вектора. На этапе обучения на вход данная нейронная сеть принимает вектор признаков и число — номер класса, к которому относится вектор.

На этапе классификации LSTM-сеть принимает вектор признаков нового запроса, а на выходе выдаёт два числа — номер класса, на который наиболее похож запрос, и число от 0 до 1 как вероятность того, что данный запрос принадлежит к этому классу.

В результате работы метод был реализован и подготовлены тестовые данные. В дальнейшем планируется провести серию экспериментов для оценки точности работы метода.

ЛИТЕРАТУРА

1. OWASP Top 10 Application Security Risks — 2017. `https://www.owasp.org/index.php/Top_10_2017-Top_10`
2. *Halfond W. G. J., Viegas J., and Orso A.* A classification of SQL injection attacks and countermeasures // Proc. ISSSE 2006, Washington, USA, Mar. 2006. `https://pdfs.semanticscholar.org/81a5/02b52485e52713ccab6d260f15871c2acdcb.pdf`
3. *Rawat R. and Shrivastav S. K.* SQL injection attack detection using SVM // Intern. J. Comput. Appl. 2012. V. 42. No. 13. P. 1–4.
4. *Buehrer G. T., Weide B. W., and Sivilotti P. A. G.* Using parse tree validation to prevent SQL injection attacks // Proc. SEM'05. N. Y.: ACM, 2005. P. 106–113.
5. *Cheng Y.* Mean shift, mode seeking, and clustering // IEEE Trans. Pattern Analysis Machine Intelligence. 1995. V. 17. Iss. 8. P. 790–799.
6. *Schmidhuber J. and Hochreiter S.* Long short-term memory // Neural Computation. 1997. No. 9. P. 1735–1780.

УДК 004.94 DOI 10.17223/2226308X/10/47

МЕТОД ИДЕНТИФИКАЦИИ СОГЛАШЕНИЙ О ВЫЗОВЕ ФУНКЦИЙ В БИНАРНЫХ ПРИЛОЖЕНИЯХ

М. А. Станчин, Н. В. Сороковиков

Предлагается метод определения соглашений о вызове функций в бинарных приложениях.

Ключевые слова: *статический анализ, бинарные приложения, соглашение о вызове.*

При статическом анализе бинарных приложений возникает задача определения соглашений о вызовах функций. *Соглашением о вызове* (calling convention) называется способ передачи параметров (аргументов) подпрограммам [1]. В известных средствах анализа бинарных приложений (например, Radare2, Angr, IDA Pro) эта задача решается с использованием встроенных инструментов, что, как правило, не позволяет использовать их в собственных разрабатываемых средствах анализа. В данной работе предлагается собственный метод анализа соглашений о вызове функций.

Для определения соглашения о вызове необходимо определить следующие параметры:

1) наличие регистров для передачи параметров в функции;
2) используемые регистры для передачи параметров в функции;
3) используемые области памяти;
4) мнемоника инструкции возврата.

Например, соглашение о вызове cdecl использует следующие параметры: для передачи аргументов в функции используются области памяти с отрицательным смещением вместо регистров, а мнемоника инструкции возврата — retn.

Метод определения соглашения о вызове функции

Пусть имеется набор физических и виртуальных адресов функций, соглашения о вызовах которых нужно определить.

1) Строится граф потока управления исследуемой функции для определения её конца и возможных условий выхода из нее.
2) Строится таблица использования неинициализированных регистров внутри функции: в случае нахождения такого регистра в инструкции в таблицу заносится метка об его использовании.
3) Проверяется наличие пролога функции и выясняется адресация относительно отрицательных смещений в стеке.
4) Проверяется наличие аргументов у инструкции возврата (мнемоника «ret/retn»).
5) По завершении всех проверок на основе полученных данных выдаётся определённый тип соглашения о вызове исследуемой функции, а в случае неспособности определить тип соглашения достоверно (например, если присутствуют несколько признаков различных соглашений о вызове) применяется мажоритарный принцип, основанный на статистике использования того или иного соглашения о вызове.
6) В случае отсутствия каких-либо признаков известных соглашений о вызове считается, что тип соглашения функции равен «user call».

На данный момент определяются следующие соглашения о вызове: cdecl_x86, cdecl_x86_64, fastcall_x86, fastcall_x86_64, stdcall_x86, stdcall_x86_64. Программная реализация метода доступна в сети Github [2].

ЛИТЕРАТУРА

1. *Agner F.* Calling Conventions for Different C++ Compilers and Operating Systems. 2017. `http://www.agner.org/optimize/calling_conventions.pdf`
2. CCPTool. `https://github.com/ACIDYWE/CCPTool`