# Design and Implementation of a Documentation Tool for Interactive Commandline Sessions

Andreas Dewald, Felix C. Freiling, Tim Weber

IMF 2011

UNIVERSITÄT MANNHEIM

# Overview

- Motivation
- You don‘t know script?
- Forscript
- Summary

# Motivation

# Motivation

- Need for documentation tools in digital investigations
  - For own documentation as well as for other experts
- Easy for a commandline session:
  - Record everything that is typed on the keyboard
  - Record everything that is sent to the screen
- `script` seems to do the job:
  - „It is useful for students who need a hardcopy record of an interactive session as proof of an assignment ...“ [script manpage]
  - But is it suitable for digital invesigation?
  - Let‘s examine that!

# You don‘t know script?

How does script work?
What is the output of script?
Problems with script

# How does script work?

- It creates a new pseudo terminal (PTY),
- Attaches itself to the master side,
- Launches a subprocess that
  - Launches the actual client application as a subchild
  - Records the applications output stream
- Parent process forwards user input to the client application

# How does script work?

- It creates a new pseudo terminal (PTY),
- Attaches itself to the master side,

```
$ 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
```

# What is the output of script?

- Script creates a typescript file, starting with the line
  `Script started on X\n`
  - `X` is human readable start-date and time
- Everything the client application sends to the terminal is appended to this file byte by byte
- If not invoked with `-q` flag, after termination of the child process, the following footer is appended:
  `Script done on Y\n`
- Using `-s`, script outputs timing data to stderr
  - tuples of delay in seconds and byte count:

```
0.725168  56
0.006549  126
0.040017  1
4.727988  1
```

- scriptreplay uses this information to replay the entire session

# What is the output of script?

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

- If not invoked with `-q` flag, after termination of the child process, the following footer is appended:
  `Script done on Y\n`
- Using `-s`, script outputs timing data to stderr
  - tuples of delay in seconds and byte count:

```
0.725168   56
0.006549   126
0.040017   1
4.727988   1
```

- scriptreplay uses this information to replay the entire session

# Problems with script

- This way, script records everything printed to the terminal
- Recall our task:
  - „Record everything that is typed on the keyboard“
  - „Record everything that is sent to the screen“
- Is that the same?
  - Usually, every typed character is also printed to the terminal
  - What about control characters?
    - TAB
    - Ctrl - C (can be circumvented with ECHOCTL bit set, which prints ^C, but does not solve the problem itself)

# Problems with script

- Data about the environment is not logged:
  - Character set, encoding, terminal type
  - Leads to a high level of uncertainty when interpreting typescript
- Timing information has to be piped to a file manually
  - Can not be stored in the same file as the typescript
- Appending to a typescript file is possible, of course, but ambiguous
  - Line „`Script started on X`“ indicates next session
  - scriptreplay has to skip this line when replaying the session
    - It does only skip the first line of a typescript file
    - After second one, timing information doesn‘t match

# Problems with script

- Only very little documentation besides man-page
  - Software used in digital investigations should provide extensive documentation

- But: Script was never meant to be a forensic tool!
- Let‘s build a forensic `script` ... `forscript`

# Forscript

Forscript
The forscript file format
Properties of the file format

# Forscript

- Forscript has the same command line user interface as script
  - Users used to script can seamlessly switch to forscript
- Forscript defines an portable and extensible file format that allows appending in a natural way and contains
  - (real) user input && application output
  - Timing information
  - Detailed metadata about the environment
- Forscript comes with a detailed user manual and documentation, following the paradigm of literate programming (LP)
  - This Paper is the program!
  - LP is already used in area of high assurance systems

# The forscript file format

- Forscript uses own file format
- Efficient combination of output and metadata within a single file
- File consists of the output stream of the client application, but includes blocks of additional data (*control chunks*) at arbitrary positions
  - Control chunks are started by a *shift out* byte (`0x0e`) and terminated by a *shift in* byte (`0x0f`)
  - A control chunk can be an *input chunk* or a *metadata chunk*
  - ...

# Input chunks

- Contain data that is sent to the client applications input stream
  - Real user input
- Arbitrary length (terminate at the already mentioned *shift in* byte)
  - If a *shift in* or *shift out* byte needs to appear in input chunk, it is escaped by prepending a *data link escape* byte (`0x10`)
  - If a *data link escape* byte needs to appear, it is doubled

**User sends:**

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

**Representations in transcript file:**

```
0x0e 0x4e 0x10 0x0f 0x00 0x61 0x74 0x10 0x10 0x0f
```

# Metadata chunks

- Contain additional information about the file or application status
  - Environment variables, terminal settings, time stamps, ...
- Contain another *shift out* byte at the beginning, followed by a *type byte*:

| binary value | type name | size |
|---|---|---|
| 0x01 | file version | 1 byte |
| 0x02 | begin of session | 10 bytes |
| 0x03 | end of session | 1 byte |
| 0x12 | environment variables | arbitrary number of C strings |
| 0x13 | locale settings | 7 C strings |
| 0x16 | delay | two 4-byte values |

- Escaping same as for input chunks

# Properties of the file format

- New chunk types can be introduced whithout breaking compatibility with older tools
- Simple escaping rules, no fixed lengthes of any type
- Switching between input and output data occurs very often
  - Format is designed to require very little storage overhead for these operations
- Converting a `forscript` file to a `script` file is basically as easy as removing everything between *shift out* and *shift in* bytes
  - Of course, respecting escaping rules

# Summary

# Summary

- Analysis of script
- Script lacks features that are crucial for digital investigations
- Design and implemented of forscript to meet the requirements of a forensic tool
- Documentation via literate programming
  - Resulting program is a single C file, written conforming to the C99, POSIX-1.2001 and System V r4 standards for portability
- forscript has been released by the third author (Tim Weber) as free software at `http://scytale.name/proj/forscript/`

# Andreas Dewald

http://pi1.informatik.uni-mannheim.de/
dewald@uni-mannheim.de


Pi1 - Laboratory for Dependable Distributed Systems

UNIVERSITÄT MANNHEIM

# Example Transcript File

*File version* chunk

- Version is `1`

```
0e 0e 01 01 0f                                  |.....          |
```

*Start of session* chunk

- time is 1266864371.072190947 = February, 22, 2010, 18:46:11 UTC
- timezone of 60 (`00 3c`) which translates to UTC+01:00

```
                0e 0e 02  4b 82 d0 f3 04 4d 8b e3  |     ...K....M..|
00 3c 0f                                           |.<.             |
```

# Example Transcript File

Environment Variabes as `name=value` pairs

- Separated by *null bytes*

```
          0e 0e 12 53 53  48 5f 41 47 45 4e 54 5f  |    ...SSH_AGENT_|
50 49 44 3d 31 36 33 30  00 47 50 47 5f 41 47 45  |PID=1630.GPG_AGE|
4e 54 5f 49 4e 46 4f 3d  2f 74 6d 70 2f 67 70 67  |NT_INFO=/tmp/gpg|
2d 4b 50 62 79 65 43 2f  53 2e 67 70 67 2d 61 67  |-KPbyeC/S.gpg-ag|
65 6e 74 3a 31 36 33 31  3a 31 00 54 45 52 4d 3d  |ent:1631:1.TERM=|
72 78 76 74 00 53 48 45  4c 4c 3d 2f 62 69 6e 2f  |rxvt.SHELL=/bin/|
62 61 73 68 00 57 49 4e  44 4f 57 49 44 3d 32 37  |bash.WINDOWID=27|
32 36 32 39 38 34 00 55  53 45 52 3d 73 63 79 00  |262984.USER=scy.|
53 53 48 5f 41 55 54 48  5f 53 4f 43 4b 3d 2f 74  |SSH_AUTH_SOCK=/t|
6d 70 2f 73 73 68 2d 64  63 74 77 4b 42 31 36 30  |mp/ssh-dctwKB160|
37 2f 61 67 65 6e 74 2e  31 36 30 37 00 50 41 54  |7/agent.1607.PAT|
48 3d 2f 68 6f 6d 65 2f  73 63 79 2f 62 69 6e 3a  |H=/home/scy/bin:|
2f 75 73 72 2f 6c 6f 63  61 6c 2f 62 69 6e 3a 2f  |/usr/local/bin:/|
75 73 72 2f 62 69 6e 3a  2f 62 69 6e 3a 2f 75 73  |usr/bin:/bin:/us|
```

UNIVERSITÄT MANNHEIM

# Example Transcript File

Delay of 0.065087679 seconds before promt is printed

```
    0e 0e 16 00 00 00 00  03 e1 28 bf 0f 1b 5d 30  | .........(...]0|
3b 73 63 79 40 62 69 6a  61 7a 3a 7e 07 1b 5b 31  |;scy@bijaz:~..[1|
3b 33 32 6d 73 63 79 1b  5b 30 3b 33 32 6d 40 1b  |;32mscy.[0;32m@.|
5b 31 3b 33 32 6d 62 69  6a 61 7a 1b 5b 31 3b 33  |[1;32mbijaz.[1;3|
34 6d 20 7e 20 1b 5b 30  3b 33 36 6d 6d 61 73 74  |4m ~ .[0;36mmast|
65 72 20 3f 20 1b 5b 31  3b 33 30 6d 30 2e 31 31  |er ? .[1;30m0.11|
20 1b 5b 30 3b 33 37 6d  31 39 3a 34 36 20 1b 5b  | .[0;37m19:46 .[|
30 3b 33 33 6d 1b 5b 31  3b 33 32 6d 24 1b 5b 30  |0;33m.[1;32m$.[0|
6d 20                                             |m                |
```

Character `e` is typed, enclosed by `0e` and `0f` (input chunk)

```
          0e 0e 16 00 00 00  01 11 67 80 66 0f 0e 65  |m ........g.f..e|
0f                                                    |.                |
```

Then the character is echoed to the terminal, took 0.0079911 seconds

```
    0e 0e 16 00 00 00 00  00 79 ef 3c 0f 65           | ........y.<.e   |
```

UNIVERSITÄT MANNHEIM

# Example Transcript File

User types `echo -l`, is echoed, too

```
                                                  0e 0e  |                ..|
16 00 00 00 00 05 b9 48  10 10 0f 0e 63 0f 0e 0e  |.......H....c...|
16 00 00 00 00 00 79 a5  09 0f 63 0e 0e 16 00 00  |......y...c.....|
00 00 0a 7d bf 1e 0f 0e  68 0f 0e 0e 16 00 00 00  |...}....h.......|
00 00 79 db 51 0f 68 0e  0e 16 00 00 00 00 0b 71  |..y.Q.h........q|
c4 94 0f 0e 6f 0f 0e 0e  16 00 00 00 00 00 79 fc  |....o.........y.|
54 0f 6f 0e 0e 16 00 00  00 02 09 89 aa a1 0f 0e  |T.o.............|
20 0f 0e 0e 16 00 00 00  00 00 79 f2 83 0f 20 0e  | .........y... .|
0e 16 00 00 00 01 2f 35  2a bc 0f 0e 2d 0f 0e 0e  |....../5*...-...|
16 00 00 00 00 00 79 bb  20 0f 2d 0e 0e 16 00 00  |......y. .-.....|
00 00 14 fb 28 4d 0f 0e  6c 0f 0e 0e 16 00 00 00  |....(M..l.......|
00 00 7a 01 3d 0f 6c 0e  0e 16 00 00 00 00 2b 64  |..z.=.l.......+d|
b7 45 0f                                          |.E.             |
```

# Example Transcript File

Typing `l` was a mistake, user presses the backspace key (ASCII 127)

```
0e 7f 0f                                                |   ...           |
```

After usual delay, the following is sent to the terminal:

- ASCII backspace character (`08`) to position the cursor on the `l`
- ANSI code CSI K, (`1b 5b 4b`), which causes terminal to make all characters at or right of the cursors position disappear

```
                  0e 0e  16 00 00 00 00 00 79 c2  |       ........y.|
7e 0f 08 1b 5b 4b                                 |~...[K           |
```

# Example Transcript File

For even more details, please refer to the paper.