SEEdit: SELinux Security Policy Configuration System with Higher Level Language

Yuichi Nakamura, Yoshiki Sameshima

Hitachi Software, Japan

{ynakam,same}@hitachisoft.jp

Toshihiro Tabata

Okayama University, Japan

tabata@cs.okayama-u.ac.jp

HitachiSoft

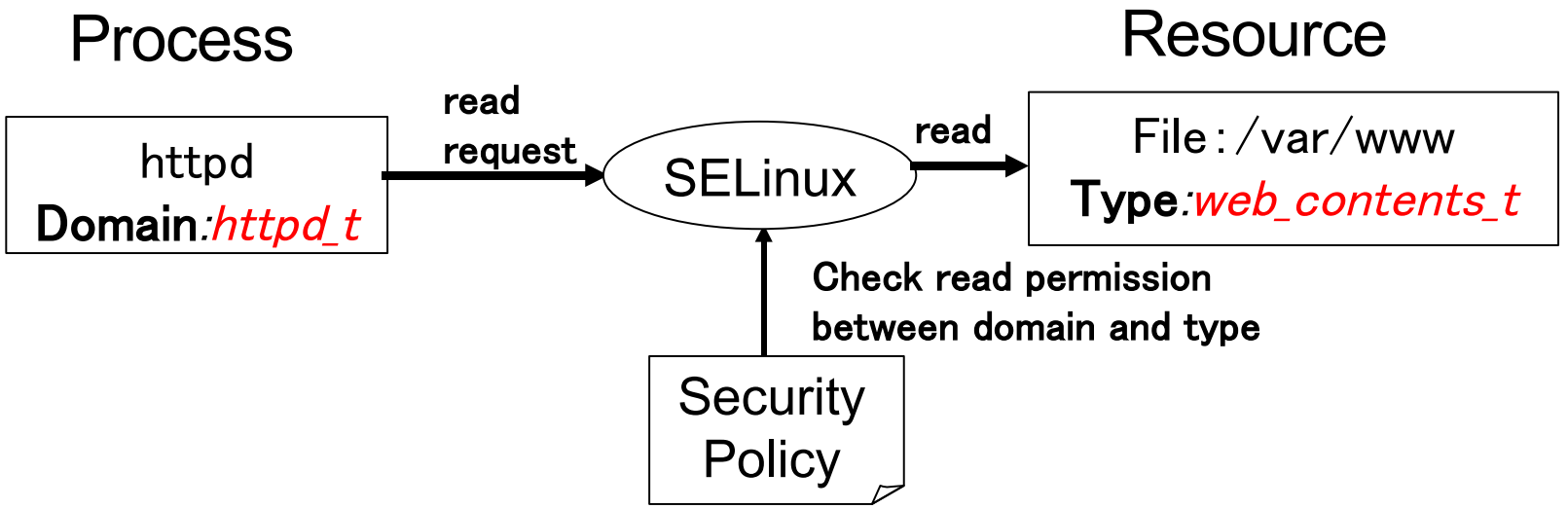
創る、支える、拓く

1. Introduction

What is SELinux?

- Security-Enhanced Linux
  - Developed by NSA (http://www.nsa.gov/selinux)
  - Security enhancement in the Linux kernel layer
- Confine behavior of attackers by access control feature
  - Least privilege (Type Enforcement:TE)
  - Mandatory Access Control (MAC)
    - No one (including root) can avoid
- Widely used for servers
  - Enabled on Redhat, CentOS by default at installation time
  - Also useful for embedded devices
    - Small enough for CE Linux devices, low overhead

TE (Type Enforcement): The Access Control Model



- Label based access control
  - **Domain** labels are assigned to processes
  - **Type** labels are assigned to resources
- The “security policy”
  - Set of access rules are written by SELinux policy language
    - Domain is not allowed nothing by default, only accesses permitted in the security policy are permitted
  - Security policy must be created to use SELinux

allow httpd_t web_contents_t file:{ read };

Domain Type Permission

The purpose of our research

- Bad reputation of SELinux: SELinux is difficult, unusable
  - SELinux is included in major Linux distros, but sysadmins/engineers are often recommended to disable SELinux

- Why? : Security policy configuration is difficult
  - Fine grained permissions (more than 700), label configurations (often more than 1,000), access rules (often more than 100,000)
  - Hard to write, understand

- What we want to do
  - make it easy to write, understand the security policy

2. Problems in the existing method

Existing method against the difficulty of security policy

- Refpolicy : the most popular
  - Developed by the SELinux community
  - Security policies are usually created using refpolicy
- The approach of refpolicy
  - Sample configurations
    - Prepare as many configurations as possible by the power of SELinux community
    - Configurations for most applications in Fedora and Cent OS are covered
  - Macros
    - For the convenience of policy writers, macros are defined to write commonly used sentences in short expressions
- Refpolicy works very well if system is used as expected by refpolicy developers
    - E.g. If we use Cent OS as default configuration, we do not have to do almost nothing for SELinux.

Limitations of refpolicy

- Preparing sample configurations for everything is impossible
  - Customizing refpolicy is necessary in systems that are not expected by refpolicy developers
    - E.g. Commercial applications, embedded system
- To customize, we have to write and understand refpolicy configurations
  - Understanding is also important because people often do not want to use what they can not understand.
- However, writing/understanding refpolicy configurations for is difficult

Why writing/understanding refpolicy is difficult?

- #1 Amount of configuration lines
  - More than 100,000 configuration lines
    - To support as many use cases as possible, configurations for many applications, conditional rules are included
  - Size is also a problem for resource constrained embedded devices
- #2 Number of configuration elements
  - More than 700 Permissions, 1,000 types, 1,000 macros..
- #3 Type configuration
  - Sysadmins have been identifying resources as “file name”, so not familiar with types

* Example:

apache_content_template(sys)

→ A macro. To understand what is configured we have to look for the definition, sometimes definition is nested.

/var/www(/.*)? gen_context(system_u:object_r:httpd_sys_content_t,s0)

→ Type configuration to assign httpd_sys_content_t type under /var/www

3. SEEdit (SELinux Policy Editor)

Our Approach against the security policy problem

- We propose tool “SEEdit”
  - SEEdit = SPDL + SPDL Tools
    - DIY Tool to create the security policy
  - SPDL (Simplified Policy Description Language)
    - Higher level language
    - Reduce number of permissions
    - Hide type configurations
  - SPDL tools
    - Help to write configurations with SPDL
- Write only necessary configurations from zero by SEEdit(without reusing refpolicy), so number of configuration lines and size are expected to be reduced

The Architecture of SEEdit

Configurations written in SPDL
domain httpd_t;
allow /var/www/** r,s;

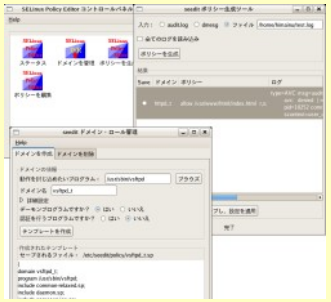
Configurations written in SELinux Policy Language
domain httpd_t,domain;
type var_www_t;
allow httpd_t var_www_t:file r_file_perms;
/var/www(/.*) system_u:object_r:var_www_t

Simplified Policy

SPDL Converter

The Security Policy

SPDL Tools



The SPDL

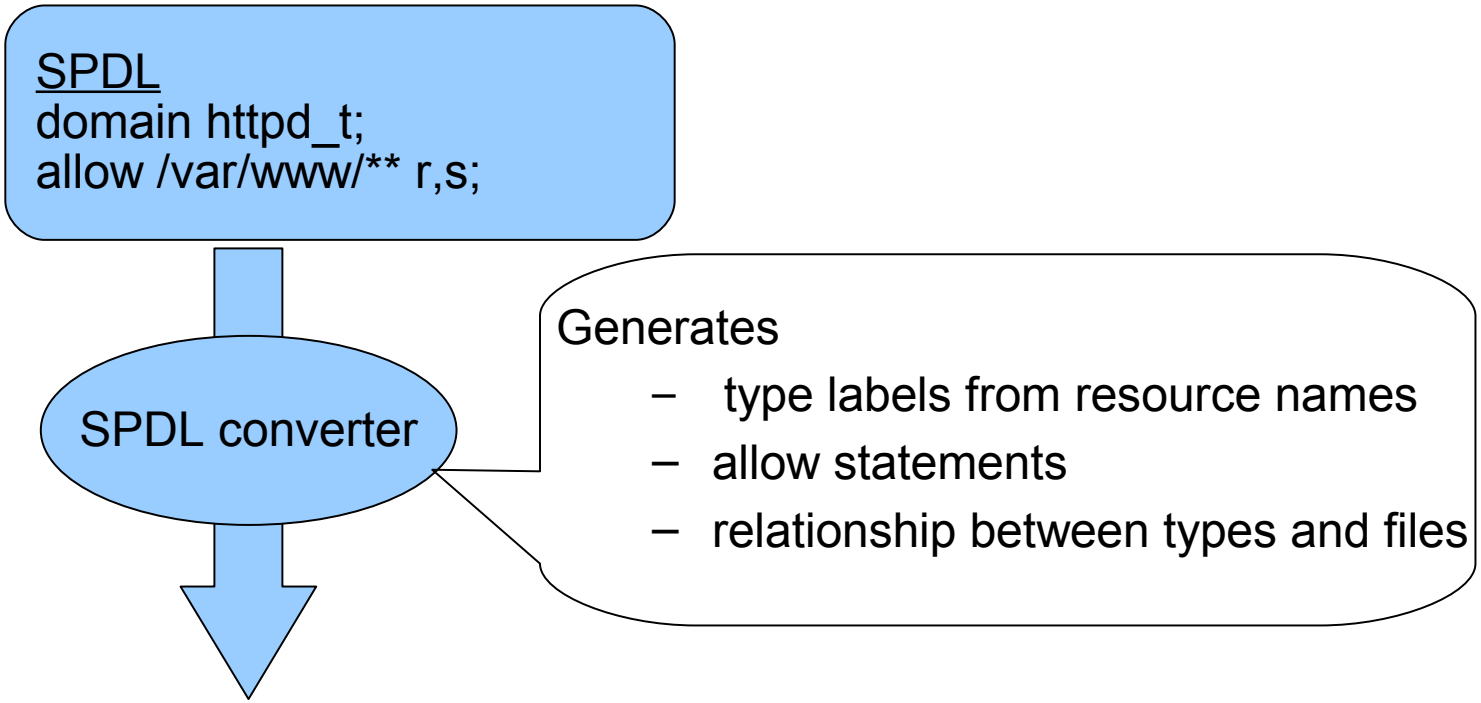
- Type configurations are hidden
  - Identify resources with names not types

- Number of permissions are reduced by Integrated permission
  - Integrated permission “r” for file grants 14 SELinux permissions related to read files

* Example: Granting httpd_t domain read access to files and port 80

```
domain httpd_t;
program /usr/sbin/httpd
allow /var/www/** r;
allownet –protocol tcp –port 80 server;
```

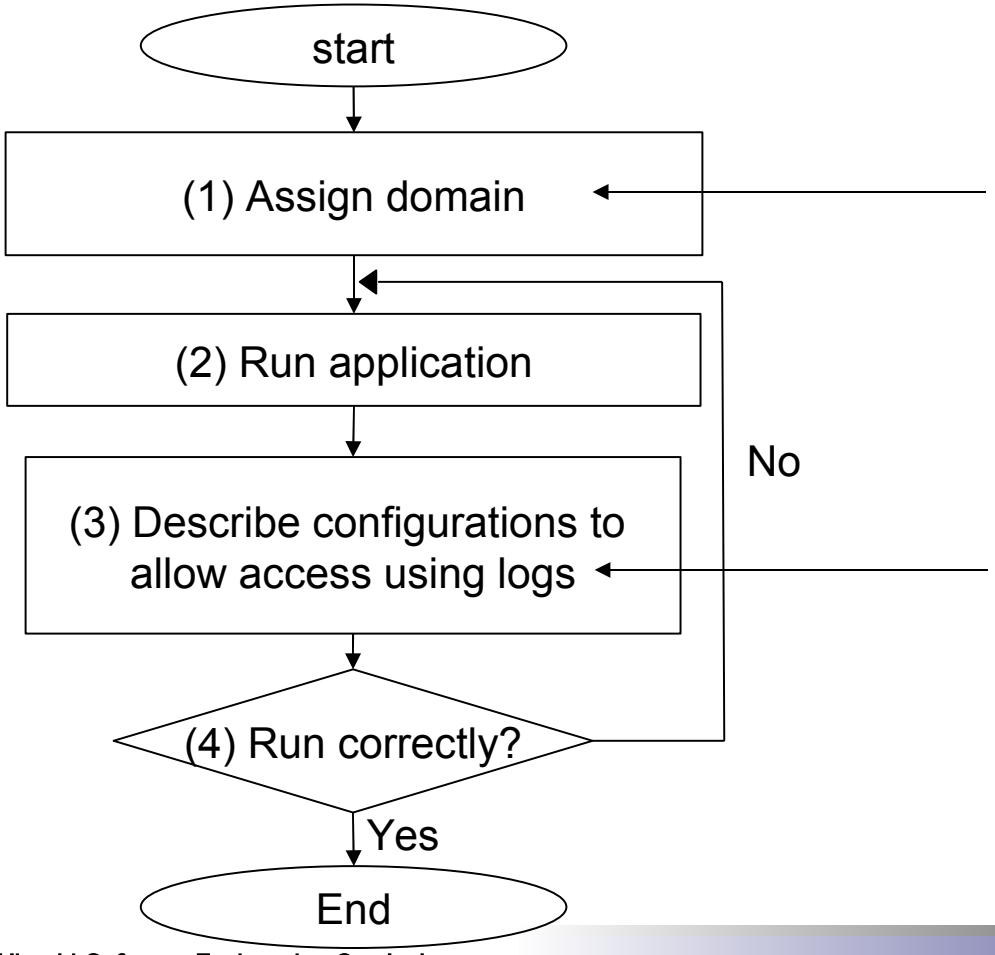
Converting SPDL to SELinux Policy Language



SPDL Tools

- SPDL tools aim to help writing security policy

Typical process of writing the security policy



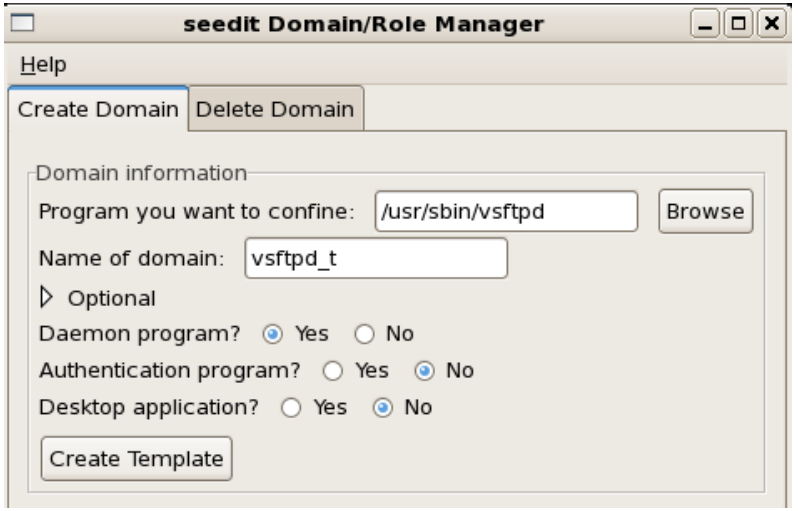
SPDL Tools

Template generator

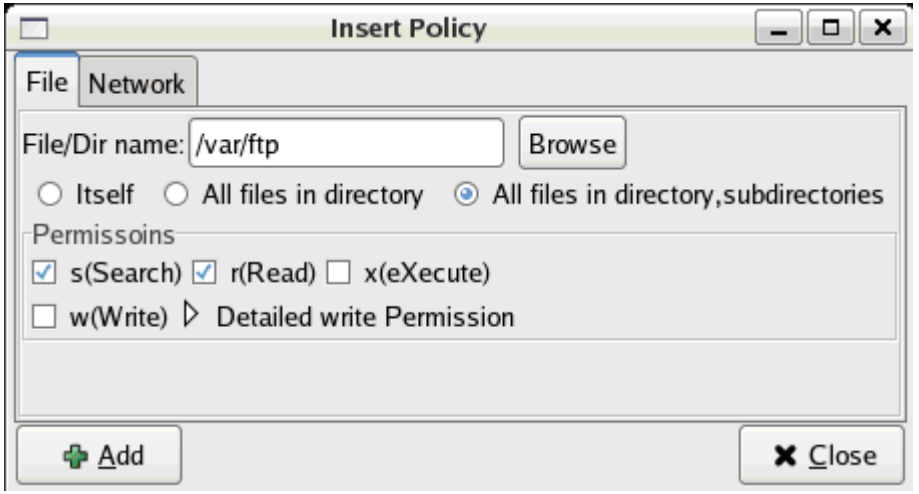
Allow generator

Template generator

- Generate typical configuration
  - Daemon? Desktop application?

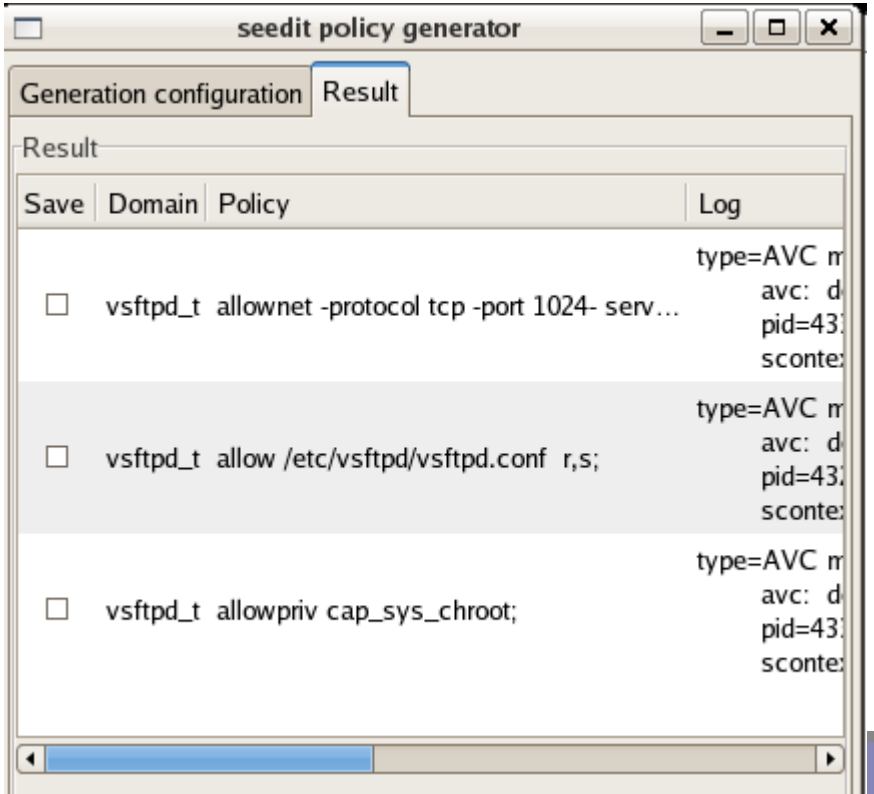


- Input knowledge about the target application
  - What file does it access?
  - What port does it use?



Allow generator

- Generate policy by audit2allow's approach
- Generate configurations from access logs
  - E.g
    - Log : httpd_t domain read accessed /var/www
    - Generated SPDL: in httpd_t, allow /var/www r;
- Do not have to write configurations by hand



4. Evaluation

Experimental setup

- Created policy for PC server system and embedded system

- PC
  - Linux: Cent OS5
  - Running Services:
    - auditd,avahidaemon,crond,cupsd,dhclient,gdm,httpd,klogd,mc-stransd,named,ntpd,portmap,samba,send-mail,sshd,syslogd

  → Configured 16 domains in the security policy
- Embedded System
  - Hardware:
    - CPU: SH7751R@240Mhz, RAM:64MB, FlashRom:64MB
  - Linux: Hand-maid Linux (Linux distribution is not used)
  - Running Services:
    - httpd,vsftpd,syslogd,klogd,portmap

  → Configured 5 domains in the security policy

Writing configurations

- The amount of lines
  - 401 lines for PC system
  - 174 lines for embedded system

  → Does not take so much time to describe such amount

- Number of configuration elements
  - Permissions: 700(before) -> 76(SPDL)
  - Macros: 2,000over(before) -> about 10 statements(SPDL)
  - Type configurations: Necessary(before) -> not necessary (SPDL)

- Template Generator
  - Assuming the tool user knows path of application's config files, log files, port number, 50% configurations are described by the tool.

- Allow Generator
  - Most of configurations generated by the tool could be used without modification

Understanding configurations

Configurations by SPDL

(allow httpd to read /var/www and port 80)

```
# Assign httpd_t domain to http daemon
1 domain httpd_t;
2 program /usr/sbin/httpd;
# Permit httpd_t to read /var/www
3 allow /var/www/** s,r;
# Permit httpd_t to wait connection on  tcp port 80
4 allowcom -protocol tcp -port 80 server;
```

Similar configurations in refpolicy

```
# Assign httpd_t domain to http daemon
1  type httpd_t;
2  type httpd_exec_t;
3  role system_r types httpd_t;
4  init_daemon_domain(httpd_t,httpd_exec_t)
5 /usr/sbin/httpd -- gen_context(system_u:object_r:httpd_exec_t,s0)
# Permit httpd_t to read /var/www
6  apache_content_template(sys)
7 /var/www(/.*)? gen_context(system_u:object_r:httpd_sys_content_t,s0)
8  allow httpd_t httpd_sys_content_t:dir list_dir_perms;
9  read_files_pattern(httpd_t,httpd_sys_content_t,httpd_sys_content_t)
10  read_lnk_files_pattern(httpd_t,httpd_sys_content_t,httpd_sys_content_t)
# Permit httpd_t to wait connection on   tcp port 80
11 corenet_all_recvfrom_unlabeled(httpd_t)
12 corenet_all_recvfrom_netlabel(httpd_t)
13 corenet_tcp_sendrecv_all_if(httpd_t)
14 corenet_udp_sendrecv_all_if(httpd_t)
15 corenet_tcp_sendrecv_all_nodes(httpd_t)
16 corenet_udp_sendrecv_all_nodes(httpd_t)
17 corenet_tcp_sendrecv_all_ports(httpd_t)
18 corenet_udp_sendrecv_all_ports(httpd_t)
19 corenet_tcp_bind_all_nodes(httpd_t)
20 corenet_tcp_bind_http_port(httpd_t)
21 gen_context(system_u:object_r:http_port_t,s0)
```

Size

- In embedded system, size is very important
- Refpolicy based security policies are 2-5MB

- The footprint of created policy for the embedded system
  - File size : 71KB
  - RAM Usage : 465KB

  → Not significant problem

- The size is small because unnecessary configurations are not included, only necessary configurations were described

Usability-Security Tradeoffs

- Integrated permissions
  - Multiple SELinux permissions are merged to one integrated permission, so granularity is reduced.
  - Ex: Integrated permission “r”
    - read permissions to file,symbolic link are merged
    - To allow access to symbolic link not normal file is impossible
  - To solve this, we have to support new SPDL syntax to allow single SELinux permission.

- Audit2allow approach in allow generator
  - Unnecessary accesses may be allowed, if we use generated configurations blindly.
  - Example:
    - If there is a bug in a target application, and the application accesses /etc/shadow by mistake. → Rules allowing access to /etc/shadow is generated
  - We have to check output of allow generator.
    - Some tool to check mistake may be useful

5. Summary

Summary and future works

- Conclusion
  - SEEdit makes it easy to write, understand security polcy configurations with SPDL and SPDL tools.
    - SPDL simplifies syntax to describe security policy configurations
    - SPDL tools help to write configurations by using knowledge of users and access logs.

- Future works
  - Current SEEdit can not be used for refpolicy based security policy
    - Refpolicy can not be reused because SPDL converter can not generate configurations compatible with refpolicy
  - Have to improve SPDL converter to generate configurations appendable to existing refpolicy configurations

- Availability
  - Available at http://seedit.sourceforge.net/
  - Last update of web page is 2008, but code is still updated in 2009. Latest code is available in subversion
    - *svn co https://seedit.svn.sourceforge.net/svnroot/seedit/trunk*