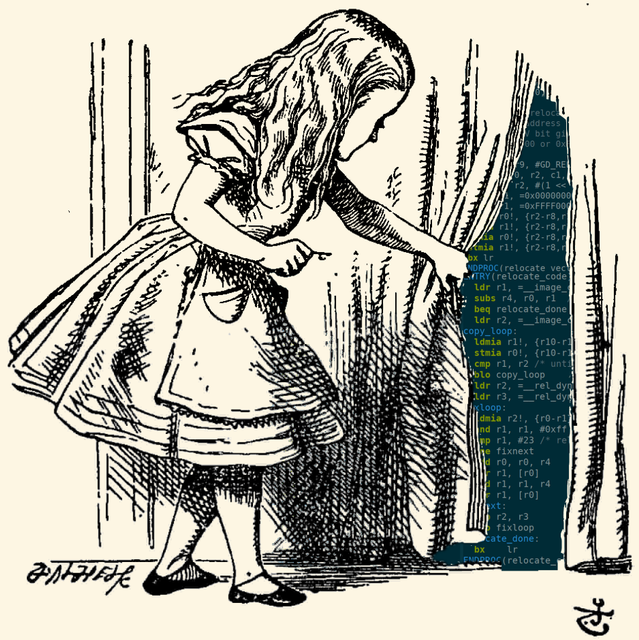
Bushwhacking your way around a bootloader

Rebecca ".bx" Shapiro

2018.06.16



Tools and techniques for traversing treacherous code bases

-or- How I managed to develop understanding of U-Boot

```
[user@work ~]$ cd thesis/presentations/recon
```

Meet Das U-Boot bootloader

```
[user@boot-dev ~]$ cloc u-boot/
   13518 text files.
   12700 unique files.
    4701 files ignored.

github.com/AlDanial/cloc v 1.76  T=4.02 s (2196.7 files/s, 504571.1 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C                             3958         177722         230606         911861
C/C++ Header                  3540          64684         108111         429854
Assembly                       236           5927          10632          24037
Python                         119           4380           9180          12486
Perl                             6           1660           1346           9850
make                           911           2263           4664           8500
Bourne Shell                    32            427            626           2164
C++                              1            233             58           1588
yacc                             2            169             75           1076
Glade                            1             58              0            603
lex                              2             98             41            539
NAnt script                      1             91              0            367
YAML                             1             13             25            347
Bourne Again Shell               3             75             66            316
Markdown                         1             80              0            283
DOS Batch                        3             20              0            176
CSS                              2             24             10             90
Kermit                           3              4             20             83
Tcl/Tk                           1              5              5             28
sed                              2              1             27             24
INI                              2              3              0             14
XSLT                             1              0              1              9
-------------------------------------------------------------------------------
SUM:                          8828         257937         365493        1404295
-------------------------------------------------------------------------------
[user@boot-dev ~]$
```

Meet Das U-Boot bootloader

```
[user@boot-dev ~]$ cloc u-boot/
   13518 text files.
   12700 unique files.
    4701 files ignored.

github.com/AlDanial/cloc v 1.76  T=4.02 s (2196.7 files/s, 504571.1 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C                             3958         177722         230606         911861
YAML                             1             13             25            347
Bourne Again Shell               3             75             66            316
Markdown                         1             80              0            283
DOS Batch                        3             20              0            176
CSS                              2             24             10             90
Kermit                           3              4             20             83
Tcl/Tk                           1              5              5             28
sed                              2              1             27             24
INI                              2              3              0             14
XSLT                             1              0              1              9
-------------------------------------------------------------------------------
SUM:                          8828         257937         365493        1404295
-------------------------------------------------------------------------------
[user@boot-dev ~]$
```

"Only" 111 MB to build bootloaders for resource-constrained systems

```
[user@boot-dev ~]$ make -C u-boot distclean
make: Entering directory '/home/user/u-boot'
make: Leaving directory '/home/user/u-boot'
[user@boot-dev ~]$ rm -rf u-boot/.git
[user@boot-dev ~]$ du -sh u-boot/
111M  u-boot/
```

The existential question.

Overall research goals

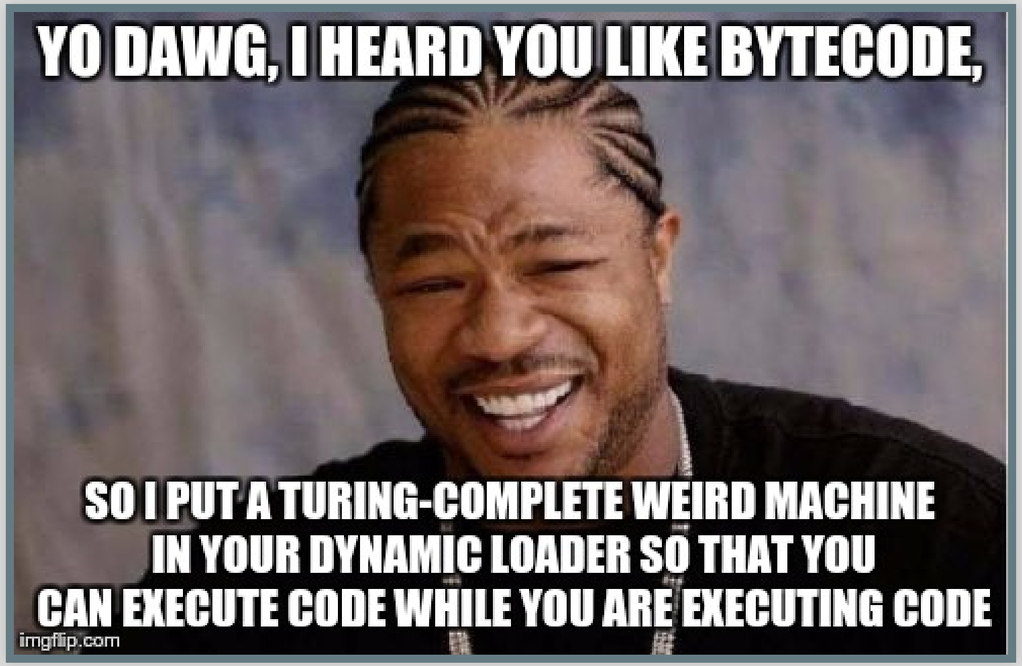
1. **Identify weaknesses** underlying (boot)loader security
2. **Develop (boot)loader hardening techniques** that:
   - are **realistic**
   - lend themselves to **formal reasoning**
   - can be **retroactively applied** to **existing** loaders
3. Demonstrate technique **feasibility**

The existential question.

* *Flashback to 2012* *

The Turing-complete ELF-metadata weird machine

a brainfuck to ELF-metadata compiler @ https://github.com/bx/elf-bf-tools



(ELF is *NIX's file format for executables, libraries, etc.)

AS SEEN ON YouTube DEF CON, 29C3, USENIX WOOT, PoC || GTFO

([1], [2], [3], [4])

The existential question.

Overall research goals

1. **Identify weaknesses** underlying (boot)loader security
2. **Develop (boot)loader hardening techniques** that:
   - are **realistic**
   - lend themselves to **formal reasoning**
   - can be **retroactively applied** to **existing** loaders
3. Demonstrate technique **feasibility**

The existential question.

* The ***ultimate*** goal *

1. **Identify**
2. **Develop**
   - are r
   - lend
   - can b
3. Demonstr

g loaders

The existential question.

Overall research goals

1. **Identify weaknesses** underlying (boot)loader security
2. **Develop (boot)loader hardening techniques** that:
   - are **realistic**
   - lend themselves to **formal reasoning**
   - can be **retroactively applied** to **existing** loaders
3. Demonstrate technique **feasibility**

This talk for those in a hurry

- ~~Introduce goals and case study~~
- Generalizing about bootloaders
- Debugging U-Boot (as according to U-Boot)
- My instrumentation toolsuite
  - An attempt at something better
- Techniques for identifying code <–> data
- A test drive through a simple example

Properties of a bootloader

- They load and prepare images for execution
- They initialize resources/hardware
- Sometimes they self-relocate

In general, bootloaders

- Allocate **non-overlapping** addresses
- Manage address **alignment requirements**
- **Prepare memory map** for target
  - **Load** target image into memory
  - **Patch** (link) loaded images
- **Extract** and **enforce** requirements and restrictions imposed by both resources and target


Let's begin exploring U-Boot

How to debug a bootloader

(with GDB and QEMU)

```
(gdb) target remote | qemu-system-arm -gdb stdio -M beaglexm -sd sd.img -S
Remote debugging using | qemu-system-arm -gdb stdio -M beaglexm -sd sd.img -S
0x40014000 in ?? ()
(gdb) file u-boot-spl
Reading symbols from u-boot-spl...done.
(gdb) break jump_to_image_no_args
Breakpoint 1 at 0x4020127c: file arch/arm/cpu/armv7/omap-common/boot-common.c,
line 229.
(gdb) c
Continuing.
```

How to debug a bootloader

(with GDB and QEMU)

```
(gdb) target remote | qemu-system-arm -gdb stdio -M beaglexm -sd sd.img -S
Remote debugging using | qemu-system-arm -gdb stdio -M beaglexm -sd sd.img -S
0x40014000 in ?? ()
(gdb) file u-boot-spl
Reading symbols from u-boot-spl...done.
(gdb) break jump_to_image_no_args
Breakpoint 1 at 0x4020127c: file arch/arm/cpu/armv7/omap-common/boot-common.c,
line 229.
(gdb) c
Continuing.
Breakpoint 1, jump_to_image_no_args (spl_image=0x80000000 <spl_image>)
    at arch/arm/cpu/armv7/omap-common/boot-common.c:229
(gdb) si
0x80100000 in ?? ()
```

How to debug a bootloader

(with GDB and QEMU)

```
(gdb) target remote | qemu-system-arm -gdb stdio -M beaglexm -sd sd.img -S
Remote debugging using | qemu-system-arm -gdb stdio -M beaglexm -sd sd.img -S
0x40014000 in ?? ()
(gdb) file u-boot-spl
Reading symbols from u-boot-spl...done.
(gdb) break jump_to_image_no_args
Breakpoint 1 at 0x4020127c: file arch/arm/cpu/armv7/omap-common/boot-common.c,
line 229.
(gdb) c
Continuing.
Breakpoint 1, jump_to_image_no_args (spl_image=0x80000000 <spl_image>)
    at arch/arm/cpu/armv7/omap-common/boot-common.c:229
(gdb) si
0x80100000 in ?? ()
(gdb) file u-boot
A program is being debugged already.
Are you sure you want to change the file? (y or n) y
Load new symbol table from "u-boot"? (y or n) y
Reading symbols from u-boot...done.
Error in re-setting breakpoint 1: Function "jump_to_image_no_args" not
defined
```

How to debug a bootloader

(with GDB and QEMU)

```
(gdb) target remote | qemu-system-arm -gdb stdio -M beaglexm -sd sd.img -S
Remote debugging using | qemu-system-arm -gdb stdio -M beaglexm -sd sd.img -S
0x40014000 in ?? ()
(gdb) file u-boot-spl
Reading symbols from u-boot-spl...done.
(gdb) break jump_to_image_no_args
Breakpoint 1 at 0x4020127c: file arch/arm/cpu/armv7/omap-common/boot-common.c,
line 229.
(gdb) c
Continuing.
Breakpoint 1, jump_to_image_no_args (spl_image=0x80000000 <spl_image>)
    at arch/arm/cpu/armv7/omap-common/boot-common.c:229
(gdb) si
0x80100000 in ?? ()
(gdb) file u-boot
A program is being debugged already.
Are you sure you want to change the file? (y or n) y
Load new symbol table from "u-boot"? (y or n) y
Reading symbols from u-boot...done.
Error in re-setting breakpoint 1: Function "jump_to_image_no_args" not
defined.
(gdb) break relocate_done
Breakpoint 2 at 0x801020b4: file arch/arm/lib/relocate.S, line 134.
(gdb) c
Continuing
```

How to debug a bootloader

(with GDB and QEMU)

```
(gdb) target remote | qemu-system-arm -gdb stdio -M beaglexm -sd sd.img -S
Remote debugging using | qemu-system-arm -gdb stdio -M beaglexm -sd sd.img -S
0x40014000 in ?? ()
(gdb) file u-boot-spl
Reading symbols from u-boot-spl...done.
(gdb) break jump_to_image_no_args
Breakpoint 1 at 0x4020127c: file arch/arm/cpu/armv7/omap-common/boot-common.c,
line 229.
(gdb) c
Continuing.
Breakpoint 1, jump_to_image_no_args (spl_image=0x80000000 <spl_image>)
    at arch/arm/cpu/armv7/omap-common/boot-common.c:229
(gdb) si
0x80100000 in ?? ()
(gdb) file u-boot
A program is being debugged already.
Are you sure you want to change the file? (y or n) y
Load new symbol table from "u-boot"? (y or n) y
Reading symbols from u-boot...done.
Error in re-setting breakpoint 1: Function "jump_to_image_no_args" not
defined.
(gdb) break relocate_done
Breakpoint 2 at 0x801020b4: file arch/arm/lib/relocate.S, line 134.
(gdb) c
Continuing.
Breakpoint 2, relocate_done () at arch/arm/lib/relocate.S:134
134   in arch/arm/lib/relocate.S
(gdb) break board_init_r
Breakpoint 3 at 0x80104e7c: file common/board_r.c, line 957.
```

How to debug a bootloader

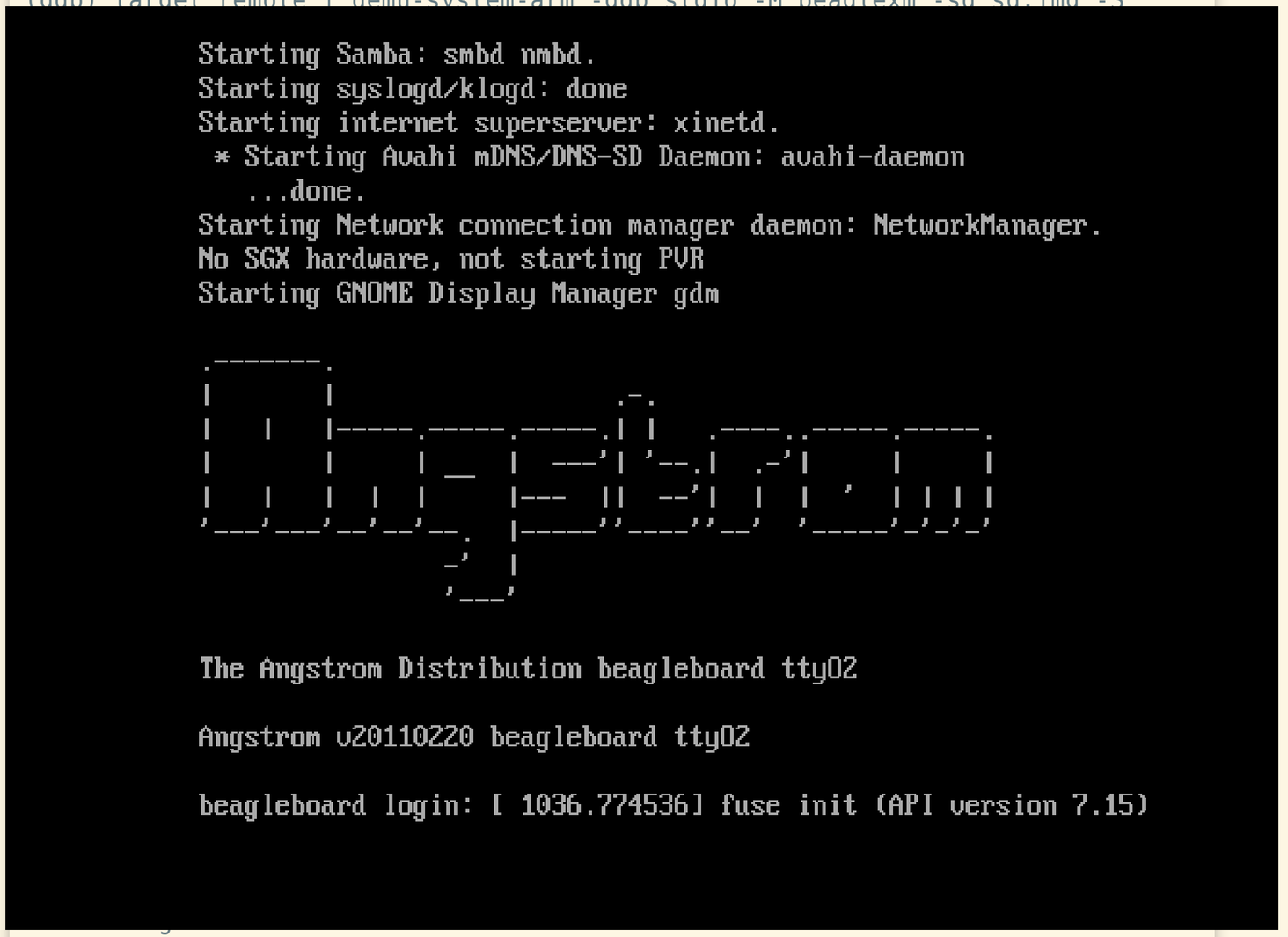
(with GDB and QEMU)

```
(gdb) target remote | qemu-system-arm -gdb stdio -M beaglexm -sd sd.img -S
Remote debugging using | qemu-system-arm -gdb stdio -M beaglexm -sd sd.img -S
0x40014000 in ?? ()
(gdb) file u-boot-spl
Reading symbols from u-boot-spl...done.
(gdb) break jump_to_image_no_args
Breakpoint 1 at 0x4020127c: file arch/arm/cpu/armv7/omap-common/boot-common.c,
line 229.
(gdb) c
Continuing.
Breakpoint 1, jump_to_image_no_args (spl_image=0x80000000 <spl_image>)
    at arch/arm/cpu/armv7/omap-common/boot-common.c:229
(gdb) si
0x80100000 in ?? ()
(gdb) file u-boot
A program is being debugged already.
Are you sure you want to change the file? (y or n) y
Load new symbol table from "u-boot"? (y or n) y
Reading symbols from u-boot...done.
Error in re-setting breakpoint 1: Function "jump_to_image_no_args" not
defined.
(gdb) break relocate_done
Breakpoint 2 at 0x801020b4: file arch/arm/lib/relocate.S, line 134.
(gdb) c
Continuing.
Breakpoint 2, relocate_done () at arch/arm/lib/relocate.S:134
134   in arch/arm/lib/relocate.S
(gdb) break board_init_r
Breakpoint 3 at 0x80104e7c: file common/board_r.c, line 957.
(gdb) c
Continuing.
^Comap_gpmc_write: bad SDRAM idle mode 3
omap_i2c_write: Bad register 0x0000cc
```

How to debug a bootloader

(with GDB and QEMU)

```
(gdb) target remote | qemu-system-arm -gdb stdio -M beaglexm -sd sd.img -S
```




```
^Comap_gpmc_write: bad SDRAM idle mode 3
omap_i2c_write: Bad register 0x0000cc
```

How to debug a self-relocating bootloader

As according to U-Boot

(quoted from doc/README.arm-relocation)

start debugger

```
[hs@pollux u-boot]$ arm-linux-gdb u-boot
```

connect to target

```
(gdb) target remote localhost:4444
```

execute until relocation complete

```
(gdb) break _relocation_done
(gdb) c
```

discard symbol-file

```
(gdb) symbol-file
Discard symbol table from `/home/hs/celf/u-boot/u-boot'? (y or n) y
No symbol file now.
```

load new symbol table at relocated address

```
gdb) add-symbol-file u-boot 0x8ff08000
add symbol table from file "u-boot" at
  .text_addr = 0x8ff08000
(y or n) y
Reading symbols from /home/hs/celf/u-boot/u-boot...done.
```

How to debug a self-relocating bootloader

As according to U-Boot

(quoted from doc/README.arm-relocation)

start debugger

```
[hs@pollux u-boot]$ arm-linux-gdb u-boot
```

connect to target

```
(gdb) target remote localhost:4444
```

execu ete

```
(gdb) break _relocatio
(gdb) c
```

```
(gdb) symbol-file
Discard symbol table f                    y or n) y
No symbol file now.
```

load new s address

```
gdb) add-symbol-file u
add symbol table from
  .text_addr = 0x8ff08
(y or n) y
Reading symbols from /home/hs/celf/u-boot/u-boot...done.
```

"Demystifying" magic numbers

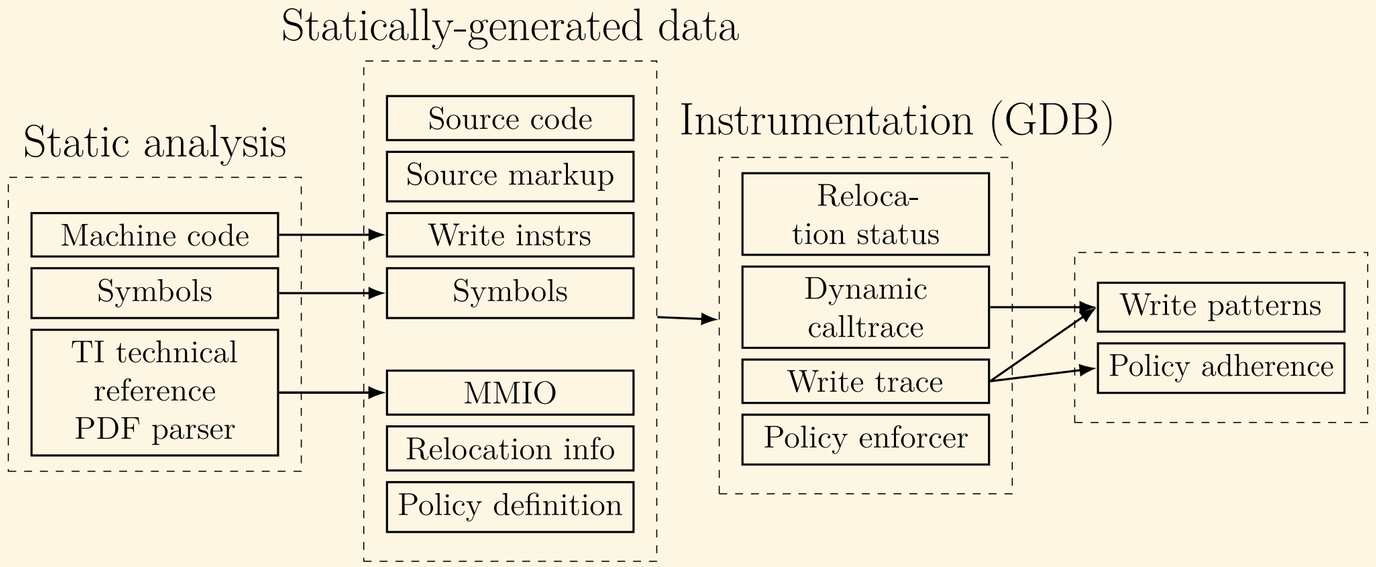
```
Program received signal SIGSTOP, Stopped (signal).
0x8ff17f18 in serial_getc () at serial_mxc.c:192
192       while (__REG(UART_PHYS + UTS) & UTS_RXEMPTY);
(gdb)

add-symbol-file u-boot 0x8ff08000
                ^^^^^^^^^^
                get this address from u-boot bdinfo command
                or get it from gd->relocaddr in gdb

 => bdinfo
rch_number = XXXXXXXXXX
boot_params = XXXXXXXXXX
DRAM bank   = XXXXXXXXXX
-> start    = XXXXXXXXXX
-> size     = XXXXXXXXXX
ethaddr     = XXXXXXXXXX
ip_addr     = XXXXXXXXXX
baudrate    = XXXXXXXXXX
TLB addr    = XXXXXXXXXX
relocaddr   = 0x8ff08000
           ^^^^^^^^^^
reloc off   = XXXXXXXXXX
irq_sp        = XXXXXXXXXX
sp start    = XXXXXXXXXX
FB base     = XXXXXXXXXX

or interrupt execution by any means and re-load the symbols at the location
specified by gd->relocaddr -- this is only valid after board_init_f.
```

Can we do better?

Toolsuite overview

Featuring:

- Static and dynamic analysis
- Instrumentation (via GDB)
  - Mediates **all** memory writes
- Language to express (and enforce) memory write polices
- 32-bit ARM only (for now)



Source code at http://typedregions.com

Relocation reconnaissance

Tools at
https://typedregions.com

Identifying relocation phases

Block write operation

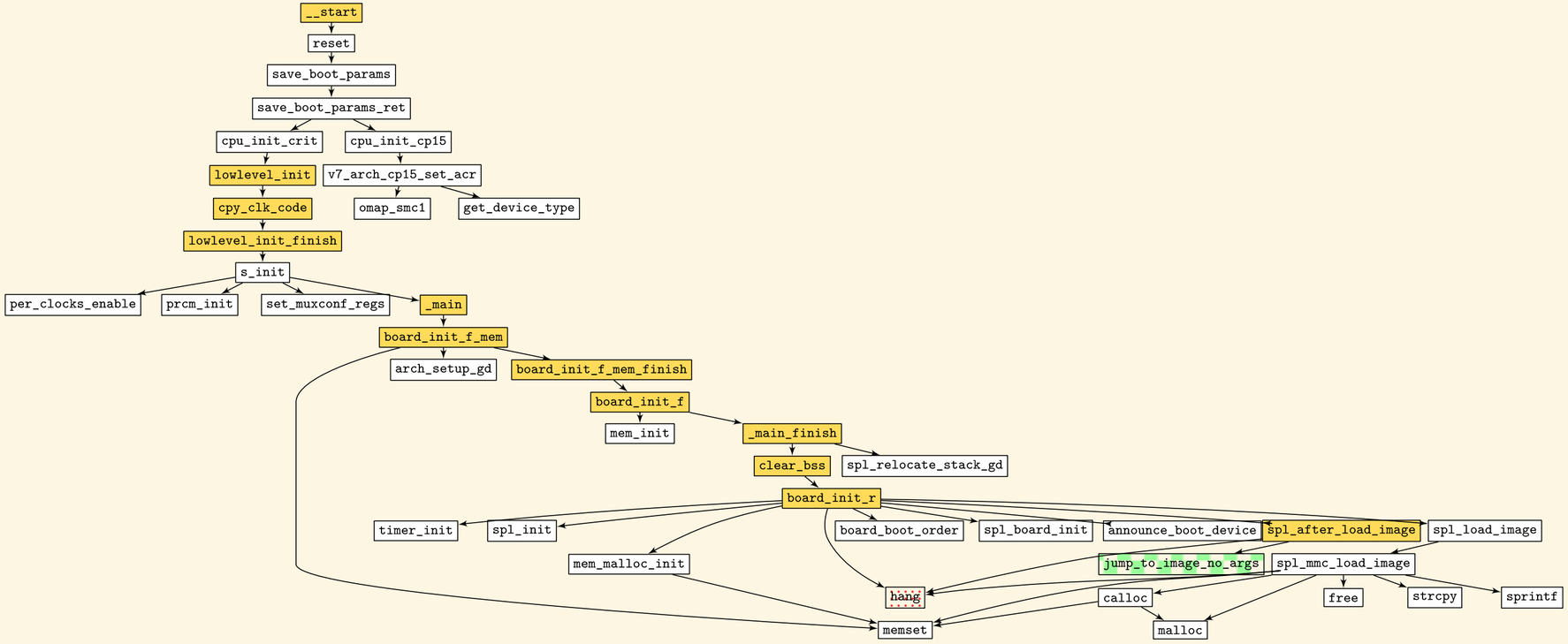
($ip, offset in image, destination, size, call stack)

```
[262144x] memset @ pc=0x40208a3e wrote 1048576 bytes to 0x80208000 str
[49233x] clbss_l @ pc=0x402025c4 wrote 196932 bytes to 0x80000000 strcc<-- Zero BSS
[128x] mmc_read_data @ pc=0x40206cfa wrote 512 bytes to 0x4020f2c0
str.w
[128x] mmc_read_data @ pc=0x40206cfa wrote 512 bytes to 0x4020f2c0     <-- Read target image to memory
str.w
[128x] mmc_read_data @ pc=0x40206cfa wrote 512 bytes to 0x80104bc0     <-- Copy data to stack
str.w                                                                 <-- Relocate "go_to_speed" function
[128x] mmc_read_data @ pc=0x40206cfa wrote 512 bytes to 0x80104dc0     <-- Copy partition data
str.w
[83x] memset @ pc=0x40208a3e wrote 332 bytes to 0x80208038 str         <-- Relocate bookkeeping data
[9x] next2 @ pc=0x402009c4 wrote 288 bytes to 0x4020f840 stmia         <-- Function call/stack
[54x] memset @ pc=0x40208a3e wrote 216 bytes to 0x4020fe10 str
[30x] memcpy @ pc=0x40208a72 wrote 120 bytes to 0x800200c0 str
[16x] memcpy @ pc=0x40208a72 wrote 64 bytes to 0x4020f118 str
[1x] omap_smc1 @ pc=0x40200970 wrote 40 bytes to 0x40200234 push
```

# write operations	~400,000
# block writes	10,000

U-Boot's static call graph (for reference)

Generated using IDA Pro (and then simplified by hand)



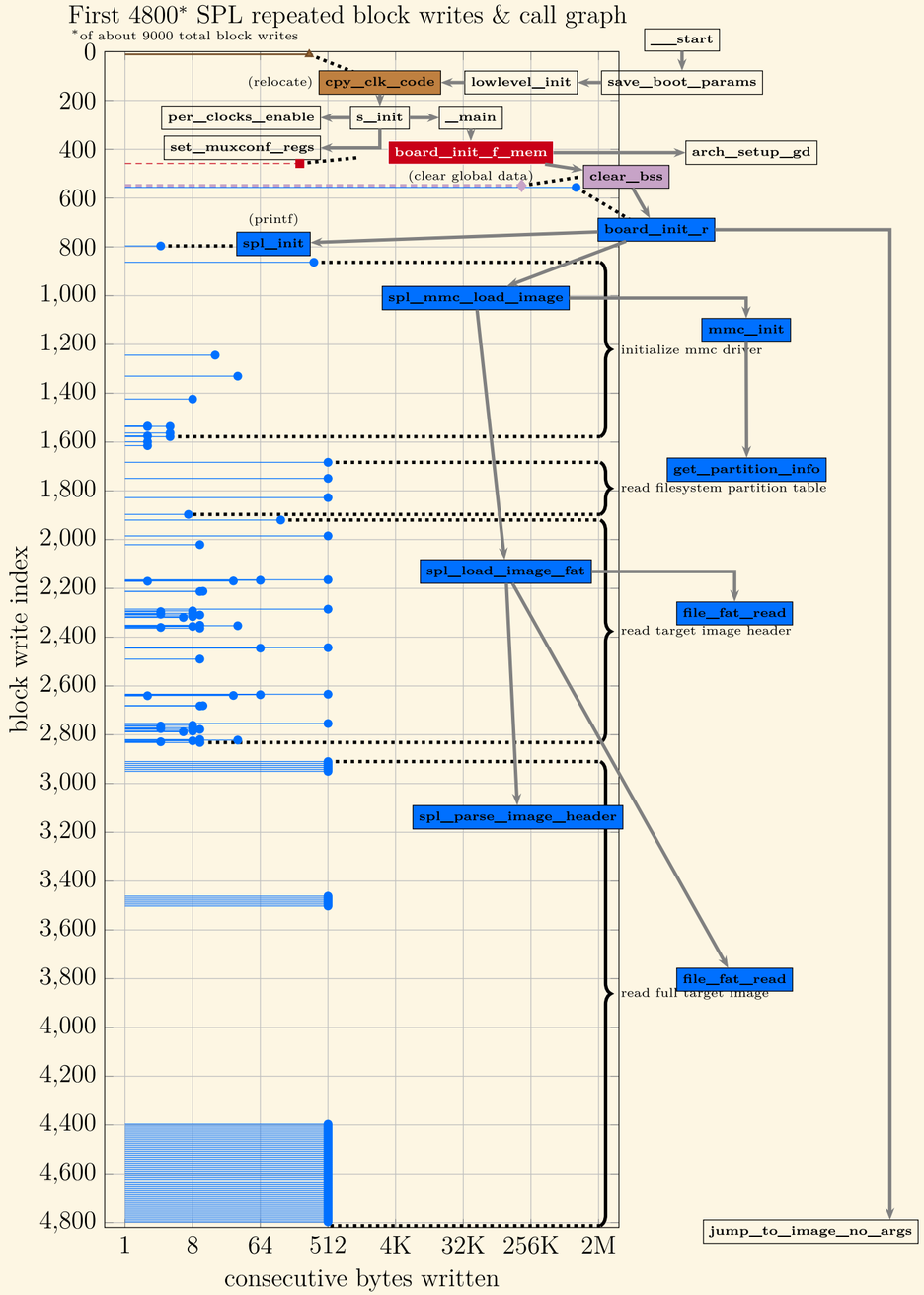
Calltrace of successful U-Boot execution

Example output from U-Boot SPL execution

```
> save_boot_params {arch/ahrm/cpu/armv7/omap-common/lowlevel_init.S::22}
```

This is a javascript-enabled demo

Calltrace and write data combined



Other magic numbers

```
switch(beagle_revision()) {
 case REVISION_C4:
  if (identify_xm_ddr() == NUMONYX_MCP) {
    __raw_writel(0x4, SDRC_CS_CFG); /* 512MB/bank */
    __raw_writel(SDP_SDRC_MDCFG_0_DDR_NUMONYX_XM, SDRC_MCFG_0);
```

```
argument = 0x0000 << 16;
err = mmc_send_cmd(MMC_CMD55, argument, resp);
if (err == 1) {
 mmc_card_cur->card_type = SD_CARD;
} else {
 mmc_card_cur->card_type = MMC_CARD;
```

Other magic numbers

```
switch(beagle_revision()) {
 case REVISION_C4:
  if (identify_xm_ddr() == NUMONYX_MCP) {
    __raw_writel(0x4, SDRC_CS_CFG); /* 512MB/bank */
    __raw_writel(SDP_SDRC_MDCFG_0_DDR_NUMONYX_XM, SDRC_MCFG_0);
```

```
argument = 0x0000 << 16;
err = mmc_send_cmd(MMC_CMD55, argument, resp);
if (err == 1) {
 mmc_card_cur->card_type = SD_CARD;
} else {
 mmc_card_cur->card_type = MMC_CARD;
```

...and so I built a PDF scraper

```
for o in page:
    if cls.is_not_in_table_bounds(tablestart, tableend, o):
        continue
    if isinstance(o, layout.LTRect) and (abs(o.bbox[1] - o.bbox[3]) < 1.5):  # is a line
        # if left edge of line is not near left edge of name col_info
        if abs(namecol.bbox[0] - (o.bbox[0] + offset)) > 5:
            continue
        odiff = cls.vertical_offset(namecol, o)
        if (odiff > 0) and (closest_rect is None):
            closest_rect = o
```

Other magic numbers

```
switch(beagle_revision()) {
 case REVISION_C4:
  if (identify_xm_ddr() == NUMONYX_MCP) {
    __raw_writel(0x4, SDRC_CS_CFG); /* 512MB/bank */
    __raw_writel(SDP_SDRC_MDCFG_0_DDR_NUMONYX_XM, SDRC_MCFG_0);
```

```
argument = 0x0000 << 16;
err = mmc_send_cmd(MMC_CMD55, argument, resp);
if (err == 1) {
 mmc_card_cur->card_type = SD_CARD;
} else {
 mmc_card_cur->card_type = MMC_CARD;
```

...and so I built a PDF scraper

```
for o in page:
    if cls.is_not_in_table_bounds(tablestart, tableend, o):
        continue
    if isinstance(o, layout.LTRect) and (abs(o.bbox[1] - o.bbox[3]) < 1.5):  # is a line
        # if left edge of line is not near left edge of name col_info
        if abs(namecol.bbox[0] - (o.bbox[0] + offset)) > 5:
            continue
        odiff = cls.vertical_offset(namecol, o)
        if (odiff > 0) and (closest_rect is None):
            closest_rect = o
```

that transforms

Table 10-28. GPMC Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
GPMC_REVISION	R	32	0x0000 0000	0x6E00 0000
GPMC_SYSCONFIG	RW	32	0x0000 0010	0x6E00 0010
GPMC_SYSSTATUS	R	32	0x0000 0014	0x6E00 0014
GPMC_IRQSTATUS	RW	32	0x0000 0018	0x6E00 0018

Other magic numbers

```
switch(beagle_revision()) {
 case REVISION_C4:
  if (identify_xm_ddr() == NUMONYX_MCP) {
    __raw_writel(0x4, SDRC_CS_CFG); /* 512MB/bank */
    __raw_writel(SDP_SDRC_MDCFG_0_DDR_NUMONYX_XM, SDRC_MCFG_0);
```

```
argument = 0x0000 << 16;
err = mmc_send_cmd(MMC_CMD55, argument, resp);
if (err == 1) {
 mmc_card_cur->card_type = SD_CARD;
} else {
 mmc_card_cur->card_type = MMC_CARD;
```

...and so I built a PDF scraper

```
for o in page:
    if cls.is_not_in_table_bounds(tablestart, tableend, o):
        continue
    if isinstance(o, layout.LTRect) and (abs(o.bbox[1] - o.bbox[3]) < 1.5):  # is a line
        # if left edge of line is not near left edge of name col_info
        if abs(namecol.bbox[0] - (o.bbox[0] + offset)) > 5:
            continue
        odiff = cls.vertical_offset(namecol, o)
        if (odiff > 0) and (closest_rect is None):
            closest_rect = o
```

that transforms

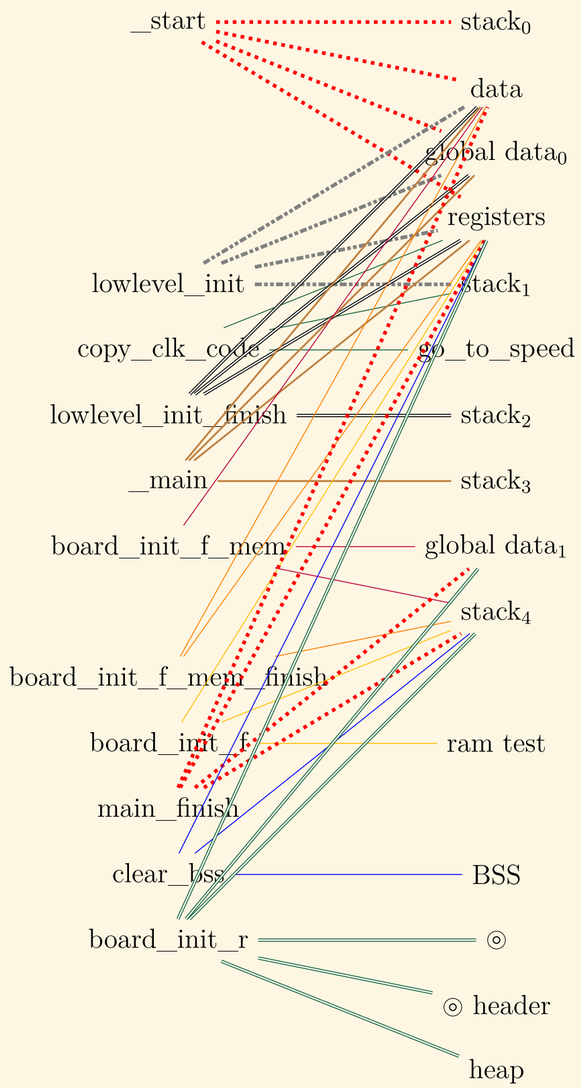
Table 10-28. GPMC Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
GPMC_REVISION	R	32	0x0000 0000	0x6E00 0000
GPMC_SYSCONFIG	RW	32	0x0000 0010	0x6E00 0010
GPMC_SYSSTATUS	R	32	0x0000 0014	0x6E00 0014
GPMC_IRQSTATUS	RW	32	0x0000 0018	0x6E00 0018

into

```
CM_FCLKEN_IVA2,RW,W,32,0x00000000,0x48004000,Table 3-93. IVA2_CM Register Summary
CM_CLKEN_PLL_IVA2,RW,W,32,0x00000004,0x48004004,Table 3-93. IVA2_CM Register Summary
CM_IDLEST_IVA2,R,C,32,0x00000020,0x48004020,Table 3-93. IVA2_CM Register Summary
CM_IDLEST_PLL_IVA2,R,C,32,0x00000024,0x48004024,Table 3-93. IVA2_CM Register Summary
CM_AUTOIDLE_PLL_IVA2,RW,W,32,0x00000034,0x48004034,Table 3-93. IVA2_CM Register Summary
```

Devising code and data relationships



A simpler example

Hello, world!

```c
#include <stdio.h>
#include <string.h>

#define SIZE 512
char memory[SIZE];

void do_nothing() {}

void say_hello() {
  printf("Hello, world\n");
}

void modify_memory() {
  for (int i = 0; i <= SIZE; i++) {
    memory[i] = 'A';
  }
}

int main(int argc, char *argv[]) {
  say_hello();
  modify_memory();
  do_nothing();
  return 0;
}
```

(Built with -static,-no-pie)

Calltrace of Hello, world!

```
> _libcstartmain {arm-linux-gnueabihf-glibc/src/glibc-2.27/csu/libc-start.c::137}
```

This is a javascript-enabled demo

Checking for unexpected writes

1. Configure instrumentation suite to work with sample
2. Run sample through instrumentation suite's **static analysis**
3. Construct policy to target "substages" and memory regions of interest
4. Import policy
5. Execute **dynamic analysis**
6. Use **post-analysis** to highlight policy violations

Checking for unexpected writes

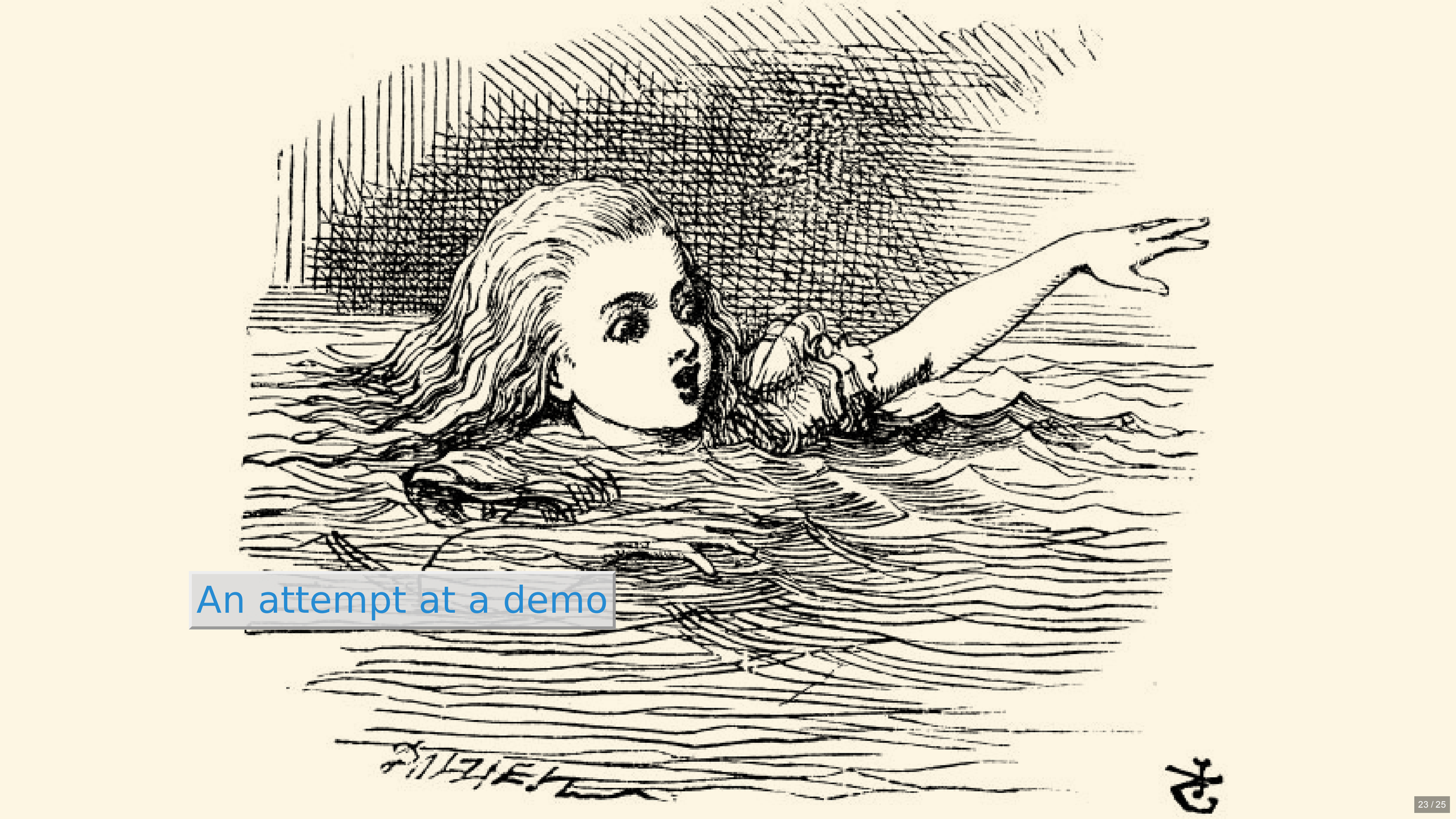
1. Configure instrumentation suite to work with sample
2. Run sample through instrumentation suite's **static analysis**
3. Construct policy to target "substages" and memory regions of interest
4. Import policy
5. Execute **dynamic analysis**
6. Use **post-analysis** to highlight policy violations

Region definitions

```
regions:
 ALL:
  type: "global"
  addresses: [0, 0xFFFFFFFF]
  subregions:
   buffer:
     type: "global"
     addresses: [0x8bb98, 0x8bd98]
   stack:
     type: "stack"
     addresses: [0xfffe0000, 0xffff0000]
   ro:
     type: "global"
     addresses: [[0x0, 0x8bb898],
       [0x8bd98,0xfffe0000],
       [0xffff0000,0xffffffff]]

stagename: "_single"
```

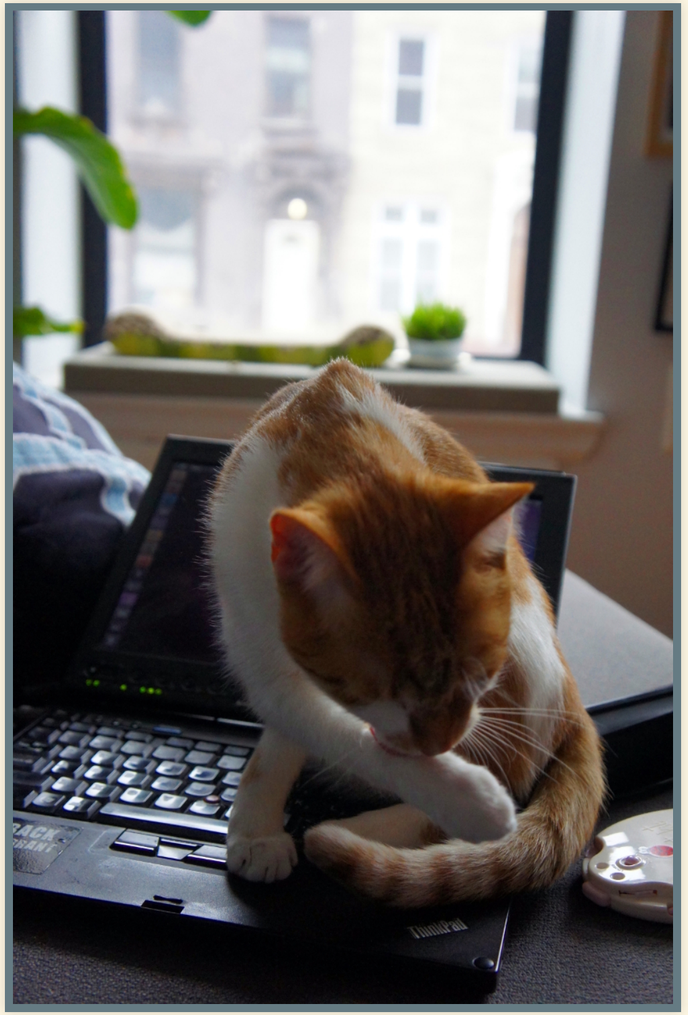
Substage/policy definitions

```
_start:
  substage_type: "bookkeeping"
  new_regions: ["ALL.buffer", "ALL.stack",
"ALL.ro"]
  reclassified_regions:
    ALL.ro: "global"
    ALL.stack: "stack"
    ALL.buffer: "global"

main:
  substage_type: "bookkeeping"

modify_memory:
  substage_type: "bookkeeping"
  reclassified_regions:
    ALL.ro: "readonly"

do_nothing:
  substage_type: "bookkeeping"
  reclassified_regions:
    ALL.ro: "global"
```



An attempt at a demo

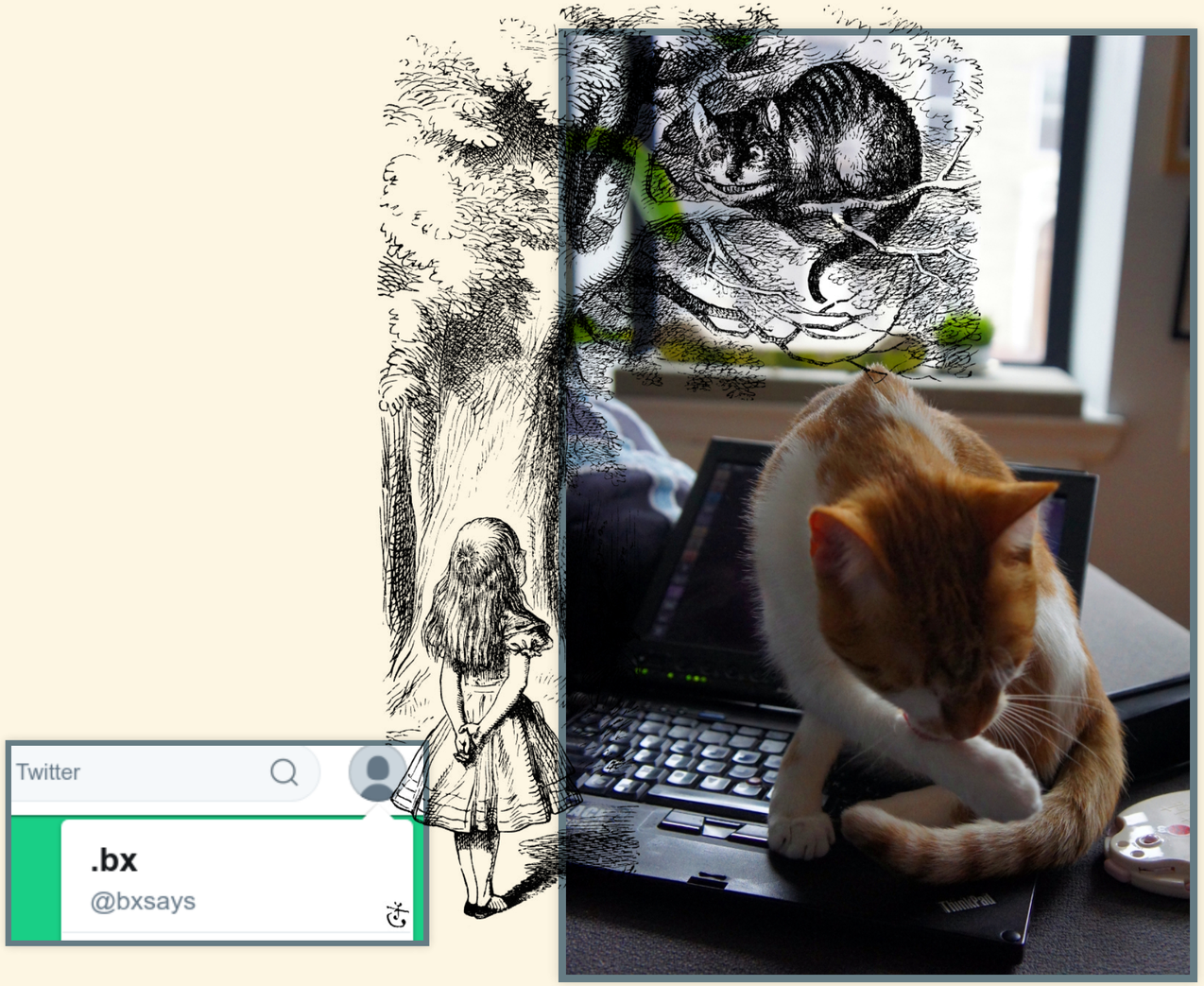
Thank you



More details and tools at: http://typedregions.com

Many thanks **Sergey Bratus**, my PhD advisor

Thank you



bx@narfindustries.com

More details and tools at: http://typedregions.com

Many thanks **Sergey Bratus**, my PhD advisor

References

See also
typedregions.com

[1]. R. .. Shapiro, “The care and feeding of weird machines found in executable metadata,” in *Chaos communication congress*, 2012.

[2]. R. Shapiro, S. Bratus, and S. W. Smith, “Weird machines in ELF: A spotlight on the underappreciated metadata,” in *Workshop on offensive technologies*, Washington, D.C., 2013.

[3]. R. .. Shapiro, “Returning from ELF to libc,” in *POC or GTFO*, vol. 0x0, 2013.

[4]. R. .. Shapiro, “Calling putchar() from an ELF weird machine,” in *POC or GTFO*, vol. 0x02, 2013.

[5]. J. Bangert, S. Bratus, R. Shapiro, and S. W. Smith, “The page-fault weird machine: Lessons in instruction-less computation,” in *Workshop on offensive technologies*, Washington, D.C., 2013.

[6]. J. Bangert, S. Bratus, R. Shapiro, M. E. Locasto, J. Reeves, S. W. Smith, and A. Shubina, “ELFbac: Using the loader format for intent-level semantics and fine-grained protection,” Dartmouth College, Computer Science, Hanover, NH, TR2013-727, May 2013.

[7]. S. Bratus, M. E. Locasto, and M. L. Patterson, “Exploit programming: From buffer overflows to "weird machines” and theory of computation,” in *USENIX; login*, 2011, pp. 13–21.

[8]. S. Bratus and J. Bangert, “ELFs are dorky, elves are cool,” in *POC or GTFO*, vol. 0x0, 2013.

[9]. J. Oakley and S. Bratus, “Exploiting the hard-working DWARF: Trojan and exploit techniques with no native executable code,” in *Workshop on offensive technologies*, 2011, p.