



12 Systrace

12.1 Einführung

Das Werkzeug *Systrace* von Niels Provos überwacht nicht ein gesamtes System, sondern einzelne Prozesse. Hierbei analysiert *Systrace* die von den Prozessen durchgeführten Systemaufrufe und vergleicht diese mit einem Regelsatz an erlaubten Aufrufen. Verletzt ein Prozess den Regelsatz, so kann *Systrace* den Prozess beenden.

Damit verhindert *Systrace* zwar nicht die Sicherheitslücke und ihre Ausnutzung zum Beispiel durch einen Bufferoverflow, aber deren negative Auswirkungen. Der Angreifer kann den Bufferoverflow zwar durchführen, aber der von ihm im Rahmen des Bufferoverflows injizierte Code wird wahrscheinlich nicht erlaubte Systemaufrufe enthalten und von *Systrace* gestoppt werden.

Systemaufrufe (engl. *systemscall*) werden von den Anwendungen verwendet, um Zugang zum Kernel zu erhalten. So kann eine Anwendung über Systemaufrufe Dateien öffnen, Netzwerkverbindungen aufbauen und neue Prozesse starten. Dies ist nur mit Systemaufrufen möglich. Alle sicherheitskritischen Vorgänge werden so von Anwendungen auf den Kernel übertragen. Die meisten modernen UNIX-Systeme verfügen über etwa 200 Systemaufrufe für die verschiedensten Funktionen. Ein Angreifer wird meist versuchen über einen Systemaufruf eine zusätzlich Shell zu starten (`fork()` oder `execve()`). Eine Überwachung der Systemaufrufe, wie *Systrace* sie durchführt kann daher die schädlichen Auswirkungen eines Angriffs meist vereiteln.

Dies soll kurz am Beispiel eines Webservers deutlich gemacht werden. Normalerweise darf die Webserver-Applikation jeden Systemaufruf durchführen. Sie benötigt jedoch nur eine geringe Auswahl davon:

- Zugriff auf Port 80
- Öffnen und Schreiben der Protokolldateien
- Lesen der Konfiguration und der Webseiten

Dieser Webserver muss jedoch keine Shell starten oder gar sicherheitskritische Dateien, wie die `/etc/passwd` oder die `/etc/shadow` lesen. Der Funktionsumfang des Programms lässt sich recht einfach durch die aufgerufenen Systemaufrufe definieren. Diese Eigenschaft wird von *Systrace* ausgenutzt. Es beobachtet den Prozess und ver-

gleicht die aufgerufenen Systemaufrufe mit der definierten Richtlinie. Ein Angreifer kann dann nicht Applikationsfremde Systemaufrufe verwenden.



Achtung:

Allerdings tritt durch Systrace ein gewisser Performanceverlust auf. Dieser hängt ab von der Anzahl der verwendeten Systemaufrufe. Systrace erlaubt in den Richtlinien auch die Verwendung von regulären Ausdrücken. Diese sollten aus Geschwindigkeitsgründen vermieden werden. Sie müssen jedoch in jedem Fall prüfen, ob der Einsatz unter diesen Umständen möglich ist.

12.2 Installation von Systrace

Systrace besteht aus einem Kernel-Patch und Userland-Applikationen. Sie können den Kernel-Patch und die Applikationen von der Systrace-Homepage <http://www.citi.umich.edu/u/provos/systrace/index.html> herunterladen. Zunächst sollten Sie Ihren Kernel patchen. Im Moment existieren zwei Patches für den Linux Kernel 2.4.24 und den Linux Kernel 2.6.1.

Um den Patch anzuwenden, wechseln Sie in das Sourcecode-Verzeichnis Ihres Kernel und rufen den folgenden Befehl auf:

```
patch -p1 --dry-run <../systrace-linux-<version>.diff
```

Wenn der Patch ohne Fehlermeldung durchläuft, können Sie in einem zweiten Durchgang die Option `--dry-run` weglassen. Dann wird der Patch tatsächlich angewendet. Wenn der Patch nicht erfolgreich ist, sollten Sie Ihren Kernel-Quelltextbaum überprüfen und möglicherweise neu installieren. Vielleicht kollidiert der Systrace-Patch auch mit weiteren von Ihnen oder Ihrem Distributor verwendeten Patches.

Sobald Sie den Kernel gepatcht haben, können sie *Systrace* in der Kernel-Konfiguration aktivieren (Abbildung 12.1).

Anschließend übersetzen, installieren und booten Sie den neuen Kernel. Nun müssen Sie das *Systrace*-Device erzeugen, falls Sie kein `devfs`-Dateisystem verwenden. Hierfür nutzen Sie den folgenden Befehl:

```
# mknod -m600 /dev/systrace c 10 226
```

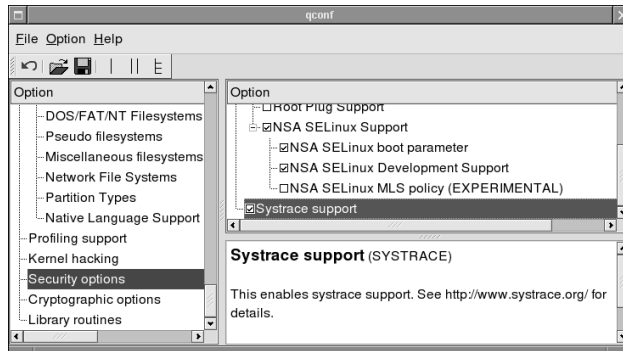


Abbildung 12.1 Beim Linux Kernel 2.6 schalten Sie Systrace bei den Security Options an.



Achtung:

An einigen Stellen im Internet liest man, dass das Gerät mit dem Minor-Code 223 erzeugt werden soll. Das ist definitiv falsch bei Einsatz der aktuellen Version. Wenn Sie die Fehlermeldung *No such device* bei dem Aufruf von Systrace erhalten, haben Sie das Gerät falsch angelegt.

Nun müssen Sie das *Systrace*-Werkzeug übersetzen und installieren. Dabei treten häufig Probleme mit der Datei *linux/systrace.h* auf. Sie können den Ort dieser Datei bei dem Aufruf des `configure`-Befehls angeben:

```
$ CPPFLAGS=-I/usr/src/linux-2.6.1/include ./configure
$ make
$ sudo make install
```

Auf einigen Linux-Distributionen schlägt bei dem `make`-Aufruf der Regressionstest fehl. Dies können Sie vernachlässigen. Das Werkzeug ist dennoch erfolgreich übersetzt worden.

Systrace verfügt auch über ein grafisches Benachrichtigungswerkzeug, wenn ein Programm eine Richtlinie verletzt. Dieses Werkzeug aus dem Paket *gtk-systrace* sollte auch installiert werden. Hier genügt ein einfacher Aufruf von:

```
$ ./configure
$ make
$ sudo make install
```

Anschließend sollte die folgende Verknüpfung erzeugt werden:

```
$ sudo cd /usr/X11R6/bin && ln -s /usr/local/bin/notification xsystrace
```

12.3 Systrace Anwendung

Der *systrace*-Befehl kann nun in drei verschiedenen Modi eingesetzt werden.

- *Lern-Modus*. In diesem Modus überwacht *Systrace* das Verhalten einer Anwendung und erzeugt eine Richtlinie, die genau dieses Verhalten erlaubt. Hiermit können Sie sehr einfach eine Richtlinie für eine Applikation erzeugen. Es ist natürlich wichtig, dass Sie dabei sämtliche benötigten Funktionen der Applikation nutzen und keine Sicherheitslücken ausnutzen!
- *Nachfrage-Modus*. In diesem Modus kontrolliert *Systrace* das Verhalten einer Anwendung anhand einer bereits existierenden Richtlinie. Ist bisher keine Richtlinie vorhanden, so legt *Systrace* eine an. Verletzungen der Richtlinie werden an den Benutzer mit dem grafischen Client gemeldet.
- *Durchsetzungs-(Enforcement)-Modus*. In diesem Modus setzt *Systrace* eine Richtlinie ohne Nachfrage durch. Verletzungen werden abgelehnt und protokolliert.

Systrace eignet sich für die Überwachung von Programmen, die mit unbekanntem Eingaben umgehen müssen. Insbesondere Netzwerkdienste können mit *Systrace* gesichert werden. Wenn Sie das betroffene Programm in einer sicheren Umgebung aufrufen und nutzen können, macht es Sinn *Systrace* im Lern-Modus zu starten. Hierfür verwenden Sie den Aufruf `systrace -A programm`. Ist das nicht der Fall, sollten Sie den Nachfrage-Modus wählen: `systrace programm`. Im Nachfrage-Modus wird *Systrace* bei dem Aufruf eines bisher nicht erlaubten Systemaufrufes den Befehl `xsystrace` starten. Falls Ihnen keine grafische Oberfläche zur Verfügung steht, können Sie mit der Option `-t` die Nachfrage auf der Textoberfläche aktivieren.

Weitere Optionen, die Sie bei dem Aufruf angeben können, sind:

- `-a`. Dies aktiviert die automatische Durchsetzung der Richtlinien ohne Nachfrage. Verletzungen werden abgelehnt und protokolliert.
- `-A`. Dies aktiviert den automatischen Lern-Modus. Jeder Systemaufruf wird erlaubt und in der Richtlinie entsprechend konfiguriert.
- `-c uid:gid`. Der Benutzer und die Gruppe, unter der *Systrace* das zu überwachende Programm zu startet.
- `-C`. Cradle-Modus. Hier verwendet *Systrace* einen zentralen Cradle-Server für die Verwaltung der *Systrace*-Prozesse. Damit ist es möglich eine UI anzuheften und wieder abzulösen.
- `-d dir`. Richtlinienverzeichnis.
- `-f file`. Die Richtlinien in dieser Datei werden zu den allgemeinen Richtlinien hinzugefügt.
- `-g gui`. Ein alternatives grafisches Benachrichtigungsprogramm.
- `-i`. Kindprozesse erben die Richtlinie des Elternprozesses.
- `-p pid`. In dieser Datei speichert *Systrace* seine Prozessnummer.
- `-t`. Verwendet Textmodus für interaktive Updates.

- -u. Deaktiviert das Aliasing von Systemaufrufen. Bei dem Aliasing fasst *Systrace* mehrere gleichartige Systemaufrufe unter einem Namen zusammen.
- -U. Ignoriert die Benutzereigenen Richtlinien in *~/systrace* und verwendet die globalen Richtlinien.

Eine typische Richtliniendatei für den Apache-Webserver ist im folgenden abgedruckt:

```
Policy: /usr/sbin/httpd, Emulation: native
native-__semctl: permit
native-__sysctl: permit
native-accept: permit
native-bind: sockaddr eq "inet-[0.0.0.0]:443" then permit
native-bind: sockaddr eq "inet-[0.0.0.0]:80" then permit
native-break: permit
native-chdir: filename eq "/" then permit
native-chown: filename match "/var/www/logs/*" then permit
native-chroot: permit
native-close: permit
native-connect: sockaddr sub ":53" then permit
native-dup2: permit
native-exit: permit
native-fcntl: permit
native-flock: permit
native-fork: permit
native-fsread: filename eq "/dev/arandom" then permit
native-fsread: filename eq "/dev/null" then permit
native-fsread: filename eq "/dev/tty" then permit
native-fsread: filename eq "/etc" then permit
native-fsread: filename eq "/etc/group" then permit
native-fsread: filename eq "/etc/hosts" then permit
native-fsread: filename eq "/etc/malloc.conf" then permit
native-fsread: filename eq "/etc/resolv.conf" then permit
native-fsread: filename eq "/etc/spwd.db" then permit
native-fsread: filename eq "/etc/ssl" then permit
native-fsread: filename eq "/htdocs" then permit
native-fsread: filename eq "/usr/libexec/ld.so" then permit
native-fsread: filename eq "/usr/sbin/suexec" then permit
native-fsread: filename eq "/var/run/ld.so.hints" then permit
native-fsread: filename eq "/var/www" then permit
native-fsread: filename eq "<non-existent filename>" then deny[enoent]
native-fsread: filename match "/etc/ssl/*" then permit
native-fsread: filename match "/htdocs/*" then permit
native-fsread: filename match "/usr/lib/*" then permit
native-fsread: filename match "/usr/share/*" then permit
native-fsread: filename match "/var/www/*" then permit
native-fstat: permit
native-fstatfs: permit
```

```
native-fswrite: filename eq "/dev/null" then permit
native-fswrite: filename eq "/dev/tty" then permit
native-fswrite: filename match "/logs/*" then permit
native-fswrite: filename match "/var/www/logs/*" then permit
native-getdirentries: permit
native-geteuid: permit
native-getpid: permit
native-getrlimit: permit
native-getsockname: permit
native-gettimeofday: permit
native-getuid: permit
native-ioctl: permit
native-issetugid: permit
native-kill: permit
native-listen: permit
native-lseek: permit
native-mmap: permit
native-mprotect: permit
native-munmap: permit
native-pread: permit
native-pwrite: permit
native-read: permit
native-recvfrom: permit
native-select: permit
native-semget: permit
native-semop: permit
native-sendto: true then permit
native-setegid: permit
native-seteuid: permit
native-setgid: permit
native-setgroups: permit
native-setrlimit: permit
native-setsid: permit
native-setsockopt: permit
native-setuid: permit
native-shutdown: permit
native-sigaction: permit
native-sigprocmask: permit
native-sigreturn: permit
native-socket: permit
native-umask: permit
native-wait4: permit
native-write: permit
native-writev: permit
```

12.4 Systrace-Beispiel

Im folgenden soll nun ein Beispiel an Hand des *Vsftpd*-FTP-Servers durchgespielt werden. Dies ist ein FTP-Server der in den aktuellen Distributionen häufig eingesetzt wird. FTP-Server erlauben häufig anonyme Anmeldungen und sind daher besonders gefährdet.

Zunächst sollten Sie den *Vsftpd*-Server mit *Systrace* im Lernmodus als *root* starten:

```
# systrace -A /usr/sbin/vsftpd
```

Anschließend beenden Sie den *Vsftpd*-Server wieder. Hierzu ermitteln Sie die Prozessnummer des FTP-Servers (nicht des *Systrace*-Prozesses und beenden den FTP-Server. *Systrace* beendet sich dann selbst. Sie werden nun in Ihrem Heimatverzeichnis ein Verzeichnis *./systrace* in dem sich eine Datei *usr_sbin_vsftpd* befindet. Diese Datei hat nun den folgenden Inhalt:

```
Policy: /usr/sbin/vsftpd, Emulation: linux
linux-newuname: permit
linux-brk: permit
linux-fsread: filename eq "/etc/ld.so.preload" then permit
linux-fsread: filename eq "/etc/ld.so.cache" then permit
linux-fstat64: permit
linux-old_mmap: permit
linux-close: permit
linux-fsread: filename eq "/usr/lib/libwrap.so.0" then permit
linux-read: permit
linux-fsread: filename eq "/lib/libnsl.so.1" then permit
linux-fsread: filename eq "/lib/libpam.so.0" then permit
linux-fsread: filename eq "/lib/libcap.so.1" then permit
linux-fsread: filename eq "/lib/libdl.so.2" then permit
linux-fsread: filename eq "/lib/tls/libc.so.6" then permit
linux-mprotect: permit
linux-ni_syscall-34: permit
linux-munmap: permit
linux-getuid: permit
linux-fsread: filename eq "/etc/vsftpd/vsftpd.conf" then permit
linux-mmap2: permit
linux-socket: sockdom eq "AF_INET" and socktype eq "SOCK_STREAM"
↳ then permit
linux-setsockopt: true then permit
linux-rt_sigprocmask: permit
linux-rt_sigaction: permit
linux-bind: sockaddr eq "inet-[0.0.0.0]:21" then permit
linux-listen: true then permit
linux-accept: true then permit
```

Wenn Sie nun den FTP-Server wieder unter der Überwachung von *Systrace* starten, wird *Systrace* die Einhaltung dieser Regeln kontrollieren.

```
# systrace -a /usr/sbin/vsftpd
```

Wenn Sie nun eine Verbindung zu dem FTP-Server aufbauen, wird diese nicht erlaubt werden und eine Protokollierung auslösen:

```
Oct 11 15:24:17 bibo systrace: deny user: root, prog: /usr/sbin/vsftpd, pid:
↳ 2496(6)[0], policy: /usr/sbin/vsftpd, filters: 27, syscall:
↳ linux-clone(120), args: 8
```

Wenn Sie interaktiv die Richtlinien des Prozesses verändern möchten, so können Sie *Systrace* ohne Optionen aufrufen.

```
# systrace /usr/sbin/vsftpd
```

Sobald nun ein Client eine Verbindung zum FTP-Server aufbaut, wird die Benachrichtigungsanwendung (*xsystace*) gestartet. Sie werden gefragt, ob Sie die erforderlichen Systemaufrufe zulassen möchten oder nicht (Abbildung 12.2).

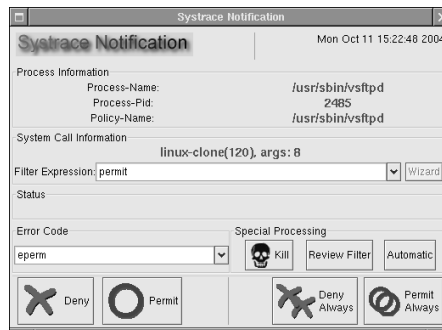


Abbildung 12.2 Systrace fängt den Systemaufruf ab und fragt den Anwender um Erlaubnis.

Wenn Sie bei allen Anfragen diese mit PERMIT bestätigen, werden die folgenden Einträge zur Richtlinie hinzugefügt:

```
linux-clone: permit
linux-dup2: permit
linux-getpeername: true then permit
linux-fsread: true then permit
linux-fcntl64: permit
linux-setsid: permit
linux-getpgrp: permit
linux-umask: permit
linux-setgroups: permit
linux-chdir: permit
```

```
linux-chroot: permit  
linux-setgid: permit  
linux-setuid: permit  
linux-alarm: permit  
linux-write: permit
```

Nun kann der *Vsftpd*-FTP-Server immer unter der Überwachung von *Systrace* gestartet werden. Dies zeigt, wie einfach die Erzeugung der *Systrace*-Richtlinien ist.

