

Heisenbug hunting from the trenches

Andrei Terechko



andrei@vectorfabrics.com
<http://www.vectorfabrics.com>

Outline

- Is my software free of bugs?
- Dynamic analysis to the rescue
- Case studies:
 1. TCP/IP software stack
 2. H.264 reference software
 3. Visualization Toolkit
 4. Boost C++
 5. Car navigation software

Wake-up call

Thread 0

```
t1 = t2 = sum = 0;  
spawn Thread 1;  
spawn Thread 2;  
while !t1 && !t2;  
print sum;
```

Thread 1

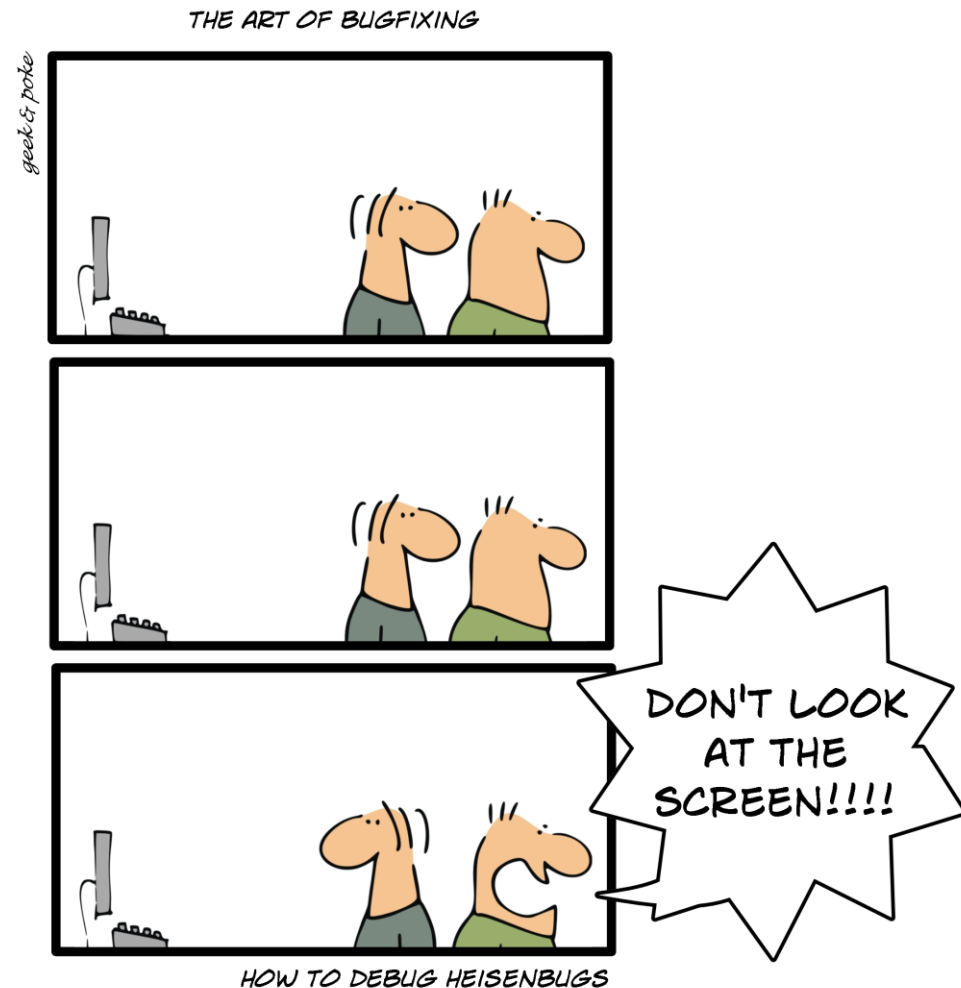
```
repeat 20:  
    sum++;  
t1 = 1;
```

Thread 2

```
repeat 20:  
    sum++;  
t2 = 1;
```

Heisenbugs and other dynamic bugs

- **Wikipedia on Heisenbug:**
a software bug that seems to disappear or alter its behavior when one attempts to study it
- hard to catch in development
- crash systems in production



Static and dynamic bugs

static bugs,
“spell checker”

```
int a[2] = {0, 1};  
int b[2] = {2, 3};
```

```
int foo(int x)  
{  
    return a[x];  
}
```

```
#include <stdio.h>  
int foo(int x); src1.c
```

```
! int main(void)  
{  
    printf("a[2]=%d\n", foo(2));  
    return 0;  
} src2.c
```

dynamic bugs,
“behavior checker”

```
int a[2] = {0, 1};  
int b[2] = {2, 3};
```

```
! int foo(int x)  
{  
    return a[x];  
} src1.c
```

```
#include <stdio.h>  
int foo(int x);  
  
int main(void)  
{  
    printf("a[2]=%d\n", foo(2));  
    return 0;  
} src2.c
```

How to detect the dynamic bugs?

Dynamic,
multithreading

Dynamic analysis
(Pareon Verify)

Quality certification
(SIL, ISO)

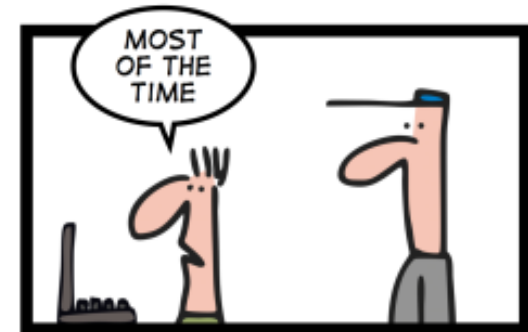
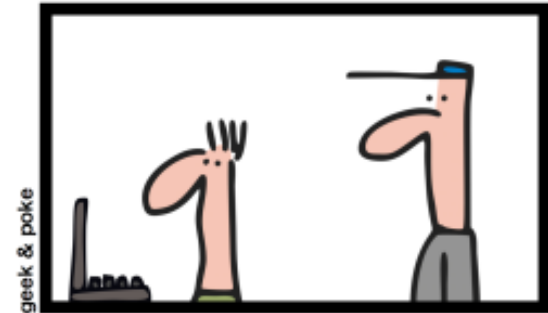
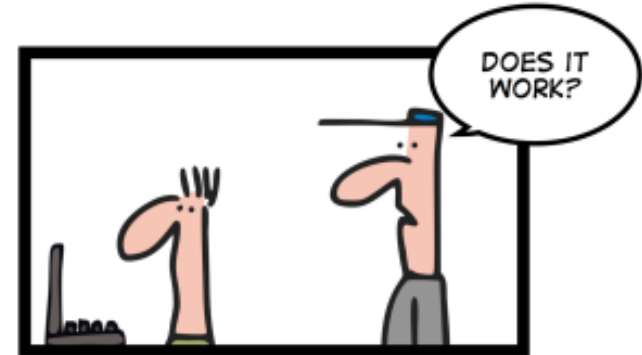
Static analysis
(Coverity, Klocwork)

Unit testing frameworks
(CPPUnit)

standards checker
(MISRA, QA-C)

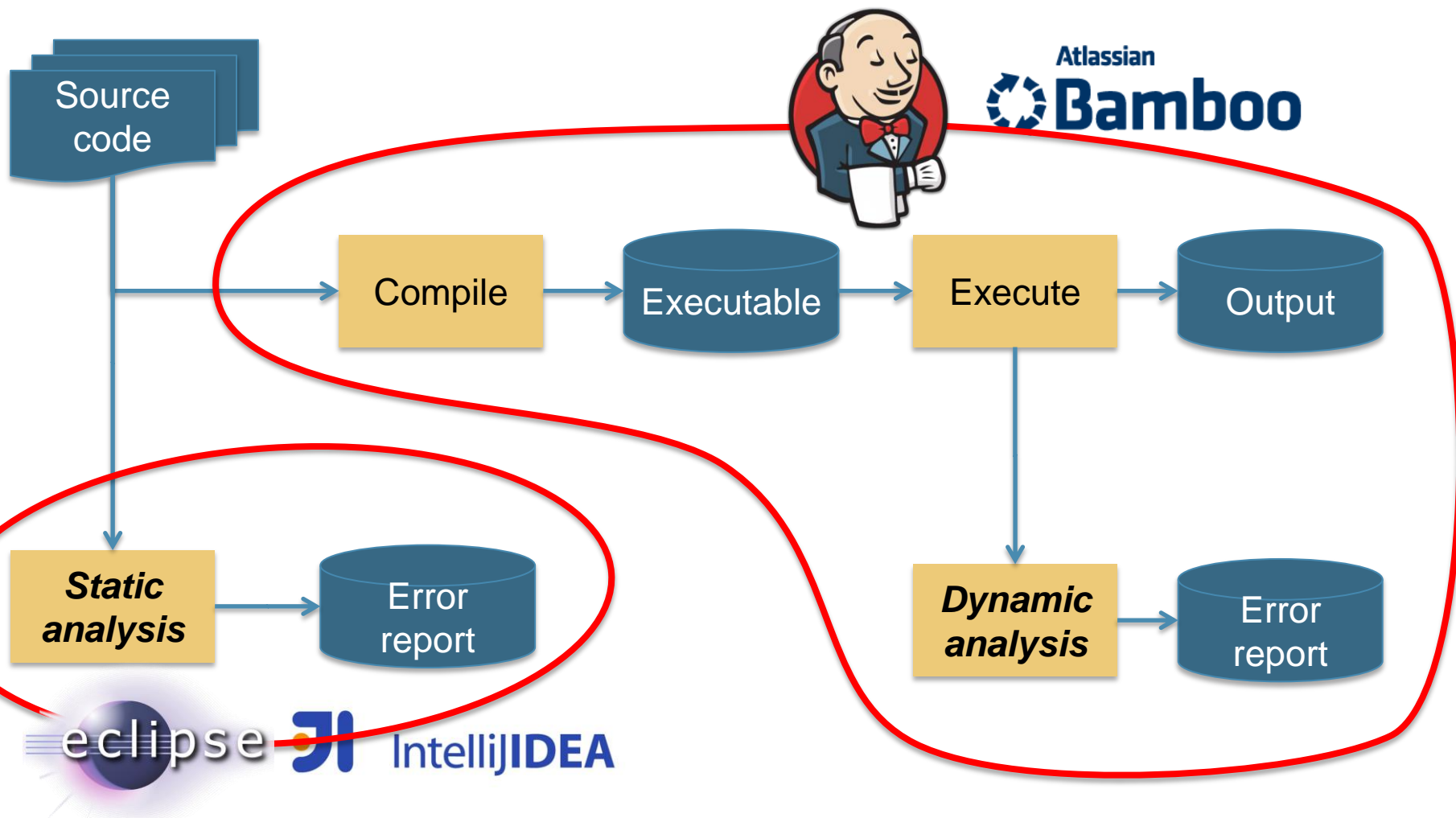
Compiler
(gcc, clang)

Static,
single thread



CONCURRENCY

Dynamic analysis @ Continuous Integration



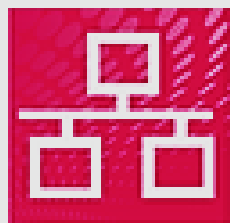
Top quality of the tested projects

- Agile development techniques
- Deploy Continuous Integration
- Excellent set of unit tests
- Many integration tests
- Some have conformance tests
- Use code coverage to add new tests
- Use static analysis tools
- Eager to adopt dynamic analysis tools



TCP/IP stack for Internet-of-Things

<https://github.com/tass-belgium/picotcp>



picoTCP

Intelligent
Systems

ALTRAN



SUPPORTING YOUR INTERNET OF THINGS

ABOUT NEWS PROJECTS CONTACT

picoTCP is the answer for a size, speed and feature conscious open source TCP/IP stack for embedded devices.

TCP/IP stack on GitHub

```
17 test/examples/dhcp_client.c
@@ -7,7 +7,6 @@
7 7 /** START DHCP Client */
8 8 #ifdef PICO_SUPPORT_DHCP
9 9 static uint8_t dhcpclient_devices = 0;
10 -static uint32_t dhcpclient_xid = 0;
...
66 58 void app_dhcp_client(char *arg)
67 59 {
68 60     char *sdev = NULL;
69 61     char *nxt = arg;
70 62     struct pico_device *dev = NULL;
63 + uint32_t dhcpclient_xid;
...
91 84 printf("Starting negotiation\n");
92 85
93 86 if (pico_dhcp_initiate_negotiation(dev, &callback_dhcpclient, &dhcpclient_xid) < 0) {
...

```

Global variable moved to stack

Pointer to stack variable passed to a callback function

Pointer dereferenced after stack deallocation

<https://github.com/tass-belgium/picotcp/commit/fea3349751fb5dc473539852ba880a7e762b7cfc>

Use after deallocation

[M0212] Use after deallocation detected:

the **write** in

function bar at dealloc.c:8

called from function main at dealloc.c:12

^^^ application start ^^^

follows after an earlier **deallocation** in

function foo called from

function main at dealloc.c:11

^^^ application start ^^^

where the object was originally created through

the **stack object** of size 4 allocated as 'x' in

function foo at dealloc.c:3

called from function main at dealloc.c:11

^^^ application start ^^^



TCP/IP bug fixes on GitHub


 tass-belgium / picotcp

SNTP: Fixed short allocation for server string

When the sntp cookie is created, the server string was being allocated using `strlen()`, which would overflow when a `strcpy` is used from the same source by putting the string terminators out of the allocated object bounds.

Bug discovered by Pareon Verify.

 master  v1.5.1 ... v1.4.2



 danielinux authored on May 21


 tass-belgium / picotcp

IPv4: Check packet len before processing

When a packet is received, the length in the header must be checked against the actual IP buffer length. If the header length has been altered, or it's set to a bigger value on purpose by an attacker, the CRC function may violate the heap memory boundaries.

Discovered using Pareon Verify.

 master  v1.5.1 ... v1.4.2

 danielinux authored on May 26

1 parent [2d712fa](#)

H.264 reference software

- H.264 - video coding standard
- Golden reference implementation
- Mature open-source project
- 117K lines of C
- <http://iphome.hhi.de/suehring/tml>



Array bound violation

[M0443] **Array bound violation(s)** detected:

the read in

```
function biari_init_context at ldecod/src/biaridecod.c:299
called from function init_contexts at ldecod/src/context_ini.c:114
called from function decode_one_frame at ldecod/src/image.c:943
called from function DecodeOneFrame at ldecod/src/ldecod.c:1254
called from function main at ldecod/src/decoder_test.c:245
```

performed 420 access of size 1 between the offsets of 240 and 658 bytes in the static object of size 720 `INIT_FLD_MAP_P' from ctx_tables.h:895

where array index 21 is outside of array `INIT_FLD_MAP_P[][0..7][][]' in

```
function init_contexts at ldecod/src/context_ini.c:114
called from function decode_one_frame at ldecod/src/image.c:943
called from function DecodeOneFrame at ldecod/src/ldecod.c:1254
called from function main at ldecod/src/decoder_test.c:245
```

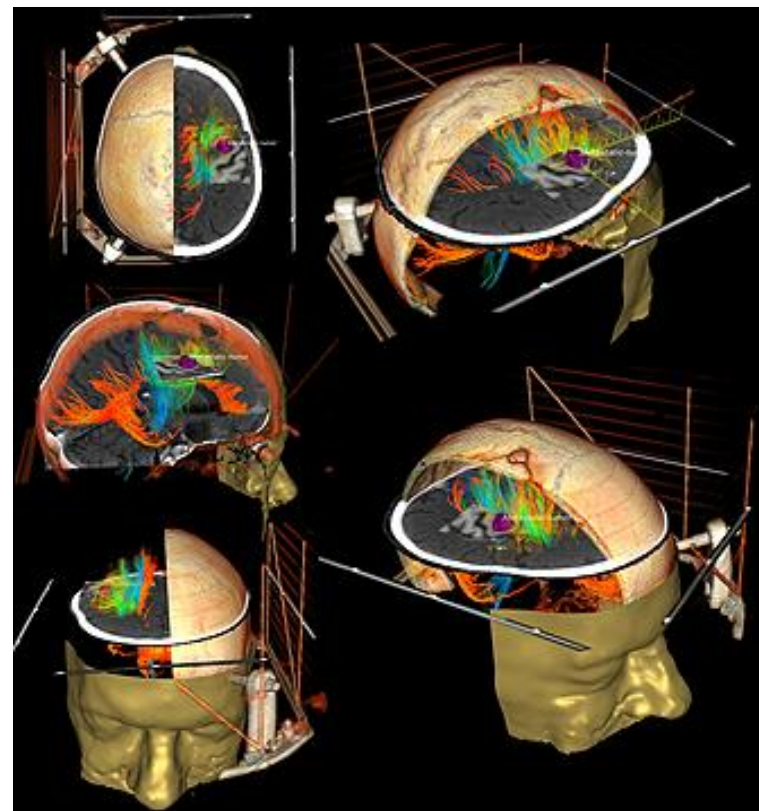
- Exact specification of the faulty index in the array
- Bug fixed two days after the submission
- <https://ipbt.hhi.fraunhofer.de/mantis/view.php?id=348>

Visualization ToolKit

- VTK – Visualization ToolKit
- Mature open-source project
- 1.5M lines of C++
- 800K lines of C
- <http://www.vtk.org>



Supporting Proven Open-Source Platforms



Visualization ToolKit

```
memcpy (cword, "abcdefgh", 8);
```

```
Swap2BERange (cword, 8);
```

```
...
```

```
void Swap2BERange(T* first, size_t num)
```

```
{
```

```
    // Swap one value at a time.
```

```
    T* last = first + num;
```

```
    for(T* p=first; p != last; ++p) {
```

```
        Swap(reinterpret_cast<char*>(p));
```

```
    }
```

```
}
```

T is not char!



- Pointers and casts, C++ templates and classes
- But the code looks OK and...
- Test succeeds

Read from uninitialized stack object

```
[M0203] Read(s) from uninitialized stack object detected:  
the read in  
function vtkByteSwapper<2ul>::Swap at vtkByteSwap.cxx:43  
called from function vtkByteSwapRange<short> at vtkByteSwap.cxx:75  
called from function vtkByteSwapBERange<short> at vtkByteSwap.cxx:193  
called from function vtkByteSwap::SwapBERange at vtkByteSwap.cxx:240  
called from function vtkByteSwap::Swap2BERange at vtkByteSwap.cxx:298  
called from function TestByteSwap at otherByteSwap.cxx:57  
called from function otherByteSwap at otherByteSwap.cxx:160  
called from function main at vtkCommonCoreCxxTests.cxx:372  
performed 1 access of size 1 at an offset of 8 bytes from the start of  
the stack object of size 1024 allocated as 'cword' in  
function TestByteSwap at otherByteSwap.cxx:32  
called from function otherByteSwap at otherByteSwap.cxx:160  
called from function main at vtkCommonCoreCxxTests.cxx:372
```

- Easy patch based on the clear error message
- Bug reported in 6.1.0 and fixed in 6.2.0
- <http://www.vtk.org/Bug/view.php?id=14997>

Boost C++ libraries

- Boost C++
- High-quality 80 libraries
- Peer-reviewed
- Highly portable
- 25M lines of C++
- <http://www.boost.org>



Boost C++ libraries

```
193     if(code <= (int)REG_E_UNKNOWN)
194     {
195         std::string p;
196         if((e) && (e->re_magic == magic_value))
197             p = static_cast<c_regex_type*>(e->guts)
198         else
```

- Magic value determines if a struct is initialized or not
- Attacker can exploit this bug to cause a denial of service
- Test succeeds

Read from uninitialized stack object

[M0203] Read(s) from uninitialized stack object detected:
the read in function **regcompW** at wide_posix_api.cpp:81
called from function main at wide_posix_api_check.cpp:44
called from function main_thread
performed 1 access(es) of size 4 at the start of
the stack object of size 40 allocated as `re' in
function main at wide_posix_api_check.cpp:42
called from function main_thread and
the resulting value is used in evaluating the condition in
function **regcompW** at wide_posix_api.cpp:81
called from function main at wide_posix_api_check.cpp:44
called from function main_thread

- Fixed
- <https://svn.boost.org/trac/boost/ticket/11472>

Features:

- Map display in 2D
- Route planning
- Route guidance
- Speech instructions



High quality code:

- Open source: peer reviewed
- Code quality confirmed by Coverity

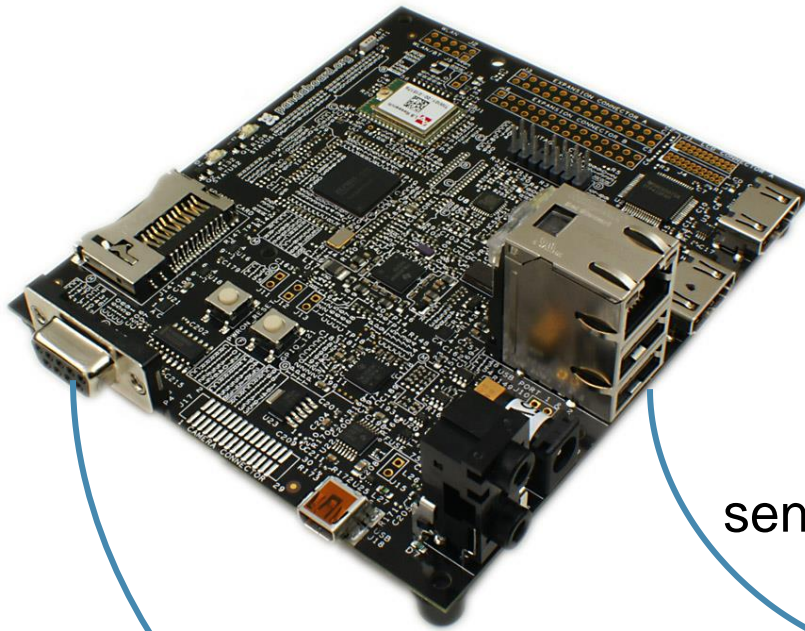
<http://sourceforge.net/projects/navit/>



Execution on target, analysis on host

Target runs instrumented Navit

Host does analysis



Yocto Linux "Dizzy"



Ubuntu 14.04

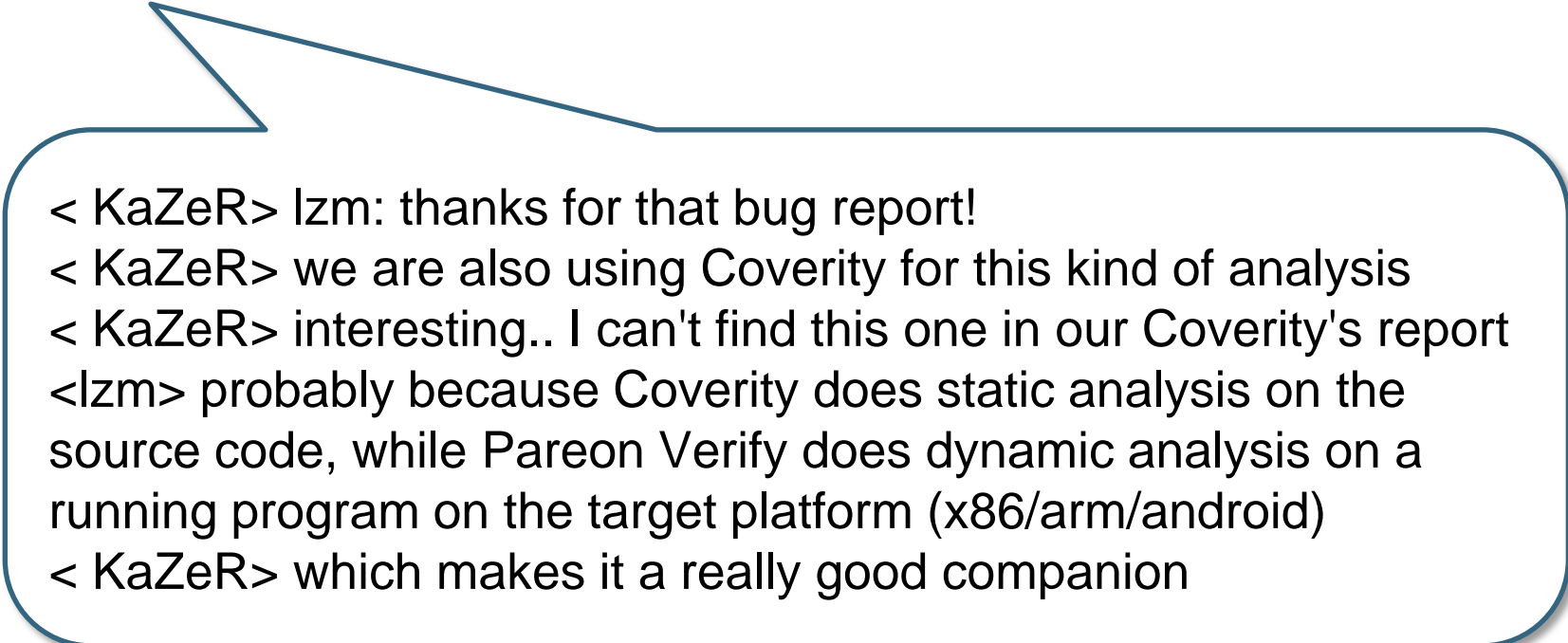
sends data to PC

USB "ethernet"

Serial port

Pareon is a good companion

We (Izm) filed a bug: <http://trac.navit-project.org/ticket/1316>
NAVIT developer (KaZeR) on IRC #navit; Aug 26 2015:



< KaZeR> Izm: thanks for that bug report!
< KaZeR> we are also using Coverity for this kind of analysis
< KaZeR> interesting.. I can't find this one in our Coverity's report
<Izm> probably because Coverity does static analysis on the source code, while Pareon Verify does dynamic analysis on a running program on the target platform (x86/arm/android)
< KaZeR> which makes it a really good companion

Bug was fixed day after reporting

Memory leak

[M0181] Memory leak detected:

```
the heap object of size 4 allocated through a call to `g_malloc0_n' in
function callback_list_new at /data/lessandro/navit/navit/callback.c:43
called from function route_new at /data/lessandro/navit/navit/route.c:487
...
called from function main_thread
has become unreachable.
```

This object was last accessible from:

```
the heap object of size 96 allocated through a call to `g_malloc0_n' in
function route_new at /data/lessandro/navit/navit/route.c:476
called from function start_element at /data/lessandro/navit/navit/xmlconfig.c:674
...
```

```
at offset 60 which was deallocated in function g_free in
function route_destroy at /data/lessandro/navit/navit/route.c:4075
called from function navit_destroy at /data/lessandro/navit/navit/navit.c:3521
...
```

This object was last accessed by:

```
the write in
function callback_list_add at /data/lessandro/navit/navit/callback.c:113
called from function route_add_attr at /data/lessandro/navit/navit/route.c:3949
called from function navigation_set_route at /data/lessandro/navit/navit/navigation.c:4354
...
called from function main_thread
```

Memory leak

```
struct route *
route_new(struct attr *parent, struct attr **attrs)
{
    ...
    this->cb12=callback_list_new();
    ...
}
```

← **Allocated here**

```
void
route_destroy(struct route *this_)
{
    this_>refcount++; /* avoid recursion */
    route_path_destroy(this_>path2,1);
    route_graph_destroy(this_>graph);
    route_clear_destinations(this_);
    route_info_free(this_>pos);
    map_destroy(this_>map);
    map_destroy(this_>graph_map);
    >g_free(this_);
}
```

← **Reference lost here**

← **Should have called:** `callback_list_destroy(this_>cb12)`

Take-away

1. Your reputation is at risk if you think your code is bug-free
2. Static error is an easy catch, dynamic bugs slip through
3. Dynamic analysis = “*really good companion*” in debugging

Find critical bugs and
optimize your software
with Pareon

<http://vectorfabrics.com>



Andrei Terechko

andrei@vectorfabrics.com

Zaltbommel, The Netherlands