POSTER: Automatic Patch Generation for Security Functional Vulnerabilities with GAN

Ya Xiao, Danfeng (Daphne) Yao
Department of Computer Science, Virginia Tech
{yax99, danfeng} @vt.edu

***Abstract*—Patching vulnerabilities is not an easy work for developers without corresponding expertise. Our ongoing work pays a special attention to those complicated bugs which cripple security mechanism due to misconfiguration. By applying an overall pattern classifier and parallel GANs, we aim to develop a system which incorporates the automatic diagnosis function to generate patches for vulnerable code snippets as well as the intelligible property which helps the developer to understand the intention of these code snippets. This work is still ongoing. Therefore, complete experiment results are not available yet.**

I. INTRODUCTION

Most security incidents have relation with software vulnerabilities [1]. Among them, vulnerabilities caused by misconfiguration or misuse in security functional code count non-negligible part in CVE database. For example, there are vulnerability types like CWE-312 with the description "Clear-text Storage of Sensitive Information", CWE-325 with the description "Missing Required Cryptographic Step". However, existing automatic patch generation systems [2] [3] pay few attention on them. One reason is that it is hard for those approaches to search complicated fixes more than one line which are often needed in these cases. Researches [4]–[6] applying deep learning techniques is also emerging in this field but also limited to some simple one-line bugs. To handle these more complicated vulnerabilities, there are several common challenges: First, more semantic information is required to be caught so that the tool can deal with the security function of code snippets correctly. This requirement improves the difficulty of embedding a program in a suitable way. Many existing program embedding ways like directly embedding it from a token string of a program may not work. Second, training a good model to generate correct patches is a so complicated task that may require a massive amount of training data, which increases the difficulty to collect dataset and also means extremely costly in the training process.

II. FRAMEWORK

In our works, we incorporate the intelligible property by leveraging the unique characteristic of those security functional codes. That is, they often have some common intentions. Specifically, when establishing a secure connection, there are some specific steps in TLS/SSL protocol, like cipher suite negotiation, authentication, key exchange, application data exchange. Code snippets with the same intention share many commonalities. We call it a pattern here. Therefore, we understand the code intention by a pattern classification first. Then, we treat each pattern separately and apply deep learning techniques to training parallel patch generators of vulnerable code snippets for each pattern. This framework also degrades the challenges of massive training data and elaborate embedding design by properly breaking down a complicated task into several simpler ones.

We apply Generative Adversarial Networks (GANs) [7] in our automatic patch generation process. A discriminator to distinguish code snippets into high-security level and low-security level is pre-trained and a generator accepts low-level code snippets and transfers it into high-security level ones. During the adversarial training process, the discriminator distinguishes the output of generator and high-security level ones. The generator receives the judges from discriminator as indicators to adjust its gradients until its outputs are indistinguishable with high-security level codes.

The framework of our approach is as follows:

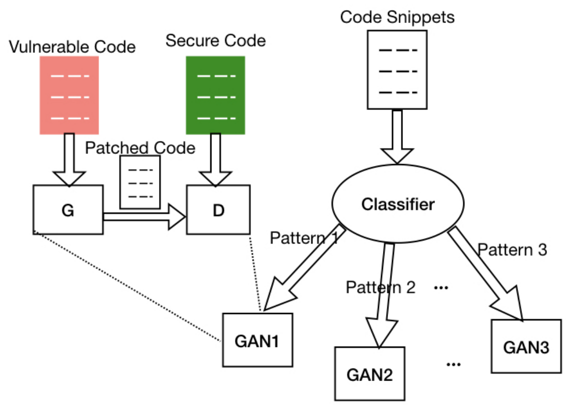

Figure 1: Framework of our approach

Compared with existing works for automatic patches generators, we expect this approach to realize new improvements including highlighting more complicated and convert vulnerabilities on security functional codes, incorporating vulnerabilities detection function as well as patches generation, and intelligible property helping developers to understand the codes.

REFERENCES

[1] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer *et al.*, "The matter of heart-

bleed," in *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, 2014, pp. 475–488.

[2] C. Le Goues, T. Nguyen, S. Forrest, and W. Weimer, "Genprog: A generic method for automatic software repair," *Ieee transactions on software engineering*, vol. 38, no. 1, p. 54, 2012.

[3] F. Long and M. Rinard, "Staged program repair with condition synthesis," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 2015, pp. 166–178.

[4] R. Gupta, S. Pal, A. Kanade, and S. Shevade, "Deepfix: Fixing common c language errors by deep learning." in *AAAI*, 2017, pp. 1345–1351.

[5] R. Gupta, A. Kanade, and S. Shevade, "Deep reinforcement learning for programming language correction," *arXiv preprint arXiv:1801.10467*, 2018.

[6] J. Harer, O. Ozdemir, T. Lazovich, C. P. Reale, R. L. Russell, L. Y. Kim, and P. Chin, "Learning to repair software vulnerabilities with generative adversarial networks," *arXiv preprint arXiv:1805.07475*, 2018.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.