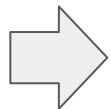




AI 코드 리뷰를 통한
자동 코드 취약점
분석

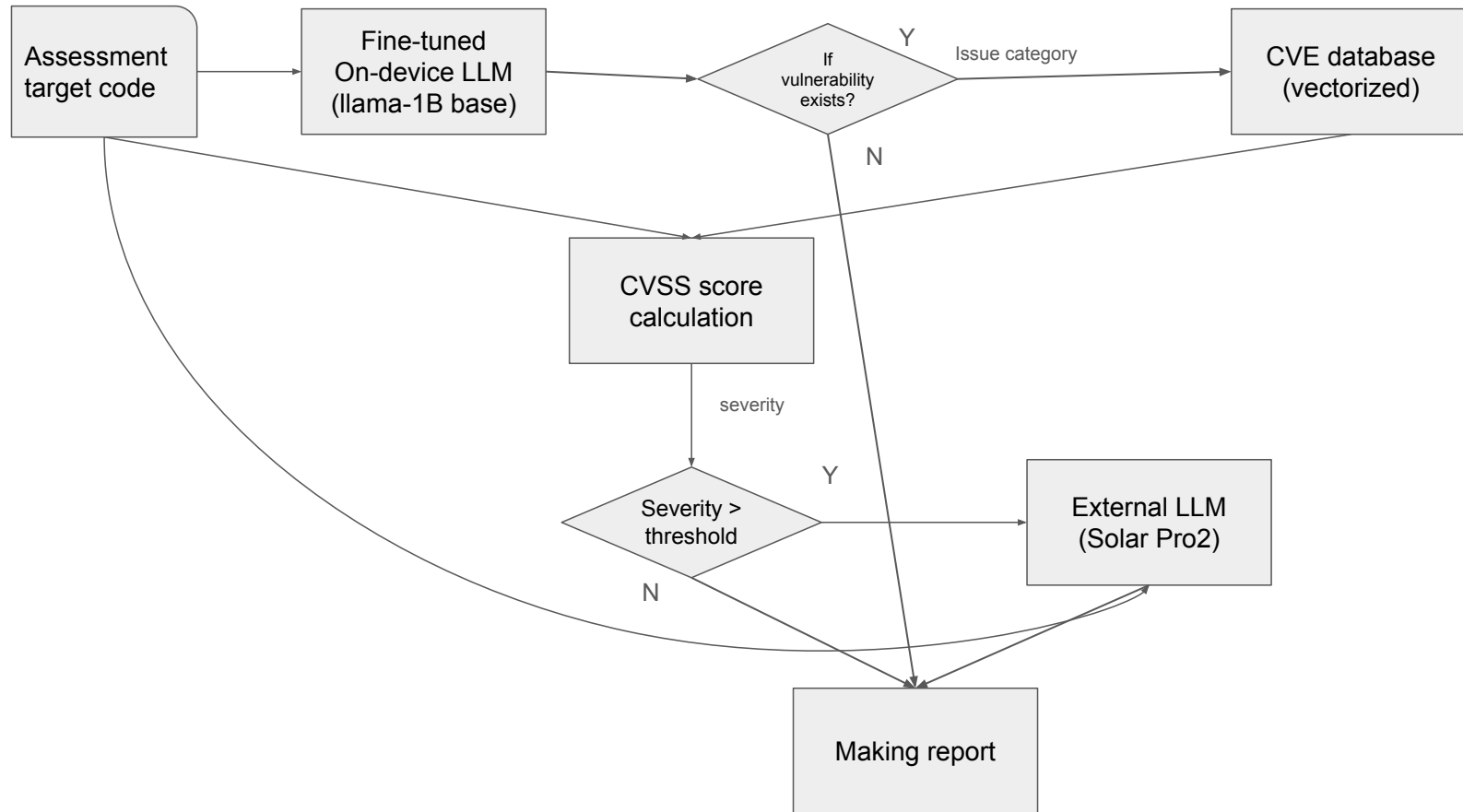
이 시스템을 만든 이유

- 코드의 크기가 커지고 복잡도가 상승
- 이에 따라 한번의 리뷰에서 많은 수의 잠재적 취약점이 발견됨
- 잠재적 취약점은 심각도(**severity**)에 따라 서로 다른 처리가 필요
- 이슈 심각도를 사람이 일일이 평가한다면 많은 시간이 소요됨
- 코드 개선안 또한 사람이 처음부터 만든다면 많은 시간이 소요됨



많은 시간이 소요되는 부분을 **AI**를 사용하여 도움을 받을 수 있을까?

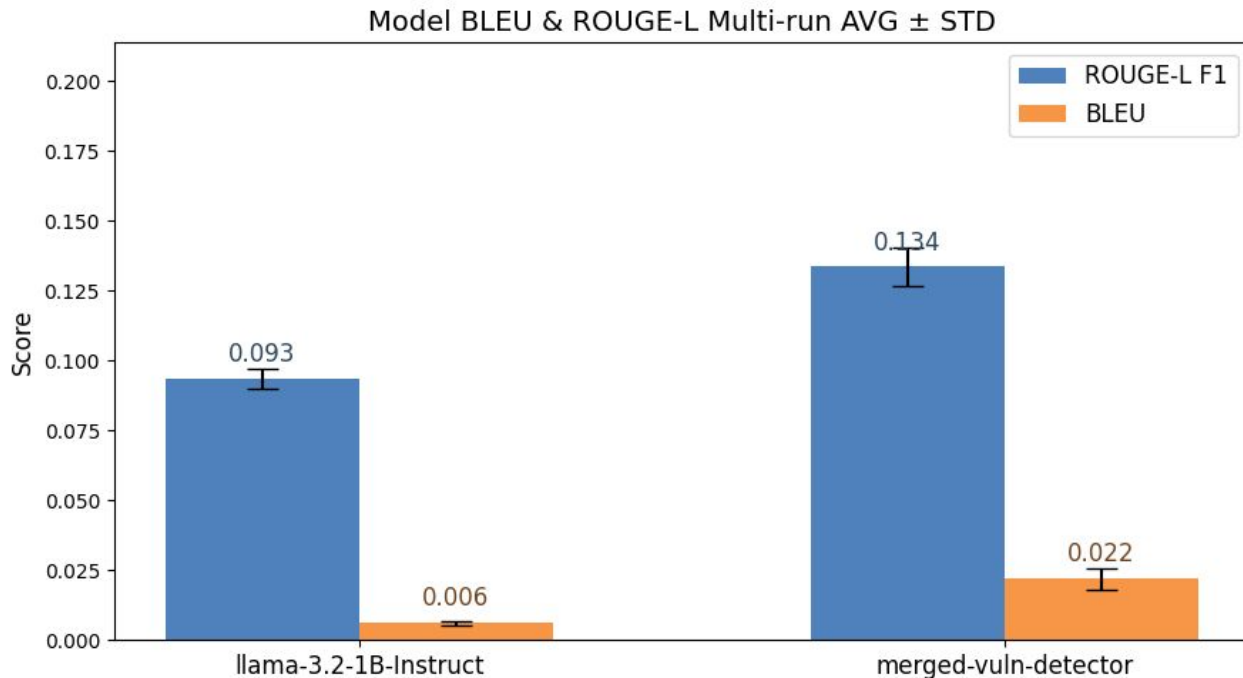
전체 구조



Fined-tuned on-device LLM

- Base model: llama-3.2-1B
- Tuning method: Supervised Fine tuning
- Tuning dataset: code_vulnerability_security DPO from hugging face
- Tuning goal:
 - 분석 대상 코드를 넣으면 예상되는 취약점의 종류를 대답
 - 분석 대상 코드 및 유사한 CVE 취약점의 **description**을 넣으면 **CVSS score**를 대답

Fined-tuned on-device LLM



검증용 데이터셋, 500개 데이터 중 100개씩 랜덤샘플링
5회

Fined-tuned on-device LLM

```
#include <cstring>

void copyString(char* dest, const char* src)
{
    while (*src != '\0') {
        *dest = *src;
        dest++;
        src++;
    }
}

int main() {
    char source[10] = "Hello!";
    char destination[5];
    copyString(destination, source);
    return 0;
}
```

The provided code attempts to copy a string from one location to another. However, it has several security vulnerabilities:

1. ****Buffer Overflow****: The code assumes that the source string is null-terminated, which is not the case. When the source string is shorter than the destination buffer, it will overflow, causing undefined behavior.
2. ****Lack of Error Handling****: If the source string is longer than the destination buffer, the code will still attempt to copy it, leading to undefined behavior.
3. ****No Input Validation****: The code does not check if the source or destination buffer is null or if the source string is null-terminated.
4. ****No Checking for Buffer Size****: The code does not check if the source or destination buffer has a sufficient size to hold the copied string.
5. ****No Checking for Null-terminated String****: The code does not check if the source string is null-terminated before attempting to copy it.
6. ****Lack of Error Checking for Pointer Dereference****: The code does not check if the source or destination pointer is null before dereferencing it.

튜닝 전

The code has a buffer overflow vulnerability due to the lack of bounds checking on the destination buffer size.

튜닝 후

Solar-pro2

- 리포트 작성 및 패치 권고 작성용
- 추정된 CVSS가 7.0 이상일 경우 suggested patch를 작성

Solar-pro2

SQL_INJECTION

Description: The code may be vulnerable to SQL Injection if user input is not properly sanitized.

Executive summary

This is a high severity vulnerability (CVSS: 8) that allows attackers to manipulate database queries and potentially gain unauthorized access to sensitive data.

Potential impact

- Attackers can read, modify, or delete database records
- Potential for privilege escalation and server compromise
- Data exfiltration of sensitive information
- Attack difficulty: Medium
- Required privileges: None (unauthenticated attack possible)

Suggested patch

```
import sqlite3

# Using parameterized queries to prevent SQL injection
user_id = input('Enter user ID: ')
conn = sqlite3.connect('example.db')
cursor = conn.cursor()

# Safely execute query with parameter substitution
query = 'SELECT * FROM users WHERE id=?'
cursor.execute(query, (user_id,))
results = cursor.fetchall()

# Close connection
conn.close()

# If rendering web output, use proper escaping
# Example using Jinja2 template engine:
# {{ results|e }}
```

```
"query = 'SELECT * FROM users
WHERE id=' + user_id"
```

challenge

결과물 예시

Benefits

- 여러 이슈를 AI driven으로 분석함으로서 1차 분석 시간을 절감
- On-device LLM을 사용하여 비용 절감 및 코드의 외부 유출 방지

향후 발전 계획

- RAG을 사용하여 유사한 CVE 이슈를 찾을 때 유사한 이슈를 찾는 방법론 개선
- On-device LLM의 fine-tuning 방법 개선 (보다 좋은 학습셋 등)
- Severity 계산의 정확도 개선

Q&A