МЕТОДИКА АДАПТИВНОГО ВЫЯВЛЕНИЯ УЯЗВИМОГО ПРОГРАММНОГО КОДА НА ОСНОВЕ МАШИННОГО ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

Исаев Н.Э.¹, Магонов Р.В.¹, Менисов А.Б.¹,

Научный руководитель – кандидат технических наук Менисов А.Б.¹,

1- Военно-космическая академия имени А.Ф. Можайского
e-mail: vka@mil.ru

Введение. Современный мир цифровых технологий находится в эпохе беспрецедентного роста кибератак. По данным ведущих компаний в сфере кибербезопасности, таких как «Лаборатория Касперского» [1] и «Positive Technologies» [2], за последние годы наблюдается значительное увеличение числа атак на слабозащищенные организации. Это говорит о том, что злоумышленники всё чаще используют незащищенные места в инфраструктуре компаний, которые не успевают реагировать на новые угрозы. Основным объектом этих атак, как показывают исследования, остаются уязвимости программного обеспечения. Ошибки в исходном коде или недостатки в архитектуре систем дают злоумышленникам возможность проникнуть внутрь инфраструктуры и использовать ее в своих интересах.

Основная часть. Использование машинного обучения (ML) в выявлении уязвимостей программного кода имеет преимущество над традиционными методами анализа и оно заключается в его способности выявлять сложные закономерности в данных, которые трудно заметить при помощи как раз традиционных методов анализа. Модели ML могут обучаться на больших наборах данных, содержащих примеры как уязвимого, так и безопасного кода, и затем применять свои знания для автоматической классификации новых образцов.

Одной из основных проблем существующих моделей машинного обучения является их снижение производительности при изменении структуры данных или появлении новых типов уязвимостей. Это происходит по нескольким причинам: зависимость от обучающих данных, спецификация сферы деятельности, необходимость постоянной адаптации под потребителя.

Адаптивность моделей для анализа уязвимостей в коде можно существенно улучшить, используя методы обучения с подкреплением (RL). Этот подход позволяет моделям динамически адаптироваться к изменениям в данных, эффективно выявляя новые, ранее неизвестные уязвимости. Говоря про преимущества RL, отмечу:

1. Обучение на основе обратной связи: RL отличается от других подходов тем, что модели обучаются через взаимодействие с окружающей средой. Это позволяет моделям "учиться" на ошибках и улучшать свои прогнозы с течением времени.

2. Адаптивность к изменениям в данных: программный код постоянно меняется, а с ним и характер уязвимостей. RL позволяет моделям быть более устойчивыми к таким изменениям. В отличие от методов с учителем, где модель статически обучена на фиксированном наборе данных, RL динамически адаптируется к новым образцам.

В рамках обучения с подкреплением применительно к анализу уязвимостей кода была разработана архитектура, состоящую из двух основных компонентов: агента и критика.

Модель критика играет ключевую роль в процессе обучения. Её задача заключается в оценке качества решений, принимаемых агентом, и предоставлении обратной связи. Основные функции работы модели критика: оценка решений агента, выработка стратегии обучения, улучшение адаптивности, снижение уровня ложных срабатываний:

Выводы. Обеспечение безопасности программного обеспечения становится все более критичным в условиях стремительного роста кибератак. Количество уязвимостей,

используемых злоумышленниками, увеличивается, что подчеркивает важность их своевременного выявления.

Методы машинного обучения показали свою эффективность в автоматизации процессов выявления уязвимостей, сокращая время анализа и улучшая его точность.

Обучение с подкреплением обеспечивает значительные преимущества благодаря своей адаптивности: применение RL особенно актуально в условиях, когда атаки постоянно эволюционируют, а новые уязвимости появляются быстрее, чем их можно обнаружить традиционными методами.

Для дальнейшего повышения эффективности необходимо использование промежуточных представлений кода (Intermediate Representation, IR): Low Level Virtual Machine (LLVM) IR, абстрагируют особенности различных языков программирования, а также обладают структурой Static Single Assignment (SSA), которая предоставляет явные зависимости между переменными и операциями, упрощая анализ. Это позволяет обнаруживать уязвимости на уровне низкоуровневой логики программы, что увеличивает переносимость решений на разные платформы и языки.

Список использованных источников:

1. Отчет «Лаборатории Касперского» [Электронный ресурс]. URL: <https://www.kaspersky.ru/about/press-releases/laboratoriya-kasperskogo-v-pervom-polugodii-2024-goda-zafiksirovan-kratnyj-rost-atak-na-sfery-telekoma-i-stroitelstva> (дата обращения 19.01.2025).

2. Отчет «Positive Technologies» [Электронный ресурс]. URL: <https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2024-q1/> (дата обращения 19.01.2025).