감정 단어를 활용한 보안 버그의 분류

김영경*, 허진석** 김미수*, 이은석*
*성균관대학교 소프트웨어대학
**성균관대학교 정보통신대학
e-mail : { agnes66, mrhjs225,misoo12,leees}@skku.edu

Classification of Security Bugs Using emotional word

Young-Kyoung Kim*, Jin-Seok Heo**, Misoo Kim *, Eun--seok Lee*
*College of Software, Sung-Kun-Kwan University
**College of Information and Communication Engineering, Sung-Kyun-Kwan University

요 약

최근 보안 버그의 중요성이 증가함에 따라, 버그 리포트 중 보안과 관련된 리포트를 빠르게 분류하는 기술이 필요하다. 기존 기술들은 버그 리포트의 단어들을 가지고 기계학습을 위한 훈련 데이터를 생성한다. 이 때 기계학습에 잡음이 발생하면 성능을 떨어뜨릴 수 있다. 이를 보완하기 위해 본 연구에서는 감정 단어를 활용하여 잡음을 줄인 보안 버그리포트를 자동으로 식별하는 기계학습기반 기술을 제안한다. 제안 기술은 기계학습 시 사용되는 훈련 데이터의 품질을 높이기 위해 감정 단어를 활용한다. 실험 결과 감정 단어를 활용했을 때 기존 기술 대비 보안 버그를 분류하는 정확도가 3.03% 향상되었다.

1. 서론

최근 소프트웨어의 보안 버그의 수가 증가하고 있다[1]. 사람들의 개인정보를 담고 있는 소프트웨어에서 보안 버그가 발생시 개인정보 유출 및 금전적 손해로 이어질 수 있다. 이러한 보안 버그를 고치기 위해서는 보안 버그 리포트(Security Bug Report: SBR) 형태로 개발자에게 알려질 수 있다. 그러나 발생하는 버그리포트의 수가 증가함에 따라 SBR 의 분류를 사람이 일일이 하기 힘들다[3]. 따라서 자동 SBR 분류 기술이 필요하다.

SBR 자동 분류 기술에 관한 연구는 활발히 이루어져 왔다 [1,7]. 이 기술은 기본적으로 버그리포트의 내용을 자연어 처리 후 기계학습을 통해 버그리포트를 분류한다[7]. 그러나 학습되는 데이터에 보안 버그리포트의 수가 매우 적기 때문에, 학습데이터의 불균형이 존재하고, 이는 분류 정확도에 악영향을 끼친다. 이를 해결하기 위해 훈련에 잡음이 될 수 있는 단어를 축소하는 것이 필요하다[2].

버그를 설명하고 있지 않는 단어는 기계학습 시 잡음이 된다. 감정 단어는 버그리포트 작성 시 빈번하게 사용되기 때문에 기계학습 데이터에 포함이 될 수 있고 이는 성능저하로 이어진다. 우리는 이러한 감정 단어를 고려한 기계학습 기반 기술을 제안한다. 학습 데이터 불균형을 해결하기 위해 감정 단어를 고려하여 보안 버그 리포트가 아닌 리포트(Non-Security Bug Report: NSBR)를 필터링한다. 정제된 리포트로 기계학습을 통해 SBR 예측모델을 생성한다.

본 논문의 구성은 다음과 같이 이루어진다. 2 장 에서는 보안 버그의 정의 및 관련 연구를 설명하고 3 장에서는 SBR 자동 분류 기술을 제안한다. 4 장에서는 방법론을 통한 실험방법들에 대해 설명하고 5 장에서는 실험 결과 및 분석을 통해 해당 기술의 성능을 검증한다. 마지막 6 장에서 결론과 추가 연구를 제안한다.

2. 관련 연구

2.1. 보안 버그

보안 버그에 관한 연구는 꾸준히 진행되고 있다. F.Peters et al.은 사용자의 부적절한 접근을 통해 소프트웨어 자체에 손상을 입히는 경우와 정보 유출에 영향을 주는 버그를 보안 버그로 정의한다[2]. Z.Wan et al은 소프트웨어 내의 메모리에 손상을 입힐 수 있으며, 소프트웨어가 제공하는 서비스에 손상을 입힐 수 있는 취약점을 보안 버그로 정의했다[4]. 정보 보안 취약점 표준 코드(Common Vulnerabilities and Exposures: CVE)에 따르면 CVE-2014-0092, CVE-2015-0208, CVE-2015-0288 등에서 예외처리 버그가 보안 취약을 유발한다고 밝혀졌다. 따라서 본 연구에서는 보안 버그를 F.Peters et al.의 기준과 함께, 메모리에 손상을 입힐 수 있는 취약점을 포함하여 보안 버그라 정의한다.

2.2. 기계학습 기반 보안 버그 리포트 분류

최근에 기계학습을 이용한 보안 버그 리포트의 종류를 분류하는 다양한 연구들이 제안되었다. M.Gegick et al.은 SBR 을 분류하기 위해 분류가 되어 있는 BR

의 통계 모델을 훈련시켜 NSBR 로 잘못 표기된 SBR 을 식별하는 기술을 제안했다 [5]. D. Behl et al 은 보안 버그 분류를 위해 데이터 불균형을 보완하는 Naïve Bayes 분류기를 제안했다 [8]. 그러나 학습되는 데이터에 보안 버그 리포트의 수가 매우 적기 때문에, 학습데이터의 불균형이 존재하고, 이는 분류 정확도에 악영향을 끼친다.

위 문제를 해결하기 위해 가장 최근에 F. Peters et al. 은 SBR 과 NSBR 에 동시에 존재하는 보안 교차 단어(Security Cross Word: SCW)를 통해 버그 리포트를 필터링하여 훈련 데이터의 품질을 향상하여 SBR 을 분류하는 기술을 제안했다[2].

기존 기술들은 모두 기계학습을 위해 버그 리포트의 단어들로 훈련 데이터를 생성한다. 이 때 감정 단어가 기계학습 시 잡음을 유발할 수 있다. 본 연구에서는 감정 단어를 활용해 버그 리포트를 필터링하고, 훈련데이터를 강화하고자 한다.

3. 제안 기술

3.1. 버그 리포트 필터링

우선 버그 리포트의 정보 추출을 위해 자연어 전처리 과정을 거친다. 기존 연구에서 보안 버그 분류를 위해 사용한 자연어 전 처리 순서에 따른다[2]. 전 처리 과정은 토큰화와 불용어 제거 단계로 나누어진다. 토큰화 단계에서는 수집한 버그 리포트를 단어 단위로 나눈다. 불용어 제거 단계는 'me, he, is '와 같이 의미가 없는 관사, 조사를 제거하는 단계이다.

버그리포트는 사람이 작성하기 때문에 감정 단어를 포함할 수 있다. Q. Umer, Y. Geunseok 등은 감정 단어를 활용하여 버그의 심각도를 예측하는 기술을 제안했다[9,10]. 이처럼 감정 단어는 버그리포트에 자주 등장한다. 따라서 버그를 설명하지 않는 단어임에도 불구하고 기계학습 데이터로 활용될 수 있다. 우리는 버그와 무관한 정보가 섞이는 잡음을 제거하기 위해 감정 단어[1]를 제거한다. 이를 통해 버그를 설명하는 단어들을 명확하게 식별하고자 한다.

위의 과정을 통해 문서 속 단어 데이터가 도출된다. SCW 을 제거하기 위해 사용되는 문서-단어 행렬을 생성하기 위해 단어 중요도 계산 수식인 tf-idf(Team Frequency-Inverse Document Frequency)를 사용한다. (식 1~3)은 TF-IDF 를 계산하기 위한 식으로 기존의 텍스트 마이닝을 사용한 연구를 참조하였다[6].

$$\mathrm{tf}(t, d) = 0.5 + \frac{0.5 \times f(t,d)}{\max\{f(w,d) : w \in d\}} \quad \text{(식 1)}$$

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \quad \text{(식 2)}$$

$$\mathrm{tf-idf}(t, d, D) = \mathrm{tf}(t, d) \times \mathrm{idf}(t, D) \quad \text{(식 3)}$$

위의 식에서 N 은 버그 리포트의 전체 수이고 d 는 단일 문서, t 는 용어, D 는 문서의 집합을 뜻한다. tf(t, d)는 문서에 단어 별로 자주 나타나는 빈도수를 의미한다. idf(t, D)는 문서의 집합 내에 단어가 얼마나 나타나는지를 의미한다. 두 계산 값을 이용해 tf-idf(t, d, D)값을 계산한다. SBR 에 존재하는 단어들을 tf-idf(t, d, D)값을 기준으로 정렬 시 상위 100 개의 단어를 보안 관련 단어(Security Related Keyword: SRK)로 추출한다. 이 단어들을 대상으로 문서당 단어의 빈도수로 이루어진 문서-단어의 행렬을 형성한다.

다음으로 SRK 들의 점수를 계산한다. 이를 위해 분류기가 NSBR 과 SBR 을 구별할 때 혼동을 주는 SCW 를 활용하여 훈련데이터에서 NSBR 을 필터링하는 알고리즘을 참조했다[2]. 참고한 알고리즘은 초기 훈련 데이터의 NSBR 에 SRK 단어가 얼마나 나타나는지를 의미하는 단어 빈도수와 그 빈도수를 강화하기 위한 필터값을 이용하여 SRK 들을 점수화시킨다.

형성된 행렬과 SRK 의 점수를 토대로 기존의 보안 버그 리포트 분류 연구 중 버그 리포트 점수화 알고리즘을 참조했다[2]. (식 4)를 토대로 계산되며 최종적으로 0~1 사이의 점수가 도출된다.

$$\text{Report Score} = \frac{\prod_{i=1}^{|M*|} m_i}{\prod_{i=1}^{|M*|} m_i + \prod_{i=1}^{|M'|} (1 - m_i)} \quad \text{(식 4)}$$

(식 4)에서 M 은 버그 리포트의 단어 별 점수들의 dictionary, M*은 버그 리포트의 SRK 점수 목록, M'은 버그 리포트의 SRK 보완 점수 목록, m 은 M*의 원소를 의미한다.

실제 SBR 은 NSBR 의 수보다 훨씬 작다. 따라서 기계학습 시 발생하는 데이터 불균형 문제를 해결하기 위해 임계 값 이상의 값을 가지는 리포트는 NSBR 으로 취급하고 학습데이터에서 제거한다.

3.2. SBR 식별

Naïve Bayes 는 효율적인 기계학습 알고리즘 중 하나이다[7]. 다른 알고리즘에 비해 데이터 불균형 환경에서 소프트웨어 결함을 찾는데 뛰어났다[12,14]. 3.2 의 과정을 통해 불균형을 해소하긴 했지만 성능의 향상을 위해 Naïve Bayes 분류기를 사용하여 SBR 을 식별하기 위한 분류모델을 만든다.

4. 실험

4.1. 실험 대상

기존의 연구 결과와 비교하기 위해 기존 보안 버그 리포트의 분류에 관한 연구에서 사용된 대상인 ambary, wicket, camel, derby 프로젝트를 대상으로 실험한다.

4.2. 평가 방법

알고리즘의 성능을 평가하는 지표 중 하나인 f-measure 을 사용하여 본 연구의 결과를 평가한다. F-measure 값은 정확도와 재현율의 조화 평균값으로 정

[1] http://www.psychpage.com/learning/library/assess/feelings.html

확도와 재현율의 트레이드오프를 적절히 통합한다. 혼동행렬은 실제의 값과 예측 값의 참, 거짓을 판단하여 총 4 가지 경우의 수를 계산한다. 혼동행렬의 구성은 <표 1>과 같다.

	실제		
예측	구분	SBR	NSBR
	SBR	TP	FP
	NSBR	FN	TN

<표 1> 혼동행렬

이를 이용하여 재현율(recall)과 정확도(precision)를 계산한다. 최종적으로 둘의 트레이드오프를 잘 통합하여 정확성을 한번에 나타내는 f-measure 을 계산한다. (식 5-7)은 이를 계산하기 위한 수식들이다.

$$\text{Precision} = \frac{TP}{TP+FP} \quad \text{(식 5)}$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad \text{(식 6)}$$

$$\text{F} - \text{measure} = \frac{2\times Precision\times Recall}{Precision+Recall} \quad \text{(식 7)}$$

4.3. 연구 질문

RQ. 잡음제거에 감정 단어를 활용했을 때 성능

기존의 보안 버그 리포트 분류 연구[2]의 제안 기술인 FARSEC 과의 비교를 통해 본 연구의 SBR 분류 성능을 평가한다. 결과적으로 감정단어의 제거가 SBR 분류 성능을 얼마나 향상시키는지 확인한다.

5. 실험 결과

RQ. 잡음제거에 감정 단어를 활용했을 때 성능

불용어 제거 단계에서 감정 단어를 추가함으로써 <표 2>와 같은 결과를 얻을 수 있었다.

필터 구분	기존 연구	감정 단어 추가
Farsecsq	7.2	7.2
Farsectwo	8.0	8.1
Farsecnone	14.6	15.3
평균	9.9	10.2

<표 2> 감정 단어 추가의 f-measure 결과값

Farsecsq, Farsectwo 와 Farsecnone 은 F.Peters 의 연구에서 제안한 필터값으로 각각 원래의 단어 빈도수에 제곱한 것, 2 배를 한 것, 아무 값도 곱하지 않은 것이다[2]. 필터값이 Farsecsq 일 때는 기존의 연구에 비해서 성능의 향상이 없었다. 하지만 필터가 Farsectwo 와 Farsecnone 일 때는 각각 1.3%, 4.8% 가량의 성능 향상이 있음을 확인할 수 있다. 불용어에 감정 단어를 추가하였을 때 평균적으로 3.03%가량 성능이 향상했다.

6. 결론 및 제안 연구

보안 버그의 중요성이 높아짐에 따라 빠른 보안 버그의 수정이 필요해졌다. 따라서 개발자들에게 알려지는 BR 중 SBR 을 자동으로 정확하게 식별해야 한다. 본 연구에서는 보안 버그 리포트의 자동 식별을 위해 감정 단어를 활용하여 보다 정확하게 식별하고자 했다.

불용어 제거 단계에서 감정 단어를 제거를 통해 기계학습 시 잡음을 제거함으로써 보안 버그 리포트를 보다 정확하게 식별할 수 있었다.

우리는 향후 더 정확한 식별을 위해 빅데이터와 딥러닝을 적용하는 연구를 진행할 계획이다.

ACKNOWLEDGMENTS

본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 SW 중심대학지원사업의 연구결과로 수행되었으며(2015-0-00914), 2018 년도 정부(교육부)의 재원으로 한국과학창의재단(2018 년도 학부생 연구프로그램)의 지원을 받아 수행된 연구임

참고문헌

[1] Cvedetails.com. (2018). CVSS Score Distribution Reports and Trends Over Time. [online] Available at: https://www.cvedetails.com/cvss-score-charts.php [Accessed 25 Jul. 2018].

[2] F. Peters et al. "Text Filtering and Ranking for Security Bug Report Prediction," IEEE Transactions on Software Engineering, 2017.

[3] L. Erlikh “Leveraging legacy system dollars for e-business,” IT Professional, vol. 2, no. 3, pp. 17-23, 2000.

[4] Z. Wan et al. “Bug characteristics in blockchain systems: A largescale empirical study,” Proc. of the 2017 IEEE/ACM 14th International Conference on Mining Software Repositories, pp. 413-424, 2017.

[5] M. Gegick et al. “Identifying security bug reports via text mining - an industrial case study,” Proc. of the 7th IEEE Working Conference on Mining Software Repositories, pp. 11-20, 2010

[6] R. Coelho et al. “Unveiling Exception Handling Bug Hazards in Android based on GitHub and Google Code Issues,” Proc. of the 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, pp. 134-145, 2015.

[7] J. Xuan et al. “Automatic Bug Triage Using Semi-Supervised Text Classification,” Proc. of 22nd International Conference on Software Engineering and Knowledge Engineering, 2010

[8] D. Behl et al. "A bug Mining tool to identify and analyze security bugs using Naive Bayes and TF-IDF," Proc. of the 2014 International Conference on Reliability Optimization and Information Technology, pp. 294-299, 2014.

[9] Q. Umer, et al. "Emotion Based Automated Priority Prediction for Bug Reports," IEEE Access, vol. 6, pp. 35743-35752, 2018.

[10] Y. Geunseok, et al. "Analyzing emotion words to predict severity of software bugs: a case study of open source projects," Proc. of the Symposium on Applied Computing, pp1280-1287, 2017.