JavaScript Code Deobfuscation using Neural Program Learning Models

Amin Fallahi<afallahi@syr.edu>

Neural Program Learning Course Project Proposal

Track: Novel Application

April 9, 2019

1 Introduction

One of the trending methods for improving software security and hiding program codes from users is JavaScript code obfuscation. JavaScript codes are meant to run on the browsers and are widely used to create dynamic content in most web-pages. Many content providers try to hide the code from the user to improve their security and obscure the real function of code. Using a simple obfuscation technique, the providers make their program secure against attackers and are able to inject malicious code that can help advertisement, tracking the user, collecting user info, cross-site scripting attack, etc. Obfuscation targets changing the code in a way that makes it hard for the human to understand what the code is actually doing by methods like changing variable/function names, altering execution order, loop unrolling, character encoding, etc.
In contrast, code deobfuscation methods try to gain as much information from obfuscated codes and extract the program from the unreadble code.
In this project, we use neural program learning methods to recover identifier (variables and functions) names from obfuscated JavaScript code. We study current JavaScript obfuscation and deobfuscation methods and how much identifier names the current approaches to deobfuscation can predict. Then, we implement a program learning model and train it using available data sets. Finally, we evaluate our work and try to optimize the model to achieve better performance than previous research.

2 Related Work

Research on code deobfuscation has been a trending topic during the recent years. Jiang et al. have proposed a method using Convolutional Neural Networks for deobfuscation detection [1]. Raychev et al. have developed a statistical model for recovering program properties (variable and function names) from obfuscated JavaScript code [2] and have implemented a currently online website named JSNice [3] which can predict the correct names for 63% of identifiers from obfuscated code for their test data. They also have designed Nice2Predict [4] which extracts statistical info out of obfuscated JavaScript code and is a base for JSNice. Raychev et al. also have done research on code completion and program synthesis [5] which is relevant to our problem. Bichsel et. al. have developed and tested a statistical method for deobfuscating Android applications [6] with 79.1% success in recovering program identifier names. Another relevant research is [7] which studies filling program missing code slots using the behavior trained from the program input/output data. Lin et al. have done relevant research on translating natural language templates to program templates using Recurrent Neural Networks [8].

3 Dataset

Researchers in Secure, Reliable, and Intelligent Systems Lab at ETH Zurich have collected a data set of 150000 JavaScript programs as a part of their research [2] which is publicly available at [9].

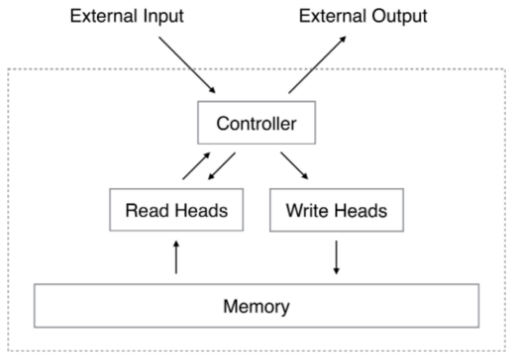
4 Neural Program Learning Model

There are multiple possible approaches for recovering program variable and function names:

- Predicting the obfuscated program tokens using context which is a form of question answering, short answer grading, and translation tasks.
- Using sample input/output data to learn the behavior and recovering the code with attention to an available obfuscated source code. However, a perfect NPL model which works on any programming task is currently not available to our knowledge. But the effectiveness of current models on recovering obfuscated programs for a certain task is subject to research.

- Using obfuscated-deobfuscated program pairs to train the network to translate obfuscated codes.

For our purpose, we want to use Neural Turing Machine (NTM) which uses a neural network as a controller that issues read/write commands to an external memory (a 2d matrix) by attentional processes. The attention uses content-based and location-based addressing. In each iteration, the network gets an input, produces read/write operations and uses read and write heads to alter the memory [10, 11]. We use NTM with an LSTM (Long Short-Term Memory) [12] neural network as the controller.



Neural Turing Machine Architecture [10]

This task (project) will be done by the author of this proposal as the project course for CIS 700-Neural Program Learning course offered by Prof. Katz.

References

[1] W. Jiang, "Method for detecting javascript code obfuscation based on convolutional neural network," *International Journal of Performability Engineering*, vol. 14, 12 2018.

[2] V. Raychev, M. Vechev, and A. Krause, "Predicting program properties from 'big code'," *Commun. ACM*, vol. 62, no. 3, pp. 99–107, 2019. [Online]. Available: http://doi.acm.org/10.1145/3306204

[3] Js nice: Statistical renaming, type inference and deobfuscation. [Online]. Available: https://jsnice.org

[4] Nice2predict. [Online]. Available: https://jsnice.org

[5] V. Raychev, M. Vechev, and E. Yahav, "Code completion with statistical language models," *SIGPLAN Not.*, vol. 49, no. 6, pp. 419–428, Jun. 2014. [Online]. Available: http://doi.acm.org/10.1145/2666356.2594321

[6] B. Bichsel, V. Raychev, P. Tsankov, and M. Vechev, "Statistical deobfuscation of android applications," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 343–355. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978422

[7] M. Bošnjak, T. Rocktäschel, J. Naradowsky, and S. Riedel, "Programming with a differentiable forth interpreter," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, pp. 547–556. [Online]. Available: http://dl.acm.org/citation.cfm?id=3305381.3305438

[8] X. V. Lin, C. Wang, D. Pang, K. Vu, L. Zettlemoyer, and M. D. Ernst, "Program synthesis from natural language using recurrent neural networks," University of Washington Department of Computer Science and Engineering, Seattle, WA, USA, Tech. Rep. UW-CSE-17-03-01, Mar. 2017.

[9] Datasets - learning from "big code". [Online]. Available: https://learnbigcode.github.io/datasets/

[10] A. Graves, G. Wayne, and I. Danihelka, “Neural turing machines,” *CoRR*, vol. abs/1410.5401, 2014. [Online]. Available: http://arxiv.org/abs/1410.5401

[11] N. Kant, “Recent advances in neural program synthesis,” *CoRR*, vol. abs/1802.02353, 2018. [Online]. Available: http://arxiv.org/abs/1802.02353

[12] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735