")) == NULL) {
fprintf (stderr, "Cannot open output file %s\n", argv [1]);
outputPtr = stdout;
}
else
printf("\nOutput to file: %s\n", argv [1]);
}
}
```

## The SELinux Notebook - Sample Policy Source

```
// Check if enforcing or not
if (security_getenforce () == ENFORCING)
    fprintf (outputPtr, "SELinux is in Enforcing mode\n");
else
    fprintf (outputPtr, "SELinux is in Permissive mode\n");

// Get a display handle
Display *dpy = XOpenDisplay(NULL);

// Get the SELinux Extension opcode
if (!XQueryExtension (dpy, "SELinux", &X_SELinuxExtensionOpcode, &event,
&error)) {
    perror ("XSELinux extension not available");
    exit (1);
}
else {
    fprintf (outputPtr, "\n***** Display Initial Window Information
*****\n");
    fprintf (outputPtr, "\nXQueryExtension for XSELinux Extension -
Opcode: %d Events: %d Error: %d \n", X_SELinuxExtensionOpcode, event,
error);
}
// Have XSELinux Object Manager

// Set our own handler for errors as the default displays error and exits.
XSetErrorHandler (CatchXErrorHandler);

// Set our own handler for _XReply errors.
XExtCodes *codes = XInitExtension (dpy, "SELinux");
XSetError (dpy, codes->extension, CatchXreplyErrorHandler);

// Now open a window
Window w = XCreateSimpleWindow (dpy, DefaultRootWindow(dpy), 0, 0, 500,
50, 0, 0, 0);

// Get and print Client context information
if (result = getcon (&domainContext) < 0) {
    perror ("Could not get Client context");
    exit (1);
}
fprintf (outputPtr, "\nlibselenium getcon - Domain Context: %s for WinID:
%d\n", domainContext, w);
sprintf (windowNameString, "%s - %s", argv[0], domainContext);

// Get the SELinux OM Version
fprintf (outputPtr, "\nCalling SELinuxQueryVersion (0) for this
display:\n");
SELinuxQueryVersion (dpy);

fprintf (outputPtr, "\nCalling SELinuxListProperties (14) before Drawing
Window (WinID: %d):\n", w);
SELinuxListProperties (dpy, w);

// Show the app name and SELinux context in the Window
windowNamePtr = windowNameString;
if (XStringListToTextProperty ((char **) &windowNamePtr, 1, &windowName) ==
0) {
    perror ("Structure allocation for windowName failed");
    exit (1);
}
XSetWMProperties (dpy, w, &windowName, NULL, NULL, 0, NULL, NULL, NULL);

freecon (domainContext);
XSelectInput (dpy, w, StructureNotifyMask);
XMapWindow (dpy, w);
XFlush (dpy);

// This function is called to display the start-up context info.
DisplayInitialContextInfo (dpy, w);

// This function can be called to set any context info using the
// SELinuxSet... functions.
SetContextTest (dpy);

XFlush (dpy);
```

```
fflush (outputPtr);

// Set the selection owner to our wndow
XSetSelectionOwner (dpy, XA_PRIMARY, w, CurrentTime);

while (1) {
// Uncomment the XSetSelectionOwner here to ensure the app always selects
// XSetSelectionOwner (dpy, XA_PRIMARY, w, CurrentTime);
printf ("WinID: %d waiting for SelectionRequest event\n", (Window)w);
XNextEvent (dpy, &xevent);

    if (xevent.type == SelectionRequest) {
        request_event=&(xevent.xselectionrequest);
        printf ("Have SelectionRequest event\n");
// Comment out DisplaySelectionContextInfo to stop screen clutter:
        DisplaySelectionContextInfo (dpy, w);
        if (request_event->target == XA_STRING)
            XChangeProperty (dpy, request_event->requestor, request_event-
>property, XA_STRING, 8, PropModeReplace, buffer, sizeof (buffer));
    }
}

/*****
/*
/* These functions display or set (optional) information using the
/* XSELinux functions. They have been moved here to avoid cluttering the
/* X code that selects the data.
/*
/*
*****/

// This function is called to display the start-up context info for tests.
int DisplayInitialContextInfo (Display *dpy, Window w)
{
XIDeviceInfo *devices, device;
int ndevices, counter;
    fprintf (outputPtr, "\nCalling SELinuxGetWindowCreateContext (6) for this
display:\n");
    SELinuxGetWindowCreateContext (dpy);

    fprintf (outputPtr, "\nCalling SELinuxGetClientContext (22) for this
Resource (WinID: %d):\n", w);
    SELinuxGetClientContext (dpy, w);

    fprintf (outputPtr, "\nCalling SELinuxGetWindowContext (7) for this Window
(WinID: %d):\n", w);
    SELinuxGetWindowContext (dpy, w);

// Do Device stuff
    fprintf (outputPtr, "\nCalling SELinuxGetDeviceCreateContext (2) for this
display:\n");
    SELinuxGetDeviceCreateContext (dpy);

    fprintf (outputPtr, "\nCalling SELinuxGetDeviceContext (4) for this
display:\n", w);
    devices = XIQueryDevice(dpy, XIAllDevices, &ndevices);

    for (counter = 0; counter < ndevices; counter++) {
        device = devices [counter];
        fprintf (outputPtr, "\nDevice %s is a ", device.name);
        switch (device.use) {
            case XIMasterPointer: fprintf (outputPtr, "master pointer\n");
                break;
            case XIMasterKeyboard: fprintf (outputPtr, "master keyboard\n");
                break;
            case XISlavePointer: fprintf (outputPtr, "slave pointer\n");
                break;
            case XISlaveKeyboard: fprintf (outputPtr, "slave keyboard\n");
                break;
            case XIFloatingSlave: fprintf (outputPtr, "floating slave\n");
                break;
        }
        SELinuxGetDeviceContext (dpy, device.deviceid);
    }
}
```

```
XIFreeDeviceInfo (devices);

// Do Properties
fprintf (outputPtr, "\nCalling SELinuxGetPropertyCreateContext (9) this
display:\n");
SELinuxGetPropertyCreateContext (dpy);

fprintf (outputPtr, "\nCalling SELinuxGetPropertyUseContext (11) for this
display:\n");
SELinuxGetPropertyUseContext (dpy);

fprintf (outputPtr, "\nCalling SELinuxListProperties (14) after Drawing
Window (WinID: %d):\n", w);
SELinuxListProperties (dpy, w);

// Do Selections
fprintf (outputPtr, "\nCalling SELinuxGetSelectionCreateContext (16) for
this display:\n");
SELinuxGetSelectionCreateContext (dpy);

fprintf (outputPtr, "\nCalling SELinuxGetSelectionUseContext (18) for this
display:\n");
SELinuxGetSelectionUseContext (dpy);

fprintf (outputPtr, "\nCalling SELinuxListSelections (21) for this
display:\n");
SELinuxListSelections (dpy);
}

// This function is called to display the context info when selection made
for tests.
int DisplaySelectionContextInfo (Display *dpy, Window w)
{
    fprintf (outputPtr, "\n***** Display Selection Window Information
*****\n");

    // Call the SELinux extension codes For Selections - with XA_PRIMARY
    fprintf (outputPtr, "\nCalling SELinuxGetSelectionContext (19) with
XA_PRIMARY for this Window (WinID: %d)\n", w);
    SELinuxGetSelectionContext (dpy, XA_PRIMARY);

    fprintf (outputPtr, "\nCalling SELinuxGetSelectionDataContext (20) with
XA_PRIMARY for this Window (WinID: %d)\n", w);
    SELinuxGetSelectionDataContext (dpy, XA_PRIMARY);

    // Call the SELinux extension codes For Properties - with XA_WM_NAME
    fprintf (outputPtr, "\nCalling SELinuxGetPropertyContext (12) with WM_NAME
for this Window:\n");
    SELinuxGetPropertyContext (dpy, w, XA_WM_NAME);

    fprintf (outputPtr, "\nCalling SELinuxGetPropertyDataContext (13) with
WM_NAME for this Window:\n");
    SELinuxGetPropertyDataContext (dpy, w, XA_WM_NAME);

    fflush (outputPtr);
}

// This function can be called to set any context info for tests.
// NOTE - These are currently commented out
int SetContextTest (Display *dpy)
{
    /*
    fprintf (outputPtr, "\nAdding any compiled SELinuxSet... function context
entries\n");

    fprintf (outputPtr, "\nCalling SELinuxSetPropertyUseContext (10) for this
display\n");
    SELinuxSetPropertyUseContext (dpy, "user_u:object_r:unconfined_t");
    fprintf (outputPtr, "\nCalling SELinuxGetPropertyUseContext (11) for this
display:\n");
    SELinuxGetPropertyUseContext (dpy);

    fprintf (outputPtr, "\nEnd SELinuxSet... functions\n");
    */
}
```

```
*/  
}
```

5. In the `./notebook-source/x-windows/x-select+paste` directory, produce the `X-paste.c` source file with the following entries:

```
/*  
*/  
/* This is the X-paste application for the Notebook X-Windows demos. */  
/* It retrieves the contexts via the XSELinux OM and the selected item */  
/* from the X-select application. */  
/* */  
/* Copyright (C) 2010 Richard Haines */  
/* */  
/* This program is free software: you can redistribute it and/or modify */  
/* it under the terms of the GNU General Public License as published by */  
/* the Free Software Foundation, either version 3 of the License, or */  
/* (at your option) any later version. */  
/* */  
/* This program is distributed in the hope that it will be useful, */  
/* but WITHOUT ANY WARRANTY; without even the implied warranty of */  
/* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the */  
/* GNU General Public License for more details. */  
/* */  
/* You should have received a copy of the GNU General Public License */  
/* along with this program. If not, see <http://www.gnu.org/licenses/>. */  
/* */  
/*****  
  
#include <X11/Xlib.h>  
#include <X11/Xatom.h>  
#include <X11/Xlibint.h>  
#include <X11/Xutil.h>  
#include <X11/extensions/XInput2.h>  
  
#include <stdio.h>  
#include <stdlib.h>  
  
#include <selinux/selinux.h>  
  
#define ENFORCING 1  
  
// The Error handler functions are in XSELinuxOMFunctions.c  
extern int CatchXErrorHandler ();  
  
extern int CatchXreplyErrorHandler ();  
  
// SELinux Context stuff  
security_context_t domainContext;  
  
//Hold the opcode from the XQueryExtension call  
extern int X_SELinuxExtensionOpcode;  
  
// Set output to stdout, but allow output to a file with option 'o'  
// Declared here as used by functions in XSELinuxOMFunctions.c to output info  
FILE *outputPtr;  
  
int main (int argc, char **argv)  
{  
    int propertyFormat, result, counter;  
    unsigned long propertyItems, stringLength, holder;  
    char *selectedData;  
    int event, error;  
    XEvent xevent;  
    Atom propertyType;  
    Window Sown;  
    XTextProperty windowName;  
    char windowNameString [100] = " ";  
    char *windowNamePtr;  
  
    // Set output to stdout but allow output to a file with command line option  
    outputPtr = stdout;
```

```
// Use argv [1] as the filename for output
if (argc == 2) {
    if ((outputPtr = fopen (argv [1], "w")) == NULL) {
        fprintf (stderr, "Cannot open output file %s\n", argv [1]);
        outputPtr = stdout;
    }
    else
        printf("\nOutput to file: %s\n", argv [1]);
}

// Check if enforcing or not
if (security_getenforce () == ENFORCING)
    fprintf (outputPtr, "SELinux is in Enforcing mode\n");
else
    fprintf (outputPtr, "SELinux is in Permissive mode\n");

// Get a display handle
Display *dpy = XOpenDisplay(NULL);

// Get the SELinux Extension opcode
if (!XQueryExtension (dpy, "SELinux", &X_SELinuxExtensionOpcode, &event,
&error)) {
    perror ("XSELinux extension not available");
    exit (1);
}
else {
    fprintf (outputPtr, "\n***** Display Initial Window Information
*****\n");
    fprintf (outputPtr, "\nXQueryExtension for XSELinux Extension -
Opcode: %d Events: %d Error: %d \n", X_SELinuxExtensionOpcode, event, error);
}
// Have XSELinux Object Manager

// Set our own handler for errors as the default displays error and exits.
XSetErrorHandler (CatchXErrorHandler);

// Set our own handler for _XReply errors.
XExtCodes *codes = XInitExtension (dpy, "SELinux");
XSetError (dpy, codes->extension, CatchXreplyErrorHandler);

// Now open a window
Window w = XCreateSimpleWindow (dpy, DefaultRootWindow(dpy), 0, 0, 500,
50, 0, 0, 0);

// Get and print Client context information
if (result = getcon (&domainContext) < 0) {
    perror ("Could not get Client context");
    exit (1);
}
fprintf (outputPtr, "\nlibselinux getcon - Domain Context: %s for WinID:
%d\n", domainContext, w);
sprintf (windowNameString, "%s - %s", argv[0], domainContext);

// Get the SELinux OM Version
fprintf (outputPtr, "\nCalling SELinuxQueryVersion (0) for this
display:\n");
SELinuxQueryVersion (dpy);

fprintf (outputPtr, "\nCalling SELinuxListProperties (14) before Drawing
Window (WinID: %d):\n", w);
SELinuxListProperties (dpy, w);

// Show the app name and SELinux context in the Window
windowNamePtr = windowNameString;
if (XStringListToTextProperty ((char **) &windowNamePtr, 1, &windowName) ==
0) {
    perror ("Structure allocation for windowName failed");
    exit (1);
}
XSetWMProperties (dpy, w, &windowName, NULL, NULL, 0, NULL, NULL, NULL);

freecon (domainContext);
XSelectInput (dpy, w, StructureNotifyMask);
XMapWindow (dpy, w);
XFlush (dpy);
```

```
// This function is called to display the start-up context info.
DisplayInitialContextInfo (dpy, w);

// This function can be called to set any context info using the
// SELinuxSet... functions.
SetContextTest (dpy);

XFlush (dpy);
fflush (outputPtr);

XSelectInput (dpy, w, StructureNotifyMask+ExposureMask);

while (1) {

// XGetSelectionOwner returns the current owner of the Selection in Sown, or
None if no selections made
Sown = XGetSelectionOwner (dpy, XA_PRIMARY);
fprintf (outputPtr, "\nWaiting for Selection Owner\n");
if (outputPtr != stdout)
    printf("\nWaiting for Selection Owner\n");

    if (Sown != None) {
// XConvertSelection converts the specified selection to the specified target
type
        XConvertSelection (dpy,
            XA_PRIMARY, // Selection atom
            XA_STRING, // Target atom
            XA_STRING, // Property name
            Sown, // The current owner
            CurrentTime);

        XFlush (dpy);

// This function is called to display the context info when selection made
for tests.
        DisplaySelectionContextInfo (dpy, w, Sown);

        // Query how much data to retrieve
        result = XGetWindowProperty (dpy, Sown,
            XA_STRING, // Atom Property name
            0, 0, // offset & length set to zero as this is a
query
            0, // Do not delete
            XA_STRING, // For this test require a STRING
            &propertyType, // return property type
            &propertyFormat, // returned property format
            &propertyItems, // returned number items
            &stringLength, // number of bytes to read
            (unsigned char **)&selectedData); // Returned data

        if (result == Success) {
// Check if selected text is present (X-select running or if
in unconfined_t
// then could be any selected text)
            if (stringLength > 0) {
                fprintf (outputPtr, "\nThis WinID: %d has data selected by
WinID: %d\n",
                    (Window)w , (Window)Sown);
                if (outputPtr != stdout) {
                    printf("\nThis WinID: %d has data selected by WinID:
%d\n",
                        (Window)w , (Window)Sown);
                    fflush (outputPtr);
                }

                result = XGetWindowProperty (dpy, Sown, XA_STRING, 0,
stringLength,
                    0, AnyPropertyType, &propertyType,
                    &propertyFormat, &propertyItems,
                    &holder, (unsigned char **)&selectedData);
                if (result == Success) {
                    fprintf (outputPtr, "The data selected is \"%s\" with
Atom Name: %s and a length of %d bytes\n", selectedData, (XGetAtomName (dpy,
propertyType)), stringLength);
                }
            }
        }
    }
}
}
```

```
        XFree (selectedData);
    }
    else fprintf (outputPtr, "Failed to read selected data\n");
}
}
}
}
fflush (outputPtr);
sleep(2);
}
}

/*****
/*
/* These functions display or set (optional) information using the
/* XSELinux functions. They have been moved here to avoid cluttering the
/* code that pastes the data.
/*
/*
/*****/

// This function is called to display the start-up context info for tests.
int DisplayInitialContextInfo (Display *dpy, Window w)
{
    XIDeviceInfo *devices, device;
    int ndevices, counter;

    fprintf (outputPtr, "\nCalling SELinuxGetWindowCreateContext (6) for this
display:\n");
    SELinuxGetWindowCreateContext (dpy);

    fprintf (outputPtr, "\nCalling SELinuxGetClientContext (22) for this
Resource (WinID: %d):\n", w);
    SELinuxGetClientContext (dpy, w);

    fprintf (outputPtr, "\nCalling SELinuxGetWindowContext (7) for this Window
(WinID: %d):\n", w);
    SELinuxGetWindowContext (dpy, w);

    // Do Device stuff
    fprintf (outputPtr, "\nCalling SELinuxGetDeviceCreateContext (2) for this
display:\n");
    SELinuxGetDeviceCreateContext (dpy);

    fprintf (outputPtr, "\nCalling SELinuxGetDeviceContext (4) for this
display:\n", w);
    devices = XIQueryDevice(dpy, XIAllDevices, &ndevices);

    for (counter = 0; counter < ndevices; counter++) {
        device = devices [counter];
        fprintf (outputPtr, "\nDevice %s is a ", device.name);
        switch (device.use) {
            case XIMasterPointer: fprintf (outputPtr, "master pointer\n");
                break;
            case XIMasterKeyboard: fprintf (outputPtr, "master keyboard\n");
                break;
            case XISlavePointer: fprintf (outputPtr, "slave pointer\n");
                break;
            case XISlaveKeyboard: fprintf (outputPtr, "slave keyboard\n");
                break;
            case XIFloatingSlave: fprintf (outputPtr, "floating slave\n");
                break;
        }
        SELinuxGetDeviceContext (dpy, device.deviceid);
    }
    XIFreeDeviceInfo (devices);

    // Do Properties
    fprintf (outputPtr, "\nCalling SELinuxGetPropertyCreateContext (9) this
display:\n");
    SELinuxGetPropertyCreateContext (dpy);

    fprintf (outputPtr, "\nCalling SELinuxGetPropertyUseContext (11) for this
display:\n");
    SELinuxGetPropertyUseContext (dpy);
}
```

```
    fprintf (outputPtr, "\nCalling SELinuxListProperties (14) after Drawing
Window (WinID: %d):\n", w);
    SELinuxListProperties (dpy, w);

// Do Selections
    fprintf (outputPtr, "\nCalling SELinuxGetSelectionCreateContext (16) for
this display:\n");
    SELinuxGetSelectionCreateContext (dpy);

    fprintf (outputPtr, "\nCalling SELinuxGetSelectionUseContext (18) for this
display:\n");
    SELinuxGetSelectionUseContext (dpy);

    fprintf (outputPtr, "\nCalling SELinuxListSelections (21) for this
display:\n");
    SELinuxListSelections (dpy);
}

// This function is called to display the context info when selection
detected.
int DisplaySelectionContextInfo (Display *dpy, Window w, Window Sown)
{
    fprintf (outputPtr, "\n***** Have Selection Owner so display Window
*****\n");
    fprintf (outputPtr, "*****      Selection & Property Information
*****\n");

// Call the SELinux extension codes For Selections - with XA_PRIMARY)
    fprintf (outputPtr, "\nCalling SELinuxGetSelectionContext (19) with
XA_PRIMARY for Selection Owner Window (WinID: %d)\n", Sown);
    SELinuxGetSelectionContext (dpy, XA_PRIMARY);

    fprintf (outputPtr, "\nCalling SELinuxGetSelectionDataContext (20) with
XA_PRIMARY for Selection Owner Window (WinID: %d)\n", Sown);
    SELinuxGetSelectionDataContext (dpy, XA_PRIMARY);

    fprintf (outputPtr, "\nCalling SELinuxGetSelectionContext (19) with
XA_PRIMARY for this Window (WinID: %d)\n", w);
    SELinuxGetSelectionContext (dpy, XA_PRIMARY);

    fprintf (outputPtr, "\nCalling SELinuxGetSelectionDataContext (20) with
XA_PRIMARY for this Window (WinID: %d)\n", w);
    SELinuxGetSelectionDataContext (dpy, XA_PRIMARY);

// Call the SELinux extension codes For Properties - with XA_WM_NAME)
    fprintf (outputPtr, "\nCalling SELinuxGetPropertyContext (12) with WM_NAME
for Property Owner Window:\n");
    SELinuxGetPropertyContext (dpy, Sown, XA_WM_NAME);

    fprintf (outputPtr, "\nCalling SELinuxGetPropertyDataContext (13) with
WM_NAME for Property Owner Window:\n");
    SELinuxGetPropertyDataContext (dpy, Sown, XA_WM_NAME);

    fprintf (outputPtr, "\nCalling SELinuxGetPropertyContext (12) with WM_NAME
for this Window:\n");
    SELinuxGetPropertyContext (dpy, w, XA_WM_NAME);

    fprintf (outputPtr, "\nCalling SELinuxGetPropertyDataContext (13) with
WM_NAME for this Window:\n");
    SELinuxGetPropertyDataContext (dpy, w, XA_WM_NAME);
}

// This function can be called to set any context info for tests.
// NOTE - These are currently commented out
int SetContextTest (Display *dpy)
{
/*
    fprintf (outputPtr, "\nAdding any compiled SELinuxSet... function context
entries\n");

    fprintf (outputPtr, "\nEnd SELinuxSet... functions\n");
*/
}
```

- From the `./notebook-source/x-windows/x-select+paste` directory, compile and link the X-select and X-paste applications as follows:

```
gcc X-paste.c ../x-common/XSELinuxOMFunctions.c -o X-paste
-l selinux -l X11 -l Xi
```

```
gcc X-select.c ../x-common/XSELinuxOMFunctions.c -o X-select
-l selinux -l X11 -l Xi
```

- Copy the X-select and X-paste application binaries to `/usr/local/bin` as follows:

```
cp X-select /usr/local/bin
cp X-paste /usr/local/bin
```

- The applications can be tested by calling them from separate virtual terminals, although they will only be running in the `unconfined_t` domain as shown in [Figure 4.1](#) (until the policy module is built as described in the next section). Note that the `x_contexts` file loaded in the previous section is the standard (non-poly) version.

### 4.3.3 Building the X-select and X-paste Loadable Module

This loadable module is to enforce policy on the X-select and X-paste applications when they are run in the `x_select_paste_t` domain using the SELinux `runcon` commands as follows:

```
# Note the runcon commands would be run from different virtual
# terminals to activate and test the applications.

runcon -t x_select_paste_t X-select
runcon -t x_select_paste_t X-select
```

The policy has a `poly-selection` boolean that by default is set to `FALSE` and controls what policy rules are enforced depending on what version of the `x_contexts` file is loaded (although note that the boolean does NOT control what file is loaded, that is a user copy function):

- Testing the standard `x_contexts` file `poly-selection = FALSE`.
- Testing the polyinstantiated `x_contexts` file `poly-selection = TRUE`.

The [Testing Derived Labels](#) and [Testing Polyinstantiated Labels](#) sections run through a number of tests to check what happens with each setting.

To build the loadable module:

- In the `./notebook-source/x-windows/x-select+paste` directory, produce the `x_select_paste.conf` policy configuration file with the following entries:

```
module x_select_paste 1.0.0;
#
```

```
#####  
#  
# This Loadable Module will manage the X-select and X-paste apps using #  
# x_context entries supported by policy rules for testing two #  
# selection scenarios: #  
# #  
# 1) Adding a 'user' prefix to form a 'derived type' and using #  
# type_transition rules similar to the RefPolicy. This does not work #  
# as explained in the PROBLEM section. #  
# This is controlled by setting the "poly-selection" boolean to FALSE #  
# and copying the "x_contexts-file-with-new-labels" to x_contexts. #  
# #  
# 2) Using polyinstantiation and type_member rules. This works okay. #  
# This is controlled by setting the "poly-selection" boolean to TRUE #  
# and copying the "x_contexts-file-with-new-polylabels" to x_contexts. #  
# #  
# Note that additional rules have been added to allow the XSELinuxGET.. #  
# functions to query contexts etc. for the various windows. #  
# #  
# dontaudit rules have also been added to stop unconfined_t getting to #  
# the x_select_paste_t domain info. #  
# #  
# Scenario 1 PROBLEM: #  
# Cannot find a way to stop selections in unconfined_t being picked #  
# up by the X-paste application when running in the x_select_paste_t #  
# domain. For example run: #  
# runcon -t x_select_paste_t X-paste #  
# Then select some text in another window running under unconfined_t. #  
# #  
# The problem seems to revolve around primary_xselection_t that allows all #  
# selections to be seen and used as the object type_transition rule has no #  
# effect at all. #  
# #  
# It seems that using polyinstantiation for selections is the only option #  
# that works (or do you know better !!! - Also tried using #  
# "constrain x_selection ..", however could not get this to work either. #  
# #  
#####  
#  
require {  
    type unconfined_t;  
    role unconfined_r;  
  
# Event types required from the x_contexts file:  
    type x11_destroynotify_xevent_t, x11_propertynotify_xevent_t;  
    type x11_confignotify_xevent_t, x11_externotify_xevent_t;  
    type x11_focusout_xevent_t, x11_focusedin_xevent_t;  
    type x11_mapnotify_xevent_t, x11_reparentnotify_xevent_t;  
    type x11_expose_xevent_t, x11_leavenotify_xevent_t;  
    type x11_selectionnotify_xevent_t, x11_unmapnotify_xevent_t;  
    type x11_selectionrequest_xevent_t;  
  
# Extension types required from the x_contexts file:  
    type big-requests_xextension_t, xkeyboard_xextension_t;  
    type selinux_xextension_t, xinputextension_xextension_t;  
    type undefined_xextension_t;  
  
# Property types required from the x_contexts file:  
    type wm_name_xproperty_t, string_xproperty_t;  
    type wm_class_xproperty_t, wm_client_machine_xproperty_t;  
    type wm_command_xproperty_t, wm_hints_xproperty_t;  
    type wm_normal_hints_xproperty_t;  
    type undefined_xproperty_t;  
    type resource_manager_xproperty_t;  
  
# Selection types required from the x_contexts file:  
    type primary_xselection_t, undefined_xselection_t;  
  
    class x_property { create read write getattr };  
    class x_selection { read getattr setattr };  
    class x_extension { query use };  
    class x_event { send receive };  
    class x_synthetic_event { send receive };
```

```
class x_drawable { read get_property getattr send list_property setattr
show receive set_property create manage add_child list_child blend };
class x_gc { create setattr };
class x_keyboard { read getattr use getfocus };
class x_resource { read };
class x_client { getattr };
class x_pointer { getattr read };

class file {read entrypoint getattr execute write execute_no_trans
create };
class process { transition siginh signal rlimitinh noatsecure sigchld };
class dir { search getattr write add_name };
class fd { use };
class chr_file { read write getattr };
class lnk_file { read };
class filesystem getattr;

class unix_stream_socket { create connect connectto read write getattr };
class security { check_context };
class fifo_file { read };
}

# These type entries have not been allocated any allow rules as they are not
# used (although I thought they would be !!). They were flagged by sechecker
# and have been left in for reference only:
# user_primary_xselection_t, user_wm_command_xproperty_t,
# user_x11_selectionnotify_xevent_t, user_wm_class_xproperty_t,
# user_wm_hints_xproperty_t, user_x11_selectionrequest_xevent_t,
# user_wm_normal_hints_xproperty_t, user_undefined_xselection_t
#

# Have a boolean to set either derived selections (false) that do not seem
# to work at all, or set selections using polyinstantiation that requires
# a type_member statement and poly_primary entry in the x_contexts file.
# Polyinstantiation works fine.
#
bool poly-selection false;

# The domain is x_select_paste_t
type x_select_paste_t;

##### Start Derived type entries #####
# Derive a specific 'type' by adding a 'prefix'. In this case 'user'.
# A derived type will be required for each entry in the x_context file
# that the application will need to 'use'. The derived type will then
# need a 'type_transition' for the object.
#
# Event types required from the x_contexts file:
type user_x11_destroynotify_xevent_t;
type user_x11_propertynotify_xevent_t;
type user_x11_confignotify_xevent_t;
type user_x11_enternotify_xevent_t;
type user_x11_focusout_xevent_t;
type user_x11_focussin_xevent_t;
type user_x11_mapnotify_xevent_t;
type user_x11_reparentnotify_xevent_t;
type user_x11_expose_xevent_t;
type user_x11_leavenotify_xevent_t;
type user_x11_selectionnotify_xevent_t;
type user_x11_selectionrequest_xevent_t;
type user_x11_unmapnotify_xevent_t;

## NO derived x_extension types are used.

# Property types required from the x_contexts file:
type user_undefined_xproperty_t;
type user_wm_name_xproperty_t;
type user_string_xproperty_t;
type user_wm_class_xproperty_t;
type user_wm_client_machine_xproperty_t;
type user_wm_command_xproperty_t;
type user_wm_hints_xproperty_t;
type user_wm_normal_hints_xproperty_t;

# Selection types required from the x_contexts file:
```

```
type user_primary_xselection_t;
type user_undefined_xselection_t;
##### End Derived type entries #####

# Allow executable to move into the x_select_paste_t domain
# using runcon for the type transition:
role unconfined_r types { x_select_paste_t };
allow x_select_paste_t unconfined_t : file entrypoint;
allow unconfined_t x_select_paste_t : process transition;

#
##### Start object type_transition for derived types #####
#
# Need type_transition entry for each of the derived type entries defined
# above. The format is as follows:
#type_transition <source_domain> <target_type> : <object_class>
<default_type>
#
# type_transition the x_drawable object to our domain:
type_transition x_select_paste_t unconfined_t : x_drawable x_select_paste_t;

# Event types required from the x_contexts file:
type_transition x_select_paste_t x11_destroynotify_xevent_t : x_event
user_x11_destroynotify_xevent_t;
type_transition x_select_paste_t x11_propertynotify_xevent_t : x_event
user_x11_propertynotify_xevent_t;
type_transition x_select_paste_t x11_confignotify_xevent_t : x_event
user_x11_confignotify_xevent_t;
type_transition x_select_paste_t x11_enternotify_xevent_t : x_event
user_x11_enternotify_xevent_t;
type_transition x_select_paste_t x11_focusout_xevent_t : x_event
user_x11_focusout_xevent_t;
type_transition x_select_paste_t x11_focussin_xevent_t : x_event
user_x11_focussin_xevent_t;
type_transition x_select_paste_t x11_mapnotify_xevent_t : x_event
user_x11_mapnotify_xevent_t;
type_transition x_select_paste_t x11_reparentnotify_xevent_t : x_event
user_x11_reparentnotify_xevent_t;
type_transition x_select_paste_t x11_expose_xevent_t : x_event
user_x11_expose_xevent_t;
type_transition x_select_paste_t x11_leavenotify_xevent_t : x_event
user_x11_leavenotify_xevent_t;
type_transition x_select_paste_t x11_selectionnotify_xevent_t : x_event
user_x11_selectionnotify_xevent_t;
type_transition x_select_paste_t x11_selectionrequest_xevent_t : x_event
user_x11_selectionrequest_xevent_t;
type_transition x_select_paste_t x11_unmapnotify_xevent_t : x_event
user_x11_unmapnotify_xevent_t;

# As each Window has its own properties it is important to make sure
# the undefined_xproperty_t is transitioned to the user domain:
type_transition x_select_paste_t undefined_xproperty_t : x_property
user_undefined_xproperty_t;

####
# These booleans are needed to allow the application name and context to
# be displayed in the title bar in the window when using polyinstantiated
# selections. Could not figure out how else to fix this !!
if (!poly-selection) {
# Don't transition this object if title bar info to be displayed when using
# polyinstantiated selections:
    type_transition x_select_paste_t wm_name_xproperty_t : x_property
user_wm_name_xproperty_t;
}
if (poly-selection) {
# Also need this to allow info to be displayed:
    allow x_select_paste_t wm_name_xproperty_t:x_property { write create };
}
####

type_transition x_select_paste_t string_xproperty_t : x_property
user_string_xproperty_t;
type_transition x_select_paste_t wm_class_xproperty_t : x_property
user_wm_class_xproperty_t;
```

```
type_transition x_select_paste_t wm_client_machine_xproperty_t : x_property
user_wm_client_machine_xproperty_t;
type_transition x_select_paste_t wm_command_xproperty_t : x_property
user_wm_command_xproperty_t;
type_transition x_select_paste_t wm_hints_xproperty_t : x_property
user_wm_hints_xproperty_t;
type_transition x_select_paste_t wm_normal_hints_xproperty_t : x_property
user_wm_normal_hints_xproperty_t;

# Selection types required from the x_contexts file:
# primary_xselection_t does not have any effect at all:
type_transition x_select_paste_t primary_xselection_t : x_selection
user_primary_xselection_t;
type_transition x_select_paste_t undefined_xselection_t : x_selection
user_undefined_xselection_t;
#
##### End object type_transition #####
#

#
### Boolean "poly-selection" set to "TRUE" for conditional policy rules ###
#
if (poly-selection) {

    # This type_member rules enforces polyinstantiation of the
    # "poly_selection PRIMARY primary_xselection t" x_contexts entry:
    type_member x_select_paste_t primary_xselection_t : x_selection
x_select_paste_t;
    # Additional allow rules:
    allow x_select_paste_t self:x_selection { getattr setattr read };

    # This one stops the title bar being displayed in the Window:
    # type_member x_select_paste_t user_wm_name_xproperty_t : x_property
x_select_paste_t;
    # allow x_select_paste_t self:x_property { write create };

}

#
##### End Boolean "poly-selection" conditional policy rules #####
#

#
##### Standard allow rules to display results, write logs etc. etc. #####
#
# Allow the test applications to write to log files:
allow x_select_paste_t unconfined_t : dir write;
allow x_select_paste_t unconfined_t : dir add_name;
allow x_select_paste_t unconfined_t : file create;

# Usual stuff for shared libraries, signals etc.
allow unconfined_t x_select_paste_t : dir search;
allow unconfined_t x_select_paste_t : process { siginh signal rlimitinh
noatsecure };
allow unconfined_t x_select_paste_t : file read;

allow x_select_paste_t unconfined_t : chr_file { read write getattr };
allow x_select_paste_t unconfined_t : dir { search getattr };
allow x_select_paste_t unconfined_t : fd use;
allow x_select_paste_t unconfined_t : process sigchld;
allow x_select_paste_t unconfined_t : file { read getattr execute };
allow x_select_paste_t unconfined_t : lnk_file read;
allow x_select_paste_t unconfined_t : unix_stream_socket connectto;
allow x_select_paste_t unconfined_t : filesystem getattr;
allow x_select_paste_t unconfined_t : file write;
allow x_select_paste_t unconfined_t : security check_context;

allow x_select_paste_t self : dir search;
allow x_select_paste_t self : file read;
allow x_select_paste_t self : unix_stream_socket { create connect getattr
read write };
allow x_select_paste_t self : process signal;

#
```

```
##### Start allow rules for derived objects #####
##### This first batch are for the X-select application #####
#
dontaudit unconfined_t x_select_paste_t : lnk_file read;
dontaudit unconfined_t x_select_paste_t : fd use;
dontaudit unconfined_t x_select_paste_t : fifo_file read;

allow x_select_paste_t unconfined_t : x_keyboard { getattr read };
allow x_select_paste_t unconfined_t : x_pointer { getattr read };
allow x_select_paste_t self : x_gc { create setattr };
allow x_select_paste_t self : x_resource read;

allow x_select_paste_t unconfined_t : x_drawable { get_property getattr
add_child };
allow x_select_paste_t self : x_drawable { create blend setattr receive
getattr set_property list_property show };

allow x_select_paste_t big-requests_xextension_t : x_extension { query use };
allow x_select_paste_t selinux_xextension_t : x_extension { query use };
allow x_select_paste_t xkeyboard_xextension_t : x_extension { query use };
allow x_select_paste_t xinputextension_xextension_t : x_extension { query use
};
allow x_select_paste_t undefined_xextension_t : x_extension { query use };

# Need this to select data but note it is not user_primary_xselection_t:
allow x_select_paste_t primary_xselection_t : x_selection setattr;

allow x_select_paste_t resource_manager_xproperty_t : x_property read;
allow x_select_paste_t user_undefined_xproperty_t : x_property { write create
};
allow x_select_paste_t user_wm_client_machine_xproperty_t : x_property
{ write create };
allow x_select_paste_t user_wm_name_xproperty_t : x_property { write
create };

allow x_select_paste_t user_x11_destroynotify_xevent_t : x_event receive;

#
##### These are for the X-paste application #####
#

allow x_select_paste_t self : x_drawable get_property;
allow x_select_paste_t self : x_client getattr;
allow x_select_paste_t unconfined_t : x_drawable setattr;

# Need this to read data but note it is not user_primary_xselection_t:
allow x_select_paste_t primary_xselection_t : x_selection { getattr read };

# Need this to allow the derived method to display the app name & context
# on title bar:
allow unconfined_t user_wm_name_xproperty_t : x_property read;
dontaudit unconfined_t user_wm_client_machine_xproperty_t : x_property read;

allow x_select_paste_t user_wm_name_xproperty_t : x_property getattr;
allow x_select_paste_t wm_name_xproperty_t : x_property getattr;
allow x_select_paste_t string_xproperty_t : x_property read;
allow x_select_paste_t user_string_xproperty_t : x_property { write create
read };

dontaudit unconfined_t user_x11_propertynotify_xevent_t : x_event receive;
dontaudit unconfined_t user_x11_reparentnotify_xevent_t : x_event receive;
dontaudit unconfined_t user_x11_confignotify_xevent_t : x_event receive;
dontaudit unconfined_t user_x11_confignotify_xevent_t : x_synthetic_event
{ send receive };
dontaudit unconfined_t user_x11_focusout_xevent_t : x_event receive;
dontaudit unconfined_t user_x11_focusin_xevent_t : x_event receive;
dontaudit unconfined_t user_x11_mapnotify_xevent_t : x_event receive;
dontaudit unconfined_t user_x11_unmapnotify_xevent_t : x_event receive;
dontaudit unconfined_t user_x11_enternotify_xevent_t : x_event receive;
dontaudit unconfined_t user_x11_leavenotify_xevent_t : x_event receive;
dontaudit unconfined_t user_x11_destroynotify_xevent_t : x_event receive;
dontaudit unconfined_t user_x11_expose_xevent_t : x_event receive;

allow x_select_paste_t user_x11_propertynotify_xevent_t : x_event receive;
allow x_select_paste_t user_x11_reparentnotify_xevent_t : x_event receive;
```

```
allow x_select_paste_t user_x11_confignotify_xevent_t : x_event receive;
allow x_select_paste_t user_x11_confignotify_xevent_t : x_synthetic_event
receive;
allow x_select_paste_t user_x11_focusout_xevent_t : x_event receive;
allow x_select_paste_t user_x11_foucusin_xevent_t : x_event receive;
allow x_select_paste_t user_x11_mapnotify_xevent_t : x_event receive;
allow x_select_paste_t user_x11_unmapnotify_xevent_t : x_event receive;
allow x_select_paste_t user_x11_externotify_xevent_t : x_event receive;
allow x_select_paste_t user_x11_expose_xevent_t : x_event receive;
allow x_select_paste_t user_x11_leavenotify_xevent_t : x_event receive;
#
##### End allow rules for derived objects #####
#

# This rule will allow the small X-window to be displayed. Change to the
# dontaudit rules to stop this being displayed.
allow unconfined_t x_select_paste_t : x_drawable { get_property receive
getattr manage set_property setattr send show read list_child };

#dontaudit unconfined_t x_select_paste_t : x_drawable { get_property receive
getattr manage set_property setattr send show read list_child };
```

2. Compile and load the policy module using the following SELinux commands:

```
checkmodule -m x_select_paste.conf -o x_select_paste.mod
semodule_package -o x_select_paste.pp -m x_select_paste.mod
semodule -v -s modular-test -i x_select_paste.pp
```

3. The policy modules loaded should now consist of the following:

```
semodule -l
x_context_base      1.0.0
x_select_paste      1.0.0
```

4. The system is now ready for testing various select / paste scenarios. Note that by default the poly-selection boolean is set to FALSE and the x\_contexts-file-with-new-labels file has been installed as the /etc/selinux/modular-test/contexts/x\_contexts file.

### 4.3.4 Testing Derived Labels

The following steps will determine if the test set-up is correct:

7. Check the correct modules are loaded by:

```
semodule -l
x_context_base      1.0.0
x_select_paste      1.0.0
```

8. Check the Boolean is set correctly by:

```
getsebool poly-selection
poly-selection --> off

# If 'on', then run:
setsebool -P poly-selection false
```

9. Ensure the correct x\_contexts file is installed. This can be done by checking that there are no poly\_ entries in the

/etc/selinux/modular-test/contexts/x\_contexts file. If the file is not correct, then copy the correct version over by:

```
cp $HOME/notebook-source/x-windows/x-contexts-base-module/x_contexts-file-with-new-labels /etc/selinux/modular-test/contexts/x_contexts
```

10. If the X-select and X-paste applications were not built as described in the [Building the X-select and X-paste Applications](#) section, then the executables can be copied from the ./notebook-source/x-windows/x\_select+paste directory to the /usr/local/bin directory. They should default to unconfined\_t that can be checked as follows:

```
ls -Z /usr/local/bin
-rwxr-xr-x. root root system_u:object_r:unconfined_t X-paste
-rwxr-xr-x. root root system_u:object_r:unconfined_t X-select
```

11. Open two virtual terminal sessions so that the applications can be run. A third can be opened for monitoring the audit log for errors.
12. Run `setenforce 1` for enforcing mode.

### Test 1:

The X-select and X-paste applications are called directly, one in each terminal session and will therefore run under the unconfined\_t domain:

Terminal 1: X-paste

Terminal 2: X-select

The results can be seen in [Figure 4.1](#) above where "Hello World" is displayed on Terminal 1 (note that if any text has been selected by another window, then that text will probably be displayed instead of "Hello World").

There is other information displayed that shows the various context information using the `SELinuxGet..` functions that can be examined if required.

To exit the applications 'Ctrl c' is used.

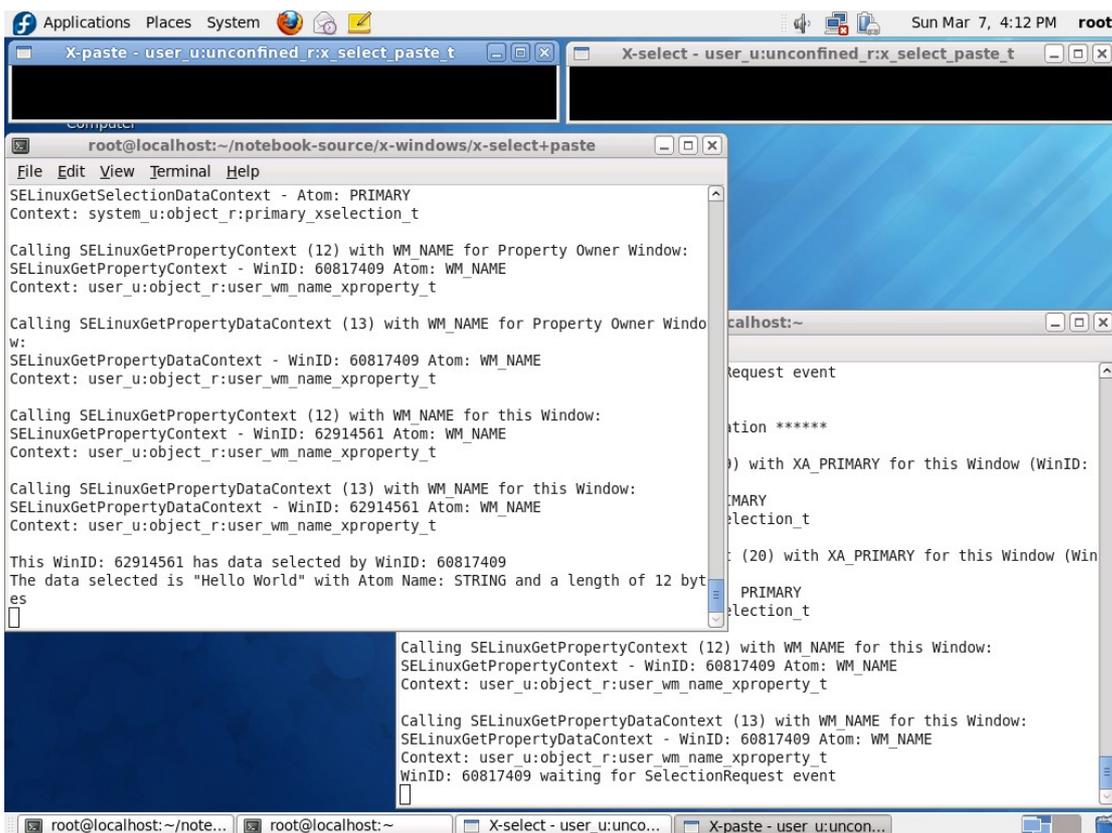
### Test 2:

The applications are then loaded using runcon:

Terminal 1: `runcon -t x_select_paste_t X-paste`

Terminal 2: `runcon -t x_select_paste_t X-select`

The results can be seen in [Figure 4.2](#) where "Hello World" is displayed on Terminal 1.



**Figure 4.2: Running Test 2 - With X-select and X-paste using standard x\_contexts entries in the x\_select\_paste\_t domain.**

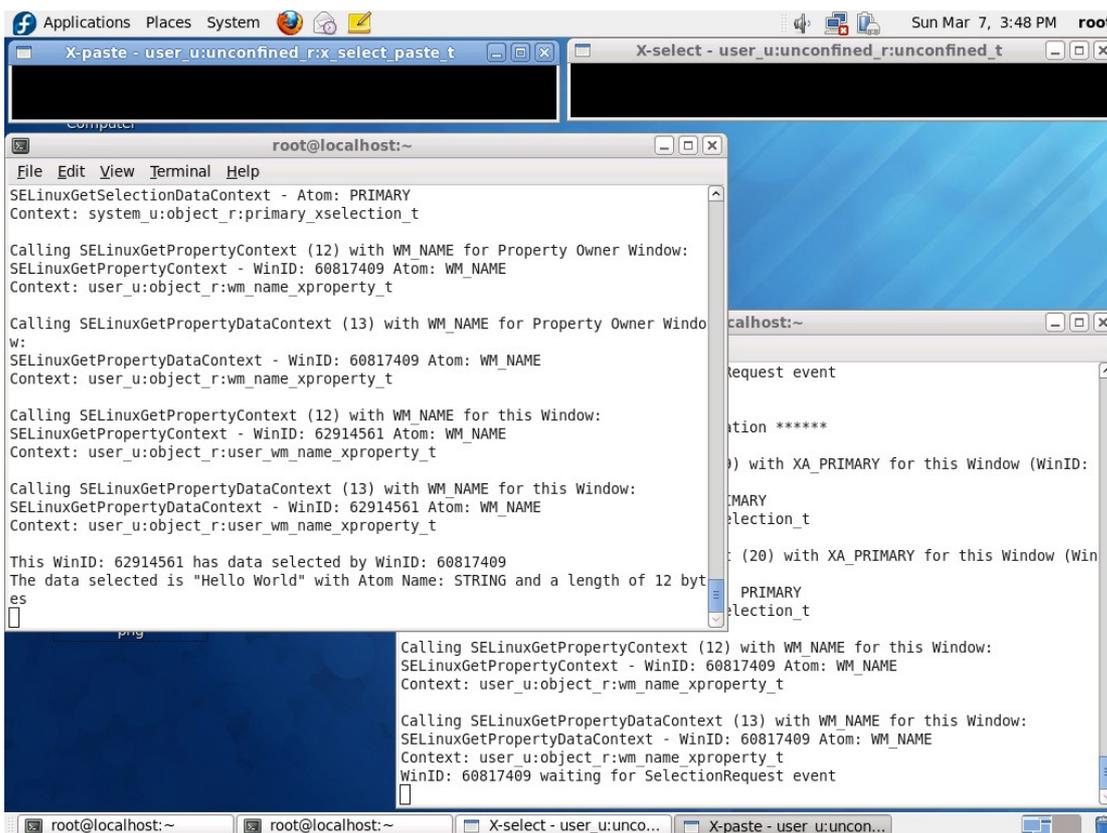
### Test 3:

The applications are then loaded as follows:

Terminal 1: `runcon -t x_select_paste_t X-paste`

Terminal 2: `X-select`

As shown in [Figure 4.3](#), the X-paste application still receives "Hello World", showing that selections are not blocked.



**Figure 4.3: Running Test 3 - With X-select (unconfined\_t) and X-paste (x\_select\_paste\_t) using standard x\_contexts entries.**

## Test 4:

With this test the poly-selection boolean is set to TRUE:

```
setsebool -P poly-selection true
```

The applications are then loaded as follows:

Terminal 1: `runcon -t x_select_paste_t X-paste`

Terminal 2: `X-select`

The results are the same as Test 3 in that “Hello World” is displayed.

### 4.3.4.1 Derived Object Test Conclusions

As can be seen the selected text can be pasted from both the `unconfined_t` and `x_select_paste_t` domains. This means that using standard reference policy type `x_contexts` file entries for selections, separation cannot be achieved (although note that the MLS version of reference policy may do - need to check one day).

If the policy is analysed, it will be seen that even though a type transition has been defined for the `primary_xselection_t` object:

```
# Extracts from the x_select_paste.conf policy:
```

```
# Added type with derived name to transition new object
instances:
type user_primary_xselection_t;

Added type transition for the object:
type_transition x_select_paste_t primary_xselection_t :
    x_selection user_primary_xselection_t;
```

a new object is never created:

```
# audit2allow never indicated that an allow rule was needed
# like this (that would be required if a new instance was
created)

allow x_select_paste_t user_primary_xselection_t : x_selection { read ... };
```

### 4.3.5 Testing Polyinstantiated Labels

The following steps will determine if the test set-up is correct:

1. Check the correct modules are loaded by:

```
semodule -l
x_context_base      1.0.0
x_select_paste      1.0.0
```

2. Check the Boolean is set correctly by:

```
getsebool poly-selection
poly-selection --> on

# If 'on', then run:
setsebool -P poly-selection true
```

3. Ensure the correct `x_contexts` file is installed. This can be done by checking that there are `poly_` entries in the `/etc/selinux/modular-test/contexts/x_contexts` file. If the file is not correct, then copy the correct version over by:

```
cp $HOME/notebook-source/x-windows/x-contexts-base-module/x_contexts-file-
with-new-polylabels /etc/selinux/modular-test/contexts/x_contexts
```

4. If the X-select and X-paste applications were not built as described in the [Building the X-select and X-paste Applications](#) section, then the executables can be copied from the `./notebook-source/x-windows/x_select+paste` directory to the `/usr/local/bin` directory. They should default to `unconfined_t` that can be checked as follows:

```
ls -Z /usr/local/bin
-rwxr-xr-x. root root system_u:object_r:unconfined_t X-paste
-rwxr-xr-x. root root system_u:object_r:unconfined_t X-select
```

5. Open two virtual terminal sessions so that the applications can be run. A third can be opened for monitoring the audit log for errors.
6. Run `setenforce 1` for enforcing mode.

### Test 1:

The X-select and X-paste applications are called directly, one in each terminal session and will therefore run under the `unconfined_t` domain:

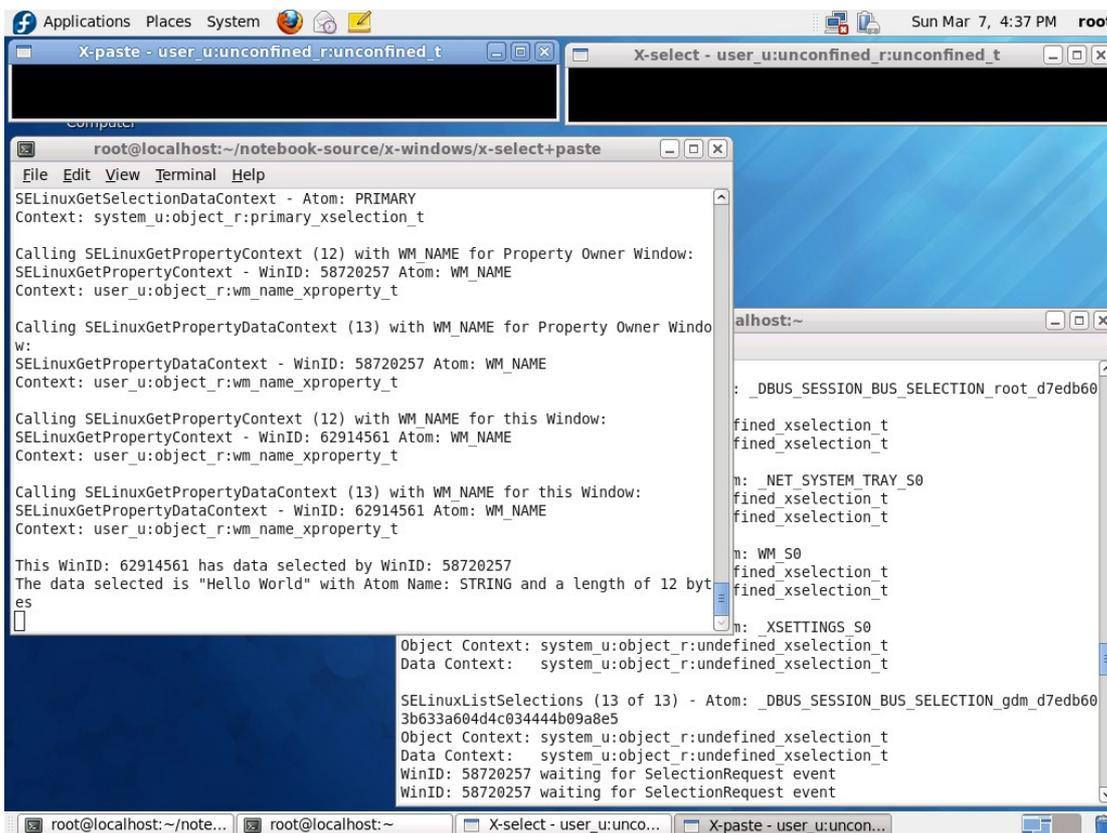
Terminal 1: X-paste

Terminal 2: X-select

The results can be seen in [Figure 4.4](#) where "Hello World" is displayed on Terminal 1 (note that if any text has been selected by another window, then that text will probably be displayed instead of "Hello World").

There is other information displayed that shows the various context information using the `SELinuxGet..` functions that can be examined if required.

To exit the applications 'Ctrl c' is used.



**Figure 4.4: Running Test 1 - With X-select and X-paste using polyinstantiated x\_contexts entries in the unconfined\_t domain.**

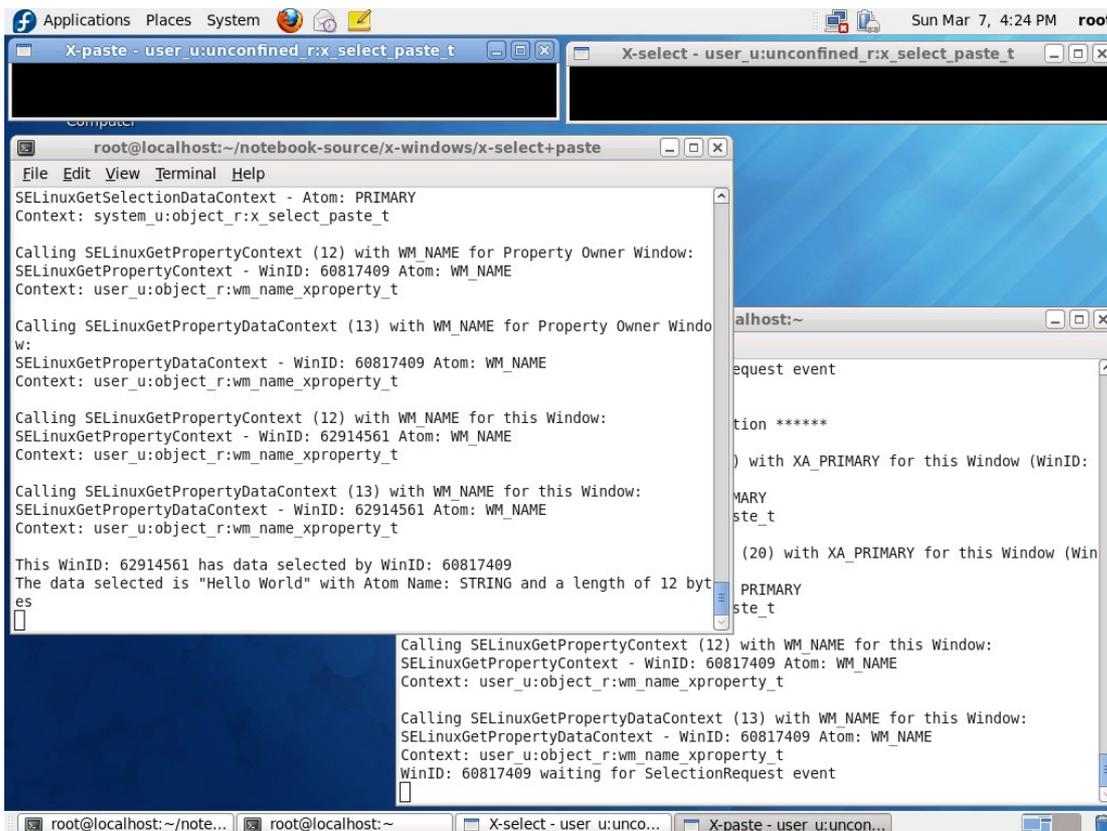
### Test 2:

The applications are then loaded using runcon:

Terminal 1: `runcon -t x_select_paste_t X-paste`

Terminal 2: `runcon -t x_select_paste_t X-select`

The results can be seen in where "Hello World" is displayed on Terminal 1.



**Figure 4.5: Running Test 2 - With X-select and X-paste using polyinstantiated x\_contexts entries in the x\_select\_paste\_t domain.**

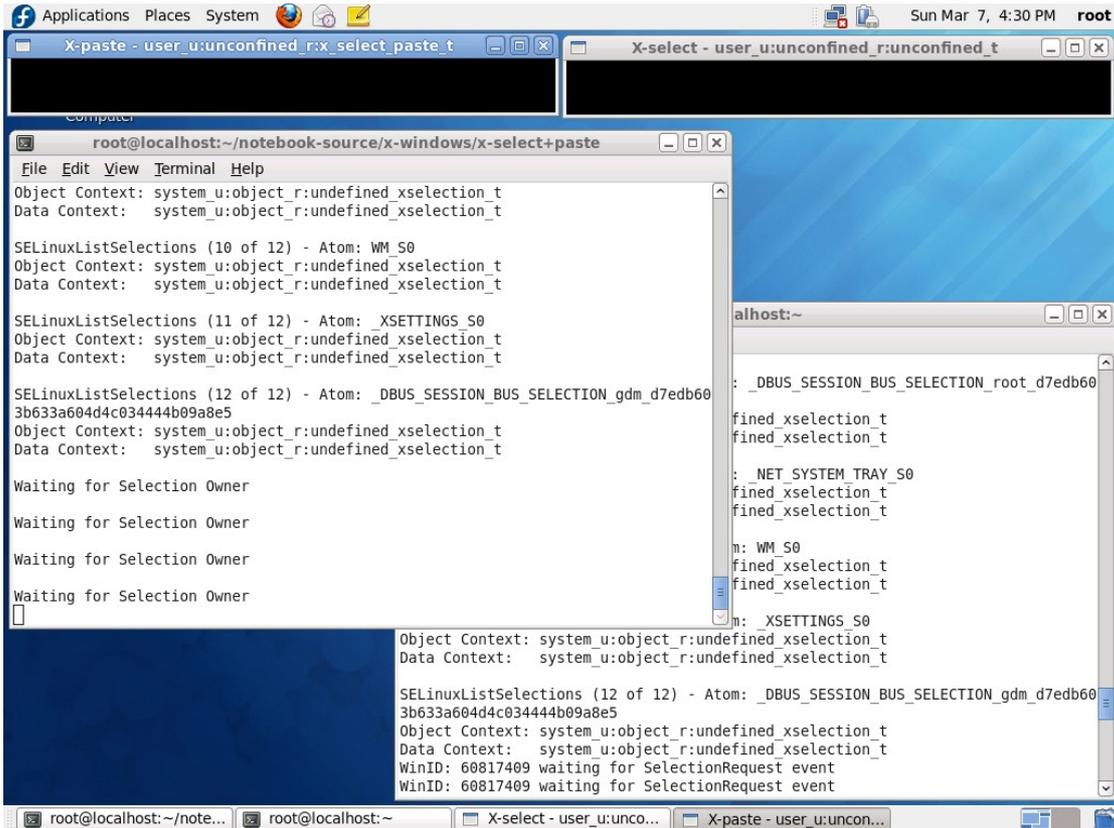
### Test 3:

The applications are then loaded as follows:

Terminal 1: `runcon -t x_select_paste_t X-paste`

Terminal 2: `X-select`

As shown in [Figure 4.6](#), the X-paste application does NOT receive "Hello World" as the selections are blocked by the polyinstantiation functionality.



**Figure 4.6: Running Test 3 - With X-select (unconfined\_t) and X-paste (x\_select\_paste\_t) using polyinstantiated x\_contexts entries.**

**Test 4:**

With this test the poly-selection boolean is set to FALSE:

```
setsebool -P poly-selection false
```

The applications are then loaded as follows:

Terminal 1: `runcon -t x_select_paste_t X-paste`

Terminal 2: `X-select`

As shown in [Figure 4.7](#), the X-paste application running on terminal 1 does not receive "Hello World" for the following reasons:

1. The selections are being detected by the X-paste application because the `type_member` rule has been disabled, therefore polyinstantiation is not being enforced by the policy (as to enforce polyinstantiation both the `poly_` entries in the `x_contexts` file is required plus a supporting `type_member` rule (and of course any allow rules)).
2. The application name and context is not displayed in the X-Window title bar and the terminal screen shows two error returns when getting the property context entries as shown below (the `resourceID: 39` is `WM_NAME` - see `Xatom.h`).

```
Calling SELinuxGetPropertyContext (12) with WM_NAME for Property Owner Window:
```

```
The SELinuxGetPropertyContext (12) function returned an _XReply error:
BadMatch - Lookup failed for resourceID: 39

Calling SELinuxGetPropertyDataContext (13) with WM_NAME for Property Owner Window:
The SELinuxGetPropertyDataContext (13) function returned an _XReply error:
BadMatch - Lookup failed for resourceID: 39
```

3. If `apol` is used to view the Conditional Expressions for the policy, the following will be seen:

```
conditional expression 1: [poly-selection]

TRUE list:
allow x_select_paste_t wm_name_xproperty_t : x_property { write create }; [Disabled]
allow x_select_paste_t x_select_paste_t : x_selection { getattr setattr read }; [Disabled]
type_member x_select_paste_t primary_xselection_t : x_selection x_select_paste_t; [Disabled]

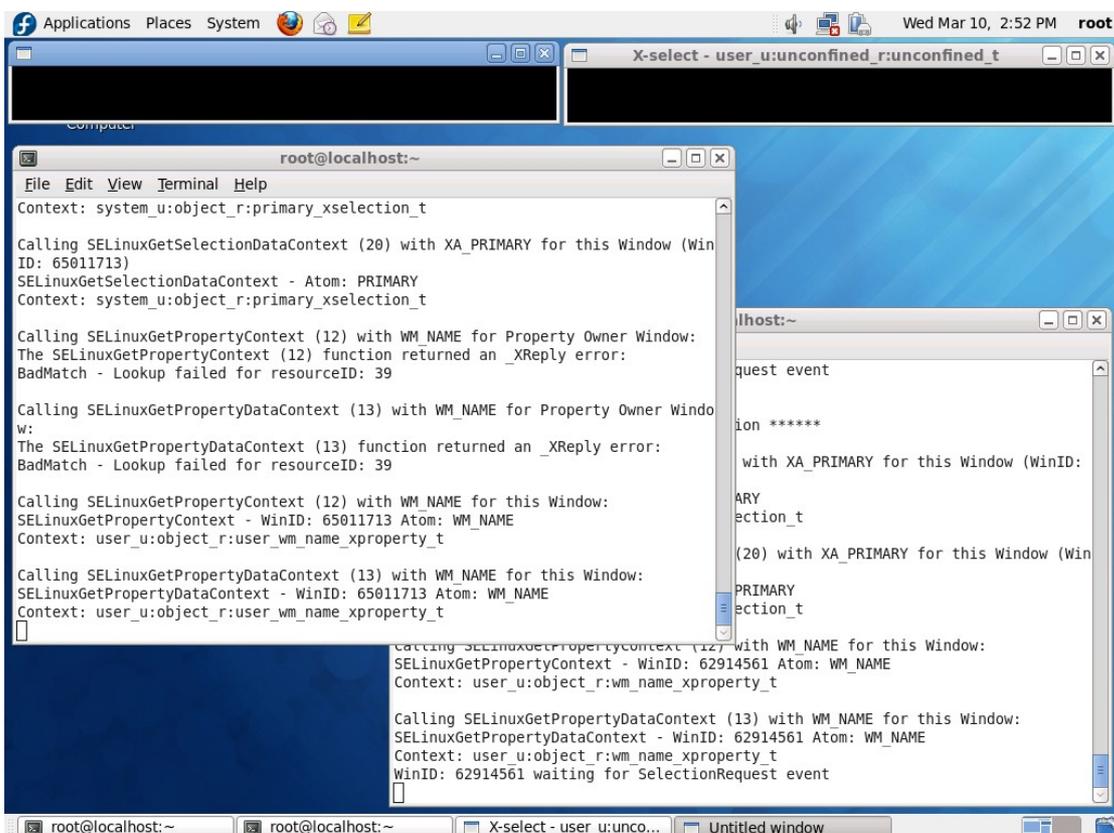
FALSE list:
type_transition x_select_paste_t wm_name_xproperty_t : x_property user_wm_name_xproperty_t;
```

Whereas, they should be:

```
conditional expression 1: [poly-selection]

TRUE list:
allow x_select_paste_t wm_name_xproperty_t : x_property { write create }; [Enabled]
allow x_select_paste_t x_select_paste_t : x_selection { getattr setattr read }; [Enabled]
type_member x_select_paste_t primary_xselection_t : x_selection x_select_paste_t; [Enabled]

FALSE list:
type_transition x_select_paste_t wm_name_xproperty_t : x_property user_wm_name_xproperty_t;
```



**Figure 4.7: Running Test 4 - With X-select (unconfined\_t) and X-paste (x\_select\_paste\_t) using polyinstantiated x\_contexts entries.**

### 4.3.5.1 Polyinstantiated Object Test Conclusions

As can be seen the selected text cannot be pasted between the `unconfined_t` and `x_select_paste_t` domains. This means that using polyinstantiated entries will allow selections to be isolated.

If the policy is analysed, it will be seen that the policy enforces the separation with a type member rule. The X-Windows object manager / XACE manages the actual selection polyinstantiation.

```
# Extracts from the x_select_paste.conf policy:

# This type_member rules enforces polyinstantiation of the
# "poly_selection PRIMARY primary_xselection_t" x_contexts entry:
type_member x_select_paste_t primary_xselection_t : x_selection
x_select_paste_t;

# Additional allow rules:
allow x_select_paste_t self:x_selection { getattr setattr read };
```

## 4.4 Building the XSELinux Function Test Application

The `x-setest` application allows a user to execute all of the SELinuxGet/Set.. functions that are integrated with the X-Windows object manager. The application is shown in [Figure 4.8](#) and should be easy to drive.



**Figure 4.8: x-setest menu** - Each XSELinux function can be executed and the domain context and window ID can be displayed.

This application does not require any specific policy module to run, however it will require permissions to be granted if you want to obtain information when running in other domains than the default. This has been tested with the Reference Policy once the X-windows object manager is running by setting the

xserver\_object\_manager boolean to TRUE. Important note - The new x\_keyboard and x\_pointer object classes and their permissions must be available. Red Hat F-12 builds from release selinux-policy-3.6.32-99.fc12 will have these added.

The [Calling the XSELinux Functions](#) section explains some of the issues around error handling and the source code has plenty of comments.

The functions 12, 13, 19, 20 & 22 return an XError of BadAlloc when access is denied and generates a USER\_AVC entry in the audit.log. Note however, XErrors are checked first and are not logged in audit.log, only USER\_AVC errors will be logged

When entering Atom names, the application will check if they are valid, however they are NOT checked to see if they are valid for the specific function (e.g. PRIMARY can be entered for a GetProperty... function, but it will fail with BadMatch).

Window and Resource IDs entered are not checked by the application and if incorrect the function will fail with BadMatch.

The 'o' option allows an output file to be specified to log the session, only minimum information is then displayed on the screen.

The application requires the following to be installed if recompiled:

- libX11, libX11-common, libX11-devel - These are standard Xlib packages.
- libXi, libXi-devel - These are required for retrieving Xdevice information.
- The XSELinuxOMFunctions.c and Xlib-selinux.h files that are located in the ./x-windows/x-common directory. The contents of these files are shown in the [Building the X-select and X-paste Applications](#) section.

The application source code is available at ./x-windows/x-setest/X-setest.c and is as follows:

```

/*****
/*
/* The XSELinux test application for the Notebook X-Windows demos.
/* It makes use of the functions in XSELinuxOMFunctions.c.
/* They are used to retrieve contexts and add them as required for the
/* X-Windows Object Manager test examples.
/*
/* Copyright (C) 2010 Richard Haines
/*
/* This program is free software: you can redistribute it and/or modify
/* it under the terms of the GNU General Public License as published by
/* the Free Software Foundation, either version 3 of the License, or
/* (at your option) any later version.
/*
/* This program is distributed in the hope that it will be useful,
/* but WITHOUT ANY WARRANTY; without even the implied warranty of
/* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
/* GNU General Public License for more details.
/*
/* You should have received a copy of the GNU General Public License
/* along with this program. If not, see <http://www.gnu.org/licenses/>.
/*
/*****
#include <X11/Xlib.h>

```

```
#include <X11/Xatom.h>
#include <X11/Xlibint.h>
#include <X11/Xutil.h>
#include <X11/extensions/XInput2.h>

#include <stdio.h>
#include <stdlib.h>

#include <selinux/selinux.h>
#include "../x-common/Xlib-selinux.h"

#define ENFORCING 1

// The Error handler functions are in XSELinuxOMFunctions.c
extern int CatchXErrorHandler ();

extern int CatchXreplyErrorHandler ();

//Hold the opcode from the XQueryExtension call in XSELinuxOMFunctions.c
extern int X_SELinuxExtensionOpcode;

// Set output to stdout, but allow output to a file with option 'o'
// Declared here as used by functions in XSELinuxOMFunctions.c to output info
FILE *outputPtr;

// Display the selection menu
int Menu ()
{
    printf ("\nXSELinux Functions:\n");
    printf ("0) QueryVersion\n");
    printf ("1) SetDeviceCreateContext (context) 2)
GetDeviceCreateContext\n");
    printf ("3) SetDeviceContext (device+context) 4) GetDeviceContext (lists
all)\n");
    printf ("5) SetWindowCreateContext (context) 6)
GetWindowCreateContext\n");
    printf ("7) GetWindowContext (win_id)\n");
    printf ("8) SetPropertyCreateContext (context) 9)
GetPropertyCreateContext\n");
    printf ("10) SetPropertyUseContext (context) 11)
GetPropertyUseContext\n");
    printf ("12) GetPropertyContext (win_id+atom) 13) GetPropertyDataContext
(win_id+atom)\n");
    printf ("14) ListProperties (win_id)\n");
    printf ("15) SetSelectionCreateContext (context) 16)
GetSelectionCreateContext\n");
    printf ("17) SetSelectionUseContext (context) 18)
GetSelectionUseContext\n");
    printf ("19) GetSelectionContext (atom) 20) GetSelectionDataContext
(atom)\n");
    printf ("21) ListSelections\n");
    printf ("22) GetClientContext (resource_id)\n");
    printf ("\n");
    printf ("d) Display domain context h) Help\n");
    printf ("m) Display menu o) Set output file\n");
    printf ("q) Quit w) Display windowID\n");
    return 0;
}

// Display info
int Help ()
{
    printf ("X-setest information:\n");
    printf ("The SELinux X-Windows Object Manager (XSelinux) has a number of
built-in\n");
    printf ("functions that can be called by SELinux-aware applications to Get..
and\n");
    printf ("Set.. information. This application will allow each of these
functions to be\n");
    printf ("called and display any information and/or error messages that are
generated.\n");

    printf ("\nThe functions 12, 13, 19, 20 & 22 return an XError of BadAlloc when
access\n");
    printf ("is denied and generates a USER_AVC entry in the audit.log.\n");
}
```

```
printf ("Note: XErrors are checked first and not logged in audit.log, only AVC
errors\n");
printf ("with entries such as \"for
request=SELinux:SELinuxGetClientContext\".\n");

printf ("\nWhen entering Atom names, they are checked for validity, however
they are NOT\n");
printf ("checked to see if they are valid for the specific function (e.g. you
can enter\n");
printf ("PRIMARY for a GetProperty... function, but it will fail with
BadMatch).\n");

printf ("\nNote that Window and Resource IDs entered are not checked by X-
setest and if\n");
printf ("incorrect the function will fail with BadMatch.\n");

printf ("\nThe 'o' option allows an output file to be specified to log the
session\n");
printf ("however, only minimum information is then displayed on the
screen.\n");
}

int main (int argc, char **argv)
{
char answer1 [80], answer2 [80], outFileNames [80], windowNameString [100] = " ";
int result, counter, ndevices;
unsigned long resourceID, deviceID;
int event, error, index;
security_context_t domainContext;
Atom atomName;
Window windowID;
XTextProperty windowName;
XDeviceInfo *devices, device;
char *windowNamePtr;

// Set output to stdout, but allow output to a file with option 'o'
outputPtr = stdout;

// Get a display handle
Display *dpy = XOpenDisplay (NULL);

// Get the SELinux Extension opcode
if (!XQueryExtension (dpy, "SELinux", &X_SELinuxExtensionOpcode, &event,
&error)) {
perror ("XSELinux extension not available");
exit (1);
}
else
printf ("\nXQueryExtension for XSELinux Extension - Opcode: %d Events: %d
Error: %d \n", X_SELinuxExtensionOpcode, event, error);
// Have XSELinux Object Manager

// Set our own handler for errors as the default displays error and exits.
XSetErrorHandler (CatchXErrorHandler);

// Set our own handler for _XReply errors.
XExtCodes *codes = XInitExtension (dpy, "SELinux");
XSetErrorHandler (dpy, codes->extension, CatchXreplyErrorHandler);

// Now open a window
Window w = XCreateSimpleWindow (dpy, DefaultRootWindow (dpy), 0, 0, 500, 50,
0, 0, 0);

// Get and print Client context information
if (result = getcon (&domainContext) < 0) {
perror ("Could not get Client context");
exit (1);
}
printf ("\nlibselinux getcon - Domain Context: %s for WinID: %d\n",
domainContext, w);
sprintf (windowNameString, "%s - %s", argv[0], domainContext);

// Show the app name and SELinux context in the Window
```

```
windowNamePtr = windowNameString;
if (XStringListToTextProperty((char **)&windowNamePtr, 1, &windowName) == 0) {
    perror ("Structure allocation for windowName failed");
    exit (1);
}
XSetWMProperties (dpy, w, &windowName, NULL, NULL, 0, NULL, NULL, NULL);

// freecon (domainContext);
XSelectInput (dpy, w, StructureNotifyMask);
XMapWindow (dpy, w);
XFlush (dpy);

// Display menu
Menu ();
// and wait for input
for (;;) {
    if (security_getenforce () == ENFORCING)
        printf ("SELinux is currently in Enforcing mode\n");
    else
        printf ("SELinux is currently in Permissive mode\n");

    printf ("\nSelect a function or \'m\' to redisplay the menu: ");
    fgets (answer1, sizeof(answer1), stdin);
    switch (answer1 [0]) {
    case 'd':
        fprintf (outputPtr, "This Domain Context: %s\n", domainContext);
        break;
    case 'h':
        Help ();
        break;
    case 'm':
        Menu ();
        break;
    case 'q':
        fflush (outputPtr);
        exit (0);
        break;
    case 'w':
        fprintf (outputPtr, "This WindowID: %d\n", w);
        break;
    case 'o':
        printf ("\nFilename for output or return for screen output: ");
        fgets (outFileName, sizeof(outFileName), stdin);
        outFileName [strlen (outFileName) - 1] = '\0';
        if (strlen (outFileName) == 0)
            outputPtr = stdout;
        else if ((outputPtr = fopen (outFileName, "w")) == NULL) {
            fprintf (stderr, "Cannot open output file %s\n", outFileName);
            outputPtr = stdout;
        }
        if (outputPtr != stdout)
            printf("\nOutput to file: %s\n", outFileName);

        if (security_getenforce () == ENFORCING)
            fprintf (outputPtr, "SELinux is in Enforcing mode\n");
        else
            fprintf (outputPtr, "SELinux is in Permissive mode\n");
        break;

    default:
        index = atoi (answer1);
        switch (index) {
        case 0:
            fprintf (outputPtr, "\nCalling SELinuxQueryVersion (0) for this
display:\n");
            SELinuxQueryVersion (dpy);
            break;
        case 1:
            // Get Context
            printf ("Enter Device Create Context: ");
            fgets (answer1, sizeof(answer1), stdin);
            // Remove cr
            answer1 [strlen (answer1) - 1] = 0;
            fprintf (outputPtr, "\nCalling SELinuxSetDeviceCreateContext (1)
for this display\n");
```

```
        SELinuxSetDeviceCreateContext (dpy, answer1);
        break;
    case 2:
        fprintf (outputPtr, "\nCalling SELinuxGetDeviceCreateContext (2)
for this display:\n");
        SELinuxGetDeviceCreateContext (dpy);
        break;
    case 3:
        // Get Context
        printf ("Enter Device Context: ");
        fgets (answer1, sizeof(answer1), stdin);
        answer1 [strlen (answer1) - 1] = 0;
        // Get Device ID
        printf("Enter Device ID: ");
        fgets (answer2, sizeof(answer2), stdin);
        deviceID = atoi (answer2);
        fprintf (outputPtr, "\nCalling SELinuxSetDeviceContext (3) for
this display\n");
        SELinuxSetDeviceContext (dpy, answer1, deviceID);
        break;
    case 4:
        fprintf (outputPtr, "\nCalling SELinuxGetDeviceContext (4) for
this display:\n", w);
        devices = XIQueryDevice(dpy, XIAllDevices, &ndevices);
        for (counter = 0; counter < ndevices; counter++) {
            device = devices[counter];
            fprintf (outputPtr, "\nDevice %s is a ", device.name);
            switch(device.use) {
                case XIMasterPointer:
                    fprintf (outputPtr, "master pointer\n");
                    break;
                case XIMasterKeyboard:
                    fprintf (outputPtr, "master keyboard\n");
                    break;
                case XISlavePointer:
                    fprintf (outputPtr, "slave pointer\n");
                    break;
                case XISlaveKeyboard:
                    fprintf (outputPtr, "slave keyboard\n");
                    break;
                case XIFloatingSlave:
                    fprintf (outputPtr, "floating slave\n");
                    break;
            }
            SELinuxGetDeviceContext (dpy, device.deviceid);
        }
        XIFreeDeviceInfo (devices);
        break;
    case 5:
        // Get Context
        printf ("Enter Window Create Context: ");
        fgets (answer1, sizeof(answer1), stdin);
        answer1 [strlen (answer1) - 1] = 0;
        fprintf (outputPtr, "\nCalling SELinuxSetWindowCreateContext (5)
for this display\n");
        SELinuxSetWindowCreateContext (dpy, answer1);
        break;
    case 6:
        fprintf (outputPtr, "\nCalling SELinuxGetWindowCreateContext (6)
for this display:\n");
        SELinuxGetWindowCreateContext (dpy);
        break;
    case 7:
        // Get WinID:
        printf ("Enter Window ID or return for this window: ");
        fgets (answer1, sizeof(answer1), stdin);
        answer1 [strlen (answer1) - 1] = 0;
        if (answer1 [0] == 0)
            windowID = w;
        else
            windowID = (atoi (answer1));

        fprintf (outputPtr, "\nCalling SELinuxGetWindowContext (7) for
Window (WinID: %d):\n", windowID);
        SELinuxGetWindowContext (dpy, windowID);
```

```
        break;
    case 8:
        // Get Context
        printf ("Enter Property Create Context: ");
        fgets (answer1, sizeof(answer1), stdin);
        answer1 [strlen (answer1) - 1] = 0;
        fprintf (outputPtr, "\nCalling SELinuxSetPropertyCreateContext (8)
for this display\n");
        SELinuxSetPropertyCreateContext (dpy, answer1);
        break;
    case 9:
        fprintf (outputPtr, "\nCalling SELinuxGetPropertyCreateContext (9)
this display:\n");
        SELinuxGetPropertyCreateContext (dpy);
        break;
    case 10:
        // Get Context
        printf ("Enter Property Use Context: ");
        fgets (answer1, sizeof(answer1), stdin);
        answer1 [strlen (answer1) - 1] = 0;
        fprintf (outputPtr, "\nCalling SELinuxSetPropertyUseContext (10)
for this display\n");
        SELinuxSetPropertyUseContext (dpy, answer1);
        break;
    case 11:
        fprintf (outputPtr, "\nCalling SELinuxGetPropertyUseContext (11)
for this display:\n");
        SELinuxGetPropertyUseContext (dpy);
        break;
    case 12:
        // Get WinID:
        printf ("Enter Window ID or return for this window: ");
        fgets (answer1, sizeof(answer1), stdin);
        answer1 [strlen (answer1) - 1] = 0;
        if (answer1 [0] == 0)
            windowID = w;
        else
            windowID = (atoi (answer1));

        // Get ATOM
        printf("Enter Property Atom Name: ");
        fgets (answer1, sizeof(answer1), stdin);
        answer1 [strlen (answer1) - 1] = 0;
        atomName = XInternAtom (dpy, answer1, xTrue);
        if (atomName == None) {
            printf ("Invalid Atom Name\n");
            break;
        }
        fprintf (outputPtr, "\nCalling SELinuxGetPropertyContext (12) with
%s for Window (WinID: %d):\n", (XGetAtomName (dpy, atomName)), windowID);
        SELinuxGetPropertyContext (dpy, windowID, atomName);
        break;
    case 13:
        // Get WinID:
        printf ("Enter Window ID or return for this window: ");
        fgets (answer1, sizeof(answer1), stdin);
        answer1 [strlen (answer1) - 1] = 0;
        if (answer1 [0] == 0)
            windowID = w;
        else
            windowID = (atoi (answer1));
        // Get ATOM
        printf ("Enter Property Atom Name: ");
        fgets (answer1, sizeof(answer1), stdin);
        answer1 [strlen (answer1) - 1] = 0;
        atomName = XInternAtom (dpy, answer1, xTrue);
        if (atomName == None) {
            printf ("Invalid Atom Name\n");
            break;
        }
        fprintf (outputPtr, "\nCalling SELinuxGetPropertyDataContext (13)
with %s for Window (WinID: %d):\n", (XGetAtomName (dpy, atomName)), windowID);
        SELinuxGetPropertyDataContext (dpy, windowID, atomName);
        break;
    case 14:
```

```
// Get WinID:
printf ("Enter Window ID or return for this window: ");
fgets (answer1, sizeof(answer1), stdin);
answer1 [strlen (answer1) - 1] = 0;
if (answer1 [0] == 0)
    windowID = w;
else
    windowID = (atoi (answer1));
fprintf (outputPtr, "\nCalling SELinuxListProperties (14) for
Window (WinID: %d):\n", windowID);
SELinuxListProperties (dpy, windowID);
break;
case 15:
    // Get Context
    printf ("Enter Selection Create Context: ");
    fgets (answer1, sizeof(answer1), stdin);
    answer1 [strlen(answer1) - 1] = 0;
    fprintf (outputPtr, "\nCalling SELinuxSetSelectionCreateContext
(15) for this display\n");
    SELinuxSetSelectionCreateContext (dpy, answer1);
    break;
case 16:
    fprintf (outputPtr, "\nCalling SELinuxGetSelectionCreateContext
(16) for this display:\n");
    SELinuxGetSelectionCreateContext (dpy);
    break;
case 17:
    // Get Context
    printf ("Enter Selection Use Context: ");
    fgets (answer1, sizeof(answer1), stdin);
    answer1 [strlen (answer1) - 1] = 0;
    fprintf (outputPtr, "\nCalling SELinuxSetSelectionUseContext (17)
for this display\n");
    SELinuxSetSelectionUseContext (dpy, answer1);
    break;
case 18:
    fprintf (outputPtr, "\nCalling SELinuxGetSelectionUseContext (18)
for this display:\n");
    SELinuxGetSelectionUseContext (dpy);
    break;
case 19:
    // Get ATOM
    printf ("Enter Selection Atom Name: ");
    fgets (answer1, sizeof(answer1), stdin);
    answer1 [strlen (answer1) - 1] = 0;
    atomName = XInternAtom (dpy, answer1, xTrue);
    if (atomName == None) {
        printf ("Invalid Atom Name\n");
        break;
    }
    fprintf (outputPtr, "\nCalling SELinuxGetSelectionContext (19)
with %s for this display\n", (XGetAtomName (dpy, atomName)));
    SELinuxGetSelectionContext (dpy, atomName);
    break;
case 20:
    // Get ATOM
    printf ("Enter Selection Atom Name: ");
    fgets (answer1, sizeof(answer1), stdin);
    answer1 [strlen (answer1) - 1] = 0;
    atomName = XInternAtom (dpy, answer1, xTrue);
    if (atomName == None) {
        printf ("Invalid Atom Name\n");
        break;
    }
    fprintf (outputPtr, "\nCalling SELinuxGetSelectionDataContext (20)
with %s for this Window:\n", (XGetAtomName (dpy, atomName)));
    SELinuxGetSelectionDataContext (dpy, atomName);
    break;
case 21:
    fprintf (outputPtr, "\nCalling SELinuxListSelections (21) for this
display:\n");
    SELinuxListSelections (dpy);
    break;
case 22:
    printf ("Enter Resource ID or return for this window: ");
```

```
fgets (answer1, sizeof(answer1), stdin);
answer1 [strlen (answer1) - 1] = 0;
if (answer1 [0] == 0)
    resourceID = w;
else
    resourceID = (atoi (answer1));
fprintf (outputPtr, "\nCalling SELinuxGetClientContext (22) for
this Resource: %d\n", resourceID);
SELinuxGetClientContext (dpy, resourceID);
break;
default:
    printf ("\nInvalid Selection\n");
    Menu ();
    break;
}
}
}
```

The X-setest application can be built using the following command:

```
gcc X-setest.c ../x-common/XSELinuxOMFunctions.c -o X-setest
-l selinux -l X11 -l Xi
```

The X-setest application can be called as follows:

```
# Output all information to the screen:
X-setest

# Output all information to a specified file [log_file_name],
# with minimum information displayed on the screen:
X-setest [log_file_name]
```

## 5. Appendix A - Policy Investigation Tools

### 5.1 Introduction

This section describes the tools used to investigate the modular-test (base + ext\_gateway + netlabel + int\_gateway + move\_file modules) policy for the message filter project during its development, debugging and validation.

Points to note:

1. When viewing a policy via investigation tools such as apol, the rules and statements that contain permissions, roles etc. have been resolved and will therefore not look the same as in the original source code. For example in the ext\_gateway.conf module there are two separate but common rules (as they specify permissions required for that part of the policy):

```
# Allow the client/server to send/rcv packets:
allow unconfined_t default_secmark_packet_t : packet { send rcv };

# Required to allow the iptables to load as needs to relabel:
allow unconfined_t default_secmark_packet_t : packet relabelto;
```

When they are viewed via the investigation tools, the two rules would have been amalgamated and displayed as:

```
allow unconfined_t default_secmark_packet_t : packet { send rcv
relabelto };
```

2. When investigating loadable modules that have had additional configuration added via semanage (user, port etc.), then the binary policy file will contain this information. However when using the tools to view the packaged module source, these changes will not appear (see [sechecker](#) for an example).

### 5.2 Using audit2allow and audit2why

audit2allow is a very useful tool, however it needs to be used with caution as it gives permissions that are not always required; does not define any types; shows role transitions as process transitions; and has various other features.

The process used to debug most of the policy revolved around audit2allow and monitoring the audit log, updating the policy with the results and removing any permissions thought not to be required, testing the policy and then re-running audit2allow and so on.

There were times when audit2allow (or anything else for that matter) was of no help, particularly when the system hung during boot or at login time. It was just a case of keeping track of the changes, determining the differences and finding a fix.

audit2why is also useful, however it does not like AVC granted messages in the audit log.

### 5.3 Using `seaudit` and `setroubleshoot`

During the development these audit tools were initially used, however it was found that after a while it was easier to clear the audit log (`>audit.log`), then tail it (`tail -f audit.log`) and 'see' the problem flashing past, run `audit2allow` and then interpret the results.

### 5.4 Using `sediffx`

The `sediff(1)` (command line) and `sediffx(1)` (GUI) compares two policies and finds the differences between them. These are part of the SETools package and have extensive help (see the `/usr/share/setools-3.3` directory) and man pages (`man sediff` and `man sediffx`) that should be read before using the tools.

The GUI version was used for testing the `modular-test` policy as the differences in two policies needed to be found for the following reason:

- When testing the `ext_gateway` module, a new role was created called `message_filter_r`. This role needed to be associated with a user and can be achieved by one of two ways:
  1. Add a `user` statement to the policy and associate the role. This worked with no problems, however as also experimenting with MLS policies it would mean having two `gateway` modules (as one `user` statement needs level and range, the other does not).
  2. Use `semanage` to associate the user to the new role. This did not work as expected using the following command:

```
semanage user -m -R "message_filter_r" user_u
```

The result was that the policy caused SELinux to lock the system so no login was possible, therefore the repair disk had to be used to change the policy, as the system had been configured with the `save-previous = true` set in the `/etc/selinux/semanage.conf` file. Therefore the original (good) binary policy was available and copied over to the `./modular-test/policy` directory, the system restarted, and the differences investigated.

To check the differences between the two policies, the original (good) and current (bad) were loaded as shown in [Figure 5.1](#):

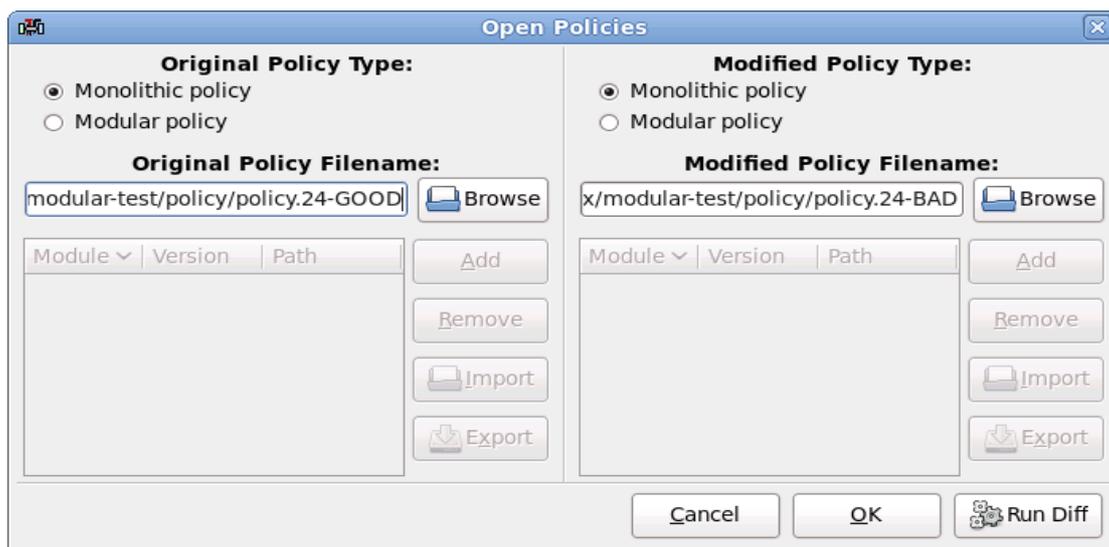


Figure 5.1: Opening the two policies in sediffx

The ‘Run Diff’ was run and [Figure 5.2](#) shows the differences between these two policies. As can be seen, the `unconfined_r` role has been removed by the `semanage` command:

```
semanage user -m -R "message_filter_r" user_u
```

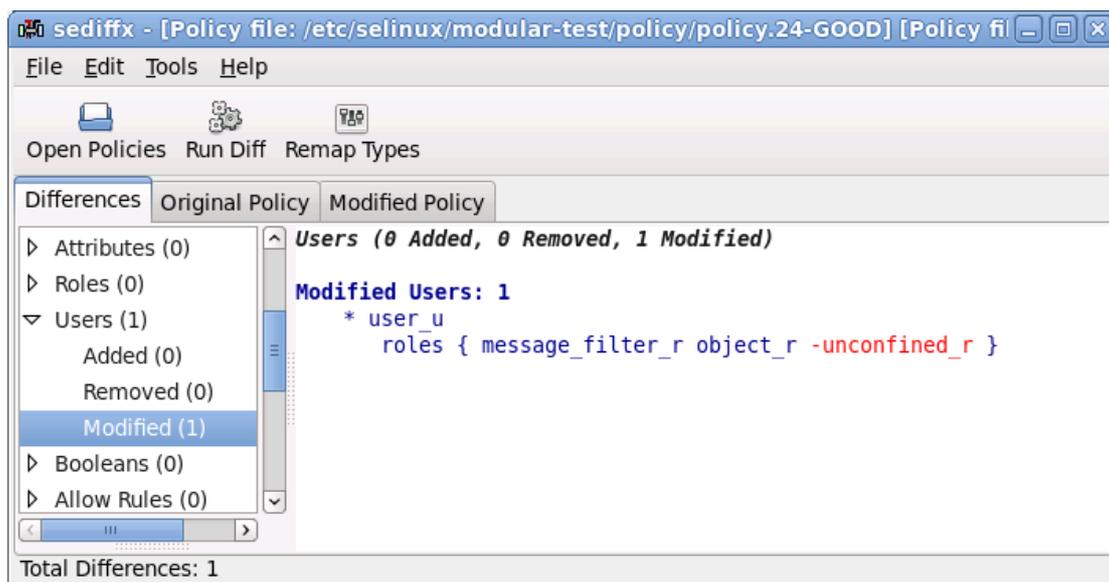


Figure 5.2: sediffx showing the differences in the two policies

The fix for this is to add all the roles when updating a user with `semanage` as follows:

```
semanage user -m -R "message_filter_r unconfined_r" user_u
```

## 5.5 Using sechecker

This command line application is part of the SETools package and is used to analyse a policy for various flaws. It has extensive help and man pages (`man sechecker`) that should be read before using the tool.

The `sechecker(8)` command has a set of pre-built modules<sup>5</sup> that can be run individually or from a profile containing a list of modules (a number of profiles are supplied – see the `/usr/share/setools-3.3` directory that also contains help files). Each of these modules will check for a specific set of flaws (e.g. find users without roles).

There are also ‘utility modules’ that find basic information (e.g. find domains) and are used by the modules when checking for flaws. Each module function is described in [Table 5-2](#) and [Table 5-3](#) along with comments on the test results for the `modular-test` policy. New modules can be written for `sechecker`, however the source code is required (that contains a module template source file to help with the development).

Note that some modules will work on packaged modules and source files only (as they have the attribute identifiers available). [Table 5-2](#) and [Table 5-3](#) has a column that specifies what type of policy (module, source or binary) each module supports.

### 5.5.1 Testing the Policy

For testing the `modular-test` policy, `sechecker` was run using the `modular-source`<sup>6</sup> policy with the `modular-test.profile` and using the binary policy with the `modular-test-binary.profile`. The two profiles are shown in [Table 5-1](#) (note that the `modular-test.profile` is in fact a copy of the `all-checks-no-mls.sechecker` that is supplied with `sechecker`).

The reason for running on both types of policy is to show the differences, as the module source does not contain the user association with the `message_filter_r` role, and the binary policy does not show that an attribute is not used by any rules.

| modular-test.profile  | modular-test-binary.profile  |
|---|--|
| <pre> &lt;sechecker version="1.1"&gt; &lt;profile&gt;   &lt;module name="find_domains"&gt;     &lt;output value="quiet"/&gt;     &lt;option name="domain_attribute"&gt;       &lt;item value="domain"/&gt;     &lt;/option&gt;   &lt;/module&gt;   &lt;module name="find_file_types"&gt;     &lt;output value="quiet"/&gt;     &lt;option name="file_type_attribute"&gt;       &lt;item value="file_type"/&gt;     &lt;/option&gt;   &lt;/module&gt;   &lt;module name="domain_and_file"&gt;     &lt;output value="short"/&gt;   &lt;/module&gt;   &lt;module name="attrs_wo_types"&gt;     &lt;output value="short"/&gt;   &lt;/module&gt;   &lt;module name="roles_wo_types"&gt;     &lt;output value="short"/&gt;   &lt;/module&gt;   &lt;module name="users_wo_roles"&gt;     &lt;output value="short"/&gt;   &lt;/module&gt;   &lt;module name="roles wo allow"&gt; </pre> | <pre> &lt;sechecker version="1.1"&gt; &lt;profile&gt;   &lt;module name="roles_wo_types"&gt;     &lt;output value="short"/&gt;   &lt;/module&gt;   &lt;module name="users_wo_roles"&gt;     &lt;output value="short"/&gt;   &lt;/module&gt;   &lt;module name="roles wo allow"&gt;     &lt;output value="short"/&gt;   &lt;/module&gt;   &lt;module name="types wo allow"&gt;     &lt;output value="short"/&gt;   &lt;/module&gt;   &lt;module name="roles wo users"&gt;     &lt;output value="short"/&gt;   &lt;/module&gt;   &lt;module name="spurious_audit"&gt;     &lt;output value="short"/&gt;   &lt;/module&gt;   &lt;module name="inc_mount"&gt; </pre> |

<sup>5</sup> Not to be confused with the policy ‘loadable modules’.

<sup>6</sup> Actually the packaged modules (`base.pp`, `gateway.pp`, `netlabel.pp` and `move_file.pp`).

|  |   |
|--|---|
| <pre> &lt;output value="short"/&gt; &lt;/module&gt; &lt;module name="types_wo_allow"&gt;   &lt;output value="short"/&gt; &lt;/module&gt; &lt;module name="attrs_wo_rules"&gt;   &lt;output value="short"/&gt; &lt;/module&gt; &lt;module name="roles_wo_users"&gt;   &lt;output value="short"/&gt; &lt;/module&gt; &lt;module name="spurious_audit"&gt;   &lt;output value="short"/&gt; &lt;/module&gt; &lt;module name="inc_mount"&gt;   &lt;output value="short"/&gt; &lt;/module&gt; &lt;module name="domains_wo_roles"&gt;   &lt;output value="short"/&gt; &lt;/module&gt; &lt;module name="inc_dom_trans"&gt;   &lt;output value="short"/&gt; &lt;/module&gt; &lt;module name="find_net_domains"&gt;   &lt;output value="quiet"/&gt;   &lt;option name="net_obj"&gt;     &lt;item value="netif"/&gt;     &lt;item value="tcp_socket"/&gt;     &lt;item value="udp_socket"/&gt;     &lt;item value="node"/&gt;     &lt;item value="association"/&gt;   &lt;/option&gt; &lt;/module&gt; &lt;module name="find_port_types"&gt;   &lt;output value="quiet"/&gt; &lt;/module&gt; &lt;module name="find_node_types"&gt;   &lt;output value="quiet"/&gt; &lt;/module&gt; &lt;module name="find_netif_types"&gt;   &lt;output value="quiet"/&gt; &lt;/module&gt; &lt;module name="inc_net_access"&gt;   &lt;output value="short"/&gt; &lt;/module&gt; &lt;module name="unreachable_doms"&gt;   &lt;output value="short"/&gt; &lt;/module&gt; &lt;/profile&gt; &lt;/sechecker&gt; </pre> | <pre> &lt;output value="short"/&gt; &lt;/module&gt; &lt;module name="inc_net_access"&gt;   &lt;output value="short"/&gt; &lt;/module&gt; &lt;/profile&gt; &lt;/sechecker&gt; </pre> |
|--|---|

**Table 5-1: sechecker profiles** – *The profiles used to check the modular-test packages and the binary policy files.*

The sechecker commands were each run twice, once with `-v` (for verbose output that will detail any issues found in gory detail) and once without the `-v` option:

```

sechecker --fcfile=/etc/selinux/modular-test/contexts/files/file_contexts -p
modular-test.profile modular-test.list > modular-test-results.txt

sechecker --fcfile=/etc/selinux/modular-test/contexts/files/file_contexts -v -p
modular-test.profile modular-test.list > modular-test-verbose-results.txt

sechecker --fcfile=/etc/selinux/modular-test/contexts/files/file_contexts -p
modular-test-binary.profile /etc/selinux/modular-test/policy/policy.23 >
modular-test-binary-results.txt

sechecker --fcfile=/etc/selinux/modular-test/contexts/files/file_contexts -v -p
modular-test-binary.profile /etc/selinux/modular-test/policy/policy.23 >
modular-test-binary-verbose-results.txt

```

Note that the binary policy is referenced by its full path name, but the modular policy is referenced by a file called `modular-test.list`. The contents of this file is as follows, and can be built by the [apol tool](#) described later:

```

# modular-test lists the modules to be tested:
#
policy_list 1 modular
/etc/selinux/modular-test/modules/active/base.pp

```

```
/etc/selinux/modular-test/modules/active/modules/ext_gateway.pp
/etc/selinux/modular-test/modules/active/modules/int_gateway.pp
/etc/selinux/modular-test/modules/active/modules/move_file.pp
/etc/selinux/modular-test/modules/active/modules/netlabel.pp
```

### 5.5.2 The Results

The output from the `modular-test-binary.profile` without the `-v` option is shown below, however the main results are shown in:

- [Table 5-2](#) that describes the results for each module and the authors interpretation / action regarding any policy changes.
- [Table 5-3](#) that describes the results from the utility modules.

```
Module name: inc_mount Severity: med
This module finds domains that have incomplete mount permissions.
In order for a mount operation to be allowed by the policy the following rules
must be present:
  1) allow somedomain_d sometype_t : filesystem { mount };
  2) allow somedomain_d sometype_t : dir { mounon };

This module finds domains that have only one of the rules listed above.

Found 0 types.
-----
Module name: inc_net_access Severity: med
This module finds all network domains in a policy which do not have the
required permissions needed to facilitate network communication. For network
domains to communicate, the following conditions must be true:
  1) the domain must have read or receive permissions on a socket of the same
type
  2) the domain must have send or receive permissions on an IPsec association
(see find_assoc_types)
  3) the domain must have send or receive permissions on netif objects for a
netif type (see find_netif_types)
  4) the domain must have send or receive permissions on node objects for a
node type (see find_node_types)
  5) the domain must have send or receive permissions on port objects for a
port type (see find_port_types)

Found 3 network domains with insufficient permissions.

int_gateway_t, ext_gateway_t, unconfined_t
-----
Module name: roles_wo_allow Severity: low
This module finds roles defined in the policy that are not used in any role
allow rules. It is not possible to transition to or from any role that does not
have any role allow rules.

Found 0 roles.
-----
Module name: roles_wo_types Severity: low
This module finds roles in the policy that have no types. A role with no types
cannot form a valid context.

Found 0 roles.
-----
Module name: roles_wo_users Severity: low
This module finds roles that are not assigned to users. If a role is not
assigned to a user it cannot form a valid context.

Found 0 roles.
-----
Module name: spurious_audit Severity: low
This module finds audit rules in the policy which do not affect the auditing of
the policy. This could happen in the following situations:
  1) there is an allow rule with the same key and permissions for a dontaudit
rule
  2) there is an auditallow rule without an allow rule with the same key or
with permissions that do not appear in an allow rule with the same key.
```

```
Found 1 rules.
dontaudit ext_gateway_t unconfined_t : filesystem getattr ;
```

-----  
**Module name: types\_wo\_allow Severity: low**

This module finds types defined in the policy that are not used in any allow rules. A type that is never granted an allow rule in the policy is a dead type. This means that all attempted access to the type will be denied including attempts to relabel to a (usable) type. The type may need to be removed from the policy or some intended access should be granted to the type.

Found 1 types.

```
socket_t
```

-----  
**Module name: users\_wo\_roles Severity: low**

This module finds all the SELinux users in the policy that have no associated roles. Users without roles may appear in the label of a file system object; however, these users cannot login to the system or run any process. Since these users cannot be used on the system, a policy change is recommended to remove the users or provide some intended access.

Found 0 users.

## The SELinux Notebook - Sample Policy Source

| <i>Module Name</i> | <i>Module Description</i>  | <i>Valid for Binary, Module or Source files</i> | <i>Comments on running sechecker on the modular-test policy with -v (verbose) option</i>   |
|--------------------|--|---|--|
| attrs_wo_rules     | This module finds attributes in the policy that are not used in any rules; These attributes will get thrown out by the compiler and have no effect on the security environment. They are unnecessary and should be removed.  | Modules and Source only                         | This module found an attribute called <code>message_filter_domains</code> that is not used (it was added to the modules and had the domain types added).<br><b>Decision:</b> The attribute can be removed from the policy.   |
| attrs_wo_types     | This module finds attributes in the policy that are not associated with any types. Attributes without types can cause type fields in rules to expand to empty sets and thus become unreachable. This makes for misleading policy source files.   | Modules and Source only                         | This module did not find any attributes without types.   |
| domain_and_file    | This module finds all types in the policy treated as both a domain and a file type. See <code>find_domains</code> and <code>find_file_types</code> modules for details about the heuristics used to determine these types. It is considered bad security practice to use the same type for a domain and its data objects because it requires that less restrictive access be granted to these types. | Modules and Source only                         | This module found three types associated to domains and files ( <code>unconfined_t</code> , <code>ext_gateway</code> and <code>int_gateway_t</code> ).<br>This probably occurred in the policy because the base is all <code>unconfined_t</code> .<br><b>Decision:</b> Without building a more complex policy it is thought that this is an acceptable risk. |
| domains_wo_roles   | This module finds all domains in the policy not associated with a role. These domains cannot have a valid security context. The <code>object_r</code> role is not considered in this check.  | Modules and Source only                         | This module did not find any domains without roles.  |
| imp_range_trans    | This module finds impossible range transitions in a policy. A range transition is possible if and only if all of the following conditions are satisfied:<br>1) there exist TE rules allowing the range transition to occur.<br>2) there exist RBAC rules allowing the range transition to occur.<br>3) at least one user must be able to transition to the target MLS range.                         | Binary, Modules and Source                      | As the modular-test policy is not MLS, then this was not run.  |

## The SELinux Notebook - Sample Policy Source

| <i>Module Name</i> | <i>Module Description</i>  | <i>Valid for Binary, Module or Source files</i> | <i>Comments on running sechecker on the modular-test policy with -v (verbose) option</i>   |
|--------------------|--|---|--|
| inc_dom_trans      | <p>This module finds potential domain transitions missing key permissions. A valid domain transition requires the following:</p> <ol style="list-style-type: none"> <li>1) the starting domain can transition to the end domain for class process.</li> <li>2) the end domain has some type as an entrypoint.</li> <li>3) the starting domain can execute that entrypoint type.</li> <li>4) (optional) a type_transition rule specifying these three types.</li> </ol>   | Modules and Source only                         | This module did not find any incomplete domain transitions.  |
| inc_mount          | <p>This module finds domains that have incomplete mount permissions. In order for a mount operation to be allowed by the policy the following rules must be present:</p> <ol style="list-style-type: none"> <li>1) allow somedomain_d sometype_t : filesystem { mount };</li> <li>2) allow somedomain_d sometype_t : dir { mounon };</li> </ol> <p>This module finds domains that have only one of the rules listed above.</p>   | Binary, Modules and Source                      | This module did not find any domains with incomplete mount permissions.  |
| inc_net_access     | <p>This module finds all network domains in a policy which do not have the required permissions needed to facilitate network communication. For network domains to communicate, the following conditions must be true:</p> <ol style="list-style-type: none"> <li>1) the domain must have read or receive permissions on a socket of the same type.</li> <li>2) the domain must have send or receive permissions on an IPsec association (see find_assoc_types).</li> <li>3) the domain must have send or receive permissions on netif objects for a netif type (see find_netif_types).</li> <li>4) the domain must have send or receive permissions on node objects for a node type (see find_node_types).</li> <li>5) the domain must have send or receive permissions on port objects for a port type (see find_port_types).</li> </ol> | Binary, Modules and Source                      | <p>This module found three network domains with insufficient permissions (unconfined_t, ext_gateway_t and int_gateway_t).</p> <p><b>Decision:</b> As the policy modules were built for minimum privilege, adding additional (and not required) permissions would add no value to the policy.</p> <p>-----</p> <p>Note: Try running this module with the NetLabel loadable module detailed in <a href="#">Appendix B – NetLabel Module Support for network_peer_controls</a> as there will then be an additional network domain found (network_peer_t).</p> |

## The SELinux Notebook - Sample Policy Source

| <i>Module Name</i> | <i>Module Description</i>   | <i>Valid for Binary, Module or Source files</i> | <i>Comments on running sechecker on the modular-test policy with -v (verbose) option</i>  |
|--------------------|---|---|---|
| roles_wo_allow     | This module finds roles defined in the policy that are not used in any role allow rules. It is not possible to transition to or from any role that does not have any role allow rules.  | Binary, Modules and Source                      | This module did not find any roles without an allow rule.   |
| roles_wo_types     | This module finds roles in the policy that have no types. A role with no types cannot form a valid context.   | Binary, Modules and Source                      | This module did not find any roles without types.   |
| roles_wo_users     | This module finds roles that are not assigned to users. If a role is not assigned to a user it cannot form a valid context.   | Binary, Modules and Source                      | On the modular policy files sechecker reported one role without a user ( <code>message_filter_r</code> ). The reason for this is because the user association ( <code>user_u</code> ) was added with <code>semanage</code> .<br><br>Note: Running <code>sechecker</code> on the binary policy does not report this error as <code>semanage</code> has added the association.<br><br><b>Decision:</b> Leave as it is, however <code>sechecker</code> could be modified at some stage to check the policy store !!. |
| spurious_audit     | This module finds audit rules in the policy which do not affect the auditing of the policy. This could happen in the following situations:<br>1) there is an <code>allow</code> rule with the same key and permissions for a <code>dontaudit</code> rule.<br>2) there is an <code>auditallow</code> rule without an <code>allow</code> rule with the same key or with permissions that do not appear in an <code>allow</code> rule with the same key. | Binary, Modules and Source                      | This module found one spurious audit rule:<br><code>dontaudit ext_gateway_t unconfined_t : filesystem getattr ;</code><br><b>Decision:</b> Review policy and update the <code>getattr</code> permission as required.  |
| types_wo_allow     | This module finds types defined in the policy that are not used in any allow rules. A type that is never granted an allow rule in the policy is a dead type. This means that all attempted access to the type will be denied including attempts to relabel to a (usable) type. The type may need to be removed from the policy or some intended access should be granted to the type.   | Binary, Modules and Source                      | This module found one type without and allow rule ( <code>socket_t</code> ). This was added to the <code>netlabel.conf</code> module but never used.<br><br><b>Decision:</b> Remove <code>socket_t</code> .   |

## The SELinux Notebook - Sample Policy Source

| <i>Module Name</i> | <i>Module Description</i>   | <i>Valid for Binary, Module or Source files</i> | <i>Comments on running sechecker on the modular-test policy with -v (verbose) option</i>   |
|--------------------|---|---|--|
| unreachable_doms   | <p>This module finds all domains in a policy which are unreachable. A domain is unreachable if any of the following apply:</p> <ol style="list-style-type: none"> <li>1) There is insufficient type enforcement policy to allow a transition.</li> <li>2) There is insufficient RBAC policy to allow a transition.</li> <li>3) There are no users with proper roles to allow a transition.</li> </ol> <p>However, if any of the above rules indicate an unreachable domain, yet the domain appears in the system default contexts file, it is considered reachable.</p> | Modules and Source only                         | <p>This module found no unreachable domains.</p> <p>-----</p> <p>Note: Try running this module with the NetLabel loadable module detailed in <a href="#">Appendix B – NetLabel Module Support for network_peer_controls</a> as there will then be one unreachable domain found (netlabel_peer_t).</p> <p>This was never intended as a domain, only a label for the NetLabel test. It is suspected that they were found by the <a href="#">find_domains</a> utility module (Bullet 2 - it is the source of a TE rule for object class other than filesystem).</p> |
| users_wo_roles     | <p>This module finds all the SELinux users in the policy that have no associated roles. Users without roles may appear in the label of a file system object; however, these users cannot login to the system or run any process. Since these users cannot be used on the system, a policy change is recommended to remove the users or provide some intended access.</p>  | Binary, Modules and Source                      | <p>This module did not find any users without roles.</p>   |

**Table 5-2: Modules in Version 1.1 of sechecker (8)** – *The Comments column covers the authors interpretation of the test results on the modular-test policy base and loadable modules using the sechecker modules and profiles.*

## The SELinux Notebook - Sample Policy Source

| <i>Module Name</i> | <i>Module Description</i>  | <i>Valid for Binary, Module or Source files</i> | <i>Comments on running sechecker on the modular-test policy with -v (verbose) option</i>  |
|--------------------|--|---|---|
| find_assoc_types   | This module finds types with an unlabeled SID.   | Binary, Modules and Source                      | This module does not output a report using the standard profiles, however it can be run with the <code>-v</code> and <code>-m</code> options as follows:<br><pre>sechecker -v -m find_assoc_types &lt;policy&gt;</pre> Running this on the modular-test policy will result in finding <code>unconfined_t</code> as the unlabeled SID.   |
| find_domains       | This is a utility module which finds types in a policy that are treated as a domain. A <code>type</code> is considered a domain if any of the following is true:<br>1) it has an <code>attribute</code> associated with domains.<br>2) it is the source of a TE rule for object class other than <code>filesystem</code> .<br>3) it is the default type in a <code>type_transition</code> rule for object class <code>process</code> .<br>4) it is associated with a role other than <code>object_r</code> .                   | Modules and Source only                         | This module does not output a report using the standard profiles, however it can be run with the <code>-v</code> and <code>-m</code> options as follows:<br><pre>sechecker -v -m find_domains &lt;policy&gt;</pre> Running this on the modular-test policy will result in finding four domains ( <code>move_file_t</code> , <code>ext_gateway_t</code> , <code>int_gateway_t</code> and <code>unconfined_t</code> ).  |
| find_file_types    | This module finds all types in the policy treated as a file type. A <code>type</code> is considered a file type if any of the following is true:<br>1) it has an <code>attribute</code> associated with file types.<br>2) it is the source of a rule to allow <code>filesystem</code> <code>associate</code> permission.<br>3) it is the default type of a <code>type transition</code> rule with an object class other than <code>process</code> .<br>4) it is specified in a context in the <code>file_contexts</code> file. | Modules and Source only                         | This module does not output a report using the standard profiles, however it can be run with the <code>-v</code> and <code>-m</code> options as follows:<br><pre>sechecker -v -m find_file_types &lt;policy&gt;</pre> Running this on the modular-test policy will result in finding nine file types ( <code>out_file_t</code> , <code>out_queue_t</code> , <code>move_file_exec_t</code> , <code>in_file_t</code> , <code>in_queue_t</code> , <code>secure_services_exec_t</code> , <code>ext_gateway_t</code> , <code>int_gateway_t</code> and <code>unconfined_t</code> ). |

## The SELinux Notebook - Sample Policy Source

| <i>Module Name</i> | <i>Module Description</i>   | <i>Valid for Binary, Module or Source files</i> | <i>Comments on running sechecker on the modular-test policy with -v (verbose) option</i>  |
|--------------------|---|---|---|
| find_net_domains   | <p>This module finds all types in a policy considered to be network domains. A type is considered a network domain if it is the subject of TE rules involving certain object classes, which are currently defined as:</p> <ol style="list-style-type: none"> <li>1) netif</li> <li>2) tcp_socket</li> <li>3) udp_socket</li> <li>4) node</li> <li>5) association</li> </ol> <p>These values can be overridden in this module's profile.</p> | Binary, Modules and Source                      | <p>This module does not output a report using the standard profiles, however it can be run with the <code>-v</code> and <code>-m</code> options as follows:</p> <pre>sechecker -v -m find_net_domains &lt;policy&gt;</pre> <p>Running this on the modular-test policy will result in finding three net domains (ext_gateway_t, int_gateway_t and unconfined_t).</p> |
| find_netif_types   | <p>This module finds all types in a policy treated as a netif type. A type is considered a netif type if it is used in the context of a netifcon statement or the context of the netif initial SID.</p>   | Binary, Modules and Source                      | <p>This module does not output a report using the standard profiles, however it can be run with the <code>-v</code> and <code>-m</code> options as follows:</p> <pre>sechecker -v -m find_netif_types &lt;policy&gt;</pre> <p>Running this on the modular-test policy will result in finding that the only use of netif is in the initial SID for unconfined_t.</p> |
| find_node_types    | <p>This module finds all types in a policy treated as a node type. A type is considered a node type if it is used in the context of a nodecon statement or the context of the node initial SID.</p>   | Binary, Modules and Source                      | <p>This module does not output a report using the standard profiles, however it can be run with the <code>-v</code> and <code>-m</code> options as follows:</p> <pre>sechecker -v -m find_node_types &lt;policy&gt;</pre> <p>Running this on the modular-test policy will result in finding that the only use of node is in the initial SID for unconfined_t.</p>   |

## The SELinux Notebook - Sample Policy Source

---

| <i>Module Name</i> | <i>Module Description</i>  | <i>Valid for Binary, Module or Source files</i> | <i>Comments on running sechecker on the modular-test policy with -v (verbose) option</i>   |
|--------------------|--|---|--|
| find_port_types    | This module finds all types in a policy treated as a port type. A type is considered a port type if it is used in the context of a portcon statement or the context of the port initial SID. | Binary, Modules and Source                      | This module does not output a report using the standard profiles, however it can be run with the -v and -m options as follows:<br><pre>sechecker -v -m find_port_types &lt;policy&gt;</pre> Running this on the modular-test policy will result in finding that the only use of port is in the initial SID for unconfined_t. |

**Table 5-3: Utility Modules in Version 1.1 of sechecker (8)** – *The Comments column covers the authors interpretation of the test results on the modular-test policy base and loadable modules using the sechecker modules and profiles.*

### 5.6 Using apol

The apol application is part of the SETools package and has extensive help (see the ‘Help’ tab or information in the /usr/share/setools-3.3 directory).

The author had problems displaying all the apol window on the screen but resolved this as described in the General Information section of volume 1.

The application analyses many different aspects of a policy that are not covered in this Notebook (the apol documentation is comprehensive though), however to attempt some analysis of the message filter policy, two scenarios are presented from the ‘Analysis’ tab<sup>7</sup> that were carried out using the binary policy file:

1. [Direct Relabel](#) - To show what can be relabeled by unconfined\_t as the [security policy](#) stated minimum required.
2. [Transitive Information Flow](#) - This shows other paths that may be available to allow information to flow that is not directly enabled by the policy, and could therefore be used to allow unauthorised access.

The majority of text comes directly from apol as it has a facility to copy the analysis information to the clipboard.

#### 5.6.1 General Information

The binary policy file was used as this has a complete picture of the policy. The policy source file could have been used however apol only supports the base or monolithic source. The other alternative is to use the packaged files that can be opened directly from the File>Open tab, selecting the Modular policy option, and then selecting the base module first then Add the other modules to the list as shown in [Figure 5.3](#). This list may then be exported for future use, with a sample as follows:

```
policy_list 1 modular
/etc/selinux/modular-test/modules/active/base.pp
/etc/selinux/modular-test/modules/active/modules/ext_gateway.pp
/etc/selinux/modular-test/modules/active/modules/int_gateway.pp
/etc/selinux/modular-test/modules/active/modules/move_file.pp
/etc/selinux/modular-test/modules/active/modules/netlabel.pp
```

---

<sup>7</sup> The ‘Analysis’ tab requires a ‘permissions’ file to be loaded (via Tools>Open Default Perm Map) that adds weighting to object permissions (see the apol Help>Information Flow Analysis tab). However the one supplied (apol\_perm\_mapping\_ver21) does not have all the new object classes added. The author updated this file (available in source package), however it made no difference to the findings (not that any were expected !!!).

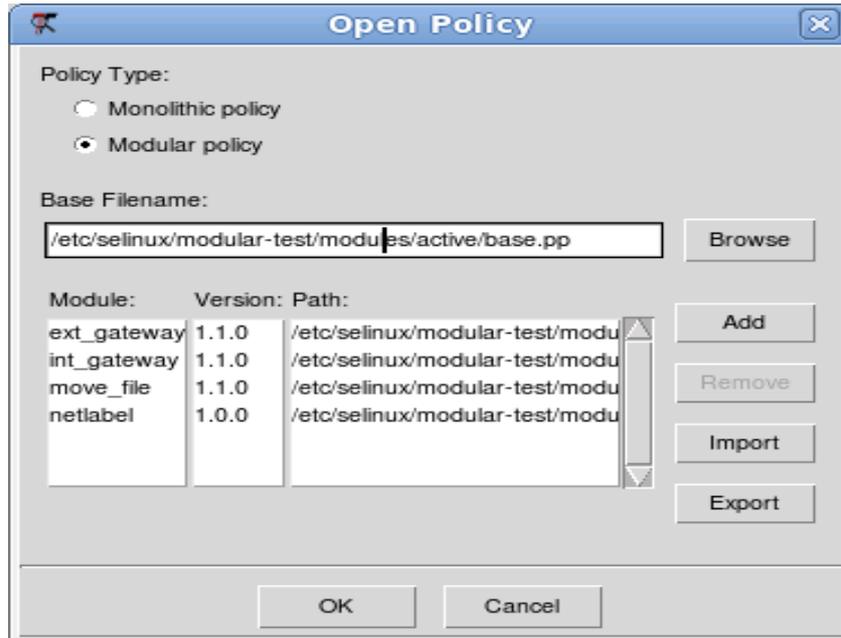
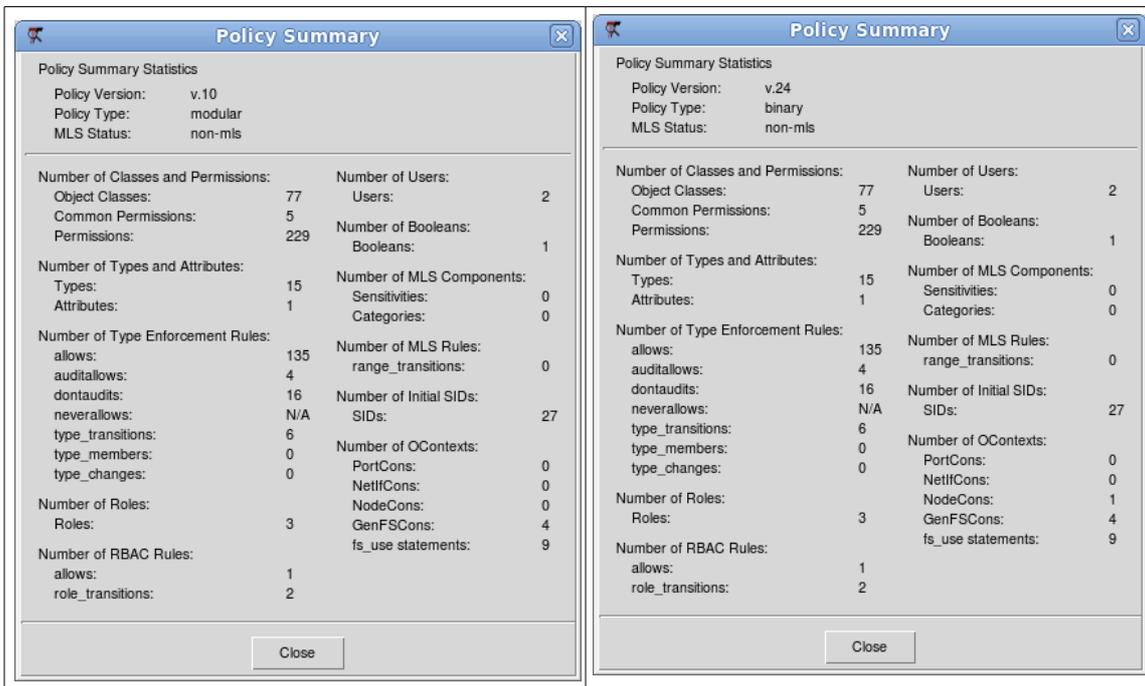
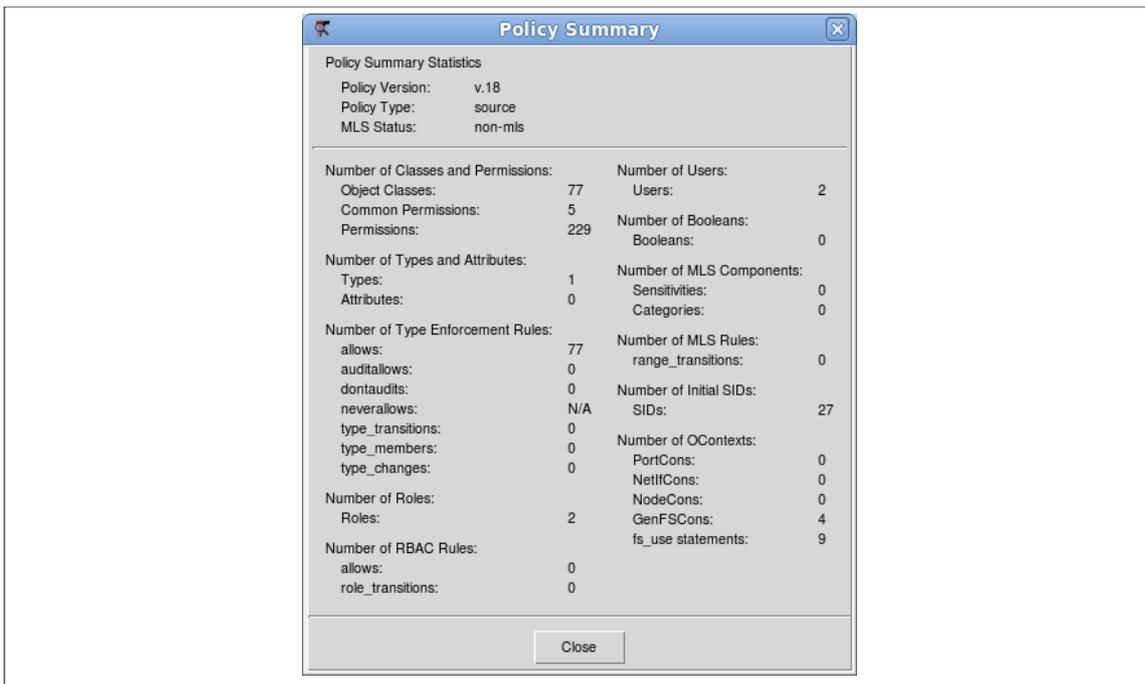


Figure 5.3: Opening the package policy files for analysis

Once the policy has been opened, a summary can be displayed using the Query>Policy Summary tabs. [Figure 5.4](#) shows one for each of the possible options: the packages, the binary policy and the base module source. As stated earlier the binary policy will be used for analysis.



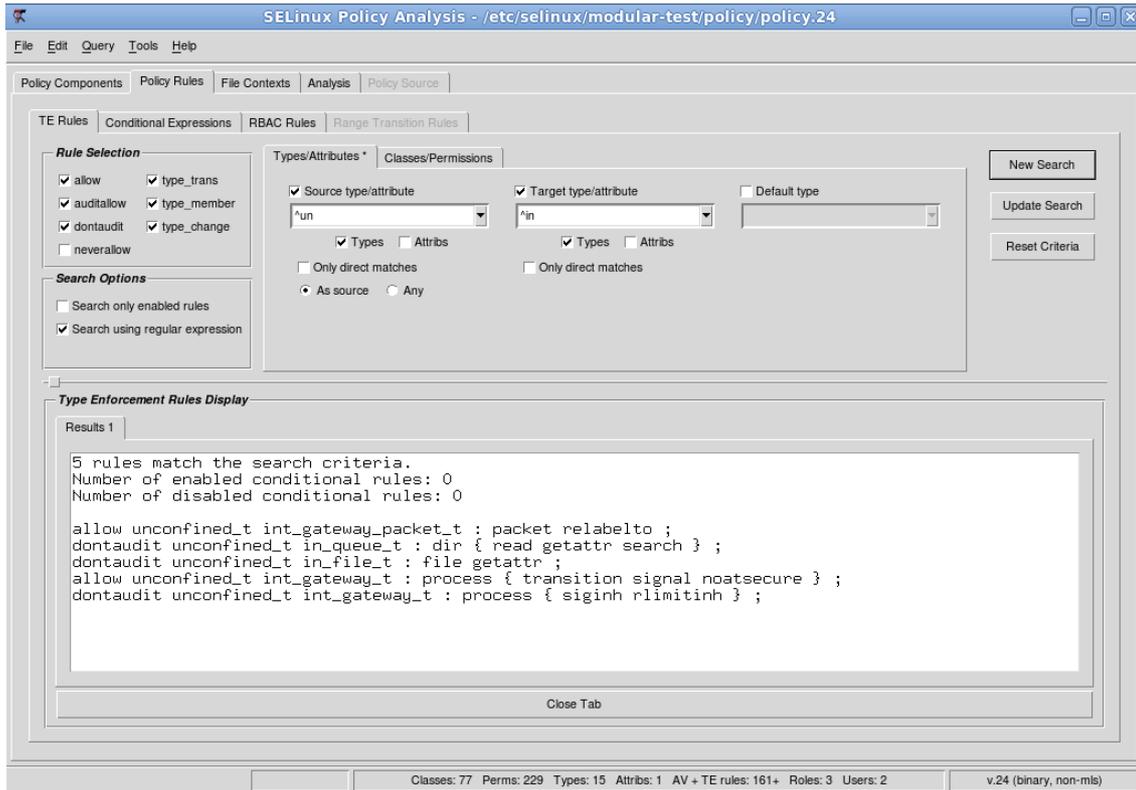


**Figure 5.4: The Modular, Binary and Base Source Policy Summaries**

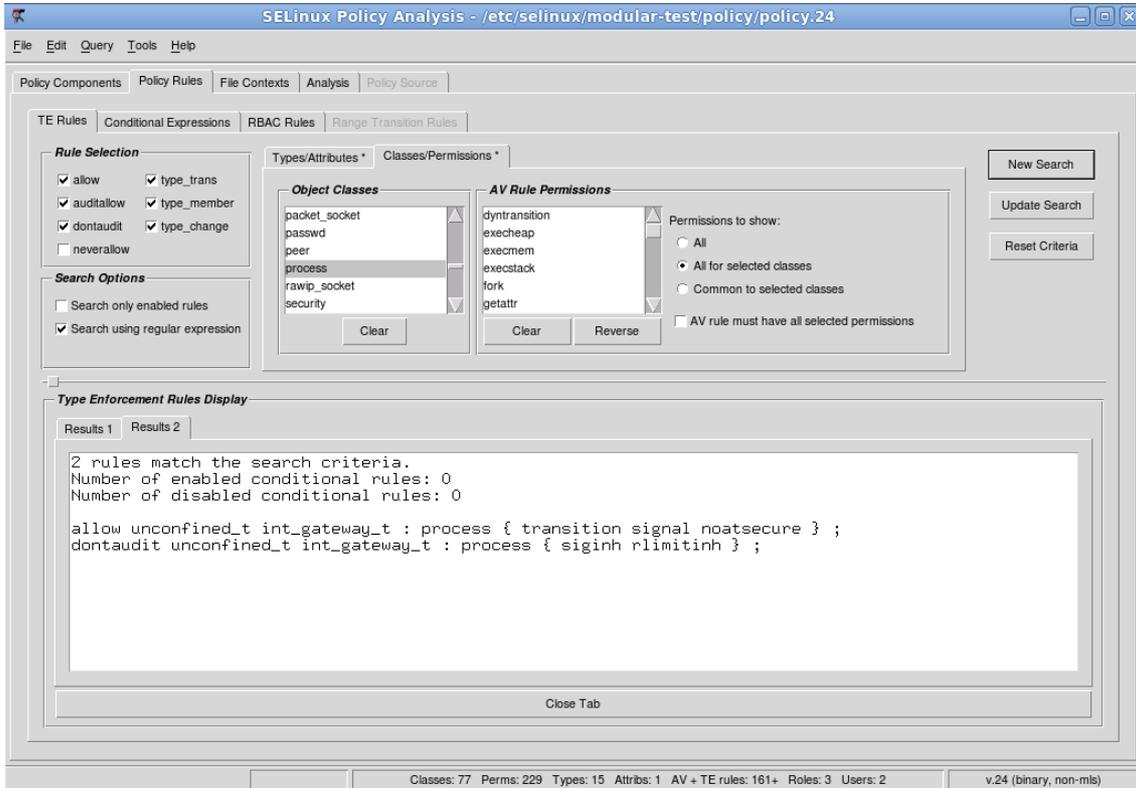
### 5.6.2 Type Enforcement Rules

[Figure 5.5](#) and [Figure 5.6](#) show how flexible `apol` can be in searching and analysing a policy. They show a search for TE Rules (via the Policy Rules>TE Rules tabs) using a regular expression with the source set to `^un` and the target `^in` that will find all rules starting with these characters. [Figure 5.5](#) shows that five rules were found, however when the Class/Permissions tab is set to select the process class ([Figure 5.6](#)), only two are found.

# The SELinux Notebook - Sample Policy Source



**Figure 5.5: Type Enforcement Rules (1) - Finding TE rules using a regular expression.**



**Figure 5.6: Type Enforcement Rules (2) - Finding TE rules using a regular expression with Class/Permissions tab set to show only the process class.**

## 5.6.3 Direct Relabel

The objective of the direct relabel analysis is to show what can be relabeled by `unconfined_t` as this needed to be the least possible. When the policy was written it was decided to only allow the message filter packets to be relabeled as the `iptables` needed to be loaded under `unconfined_t` and therefore required these permissions.

Note that to be able to

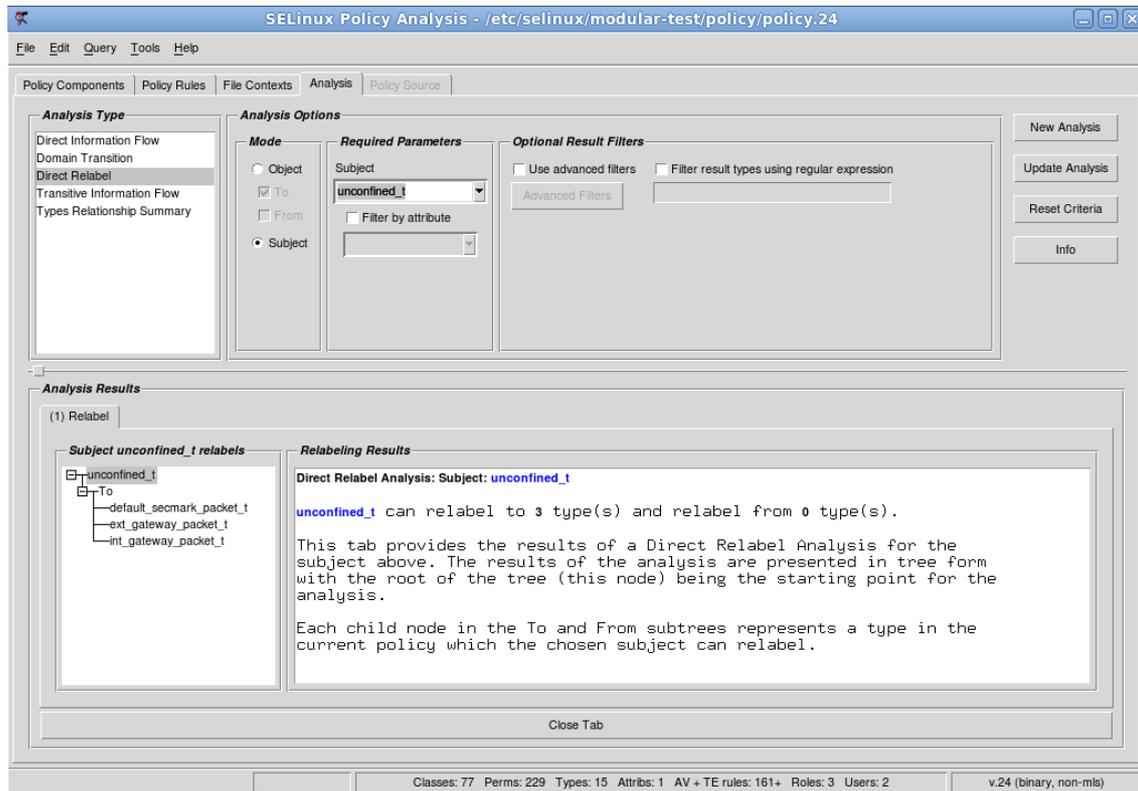
### 5.6.3.1 ap01 Direct Relabel Analysis

**Direct Relabel Analysis: Subject: `unconfined_t`**

`unconfined_t` can relabel to **3** type(s) and relabel from **0** type(s).

This tab provides the results of a Direct Relabel Analysis for the subject above. The results of the analysis are presented in tree form with the root of the tree (this node) being the starting point for the analysis.

Each child node in the To and From subtrees represents a type in the current policy which the chosen subject can relabel.



**Figure 5.7: Direct Relabel - Subject: `unconfined_t`**

Each of the nodes for the `unconfined_t` subject are as follows:

**default\_secmark\_packet\_t:**

```
unconfined_t can relabel to default_secmark_packet_t
allow unconfined_t default_secmark_packet_t : packet { send recv relabelto } ;
```

**ext\_gateway\_packet\_t:**

```
unconfined_t can relabel to ext_gateway_packet_t
allow unconfined_t ext_gateway_packet_t : packet relabelto ;
```

**int\_gateway\_packet\_t:**

```
unconfined_t can relabel to int_gateway_packet_t
allow unconfined_t int_gateway_packet_t : packet relabelto ;
```

### 5.6.4 Transitive Information Flows

This shows other paths that may be available to allow information to flow that is not directly enabled by the policy, and could therefore be used to allow unauthorised access. The `in_file_t` to / from `unconfined_t` was analysed in an attempt to write an application that would ‘plant’ a file in the message filters ‘`in_queue`’ when in enforcing mode (and assuming no access to the policy build tools). The author failed miserably<sup>8</sup> – any offers !!!!

#### 5.6.4.1 ap01 Transitive Information Flows Analysis

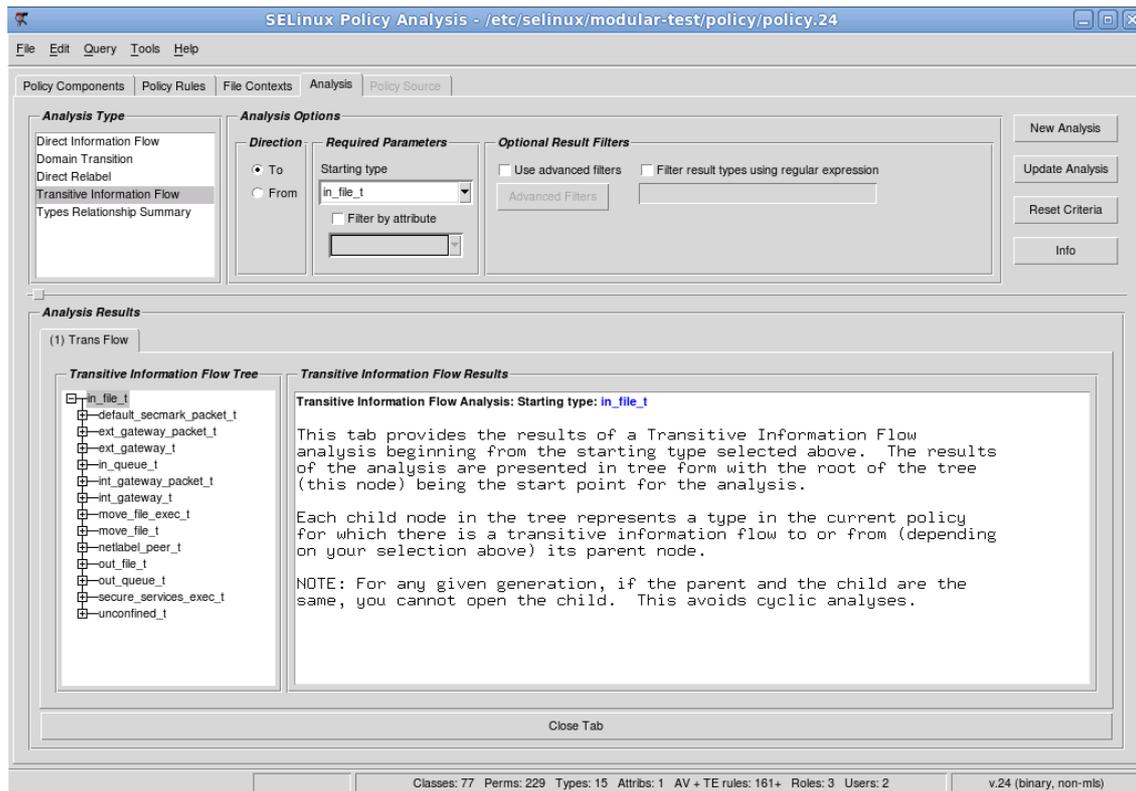
**Transitive Information Flow Analysis:** Starting type: `in_file_t` ([To](#) and [From](#))

This tab provides the results of a Transitive Information Flow analysis beginning from the starting type selected above. The results of the analysis are presented in tree form with the root of the tree (this node) being the start point for the analysis.

Each child node in the tree represents a type in the current policy for which there is a transitive information flow to or from its parent node.

---

<sup>8</sup> The kernel exploit from [Brad Spengler](#) is known but was not used (also see “[SELinux hardening for mmap\\_min\\_addr protections](#)” [Ref. 16]).



**Figure 5.8: Transitive Information Flow - Starting type : `in_file_t` showing the 'to' direction.**

Note that only the node for `unconfined_t` has been shown.

The first entry is with the 'To' direction selected, and the second with the 'From' direction selected (this entry has been edited<sup>9</sup> as it lists all objects that `unconfined_t` is allowed to relabel i.e. all of the object classes with a relabel permission).

### Information Flows TO `in_file_t` FROM `unconfined_t`:

```

Information flows to in_file_t from unconfined_t (find more flows)

Apol found the following number of information flows: 2

Flow 1 requires 3 steps(s).
unconfined_t -> ext_gateway_packet_t -> ext_gateway_t -> in_file_t
allow unconfined_t ext_gateway_packet_t : packet relabelto ;
allow ext_gateway_t ext_gateway_packet_t : packet { send recv } ;
allow ext_gateway_t in_file_t : file { write create getattr } ;

Flow 2 requires 2 steps(s).
unconfined_t -> ext_gateway_t -> in_file_t
allow ext_gateway_t unconfined_t : filesystem { getattr associate } ;
allow ext_gateway_t unconfined_t : association rcvfrom ;
allow ext_gateway_t unconfined_t : chr_file { read write getattr } ;
allow ext_gateway_t unconfined_t : dir search ;
allow ext_gateway_t unconfined_t : fd use ;
allow ext_gateway_t unconfined_t : file { read getattr execute } ;
allow ext_gateway_t unconfined_t : lnk_file read ;
allow ext_gateway_t unconfined_t : packet { send recv } ;
allow ext_gateway_t in_file_t : file { write create getattr } ;
    
```

<sup>9</sup> By selecting the 'Use advanced filters' check box and then 'Advanced Filters', it is possible to refine the search, for example excluding all permission weights below 10 (that will get permissions such as read, write and relabel).

### Information Flows FROM `in_file_t` TO `unconfined_t`:

```
Information flows from in_file_t to unconfined_t (find more flows)

Apol found the following number of information flows: 2

Flow 1 requires 2 steps(s).
  in_file_t -> move_file_t -> unconfined_t
    allow move_file_t in_file_t : file { read unlink } ;
    allow move_file_t unconfined_t : fd use ;
    allow move_file_t unconfined_t : chr_file { read write getattr } ;

Flow 2 requires 3 steps(s).
  in_file_t -> move_file_t -> unconfined_t -> unconfined_t
    allow move_file_t in_file_t : file { read unlink } ;
    allow move_file_t unconfined_t : fd use ;
    allow move_file_t unconfined_t : chr_file { read write getattr } ;
    allow unconfined_t unconfined_t : process { fork transition sigchld sigkill
sigstop signull signal ptrace getsched setsched getsession getpgid setpgid getcap
setcap share getattr setexec setfscreate noatsecure siginh setrlimit rlimitinh
dyntransition setcurrent execmem execstack execheap setkeycreate setsockcreate } ;
.....
.....
.....
    allow unconfined_t unconfined_t : unix_stream_socket { ioctl read write
create getattr setattr lock relabelfrom relabelto append bind connect listen
accept getopt setopt shutdown recvfrom sendto rcv_msg send_msg name_bind
connectto newconn acceptfrom } ;
```

## 6. Appendix B – NetLabel Module Support for network\_peer\_controls

### 6.1 Introduction

This is an enhanced NetLabel module to enable a NetLabel `netlabel_peer_t` label to be added to the network connection.

The [previous NetLabel module](#) used the standard F-12 Policy Capabilities<sup>10</sup> `network_peer_controls` (set to '0'). This exercise will set the `network_peer_controls` to '1' by updating the base module with a `policycap` statement, allowing the use of these new controls.

### 6.2 Configuration

The following steps are required to build the enhanced NetLabel module, it is assumed that the NetLabel services have already been installed from the previous NetLabel module exercise.

1. Ensure you are logged on as 'root' and SELinux is running in permissive mode (`setenforce 0`) to perform the build process.
2. Edit the `./notebook-source/modular-base-policy/base.conf` file to remove the '#' from the `policycap` statement as shown:

```
#
# This policycap statement will be used in a netlabel module exercise
# to show network_peer_controls. For now comment out:
policycap network_peer_controls;
```

3. Compile and link the base module so that the `network_peer_controls` are enabled:

```
checkmodule -o base.mod base.conf
semodule_package -o base.pp -m base.mod -f base.fc -s seusers -u users_extra
semodule -s modular-test -b base.pp
```

4. The following command will return '1' if the policy enabled the `network_peer_controls`:

```
cat /selinux/policy_capabilities/network_peer_controls
1
```

5. Produce a `netlabel_policycap.conf` loadable module file with a text editor containing the contents shown below:

```
module netlabel 2.0.0;

#
#####
#
# This Loadable Module will allow the netlabels to be added and checked #
# within the client / server applications that form part of the SECMARK #
```

<sup>10</sup> See the SELinux Filesystem section in Volume 1 - The Foundations.

```
# test examples. #
# #
# The following needs to happen to enable Netlabel to work as it is not #
# installed by default in F-12: #
# #
# (1) Download and install netlabel_tools #
# #
# (2) Install this loadable module. #
# #
# (3) Run the following netlabelctl command: #
#     netlabelctl unlbl add interface:lo address:127.0.0.1 \ #
#         label:system_u:object_r:netlabel_peer_t #
# #
# (4) Run netlabelctl -p unlbl list command to check all is okay. #
# #
# (5) Run the secure and standard client/server that should now display #
#     the netlabel_peer_t as the peer context. #
# #
# Important note: The polycap network_peer_controls; statement must be #
#     added to the base policy before the peer object can be #
#     used, otherwise the tcp_socket object will be used #
#     instead: #
# /selinux/policy_capabilities/network_peer_controls = 0 (use tcp_socket) #
# /selinux/policy_capabilities/network_peer_controls = 1 (use peer) #
# #
#####
#
require {
    type ext_gateway_t, unconfined_t;
    class peer { recv };
    class netif { ingress egress };
    class node { recvfrom sendto};
}

# Use this to label the peer level:
type netlabel_peer_t;

# These are used when /selinux/policy_capabilities/network_peer_controls = 1

# These are for unconfined_t ports:
allow unconfined_t netlabel_peer_t : peer recv;
allow netlabel_peer_t unconfined_t : netif ingress;
allow netlabel_peer_t unconfined_t : node recvfrom;

# These are for the external gateway port:
allow ext_gateway_t netlabel_peer_t : peer recv;
allow ext_gateway_t unconfined_t : netif egress;
allow ext_gateway_t unconfined_t : node sendto;

#
##### START OPTIONAL SECTION #####
#
optional {
    require {
        # This is defined in the int_gateway.conf module:
        type int_gateway_t;
    }
    allow int_gateway_t netlabel_peer_t : peer recv;
    allow int_gateway_t unconfined_t : netif egress;
    allow int_gateway_t unconfined_t : node sendto;
}
#
##### END OPTIONAL SECTION #####
#
```

### 6. Compile and link the new NetLabel module:

```
checkmodule -m netlabel_policycap.conf -o netlabel.mod

semodule_package -o netlabel.pp -m netlabel.mod
```

```
semodule -v -s modular-test -i netlabel.pp
```

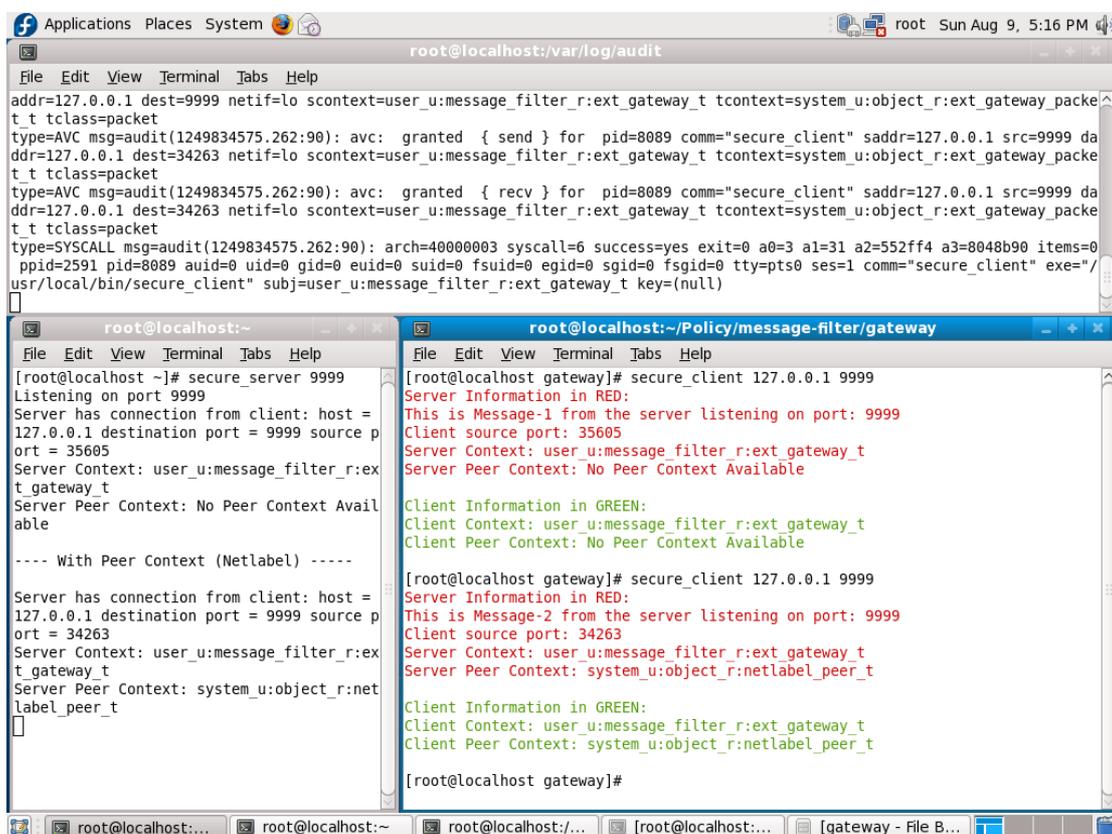
8. Run the following command to add the `netlabel_peer_t` label as follows:

```
netlabelctl unlbl add interface:lo address:127.0.0.1 \  
label:system_u:object_r:netlabel_peer_t
```

9. Run enforcing mode:

```
setenforce 1
```

10. Run either the client / server or `secure_client` / `secure_server` applications as shown in the [SECMARK tests](#). There should now be a peer context displayed as shown in the ‘With Peer Context (NetLabel)’ section of [Figure 6.1](#).



```
addr=127.0.0.1 dest=9999 netif=lo scontext=user_u:message_filter_r:ext_gateway_t tcontext=system_u:object_r:ext_gateway_packe  
t tclass=packet  
type=AVC msg=audit(1249834575.262:90): avc: granted { send } for pid=8089 comm="secure_client" saddr=127.0.0.1 src=9999 da  
ddr=127.0.0.1 dest=34263 netif=lo scontext=user_u:message_filter_r:ext_gateway_t tcontext=system_u:object_r:ext_gateway_packe  
t tclass=packet  
type=AVC msg=audit(1249834575.262:90): avc: granted { recv } for pid=8089 comm="secure_client" saddr=127.0.0.1 src=9999 da  
ddr=127.0.0.1 dest=34263 netif=lo scontext=user_u:message_filter_r:ext_gateway_t tcontext=system_u:object_r:ext_gateway_packe  
t tclass=packet  
type=SYSCALL msg=audit(1249834575.262:90): arch=40000003 syscall=6 success=yes exit=0 a0=3 a1=31 a2=552ff4 a3=8048b90 items=0  
ppid=2591 pid=8089 auid=0 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=1 comm="secure_client" exe="/  
usr/local/bin/secure_client" subj=user_u:message_filter_r:ext_gateway_t key=(null)
```

```
[root@localhost ~]# secure_server 9999  
Listening on port 9999  
Server has connection from client: host =  
127.0.0.1 destination port = 9999 source p  
ort = 35605  
Server Context: user_u:message_filter_r:ex  
t_gateway_t  
Server Peer Context: No Peer Context Avail  
able  
---- With Peer Context (NetLabel) ----  
Server has connection from client: host =  
127.0.0.1 destination port = 9999 source p  
ort = 34263  
Server Context: user_u:message_filter_r:ex  
t_gateway_t  
Server Peer Context: system_u:object_r:net  
label_peer_t
```

```
[root@localhost gateway]# secure_client 127.0.0.1 9999  
Server Information in RED:  
This is Message-1 from the server listening on port: 9999  
Client source port: 35605  
Server Context: user_u:message_filter_r:ext_gateway_t  
Server Peer Context: No Peer Context Available  
  
Client Information in GREEN:  
Client Context: user_u:message_filter_r:ext_gateway_t  
Client Peer Context: No Peer Context Available  
  
[root@localhost gateway]# secure_client 127.0.0.1 9999  
Server Information in RED:  
This is Message-2 from the server listening on port: 9999  
Client source port: 34263  
Server Context: user_u:message_filter_r:ext_gateway_t  
Server Peer Context: system_u:object_r:netlabel_peer_t  
  
Client Information in GREEN:  
Client Context: user_u:message_filter_r:ext_gateway_t  
Client Peer Context: system_u:object_r:netlabel_peer_t  
  
[root@localhost gateway]#
```

**Figure 6.1: Running the secure client / server – Once with no NetLabel and once with NetLabel enabled.**

## 7. Appendix C – Labeled IPsec Module Example

### 7.1 Introduction

This section shows a sample IPsec module and configuration files that have been built to support the simple message filter. It is in two parts:

**Manual Configuration** – This shows the files required to configure IPsec manually making all the entries in the SAD and SPD databases. Important note: The encryption keys are pre-generated. If this type of configuration is to be used, then generate new keys as described in the [IPsec-HOWTO](#) [Ref. 12].

**Key Exchange Configuration** – This shows the configuration files required for `racoon` to manage the key exchange and security context. Unfortunately, `racoon` core dumps on F-12 using the `modular-test` policy (but does work with Red Hat targeted policy - The reason seems to be linked with using loopback to run IPsec. When an MCS / MLS policy is used with loopback it works, however if MCS or MLS is not configured it core-dumps).

Notes:

1. F-12 does not have IPsec tools installed as standard, therefore `yum` can be used to install it as shown below:

```
yum install ipsec-tools
```

2. The IPsec configuration files have entries for the Internal Gateway (`int_gateway_t`). If this module is not loaded, then the entries need to be removed.
3. F-12 does not have IPsec services enabled for loopback by default, therefore the following commands need to be run:

```
echo 0 > /proc/sys/net/ipv4/conf/lo/disable_xfrm  
echo 0 > /proc/sys/net/ipv4/conf/lo/disable_policy
```

Be aware though that this re-configuration will only be valid until the next re-boot.

### 7.2 Manual IPsec Configuration

The steps required to install the module and configure IPsec are as follows:

1. Ensure you are logged on as 'root' and SELinux is running in permissive mode (`setenforce 0`) to perform the build process.
2. Produce a `ipsec.conf` loadable module file with a text editor containing the contents shown below:

```
module ipsec 1.0.0;  
#  
#####  
#  
# This Loadable Module will allow Labeled IPsec to manage labeling for #  
# the client / server applications that form part of the test examples #
```

```
# in the SELinux Notebook. #
# #
#####
#
require {
type ext_gateway_t, unconfined_t;
class association { setcontext polmatch sendto recvfrom };
}

# This allows unconfined to set the SPD and SAD context entries:
allow unconfined_t ext_gateway_t : association { setcontext };

# This allows the external gateway to work with Labeled IPSec:
allow ext_gateway_t self : association { setcontext polmatch sendto recvfrom
};

# Allows Racoon running in unconfined_t to polmatch (in fact
# racoon needs to polmatch all entries):
allow unconfined_t ext_gateway_t:association polmatch;

#
##### START OPTIONAL SECTION (for internal gateway) #####
#
optional {
    require {
        # This is defined in the int_gateway.conf module:
        type int_gateway_t;
    }
    allow unconfined_t int_gateway_t:association { setcontext polmatch };
    allow int_gateway_t self : association { setcontext polmatch sendto recvfrom
};
}
#
##### END OPTIONAL SECTION #####
#
```

### 3. Compile and link the new IPSec module:

```
checkmodule -m ipsec.conf -o ipsec.mod
semodule_package -o ipsec.pp -m ipsec.mod
semodule -v -s modular-test -i ipsec.pp
```

### 4. Create an IPSec configuration file (ipsec\_manual\_SA) that will generate both the SAD and SPD database entries that allow IPSec to be configured manually:

```
# setkey -f configuration file entries for MANUAL SA configuration
#
# If the Internal Gateway module (int_gateway.conf) is not loaded,
# then the entries should be removed from this file.
#
# Flush the SAD and SPD
flush;
spdflush;

#
##### Security Association Database entries #####
#
# Important notes:
# 1) The security context (-ctx) entries MUST match
#     the actual running context of the process or it will fail to
#     match (therefore racoon is the best configuration option as
#     these are automatically exchanged).
# 2) If the manual configuration is used in a live environment,
#     then DO NOT use these encryption keys, generate your own.
#
# Authentication Header info
```

```
# AH SAs using 128 bit long keys
add 127.0.0.1 127.0.0.1 ah 0x200
-ctx 1 1 "user_u:message_filter_r:ext_gateway_t"
-A hmac-md5 0xc0291ff014dccdd03874d9e8e4cdf3e6;

add 127.0.0.1 127.0.0.1 ah 0x250
-ctx 1 1 "user_u:message_filter_r:int_gateway_t"
-A hmac-md5 0xc0291ff014dccdd03874d9e8e4cdf3e6;

add 127.0.0.1 127.0.0.1 ah 0x300
-ctx 1 1 "user_u:unconfined_r:unconfined_t"
-A hmac-md5 0x96358c90783bbfa3d7b196ceabe0536b;
#
# Encapsulated Security Payload info
# The -ctx context MUST be exact else get "connect: No such process"
# message when running client:
# ESP SAs using 192 bit long keys (168 + 24 parity)
add 127.0.0.1 127.0.0.1 esp 0x201
-ctx 1 1 "user_u:message_filter_r:ext_gateway_t"
-E 3des-cbc 0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;

add 127.0.0.1 127.0.0.1 esp 0x251
-ctx 1 1 "user_u:message_filter_r:int_gateway_t"
-E 3des-cbc 0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;

add 127.0.0.1 127.0.0.1 esp 0x301
-ctx 1 1 "user_u:unconfined_r:unconfined_t"
-E 3des-cbc 0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df;

#
##### Security Policy Database entries #####
#
# Note that the only part of the security context matched against is
# the 'type' (e.g. ext_gateway_t).

# Security policies for external gateway:
spdadd 127.0.0.1 127.0.0.1 tcp
-ctx 1 1 "system_u:object_r:ext_gateway_t"
-P out ipsec esp/transport//require
ah/transport//require;

spdadd 127.0.0.1 127.0.0.1 tcp
-ctx 1 1 "system_u:object_r:ext_gateway_t"
-P in ipsec esp/transport//require
ah/transport//require;

# Security policies for internal gateway:
spdadd 127.0.0.1 127.0.0.1 tcp
-ctx 1 1 "system_u:object_r:int_gateway_t"
-P out ipsec esp/transport//require
ah/transport//require;

spdadd 127.0.0.1 127.0.0.1 tcp
-ctx 1 1 "system_u:object_r:int_gateway_t"
-P in ipsec esp/transport//require
ah/transport//require;

# Security policies for unconfined_t:
spdadd 127.0.0.1 127.0.0.1 tcp
-ctx 1 1 "system_u:object_r:unconfined_t"
-P out ipsec esp/transport//require
ah/transport//require;

spdadd 127.0.0.1 127.0.0.1 tcp
-ctx 1 1 "system_u:object_r:unconfined_t"
-P in ipsec esp/transport//require
ah/transport//require;
```

5. Activate the IPSec configuration by running setkey:

```
setkey -f ipsec_manual_SA
```

6. The configuration can be checked using the setkey commands as shown:

```
# This command will list the Security Association
# Database entries:

setkey -D

# A list should follow that starts:

127.0.0.1 127.0.0.1
ah mode=transport spi=512(0x00000200) reqid=0(0x00000000)
A: hmac-md5 c0291ff0 14dcccdd0 3874d9e8 e4cdf3e6
seq=0x00000000 replay=0 flags=0x00000000 state=mature
created: Sep 14 16:48:48 2009 current: Sep 14 16:49:10 2009
diff: 22(s)hard: 0(s) soft: 0(s)
last:                hard: 0(s) soft: 0(s)
current: 0(bytes) hard: 0(bytes) soft: 0(bytes)
allocated: 0 hard: 0 soft: 0
security context doi: 1
security context algorithm: 1
security context length: 38
security context: user_u:message_filter_r:ext_gateway_t
sadb_seq=1 pid=3216 refcnt=0
.....
.....
```

```
# This command will list the Security Policy
# Database entries:

setkey -DP

# A list should follow that starts:

127.0.0.1[any] 127.0.0.1[any] tcp
out prio def ipsec
esp/transport//require
ah/transport//require
created: Sep 14 16:48:48 2009 lastused:
lifetime: 0(s) validtime: 0(s)
security context doi: 1
security context algorithm: 1
security context length: 32
security context: system_u:object_r:ext_gateway_t
spid=209 seq=1 pid=3219
refcnt=1
.....
.....
```

7. Because the IPsec service has not been enabled in F-12 for loopback, the following commands need to be run:

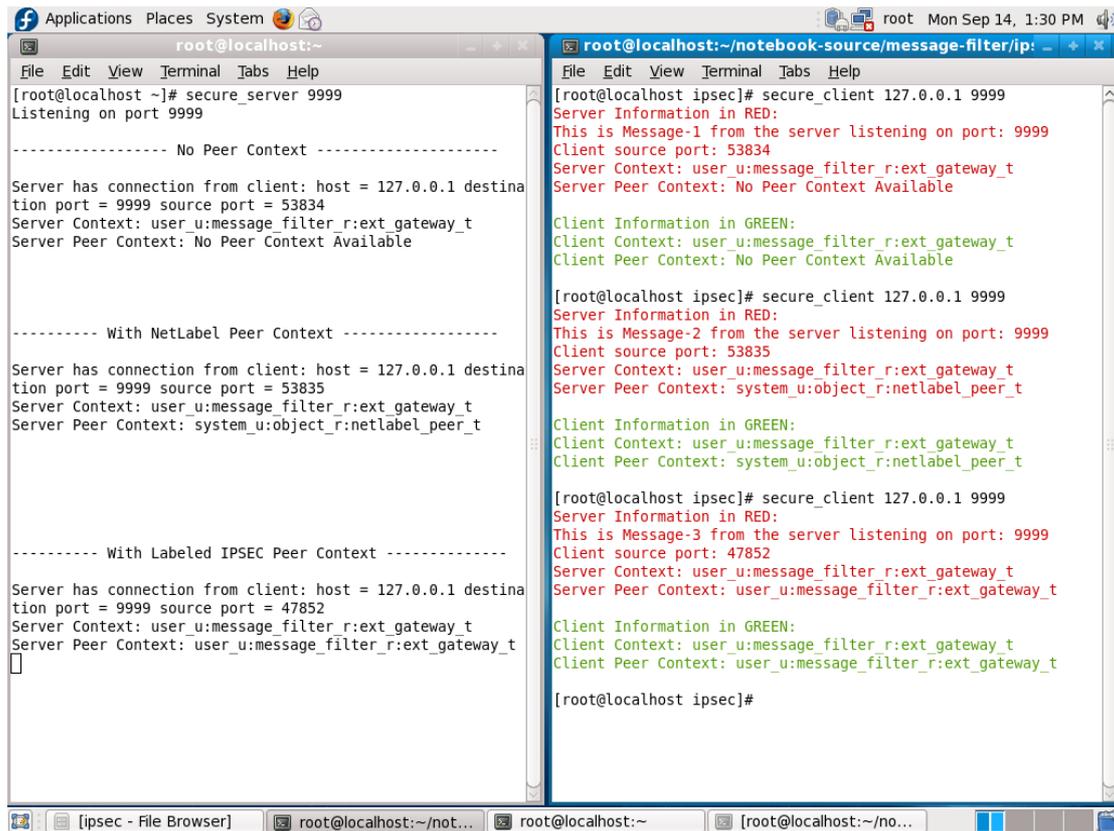
```
# The default for F-12 is that ipsec is disabled for
# loopback. These commands will enable until a re-boot:

echo 0 > /proc/sys/net/ipv4/conf/lo/disable_xfrm
echo 0 > /proc/sys/net/ipv4/conf/lo/disable_policy
```

8. Run enforcing mode:

```
setenforce 1
```

9. Run either the client / server or secure\_client / secure\_server applications as shown in the [SECMARK tests](#). There should now be a peer context displayed as shown in the 'With Labeled IPsec Peer Context' section of [Figure 7.1](#).



**Figure 7.1: Running the secure client / server – Once with no NetLabel, once with NetLabel enabled and once with Labeled IPsec enabled (note that this label takes precedence over the Fallback NetLabel).**

## 7.3 Key Exchange IPsec Configuration

The configuration requirements are shown below, however as mentioned above the racoon IKE daemon will core dump when using the simple policy configured as shown in the [Building a Basic Policy](#) section. The steps required to install the module and configure IPsec are as follows:

1. Perform steps 1, 2 and 3 as for the manual configuration to build the IPsec module.
2. Create an IPsec configuration file (`ipsec_racoon_SA`) that will generate only the SPD database entries. The SAD entries will be populated by racoon as it exchanges the key and context information:

```
# setkey -f configuration file entries for RACOON SA configuration
#
# Flush the SAD and SPD
flush;
spdflush;

#
##### Security Policy Database entries #####
#
# Security policies for external gateway:
spdadd 127.0.0.1 127.0.0.1 tcp
-ctx 1 1 "system_u:object_r:ext_gateway_t"
-P out ipsec esp/transport//require
ah/transport//require;
```

```
spdadd 127.0.0.1 127.0.0.1 tcp
-ctx 1 1 "system_u:object_r:ext_gateway_t"
-P in ipsec esp/transport//require
ah/transport//require;

# Security policies for internal gateway:
spdadd 127.0.0.1 127.0.0.1 tcp
-ctx 1 1 "system_u:object_r:int_gateway_t"
-P in ipsec esp/transport//require
ah/transport//require;

spdadd 127.0.0.1 127.0.0.1 tcp
-ctx 1 1 "system_u:object_r:int_gateway_t"
-P in ipsec esp/transport//require
ah/transport//require;

# Security policies for unconfined_t:
spdadd 127.0.0.1 127.0.0.1 tcp
-ctx 1 1 "system_u:object_r:unconfined_t"
-P out ipsec esp/transport//require
ah/transport//require;

spdadd 127.0.0.1 127.0.0.1 tcp
-ctx 1 1 "system_u:object_r:unconfined_t"
-P in ipsec esp/transport//require
ah/transport//require;
```

3. Activate the IPSec configuration by running setkey:

```
setkey -f ipsec_racoon_SA
```

4. The configuration can be checked using the setkey commands as shown:

```
# This command will list the Security Association
# Database entries:

setkey -D
No SAD entries.

# Note that there should be NO SAD entries as racoon will
# add these during the key exchange process.
```

```
# This command will list the Security Policy
# Database entries:

setkey -DP

# A list should follow that starts:
```

5. Because the IPSec service has not been enabled in F-12 for loopback, the following commands need to be run:

```
# The default for F-12 is that ipsec is disabled for
# loopback. These commands will enable until a re-boot:

echo 0 > /proc/sys/net/ipv4/conf/lo/disable_xfrm
echo 0 > /proc/sys/net/ipv4/conf/lo/disable_policy
```

6. Check the `/etc/racoon/racoon.conf` file. For F-12 its contents should resemble that shown (the commented out sections have been removed). This describes the key exchange as an anonymous exchange and therefore should work with loopback. If the contents of the `racoon.conf` file are different, then save the file and replace the contents with that below:

```
# Racoon IKE daemon configuration file.
# See 'man racoon.conf' for a description of the format and
entries.

path include "/etc/racoon";
path pre_shared_key "/etc/racoon/psk.txt";
path certificate "/etc/racoon/certs";
path script "/etc/racoon/scripts";

sainfo anonymous
{
lifetime time 1 hour ;
encryption_algorithm 3des, blowfish 448, rijndael ;
authentication_algorithm hmac_sha1, hmac_md5 ;
compression_algorithm deflate ;
}
```

7. Start a new virtual terminal session so that the `racoon` service can be run using the command below (in the foreground with debug). If F-12 is being used with the `ipsec-tools` tools, then the chances are `racoon` will core dump once the client tries to contact the server.

```
racoon -Fd
```

8. Run enforcing mode:

```
setenforce 1
```

9. Run either the `client` / `server` or `secure_client` / `secure_server` applications as shown in the [SECMARK tests](#). There should be a peer context displayed as shown in the 'With Labeled IPSec Peer Context' section of [Figure 7.1](#) (however `racoon` core dumps once the client tries to contact the server).

## 8. Appendix D – Implementing a Constraint

### 8.1 Introduction

The objective of this section is to show how a constraint can further limit access permissions. The example given will add a simple role constraint to the base policy described in the [Building the Base Module Policy](#) section by adding the following:

```
constrain process transition ( r1 == r2 );
```

The impact of this constraint will be that when a transition is required, the role of the source (or current) process must be the same as the role of the target process (or new process being exec'ed).

While the majority of applications will load and execute when using the example base policy, when attempting to load the external gateway module described in the [Building the SECMARK Test Loadable Module](#) section, the result will be:

```
# Ensure enforcement mode:
setenforce 1

# Run the external gateway:
secure_server 9999

# The following will be displayed with the constraint enforced:
bash: /usr/local/bin/secure_server: Permission denied
```

This is because the roles are not equal as the source process is the shell running with `unconfined_r` and the external gateway with `message_filter_r`.

### 8.2 Configuration

The following steps are required to add the constraint to the base policy and test the results:

1. Ensure you are logged on as 'root' and SELinux is running in permissive mode (`setenforce 0`) to perform the build process.
2. Edit the `./notebook-source/modular-base-policy/base.conf` file to remove the '#' from the `constrain` statement as shown:

```
#
# This role constraint statement will be used to show limiting
# a role transition in the external gateway. For now comment out:
#
constrain process transition ( r1 == r2 );
```

3. Compile and link the base module so that the `constrain` is enabled:

```
checkmodule -o base.mod base.conf
semodule_package -o base.pp -m base.mod -f base.fc -s seusers -u users_extra
semodule -s modular-test -b base.pp
```

4. Run enforcing mode:

```
setenforce 1
```

5. Run the external gateway as shown:

```
secure_server 9999
```

6. The following error should be displayed as the source and target process roles are not equal:

```
bash: /usr/local/bin/secure_server: Permission denied
```

7. The role constraint (or any other if required) can be tried using different operators such as `!=`, `dom`, `domby` or `incomp` as described in the [constrain statement](#) section. Note the only other operator that will stop the transition is 'domby', however this can be overcome by adding a further line to the base policy to implement the [role dominance rule](#) as shown (but note that this rule has been deprecated and `checkmodule` will issue a warning):

```
# Add the dominance rule after the following statement:
allow unconfined_t self:x_application_data *;
#
dominance { role message_filter_r { role unconfined_r };}
```

8. Once testing has been completed it is recommended that the `constrain` and `dominance` statements are removed and the policy rebuilt.

### 8.3 Reference Policy Constraints Information

The reference policy source has all the constraints listed in the `policy/constraints` file and the MLS / MCS constraints listed in the `policy/mls` and `policy/mcs` files respectively. All these constraints make extensive use of attributes to hold the types to be managed.

For example in the `policy/constraints` file used by F-12, role changing for transitions are managed by the `constrain` statement as shown:

```
#
# SELinux process role change constraint:
#
constrain process { transition noatsecure siginh rlimitinh }
(
  r1 == r2
  or ( t1 == can_change_process_role and t2 == process_user_target )
  or ( t1 == cron_source_domain and t2 == cron_job_domain )
  or ( t1 == can_system_change and r2 == system_r )
  or ( t1 == process_uncond_exempt )
);
```

As can be seen the 'r1 == r2' is what was used in the external gateway example above, however to allow other scenarios, attributes are used to 'hold' the types that can change the constraint conclusion. For example, if the external gateway module was written as a reference policy source module, then to allow the role change:

- The `unconfined_t` domain type could be added to the `can_change_process_role` attribute and the `ext_gateway_t` domain type added to the `process_user_target` attribute.

or

- The `unconfined_t` domain type could be added to the `process_uncond_exempt` attribute.

Either of these would allow the transition to take place.

## 9. Appendix E - GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### 0. Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

### 1. Applicability and Definitions

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

### 2. Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

### 3. Copying In Quantity

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

---

# The SELinux Notebook - Sample Policy Source

---

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## 4. Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

## 6. Collections Of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## 7. Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## 8. Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## 9. Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

## 10. Future Revisions Of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

## 11. Relicensing

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.