Schloss Dagstuhl
25 April 2014

LinuxTesting.org

The Experience of Linux Driver Verification

Alexey Khoroshilov
Vadim Mutilin

ISP RAS
Institute for System Programming of the Russian Academy of Sciences

Linux Verification Center

LinuxTesting.org

Institute for System Programming of Russian Academy of Sciences

VERIFICATION CENTER OF THE OPERATING SYSTEM Linux

founded in 2005

- User Space Verification
- Application Binary/Program Interface Stability
- **Linux Driver Verification Program**
- Linux File System Verification
- Deductive Verification

Software Model Checking

- Reachability problem

Verification Tools World

```
int main(int argc,char* argv[])
{
 ...
 other_func(var);
 ...
}
```

```
void other_func(int v)
{
   ...
   assert( x != NULL);
}
```

Device Driver World

```
int usbpn_open(struct net_device *dev) { ... };
int usbpn_close(struct net_device *dev) { ... };
struct net_device_ops usbpn_ops = {
    .ndo_open = usbpn_open, .ndo_stop = usbpn_close
};
int usbpn_probe(struct usb_interface *intf, const struct usb_device_id *id){
    dev->netdev_ops = &usbpn_ops;
    err = register_netdev(dev);
}
void usbpn_disconnect(struct usb_interface *intf){...}

struct usb_driver usbpn_struct = {
    .probe = usbpn_probe, .disconnect = usbpn_disconnect,
};
int __init usbpn_init(void){ return usb_register(&usbpn_struct);}
void __exit usbpn_exit(void){usb_deregister(&usbpn_struct );}

module_init(usbpn_init);
module_exit(usbpn_exit);
```

No explicit calls to init/exit procedures

Device Driver World

```
int usbpn_open(struct net_device *dev) { ... };
int usbpn_close(struct net_device *dev) { ... };
struct net_device_ops usbpn_ops = {
    .ndo_open = usbpn_open, .ndo_stop = usbpn_close
};
int usbpn_probe(struct usb_interface *intf, const struct usb_device_id *id){
    dev->netdev_ops = &usbpn_ops;
    err = register_netdev(dev);
}
void usbpn_disconnect(struct usb_interface *intf){...}

struct usb_driver usbpn_struct = {
    .probe = usbpn_probe, .disconnect = usbpn_disconnect,
};
int __init usbpn_init(void){ return usb_register(&usbpn_struct);}
void __exit usbpn_exit(void){usb_deregister(&usbpn_struct );}

module_init(usbpn_init);
module_exit(usbpn_exit);
```

Callback interface procedures registration

No explicit calls to init/exit procedures

Device Driver World

```
int usbpn_open(struct net_device *dev) { ... };
int usbpn_close(struct net_device *dev) { ... };
struct net_device_ops usbpn_ops = {
    .ndo_open = usbpn_open, .ndo_stop = usbpn_close
};
int usbpn_probe(struct usb_interface *intf, const struct usb_device_id *id){
    dev->netdev_ops = &usbpn_ops;
    err = register_netdev(dev);
}
void usbpn_disconnect(struct usb_interface *intf){...}

struct usb_driver usbpn_struct = {
    .probe = usbpn_probe, .disconnect = usbpn_disconnect,
};
int __init usbpn_init(void){ return usb_register(&usbpn_struct);}
void __exit usbpn_exit(void){usb_deregister(&usbpn_struct );}

module_init(usbpn_init);
module_exit(usbpn_exit);
```

Callback interface procedures registration

No explicit calls to init/exit procedures

Driver Environment Model

```
int main(int argc,char* argv[])
{
  usbpn_init()
  for(;;) {
    switch(*) {
     case 0: usbpn_probe(*,*,*);break;
     case 1: usbpn_open(*,*);break;
     ...
    }
  }
  usbpn_exit();
}
```

Driver Environment Model (2)

- Order limitation
  - open() after probe(), but before remove()
- Implicit limitations
  - read() only if open() succeed
- and it is specific for each class of drivers

Scenarios of Handler Invocations

Scenarios of Several Groups

Driver Environment Model (3)

- Correct & Complete
  - should reproduce the same scenarios of interaction with a driver as in the real kernel environment

Driver Environment Model (3)

- Correct & Complete
  - should reproduce the same scenarios of interaction with a driver as in the real kernel environment
- Simple Enough
  - should satisfy the requirements of static verification tools

π-model of Environment

π-model of Environment

Modeling Driver Environment

Generated Environment in C

```
void main(int argc, char* argv[]) {
  while(true) {
    switch(__VERIFIER_nondet_int()) {
      case 0: //process P1, action α1:
        if(s->state == K ) {
          ...
        }
        break;
      case 1: //process P1, action α2:
      ...
    }
  }
  break_loop:
}
```

Model Checking and Linux Kernel

- Reachability problem

Error Location?

```
int f(int y)
{
struct urb *x;

  x = usb_alloc_urb(0,GFP_KERNEL);
  ...
  usb_free_urb(x);

  return y;
}
```

Error Location?

```
int f(int y)
{
struct urb *x;

  x = usb_alloc_urb(0,GFP_KERNEL); // allocate new URB
  ...
  usb_free_urb(x); // deallocate URB: assert(x is NULL or previously allocated URB)

  return y;
}


  …
  // after module exit: assert( all allocated URBs are deallocated)
```

Instrumentation

```
int f(int y)
{
struct urb *x;

  x = usb_alloc_urb(0,GFP_KERNEL);

  ...

  usb_free_urb(x);

  return y;
}
```

```
set URBS = empty;

int f(int y)
{
struct urb *x;

  x = usb_alloc_urb();
  add(URBS, urb);
  ...
  assert(contains(URBS, x));
  usb_free_urb(x);
  remove(URBS, urb);

  return y;
}
  …
  // after module exit
  assert(is_empty(URBS));
```

Aspect-Oriented Notation

```
// Model state: set of allocated URBs
set URBS = empty;

// Model functions
struct urn * ldv_usb_alloc_urb(void)
{
  void *urb;
  urb = ldv_alloc();
  if (urb) {
    add(URBS, urb);
  }
  return urb;
}

void ldv_usb_free_urb(struct urb *urb)
{
  if (urb) {
    remove(URBS, urb);
  }
}
```

```
// Pointcut declarations
pointcut USB_ALLOC_URB:
  call(struct urb *usb_alloc_urb(int, gfp_t));
pointcut USB_FREE_URB:
  call(void usb_free_urb(struct urb *));

// Update model state
around: USB_ALLOC_URB {
  return ldv_usb_alloc_urb();
}
around: USB_FREE_URB {
  ldv_usb_free_urb($arg1);
}

// Assertions
before: USB_FREE_URB {
  assert(contains(URBS, $arg1));
}
after: MODULE_EXIT {
  assert(is_empty(URBS));
}
```

Instrumentation

```
int f(int y)
{
struct urb *x;

  x = usb_alloc_urb(0,GFP_KERNEL);
  ...
  usb_free_urb(x);

  return y;
}
```

```
int f(int y)
{
struct urb *x;

  x = ldv_usb_alloc_urb();
  ...
  assert(contains(URBS, x));
  ldv_usb_free_urb(x);

  return y;
}


  …
  // after module exit
  assert(is_empty(URBS));
```

Rule Instrumentor

- **CIF** – C Instrumentation Framework
  - gcc-based aspect-oriented programming tool for C language
  - available under GPLv3:

    http://forge.ispras.ru/projects/cif

Model Checking and Linux Kernel

- Reachability problem

Linux Driver Verification

http://linuxtesting.org/ldv

Cmd Stream

- C Program
  - CC Command-set
    - compiler options (including defines)
    - source files
  - LD Command
    - linker options
    - object files

Cmd Stream (2)

CmdStream
Transformation Tool
CmdStream-upd
file1.c
file2.c
file1-upd.c

Linux Driver Verification

http://linuxtesting.org/ldv

LDV-Core: Build Cmd Extractor

- extract CmdStream from original build process
- divide CmdStream into small parts per kernel module

Linux Driver Verification

End of Linux drivers specific

http://linuxtesting.org/ldv

Error Traces in Common Format

Error Traces in Common Format (2)

[LINE] [FILE] [TYPE] [KIND] …[TEXT]

- Flexible
- Extensible
- Easy enough to convert to
- Easy to process

Error Traces in Common Format (3)

```
Line 18919:    N8417 -{mutex_lock(__cil_tmp18)}-> N10754
Line 23193:    N10754 -{Function start dummy edge}-> N10755
Line 23198:    N10755 -{[!((ldv_mutex) == (1))]}-> N10758
Line 23203:    N10758 -{ldv_blast_assert()}-> N10704
Line 23088:    N10704 -{Function start dummy edge}-> N10705
Line 23092:    N10705 -{Label: LDV_ERROR}-> N10706
Line 23094:    N10706 -{Goto: LDV_ERROR}-> N10706

2299  CALL   : mutex_lock(__cil_tmp18)
108 "model0032.c" ASSUME   : !((ldv_mutex) == (1))
108  CALL   : ldv_blast_assert()
9 "assert.h" BLOCK   : LDV_ERROR:
6  BLOCK   : goto LDV_ERROR;
```

Linux Driver Verification

http://linuxtesting.org/ldv

Demo

BLAST → CPAchecker

Total changes	Safe → Unsafe	Safe → Unknown	Unsafe → Unknown	Unknown → Safe	Unknown → Unsafe
171	3	87	12	59	10

+2 Correct
-1 Incorrect

-12 Was correct

+10 Correct

Different bugs!

Total known unsafes – 65
BLAST – 53
CPAcheker – 53

Bugs Found

http://linuxtesting.org/results/ldv

>150 patches already applied

Problems in Linux Kernel

This section contains information about problems in Linux kernel found within Linux Driver Verification program.

No.	Type	Brief	Added on	Accepted	Status
L0150	Crash	USB: cdc-acm: double usb_autopm_put_interface() in acm_port_activate()	2014-04-12	https://lkml.org/lkml/2014/4/11/627 commit	Fixed in kernel 3.15-rc2
L0149	Crash	rtl8187: fix use after free on failure path in rtl8187_probe()	2014-03-29	https://lkml.org/lkml/2014/3/28/425 commit	Fixed in kernel 3.15-rc1
L0148	Leak	adv7180: interrupt leak on failure paths in init_device()	2014-03-14	https://lkml.org/lkml/2014/3/14/479 commit	Fixed in kernel 3.15-rc1
L0147	Leak	p54usb: fix leaks at failure path in p54u_probe()	2014-03-09	https://lkml.org/lkml/2014/3/7/503 commit	Fixed in kernel 3.15-rc1
L0146	Crash	staging: dgap: absence of proper error handling in dgap_start()	2014-03-09	https://lkml.org/lkml/2014/2/23/92 commit	Fixed in kernel 3.15-rc1
L0145	Crash	drm/vmwgfx: avoid null pointer dereference at failure paths	2014-03-01	https://lkml.org/lkml/2014/2/28/501 commit	Fixed in kernel 3.14-rc5
F0007	Crash	f2fs: f2fs unmount hangs if f2fs_init_acl() fails during mkdir syscall	2014-02-17	https://lkml.org/lkml/2014/2/6/18 commit	Fixed in kernel 3.15-rc1
L0144	Deadlock	drivers/message/i2o/i2o_config.c: fix deadlock in compat_ioctl(I2OGETIOPS)	2014-02-11	https://lkml.org/lkml/2014/1/31/468 commit	Fixed in kernel 3.14-rc3
L0143	Leak	staging: gdm72xx: fix leaks at failure path in gdm_usb_probe()	2014-02-07	https://lkml.org/lkml/2014/2/5/748 commit	Fixed in kernel 3.14-rc3

Lessons Learnt

Lessons Learnt

- Language features support

gcc extensions, e.g. zero-length arrays

```
struct line {
        int length;
        char contents[0];
      };
```

etc.

Lessons Learnt

- Language features support
- No premature UNKNOWN

Error: Unsupported C feature (recursion) in line 60858:
tmp = gma_power_begin(tmp24, tmp25);
(CallstackTransferRelation.getAbstractSuccessors)

Bug Finder vs. Safe Prover

Lessons Learnt

- Language features support
- No premature UNKNOWN
- Efficiently ignore irrelevant details

Ten of thousands irrelevant transitions vs. dozens of relevant ones

Lessons Learnt

- Language features support
- No premature UNKNOWN
- Efficiently ignore irrelevant details
- Engineering Matters

BLAST

Berkeley

Lazy

Abstraction

Software Verification

Tool

BLAST is a software model checker for C programs.

It uses counterexample-driven automatic abstraction refinement to construct an abstract model which is model checked for safety properties.

ISPRAS BLAST 2.6 Release Notes

Speedup ranges from **8 times** on small-sized programs to **30 times** on medium-sized programs

- Logarithmic algorithm for useful-blocks (significantly speedup of trace analysis)
- Improved integration with SMT solvers
  - efficient string concatenation
  - caching of converted formulae
  - optimization of CVC3 options for BLAST use cases
- Formulae normalization moved to solvers since solvers do it faster
- Alias analysis speedup
  - must-aliases are handled separately and faster than may-aliases
  - removed unnecessary debug prints from alias iteration (even a check for debug flag impacts performance significantly in hot places)
- BLAST-specific tuning of OCaml virtual machine options

SVCOMP'12 Results

Competition candidate	BLAST 2.7	CPAchecker ABE 1.0.10	CPAchecker Memo 1.0.10	ESBMC 1.17	FShell 1.3	LLBMC 0.9	Predator 20111011	QARMC -HSF	SATabs 3.0	Wolverine 0.5c
Affiliation	Moscow, Russia	Passau, Germany	Paderborn, Germany	Southampton, UK	Vienna, Austria	Karlsruhe, Germany	Brno, Czechia	Munich, Germany	Oxford, UK	Princeton, USA
ControlFlowInteger 93 files, max score: 144	71 9900 s	141 1000 s	140 3200 s	102 4500 s	28 580 s	100 2400 s	17 1100 s	140 4800 s	75 5400 s	39 580 s
DeviceDrivers 59 files, max score: 103	72 30 s	51 97 s	51 93 s	63 160 s	20 3.5 s	80 1.6 s	80 1.9 s	--	71 140 s	68 65 s
DeviceDrivers64 41 files, max score: 66	55 1400 s	26 1900 s	49 500 s	10 870 s	0 0 s	1 110 s	0 0 s	--	32 3200 s	16 1300 s
HeapManipulation 14 files, max score: 24	--	4 16 s	4 16 s	1 220 s	--	17 210 s	20 1.0 s	--	--	--
SystemC 62 files, max score: 87	33 4000 s	45 1100 s	36 450 s	67 760 s	--	8 2.4 s	21 630 s	8 820 s	57 5000 s	36 1900 s
Concurrency 8 files, max score: 11	--	0 0 s	0 0 s	6 270 s	0 0 s	--	0 0 s	--	1 1.4 s	--
Overall 277 files, max score: 435	231 15000 s	267 4100 s	280 4300 s	249 6800 s	48 580 s	206 2700 s	138 1700 s	148 5600 s	236 14000 s	159 3800 s

SVCOMP'14 Results

Competition candidate	BLAST 2.7.2	CBMC	CPAchecker	CPAlien	CSeq-Lazy	CSeq-MU	ESBMC 1.22	FrankenBit	LLBMC	Predator	Symbiotic 2	Threader	UFO	Ultimate Automizer	Ultimate Kojak
Representing Jury Member	Vadim Mutilin	Michael Tautschnig	Stefan Löwe	Petr Muller	Bernd Fischer	Gennaro Parlato	Lucas Cordeiro	Arie Gurfinkel	Stephan Falke	Tomas Vojnar	Jiri Slaby	Corneliu Popeea	Aws Albarghouthi	Matthias Heizmann	Alexander Nutz
Affiliation	Moscow, Russia	London, UK	Passau, Germany	Brno, Czechia	Stellenbosch, South Africa	Southampton, UK	Manaus, Brazil	Pittsburgh, USA	Karlsruhe, Germany	Brno, Czechia	Brno, Czechia	Munich, Germany	Pittsburgh, USA	Freiburg, Germany	Freiburg, Germany
BitVectors 49 tasks, max. score: 86	--	**86** **2 300 s**	**78** **690 s**	--	--	--	77 1 500 s	--	**86** **39 s**	-92 28 s	39 220 s	--	--	--	-23 1 100 s
Concurrency 78 tasks, max. score: 136	--	**128** **29 000 s**	0 0.0 s	--	**136** **1 000 s**	**136** **1 200 s**	32 30 000 s	--	0 0.0 s	0 0.0 s	-82 5.7 s	100 3 000 s	--	--	0 0.0 s
ControlFlow 843 tasks, max. score: 1261	508 32 000 s	397 42 000 s	**1009** **9 000 s**	455 6 500 s	--	--	949 35 000 s	**986** **6 300 s**	**961** **13 000 s**	511 3 400 s	41 39 000 s	--	912 14 000 s	164 6 000 s	214 5 100 s
ControlFlowInteger 181 tasks, max. score: 255	64 7 800 s	-298 35 000 s	179 4 800 s	121 3 400 s	--	--	85 24 000 s	149 5 300 s	74 10 000 s	-28 2 200 s	-151 22 000 s	--	184 9 500 s	33 5 800 s	57 5 000 s
Loops 65 tasks, max. score: 99	25 320 s	99 1 100 s	68 600 s	-16 91 s	--	--	88 3 600 s	76 50 s	95 160 s	27 14 s	26 4.9 s	--	44 44 s	26 170 s	29 150 s
ProductLines 597 tasks, max. score: 929	639 24 000 s	918 6 600 s	928 3 500 s	715 3 100 s	--	--	928 7 500 s	905 950 s	925 2 600 s	929 1 200 s	347 17 000 s	--	927 4 800 s	0 0.0 s	0 0.0 s
DeviceDrivers64 1428 tasks, max. score: 2766	**2682** **13 000 s**	2463 390 000 s	2613 28 000 s	--	--	--	2358 140 000 s	**2639** **3 000 s**	0 0.0 s	50 9.9 s	980 2 200 s	--	**2642** **5 700 s**	--	0 0.0 s
HeapManipulation 80 tasks, max. score: 135	--	**132** **12 000 s**	107 210 s	71 70 s	--	--	97 970 s	--	**107** **130 s**	**111** **9.5 s**	105 15 s	--	--	--	18 35 s
MemorySafety 61 tasks, max. score: 98	--	4 11 000 s	**95** **460 s**	9 690 s	--	--	-136 1 500 s	--	**38** **170 s**	**14** **39 s**	-130 7.5 s	--	--	--	0 0.0 s
Recursive 23 tasks, max. score: 39	--	**30** **11 000 s**	0 0.0 s	--	--	--	-53 4 900 s	--	3 0.38 s	-18 0.12 s	6 0.93 s	--	--	**12** **850 s**	**9** **54 s**
SequentializedConcurrent 261 tasks, max. score: 364	--	**237** **47 000 s**	97 9 200 s	--	--	--	**244** **38 000 s**	--	**208** **11 000 s**	-46 7 700 s	-32 770 s	--	83 4 800 s	49 3 000 s	9 1 200 s
Simple 45 tasks, max. score: 67	30 5 400 s	**66** **15 000 s**	**67** **430 s**	--	--	--	31 27 000 s	37 830 s	0 0.0 s	0 0.0 s	-22 13 s	--	**67** **480 s**	--	0 0.0 s
Overall 2868 tasks, max. score: 4718	--	**3 501** **560 000 s**	**2 987** **48 000 s**	--	--	--	975 280 000 s	--	**1 843** **24 000 s**	-184 11 000 s	-220 42 000 s	--	--	399 10 000 s	139 7 600 s

Lessons Learnt

- Language features support
- No premature UNKNOWN
- Efficiently ignore irrelevant details
- Engineering Matters

Competition Consideration

Competition Consideration

- Bug Finder vs. Safe Prover

Competition Consideration

- Bug Finder vs. Safe Prover
- Benchmarks – Future, not Status Quo

Competition Consideration

- Bug Finder vs. Safe Prover
- Benchmarks – Future, not Status Quo
  - Pointers, Sets, Maps, not Integers
  - Parallel (?)
  - Function Pointers (?)

Competition Consideration

- Bug Finder vs. Safe Prover
- Benchmarks – Future, not Status Quo
  - Pointers, Sets, Maps, not Integers
  - Parallel (?)
  - Function Pointers (?)
- Error traces

Competition Consideration

- Bug Finder vs. Safe Prover
- Benchmarks – Future, not Status Quo
  - Pointers, Sets, Maps, not Integers
  - Parallel (?)
  - Function Pointers (?)
- Error traces
- Wall time matters

LinuxTesting.org

Thank you!

Alexey Khoroshilov
Vadim Mutilin
http://linuxtesting.org/ldv

ISP RAS
Institute for System Programming of the Russian Academy of Sciences