СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ РЕШЕНИЯ ЗАДАЧИ ОБНАРУЖЕНИЯ ПРИЗНАКОВ СЕТЕВОЙ РАЗВЕДКИ

Н.П. Шараев, С.Н. Петров

В настоящее время около 15 % крупнейших организаций детектируют таргетированные или APT атаки на сетевую инфраструктуру. Указанные атаки крайне опасны для организаций, что связано с созданием киберпреступниками вредоносного программного обеспечения с учетом специфики работы информационных систем и сетевой инфраструктуры организаций. Исследование проводимых APT атак показал, что в общем случае таргетированные атаки состоят из четырех этапов: подготовка, проникновение, распространение (закрепление) и достижение цели. На каждом из представленных этапов дополнительно проводится процедура сокрытия (очистки), что позволяет злоумышленникам оставаться в сетевой инфраструктуре незамеченными. Обнаружить подобный тип атак на последней стадии крайне сложно, а в отдельных случаях невозможно. По данной причине целесообразно провести обнаружение и анализ таргетированной атаки на этапе подготовки. На указанном этапе злоумышленники проводят процедуру сетевой разведки инфраструктуры организации, которую можно выявить с помощью методов машинного обучения.

В качестве перспективных и быстродейственных алгоритмов машинного обучения, позволяющих провести обнаружения признаков сетевой разведки, выбраны: логистическая регрессия, квадратичный дискриминантный анализ, метод опорных векторов, метод ближайших соседей (K-ближайших соседей), «наивный» байесовский классификатор, дерево принятия решений и многослойный персептрон (нейронная сеть прямого распределения). Для всех представленных методов проведено обучение и тестирование с использованием различных гиперпараметров. Обучение алгоритмов проводилось на основе сгенерированного датасета, состоящего из 1000 уникальных событий (250 событий сетевой разведки и 750 легитимных событий). Наилучшие результаты эффективности и быстродействия показал метод дерева принятия решений с параметрами criterion = «gini» и splitter = «random». Дополнительно проведено улучшение отдельных алгоритмов с помощью процедур бэггинга и бустинга, а также представление алгоритма с наилучшими параметрами в виде программного кода, что позволило увеличить скорость работы приблизительно в 2 раза.