효율적인 프로그램 행위 모델링을 위한 데이터 임베딩 연구

안선우*, 김현준*, 하회리*, 안성관*, 백윤흥*
*서울대학교 전기정보공학부, 서울대학교 반도체 공동연구소
swahn@sor.snu.ac.kr, hjkim@sor.snu.ac.kr, wrha@sor.snu.ac.kr, sgahn@sor.snu.ac.kr, ypake@snu.ac.kr

A Study on Data Embedding for Efficient Program Behavior Modeling

Sunwoo Ahn*, Hyunjun Kim*, Whoi Ree Ha, Seonggwan Ahn, Yunheung Paek*
*Dept. of Electrical and Computer Engineering and Inter-university Semiconductor Research Center, Seoul National University

요 약

최근 프로그램은 그 크기와 복잡도가 나날이 증가하고 있어, 프로그램 행위 모델링에 대한 중요성은 다양한 분야에서 증대되고 있다. 프로그램의 동적 분석은 런타임에 생성되는 데이터가 너무나 많아, 많은 데이터를 활용하기 용이한 딥러닝 기술이 사용되고 있다. 하지만, 기존의 연구들은 연산과 연산의 매개변수 중 매개변수에 대한 고려가 충분치 않았다. 이는 매개변수가 딥러닝에 알맞은 입력으로 표현되기 어렵기 때문인데, 우리는 이를 해결하기 위해 매개변수로 인해 발생되는 런타임 행위에서 특징적인 값들을 추출하는 것으로 대체하였다. 또한, 연산과 특징적인 값들이 여러 개의 LSTM-RNN 으로 처리됨을 보이고, 이 결과를 시각화 하여 효과적임을 보였다.

1. 서론

프로그램 행위 모델링은 침입 탐지, 취약점 탐지, 소프트웨어 테스트 등 다양한 분야에서 적용되고 있다. 프로그램 행위 모델링을 하는 방법에는 크게, 프로그램을 실행하지 않고 분석하는 정적 분석과 프로그램을 수행하며 얻은 정보를 활용하는 동적 분석이 있다. 정적 분석에는 정보량에서 그 한계가 있기 때문에, 정확도를 최대한으로 끌어내기 위해서는 동적 정보를 활용하는 동적 분석이 필요하다. 동적 분석을 하는 연구들은 일반적으로 프로그램을 수행하며 수행된 연산을 위주로 프로그램을 분석한다.[1-5] 특히, [1,2,5]은 최근 방대한 데이터를 처리하는데 뛰어난 성능을 보이는 딥러닝을 차용하여, 프로그램 행위 중 발생하는 많은 양의 데이터를 학습하여 프로그램 행위 모델링을 하고 있다. 딥러닝을 사용하는 연구의 또 다른 장점은 기존 연구[3,4]가 사람이 정제한 데이터에 의존하기 때문에 모델의 능력도 전문가의 지식에 한정되는 반면에, 딥러닝은 데이터 정제 방법을 학습 과정에서 배우기 때문에 데이터에 내재된 다양한 특성을 파악할 수 있다.

하지만, [6]에 따르면, 프로그램 행위는 연산 수행에 의해 변경된 상태의 변화 과정으로 정의할 수 있으므로, [1-5]의 방식은 프로그램 행위의 두 가지 정보 (연산, 상태) 중 상태 정보를 배제한체 프로그램 행위 모델링을 하고 있다. 우리는 각 상태 정보는 연산의 매개변수에 포함되어 있다는 점에서 착안하여, 프로그램 행위 모델링을 연산과 연산의 매개변수를 활용하려 한다. 이 때, 매개변수는 가능한 값의 범위가 너무나 넓어 인공신경망의 입력으로는 적절치 않기 때문에 추가적인 정제가 필요하다. 예를 들어, 64-bit 기계에서 정수 값은 -2^{32} ~ 2^{32} 사이의 값을 가질 수 있다. 이 값을 그대로 사용하게 되면, flag 와 같이 값의 의미가 정수 공간에 있지 않은 경우, 모델이 더 가까운 값을 의미 상 가깝다고 오인하게 된다. 이를 방지하기 위해 데이터를 중립적으로 표현하는 one-hot encoding 을 활용하면 vector 크기가 너무 커지게 된다. 따라서, 우리는 프로그램이 계층적 구조를 가진다는 점에 착안하여, 매개변수를 대체할 수 있는 특징적인 값들을 찾았다. 어떠한 연산은 그 연산을 수행하기 위한 내부 연산이 존재하고, 이 내부 연산은 매개 변수에 의해 조정된다. 즉, 내부 연산을 기존 연구[1-5]이 연산을 입력을 표현한 방식을 차용하여 표현하면 매개변수를 효율적으로 표현할 수 있다.

이 표현 방식을 이용해 우리는 프로그램 행위 모델링을 인공신경망으로 하였다. 학습된 모델이 프로그

램을 적절히 모델링 하고 있는지 증명하기 위해, 우리는 모델 내부의 vector 가 적절한 형태로 구성되었는지 시각화하여 보였다.

2. 인공신경망 기반의 모델

프로그램이 수행되며 발생하는 연산들은 시계열 데이터이므로, 우리는 시계열 데이터를 처리하는데 능한 LSTM-RNN[7]을 이용하였다. 연산 시계열 정보와 상태 시계열 정보가 섞이지 않도록 하기 위하여, 두 개의 LSTM-RNN 모델을 사용하였다. LSTM-RNN 모델은 임베딩 층, LSTM-RNN 층, softmax 층으로 구성되어 있다. 연산 시계열을 위한 LSTM-RNN 모델과 상태 시계열을 위한 LSTM-RNN 모델은 각각 아래 수식을 최적화하여 모델의 매개변수를 조정하게 된다.

$$\underset{\Theta}{\operatorname{argmax}} \sum_{i=0}^{n} H_f(f_i, F_{i-1}; \Theta)$$

where:

Θ = the network parameters
n = the number of functions
f_i = a specific function
F_i = a sequence of functions
$= f_0, f_1, \ldots, f_t$
F_{-1} = an empty sequence

(수식 1) 연산 시계열을 위한 모델 최적화.

$$\underset{\Theta}{\operatorname{argmax}} \sum_{i=0}^{n} \sum_{j=0}^{k_i} \tilde{H}_c(c_{i,j}, C_{i,j-1}, cx_i; \Theta)$$

where:

Θ = the network parameters
n = the number of functions
k_i = the length of the CV sequence
$c_{i,j}$ = a specific CV corresponding to f_i
$C_{i,j}$ = CV sequence corresponding to f_i
$= c_{i,0}, c_{i,1}, \ldots, c_{i,k_i}$
cx_i = the context vector of f_i extracted from the LSTM layer of the function sub-network
$C_{i,-1}$ = an empty sequence

(수식 2) 상태 시계열을 위한 모델 최적화.

이때, 상태 정보가 어떤 연산에 연관되어 있는지 알 수 있도록 연산 시계열을 위한 LSTM-RNN 층의 출력(context vector)을 상태 시계열을 위한 LSTM-RNN 층의 입력으로 넣었다. 연산 시계열을 위한 모델의 수식으로 최적화된 모델은 i 번째 연산이 발생했을 때, 1~i-1 번째 연산들을 고려하여 i 번째 연산을 잘 예측하는지를 측정하여 프로그램 행위 모델링을 하게 된다. 상태 시계열을 위한 모델의 수식으로 최적화된 모델은 i 번째 연산의 j 번째 내부 연산이 발생하였을 때, 1~i-1 번째 연산들과 1~j-1 번째 연산들을 고려하여 j 번째 내부 연산을 잘 예측하는지를 측정하여 프로그램 행위 모델링을 하게 된다.

3. 실험 결과

프로그램 행위 모델링이 잘 되었는지 확인하기 위하여, 우리는 시스템 호출을 연산, 시스템 호출 내부의 분기 연산을 내부 연산으로 설정하여 MySQL 과 GNU Screen 의 행위를 모델링하였다. 시스템 호출 내부의 분기 연산은 시스템 호출의 매개변수에 의해 변경되므로 앞서 언급한 매개변수를 대체할 수 있는 내부 연산으로서 분기 연산은 적절하다.

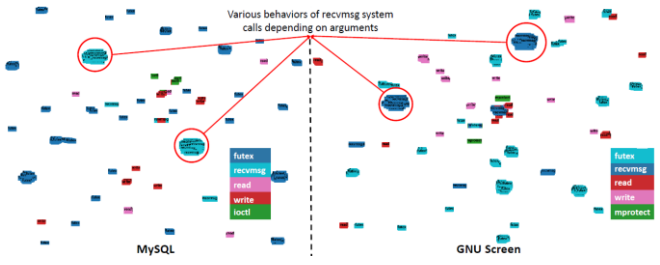
학습을 위한 데이터를 모으기 위해, 우리는 연구실에 가동 중인 서버에서 프로세스가 호출하는 시스템 호출 시퀀스와 이들의 분기 연산 시퀀스를 모았다. 특히, MySQL 과 GNU Screen 의 경우 프로그램 행위 모델링이 정확히 되었는지 확인할 목표 프로그램이므로, 데이터를 모을 때 활발히 사용되도록 하였다. 데이터의 시스템 호출 시퀀스 개수와 분기 연산 시퀀스 길이는 각각 1,921 개, 2,264,553 개이다.

모델의 초월 매개변수는 <표 1>에 정리되어 있다.

<표 1> 모델의 초월 매개변수

초월 매개변수	연산 시계열을 위한 모델	상태 시계열을 위한 모델
연산 종류	400	4096
LSTM 크기	8	64
Epoch 수	20	
Batch 크기	256	
최적화 알고리즘	Adam	
Learning rate	10^{-4}	

이렇게 학습된 모델이 실제 프로그램 행위를 잘 반영하고 있는지 확인하기 위해, 우리는 시스템 호출의 내부 분기 연산의 임베딩 결과가 시스템 호출의 행위를 잘 나타내고 있는지 시각화 해보았다. 각 시스템 호출의 내부 분기 연산들의 임베딩 vectors 를 평균 내어, 해당 시스템 호출을 대표하도록 하였다. 평균 내어진 임베딩 vectors 의 시각화를 위해 Embedding Projector[8]를 이용하여 Uniform Manifold Approximation and Projection (UMAP)[9] 알고리즘을 적용하였다.



(그림 2) 내부 연산 임베딩 vector 시각화

직관적으로 해석할 수 있도록, 모든 시스템 호출을 시각화하는 대신, 가장 빈번하게 발생된 5 개의 시스템 호출을 시각화하였다. 해당 시스템 호출은 그림의 오른쪽 아래에 제시되어 있다. (그림 1)을 보면 같은 시스템 호출이 몇가지의 클러스터를 형성하고 있음을 알 수 있다. 이는 시스템 호출이 매개변수 변화에 따라 몇 가지 기능을 하고 있는 것들이 시각화되었다고 볼 수 있다. 더 자세한 설명을 위해 예를 들자면, “recvmsg”라는 시스템 호출은 MSG_OOB, MSG_PEEK, MSG_WAITALL 과 같은 행위를 하도록 구현되어 있고, 이는 3 번째 매개변수인 flags 를 통해 결정된다. (그림 1)의 붉은 선 표시를 보면, 이러한 사항들이 반영되어 임베딩이 적절히 되었음을 알 수 있다.

4. 결론

본 연구에서는 프로그램 행위 모델링을 위하여 내부 연산을 활용하는 것이 더 정확하고 효율적인 연프로그램 행위 모델링이 잘 되었는지 확인하기 위하여, 우리는 매개변수를 내부 연산으로 대체하여 학습한 모델에서 추출한 임베딩 vector 를 추출하여 시각화하였다. 이 시각화된 vectors 를 분석한 결과, 제시된 모델이 프로그램 행위 모델링을 적절히 하고 있음을 확인하였다.

5. 사사문구

이 논문은 2021 년도 BK21 FOUR 정보기술 미래인재 교육연구단과 2021 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2020R1A2B5B03095204).

참고문헌

[1] L. Chen, S. Sultana and R. Sahita, Henet: A deep learning approach on intel® processor trace for effective exploit detection, IEEE Security and Privacy Workshops (2018), pp. 109–115.

[2] G. Kim, H. Yi, J. Lee, Y. Paek and S. Yoon, LSTM-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems, arXiv preprint arXiv:1611.01726 (2016).

[3] X. Shu, D. Yao and N. Reamakrishnan, Unearthing stealthy program attacks buried in extremely long execution paths, Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (2015), pp. 401–413.

[4] K. Xu, K. Tian, D. Yao and B. G. Ryder, A sharper sense of self: Probabilistic reasoning of program behaviors for anomaly detection with context sensitivity, 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (2016), pp. 467–478.

[5] H. Yi, G. Kim, J. Lee, S. Ahn, Y. Lee, S. Yoon and Y. Paek, Mimicry resilient program behavior modeling with lstm based branch models, arXiv preprint arXiv:1803.09171 (2018).

[6] A. Armando and G. Lowe, Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security, JointWorkshop, ARSPAWITS 2010, Paphos, Cyprus, March 27-28, 2010, Revised Selected Papers. Vol. 6186. Springer, 2010

[7] S. Hochreiter, and J. Schmidhuber, “Long short-term memory,” Neuralcomputation, vol. 9, 8, pp. 1735–1780, 1997.

[8] D. Smilkov, N. Thorat, C. Nicholson, E. Reif, F. B. Vi´egas and M. Wattenberg, Embedding projector: Interactive visualization and interpretation of embeddings, arXiv preprint arXiv:1611.05469 (2016).

[9] L. McInnes, J. Healy and J. Melville, Umap: Uniform manifold approximation and projection for dimension reduction, arXiv preprint arXiv:1802.03426 (2018).