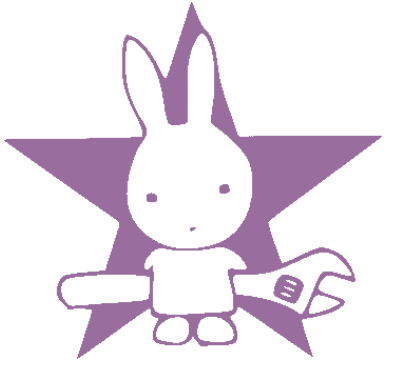
Aus meinem Werkzeugkasten: extrace

Leah Neukirchen

<leah@vuxu.org>



Easterhegg 21, Regensburg

2024-03-31

Welche Prozesse laufen auf meinem System?

- Die Klassiker `ps`, `top`, `htop` zeigen nur Prozesse an, die genau gerade oder schon lange genug laufen.
- Viele Prozesse sind aber sehr kurzlebig und schwer einzufangen:
  - Cronjobs
  - Diverse Helper die durch `udev`, `acpid`, `dhcpcd` etc. aufgerufen werden
  - Werkzeuge ala `execline` rufen sofort andere Programme auf.
  - Komplizierte Skripte wie `./configure` rufen abertausende Programme auf.

Bisherige Ansätze

- Lokale Prozessbäume beobachten ist möglich mit `strace`, aber ineffizient.
- Shell-Skripte kann man mit `-x` tracen, oft unzufriedenstellend da viel Kontext fehlt.
- Das Audit-Subsystem erfordert Konfiguration und erzeugt schwerverständliche Logdateien.
- `bcc-tools` haben `execsnoop`, gabs 2014 noch nicht. ;)

extrace

Mein Werkzeug extrace erlaubt:

- Globales Tracing *aller* Prozesse, die auf einem System ausgeführt werden. (Filtern auf Kindprozesse ist möglich.)
- Ausgabe von PID, Laufzeit, Rückgabewert, User, Umgebungsvariablen, Arbeitsverzeichnis
- In einem übersichtlichen Baumformat

Beispiel: ein SSH-Login, was mein Zsh-Prompt so macht

```
% extrace -t -p $(pidof sshd)
1982+ /usr/bin/sshd -D -R
    1985+ -zsh
      1986+ getent group leah
      1986- getent exited status=0 time=0.001s
      1988+ touch /home/leah/.zdirs
      1988- touch exited status=0 time=0.001s
      1989+ tty
      1989- tty exited status=0 time=0.001s
        1991+ jj root --ignore-working-copy
        1991- jj exited status=1 time=0.012s
        1994+ git rev-parse --show-prefix
        1994- git exited status=128 time=0.001s
    1985- -zsh exited status=0 time=4.041s
1982- /usr/bin/sshd exited status=255 time=4.277s
```

Einsatzzwecke

Ursprünglich entwickelt, um `autoconf`-Skripte zu debuggen, aber halt sich als extrem wertvoll herausgestellt:

- Was läuft überhaupt so im Hintergrund? (z. B. löst eine RA-Fehlkonfiguration alle 5s `dhcpcd`-Hooks aus…)
- Wurde der Prozess mit den richtigen Umgebungsvariablen aufgerufen?
- Welche Compiler-Flags waren aktiv als diese Objektdatei kompliert wurde?
- Welche Aufrufe eines Programms sind besonders langsam?
- Was macht mein `maildrop` genau wenn eine Mail angeliefert wird?
- Was führt mein `sshd` sonst so alles hinter meinem Rücken aus ._.

Nachteile

- Benötigt erweiterte Rechte (`CAP_NET_ADMIN`) unter Linux.
- Der Linux-Kernel braucht `CONFIG_CONNECTOR=y` sowie `CONFIG_PROC_EVENTS=y` (ist aber bei fast allen Distrokerneln an, Embedded oder selbstgebaut oft aus).
- Die Schnittstelle hat inhärent kritische Wettläufe, bei Systemen unter hoher Last gehen Events verloren (wird aber erkannt und gemeldet).
- Fehlschlagende Programmaufrufe (Programmdatei fehlt etc.) werden nicht erkannt.

Ausblick

- Es gibt eine experimentelle Neufassung mittels eBPF, aber kompliziert unabhängig von der Kernelversion verfügbar zu machen.
- Es gibt Portierungen für OpenBSD und FreeBSD (mittels `kevent`).
- Verfügbar unter der GPL2 auf `https://github.com/leahneukirchen/extrace` sowie fertig für Debian, NixOS, openSUSE, Ubuntu, Void.
- **Noch Fragen?**
- Weitere Tools siehe `https://github.com/leahneukirchen/leahutils`