ИСП РАН

Инструментальные средства и технологии выявления уязвимостей в программном обеспечении

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

12 февраля 2020

Уязвимости – причина компьютерных атак

Принципиальное наличие **уязвимостей*** в ПО и аппаратуре

- функциональные
- архитектурные
- программного кода/микрокода

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

Утечка информации о уязвимостях

Хакеры «для интереса»

- Взломы ради известности
- Ограничены ресурсы
- Только известные эксплоиты

Хакеры «для ущерба»

- Вандализм
- Ограничены ресурсы

Преступность

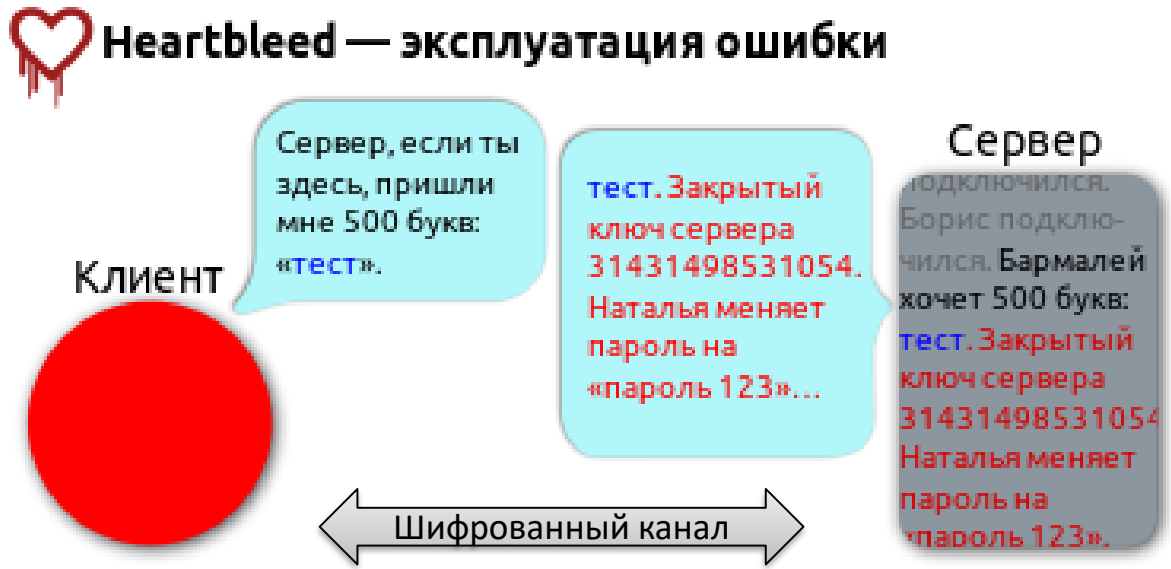
- Взлом ради прибыли
- Существенные ресурсы
- Синдикаты
- Специально разработанные программы для кражи данных

Спецслужбы

- Атаки на критические информационные инфраструктуры, промышленный шпионаж
- Практически неограниченные ресурсы
- Сложное ВПО и программы для взлома
- Постоянные угрозы

Рост ресурсов и сложности атак

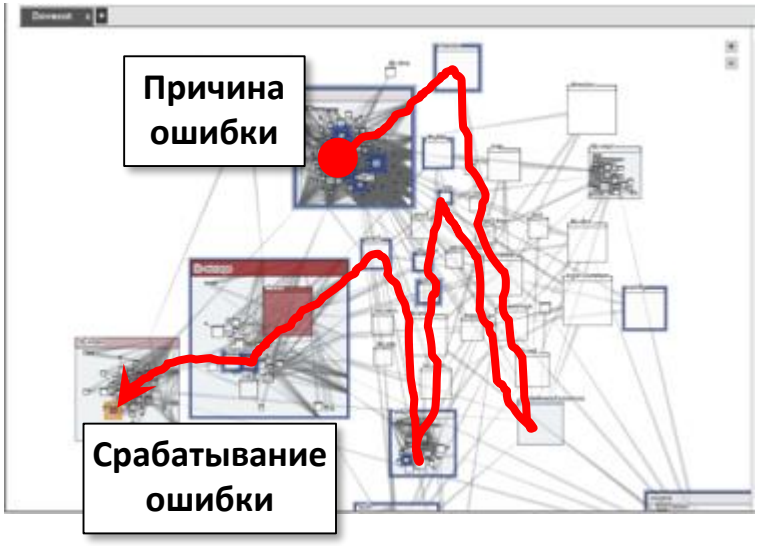
Уязвимость HeartBleed (библиотека OpenSSL) (500000 сайтов заражено, $500 млн. потерь)



- 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);
...
116         return file_hdr;
```

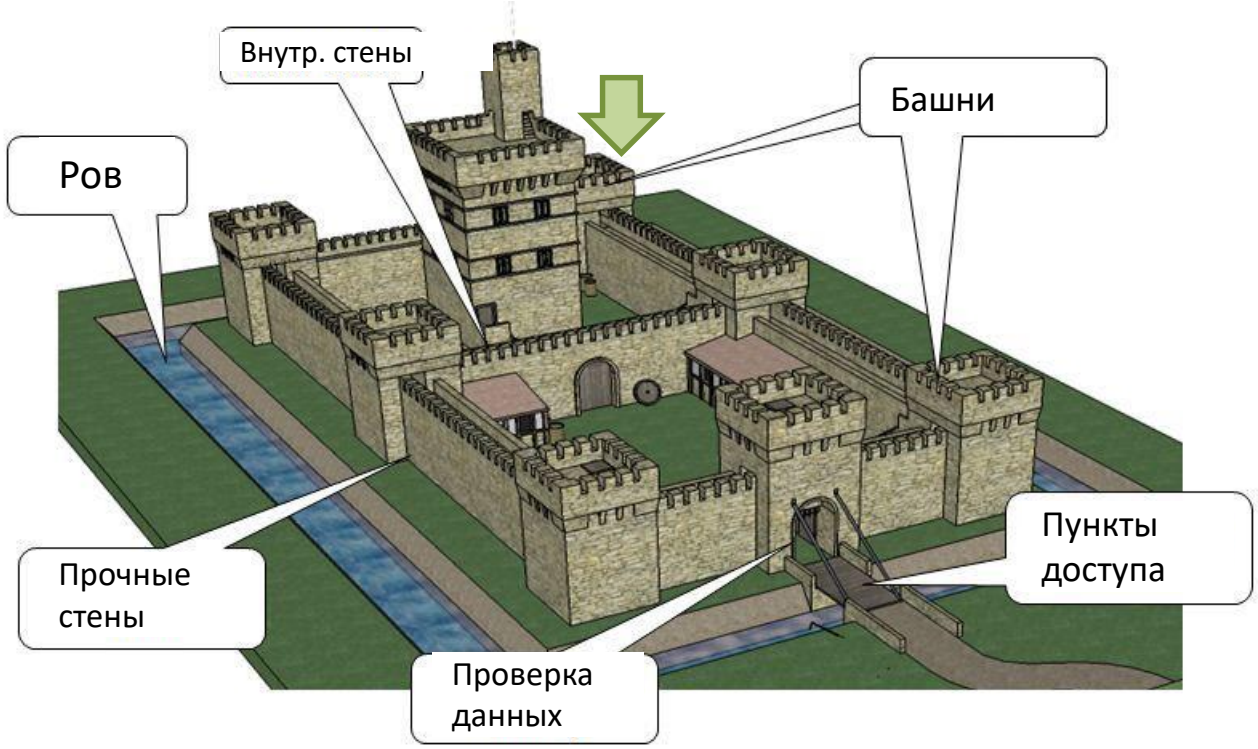
Функция в другом модуле. Ранее считанные извне данные определяют размер копируемой

```
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

Классические методы защиты не являются основными

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



Вызовы

- Эскалация размеров. ПО – большие данные (размер Astra Linux – 150 млн. строк кода, размер Debian – более миллиарда строк)
- Сложность среды разработки и сборки (*например, компилятор может добавить уязвимость, которой нет в исходном коде*)
- Облачно-мобильные платформы, ИИ, «Интернет вещей» (отсутствие локально изолированных систем)…

Зарубежный опыт: разработка безопасного ПО

- Госстандарты: Common Criteria (развитие с 1999), документы МО США (Program Protection Plan для всех этапов разработки, с 2000х)
- Технологии, инструменты анализа выбираются сертификационными лабораториями → разработчики ПО вынуждены пользоваться инструментами требуемого уровня
- Заложено непрерывное обновление стандартов по мере развития технологий (текущая версия CC – 3.1)
- Принятие нормативных документов привело к взрывному росту рынка технологий анализа
  https://samate.nist.gov/index.php/Source_Code_Security_Analyzers.html

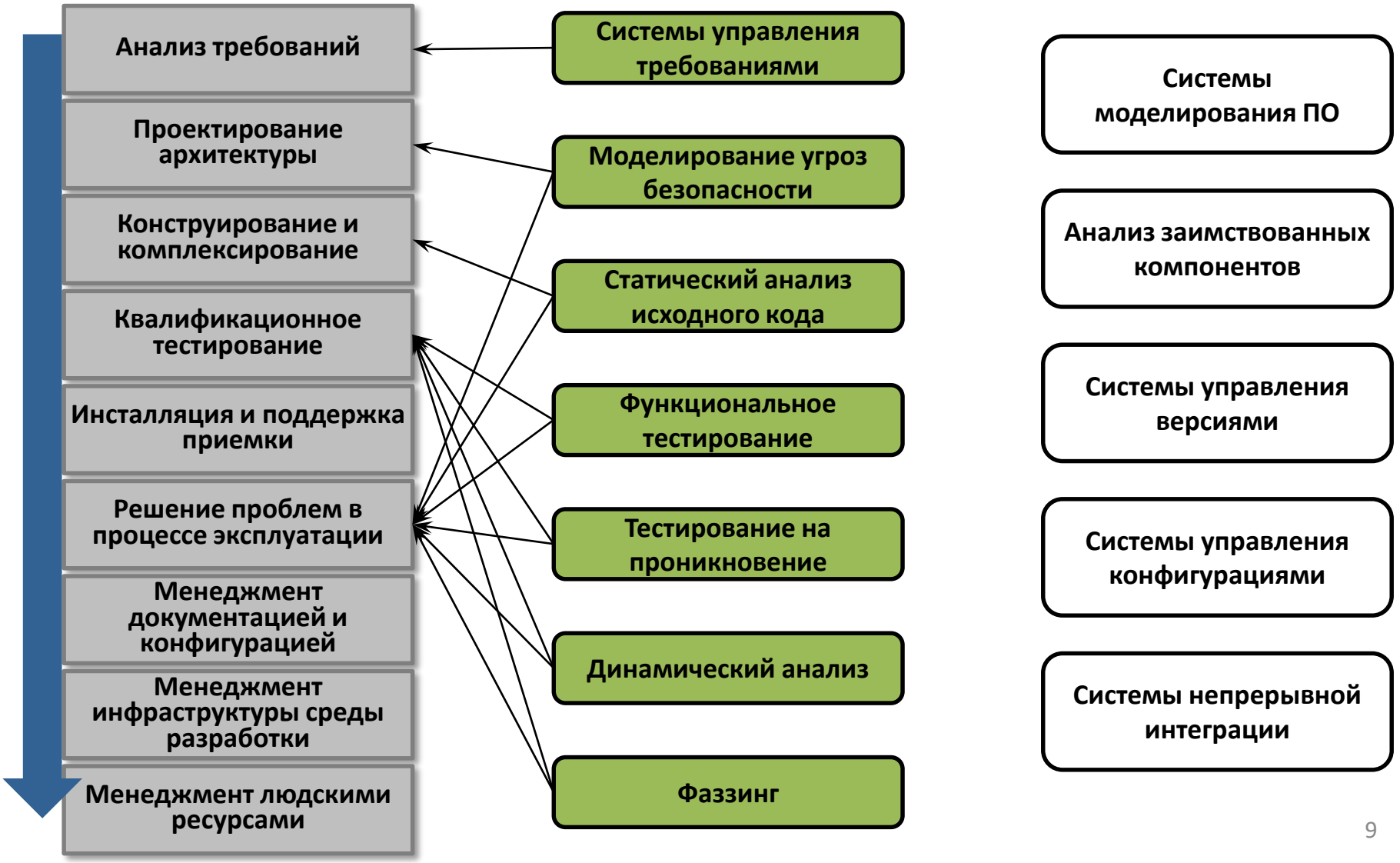
Документы ФСТЭК России

- **ГОСТ Р 56939-2016** «Защита информации. Разработка безопасного программного обеспечения. Общие требования» Утверждён и введен в действие **01.06.2016**

- «**Методика выявления уязвимостей и недекларированных возможностей в программном обеспечении**» утверждена ФСТЭК России 11 февраля 2019 г.
  - Применяется при проведении сертификационных испытаний **с 1 июня 2019 г.**

- ***Задано направление на применение в системе сертификации автоматизированных и автоматических технологий анализа безопасности программного кода***

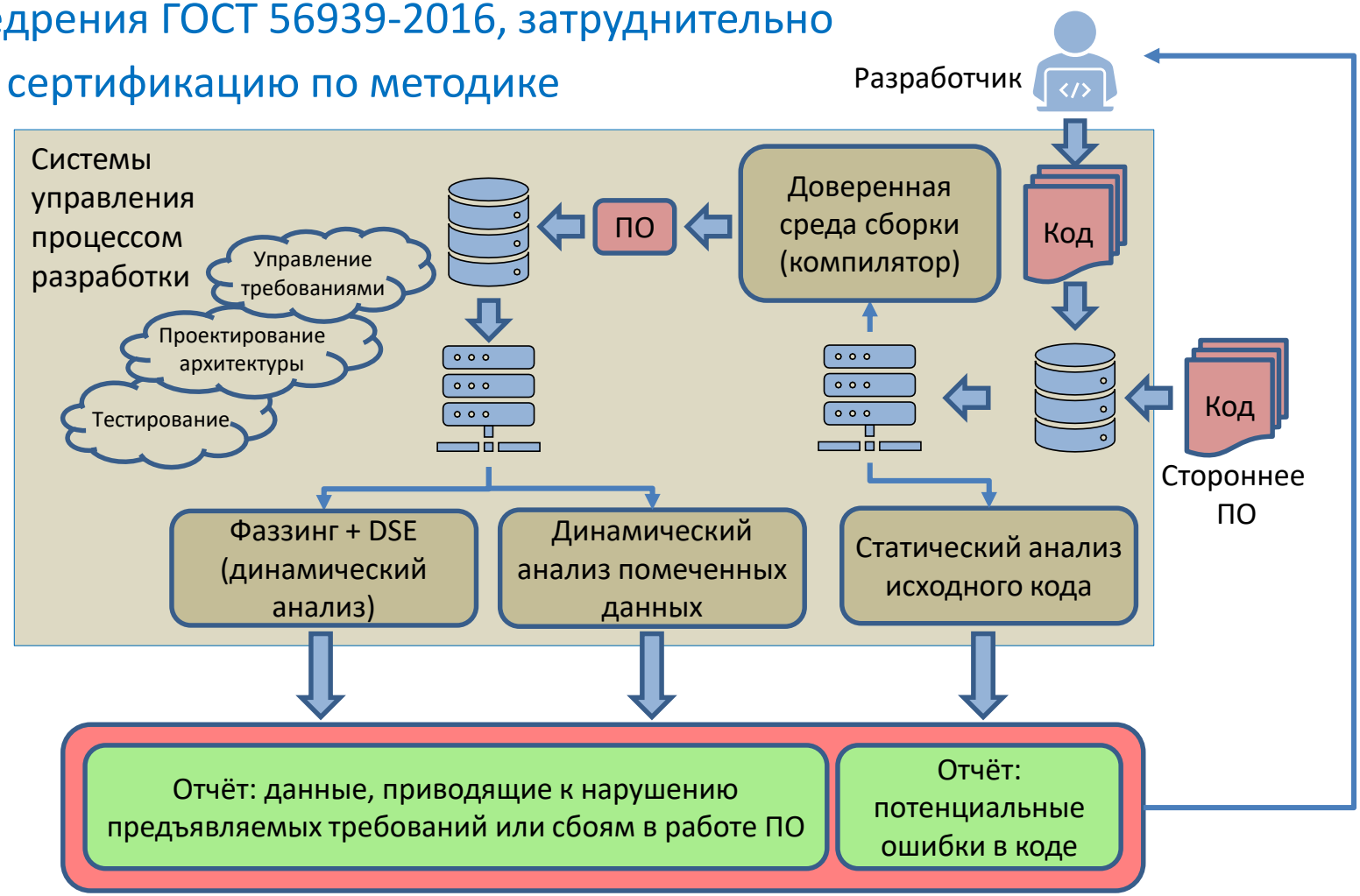
ГОСТ Р 56939-2016



Методика выявления уязвимостей и НДВ

Технологии анализа безопасности программного кода в процессах разработки

Без внедрения ГОСТ 56939-2016, затруднительно пройти сертификацию по методике



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

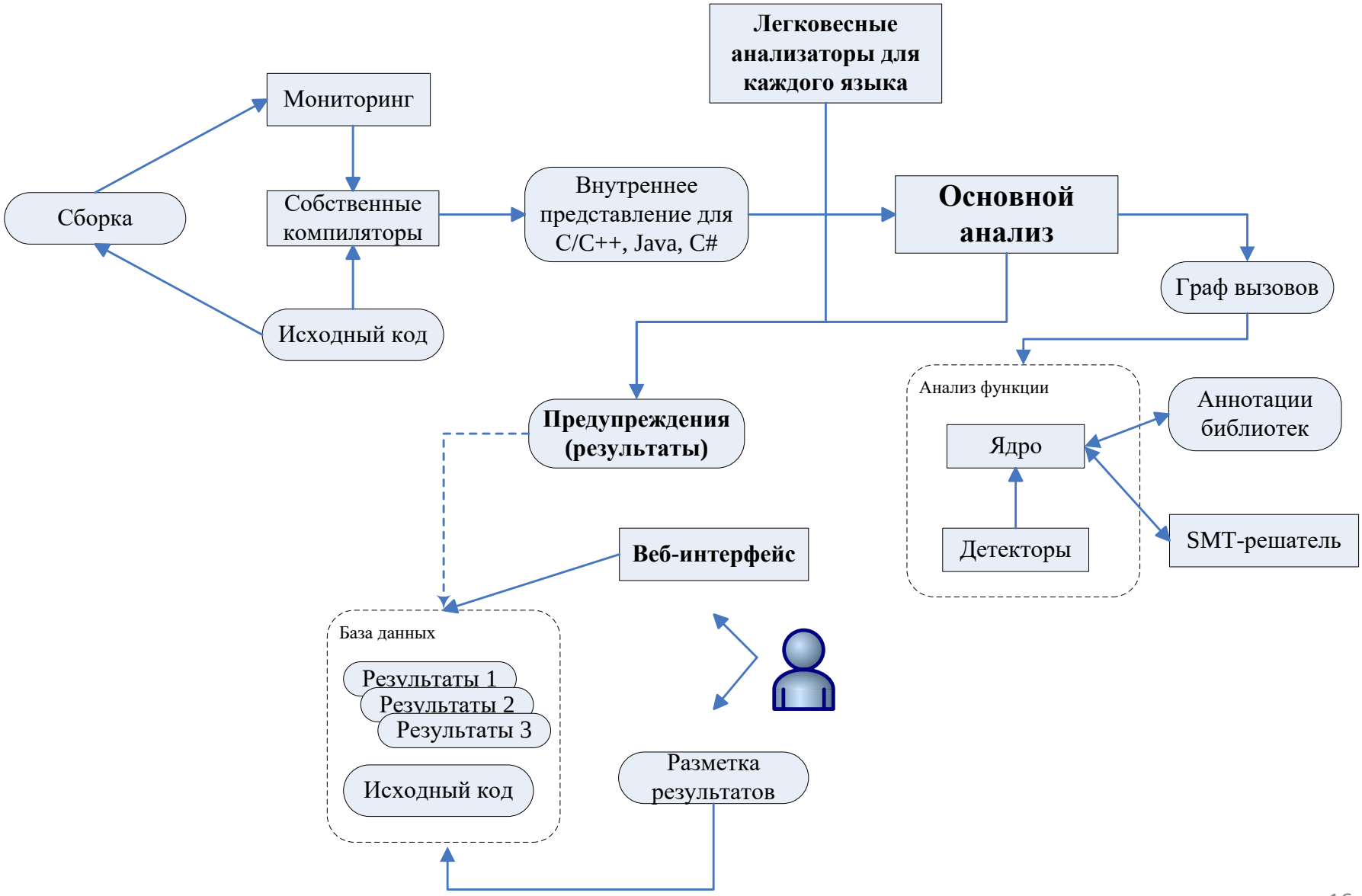
Статический анализ исходного кода

Технологии: перехват сборки, межпроцедурный контекстно-чувствительный анализ, SMT-решатели, …

Доступные на рынке инструменты:

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

Архитектура Svace



Фаззинг и динамический анализ

Технологии:

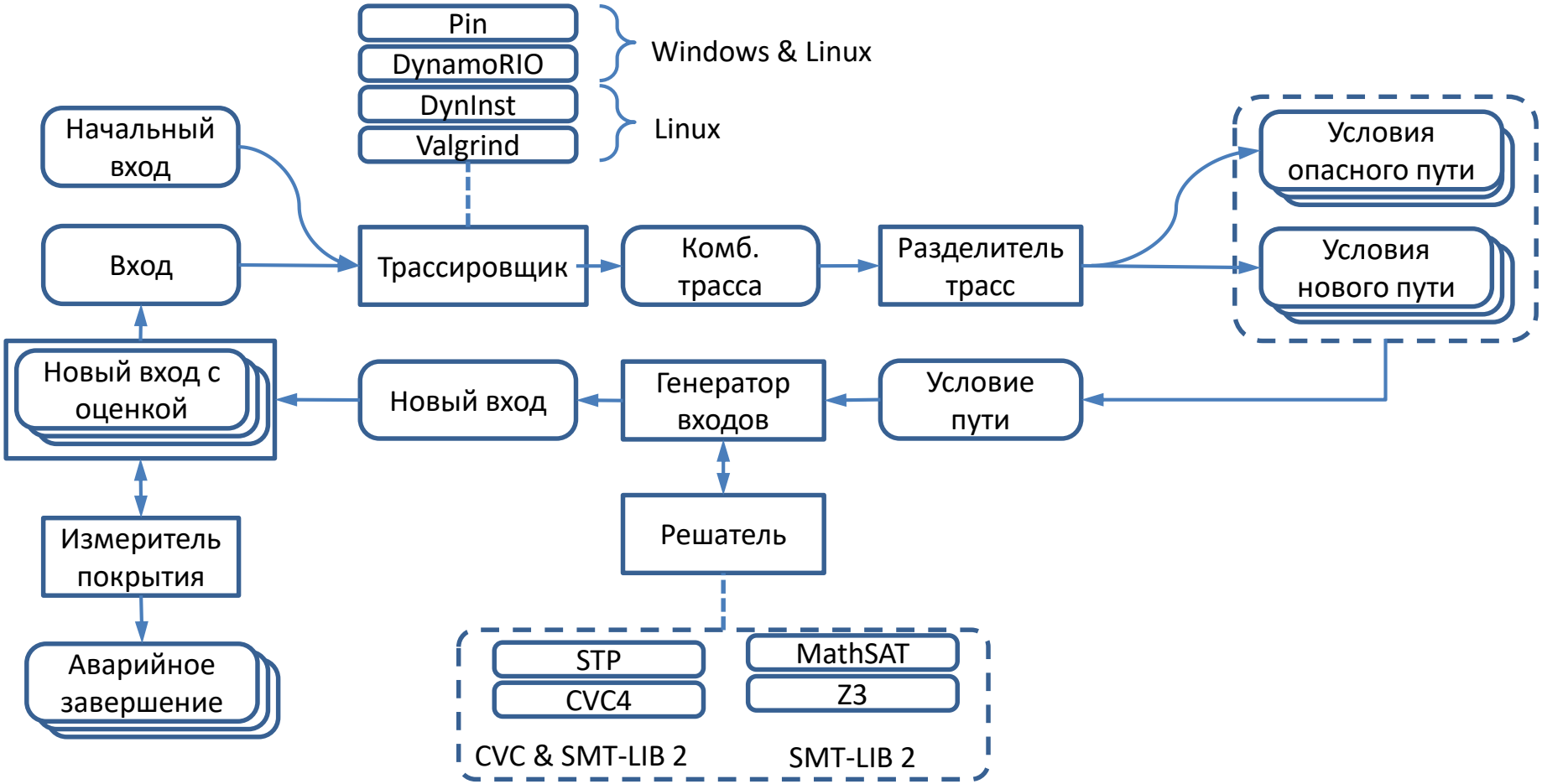
1. Фаззинг + Sanitizers
2. Динамическое символьное исполнение
3. Комбинированные инструменты

Доступные на рынке инструменты:

- Anxiety (ИСП РАН, Россия)
- ИСП Фаззер (ИСП РАН, Россия)
- 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



Динамический анализ помеченных данных

Технологии:

1. Контролируемое выполнение в виртуальной машине
2. Автоматизированная обратная инженерия бинарного кода
3. Анализ потоков данных и управления на уровне бинарного кода

Доступные в РФ инструменты:

- Среда анализа бинарного кода ТРАЛ (ИСП РАН, Россия)

Известные инструменты:

- MAYHEM (ForAllSecure, США)
- APAC (2013-2015), VET (2013), CGC (2016), CHESS (2019) (Проекты DARPA)

Технологии ИСП РАН жизненного цикла разработки безопасного 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

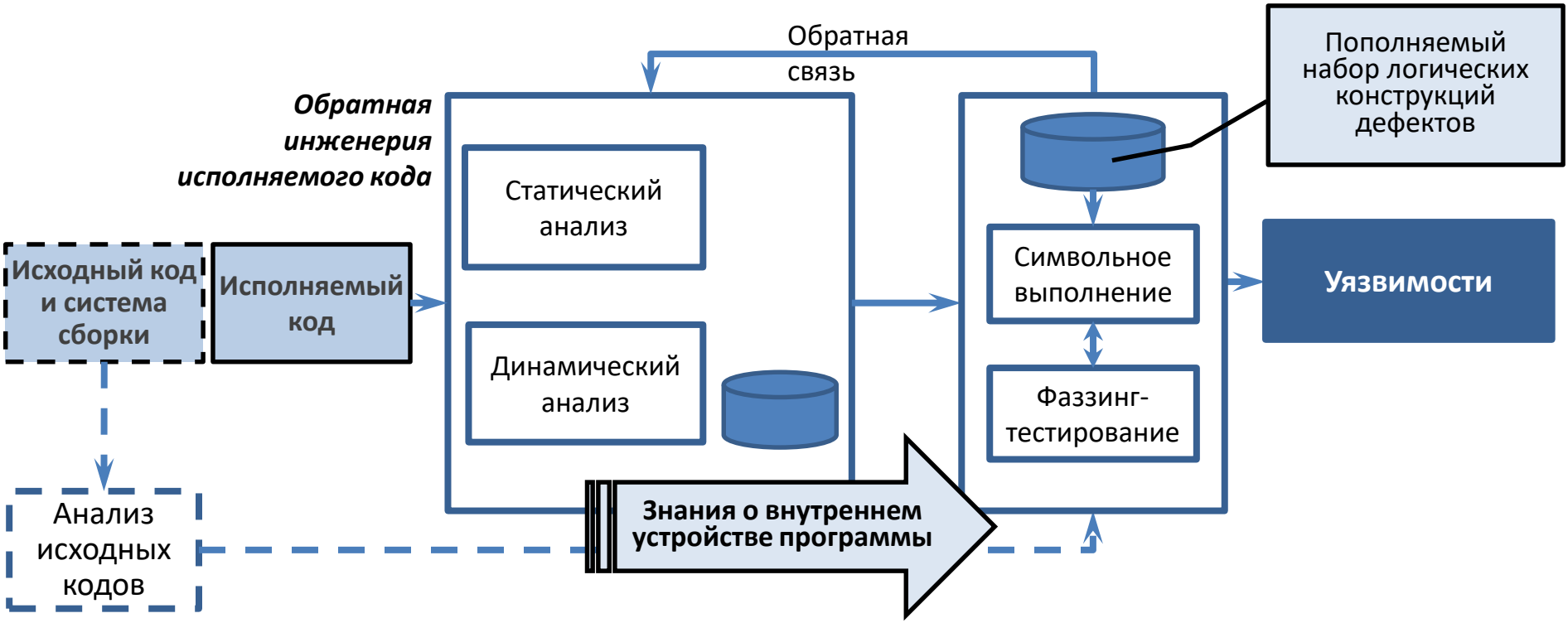
- **Доверенный компилятор**
  - Диверсификация кода
  - ИСП–Обфускатор: защита кода от анализа
  - Доработка системных защитных механизмов (ASLR …)

Реагиро вание

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

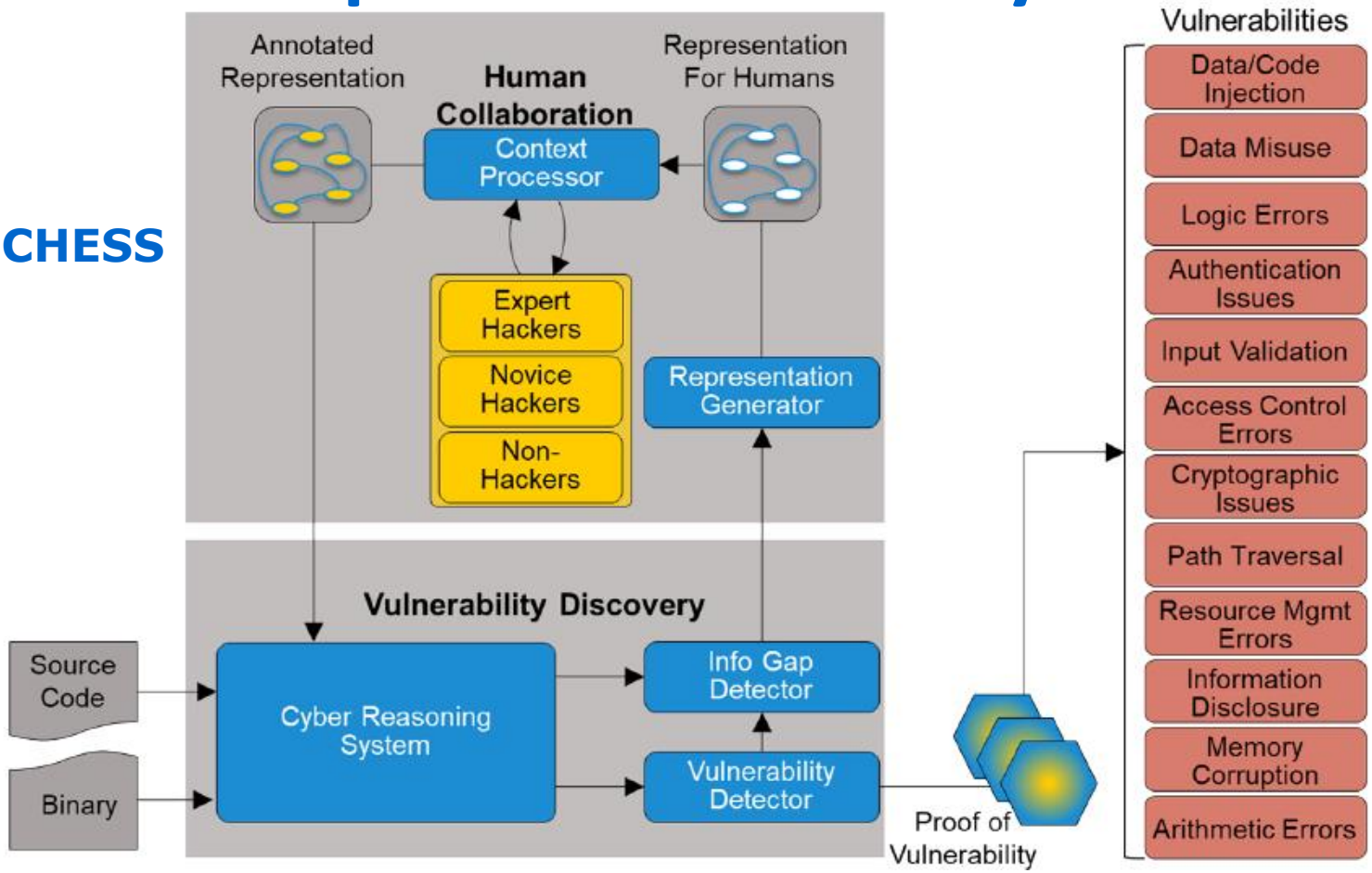
Необходимый этап: непрерывный поиск уязвимостей после сдачи в эксплуатацию

- Нет ограничения по времени
- Требует больших ресурсов



Зарубежный опыт: автоматизированный поиск уязвимостей

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

Развитие нормативной базы в РФ 2020-2021 г.

- ГОСТ Р «Защита информации. Разработка безопасного программного обеспечения. **Руководство по оценке безопасности разработки программного обеспечения**»
- ГОСТ Р «Защита информации. Разработка безопасного программного обеспечения. **Руководство по проведению статического анализа. Общие требования**»
- ГОСТ Р «Защита информации. Разработка безопасного программного обеспечения. **Руководство по проведению динамического анализа. Общие требования**»
- ГОСТ Р «Защита информации. Разработка безопасного программного обеспечения. **Доверенный компилятор языков Си/Си++. Общие требования**»
- ГОСТ Р «Защита информации. Разработка безопасного программного обеспечения. **Управление безопасностью программного обеспечения при использовании заимствованных и привлекаемых компонентов**»
- Пересмотр (**обновление**) **ГОСТ Р 56939-2016** «Защита информации. Разработка безопасного программного обеспечения. **Общие требования**»
- Пересмотр (**обновление**) **Методики выявления уязвимостей и недекларированных возможностей в программном обеспечении**
- **…**

ИСП РАН – ведущая научная школа в области «анализа программ и кибербезопасности»

- 2018 г.: решение Президиума РАН о новом научном направлении
- 2019 г.:
  - ✓ Курсы по повышению квалификации сотрудников Испытательных Лабораторий (ФСТЭК России)
  - ✓ создание подкомитета 4 «Разработка безопасного программного обеспечения» в ТК 362 «Защита информации»
  - ✓ создание ФСТЭК России и ИСП РАН Центра компетенции в области кибербезопасности (при поддержке Президиума РАН и Минобрнауки)
- 2020 г.: внедрение инструментов в Испытательных Лабораториях (Синклит, ЦБИ, Эшелон и др.) и у разработчиков (РусБИТех, ИВК, Код безопасности и др.)

Успешный опыт конструктивного взаимодействия:
регулятор&наука&бизнес

Спасибо за внимание

Фаззинг и динамический анализ
Crusher