Journal of the Korea Academia-Industrial cooperation Society
Vol. 24, No. 9 pp. 16-23, 2023

https://doi.org/10.5762/KAIS.2023.24.9.16
ISSN 1975-4701 / eISSN 2288-4688

딥러닝 기반 소스코드 취약점 자동 패치 시스템 설계 및 구현

이관목[1*], 이상도[2], 이용준[1]
[1]극동대학교 일반대학원 인공지능보안학과, [2]육군사관학교

Design and Implementation of Deep Learning-based Source Code Vulnerability Automatic Patching System

Kwan-Mok Lee[1*], Sang-Do Lee[2], Young-Jun Lee[1]
[1]Department of Artificial Intelligence Security, Graduate School, Far East University
[2]Korea Military Academy

요 약 소스코드 내의 악성코드를 식별할 수 있는 기법으로는 소스코드 기반 분석과 바이너리 기반 분석, 2가지 방법이 있다. 하지만 소스코드 기반 분석 방법은 정확도가 높지만 악성코드가 들어있는 소스코드를 수집하기가 어렵고, 바이너리 기반 분석은 바이너리를 수집하는 것은 용이하지만 악성코드를 검출 및 분석 정확도가 떨어진다는 한계점이 있다. 본 연구에서는 이러한 기존의 소스코드 취약점 진단 기술 및 시스템의 문제점을 해결하기 위해 딥러닝을 이용하여 소스코드 취약점 제거를 위한 소스코드 데이터셋을 학습시켜 자동화된 패치까지 제공함으로써 일관된 소스코드 취약점 제거를 가능하게 하는 모델을 제안하고, 이를 시스템을 구현하였으며, 구현된 시스템에 대한 실험을 통해 성능평가를 수행하였다. 성능평가 결과, 소스코드 취약점 자동패치 정확도는 99.8%, 소프트웨어 취약점에 대한 점수화 확인, 발견된 취약점별 추적정보 확인, 행렬형 보기의 분류 정확도 확인 그리고 취약점 분석에 따른 자동 진단율은 96.5%의 결과를 얻을 수 있었다.

Abstract Source-code-based and binary-based analysis techniques can identify malicious code in source code. Although the source code-based analysis method has high accuracy, it is difficult to collect source codes containing malicious codes. Although it is easy to collect binaries, the binary-based analysis method has limitations in that the accuracy of detecting and analyzing malicious codes is low. In this study, we propose a model that can learn source-code datasets to remove source code vulnerabilities using deep learning and automatically remove them. The model was implemented, and its performance was evaluated. The performance evaluation showed promising results, and the automatic patching of source code vulnerabilities achieved 99.8% accuracy. In addition, software vulnerabilities were scored 10 times, and the presence or absence of tracking information for each identified vulnerability was confirmed on five separate occasions. Notably, the automatic diagnosis rate based on the analysis was 96.5%.

Keywords : Deep Learning, Source Code, Vulnerability, Auto-patching, CVSS

1. 서론

최근 들어, ICT 기술의 급격한 발달은 자율주행 자동차, 헬스케어, 지능형 물류 등 다양한 산업 분야에서 최첨단 기술 및 시스템을 통해 많은 사용자에게 매우 유용한 도움을 주고 있으며, 사회 전반에 걸쳐 매우 혁신적인 변화를 일으키고 있다. 하지만 이러한 혁신적인 변화의 이면에는 표준화되고 점점 고도화되는 악성코드의 대량

*Corresponding Author : Kwan-Mok Lee(Department of Artificial Intelligence Security, Graduate School, Far East University)
email: bush@code1.co.kr
Received July 7, 2023 Revised July 27, 2023
Accepted September 1, 2023 Published September 30, 2023

생산과 지속되는 무분별한 APT(Advanced Persistent Threat) 공격 등의 좋지 않은 변화도 일어나고 있다. ICT 시스템에 매우 큰 위협을 가하는 사이버 공격 또한 계속 고도화를 넘어 이제는 더욱 지능화되고 있어서 기존의 대응 방법으로는 적절히 대응하기가 어려운 한계에 이르렀다.

코로나 감염병 범유행 이후 비대면 활동이 증가함에 따라 기업에서도 재택근무 환경을 권장하였으며, 이에 따라 급변한 비대면 원격근무 환경에서는 보안에 취약할 수밖에 없는 여러 취약점이 나타났다. 이러한 취약점은 기업 내부에 침투하는 해커들에 의해 이용당했으며, 이를 통해 기업의 주요 정보가 유출되는 사고가 발생하였다[1].

2022년 3월, 삼성전자가 해킹그룹인 '랩서스(LAPSUS$)'의 공격으로 인해 일부 기밀 자료가 유출된 사실이 드러났으며, 전문가들은 이렇게 유출된 소스코드를 이용한 추가적인 피해사례가 앞으로 계속 발생할 수 있다고 우려하였다[2].

소스 코드는 기업 또는 기관이 보유 중인 수많은 디지털 자산 중 매우 중요한 자산 중 하나이다. 소스 코드는 기획부터 시작하여 코딩, 테스트, 디버깅, 상용화 또는 배포 등을 거치며 매우 정교하고 세밀하게 다듬어지는 데에 매우 많은 시간이 소요된다. 특히, 기술을 기반으로 한 기업들에 있어서 소스 코드는 해당 기업의 서비스 기획서 또는 제품의 설계도와 다름이 없다. 그러므로 소스 코드는 디지털 자산 중에 가장 높은 가치를 가진 자산이며, 실무환경에서 기업과 기관들 대부분이 그렇게 인식하고 있다[2,3]. 따라서 소스 코드에서부터 기업의 가치가 창출된다고 해도 과언이 아닐 정도로 소스 코드는 기업에 있어서 매우 중요하고 핵심적인 디지털 자산이다[4].

2022년에 발표된 버라이즌(Verizon)社의 데이터 침해 조사 보고서(2022 Data Breach Investigations Report; DBIR)에 따르면 2016년 이후로 트로이목마와 같은 단일성 악성코드의 공격이 줄어든 대신 취약점을 찾아서 오랜 기간 공격하는 지능형 지속적 공격인 APT 공격과 같은 사이버 공격이 증가했다고 하였다[5]. 또한 2021년 발표된 'CISCO 중소기업의 사이버 보안: 아시아·태평양 디지털 방어 보고서[6]'에 따르면 한국의 중소기업 중 33%가 사이버 공격을 경험했다고 하였고, 그중 약 85%가 악성코드에 의한 공격이라고 보고되었다[5]. 기존의 사이버 공격 대응 방법은 전문 인력에 의존한 보안관제 시스템 구축을 통한 모니터링이었다. 이러한 방법은 모듈화로 인해 악성코드가 대량으로 생산되거나, 변이된 사이버 공격 또는 APT공격의 경우, 적절히 대응하기 어렵고, 중소기업의 경우에는 보안관제 시스템을 자체적으로 구축하기 쉽지 않다는 한계점이 있다[7].

이러한 악성코드 식별은 소스코드 기반 분석과 바이너리 기반 분석, 2가지로 구분할 수 있다[8].

첫 번째, 소스코드 기반 분석은 악성코드의 소스코드를 수집한 후 사용된 변수 이름, 함수 이름, 문법 등을 분석하여 저자의 특징을 정의함으로써 저자를 식별하는 것이고, 두 번째, 바이너리 기반 분석은 악성코드의 바이너리를 추출 및 분석함으로써 저자의 특징을 정의하고 식별하는 방법이다[7]. 소스코드 기반 분석 방법은 정확도가 높지만, 악성코드가 들어있는 소스코드를 수집하기가 어려운 한계점이 있고, 바이너리 기반 분석은 바이너리를 수집하기는 쉽지만, 악성코드를 검출 및 분석 정확도가 떨어진다는 한계점이 있다[9]. 두 분석 방법 모두 소스코드, 바이너리로의 변환 후 특징을 추출한 후 분석하기 때문에 악성코드의 표준화와 모듈화로 대량으로 생산이 가능해진 최근의 다양한 악성코드를 식별 및 분석하기에는 시간적인 측면에서 많은 어려움이 있고, 식별하여 분석하는 과정에서 보안 취약점 분석 관련 전문 인력을 투입해야 하는 인력적인 한계점이 있다[10,11].

이에 본 연구에서는 소스코드 분석 방법에서의 분석 및 특징 추출의 시간이 오래 소요되는 한계점을 극복하고, 분석 결과의 정확도를 높이기 위해 딥러닝을 기반으로 한 소스코드 취약점 자동 패치 기법을 설계하고, 이를 시스템으로 구현한다.

본 논문의 구성은 1장은 서론으로서 연구 배경과 목적, 연구 내용 범위 및 논문의 구성에 관하여 서술한다. 2장에서는 본 연구를 통해 제안하는 딥러닝 기반의 소스코드 취약점 자동 패치 기법과 이를 이용한 자동패치 시스템의 설계를 기술한다. 3장에서는 딥러닝 기반 소스코드 취약점 자동 패치 시스템을 구현한 내용에 관하여 상세히 기술한다. 4장에서는 구현된 딥러닝 기반 소스코드 취약점 자동 패치 시스템의 성능을 실험 및 평가함으로써 제안한 딥러닝 기반 소스코드 취약점 자동 패치 시스템의 성능을 검증한다. 마지막 5장은 결론으로서 본문의 결과를 기술하고 향후 연구에 관하여 서술한다.

2. 딥러닝 기반 소스코드 취약점 자동 패치 시스템 설계

2.1 시스템 구성

기존의 소스코드 취약점 점검도구는 취약점만 점검하는 한계를 가지며 개발자가 수동으로 취약점을 제거하도록 가이드를 제공하는 수준에 그치지만 본 연구를 통해 개발되는 시스템은 가장 공격을 많이 받은 대상인 자바(Java) 언어로 만들어진 프로그램을 대상으로 딥러닝을 통한 소스코드 취약점을 자동으로 제거하는 시스템이다. 시스템의 구성은 Fig. 1과 같다.

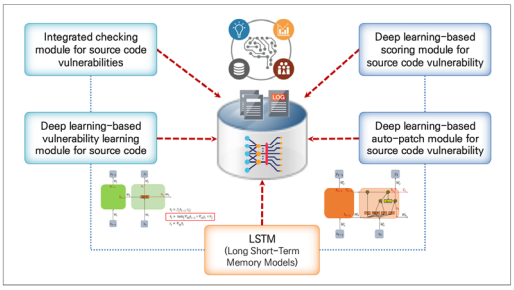

Fig. 1. System diagram

본 연구에서 제안하는 딥러닝 기반 소스코드 취약점 자동 패치 시스템은 다음의 4가지 모듈로 구성된다.

2.1.1 소스코드 취약점 통합 점검 모듈

소스코드 취약점 통합 점검 모듈은 취약점 진단 요청 처리와 진단 후 진단 데이터 처리, 진단 결과에 대한 처리, 정적 진단 도구의 제어 및 관리를 위한 시스템 모듈이다. 진단하고자 하는 소스는 민감한 사항이 될 수 있으므로 외부 유출 방지를 고려하여 보안 통신 프로그램을 사용하여 안전하게 파일을 전송한다.

2.1.2 딥러닝 소스코드 취약점 학습 모듈

딥러닝 소스코드 취약점 학습 모듈은 소스코드 취약점에 대한 데이터를 수집하여 딥러닝 알고리즘에 훈련 및 학습시키고, 해당 데이터셋을 기반으로 하여 검증을 위한 소스코드 입력 시, 취약점을 검출 후 자동으로 제거하기 위한 프로그래밍 학습 모듈이다. 이 모듈은 행안부 소스코드 취약점 소스코드를 데이터세트로 구성하여 소스코드의 취약점을 학습한다.

2.1.3 딥러닝 소스코드 취약점 점수화 모듈

딥러닝 소스코드 취약점 점수화 모듈은 취약점 점수화를 위한 취약점 평가는 아래 취약점 평가표를 기준으로 계산하며, 이진 분류(Binary classify)에서 결과를 통계적으로 분석하여 값을 측정하는 모듈이다. 이 모듈은 국제표준에 기초한 보안 위협 점수화 및 국제 침해 사고 대응협의체(Forum of Incident Response and Security Teams; FIRST)의 취약점 평가(Common Vulnerability Scoring System; CVSS) 국제표준을 적용하였다.

2.1.4 딥러닝 소스코드 취약점 자동 제거 모듈

딥러닝 소스코드 취약점 자동 제거 모듈은 소스코드 취약점 제거 히스토리 학습을 통해 취약점을 제거 모듈이다. 딥러닝 신경망을 통해 보안상 약점에 해당하는 패턴을 학습시켜 분류하고 결과를 분석하며, 취약점 제거를 위해 순서가 있는 데이터 학습에 적합한 LSTM(Long Short Term Memory) 모델로 구성한 오토인코더(Autoencoder)를 이용하여 취약점 제거 프로그래밍을 학습하고, 학습된 내용을 기반으로 소스코드 취약점을 자동으로 제거한다.

상기에 기술한 4가지 모듈에 의한 소스코드 취약점 자동 패치 처리 프로세스는 Fig. 2와 같다.

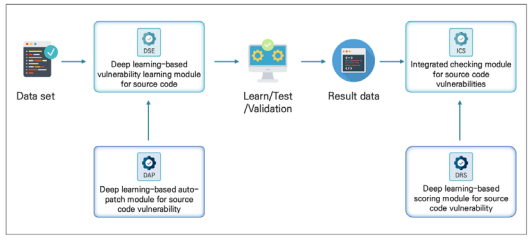

Fig. 2. System process

2.2 소스코드 취약점 통합 점검 모듈 설계

딥러닝 기반 소스코드 취약점 자동 패치 시스템의 모듈에 대한 설계 내용을 각 세부 기능(세부 모듈)별로 기술한다.

2.2.1 진단 도구 관리를 위한 시스템 관리

진단 도구 관리를 위한 시스템 관리 기능은 취약점 진단 요청 처리와 진단 후 진단 데이터 처리, 진단 결과에 대한 처리, 정적 진단 도구의 제어 및 관리를 위한 기능이다.

진단 도구 관리를 위한 기능 개발 시 이슈 사항으로는 웹을 통해 사용자 로컬 시스템을 제어할 수 없다는 것인데, 이러한 이슈 사항은 위저드 실행프로그램을 사용자가 내려받아 설치하여 실행하는 방식을 통하여 로컬 시스템의 소스 파일을 취합한 후 서버로 자료를 올리는 방식으로 해결할 수 있다. 진단하고자 하는 소스 파일은 민감한 사항이 될 수 있으므로 외부 유출 방지를 고려하여

보안 통신 프로그램을 사용하여 안전하게 파일을 전송한다. 진단 도구 관리 프로세스는 Fig. 3과 같다.

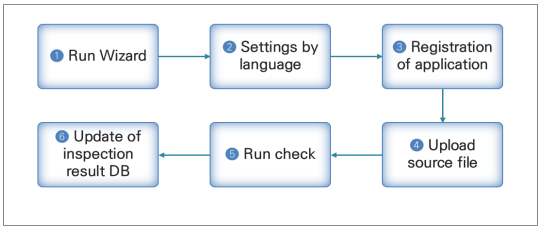

Fig. 3. Diagnostic tool management process

2.2.2 소스코드 취약점 진단 관리

① 권한 등록 및 관리

기존 등록되어있는 메뉴 정보와 사용자 그룹 정보를 매칭하여 등록하고, 권한 확인은 권한 매칭 테이블에서 사용자 그룹코드를 이용하여 각각의 권한을 조회한다. 권한관리 이벤트 시나리오는 Fig. 4와 같다.

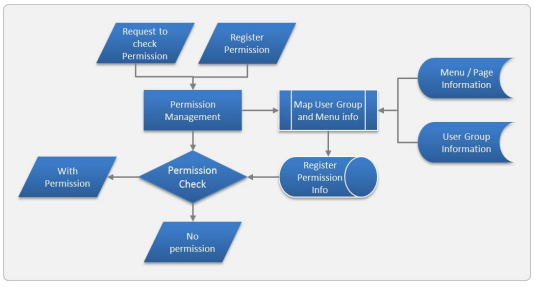

Fig. 4. Scenarios of rights management events

② 사용자 관리

사용자 또는 관리자가 계정(ID)을 생성, 사용자가 등록(패스워드 변경 후) 후 관리자가 사용자의 그룹과 부서 권한을 등록한다. 권한관리 이벤트 시나리오는 Fig. 5와 같다.

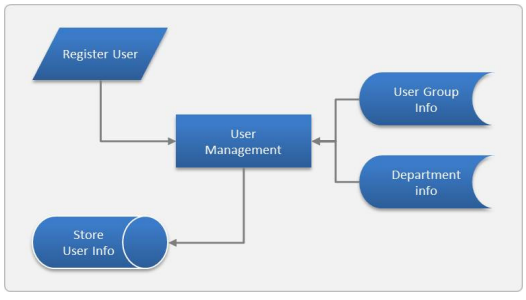

Fig. 5. Scenarios for user management events

2.2.3 소스코드 취약점 진단 관리

딥러닝 소스코드 취약점 학습 모듈은 소스코드 취약점에 대한 데이터를 수집하여 취약점을 자동으로 제거하기 위한 프로그래밍을 학습한다. 또한 행정안전부 소스코드 취약점 소스코드를 데이터세트로 구성하여 취약점 소스코드, 취약점 제거 코드를 학습한다.

딥러닝의 판정 및 검증을 위하여 딥러닝 엔진에서 소스코드 데이터를 학습한 후, 해당 결과로부터 생성된 모델에서 주어진 데이터셋에 대한 판정 정확률을 확인할 수 있다. 딥러닝기반 소스코드 취약점 학습 프로세스는 Fig. 6과 같다.

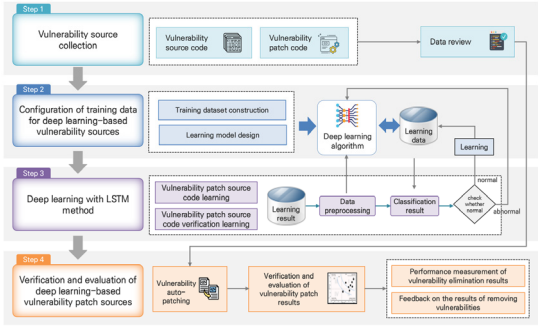

Fig. 6. Process of Deep learning-based source code vulnerability learning

2.2.4 딥러닝 소스코드 취약점 점수화 모듈 설계

딥러닝 소스코드 취약점 점수화 모듈(DRS)은 취약점 점수화를 위한 취약점 평가는 아래 취약점 평가표를 기준으로 계산하며, 이진 분류에서 결과를 통계적으로 분석하여 값을 측정할 수 한다. Fig. 7은 취약점 점수화를 위한 평가표이다.

	True Positive (abnormal)	True Negative (normal)
Predicted Positive (abnormal)	A (True Positive)	B (False Positive)
Predicted Negative (normal)	C (False Negative)	D (True Negative)

Fig. 7. Vulnerability Scorecard

평가 항목은 다음과 같이 정확도, 민감도, 특이도, 정확율 4가지이며, 다음 수식과 같다.

① 정확도(Accuracy) : 전체 결과 중에 정답으로 분류한 경우

$$Accuracy = \frac{(A+B)}{(A+B+C+D)} \quad (1)$$

② 민감도(Sensitivity) : 실제 비정상 행위 중에 예측된 비정상 행위의 비율

$$Sensitivity = \frac{(A)}{(A+C)} \frac{(A+B)}{(A+B+C+D)} \quad (2)$$

③ 특이도(Specificity) : 예측된 정상 행위 중 실제 정상 행위의 비율

$$Specificity = \frac{D}{(B+D)} \quad (3)$$

④ 정확률(Precision) : 예측된 비정상 행위 중 실제 비정상 행위의 비율

$$Precision = \frac{A}{(A+B)} \quad (4)$$

취약점 점수화 기준은 CVSS를 따르며, 취약점에 대한 위험도를 측정하기 위해서 평가 항목과 평가 값 및 수치를 분류하고, 각 평가 항목에 대한 평가 등급을 확인한 후 평가 점수 계산을 수행한다. Fig. 8은 CVSS 점수화 프로그램 화면이다.

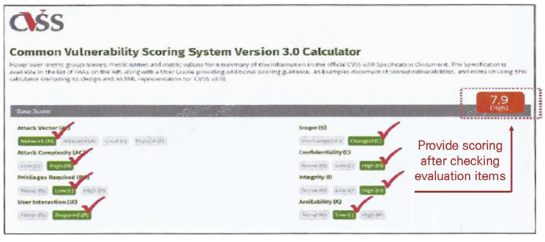

Fig. 8. CVSS scoring program

2.2.5 딥러닝 소스코드 취약점 자동 제거 모듈 설계

① 소스코드 취약점 제거 학습

소스코드 취약점 제거 학습 기능은 취약점 제거를 위해 순서가 있는 데이터 학습에 적합한 LSTM(Long Short Term Memory) 모델로 구성한 비지도학습 방법 Autoencoder를 이용하여 취약점 제거 프로그래밍을 학습한다. 소스코드 취약점이 있는 소스코드 패턴을 학습하여 특정 패턴을 설정하여 임계값 이상의 값이 나올 경우, 취약점 소스코드로 판단한다.

② LSTM(Long Short Term Memory) 모델 적용

순환신경망(RNN : Recurrent Neural Network)은 입력 시, 이전의 입력 결과가 다음의 입력으로 들어가고, 이후에 출력 값에 영향을 미쳐 입력순서에 따른 종속적인 성질이 나타나므로 소스코드 취약점 즉, 소프트웨어 실행 시 취약점이 활성화되는 흐름이나 기검출된 타 취약점과 이어져 연속된 행위가 발생하기도 하는 순서가 존재하는 데이터 학습에 유리하다.

LSTM은 이러한 RNN의 한계를 극복하기 위해 개선된 모델로, 뒤로 갈수록 이전 데이터의 내용 전달이 약해지는 장기 의존성(Long-Term Dependency)을 가진 RNN 모델에 비해 소스코드 취약점의 입력데이터 순서에 더욱 종속적인 성질을 가진다는 점에서 보완에 대한 장점을 가진다. Fig. 8은 LSTM(Long Short Term Memory) 모델의 구조를 나타낸 것이다[12,13].

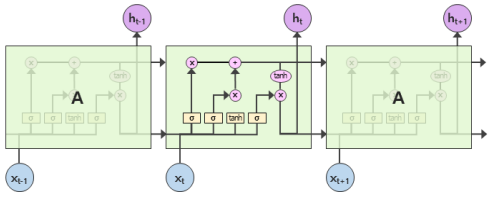

Fig. 9. Long Short Term Memory(LSTM) model structure

3. 딥러닝 기반 소스코드 취약점 자동 패치 시스템 구현

3.1 딥러닝 소스코드 취약점 학습 및 소스코드 취약점 자동 제거 모듈 구현

학습이 시작되면 딥러닝 소스코드 취약점 학습 메인화면 좌측의 Console 화면을 통해 데이터셋의 정보 및 학습 진행 상황이 Fig. 10과 같이 확인이 가능하다.

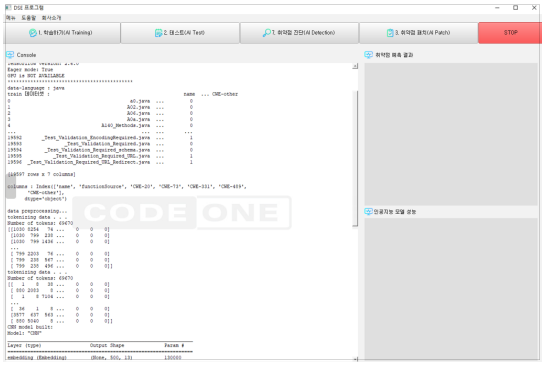

Fig. 10. Vulnerability learning progress

취약점 제거 완료 후 Fig. 11과 같이 완료 팝업을 통해 취약점 제거 전, 후의 코드를 확인하는 것이 가능하고 추가로 다시 검사하기 버튼을 클릭하면 제거된 후 취약점 테스트를 다시 진행한다.

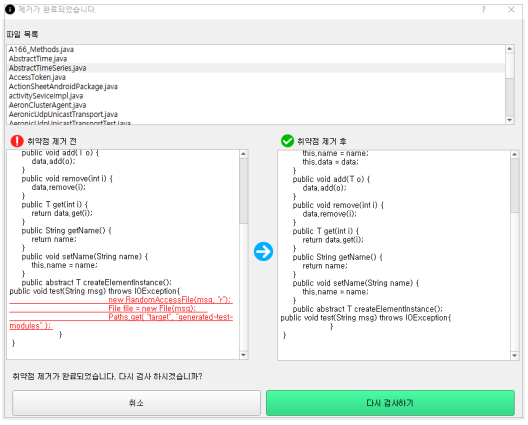

Fig. 11. Source code analysis screen before/after vulnerability removal

취약점 예측 완료 후 취약점 예측 결과 화면을 통해 취약점 진단 결과를 확인할 수 있다. 소스코드 취약점 자동 패치 정확도는 취약점 제거 이후 재진단한 결과를 통해 소스코드 취약점 제거에 대한 정확도를 확인할 수 있다.

3.2 소스 코드 취약점 통합 점검 시스템 및 소스코드 취약점 점수화 구현

딥러닝 소스코드 취약점 점검 후 결과로 생성된 결과 파일을 선택하면 Fig. 12와 같이 CVSS 점수를 확인할 수 있다.

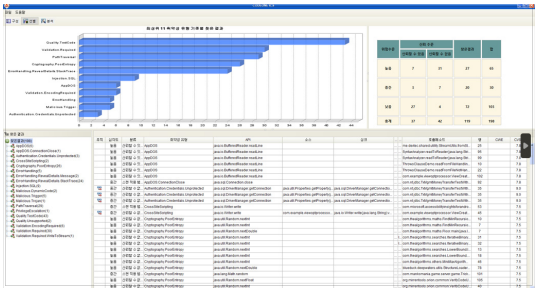

Fig. 12. CVSS score check screen

소스 코드 취약점 통합 점검 프로그램의 메인화면의 선별 탭을 클릭하여 확인할 수 있으며, ICS 프로그램에서 취약성 유형 목록과 CVSS 목록의 출력을 확인할 수 있다.

사전에 정의된 CVSS 점수 및 취약성 유형과 출력된 결과를 비교하여 일치 여부의 검증이 가능하다. 출력된 결과 중에서 임의로 선정한 취약점 유형과, 해당하는 유형의 CVSS (사전에 정의된 취약점 유형의 CVSS)와 비교도 가능하다.

4. 실험 평가

본 연구를 통해 개발된 딥러닝 기반 소스코드 취약점 자동 패치 시스템의 성능을 실험하기 위한 실험환경은 Table 1과 같이 CPU i5-10400, 2.90GHz, RAM 64GB, Windows 10 Pro 운영체제를 이용하였다.

Table 1. Test environment

Item	Experiment environment
CPU	Intel(R) Core(TM) i5-10400 CPU @ 2.90GHz
RAM	64.0GB
System	64-bit operating system, x64-based processor
OS	Windows 10 Pro

구현한 소스 코드 취약점 통합 점검 시스템을 통해 java, c. cpp 소스코드에 대한 취학점 학습을 진행하였으며, 취약점 학습 및 취약점 제거 학습 데이터셋의 실험 결과는 Table 2와 같다.

Table 2. Test Results of vulnerability learning and vulnerability elimination learning datasets

Test Items		Test results
Vulnerability training dataset	Java	19,597
	c, cpp	20,000
Vulnerability removal training dataset	Java	100
	c, cpp	100

취약점 학습 데이터셋은 Java 소스코드의 경우, 19,597개, c 및 cpp 소스코드의 경우, 20,000개를 이용하여 테스트하였으며, 취약점 제거 학습을 위한 데이터셋은 Java 소스코드의 경우, 100개, c 및 cpp 소스코드의 경우, 100개를 이용하여 테스트를 진행 하였다.

이후 개발된 딥러닝 기반 소스코드 취약점 자동 패치 시스템의 전반적인 성능을 평가하기 위해 소스코드 취약점 자동 패치 정확도, 소프트웨어 취약점 점수화, 발견된 취약점별 추적정보 존재 유무, 행렬형 보기의 분류 정확도, 취약점 분석에 따른 자동 진단율에 대한 실험을 진행하였으며, 그 결과는 Table 3과 같다.

Table 3. Performance test results

Test Items	Performance test results
Source Code Vulnerability Automatic Patch Accuracy	99.8%
Software vulnerability scoring	≥ 10 times
Existence of tracking information for each discovered vulnerability	≥ 5 instances
Classification accuracy of the matrix view	10 times out of 10
Automatic detection rate based on vulnerability analysis	96.5%

실험 결과, 소스코드 취약점 자동패치 정확도는 99.8%, 소프트웨어 취약점 점수화는 10개 이상 확인. 발견된 취약점별 추적정보 존재 유무는 5회 이상 확인되었으며, 행렬형 보기의 분류 정확도는 10회 중 10회 확인, 취약점 분석에 따른 자동 진단율은 96.5%로 나타났다.

따라서 본 연구를 통해 개발된 딥러닝 기반 소스코드 취약점 자동 패치 시스템의 성능은 매우 높게 나타난 것을 확인할 수 있었다. 또한 본 시스템을 통해 해당 소스코드 내에 있는 취약점을 한눈에 알아볼 수 있고, 취약점이 자동으로 패치되는 것은 기존 취약점 분석 시스템 대비 장점으로 평가할 수 있다.

5. 결론

본 연구에서는 딥러닝을 이용하여 소스코드 취약점 제거를 위한 소스코드 데이터셋을 학습시켜 자동화된 패치까지 제공함으로써 일관된 소스코드 취약점 제거를 가능하게 하는 모델을 제안하고, 자동으로 소스코드 취약점 제거 기능을 수행하는 시스템을 구현하고 이에 대한 성능평가를 수행하였다.

성능평가 결과, 취약점 학습 데이터셋은 Java 소스코드의 경우, 19,597개, c 및 cpp 소스코드의 경우, 20,000개를 이용하여 테스트하였으며, 취약점 제거 학습을 위한 데이터셋은 Java 소스코드의 경우, 100개, c 및 cpp 소스코드의 경우, 100개로 테스트를 진행하였다. 또한 실험 결과, 소스코드 취약점 자동 패치 정확도는 99.8%, 소프트웨어 취약점 점수화는 10회 이상. 발견된 취약점별 추적정보 존재 유무는 5회 이상 확인되었으며, 행렬형 보기의 분류 정확도는 10회 중 10회 확인, 취약점 분석에 따른 자동 진단율은 96.5%로 나타났다.

따라서 본 연구를 통해 개발된 딥러닝 기반 소스코드 취약점 자동 패치 시스템의 성능은 매우 높게 나타난 것을 확인할 수 있었으며, 본 시스템을 통해 해당 소스코드 내에 있는 취약점을 한눈에 알아볼 수 있고, 취약점이 자동으로 패치되는 것은 기존 취약점 분석 시스템 대비 장점으로 평가할 수 있다.

소프트웨어 보안을 위해서는 소스코드 취약점 진단 시스템을 통한 취약점 검출과 제거가 되어야 하지만, 기존 소스코드 취약점 진단 시스템은 취약점 검출은 취약점에 대하여 단순한 진단만 수행할 뿐 취약점 부분을 자동으로 제거하는 기능은 없는 상황이다. 본 연구를 통해 개발된 딥러닝 기반 소스코드 취약점 자동 패치 시스템은 소스코드 취약점에 대해 국제표준에 기반하여 CWE 점수화를 제공함으로써 Java 소스코드 취약점의 위험 순위를 제공뿐만 아니라 딥러닝을 통해 소스코드의 취약점을 검출하고, 자동으로 제거까지 할 수 있다는 점에서 연구의 의의가 있다고 할 수 있다.

향후 추가적인 연구를 통해 소스코드 취약점 점검 결과 및 자동 취약점 제거 결과에 따라 프로젝트 수행 과정에 대한 시각화 화면을 제공하여 일관성 있는 소스코드 취약점 제거 프로젝트 관리 기능을 개발할 계획이다. 또한 이 시스템의 효과를 최대화하기 위하여 앞으로의 연구에서는 다양한 프로그래밍 언어에 대한 적용성을 확장하고 동시에 더 다양한 취약점 패턴을 학습할 수 있도록 데이터셋을 확장하여 실시간 취약점 검출 및 패치 기능도 함께 구현할 계획이다.

References

[1] D. Y. Jeong, Types and Prospects of Cyber Attacks in Various Cases, Ministry of Science and ICT, 2023. https://blog.naver.com/with_msip/222993851861

[2] G. E. Kim, Experts on hacking of Samsung Electronics "high possibility of secondary damage by exploiting source code", TechM, March 2022.. https://www.techm.kr/news/articleView.html?idxno=94995

[3] T. Marjanov, I. Pashchenko, F. Massacci, "Machine

Learning for Source Code Vulnerability Detection: What Works and What Isn't There Yet", *IEEE Security & Privacy*, Vol.20, pp.60-76, 2022.
DOI: https://doi.org/10.1109/msec.2022.3176058

[4] G. Y. Moon,Why does everyone become tolerant of source code leak accidents, Boannews, Dec. 2021. https://www.boannews.com/media/view.asp?idx=103393

[5] Verizon, Data Breach Investigations Report(DBIR), 2022. https://www.verizon.com/business/resources/reports/dbir/2022/master-guide

[6] CISCO SECURE, Cybersecurity for SMBs: Asia Pacific Businesses Prepare for Digital Defense, 2021. https://www.cisco.com/c/dam/global/ko_kr/products/security/meet-max-report-2021/assets/data/Infographics.pdf

[7] S. W. Lee, *A Study on Malware Authorship Attribution Model through Automated Analysis*, Master's thesis, Jeju University, Jeju, Korea, 2022.

[8] E. Stamatatos, "A survey of modern authorship attribution methods", *Journal of the American Society for information Science and Technology*, Vol.60, No.3, pp.538-556, 2009.
DOI: https://doi.org/10.1002/asi.21001

[9] W. A. Arbaugh, W. L. Fithen, J. McHugh, "Windows of Vulnerability: A Case Study Analysis", *Computer, IEEE*, Vol.33, No.12, pp.55-59, 2000.
DOI: https://doi.org/10.1109/2.889093

[10] M. Campbell, "Beyond Zero Trust: Trust is a Vulnerability", *Computer, IEEE*, Vol.53, No.10, pp.110-113, 2020.

[11] T. Alladi, V. Chamola, B. Sikdar, K. R Choo, "Consumer IoT: Security Vulnerability Case Studies and Solutions", *IEEE Consumer Electronics Magazine*, Vol.9, No 2, pp.17-25, 2020.

[12] C. H. Cho, *Research on an efficient insider threat detection based on LSTM autoencoder using user attribute information*, Doctoral dissertation, Sungkyunkwan University, Suwon, Korea, 2018.

[13] S. Hochreiter, J. Schmidhuber, "Long short-term memory", Neural Computation, Vol.9, No.8, pp.1735-1780, 1997.
DOI: https://doi.org/10.1162/neco.1997.9.8.1735

이 관 목(Kwan-Mok Lee) [정회원]

- 2006년 8월 ~ 현재 : ㈜코드원 대표이사
- 2021년 2월 : 서울미디어대학원대 뉴미디어학부 (미디어경영학 석사)
- 2021년 2월 ~ 현재 : 극동대학교 일반대학원 인공지능보안학과 (박사과정)

〈관심분야〉
정보보호, 정보통신

이 상 도(Sang-Do Lee) [정회원]

- 2005년 2월 : 성균관대 정보통신대학원 정보보호학과 (석사)
- 2018년 2월 : 숭실대 컴퓨터학과 (박사)
- 2021년 8월 ~ 현재 : 육군사관학교 컴퓨터학과 조교수

〈관심분야〉
정보보호, 정보통신, 사이버전

이 용 준(Young-Joon Lee) [종신회원]

- 2005년 2월 : 숭실대학교 컴퓨터학과 박사
- 2010년 2월 ~ 2016년 3월 : 한국인터넷진흥원 수석연구위원
- 2016년 4월 ~ 2020년 3월 : 국방보안연구소 연구관
- 2021년 4월 ~ 현재 : 극동대학교 해킹보안학과 교수

〈관심분야〉
해킹보안, 국방보안, 인공지능보안