

LinuxCon North America 2012 (San Diego)

CaitSith

a new type of rule based in-kernel access control

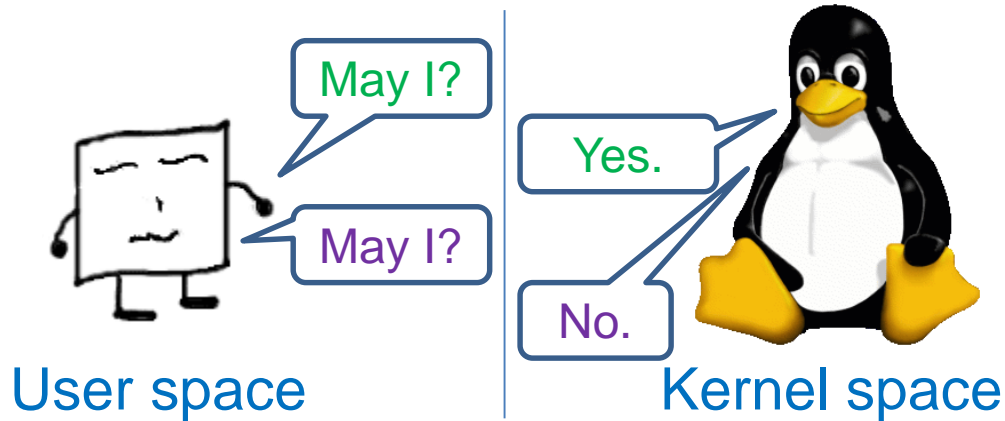
Tetsuo Handa, NTT

Who am I?

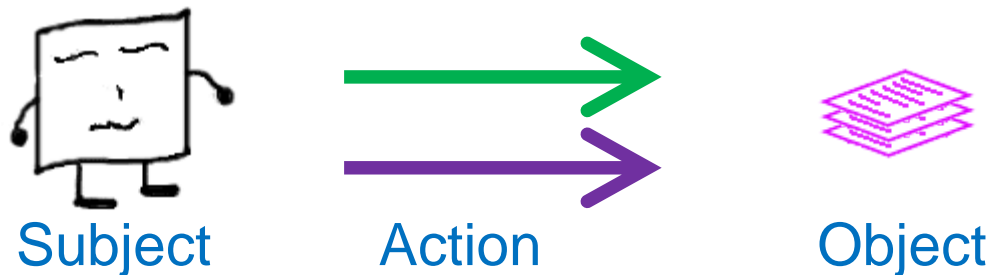
- ▶ A retired employed programmer
- ▶ My involvement with Linux
 - ▶ 2001.10-2003.3 Developing user space applications that run on Linux systems.
 - ▶ 2003.4-2012.3 Developing kernel mechanisms for improving security of Linux systems.
 - ▶ 2012.4- Providing user support service for troubleshooting Linux systems.

What do I speak today?

- ▶ Security, especially access control in kernel space.



- ▶ Subject(processes), Action, Object(resources)
 - ▶ That's all. However, it is extremely difficult to develop rules that match user's needs.



Structure of this presentation?

- ▶ Chapter 1 --- Introduction: Summarized description of what has happened before CaitSith
 - ▶ For users who are interested in in-kernel access control.
- ▶ Chapter 2 --- Things I experienced with continuing enhancement of access control functionality
 - ▶ For developers who are developing access control modules.
- ▶ Chapter 3 --- Things I experienced with continuing enhancement of ease of use
 - ▶ For users who are seeking for simpler in-kernel access control modules.
- ▶ Chapter 4 --- CaitSith
 - ▶ For users who are interested in my proposal.

Chapter 1

Introduction: Summarized description of what has happened before CaitSith

Everyone's security varies?

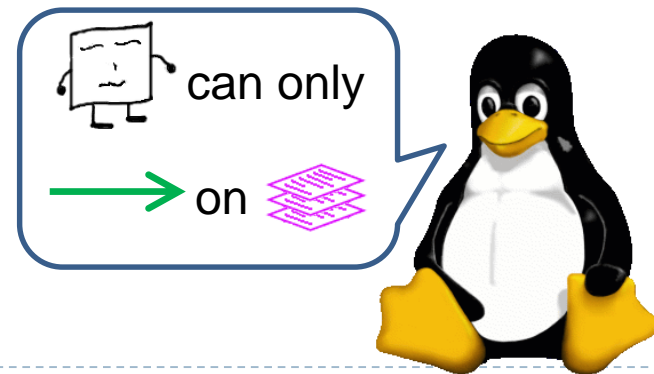
- ▶ What people associate with the word "security" depends on their skill levels and beliefs.
 - ▶ Give people choices, rather than forcing the only one.
- ▶ My belief is that visualization is important.
 - ▶ Many of today's security issues come from invisibleness.
 - ▶ Traceability leads to satisfaction.

Attempts for doing access control in the kernel space

- ▶ Threats are in the behavior of the user space.
- ▶ But, doing access control in the user space is problematic.
 - ▶ It can be bypassed when deprived of control.
 - ▶ Its level and granularity varies.
- ▶ Let's do access control in the kernel space, in addition to access control in the user space.
 - ▶ Although what in-kernel access control can do is limited, in-kernel access control can provide a baseline restriction and will be useful.

Mandatory Access Control(MAC)

- ▶ Implementation that does access control in the kernel space.
 - ▶ It cannot be bypassed because it is done in the kernel space.
- ▶ Currently, SELinux, SMACK, TOMOYO and AppArmor are in the Linux mainline kernel.
 - ▶ They are all ****allowlisting**** approach which uses Linux Security Modules (LSM) interface.
 - ▶ They do access control from the point of view of ****subjects****.



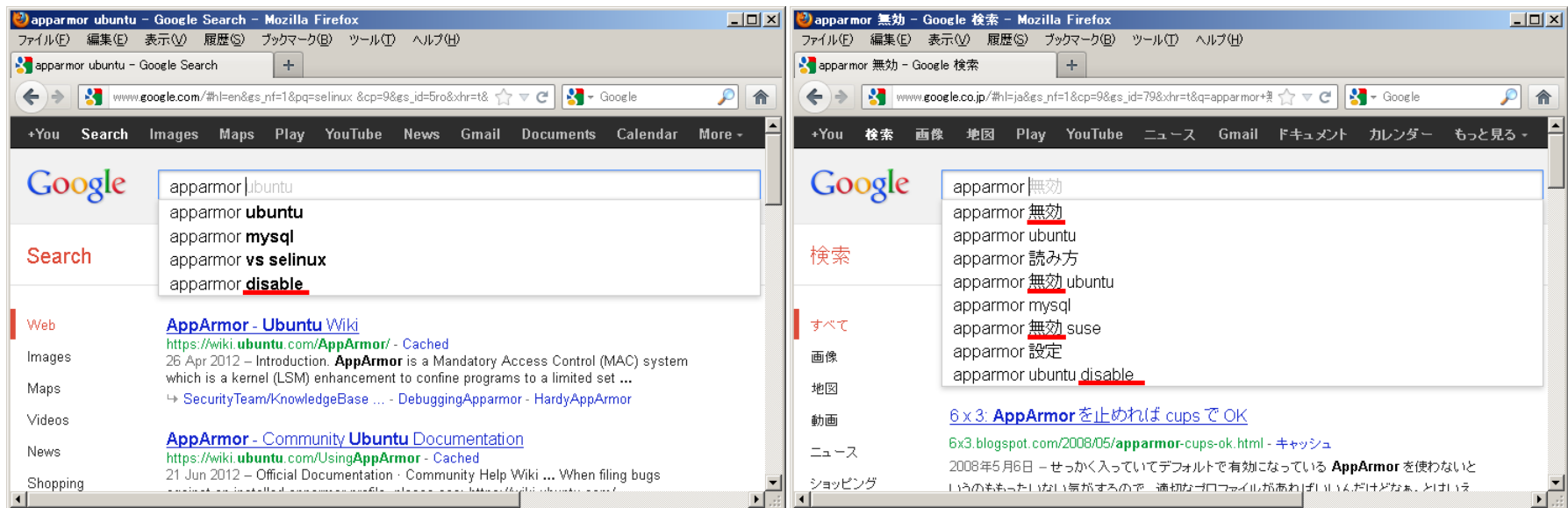
Distributor's kernels enable some of them, but...

- ▶ Many people are still disabling SELinux.



Distributor's kernels enable some of them, but...

- ▶ Even AppArmor which was claimed to be easier than SELinux is disabled.



Distributor's kernels enable some of them, but...

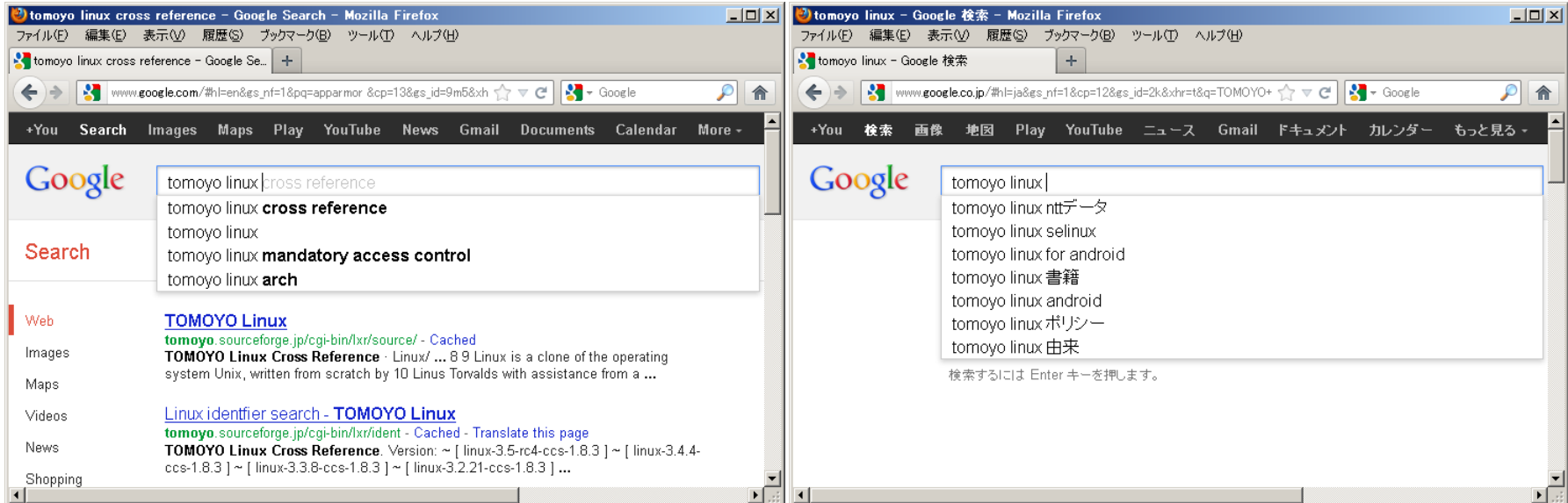
▶ What about SMACK?



- ▶ Not disabled yet, for SMACK is not enabled without user's explicit configuration.

Distributor's kernels enable some of them, but...

▶ What about TOMOYO?



- ▶ Not disabled yet, for TOMOYO is not enabled without user's explicit configuration.

Distributor's kernels enable some of them, but...

- ▶ **Reasons to disable them?**
 - ▶ Fears to use without understanding their configuration.
 - ▶ Fears to miss permissions in the allowlisting approach.
- ▶ **There are a lot of documentation.**
 - ▶ Too long, didn't read.
 - ▶ They are for developers, not for users.

"all or nothing" problem

- ▶ Ideally, access control is enforced on all subjects and all objects.
- ▶ In reality, it is considered as "Well done" if access control is enforced on some specific subjects.
- ▶ When troubles occur, they will have no choice but to **disable entirely** unless they know how to configure policy and current policy configurations.

Reasons why TOMOYO insisted on manageability for users?

- ▶ **No resource to distribute ready-made policy.**
 - ▶ TOMOYO has no background distributors compared to SELinux(RedHat) and AppArmor(SUSE/Ubuntu).
- ▶ **Unable to troubleshoot if ready-made policy is used.**
 - ▶ TOMOYO would follow the same way where SELinux and AppArmor stray into.
- ▶ **What people want to allow/deny varies.**
 - ▶ It is impossible to develop ready-made policy that can cover everybody's needs.
- ▶ **Have to disable upon troubles if there is only one switch.**
 - ▶ "all or nothing" problem

Problems with developing policy configuration

- ▶ An "access control" restricts access requests based on rules defined beforehand.
 - ▶ It is an implicit requirement that users can define rules beforehand.
 - ▶ However, to define rules beforehand, users have to be familiar with internal structure of Linux systems.
 - ▶ Not many users are familiar with Linux internal because it is a world where they usually don't care.
- ▶ However, paying attention for burden of defining rules has generally been viewed as unimportant.
 - ▶ TOMOYO had been paying attention for this burden.

TOMOYO had been struggled in order to keep TOMOYO enabled.

- ▶ Made possible to enable/disable on a per-action basis using profiles.
 - ▶ This allows users to choose the action coverage based on their skills.
- ▶ Made possible to enable/disable on a per-domain basis using profiles.
 - ▶ This allows users to choose the subjects coverage based on their skills.

TOMOYO had been struggled in order to keep TOMOYO enabled.

- ▶ Made possible to handle policy violations interactively.
 - ▶ This allows users to judge unexpected access requests on a case-by-case basis.
- ▶ Made possible to understand all states using tree style domain transitions.
 - ▶ This allows users to understand what's going on.

Meanwhile, I received unexpected requests from RHEL users...

- ▶ "I want to apply access control on specific resources (files), rather than applying on specific processes."
- ▶ TOMOYO was allowing users to enable/disable access control on a per-action basis and a per-domain basis, but was not allowing users to enable/disable access control on a per-file basis.

After all, did existing MAC implementations respond to user's needs?

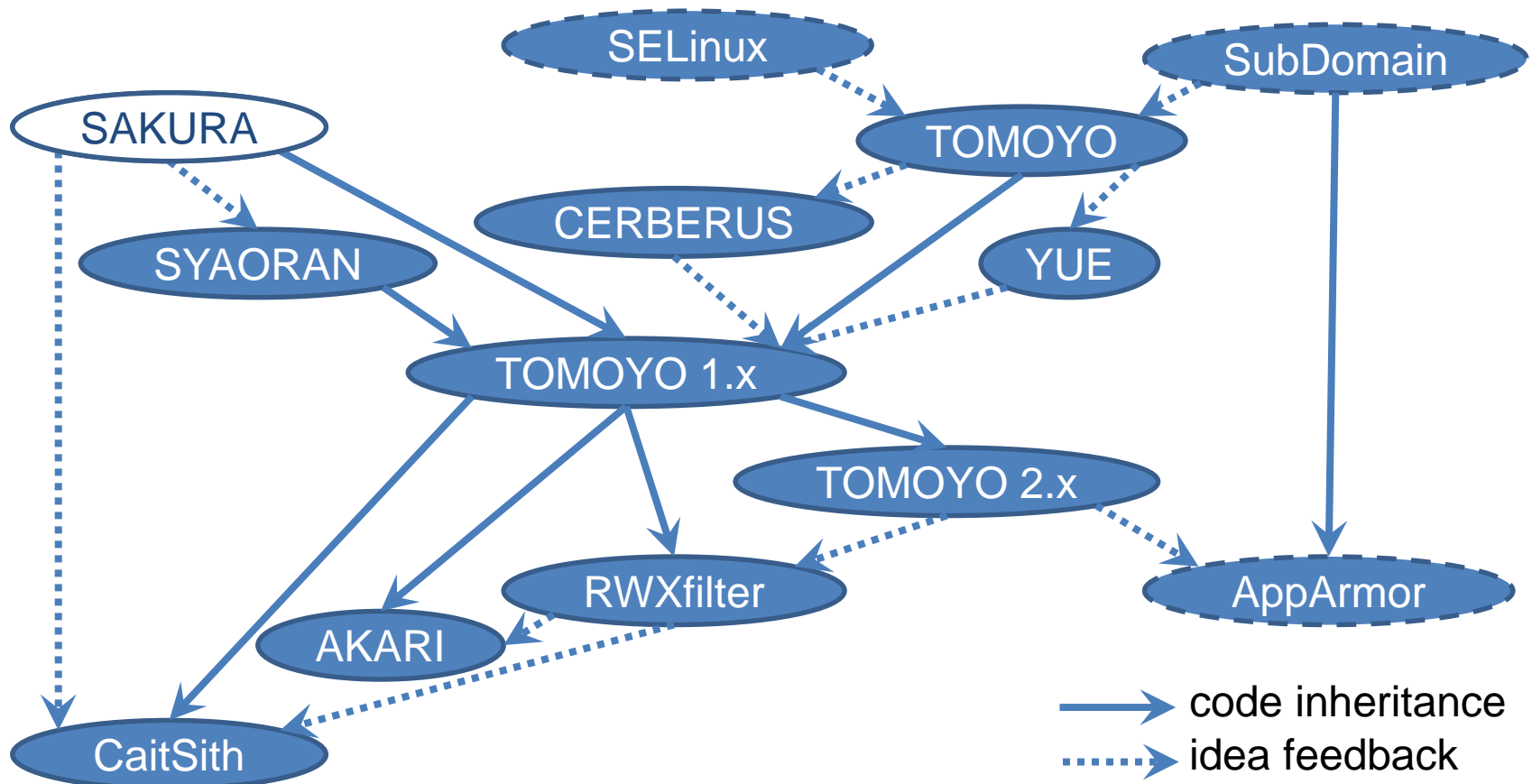
- ▶ User's needs are not always allowlisting nor from the point of view of processes.
 - ▶ Some wants to apply from the point of view of resources.
 - ▶ Some are not interested in managing domains.
 - ▶ These are limitations for TOMOYO.
- ▶ We might want to try a fundamental course-changing move.
 - ▶ That's the trigger for developing CaitSith.

Chapter 2

Things I experienced with continuing
enhancement of access control
functionality

SAKURA (2003.4-)

- ▶ An attempt for protecting Linux systems without policy management.



SAKURA

- ▶ I tried SELinux, but I soon gave up because SELinux was too difficult to use.
 - ▶ Can't we omit policy management by specializing for protection from tampering?
- ▶ Code name: "Security Advancement Know-how Upon Read-only Approach for Linux"
- ▶ Topics on SAKURA
 - ▶ Protection from tampering via read-only mounting
 - ▶ System-wide access restriction
 - ▶ Spontaneous permission abandonment via modification of user space programs

Protection from tampering via read-only mounting

- ▶ Mounting filesystems as read-only wherever possible in order to reduce the risk of tampering files.
 - ▶ I modified the kernel to report pathnames which failed with -EROFS error in order to help separating read-only directories and writable directories.
 - ▶ Mount all partitions except partitions which need to be writable (e.g. /var and /tmp) read-only, and store read-only partitions into read-only medium in order to protect from direct tampering attacks (e.g. writing to block device files which corresponds to read-only mounted partitions).

System-wide access restriction

- ▶ Mounting a writable filesystem over a read-only filesystem ruins tamper-proof protection.
 - ▶ Restrict namespace related actions (e.g. mount, chroot, pivot_root) for system-wide.
 - ▶ An example configuration looks like below.

```
allow_mount devpts /dev/pts/ devpts 0x0
allow_mount any / --remount 0x0
allow_mount securityfs /sys/kernel/security/ securityfs 0x0
allow_mount none /proc/sys/fs/binfmt_misc/ binfmt_misc
0x0
allow_chroot /etc/avahi/
allow_chroot /var/empty/sshd/
```

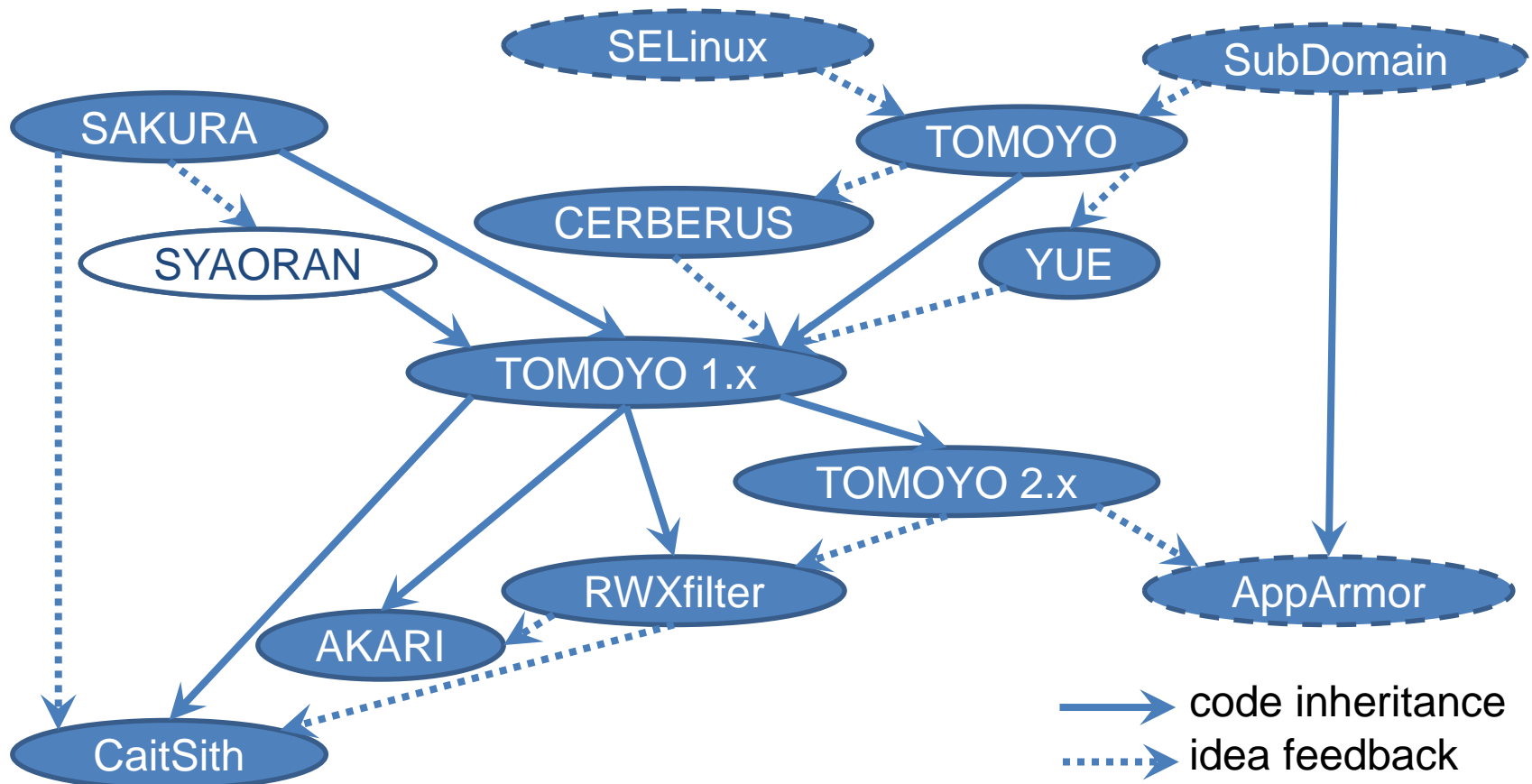
Note that the name of processes are not specified.

Spontaneous permission abandonment via modification of user space programs

- ▶ To make system-wide access restriction more efficient, I added spontaneous permission abandonment by appending a original field to `task_struct` of Linux 2.4 kernels.
 - ▶ Allow user space programs to discard permissions to call `execve()`, `chroot()`, `pivot_root()`, `mount()` and a permission to regain effective UID = 0 on a per-`task_struct` basis.
 - ▶ Temporal discard which the permissions will be regained after successful `execve()` request.
 - ▶ Permanent discard which the permissions will not be regained after `execve()` request.
- ▶ We could not afford resource to modify user space programs and this feature was removed in TOMOYO 1.4.
 - => But a more wider spontaneous permission abandonment feature was added to Linux 3.5 as "seccomp mode 2".

SYAORAN (2004.10-)

- ▶ Tamper-proof filesystem for /dev partition.



SYAORAN

- ▶ SAKURA made it possible to mount / partition as read-only, but /dev cannot be mounted as read-only.
 - ▶ Existing /dev filesystems (e.g. devfs and devtmpfs) allowed modification of directory entries via requests from user space.
 - ▶ Tampering files in /dev partition is a severe problem.
 - ▶ What happens if /dev/null has attributes of /dev/zero ?
- ▶ Code name: "Simple Yet All-important Object Realizing Abiding Nexus"
- ▶ Topics on SYAORAN
 - ▶ Tamper-proof filesystem for /dev partition

Tamper-proof filesystem for /dev partition

- ▶ I developed a dedicated filesystem for /dev by adding attribute checking logic to tmpfs filesystem.
 - ▶ By using this filesystem, we can enforce combination of filenames and their attributes.
 - ▶ For example, /dev/null always has char-1-3 and /dev/zero always has char-1-5.

Tamper-proof filesystem for /dev partition

- ▶ Configuration file looks like below.

#filename	perm	owner	group	flags	type	major	minor
pts	755	0	0	0	d		
shm	755	0	0	0	d		
null	666	0	0	0	c	1	3
zero	666	0	0	0	c	1	5
random	644	0	0	0	c	1	8
urandom	644	0	0	0	c	1	9
tty	666	0	0	0	c	5	0
tty0	600	0	0	12	c	4	0
tty1	600	0	0	12	c	4	1

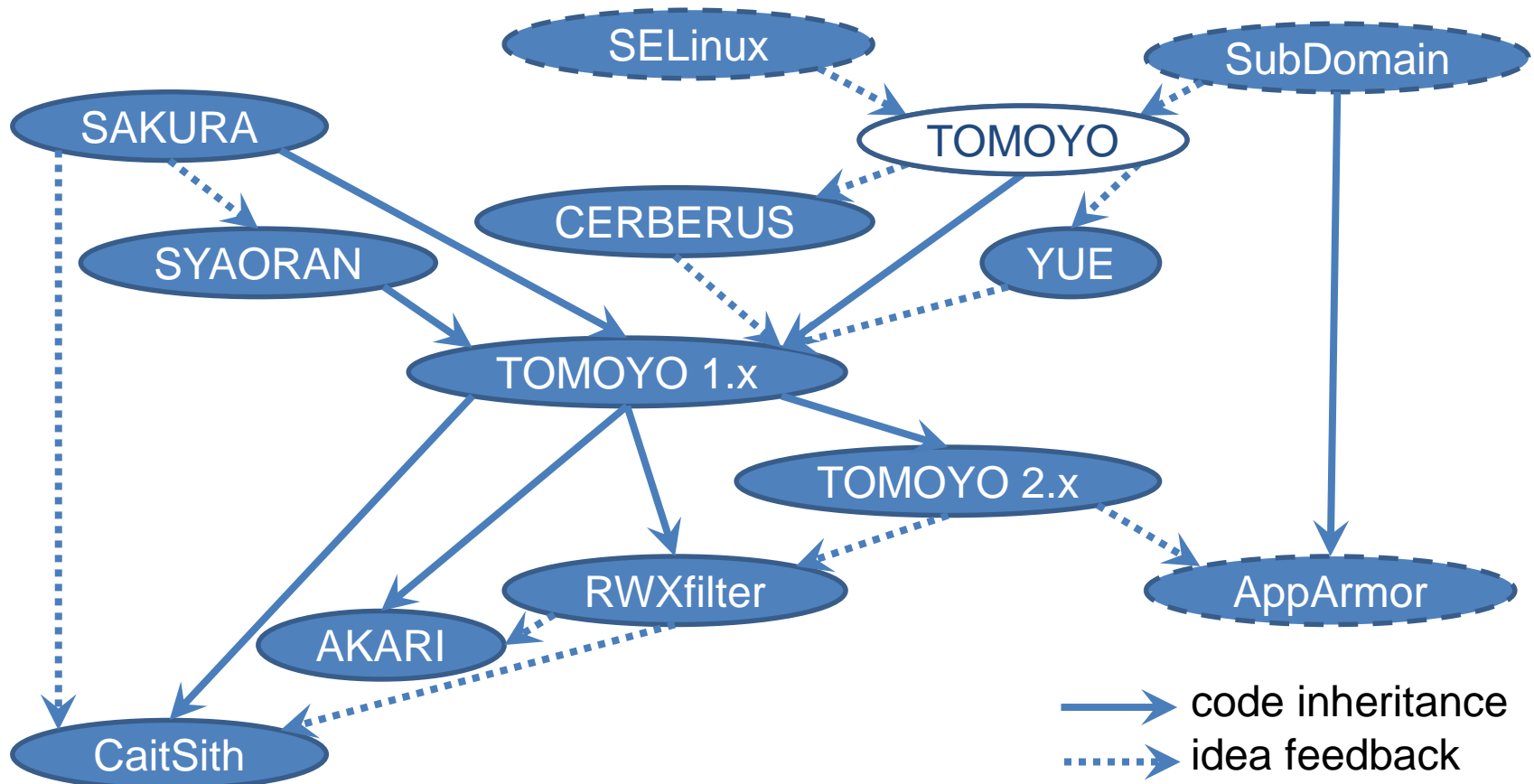
Note that the permitted actions
(create/delete/chmod/chown/chgrp) are restricted

Managing policy is inevitable?

- ▶ By using SYAORAN for /dev and using SAKURA for wherever possible, the risk of tampering files can be reduced.
 - ▶ Although the combination of SAKURA and SYAORAN made it impossible to tamper files stored into read-only medium, storing into read-only medium also makes it impossible to handle software updates.
 - ▶ To make it possible to handle software updates while protecting from tampering, we should consider also using policy based protection.
- => Leads to TOMOYO.

TOMOYO (2003.7-)

- ▶ An attempt for implementing manageable policy.



TOMOYO

- ▶ SELinux's policy is too difficult to use. Can't we develop original policy that covers only what we need?
 - ▶ Let's generate policy that allows only behavior of processes which we ever observed.
- ▶ Code name: "Task Oriented Management Obviates Your Onus on Linux"
- ▶ Topics on TOMOYO
 - ▶ System-wide domain transition tracing function
 - ▶ Access request tracing function on a per-domain basis
 - ▶ Access request restricting function on a per-domain basis

System-wide domain transition tracing function

- ▶ Append a original field to task_struct of Linux 2.4 kernels.
- ▶ Form a tree style state transition using the fork()/execve() mechanism.
- ▶ Use each state in the tree as a domain.

```
192.168.0.6 - Terra Term VI
root@ccsecurity ~]# ccs-pstree
1 init (1) <kernel> /sbin/init
1 +- udevd (2885) <kernel> /sbin/init /etc/rc.d/rc.sysinit /sbin/start_udev /sbin/udev
1 +- udevd (3548) <kernel> /sbin/init /etc/rc.d/rc.sysinit /sbin/start_udev /sbin/udev
1 +- dhclient (3355) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S10network /etc/sysconfig/network-scripts/ifup /etc/sysconfig/network-scripts/ifup-eth /sbin/dhclient
1 +- auditd (3401) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S11auditd /bin/bash /usr/sbin/auditd
1 +- rsyslogd (3417) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S12rsyslog /bin/bash /sbin/rsyslogd
1 +- dbus-daemon (3429) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S22messasaebus /bin/bash /bin/dbus-daemon
1 +- avahi-daemon (3440) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S24avahi-daemon /bin/bash /usr/sbin/avahi-daemon
1 +- avahi-daemon (3441) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S24avahi-daemon /bin/bash /usr/sbin/avahi-daemon
1 +- sshd (3471) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S55sshd /usr/sbin/sshd
1 +- sshd (3554) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S55sshd /usr/sbin/sshd /usr/sbin/sshd /bin/bash
1 +- bash (3556) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S55sshd /usr/sbin/sshd /usr/sbin/sshd /bin/bash
1 +- ccs-pstree (3587) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S55sshd /usr/sbin/sshd /usr/sbin/sshd /bin/bash /usr/sbin/ccs-pstree
1 +- xinetd (3479) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S56xinetd /bin/bash /usr/sbin/xinetd
1 +- sendmail (3495) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S30sendmail /bin/bash /usr/sbin/sendmail
1 +- sendmail (3503) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S30sendmail /bin/bash /usr/sbin/sendmail
1 +- crond (3512) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S30crond /bin/bash /usr/sbin/crond
1 +- smbd (3521) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S31smb /bin/bash /usr/sbin/smbd
1 +- smbd (3532) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S31smb /bin/bash /usr/sbin/smbd
1 +- ccs-auditd (3531) <kernel> /sbin/init /bin/sh /etc/rc.d/rc /etc/rc.d/rc3.d/S30local /usr/sbin/ccs-auditd
1 +- mingetty (3541) <kernel> /sbin/init /bin/sh /sbin/mingetty
1 +- mingetty (3543) <kernel> /sbin/init /bin/sh /sbin/mingetty
1 +- mingetty (3545) <kernel> /sbin/init /bin/sh /sbin/mingetty
1 +- mingetty (3549) <kernel> /sbin/init /bin/sh /sbin/mingetty
1 +- mingetty (3551) <kernel> /sbin/init /bin/sh /sbin/mingetty
1 +- mingetty (3553) <kernel> /sbin/init /bin/sh /sbin/mingetty
root@ccsecurity ~]#
```

```
192.168.0.6 - Terra Term VI
Domain Transition Editor >>> 221 domains '?' for help
kernel
0: <kernel>
1: /sbin/init
2: /bin/kill
3: /bin/sh
4: /bin/awk
5: /bin/cat
6: /bin/grep
7: /bin/pymouth
8: /etc/rc.d/rc
9: /bin/pymouth
10: /etc/rc.d/rc3.d/S10network
11: /bin/grep
12: /bin/tarcp
13: /bin/ls
14: /bin/sed
15: /bin/sort
16: /bin/touch
17: /etc/sysconfig/network-scripts/ifup
18: /bin/awk
19: /bin/sed
20: /etc/sysconfig/network-scripts/ifup-eth
21: /bin/awk
22: /bin/cat
23: /bin/grep
24: /bin/ipcalc
25: /bin/sed
26: /etc/sysconfig/network-scripts/ifup-ipv6
27: /bin/awk
28: /bin/sed
29: /etc/sysconfig/network-scripts/ifup-post
30: /bin/awk
31: /bin/hostname
32: /bin/sed
33: /etc/sysconfig/network-scripts/ifup-aliases
34: /bin/awk
35: /sbin/ip
36: /etc/sysconfig/network-scripts/ifup-routes
37: /sbin/dhclient
38: /sbin/dhclient-script
39: /bin/awk
40: /bin/cat
41: /bin/cut
42: /bin/grep
43: /bin/ipcalc
44: /bin/mktemp
45: /bin/m
46: /bin/sed
47: /sbin/arping
48: /sbin/consoletype
49: /sbin/ip
50: /sbin/restorecon
51: /usr/bin/logger
52: /sbin/ethtool
53: /sbin/ip
54: /etc/sysconfig/network-scripts/init.ipv6-global
55: /sbin/ip
56: /sbin/evsctl
57: /sbin/arp
58: /sbin/syctl
59: /etc/rc.d/rc3.d/S11auditd
60: /bin/bash
61: /sbin/auditd
62: /sbin/auditd
63: /bin/touch
64: /sbin/auditctl
65: /etc/rc.d/rc3.d/S12rsyslog
66: /bin/bash
67: /sbin/rsyslogd
68: /bin/touch
69: /etc/rc.d/rc3.d/S22messasaebus
70: /bin/bash
71: /bin/dbus-daemon
72: /bin/dbus-uidgen
73: /bin/touch
74: /etc/rc.d/rc3.d/S24avahi-daemon
75: /bin/bash
76: /usr/sbin/avahi-daemon
77: /bin/cp
78: /bin/touch
79: /etc/rc.d/rc3.d/S25netfs
80: /bin/awk
81: /bin/mount
82: /bin/touch
83: /etc/rc.d/rc3.d/S25udev-post
84: /sbin/udevadm
85: /etc/rc.d/rc3.d/S25sshd
86: /bin/touch
87: /sbin/rmnetel
88: /usr/sbin/sshd
89: /usr/sbin/sshd
90: /bin/awk
91: /bin/grep
92: /bin/postname
93: /sbin/consoletype
94: /usr/bin/dircolors
95: /bin/id
96: /usr/bin/pass
```

Access request tracing function on a per-domain basis

- ▶ Started as an access analysis tool.
 - ▶ Trace open() and execve() requests using pathnames and sort the output by domain as a key.



```
<<< Domain Policy Editor >>>      22 entries      '?' for help
<kernel>
0: file chmod      /var/log/boot.log 0644
1: file create     /var/log/boot.log 0644
2: file execute    /sbin/init exec.realpath="/sbin/init" exec.argv[0]="/sbin/init"
3: file execute    /sbin/modprobe exec.realpath="/sbin/modprobe" exec.argv[0]="/sbin/modprobe"
4: file getattr    /var/lib/plymouth/boot-duration
5: file ioctl      /dev/console 0x541D
6: file ioctl      /dev/tty1 0x4B71
7: file ioctl      /dev/tty1 0x5401
8: file ioctl      /dev/tty1 0x5404
9: file ioctl      /dev/tty1 0x5457
10: file ioctl     /dev/tty1 0x5602
11: file ioctl     /dev/ttyS0 0x5401
12: file ioctl     /dev/ttyS0 0x5404
13: file ioctl     /dev/ttyS0 0x5457
14: file ioctl     socket:[family=1:type=1:protocol=0] 0x541B
15: file read      /dev/console
16: file truncate  /var/lib/plymouth/boot-duration
17: file unlink    /var/log/boot.log
18: file write     /dev/console
19: file write     /var/lib/plymouth/boot-duration
20: file write     /var/log/boot.log
```

Caveat:
These screenshots include other requests because these are taken using TOMOYO 1.8 on CentOS 6.3.

Access request tracing function on a per-domain basis

```
192.168.0.6 - Tera Term VT
<<< Domain Policy Editor >>> 70 entries '?' for help
kernel> /sbin/init
0: file execute /bin/kill exec.realpath="/bin/kill" exec.argv[0]="/bin/kill"
1: file execute /bin/sh exec.realpath="/bin/bash" exec.argv[0]="/bin/sh"
2: file execute /etc/rc.d/rc.sysinit exec.realpath="/etc/rc.d/rc.sysinit" exec.argv[0]="/etc/rc.d/rc.sysinit"
3: file setattr /etc/group
4: file setattr /etc/init/
5: file setattr /etc/init/control-alt-delete.conf
6: file setattr /etc/init/init-system-dbus.conf
7: file setattr /etc/init/kexec-disable.conf
8: file setattr /etc/init/plymouth-shutdown.conf
9: file setattr /etc/init/prefdm.conf
10: file setattr /etc/init/quit-plymouth.conf
11: file setattr /etc/init/rc.conf
12: file setattr /etc/init/rc3-emergency.conf
13: file setattr /etc/init/rc3-sulogin.conf
14: file setattr /etc/init/rc3.conf
15: file setattr /etc/init/serial.conf
16: file setattr /etc/init/splash-manager.conf
17: file setattr /etc/init/start-ttys.conf
18: file setattr /etc/init/tty.conf
19: file setattr /etc/ld.so.cache
20: file setattr /etc/nsswitch.conf
21: file setattr /etc/passwd
22: file setattr /lib/libc-2.12.so
23: file setattr /lib/libdbus-1.so.3.4.0
24: file setattr /lib/libgcc_s-4.4.6-20120305.so.1
25: file setattr /lib/libnih-dbus.so.1.0.0
26: file setattr /lib/libnih.so.1.0.0
27: file setattr /lib/libnss_files-2.12.so
28: file setattr /lib/libpthread-2.12.so
29: file setattr /lib/librt-2.12.so
30: file setattr proc/self/td/
31: file ioctl /dev/console 0x5401
32: file ioctl /dev/console 0x5402
33: file ioctl /dev/console 0x540B
34: file ioctl /dev/null 0x4B4E
35: file read /dev/console
36: file read /dev/null
37: file read /dev/urandom
38: file read /etc/group
39: file read /etc/init/
40: file read /etc/init/control-alt-delete.conf
41: file read /etc/init/init-system-dbus.conf
42: file read /etc/init/kexec-disable.conf
43: file read /etc/init/plymouth-shutdown.conf
44: file read /etc/init/prefdm.conf
45: file read /etc/init/quit-plymouth.conf
46: file read /etc/init/rc.conf
47: file read /etc/init/rc3-emergency.conf
48: file read /etc/init/rc3-sulogin.conf
49: file read /etc/init/rc3.conf
50: file read /etc/init/serial.conf
51: file read /etc/init/splash-manager.conf
52: file read /etc/init/start-ttys.conf
53: file read /etc/init/tty.conf
54: file read /etc/ld.so.cache
55: file read /etc/nsswitch.conf
56: file read /etc/passwd
57: file read /lib/ld-2.12.so
58: file read /lib/libc-2.12.so
59: file read /lib/libdbus-1.so.3.4.0
60: file read /lib/libgcc_s-4.4.6-20120305.so.1
61: file read /lib/libnih-dbus.so.1.0.0
62: file read /lib/libnih.so.1.0.0
63: file read /lib/libnss_files-2.12.so
64: file read /lib/libpthread-2.12.so
65: file read /lib/librt-2.12.so
66: file read /var/run/utmp
67: file write /dev/console
68: file write /dev/null

192.168.0.6 - Tera Term VT
<<< Domain Policy Editor >>> 58 entries '?' for help
kernel> /sbin/init/bin/sh
0: file execute /bin/awk exec.realpath="/bin/gawk" exec.argv[0]="/bin/awk"
1: file execute /bin/cat exec.realpath="/bin/cat" exec.argv[0]="/bin/cat"
2: file execute /bin/grep exec.realpath="/bin/grep" exec.argv[0]="/bin/grep"
3: file execute /bin/plymouth exec.realpath="/bin/plymouth" exec.argv[0]="/bin/plymouth"
4: file execute /etc/rc.d/rc exec.realpath="/etc/rc.d/rc" exec.argv[0]="/etc/rc.d/rc"
5: file execute /sbin/initctl exec.realpath="/sbin/initctl" exec.argv[0]="/sbin/initctl"
6: file execute /sbin/mingetty exec.realpath="/sbin/mingetty" exec.argv[0]="/sbin/mingetty"
7: file execute /sbin/telinit exec.realpath="/sbin/telinit" exec.argv[0]="/sbin/telinit"
8: file setattr /bin/cat
9: file setattr /bin/grep
10: file setattr /etc/inittab
11: file setattr /etc/ld.so.cache
12: file setattr /etc/nsswitch.conf
13: file setattr /etc/passwd
14: file setattr /etc/sysconfig/init
15: file setattr /lib/libc-2.12.so
16: file setattr /lib/libdl-2.12.so
17: file setattr /lib/libnss_files-2.12.so
18: file setattr /lib/libnss_files-2.12.so
19: file setattr /lib/libtinfo.so.5.7
20: file setattr /sbin/initctl
21: file setattr /sbin/telinit
22: file setattr pipe:[2707]
23: file setattr pipe:[4815]
24: file setattr pipe:[4822]
25: file setattr pipe:[5304]
26: file setattr pipe:[5312]
27: file setattr proc:/meminfo
28: file setattr sysfs:/kernel/kexec_crash_size
29: file ioctl /dev/console 0x5401
30: file ioctl /dev/null 0x5401
31: file ioctl pipe:[2707] 0x5401
32: file ioctl pipe:[4815] 0x5401
33: file ioctl pipe:[5304] 0x5401
34: file ioctl pipe:[5312] 0x5401
35: file read /dev/
36: file read /dev/console
37: file read /dev/tty
38: file read /etc/ld.so.cache
39: file read /etc/nsswitch.conf
40: file read /etc/passwd
41: file read /etc/sysconfig/init
42: file read /lib/ld-2.12.so
43: file read /lib/libc-2.12.so
44: file read /lib/libdl-2.12.so
45: file read /lib/libnss_files-2.12.so
46: file read /lib/libtinfo.so.5.7
47: file read pipe:[2707]
48: file read pipe:[4815]
49: file read pipe:[5304]
50: file read pipe:[5312]
51: file read proc:/meminfo
52: file truncate sysfs:/kernel/kexec_crash_size
53: file write /dev/console
54: file write /dev/null
55: file write /dev/tty
56: file write sysfs:/kernel/kexec_crash_size
```

Access request tracing function on a per-domain basis

```
192.168.0.6 - Tera Term VT
<<< Domain Policy Editor >>> 80 entries '?' for help

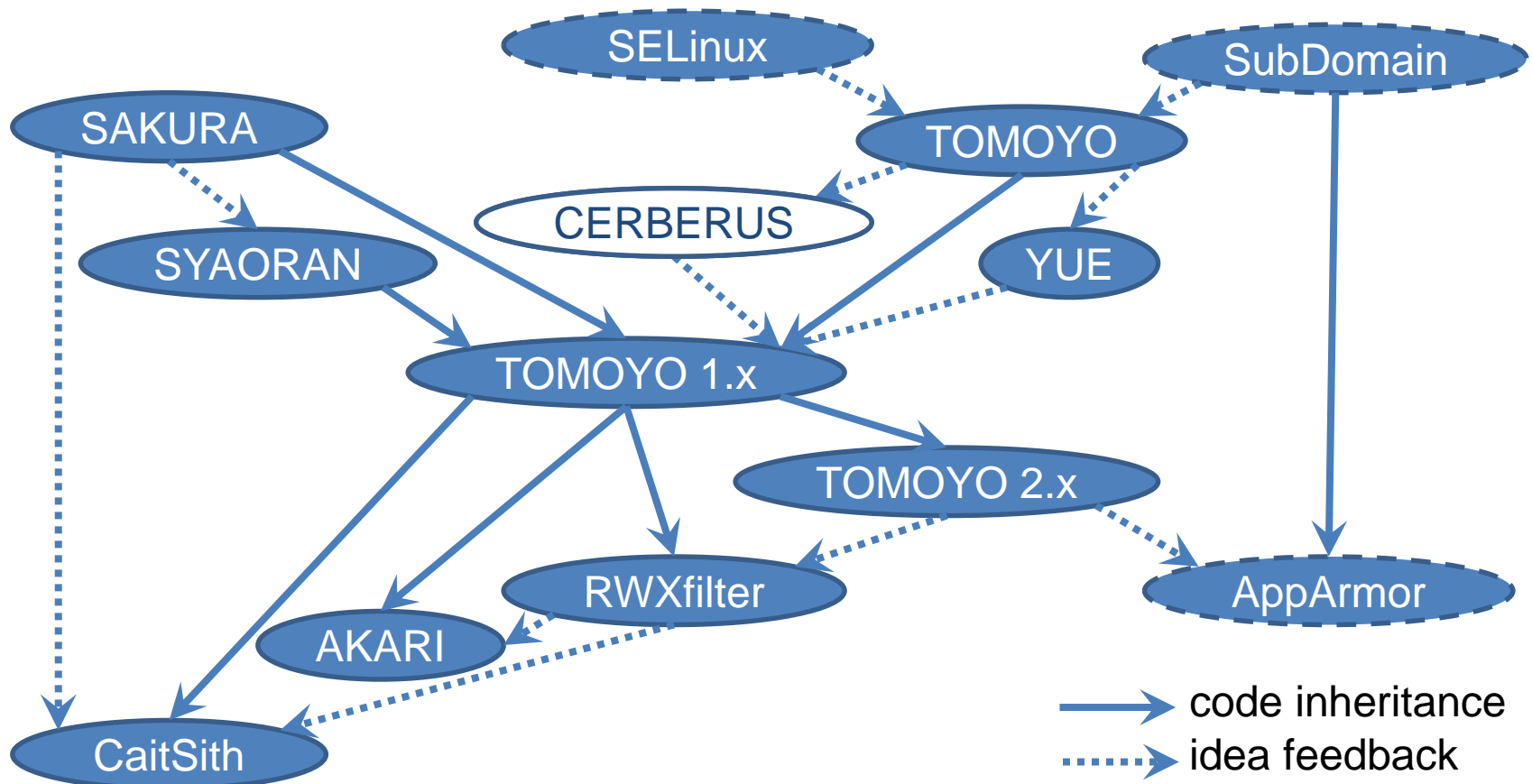
kernel> /sbin/init /bin/sh /etc/rc.d/rc
0: file execute /bin/plymouth exec.realpath="/bin/plymouth" exec.argv[0]="/bin/plymouth"
1: file execute /etc/rc.d/rc3.d/S10network exec.realpath="/etc/rc.d/init.d/network" exec.argv[0]="/etc/rc3.d/S10network"
2: file execute /etc/rc.d/rc3.d/S11auditd exec.realpath="/etc/rc.d/init.d/auditd" exec.argv[0]="/etc/rc3.d/S11auditd"
3: file execute /etc/rc.d/rc3.d/S12rsyslog exec.realpath="/etc/rc.d/init.d/rsyslog" exec.argv[0]="/etc/rc3.d/S12rsyslog"
4: file execute /etc/rc.d/rc3.d/S22messagebus exec.realpath="/etc/rc.d/init.d/messagebus" exec.argv[0]="/etc/rc3.d/S22messagebus"
5: file execute /etc/rc.d/rc3.d/S24avahi-daemon exec.realpath="/etc/rc.d/init.d/avahi-daemon" exec.argv[0]="/etc/rc3.d/S24avahi-daemon"
6: file execute /etc/rc.d/rc3.d/S25netfs exec.realpath="/etc/rc.d/init.d/netfs" exec.argv[0]="/etc/rc3.d/S25netfs"
7: file execute /etc/rc.d/rc3.d/S26udev-post exec.realpath="/etc/rc.d/init.d/udev-post" exec.argv[0]="/etc/rc3.d/S26udev-post"
8: file execute /etc/rc.d/rc3.d/S55sshd exec.realpath="/etc/rc.d/init.d/sshd" exec.argv[0]="/etc/rc3.d/S55sshd"
9: file execute /etc/rc.d/rc3.d/S56xinetd exec.realpath="/etc/rc.d/init.d/xinetd" exec.argv[0]="/etc/rc3.d/S56xinetd"
10: file execute /etc/rc.d/rc3.d/S80sendmail exec.realpath="/etc/rc.d/init.d/sendmail" exec.argv[0]="/etc/rc3.d/S80sendmail"
11: file execute /etc/rc.d/rc3.d/S90crond exec.realpath="/etc/rc.d/init.d/crond" exec.argv[0]="/etc/rc3.d/S90crond"
12: file execute /etc/rc.d/rc3.d/S91smb exec.realpath="/etc/rc.d/init.d/smb" exec.argv[0]="/etc/rc3.d/S91smb"
13: file execute /etc/rc.d/rc3.d/S99local exec.realpath="/etc/rc.d/rc.local" exec.argv[0]="/etc/rc3.d/S99local"
14: file execute /sbin/consoletype exec.realpath="/sbin/consoletype" exec.argv[0]="/sbin/consoletype"
15: file execute /sbin/initctl exec.realpath="/sbin/initctl" exec.argv[0]="/sbin/initctl"
16: file execute /sbin/runlevel exec.realpath="/sbin/runlevel" exec.argv[0]="/sbin/runlevel"
17: file getattr /
18: file getattr /bin/plymouth
19: file getattr /dev/console
20: file getattr /etc/ld.so.cache
21: file getattr /etc/nsswitch.conf
22: file getattr /etc/passwd
23: file getattr /etc/profile.d/lang.sh
24: file getattr /etc/rc.d/init.d/auditd
25: file getattr /etc/rc.d/init.d/avahi-daemon
26: file getattr /etc/rc.d/init.d/crond
27: file getattr /etc/rc.d/init.d/functions
28: file getattr /etc/rc.d/init.d/messagebus
29: file getattr /etc/rc.d/init.d/netfs
30: file getattr /etc/rc.d/init.d/network
31: file getattr /etc/rc.d/init.d/rsyslog
32: file getattr /etc/rc.d/init.d/sendmail
33: file getattr /etc/rc.d/init.d/smb
34: file getattr /etc/rc.d/init.d/sshd
35: file getattr /etc/rc.d/init.d/udev-post
36: file getattr /etc/rc.d/init.d/xinetd
37: file getattr /etc/rc.d/rc
38: file getattr /etc/rc.d/rc.local
39: file getattr /etc/rc.d/rc3.d/
40: file getattr /etc/sysconfig/i18n
41: file getattr /etc/sysconfig/init
42: file getattr /lib/libc-2.12.so
43: file getattr /lib/libdl-2.12.so
44: file getattr /lib/libnss_files-2.12.so
45: file getattr /lib/libtinfo.so.5.7
46: file getattr /sbin/initctl
47: file getattr /usr/lib/gconv/gconv-modules.cache
48: file getattr /usr/lib/locale/locale-archive
49: file getattr /usr/share/locale/locale.alias
50: file getattr proc/meminfo
51: file ioctl /dev/console 0x5401
52: file ioctl /dev/console 0x540F
53: file ioctl /dev/console 0x5410
54: file ioctl /etc/rc.d/rc 0x5401
55: file read /bin/bash
56: file read /dev/console
57: file read /dev/tty
58: file read /etc/ld.so.cache
```

Access request restricting function on a per-domain basis

- ▶ I attempted to generate SELinux's policy from the output.
 - ▶ I gave up because I could not map TOMOYO's pathnames into SELinux's labels.
- ▶ Instead, I added function to restrict access requests based on the observed output
 - ▶ At this time, I didn't distinguish requests for modifying directory entries.
 - ▶ In other words, the granularity was similar to DAC's rwx.

CERBERUS (2004.1-)

- ▶ One of advanced usages of TOMOYO, which protects from login brute force attacks.

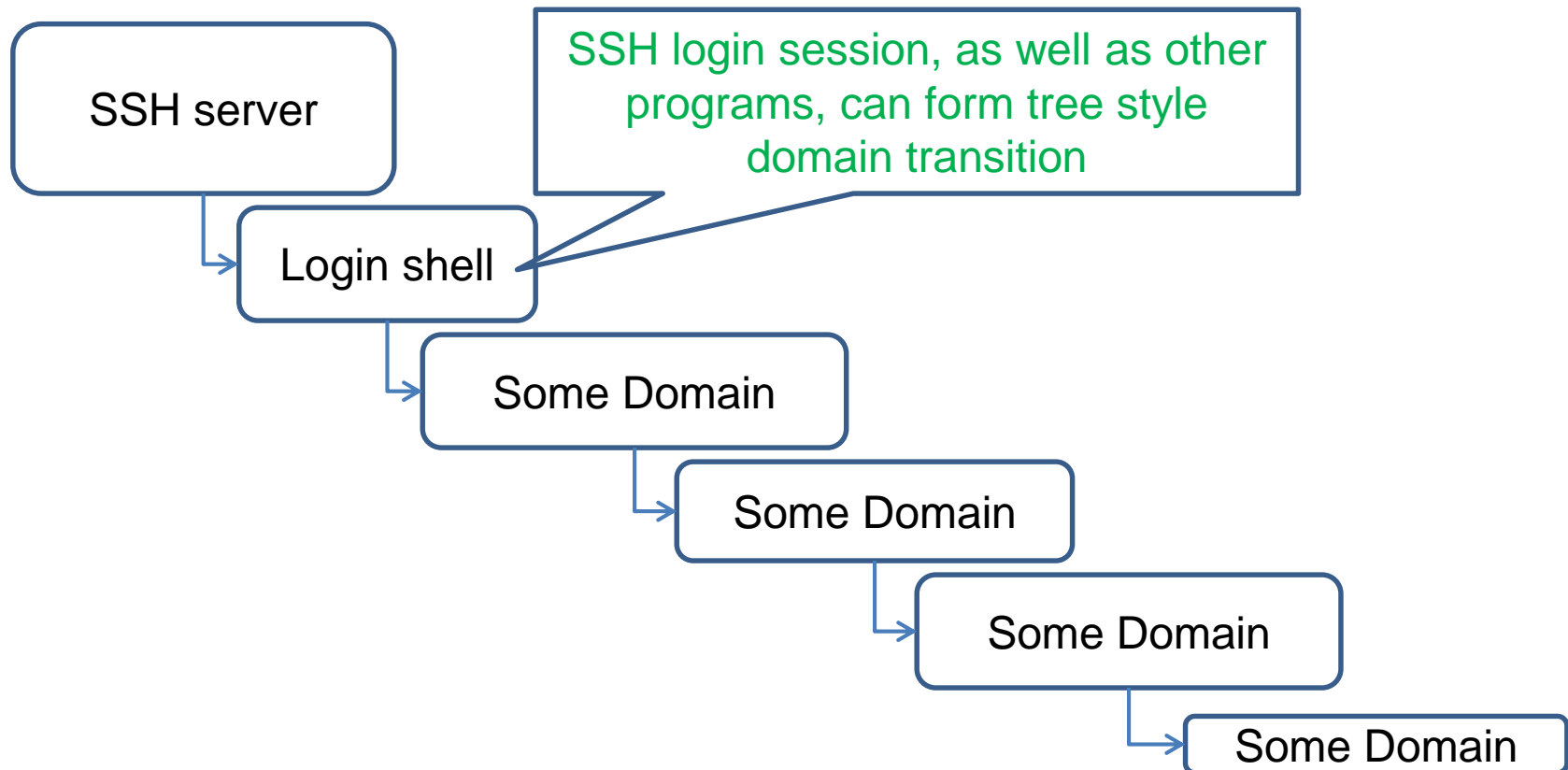


CERBERUS

- ▶ SSH brute force attacks can break protection by MAC.
 - ▶ Why not to enforce extra authentications?
- ▶ Code name: "Chained Enforceable Re-authentication Barrier Ensures Really Unbreakable Security"
- ▶ Topics on CERBERUS
 - ▶ Anti brute force technique via multiplexed user authentication.

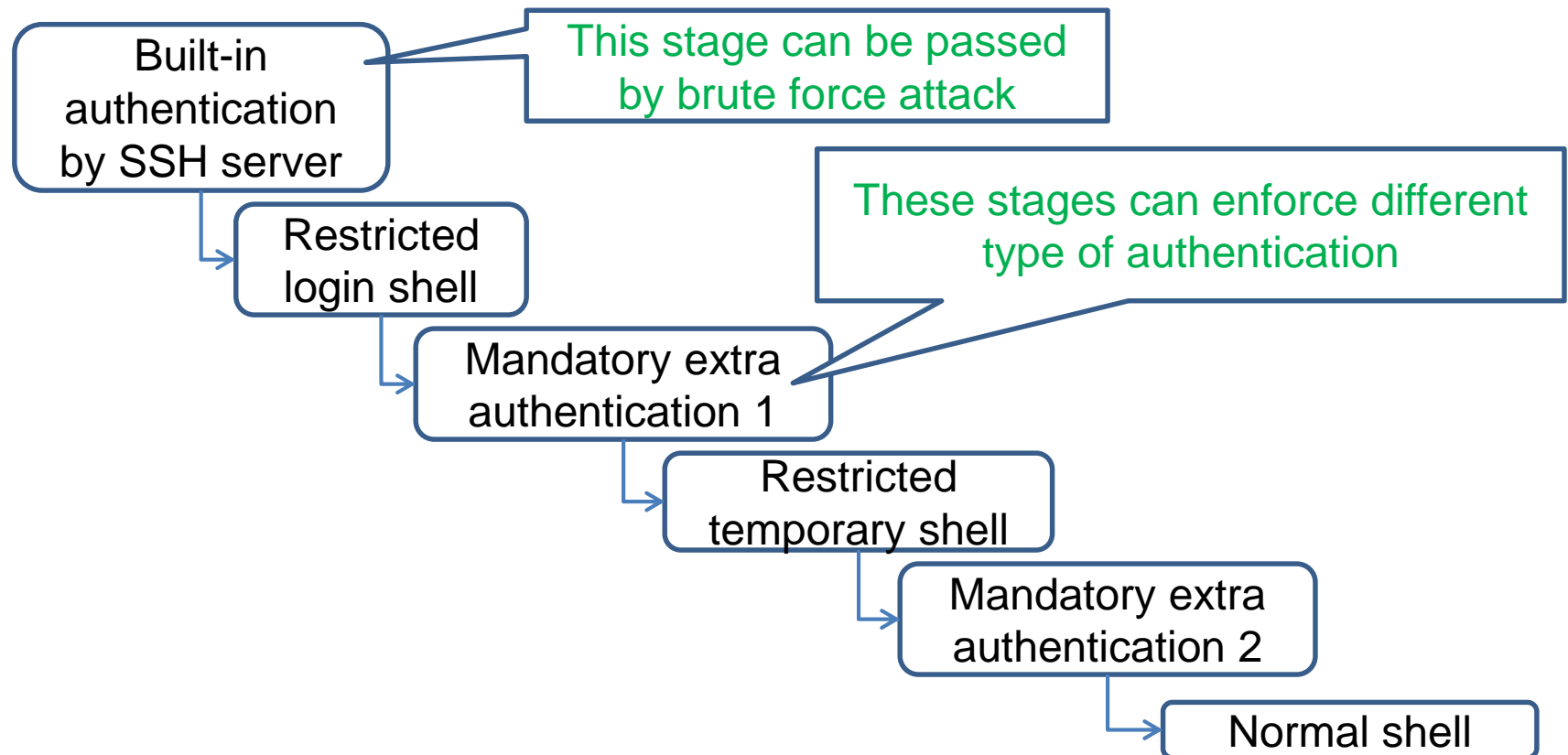
Anti brute force technique via multiplexed user authentication.

- ▶ I noticed that we can deploy user authentications for multiple times, by using TOMOYO's tree style state transition.



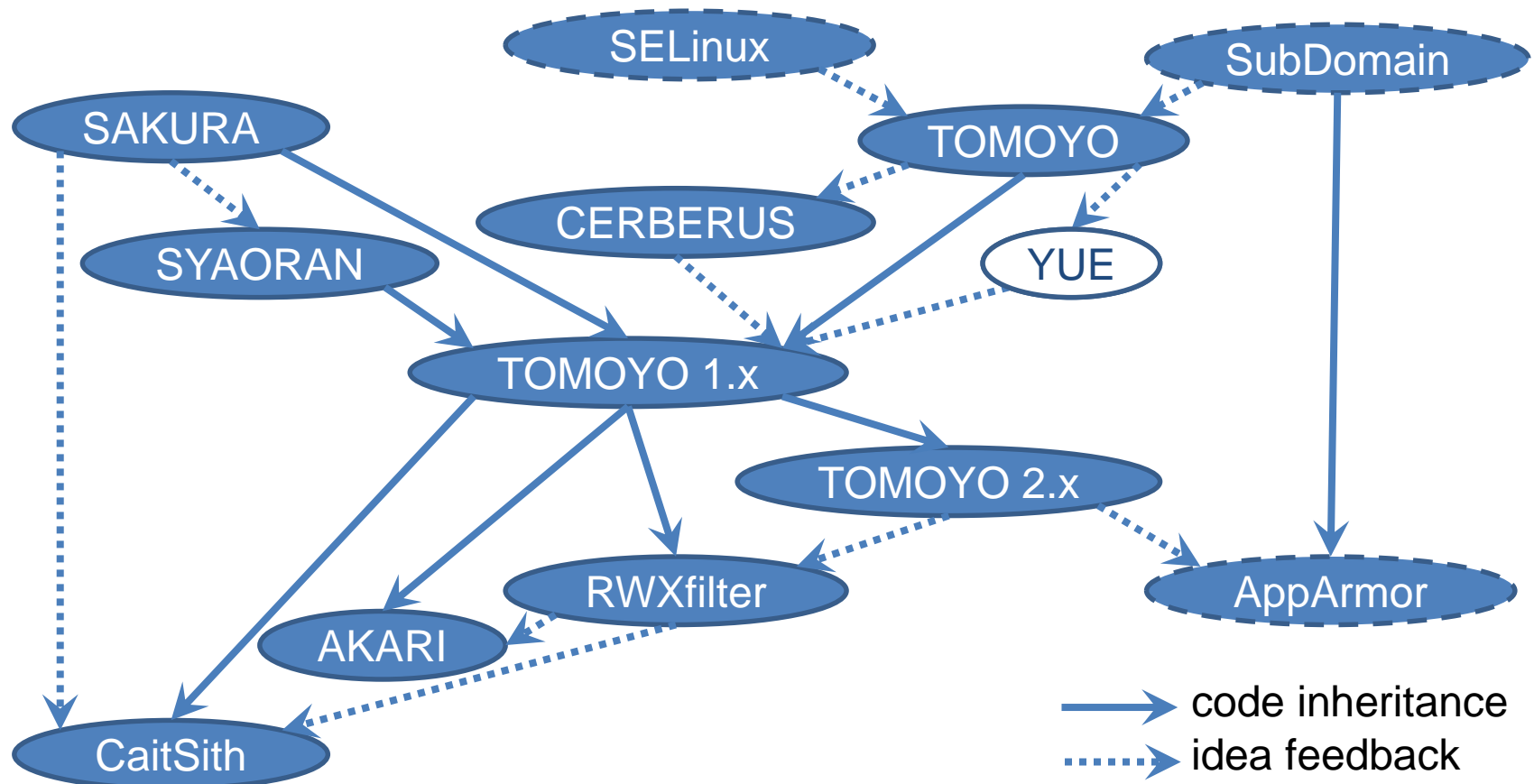
Anti brute force technique via multiplexed user authentication.

- ▶ I noticed that we can deploy user authentications for multiple times, by using TOMOYO's tree style state transition.



YUE (2004.1-)

- ▶ One of advanced usages of TOMOYO, which divides privileges for administrative jobs.

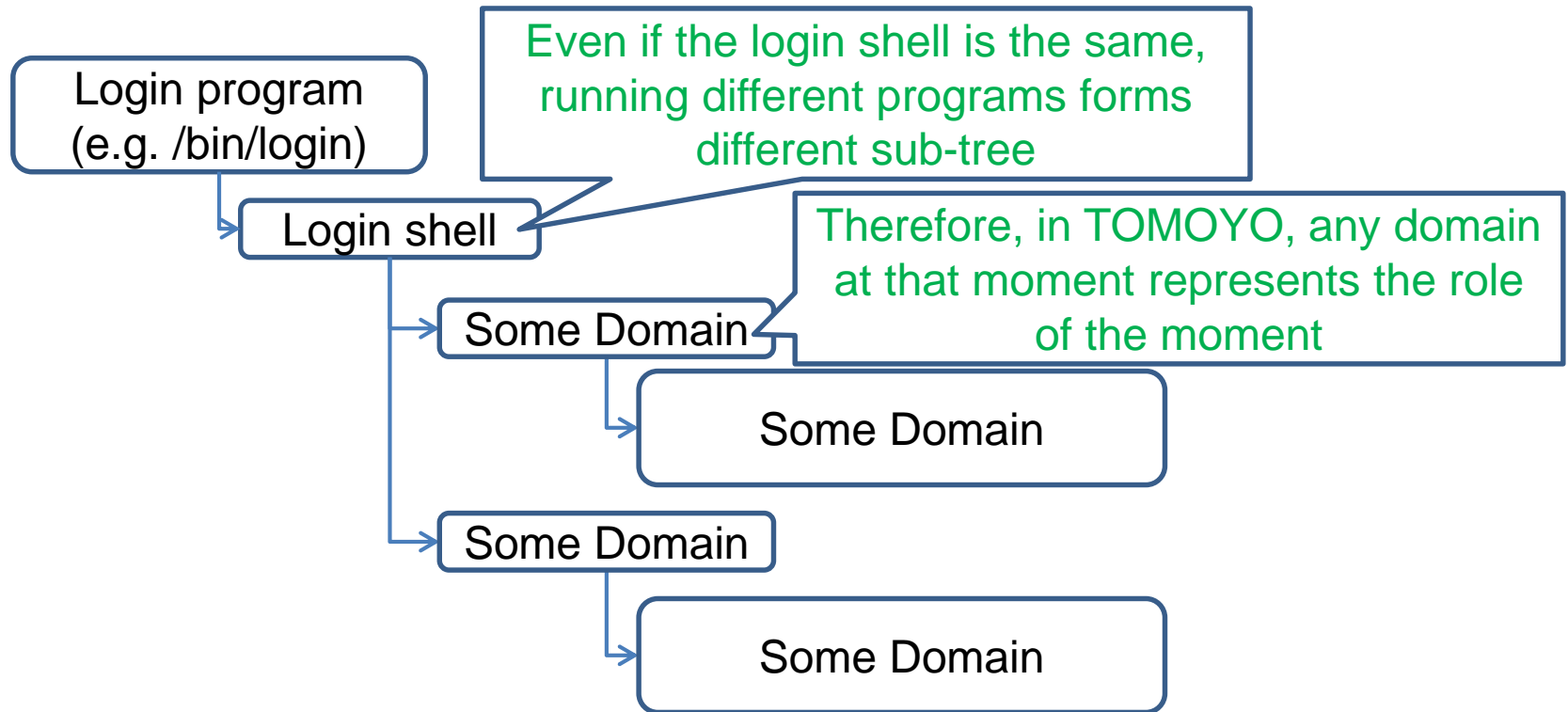


YUE

- ▶ The root privileges are needed for doing administrative jobs.
 - ▶ But only one root user can exist.
 - ▶ Why not to divide privileges using TOMOYO's tree style state transition?
- ▶ Code name: "Your User-role Enforcer"
- ▶ Topics on YUE
 - ▶ Privilege division technique like Role Based Access Control (RBAC)

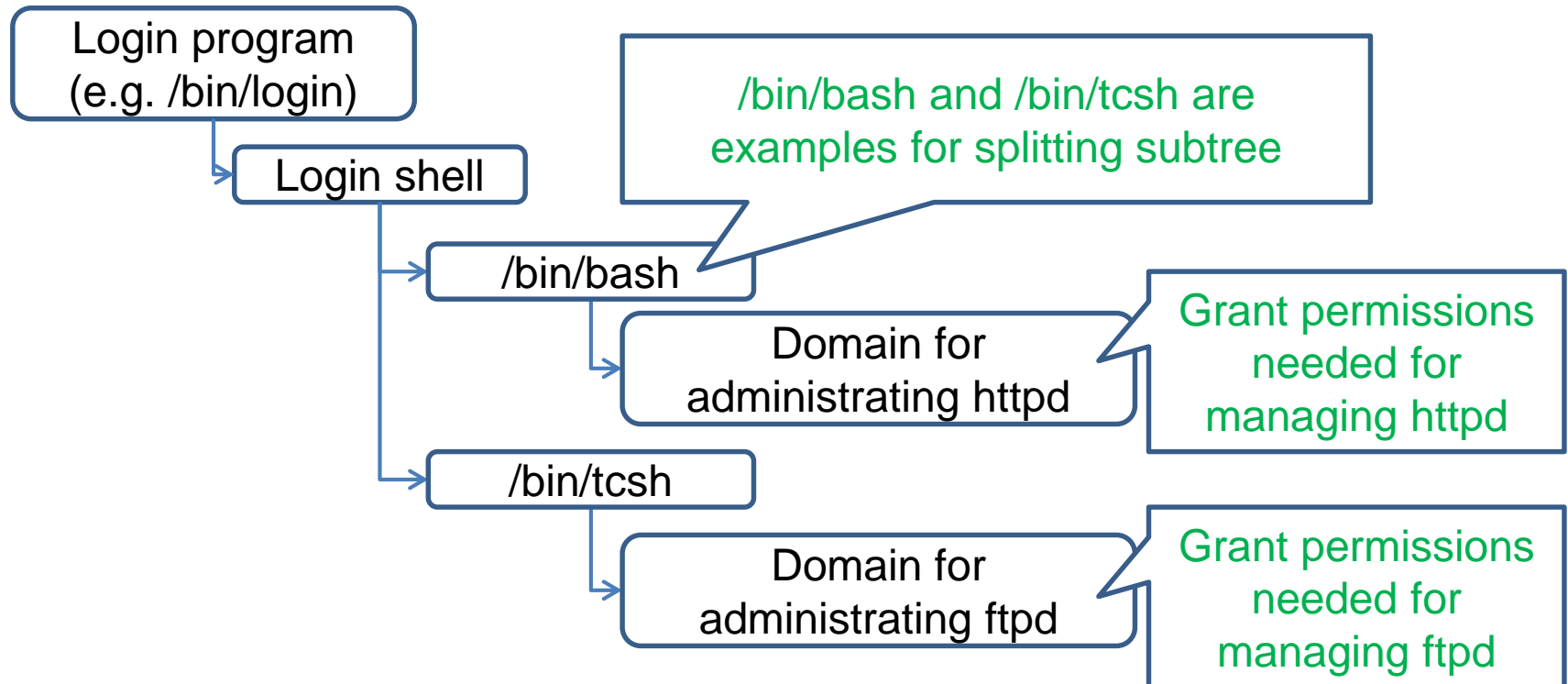
Privilege division technique like Role Based Access Control (RBAC)

- ▶ I noticed that we can divide privileges for administrative jobs into arbitrary groups, by using TOMOYO's tree style state transition.



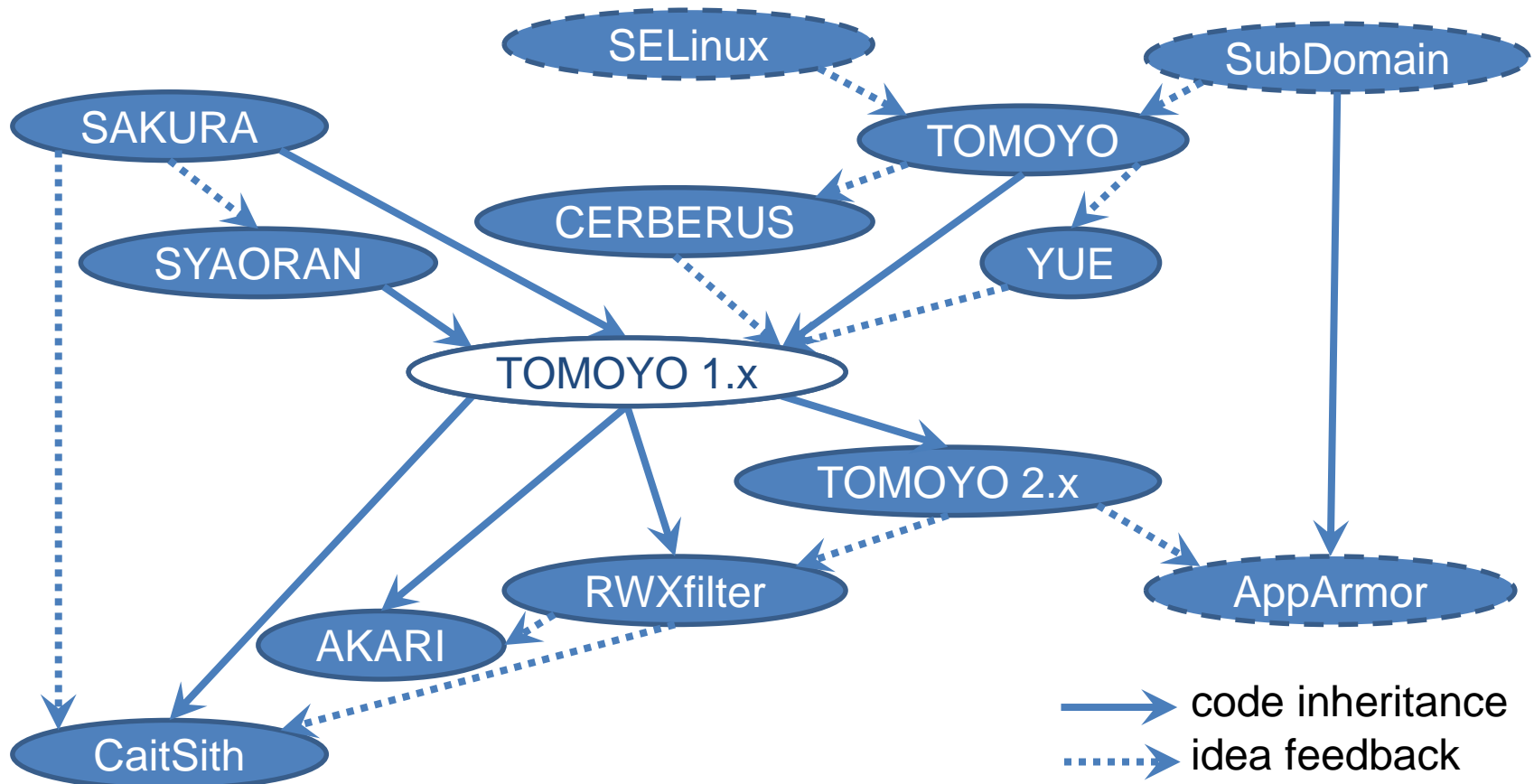
Privilege division technique like Role Based Access Control (RBAC)

- ▶ I noticed that we can divide privileges for administrative jobs into arbitrary groups, by using TOMOYO's tree style state transition.



TOMOYO 1.x (2005.11-)

- ▶ The origin of my various derivative works.



TOMOYO 1.0 (2005.11-2006.3)

- ▶ A version which was published as a GPL open source software with outcomes since 2003.4.
 - ▶ SAKURA which handles restriction of namespace manipulation on system-wide basis
 - ▶ TOMOYO which handles restriction of access requests on a per-domain basis
 - ▶ SYAORAN which protects /dev partition
 - ▶ CERBERUS which protects from login brute force attacks
 - ▶ YUE which divides privileges for administrative jobs

TOMOYO 1.1 (2006.4-2006.9)

- ▶ User space programs behaves differently depending on names.
 - ▶ Gear towards more strictly restricting names which affects program's behavior, in addition to restricting whether the pathname is readable/writable/executable or not.
 - ▶ Differentiate directory entry modification actions (i.e. mkdir, rmdir, create, unlink, mksock, mkfifo, mkchar, mkbblock, link, symlink, rename, truncate) from "write" action.
- ▶ Made it possible to handle policy violations interactively.
 - ▶ Give users a chance to handle unexpected events which sometimes occur upon running software updaters(e.g. yum/apt).

TOMOYO 1.2 (2006.9-2006.11)

- ▶ Check invocation name (a.k.a. argv[0]) upon checking program's execute permission.
 - ▶ Because multi-call programs (e.g. busybox) behave differently depending on the invocation name.
- ▶ Check current process's user ID etc. and file's owner ID etc. upon checking permissions.
 - ▶ Checking only pathname is not sufficient.

TOMOYO 1.3 (2006.11-2006.3)

- ▶ Made it possible to specify whether to enforce access control or not, on a per-domain basis by introducing profile number which takes an integer between 0 and 255.
 - ▶ Profile number allowed users to use TOMOYO like SELinux's targeted policy.
 - ▶ Profile number also allowed users to use different functionality on different programs.
- ▶ Made it possible to suppress/reset domain transitions as needed.
 - ▶ Support various patterns of domain transition.

TOMOYO 1.4 (2007.4-2007.9)

- ▶ Support pathname subtraction operator.
 - ▶ Generally, filenames starting with a dot should be treated differently.
 - ▶ `/var/www/html/¥*¥-.¥*`
 - ▶ For example, differentiate `/var/www/html/.htaccess` from `/var/www/html/index.html`
- ▶ Support x86_64 architecture.
- ▶ Started developing an LSM version called TOMOYO 2.x and started challenges for inclusion into Linux mainline kernel.
 - ▶ TOMOYO Linux project had a BoF session at Ottawa Linux Symposium 2007.

TOMOYO 1.5 (2007.9-2008.3)

- ▶ Improved usability in order to differentiate TOMOYO from AppArmor.
 - ▶ At that time, TOMOYO was challenging for inclusion into Linux mainline kernel. But since both TOMOYO and AppArmor used pathnames in their policy configuration, TOMOYO and AppArmor are regarded as "no need to include both implementations into Linux mainline".
 - ▶ I tried to show how attentive TOMOYO is.
- ▶ Made possible to run TOMOYO 1.x in parallel with SELinux.
 - ▶ The label based MAC and the name based MAC play complementary role.
 - ▶ Thus, these should be able to run in parallel.

TOMOYO 1.6 (2008.4-)

- ▶ **Various functionality/usability enhancements**
 - ▶ check argv[]/envp[] upon program execution
 - ▶ support execute handler which intercepts program execution request and validates/sanitizes argv[]/envp[] etc.
 - ▶ This can silently terminate a process if the process issued suspicious execve() request (e.g. /bin/sh without appropriate argv[]/envp[])
 - ▶ support stateful acl
 - ▶ and many more
- ▶ **This is the base code for RWXfilter and TOMOYO 2.2.**

TOMOYO 1.7 (2009.9-)

- ▶ Current stable version.
- ▶ Checks not only pathnames but also various attributes passed together.
- ▶ Changed module name of TOMOYO 1.x to CCSecurity since TOMOYO 2.2 was included into Linux mainline kernel.
 - ▶ Took occasion to review the division of the roles.
 - ▶ Integrate system-wide access restrictions (SAKURA) into a per-domain access restrictions (TOMOYO).
 - ▶ Integrate access restrictions in /dev filesystem (SYAORAN) into a per-domain access restrictions (TOMOYO).

System-wide access restriction vs. A per-domain access restriction

- ▶ Why did I treat system-wide access restriction and a per-domain access restriction separately?
 - ▶ Mainly enthusiasm for the code names.
- ▶ In order to restrict more precisely, why not to specify on a per-domain basis rather than system-wide basis, for TOMOYO 1.x can apply access restrictions to all processes?
 - ▶ I thought (at that time) "Definitely".
- ▶ As a background of this decision, I was targeting for more finer grained restriction in order to support LXC (pivot_root) users.
 - ▶ Since TOMOYO 2.2 supported only TOMOYO (a per-domain access restriction), I removed SAKURA (system-wide access restriction) from TOMOYO 1.7.

Access restriction by filesystems vs. Access restriction by domains

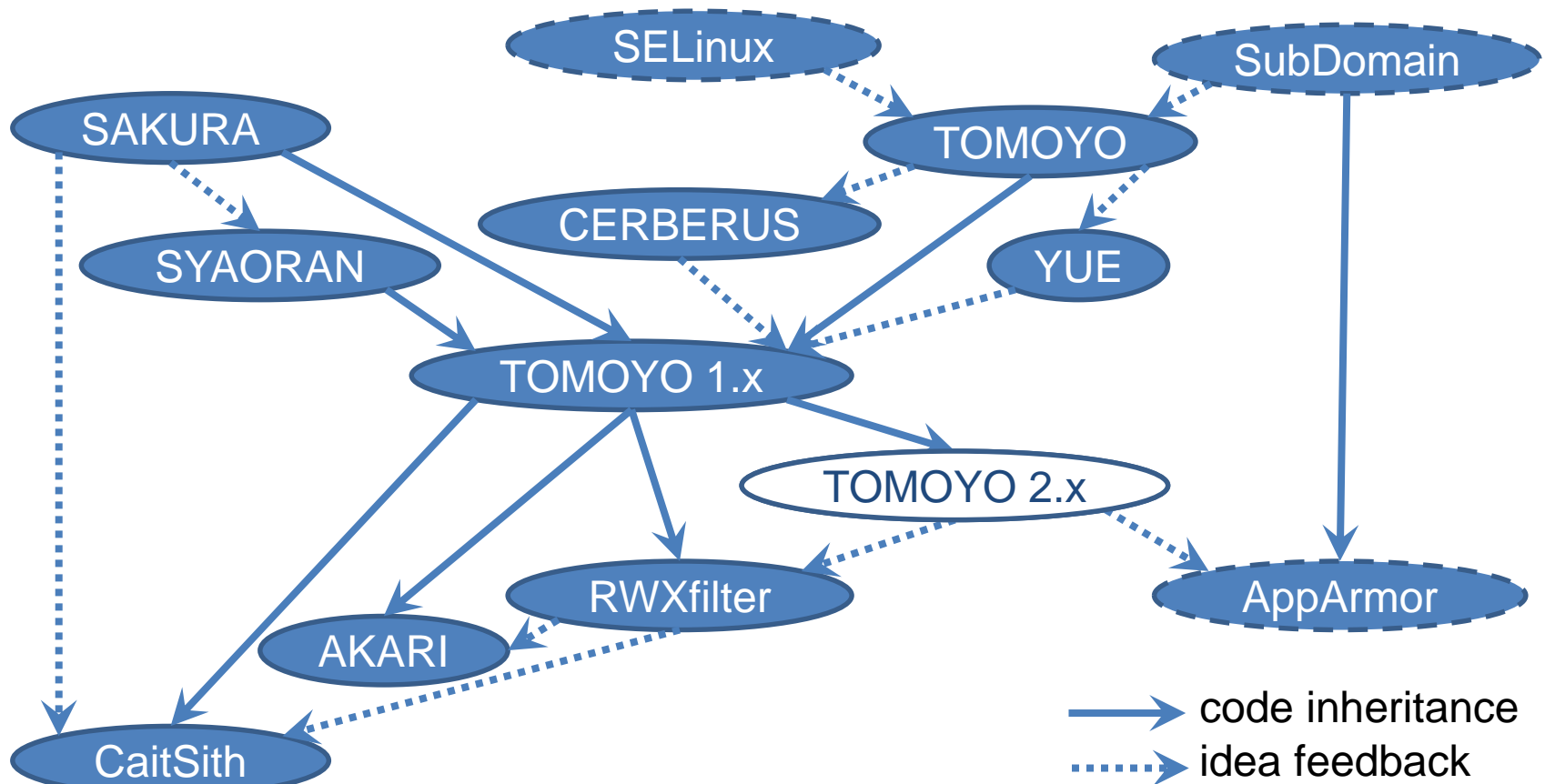
- ▶ Why did I have to restrict at filesystem layer? Now, TOMOYO can check not only filenames but also file's attributes. Should I continue maintaining /dev filesystem?
 - ▶ I thought (at that time) "Not worth maintaining".
 - ▶ I removed SYAORAN filesystem from TOMOYO 1.7, for SYAORAN filesystem conflicts with udev approach.

TOMOYO 1.8 (2010.11-)

- ▶ Current latest version which supports Linux 2.6.27-3.5 kernels.
- ▶ Reviewed internal structures, removed redundant/legacy functionality, renamed keywords in the policy syntax.
- ▶ Made it possible to preserve kernel ABI by introducing hooks into `fork()` and `exit()` (instead of appending original fields to `task_struct` which results in kernel ABI breakage).
 - ▶ It became possible to treat like distributor's stock kernels.
- ▶ This is the base code for AKARI and CaitSith.

TOMOYO 2.x (2007.6-)

- ▶ The mainlined version of TOMOYO 1.x.



TOMOYO 2.2 (2009.6-2010.10)

- ▶ A version which was mainlined in Linux 2.6.30 kernel.
- ▶ Only core function of TOMOYO 1.6 is implemented.
- ▶ Addition of missing LSM hooks for file related actions has completed by Linux 2.6.33 kernel.

TOMOYO 2.3 (2010.10-2011.10)

- ▶ A version which was included into Linux 2.6.36-3.0 kernels.
- ▶ Major functionality regarding file related actions in TOMOYO 1.7 is implemented.

TOMOYO 2.4 (2011.10-2012.1)

- ▶ A version which was included into Linux 3.1 kernel.
- ▶ A version which has practically usable function.
- ▶ Major functionality regarding file related actions in TOMOYO 1.8 is implemented.

TOMOYO 2.5 (2012.1-)

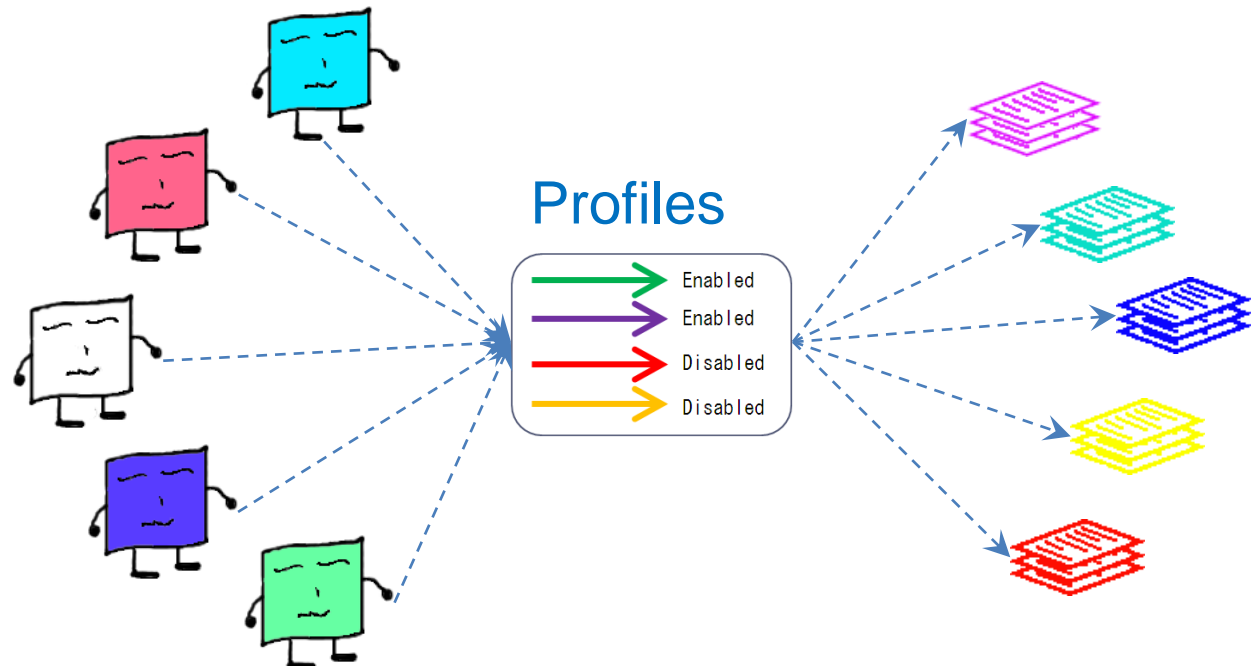
- ▶ A version which is included into Linux 3.2 and later kernels.
 - ▶ It is possible to backport this version up to Linux 2.6.33 kernel without modifying outside of security/tomoyo/ directory.
- ▶ Major functionality in TOMOYO 1.8 is implemented.
- ▶ Not yet implemented functionality
 - ▶ execute handler
 - ▶ Checking permission of incoming network packets.
 - ▶ capability (Maybe seccomp mode 2 can substitute?)
 - ▶ Checking permission of binary loader programs.
 - ▶ Running with other LSM modules in parallel.

Things I achieved

- ▶ In-kernel access control which takes into account side effects in the user space
 - ▶ Preserving only "whether the file is readable/writable/executable or not" is not sufficient.
 - ▶ Firewall which checks various parameters which are represented in the form of string or numeric values.
- ▶ Know all possible behaviors from boot to shutdown.
 - ▶ Covers all processes
 - ▶ Use `task_struct` for defining domains.
 - ▶ Sense of safety that can cover all processes
 - ▶ TOMOYO 1.8 is used in Android devices.
 - ▶ Focuses on preventing from unwanted behaviors.

Things I struggled

- ▶ Implement functionality which will be useful, while keeping psychological barrier as low as possible.
- ▶ First step(2005.11)
 - ▶ I made it possible to enable/disable on a per-action basis.

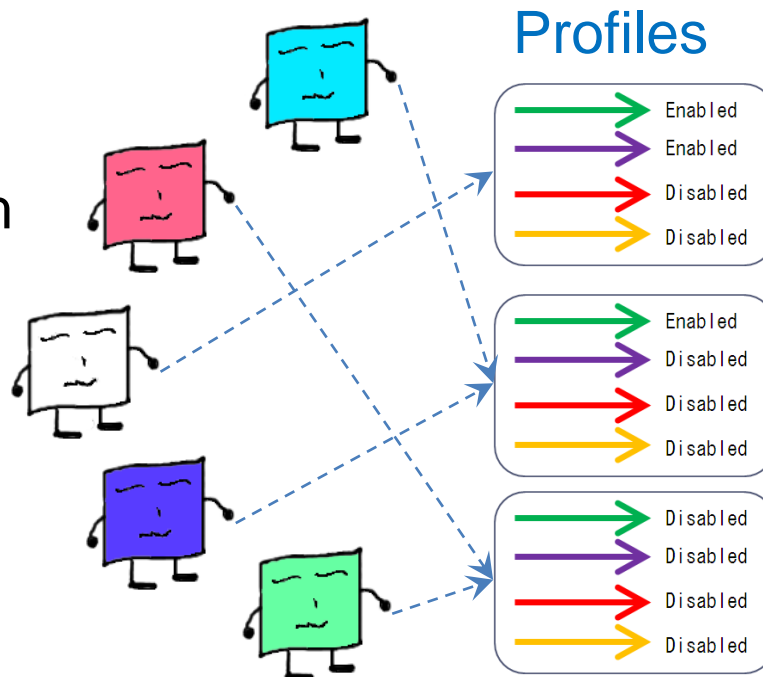


Things I struggled

▶ Second step(2006.11)

- ▶ Even though TOMOYO can apply restrictions on all processes, user's skill are not catching up.
- ▶ I made it possible to enable/disable access restrictions on a per-domain basis.

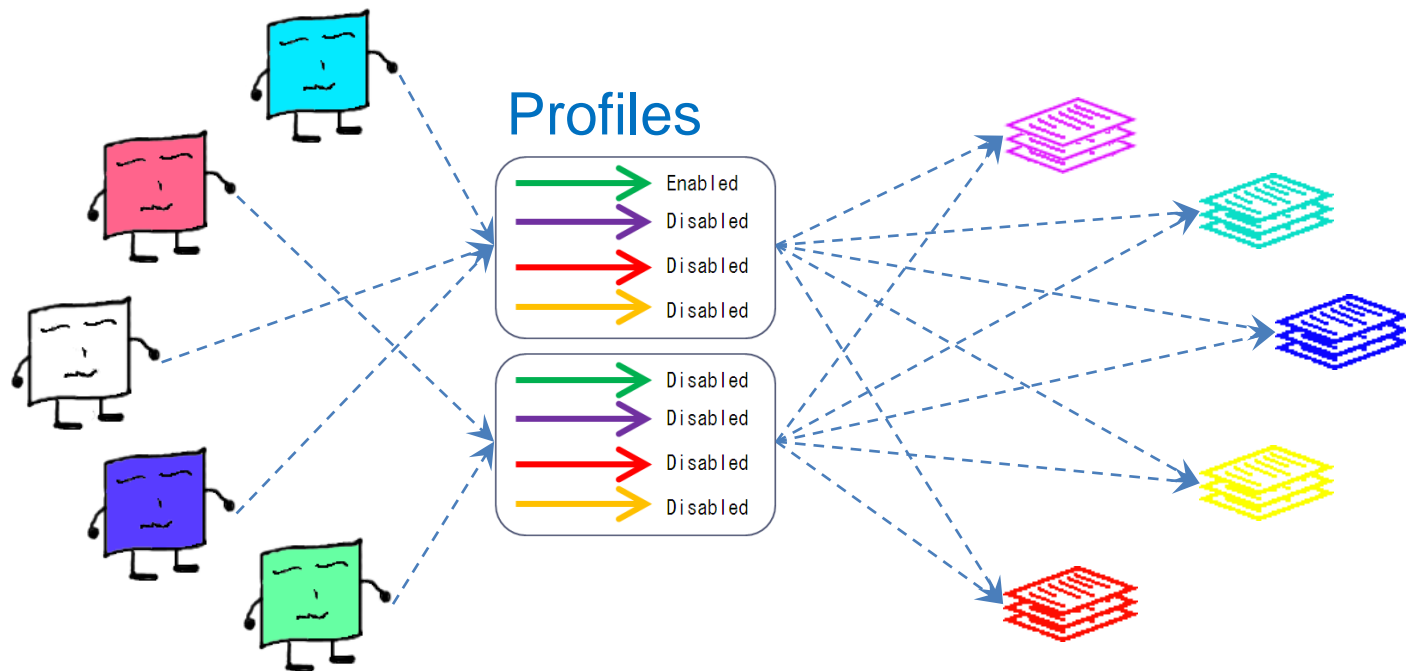
- ▶ This also made it possible to use build-up approach by switching profiles after the policy is developed.



Things I struggled

▶ Third step(2011)

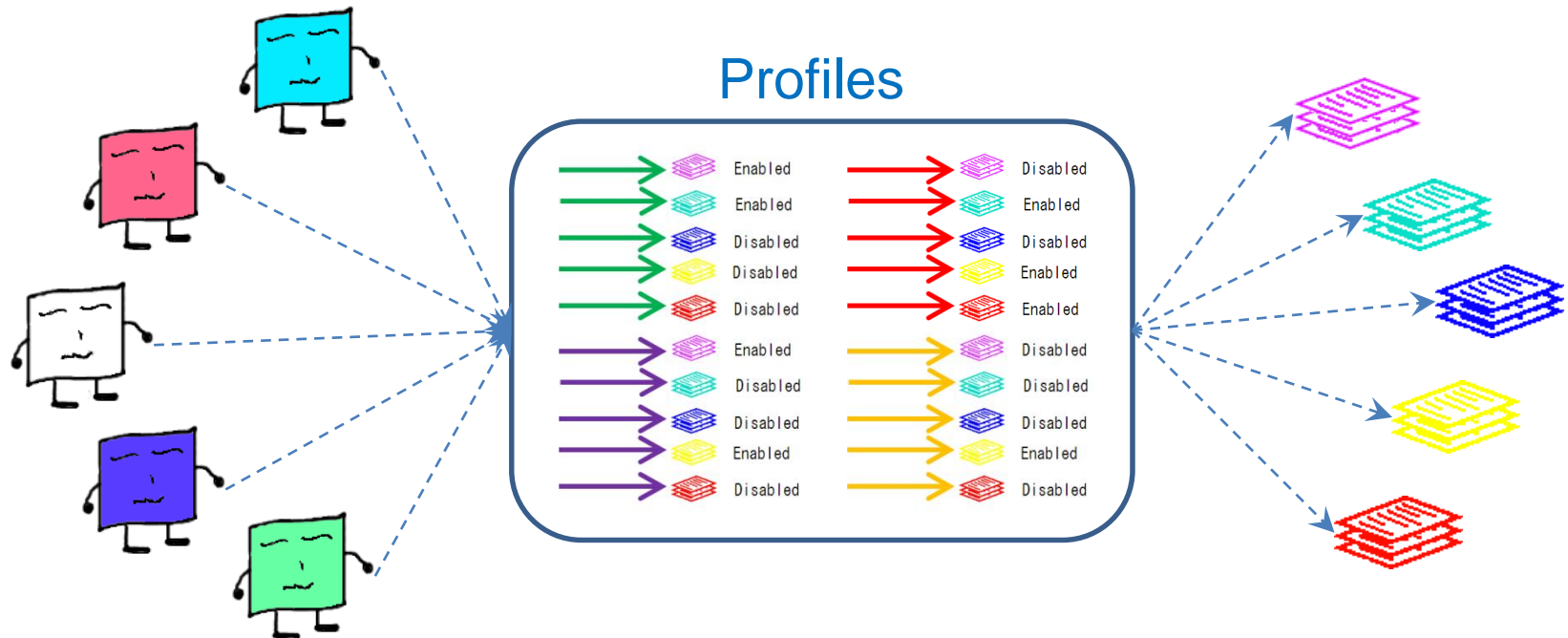
- ▶ Even if restricting only file related actions, it is too difficult to switch all files to enforcing mode at once.



Things I struggled

▶ Third step(2011)

- ▶ I considered profiles on a per-filename basis.
 - ▶ I didn't implement because it will become too messy.



Things I struggled

▶ Third step(2011)

- ▶ I considered denylisting approach.
 - ▶ I didn't implement because it conflicts access control modes in the profiles
 - ▶ The permissive mode is defined as "check access requests but do not reject", but denylisting will make the permissive mode no longer permissive.
 - ▶ It is impossible to apply denylisting approach before defining domains.
 - ▶ How should TOMOYO handle denylist when TOMOYO automatically generated domains?

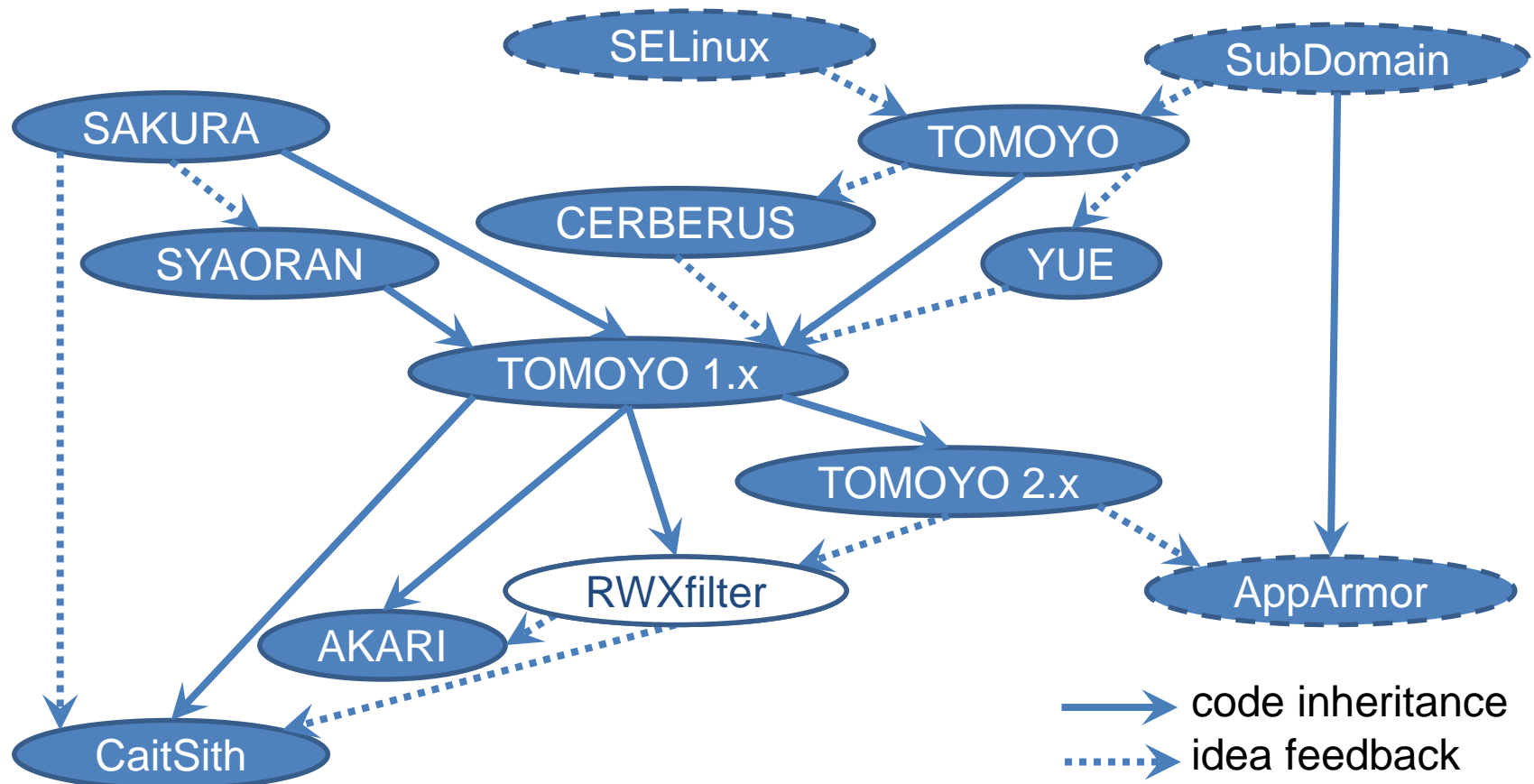
=> Leads to CaitSith

Chapter 3

Things I experienced with continuing
enhancement of ease of use

RWXfilter (2010.2-2010.4)

- ▶ The trigger for seriously considering Linux user's real opinions.



A request from RHEL users.

- ▶ SELinux is too difficult for us to use. Please develop a single function access control mechanism that can be loaded into RHEL kernels as a loadable kernel module.
 - ▶ I decided to implement a loadable kernel module that makes use of LSM interface.
- ▶ Please allow users to apply access control on only specific files.
 - ▶ I decided to filter only "read/write/execute" actions in order to minimize barrier for users.
 - ▶ Code name: "Read/Write/Execute filter", or in short "RWXfilter".

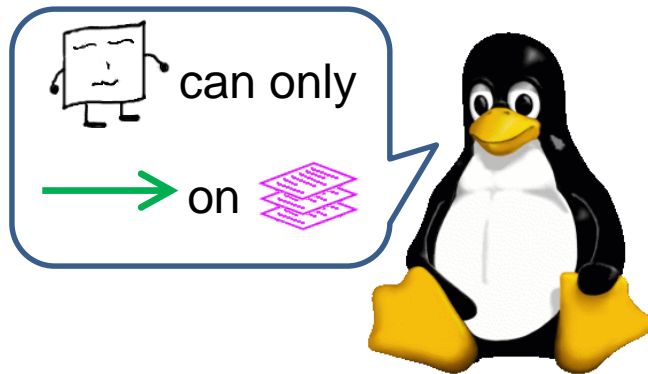
Dilemmas

- ▶ Existing MAC implementations assume "define domains first, and then associate permissions/resources to domains".
 - ▶ But it is difficult to apply such approach to all processes.
- ▶ But it ruins the value of MAC if there is a process which such approach is not applied.
 - ▶ But we cannot impose on users the burden of managing every process only for applying access control on some specific resources.

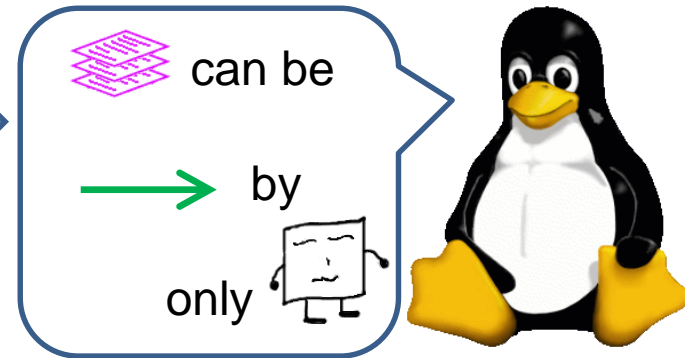
I reversed the viewpoints.

- ▶ Switch from "define domains first, and then associate actions/resources" to "define resources first, and then associate actions/domains".

Capability model



Access control list model



Policy syntax for RWXfilter

- ▶ "Resource" "Access Control Mode"

 - "Action1" by "Domain1 which action1 is allowed"

 - "Action2" by "Domain2 which action2 is allowed"

 - "Action3" by "Domain3 which action3 is allowed"

- ▶ "Resource" is TOMOYO's pathname representation.
- ▶ "Access Control Mode" is either "permissive" or "enforcing".
- ▶ "ActionX" is one of "read", "write" or "execute".
- ▶ "DomainX which actionX is allowed" is TOMOYO's domainname representation.

An example of RWXfilter's policy

- ▶ `/etc/shadow` enforcing

 - read by `<kernel> /sbin/init /sbin/mingetty /bin/login`

 - read by `<kernel> /usr/sbin/sshd`

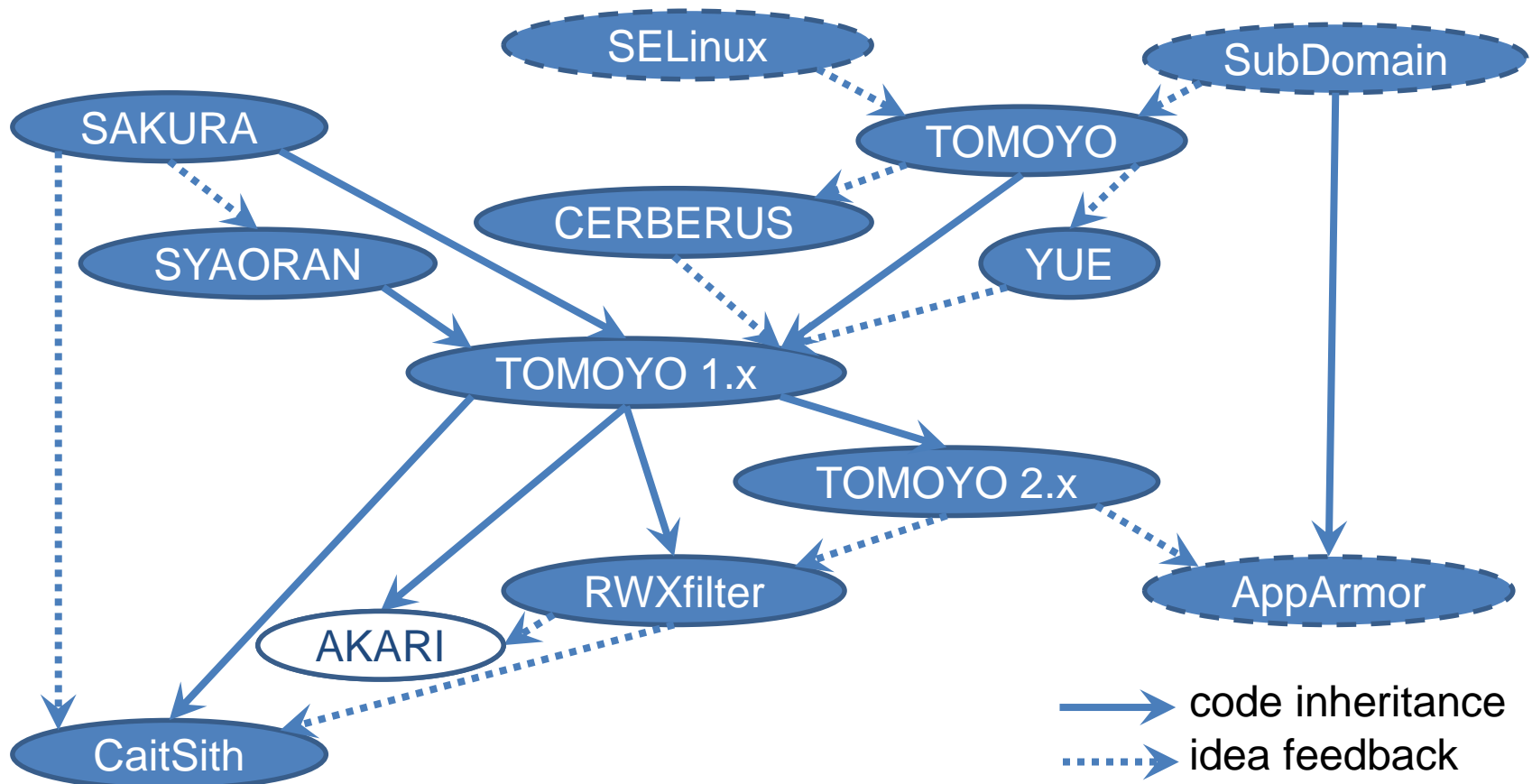
- ▶ This example will allow opening `/etc/shadow` for reading to only processes which are in "`<kernel> /sbin/init /sbin/mingetty /bin/login`" domain or "`<kernel> /usr/sbin/sshd`" domain.
- ▶ This example will deny opening `/etc/shadow` for writing, for access control mode for this pathname is "enforcing".

Consequence

- ▶ **RWXfilter was shelved because I could not establish commercial support.**
 - ▶ But RWXfilter triggered me to explore the possibility of access controls from a different point of view.
=> Leads to CaitSith.
- ▶ **I established an approach for appending another LSM module as a loadable kernel module without disabling SELinux.**
 - ▶ I demonstrated Yama with TOMOYO 2.x at Linux Security Summit held in conjunction with LinuxCon North America 2010 (Boston).
=> Leads to AKARI.

AKARI (2010.10-)

- ▶ The loadable kernel module version of TOMOYO 1.8.



Origination

- ▶ TOMOYO 2.x remains unsupported in Fedora and RHEL kernels.
 - ▶ https://bugzilla.redhat.com/show_bug.cgi?id=542986
- ▶ Replacing the kernel package is a strong psychological barrier.
 - ▶ Can't we somehow try TOMOYO more easily?

What did I think?

- ▶ There are functionality which cannot be implemented as a loadable kernel module, for some hooks are not provided by LSM.
 - ▶ TOMOYO started as an access analysis tool.
 - ▶ For analysis purpose, it would be acceptable that some functionality cannot be implemented?
 - ▶ Just try to get used to it.
- => I applied the approach which I established via RWXfilter to TOMOYO 1.8.

Consequence

- ▶ We now can use major functionality of TOMOYO 1.8 on Fedora and RHEL kernels.
- ▶ Especially useful for analysis purpose because AKARI is a loadable kernel module similar to RWXfilter.
- ▶ <https://akari.osdn.jp/>

Access

Keeping

And

Regulating

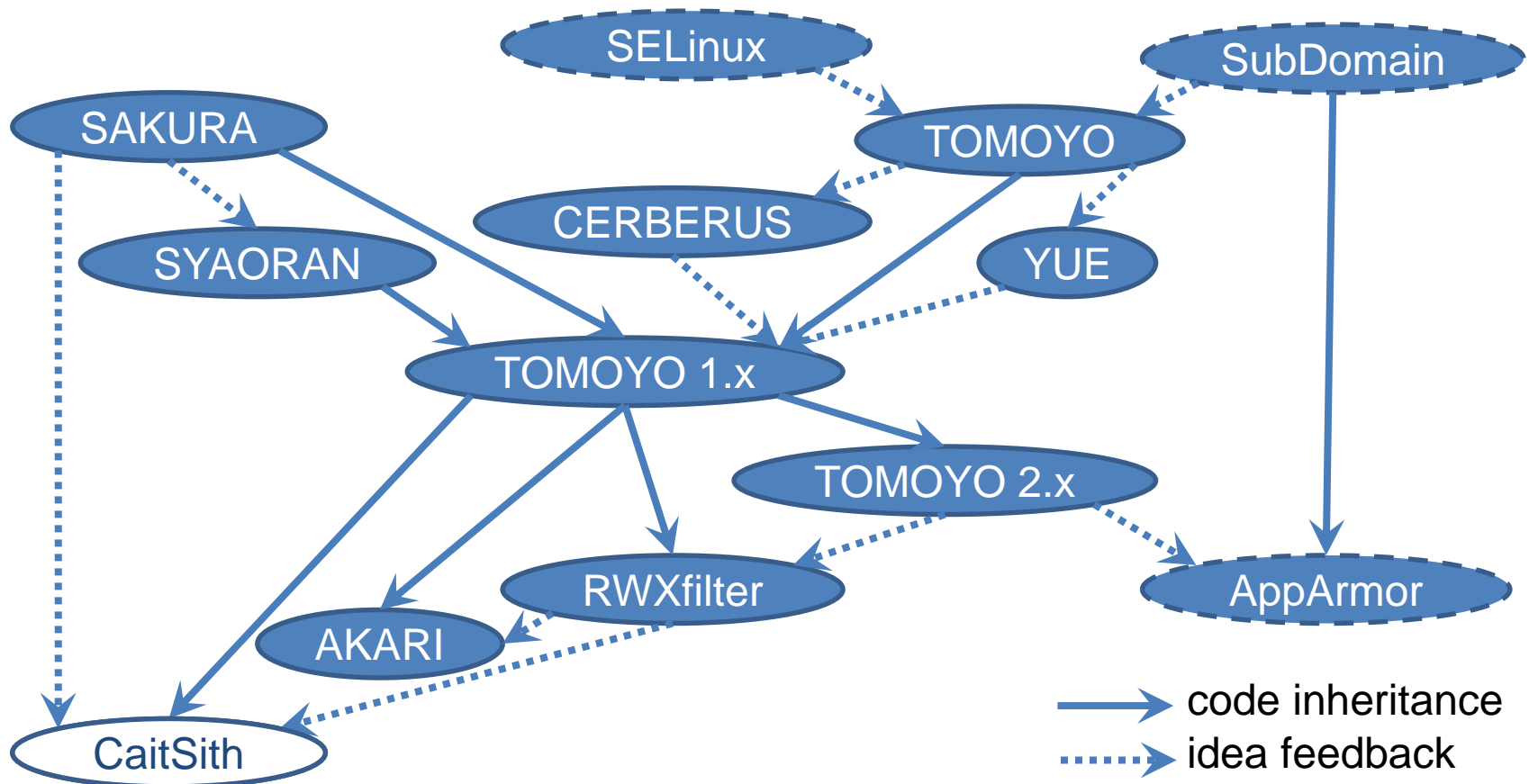
Instrument.

Chapter 4

CaitSith

CaitSith (2012.4-)

- ▶ The most powerfully and flexibly configurable policy syntax derived from 9 years of my experience.



Starting point for CaitSith

- ▶ Threats are in the behavior of the user space.
 - ▶ But the threats are getting to shift to areas where in-kernel access control cannot deal with.
 - ▶ For example, mails/tweets by error exposed to or shared by unexpected peers because of bugs in the user space applications.
 - ▶ It's a limitation for in-kernel access control.
- ▶ The "seccomp mode 2" became available in Linux 3.5.
=> Maybe we no longer need to go off the deep end at the kernel side.

What is the way MAC needs to be?

- ▶ Things I experienced with continuing enhancement of access control functionality:
 - ▶ A per-domain basis access control can do better than system wide access control, as long as access control is applied on all domains.
 - => If there is a domain which access control is not applied, it becomes a hole which would not have existed if system wide access control is used.
 - ▶ In reality, there are usually domains which access control is not applied.
 - => But system wide access control alone is not sufficient.

What is the way MAC needs to be?

- ▶ Things I experienced with continuing enhancement of access control functionality:
 - ▶ There are users who want to restrict access using denylisting approach rather than allowlisting approach.
=> Denylisting approach is difficult for TOMOYO because TOMOYO automatically creates domain.

What is the way MAC needs to be?

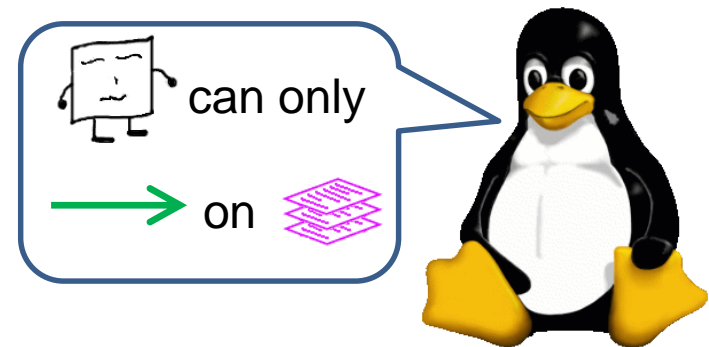
- ▶ Things I experienced with continuing enhancement of ease of use:
 - ▶ There are users who want to restrict access from the point of view of resources rather than that of processes.
=> Access controls which depends on "define domains first" cannot handle this request.
 - ▶ There are users who want to restrict access on specific resources.
=> Access controls which depends on allowlisting cannot handle this request.

What is the way MAC needs to be?

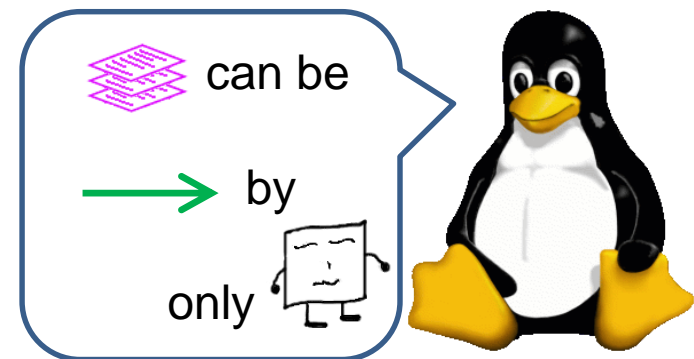
- ▶ **My conclusion:**
 - ▶ It's time to break dependence on domains (Domain Type Enforcement) and allowlisting.
 - ▶ Let's consider from scratch.

How can I take advantage of both approaches?

- ▶ TOMOYO which was made from the point of view of ****subjects**** and focused on enhancing functionality




- ▶ RWXfilter which was made from the point of view of ****objects**** and focused on enhancing usability







Why not to use Action as a key?



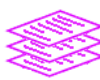
▶ Capability model + Access control list model

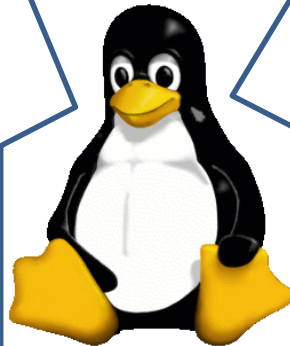
=> Action check list model


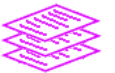
Check if  is requested.

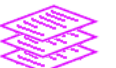
Check if  by  is requested.


Check if  on  is requested.

Check if  by  and on  is requested.



Grant or deny the request if by  and on  are true.

Grant or deny the request if on  is true.

Grant or deny the request if by  is true.

Grant or deny the request.

My proposed syntax

- ▶ `acl "Action" "Whether to check Action or not"`
 - `audit "Audit pattern specifier"`
 - `"Decision1" "Whether to use Decision1 or not"`
 - `"Decision2" "Whether to use Decision2 or not"`
 - `"Decision3" "Whether to use Decision3 or not"`
- ▶ Specify Action as a key, and enumerate conditions as needed.
 - ▶ Not using mandatory (positional) parameters
 - ▶ All parameters (including domains) are optional.
- ▶ Split into two phases: "whether to check or not" and "whether to grant/deny or not".
 - ▶ Not using profiles because there is no need to specify enabled/disabled.

Characteristic points of proposed syntax

- ▶ Supports both allowlisting approach and denylisting approach.
- ▶ Supports both the point of view of subjects and the point of view of objects, using actions as a key.
- ▶ Allows users to fully utilize TOMOYO's parameter validation capabilities.
- ▶ Allows users to apply single function restrictions like RWXfilter.
- ▶ Allows users to easily apply system wide restriction because action is the key.

=> Above topics are explained later using example policy.

How does policy file look like?

POLICY_VERSION=20120401

quota memory audit 16777216

quota memory query 1048576

quota audit[1] allowed=0 denied=1024 unmatched=1024

0 acl modify_policy

audit 1

1 deny task.uid!=0

1 deny task.euid!=0

100 allow task.exe="/usr/sbin/caitsith-loadpolicy"

100 allow task.exe="/usr/sbin/caitsith-queryd"

10000 deny

How does policy file look like?

```
POLICY_VERSION=20120401
quota memory audit 16777216
quota memory query 1048576
quota audit[1] allowed=0 denied=1024 unmatched=1024
```

Header part defines quota
and groups

```
0 acl modify_policy
  audit 1
  1 deny task.uid!=0
  1 deny task.euid!=0
  100 allow task.exe="/usr/sbin/caitsith-loadpolicy"
  100 allow task.exe="/usr/sbin/caitsith-queryd"
  10000 deny
```

Body part defines rules

How does policy file look like?

POLICY_VERSION=20120401

quota memory audit 16777216

quota memory query 1048576

quota audit[1] allowed=0 denied=10000 unmatched=1024

0 acl modify_policy

audit 1

1 deny task.uid!=0

1 deny task.euid!=0

100 allow task.exe="/usr/sbin/caitsith-loadpolicy"

100 allow task.exe="/usr/sbin/caitsith-queryd"

10000 deny

Version of policy syntax

Allow up to 16MB of kernel memory
for spooling audit logs

How does policy file look like?

POLICY_VERSION=20120401

quota memory audit 16777216

quota memory query 1048576

quota audit[1] allowed=0 denied=1024 unmatched=1024

0 acl modify_policy

audit 1

1 deny task.uid!=0

1 deny task.euid!=0

100 allow task.exe="/usr/sb

100 allow task.exe="/usr/sb

10000 deny

Apply audit log quota
defined in audit[1] line

Allow spooling up to 0 logs when
matched "allow" line, up to 1024 logs
when matched "deny" line, up to
1024 logs when did not match
"allow" line nor "deny" line

How does policy file look like?

POLICY_VERSION=20120401

Checking priority when there are multiple acl blocks with the same action

quota [1] allowed=0 denied=1024 unmatched=1024

0 acl modify_policy

audit 1

1 deny + .uid!=0

1 deny .euid!=0

100 ask.exe="/u

Additional conditions for checking whether to check this action or not.
Unconditionally checked if omitted

Name of action to check. In this example, changing policy configuration

sith-queryd"

How does policy file look like?

```
POLICY_VERSION=20120401
```

```
quota memory audit 16777216
```

```
quota memory query 1048576
```

```
0 acl memory policy unmatched=1024
```

Decision priority when there are multiple decision lines within this acl block

```
0 acl memory policy
```

```
audit
```

```
1 deny task.uid!=0
```

```
1 deny task.euid!=0
```

```
100 allow task.exe="/usr/sbin/caitsith-loadpolicy"
```

```
100 allow task.exe="/usr/sbin/caitsith-queryd"
```

```
10000 deny
```

How does policy file look like?

POLICY_VERSION=20120401

quota memory audit 16777216

quota memory query 1048576

quota audit[1] allowed=0 denied=1024 unmatched=1024

Decision is either "allow" or "deny"

0 acl memory_

audit 1

1 deny task.uid!=0

1 deny task.euid!=0

100 allow task.exe="/usr/sbin/caitsith-loadpolicy"

100 allow task.exe="/usr/sbin/caitsith-queryd"

10000 deny

How does policy file look like?

POLICY_VERSION=20120401

quota memory audit 16777216

quota memory query 1048576

quota audit[1] allowed=0 denied=1024 unmatched=1024

0 acl modify_policy

audit 1

1 deny task.uid!=0

1 deny task.euid!=0

100 allow task.exe="/usr/sbin/caitsith-loadpolicy"

100 allow task.exe="/usr/sbin/caitsith-queryd"

10000 deny _____

Additional conditions for deciding whether to apply the decision or not.
The decision is unconditionally applied if omitted

How does policy file look like?

```
POLICY_VERSION=20120  
quota memory audit 167772  
quota memory query 10485  
quota audit[1] allowed=0 de
```

```
0 acl modify_policy
```

```
audit 1
```

```
1 deny task.uid!=0
```

```
1 deny task.euid!=0
```

```
100 allow task.exe="/usr/sbin/caitsith-loadpolicy"
```

```
100 allow task.exe="/usr/sbin/caitsith-queryd"
```

```
10000 deny
```

This acl block defines below rules.

- (1) Deny changing policy configuration if current thread's user id or effective user id is not 0
- (2) Allow changing policy configuration if /proc/self/exe is either /usr/sbin/caitsith-loadpolicy or /usr/sbin/caitsith-queryd
- (3) Deny changing policy configuration otherwise

How does audit log look like?

- ▶ Trying to modify policy configuration by bash will be denied.

```
# echo '1000 acl modify_policy' > /proc/caitsith/policy
```

```
-bash: echo: write error: Operation not permitted
```

- ▶ And a denied log will be generated.

```
0 acl modify_policy
```

```
audit 1
```

```
1 deny task.uid!=0
```

```
1 deny task.euid!=0
```

```
100 allow task.exe="/usr/sbin/caitsith-loadpolicy"
```

```
100 allow task.exe="/usr/sbin/caitsith-queryd"
```

```
10000 deny
```

Doesn't match

Doesn't match

Doesn't match

Matches

Doesn't match

How does audit log look like?

- ▶ Below is a denied log generated by trying to modify policy configuration by bash.
 - ▶ #2012/07/11 14:06:21# global-pid=3584 result=denied priority=0 / modify_policy task.pid=3584 task.ppid=3582 task.uid=0 task.gid=0 task.euid=0 task.egid=0 task.suid=0 task.sgid=0 task.fsuid=0 task.fsgid=0 task.type!=execute_handler task.exe="/bin/bash" task.domain="/usr/sbin/sshd"



This log is readable from `/proc/caitsith/audit` and is saved by `caitsith-auditd` program

How does audit log look like?

- ▶ Below is a denied log generated by trying to modify policy configuration by bash.

- ▶ #2012/07/11 14:06:21# global-pid=3584 result=denied
priority=0 / modify_policy task.pid=3584 task.ppid=3582
task.uid=0 task.gid=0 task.euid=0 task.egid=0 task.suid=0
task.sgid=0 task.fsuid=0 task.fsgid=0
task.type!=execute_handler task.exe="/bin/bash"
task.domain="/usr/sbin/sshd"

This log was generated by "0
acl modify_policy" block

Result is one of "allowed" or
"denied" or "unmatched"

How does audit log look like?

- ▶ Below is a denied log generated by trying to modify policy configuration by bash.
 - ▶ #2012/07/11 14:06:21# global-pid=3584 result=denied priority=0 / modify_policy task.pid=3584 task.ppid=3582
task.uid=0 task.gid=0 task.euid=0 task.egid=0 task.suid=0
task.sgid=0 task.fsuid=0 task.fsgid=0
task.type!=execute_handler task.exe="/bin/bash"
task.domain="/usr/sbin/sshd"



These are variables within the access request.
These variables can be used as conditions as needed

How to update policy?

- ▶ Trying to modify policy configuration by caitsith-loadpolicy will be allowed.

```
# (echo '0 acl modify_policy'; echo '1 deny task.gid!=0') |  
/usr/sbin/caitsith-loadpolicy
```

- ▶ But an allowed log will not be generated.

```
0 acl modify_policy
```

```
audit 1
```

```
1 deny task.uid!=0
```

```
1 deny task.euid!=0
```

```
100 allow task.exe="/usr/sbin/caitsith-loadpolicy"
```

```
100 allow task.exe="/usr/sbin/caitsith-queryd"
```

```
10000 deny
```

Doesn't match

Doesn't match

Matches

How does user space daemon work?

POLICY_VERSION=20120401

quota memory audit 16777216

quota memory query 1048576

quota audit[1] allowed=0 denied=1024 unmatched=1024

0 acl modify_policy

audit 1

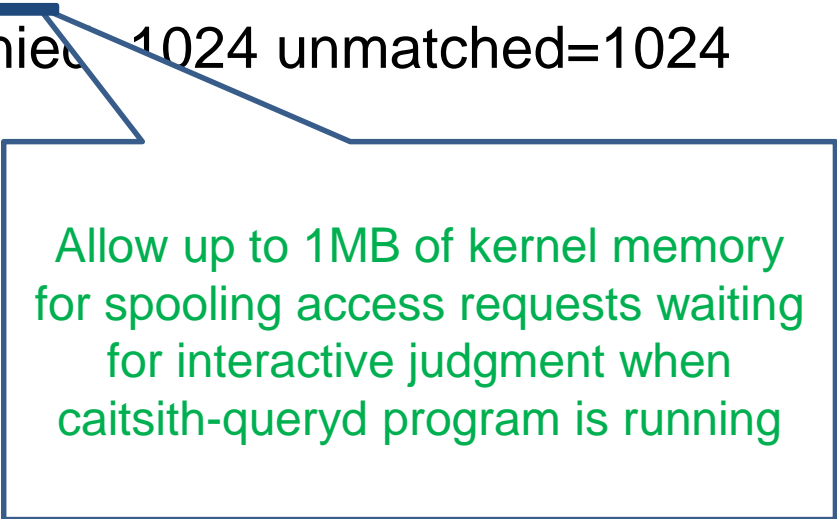
1 deny task.uid!=0

1 deny task.euid!=0

100 allow task.exe="/usr/sbin/caitsith-loadpolicy"

100 allow task.exe="/usr/sbin/caitsith-queryd"

10000 deny



Allow up to 1MB of kernel memory
for spooling access requests waiting
for interactive judgment when
caitsith-queryd program is running

How does user space daemon work?

```
POLICY_VERSION=20120401
```

```
quota memory audit 10777010
```

```
quota memory query 1
```

```
quota audit[1] allowed
```

```
0 acl modify_policy
```

```
audit 1
```

```
1 deny task.uid!=0
```

```
1 deny task.euid!=0
```

```
100 allow task.exe="/usr/sbin/caitsith-loadpolicy"
```

```
100 allow task.exe="/usr/sbin/caitsith-queryd"
```

```
10000 deny
```

A denied log is generated and then access request is spooled for interactive judgment. If caitsith-queryd allows, the permission check continues as if the access request did not match the "deny" line. The access request is denied otherwise

How does user space daemon work?

- ▶ Below is a query shown by caitsith-queryd program.
 - ▶ #2012/07/11 14:06:21# global-pid=3584 result=denied priority=0 / modify_policy task.pid=3584 task.ppid=3582 task.uid=0 task.gid=0 task.euid=0 task.egid=0 task.suid=0 task.sgid=0 task.fsuid=0 task.fsgid=0 task.type!=execute_handler task.exe="/bin/bash" task.domain="/usr/sbin/sshd"

Allow? ('Y'es/'N'o/'R'etry/'S'how policy/'A'dd to policy and retry):



Identical with audit log, except that a prompt line for manual decision is shown.

Characteristic points of proposed syntax

- ▶ Supports both allowlisting approach and denylisting approach.

```
1000 acl execute task.exe="/usr/sbin/httpd"
```

```
audit 1
```

```
100 allow path="/var/www/cgi-bin/counter.cgi"
```

```
200 deny
```

allowlisting approach ends with an unconditional "deny" line

```
2000 acl execute task.exe="/usr/sbin/httpd"
```

```
audit 1
```

```
100 deny path="/bin/sh"
```

```
200 allow
```

denylisting approach ends with an unconditional "allow" line

Characteristic points of proposed syntax

- ▶ Supports both the point of view of subjects and the point of view of objects, using actions as a key.

```
1000 acl execute task.exe="/usr/sbin/httpd"
```

```
audit 1
```

```
100 allow path="/usr/sbin/suexec"
```

```
200 deny
```

/usr/sbin/httpd can execute only /usr/sbin/suexec

```
2000 acl execute path="/usr/sbin/suexec"
```

```
audit 1
```

```
100 allow task.exe="/usr/sbin/httpd"
```

```
200 deny
```

/usr/sbin/suexec can be execute by only /usr/sbin/httpd

Characteristic points of proposed syntax

- ▶ Supports both the point of view of subjects and the point of view of objects, using actions as a key.

```
1000 acl inet_stream_listen task.exe="/usr/sbin/sshd"
```

```
audit 1
```

```
100 allow port=22
```

```
200 deny
```

/usr/sbin/sshd can listen to TCP sockets at only port 22

```
2000 acl inet_stream_listen port=22
```

```
audit 1
```

```
100 allow task.exe="/usr/sbin/sshd"
```

```
200 deny
```

TCP socket's port 22 can be listened by only /usr/sbin/sshd

Characteristic points of proposed syntax

- ▶ Allows users to fully utilize TOMOYO's parameter validation capabilities.

Check ioctl requests on /dev/kvm device

```
0 acl ioctl path.type=char path.dev_major=10
```

```
path.dev_minor=232
```

```
audit 1
```

Only /usr/libexec/qemu-kvm can issue ioctl requests on /dev/kvm device

```
100 deny task.exe!=" /usr/libexec/qemu-kvm"
```

```
200 allow
```

```
cmd=@PERMITTED_DEV_KVM_IOCTL_CMD_NUMBERS
```

```
300 deny
```

Only ioctl command numbers defined by "number_group PERMITTED_DEV_KVM_IOCTL_CMD_NUMBERS" lines in the header part of policy file are permitted

Characteristic points of proposed syntax

- ▶ Allows users to apply single function restrictions like RWXfilter.

Check TCP socket's connect requests

Only port numbers defined by "number_group PERMITTED_INET_CONNECT_PORTS" lines are permitted

```
1000 acl inet_stream_connect
```

```
audit 1
```

```
100 deny port!=@PERMITTED_INET_CONNECT_PORTS
```

```
100 allow
```

```
ip=@PERMITTED_INET_CONNECT_ADDRESSES
```

```
200 deny
```

Only IPv4/IPv6 addresses defined by "ip_group PERMITTED_INET_CONNECT_ADDRESSES" lines are permitted

Characteristic points of proposed syntax

- ▶ Allows users to easily apply system wide restriction because action is the key.

100 acl mount

audit 1

0 deny task.exe!=" /bin/mount"

1 allow target="/proc/" fstype="proc" flags=0x0

1 allow target="/sys/" fstype="sysfs" flags=0x0

1 allow target="/dev/pts/" fstype="devpts" flags=0x0

1 allow target="/dev/shm/" fstype="tmpfs" flags=0x0

1 allow target="/" fstype="--remount" flags=0x1

1 allow target="/" fstype="--remount" flags=0x400

2 deny

Only /bin/mount can issue mount requests

How to use CaitSith?

- ▶ Installation steps are almost same with those of TOMOYO 1.8.
 - ▶ Because CaitSith shares same kernel patches used by TOMOYO 1.8.
 - ▶ Applying kernel patches is easy because most of hooks are already embedded into LSM.
- ▶ Usage steps are
 - (1) Define acl blocks you want to check
 - (2) Edit decision lines in the blocks from audit logs
 - (3) Terminate the blocks with unconditional deny (or allow) line

Please try CaitSith.

▶ <https://caitsith.osdn.jp/>

C
h
a
r
a
c
t
e
r
i
s
t
i
c

a
c
t
i
o
n

i
n
s
p
e
c
t
i
o
n

t
o
o
l
.

S
e
e

i
f

t
h
i
s

h
e
l
p
s
.