

# Mengawasi Sistem Linux dengan Syscalltracker

Kadang kita ingin mengawasi sistem Linux kita secara detail. Pasang *log* di sana-sini, pasang *intrusion detection*, *tool integrity checking*, dan sebagainya. Tapi kadang terasa kurang memadai karena programnya bisa diakali, diutak-atik user, bahkan di-*disable*. Terus bagaimana? Ada pengawas sistem di level kernel dengan tingkat keandalan tinggi.

Suatu hari penulis sedang “nongkrong” di milis *user openMosix*. Kebetulan bahasannya tentang seorang user yang menanyakan kejanggalan sistem akibat *patch* kernel. Lalu user lain di milis itu mencoba membantu merekonstruksi permasalahan dengan bantuan tool analisis bernama Syscalltracker. Saya langsung bertanya-tanya, “*Tool apaan nih?*” Bisa dibuat apa *aja?*” Saya sendiri mendengar nama serem seperti ini langsung membayangkan mekanisme yang “njelimet” serta syntax yang aneh-aneh.

Syscalltracker (<http://syscalltrack.sourceforge.net>) adalah suatu program, persisnya modul kernel, untuk membantu melacak *system call* yang terjadi dalam mekanisme internal kernel Linux. Gunanya apa? Secara umum bisa kita gunakan untuk melacak penyebab keganjilan pada sistem yang sukar dipecahkan program biasa. Contoh mudahnya begini. Misal Anda memiliki suatu file konfigurasi di direktori / etc. Nama file ini misal *abc.conf*. Anda sudah mengeset isi file ini untuk setting suatu program yang baru Anda instalasi. Sedang enak-enaknya program tersebut berjalan, isi file konfigurasi ini tiba-tiba hilang. Siapa yang dicurigai? Program ini sendiri? Script pembersih file bawaan distro? Atau jangan-jangan ada *exploit*? Buka log kiri-kanan, jalankan ‘ps’, intip dengan ‘who’, sepertinya si Linux normal saja. Buat file konfigurasi lagi, ternyata hilang lagi. *Kesel kan?*

Jalan keluar dari permasalahan seperti di atas terletak pada fakta, bahwa semua aksi atau perintah pada sistem pasti menjalankan satu atau lebih prosedur internal sistem atau yang lebih dikenal dengan *system call*. Anda menghapus file? Berarti Anda memanggil *system call unlink*. Anda

menjalankan program shell script? Berarti Anda memanggil *system call exec()* atau *execve()*. Jadi hampir semua aksi Anda diterjemahkan menjadi *system call*. Inilah ide dasar mengapa pelacakan berbasis *system call* bisa menjadi senjata ampuh pelacakan sistem.

Tertarik? Anda ingin mencoba? Kita langsung percobaan saja. Penulis di sini menggunakan distribusi Red Hat 7.3. Arahkan browser ke <http://syscalltrack.sourceforge.net> dan download tarball-nya dari bagian “Download”. Penulis menggunakan paket *syscalltrack-0.82.tar.gz* yang berukuran kurang lebih 500 kilobyte. Uraikan paket ini misal di direktori */usr/src*.

```
# tar zxvf syscalltrack-0.82.tar.gz -C /usr/src
```

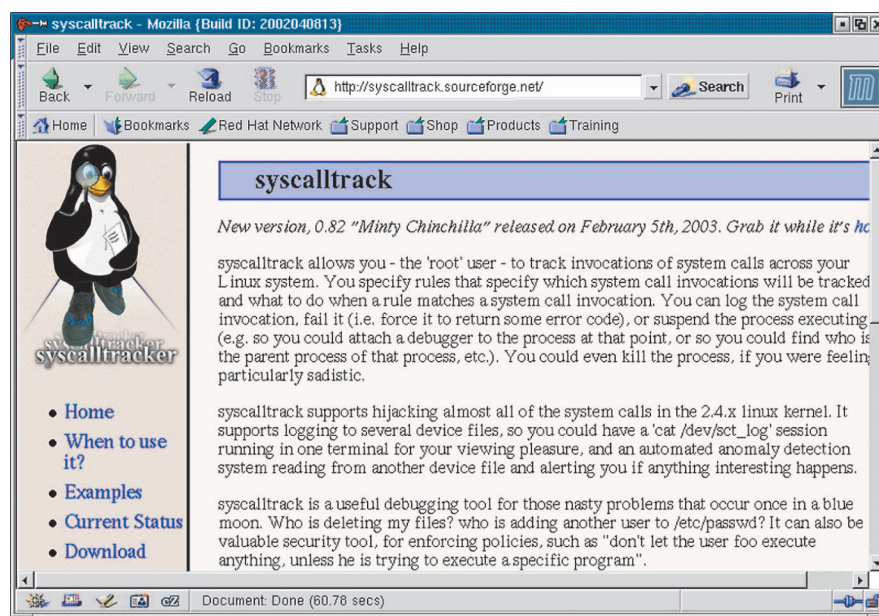
Berikutnya cek apakah Anda sudah memiliki *source code* kernel Linux di */usr/src*.

```
# ls -al /usr/src/linux-2.4.*
```

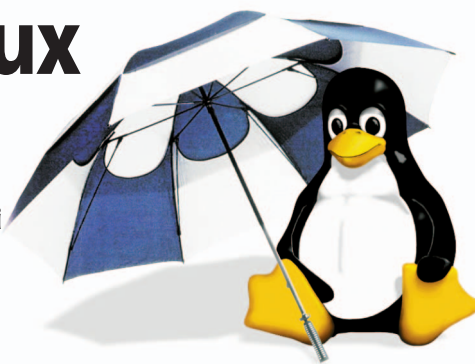
Jika belum ada isi *source kernel*, ambil CD Linux Red Hat, lalu instalasi dengan perintah berikut ini:

```
# rpm --replacepkgs -Uvh /path/to/your/RPM/kernel-source-2.4.18-3.i386.rpm
```

Perlu diperhatikan, *syscalltracker* harus di-*compile* berdasarkan *source kernel* yang sedang dijalankan pada sistem Linux Anda. Contohnya, jika Anda memakai kernel standar bawaan Red Hat Linux 7.3, maka Anda harus men-*compile* dengan *source code* dari kernel Red Hat Linux 7.3. Untuk itulah kita instalasi RPM dari *source code* kernel seperti di atas. Atau jika Anda menggunakan *source code* kernel bukan bawaan distro, maka nantinya arahkan kompilasi *syscalltracker* ke *source kernel* tersebut.



➤ [Homepage Syscalltracker.](http://syscalltrack.sourceforge.net)



```

root@localhost:~ - Shell - Konsole
Session Edit View Settings Help

[root@localhost root]# sct_load
sct_load: syscalltrack 0.82
MODPATH = /lib/modules/2.4.18-3/syscalltrack-0.82
ctrl_major=254
log_major=253
modules loaded succesfully, have a nice syscalltracking!
[root@localhost root]#

```

#### Module Syscalltracker di load.

Catatan lain, pastikan Anda menggunakan versi compiler (gcc) yang sama dengan yang digunakan untuk meng-compile kernel Anda. Jadi jika Anda menggunakan gcc-2.95 untuk kompilasi kernel, gunakan versi 2.95 juga untuk kompilasi syscalltracker. Untuk contoh dalam tulisan ini, Anda cukup menggunakan gcc bawaan Red Hat Linux karena diasumsikan Anda menggunakan kernel standar bawaan Red Hat.

Selain membutuhkan source kernel, untuk mempermudah kompilasi syscalltracker, Anda juga butuh file konfigurasi kernel dari distribusi Red Hat ini. Coba cek isi direktori /boot.

```
# ls -al /boot/config*
```

Jika ada output semisal config-2.4.18-3, maka Anda tinggal *copy* file ini ke /usr/src/linux-2.4. Ubah namanya menjadi '.config'.

```
# cp /boot/config-2.4.18-3 /usr/src/linux-2.4/.config
```

Jika file ini hilang atau tidak ditemukan, cara termudah adalah instalasi ulang paket kernel.

```
# rpm -Uvh /path/to/your/RPM/kernel-2.4.18-3.i386.rpm
```

Sesuaikan paket dengan arsitektur prosesor Anda. Jika Anda memakai sistem multiprocessor, pakai tipe '-SMP'. Jika memerlukan PCMCIA, pilih nama kernel yang mengandung kata '-PCMCIA'. Namun secara umum pilihan 'i386' sudah cukup

aman dalam kebanyakan kasus. Setelah paket kernel terinstalasi, copy file konfigurasi seperti cara di atas.

Atau Anda ingin file '.config' hasil konfigurasi manual? Caranya mudah. Pertama pindah ke direktori source code kernel Linux Anda, lalu jalankan mrproper dan menuconfig.

```
# cd /usr/src/linux-2.4.18-3
```

```
# make mrproper
```

```
# make menuconfig
```

Lakukan setting pada kernel, lalu "Exit". Otomatis di direktori source kernel Anda akan tercipta file '.config'. Sekarang kita langsung compile dan instalasi syscalltracker.

```
# cd /usr/src/syscalltrack-0.82
```

```
# ./configure(atau ./configure --with-
```

```
linux = /path/ke/direktori/source/kernel/anda)
```

```
# make && sudo make install
```

Setelah kompilasi sukses, Anda akan menemui dua module berikut:

```
/lib/modules/2.4.18-3/syscalltrack-0.82/sct_rules.o
```

```
/lib/modules/2.4.18-3/syscalltrack-0.82/sct_hijack.o
```

Ini adalah dua module utama yang akan menangani pelacakan sistem Anda. Berikutnya Anda perlu me-load dua module ini ke dalam kernel. Sudah disediakan shell script oleh pembuat syscalltracker untuk hal ini.

```
# sct_load
```

Lalu *cross check* apakah dua modul ini sudah dimuat, dengan perintah lsmod. Anda mestinya akan menemukan keluaran yang mengandung baris berikut.

```
# lsmod
```

Module	Size	Used by	Not tainted
sct_rules	257788	2	
sct_hijack	110176	1	[sct_rules]

Selamat! Anda sukses me-load dan menjalankan syscalltracker. Tapi ini masih belum berfungsi maksimal. Anda perlu menulis rule yang akan dimuat oleh syscalltracker dan dijalankannya. Penulis akan memberi contoh satu rule sederhana.

```
rule
```

```
{
```

```

root@localhost:/home/mulyadi/script-syscalltracker - Shell - Konsole
Session Edit View Settings Help

[root@localhost script-syscalltracker]# sct_config upload ./coba.conf | more
handling rule:
rule: cegah_delete (1)
  when: before
  syscall: 10
  action: type = FAIL pid = -1
handling rule:
rule: unlink_rule1 (2)
  when: before
  syscall: 10
  action: type = LOG log_format "%comm : %params delete by %uid --> %suid\n" priority 4
handling rule:
rule: connect_rule1 (3)
  when: after
  syscall: 196710
  action: type = LOG log_format "%comm : %params try socket connect by %uid --> %suid\n" priority 4
handling rule:
rule: fork_rule1 (4)
  when: after
  syscall: 2
  action: type = LOG log_format "%comm : %params fork by %uid --> %suid\n" priority 4

```

#### Syscalltracker me-load rule yang telah didefinisikan.

```

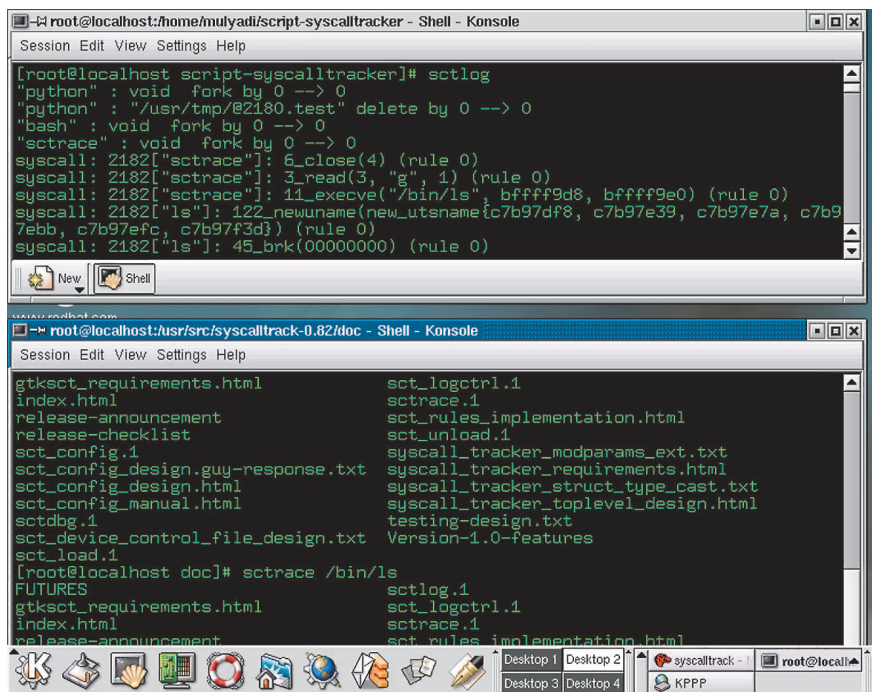
syscall_name = unlink
rule_name = unlink_rule1
action
{
    type = LOG
    log_format {%comm : %params delete
by %euid -> %suid}
}
when = before
}

```

Tiap rule pada syscalltracker dimulai dengan *reserved word* "rule". Lalu kurung kurawal "{". Berikutnya Anda set, pada system call yang mana syscalltracker akan men-intercept pada variable **syscall\_name** = . Ada banyak system call yang bisa Anda awasi. Untuk daftarnya coba lihat isi file /usr/local/lib/syscalltrack-0.82/syscalls.dat-2.4.18-3. Makna tiap system call kadang ada yang mudah ditebak dari namanya, ada juga yang agak sulit. Untuk contoh ini, saya ambil misal system call *unlink()*. System call ini pasti dijalankan setiap kali ada proses penghapusan file. Saya pikir cukup menarik untuk percobaan awal, jadi idenya kita ingin tahu sistem Linux kita menghapus file apa saja saat sedang berjalan.

Pada variable **rule\_name**, Anda tulis nama untuk rule ini. Anda bisa secara bebas mengisinya, misal saja Anda namakan **unlink\_rule1**.

Lalu pada section **action**, Anda tulis tindakan yang akan dilakukan. Syscalltraker mendukung beberapa action, tapi di sini kita gunakan tipe **LOG**. Action ini akan



➤ Satu Konsole menjalankan "sctrace /bin/lis", Konsole lain melihat isi log.

menulis hasil pelacakan ke device **/dev/sct\_log**. Sedang direncanakan untuk bisa melakukan intercept syscall lalu melakukan rewriting. Jika rencana ini bisa diimplementasikan, Anda bakal bisa memainkan sistem sesuka Anda.

Untuk action **LOG**, Anda perlu mendefinisikan format output. Ada beberapa macro yang bisa membantu Anda mendapatkan output yang diinginkan, yaitu:

- **ruleid**: nomor dari rule yang sesuai syscall yang diintercept.
- **sid**: nomor dari syscall yang diintercept.

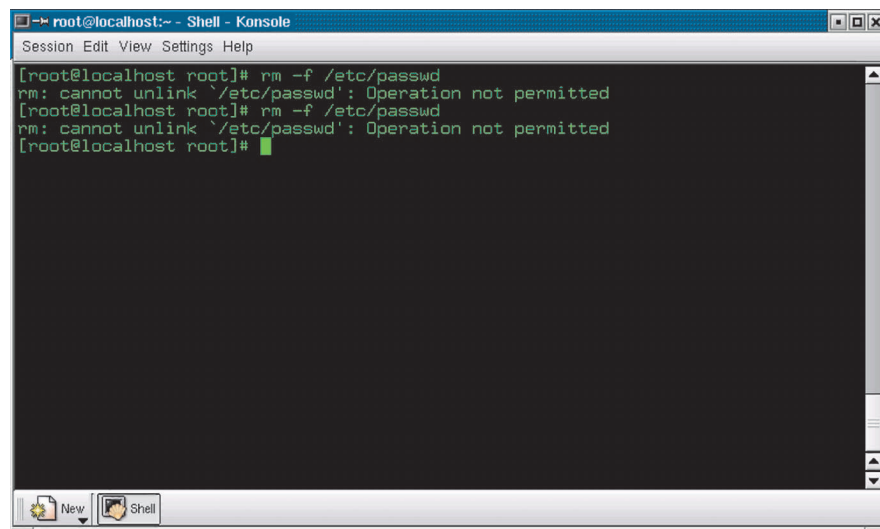
- **sname**: nama dari system call.
- **params**: parameter dari system call.
- **pid**: ID dari proses yang memanggil syscall.
- **uid**: ID dari user yang menjalankan proses ini.
- **euid**: user ID efektif dari user yang menjalankan proses.
- **suid**: user ID tercatat yang menjalankan proses.
- **gid**: ID dari group dari user yang menjalankan proses ini.
- **egid**: group ID efektif dari user yang menjalankan proses ini.
- **sgid**: group ID tercatat dari user yang menjalankan proses ini.
- **comm**: nama dari perintah yang dijalankan oleh proses ini.
- **retval**: nilai kembalian dari system call. Berguna untuk LOG bertipe "after".

Untuk contoh ini, saya menggunakan:

```
log_format {%comm : %params delete by
%euid -> %suid}
```

➔ Artinya, saya menginginkan dicatat perintah yang menjalankan system call *unlink()*, lalu user id efektif dan user id tercatat.

Pada variabel **when**, kita bisa definisikan "before" atau "after". Perbedaannya cukup



➤ Perintah menghapus file /etc/passwd digagalkan Syscalltracker.



jelas. Jika “before” pencatatan akan dilakukan sebelum system call dijalankan. Jika “after”, pencatatan akan dilakukan setelah system call dijalankan.

Sebagai penutup rule, tulis kurung kurawal tutup “}”. Seluruh rule ini bisa Anda tulis pada sembarang file text. Misal Anda namai coba.conf dan Anda tempatkan di /tmp. Berikutnya Anda perlu meload rule ini ke dalam syscalltracker.

```
# sct_config upload /tmp/coba.conf
```

Perintah di atas akan menampilkan output “Successfully uploaded rules from file ‘/tmp/coba.conf’ “. Jika tidak ada output seperti ini, cek ulang rule Anda. Anda bisa juga membaca dokumentasi yang terletak pada direktori hasil ekstrak source code syscalltracker, subdirektori doc.

OK, semua sudah dikerjakan. Tinggal menguji coba. Buka dua console, misal xterm pada X Window. Pada satu console, jalankan perintah sctlog.

```
# sctlog
```

Sctlog akan menampilkan ke stdout semua hasil intercept system call (syscall), jika terkena rule yang telah Anda definisikan. Di console lain, misal Anda lakukan sebagai berikut:

```
# cd /tmp
# touch ./hehe (perintah untuk membuat file bernama hehe)
# rm -f ./hehe
```

Maka pada console output sctlog, akan ada keluaran sebagai berikut:

```
"rm" : "./hehe" delete by 0 -> 0
```

Dari sini Anda bisa tahu, bahwa telah ada action sebagai berikut. Perintah “rm” dengan parameter “./hehe” menjalankan system call *unlink()* alias menghapus file. Perintah dijalankan dengan user ide efektif 0 alias root.

Satu lagi contoh rule seperti berikut ini.

```
rule
{
  syscall_name = unlink
  rule_name = cegah_delete
  filter_expression {PARAMS[1] = "/etc/
passwd" && UID == 0}
  action {
```

```
    type = FAIL
    error_code = -1
  }
  when = before
}
```

Contoh rule kedua ini mirip dengan contoh rule pertama, tapi di sini kita menggunakan action FAIL. Khusus untuk FAIL, kita harus mendefinisikan “error\_code” alias nilai kembalian system call. Di sini saya pakai nilai -1 alias “operation not permitted”. Daftar lengkap error code bisa Anda lihat di /usr/include/asm/errno.h.

Pada baris “filter expression”, saya mendefinisikan kondisi bahwa pengecekan dilakukan jika parameter pertama / PARAMS[1] (file yang akan dihapus) adalah /etc/passwd.

Catatan: ini mungkin tidak sempurna, mengingat bisa saja delete dilakukan sbb “cd /etc && rm -f ./passwd”. Variable UID dicek apakah sama dengan 0, berarti pengecekan dilakukan terhadap user ide root.

Sekarang coba load ulang rule di atas, dengan cara sebagai berikut.

```
# sct_config delete
# sct_config upload ./coba.conf
```

Perhatikan bahwa urutan rule juga penting. Misalnya Anda tulis rule “cegah\_delete” sebelum “unlink\_rule1”. Jika ada operasi perintah “rm -f /etc/passwd”, maka operasi “delete” akan match dengan rule “cegah\_delete” terlebih

dahulu dan operasi akan digagalkan.

Selanjutnya proses logging akan diabaikan. Tapi jika Anda balik (“unlink\_rule1” terlebih dahulu daripada “cegah\_delete”), maka operasi delete hanya akan di-log tanpa dicegah.

Selamat jika Anda berhasil sampai di sini, karena Anda sudah satu langkah dalam usaha menginvestigasi sistem. Memang tidak secanggih sistem intrusion detection lainnya, tapi syscalltracker menawarkan fleksibilitas yang cukup tinggi. Sebagai tambahan, jika Anda ingin melihat rule yang telah dimuat, ketikkan:

```
# sct_config download
```

Untuk menghapus semua rule yang telah dimuat, ketikkan:

```
# sct_config delete
```

Terakhir, jika Anda tidak lagi memerlukan syscalltracker, Anda bisa unload module-nya dengan perintah:

```
# sct_unload
```

Ada kemungkinan perintah ini gagal dengan output “Device or resource busy”. Jika demikian, ada kemungkinan modul syscalltracker masih berjalan. Biarkan saja sementara dan coba lagi beberapa saat kemudian. Modul syscalltracker ini tidak berbahaya bagi sistem dan tidak akan menambah beban kerja kernel secara berlebihan (kecuali ada rule sangat banyak). Jadi cukup aman sekalipun Anda tidak unload.

Mulyadi Santosa ([a\\_mulyadi@softhome.net](mailto:a_mulyadi@softhome.net))

```
root@localhost:~/home/mulyadi/script-syscalltracker - Shell - Konsole
Session Edit View Settings Help
[root@localhost script-syscalltracker]# sctlog
"python" : void fork by 0 --> 0
"python" : "/usr/tmp/@Z207.test" delete by 0 --> 0
"bash" : void fork by 0 --> 0
"bash" : void fork by 0 --> 0
"bash" : void fork by 0 --> 0
"bash" : void fork by 0 --> 0
"rm" : "./sampah" delete by 0 --> 0
```

▲ Syscalltracker mencatat aktivitas fork dan delete.