Андрей Белеванцев
д.ф.-м.н., профессор кафедры СП
ВМК МГУ, ИСП РАН
abel@ispras.ru

Методы системного программирования в кибербезопасности

3 декабря 2025 года

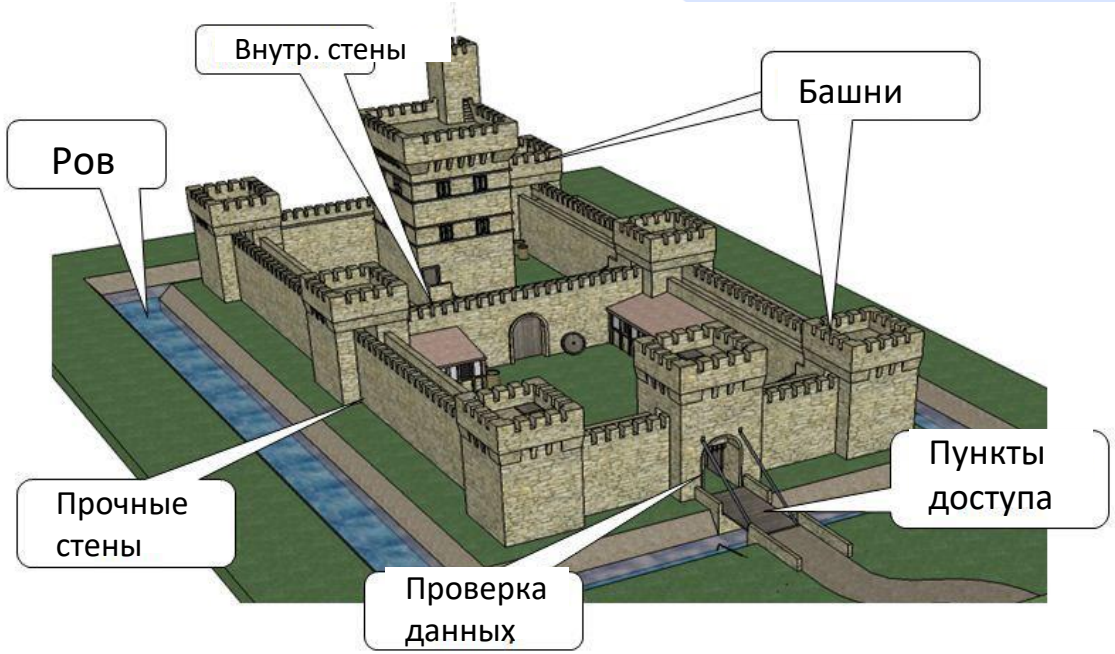
МГУ ВМК

ИСП РАН

Кибербезопасность – устоявшаяся научная дисциплина

- Не сводится к информационной безопасности
  - Классические методы необходимы, но недостаточны
  (защита по периметру, проверка доступа, криптография, антивирусы и др.)

- 2018: решение Президиума РАН о новом научном направлении
- 2021: новая специальность ВАК 1.2.4 «Кибербезопасность»
- Программа фундаментальных научных исследований
  - Модели и методы анализа и защиты программно-аппаратных систем
  - Инструментальные средства создания безопасных программ
- Регуляторика: ГОСТ Р 56939-2016 (2024)
  - Стандарты по статическому анализу и безопасной компиляции 2024 года
  - За рубежом: EU Cybersecurity Act, Common Criteria, документы NIST…



Причина развития кибербезопасности: сложность ПО

ИСП РАН

Сейчас программы:
Быстро растут
Усложняются
Не бывают изолированными

В то же время программам необходимы:
Эффективность
Продуктивность
Безопасность

GitHub
>100 млн разработчиков
(в 2016 – 5,8 млн)
>420 млн проектов

ОС, фреймворки
PyTorch 7 млн строк кода
TensorFlow 10 млн строк кода
Debian >2 млрд строк кода

Большие данные
2020 – 64 зеттабайт (1 зеттабайт = 1 млрд ТВ)
2022 – 97 зеттабайт
2025 – 180 зеттабайт

Эффективность и продуктивность – средства разработки

Эффективность и продуктивность обеспечивает стек системного ПО – средства разработки, облачные технологии, инфраструктуры искусственного интеллекта…

Компьютерные оптимизации и наборы инструментов

- **iOS**: Objective-C/LLVM – Swift/LLVM (2014)
- **Android**: Java/Dalvik (2010), Java/Android Runtime (ART) (2014), Jack (2016), Kotlin (2019)
- **Tizen**: C++, JavaScript/WebKit & V8 (2012), C# / Roslyn (2016)
- **Десктопные ОС**: Оптимизации link time / GCC, LLVM, Just-in-Time оптимизации / LLVM, WebKit, …
- **Многоядерные процессоры, GPU- и NPU-ускорители**: стандарты OpenCL, OpenMP / GCC, LLVM
- **Наш опыт**: 5 официальных ревьюеров GCC, OpenMP для GPU/CUDA в GCC, опережающая компиляция для JavaScript, оптимизация размера дистрибутива ОС Tizen (уменьшение на ~20%)…

Сотни миллионов строк кода – справиться в одиночку невозможно:

- Закрытые компиляторы Intel, IBM, Microsoft проиграли конкуренцию (используют LLVM/Clang)
- «Эльбрус»: собственный компилятор GCC, затраты на поддержку своей отдельной кодогенерации, трата ресурсов и отставание от основной версии
- **Наш опыт**: исследования по поддержке стандарта OpenCL для FPGA (2012 г.)
  Спустя 3-4 года это стало мейнстримом для основных производителей Xilinx / Altera

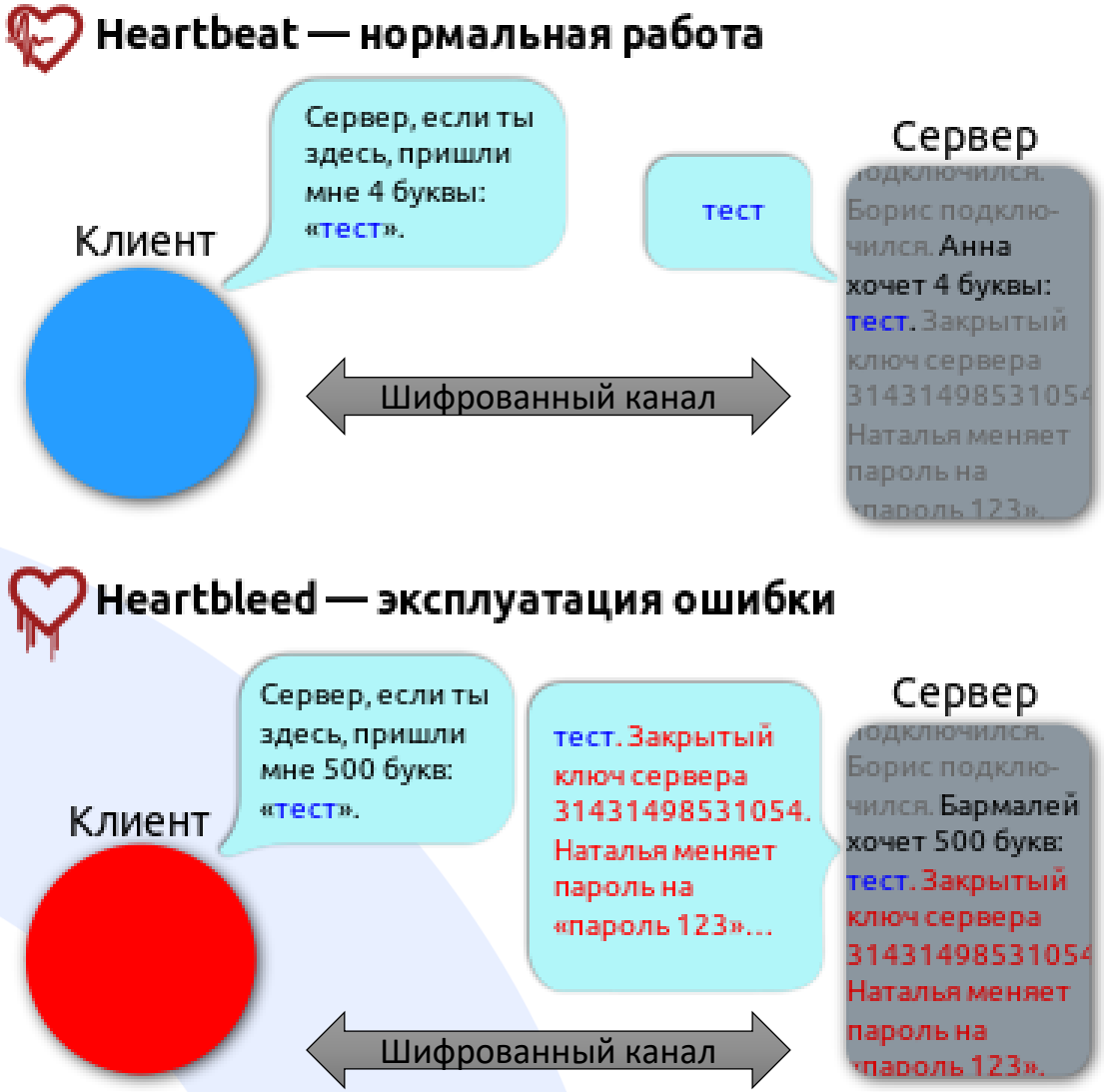
Безопасность – основная характеристика

ИСП РАН

Ключевой вызов: обеспечивать высокий уровень **безопасности** при сохранении на конкурентоспособном уровне **эффективности** и **продуктивности**

Безопасность связана с **уязвимостями**. Основная причина уязвимостей – **ошибки в ПО**.
Границы между ошибками, закладками, уязвимостями **размыты**.

Пример: уязвимость Heartbleed (библиотека OpenSSL)



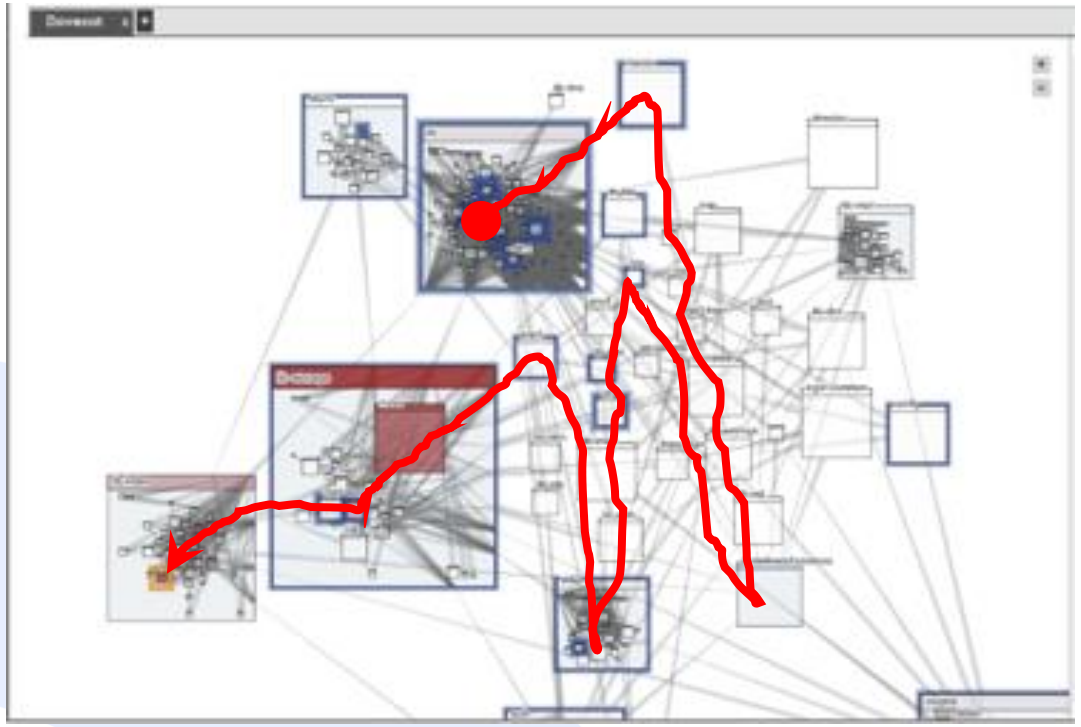
500000 сайтов заражено
$500 млн потерь

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

Версия OpenSSL с уязвимостью была выпущена в марте 2012 года и обнаружена только через два года

Пример: сложная ошибка в коде, приведшая 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 памяти.

```
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

Пример: небезопасная оптимизация в компиляторе

ИСП РАН

```
bool auth() {
  char buf[N];
  bool res;

  read_password(buf, N);
  res = check_password(buf);

  memset(buf, 0, N);
  return res;
}
```

Работа с паролем (чувствительными данными)

Компилятор удаляет обнуление буфера с паролем, т.к. с его точки зрения после обнуления буфер не используется. При этом пароль останется в памяти.

Кибербезопасность: направления научных исследований*

- Анализ известных и вновь выявляемых уязвимостей, их систематизация, разработка методов интеллектуального поиска новых классов уязвимостей
- Методы проектирования, моделирования, анализа, трансформации программ для выявления потенциальных уязвимостей в программных системах с учетом специфики фаз жизненного цикла
  - Статический анализ, динамический анализ и фаззинг-тестирование
  - Обратная инженерия бинарного кода, восстановление алгоритмов и моделей поведения
- Методы, алгоритмы и средства обеспечения устойчивого функционирования программно-аппаратных систем в условиях злонамеренного воздействия
  - Безопасная компиляция, доверенная среда выполнения, обфускация (запутывание) и диверсификация кода
- Моделирование, анализ и верификация криптографических протоколов
- Гомоморфное шифрование, безопасность хранилищ данных
- Методы анализа описаний цифровой аппаратуры на предмет наличия закладок и НДВ

*** Паспорт специальности ВАК 1.2.4 «Кибербезопасность», программа фундаментальных научных исследований**

Стек технологий обеспечения кибербезопасности

ИСП РАН

Инструменты

- Статический анализатор исходного кода программ
- Комплекс динамического анализа и фаззинг-тестирования
- Безопасный компилятор
- Инструмент определения поверхности атаки
- Инструмент обратной инженерии бинарного кода и поиска утечек конфиденциальных данных
- Инструмент отслеживания используемых библиотек и компонент

Методы

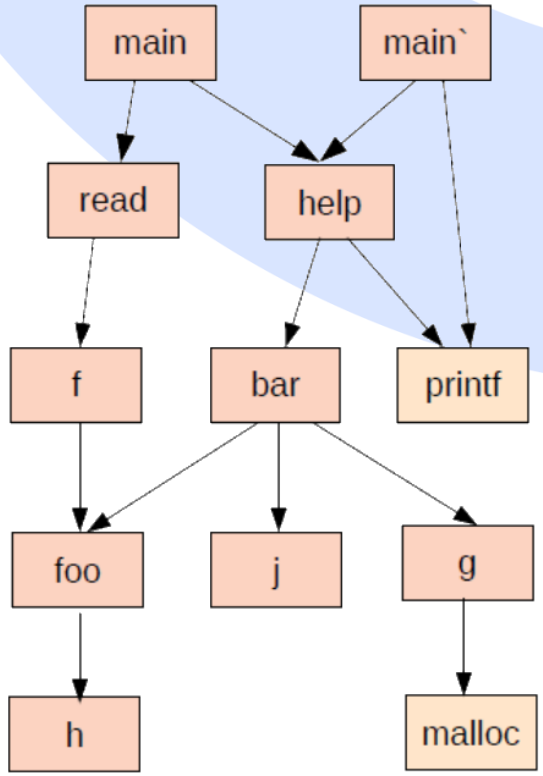
Анализ потоков управления и данных программы

- Абстрактная интерпретация
- Символьное выполнение
- Анализ чувствительных данных
- Моделирование поведения программы
- Эмуляция и бинарная трансляция
- Восстановление структуры программы из бинарного кода
- Формальная верификация
- Поиск «клонов» кода
- Анализ трасс выполнения

Многоуровневый статический анализ исходного кода (I)

Вызов: работа моделей и алгоритмов анализа в ограничениях десятков миллионов строк кода

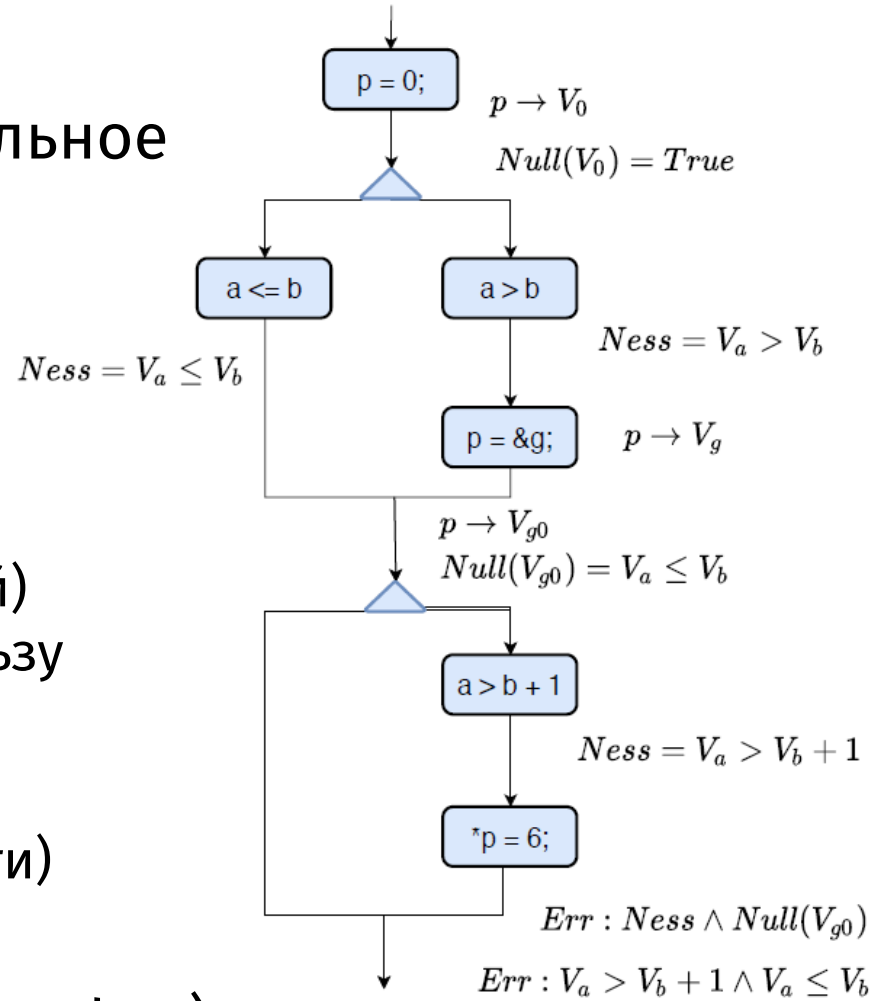
- Уровень 1: универсальное абстрактное синтаксическое дерево («шаблонные» ошибки)
- Уровень 2/3: методы глубокого анализа
  - Модель памяти и данных программы
    - Интервальная арифметика [a, b] + «выколотая точка» для размеров и смещений в ячейках памяти
    - Определение классов эквивалентности значений
  - Межпроцедурный анализ потока данных (резюме функции, обходы снизу-вверх/сверху-вниз)
    - Резюме параметризуется аргументами функции
    - Символьные вычисления над внешними (неизвестными) значениями



Многоуровневый статический анализ исходного кода (II)

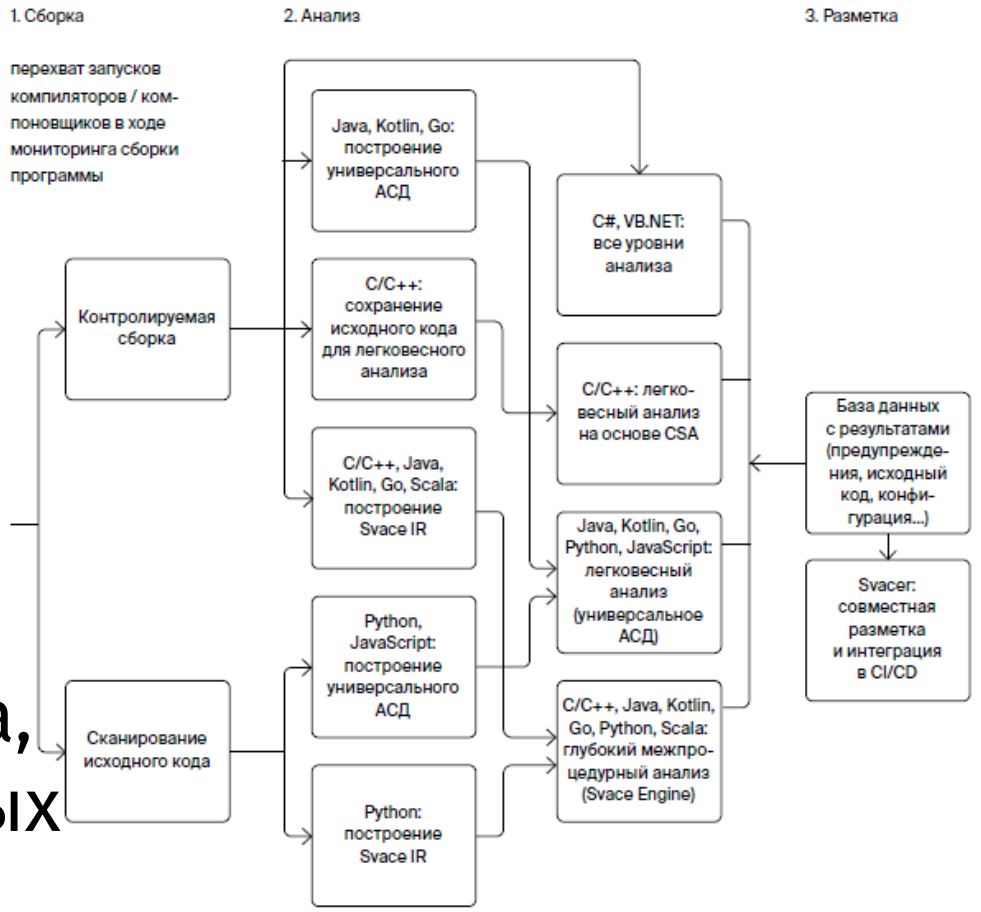
ИСП РАН

- **Вызов: работа моделей и алгоритмов анализа в ограничениях десятков миллионов строк кода**
  - Уровень 3: чувствительность к путям (символьное выполнение)
    - Отслеживание предикатов пути в ходе анализа, передача через резюме
    - SMT-солверы и логики: битовых векторов, вещественных чисел, массивов, кванторы…
    - Теория QF_UF (равенство символьных выражений) позволяет отказаться от нумерации значений в пользу обычного символьного выполнения
    - Возможное улучшение: сепарационная логика (расширение логики Хоара для динамической памяти)
  - Анализ указателей, девиртуализация, IFDS-задача потока данных (достижимость на графах)



Реализация методов статического анализа: Svace

- Разрабатывается с 2003 года
  - Добавлялись методы анализа потока данных, символьного выполнения, чувствительных данных
  - Добавлялась поддержка популярных языков (C, C++, Java, C#, Go, Kotlin, Python, Scala...)
- 10 языков, более 70 классов ошибок и 1000 детекторов
- Анализ 10-20 млн. строк за 5-7 часов
- Эксперименты с направлением анализа, распространением чувствительных данных
- Ведение истории анализов
- Разметка предупреждений

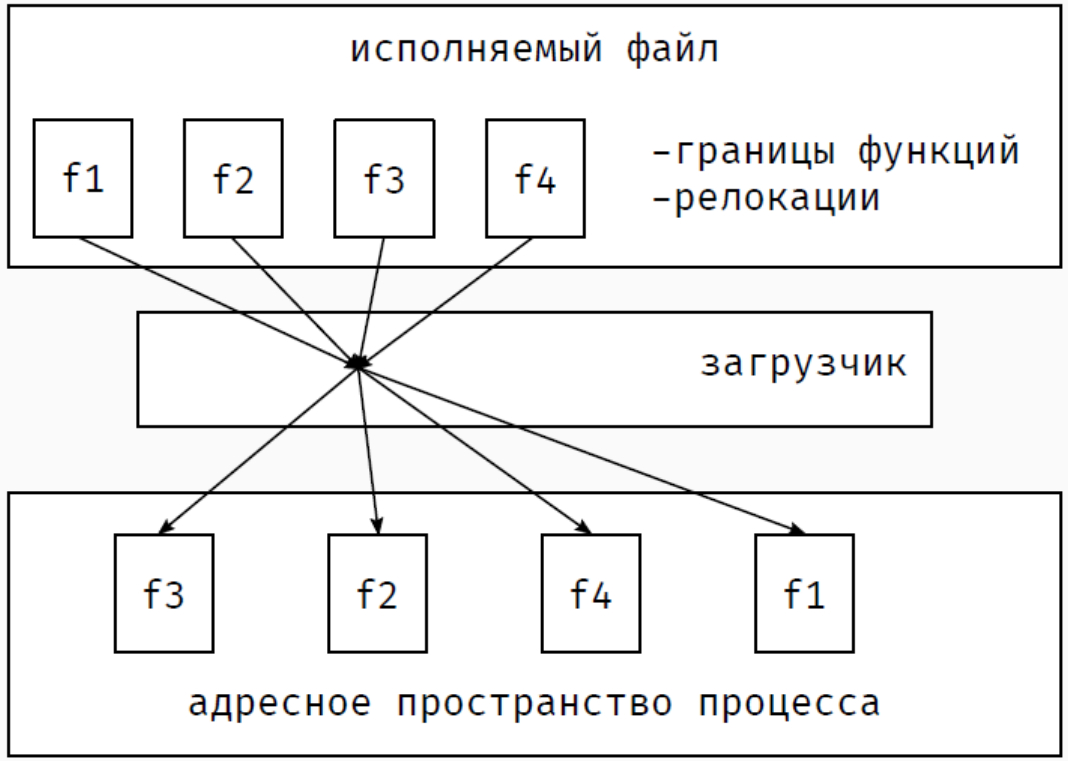


Безопасная компиляция

ИСП РАН

Вызов: ограничить оптимизации, не теряя производительности

- Надежные оптимизации
  - Не предполагать корректного поведения программы
- Выдача предупреждений о потенциально опасном коде
  - Консервативный анализ потока данных
- Динамическая защита кода
  - Снижение критичности уязвимости (санитайзеры)
  - Диверсификация на этапе компиляции и загрузки
- Реализация (C/C++): на основе компиляторов GCC / Clang



Динамический анализ и фаззинг

ИСП РАН

Вызов: модель поведения и данных программы для «умного» перебора входных данных в больших программах

- Динамический анализ: мониторинг выполнения программы + выбор очередных входных данных
  - Символьное выполнение и стратегии обхода путей выполнения
  - Увеличение покрытия, марковские цепи, автоматы Мили…
  - Поддержка структуры данных: протоколы, форматы…
  - Подсказки «интересных» мест программы от статического анализа
- Анализ «клонов» кода: поиск ошибок по всему дистрибутиву
- Выявление дефектов в программно-аппаратных системах, масштабируемость, …

Анализ бинарного кода: обратная инженерия

Вызов: модели бинарных программ, пригодные для разнообразных процессорных архитектур и программ разных классов

- Анализ набора полносистемных трасс выполнения
  - Автоматизация восстановления алгоритмов
  - Утечки чувствительных данных (в том числе через границы процессов)
  - Методы: дизассемблирование, восстановление процессов, анализ потоков данных, анализ чувствительных данных, динамическая двоичная трансляция…
- Восстановление алгоритмов из бинарного кода
  - Специализированное промежуточное представление, позволяющее единообразно проводить анализ бинарного кода различных процессорных архитектур
  - Обратная инженерия бинарного кода по набору трасс для ПО всех уровней

Машинное обучение в анализе программ

ИСП РАН

Вызов: мало данных для обучения, «строгая» область

- Помощник, но не заменитель компиляторных алгоритмов
- Классическое машинное обучение
  - Фильтрация ложных срабатываний (например, по метрикам программы)
  - Автоматическое исправление простых ошибок
  - Вывод правил для поиска ошибок из исправлений в коде
- Большие языковые модели
  - Помогают в разработке анализа (например, пополнение базы тестов, разработка моделей функций)
  - Поиск «алгоритмических» ошибок
  - Оценка истинности предупреждений (предоставляется контекст от анализатора) +3-14% точности к анализатору, +3-5% дополнительно с дообучением
  - Тестирование компиляторов: LLM как «мутатор»

Davide Italiano and Chris Cummins. Finding Missed Code Size Optimizations in Compilers using Large Language Models.

Безопасность искусственного интеллекта: методы и технологии

Эксплуатация моделей

- Методы защиты моделей от атак на этапе эксплуатации
- Методы обнаружения дрейфа данных
- Инструменты оценки устойчивости обученных моделей к атакам
- Инструменты маркирования сгенерированного контента

Обучение моделей

- Методы объяснения моделей
- Инструменты для повышения доверия к предобученным моделям
- Методы федеративного обучения
- Методы выявления предвзятости моделей

Датасеты

- Инструменты проверки наличия аномалий в наборах данных

Фреймворки

- Доверенные фреймворки и библиотеки машинного обучения

Центр доверенного искусственного интеллекта в ИСП РАН (с 2019 г.)

Задел: разработанные методы обеспечения кибербезопасности (статический анализ, динамический анализ, фаззинг…)

Выводы

ИСП РАН

- Кибербезопасность: широкий спектр направлений исследования
  - Статический и динамический анализ, безопасная компиляция, анализ чувствительных данных, обратная инженерия бинарного кода…
- Новое в методах анализа программ
  - Многоязыковые программы / языки запросов и пользовательские детекторы
  - Моделирование «внешнего» мира / новые виды логик (incorrectness logic)
  - Моделирование поведения программы в фаззинге
  - Применение машинного обучения
- Новые вызовы – искусственный интеллект
  - Применение классических методов анализа к фреймворкам ИИ
  - Защита от отравления данных и кражи данных, защита моделей

Необходимо непрерывное развитие фундаментальных методов в ограничениях больших программных систем

Научные школы, распределенные сообщества, образование

Спасибо!