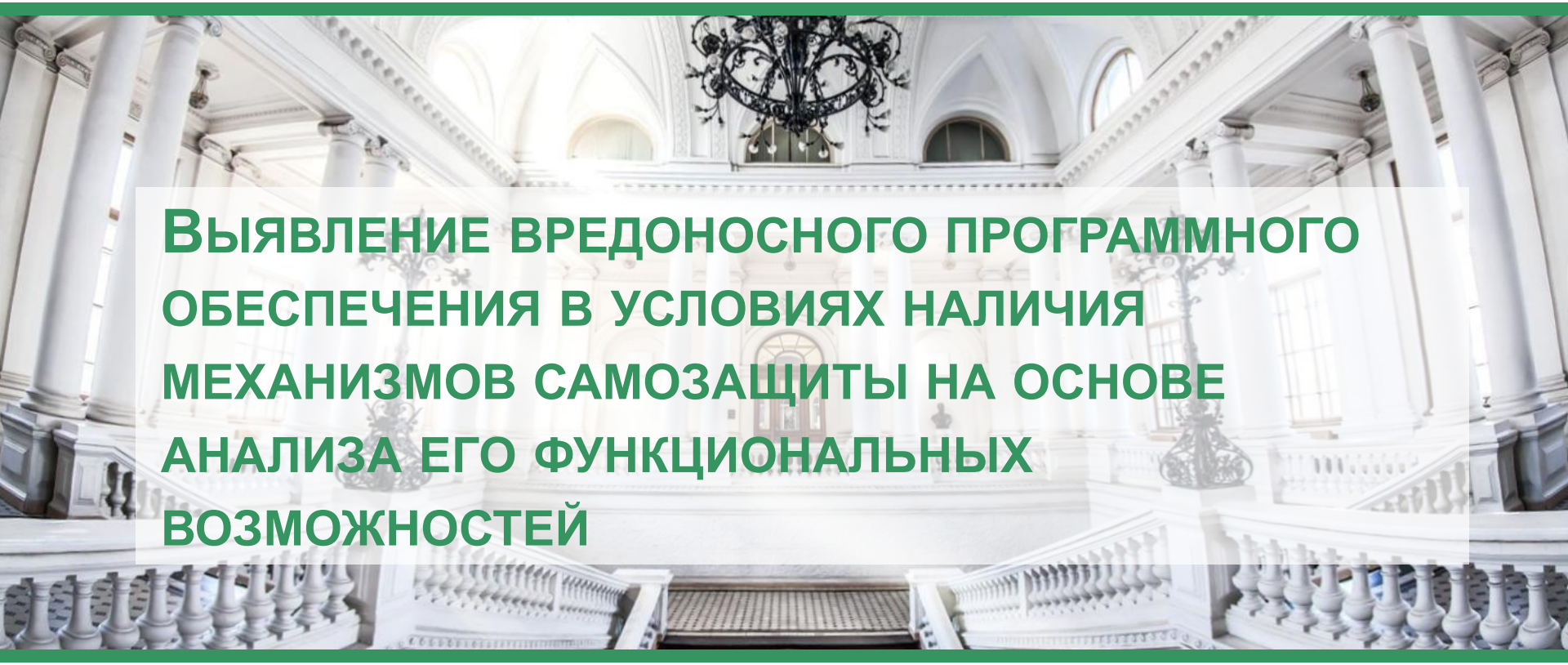
ПОЛИТЕХ
Санкт-Петербургский
политехнический университет
Петра Великого

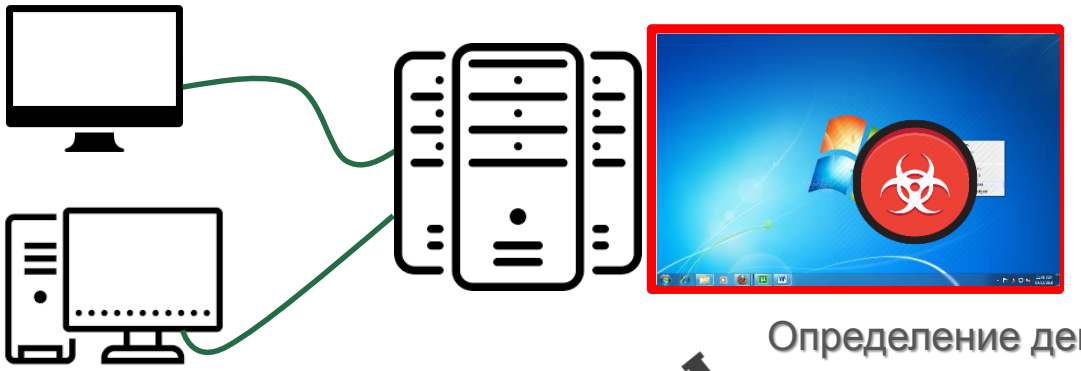


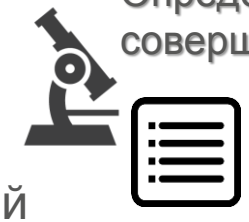
Выявление вредоносного программного обеспечения в условиях наличия механизмов самозащиты на основе анализа его функциональных возможностей

аспирант, ассистент кафедры ИБКС Жуковский Е.В.

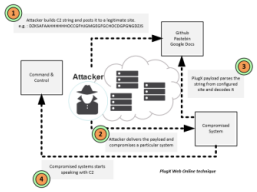
2019

Выявление и анализ вредоносного ПО на основе выполнения кода в изолированной среде





Определение действий, совершаемых вредоносным ПО



Обеспечивает:

- Высокую скорость работы
- Малое количество ложных срабатываний

Задачи, решаемые путем динамического анализа вредоносного ПО

Исследование путей распространения и инфраструктуры управления (C&C серверов)	Выявление скомпрометирован-ных систем и последствий заражения	Определение индикаторов компрометации (IoC,Indicators of Compromise)	Составление сигнатур и правил

Постановка цели и задач

Цель: Выявление возможности выполнения вредоносных операций программным обеспечением, обладающим механизмами самозащиты *(определить функциональные возможности ПО)*

Сложности:

1. Статический анализ: возможность упаковки / шифрования кода
2. Динамический анализ: вредоносные действия не будут выполнены при анализе в изолированной среде

Задачи:

1. Обеспечить выполнение потенциально опасных операций, в случае наличия таких функциональных возможностей в ПО *(максимизировать покрытие анализируемого кода).*
2. Выявить операции, представляющие угрозу нарушения информационной безопасности.

Механизмы самозащиты вредоносного ПО

- Затруднение анализа ВПО
  - Обфускация, использование промежуточного представления кода (виртуализация)
  - Упаковка / шифрование бинарного кода
  - Самомодификация кода
  - Приемы затрудняющие дизассемблирование

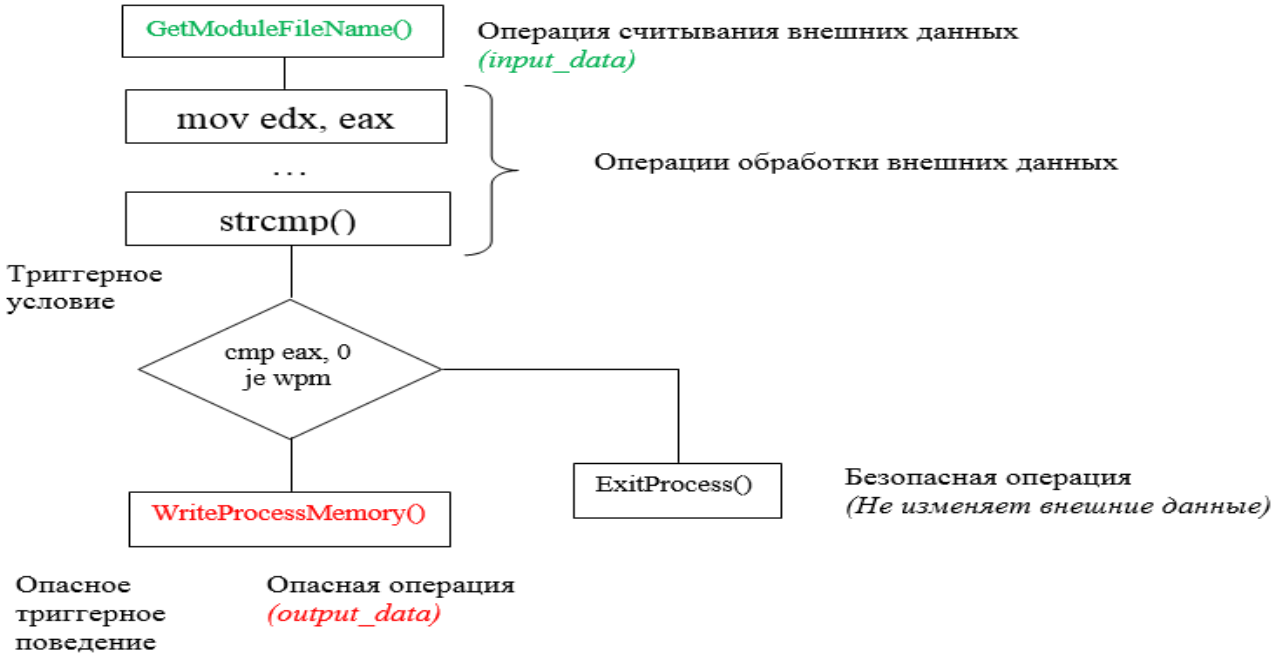
Затрудняют статический анализ

- Обнаружение факта исследования и изменение поведения ВПО *(триггерное поведение)*
  - Особенности процесса отладки и мониторинга
  - Признаки применения технологии виртуализации
  - Сигнатуры известных систем автоматизированного анализа
  - Отсутствие пользовательской активности
  - Наличие сетевого соединения, наличие определенных файлов

Ограничивают эффективность динамического анализа

Триггерное поведение

- ***Триггерное поведение*** – осуществление операций, изменяющих состояние системы при выполнении определенных условий
- ***Опасное триггерное поведение*** – триггерное поведение, связанное с выполнением операций, приводящих к возможности нарушения информационной безопасности системы
- ***Триггерное условие*** – условие выполнения целевого кода, изменяющего состояние системы

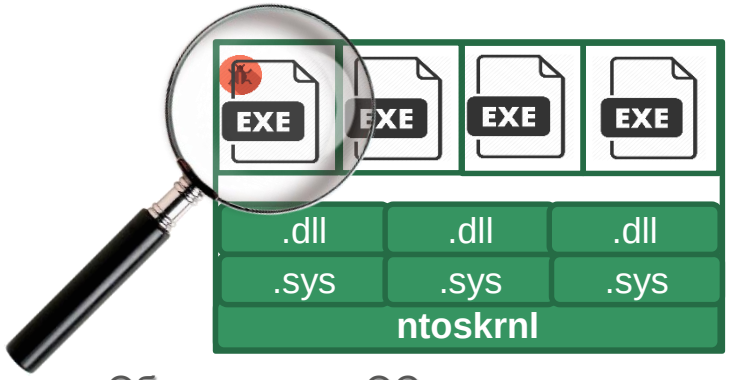


Отслеживание потоков передачи данных (операций ввода/вывода)

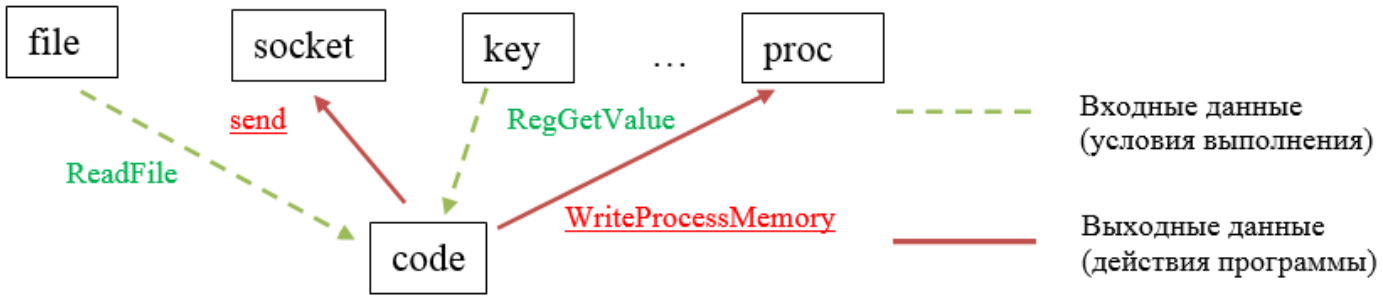
Разбиение всех WinAPI-функций на 2 пересекающихся класса:

$O_{in} = \{F_1, F_2, .. F_n\}$ – функции, осуществляющие считывание внешних данных

$O_{out} = \{F_1, F_2, .. F_k\}$ – функции, осуществляющие изменение внешних данных



Объекты ядра ОС



Триггерные условия: примеры

- Зависимость поведения программы от *внешних условий*
- *Внешние условия* – зависимость от состояния объектов ядра

Условие выполнения	Принцип проверки
Отнесение пользователя системы к целевой группе	Поиск критических файлов в системе
Соответствие требуемой локализации системы	Проверка используемых языков
Доступ к управляющему серверу	Возможность сетевого соединения с узлом
Отсутствие антивирусных средств	Присутствие процессов и служб, файлов и директорий
Отсутствие признаков анализа	Присутствие процессов и служб, файлов и директорий Признаки применения технологии виртуализации

Время	Системные события	Сетевые соединения	Файлы	Объекты реестра	Объекты синхронизации

Существующие подходы к выявлению триггерного поведения

№	Публикация	Описание принципов работы	Скорость анализа	Полнота анализа
1	Temporal Search: Detecting Hidden Malware Timebombs with Virtual Machines, 2006	Выявление зависимости поведения программы от таймеров с использованием виртуализации	Средняя	Низкая
2	Exploring Multiple Execution Paths for Malware Analysis, 2007	Символьное выполнение, Taint-анализ.	Средняя	Высокая
3	Automatically Identifying Trigger-based Behavior in Malware, 2007	Использование полносистемного эмулятора QEMU. Символьное выполнение, Taint-анализ.	Средняя	Высокая
4	Identifying Dormant Functionality in Malware Programs, 2010	Использование QEMU для определения типа ВПО, а затем нахождение не исполненного кода на основе анализа CFG	Высокая	Низкая
5	PyTrigger: A System to Trigger & Extract User-Activated Malware Behavior, 2013	Генерация действий пользователя из БД, виртуализация	Низкая	Низкая
6	GOLDEN EYE: Efficiently and Effectively Unveiling Malware's Targeted Environment, 2014	Динамический анализ с модификацией среды выполнения, виртуализация	Низкая	Высокая
7	BareCloud: Bare-metal Analysis-based Evasive Malware Detection, 2014	Сравнение выполнения программы в разных средах, виртуализация	Высокая	Низкая
8	MalGene: Automatic Extraction of Malware Analysis Evasion Signature, 2015	Сравнение выполнения программы в разных средах, виртуализация	Высокая	Низкая

Этапы анализа и выявления вредоносного ПО

Классификация операций

- **Класс 1** (C_1) – Безопасные операции. Операции, не представляющие опасности. Имеющие высокую встречаемость как во вредоносном, так и в легитимном ПО.
- **Класс 2** (C_2) – Подозрительные операции. Операции, которые часто используются во вредоносном ПО, но также могут присутствовать в легитимном. Механизмы самозащиты (анти-отладка, анти-ВМ). Операции передачи управления.
- **Класс 3** (C_3)– Опасные операции. Операции, которые могут приводить к нарушению ИБ. Чаще всего встречаются во вредоносном ПО.

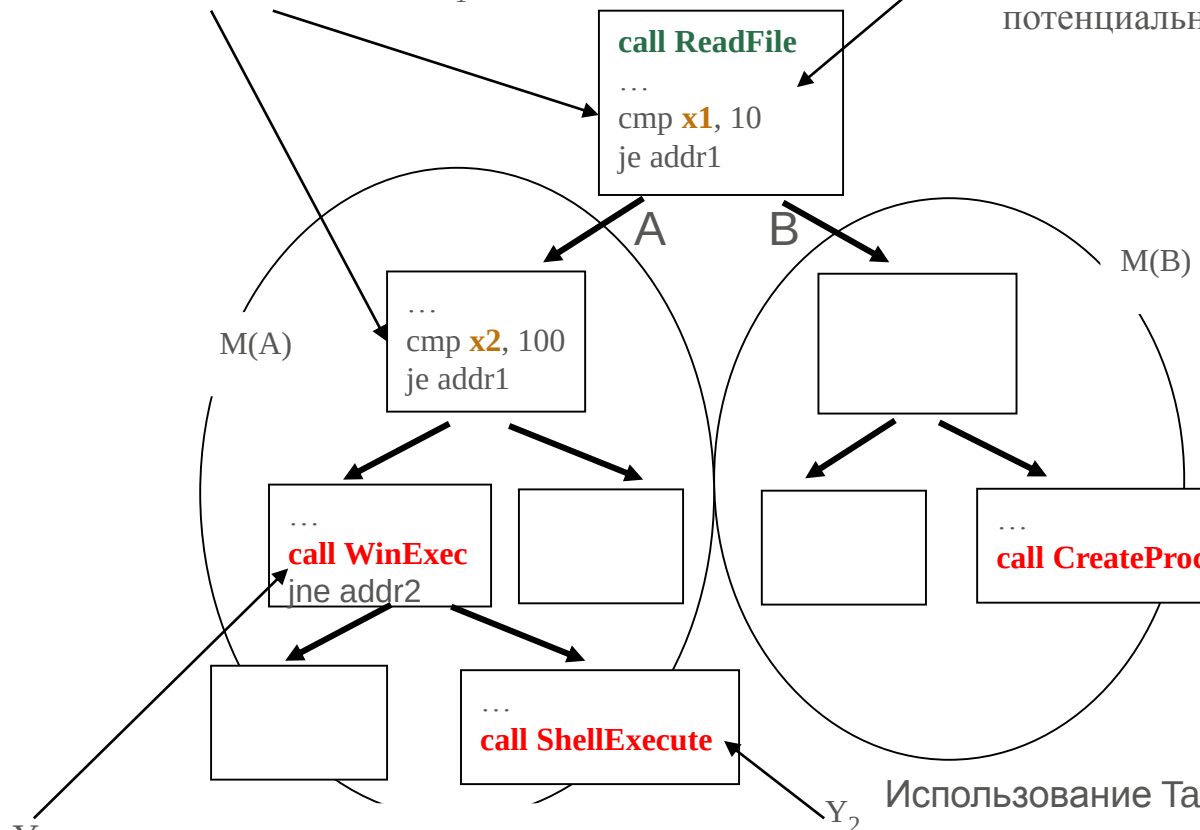
Функция определения класса:
$D(function) = \{C_1, C_2, C_3\}$

Безопасные операции	Подозрительные операции	Опасные операции
CreateFile	StartService	WriteProcessMemory
ReadFile	IsDebuggerPresent	NtDuplicateObject
RegCloseKey	SetUnhandledExceptionFilter	NtMapViewOfSection

Применение символьного выполнения для определения условий достижимости участков кода

$X_1, X_2, \ldots, X_n$ – переменные (в т.ч. регистры и память), от которых зависит достижимость Y_1

X – место считывания внешних данных, от которых зависит выполнение потенциально опасных операций



M–Оценка «опасности» пути:

$M^k(A) = \sum_R D(r) = 2 + 2 = 4$

$M^k(B) = 2$

R–количество достижимых функций,

D – класс «опасности» функции,

k – глубина анализа графа, k = 9

Y_1 – целевая точка выполнения программы, приводящая к осуществлению потенциально опасных операций

Использование Taint-анализа и построение символьного уравнения:

$H(Y_1) = x_1 < 10 \ \& \ x_2 > 100 \ \& \ \ldots$ - уравнение, подаваемое SMT-решателю для определения условий достижения Y_1

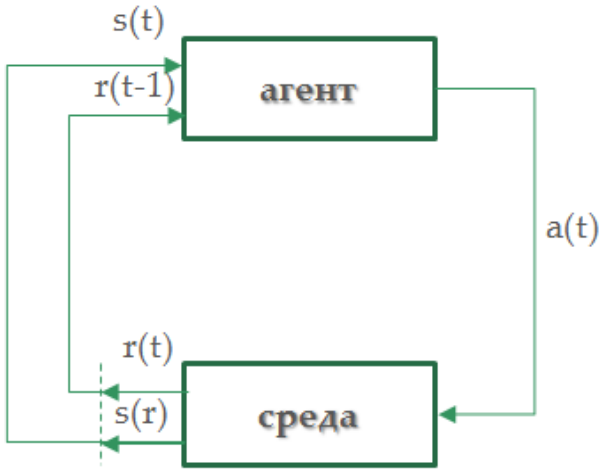
ПОЛИТЕХ

Этапы анализа путей выполнения программы

1. • Выявление функций и инструкций в программе, отвечающих за считывание внешних данных – бинарная инструментация (DBI)
2. • Оценка покрытия кода при анализе и определение потенциально опасных путей выполнения с триггерным поведением – поэтапный статический анализ + DBI
3. • Определение условий выполнения целевого кода и формирование требуемого входного потока данных – динамическое символьное выполнение
4. • Построение трасс выполнения программы, обеспечив максимальное покрытие исполняемого кода
5. • Анализ полученных трасс выполнения – машинное обучение

Применение обучения с подкреплением

Reinforcement learning



Действие **a(t)** – выбор дальнейшего пути выполнения программы (*следовать по текущему, либо решать символьное уравнение и менять путь*)

Вознаграждение $\mathbf{r(t) = r_1(t) – r_2(t)}$

Положительное вознаграждение $\mathbf{r_1(t)}$ – количество функций, вызванных по выбранному пути выполнения, с учетом их потенциальной опасности.

Отрицательное вознаграждение $\mathbf{r_2(t)}$ – количество выполненных базовых блоков кода, которые можно было пропустить при выполнении программы для достижения целевых опасных функций, а также время работы программы, в том числе факт ее зависания (зацикливания).

Состояние **s(t)** – характеризуется графом потока передачи управления и зависимости по данным и выполненными функциями

Среда - результат выполнения выбранного пути программы
Осуществляет выполнение программы с использованием DBI

Агент – подпрограмма, принимающее решение по какому из возможных путей продолжать выполнение

Цель агента максимизировать ожидаемую награду: Rt = rt+1 + rt+2 + rt+3 + … + rT.

Условием остановки анализа является достижение заданного ограничения времени на анализ программы, завершение анализа всех путей выполнения (достижение всех инструкций передачи управления внешним модулям).

ПОЛИТЕХ

Принятие решения агентом

Решение агента о выборе пути дальнейшего выполнения принимается на основе следующих параметров:

- текущий путь выполнения согласно состоянию памяти и регистров;
- тип проверки (тип объекта): реестр, файл, сетевое соединение, время и другие;
- условие проверки (WinAPI-функция, связанная с проверкой);
- степень опасности функции условия;
- достижимость невыполненного кода;
- отличается ли в других средах выполнения текущее состояние (с учетом отключения факторов внесения случайности);
- количество новых инструкций до следующего ветвления;
- количество достижимых вызовов новых функций при глубине анализа k (изменяемый параметр k);
- опасность достижимых функций.

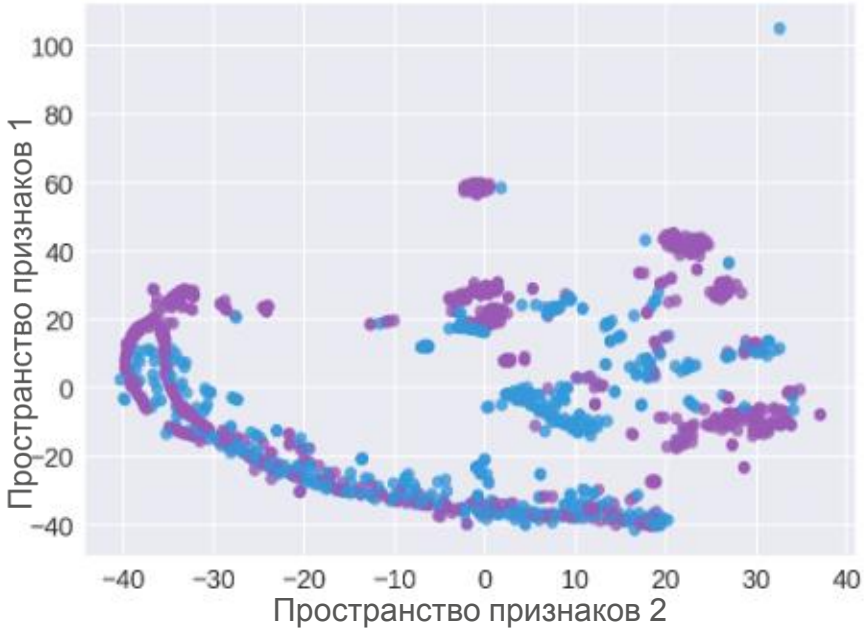
Компромисс между **исследованием** (решением символьного уравнения для достижения другого пути) и **эксплуатацией** (выполнением программы в текущих условиях)

Архитектура системы

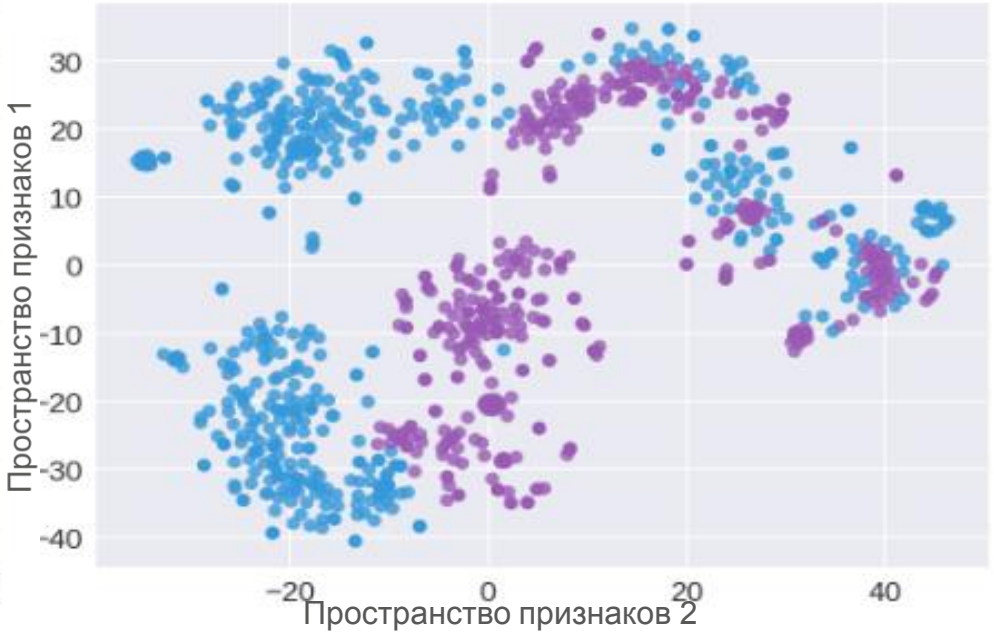
Анализ распределения значений признаков

Визуализация значений признаков с помощью алгоритма стохастического вложения соседей с t-распределением (t-SNE)

Частоты вызовов функций



Последовательности вызовов функций



- Эффективнее для классификации применение в качестве признаков - последовательностей вызовов функций

Полученные результаты классификации

Перекрестное тестирование с разбиением по 4 блока

	Точность	F1-мера	ROC-AUC
Логистическая регрессия	0.988	0.988	0.997
Градиентный бустинг	1	1	1
Адаптивный бустинг	0,984	0,983	0,983
Случайный лес (критерий Джини)	1	1	1
Случайный лес (энтропия)	1	1	1
Нейронная сеть	0.974	0.977	0.971
Рекуррентная нейронная сеть	0.988	0.987	0.850
Нейронная сеть с LSTM	0,978	0.981	0.975

Выборка исполняемых файлов: 500 (VirusTotal) / 700 (системные файлы ОС)
Требование: отсутствие зависимости от сторонних исполняемых файлов

ПОЛИТЕХ

Сложности и ограничения

Факторы, влияющие на эффективность анализа

Обфускация кода, графа потока управления	Нелинейная передача управления – (обработчики исключений и событий)	Криптография (контрольные суммы, хэш-функции)	Огромное количество ветвлений в программе

Сложность вычисления достижимости пути SMT-решателем

Не удалось покрыть весь код при тестировании – ставилось ограничение по максимальному времени решения уравнений

ПОЛИТЕХ

Достоинства подхода

Нахождение и обход ранее неизвестных механизмов самозащиты

Возможность определения условий для выполнения вредоносного кода

Возможность анализа многоэтапных целевых атак

ПОЛИТЕХ

Дальнейшие исследования

Поддержка анализа передачи управления в сторонние модули

Формирование выборки с наличием подтвержденного триггерного поведения

Многоклассовая классификация: определение класса ВПО

Решение проблемы формирования обучающей и тестовой выборок: строится на основе результата анализа антивирусных средств (VirusTotal)

ПОЛИТЕХ

Спасибо за внимание