LinuxTesting.org

Нахождение всех ошибок в модулях ядра Linux за один запуск верификатора

Мордань Виталий
аспирант 1-го года обучения МГУ ВМК
mordan@ispras.ru

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

Существующие проблемы

Система верификации LDV Tools использует статические верификаторы, которые останавливаются после нахождения первой ошибки

- Единовременно может проверяться только одно правило (проверка всех модулей ядра Linux занимает 1-2 дня)
- **Только одна ошибка может быть найдена для заданного правила и модуля ядра**

Пример программы с двумя ошибками

```
1:  void error()
2:  {
3:          ERROR: goto ERROR;
4:  }
5:  int main()
6:  {
7:          int var_1;
8:          int var_2;
9:          // do something
10:         if (var_1 < 0) {error();}
11:         if (var_2 > 0) {error();}
12:         return 0;
13:  }
```

Реальный пример

Ошибки, найденные с помощью LDV Tools

1. drivers/usb/gadget/inode.c: lack of unlock data->lock mutex on error path in ep_read() – 2011-03-22
2. drivers/usb/gadget/inode.c: lack of unlock data->lock mutex on error path in ep_write() – 2011-06-08

Многоаспектная верификация

Цель – найти все ошибки в заданном модуле ядра Linux, проверив его на всех известных правилах корректности

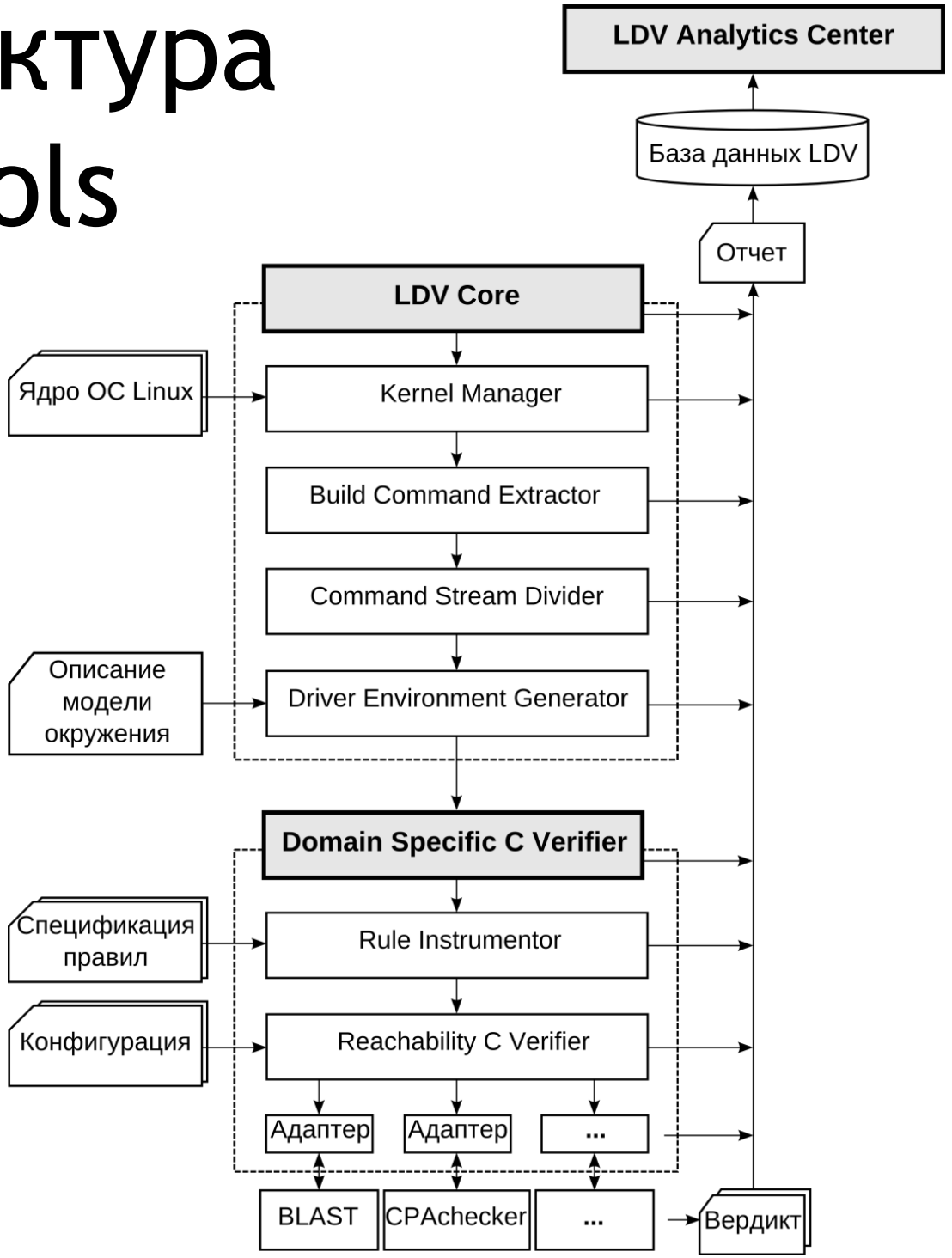
Для начала нужно научиться находить несколько ошибок для одного правила

- Выбрать статический верификатор
- Внести изменения в LDV Tools

Статический верификатор CPAchecker

- Имеет опцию для поиска нескольких ошибок
- Интегрирован в LDV Tools

Архитектура LDV Tools



Измения в компонентах LDV Tools

1. Адаптер для CPAchecker
2. Reachability C Verifier
3. Загрузка отчета в базу данных LDV

Первые эксперименты

1. Обе известные ошибки были найдены
2. Для более глубокого анализа –
   все найденные ошибки на ядре 3.12-rc1
   (20 модулей)

Результаты поиска всех ошибок на 20 модулях ядра Linux

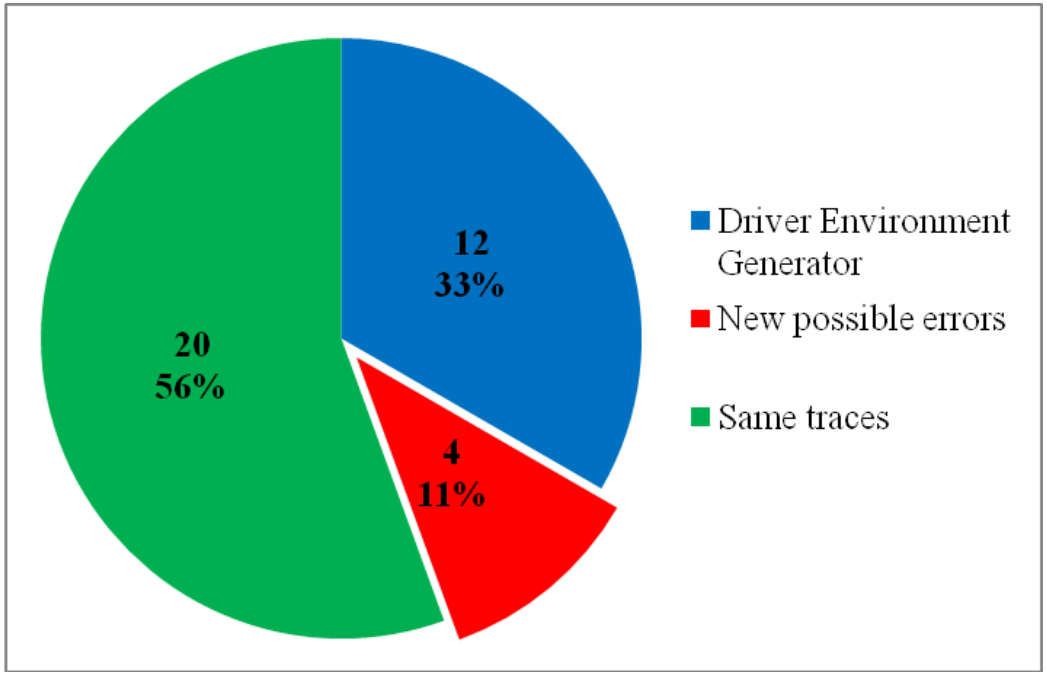
- Для 9 модулей найдено 110 ошибок
- Для 11 – ‘unknown’ (не хватило времени/памяти)

  Причина: CPAchecker выводил трассы ошибок только после завершения анализа.

  - Подготовлен патч для возможности печати трасс ошибок сразу после их нахождения

После применения патча

- Найдено 451 ошибок для 20 модулей
- Ручная разметка причин появления новых ошибок
  - Одни и те же ошибки (одинаковые трассы)
  - Для 12 модулей были выданы ложные сообщения об ошибках из-за некорректной модели окружения
  - Только 4 модуля содержат потенциально новые ошибки (5)
  - Идеальное число ошибок – 25 (20 старых + 5 новых)



Эквивалентность трасс ошибок

Трассы ошибок называются эквивалентными, если они соответствуют одной ошибке

- В случае ложной ошибки – причина ложного срабатывания должна быть одной
- В случае истинной ошибки – ее исправление должно быть одинаково

Все трассы ошибок можно разбить на классы эквивалентности и рассматривать по одной из каждого класса

Алгоритм «дерево вызовов функций»

- Две трассы ошибок эквивалентны, если у них совпадают деревья вызовов функций
- Получено 433 класса эквивалентности для 451 ошибки
- Простой алгоритм, но не оптимальный

Алгоритм «модельные функции»

- Модельные функции содержат основную логику правил корректности в LDV Tools
- Две трассы ошибок эквивалентны, если у них совпадают деревья вызовов модельных функций, в которых каждое поддерево содержит вызов хотя бы одной модульной функции

Алгоритм «модельные функции»

Например, «равные» участки трасс

```
1: f(); // does not call any model functions
2: if (x) {g();} // does not call any model functions
3: else {h();} // does not call any model functions
4: ldv_model_function(); // calls LDV model function
```

и

```
1: ldv_model_function(); // calls LDV model function
```

Алгоритм «модельные функции»

- Получено 164 класса эквивалентности для 451 ошибки
  - Лучше предыдущего (433)
  - Далеко от идеала (25)

Новый Driver Environment Generator

- Найдено 1997 трасс ошибок для 15 модулей, 4 модуля стало «unknown», 1 – «safe»
- Получен 481 класс эквивалентности для алгоритма «модельные функции»
- Причина: изменились вызовы probe функций модулей ядра Linux

Функции «probe»

Типичная ситуация:

1: probe(); // returns -1, probe failed

2: // … any number of same probe calls

3: probe(); // returns -1, probe failed

4: ldv_check_final_state(); // error will be found

Функции «probe»

- Много различных трасс, соответствующих одной ошибке
- Необходимо считать их эквивалентными
- Получено 63 класса эквивалентности для 1997 ошибок
- Практически все ложные срабатывания ушли

Результирующая таблица

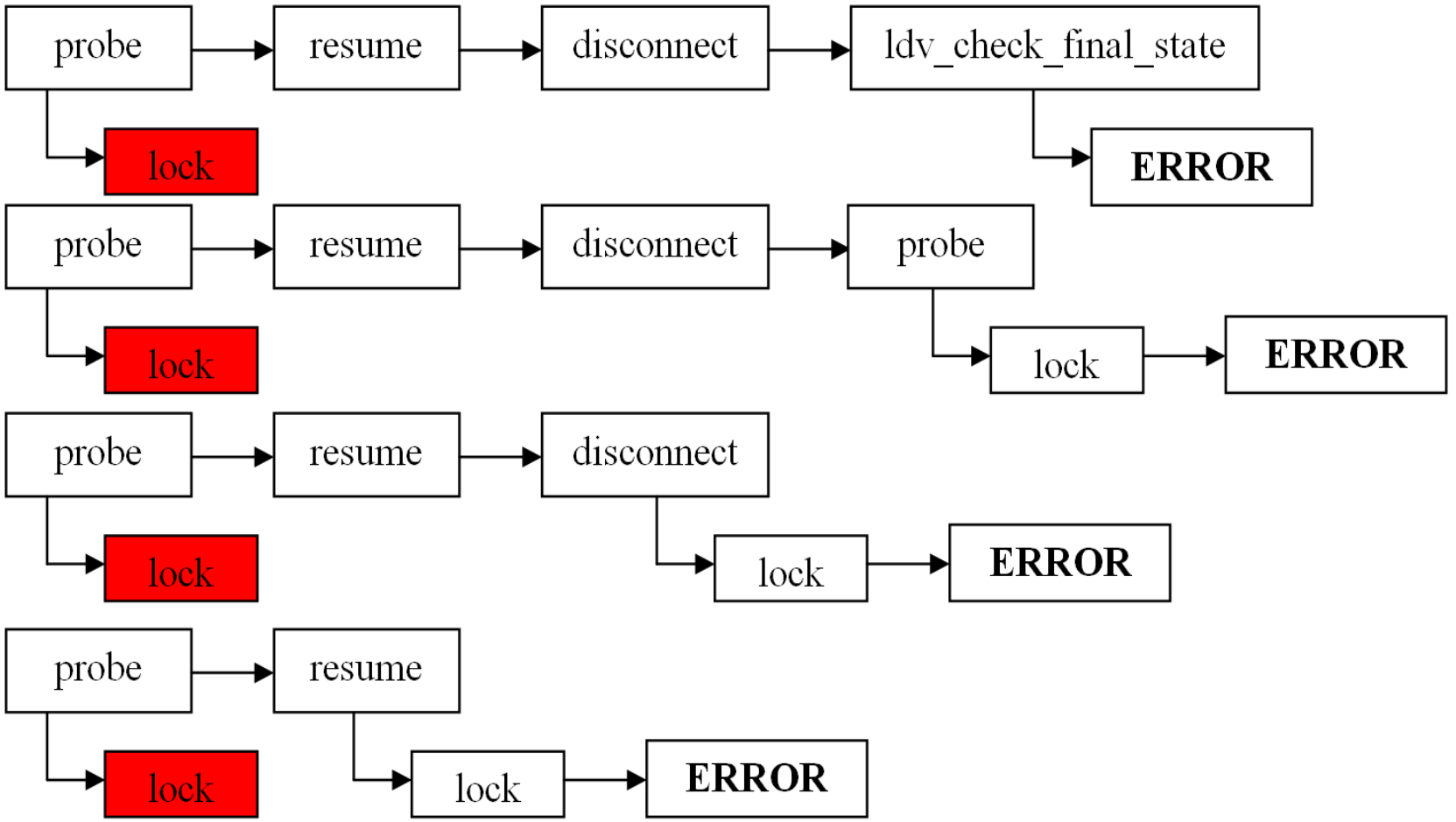
Module	Main	Rule	Time (s)	Memory (Mb)	First run	CPAchecker patch	Tree	Model functions	New DEG	New DEG + MF	New DEG + MF-probe	Manual (ideal)
drivers/net/ethernet/dec/tulip/de4x5.ko	main0	39_7a	137	2717	10	10	8	1	unk.	unk.	unk.	1
net/rxrpc/af-rxrpc.ko	main0	39_7a	900	4208	unk.	6	6	2	6	2	2	1
drivers/media/usb/pvrusb2/pvrusb2.ko	main5	106_1a	95	1046	3	3	3	2	3	2	2	1
drivers/infiniband/hw/nes/iw_nes.ko	main0	129_1a	71	1281	6	6	2	2	unk.	unk.	unk.	2
drivers/input/tablet/gtco.ko	main0	132_1a	900	11000	unk.	14	13	2	65	65	1	1
drivers/isdn/gigaset/bas_gigaset.ko	main0	132_1a	15	398	2	2	2	1	17	2	1	1
drivers/media/rc/imon.ko	main0	132_1a	900	951	unk.	7	7	3	136	15	3	3
drivers/staging/gdm724x/gdmulte.ko	main2	132_1a	33	584	3	3	3	1	4	4	1	1
drivers/staging/gdm72xx/gdmwm.ko	main3	132_1a	33	1170	3	3	3	1	6	5	1	1
drivers/staging/media/as102/dvb-as102.ko	main5	132_1a	900	5009	unk.	18	18	3	2	2	2	1
drivers/staging/media/lirc/lirc_imon.ko	main0	132_1a	900	415	unk.	5	5	2	394	15	1	1
drivers/staging/media/lirc/lirc_sasem.ko	main0	132_1a	900	457	unk.	8	6	2	234	234	1	1
drivers/staging/wlan-ng/prism2_usb.ko	main0	132_1a	900	7467	unk.	6	6	2	1	1	1	1
drivers/usb/misc/ftdi-elan.ko	main0	132_1a	900	8019	unk.	209	204	12	947	47	35	1
drivers/usb/wusbcore/wusb-cbaf.ko	main0	132_1a	900	4400	unk.	107	107	106	154	76	1	1
drivers/net/tun.ko	main0	147_1a	51	2424	51	4	4	2	17	4	4	2
drivers/net/wireless/ath/carl9170/carl9170.ko	main0	147_1a	128	1861	26	26	26	15	11	7	7	1
drivers/net/wireless/ath/carl9170/carl9170.ko	main1	147_1a	900	1739	unk.	4	4	2	unk.	unk.	unk.	1
drivers/scsi/megaraid/megaraid_mm.ko	main0	43_1a	15	251	6	6	2	2	safe	safe	safe	2
drivers/net/wireless/ath/ath9k/ath9k_htc.ko	main1	68_1	735	15000	unk.	4	4	1	unk.	unk.	unk.	1
Total			**172 min.**		**110**	**451**	**433**	**164**	**1997**	**481**	**63**	**25**

Открытые проблемы

- Одна ошибка, которая проявляется позже в нескольких местах
- Получается много трасс ошибок, соответствующих одной ошибке

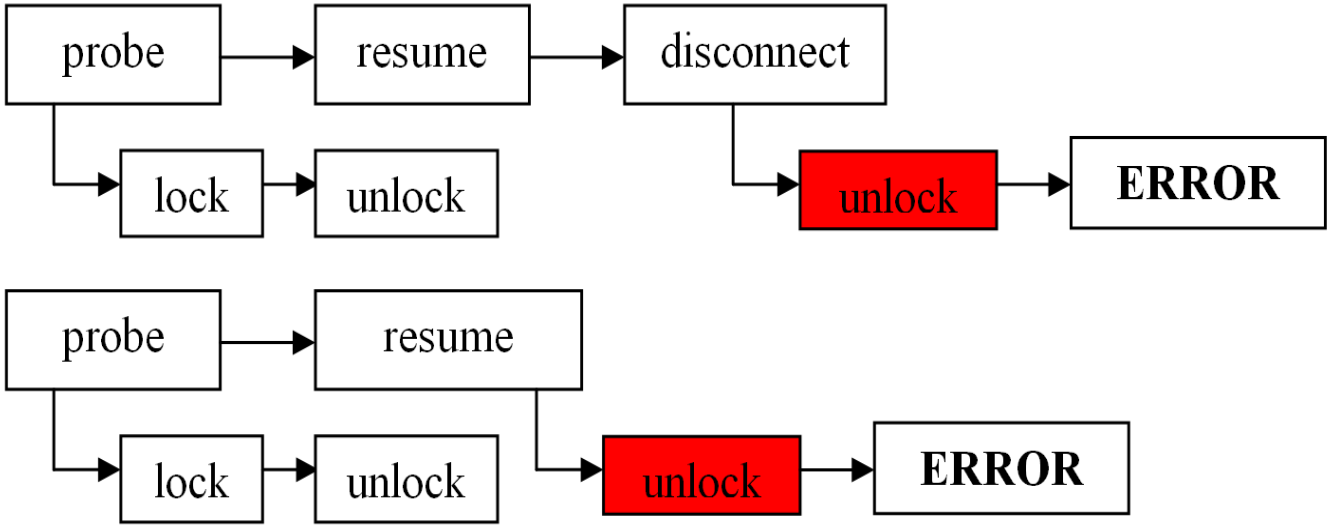
Пример

1 ошибка в probe, 4 разных трассы:



Открытые проблемы

2 различные ошибки и 2 трассы:



- Как различать подобные ситуации?

Возможное решение

Полуавтоматический подход

- Эксперт находит ошибку в первой трассе
- Некий скрипт удаляет из рассмотрения все трассы с данной ошибкой, и процедура повторяется

Планы

- Найти все возможные проблемы подхода (подобно «probe»)
- Максимально возможно приблизить результат к идеальному
- Проанализировать все возможные новые ошибки
- Обобщить подход для анализа нескольких правил сразу (многоаспектная верификация)

Заключение

- Проведены первые эксперименты в области поиска нескольких ошибок (на основе LDV Tools и CPAchecker)
- Предложенный подход уже может быть использован для нахождения многих ошибок в модулях ядра Linux
- Планируется дальнейшее улучшение подхода и обобщение на идеи многоаспектной верификации

LinuxTesting.org

Спасибо за внимание

Мордань Виталий
mordan@ispras.ru

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