**캡스톤 디자인 Ⅱ**

**종합설계 프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *SOSfinder* |
| 팀 명 | *SOFA* |
| 문서 제목 | 최종보고서 |

|  |  |
| --- | --- |
| **Version** | 1.3 |
| **Date** | 2017-12-19 |

|  |  |
| --- | --- |
| **팀원** | 차진원 (조장) |
| 박민경 |
| 남혜인 |
| 강지형 |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 “SOSfinder”를 수행하는 팀 “SOFA”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “ SOFA”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | [SOFA]2학기\_최종보고서.doc |
| **원안작성자** | 전체 |
| **수정작업자** | 박민경 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2017. 12. 16 | 박민경 | 1.0 | 최초 작성 | 전체 항목 작성 |
| 2017. 12. 17 | 차진원 | 1.1 | 내용 추가 | 연구 개발 및 결과물 내용 추가 |
| 2017. 12. 18 | 강지형 | 1.2 | 내용 수정 | 일부 항목 내용 수정 |
| 2017.12.19 | 박민경 | 1.3 | 내용 수정 | 전체 항목 일부 수정 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**목 차**

[1 개요 4](#_Toc501415027)

[1.1 프로젝트 개요 4](#_Toc501415028)

[1.2 추진 배경 및 필요성 5](#_Toc501415029)

[1.2.1 IoT 기기 사용 증가에 따른 보안 위협 증가 5](#_Toc501415030)

[1.2.2 IoT 기기의 소스코드 공개여부 8](#_Toc501415031)

[1.2.3 설문조사를 통해 알아본 본 프로젝트의 필요성 9](#_Toc501415032)

[1.2.4 최근 기술 동향 11](#_Toc501415033)

[2 개발 내용 및 결과물 13](#_Toc501415034)

[2.1 목표 13](#_Toc501415035)

[2.2 연구/개발 내용 및 결과물 14](#_Toc501415036)

[2.2.1 연구/개발 내용 14](#_Toc501415037)

[2.2.2 시스템 기능 요구사항 21](#_Toc501415038)

[2.2.3 시스템 비기능(품질) 요구사항 21](#_Toc501415039)

[2.2.4 시스템 구조 및 설계도 23](#_Toc501415040)

[2.2.5 활용/개발된 기술 24](#_Toc501415041)

[2.2.6 현실적 제한 요소 및 그 해결 방안 26](#_Toc501415042)

[2.2.7 결과물 목록 27](#_Toc501415043)

[2.3 기대효과 및 활용방안 28](#_Toc501415044)

[3 자기평가 29](#_Toc501415045)

[4 참고 문헌 30](#_Toc501415046)

[4.1 테스트 케이스 31](#_Toc501415047)

# 개요

## 프로젝트 개요

최근 IoT(Internet of Things, 사물인터넷) 기기의 확산이 진행되면서, IoT 기기를 대상으로 하는 악성 공격들이 이루어지고 있다. 하지만 IoT 기기는 취약 부분에 대한 점검과 보수보다는 기능적인 추가를 더욱 우선시하여 발전하고 있는 추세이다. 앞으로 점점 더 많은 기기가 공격의 대상이 될 수 있으며 각각의 IoT 기기가 취약점을 가지는지에 대한 파악이 중요하다.

본 프로젝트에서는 최근 취약점이 알려진 오픈소스의 특정 소스코드 모듈을 다양한 플랫폼을 대상으로 컴파일하여 바이너리 코드 레벨의 시그니처 집합을 생성함으로써, ‘사용자의 IoT(Internet of Things, 사물인터넷) 기기의 취약점 포함 여부를 파악할 수 있는 서비스’를 구현한다. 여기서 사용자는 ‘Power User’로서, IoT 기기의 펌웨어를 이용하는 보안기관이나 IoT 소프트웨어의 개발자를 의미한다. 본 프로젝트에서 제공하는 기능은 아래와 같다.

첫째, IoT 기기 내부의 오픈소스 취약점 점검 기술이다. 사용자에게 자신의 IoT 기기에 있는 소프트웨어를 업로드하게 하여 해당 기기가 가진 취약점을 점검한다.

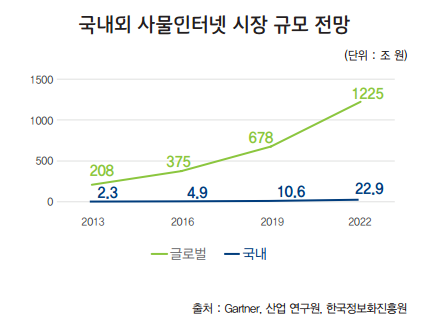
둘째, 취약점 정보 업로드 기술이다. 서비스를 제공하지 않는 소프트웨어 또는 최근에 발표된 취약점을 가진 소프트웨어를 사용자로부터 업로드 할 수 있게 하여, 사용자의 참여로 취약점 데이터베이스의 시그니처를 확장한다.

## 추진 배경 및 필요성

### IoT 기기 사용 증가에 따른 보안 위협 증가

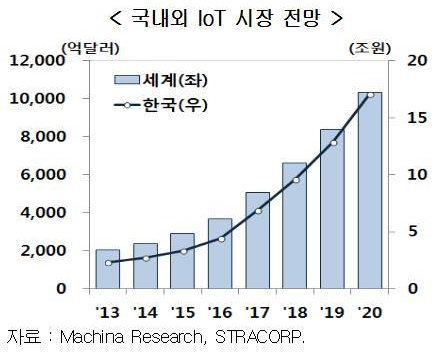
빠른 ICT(Information Communication Technology, 정보통신기술)의 발전으로 인해, 전 세계는 인터넷을 통해 서로 소통하는 환경에서 살고 있다. 아날로그 및 전자 기계 장치의 사용에서 디지털 기술로 넘어오는 제 3차 산업 혁명을 넘어서 현시대는 초연결성, 초지능성에 의한 생산성 향상을 기대하는 제4차 산업 혁명을 맞이하고 있다. 제 4차 산업 혁명은 ICT의 융합으로 이룬 혁명 시대를 의미하며 로봇 공학, 인공 지능, 나노 기술, 생명 공학, 사물인터넷, 자율 차량 등을 포함한 여러 분야에서 새로운 기술 혁신을 기대하고 있다. 본 프로젝트는 IoT(Internet of Things, 사물인터넷)를 통한 세계의 발전에 초점을 맞추고 있다.

IoT는 다양한 사물이 각기 부착된 통신장치와 센서를 통해 네트워크에 연결되고 정보를 공유할 수 있는 기술이다. 또한 가트너(Gartner)가 선정하는 10대 전략기술에 2012년부터 매년 내면 선정되어 ICT시장의 산업을 이끄는 핵심 부가가치 산업으로 급부상하고 있다. 이러한 기술을 활용하여 실생활영역에 적용되면서 다양한 경제적 가치와 더불어 효율성 및 편의성이 한층 높아질 것으로 기대되고 있다. 국내외 사물인터넷 시장은 많이 증가할 것으로 보인다.



[그림 1] 국내외 사물인터넷 시장 규모 전망

또한 현대경제연구원이 발표한 ‘사물인터넷(IoT) 관련 유망산업 동향 및 시사점’ 보고서에 따르면 세계 IoT 시장은 2015년 약 3천억 달러에서 2020년 1조 달러로 연평균 28.8% 성장할 것이고, 국내 IoT 시장은 연평균 38.5% 성장할 것이라는 전망이 나왔다. 같은 기간 국내 IoT 시장 규모도 3조3천억원에서 17조1천억원으로 연평균 38.5% 성장할 것으로 예상됐다.



[그림 2] 국내외 IoT 시장 전망, 한국경제연구원

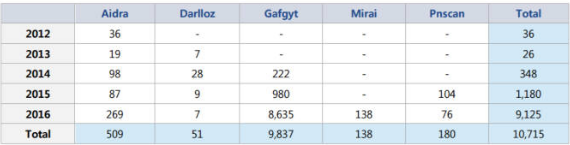
하지만 빠른 속도로 발전하는 IoT 기기에 대한 보안 위협 사례가 발표되면서, IoT 기기의 보안 문제는 아주 중요한 문제가 되었다. IoT의 확산에 따라 해킹이 더욱 쉬워지게 되었고, 네트워크 인프라가 점점 더 넓어짐에 따라 해킹할 수 있는 환경이 넓어지게 되었다. 2015년에 진행한 한 기사의 설문조사에 의하면 그 해의 가장 큰 보안 위협은 ‘IoT(사물인터넷) 기기의 보안위협’ 이라고 응답한 사람이 가장 많다고 한다.[[1]](#footnote-1)



[그림 3] 보안뉴스 기사의 설문조사[[2]](#footnote-2)

Akamai 위협 연구팀은 최근 인터넷에 연결된 수백만 대의 사물인터넷(IoT) 기기가 웹 기반 자격 증명 스터핑 공격의 소스로 사용되는 사례를 보고했다. 이러한 문제점은 이전에도 보고된 바 있으나, 수정되지 않고 IoT 기기에 적용되었다. 이 기기들은 Dyn이라 불리는 DNS 호스트에 DDoS 공격을 가하여 큰 문제를 발생시켰다.

사물인터넷 기기의 보안 위협은 리눅스 운영체제와 밀접한 연관이 있다. IoT 기기에 탑재되는 운영체제는 주로 임베디드 리눅스이며 리눅스는 오픈소스 소프트웨어이다. 따라서 IoT 업체들은 개발에 용이한 리눅스 운영체제와 오픈소스 소프트웨어를 많이 사용한다. IoT에서 제공하는 여러 어플리케이션 레벨의 소프트웨어들 역시 오픈소스를 활용한 소프트웨어인 경우가 다수이다. 오픈소스 소프트웨어는 다양한 개발자들의 참여로 코드 품질이 매우 높지만 생산 속도가 빠르고 많은 오픈소스를 생산하는 만큼 많은 취약점을 보유 및 패치한다. 만약 특정 IoT 기기가 보안에 취약한 오픈소스 소프트웨어를 포함한다면 IoT 기기 역시 취약점을 가지고 있기 때문에 보안위협에 영향을 끼칠 것이다. 그 밖에도 IoT 서비스는 기술 자체 혹은 구현 방법의 문제점으로 인해 다양한 취약점이 존재할 수 있다. 안랩(AhnLab)에 따르면 2008년 리눅스 악성코드가 처음 보고된 후 2016년 10월 기준으로 1만개를 넘어섰다고 한다. 이것은 특정 기기만 감염시키는 악성코드와 지속해서 변형이 나오지 않는 것을 제외한 수치이다.



[그림 4] 안랩이 공개한 주요 리눅스 악성코드 발견 현황

이처럼 IoT 기기를 노리는 악성코드는 급증하지만 대응은 쉽지 않은 것이 현실이다. IoT 기기는 상당수 보안을 고려하지 않고 설계되었다. 안전한 IoT 사용을 위해서, 기기 제조사가 보안을 강화한 펌웨어를 업데이트하고, 사용자는 지정되어 있는 초기 비밀번호를 바꿔야 한다. 보편적 사용자는 사용하는 기기가 위험에 유출되어 있는지에 대한 인식이 부족하고 기기가 보유한 취약성에 대해 바로 알 수 없다. 따라서 본 프로젝트는 IoT 기기에 들어있는 취약한 소프트웨어를 진단해주는 웹 서비스를 설계하고자 한다.[[3]](#footnote-3)

### IoT 기기의 소스코드 공개여부

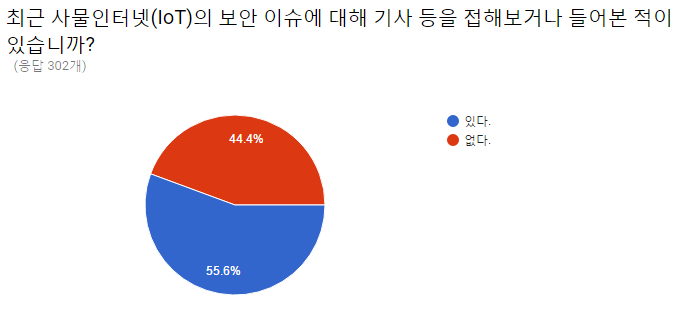
최근에 IoT 보안이 주목을 받으면서 이와 관련한 프로젝트들이 활성화 되었다. 하지만 관련 프로젝트들은 소스코드를 기반으로 검사하여 취약한 IoT 기기를 찾는 형태이다. 바이너리 코드 기반으로 검사를 하는 경우는, 주석에 달려있는 정보를 가지고 오픈소스의 존재 유무를 따져 라이선스 문제에 대한 대책으로 나와있는 상황이다. 이러한 경우 소스코드에 대한 지식이 있는 개발자만이 주로 사용하여, 프로그램 또는 제품에 라이선스로 인한 문제를 피해갈 수 있게 한다.

그러나 중요한 것은 소비자들이 주로 사용하는 IoT 기기의 소스코드 공개는 대부분 이루어지지 않는다. 또 오픈소스 라이선스의 문제로 인한 소스코드의 공개에도 부분적 공개나, 사용 출처를 표시하는 것으로 그치는 경우도 많다. 만약 제품의 전체 소스코드가 공개되어도 컴퓨터의 프로그래밍 언어를 공부하지 않은 일반 사용자의 경우에는 소스코드가 일반적인 문자에 지나지 않은 의미를 가진다. 이처럼 소스코드는 제한적으로 공개되기도 하지만 공개가 되어도 일반 사용자는 소스코드를 이용한 검사를 함에 어려움이 있다.

따라서 일반적인 사용자에게는 ‘실행 파일만 가지고 자신이 소유하고 있는 IoT 기기가 취약한지 한눈에 알려주는 서비스’가 필요하다.

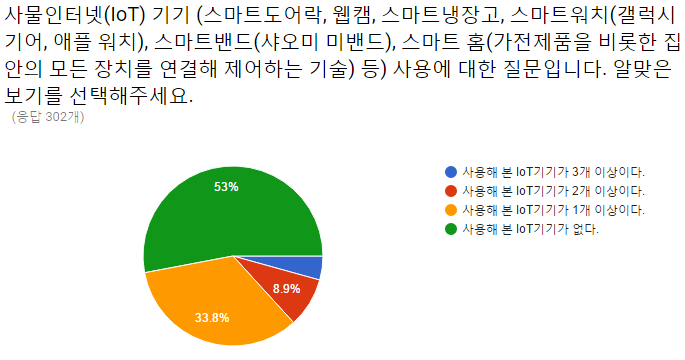
### 설문조사를 통해 알아본 본 프로젝트의 필요성

본 프로젝트와 관련해서 사람들의 IoT 사용 현황에 대한 설문조사를 진행하였다 SNS를 통해 설문조사 링크를 배포하여, 설문 대상자를 IT지식이 있는 사람부터 관련 지식이 전혀 없는 사람까지 포괄하게 하였다. 5일 동안 총 302명이 응답하였고, 결과는 다음과 같다.



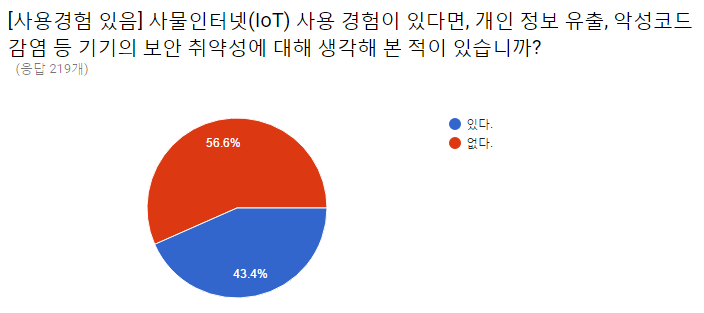
[그림 5] 설문조사 질문1에 대한 응답 결과

최근 IoT보안에 대한 뉴스 속보와 기사 등을 통해 보안 이슈를 접해본 사람들은 55.6%이었다. 절반 이상 정도의 사람들이 IoT에 대한 관심과 보안 문제에 대한 인식을 가지고 있음을 알 수 있다.



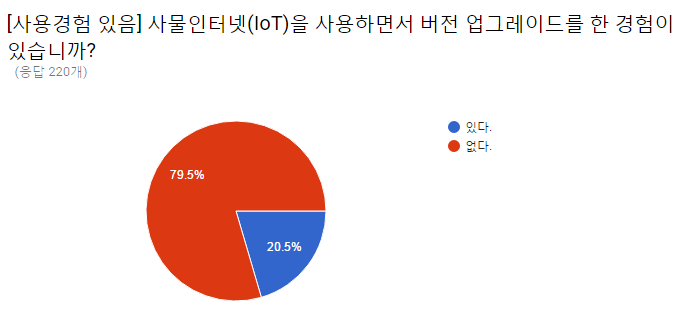
[그림 6] 설문조사 질문2에 대한 응답 결과

IoT 기기 사용 현황에 대한 질문이다. IoT시장이 빠르게 성장하고 있고, IoT 기기가 늘어나고 있는 현 실태에 맞게 사용해 본 사람의 수도 적지 않은 숫자로 약 47%에 달했다.



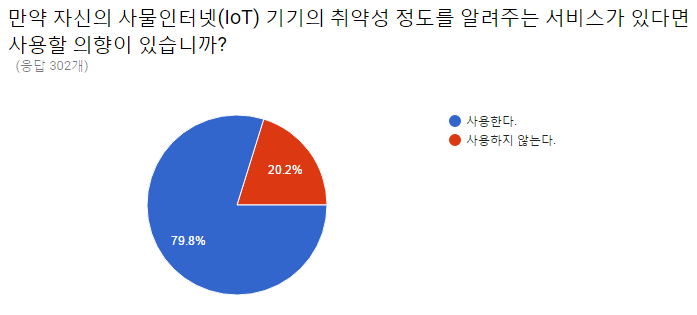
[그림 7] 설문조사 질문3에 대한 응답 결과

사용경험이 있는 사람들을 대상으로 한 보안 취약성에 대한 관심을 묻는 질문이다. IoT 기기를 사용해본 사람들이 자신의 기기의 보안에 대해 크게 관심이 있지는 않은 것으로 보여진다. 이를 통해, 사람들에게 IoT 기기 보안에 대한 관심과 보안 취약성에 대한 경각심을 심어주어 사용에 있어서 주의를 갖게 할 필요가 있음을 알 수 있다.



[그림 8] 설문조사 질문4에 대한 응답 결과

응답 결과와 같이, IoT 기기를 사용해 보았지만 버전 업그레이드를 한 경험이 있는 사람은 20.5%로 매우 적은 것을 알 수 있다. 질문3에서 보안 취약성에 대해 생각을 해 본 사람이 적은 것과 연관시켰을 때, IoT 기기 사용자들은 보안 취약성에 대한 인지가 부족하기 때문에 보안을 위한 버전 업그레이드에 대한 중요성을 느끼지 못하고 있다. 즉, 본 프로젝트와 같은 서비스 제공을 통해 사용자에게 자신의 기기의 위험성을 점검해 볼 필요성을 일깨워야 한다.



[그림 9] 설문조사 질문5에 대한 응답 결과

앞의 응답결과에서 언급하였듯이, IoT 기기 보안에 있어서 본 프로젝트의 필요성을 충분히 알 수 있다. 마지막으로, 질문5의 응답 결과를 보면, 본 프로젝트와 같은 서비스가 제공된다면 사람들의 사용 의향은 79.8%에 달했다.

전체적으로 설문조사 결과를 살펴보면, 많은 사람들이 IoT 보안 이슈에 대해 들어본 적은 있지만, 자신의 기기의 보안 취약성에 대해서는 인지하지 않고 있다. 따라서 본 프로젝트는 사용자들에게 IoT 기기 취약성에 대한 경각심을 심어주고, 기기의 보안 위험성에 대해 편리하고 쉽게 접하게 하기 위해 필요하다고 할 수 있다.

### 최근 기술 동향

현재 ‘BLACKDUCK’사에서 소프트웨어 속 오픈소스를 탐지하는 서비스를 제공하고 있다.[[4]](#footnote-4) 하지만 당사에서 추구하는 목표는 오픈소스의 취약점 탐지가 아닌 라이선스로 발생하는 법적 문제를 막기 위해 개발자에게 문제의 소지를 줄여주는 역할을 한다. 또한 고려대학교에서 시작한 ‘IoT Cube’ 서비스는 바이너리 파일 기반이 아닌 소스코드 기반으로 IoT 기기의 취약점 검사를 진행하여 소스코드를 보유한 경우에 사용할 수 있다.[[5]](#footnote-5) 하지만 일반 사용자의 경우 소스코드에 대해 인지하지 못하는 사용자가 많으며, 소프트웨어를 구성하는 소스코드를 확보함에 있어서 어려움이 있다. 2014년 ACM PLDI(Programming Language Design and Implementation**)**에서 발표된 ‘Tracelet’에서는 바이너리 코드의 정적 분석으로 코드를 찾는 방법에 대해 설명하고 있다.*[[6]](#footnote-6)* 해당 연구가 같은 domain을 지니는 대상에 대해 쉽게 사용가능 하지만 cross-domain일 경우 사용함에 있어 많은 cost가 발생한다고 언급하고 있다.

이처럼 사용자가 쉽게 이용하기 보다는 개발자들에게 문제점을 줄이기 위한 방안으로 많은 기술들이 제공되고 있으며, 여러 환경을 통하여 다루지 못하는 경우가 많이 있다. 따라서 사용자의 관점에서 쉽게 이용할 수 있고, 많은 환경을 지원하는 취약점 탐지 서비스가 필요하다.

# 개발 내용 및 결과물

## 목표

본 프로젝트는 사용자의 IoT 기기의 펌웨어에 대해 소스코드가 존재하지 않는 환경에서 바이너리 코드를 분석 비교하여, 취약점을 가진 오픈소스 코드가 사용되었는지 확인하고 진단 결과를 보여주는 웹 서비스이다. 1학기 캡스톤 디자인 수업을 통해 서비스의 개발을 완료하였다.

따라서 이번 2학기 캡스톤 디자인 프로젝트에서는 1학기 때 진행한 서비스의 성능 향상과 코드리뷰를 통한 리팩토링을 목표로 한다. 이를 위해 첫째로, 취약점 진단 모듈의 성능 향상을 위해 알고리즘을 수정한다. 기존의 사용했던 방식이 아닌, 슬라이딩 윈도우 방식을 기준을 두고 진행하여 정확하고 빠른 알고리즘을 목표로 한다.

둘째로, 서비스를 제공하는 웹 사이트의 데이터베이스 환경을 수정한다. DB Connection Pool 기능을 적용하여, 다수의 사용자가 동시에 서비스를 사용할 때 효율적인 데이터베이스의 접근을 목표로 한다.

## 연구/개발 내용 및 결과물

### 연구/개발 내용

#### 2.2.1.1 취약점 진단 모듈 코드의 Tokenize 함수 단순화

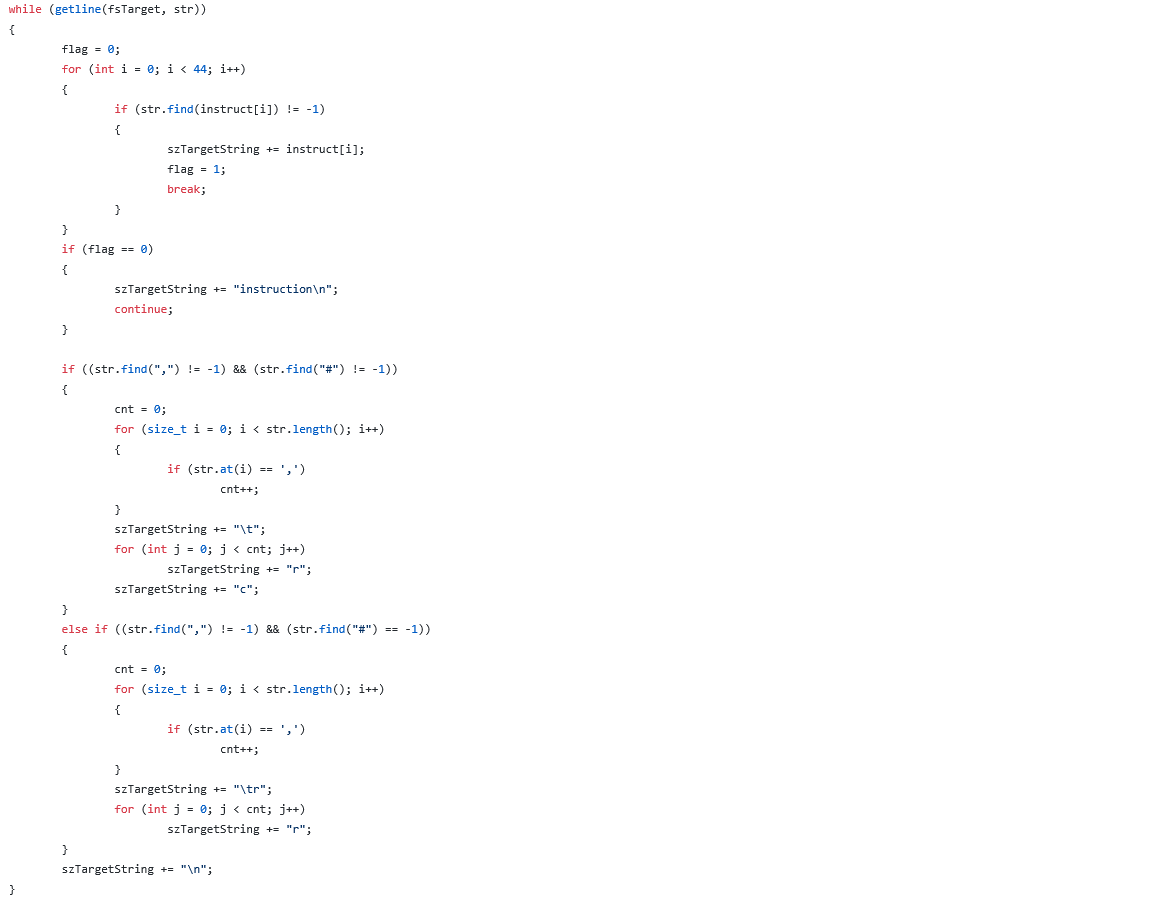


[그림 10] 변경 전 Tokenize 함수의 일부 코드

위 그림은 취약점 진단 모듈 코드의 Tokenize 함수의 변경 전 모습이다. If문이 연달아 나열되어, 일일이 모든 조건을 확인하는 형태로 되어있다. 이것은 매우 비효율적인 코드이므로, 명령어를 하나의 배열에 저장하고, 배열 안의 명령어 변수와 비교하는 방식으로 알고리즘을 수정하였다. 변경 후의 코드는 다음 그림과 같은 모습이다.

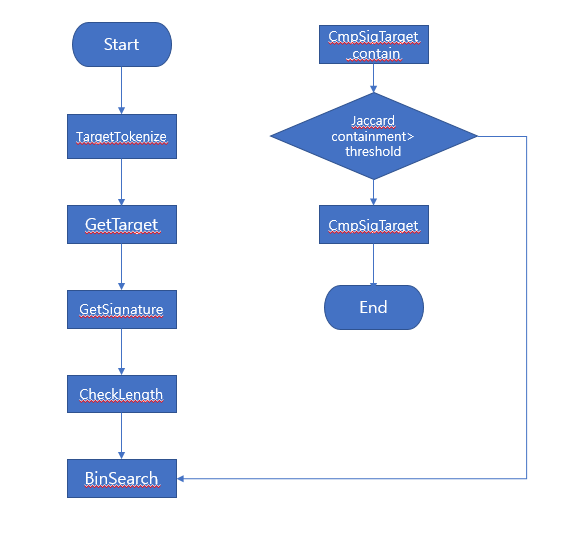


[그림 ] 변경 후 명령어들을 instruct 배열에 저장



[그림 12] 변경 후 Tokenize 함수의 일부 코드

#### 2.2.1.2 취약점 진단 알고리즘 수정

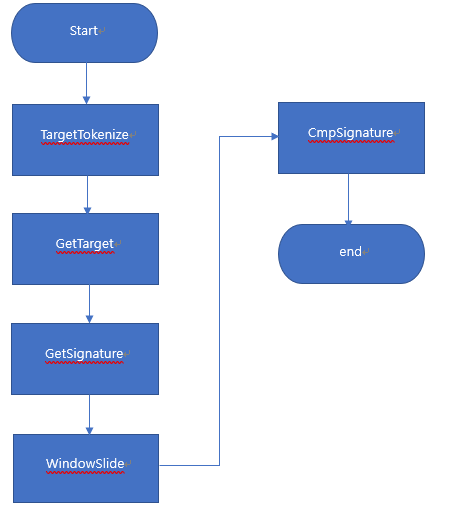


[그림 13] 변경 전 취약점 진단 알고리즘 모듈 Flow Chart

위 그림은 변경 전의 취약점 진단 알고리즘 모듈의 Flow Chart이다. 모듈의 알고리즘을 설명하면 다음과 같다.

주어진 인자를 기반으로 진단할 타겟의 어셈블리를 읽는다. 타겟의 어셈블리를 읽고 이를 뉴모닉, 메모리 주소, 레지스터, 상수로 tokenize를 진행한다. Tokenize가 진행된 타겟 파일은 데이터베이스에 저장되어 있는 시그니처 파일과 비교한다. 먼저, 데이터베이스에서 시그니처 정보를 가져와 구조체에 저장한다. Tokenize한 타겟 파일의 길이를 확인하고, 비교할 시그니처의 길이와 비교한 뒤 윈도우의 사이즈를 결정한다. Tokenize한 타겟을 반으로 나눈 다음 Jaccard containment 값을 계산한다. 기준으로 설정한 Jaccard containment 값을 넘어갈 경우, 다시 타겟의 크기를 반으로 나눈 다음 Jaccard containment 값을 계산하는 것을 반복한다. 이 값이 기준 아래로 판단될 경우 Jaccard index를 계산한다. 계산한 index 값 중 최댓값을 도출한다. 이후, 도출한 최댓값을 처음 설정한 취약도 범위에 따라 취약성의 정도를 판단한다.

하지만, 변경 전 알고리즘에서는 시그니처 파일보다 타겟 파일의 크기가 작을 경우, 비교가 안 되는 현상이 발생할 수 있다. 또한, 유사도의 비교를 진행할 때에 비교하는 두 파일의 크기가 동일한 상태에서 진행돼야 하는데, 본 알고리즘은 그렇지 않다. 마지막으로, 포함도 검사의 기준을 0.6이라는 수치로 일괄적으로 정해놓은 것이 문제였다.



[그림 14] 변경 후 취약점 진단 알고리즘 모듈 Flow Chart

따라서 취약점 진단 알고리즘을 전체적으로 수정하였다. 정확도가 떨어지는 것을 보완하기 위해, 비교 횟수를 줄일 수 있는 슬라이딩 윈도우 방식을 이용하여 수정하였다. 타겟 파일의 각 라인과 시그니처 파일의 첫 라인을 비교한 후, 동일한 경우에만 jaccard index를 계산하였다. 이에 의해, jaccard containment를 계산하는 부분과 binsearch 함수 부분을 삭제하였다.

#### 2.2.1.3 모듈 알고리즘의 수정 전후 성능 비교

성능 비교는 취약점의 포함 정도로 구분하여 총 3가지의 타겟 파일을 구성하여 진행하였다. 다음과 같이 3가지 상황으로 표현할 수 있다.

**1. 취약점을 완전히 포함하는 경우**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **타겟 파일** | **라인수** | **걸린 시간(이진탐색)** | **결과**  **(이진탐색)** | **걸린 시간**  **(윈도우슬라이드)** | **결과**  **(윈도우슬라이드)** |
| Pi3\_AI\_O3\_heartbeat-101f.txt | 2673 | 34.12 | 0.939542 | 24.17 | 1.0 |
| Pi3\_AI\_O3\_heartbeat\_101g.txt | 2679 | 34.27 | 0.9036 | 23.84 | 1.0 |

**2. 취약점을 부분적으로 포함하는 경우**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **타겟 파일** | **라인수** | **걸린 시간(이진탐색)** | **결과**  **(이진탐색)** | **걸린 시간**  **(윈도우슬라이드)** | **결과**  **(윈도우슬라이드)** |
| Pi3\_AI\_O3\_heartbeat-101f.txt | 2673 | 32.43 | 0.623245 | 23.73 | 0.356627 |
| Pi3\_AI\_O3\_heartbeat\_101g.txt | 2679 | 32.57 | 0.612383 | 23.80 | 0.337322 |

**3. 취약점을 포함하지 않는 경우**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **타겟 파일** | **라인수** | **걸린 시간(이진탐색)** | **결과**  **(이진탐색)** | **걸린 시간**  **(윈도우슬라이드)** | **결과**  **(윈도우슬라이드)** |
| Pi3\_AI\_O3\_heartbeat-101f.txt | 2673 | 0.05 | 0 | 0.07 | 0 |
| Pi3\_AI\_O3\_heartbeat\_101g.txt | 2679 | 0.05 | 0 | 0.07 | 0 |

위와 같이, 알고리즘을 수정한 후인 윈도우슬라이드 알고리즘에서는 취약점을 완전히 포함하는 경우 정확한 결과가 나오지만, 부분적으로 포함하는 경우에는 정확도가 매우 떨어지는 것을 확인할 수 있다.

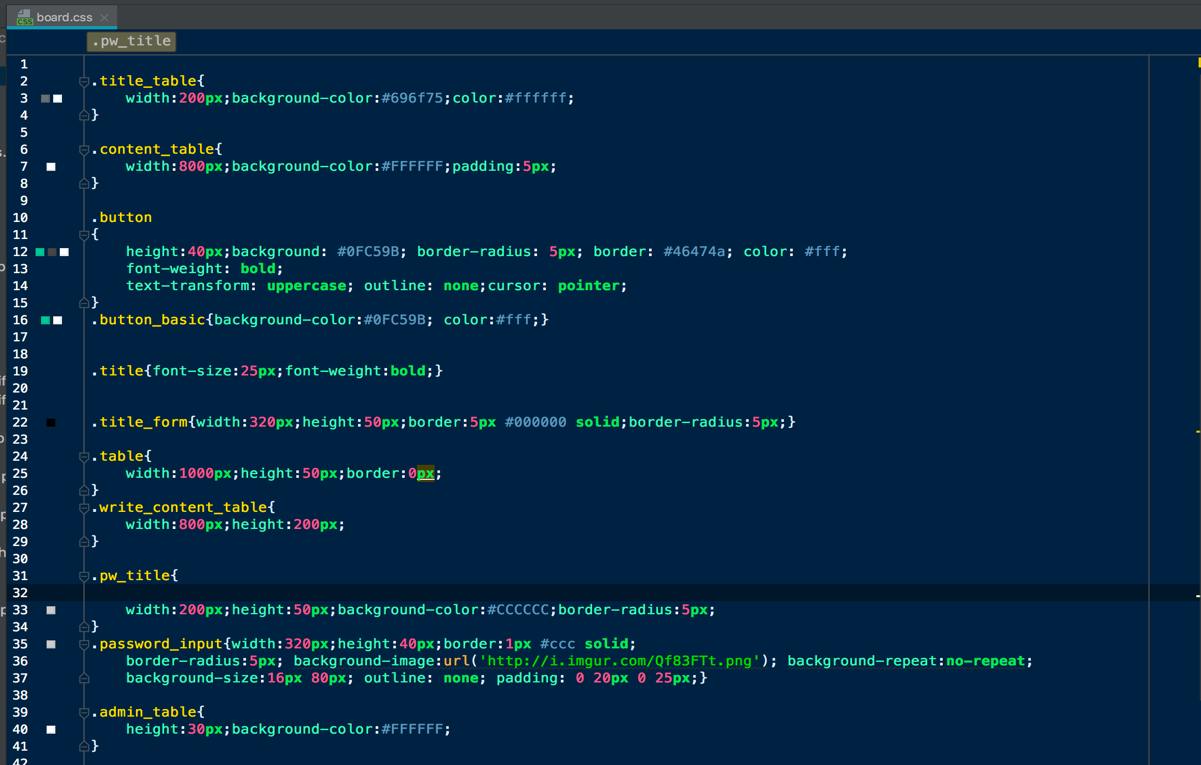
전체적으로 속도면에서는 윈도우슬라이드 알고리즘이 이진탐색 알고리즘보다 효율적이며, 정확도면에서는 이진탐색 알고리즘이 더 우수하다.

#### 2.2.1.4 웹 UI 부분의 CSS 코드 모듈화



[그림 16] 변경 전, CSS 코드의 일부분

위 그림과 같이, 웹 UI 부분의 변경 전 CSS 코드를 확인해 보면, 인라인 형태로 일일이 템플릿을 적용하는 것을 볼 수 있다. 효율적으로 UI 템플릿을 관리하기 위해, 각각의 웹 UI 파일에서 공통되는 부분의 CSS 코드를 다음과 같이 board.css라는 하나의 파일로 모듈화하여 정리하였다.

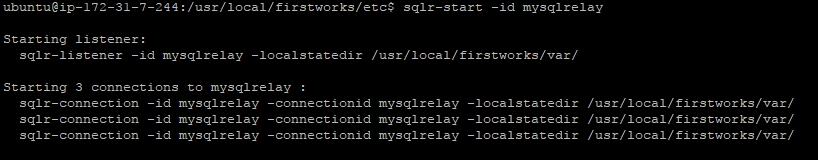


[그림 17] 변경 후 CSS 코드

#### 2.2.1.5 웹 데이터베이스의 DB Connection Pool 적용



[그림 18] DB Connection Pool 플랫폼 설치 및 설정



[그림 19] DB Connection Pool 적용 및 테스트

웹 데이터베이스의 세션을 효율적으로 관리하기 위해, DB Connection Pool을 적용하였다. MYSQL과 PHP 환경에서의 DB Pool 플랫폼으로 알려진 sqlrelay를 설치하여 환경에 맞게 세팅하였다([그림 18] 참고). 초기 연결 개수를 3개로 하여 적용한 결과 [그림 19]와 같이 3개의 프로세서가 실행되는 것을 확인하였다.

### 시스템 기능 요구사항

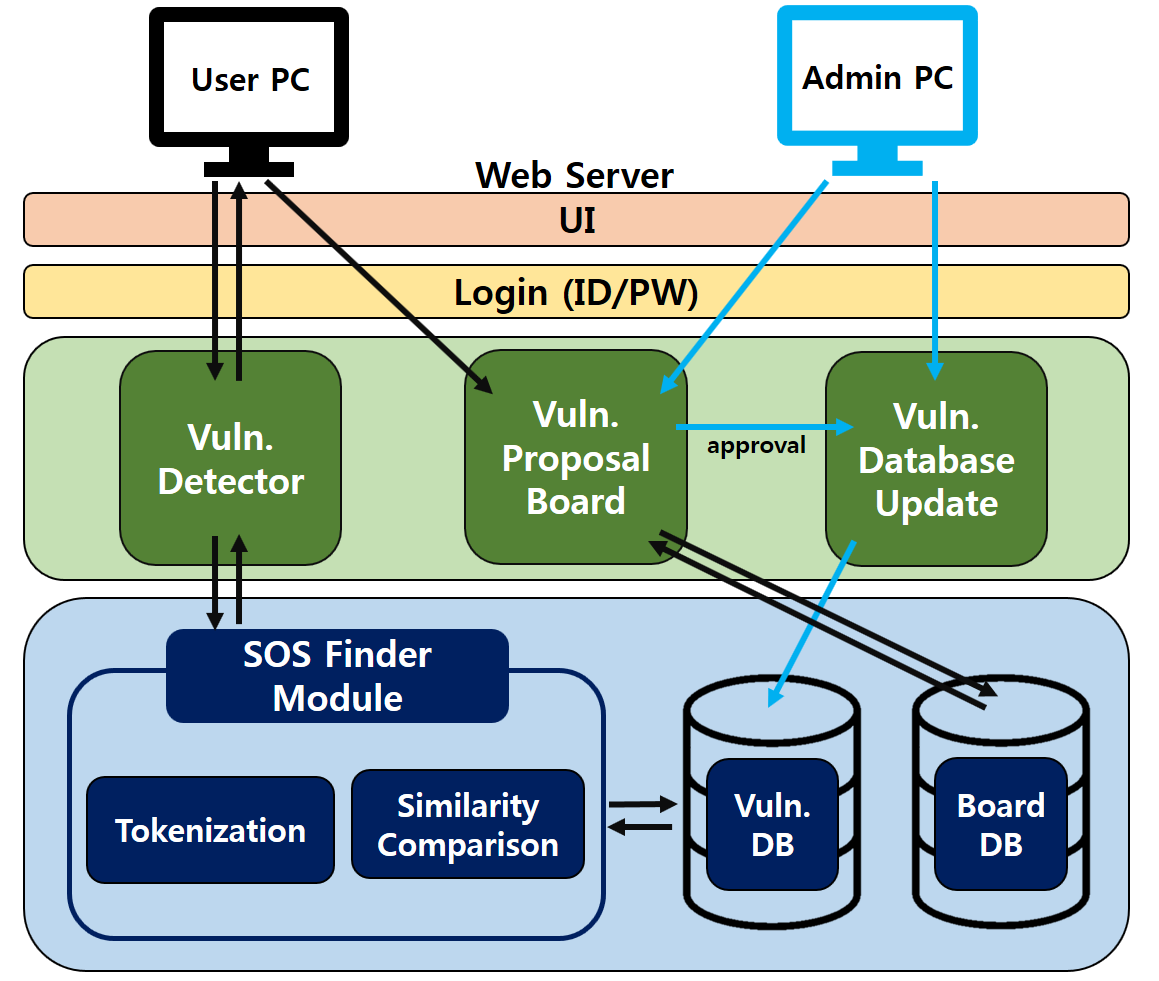
|  |  |  |
| --- | --- | --- |
| **프로그램 유형** | **IoT보안 웹 서비스** | **진행상황** |
| **요구사항** | Database Connection Pool 적용 | 완료 |
| CSS 코드 모듈화 | 완료 |

|  |  |  |
| --- | --- | --- |
| **프로그램 유형** | **취약점 점검 알고리즘 모듈** | **진행상황** |
| **요구사항** | 취약점 점검 알고리즘 전체 수정 | 완료 |
| Tokenize 함수 코드 단순화 | 완료 |

### 시스템 비기능(품질) 요구사항

|  |  |  |
| --- | --- | --- |
| **요구사항** | **내용** | **진행 상태** |
| **보안성** | 본 서비스의 이용자 중 악의적 목적으로 취약한 소프트웨어를 업로드할 수 있는 기능을 이용하는 경우, 데이터베이스를 오염시킬 수 있다. 따라서 권한을 지정한다. 권한은 일정 신용을 의미하는데, 해당 사용자가 커뮤니티에 글을 일정 수 이상 올려야 하며, 업로드한 글 중 관리자가 승인한 글이 일정 수 이상이어야 한다. 권한을 얻은 사용자에 한해서만 취약 소프트웨어 업로드 기능을 제공한다. | 완료 |
| 게시판에 사용자가 진단하는 취약점에 대한 건의사항을 업로드한다. 해당 글에 대해 관리자가 승인을 한 경우, 이 글은 ‘Good 게시글’이라고 취급한다. ‘Good 게시글’의 수가 많을 수록 신용이 높은 사용자이다. |
| **사용성** | 본 프로젝트의 목적은 IoT 기기 사용자를 대상으로 해당 기기의 소프트웨어 속 취약점 존재 여부를 판단하는 것이다. 프로젝트의 사용성은 사용자가 자신의 기기의 취약점에 대해 쉽게 파악할 수 있는 지로 구분할 수 있다. 이를 위해, 사용자가 기기의 실행 파일을 쉽게 업로드 할 수 있게 하고, 게시판을 두어 사용자와 관리자간 취약한 소프트웨어에 대한 의사소통을 가능하도록 한다. | 완료 |
| 사용성을 위하여 취약점의 결과를 가시화 할 때, 그래프를 사용하여 쉽게 파악할 수 있도록 하였다. 또한, 그래프 좌측에 퍼센트에 따른 취약성 정도를 기록하였다. 사용자와 관리자 간의 의사소통을 위해 게시판 메뉴에서 취약점에 대해 건의할 수 있도록 하였다. |
| **재사용 가능성** | 본 프로젝트의 목적은 IoT기기의 취약점을 탐지하는 것으로, 프로젝트 개발 기간의 제약 상 자카르드 인덱스를 통해 비교를 진행하나 비교의 정확도를 높이기 위해 다른 알고리즘을 적용할 수 있도록 모듈화한다. 또한 웹 서비스의 구현과 취약점 점검 모듈을 분리하여 구현한다. | 완료 |
| 웹 서비스에서 서버에 존재하는 점검 모듈을 통해 점검을 하고, 결과를 받아오는 형태로 구현하였다. |
| **정확성** | IoT 기기의 취약점 점검 모듈의 알고리즘을 수정함으로써, 정확하고 빠른 점검을 제공하며 서비스의 정확성을 높인다. | 완료 |
| ‘2.2.1.3 모듈 알고리즘의 수정 전후 성능 비교’에서 확인이 가능하다. |
| **효율성** | 웹 서비스의 Database 환경을 수정하여, 효율적인 DB 연결 및 세션 관리를 통해 효율성을 높인다. | 완료 |
| ‘2.2.1.5 웹 데이터베이스의 DB Connection Pool 적용’에서 확인이 가능하다. |

### 시스템 구조 및 설계도



[그림 20] SOSfinder 시스템구조도

- User와 Server 사이의 데이터 전달을 위해 APM(Apache & PHP & MYSQL)을 이용하여 Web Service를 구성한다.

- 사용자(Power User)는 Web Service에 로그인하여 사용자 인증을 한 후, IoT 기기에서 실행파일(바이너리 코드)을 업로드한다.

- Web Service는 Server로 실행파일을 전달하고 Server는 취약점 진단 알고리즘 모듈을 사용하여 진단한 뒤, 결과를 Web service에 전달한다.

- Web service는 진단 결과를 가시화하여 취약 정도를 그래프로 표현하고, 퍼센트 수치로 제공한다.

### 활용/개발된 기술

**1) IoT 보안 웹 서비스**

1. 본 프로젝트의 어플리케이션은 Web으로 설계하였으며 사용자가 사용하기 쉬운 인터페이스를 제공한다.

2. 사용자는 로그인을 통해 서비스의 권한을 얻고, 메인 화면에 있는 파일 업로드 기능을 통해 검사 받기를 원하는 파일을 쉽게 업로드한다.

3. 업로드 이후 검사 버튼을 누르면 서버의 진단 모듈에서 취약점을 진단하여 결과를 받아온다.

4. 그래프를 통해 사용자는 자신이 업로드한 파일이 어떤 오픈소스에 의해 취약한지 정보를 얻는다.

5. 사용자는 게시판에 검사하고 싶은 오픈소스의 취약점에 대해 건의할 수 있다. 건의한 글을 관리자가 많이 선택할 수록 사용자의 신용이 높아진다. 이로써, IoT 보안에 대한 커뮤니티를 형성한다.

**2) 취약점 점검 알고리즘 모듈**

1. TargetTokenize: 타겟 파일의 어셈블리 코드를 mnemonic, memory address, register, offset으로 분리하여 토큰화한다.

2. GetSignature: database에서 미리 저장된 시그니처 리스트를 가져와 비교 대상을 찾는다.

3. CheckLength: 타겟 파일의 토큰화된 어셈블리의 길이를 확인하고 선택된 signature의 길이를 비교하여 window size를 정한다.

4. CmpSigTarget: 시그니처와 타겟을 비교하여 유사도를 확인하고 기준치를 넘는 값들의 이름을 가져온다.

\*타겟 파일: 사용자가 업로드한 파일

|  |  |  |
| --- | --- | --- |
|  | **구분** | **SOSfinder** |
| **개발에 필요한 환경** | 운영체제 | Linux |
| 컴파일 | g++ |
| 개발언어 | C++, Html, PHP, MySQL |
| 언어의 문법적 요구사항 |  |
| **결과물을 확인하는 환경** | 운영체제 | Ubuntu, Windows7 |
| 컴파일 |  |
| 개발언어 |  |
| 언어의 문법적 요구사항 |  |

### 현실적 제한 요소 및 그 해결 방안

**1) 취약점 데이터베이스 구성의 제한**

* 본 프로젝트의 시간적 제약 때문에 취약점에 대한 데이터베이스의 구성을 위해 사용하는 소프트웨어의 다양성이 제한될 수 있다. 따라서 일부 환경 안에서 3가지 오픈소스의 취약점을 54개의 환경에서 진단하는 시그니처 파일을 총 162개 구성하였다. 또한, 확장성을 열어두기 위해 웹 서비스의 게시판 기능을 마련하여 취약점의 추가 가능성을 열어두었다.

### 결과물 목록

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 대분류 | 소분류 | 기능 | 형식 | 진행 상태 | 수정 사항 |
| **IoT보안 웹 서비스** | *레이아웃* | 사용자에게 편리한 User Interface를 제공한다. |  | 완료 |  |
| *파일 업로드* | 사용자 기기의 소프트웨어를 업로드한다. | 모듈 | 완료 |  |
| *DB 연동, 알고리즘 적용* | 업로드한 소프트웨어의 정보를 서버에 전달하고, DB에 저장 후 유사도 알고리즘을 적용한다. | 모듈 | 완료 |  |
| *대시보드* | 사용자가 업로드한 소프트웨어의 취약한 정도를 출력한다. |  | 수정 | 대시보드가 아닌 그래프와 취약성 정도를 나타내는 단어로 출력한다. |
| **취약점 점검 알고리즘 모듈** | *TargetTokenize* | Target 기기의 어셈블리 코드를 mnemonic, memory address, register, offset으로 토큰화한다 | 모듈 | 완료 |  |
| *GetSignature* | Database에서 signature list를 가져와 비교 대상을 찾는다. | 모듈 | 완료 |  |
| *CheckLength* | Target의 토큰화된 어셈블리의 길이를 확인하고 선택된 signature의 길이를 비교하여 window size를 정한다 | 모듈 | 완료 |  |
| *CmpSigTarget* | Signature와 Target을 비교하여 유사도를 확인하고 기준치를 넘는 값들의 이름을 가져온다 | 모듈 | 완료 |  |

## 기대효과 및 활용방안

**1) 보안에 미숙한 개발자들의 검정**

개발자들이 오픈소스를 사용하여 개발하였을 경우, 본 웹 서비스를 활용하여 보안 상태를 확인할 수 있다. 최근 이슈가 된 취약점을 중심으로 진단하여 개발자가 개발하는 과정에 있어서 보안에 주의하여 개발할 수 있도록 경각심을 부여한다.

**2) 파워 유저의 자신의 IoT 디바이스 검사**

본 프로젝트는 IoT기기의 실행 파일을 업로드 할 수 있는 지식을 가진 파워 유저를 대상으로 한다. 일반 사용자의 경우, IT분야의 지식이 부족할 수 있기 때문에, 실행 파일 업로드에 어려움을 겪을 수 있다. 따라서 추후에 일반 사용자도 실행 파일을 업로드할 수 있도록 매뉴얼을 제공할 예정이다. 파워유저는 자신이 사용하는 IoT 디바이스를 검사함으로써, 자신의 개인정보 유출 위험이 있는지 파악할 수 있다. 따라서 본 서비스는 사용자의 안전한 디바이스 사용을 돕는다.

**3) 취약점 관련 커뮤니티 형성**

본 서비스는 게시판 기능을 통해 사용자와 관리자간, 사용자와 다른 사용자 간의 IoT 디바이스의 보안 취약성에 대해 소통이 이뤄진다. 따라서 방화벽과 같이 디바이스 위에 올려지는 보안의 형태가 아닌, 웹 커뮤니티라는 새로운 형태의 IoT 보안을 제안한다.

# 자기평가

|  |  |
| --- | --- |
| **최종 결과물** | IoT 보안 웹 서비스 |
| **주요 평가 기준** | IoT 기기의 들어있는 오픈소스의 취약점을 정확하게 검사할 수 있는가? |
| **자기평가** | 본 프로젝트는 IoT 기기에 속한 취약점 중 OpenSSL-1.0.1f의 ‘dtls1\_process\_heartbeat’ 함수, Freetype-2.5.2의 ‘cf2\_hintmap\_build’ 함수, OpenSSH-6.4의 ‘hash\_buffer’ 함수에 대해 정확하게 찾아내고, 취약성 정도를 상세하게 진단한다. 각 취약점에 있어서, 다음과 같은 플랫폼에 대해 진단할 수 있다. 3개의 라즈베리 파이(Pi1, Pi2, Pi3) 기기, 3개의 운영체제(Raspbian, Arch Linux, Kali Linux), 4.9.2 버전의 arm-linux-gcc 컴파일러, 그리고 6개의 최적화 옵션(O0, O1, O2, O3, Ofast, Os)이다.  더 많은 취약점에 대한 확장성을 열어두기 위해, 웹 사이트형태의 IoT 보안을 제안한다. 본 서비스는 웹 사이트 상에 사용자가 서비스에 존재하지 않는 소프트웨어나 최근 유행이 된 취약점 중 진단하고 싶은 소프트웨어를 업로드 할 수 있는 ‘Board’ 기능이 존재한다. 사용자가 업로드한 파일이 적절하다고 관리자가 판단할 시, 취약점 데이터베이스에 저장한다. 이를 통해, 취약점 데이터베이스의 확장성을 두었다.  결과적으로, 본 SOSfinder 서비스는 IoT기기의 오픈소스 취약점에 대해 정확하고 상세하게 검사할 수 있고, 앞으로 더 많은 취약점에 대해 점검할 수 있는 확장성을 갖추고 있다. |

# 참고 문헌

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처 | 발행연도 | 저자 |
| 1 | 기사 | <http://www.yonhapnews.co.kr/bulletin/2016/07/12/0200000000AKR20160712109300002.HTML> | 연합뉴스 | 2016.7 | 박의래기자 |
| 2 | 기사 | http://www.boannews.com/media/view.asp?idx=45510 | 보안뉴스 | 2015.03 | 김태형 기자 |
| 3 | 기사 | <http://www.etnews.com/20161222000418> | Etnews | 2016.12 | 김인순기자 |
| 4 | 논문 | Winnowing: Local Algorithms for Document Fingerprinting | SIGMOD | 2003.12 | Saul Schleimer,  Daniel S. Wilkerson,  Alex Aiken |
| 5 | 소프트웨어 | IoT Cube | 고려대학교 | - | 고려대학교 |
| 6 | 논문 | Tracelet-Based Code Search in Executables | ACM PLDI ’14 | 2014.6 | Eran Yahav, Yaniv David |

## 테스트 케이스

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 대분류 | 소분류 | 기능 | 테스트 방법 | 기대 결과 | 테스트  결과 |
| 웹 서비스 | 기능 |  | | |  |

1. [1] 연합뉴스 "한국 사물인터넷시장, 2020년 17조…연평균 38% 성장 전망" [↑](#footnote-ref-1)
2. [2] 보안뉴스 [설문조사] 올해 가장 큰 보안위협은 ‘사물인터넷’ [↑](#footnote-ref-2)
3. [3] Etnews IoT 노리는 악성코드 1만개 넘었다 [↑](#footnote-ref-3)
4. [4] Blackduck protex BAT [↑](#footnote-ref-4)
5. [5] IoT Cube [↑](#footnote-ref-5)
6. [6] Tracelet-Based Code Search in Executables [↑](#footnote-ref-6)