ИСП РАН

Анализ, трансформация программ и кибербезопасность

Арутюн Аветисян
Институт системного программирования
им. В.П. Иванникова РАН
arut@ispras.ru
07.04.2020

Причины бурного роста ИТ в последние 10 лет (I)

Социум – основной заказчик инноваций.

Социальные медиа, поисковики, мобильные платформы.

Распространение СПО.

Linux, GCC, LLVM, QEMU, Valgrind, SFX, V8, Hadoop, Spark, Apache Ignite, Infinispan, Hazelcast.

Производство эффективной аппаратуры из компонентов общего назначения.

Высокопроизводительные системы, использование больших данных.

Причины бурного роста ИТ в последние 10 лет (II)

Игровые приставки → суперкомпьютеры

Графические акселераторы (**GPU)** используются в суперкомпьютерной области.

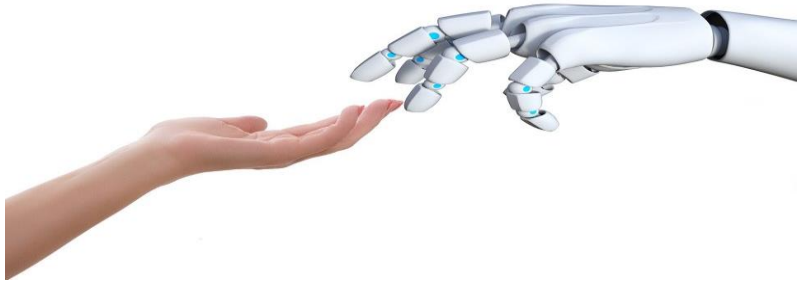
История процессора Cell (IBM, Sony, Toshiba):

- первое коммерческое применение — в игровой консоли Sony Play Station 3;
- в 2009 году самым высокопроизводительным компьютером в мире был признан IBM Roadrunner (построен по гибридной схеме из процессоров AMD Opteron и IBM Cell 8i);
- разработка процессоров Cell была свёрнута.

Поисковая машина → стек технологий для работы с большими данными (Hadoop)

Рост ИТ → развитие цифровой экономики

Цифровая экономика – мировой тренд, отражающий ускоренное внедрение ИТ во все сферы жизнедеятельности.



Уровень ИТ определяет конкурентоспособность как отдельных отраслей, так и экономики в целом

Цифровая экономика: приложения

Непрерывный доступ в сеть интернет/интранет

Киберфизические возможности

Большая вычислительная сложность

Умные устройства (дом, офис, завод)

Платформы «интернета вещей», искусственного интеллекта, …

Платформы хранения и обработки «больших» данных

Облачные платформы

Аппаратура

Проблемы современного системного ПО:

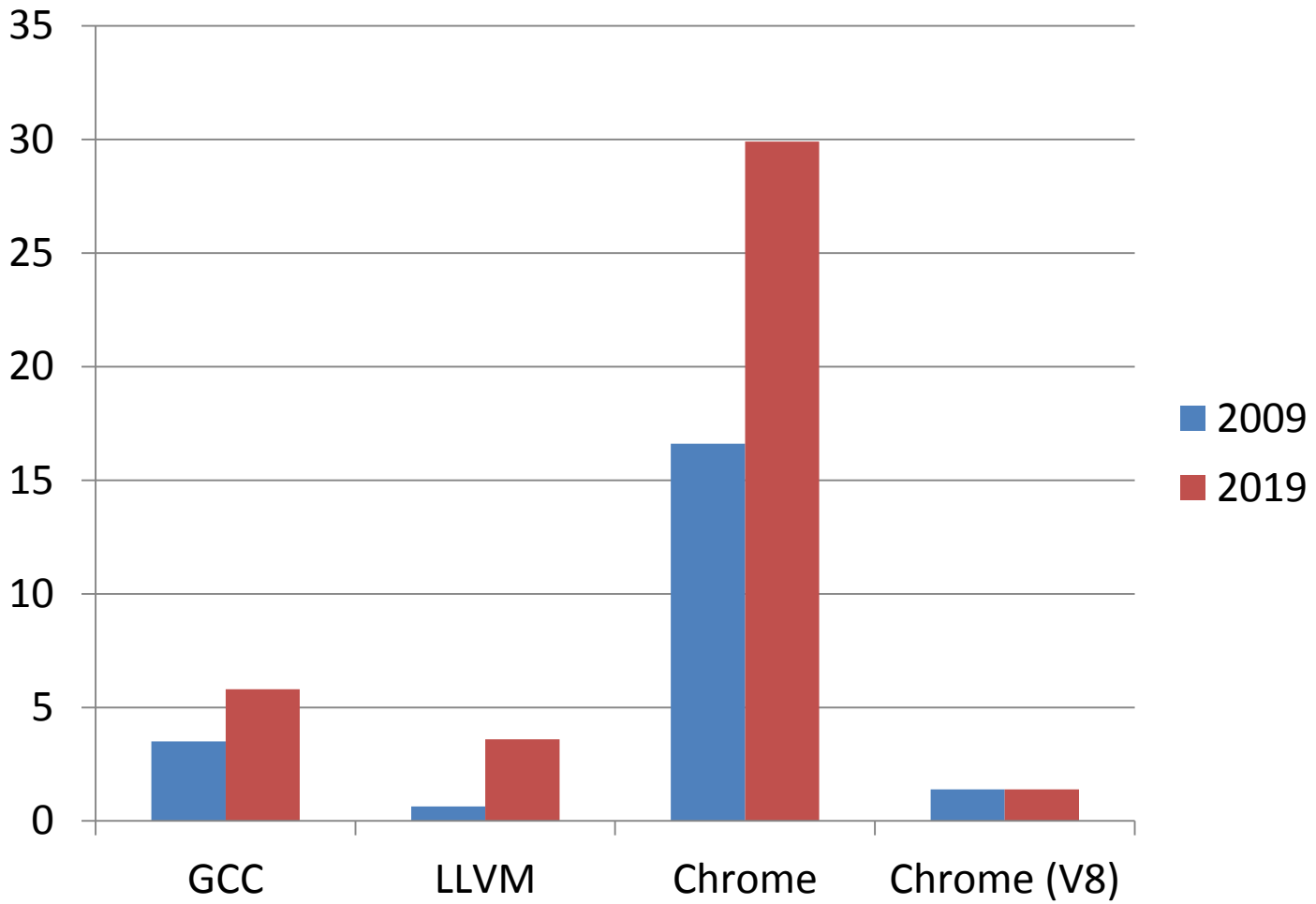
1. Эскалация размеров (Astra Linux – более 150 миллионов строк кода).
2. Сложность среды разработки и сборки.
3. Отсутствие изолированных систем.

Необходимые качества системного ПО:

- Эффективность
- Продуктивность
- Безопасность

Мобильные приложения – более **100 млрд.** загрузок, более **2 млн.** приложений в каждом из Google Market / iOS App Store, более **100 тыс.** приложений добавляется каждый месяц (данные на 2018-2019 гг.)

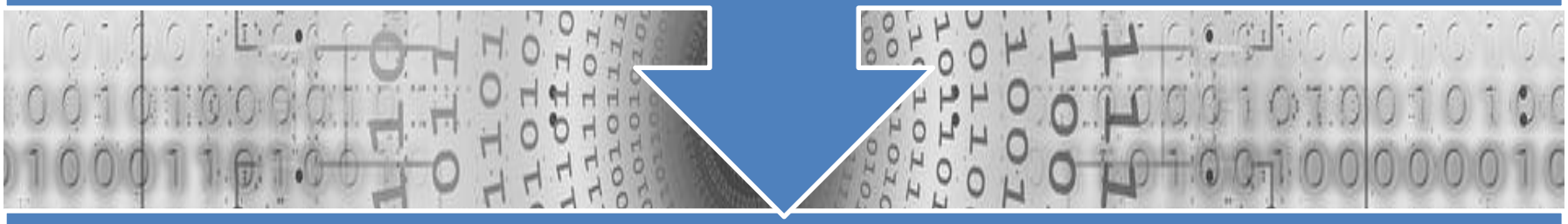
Рост размеров и сложности системного ПО



Кризис программного обеспечения: обострение

Термин веден Фридрихом Л. Бауэром (Friedrich L. Bauer) на Конференции НАТО «Инженерия программного обеспечения» в 1968 году.

Описывает проблему создания продуктивного, безопасного и эффективного ПО.



*«… нет вычислительной техники — нет проблем с разработкой программного обеспечения… **сейчас у нас есть гигантские компьютеры, и программирование стало столь же гигантской проблемой**».*

Эдсгер В. Дейкстра, «Смиренный программист» (1972)

Долговременный вызов

Три главных качества ПО:

Безопасность

Продуктивность

Эффективность

?

Дефекты – причина киберугроз

Принципиальное наличие **дефектов*** в ПО и аппаратуре:

- функциональные;
- архитектурные;
- дефекты программного кода/микрокода.

***Границы между ошибками программиста, закладками и НДВ размыты!**

Утечка информации о дефектах

Хакеры «для интереса»

- Цель – известность;
- Ограниченные ресурсы;
- Только известные эксплойты.

Хакеры «для ущерба»

- Цель – разрушение, вандализм;
- Ограниченные ресурсы.

Организованная преступность

- Цель – прибыль;
- Существенные ресурсы;
- Объединение в синдикаты;
- Специально разработанные программы для кражи данных.

Спецслужбы

- Кибервойны, промышленный шпионаж;
- Практически неограниченные ресурсы;
- Сложные вирусы и программы для взлома;
- Постоянные угрозы.

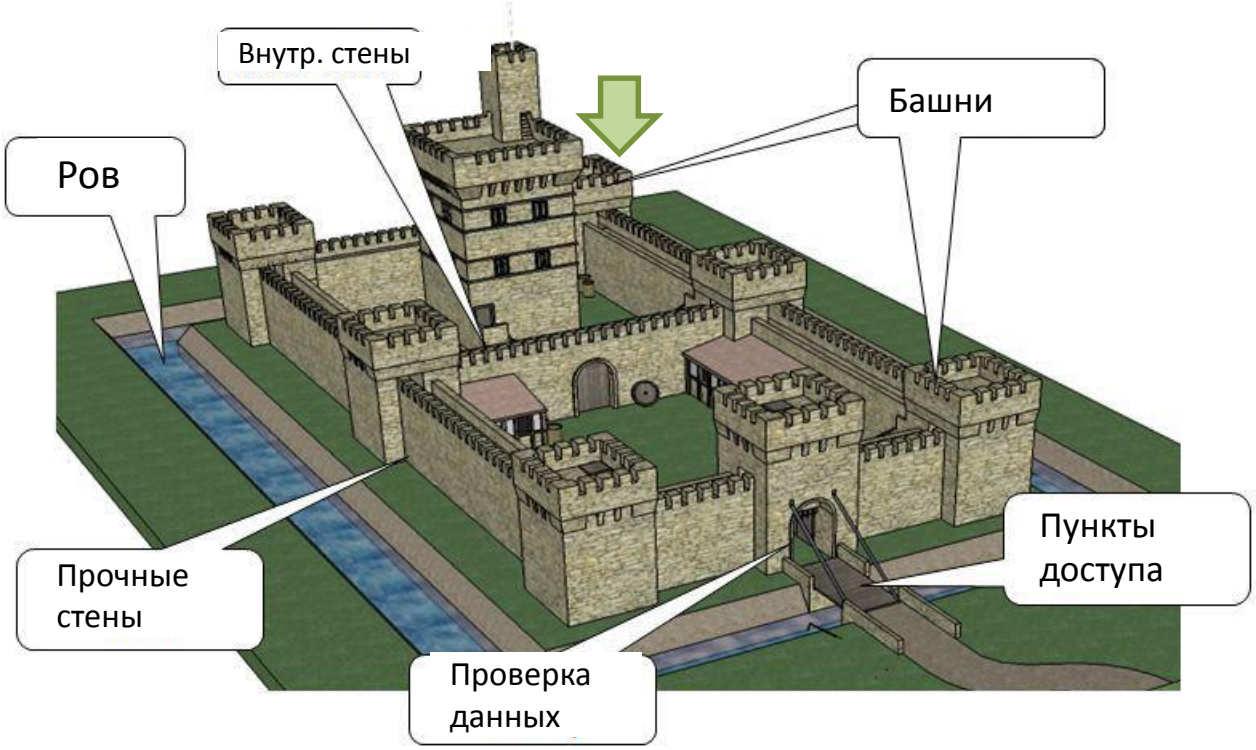
Рост ресурсов и сложности атак

Дефекты: существенный рост ущерба

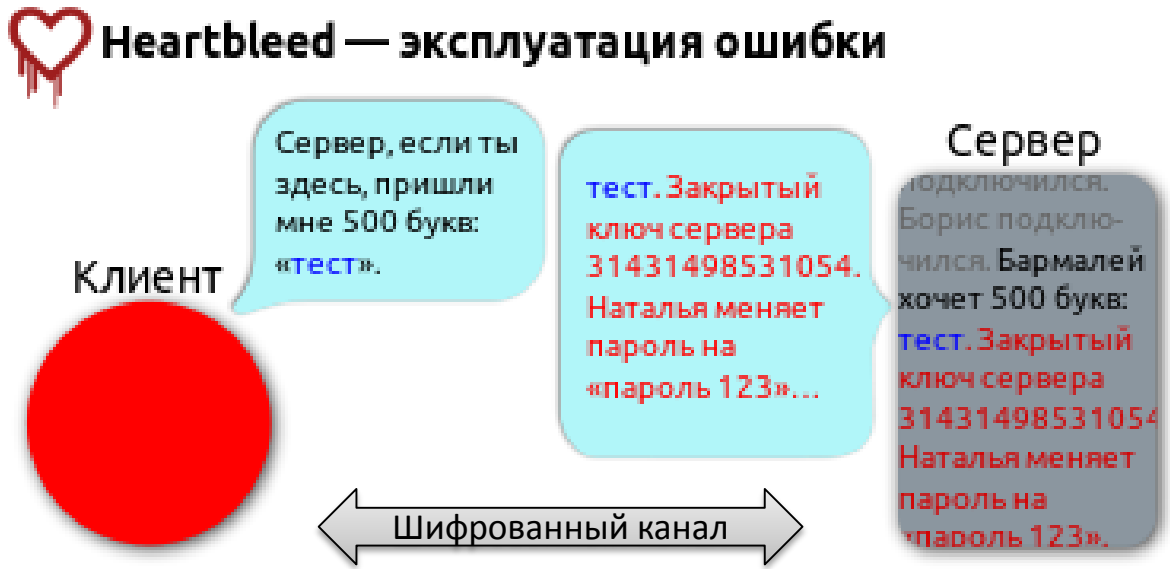
- Затраты на кибербезопасность около $2 трлн. в 2019 году (16-кратный рост по сравнению с 2013 годом, 4-кратный по сравнению с 2015 г.)
- Угрозы кибербезопасности могут иметь последствия в физическом мире, ставя под угрозу жизнь и здоровье

Классические методы защиты не являются основными

- Защита по периметру
- Организационные мероприятия (проверка доступа)
- Антивирусы и др.



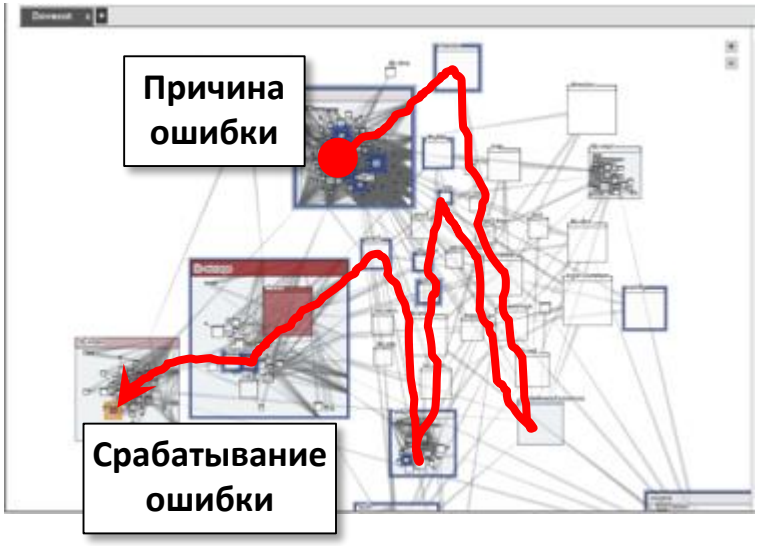
Уязвимость HeartBleed (библиотека OpenSSL) (500000 сайтов заражено, $500 млн. потерь)



- Ошибка чтения данных за границей буфера: злоумышленник контролирует длину посланного текста
- Происходит утечка пользовательских данных
- Весь обмен данных строго следует зашифрованному протоколу

Пример ошибки в современном ПО, приводящей к уязвимости

- Типы ошибки – Слабость кодирования обработки входных данных, Переполнение буфера



- Прежде чем достичь места реализации ошибки, введённые извне данные «проходят» по многим функциям разных модулей

Модуль с функцией считывания файла-архива.

```
[tainted] Call of read
95          if(read(fd, file_hdr, SIZEOF_NEWLHD) != SIZEOF_NEWLHD) {
96              free(file_hdr);
97              return NULL;
98          }
99          file_hdr->flags = unrar_endian_convert_16(file_hdr->flags);
100         file_hdr->head_size = unrar_endian_convert_16(file_hdr->head_size);
101         file_hdr->pack_size = unrar_endian_convert_32(file_hdr->pack_size);
102         file_hdr->unpack_size = unrar_endian_convert_32(file_hdr->unpack_size);
103         file_hdr->file_crc = unrar_endian_convert_32(file_hdr->file_crc);
Composite 'file_hdr' taints element 'file_hdr->name_size'
104         file_hdr->name_size = unrar_endian_convert_16(file_hdr->name_size);
105         if(file_hdr->flags & 0x100) {
...
116         return file_hdr;
```

Функция в другом модуле. Ранее считанные извне данные определяют размер копируемой

```
1484        /* Enter response type, length and copy payload */
1485        *bp++ = TLS1_HB_RESPONSE;
1486        s2n(payload, bp);
Tainted data from /home/shimnik/openssl/ssl/s3_pkt.c+239 reached a sink.
8. [SINK] *(s->s3->rrec.data + @ ) reaches the sink
⚠ 1487      memcpy(bp, pl, payload);
1488        bp += payload;
1489        /* Random paddi
1490        RAND_pseudo_byt
1491
1492        r = dtls1_write                    EAT, buffer, 3 + payload + paddi
```

Go to declaration
Go to definition
Show usage

Ключевое направление противодействия киберугрозам

Разработка новых моделей, методов и соответствующих технологий системного программирования *, позволяющих проводить глубокий *анализ и трансформацию ПО* с целью:

- **найти и устранить** на этапе разработки максимальное количество ошибок в исполняемом коде (жизненный цикл разработки безопасного ПО)
- **обеспечить устойчивость** программной системы, затруднив эксплуатацию существующих ошибок или смягчить последствия их эксплуатации, после внедрения программы

*** Наличие и зрелость такого комплекса технологий фактически определяет уровень кибербезопасности программно-аппаратных систем, создаваемых абсолютно во всех приоритетных направлениях научно-технологического развития России («сквозная» технология в цифровой экономике)**

Анализ и трансформация ПО
Прикладные задачи

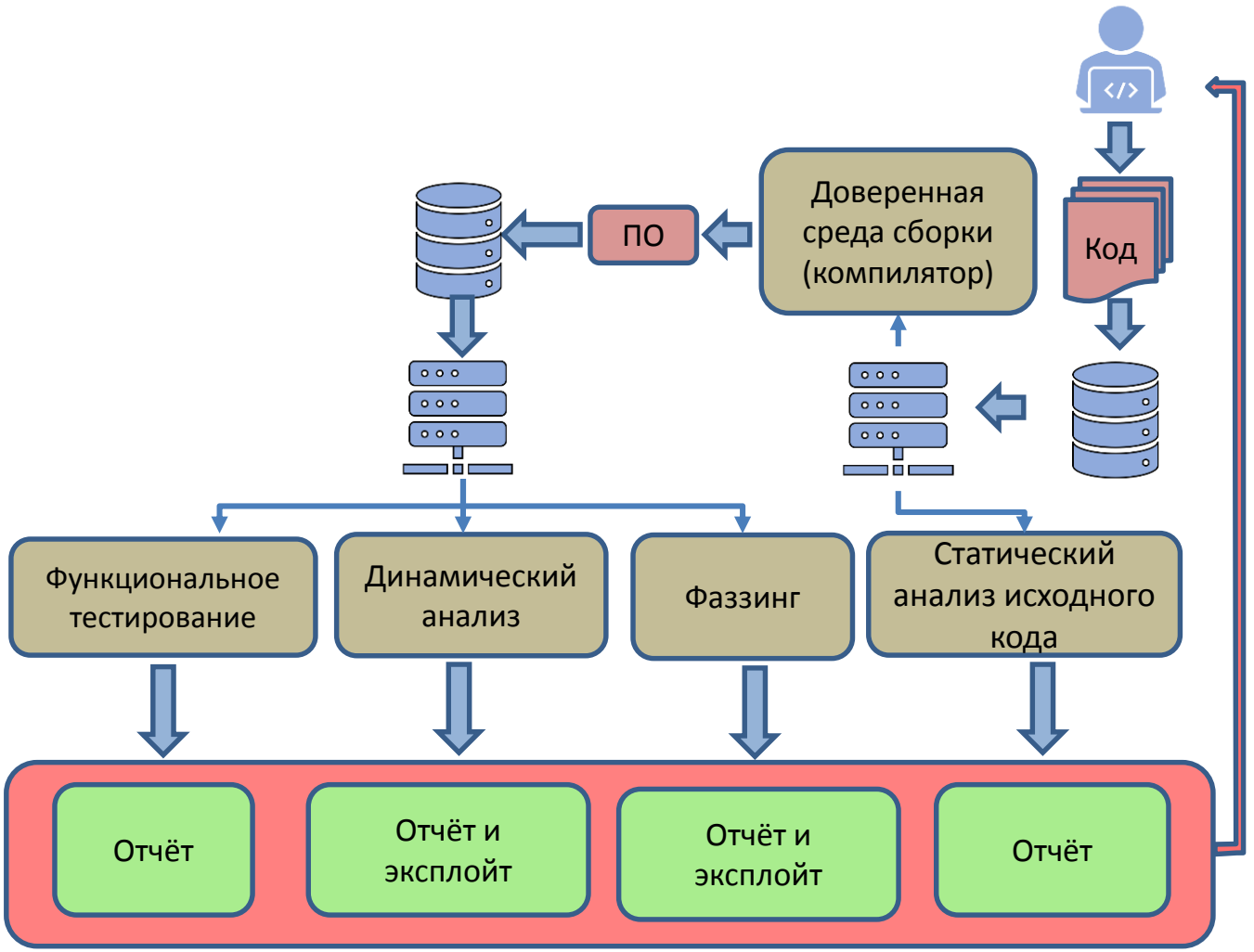
ИСП РАН

Жизненный цикл разработки безопасного ПО	Аудит безопасности целевого ПО	Сертификация и СПСИ	Противодействие эксплуатации

Фундаментальные модели и алгоритмы для создания базовых технологий:

разработки виртуальных машин	построения контролируемой среды выполнения	анализа бинарного (исполняемого) кода	компиляции и анализа исходного кода	анализа времени выполнения	построения доверенных платформ
Восстановление интерфейсов аппаратуры	Детерминированное воспроизведение и приложения на его базе	Статический анализ	Статический анализ - поиск ошибок	Гомоморфное шифрование	Взломостойкие компоненты аппаратуры
Автоматизация разработки ВМ	Инструментация кода	Динамический анализ	Статический анализ помеченных данных	Безопасная компиляция и доверенная среда выполнения	Доверенная среда выполнения
Интеграция аппаратуры в ВМ	Трассировка/ отладка	Фаззинг	Подбор входных данных для воспроизведения ошибки	Обфускация и диверсификация кода	Верифицированная модель контроля доступа
Верификация аппаратуры и эмуляторов	Интроспекция ВМ	Символьное выполнение	Формальная верификация	Песочницы и HoneyPot	Аппаратура с открытым исходным кодом
		Оценка критичности дефектов			
		Модели алгоритмов			

Технологии с точки зрения жизненного цикла разработки безопасного ПО



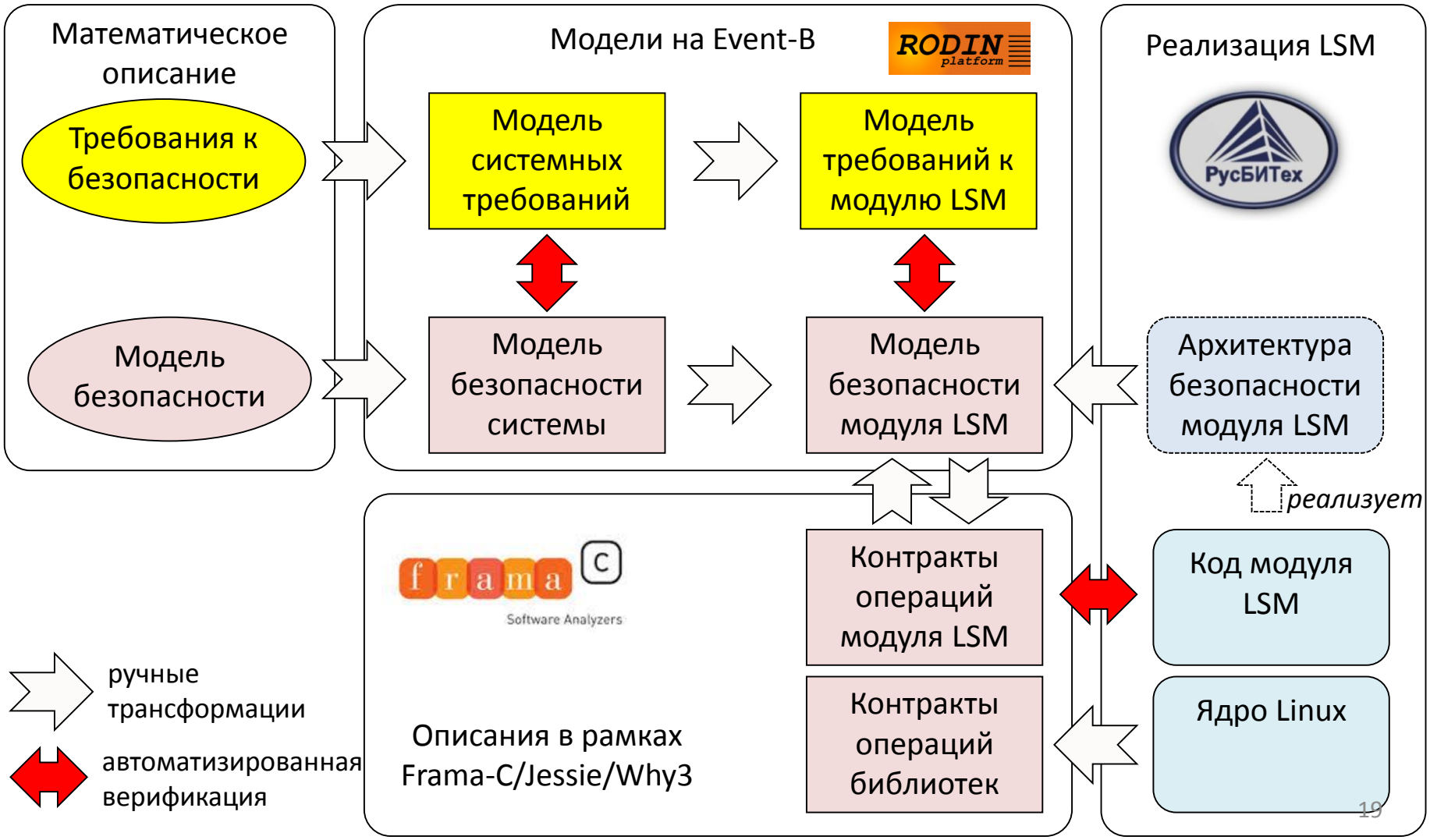
Примеры соответствующих технологий

ИСП РАН, прошедших путь

от идеи до внедрения в индустрию

Дедуктивный анализ

Верификация модели безопасности AstraLinux РусБИТех – ИКСИ – ИСП РАН



Статический анализ исходного кода

Технологии:

1. Доверенный компилятор
2. Инструменты статического анализа программ.

Доступные на рынке инструменты:

- **Svace** **(ИСП РАН, Россия)**
- Klocwork (RogueWave, США)
- Coverity (Synopsys, США)
- Java FindBugs (Maryland University, США)
- Pyhton: PyLint, PyType
- JavaScript: JSHint, JSLint, Google Closure Linter
- …

Svace: основной инструмент статического анализа в Samsung

***Svace** – необходимый инструмент жизненного цикла разработки безопасного ПО.*

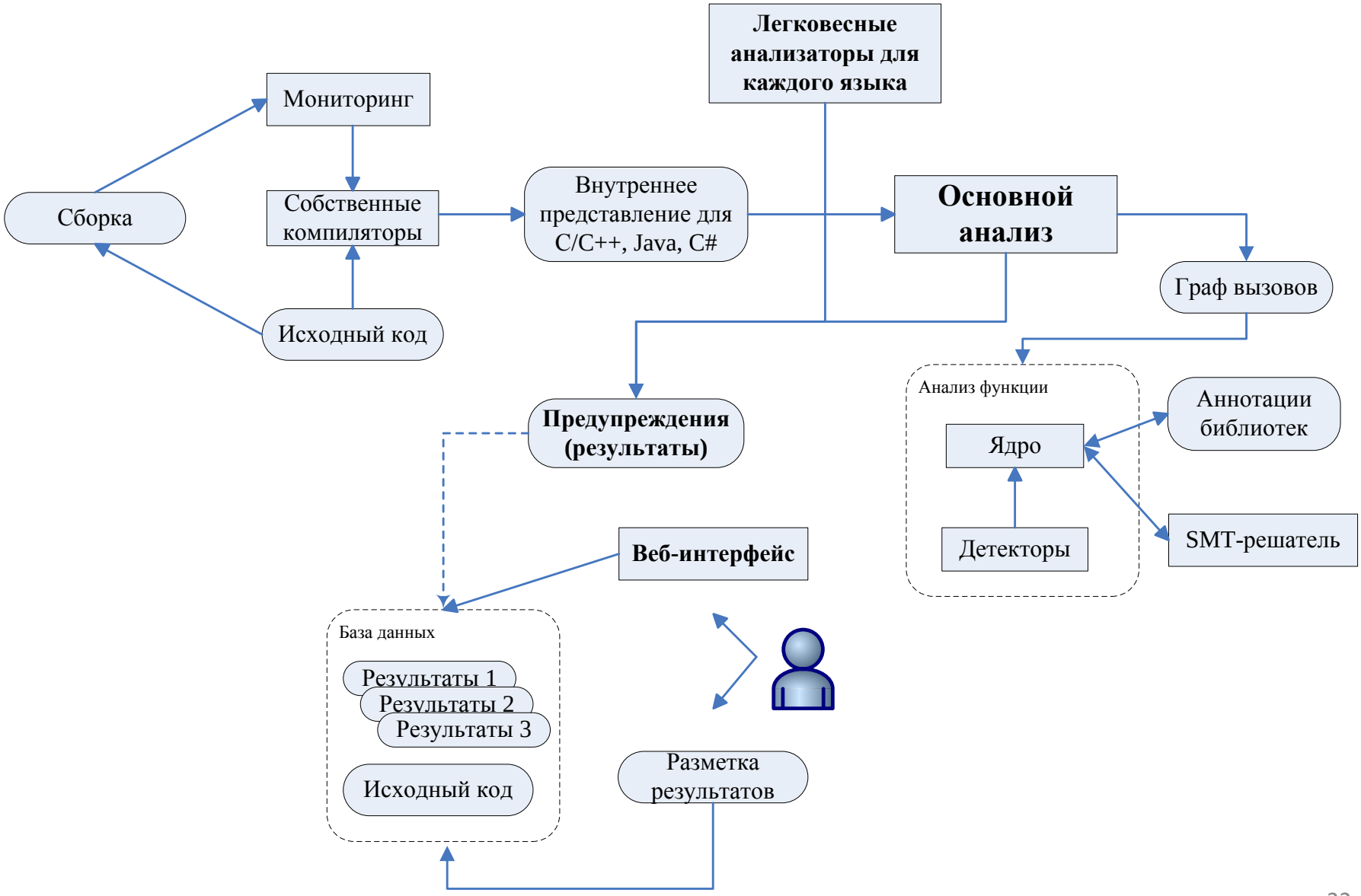
Обнаруживает более 50 классов критических ошибок в исходном коде.

Поддерживает языки C, C++, C#, Java. Добавление поддержки Kotlin и Go планируется в конце 2020 г.

Включён в Единый реестр российского ПО (№4047).

Разрабатывается с 2003 года

Архитектура Svace



Доверенная компиляция

- Уязвимости в программе могут появляться не только из-за ошибок в ее коде, но и в результате оптимизаций, выполняемых компилятором
- В набор средств для обеспечения безопасности ПО должны быть также включены средства разработки (компилятор)

Пример небезопасной оптимизации (1/2)

```
char * password = malloc(PASSWORD_SIZE);
// ... read and check password
memset(password, 0, PASSWORD_SIZE);
free(password);
```

- С точки зрения компилятора, запись нулей в массив с паролем является избыточной, т.к. далее в программе нет обращения к этому массиву, поэтому вызов функции *memset()* может быть удален в ходе оптимизации
- Удаление операций очистки массива может привести к утечке конфиденциальных данных

Пример небезопасной оптимизации (2/2)

```
char *buf = ...;
char *buf_end = ...;
unsigned int len = ...;
if (buf + len >= buf_end)
  return;
/* len too large */
if (buf + len < buf)
  return;
/* overflow, buf+len wrapped around */
/* write to buf[0..len-1] */
```

Проверка на принадлежность `buf+len` границам области `[buf, buf_end]`

- При выполнении оптимизаций компилятор в соответствии со стандартом языка может полагаться на отсутствие в программе неопределенного поведения
- Выделенный участок кода полагается на ***переполнение*** при выполнении арифметических операций с указателями, что является ***неопределенным поведением*** в соответствии со стандартом языка Си
- Указанная проверка может быть удалена при выполнении оптимизаций компилятором, как избыточная
- Удаление проверки границ перед записью в память может привести к появлению эксплуатируемой уязвимости

Доверенный компилятор (C/C++)

- Безопасная оптимизация кода
  - Отсутствие внесения дополнительных уязвимостей на этапе компиляции ПО (в т.ч. оптимизация кода с *неопределенным поведением*)
- Диагностика
  - Выдача предупреждений о потенциально небезопасном коде (напр., -Wextra-safety)
- Снижение степени потенциальной угрозы безопасности
  - Применение методов защиты на этапе компиляции (запутывание кода, sanitizers)

Фаззинг и динамический анализ

Технологии:

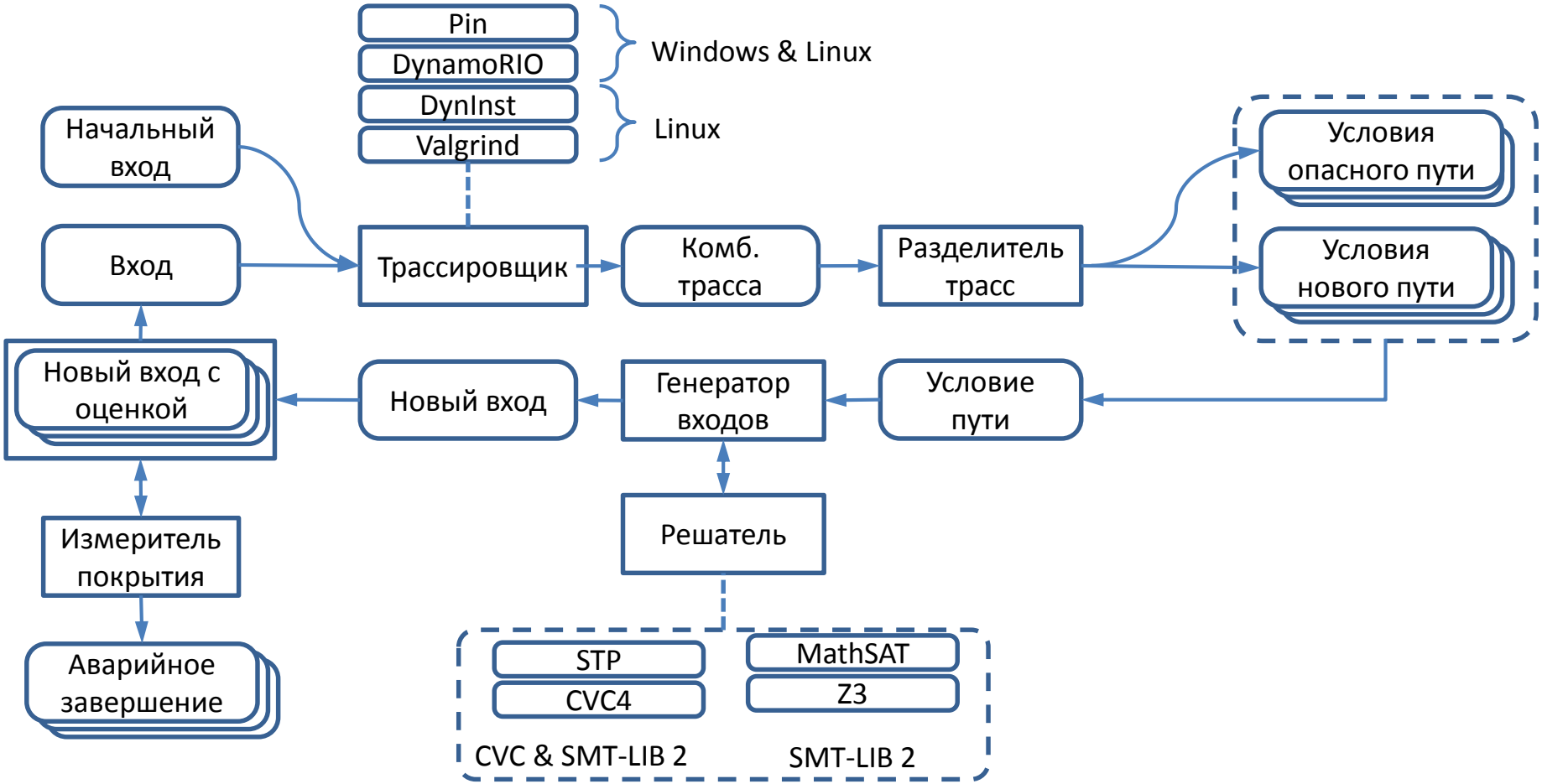
1. Фаззинг + Sanitizers
2. Динамическое символьное исполнение
3. Комбинированные инструменты.

Доступные на рынке инструменты:

- **Anxiety** **(ИСП РАН, Россия)**
- **Crusher** **(ИСП РАН, Россия)**
- MAYHEM (ForAllSecure, США)
- Peach Fuzzer (США, Peach Tech)
- Synopsys Defensics (Synopsys, США)
- angr (Open source)
- Americal Fuzzy Lop (Open source)
- Driller (Open source)
- …

Фаззинг и динамический анализ

Anxiety



ТРАЛ: инструмент анализа бинарного кода

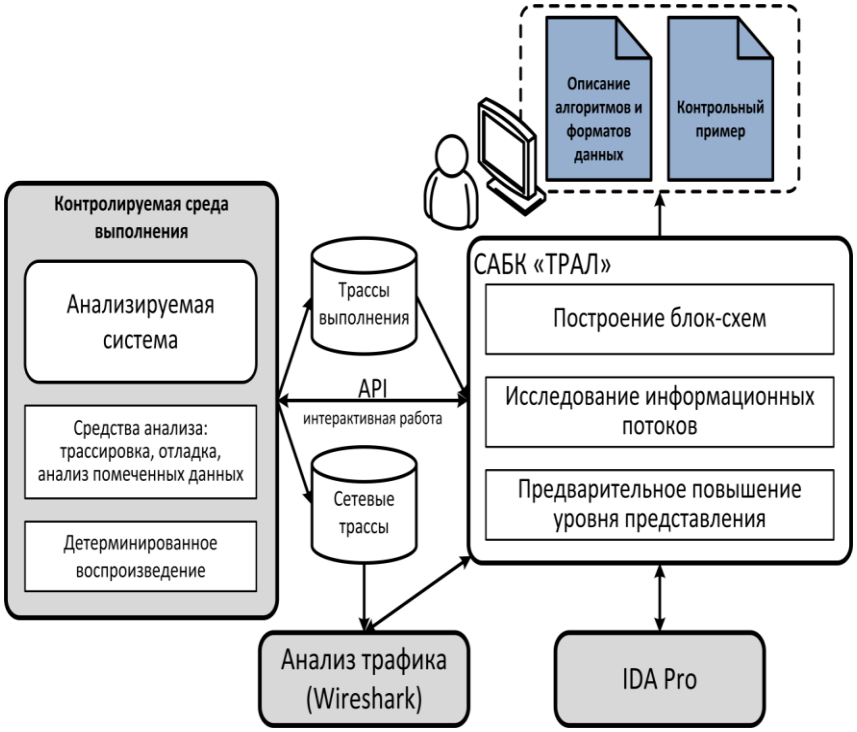
ТРАЛ *– уникальный промышленный инструмент для анализа свойств бинарного кода.*

Позволяет работать с кодом различных целевых процессорных архитектур.

Не требует наличия отладочной информации и исходных кодов.

Применим для анализа всего программного стека от загрузчика до прикладного ПО.

Включён в Единый реестр российского ПО (№5323).



Разрабатывается с 2011 года

Технологии ИСП РАН жизненного цикла разработки доверенного ПО

Разра-ботка

- **Svace и Binside: статический анализ исходного и бинарного кода**
- **Поиск клонов кода: ошибки, нарушения лицензии**
- **Дедуктивный анализ моделей безопасности**

Тестиро-вание

- **Контролируемое выполнение на базе эмулятора QEMU**
- **Crusher (ISP Fuzzer и Anxiety): фаззинг и динамическое символьное исполнение, расширенное тестирование**

Выпуск

- **Доверенный компилятор**
  - Диверсификация кода
  - ИСП–Обфускатор: защита кода от анализа
  - Доработка системных защитных механизмов (ASLR …)

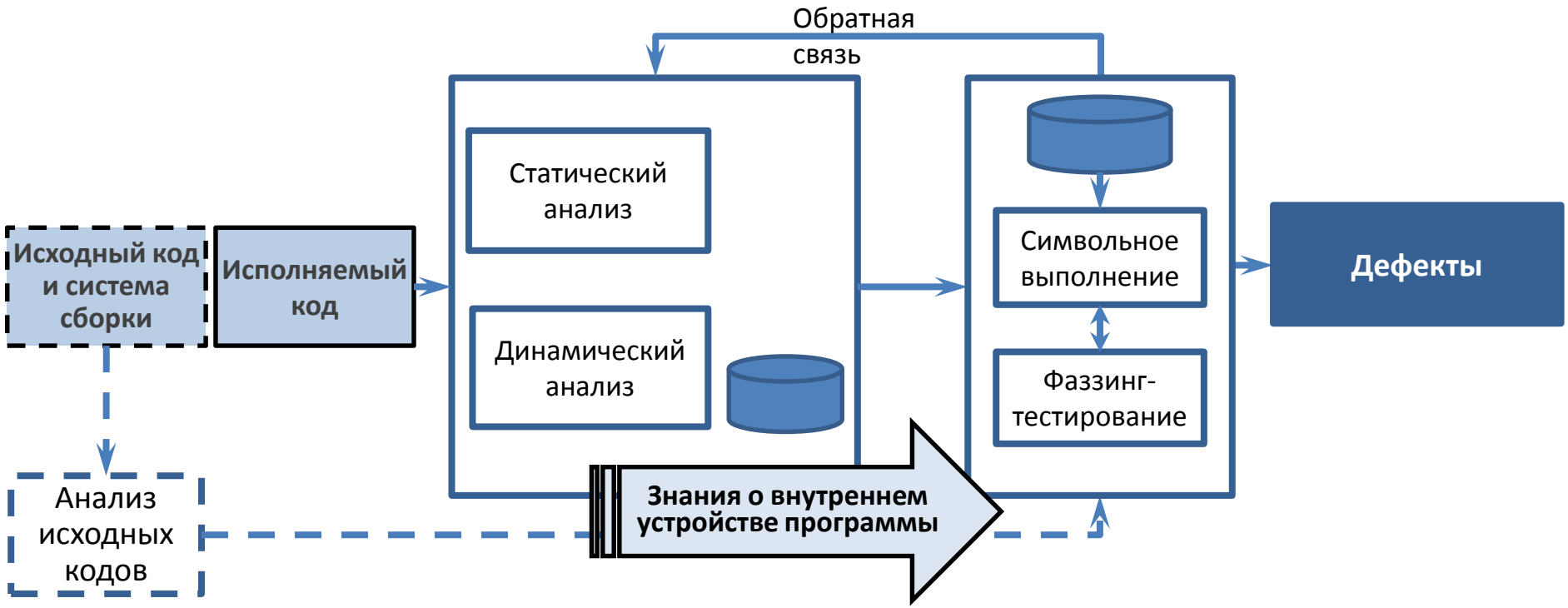
Реагиро-вание

- **Непрерывный поиск дефектов в выпущенном ПО**
- **Анализ аварийных завершений для оценки критичности программных дефектов**

Технологический процесс непрерывного поиска дефектов на этапе эксплуатации

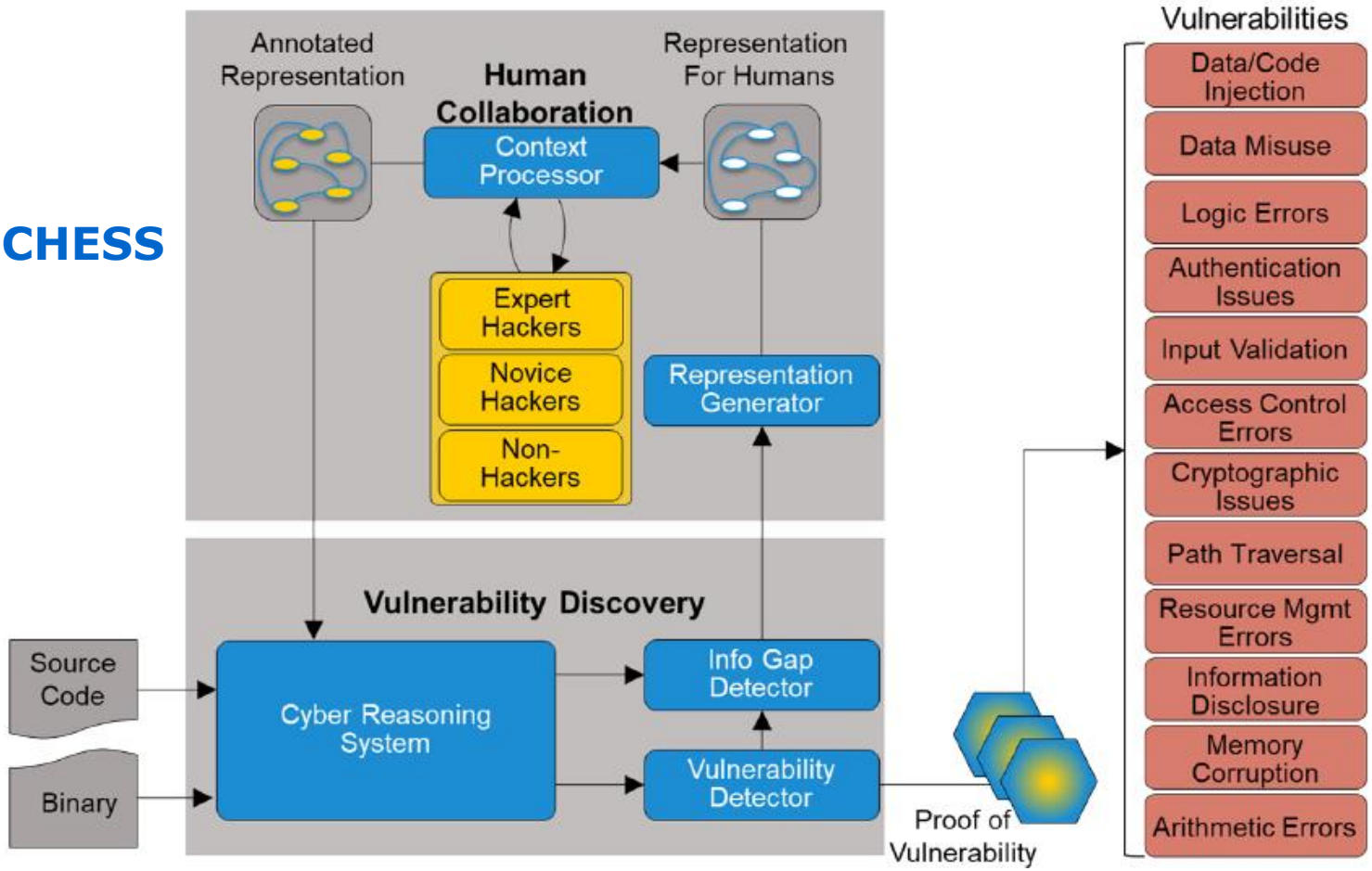
- Нет ограничения по времени
- Требует больших ресурсов

! Может и должен использоваться и после сдачи в эксплуатацию



Зарубежный опыт

DARPA CHESS

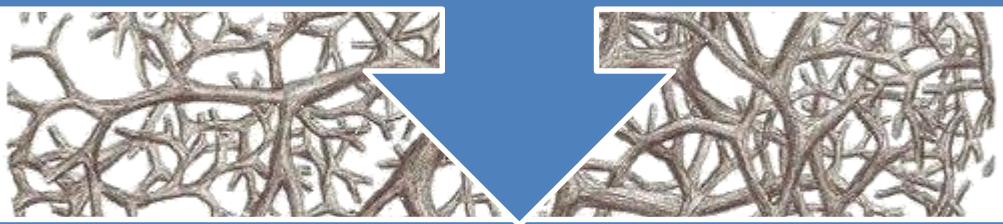


- **DARPA Cyber Grand Challenge** (2016)
  Соревнование по автономному поиску и исправлению уязвимостей в бинарном коде, победитель: Mayhem – Carnegie-Mellon-University

- Проект **DARPA CHESS** (2018, объявлен конкурс)
  Масштабируемый поиск уязвимостей за счет синергии инструментов анализа и экспертных знаний

Сложность систем, быстро меняющиеся требования и новые вызовы требуют постоянных инноваций и комплексного применения всего стека технологий.

Никакая отдельно взятая компания или даже государство не в состоянии поддерживать необходимые темпы развития и соответствующий уровень безопасности без существенных рисков.



Мировой тренд – создание некоммерческих организаций для долгосрочного развития технологий.

Примеры некоммерческих организаций

Центр компетенций Linaro

- Занимается оптимизацией свободного программного обеспечения для развития экосистемы ARM.
- Один из пяти крупнейших контрибьюторов ядра Linux; работает над десятками проектов на основе СПО.
- Создан в 2010 году и в 2016 уже насчитывал 36 компаний-участников.
- В продвижение платформы ARM вкладываются такие крупные компании, как IBM, Google, Samsung и др.
- Команда центра насчитывает 300 разработчиков.
- Дважды в год проводится конференция Linaro Connect, которая собирает более 400 участников.

Организация GlobalPlatform

- Занимается стандартизацией цифровых сервисов; нацелена на высокий уровень обеспечения безопасности.
- Пользуется финансовой поддержкой около 100 компаний (в том числе, AMD, Apple, Huawei, Samsung, NTT).
- Объединяет три технических комитета, 15 рабочих групп и 10 аккредитованных лабораторий в Китае, Франции, Испании, Южной Корее, Великобритании и Нидерландах.
- Занимается различными образовательными проектами.
- 20 февраля 2019 года GlobalPlatform подписала меморандум о взаимопонимании с российской ассоциацией «Доверенная платформа», в которую входят «Лаборатория Касперского», «Инфотекс», «Сфера», НПЦ «Элвис», а также ИСП РАН и др.

Спасибо за внимание!