0.1. *Лебедев Р.К.* Универсальный метод защиты программ от символьного исполнения, основанный на гипотезе Коллатца

В настоящий момент одним из традиционных методов обратной разработки программ является символьное исполнение. Этот подход позволяет представлять регистры процессора и содержимое оперативной памяти как переменные и составлять уравнения для них из встреченных в процессе исполнения программы условий и циклов. Уравнения затем решаются при помощи SMT-решателей. Таким образом, без полного перебора обеспечивается достижимость большинства ветвей программы, включая, например, проверку лицензионных ключей или прочих мест, достижение которых исследователями является нежелательным с точки зрения автора программы.

Однако подход символьного исполнения имеет некоторые трудноразрешимые проблемы, которые можно использовать для защиты чувствительных участков кода. Одной из таких проблем является «path explosion» (или «экспоненциальный взрыв») — экспоненциальный рост числа путей исполнения с каждым встреченным условием. Классической иллюстрацией этого явления является гипотеза Коллатца.

$$a_0 = n, n \in \mathbb{N}$$

$$a_{i+1} = \begin{cases} \frac{a_i}{2} & \text{если } a_i \equiv 0 \pmod 2 \\ 3a_i + 1 & \text{если } a_i \equiv 1 \pmod 2 \end{cases}$$

Утверждается, что для любого начального n существует такое i, что $a_i = 1$. Как можно заметить, благодаря наличию условия на каждом этапе подсчета a_i, код проверки данной гипотезы может спровоцировать экспоненциальный взрыв при анализе инструментами символьного исполнения.

Существуют работы, использующие данную гипотезу для целей защиты от символьного анализа, однако они ограничены обфускацией условий и требуют внесения нетривиальных изменений в код программы, что делает их применимыми в ограниченном числе сценариев [1, 2].

В рамках данной работы предложен универсальный метод защиты, который позволяет ограничиться добавлением в программу полностью независимого кода, выступающего в роли барьера для символьного исполнения. Это позволяет добиться следующих свойств:

- Возможность применения защиты вручную, без использования обфусцирующих компиляторов;
- Возможность защищать любой код, в том числе и не содержащий ни единого условия.

Разработанный метод показал эффективность против инструмента символьного исполнения angr [3], а влияние его на время исполнения программы оказалось незначительным.

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

Список литературы

[1] Wang Z. et al. Linear obfuscation to combat symbolic execution // Proc. European Symposium on Research in Computer Security. Springer, Berlin, Heidelberg, 2011. P. 210–226.

[2] Qin S. et al. A Method of JavaScript path obfuscation based on Collatz conjecture // Proc. 12th Web Information System and Application Conference (WISA). IEEE, 2015. P. 330–333.

[3] Shoshitaishvili Y. et al. SOK: (State of) The Art of War: Offensive Techniques in Binary Analysis // Proc. IEEE Symposium on Security and Privacy (SP). IEEE, 2016. P. 138–157.