Systemaufrufe mit Systrace reglementieren

— Wie man Systemaufrufe überwacht und steuert —

Stefan Schumacher

www.net-tex.de
stefan@net-tex.de
PGP:0xB3FBAE33

Chemnitzer Linux-Tage 2007

Id: sicherheitshandbuch-systrace.tex,v 1.15 2007/02/24 13:02:43 stefan Exp



Inhaltsverzeichnis

1. Motivation
2. Implementierung
3. Praktische Einsatzmöglichkeiten
4. Die Richtlinie und ihre Grammatik
5. Praktischer Einsatz

Literatur

- *Eriksen und Provos* Eriksen, Marius A. ; Provos, Niels: Enges Korsett: Systrace setzt Regeln für erlaubte Systemaufrufe durch. In: Linux Magazin 01/03
- *Peterman 2005* Peterman, Matthias: Systrace-Restricted Login-Shell mit NetBSD. (2005). `http://wiki.bsd-crew.de/index.php/Systrace-Restricted_Login-Shell_mit_NetBSD`
- *Provos 2006* Provos, Niels: Improving Host Security with System Call Policies. (2006). `http://www.citi.umich.edu/techreports/reports/citi-tr-02-3`
- *Thompson 1984* Thompson, Ken: Reflections on Trusting Trust. In: Communication of the ACM, Association for Computing Machinery, Inc, 1984, `http://www.ece.cmu.edu/~ganger/712.fall02/papers/p761-thompson.pdf`



1 Motivation

2 Implementierung

3 Praktische Einsatzmöglichkeiten

4 Die Richtlinie und ihre Grammatik

5 Praktischer Einsatz



Wo ist das Problem?

- Sicherheit erfordert Verteidigung in der Tiefe
- mehrere unabhängige Verteidigungslinien sichern ein System
- jede Verteidigungslinie ist aber wieder Sicherheitsrisiko
- Software ist fehlerbehaftet
- auch Quelloffene! ($\rightarrow$ (Thompson 1984))
- geschlossene Software ist nicht analysierbar



Und nun?

- Angriffe zielen auf Systemaufrufe
- Systemaufrufe ermöglichen privilegierte Operationen im Kernel
  bspw: Datei öffnen / speichern, Prozess forken / beenden
- Systemaufrufe überwachen um Angriffe zu erkennen
- oder gleich proaktiv Angriffe verhindern

Und wie?

ganz einfach:

- Zugriffe auf Systemaufrufe reglementieren

Was braucht man dazu?

- Richtlinie, die das *Normal*verhalten eines Prozesses beschreibt
- Programm, das den Prozess mit der Richtlinie vergleicht
- und Entscheidungen trifft



Und wie?

ganz einfach:

- Zugriffe auf Systemaufrufe reglementieren

Was braucht man dazu?

- Richtlinie, die das *Normal*verhalten eines Prozesses beschreibt
- Programm, das den Prozess mit der Richtlinie vergleicht
- und Entscheidungen trifft



Im Detail

- Richtlinie muss alle Eventualitäten beachten (bspw. Symlinks)
- daher recht komplex $\rightsquigarrow$ komplexe Grammatik
- automatisierte Erstellung
- Außerdem Protokolle zwecks Analyse erstellen



1 Motivation

2 Implementierung

3 Praktische Einsatzmöglichkeiten

4 Die Richtlinie und ihre Grammatik

5 Praktischer Einsatz



Implementierung, erste Ideen

Systrace im Kernel:

- + schnell
- − komplex
- − wenig portabel

Systrace im Userland:

- + portabel
- − langsam
- − unsicher (Race-Condition)



Implementierung, erste Ideen

Systrace im Kernel:

- + schnell
- − komplex
- − wenig portabel

Systrace im Userland:

- + portabel
- − langsam
- − unsicher (Race-Condition)



Implementierung, die Lösung

Hybrid
teils im Kernel

- behandelt kontextunabhängige Systemaufrufe (schnell)
- ermöglicht Sandkasten der nicht verlassen werden kann
- überwacht Forks und vererbt Richtlinie an Kindprozesse

teils im Userland:

- behandelt kontextabhängige Systemaufrufe im Userland
- Kernel-Teil blockiert fraglichen Prozess solange
- Dæmon kann Informationen vom Kernel-Teil erfragen



Implementierung, die Lösung

Hybrid
teils im Kernel

- behandelt kontextunabhängige Systemaufrufe (schnell)
- ermöglicht Sandkasten der nicht verlassen werden kann
- überwacht Forks und vererbt Richtlinie an Kindprozesse

teils im Userland:

- behandelt kontextabhängige Systemaufrufe im Userland
- Kernel-Teil blockiert fraglichen Prozess solange
- Dæmon kann Informationen vom Kernel-Teil erfragen

Problem: Race-Condition

Race-Condition

- Ein Prozess manipuliert die Argumentes eines anderen Prozesses, während dieser überprüft wird

Gegenmaßnahme:

- Kernel normalisiert Argumente des Systemaufrufes
- Normalform wird schreibgeschützt gepuffert
- Userland-Dæmon und Kernel verwenden nur noch Normalform
  $\rightsquigarrow$ Manipulation greift nicht mehr



Problem: Race-Condition

Race-Condition

- Ein Prozess manipuliert die Argumentes eines anderen Prozesses, während dieser überprüft wird

Gegenmaßnahme:

- Kernel normalisiert Argumente des Systemaufrufes
- Normalform wird schreibgeschützt gepuffert
- Userland-Dæmon und Kernel verwenden nur noch Normalform
  $\rightsquigarrow$ Manipulation greift nicht mehr
- Außerdem: Richtliniendatei schützen
  (`chflags schg`, CD-R, mtree-Signatur o.ä.)



Problem: Race-Condition

Race-Condition

- Ein Prozess manipuliert die Argumentes eines anderen Prozesses, während dieser überprüft wird

Gegenmaßnahme:

- Kernel normalisiert Argumente des Systemaufrufes
- Normalform wird schreibgeschützt gepuffert
- Userland-Dæmon und Kernel verwenden nur noch Normalform
  $\rightsquigarrow$ Manipulation greift nicht mehr
- Außerdem: Richtliniendatei schützen
  (`chflags schg`, CD-R, mtree-Signatur o.ä.)

Lauf-Modi

- Initialisierungsmodus: Systrace überwacht Programm und generiert Richtlinie, in der alle Aktionen erlaubt sind, guter Startpunkt
- Nachfragemodus: Systrace überwacht Programm und generiert Richtlinie, fragt aber bei jeder Aktion den Benutzer
- Überwachungsmodus: Systrace überwacht Programm und setzt die definierte Richtlinie durch

1 Motivation

2 Implementierung

3 Praktische Einsatzmöglichkeiten

4 Die Richtlinie und ihre Grammatik

5 Praktischer Einsatz



Schadmöglichkeiten der Programme eingrenzen

- Ein Programm mit Root-Rechten darf alles
- Apache kann eine /bin/sh forken. Muß er das können?
- Natürlich nicht, also verbieten wir es.
- Richtlinie erstellen, die alle erlaubten Systemaufrufe festlegt
- Programm überwachen und alles andere verbieten

Schadmöglichkeiten der Programme eingrenzen

- Ein Programm mit Root-Rechten darf alles
- Apache kann eine /bin/sh forken. Muß er das können?
- Natürlich nicht, also verbieten wir es.
- Richtlinie erstellen, die alle erlaubten Systemaufrufe festlegt
- Programm überwachen und alles andere verbieten

SETUID verhindern

- Programme benötigen Root-Rechte, laufen daher SETUID
- benutzen in der Regel nur sehr wenige Systemaufrufe mit Root-Rechten
- können aber trotzdem alles als Root, bspw. Shells forken
- Idee: Programm läuft als normaler Benutzer
- Systrace gestattet Root-Zugriffe auf notwendige Systemaufrufe



Nutzer einschränken

- Shell des Benutzers steht unter Systrace-Überwachung
- Richtlinie erlaubt oder verbietet Aktionen
- feine Granulierung möglich, bspw. SSH nur auf bestimmte Adressen erlauben
- Programme können komplett gesperrt werden

1 Motivation

2 Implementierung

3 Praktische Einsatzmöglichkeiten

4 Die Richtlinie und ihre Grammatik

5 Praktischer Einsatz



- Aneinanderreihung von Ausdrücken
- werden traversiert
- erster passender Ausdruck wird angewendet
- Aufbau eines Ausdrucks:
  `Systemaufruf: Argumente Operator Vergleichswert Aktion [als Benutzer] [Log]`

Filter-Ausdrücke

- *match* Wahr, wenn der Dateiname gemäß den Regeln in fnmatch(3) getroffen wird.
- *eq* Wahr, wenn das Argument des Systemaufrufes genau der Vorgabe entspricht.
- *neq* Die logische Negation des *eq*-Operators
- *sub* Wahr, wenn die angegebene Teilzeichenkette im Argument des Systemaufrufes vorkommt.
- *nsub* Die logische Negation des *sub*-Operators
- *inpath* Wahr, wenn das Argument des Systemaufrufes im Pfad der Vorgabe vorkommt.
- *re* Wahr, wenn das Argument des Systemaufrufes dem angegebenen Regulären Ausdruck entspricht.



Aktionen

- Aktionen: `deny`, `permit`, `ask`
- als Benutzer ausführen: `permit as root`
- Fehlercode: `deny [FEHLER]`
- Wenn Benutzer: `then deny[eperm], if group != wheel or user = stefan`
- Protokollierung: `log`
- Verknüpfungen: `and` und `or`



Beispiele für Ausdrücke

- `netbsd-execve: permit`
- `netbsd-execve: true then permit log`



Beispiele für Ausdrücke

- `netbsd-execve: permit`
- `netbsd-execve: true then permit log`
- `netbsd-fsread: filename eq ''/etc/passwd'' then permit`



Beispiele für Ausdrücke

- `netbsd-execve: permit`
- `netbsd-execve: true then permit log`
- `netbsd-fsread: filename eq ''/etc/passwd'' then permit`
- `netbsd-fsread: filename match ''/etc*'' then deny [eaccess] log`



Beispiele für Ausdrücke

- `netbsd-execve: permit`
- `netbsd-execve: true then permit log`
- `netbsd-fsread: filename eq ''/etc/passwd'' then permit`
- `netbsd-fsread: filename match ''/etc*'' then deny [eaccess] log`
- `netbsd-connect: sockaddr re ''inet-.192\.168\.[0,4,8]\.[4-6].:22'' then permit log`



Beispiele für Ausdrücke

- `netbsd-execve: permit`
- `netbsd-execve: true then permit log`
- `netbsd-fsread: filename eq ''/etc/passwd'' then permit`
- `netbsd-fsread: filename match ''/etc*'' then deny [eaccess] log`
- `netbsd-connect: sockaddr re ''inet-.192\.168\.[0,4,8]\.[4-6].:22'' then permit log`
- `netbsd-seteuid: uid eq ''1007'' and uname eq ''systraced'' then permit`



Beispiele für Ausdrücke

- `netbsd-execve: permit`
- `netbsd-execve: true then permit log`
- `netbsd-fsread: filename eq ''/etc/passwd'' then permit`
- `netbsd-fsread: filename match ''/etc*'' then deny [eaccess] log`
- `netbsd-connect: sockaddr re ''inet-.192\.168\.[0,4,8]\.[4-6].:22'' then permit log`
- `netbsd-seteuid: uid eq ''1007'' and uname eq ''systraced'' then permit`



1 Motivation

2 Implementierung

3 Praktische Einsatzmöglichkeiten

4 Die Richtlinie und ihre Grammatik

5 Praktischer Einsatz



Praktischer Einsatz: Apache überwachen 1/2

- Richtlinie erstellen
  `systrace -A /usr/pkg/sbin/apachectl start && /usr/pkg/sbin/apachectl stop`
- Richtlinie manuell anpassen

```
1 netbsd-chmod: filename  "/var/run/httpd.mm..sem"
2               and mode eq "600" then permit
3
4 netbsd-fswrite: filename  "/var/run/httpd.lock."
5                 then permit
6
```



Praktischer Einsatz: Apache überwachen 1/2

- Richtlinie erstellen
  `systrace -A /usr/pkg/sbin/apachectl start && /usr/pkg/sbin/apachectl stop`
- Richtlinie manuell anpassen

```
1 netbsd-chmod: filename eq "/var/run/httpd.mm.2872.sem"
2               and mode eq "600" then permit
3
4 netbsd-fswrite: filename eq "/var/run/httpd.lock.1827"
5                 then permit
6
```



Praktischer Einsatz: Apache überwachen 1/2

- Richtlinie erstellen
  `systrace -A /usr/pkg/sbin/apachectl start && /usr/pkg/sbin/apachectl stop`
- Richtlinie manuell anpassen

```
1 netbsd-chmod: filename match "/var/run/httpd.mm.*.sem"
2               and mode eq "600" then permit log
3
4 netbsd-fswrite: filename match "/var/run/httpd.lock.*"
5                 then permit log
6
```



Praktischer Einsatz: Apache überwachen 2/2

- Testlauf starten und nachsteuern
  `systrace /usr/pkg/sbin/apachectl start`
  `xterm -e tail -f /var/log/messages`
  (Echtzeitfehlermeldungen)
- Webseiten aufrufen
  Richtlinie mit Regulären Ausrdücken auf /home/www erweitern



Praktischer Einsatz: Apache überwachen 2/2

- Testlauf starten und nachsteuern
  `systrace /usr/pkg/sbin/apachectl start`
  `xterm -e tail -f /var/log/messages`
  (Echtzeitfehlermeldungen)
- Webseiten aufrufen
  Richtlinie mit Regulären Ausrdücken auf /home/www erweitern



Praktischer Einsatz: Apache ohne Root-Rechte 1/1

- `systrace -c 1002:1001 apachectl start`
  startet als www:www, verschiedene Operationen schlagen fehl
- Root-Operationen anpassen:

```
1 netbsd-fswrite: filename match
2 "/var/run/httpd.mm.*.sem" then permit
3
4 netbsd-bind: sockaddr eq "inet-192.168.0.5:80"
5 then permit
```



Praktischer Einsatz: Apache ohne Root-Rechte 1/1

- `systrace -c 1002:1001 apachectl start`
  startet als www:www, verschiedene Operationen schlagen fehl
- Root-Operationen anpassen:

```
1 netbsd-fswrite: filename match
2 "/var/run/httpd.mm.*.sem" then permit as root
3
4 netbsd-bind: sockaddr eq "inet-192.168.0.5:80"
5 then permit as root
```



Praktischer Einsatz: Systrace-Shell 1/6

Prinzip

- Login-Shell die von Systrace überwacht wird
- Richtline für die Shell und andere Programme
- Richtlinie wird vererbt oder eigene Richtlinien für Programme



Praktischer Einsatz: Systrace-Shell 2/6

Die Shell starten

„`systrace -a -i ksh`“ geht nicht in /etc/master.passwd

Kapselung in C-Programm:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4
5  int main()
6  {
7   puts("Fuehre Systrace-ueberwachte Shell aus");
8   execlp("/bin/systrace","systrace","-a","-i","/bin/ksh",NUL
9   return 0;
10 }
```



Praktischer Einsatz: Systrace-Shell 3/6

Vorgehensweise

- einloggen als überwachter Benutzer mit *normaler* /bin/ksh
- dann Shell auf Kapsel-Programm umbiegen
- Richtlinie anpassen durch ausprobieren

Praktischer Einsatz: Systrace-Shell 4/6

Richtlinien-Beispiele: Konfigurationsdateien

```
1  \# Zugriffe auf Konfigurationsdateien erlauben
2  netbsd-fsread: filename eq "/etc/man.conf" then permit
3  netbsd-fsread: filename eq "/etc/passwd" then permit
4  netbsd-fsread: filename match "/etc*" then deny
5  netbsd-fsread: filename match "/home/systraced/*" then perm
6  netbsd-fsread: filename match "/home*" then deny
7  netbsd-fsread: filename match "/tmp/*" then permit
8  netbsd-fsread: filename match "/var/tmp/*" then permit
9  netbsd-fswrite: filename match "/dev/tty" then permit
10 netbsd-fswrite: filename match "/tmp/*" then permit
11 netbsd-fswrite: filename match "/var/tmp/*" then permit
12 netbsd-fswrite: filename eq "/dev/crypto" then permit
```



Praktischer Einsatz: Systrace-Shell 5/6

Richtlinien-Beispiele: Sockets

```
1  # SSH auf bestimmte Adressen erlauben
2  netbsd-socket: sockdom eq "AF_INET" and socktype
3                 match "*" then permit
4  netbsd-connect: sockaddr eq
5                  "inet-[127.0.0.1]:22" then permit log
6  netbsd-connect: sockaddr re
7  "inet-.192\.168\.[0,4,8]\.[4-6].:22" then permit log
```



Praktischer Einsatz: Systrace-Shell 6/6

Richtlinien-Beispiele

```
1  # Ausf"uhrbare Dateien in $HOME und ftp + telnet verbieten
2  netbsd-exec: filename sub "/home/systraced" then deny log
3  netbsd-execve: filename sub "/home/systraced" then deny log
4  netbsd-execve: filename eq "/usr/bin/ftp" then permit log
5  netbsd-execve: filename eq "/usr/bin/telnet" then permit lo
6  netbsd-exec: true then permit log
7  netbsd-execve: true then permit log
8
9  netbsd-mmap: permit
10 netbsd-fsread: permit
11 netbsd-__fstat13: permit
12 netbsd-close: permit
```



Und sonst noch?

- gutes Werkzeug zur Software-Analyse
  Was macht eigentlich Acrobat Reader alles so genau?
- gutes Werkzeug für Software-Tests
  Was macht eigentlich Acrobat Reader wenn er nicht mehr auf /tmp zugreifen darf?

Fazit

- Systrace ermöglicht feingranulierte Richtlinien
- Kann SETUID ersetzen
- komplette Benutzeraktionen können überwacht und reglementiert werden
- umfangreiches Wissen über Systemaufrufe/Fehlercodes notwendig
- benötigt ebenso umfassende Sicherheitsrichtlinie



Ende

Fragen?

Kontakt: www.net-TEX.de

stefan@net-TEX.de

oder auf den Chemnitzer Linux-Tagen nochmal zuhören ;-)



Ende

Fragen?

Kontakt: www.net-TEX.de

stefan@net-TEX.de

oder auf den Chemnitzer Linux-Tagen nochmal zuhören ;-)

