

TOMOYO Linux News!

*“A Lightweight and Manageable Security System
for PC and Embedded Linux”*

PDF version of ELC2007 slide is available:

CE Linux Forum Wiki **“TomoyoLinux”**

<http://tree.celinuxforum.org/CelfPubWiki/TomoyoLinux>

(bookmark and monitor the page, *now!*)

ELC2007 Program very Late Additions:

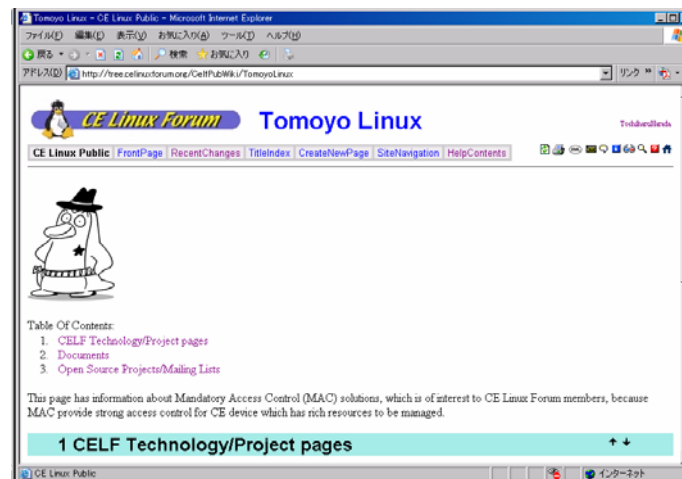
TOMOYO Linux Demonstration 19-340-C

See You again in Japan!:

TOMOYO Linux Project will have a session at upcoming **Japan Technical Jamboree 14 (April 27, 2007)**.

See You in Ottawa, too:

TOMOYO Linux Project will have a BoF session at upcoming Japan **Ottawa Linux Symposium 2007**.



CE Linux Forum
[Worldwide Embedded Linux Conference 2007](#)
April 17-19, 2007

TOMOYO Linux

*“A Lightweight and Manageable Security System
for PC and Embedded Linux”*

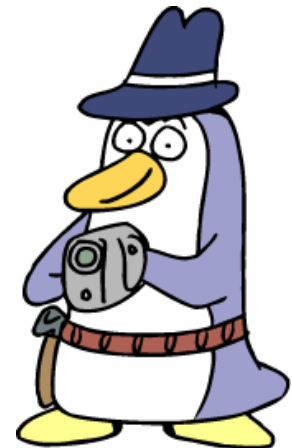
<http://tomoyo.sourceforge.jp/>



Toshiharu Harada
Tetsuo Handa
NTT DATA CORPORATION



<http://www.nttdata.co.jp/en/index.html>



Hello, world!

- This is the very first presentation abroad for our work, “TOMOYO Linux”.
- TOMOYO Linux is a MAC (Mandatory Access Control) implementation for Linux.
- TOMOYO Linux consists of a set of patches for kernel 2.4/2.6 and a couple of administrative tools. It has been ported to various distributions.
- TOMOYO Linux is available at <http://tomoyo.sourceforge.jp/> under the GPL license.
- You can browse&search the code at <http://tomoyo.sourceforge.jp/cgi-bin/lxr/source>

About NTT DATA

- <http://www.nttdata.co.jp/en/index.html>
- One of the largest SI companies in Japan.
 - Data
 - Established: May 23, 1988
 - Ordinary Income: **42,016 million yen**
 - Number of Employees: **8,406**
 - Common Stock: **142,520 million yen**
 - Net Sales: **907,281 million yen**
 - (see <http://www.nttdata.co.jp/en/aboutus/a09.html> for more detail)
- TOMOYO Linux project:
 - Launched March 2003.
 - Members: started from 2. currently 5.
 - Exceptionally small project in the largest company. :-)

TOMOYO Linux: Background

- Originated from NTT DATA CORPORATION R&D.
- Project started in March 2003.
- First public release in November 2005.
- Ported to Debian Sarge/Etch, RedHat Linux 9, Fedora Core 3-6, CentOS 4.4/5, OpenSUSE 10.1/10.2, Asianux 2.0, Ubuntu 6.10/7.04 and more (check the link below).
 - <http://sourceforge.jp/projects/tomoyo/files/>
- Suits well to embedded systems.

Topics Covered in This Session:

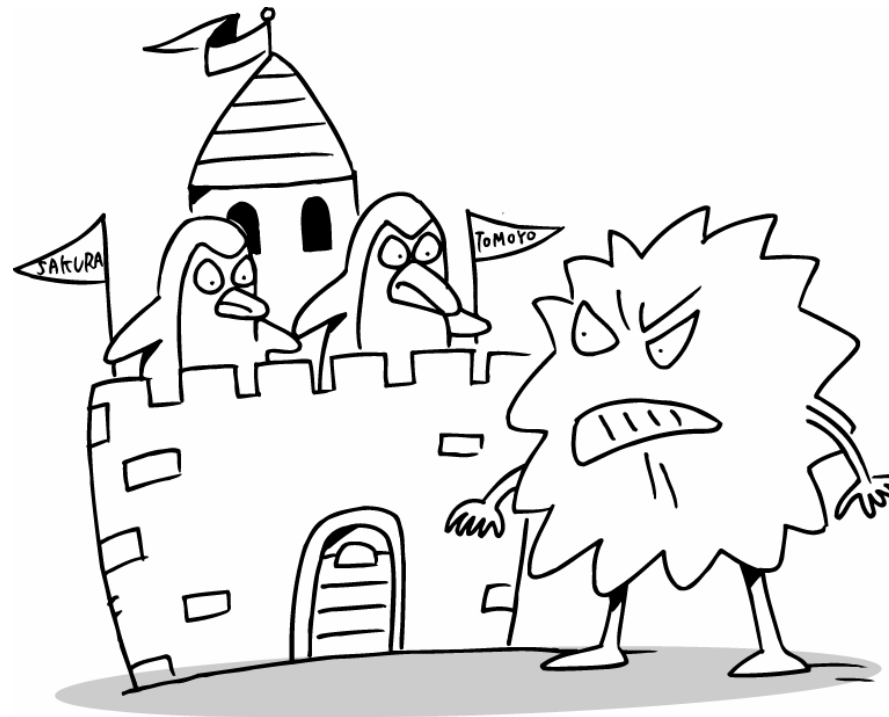
- Overview of TOMOYO Linux and MAC.
- What TOMOYO Linux can do/can't do.
- Concepts.
- Automatic policy configuring feature.
- TOMOYO Linux policy.
- How it works.
- Short demonstration.
- Comparison to other security enhanced OSs.

Topics NOT Covered in This Session:

- Whether name-based access control is evil or not.
 - It's too technical and complex.
 - TOMOYO Linux will have a BOF session in the upcoming Ottawa Linux Symposium. Please join and *help/save us!*
- In-depth demonstration.
 - Will be shown in the following tutorial session, don't miss it! Or please consider installing binary package yourself (10 minutes job).
 - <http://tomoyo.sourceforge.jp/en/1.4/install.html>

Part 1

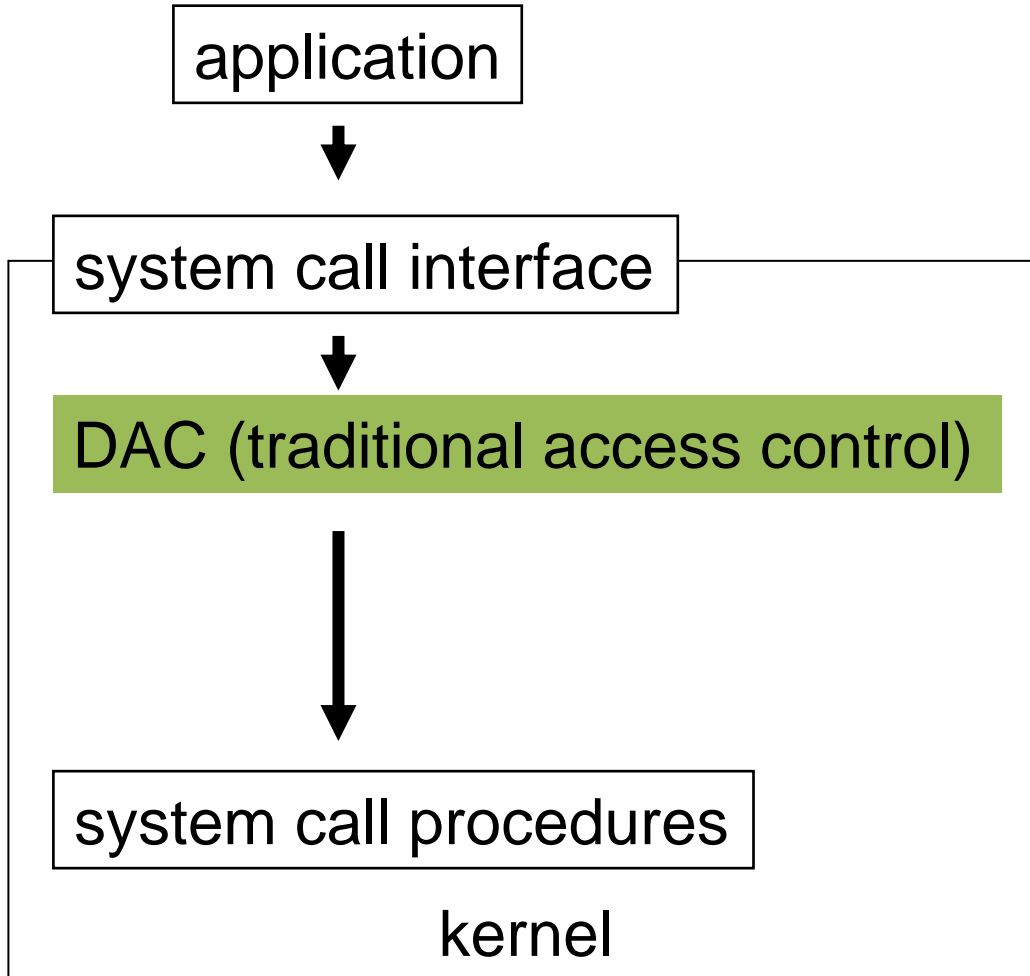
MAC and TOMOYO Linux Overview



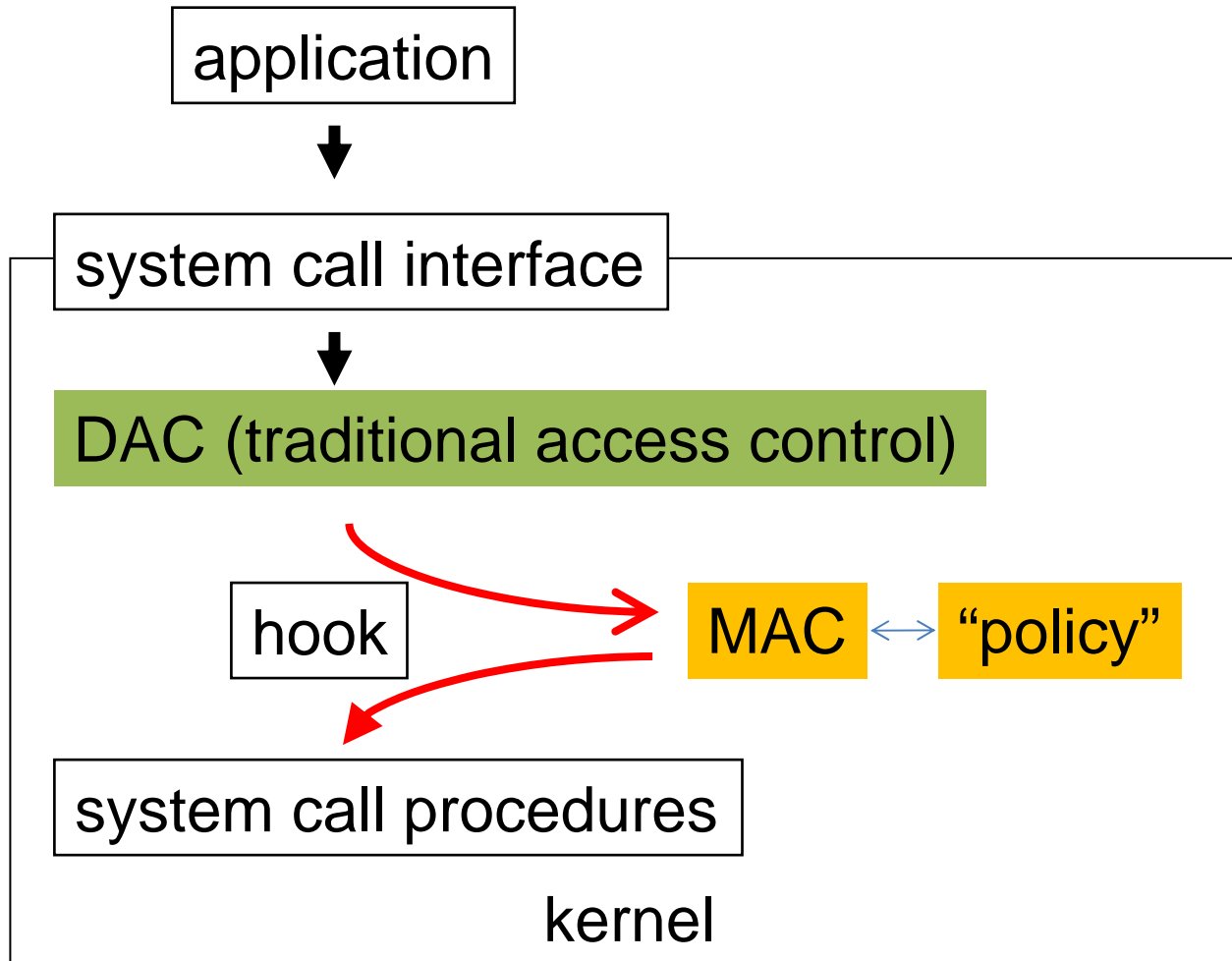
What is “MAC”?

- “Mandatory Access Control”
 - No exceptions even for “root” users and no bypass.
 - Accesses are strictly judged according to the rules (called “policy”).
 - Traditional access control is referred as DAC (Discretionary Access Control).
- Implementation.
 - (1) “Hook” a request (e.g. system call).
 - (2) “Judge” whether to execute the request or not.
 - (3) “Process” the request.
 - 2.6 kernel has a built-in framework for hooks – LSM (Linux Security Modules).

What is “hook”?



What is “hook”?



What is “policy”?

- DAC checks “rwx” attributes associated with filesystem. (very simple)
- MAC engine needs “rules” to make decision.
- “Policy” is the name for the rules.
 - MAC is useless without good policy.
- No standard policy syntax exists (so far).
- Policy is defined as a set of “conditions”.
 - *if (condition) then Access is (granted/denied).*

MAC by Animation



NTT DATA CORPORATION

What is TOMOYO Linux?

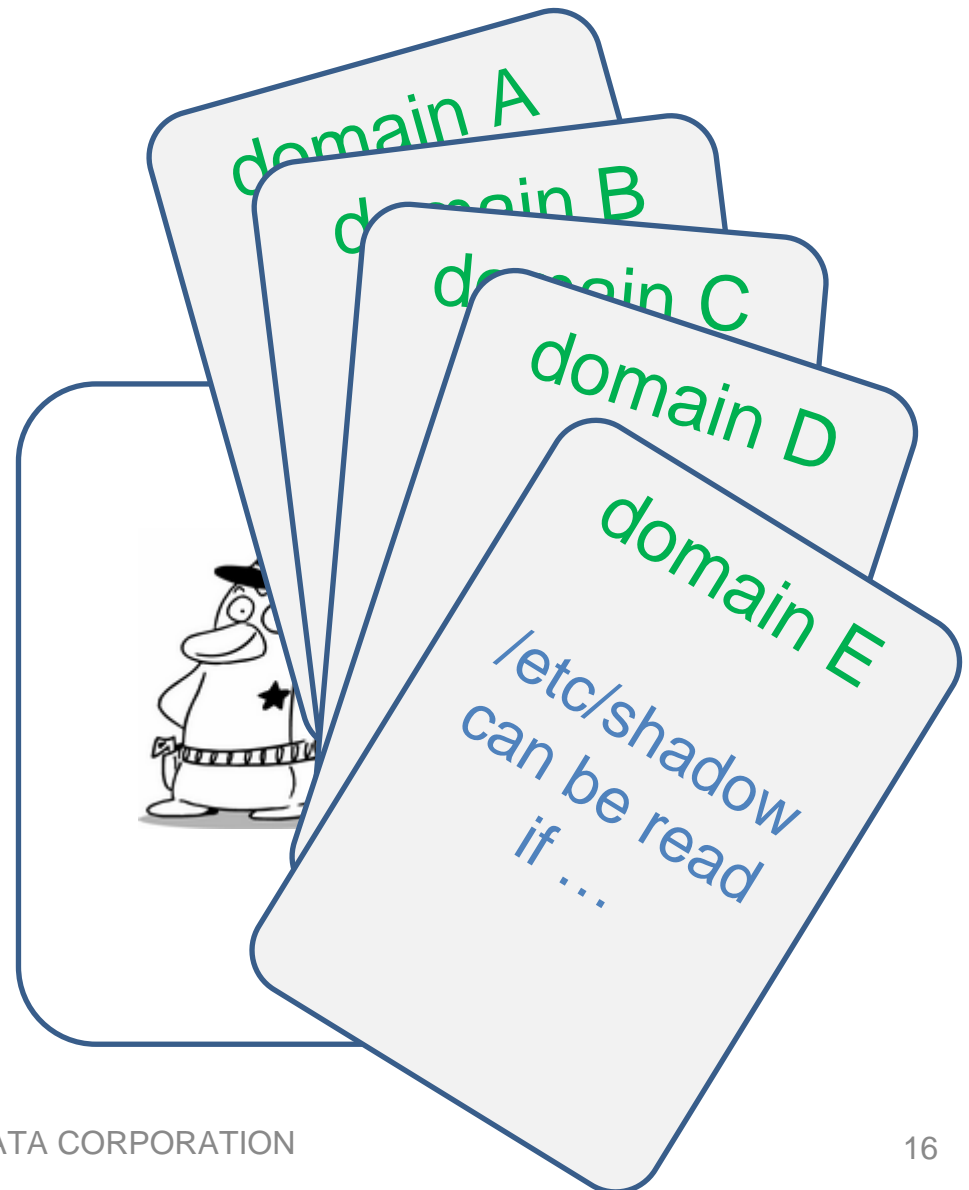
- “Lightweight” and “usable” Mandatory Access Control for Linux, with
 - “automatic policy configuring” feature.
 - administrators friendly policy language.
 - English documentation.
 - 2.4 kernel and BusyBox support.
 - no filesystem limitations (runs with any filesystem).
 - no need of libselinux nor userland program modifications.

What is TOMOYO Linux?

- TOMOYO Linux
 - is not using LSM (Linux Security Modules) .
 - Some work is in progress.
 - does not have MLS (Multi Level Security), RBAC (Role Based Access Control) mechanisms..
 - If you need MLS, use SELinux.
 - RBAC like usages are available.
 - is not included in Linux kernel source.
 - yet
 - can do unusual things and playing with TOMOYO Linux is so much fun.
 - is no guarantee. (use at your own risk, of course)

What is “domain”?

- Whether a request is legal or not depends on the “context”.
- Most MAC system refers this context as “domain”.
- “Domain” is a kind of group/unit.



TOMOYO Linux Policy

- Exceptionally simple. Any Linux/UNIX users can read and write it.

`<kernel> /usr/sbin/sshd /bin/bash /bin/csh`

is the current “domain”

```

1 /bin/grep
1 /bin/sed
4 /dev/null
4 /etc/csh.cshrc
4 /etc/group
4 /etc/nsswitch.conf
...
1 /usr/bin/[
1 /usr/bin/dircolors
1 /usr/bin/id
1 /usr/bin/test

```

mode path

mode:
1 --x
2 -w-
4 r—
6 rw-

`/bin/csh` process
that was invoked from `/bin/bash`
that was invoked from `/usr/bin/sshd`

“<kernel>” is the virtual bottom

SELinux Policy

<p>● bind.te: allowing acces</p> <pre> 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 ...100 kinds of macros </pre>	<p>● bind.fc: assigning label</p> <pre> /etc/ndc.* -- gen_context(system_u:object_r:named_conf_t,s0) /etc/ndc\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 </pre>
--	--

From Japan Technical Jamboree 12.

“*.te”: access control definitions for “*”.

“*.fc”: label definitions for “bind”. “fc” stands for “file context”.

Policy is described in terms of label, not path name. (but label definition is described in terms of path names). Macros are introduced to make policy more readable. SELinux kernel needs policy to be compiled before use.

AppArmor Policy

```
/usr/sbin/named {    -> path to executable
#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

From Japan Technical Jamboree 12.

- Above definition is applied to every instance of `/usr/sbin/named` (No process invocation history mechanism exists with AppArmor).
- In TOMOYO Linux, “<kernel> /foo /bar /usr/sbin/named” and “<kernel> /boo /bar /baz /usr/sbin/named” is distinguished and treated independently.
- SELinux simply cares about “label”.

AppArmor Policy (cont.)

- Similarities with TOMOYO Linux:
 - Both use “pathname based” definition.
- Differences:
 - TOMOYO Linux distinguishes domain by a process invocation history while AppArmor does by a single process.
 - TOMOYO Linux provides MAC for network and signals.

Domains in TOMOYO Linux

- In TOMOYO Linux
 - every process belongs to a domain.
 - every process remembers its ancestors.
 - ACL are controlled by the domain, not by a current process name.
- Utility program, “ccstree”, prints out domain information for running processes.

```
0 +- sshd (2859) <kernel> /usr/sbin/sshd
0 +- sshd (3807) <kernel> /usr/sbin/sshd
1 +- bash (3809) <kernel> /usr/sbin/sshd /bin/bash
2 +- ccstree (3942) <kernel> /usr/sbin/sshd /bin/bash /root/ccstools/ccstree
```

TOMOYO Keeps Track of Process Invocation History

<kernel> /sbin/mingetty /bin/login /bin/bash /bin/ls



<kernel> /sbin/mingetty /bin/login /bin/bash



<kernel> /sbin/mingetty /bin/login

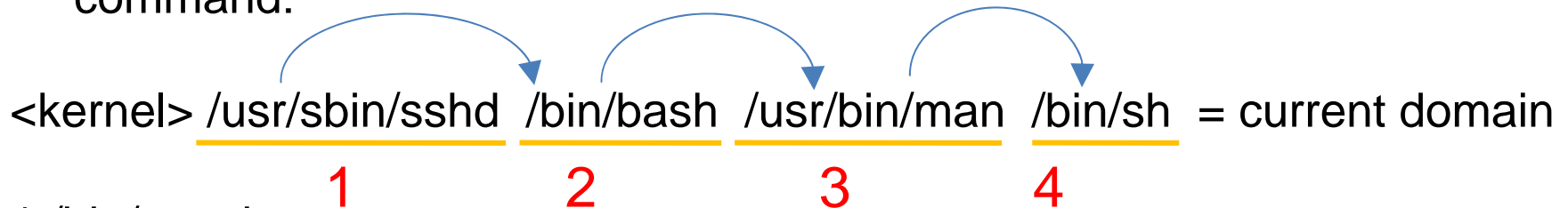


<kernel> /sbin/mingetty



A Slightly Complicated Example

Suppose that you are logged-in into a ssh server and execute a *man* command:



1 /bin/gunzip

2 /dev/null

6 /dev/tty

4 /etc/mtab

1 /usr/bin/bzip2

1 /usr/bin/gtbl

1 /usr/bin/less

1 /usr/bin/nroff

2 /var/cache/man/cat1/pstree.1.bz2

allow_truncate /var/cache/man/cat1/pstree.1.bz2

Automatic Policy Generation

- How to use:
 1. Change “mode (explained later)” of TOMOYO Linux kernel to “policy generation”.
 2. Just operate as you want.
 3. TOMOYO Linux generates the required access control lists as policy.
 4. Check it and modify it.
 5. Done
- Advantages
 - Saves time enormously.
 - The resulting policy is quite “readable”.



Demo

Come and experience TOMOYO Linux at:

19-340-C

Automatic Policy Generation

- How it works?
 - TOMOYO kernel keeps track of:
 - access requests (by original hooks).
 - process invocation history (something like *ps*tree originated from /sbin/init) – every process knows its ancestors under TOMOYO Linux.
 - assigns each individual “process invocation history” as a “domain”.
 - ex. “<kernel> /sbin/init /etc/rc.d/rc /etc/rc.d/rc.local”



Life with TOMOYO Linux

- Administrating TOMOYO Linux is easy and fun.



Life with Other MAC Systems

- Why it's not working?
- Is it failure of my APP or my policy? How am I suppose to find?
- I'd rather disable it ...



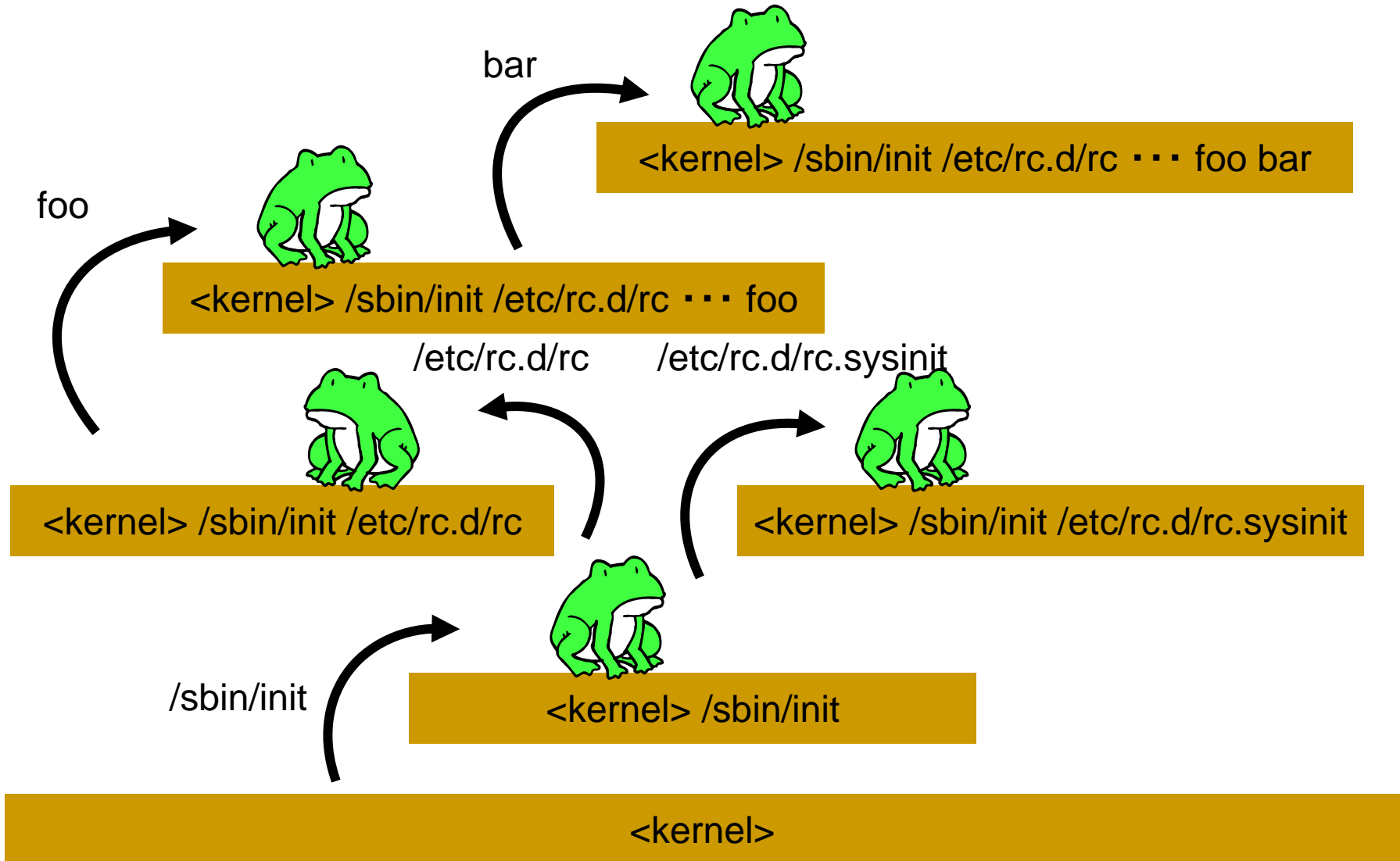
SELinux FAQ says:

- *The security of an unmodified Linux system depends on the correctness of the kernel, all the privileged applications, and each of their configurations. A problem in any one of these areas may allow the compromise of the entire system.*
- *In contrast, the security of a modified system based on the Security-enhanced Linux kernel depends primarily on the correctness of the kernel and **its security policy configuration**.*

Domain Transition

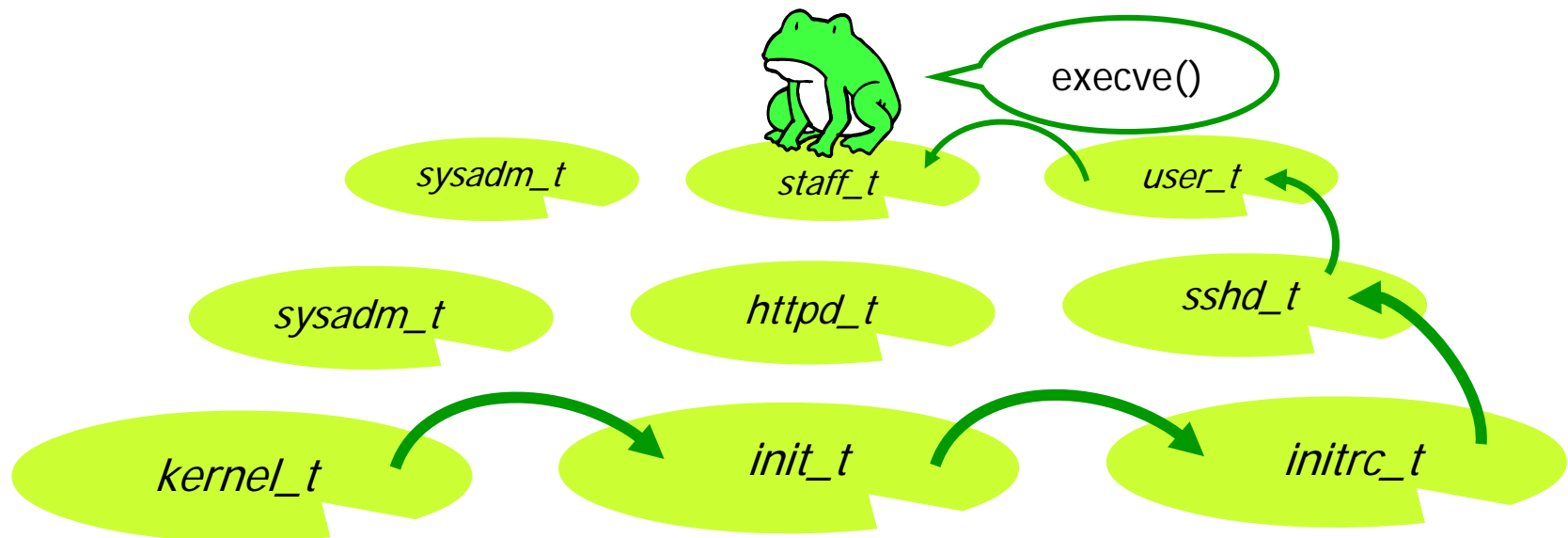
- Handled by TOMOYO Linux kernel automatically .
- Domain simply gets deeper by default.
- The name of a domain is like “<kernel>/sbin/init ...” .
 - No administrator operation necessary.
 - No file context definition necessary.
 - No need of “xattr”.

Domain Transition



Domains in SELinux

- Domain names and domain transition conditions must be defined beforehand.
- Domains are flat and have no levels.

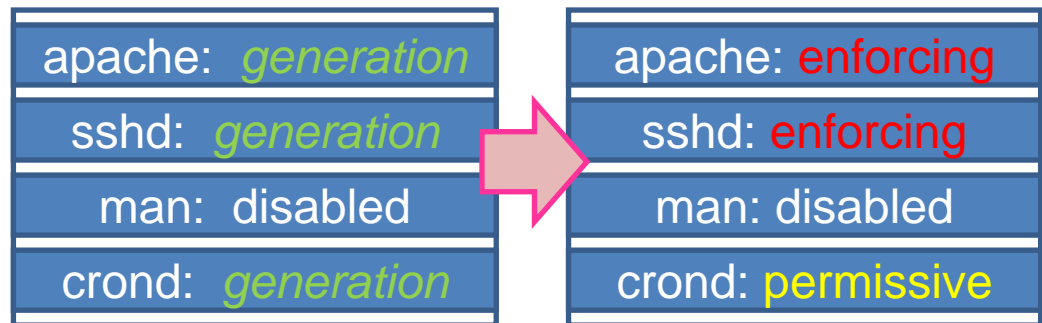


Mode

- SELinux
 - has 3 modes.
 - disabled
 - permissive (it is “ok”, even if policy says “no”)
 - enforcing (if policy says “no”, it is “no”)
 - Mode is system global in SELinux.

disabled/permissive/
enforcing

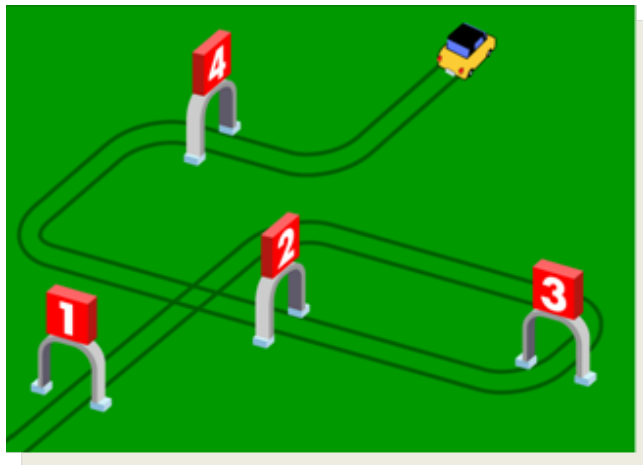
- TOMOYO Linux
 - 4 modes.
 - disabled
 - permissive
 - policy generation
 - enforcing



– Mode is selectable on a per domain basis.

Part 2

TOMOYO Linux for Embedded Systems



MAC for Embedded Systems

- Embedded systems need security, too.
- Resource issues:
 - SELinux needs “xattr” aware filesystems.
 - Tight requirements for memory and CPU.
- BusyBox issue:
 - Sharing a binary image is good for saving storage, but sharing “context” can harm MAC.
 - BusyBox has incorporated SELinux supports recently.
 - TOMOYO Linux has built-in support for BusyBox.

Affinity with Embedded Linux

- Small memory footprint.
 - 100KB for kernel code and data.
 - A few hundred KB for policy.
- Targeted Protection.
 - You can protect selectively.
- Small performance impact.
 - No checks for read()/write() system calls.

Affinity with Embedded Linux

- Program invocation via symbolic link.
 - “alias /bin/bash /bin/sh” will allow you to distinguish /bin/bash and /bin/sh .
- Program invocation via hard link.
 - You can distinguish /bin/gzip and /bin/gunzip without any prescription.

Affinity with Embedded Linux

- No xattr support required.
 - 100% filesystem independent.
- No GUI environment required.
 - Administration tasks are quite simple enough that you can do it using CUI (e.g. console or ssh session).
- No modification required.
 - You don't need any patches for your userland programs.

Affinity with Embedded Linux

- Dynamically created files.
 - Sometimes files are dynamically created in a volatile filesystem. Such files have no fixed inode-number nor determinate inode-xattr.
 - LIDS needs stable inode-number.
 - SELinux needs stable inode-xattr.
 - TOMOYO doesn't need either.

Affinity with Embedded Linux

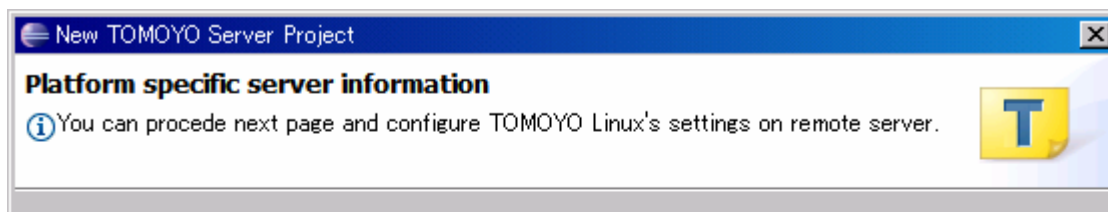
- Automatic policy generation.
 - TOMOYO will observe your system's behavior and build the exact policy for you.
 - You have got a professional tailor!
- Easy policy accommodation.
 - You can assign multiple pathnames to a file.
 - No need for accommodation of label assignment like SELinux.

Affinity with Embedded Linux

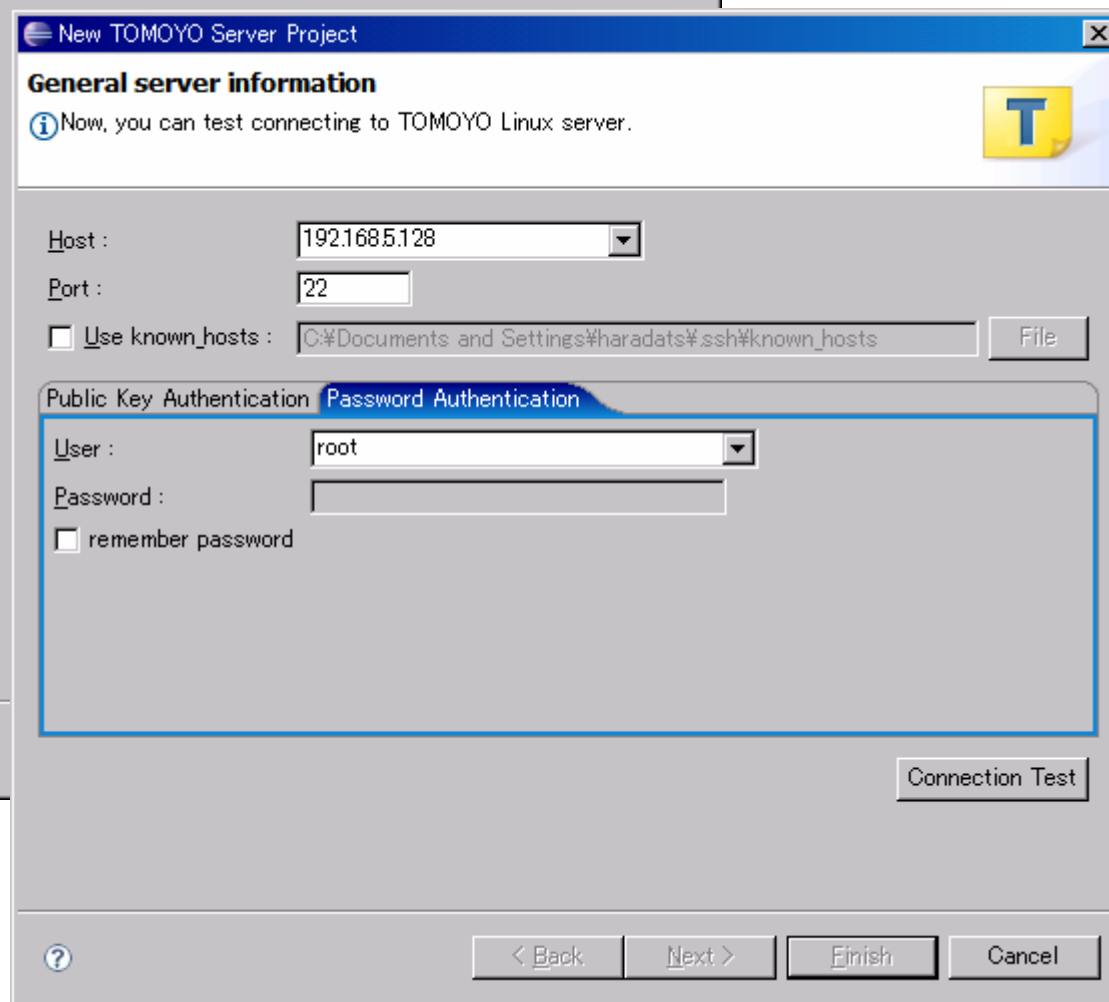
- Unusual pathnames.
 - TOMOYO Linux is a pathname based MAC.
 - TOMOYO Linux can handle any characters including white-spaces, carriage-return, non-printable characters (e.g. BS), non-ASCII characters (e.g. EUC_JP) in the pathname.
 - All names in TOMOYO Linux consist of ASCII printable characters.

GUI

- TOMOYO Linux can be fully controlled via tty and no GUI is necessary, but
- we are developing a Eclipse plug-in for GUI lovers/addicts.
- The plug-in talks to TOMOYO Linux kernel through SSH protocol.
- No server side libraries are required.
- English is supported as well as Japanese.



- Redhat Linux 9
- CentOS 4.4
- Fedora Core 3
- Fedora Core 4
- Fedora Core 5
- Fedora Core 6
- SUSE 10.1
- Debian Sarge
- Debian Etch
- Ubuntu 6.10



TOMOYO Linux (default) - TOMOYO Project Editor - Eclipse SDK

File Edit Navigate Search Project Run Window Help

test x

Home

Server
General Information about remote TOMOYO Linux server

Host : 192.168.5.128
Distribution : CentOS 4.4

TOMOYO Linux
Information on installed TOMOYO Linux is displayed.

Project : test
Version : 1.4

Connection status
The server connection, the reference to the server watch status, and the change can be done here.

Connect : Connected Disconnected

Polling : On Off

Log watch : On Off

Connection settings
Connected method like the authentication method to the server etc. is displayed.

Login user : root

Authentication method : Public key Password

Remember password : Yes No

Hostkey check : Yes No

Server settings
The project setting, the reference to a set value of the server, and the change can be done here.

Grant log path : /var/log/tomoyo/grant_log.txt

Reject log path : /var/log/tomoyo/reject_log.txt

Server operation
The server related to TOMOYO Linux can be operated here.

[Save Policy](#) : The domain, the exception, and the system policy are saved in the file on the server.

[Save Profile](#) : Profile information is saved in the file on the server.

Home Domain Policy Exception Policy System Policy Profile

Grant Log Reject Log History Backup Navigator

Outline

- Profile (11)
 - 0 - Disabled (8)
 - 1 - Learning (1)
 - 2 - Permissive
 - 3 - Enforcing (0)
 - 5 - Accept MOC
 - 6 - Permissive
 - 7 - Enforcing M
 - 9 - (0)
 - 10 - (0)
 - 13 - (0)
 - 14 - (0)
- Domain Policy (948)
- Exception Policy (8)

Properties

Property	Value

TOMOYO Linux (default) - TOMOYO Project Editor - Eclipse SDK

File Edit Navigate Search Project Run Window Help

test x

Domain Policy

Domain transition tree

```

<kernel>
├── /etc/hotplug/firmware.agent (0)
│   ├── /bin/sed (0)
│   ├── /bin/sleep (0)
│   └── /bin/uname (0)
├── /etc/rc.d/init.d/acpid (13)
│   ├── /bin/touch (2)
│   ├── /sbin/consoletype (0)
│   └── /sbin/initlog (3)
│       └── /usr/sbin/acpid (5)
├── /etc/rc.d/init.d/anacron (13)
│   ├── /bin/nice (1)
│   │   └── /sbin/initlog (3)
│   │       └── /usr/sbin/anacron
│   ├── /sbin/consoletype (0)
│   └── /sbin/initlog (1)
└── /etc/rc.d/init.d/apmd (8)
    
```

Access permissions

```

010 0 - Disabled
└── 010 1 (3)
    ├── /sbin/hotplug
    ├── /sbin/init
    └── /sbin/modprobe
    
```

Outline

- Profile (11)
 - 0 - Disabled (8)
 - 1 - Learning (1)
 - 2 - Permissive
 - 3 - Enforcing (0)
 - 5 - Accept MOC
 - 6 - Permissive
 - 7 - Enforcing M
 - 9 - (0)
 - 10 - (0)
 - 13 - (0)
 - 14 - (0)
- Domain Policy (948)
- Exception Policy (8)

Properties

Property	Value
ACL	co 3
Descen	840
Domain	<kernel>
Domain 1	
Has ini	Yes
Is initial	No
Is keep	No
Process	<kernel>

Home Domain Policy Exception Policy System Policy Profile

Grant Log Reject Log History Backup Navigator

TOMOYO Linux (default) - TOMOYO Project Editor - Eclipse SDK

File Edit Navigate Search Project Run Window Help

test x

Profile

Profile table

	0 - Disabled	1 - Learning	2 - Permissive	3 - Enforcing	5 - Accept
[-] File					
... MAC_FOR_FILE	Disabled	Learning	Permissive	Disabled	Disable
[-] Network					
... MAC_FOR_NETWORK	Disabled	Disabled	Disabled	Disabled	Disable
... MAC_FOR_BINDPORT					
... MAC_FOR_CONNECTPORT					
... RESTRICT_AUTOBIND	Off	Off	Off	Off	Off
[-] Filesystem					
... DENY_CONCEAL_MOUNT	Disabled	Disabled	Disabled	Disabled	Disable
... RESTRICT_CHROOT	Disabled	Disabled	Disabled	Disabled	Disable
... RESTRICT_MOUNT	Disabled	Disabled	Disabled	Disabled	Disable
... RESTRICT_UNMOUNT	Disabled	Disabled	Disabled	Disabled	Disable
... DENY_PIVOT_ROOT					
... RESTRICT_PIVOT_ROOT	Disabled	Disabled	Disabled	Disabled	Disable
... TRACE_READONLY					
[-] Log					
... MAX_ACCEPT_FILES					
... MAX_ACCEPT_ENTRY	2048	2048	2048	2048	2048
... MAX_GRANT_LOG	1024	1024	1024	1024	1024
... MAX_REJECT_LOG	1024	1024	1024	1024	1024
... TOMOYO_VERBOSE	Off	Off	Off	Off	Off
[+] Capability					
[-] Others					
... COMMENT	Disabled	Learning	Permissive	Enforcing	Accept Mod
... MAC_FOR_ARGV0	Disabled	Disabled	Disabled	Disabled	Disable
... MAC_FOR_SIGNAL	Disabled	Disabled	Disabled	Disabled	Disable
... ALLOW_ENFORCE_GRACE	Off	Off	Off	Off	Off
[+] Customize					

Home Domain Policy Exception Policy System Policy Profile

Grant Log Reject Log History Backup Navigator

Usages Other Than Security

- TOMOYO is not only for security!
 - You can analyze your Linux very deeply.
 - TOMOYO Linux can find out unused files and unexpected behavior for you (you can save space by deleting them).
 - “rm -rf *” will cause a disaster on normal Linux, but it will not if you are running TOMOYO Linux and keeping proper policies.
 - Just looking at the policies is so much fun. it also helps to understand OS mechanisms.

Plans



- 2007 Ottawa Linux Symposium BoF.
 - Introduce TOMOYO Linux to label-guys.
 - Is “name based access control” a fatal illness?
- We would like to merge our work to be included in main line (if we survive Ottawa).
 - Propose patches to LSM (to make pathname based MAC work).
 - Propose capabilities extension to LSM.

Please Come to See TOMOYO Linux

April 18, Room B

18-340 15:40-16:30 Presentation

18-440 16:40-17:30 Tutorial

Don't miss it!



CELF Embedded Conference 2007

TOMOYO Linux

TOMOYO Linux version 1.4

TOMOYO Linux (<http://tomoyo.sourceforge.jp/index.html.en>) is yet another way to provide a lightweight and manageable MAC ability. It is available under GPL and applicable to and suitable for both PCs and embedded devices. In this session, we will present an overview of TOMOYO Linux and explain why TOMOYO Linux is suitable for embedded devices. We will also show some demonstrations.



This project will save our earth from crackers.
Why don't you join us?

 **remove before reboot!!**