Towards Thorough Verification of Particular Critical Industrial C Programs

Evgeny Novikov, Anton Vasilyev, Ilja Zakharov
ISP RAS

CPA'21, Online, September 30, 2021

Critical Industrial C Programs

- BIOS and boot loaders
- Hypervisors
- Operating system kernels and drivers
- Libraries
- ...

index : kernel/git/torvalds/linux.git

master switch

Linux kernel source tree

Linus Torvalds

about summary refs log tree commit diff stats

log msg Found by Linux D search

Age	Commit message (Expand)	Author	Files	Lines
12 days	HID: amd_sfh: Fix potential NULL pointer dereference	Evgeny Novikov	1	-1/+5
2021-08-26	usb: musb: musb_dsps: request_irq() after initializing musb	Nadezda Lutovinova	1	-7/+6
2021-08-26	usb: dwc3: imx8mp: request irq after initializing dwc3	Nadezda Lutovinova	1	-7/+7
2021-08-26	usb: ehci-orion: Handle errors of clk_prepare_enable() in probe	Evgeny Novikov	1	-2/+6
2021-08-20	HID: thrustmaster: Fix memory leak in thrustmaster_interrupts()	Evgeny Novikov	1	-0/+1
2021-08-20	HID: thrustmaster: Fix memory leak in remove	Evgeny Novikov	1	-0/+1
2021-08-20	HID: thrustmaster: Fix memory leaks in probe	Evgeny Novikov	1	-1/+4
2021-08-18	usb: gadget: mv_u3d: request_irq() after initializing UDC	Nadezda Lutovinova	1	-9/+10
2021-08-17	mtd: rawnand: intel: Fix error handling in probe	Evgeny Novikov	1	-9/+18
2021-08-04	media: tegra-cec: Handle errors of clk_prepare_enable()	Evgeny Novikov	1	-4/+6
2021-07-22	media: platform: stm32: unprepare clocks at handling errors in probe	Evgeny Novikov	1	-8/+18
2021-07-21	USB: EHCI: ehci-mv: improve error handling in mv_ehci_enable()	Evgeny Novikov	1	-12/+11
2021-06-17	media: marvell-ccic: set error code in probe	Evgeny Novikov	1	-3/+9
2021-06-08	media: st_rc: Handle errors of clk_prepare_enable()	Evgeny Novikov	1	-4/+18
2021-06-02	media: st-hva: Fix potential NULL pointer dereferences	Evgeny Novikov	1	-2/+1
2021-06-02	media: v4l: cadence: Handle errors of clk_prepare_enable()	Evgeny Novikov	1	-1/+7
2021-06-02	media: v4l: cadence: Handle errors of clk_prepare_enable()	Evgeny Novikov	1	-1/+7
2021-05-24	net: appletalk: cops: Fix data race in cops_probe1	Saubhik Mukherjee	1	-2/+2
2021-03-10	net: pxa168_eth: Fix a potential data race in pxa168_eth_remove	Pavel Andrianov	1	-1/+1
2020-12-10	mtd: plat-ram: correctly free memory on error path in platram_probe()	Baskov Evgeiny	1	-3/+8
2020-12-03	media: s5p-jpeg: handle error condition in s5p_jpeg_probe	Baskov Evgeniy	1	-0/+2
2020-11-16	media: isif: reset global state	Evgeny Novikov	1	-2/+9
2020-11-16	media: zr364xx: propagate errors from zr364xx_start_readpipe()	Evgeny Novikov	1	-7/+24
2020-10-27	usb: gadget: goku_udc: fix potential crashes in probe	Evgeny Novikov	1	-1/+1
2020-10-14	drivers: watchdog: rdc321x_wdt: Fix race condition bugs	Madhuparna Bhowmik	1	-3/+2

Verification Results at Checking Memory Safety for Subset of Device Drivers of Linux 5.14-rc1*

- Verdicts
  - Unsafes: 285 (24 bugs, 261 false alarms)
  - Safes: 847
  - Unknowns: 890 (538 timeouts, 119 failures)
- Total code coverage is 15%

* Details of the experiment could be found in my talk at CPA'20 "Recent advances of CPAchecker within Klever"

Verification Results at Checking Memory Safety for Subset of Device Drivers of Linux 5.14-rc1

- Verdicts
  - Unsafes: 285 (**24 bugs**, 261 false alarms)
  - Safes: 847
  - Unknowns: 890 (538 timeouts**,** 119 failures)
- Total code coverage is 15%

Awesome!

Verification Results at Checking Memory Safety for Subset of Device Drivers of Linux 5.14-rc1

- Verdicts
  - Unsafes: 285 (24 bugs, **261 false alarms**)
  - Safes: 847
  - Unknowns: 890 (**538 timeouts, 119 failures**)
- Total code coverage is **15%**

This is not acceptable at verification of particular programs!

Challenges at Verification of Particular Programs

- Unclear program environments and requirements should be formalized
- Verification should not fail and run out of time/memory
- There may be no false alarms due to any reason
- There may be no missed bugs due to verification tools
- Code coverage should be 100%+

```
int array[10];
int func(int index) {
    return array[index];
}
```

- Verification results should be represented in a user-friendly way

Ideas on How to Cope with Challenges

- Develop very accurate and pretty simple environment models and requirement specifications
  - Development and decomposition of environment models (another talk today)
  - Expressing specific requirements
  - Models of error behavior
- Make verification tools extremely precise and fast
  - Using CPAchecker SMG due to active usage of arrays, lists, strings and bit precise operations
  - Fixing and optimizing CPAchecker SMG (another talk tomorrow)
- Facilitate analysis of verification results
  - Representation and analysis of violation witnesses (users should investigate and fix bugs and false alarms immediately after their detection)
  - Representation and analysis of code coverage reports (2 other talks today)

Expressing Specific Requirements (Statement *ldv_assert()*)

```
#ifdef LDV_MEMORY_SAFETY
#define ldv_assert() ({*(char *)0;})
#else
#define ldv_assert() __VERIFIER_error()
#endif

void entry_point(void) {
    ...
    if (...)
        /* ASSERT Something unexpected happened. */
        ldv_assert();
    ...
}
```

Expressing Specific Requirements (Concrete Values)

Original program source file

```
int array[10];
int func(int index) {
    return array[index];
}
```

Requirement specification 1

```
void entry_point(void) {
    array[5] = 5;
    if (func(5) != 5)
        /* ASSERT ... */
        ldv_assert();
}
```

Requirement specification 2

```
int entry_point(void) {
    array[5] = -5;
    if (func(5) != 5)
        /* ASSERT ... */
        ldv_assert();
}
```

Expressing Specific Requirements (Nondeterministic Values)

Original program source file

```
int array[10];
int func(int index) {
    return array[index];
}
```

Requirement specification

```
int entry_point(void) {
    int index = ldv_random_int(0, 9);
    int value = ldv_undef_int();

    array[index] = value;
    if (func(index) != value)
        /* ASSERT ... */
        ldv_assert();
}
```

Models of Error Behavior (Possible Failures of *malloc()*)

Original program source file

```
void func(void) {
    char *array[10];
    int i;

    for (i = 0; i < 10; i++)
        array[i] = malloc(10);

    ...

    for (i = 0; i < 10; i++)
        free(array[i]);
}
```

Environment model

```
int entry_point(void) {
    func();
}
```

Models of Error Behavior (Restrict Possible Number of Failures)

Original program source file

```
void func(void) {
    char *array[10];
    int i;

    for (i = 0; i < 10; i++)
        array[i] = ldv_alloc(10);

    ...

    for (i = 0; i < 10; i++)
        free(array[i]);
}
```

Environment model

```
int entry_point(void) {
    ldv_alloc_fails_num = 1;
    func();
}

int ldv_alloc_fails_num = 0;

void *ldv_alloc(size_t size) {
    if (ldv_alloc_fails_num > 0
        && ldv_undef_int()) {
        ldv_alloc_fails_num -= 1;
        return NULL;
    }
    void *res = malloc(size);
    ldv_assume(res);
    return res;
}
```

Models of Error Behavior (Failure Masks)

Original program source file

```
void func(void) {
    char *array[10];
    int i;

    for (i = 0; i < 10; i++)
        array[i] = ldv_alloc(10);

    ...

    for (i = 0; i < 10; i++)
        free(array[i]);
}
```

Environment model

```
int entry_point(void) {
    ldv_alloc_fail_mask = 5;
    func();
}

int ldv_alloc_fail_mask = 0;

void *ldv_alloc(size_t size) {
    if (ldv_alloc_fail_mask & 1) {
        ldv_alloc_fail_mask <<= 1;
        return NULL;
    }
    ldv_alloc_fail_mask <<= 1;
    void *res = malloc(size);
    ldv_assume(res);
    return res;
}
```

Future Work

- Consider thorough verification of several open source device drivers to evaluate suggested approaches better
- Enhance representation of verification results
- Optimize CPAchecker SMG to handle large loops and lists much more efficiently