



TOMOYO® Linux:
*A **Practical** Method to*
*Understand and **Protect***
Your Own Linux Box

Nov. 29, 2007

Toshiharu Harada

<haradats@nttdata.co.jp>

Research and Development Headquarters

NTT DATA CORPORATION

Outline

- Looking Back At Linux Security
- What is TOMOYO Linux
- How TOMOYO Linux Compares to Others?

Incidents Occur

- Let's dig in it to see how it can happen:
 1. **shell code** ... *is caused by*
 2. **buffer overflow attack** ... *is caused by*
 3. **vulnerability** ... *is caused by*
 4. **human err** ... *THE END* (can't dig in further)
- So, no one can stop incidents.

What humans can do is

- Limiting the extent of damage.
- How?
 - Brightest invention of “**Mandatory Access Control**”
 - It has become available to even Open Source Software including Linux and other mainstream OSes.
- Problem still remains ...
 - Managing proper policies is not easy.



Why Managing Policies is So Difficult?

- Because
 - It's in the bottom layer (kernel), not in the human understandable layer.
 - Programmers have to understand about the complexities that are usually encapsulated by libraries and middleware.
 - The differences of manners between Linux kernel and Human understandings.
 - Human and Linux Boxes can live without policies.



Two Approaches Towards a Single Goal

- Goal
 - To obtain the appropriate policies.
- Approaches
 - “Catering” vs. DIY
 - “Catering” means:
 - Someone cooks and deliver dishes. Users (you!) just eat their dishes.
 - DIY means
 - cook by yourself and eat by yourself
 - In other words:
 - Professional vs. Amateur

Time to Introduce the Players

- “Professional” team:
 - **SELinux** by NSA
 - Users are suppose to apply professionally ready made policies.
- “Amateur” team:
 - **TOMOYO Linux**
 - automatic “policy learning mode” is available.
- Somewhere in-between:
 - **AppArmor** (formerly known as SubDomain)
- Promising rookie:
 - **Smack** (Simplified Mandatory Access Control Kernel)

At a Glance Comparison

- <http://tomoyo.sourceforge.jp/wiki-e/?WhatIs#comparison>
(*live* complicated table with useful links)

	SELinux	Smack	AppArmor	TOMOYO Linux
Label/Pathname	label		pathname	
Mainline Status	already	#1(Jul 14, 2007) v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 (Nov 8, 2007) now in -mm tree!	#1(Apr 19, 2006) #2 #3 #4(Oct 26, 2007)	#1(Jun 13, 2007) #2 #3 #4 #5(Nov. 17, 2007)
Overview				
Overview	implementation of the research project and architecture, Flask	fairly new attempt towards usable MAC for Linux	Novell had bought the company formerly known as Immunix and ported the technology to SUSE as AppArmor. open source version is also available	developed solely by NTT DATA and was open sourced in 2005
Developed by	NSA	Casey Schaufler	Novell (was)	NTT DATA CORPORATION
Supported by	(mainlined)	project	Mercenary (will be)	project
ISO image for Live CD	N/A	N/A	N/A	w/ Ubuntu 7.10

What Items are Important?

- In my humble view:
 - Whether you like “professional security” way of thinking or not
 - Your DIY spirit (or Your love for your Linux box)
 - Number of the Linux boxes you need to manage
 - Functional requirements (this is the easier part)
 - If you need “more”, probably SELinux is the best.
- Please be advised to “read” the policies **before** you make decisions. 😊
 - *If you don't like/ understand policies, you should not choose it. Using secure OS is managing its policies. (by ME)*



“Professional Policy”

- Quote from LKML ever lasting AppArmor’s thread
- SELinux expert, Kyle Moffet wrote:
 - *Average users are not supposed to be writing security policy. To be honest, even average-level system administrators should not be writing security policy. It's OK for such sysadmins to tweak existing policy to give access to additional web-docs or such, but only expert sysadmin/ developers or security professionals should be writing security policy. **It's just too damn easy to get completely wrong.***
 - <http://lkml.org/lkml/fancy/2007/5/28/359>
- Having a SELinux is a glory, but if you use it today, you will need some hustle. If you can bare it, SELinux should be the first secure Linux for you.

Outline

- Looking Back At Linux Security
- What is TOMOYO Linux
- How TOMOYO Linux Compares to Others?



Motivation

- Questions
 - Who knows best about your Linux box?
 - Who is responsible for your Linux box?
- I assume
 - It's YOU, isn't it?
- You might not be a professional security architect or a SELinux guru, but you can be an expert of YOUR own Linux box.
- So, we are developing a DIY tool for you. That is "TOMOYO Linux".

Let's Go Back to the Needs

- The title of this presentation is “TOMOYO Linux: A Practical Method to Understand and **Protect** Your Own Linux Box”.
- Why to **protect**? (protect from what?)
 - Malicious attacks.
 - Operations by mistake.
 - Your wife skimming your secret data.

Defining a Goal

- “Protect” is OK, but why “Understand” proceeds?
- Because you need to understand your Linux box to protect it.



Defining a Goal

- ... What am I supposed to **understand** about my Linux box? I know it's running 2.6.23 kernel and its Ubuntu 7.10. Isn't that enough?
- No.
- Example?
- *Can you tell how a gnome-terminal process is invoked and what a gnome-terminal process does?*

Defining a Goal

- You might say, “I’m totally not interested in such things. WHY DO I NEED TO KNOW THEM?” (calm down, please ...)
- You need to know them to tell your Linux box those accesses are needed. That’s the way security policy works.
 - I’m sorry, but this is the truth. You can never protect unless you understand what you want to protect. (There’s a professional security model that also exists, though)

Defining a Goal

- You might say,
 - “I want to protect my Linux box, but I don’t want to spend time to analyzing my Linux box and writing down policies myself”.

- Congratulations!
- *TOMOYO Linux is just for you.*

Let's see

- How the gnome-terminal process is kicked.
- What does the gnome-terminal process access.

- With TOMOYO Linux
 - Yes. You can.

- I will demonstrate now.



How gnome-terminal was “exec”ed

```
<kernel>
  /sbin/init
    /bin/sh
      /etc/init.d/rc
        /etc/init.d/gdm
          /sbin/start-stop-daemon
            /usr/sbin/gdm
              /etc/gdm/Xsession
                /usr/bin/ssh-agent
                  /usr/bin/x-session-manager
                    /usr/bin/gnome-panel
                      /usr/bin/gnome-terminal
```



What does *THIS* gnome-terminal access?

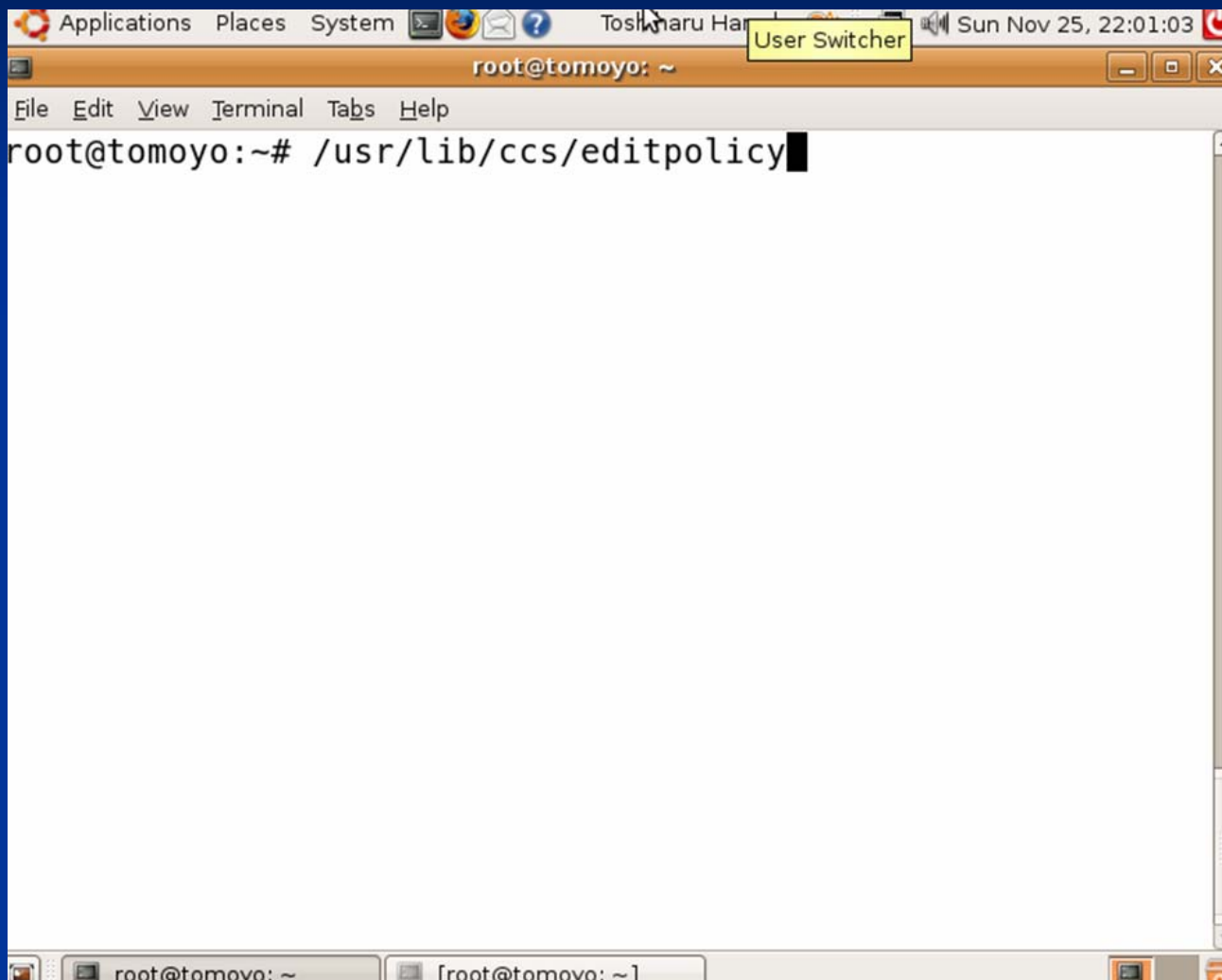
```
exec /bin/bash
exec /usr/lib/libvte9/gnome-pty-helper
read /dev/null
read /dev/urandom
read /etc/fonts/*
read /etc/gnome-vfs-2.0/modules/*
read /etc/nsswitch.conf
read /etc/passwd
read /etc/sound/events/gtk-events-2.soundlist
read /home/toshiharu/.config/user-dirs.dirs
read /home/toshiharu/.gtk-bookmarks
read /home/toshiharu/.ICEauthority
read /home/toshiharu/.Xauthority
read /tmp/gconfd-toshiharu/lock/ior
read /tmp/orbit-toshiharu/bonobo-activation-server-ior
read /usr/lib/gconv/*
read /usr/lib/gnome-vfs-2.0/modules/libfile.so
read /usr/lib/gtk-2.0/*
read /usr/lib/libglade/*
read /usr/lib/pango/1.6.0/module-files.d/libpango1.0-0.modules
read /usr/lib/pango/1.6.0/modules/pango-basic-fc.so
read /usr/share/fonts/*
read /usr/share/gnome-terminal/glade/gnome-terminal.glade2
read /usr/share/icons/*
read /usr/share/mime/*
read /usr/share/pixmaps/gnome-terminal.png
read /usr/share/themes/*
read /usr/share/vte/termcap/xterm
read /usr/share/X11/locale/*
read /var/cache/fontconfig/*
read&write /dev/pts/\$
read&write /tmp/orbit-toshiharu/bonobo-activation-register.lock
```



How Did I manage?

- Just copied and pasted the output of TOMOYO Linux policy editor.
- TOMOYO Linux policy editor
 - Displays the domains (domain transition tree)
 - Displays the result of access occurred for each domain
- Want to see it?

How Did I Get?





So what?

- With TOMOYO Linux and **without any preparations and hustle**
 - you can see how the processes are generated and what they do (access).
 - you can distinguish processes by their call chains, not by the name of the program.
 - if you know the correct “call chains”, then you can detect and exclude incorrect accesses.
- That's what title of this presentation means, “**Understand** and **Protect**”

1-2-3 You Are All Set

- Invoke policy editor program
 - 1) Choose the **domain** you want to protect
 - 2) Enter “s” key to change the mode for the selected **domain**
 - 3) Input the profile number you choose
- “Profile”
 - /etc/ccs/profile.conf (text file)
 - You can define the MAC functions as you need

Where is the profile #?

```

<<< Domain Transition Editor >>>      1543 domains  '?'
for help
<kernel>
0: 1 <kernel>
1: 1 /sbin/init
2: 1 /bin/sh
3: 1 /bin/grep
4: 1 /etc/init.d/rc
5: 1 /bin/grep
6: 1 /bin/sed
7: 1 /etc/init.d/acpi-support
8: 1 /bin/sed
9: 1 /etc/acpi/power.sh
10: 1 /sbin/on_ac_power
11: 1 /bin/grep
12: 1 /sbin/acpi_avail
13: 1 /sbin/usplash_write
14: 1 /usr/bin/expr
15: 1 /usr/bin/tput
16: 1 /usr/sbin/dmidecode

```

Let's Restrict a Shell

```

<<< Domain Transition Editor >>>      1543 domains  '?'
for help
er /usr/bin/gnome-panel /usr/bin/gnome-terminal /bin/bash
246:  1
247:  1
248:  1
249:  1
250:  1

251:  1
252:  1
253:  1
254:  1
255:  1
256:  1

257:  1
258:  1
259:  1
Enter profile number> 3

```

Let's Restrict a Shell

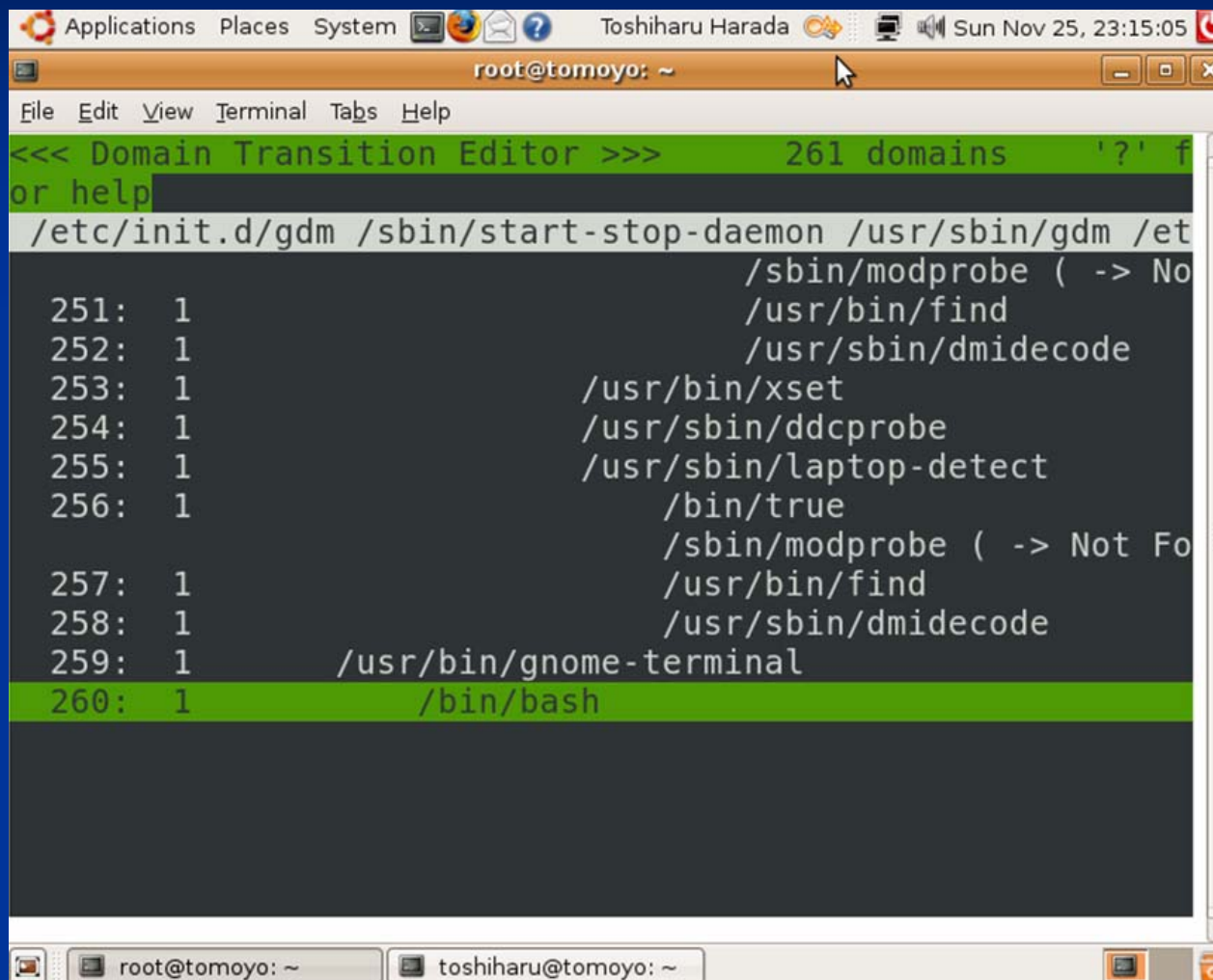
```

<<< Domain Transition Editor >>>      1543 domains      '?'
for help
<kernel> /sbin/init /bin/sh /etc/init.d/rc /etc/init.d/gdm
246: 1
247: 1
248: 1
249: 1
250: 1

251: 1
252: 1
253: 1
254: 1
255: 1
256: 1

257: 1
258: 1
259: 1
260: 3
  
```

See it again?



```

Applications Places System Toshiharu Harada Sun Nov 25, 23:15:05
root@tomoyo: ~
File Edit View Terminal Tabs Help
<<< Domain Transition Editor >>> 261 domains '?' f
or help
/etc/init.d/gdm /sbin/start-stop-daemon /usr/sbin/gdm /et
/sbin/modprobe ( -> No
251: 1 /usr/bin/find
252: 1 /usr/sbin/dmidecode
253: 1 /usr/bin/xset
254: 1 /usr/sbin/ddcprobe
255: 1 /usr/sbin/laptop-detect
256: 1 /bin/true
/sbin/modprobe ( -> Not Fo
257: 1 /usr/bin/find
258: 1 /usr/sbin/dmidecode
259: 1 /usr/bin/gnome-terminal
260: 1 /bin/bash
root@tomoyo: ~ toshiharu@tomoyo: ~

```

Outline

- Looking Back At Linux Security
- What is TOMOYO Linux
- How TOMOYO Linux Compares to Others?

Comparison with - SELinux

■ SELinux Overview

- “in tree” security enhancement
- Fine grained yet flexible MAC engine with full functionalities of Multi-Level Security, Multi-Category Security and Role Based Access Control.
- Based on the concept of “Security should be designed by professionals”. -> “reference policy”
- Well designed and supported by the wizards.

Comparison with - SELinux

- Should be ideal solution for Linux users *if*:
 - reference policy definition is finished.
 - administrators are freed from “label” management tasks.
- “Per domain permissive mode” is a missing piece. (Enforcing/Permissive mode is a system global attribute)

Comparison with - AppArmor

■ AppArmor Overview

- formerly known as SubDomain.
- same “pathname based” MAC (we are brothers)
- “domain” is per program while TOMOYO Linux domain is “process invocation tree”.
- aims to confine specified “programs” and is not intended to protect the whole system.



SELinux, AppArmor, TOMOYO Linux

- All do MAC per “domain”
- “domain” differs significantly:
 - **SELinux**
 - Domains are pre-defined in the policy
 - No hierarchy for domains. Domains are flat
 - **AppArmor** (“profile”)
 - Domains correspond to programs, such as Apache
 - Domains are pre-defined in the policy
 - No hierarchy for domain.
 - **TOMOYO Linux**
 - Domains are automatically defined and managed by the kernel
 - Domain is “process invocation history (or call chain)”



With TOMOYO Linux

- /bin/sh with different process invocation history are treated totally different domain
- It's done by the TOMOYO Linux kernel, so you don't have to define in advance
- Domain name is literally its process invocation history (no learning is needed)



More Information?

- <http://www.elinux.org/TomoyoLinux>
- <http://tomoyo.sourceforge.jp/index.html.en>
- <http://tomoyo.sourceforge.jp/wiki-e/>