SELinux & AppArmor
- Comparison of Secure OSes

Apr 18 2007

Yuichi Nakamura
Research and Development Department
Hitachi Software Engineering Co., Ltd.
ynakam@hitachisoft.jp

HitachiSoft
創る、支える、拓く

Contents

Background

- Embedded devices are being connected to networks.
  - Attackers can also reach devices
- Security of embedded devices is similar to Win 95.
  - In some devices
    - All processes are running as “root”
    - No password
- What happened to PCs will happen in near future.
  - Worm, virus, crackers…
  - Some devices were already exploited

Threats

- root can do everything
  - Privilege escalation is known even running as normal user
    - such as bugs in suid programs

- PDA, mobile phone
  - If browser open malicious page
    - Virus is executed..
      - Private data is stolen (by wiretap, key logger)
      - Springboard
- Consumer devices (TV, DVD, audio player etc)
  - Attackers can intrude from network interface
    - Download virus with data
    - Destroy system, disclose data, springboard, wiretap etc

Requirement for embedded security

- Embedded devices
  - Restricted resource, Hard to update
- Security technologies
  - Packet filtering
    - Useful, but can not protect open ports
  - IDS, Anti-virus
    - Consumes resources
    - Need update of pattern file, not effective to zero-day attack
  - Secure OS
    - Simple, effective even without security patch
    - Useful for zero-day attack
    - Hardware independent

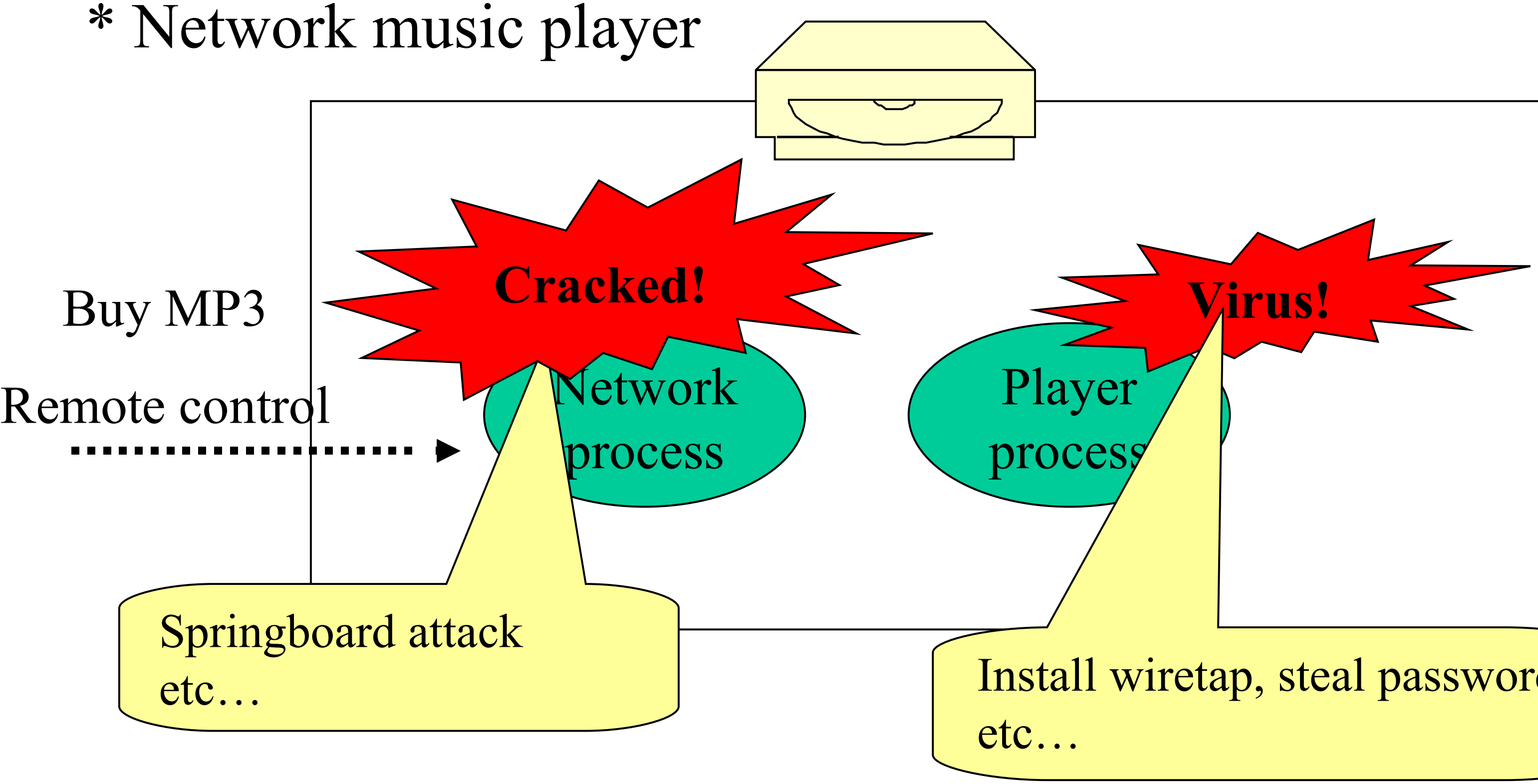
Secure OS

- Access control feature
  - Assign least privilege to process
    - Example: HTTPD can access only homepage file and configuration file.
  - MAC (Mandatory Access control)
    - No one (including root) can not bypass
- Implemented in Linux kernel
- Policy: Important component
  - Configuration of Secure OS: Subject, object, access type

Subject	object	Access type
/usr/sbin/httpd	/var/www	read
/usr/sbin/httpd	/etc/httpd.d	read
/usr/sbin/name	/var/named/	read
d…..		…..

What Secure OS can do?: Before

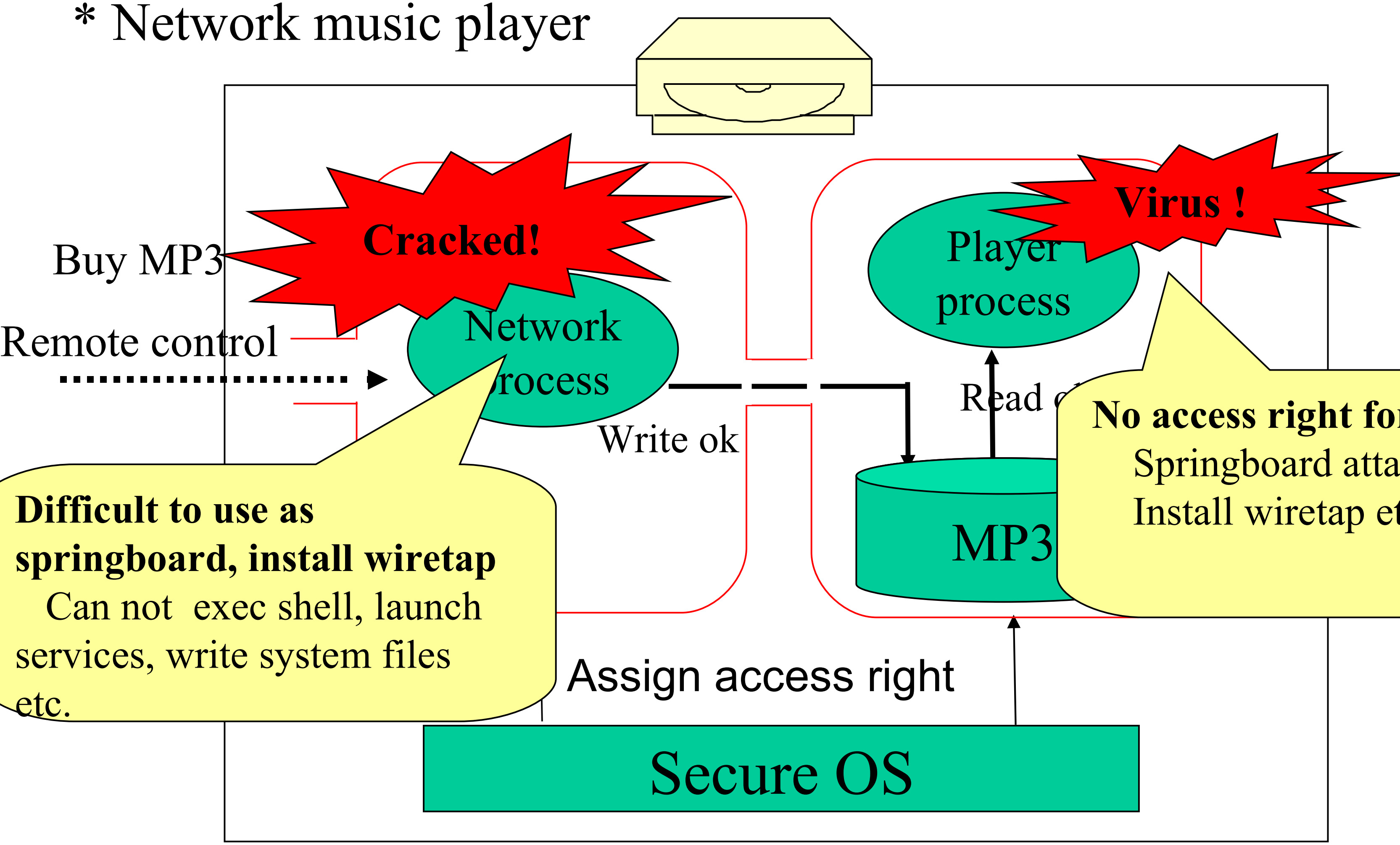
* Network music player



Attackers can do everything

What Secure OS can do?: After

* Network music player



- Attackers/malwares have limited access right
- Effective to Zero-day attacks, without security patch

SELinux and AppArmor

- Two major Open Source Secure OSes
  - Also two extreme
    - security vs. usability

- SELinux - Strict security, hard to use
  - Developed by NSA
  - Included in mainline kernel(2.6), Redhat, Fedora

- AppArmor – Not strict security, easy to use
  - Was called Subdomain, developed by Immunix
  - Now maintained by Novell
  - Included in SuSE Linux

1. Introduction of SELinux & AppArmor

HitachiSoft
創る、支える、拓く

Overview of SELinux

- Access control feature: TE
- Example of policy

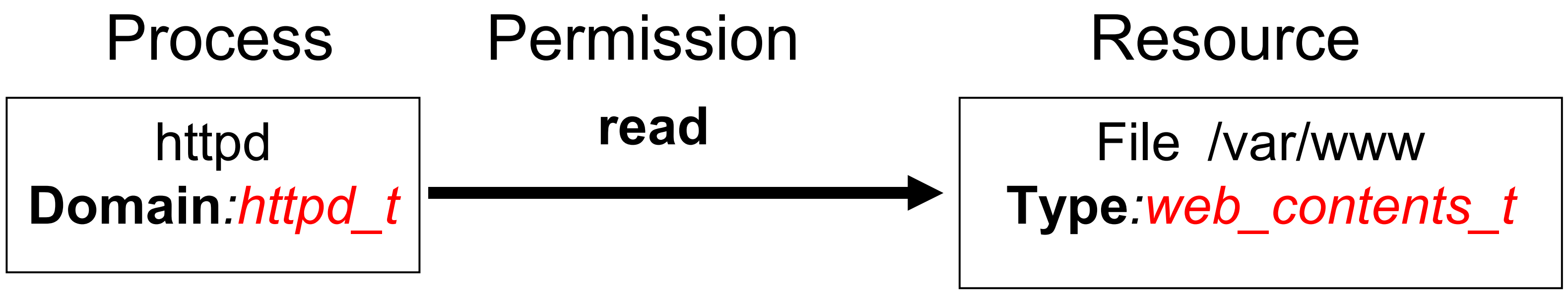
Main feature : TE (Type Enforcement)

Label based access control

Domain Identifier for process

Type Identifier(label) for resources

Controls permission between domain and type



Fine-grained access control

- File, network(port number, NIC, IP), IPC, user, other privilege ..
- About 700 permissions

Configuration of policy

- The most important feature
  - What domain can access what access to what types?

Ex Web server(domain httpd_t :allowing access to homepage

- ◆ allow specify domain, type, permission
  - ◆ **allow** httpd_t web_contents_t file:{ read };

Domain Type Permission

◆Assign label(=type) to resource

/var/www(|/.*) system_u:object_r:web_contents_t

- Many lines of allows(10k-100k) are required
- macro is used
  - Bunch of allows is summarized by macro

Example of policy

●bind.te: allowing acces

```
type named_t;
type named_exec_t;
init_daemon_domain(named_t,named_exec_t)
...
...
kernel_read_kernel_sysctls(named_t)
kernel_read_system_state(named_t)
kernel_read_network_state(named_t)
kernel_tcp_recvfrom(named_t)
....
corenet_tcp_sendrecv_all_if(named_t)
corenet_raw_sendrecv_all_if(named_t)
corenet_udp_sendrecv_all_if(named_t)
corenet_tcp_sendrecv_all_nodes(named_t)
corenet_udp_sendrecv_all_nodes(named_t)
corenet_raw_sendrecv_all_nodes(named_t)
corenet_tcp_sendrecv_all_ports(named_t)
corenet_udp_sendrecv_all_ports(named_t)
corenet_non_ipsec_sendrecv(named_t)
corenet_tcp_bind_all_nodes(named_t)
corenet_udp_bind_all_nodes(named_t)
```

…293 lines

…100 kinds of macros

●bind.fc:assigning label

```
/etc/rndc.*          --   gen_context(system_u:object_r:named_conf_t,s0)
/etc/rndc\.key        --   gen_context(system_u:object_r:dnssec_t,s0)

/usr/sbin/lwresd         --   gen_context(system_u:object_r:named_exec_t,s0)
/usr/sbin/named        --   gen_context(system_u:object_r:named_exec_t,s0)
/usr/sbin/named-checkconf --   gen_context(system_u:object_r:named_checkconf_exec_t,s0)
/usr/sbin/r?ndc        --   gen_context(system_u:object_r:ndc_exec_t,s0)

/var/log/named.*         --   gen_context(system_u:object_r:named_log_t,s0)

/var/run/ndc          -s   gen_context(system_u:object_r:named_var_run_t,s0)
/var/run/bind(/.*)?        gen_context(system_u:object_r:named_var_run_t,s0)
/var/run/named(/.*)?        gen_context(system_u:object_r:named_var_run_t,s0)

ifdef(`distro_debian',`
/etc/bind(/.*)?          gen_context(system_u:object_r:named_zone_t,s0)
/etc/bind/named\.conf  --   gen_context(system_u:object_r:named_conf_t,s0)
```

...45

Difficult to understand

Overview of AppArmor

- Easier than SELinux
- Implemented as LKM

- Recently, often compared with SELinux

Feature

- 1. Access control
  - Controls file and POSIX capability
  - Path name-based
    - Label is not used
  - Profile
    - = “policy”

- 2. GUI Tools
  - Integrated in YaST
    - Generating profile
    - Log report
    - Not so important for embedded ☺

Path name based access control

- Path name based:
  - Identify file with “path name”
  - Easy to understand
- Example:

```
/usr/sbin/httpd{

/var/www/**  r,

}
```

→ /usr/sbin/httpd can read under /var/www

Permission to file

- Basic permission: r,w,x,l
  - r read
  - w : write
  - ix : execute
  - l : link(remove file)

POSIX capability

- Controls capability
  - Capability
    - Important operation other than file access
    - Example:
      - net_bind_service: bind well-known port
      - net_raw: use raw socket
  - For detail: see $man capabilities

Configuration for profile

- Simple, easy to understand

```
/usr/sbin/named {          -> path to exectable
 #include <abstractions/base>
 #include<abstractions/nameservice>
  capability net_bind_service,
  capability setgid,
  capability setuid,
<snip>
  /var/lib/named/** rwl,
  /var/run/named.pid wl,
}
```

Common

Capability

Access to file

2.1 Comparison of feature

HitachiSoft
創る、支える、拓く

Common : LSM

- Both use LSM for implementation
- LSM: Linux Security Module
  – set of hooks in kernel to check security
  – is included in mainline from 2.6

- Using LSM:
  – SELinux, AppArmor, LIDS (for 2.6)
- Not using
  – TOMOYO Linux, LIDS (for 2.4)

Difference between SELinux and AppArmor

- Granularity of permission
  - SELinux:
    - File, network, IPC, POSIX capability etc..
  - AppArmor
    - File + POSIX capability
  - AppArmor can reach SELinux in theory, because both use LSM.

- How to identify resource
  - The most fundamental -> next

How to identify resource

- Fundamental difference
  - Affects security and usability
- Label based vs Path name based
  - Label: lower usability, higher security
    - Assign label to file
    - SELinux
  - Path name: higher usability, lower security
    - Identify file with path name
    - AppArmor, TOMOYO Linux
- Compare them by showing benefit and loss of pathname

Benefit of path-name

- High usability, easy to understand
- No need to extend file system
  - Label base: File system have to be extended to store label
- Implementing policy generation tool is easier
  - -> Next
- Nothing happens when i-node number is changed
  - -> Next

Benefit of path-name: policy generation

- Example case:
  - PHP trid to write /var/www/html/write/test.txt
  - But, access denied by Secure OS
  - Have to generate policy from log
- SELinux
  - 1) label under /var/www/html -> httpd_sys_content_t
  - 2) Log says..
    - httpd_t was denied to write to httpd_sys_content_t
  - 3) Generate policy from log
    - allow httpd_t httpd_sys_content_t:file write;
    - - > allowing write access whole “/var/www” !
  - <u>Unnecessary access is granted</u>
- AppArmor
  - 1) log says
    - /usr/sbin/httpd is denied to write /var/www/html/write/test.txt
  - 2) Generate policy(=profile) from log
  - /usr/sbin/httpd{

    /var/www/html/write/test.txt w,
  - <u>Unnecessary access is _not_ granted</u>

Benefit of path-name change of inode number

- Example /etc/mtab
- SELinux : Label is lost when inode number is changed
  - Label is associated with inode
    - /etc/mtab
    - vi, rpm changes inode
  - Solution
    - “file type transition” configuration
      - Not easy for beginner
    - Some userland have to be extended
      - Example: rpm ,vi
- AppArmor
  - No problem!

Loss by path-name

- Information flow
- tmpfiles

Loss by path-name Information flow analysis

-> Who can access the information?

- Some people say path-name based security is broken because of this
- Ex: Information flow analysis to password information
  - Initial state: Stored in /etc/shadow
  - If hardlink is created to /etc/shadow, password information can be accessed via hardlink
  - What happens in information flow analysis?
    - Have to traverse whole file tree to find hardlink
    - What if more hardlink is created during travarsal ?
  - SELinux:
    - All you have to do is to check what kind of domain can access label for /etc/shadow
    - Label is the same for hardlink

Loss by path-name tmp files

- When creating randomly named file under /tmp

- SELinux
  - Can identify such file by naming label such as httpd_tmp_t

- AppArmor
  - How to identify randomly named files?
    - result in allowing whole /tmp.

SELinux Policy Editor(SEEDIT) (1)

- Tool that makes SELinux easy
- Open Source: http://seedit.sourceforge.net/
  - Originally developed by Hitachi Software
  - Included in Fedora repository
- Main feature: SPDL
  - AppArmor-like syntax to write policy
  - example:
    - domain httpd_t
    - program /usr/sbin/httpd;
    - allow /var/www/** r; ← path-name configuration
  - This is converted to SELinux policy syntax
    - type var_www_t; ← label is generated
    - allow httpd_t var_www_t { file dir }: read;

SELinux Policy Editor(SEEDIT) (2)

- Still different from AppArmor

- Inherit drawback from label-based access control
  - change of inode
    - generated policy is label based
- Inherit good points from SELinux
  - fine-grained permission (IPC, network)
  - no patch to kernel

- Now, I am porting SPDL to work on embedded device
  - I can demo for you after presentation!
  - I hope I can release in future (not sure when)

2.2 Porting SELinux/AppArmor to embedded devices

HitachiSoft
創る、支える、拓く

Target device

- Sharp Zaurus SL-C3200
  - CPU: Intel XScale 416Mhz
  - Memory: 64MB
- Distro: Open Zaurus 3.5.4.2-rc2

- Experiences of porting SELinux and AppArmor

Kernel

- SELinux
  - No work is needed! included in mainline
- AppArmor
  - Have to obtain patch from
    - http://developer.novell.com/wiki/index.php/Novell_AppArmor
  - Very easy to patch

diffstat:

fs/namespace.c	|	3
include/linux/audit.h	|	5
include/linux/namespace.h	|	3
kernel/audit.c	|	6

All others: security/apparmor

File system

- SELinux:
  - File system must support xattr
    - ext2, ext3 supports xattr
    - after 2.6.18 jffs2 supports xattr
    - Fortunately, SL-C3200 uses ext3 ☺
- AppArmor:
  - No extension needed!

Userland

- SELinux
  - Many commands
    - load_policy, setfiles, restorecon, chcon etc..
    - Might want them to port to BusyBox to reduce size
  - libselinux
    - APIs for SELinux commands
- AppArmor
  - Only apparmor_parser
    - Profile loader
    - Some helper shell script may needed for convenience
- cross-compile with minor modification

Policy

- SELinux
  - Difficult to use sample policy (refpolicy)
    - Intended for server use
    - Need a lot of customize
    - Difficult to understand, describe
  - I used SELinux Poilcy Editor's simplified policy(SPDL)
- AppArmor
  - Much easier than refpolicy
    - Like SPDL
- Policy generation tool
  - Not available for both
    - python or perl is required
  - Have to write by hand.

2.3 Performance

Experiment

- Prepared domain/profile for 7 apps

- Memory usage
- Storage usage
- Unixbench/lmbench
- Compared with no SELinux/AppArmor kernel

Memory usage

- free command

- AppArmor
  - +1M
- SELinux
  - +1.7M
- Both need work (TODO)

Storage usage

- Total
  - SELinux + 757k(no tuning) – +244k(with tuning)
    - -> Tuning is important
  - AppArmor +157k (tuning not tried yet)

lmbench

	Overhead of AppArmor(%)	Overhead of SELinux(%)
simple syscall	0.6	0.4
simple read	31.3	74.3
simple write	42.9	98.7
simple stat	30	54.8
simple fstat	5	45.9
simple open/close	114.5	44.8
pipe latency	8.7	12.6
process fork+exit	1.9	2.6
process fork+execve	17.6	6.8
process fork+/bin/sh -c	18.2	18.1

AppArmor: overhead in file open, exec

SELinux: overhead after file open, exec

Unixbench

	Overhead of AppArmor(%)	Overhead of SELinux(%)
DhryStone 2 using register variables	0	0
Double-Precision Whetstone	0	0
Execl	15.3	5.7
FileCopy(256buf)	6.4	13.9
1024buf	0.6	8.7
4096buf	0	2.9
Pipe Throuput	5.6	24.6
Pipe-based context switch	3.9	11.7
Process creation	0	1.4
Shell scripts	19.3	30.3
System call overheads	0	0

Less overhead than null I/O

→ ???

3．SELinux activities in Japan

Our project

- Project in Japan SELinux Users Group (JSELUG)
- Our goal
  - Prepare SELinux platform, development kit for embedded devices
- 2 projects
  - seBusyBox(on going), SEDiet(not public yet)
- Developers
  - Current active
    - Yuichi, KaiGai, Shinji
  - Some other people are involved in discussion
- If you are interested in our project:
  - busybox atmark kaigai.gr.jp

seBusyBox

- Porting SELinux commands to BusyBox
- Submitted patch to BusyBox upstream
  - Accepted: coreutils, libselinux
  - On going: policycoreutils, netstat, find
- We found implicit guidelines of BusyBox
  - such as indent rule, usage of libbb
  - Japanese site, sorry:
    - http://www.kaigai.gr.jp/index.php?busybox_upstream

SEDiet

- SEDiet (SELinux Diet):
  - Activity to reduce size of SELinux
  - Reducing size of policy, userland
  - In progress.
    - Submitting patch to diet libselinux

  - More presentation in near future??

Summary

- SELinux -> more security, less usability
- AppArmor -> less security, more usability

- SELinux needs more work, but community can change it!
  - Project in progress
  - SELinux Policy Editor can simplify SELinux
  - SELinux community is bigger, upstreamed
    - More eyeballs, better implementation, more reputation
    - Let’s contribute ☺

Questions/Suggestions ?

Linux is a registered trademark of Linus Torvalds in the US and other countries

Red Hat is a registered trademark of Red Hat ,Inc in the US and other countries

SUSE is a registered trademark of SUSE LINUX AG in the US and other countries

AppArmor is a registered trademark of Novell, Inc in the US and other countries.

TOMOYO is a registered trademark of NTT Data corporation.

Other names of products, services and companies are the trademarks of their respective companies.