Kernel ABI Monitoring and Toolchain Support

guillermo.e.martinez@oracle.com

Linux Plumbers Conference 2022: Toolchains Track

September 14, 2022

ABI

- Interface between two binary programs.
- In the kernel:
  - Kernel⇔Kernel.
  - Kernel⇔Module.

Libabigail

Libabigail is a *ABI Generic Analysis and Instrumentation* library, providing a set of tools to analyse and looks for changes in the ABI, it works with ELF binary files to:

- Extract ABI-relevant type information: functions, variables.
- Serialize & de-serialize ABI information from binary files in a XML format file.
- Compare and summarize ABI changes operating on either:
  - ELF binaries files (also specifying a path).
  - RPMs, DEBs and compress format files.
- Useful to detects API/ABI breaks in libraries.
- Verify backward compatibility with new library version.

Info handled by Libabigail

libabigail uses as input:

- ELF objects (`EXEC, DYN, REL`), with debug format:
  - `DWARF`.
  - And now it supports: `CTF`!.
- ELF files without debug information, just using the symtab.
- Linux Kernel image: `vmlinux` and modules: `.ko's`

CTF: Compact C Type Format

- Associate symbols and its types.
- Lightweight debug format. So it is embedded in ELF objects.
- Debug information is organized in `CTF archives` and `dictionaries`.
  - CTF archive: vmlinux.ctfa.
- `-gctf` generates .ctf section in ELF files.
- Implementation in `libctf`.
- CTF v3: `ctf-specification`.

Dumping CTF Info

```
# GNU poke/(objcopy, objdump) can be used to looks
# inside of CTF archive (vmlinux.ctfa) in a human mode.
#
# File vmlinux.ctfa is one of the main inputs
# used for CTF reader in libabigail tools.
$ objcopy --add-section .ctf=vmlinux.ctfa vmlinux vmlinux.elf
$ objdump --ctf --ctf-parent=shared_ctf vmlinux.elf

...
CTF archive member: 3c574_cs:

  Header:
    Magic number: 0xdff2
    ...
    Parent name: shared_ctf

  Function objects:

  Variables:
  ...
  _printk -> 0x855: (kind 5) int (*) (const char *, ...) (aligned at 0x8)
  ...
```

Listing 1: dump vmlinux.ctfa

CTF in Libabigail ...

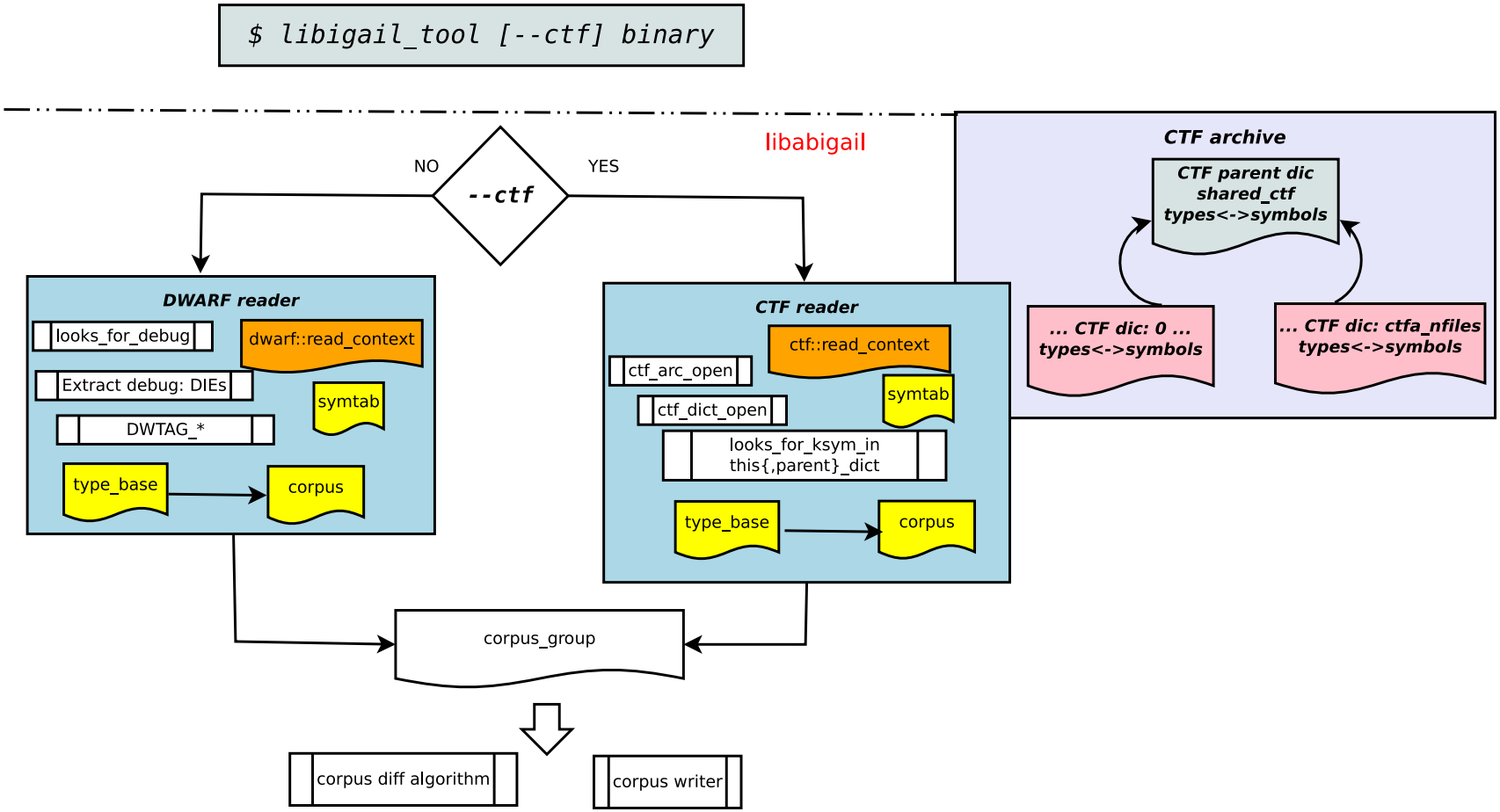

Figure 1: CTF pieces used by libabigail in CTF reader.

Libabigail Tools with CTF support

Tools with CTF support included in libabigail are:

- `abidw`: Represent ABI as XML output.
- `abidiff`: Summarize ABI changes between two binary files.
- `abipkgdiff`: Like abidiff but works on directories or compress/packages files.
- `kmidiff`: Compares the binary Kernel and Modules of two Linux Kernel trees.

Example src1.c

Compare two ABIs corpus, with libabigail (diff-object), summarising absolute changes.

```
extern void other_sym();
const int kkkk; int a[10] = {1};
enum e
{
  ONE=1, TWO=2,
};
int
f0 (char p)
{ return 1; }
struct p
{
  int a;
  char b;
};
void
f2 (char c , int b, enum e e)
{ }
```

Listing 2: src1.c

Generating ELF with CTF debug information

```
# compile with CTF debug info
$ gcc -gctf -shared src/src1.c -o src/src1.so

$ objdump -h src/src1.so

src/src1.so:     file format elf64-x86-64

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
...
 23 .ctf          00000168  0000000000000000  0000000000000000  00003486  2**0
                  CONTENTS, READONLY

# compile with DWARF and CTF debug info
$ gcc -g -gctf -shared src/src1.c -o src/src1.so
```

LISTING 3: compile.sh

Generate ABI representation

```
$ abidw --ctf src1.so --out-file src1.abi
```

```
<abi-corpus version='2.1' path='src/src1.so' architecture='elf-amd-x86_64'>
  <elf-function-symbols>
    <elf-symbol name='f0' type='func-type' binding='global-binding' ... is-defined='yes'/>
  ...
  <elf-variable-symbols>
    <elf-symbol name='a' size='40' type='object-type' ... is-defined='yes'/>
    ...
  <abi-instr address-size='64' language='LANG_C'>
    <enum-decl name='e' linkage-name='e' id='type-id-3'>
    ...
    <array-type-def dimensions='1' type-id='type-id-4' ...>
    ...
    <qualified-type-def type-id='type-id-4' const='yes' id='type-id-8'/>
    <var-decl name='kkkk' ... mangled-name='kkkk' ... elf-symbol-id='kkkk'/>
    <function-decl name='f0' visibility='default' ... alignment-in-bits='8' ... >
      <parameter type-id='type-id-2'/>
      <return type-id='type-id-4'/>
    </function-decl>
    ...
  </abi-instr>
</abi-corpus>
```

Listing 4: src1.abi

Example src2.c

```
1 extern void other_sym();
  int kkkk;
3 int a[] = {1};
  enum e
5 {
    ONE=1, TWO=2, THREE=3,
7 };
  struct p
9 {
    int a;
11  char b;
    int c;
13 };
  void
15 f0 (int p, enum e e)
  { }
17 void
  f3 (char c ,int b, struct p p)
19 { other_sym(); }
```

Listing 5: src2.c

Command line

```
# generate ABIs XML representation
$ abidw --ctf src1.so --out-file src1.abi
$ abidw --ctf src2.so --out-file src2.abi

# comparing ABIs XML references with 'libabigail' tools
$ abidiff src1.abi src2.abi

# comparing a reference ABI vs ELF binary file
$ abidiff --ctf src1.abi src2.so

# comparing a ELFs binary files
$ abidiff --ctf src1.so src2.so

# comparing ABIs XML references with 'google'
# tools
$ stgdiff -f small -o abi.diff src1.abi src2.abi
```

LISTING 6: script1.sh

Output report

```
Functions changes summary: 1 Removed, 1 Changed, 1 Added functions
Variables changes summary: 0 Removed, 2 Changed, 0 Added variables
1 Removed function:
  [D] 'function void f2(char, int, e)'    {f2}
1 Added function:
  [A] 'function void f3(char, int, p)'    {f3}
1 function with some indirect sub-type change:
  [C] 'function int f0(char)' has some indirect sub-type changes:
    return type changed:
      type name changed from 'int' to 'void'
      ...
    parameter 1 of type 'char' changed:
      type name changed from 'char' to 'int'
      ...
    parameter 2 of type 'enum e' was added
2 Changed variables:
  [C] 'int a[10]' was changed to 'int a[]':
    size of symbol changed from 40 to 4
    ...
      type name changed from 'int[10]' to 'int[]'
  [C] 'const int kkkk' was changed to 'int kkkk':
      ...
      entity changed from 'const int' to 'int'
```

Comparing Linux Kernel trees in libabigail

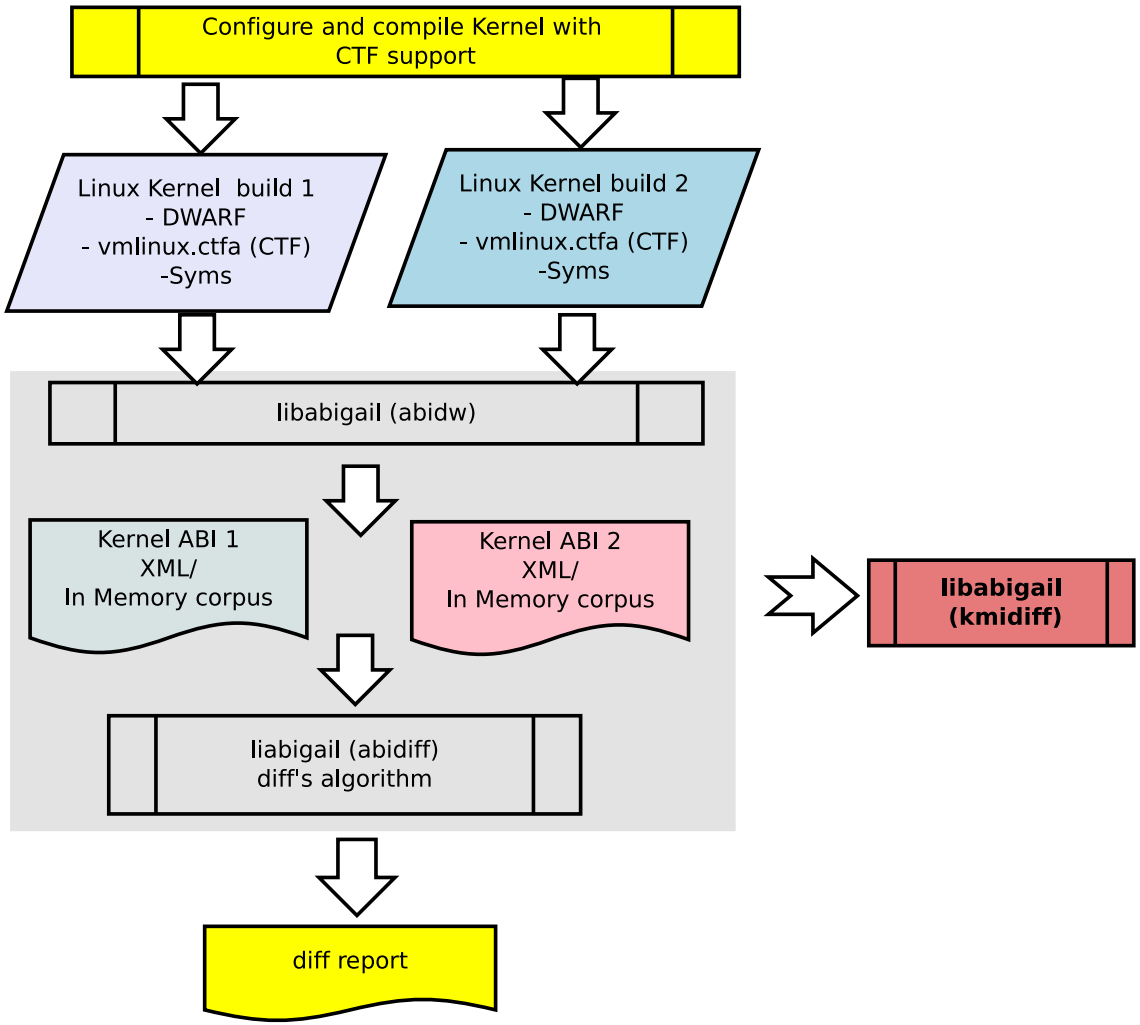

Figure 2: Workflow: comparing Linux kernels

This requires CTF for Kernel (to be upstreamed)

```
# set CONFIG_CTF=y
$ make && make ctf

$ ls /path/to/linux_build/*
..
vmlinux
vmlinux.ctfa
...
module.ko
module.ko.ctf
...
```

Listing 8: Compile Linux Kernel with CTF support

Command line

```
# generate ABIs XML representation for
# Kernel + Modules
$ abidw --ctf --lt /path/to/build_dir_kernel1_and_modules > linux1.abi
$ abidw --ctf --lt /path/to/build_dir_kernel2_and_modules > linux2.abi

# comparing ABIs XML references with 'libabigail'
# tools
$ abidiff --impacted-interfaces linux1.abi linux2.abi > kabi.diff

# comparing a Kernel Linux trees
$ kmidiff --ctf /path/to/build_dir_kernel1_and_modules \
  /path/to/build_dir_kernel2_and_modules \
  > kabi.diff

# comparing ABIs XML references with 'google'
# tools
$ stgdiff -f small -o kabi.diff linux1.abi linux2.abi
```

LISTING 9: script2.sh

Performance: corpus generation (CTF vs DWARF)

```
$ find /path/to/kernel/build -name "*.ko" | wc -l
858

$ ls -lh /path/to/kernel/build/vmlinux
-rwxr-xr-x 1 byby byby 71M Aug 18 12:18 /path/to/kernel/build/vmlinux

$ time abidw --ctf --lt /path/to/kernel/build > linux-ctf.abi
real    1m17.366s
user    0m14.778s
sys     0m1.951s

$ time abidw --lt /path/to/kernel/build > linux-dwarf.abi
real    4m47.540s
user    4m45.895s
sys     0m2.520s

$ ls -lh /path/to/kernel/build/vmlinux.ctfa
-rw-r--r-- 1 byby byby 17M Aug 14 20:10 /path/to/kernel/build/vmlinux.ctfa

$ objcopy --strip-debug /path/to/kernel/build/vmlinux
-rwxr-xr-x 1 byby byby 19M Aug 23 12:35 /path/to/kernel/build/vmlinux
```

LISTING 10: performance-script.sh

Performance: ABI diffing

Should libabigail support custom comparison algorithms?

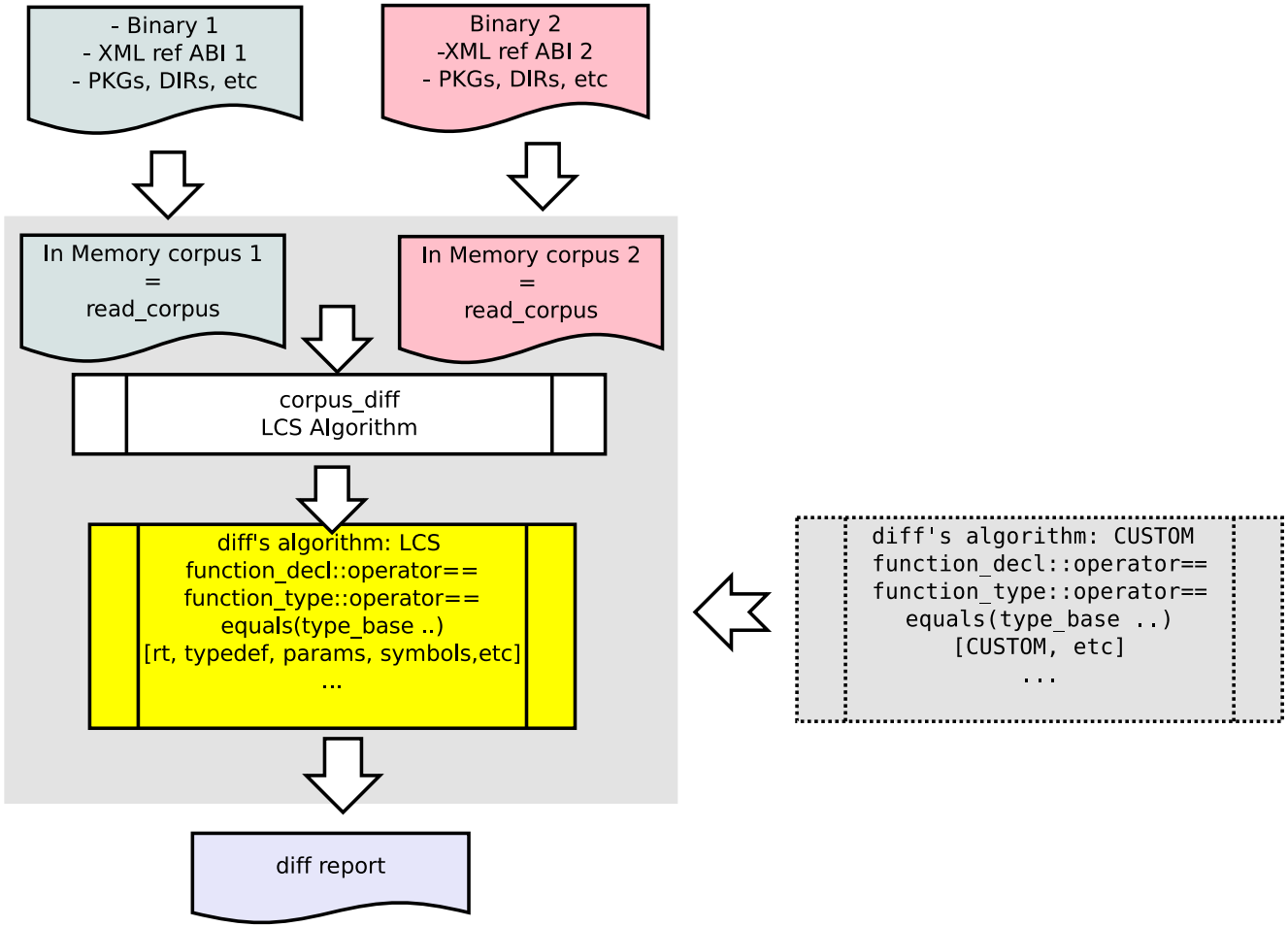

Figure 3: Custom comparison algorithm.

Examples

```
1
# Percentage time reduced using libabigail tools with CTF reader.
3 #
+----------+---------+----------+-------------+-------------------------+
5 |  binary  | version |   tool   | DWARF reader| CTF reader| improvement
|----------+---------+----------+-------------+----------+--------------|
7 |  grep    |  3.1    | abidw    |  0m0.394s   | 0m0.250s |      36%     |
|----------+---------+----------+-------------+----------+--------------|
9 | coreutils |  8.30  | abipkgdiff|  0m1.343s  | 0m1.187s |      45%     |
|----------+---------+----------+-------------+----------+--------------|
11 | finduitls |  4.6.0  | abipkgdiff|  0m0.723s  | 0m0.517s |      28%     |
|----------+---------+----------+-------------+----------+--------------|
13 | elfutils  |  0.170  | abipkgdiff|  0m0.223s  | 0m0.217s |     2.6%     |
|----------+---------+----------+-------------+----------+--------------|
15 |----------+---------+----------+-------------+----------+--------------|
| vmlinux + | 5.16-rc4| abidw    |  4m47.507s  | 1m17.459s |    300%     |
17 |   modules |         |          |             |          |             |
+-----------+---------+----------+-------------+----------+--------------+
```

Listing 11: performance-script-us.sh

Detecting kABI breakage: kABI v1

```
__attribute__((__noinline__))
const char *do_main_entry_please(const void *n, int m, int k)
{
  printk("do_main_entry_please\n");
  if (m == k)
       return NULL;
  else
       return n + m;
}
EXPORT_SYMBOL(do_main_entry_please);

/* cat Module.symvers
 * 0x5e341df8  do_main_entry_please  vmlinux EXPORT_SYMBOL
 */
```

Listing 12: kabi-v1.c

Build a module using kABI

```
#include <linux/kernel.h>
#include <linux/module.h>

int hello_init(void)
{
      char *s = do_main_entry_please(NULL, 12, 12);
      s = s;
      printk(KERN_INFO "Hello World!\n");
      return 0;
}

void hello_exit(void)
{
       printk(KERN_INFO "Bye World!\n");
}

module_init(hello_init);
module_exit(hello_exit);

MODULE_LICENSE("GPL");
```

Listing 13: linux-module.c

Module's symbols

```
$ arm-linux-gnueabihf-objdump -t linux-module.ko

linux-module.ko:     file format elf32-littlearm

SYMBOL TABLE:
00000000 l    d  .note.gnu.build-id     00000000 .note.gnu.build-id
00000000 l    d  .note.Linux    00000000 .note.Linux
00000000 l    d  .text  00000000 .text
00000000 l    d  .text.unlikely 00000000 .text.unlikely
00000000 l    d  __ksymtab      00000000 __ksymtab
00000000 l    d  __kcrctab      00000000 __kcrctab
0000000c l    d  .plt   00000000 .plt
...
00000000 g     O  .gnu.linkonce.this_module      00000240 __this_module
00000000         *UND*  00000000 __aeabi_unwind_cpp_pr0
00000000         *UND*  00000000 do_main_entry_please
00000018 g     F .text.unlikely 0000000c cleanup_module
00000020 g     F .text  00000028 init_module
00000000         *UND*  00000000 _printk
00000020 g     F .text  00000028 hello_init
00000018 g     F .text.unlikely 0000000c hello_exit
```

LISTING 14: module-syms.sh

kABI v2

```
__attribute__ ((access (read_only, 1))) __attribute__((__noinline__))
const char *do_main_entry_please(const void *n, int m, int k)
{
  printk("do_main_entry_please\n");
  if (m == k)
       return NULL;
  else
       return n + m;
}
EXPORT_SYMBOL(do_main_entry_please);

/* cat Module.symvers
 * 0xbce9bef2 do_main_entry_please  vmlinux EXPORT_SYMBOL
 */
```

Listing 15: kabi-v2.c

Loading module

```
$ insmod linux-module.ko
[   20.143755] linux_module: loading out-of-tree module taints kernel.
[   20.150282] linux_module: disagrees about version of symbol do_main_entry_please
[   20.157876] linux_module: Unknown symbol do_main_entry_please (err -22)
insmod: can't insert 'linux-module.ko': invalid parameter
```

Listing 16: kabi-load-v2.sh

Why is the kABI incompatible?

Using: `arm-linux-gnueabihf-objdump -D -j.text kabi-v{1,2}.o`

```
00000000 <do_main_entry_please>:
   0: b480        push  {r7}
   2: b085        sub sp, #20
   4: af00        add r7, sp, #0
   6: 60f8        str r0, [r7, #12]
   8: 60b9        str r1, [r7, #8]
   a: 607a        str r2, [r7, #4]
   c: 68ba        ldr r2, [r7, #8]
   e: 687b        ldr r3, [r7, #4]
   ...
  1e: 4618        mov r0, r3
  20: 3714        adds  r7, #20
  22: 46bd        mov sp, r7
  24: f85d 7b04   ldr.w r7, [sp], #4
  28: 4770        bx  lr
```

LISTING 17: kabi-vX.s

Why is the kABI incompatible?

Using: `abidw --ctf kabi-v{1,2}.o`

```
<abi-corpus version='2.1' ... architecture='elf-arm'>
  <elf-function-symbols>
    <elf-symbol name='do_main_entry_please' type='func-type' binding='global-binding'
     visibility='default-visibility' is-defined='yes'/>
  </elf-function-symbols>
  <abi-instr address-size='64' language='LANG_C'>
    <type-decl name='char' size-in-bits='8' alignment-in-bits='8' id='type-id-1'/>
    <type-decl name='int' size-in-bits='32' alignment-in-bits='32' id='type-id-2'/>
    <qualified-type-def type-id='type-id-1' const='yes' id='type-id-3'/>
    <pointer-type-def type-id='type-id-3' size-in-bits='32' ... id='type-id-4'/>
    <qualified-type-def type-id='type-id-5' const='yes' id='type-id-6'/>
    <pointer-type-def type-id='type-id-6' size-in-bits='32' ... id='type-id-7'/>
    <function-decl name='do_main_entry_please' visibility='default' binding='global' ...
     elf-symbol-id='do_main_entry_please'>
      <parameter type-id='type-id-7'/>
      <parameter type-id='type-id-2'/>
      <parameter type-id='type-id-2'/>
      <return type-id='type-id-4'/>
    </function-decl>
    <type-decl name='void' id='type-id-5'/>
  </abi-instr>
</abi-corpus>
```

Why is the kABI incompatible?

Using: `abidiff`

```
$ abidiff --ctf kabi-v{1,2}.o
$ echo $?
0
```

Listing 19: abidiff-kabi.sh

Why is the kABI incompatible?

Using: poke

```
load elf;

var fd1 = open ("kabi-v1.o", IOS_M_RDONLY);
var fd2 = open ("kabi-v2.o", IOS_M_RDONLY);

var elfv1 = Elf64_File @ fd1 : 0#B;
var elfv2 = Elf64_File @ fd2 : 0#B;

for (s in elfv1.shdr)
{
  var name = elfv1.get_section_name (s.sh_name);

  printf "Looking differences in \"%s\"\n" name;
  var secv1 = elfv1.get_sections_by_name (name);
  var secv2 = elfv2.get_sections_by_name (name);
  sdiff :a secv1[0] :b secv2[0];
}

print "Done..\n";
```

LISTING 20: cmp-elf.pk

Why is the kABI incompatible?

```
$ poke -L cmp-elf.pk
Looking differences in ""
Looking differences in ".text"
Looking differences in ".data"
Looking differences in ".bss"
Looking differences in ".ctf"
Looking differences in ".comment"
Looking differences in ".note.GNU-stack"
Looking differences in ".eh_frame"
Looking differences in ".rela.eh_frame"
Looking differences in ".symtab"
Looking differences in ".strtab"
Looking differences in ".shstrtab"
Done..
```

Listing 21: run-poke.sh

Why is the kABI incompatible?

- CRC generated by `./scripts/genksyms`.
- Safely use of `MODULE_INIT_IGNORE_MODVERSIONS`.

```
static const struct modversion_info ____versions[]
__used __section("__versions") = {
        { 0x47fadb66, "module_layout" },
        { 0x92997ed8, "_printk" },
        { 0x5e341df8, "do_main_entry_please" },
        { 0xefd6cf06, "__aeabi_unwind_cpp_pr0" },
};
```

Listing 22: linux-module.mod.c

Can we do better? Yes, with CTF

```
$ objdump --ctf linux-module.ko
  ...
Function objects:
...
_printk -> 0x30: (kind 5) int (*) (const char *, ...) (aligned at 0x8)
do_main_entry_please -> 0x2e: (kind 5) const char *(*) (const void *, int, int) (
    aligned at 0x8)

$ objdump --ctf --ctf-parent=shared_ctf vmlinux-v1.ctfa.elf
...
_printk -> 0x855: (kind 5) int (*) (const char *, ...) (aligned at 0x8)
do_main_entry_please -> 0x8002ec06: (kind 5) const char *(*) (const void *, int, int) (
    aligned at 0x8)

$ objdump --ctf --ctf-parent=shared_ctf vmlinux-v2.ctfa.elf
...
_printk -> 0x855: (kind 5) int (*) (const char *, ...) (aligned at 0x8)
do_main_entry_please -> 0x8002ec06: (kind 5) const char *(*) (const void *, int, int) (
    aligned at 0x8)
```

LISTING 23: cmp-kabi-ctf.sh

and libabigail ...

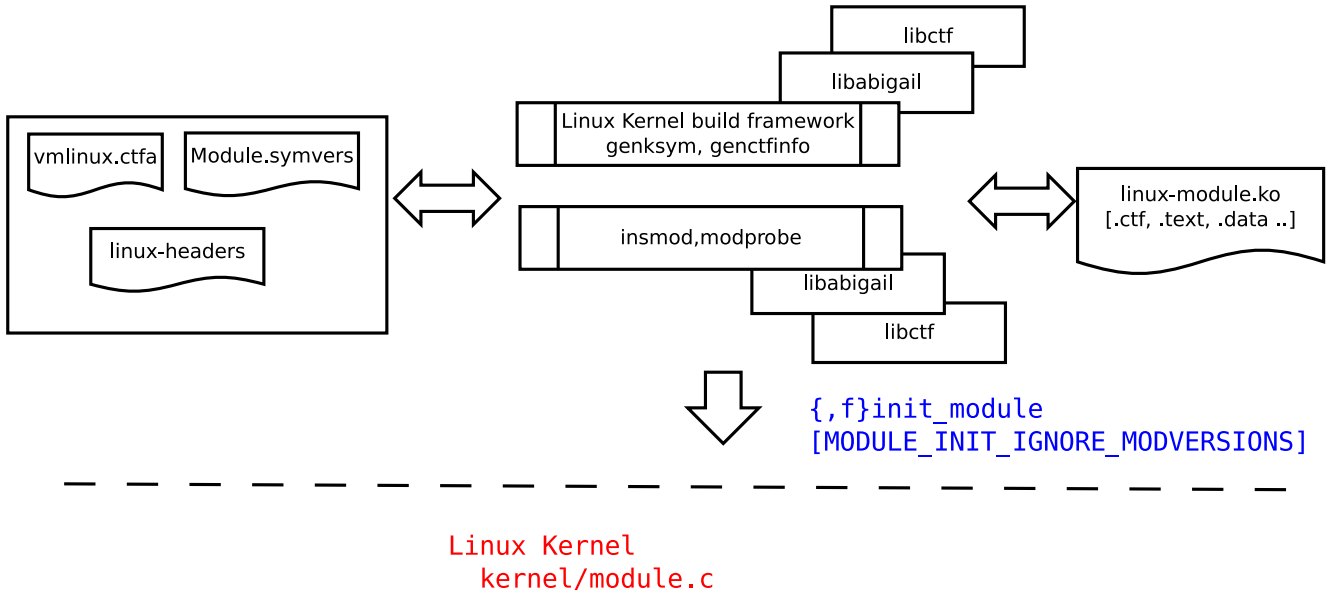

Figure 4: Using libabigail with CTF support.

And why not DWARF?

- CTF size overhead 50%-100%!.
- DWARF size overhead ~300%!!.

```
$ ls -lh ./drivers/usb/core/usbcore.ko.ctf
-rw-r--r-- 1 byby byby 226K Aug 13 21:42 ./drivers/usb/core/usbcore.ko.ctf
$ cp ./drivers/usb/core/usbcore.ko ./drivers/usb/core/usbcore.ko.strip
$ ls -lh ./drivers/usb/core/usbcore.ko
-rw-r--r-- 1 byby byby 1.1M Aug 13 21:42 ./drivers/usb/core/usbcore.ko
$ objcopy -g ./drivers/usb/core/usbcore.ko.strip
$ ls -lh ./drivers/usb/core/usbcore.ko.strip
-rw-r--r-- 1 byby byby 293K Aug 23 15:15 ./drivers/usb/core/usbcore.ko.strip
```

Listing 24: cmp-kabi-ctf.sh

Unnoticed ABI breakages

```
/* gcc-12 -g -gctf abi-v1.c -c -o abi-v1.o */

int
doit(void *p, int a, int b)
{
  char *s = (char *)p;
  if (a == b)
    printf ("%s\n", s);

  return a ? a : b;
}
```

Listing 25: abi-v1.c

Unnoticed ABI breakages

```
/*
   The ms_abi attribute tells the compiler to use the Microsoft ABI.

   x64 calling convention:
     Integer arguments are passed in registers RCX, RDX, R8, and R9.
     Floating point arguments are passed in XMM0L, XMM1L, XMM2L, and XMM3L.
     16-byte arguments are passed by reference....
     A scalar return value that can fit into 64 bits, including the __m64 type,
     is returned through RAX..
*/

/* gcc-12 -g -gctf ~/c/mabi.c -c -o ~/c/mabi-v2.o */

__attribute__((__ms_abi__))
int doit(void *p, int a, int b)
{
  char *s = (char *)p;
  if ( a == b)
    printf ("%s\n", s);

  return a ? a : b;
}
```

Listing 26: abi-v2.c

Unnoticed ABI breakages

```
$ cat test-abi.c
extern int
doit(void *p, int a, int b);

int
main()
{
  doit ("main", 12, 12);
}

gcc-12 -g -gctf -o test-abi abi-v1.o test-abi.c
$ ./test-abi
main

gcc-12 -g -gctf -o test-abi abi-v2.o test-abi.c
$ ./test-abi
Segmentation fault
```

Listing 27: test-abi.sh

Unnoticed ABI breakages

```
Breakpoint 1, main () at /home/byby/c/test-abi.c:5
5         doit ("main", 12, 12);
(gdb) s
doit (p=0x555555557dd8, a=0xc, b=0xf7df3ef0) at /home/byby/c/mabi.c:4
4         char *s = (char *)p;
(gdb) n
5         if ( a == b)
(gdb)
8         return a ? a : b;
(gdb)

rax             0x555555556004      0x555555556004                  /* sysv arg1 */
rbx             0x0                 0x0
rcx             0x555555557dd8      0x555555557dd8   /* ms arg1 */
rdx             0xc                 0xc              /* ms arg2 */  /* sysv arg2 */
rsi             0xc                 0xc                             /* sysv arg3 */
...
r8              0x7ffff7df3ef0      0x7ffff7df3ef0   /* ms arg3 */
...
(gdb) p (char *) 0x555555556004
$4 = 0x555555556004 "main"
(gdb)
```

Listing 28: gdb-session.gdb

Unnoticed ABI breakages

```
$ poke -L cmp-elf.pk
Looking differences in ""
Looking differences in ".text"
@@ 0x10+8,0x10+8 @@
-11 00 00 00 00 00 00 00    a.sh_size   0x0000000000000011UL#B
+12 00 00 00 00 00 00 00    b.sh_size   0x0000000000000012UL#B
Looking differences is ".data"
@@ 0x48+8,0x48+8 @@
-51 00 00 00 00 00 00 00    a.sh_offset   0x0000000000000051UL#B
+52 00 00 00 00 00 00 00    b.sh_offset   0x0000000000000052UL#B
...
Looking differences in ".eh_frame"
Looking differences in ".rela.eh_frame"
Looking differences in ".shstrtab"

$ abidiff abi-v{1,2}.o
$ echo $?
0
$ abidiff --ctf abi-v{1,2}.o
$ echo $?
0
```

LISTING 29: run-diff-abi.sh

Unnoticed ABI breakages

Adding ABI metadata to ELF sections ...

- `malloc`, `nonreturn`, `section`, `zero_call_used_regs`, etc.

Questions & Answers ...