Analyzing Linux Kernel interface changes by looking at binaries

Dodji Seketeli <dodji@redhat.com>

Kernel Recipes, Paris 2016

What if we could see ...

- Changes in interfaces between vmlinux and its modules
- Just by looking at:
  - Two versions of vmlinux
  - Two versions of a given kernel module
- A kind of diff
  - For (ELF) binaries
  - That shows changes in a meaningful way for programmers

There is tooling for ***almost*** that

- abidiff works on userspace ELF binaries
- Reads:
  - Basic information from ELF (symbols, etc)
  - Semantic information from ELF debug information
    - Functions, variables
    - Type hierarchies
    - Source locations of definitions
- Builds internal representation of an *ABI Corpus*
  - Globally defined functions and variables of a binary
  - Including their ***types***
- Builds internal representation of differences between *ABI Corpora*

Action: semantic diffs of binaries

```
$ abidiff libtest-v0.so libtest-v1.so
Functions changes summary: 0 Removed, 1 Changed, 0 Added function
Variables changes summary: 0 Removed, 0 Changed, 0 Added variable

1 function with some indirect sub-type change:

 [C]'function int foo(struct_type*)' at test-v1.c:10:1 has some indirect sub-type changes:
   return type changed:
     type name changed from 'int' to 'char'
     type size changed from 32 to 8 bits
   parameter 1 of type 'struct_type*' has sub-type changes:
     in pointed to type 'typedef struct_type' at test-v1.h:9:1:
       underlying type 'struct S' at test-v1.h:4:1 changed:
         1 data member change:
          type of 'priv_type* S::priv' changed:
            in pointed to type 'typedef priv_type' at test-v1.h:2:1:
              underlying type 'struct priv' at test-v1.c:3:1 changed:
                type size changed from 64 to 96 bits
                1 data member insertion:
                  'unsigned int priv::added_in_between', at offset 32 (in bits) at test-v1.c:6:1
                1 data member change:
                 'char priv::m2' offset changed from 32 to 64 (in bits)

$
```

And the flat source code diff was ...

```
$  diff -u test-v0.c test-v1.c
--- test-v0.c 2016-09-29 12:11:12.688336271 +0200
+++ test-v1.c  2016-09-29 12:28:04.275719322 +0200
@@ -1,12 +1,13 @@
-#include "include1/test-v0.h"
+#include "include2/test-v1.h"

 struct priv
 {
   int m1;
+  unsigned added_in_between;
   char m2;
 };

-int foo(struct_type *s)
+char foo(struct_type *s)
 {
   return s->priv->m2;
 }
$  diff -u include1/test-v0.h include2/test-v1.h
--- include1/test-v0.h 2016-09-29 11:32:06.363637581 +0200
+++ include2/test-v1.h  2016-09-29 11:31:27.870003380 +0200
@@ -8,4 +8,4 @@

 typedef struct S struct_type;

-int foo(struct_type *s);
+char foo(struct_type *s);
$
```

Less noise

```
$ abidiff --headers-dir1 include1  --headers-dir2 include2 libtest-v0.so libtest-v1.so
Functions changes summary: 0 Removed, 1 Changed, 0 Added function
Variables changes summary: 0 Removed, 0 Changed, 0 Added variable

1 function with some indirect sub-type change:

  [C]'function int foo(struct_type*)' at test-v1.c:10:1 has some indirect sub-type changes:
    return type changed:
      type name changed from 'int' to 'char'
      type size changed from 32 to 8 bits

$
```

Choose your differences! [1/2]

```
$ abidiff libtest2-v0.so libtest2-v1.so
Functions changes summary: 1 Removed, 1 Changed, 1 Added functions
Variables changes summary: 0 Removed, 0 Changed, 0 Added variable

1 Removed function:

  'function void function_to_remove()'    {function_to_remove}

1 Added function:

  'function void function_added()'    {function_added}

1 function with some indirect sub-type change:

  [C]'function void function(int, char)' at test2-v1.c:1:1 has some indirect sub-type changes:
    return type changed:
      type name changed from 'void' to 'int'
      type size changed from 0 to 32 bits
    parameter 2 of type 'char' was removed
$
```

Choose your differences! [2/2]

```
$ cat libtest2.abignore
[suppress_function]
  change_kind = added-function

$ abidiff --suppr libtest2.abignore libtest2-v0.so libtest2-v1.so
Functions changes summary: 1 Removed, 1 Changed, 0 Added functions (1 filtered out)
Variables changes summary: 0 Removed, 0 Changed, 0 Added variable

1 Removed function:

  'function void function_to_remove()'    {function_to_remove}

1 function with some indirect sub-type change:

  [C]'function void function(int, char)' at test2-v1.c:1:1 has some indirect sub-type changes:
    return type changed:
      type name changed from 'void' to 'int'
      type size changed from 0 to 32 bits
    parameter 2 of type 'char' was removed
$
```

Save binary interfaces

```
$ abidw  libtest-v0.so > libtest-v0.so.abi
```

Other tools in the family

- abipkgdiff: compare binaries in two packages
  - RPMs and DEBs
- fedabipkgdiff: compare binaries in remote packages
  - Queries packages built in the remote Fedora build system
  - This works just for Fedora
- Automatic ABI change review of Fedora package updates

Nothing for the Linux Kernel! (Yet)

Pipe dream

What if we had hypothetical tools to analyze kernel/modules interface changes?

```
$ kabidiff usr/lib/debug/lib/modules/4.8.0-0.rc7.git1.1.local.fc26.x86_64/vmlinux \
           usr/lib/debug/lib/modules/4.8.0-0.rc8.git1.1.local.fc26.x86_64/vmlinux
```

```
$ kabipkgdiff linux.git/master/build-dir/ linux.git/my-hack-branch/build-dir/
```

What it would take

- Handle special Linux/ELF symbol sections
  - __export_symbol, __export_symbol_gpl sections
- Support augmenting a (vmlinux) ABI Corpus
  - With ABI artifacts coming from modules
- More memory consumption optimizations

Work has just started ...

- In the dodji/kabidiff branch of the Git repo
  - git clone -b dodji/kabidiff git://sourceware.org/git/libabigail.git
- So far:
  - Added a –linux-kernel-mode to abidw
  - people.redhat.com/~dseketel/kabidiff/vmlinux.abi.txt
  - people.redhat.com/~dseketel/kabidiff/tun.ko.abi.txt
  - people.redhat.com/~dseketel/kabidiff/uio.ko.abi.txt

What do you think?

- irc://irc.oftc.net#libabigail
- https://sourceware.org/libabigail/manual/
- https://sourceware.org/libabigail/apidoc/
- https://sourceware.org/libabigail/wiki/
- https://sourceware.org/bugzilla/enter_bug.cgi?product=libabigail
- https://sourceware.org/libabigail/wiki/SubscribeToMailingList
- https://www.sourceware.org/libabigail/

Thank you
And
Enjoy Paris!