Design and Implementation of a Forensic Documentation Tool for Interactive Command-line Sessions

Abschlussvortrag

Tim Weber

15. März 2010

Protokollierung von Terminalsitzungen

- Im Zuge forensischer Ermittlungen
- Umfassende und detaillierte Aufzeichnung der durchgeführten Tätigkeiten
- Dokumentation für
  - den Ermittler selbst
  - mögliche Amtsnachfolger
  - unabhängige Gutachter o.Ä.

Einsatzbeispiel SCRIPT

Example

```
$ script -t typescript 2> timing
Script started, file is typescript
$ dd if=/dev/sda | nc 10.10.1.1 1234
...
$ logout
Script done, file is typescript
```

Was tut SCRIPT?

Example

```
$ script
Script started, file is typescript
$ ps faux
...
3704 ?      S   /usr/bin/x-terminal-emulator
3705 pts/4  Ss   \_ -bash
3843 pts/4  S+      \_ script
3844 pts/4  S+          \_ script
3845 pts/7  Ss              \_ bash -i
3878 pts/7  R+                  \_ ps faux
```

Terminal-Emulator $\Leftrightarrow$ *pts/4* $\Leftrightarrow$ SCRIPT $\Leftrightarrow$ *pts/7* $\Leftrightarrow$ BASH

Typescript

- 1:1-Ausgabe der als Kindprozess laufenden Software
- Inklusive Steuerzeichen etc.

Example

```
^[[1;32mscy@ ~ $^[[0m echo test^M
test^M
```

Timingdaten

- ASCII-Daten
- Zu wartende Zeit in Sekunden; Anzahl auszugebender Bytes

Example

```
0.163965 117
0.038713 1
9.803904 1
0.175984 1
0.023397 2
...
```

Schwächen

- Nur Bildschirmausgabe wird protokolliert, Eingaben überhaupt nicht
  - Tab-Completion? `^C`/`^D`/`^M`?
- Timinginformationen werden in separater Datei gespeichert
- Keine Informationen über die Laufzeitumgebung
  - Zeichenkodierung
  - Umgebungsvariablen
  - Fenstergröße
  - ...

Ziele der Bachelorarbeit

- Ausführliche Dokumentation des Verhaltens von SCRIPT, Aufzeigen seiner Schwächen
- Gestalten eines neuen Dateiformates, das forensischen Ansprüchen genügt
- Entwickeln einer Software, die in diesem Format protokolliert
- Dokumentieren dieser Software nach den Prinzipien des *literate programming*

Bestandteile einer FORSCRIPT-Datei

Ein einziger chronologisch fortlaufender Datenstrom, enthält:

- Ausgaben der Clientsoftware
- Eingaben des Benutzers
- Metainformationen
  - die Datei strukturierend
  - den Datenstrom ergänzend

Ausgaben der Clientsoftware

- Identisch zu SCRIPT
- Steuerzeichen werden 1:1 übernommen
- Für maximale Authentizität werden keinerlei Änderungen vorgenommen
  - Keine Umkodierung in Unicode etc.
  - Ausnahme: Escaping von Bytes mit Sonderfunktion
  - Rekonstruktion/Wiedergabe: Aufgabe der lesenden Software, die ursprüngliche Situation zu erkennen, z.B. anhand Umgebungsvariablen

Eingaben des Benutzers

- Durch vorangestelltes *shift out*, *shift out* (`0x0e 0x0e`) signalisiert
- Unveränderte Eingabebytes (Ausnahme: Escaping)
  - Druckbare Zeichen
  - Cursorbewegungen
  - Tastenkombinationen
  - Sondertasten
  - ...
- Abschluss durch *shift in* (`0x0f`)

Metainformationen

- Durch vorangestelltes *shift out* (`0x0e 0x0e`) und ein Typ-Byte signalisiert
- Festgelegte Typ-Bytes, aber in künftigen Versionen erweiterbar
  - Dateiformat-Version
  - Anfang einer neuen Sitzung
  - Umgebungsvariablen
  - Fenstergröße
  - Sprache und Zeichensatz/Encoding
  - Timinginformationen
  - ...
- Terminiert durch *shift in* (`0x0f`)

Escaping

- *shift out* (`0x0e`) und *shift in* (`0x0f`) werden durch vorangestelltes *data link escape* (`0x10`) escaped
- *data link escape* muss folglich auch escaped werden: durch sich selbst

Example

```
0x4e 0x0f 0x00 0x61 0x74 0x10
                ⇓
0x4e 0x10 0x0f 0x00 0x61 0x74 0x10 0x10
```

Zeitmessung

- Hauptschleife: Warten auf
  1. Benutzereingaben
  2. Clientausgaben
  3. Signale (Fenstergröße verändert, Client beendet)
- Vor Bearbeiten eines jedes Ereignisses wird Zeitdifferenz in Ausgabedatei geschrieben

Standards

- FORSCRIPT hält sich strikt an Unix-übergreifende Standards
  - C 99
  - POSIX.1-2001
  - System V r4
  - keine BSD-Funktionen
- Lauffähig auf Linux und NetBSD, möglicherweise auch auf anderen
- Auf OS X bislang nicht lauffähig, da dort `clock_gettime()` fehlt
  - Kann aber emuliert oder ersetzt werden

Neuerungen

- SCRIPT etwa 1997 geschrieben,
  neuere Library-Funktionen existieren:
  - `select()` statt mehreren Prozessen
  - `posix_openpt()` statt race-behaftetem Durchprobieren von PTYs
  - `clock_gettime()` mit monotonem Zeitverlauf
- FORSCRIPT ignoriert *SIGSTOP* des Childs
- `-a` mit Timing unterstützt
- Nur 1× forken ⇒ vereinfachter Code