TOMOYO Linux

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

http://tomoyo.sourceforge.jp/

Toshiharu Harada
Tetsuo Handa
NTT DATA CORPORATION

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

• 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](http://www.nttdata.co.jp/en/aboutus/a09.html) for more detail)

• TOMOYO Linux project:
  – 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).
- 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).
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”? 

application 

↓ 

system call interface 

↓ 

DAC (traditional access control) 

↓ 

system call procedures 

kernel
What is "hook"?

application

↓

system call interface

↓

DAC (traditional access control)

→ hook

MAC ↔ "policy"

system call procedures

kernel
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”.
  - \textit{if (condition) then Access is (granted/denied)}.
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
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
```

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

```
<kernel>
```
is the virtual bottom

```
is the current “domain”
```

mode path

mode: 1 --x
2 –w-
4 r—
6 rw-


**SELinux Policy**

- **bind.te**: allowing access
  - type named_t
  - type named_exec_t
  - init_daemon_domain(named_t,named_exec_t)
  - ...
  - kernel_read_kernel_syscalls(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_udp_sendrecv_all_ports(named_t)
  - corenet_tcp_sendrecv_all_ports(named_t)
  - corenet_non_psec_sendrecv(named_t)
  - corenet_tcp_bind_all_nodes(named_t)
  - corenet_udp_bind_all_nodes(named_t)
  - ...
  - 293
  - ...
  - 100 kinds of macros

- **bind.fc**: assigning label

```
/etc/nmc/*  -- gen_context(system_u:object_r:named_conf_t:s0)
/etc/nmc/key  -- gen_context(system_u:object_r:dnssec_t:s0)

/var/sbin/lwscd  -- gen_context(system_u:object_r:named_exec_t:s0)
/var/sbin/named  -- gen_context(system_u:object_r:named_exec_t:s0)
/var/sbin/named-chkconf  -- gen_context(system_u:object_r:named_chkconf_exec_t:s0)
/var/sbin/nmc  -- gen_context(system_u:object_r:nmc_exec_t:s0)
/var/log/named.*  -- gen_context(system_u:object_r:named_log_t:s0)

/var/run/nmc  -- 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)
```

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/** rw1,
    /var/run/named.pid rw1,
}
```

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”.

2007-04-18
• 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.
TOMOYO Keeps Track of Process
Invocation History

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

```
<kernel> /usr/sbin/sshd  /bin/bash  /usr/bin/man  /bin/sh  = current domain
```

1. `/bin/gunzip`
2. `/dev/null`
3. `/dev/tty`
4. `/etc/mtab`
5. `/usr/bin/bzip2`
6. `/usr/bin/gtbl`
7. `/usr/bin/less`
8. `/usr/bin/nroff`
9. `/var/cache/man/cat1/pstree.1.bz2`
10. `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”.

2007-04-18
Demo
Automatic Policy Generation

• How it works?
  – TOMOYO kernel keeps track of:
    • access requests (by original hooks).
    • process invocation history (something like `pstree` 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.

Hey boss, just got policy, please check.

Thank you.
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”.

2007-04-18

NTT DATA CORPORATION
Domain Transition

<kernel> /sbin/init /etc/rc.d/rc ・・・ foo bar

<kernel> /sbin/init /etc/rc.d/rc ・・・ foo

/etc/rc.d/rc

/etc/rc.d/rc.sysinit

<kernel> /sbin/init /etc/rc.d/rc

/foo

/bar

<kernel> /sbin/init /etc/rc.d/rc

<kernel> /sbin/init /etc/rc.d/rc.sysinit

<kernel> /sbin/init

<kernel>
Domains in SELinux

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

• **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 need “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.
You can proceed next page and configure TOMOYO Linux's settings on remote server.

Now, you can test connecting to TOMOYO Linux server.

- **Host**: 192.168.51.28
- **Port**: 22
- **User**: root
- **Password**: (empty)

Click the **Connection Test** button to test the connection.
<table>
<thead>
<tr>
<th>Profile Table</th>
<th>0 - Disabled</th>
<th>1 - Learning</th>
<th>2 - Permissive</th>
<th>3 - Enforcing</th>
<th>5 - Accept</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC FOR_FILE</td>
<td>Disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC FOR_NETWORK</td>
<td>Disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC FOR BINDPORT</td>
<td>Disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC FOR CONNECTPORT</td>
<td>Disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESTRICT_AUTOBIND</td>
<td>Off</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filesystem</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DENY CONCEAL_MOUNT</td>
<td>Disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESTRICT CHROOT</td>
<td>Disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESTRICT MOUNT</td>
<td>Disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESTRICTUnmount</td>
<td>Disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DENY PVOT ROOT</td>
<td>Disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESTRICT PVOT_ROOT</td>
<td>Disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trace READONLY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX ACCEPT FILES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX ACCEPT ENTRY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX GRANT LOG</td>
<td>1024</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX REJECT LOG</td>
<td>1024</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOMOYO_VERBOSE</td>
<td>Off</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Others</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMMENT</td>
<td>Disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC FOR ARQGO</td>
<td>Disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC FOR SIGNAL</td>
<td>Disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALLOE ENFORCE GRACE</td>
<td>Off</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customize</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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!