0.1. *Лебедев Р.К., Корякин И.А.* Метод защиты программного кода при помощи расширений архитектуры x86

Для обратной разработки программного обеспечения применяются такие инструменты как дизассемблеры, декомпиляторы и инструменты символьного исполнения. Последние два типа инструментов значительно ускоряют этот процесс, что делает актуальной задачей разработку методов противодействия им с целью защиты программного обеспечения от копирования и других нежелательных последствий обратной разработки.

Методы противодействия декомпиляторам и инструментам символьного исполнения, как правило, совершенно различны. Так, для противодействия декомпиляторам применяются методы обфускации, делающие вывод декомпилятора неудобным для прочтения, например, модификация графа потока управления [1]. Однако, эти методы редко эффективны против инструментов символьного исполнения программ, иногда эти инструменты даже используются для противодействия таким методам [2]. Против них, в свою очередь, используются специальные подходы, не слишком эффективные против декомпиляторов, например, основанные на проблеме экспоненциального взрыва и односторонних функциях.

В данной работе предложен метод защиты, эксплуатирующий особенности общего этапа работы этих инструментов — преобразования машинных инструкций процессора в понятный инструментам вид. Архитектура x86, распространенная в персональных компьютерах, содержит более тысячи различных инструкций [3], однако компиляторами обычно используется лишь малая часть этого набора. Соответственно, поддержка более редких инструкций могла быть не реализована в инструментах обратной разработки без заметной потери совместимости.

Предложенный метод основан на введении в программу инструкций из расширения системы команд x86 AES-NI, используемого для ускорения шифрования AES и нечасто встречаемого в обычных программах. Одной из его инструкций является AESENC, реализующая один раунд шифрования AES и принимающая в качестве аргументов данные и раундовый ключ. При использовании без ключа эта инструкция может использоваться для сокрытия любых численных констант в программе. Во время компиляции каждая константа x заменяется выражением времени исполнения x_{obf} следующего вида:

$$x_{\text{obf}} = \text{AESENC}(x', 0)$$

Значение x' здесь рассчитывается во время компиляции, AESDEC — операция, обратная AESENC:

$$x' = \text{AESDEC}(x, 0)$$

Метод был реализован при помощи модификации промежуточного представления LLVM и показал свою эффективность против декомпиляторов Ghidra и IDA, а также инструмента символьного исполнения angr, вызвав их полную или частичную неработоспособность. Это подтверждает гипотезу о неполной поддержке инструкций инструментами обратной разработки, поэтому при необходимости описанный подход может быть использован и с другими редкими инструкциями архитектуры x86.

Научный руководитель — д.т.н. Павский К. В.

Список литературы

[1] Junod P. et al. Obfuscator-LLVM – Software Protection for the Masses // Proc. 2015 IEEE/ACM 1st International Workshop on Software Protection. IEEE, 2015. P. 3–9.

[2] Kan Z. et al. Deobfuscating Android Native Binary Code // 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). IEEE, 2019. P. 322–323.

[3] Intel XED. [Электронный ресурс]. URL: https://intelxed.github.io/ (дата обращения 01.09.2021).