**VulnNotti: 오픈소스를 활용한 IoT 디바이스**

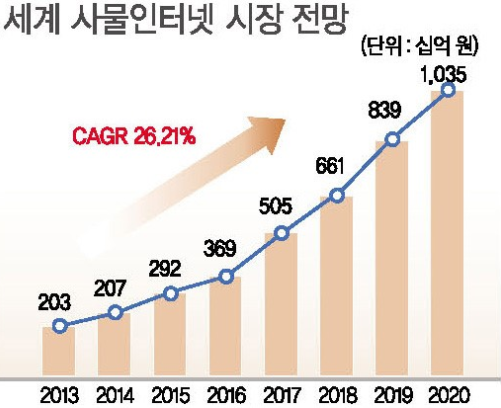
**취약점 분석 관리 도구**

2012104082 노현종

2012104108 이경수

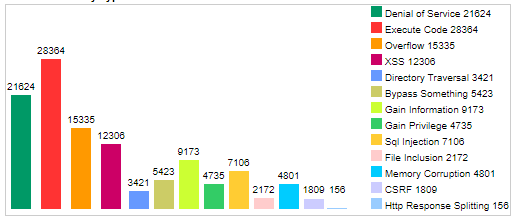
2012104115 이지윤

**개 요**

IoT (Internet of Things)는 가전제품, 모바일 장비, 웨어러블 컴퓨터 등 다양한 임베디드 시스템에 센서와 통신기능을 내장하여 인터넷에 연결하는 기술을 말한다. 인터넷으로 연결된 사물들이 인간의 개입 없이 데이터를 주고받아 스스로 분석하고 학습한 정보를 사용자에게 제공하거나 사용자가 이를 조정할 수 있다.

[그림 1] 세계 사물인터넷 시장 전망

IoT의 발전과 함께 IoT보안에 대한 중요성이 커지고 있다. IoT취약점을 이용한 공격들은 사용자들의 실생활과 밀접한 관련을 가지고 있기 때문에 심각한 사생활 침해를 유발할 수 있다. 공격자는 자동화된 공격 툴을 통해 취약한 디바이스에 접속하여 전력 사용량, 영상 정보 등을 탈취 할 수 있다. 이는 사용자의 생활패턴을 충분히 파악할 수 있는 데이터로 외부에 노출되었을 시 사생활 침해는 물론 2차적 피해를 야기할 수 있다.

다양한 IoT기기의 출시와 서비스의 확산에 따라 오픈소스를 활용한 IoT도 많아졌다. 낮은 가격과 높은 호환성으로 많은 개발자들이 오픈소스를 사용하고 있다. 하지만 사용하는 오픈소스마다 보안을 신경쓰기가 어려워 다수의 보안 사고가 증가하고 있고, 규모와 피해 정도도 점점 커지고 있다. 최근에는 개인 사생활 침해와 같은 사회적 역기능 및 심각한 우려를 초래하여 새로운 기술, 제품 및 IoT 산업 확대에 걸림돌이 되고 있다.

[그림 2] 취약점 발생 원인 - 실행코드

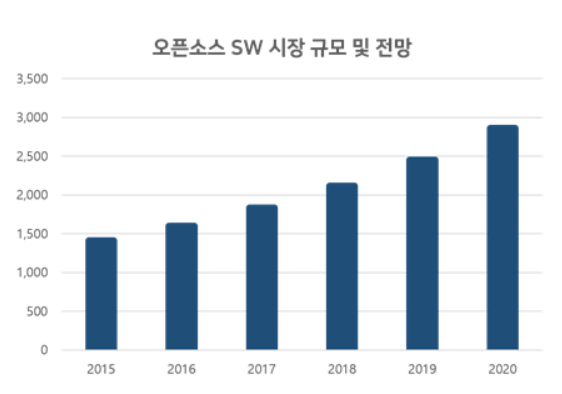
이를 해결하기 위해서는 오픈소스를 사용한 IoT디바이스의 취약점을 분석해주는 도구가 필요하다. 하지만 개발자가 개발단계에서 오픈소스의 취약점을 직접 찾고 관리하기는 쉽지 않다. ‘VulnNotti’ 프로젝트는 오픈소스를 사용한 IoT 디바이스 취약점 분석 도구로써 개발자가 개발에만 집중을 할 수 있도록 많은 기능을 제공한다. 예를 들어 개발자는 Github에 소스코드 변경 사항을 Commit만 하면 서버가 자동으로 인식하여 Commit된 코드 중 취약점을 찾아낸다. 제공하는 서비스는 소스코드 단계, 바이너리 단계, 동적 서비스 단계 분석으로 나누어서 취약점 분석을 해주고 즉각적인 알림을 제공한다. 또한 추후에 검사했던 오픈소스에서 새로운 취약점이 발견되면 사용자에게 알림을 준다.

‘VulnNotti’ 프로젝트는 기존 개발자들이 개발단계에서 소스코드의 취약점을 직접 하나하나 분석을 해야 하는 불편함을 줄이고, 개발 후 오픈소스에 취약점이 발견 되었을 때 패치를 잘 하지 않아서 생기는 보안사고를 예방할 수 있도록 개발한다

1. **서론** 
   1. **연구 배경**

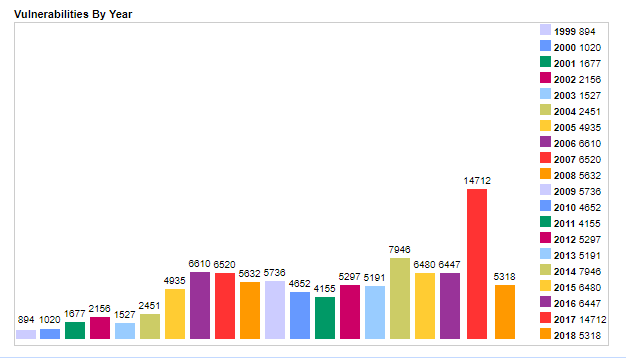
2019년 전 세계 IoT 시장 규모는 13 억 달러로 성장하며, 2020년까지 IoT Hardware는 300억 개에 달할 것으로 예상된다. 이에 따라 올해 기업 및 조직들의 약 67%가 사물인터넷을 사용할 계획을 지닌 것으로 나타났다.

모든 사물이 해킹의 대상이 될 수 있으므로 IoT 보안의 중요성이 커지고 있다. 전 세계적으로 IoT 산업의 시장 확대에 따라 보안에 대한 관심과 투자가 이루어지고 있으며, 시장 조사기관인 가트너에 따르면, 2018년 기업 IoT 보안 지출 규모가 2017년 12억 달러에서 28% 증가한 15억 달러에 이를 것으로 전망됐다. 또한 전세계 약 20%의 기업이 지난 3년 이내 최소 한 차례 이상의 IoT 기반 공격을 경험한 것으로 드러났다.

현재 IoT는기존 기기-사람-서비스 간의 보안에서, 사람의 개입 없이 기기와 기기 또는 기기와 서비스로 연결되어 데이터의 처리와 제어, 서비스 제공이 이루어지고 있다. 사용자의 인지와 무관하게 보안에 대한 공격이 발생할 수 있는 것이다. 그러므로 근본적으로 기기와 서비스의 설계 단계에서부터 보안을 고려해야 한다.

[그림 3] 오픈소스 SW 시장 규모 전망

최근 Open Source Hardware에 대한 관심이 높아지고 있다. 오픈소스를 이용하면 IoT 구현기술에 대해 모르더라도, 쉽게 하드웨어를 설계할 수 있어서, 전공자가 아니더라도 IoT를 이용한 제품을 만들 수 있다. 이런 장점으로 오픈소스를 활용한 IoT 하드웨어가 많아지고 있다.

하지만 동시에 오픈소스 보안에 대해서도 생각할 필요가 있다. 많은 프로젝트에서 오픈소스가 어디서 어떻게 사용되고 있는지 관리가 되지 않아 취약한 버전의 오픈소스가 패치되지 않을 수 있다. 또, 오픈소스 사용에 대한 정책이 존재하지 않기 때문에 업데이트를 시행한 이후에도 보안문제가 해결되지 않는 경우도 있다

[그림 4] 매년 증가하는 CVE Vulnerabilities

Cve details 2017에 따르면 [그림 4]와 같이 매년 CVE Vulnerabilities가 증가하는 것을 알 수 있다. 또한, 소나타임 (Sonatype)의 2016 소프트웨어 공급망 연구 (2016 software supply chain study)에 따르면, 써드파티 (third party) 구성요소가 엔터프라이즈 자바 애플리케이션 코드의 80~90%를 차지하고 있고 기업이 다운로드 받는 구성요소 6개 가운데 1개에 보안 취약점이 있다. 하트블리드 (Heartbleed) 버그가 발견된 지 2년이 지났지만, 시스코 보안 연구소 (Cisco Security Research)가 2015년 테스트 한 오픈SSL (OpenSSL) 버전 가운데 절반 이상이 취약점이 발견된 이후에도 패치를 하지 않아 여전히 취약한 것으로 드러났다.

* 1. **연구목표**

최근 조사에 따르면 개발자들이 사용하고 있는 오픈소스의 취약점에 대해 관리를 못하는 기업이 많다. 모르는 오픈소스 프로젝트를 대상으로 보안을 평가하고, 패치를 관리하기란 매우 어렵다. ‘VulnNotti’프로젝트는 보안 지식이 없는 개발자들이 핵심 비즈니스에 집중 할 수 있도록 쉽게 소스코드를 검증할 수 있도록 해준다.

2016년 널리 쓰이는 PHP 이메일 전송 라이브러리 중 하나인 PHP메일러 (PHPMailer)에서 ‘원격 코드 실행’ 보안 취약점이 발견됐다. 웹사이트에 수백만 개가 위험에 노출 될 수 있는 코드였다. 이후 이를 수정한 PHP메일러 5.2.18버전이 공개됐다. 하지만 PHP메일러를 사용하는 웹사이트가 대부분 패치를 하지 않은 ‘제로데이’상태였다. 보안 패치가 배포된 상황이었지만 패치를 설치하지 않은 파크앤플라이 (Park’n Fly)와 원스톱파킹 (OneStopParking.com)은 보안 공격을 당하였다. 이처럼 관련 보안 패치가 공격이 일어나기 전에 배포되었지만 모든 관련 기기 및 시스템에 설치되지 않아 일어날 수 있는 보안 공격을 막기 위해 ‘VulnNotti’ 프로젝트는 한번 검사한 소스코드를 저장하고, 이후에 취약점이 발견 될 시 사용자에게 알림을 주는 서비스를 제공한다.

1. **관련 연구**
   1. **관련 연구 1**

관련 연구로는 ‘VUDDY: A scalable approach for Vulnerable Code Clone Discovery’가 있다.[2] 이 논문은 오픈소스의 취약점을 탐지하기 위해 코드 클로닝 기법을 사용한다. VUDDY는 취약한 복제를 찾아내기 위해 전처리 과정을 거친다. 취약점이 발견된 소스코드의 함수를 추출한 후 정규화와 추상화를 진행한다. 그 다음 Fingerprint를 생성하고 추출된 Fingerprint를 이용해서 비교를 하기 위한 데이터베이스를 구축한다.

* 1. **관련 연구 2**

또 다른 관련 연구로는 왕정민의 숭실대학교 석사학위 논문 ‘오픈소스 위험도 평가 방법을 기반으로 한 오픈 소스 수집 시스템’이 있다.[1] 이 논문에서는 방대한 오픈소스를 효율적으로 수집하는 방법을 제안한다. 이 시스템을 구현하기 위해 오픈소스의 메타 정보를 이용하여 주요 키워드 셋 (Set)을 구성하고 이를 바탕으로 위험도를 평가하여 오픈소스의 수집 우선순위를 도출하는 알고리즘을 제안하였다. 또한 취약점 분석이 이루어질 때 마다 기존의 기반 데이터가 업데이트되어 더욱 명확하고 구체화된 위험성 키워드 셋 (Set)을 도출할 수 있는 시스템 프로세스를 제안하였다. 제안한 시스템의 성능을 검증하기 위해 프로토타입을 제작하여 오픈소스의 심각도 (Severity), 위협도 (Threat), 우선순위 (Priority)를 평가하여 위험성 스코어를 산출하였으며 이를 통해 수집 효율을 검증하였다.

이 방법으로 오픈소스의 메타정보(카테고리 분류, 언어, 라이선스, 오픈소스 제작자의 설명 등)를 자동으로 수집한다. ‘VulnNotti’ 프로젝트는 취약점 정보(위험등급, CVE)와 패치 이전 코드정보와 패치 후 코드정보를 수집한다. 웹에서 사용자가 입력한 소스코드와 비교할 때 필요 없는 정보들은 수집하지 않는다. 또한 패치 전, 후의 코드의 정보가 담겨 있어서 사용자가 입력한 소스코드에 패치 전의 소스코드가 있어도 Matching 시킬 수 있다.

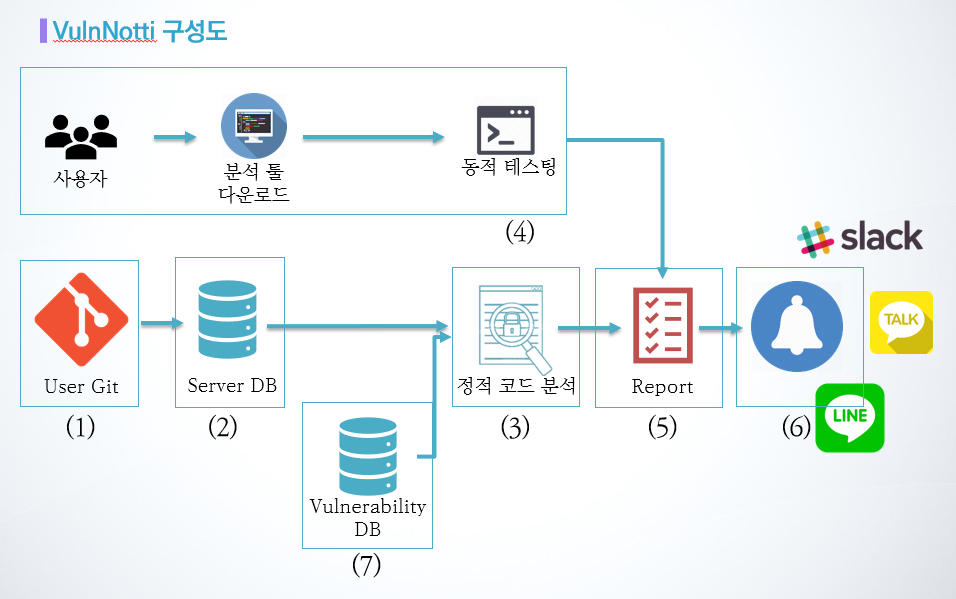
* 1. **관련 연구의 문제점**

‘VUDDY: A scalable approach for Vulnerable Code Clone Discovery’는 취약점을 검사하기 위해 함수단위로 테스트를 진행하기 때문에 함수에 변화가 생기면 취약점을 검출하지 못한다는 문제점이 있다. 예를 들어 for문을 while문으로 바꾸거나 다른 코드가 추가하거나 제거 시 해당 함수에 대한 해시값이 바뀌기 때문에 취약점을 검출할 수 없다. ‘VulnNotti’는 이런 단점을 보완할 수 있도록 함수단위의 비교가 아닌 함수를 여러 블록으로 쪼갠 후 블록단위비교를 통해 취약점을 탐지한다.

1. **프로젝트** 
   1. **기존 연구와 차이점 및 해결방안**

기존 연구에서는 코드 구조가 약간만 변경되어도 취약점을 탐지할 수 없었다. CVE 패치 정보를 이용하여 크리티컬 변수를 선정하고 그 변수는 해당 취약점에 중요한 역할을 하기에 그 변수에 영향을 주는 코드 블록을 함수 내에서 크리티컬 블록으로 선정하여 코드 구조 변경에 유연하면서 취약점 탐지 성능은 더 높아질 수 있게 하였다.

* 1. **프로젝트 내용**



[그림 5] ‘VulnNotti’ 전체 구성도

‘VulnNotti’는 크게 사용자 입력 부분(1), 서버 DB(2), 정적 코드 분석(3), 동적 테스팅 분석(4), 분석 결과 시각화(5), 실시간 알림 서비스(6), 취약점 DB(7)로 구성된다.[3]

**사용자 입력(1)**

입력으로 사용자의 Github Repository 주소를 받는다. 또는 소스코드 압축 파일을 ‘VulnNotti’ 웹사이트에 직접 업로드 할 수 있다.

여기에 등록된 사용자 소스코드는 주기적 검사 및 관리를 위해서 Server DB(2)에 저장된다.

**Server DB(2)**

사용자의 소스코드가 서버에 저장되는 DB이다. 이 곳에선 최초 업로드 시 취약점 풀 스캐닝 프로세스를 실행하고 이후엔 git push를 받을 때마다 취약점 부분 스캐닝을 진행한다.

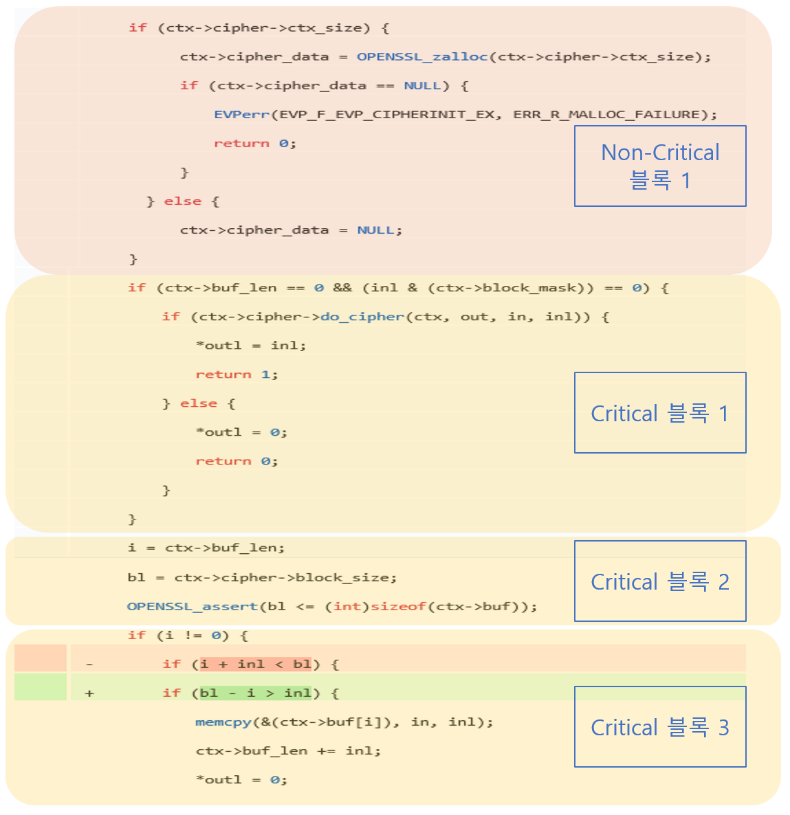
**정적 코드 분석(3)**

Server DB(2)에서 업로드 된 소스 코드를 대상으로 정적 취약점 분석 프로세스를 실행한다.

정적 코드 분석(3)은 취약한 코드 클로닝 검사를 통해 취약한 코드를 찾아낸다. 기존 코드 클론 탐지는 라이선스 위반 여부를 찾는 목적이었기에 for문을 while문으로 바꾸는 것과 같은 문장 의미 구조를 파악할 수 없었다. 하지만 본 논문에서는 코드 간 Syntax 차이를 구분할 방법과 Vulnerable Code를 탐지하기 위해 최적화된 코드 클로닝 기법을 제시한다.

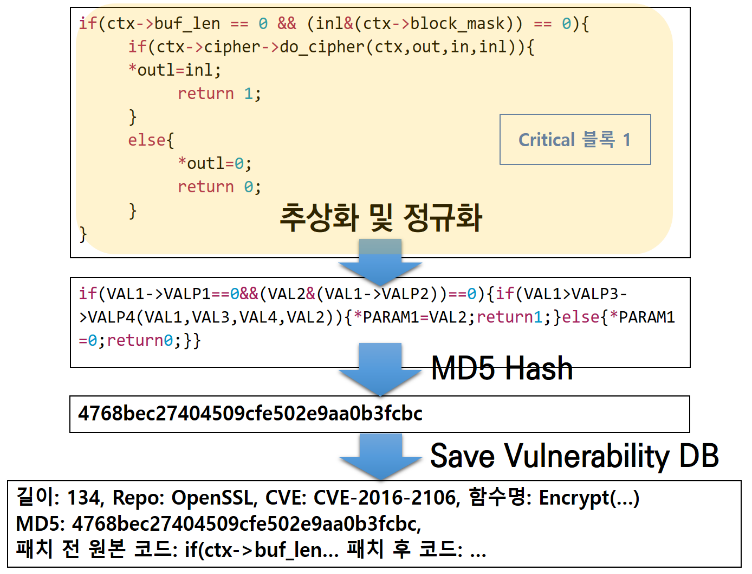
취약점 분석을 위한 코드 클로닝 기법을 위해선 먼저 검사할 소스코드와 취약점이 발견된 소스코드가 필요하다. 검사할 소스코드는 Server DB(2)에 저장 되어 있고 취약점 소스코드는 Server의 Vulnerability DB(7)에 저장된다. Vulnerability DB의 Vulnerable Code 데이터는 Git의 Commit 구조를 이용하여 수집한다. IoT에서 자주 사용되는 라이브러리를 Git 웹호스팅 회사인 Github에서 Repository를 선정하여 수집한다. Git은 코드 변경 내역인 Commit을 가지고 있기 때문에 각 Repository에서 CVE 코드로 패치 한 내역이 있는지 검색한 후 수집된 Commit을 대상으로 코드 수집을 한다. CVE 패치 Commit에서 패치 전 코드 함수는 취약한 함수이므로 Server DB에서 이 함수가 발견 되면 입력한 소스코드가 취약한 코드를 가지고 있는 것으로 판단한다. 하지만 사용자가 코드를 추가하거나 Syntax적인 변경을 한다면 취약점이 있는데도 탐지할 수 없다는 문제점이 있다.

이런 문제를 해결하기 위해 패치 되어 변경된 코드 블록과 연관된 변수는 Critical할 확률이 높다는 것에서 새로운 방법을 착안하였다. 즉, 패치 된 부분 함수를 전부 복사하여 matching을 하는 것이 아닌, 패치 전과 패치 후 코드의 변화를 이용하여 해당 함수 내부에서 취약점이 패치 된 라인을 찾고 그 라인의 블록을 정한 뒤 블록 내부의 변수 목록을 만든다. 이 변수 목록이 취약점 패치 변경이 일어난 부분 라인보다 위에서 할당되거나 쓰였다면 그 코드 블록은 Critical한 블록으로 남기고 나머지 블록은 Blind 처리한다. Critical한 부분을 만날 때 마다 위 처리 과정을 반복하면서 영향을 줄 수 있는 부분들을 모두 Propagation 한다. 이렇게 하면 취약한 코드와 연관성이 낮은 블록은 Blind 처리 되기에 탐지 효율을 높일 수 있다.



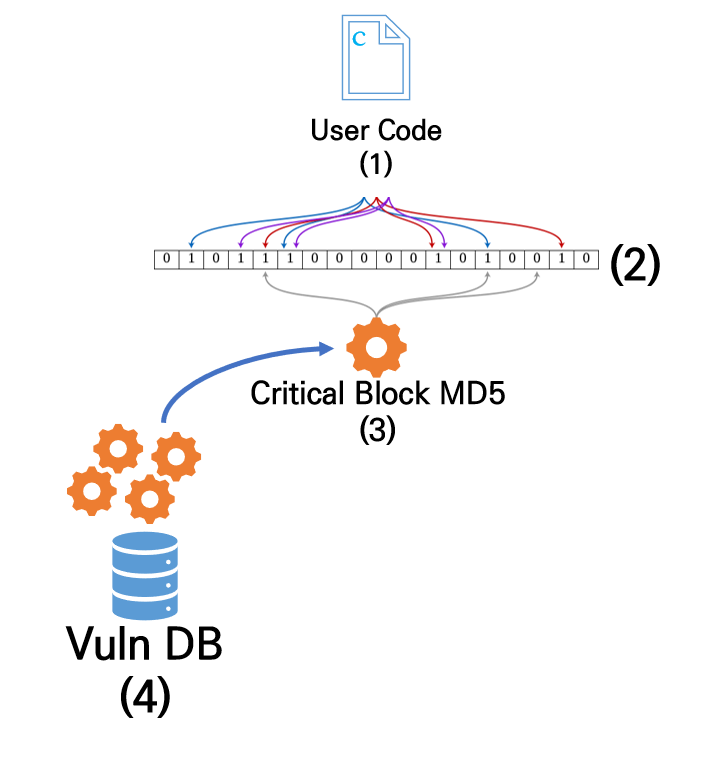
[그림 6] Non-Critical한 블록과 Critical 블록

[그림 2]는 OpenSSL 라이브러리의 CVE 패치 사례이다. 해당 CVE의 패치로 변경된 코드를 보면 변수 i, inl, b1과 연관 있는 것으로 볼 수 있다. Critical 블록 선정은 이 3가지 변수가 사용된 블록(노란색 블록)을 취약 코드 라인 위에서 선정한다. 동시에 관련성이 적은 블록은 Non-Critical(분홍색 블록) 블록으로 선정한다. 취약점 코드 클로닝 탐색은 Critical 블록을 이용해서 탐색한다.



[그림 7] Critical 블록 토큰화

Critical 블록은 [그림 2]처럼 큰 블록 단위로 나누고 나누어진 각각의 블록에서 변수 이름이 바뀌었거나 물리적 명령 행 차이에 따른 인식률 저하를 방지하기 위해 [그림 3]처럼 변수 이름을 추상화 하고 블록을 정규화한 뒤 MD5로 Hash 한다. 이렇게 처리 된 데이터는 서버의 Vulnerability DB(7)에 발견한 코드 길이, Repository 이름, CVE 코드, MD5 값, 함수명, 패치 전 코드, 패치 후 코드로 구성한다. 다른 Critical 블록들도 같은 방법으로 진행한 후 토큰화 된 모든 Critical 블록 데이터를 하나의 Chain으로 구성한 뒤 하나의 행으로 저장한다.



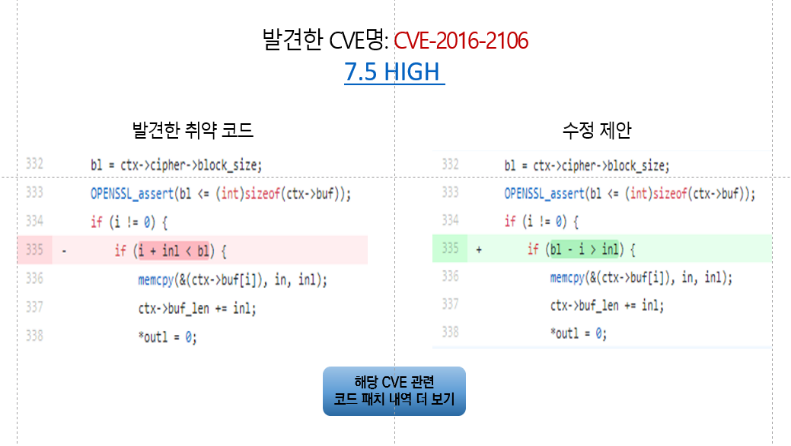
[그림 8] Bloom filter를 이용한 취약점 Matching

이런 구조의 DB로 취약점 코드를 탐지하는 방법은 [그림 4]와 같다. 먼저 사용자 소스코드 [그림 4-(1)]를 순차적으로 읽으면서 각 함수 내부의 큰 블록들을 [그림 3]과 동일한 방법으로 토큰을 만든다. 각 하나의 토큰 마다 빠른 취약점 Matching을 위해 Bloom filter 구조 [그림 4– (2)]에 추가한다. 그 후 Vulnerability DB [그림 4– (4)]에서 저장되어 있는 취약점 코드 블록의 MD5 값[그림 4–(3)]을 Bloom filter에 검사한다. Critical 블록 한 체인 안에 있는 모든 블록 HASH 값이 포함되어 있을 때 해당 CVE의 취약점이 있다고 판단한다.

**동적 테스팅(4)**

사용자는 ‘VulnNotti’ 웹사이트에서 리눅스 기반의 분석 프로그램을 다운로드 받는다. 사용자가 자신의 IoT 디바이스 위에서 이 프로그램을 실행하면 한국인터넷진흥원이 발표한 취약점 분석 및 평가 가이드라인[4]에 따라 사용자 환경에서 동적으로 취약점을 통합 분석한다. 또 네트워크 프로토콜단에서 발생하는 취약점을 테스팅하고 RC4 같은 취약한 암호화 라이브러리를 사용하는지 검사한다. 이 검사는 네트워크로 진행되며 검사가 끝나면 ‘VulnNotti’ 웹사이트에 보고서가 자동으로 업로드 된다.

**시각화 보고서(5)**



[그림 9] 검사 결과 예시

사용자에게 검사 결과를 시각화한 보고서로 보여주는 모듈이다. 사용자 코드는 크게 정적 코드 분석과 동적 테스팅으로 분석 되며 각 분석 결과를 ‘VulnNotti’ 웹사이트에서 볼 수 있다. [그림 5]와 같이 정적 코드 분석은 발견한 취약점 CVE 정보와 위험 점수를 보여주고 각 발견한 취약점은 취약 코드와 수정 제안 코드를 보여준다. 수정 제안 코드는 취약 코드 수집 시 Commit 파일에서 패치 후 코드 부분을 통 수집한다. 수정 제안 코드는 개발자가 스스로 판단하고 더 적극적으로 적용할 수 있도록 하기 위함이다. 또 동일한 CVE에 대해서 패치 한 사례를 보여줌으로써 ‘VulnNotti’ 목적인 개발자가 능동적으로 보안에 신경 쓰면서 개발을 할 수 있도록 구성한다.

**실시간 알림(6)**

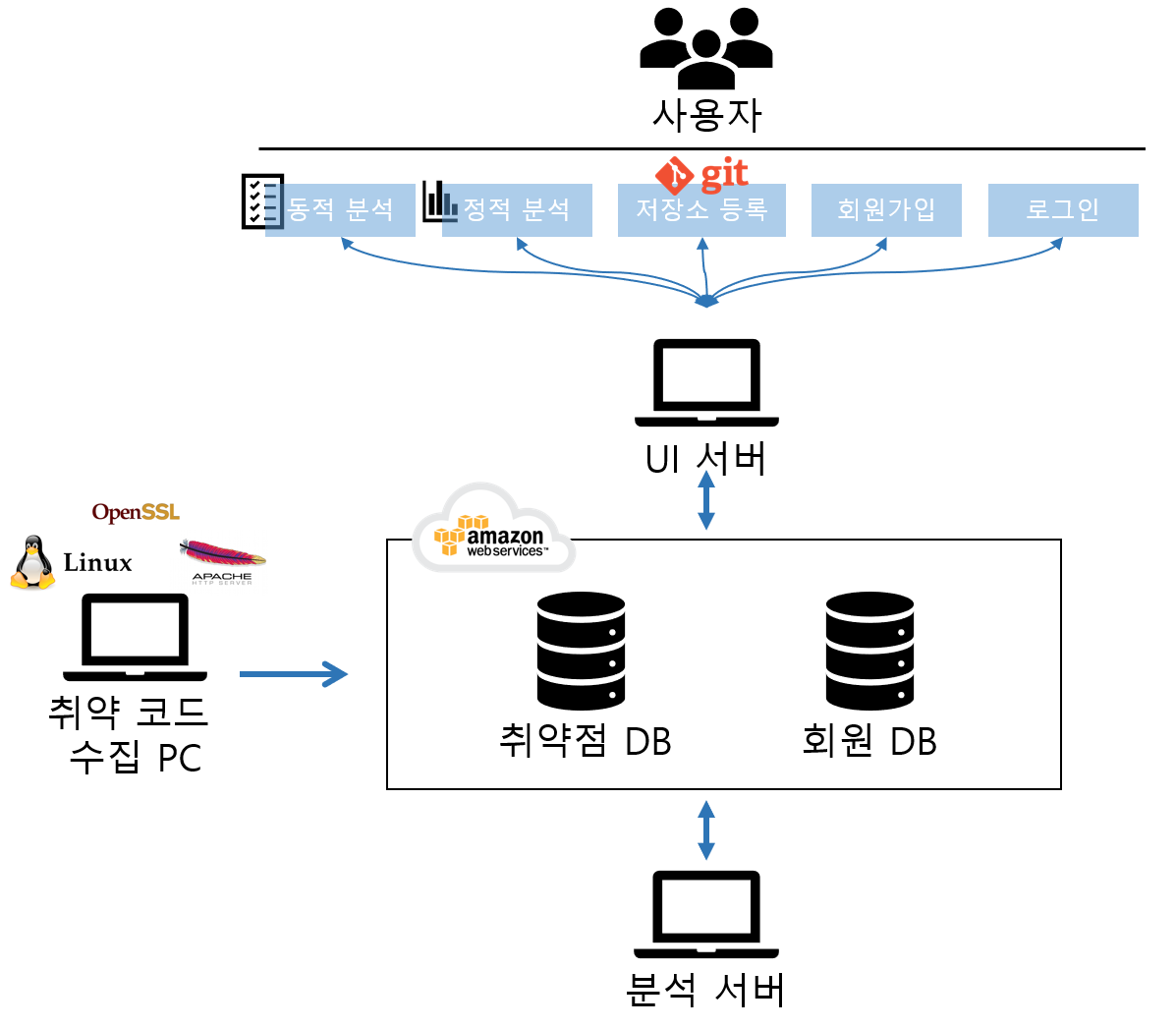
‘VulnNotti’ 서버는 지속적으로 Server DB(2)의 사용자 Git이 있는 DB를 검사한다. 새로운 Commit을 발견하면 그 Commit에 대해서 동일한 정적 코드 분석(3)을 진행한다. 그 결과를 보고서로 출력하고 사용자가 보고서를 확인할 수 있도록 Slack 같은 SNS 서비스 API를 이용하여 알림을 준다.

1. **프로젝트 결과 (노현종)**

**4.1 실험 환경**

|  |  |
| --- | --- |
| **CPU** | **I5-8250U 1.6~3.4GHz** |
| **RAM** | **DDR4 12GB** |
| **HDD** | **M.2 SSD 256GB** |
| **개발 언어** | **C#, Visual Studio 2017** |
| **운영체제** | **윈도우10 Home 64Bit** |
| **DB 엔진** | **AWS RDS MySql** |
| **수집한 오픈소스** | **Linux Kernel(18.05.25)**  **Open SSL(18.05.20)** |

**4.2 실험결과**

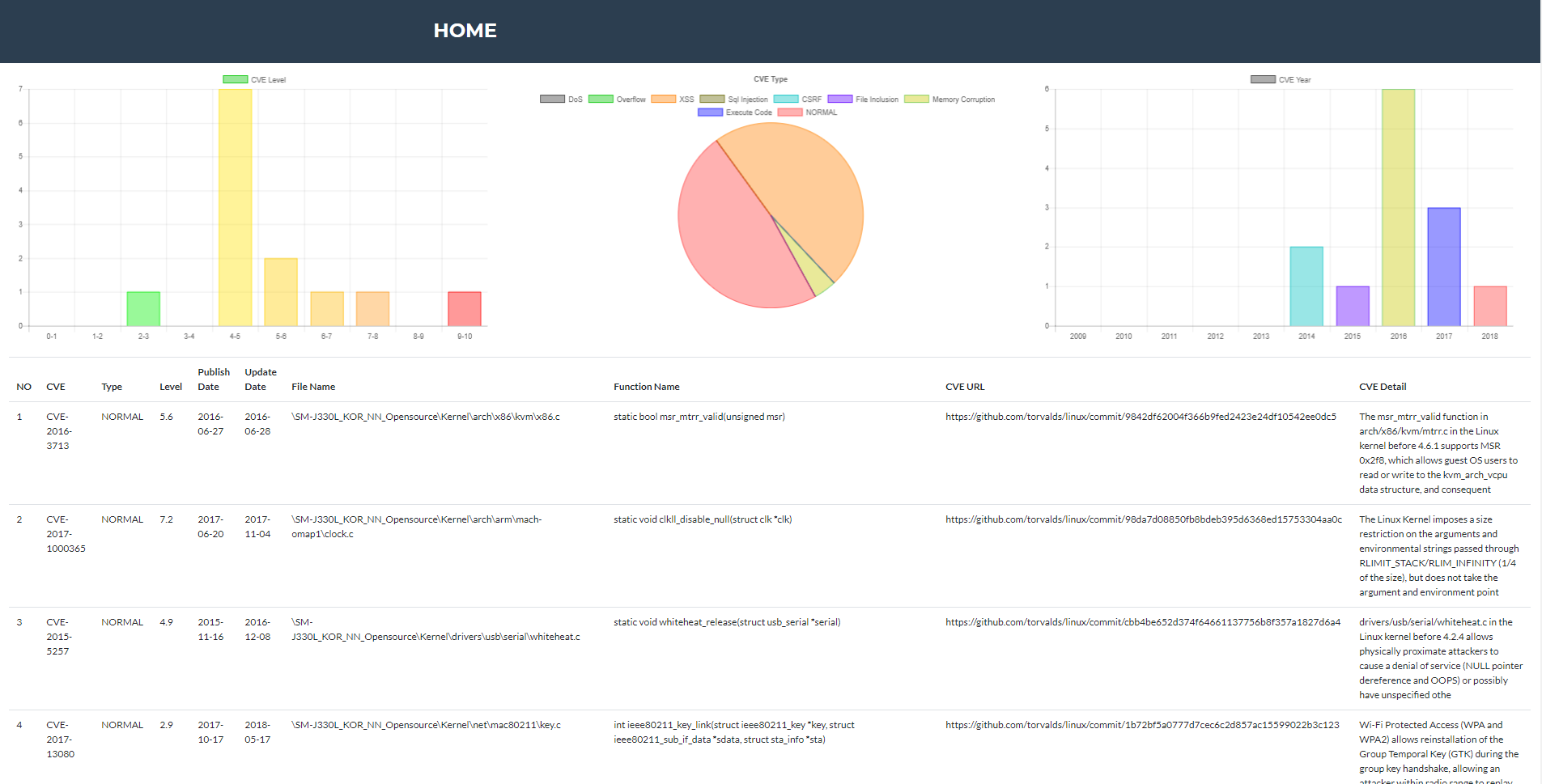


[그림 10] 아키텍처 구성도

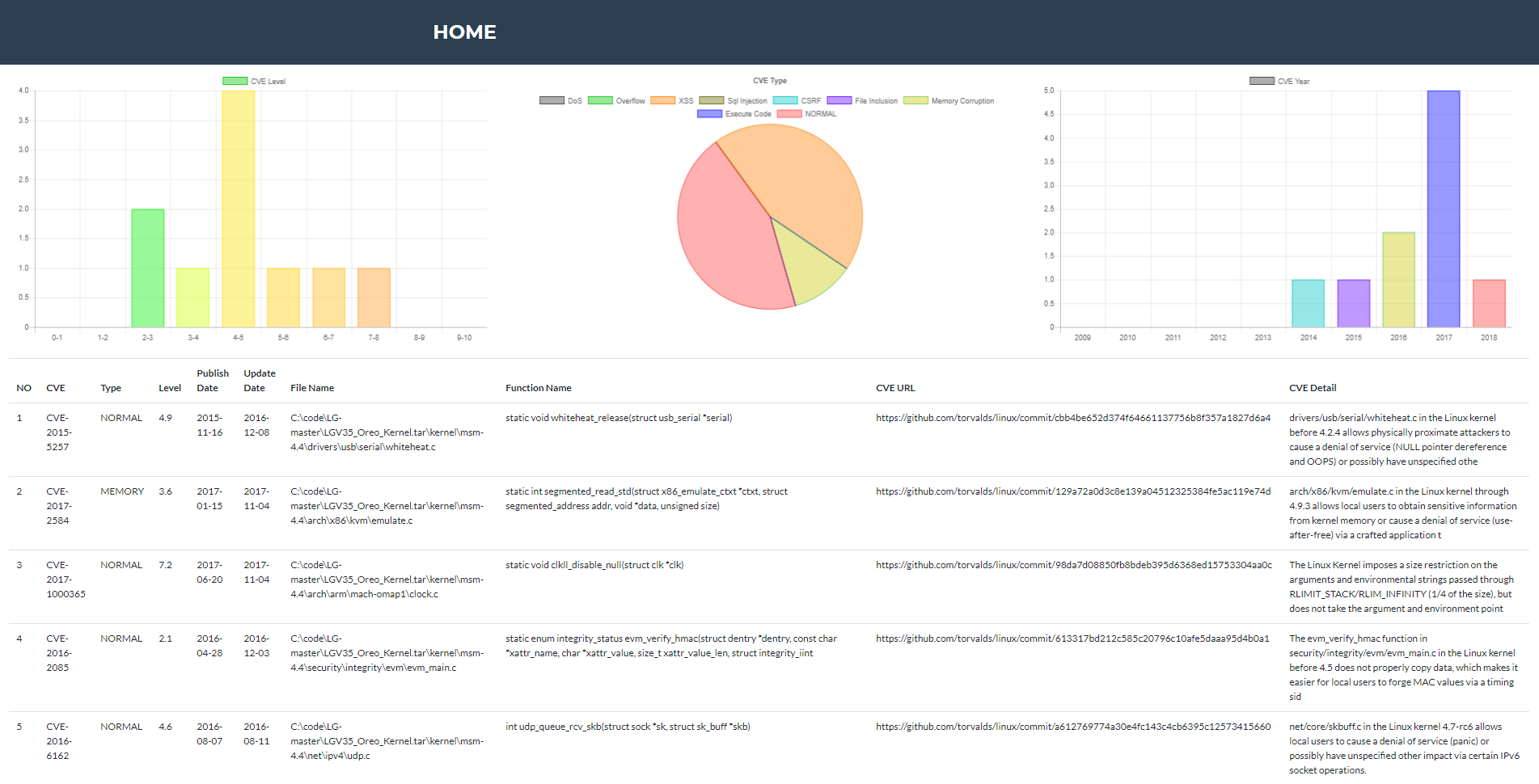
실험은 2018년에 출시된 최신 스마트폰인Samsung S9과 LG V35 ThinQ를 대상으로 진행하였다. 실험 목표는 두 제품 모두 안드로이드 운영체제 제품으로 2018년 이전에 발견된 리눅스 커널 취약점을 찾아내는 것이다.

Samsung S9에서는 13개의 취약점이 발견되었고 LG V35는 10개의 취약점이 발견하였다.

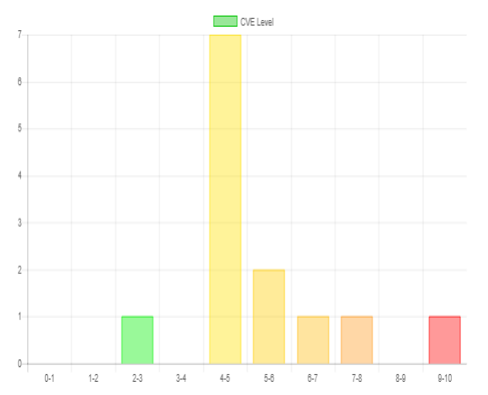
모두 2017년 이전 취약점이고 CVSS 7 이상의 고위험 취약점들도 다수 발견할 수 있었다.

****

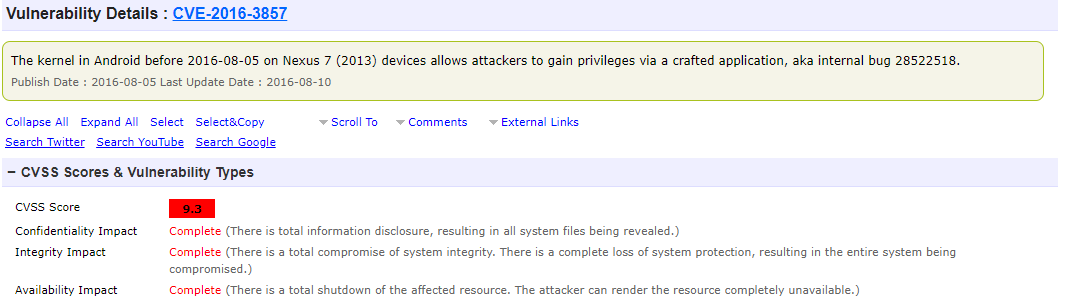
[그림 11] Samsung S9 분석 결과

****

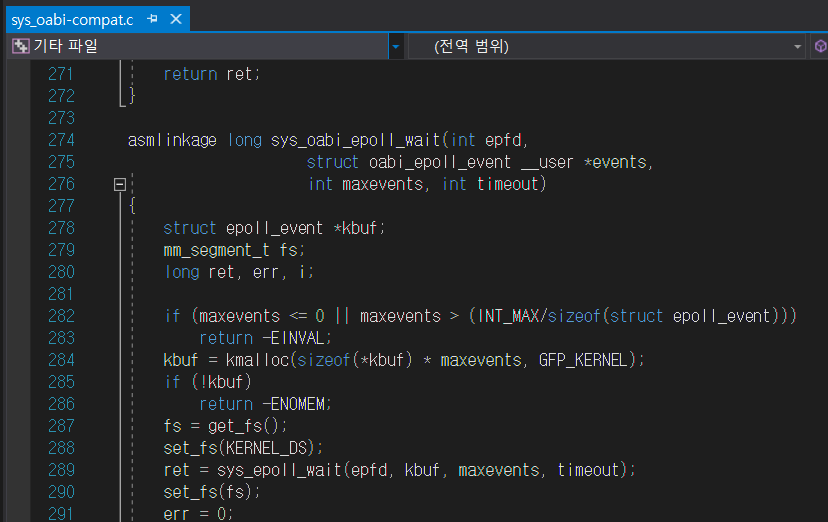
[그림 12] LG V35 ThinQ 분석 결과

****

[그림 13] Samsung S9의 CVE 점수 분포도

****

[그림 14] 발견된 취약점 중 가장 위험도가 높은 CVE

****

[그림 15] 취약점이 발견된 Samsung S9 sys\_oabi-compat.c 소스코드

****

[그림 16] 그림 15의 권장 소스코드

1. **결론**

**5.1 기대효과**

오픈소스가 현대 IT산업의 핵심 요소가 된 만큼, 앞으로 오픈소스에 대한 인기는 식지 않을 것이다. 더 많은 오픈소스가 응용프로그램에 적재될 것이고, IoT 기기 또한 마찬가지이다. 그만큼 오픈 소스의 보안이 중요해 진 것이다.

‘VulnNotti’는 정적 테스팅 단계, 동적 테스팅 단계로 나누어서 분석을 한다.

현재, 사용한 오픈소스를 탐지하거나, 오픈소스의 취약점을 알려주는 서비스는 상용화 되어있다. 기존의 것과 다르게 ‘VulnNotti’는 IoT 제품 개발에서 보안 지식이 부족한 개발자도 개발과 보안이 하나의 프로세스가 되도록 보안을 위해 취해야 할 노력을 줄임으로써 개발자가 개발에만 집중할 수 있도록 한다. 즉, 개발자가 소스코드를 작성하고 취약점을 검색한 뒤 취약한 코드를 수정을 할 필요 없이, 개발 후 git push를 하면 자동으로 코드를 분석해서 취약점을 알려줌으로써 개발자의 수고를 덜어준다. 또 기존에는 취약점의 종류인 ‘what’에 대한 설명에 집중했다면 ‘VulnNotti’는 왜 취약한 지를 설명하는 ‘why’를 최대한 설명하게끔 하여 개발자가 직관적으로 코드를 보고 학습하며 개발할 수 있도록 하는 것이 목표다. 또한 한번 검사한 소스코드는 데이터베이스에 저장한 후, 나중에 취약점이 발견 될 시 사용자에게 알림을 주는 기능을 통해, 오픈소스 패치를 잘 하지 않는 단점을 보완할 수 있다.

**5.2 추후 연구 방향**

현재 VulnNotti에서 부족한 점은 코드 변경에 유연한 취약점 탐지는 이루었지만 가장 큰 문제는 그만큼 취약점에 따라 오탐율도 높게 나타날 수 있는 것이다. 이를 해결하기 위해서는 발견한 취약점을 검증할 수 있는 모듈과 취약점을 탐지하는 컴퓨팅 파워 비용을 낮출 수 있는 방법이 필요하다. 마지막으로 현재 VulnNotti는 소스코드 분석만 지원하는데 기업이나 사용자에 따라 모든 소스코드를 가지고 있지 않을 수 있다. 바이너리 파일 상태에서 식별자를 수집하여 알려진 취약점을 검사할 수 있는 모듈 연구가 필요하다.

**참고 문헌**

[1] William Stallings, “Network Security Essentials Applications and Standards (5th Edition)

[2] VUDDY: A Scalable Approach for Vulnerable Code Clone Discovery