



***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



***TOMOYO® Linux:**  
あなたのLinuxシステムを**理解し**  
**護る**ための**実践的**な方法*

Nov. 29, 2007

原田 季栄

<haradats@nttdata.co.jp>

技術開発本部  
株式会社NTTデータ

# Outline

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

# 概要

- Linuxのセキュリティを振り返る
- TOMOYO Linuxとは
- TOMOYO Linuxと他のLinuxとの比較

## 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.

## 事件は起こるもの

- なぜ発生するのかというと:
  1. **シェルコード** ... *が実行されてしまうのは*
  2. **バッファオーバーフロー攻撃** ... *が可能であるからであり、そこには*
  3. **脆弱性が** ... *存在するが、その発端は*
  4. **人的エラー** ... \*お手上げ\* (これ以上は無理)
- だから、事件が起こらないようにするのは不可能

# 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.

# 人間にできることと言えば

- ダメージを軽減すること
- どうやって?
  - 強制アクセス制御 ( **Mandatory Access Control** ) という発明
  - Linuxを含め多くのオープンソースOSでも利用可能になってきている
- でも、問題が残ってる...
  - ポリシーの管理は簡単ではない

## 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.

## どうしてポリシーの管理は面倒なの？

### ■ なぜなら

- 人間側から見るのではなく、最下層であるカーネル側から見なければいけないから
- 通常はライブラリやミドルウェアによって隠蔽されている詳細についてプログラム開発者側が意識しなければいけないから
- Linuxのカーネルが理解する流儀と人間が理解する流儀は異なるから
- 人間もLinuxシステムも強制アクセス制御のためのポリシーを持たずとも存在できるから

## 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

## 唯一の目標への二通りのアプローチ

- 目標
  - 適切なポリシーを入手すること
- アプローチ
  - 「出前」対「自炊」
  - 「出前」とは
    - 誰かが料理して届けてくれる。ユーザ(あなた)はそれを食べるだけ。
  - 「自炊」とは
    - 自分で料理して自分で食べる
  - 言い換えると
    - 「玄人」対「素人」

## 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)

## 登場人物を紹介します

- 「玄人」チーム
  - NSA (米国家安全保障局)による**SELinux**
  - ユーザはプロが作ったポリシーを使うだけ
- 「素人」チーム
  - **TOMOYO Linux**
  - ユーザ自身が「自動学習機能」を使ってポリシーを作る
- その中間
  - **AppArmor** (以前はSubDomainと呼ばれていた)
- 前途有望なルーキー
  - **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, <a href="#">Flask</a>	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

# ちょっと比べてみたい人は

- <http://tomoyo.sourceforge.jp/wiki-e/?WhatIs#comparison>  
(役に立つリンク付き／現在も更新中)

	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, <a href="#">Flask</a>	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)*

## 何が重要なもの？

- 私見ですが
  - 「プロフェッショナルによるセキュリティ」という考え方が好きかどうか
  - 自分でやりたいと思う心(あるいはあなたのLinuxシステムへの愛)
  - 管理しなければならないLinuxシステムの数
  - 必要とする機能(これは簡単)
    - もし、あれもこれも必要ならSELinuxがベストでしょう
- **でも、判断する前に、ポリシーを「読んで」ね☺**
  - **もし、ポリシーが嫌いだったり内容を理解したくないのなら、やめておくべきです。セキュアOSを使うということは、ポリシーを管理することなのですから。**

## “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.

## 「プロが作ったポリシー」

- 延々と続くLKMLでのAppArmorに関する議論より
- SELinuxの専門家であるKyle Moffet氏曰く
  - 平均的なユーザはセキュリティポリシーを書くべきではない。正直言って、平均的なシステム管理者でさえもセキュリティポリシーを書くべきではない。Webコンテンツへのアクセスを追加する程度の既存のポリシーに対する軽微な修正なら構わないが、それでもプロのシステム管理者／開発者やセキュリティのプロだけがセキュリティポリシーを書くべきである。セキュリティポリシーをぶち壊すのはとても簡単なんだ。
  - <http://lkml.org/lkml/fancy/2007/5/28/359>
- SELinuxが存在することは誇れることですが、もし今日使いたいならば「根気」が必要です。その根気があれば、SELinuxはあなたのLinuxシステムへの第一候補となるでしょう。

# Outline

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

# 概要

- Linuxのセキュリティを振り返る
- TOMOYO Linuxとは
- TOMOYO Linuxと他のLinuxとの比較

# 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".

# 開発の動機

- 疑問
  - 誰が自分のLinuxシステムのことを最もよく知っている？
  - 誰が自分のLinuxシステムに責任を持っている？
- 私が思うに
  - それは「あなた」でしょ？
- あなたはセキュリティのプロではないかもしれないし、SELinuxの伝道師でもないかもしれない。でも、あなたのLinuxシステムに関しては専門家になれるはず。
- だから、我々は自分で何とかするためのツールである TOMOYO Linuxを開発しています。

## 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.

## 本筋に戻ります

- このプレゼンテーションのタイトルは  
“TOMOYO Linux:あなたのLinuxシステムを  
理解し**護る**ための実践的な方法”です。
- **何を護る？何から護る？**
  - 悪意ある攻撃から
  - 操作ミスから
  - あなたが秘密にしたい情報を盗み見する奥さん  
から

## Defining a Goal

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

## 目標の設定

- “護る”のは良いとして、何故“理解”が先にあるの？
- 理解しなければ護ることはできないからね。

## 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?*

## 目標の設定

- ... 私のLinuxシステムの何について**理解**すればいいの？カーネル2.6.23でUbuntu 7.10を動かしていることを知ってるけど、それでは駄目ですか？
- 駄目です。
- 例えばどんなことを知っていれば良い？
- *gnome-terminal* がどのように起動され、どのような処理をするか言えますか？

## 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)

## 目標の設定

- “そんなことに興味ないね。どうしてそんなことを知っていないといけないの?” (お静かに願います...)
- あなたのLinuxシステムにどのようなアクセスが必要なのかを伝える必要があります。それが、セキュリティポリシーが機能するために必要なのです。
  - 不便なことですが事実なのです。あなたが護るべきものが何であるかを理解しない限り、あなたはそれらを護ることができません。(プロが考えたセキュリティモデルだけがあっても駄目なのです。)



## 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.*

## 目標の設定

- きっと、こう言うでしょう
  - “私のLinuxシステムは護りたい。でも、そのために分析してポリシーを書く時間を費やしたくない。”
  
- おめでとう！
- *TOMOYO Linuxはそんなあなたのためのものです。*

## 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.

## では始めます

- どのようにgnome-terminalが起動されたか
- gnome-terminalはどんな処理をしたか
  
- TOMOYO Linuxであれば
  - それを観ることができます
  
- デモを始めます

## 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
```

## gnome-terminalはどのように起動される？

```
<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
    
```

## \*この\* gnome-terminalは何をしている？

```

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
    
```

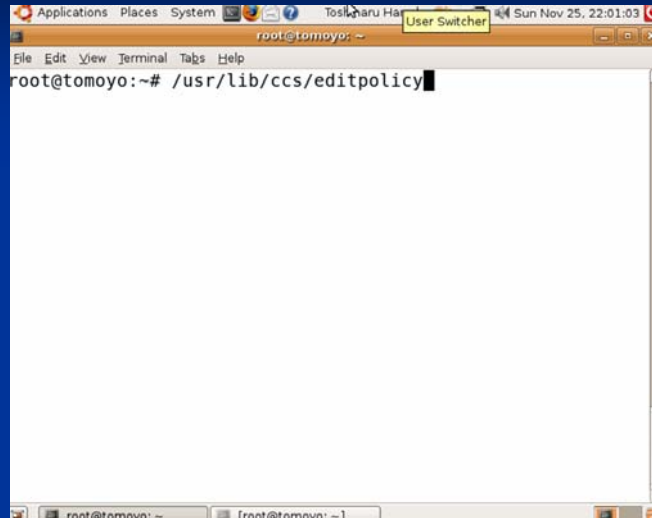
## How Did I manage?

- 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?

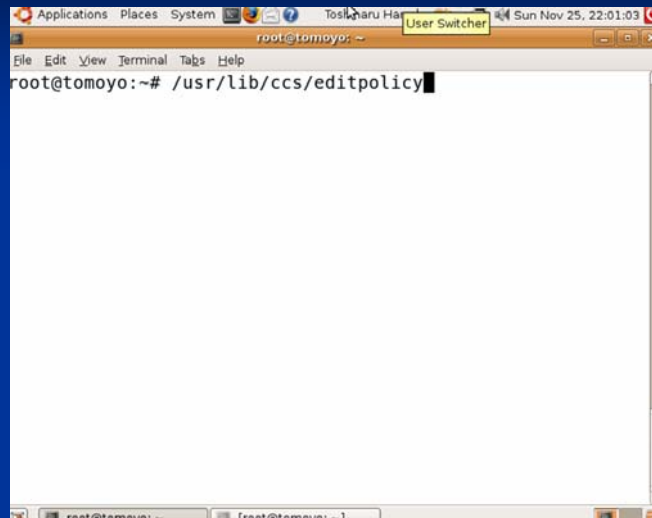
## どうやって知った？

- ポリシーエディタの表示をコピー&ペーストしただけだよ
- TOMOYO Linuxのポリシーエディタは
  - ドメイン遷移を木構造で表示してくれるよ
  - 各ドメインで発生したアクセス要求を表示してくれるよ
- 観たいでしょ？

# How Did I Get?



# どうやって知った?



# 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**”

# だからどうした？

- TOMOYO Linuxなら**準備も根気も要らないよ**
  - プロセスがどのように生成されて何をするか(どのような資源にアクセスするか)を教えてくれる世
  - プログラム名ではなくプログラムの実行履歴を用いることでプロセスを詳細に識別できるよ
  - 適切な呼び出し履歴を知ること、不適切なアクセス要求を検出して阻止することができるよ
- それがこのプレゼンテーションのタイトルである“**理解し護る**”という意味

## 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

## 1! 2! 3! はい、できあがり!

- ポリシーエディタを実行しましょう
  - 1) 保護したい**ドメイン**を選択
  - 2) その**ドメイン**のモードを切り替えるために“s” キーを押す
  - 3) プロファイル番号を入力
- プロファイル番号とは
  - /etc/ccs/profile.conf (テキストファイル)
  - 使いたい機能だけを選択してね



# 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
    
```

# プロフィール番号はどこ？

```

<<< 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
```

# シェルを制限してみよう

```
<<< 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
    
```

# シェルを制限してみよう

```

<<< 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?

# もう一度観たい？

# Outline

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

# 概要

- Linuxのセキュリティを振り返る
- TOMOYO Linuxとは
- TOMOYO Linuxと他のLinuxとの比較

## 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.

## SELinuxとの比較

- SELinuxの概要
  - メインラインに取り込まれているセキュリティ強化の実装
  - MLS/MCS/RBACを含むきめこまやかで柔軟なMAC
  - 「セキュリティのことはプロによって設計されたものを使うべし」という考え方→リファレンスポリシー
  - 
  - 魔術師によって設計されサポートされている

## 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)

## SELinuxとの比較

- 「もし」、以下の問題が解決されたらLinuxユーザにとって理想的な解決策
  - リファレンスポリシーが完成する
  - 管理者が適切なラベルを維持することの負担から解放される
- ドメイン単位の制御モード切替が無い(強制／許容の切り替えはシステム全体でしか行えない)

## 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.

## AppArmorとの比較

- AppArmorの概要
  - 昔はSubDomainと呼ばれていた
  - TOMOYOと同様にパス名ベースのMAC (だから我々は兄弟みたいな存在)
  - ドメインはプログラム単位のフラットな構造 (TOMOYOでは履歴を用いた木構造)
  - 特定のプログラムだけを保護するように設計されており、システム全体を保護することは想定していない



## 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)”

## SELinux, AppArmor, TOMOYO Linux

- いずれもドメインを単位とするMAC
- ドメインの概念が大きく異なる:
  - **SELinux**
    - ドメインはポリシーを用いて事前に定義されている
    - 階層構造を持たないためフラットである
  - **AppArmor** (“profile”)
    - ドメインは「Apache」のようにプログラム単位で作成する
    - ドメインはポリシーを用いて事前に定義されている
    - 階層構造を持たない
  - **TOMOYO Linux**
    - ドメインはカーネルにより自動的に定義され管理される
    - ドメインとはプロセスの実行履歴 (あるいは呼び出しの連鎖) である

## Dilemma of SELinux domain

- How to distinguish?
  - /bin/sh “exec”ed from sshd  
(probably administrator logged in from remote)
  - /bin/sh “exec”ed from httpd  
(possibly result of a shell code cracking)
- You can't because
  - /bin/sh and other objects can have \*ONLY\* one label at the same time
  - There's no concept of time frame in Labeled Security.

## SELinuxにおけるドメインのジレンマ

- この2つをどうやって区別する？
  - 「sshdから起動された/bin/sh」  
多分管理者がリモートからログインしているshell
  - 「httpdから起動された/bin/sh」  
もしかしたらシェルコードで実行されたshell
- 実は区別できません
  - /bin/shなどのオブジェクトは一度に”1個”のラベルしか持つことができない
  - ラベルベースのセキュリティに於いては時間軸の概念が無い

## 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)

## TOMOYO Linuxを使うと

- 異なる実行履歴を持つ /bin/sh は全く異なるドメインとして扱われる
- TOMOYO Linuxカーネルが自動的に扱ってくれるので、事前に定義する必要が無い
- ドメイン名はプロセスの実行履歴そのものであり、学習しないでも理解できる

## More Information?

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

## もっと情報が必要？

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