



캡스톤 디자인 I

종합설계 프로젝트

프로젝트 명	<i>SOSfinder</i>
팀 명	<i>SOFA</i>
문서 제목	결과보고서

Version	0.1
Date	2017-05-24

팀원	송창현 (조장)
	정성민
	박민경
	정현석
	차진원
	핫산



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	수행결과보고서-SOSfinder.doc
원안작성자	전체
수정작업자	박민경

수정날짜	대표수정자	Revision	추가/수정 항목	내용
2017. 5. 24	박민경	1.0	최초 작성	전체 항목 작성



목 차

1	개요	4
1.1	프로젝트 개요	4
1.2	추진 배경 및 필요성	5
1.2.1	IoT 기기 사용 증가에 따른 보안 위협 증가	5
1.2.2	IoT 기기의 소스코드 공개여부	8
1.2.3	설문조사를 통해 알아본 본 프로젝트의 필요성	9
1.2.4	최근 기술 동향	11
2	개발 내용 및 결과물	13
2.1	목표	13
2.2	연구/개발 내용 및 결과물	14
2.2.1	연구/개발 내용	14
2.2.2	시스템 기능 요구사항	28
2.2.3	시스템 비기능(품질) 요구사항	28
2.2.4	시스템 구조 및 설계도	30
2.2.5	활용/개발된 기술	31
2.2.6	현실적 제한 요소 및 그 해결 방안	33
2.2.7	결과물 목록	34
2.3	기대효과 및 활용방안	35
3	자기평가	36
4	참고 문헌	37
4.1	테스트 케이스	38



1 개요

1.1 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 보수 보다는 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.

IoT를 구성하는 소프트웨어는 대부분 오픈소스 소프트웨어로 이루어져 있다. 하지만 오픈소스 소프트웨어 역시 많은 취약점을 가지고 많은 사람이 패치를 하고 있다. 만약 IoT 기기에 탑재되는 오픈소스 소프트웨어가 취약한 버전을 사용한다면 IoT 기기 역시 취약점을 보유하고 있는 상태가 되기 때문에 이는 매우 심각한 문제이다.

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.

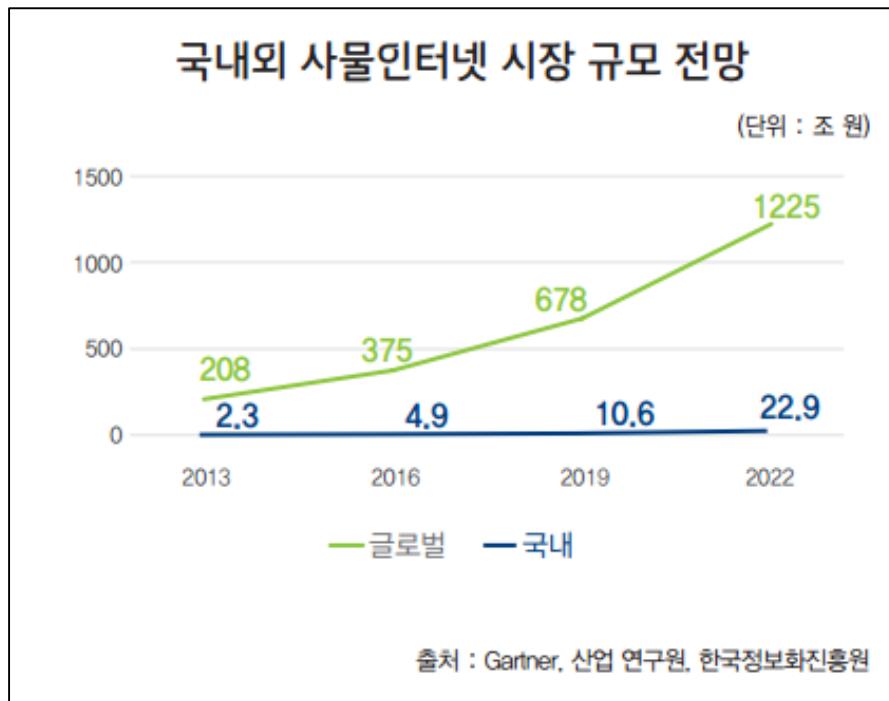


1.2 추진 배경 및 필요성

1.2.1 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 3 차 산업 혁명을 넘어서 현시대는 초연결성, 초지능성에 의한 생산성 향상을 기대하는 제 4 차 산업 혁명을 맞이하고 있다. 제 4 차 산업 혁명은 ICT 의 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.

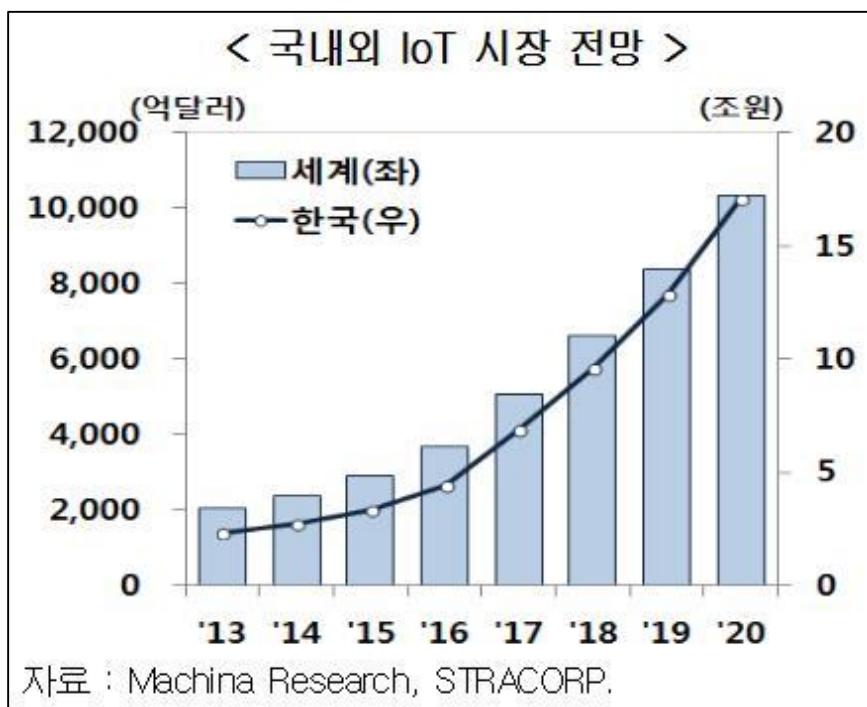
IoT는 다양한 사물이 각기 부착된 통신장치와 센서를 통해 네트워크에 연결되고 정보를 공유할 수 있는 기술이다. 또한 가트너(Gartner)가 선정하는 10 대 전략기술에 2012 년부터 매년 내면 선정되어 ICT 시장의 산업을 이끄는 핵심 부가가치 산업으로 급부상하고 있다. 이러한 기술을 활용하여 실생활영역에 적용되면서 다양한 경제적 가치와 더불어 효율성 및 편의성이 한층 높아질 것으로 기대되고 있다. 국내외 사물인터넷 시장은 많이 증가할 것으로 보인다.



[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 3조 3천억원에서 17조 1천억원으로 연평균 38.5% 성장할 것으로 예상됐다.

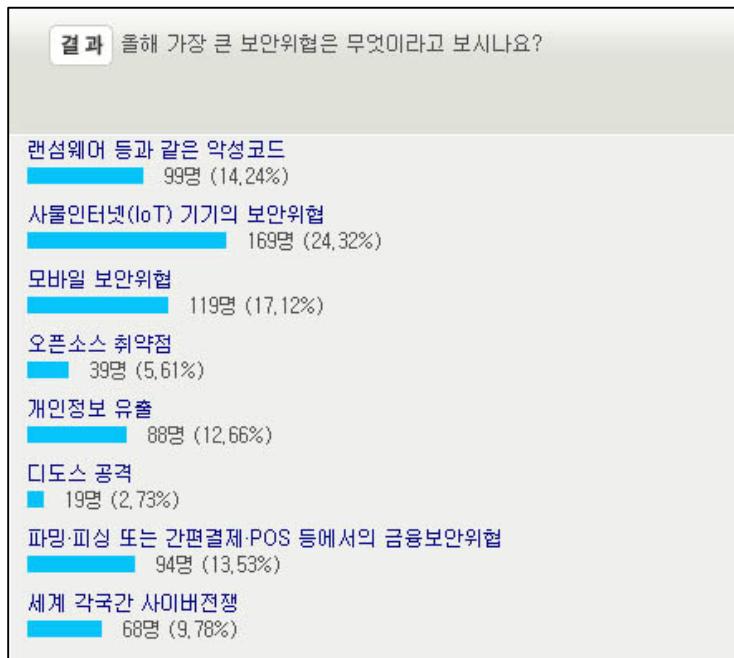


[그림 2] 국내외 IoT 시장 전망, 한국경제연구원

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

¹ [1] 연합뉴스 "한국 사물인터넷시장, 2020년 17조...연평균 38% 성장 전망"

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24



[그림 3] 보안뉴스 기사의 설문조사²

Akamai 위협 연구팀은 최근 인터넷에 연결된 수백만 대의 사물인터넷(IoT) 기기가 웹 기반 자격 증명 스타팅 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.

² [2] 보안뉴스 [설문조사] 올해 가장 큰 보안위협은 '사물인터넷'



	Aidra	Darilloz	Gafgyt	Mirai	Pnscan	Total
2012	36	-	-	-	-	36
2013	19	7	-	-	-	26
2014	98	28	222	-	-	348
2015	87	9	980	-	104	1,180
2016	269	7	8,635	138	76	9,125
Total	509	51	9,837	138	180	10,715

[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

1.2.2 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.

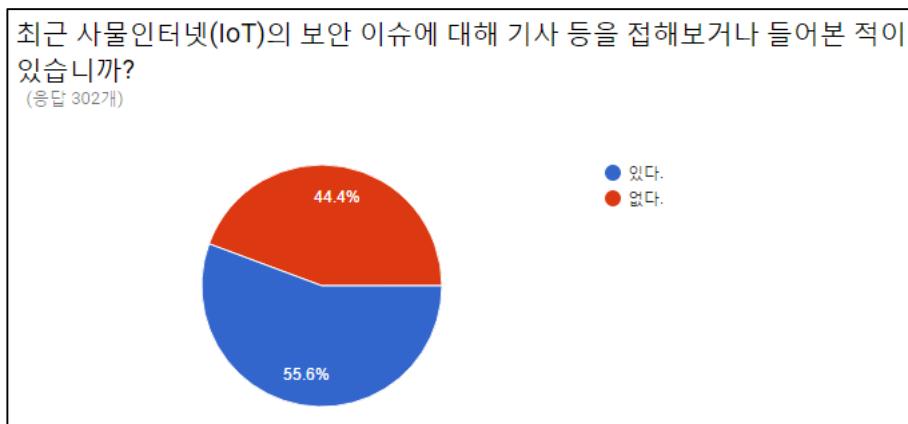
따라서 일반적인 사용자에게는 '실행 파일만 가지고 자신이 소유하고 있는 IoT 기기가 취약한지 한눈에 알려주는 서비스'가 필요하다.

³ [3] Enews IoT 노리는 악성코드 1만개 넘었다

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24

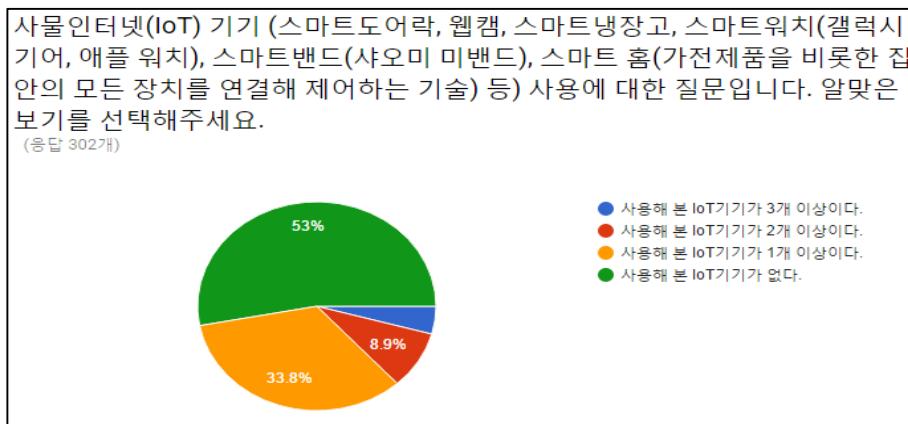
1.2.3 설문조사를 통해 알아본 본 프로젝트의 필요성

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



[그림 5] 설문조사 질문1에 대한 응답 결과

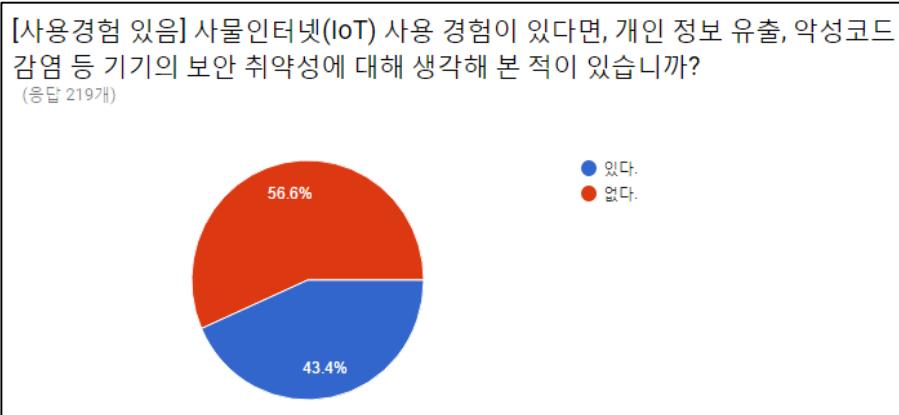
최근 IoT보안에 대한 뉴스 속보와 기사 등을 통해 보안 이슈를 접해본 사람들은 55.6%이었다. 절반 이상 정도의 사람들이 IoT에 대한 관심과 보안 문제에 대한 인식을 가지고 있음을 알 수 있다.



[그림 6] 설문조사 질문2에 대한 응답 결과

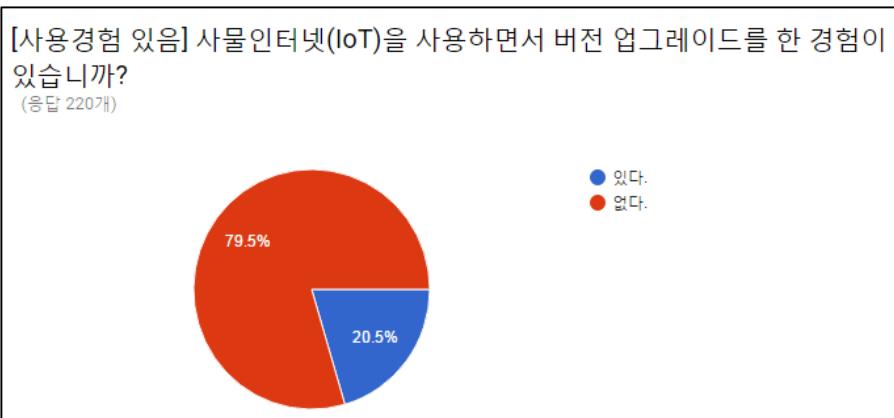


IoT 기기 사용 현황에 대한 질문이다. IoT 시장이 빠르게 성장하고 있고, IoT 기기가 늘어나고 있는 현 실태에 맞게 사용해 본 사람의 수도 적지 않은 숫자로 약 47%에 달했다.



[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.



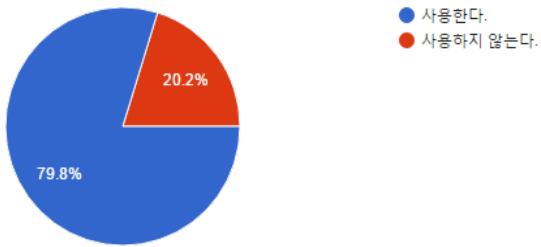
[그림 8] 설문조사 질문4에 대한 응답 결과

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.



만약 자신의 사물인터넷(IoT) 기기의 취약성 정도를 알려주는 서비스가 있다면 사용할 의향이 있습니까?

(응답 302개)



[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.

1.2.4 최근 기술 동향

현재 'BLACKDUCK'사에서 소프트웨어 속 오픈소스를 탐지하는 서비스를 제공하고 있다.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 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 해당 연구가 같은 domain을 지니는 대상에 대해 쉽게 사용가능 하지만 cross-domain일 경우 사용함에 있어 많은

⁴ [4] Blackduck BAT

⁵ [5] IoT Cube

⁶ [6] Tracelet-Based Code Search in Executables



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.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24

2 개발 내용 및 결과물

2.1 목표

본 프로젝트의 목표는 사용자의 IoT 기기의 펌웨어에 대해 소스코드가 존재하지 않는 환경에서 바이너리 코드를 분석 비교하여 취약점을 가진 오픈소스 코드가 사용되었는지 확인하고 진단 결과를 보여주는 웹 서비스를 개발하는 것이다. 웹 서비스의 사용자는 'Power User'을 대상으로 한다. 여기서 'Power User'는 IoT 기기의 펌웨어를 이용하는 보안기관이나 IoT 소프트웨어의 개발자, 또는 IT분야에 대한 지식이 충분한 자를 의미한다. 개발 목표는 크게 다음 3가지이다.

첫째, 대부분의 사용자는 본인이 사용하는 전자 제품, 센서 등이 IoT 기기에 대해 보안 위험성에 대한 인식이 부족하다. "당신의 기기는 안전한가요?"라는 문구를 토대로 사용자들에게 스스로 기기의 보안 취약점을 자각할 수 있도록 도와주는 서비스를 개발한다.

둘째, 많은 IoT 기기 개발자들은 개발에 오픈소스 코드를 많이 이용하고 있다. 하지만 사용한 오픈소스에 대한 보안성을 확실하게 보장할 수 없고, 실제로 취약점이 계속해서 알려진 상황이다. 본 서비스는 개발자들이 개발 단계에서 이 서비스를 이용하여 안전성이 높은 개발을 할 수 있도록 오픈소스의 취약 정도를 쉽게 보여준다.

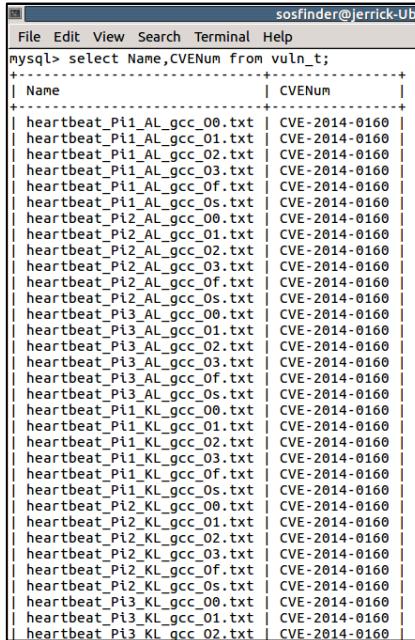
셋째, IoT 기기의 시스템 위에 모듈이 올려지는 방화벽과 같은 형태의 보안이 아닌, 웹 사이트에서 보안을 점검하는 형태의 보안 서비스를 개발한다. 웹 형태로 서비스를 제안하는 이유는 다음과 같다. 첫째, 사용자가 쉽게 서비스를 사용할 수 있으므로 접근성이 훌륭하기 때문이다. 둘째, 프로젝트의 확장이 용이하다는 웹의 특징 때문이다. 사용자로부터 취약한 펌웨어를 게시판에 업로드하게 함으로써 하나의 커뮤니티를 형성하는데, 이를 통해 다양한 환경의 펌웨어를 점검하는 모듈로 확장시킬 수 있다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24

2.2 연구/개발 내용 및 결과물

2.2.1 연구/개발 내용

2.2.1.1 취약점 데이터베이스 구성



```

sosfinder@jerrick-Ub: ~
File Edit View Search Terminal Help
mysql> select Name,CVENum from vuln_t;
+-----+-----+
| Name | CVENum |
+-----+-----+
| heartbeat_Pi1_AL_gcc_00.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_01.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_02.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_03.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_0f.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_0s.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_00.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_01.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_02.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_03.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_0f.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_0s.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_00.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_01.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_02.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_03.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_0f.txt | CVE-2014-0160 |
| heartbeat_Pi1_AL_gcc_0s.txt | CVE-2014-0160 |
| heartbeat_Pi1_KL_gcc_00.txt | CVE-2014-0160 |
| heartbeat_Pi1_KL_gcc_01.txt | CVE-2014-0160 |
| heartbeat_Pi1_KL_gcc_02.txt | CVE-2014-0160 |
| heartbeat_Pi1_KL_gcc_03.txt | CVE-2014-0160 |
| heartbeat_Pi1_KL_gcc_0f.txt | CVE-2014-0160 |
| heartbeat_Pi1_KL_gcc_0s.txt | CVE-2014-0160 |
| heartbeat_Pi2_AL_gcc_00.txt | CVE-2014-0160 |
| heartbeat_Pi2_AL_gcc_01.txt | CVE-2014-0160 |
| heartbeat_Pi2_AL_gcc_02.txt | CVE-2014-0160 |
| heartbeat_Pi2_AL_gcc_03.txt | CVE-2014-0160 |
| heartbeat_Pi2_AL_gcc_0f.txt | CVE-2014-0160 |
| heartbeat_Pi2_AL_gcc_0s.txt | CVE-2014-0160 |
| heartbeat_Pi2_AL_gcc_00.txt | CVE-2014-0160 |
| heartbeat_Pi2_AL_gcc_01.txt | CVE-2014-0160 |
| heartbeat_Pi2_AL_gcc_02.txt | CVE-2014-0160 |
| heartbeat_Pi2_AL_gcc_03.txt | CVE-2014-0160 |
| heartbeat_Pi2_AL_gcc_0f.txt | CVE-2014-0160 |
| heartbeat_Pi2_AL_gcc_0s.txt | CVE-2014-0160 |
| heartbeat_Pi2_KL_gcc_00.txt | CVE-2014-0160 |
| heartbeat_Pi2_KL_gcc_01.txt | CVE-2014-0160 |
| heartbeat_Pi2_KL_gcc_02.txt | CVE-2014-0160 |
| heartbeat_Pi2_KL_gcc_03.txt | CVE-2014-0160 |
| heartbeat_Pi2_KL_gcc_0f.txt | CVE-2014-0160 |
| heartbeat_Pi2_KL_gcc_0s.txt | CVE-2014-0160 |
| heartbeat_Pi3_AL_gcc_00.txt | CVE-2014-0160 |
| heartbeat_Pi3_AL_gcc_01.txt | CVE-2014-0160 |
| heartbeat_Pi3_AL_gcc_02.txt | CVE-2014-0160 |
| heartbeat_Pi3_AL_gcc_03.txt | CVE-2014-0160 |
| heartbeat_Pi3_AL_gcc_0f.txt | CVE-2014-0160 |
| heartbeat_Pi3_AL_gcc_0s.txt | CVE-2014-0160 |
| heartbeat_Pi3_KL_gcc_00.txt | CVE-2014-0160 |
| heartbeat_Pi3_KL_gcc_01.txt | CVE-2014-0160 |
| heartbeat_Pi3_KL_gcc_02.txt | CVE-2014-0160 |
| heartbeat_Pi3_KL_gcc_03.txt | CVE-2014-0160 |
| heartbeat_Pi3_KL_gcc_0f.txt | CVE-2014-0160 |
| heartbeat_Pi3_KL_gcc_0s.txt | CVE-2014-0160 |

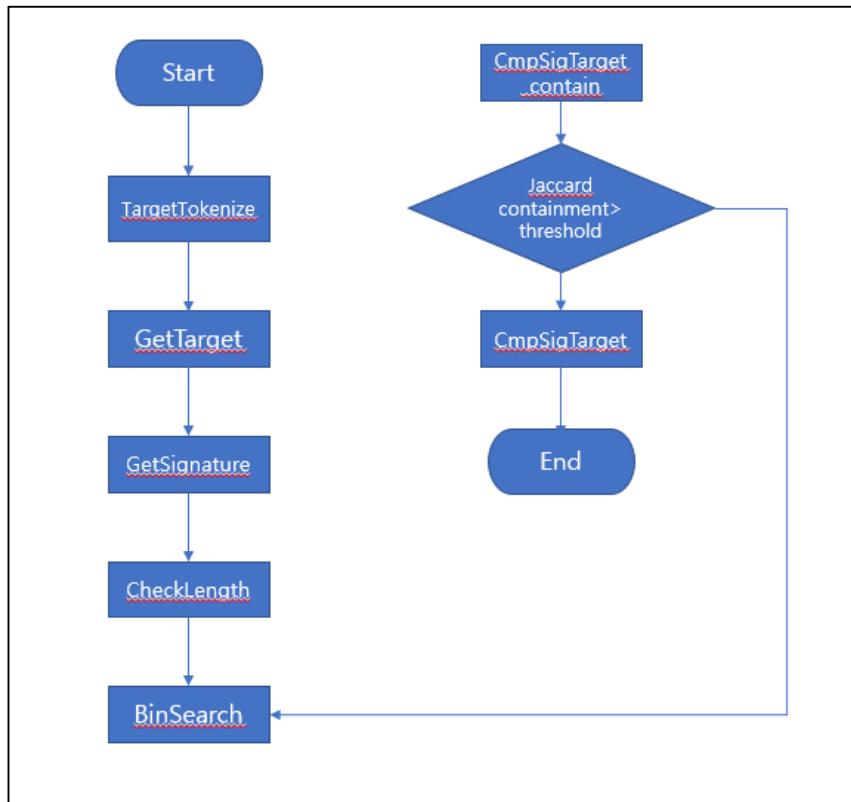
```

[그림 10] 취약점 데이터베이스 구성의 일부 캡처 화면

SOSfinder는 현재 3개의 오픈소스 취약점에 대한 정보를 보유하고 있다. 3개의 취약점은 다음과 같다. OpenSSL-1.0.1f의 'dtls1_process_heartbeat' 함수, Freetype-2.5.2의 'cf2_hintmap_build' 함수, OpenSSH-6.4의 'hash_buffer' 함수이다. 사용된 플랫폼은 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. 따라서 취약점 당 총 54개의 서로 다른 플랫폼에 해당하는 정보를 가지도록 데이터베이스를 구성하였다. 결과적으로, 현재 취약점 데이터베이스에는 162개의 어셈블리 정보가 포함되어 있다.



2.2.1.2 취약점 진단 알고리즘 모듈 개발



[그림 11] 취약점 진단 알고리즘 모듈 Flow Chart

위 그림은 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 파일을 데이터베이스속에 저장되어 있는 시그니처파일과 비교를 시작한다. 우선, 데이터베이스에서 시그니처정보를 가져와 구조체에 저장한다. Tokenize한 타겟 파일의 길이를 확인하고, 비교를 할 시그니처의 길이를 비교한 후 윈도우의 사이즈를 정한다. Tokenize한 타겟을 반으로 나눈 다음 Jaccard containment값을 계산한다. 기준으로 설정한 Jaccard containment값을 넘어갈 경우, 다시 타겟의 크기를 반으로 나눈 다음 Jaccard containment값을 계산하는 것을 반복한다. 이 값이 기준 아래로 판단될 경우 Jaccard index를 계산한다. Jaccard index값을 계산한 것 중 최댓값을 도출한다. 이후, Jaccard index의 최댓값을 처음 설정한 취약도 범위에 따라 취약성의 정도를 판단한다.

2.2.1.3 모듈 테스트

본 프로젝트에서는 비교대상(타겟파일과 시그니처파일) 사이의 많은 크기 차이가 있다. 취약점 데이터베이스 속에 구성된 시그니처는 80~2,600의 라인 수를 가진다. 테스트를 위해 구성한 간단한 실행파일의 경우 500,000~5,000,000줄의 라인 수를 가지고 있다. 크기가 다른 두 파일의 올바른 비교를 위하여 슬라이딩 윈도우 방식을 사용한다. 슬라이딩을 진행하는 방식에 따라 수행 속도 및 정확성이 달라진다. 프로젝트를 위해 처음 사용한 기본 슬라이딩 방식은 1라인 씩 진행하

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24

며, 각각의 윈도우를 집합으로 구성하고 시그니처와 자카드 인덱스(집합 유사도 비교)를 통해 두 스트리밍 데이터 간 유사도를 비교하는 방법이다. 해당 방식에 따르면 $O(n)$ 의 복잡도로 윈도우를 구성하게 되며, 데이터베이스 속의 각각의 시그니처에 대해 평균 약 1800초를 소요하였다. 하지만 본 프로젝트에서 개발하는 서비스의 특성상, 모듈의 실행 속도가 빨라야 하므로 비교 알고리즘의 개선이 필요하였다.

따라서 2.2.1.2에서 설명한 취약점 진단 모듈의 알고리즘을 사용하여 $O(\log n)$ 의 복잡도로 윈도우를 구성할 수 있다. 이를 의사코드로 나타낸 경우 다음과 같다.

```

function bin_filter

while chunk_size > 2 * signature_length

  If Jaccard_CContainment(chunk,signature) > 0.6
    bin_filter(first part of half_of_chunk)
    bin_filter(second part of half_of_chunk)

  Else
    except from detecting

  Else
    Jaccard_Index(chunk, 시그니처) take max:
  
```

데이터베이스의 일부(openSSL속 Hearbeat 취약점 36개, 평균 2600 라인)와 openSSL로 구성된 대상 파일(약 41만 라인)을 비교 성능을 실험을 진행한 경우, 다음과 같은 결과를 얻을 수 있다.



이름	라인수	걸린 시간	자카르드 인덱스	이름	라인수	걸린 시간	자카르드 인덱스
Pi3_AI_O0_heartbeat_101f.txt	3743	2664.11	0.00628077	Pi3_AI_O0_heartbeat_101f.txt	3743	0.0520024	0
Pi3_AI_O0_heartbeat_101g.txt	3760	2680.46	0.00622686	Pi3_AI_O0_heartbeat_101g.txt	3760	0.0560996	0
Pi3_AI_O1_heartbeat_101f.txt	2216	1576.77	0.0373979	Pi3_AI_O1_heartbeat_101f.txt	2216	0.0558727	0
Pi3_AI_O1_heartbeat_101g.txt	2227	1584.91	0.0373271	Pi3_AI_O1_heartbeat_101g.txt	2227	0.0559616	0
Pi3_AI_O2_heartbeat_101f.txt	2145	1534.61	0.357769	Pi3_AI_O2_heartbeat_101f.txt	2145	0.0560073	0
Pi3_AI_O2_heartbeat_101g.txt	2151	1539.59	0.338043	Pi3_AI_O2_heartbeat_101g.txt	2151	0.056011	0
Pi3_AI_O3_heartbeat_101f.txt	2673	1962.7	0.939542	Pi3_AI_O3_heartbeat_101f.txt	2673	34.1288	0.939542
Pi3_AI_O3_heartbeat_101g.txt	2679	1967.35	0.9036	Pi3_AI_O3_heartbeat_101g.txt	2679	34.2785	0.9036
Pi3_AI_Of_heartbeat_101f.txt	2673	1962.3	0.939542	Pi3_AI_Of_heartbeat_101f.txt	2673	34.1132	0.939542
Pi3_AI_Of_heartbeat_101g.txt	2679	1967.14	0.9036	Pi3_AI_Of_heartbeat_101g.txt	2679	34.2387	0.9036
Pi3_AI_Os_heartbeat_101f.txt	1965	1406.12	0.0448549	Pi3_AI_Os_heartbeat_101f.txt	1965	0.0559838	0
Pi3_AI_Os_heartbeat_101g.txt	1973	1413.66	0.0413512	Pi3_AI_Os_heartbeat_101g.txt	1973	0.048979	0
Pi3_Rb_O0_heartbeat_101f.txt	3743	2650.43	0.00628077	Pi3_Rb_O0_heartbeat_101f.txt	3743	0.0531437	0
Pi3_Rb_O0_heartbeat_101g.txt	3760	2655.17	0.00622686	Pi3_Rb_O0_heartbeat_101g.txt	3760	0.0530175	0
Pi3_Rb_O1_heartbeat_101f.txt	2216	1565.2	0.0373979	Pi3_Rb_O1_heartbeat_101f.txt	2216	0.0530188	0
Pi3_Rb_O1_heartbeat_101g.txt	2227	1572.26	0.0373271	Pi3_Rb_O1_heartbeat_101g.txt	2227	0.0529921	0
Pi3_Rb_O2_heartbeat_101f.txt	2145	1520.27	0.357769	Pi3_Rb_O2_heartbeat_101f.txt	2145	0.0531	0
Pi3_Rb_O2_heartbeat_101g.txt	2151	1525.35	0.338043	Pi3_Rb_O2_heartbeat_101g.txt	2151	0.0531285	0
Pi3_Rb_O3_heartbeat_101f.txt	2673	1940.61	0.939542	Pi3_Rb_O3_heartbeat_101f.txt	2673	34.5297	0.939542
Pi3_Rb_O3_heartbeat_101g.txt	2679	1946.01	0.9036	Pi3_Rb_O3_heartbeat_101g.txt	2679	34.592	0.9036
Pi3_Rb_Of_heartbeat_101f.txt	2673	1940.66	0.939542	Pi3_Rb_Of_heartbeat_101f.txt	2673	34.5001	0.939542
Pi3_Rb_Of_heartbeat_101g.txt	2679	1955.88	0.9036	Pi3_Rb_Of_heartbeat_101g.txt	2679	34.5943	0.9036
Pi3_Rb_Os_heartbeat_101f.txt	1965	1391.03	0.0448549	Pi3_Rb_Os_heartbeat_101f.txt	1965	0.0531403	0
Pi3_Rb_Os_heartbeat_101g.txt	1973	1426.43	0.0413512	Pi3_Rb_Os_heartbeat_101g.txt	1973	0.0526502	0
Pi3_Kl_O0_heartbeat_101f.txt	3789	2651.89	0.00426257	Pi3_Kl_O0_heartbeat_101f.txt	3789	0.0525519	0
Pi3_Kl_O0_heartbeat_101g.txt	3806	2663.58	0.00423227	Pi3_Kl_O0_heartbeat_101g.txt	3806	0.0525539	0
Pi3_Kl_O1_heartbeat_101f.txt	2186	1535.44	0.0140957	Pi3_Kl_O1_heartbeat_101f.txt	2186	0.0526914	0
Pi3_Kl_O1_heartbeat_101g.txt	2200	1547.58	0.0142255	Pi3_Kl_O1_heartbeat_101g.txt	2200	0.0528228	0
Pi3_Kl_O2_heartbeat_101f.txt	2150	1538.59	0.066722	Pi3_Kl_O2_heartbeat_101f.txt	2150	0.0523128	0
Pi3_Kl_O2_heartbeat_101g.txt	2154	1539.19	0.0657096	Pi3_Kl_O2_heartbeat_101g.txt	2154	0.0524754	0
Pi3_Kl_O3_heartbeat_101f.txt	2672	1950.47	0.123132	Pi3_Kl_O3_heartbeat_101f.txt	2672	0.0526536	0
Pi3_Kl_O3_heartbeat_101g.txt	2676	1954.16	0.121363	Pi3_Kl_O3_heartbeat_101g.txt	2676	0.0524544	0
Pi3_Kl_Of_heartbeat_101f.txt	2672	1950.49	0.123132	Pi3_Kl_Of_heartbeat_101f.txt	2672	0.0524327	0
Pi3_Kl_Of_heartbeat_101g.txt	2676	1954.1	0.121363	Pi3_Kl_Of_heartbeat_101g.txt	2676	0.0524293	0
Pi3_Kl_Os_heartbeat_101f.txt	1922	1352.15	0.0199706	Pi3_Kl_Os_heartbeat_101f.txt	1922	0.0523206	0
Pi3_Kl_Os_heartbeat_101g.txt	1930	1367.85	0.0198772	Pi3_Kl_Os_heartbeat_101g.txt	1930	0.052294	0
	2572.25	1843.18083			2572.25	7.6796223	

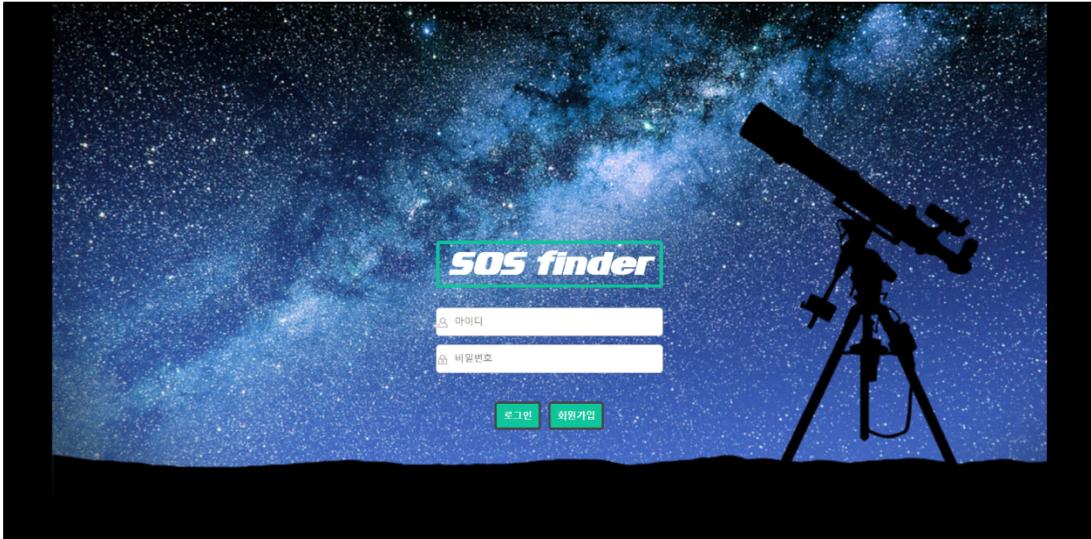
[그림 12] 알고리즘 모듈 테스트 결과

위의 결과와 같이 유사한 대상에 대해서는 처음 방식(왼쪽 데이터)과 같은 결과를 보이고 이 외의 대상에 대해서는 평균 0.05초로 빠르게 제외하는 것을 볼 수 있다. 평균적으로 각각의 시그니처마다 약 7.7초로 약 200배 이상의 속도 향상을 보인다. 추가적으로 서비스가 제공이 되고 확장되면서 데이터베이스를 구성하는 종류가 많아 질수록 제외하는 대상이 늘어날 것으로 예상이 되며, 평균 속도는 증가할 것으로 보인다.

2.2.1.4 웹 서비스 개발 (전체 기능)

1) 로그인 및 회원가입

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24



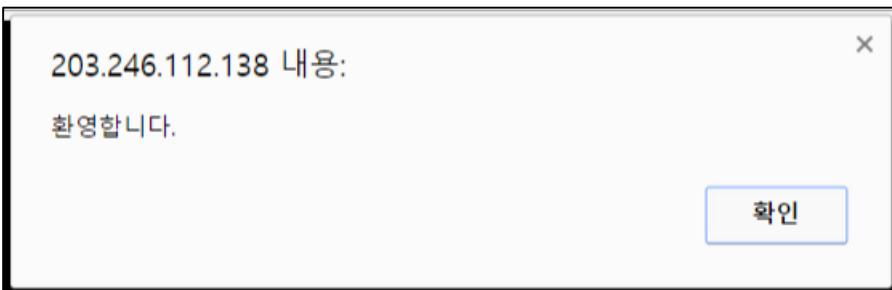
[그림 13] 로그인 페이지

SOS finder는 회원가입과 로그인을 통해 사용할 수 있는 IoT 기기 취약점 탐지 보안 웹 페이지로써 이하와 같은 절차에 맞춰 운영되고 있다.

[그림 14] 로그인 예시

위 그림은 테스트로 만든 사용자용 아이디와 비밀번호(test / 1234)가 입력된 예시이다. 사용자가 가입한 아이디와 비밀번호를 입력한 후 로그인 버튼을 누른다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24



[그림 15] 로그인 확인 메시지

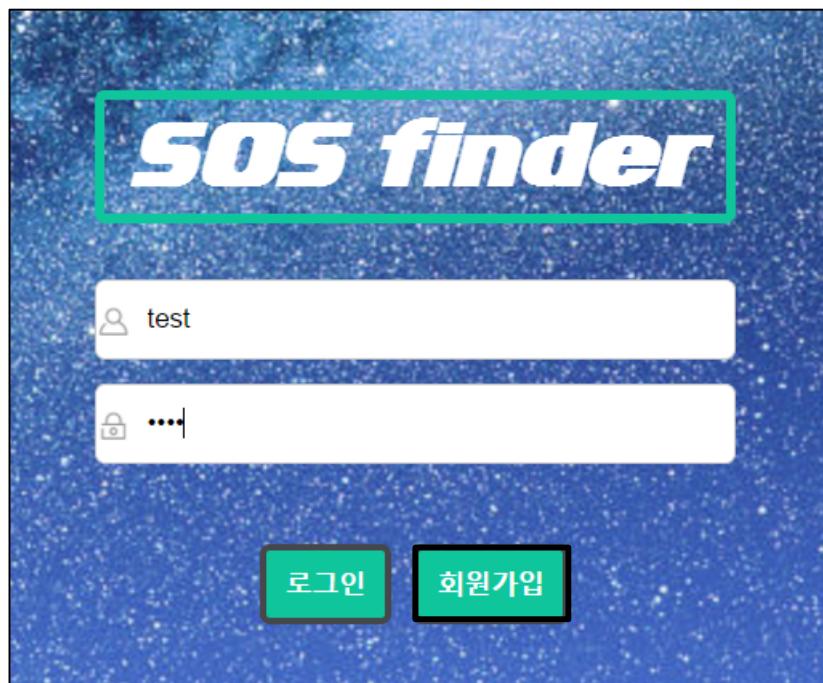
정확한 사용자 정보를 입력하였다면, 위와 같이 확인 메세지가 나오며 메인 홈페이지로 들어갈 수 있다.



[그림 16] 로그인 오류 메시지

위와 같이 잘못된 아이디나 비밀번호를 입력하면 오류 메시지가 출력된다. 회원가입과 로그인 시 주의 하도록 한다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24



[그림 17] 회원가입

SOS finder 웹 서비스를 사용하기 위해 로그인을 해야 하기 때문에 회원 등록을 하지 않았다면 위와 같이 회원가입 버튼을 누른다.

[그림 18] 회원가입 페이지

위 그림은 회원가입 페이지로써 SOS finder에서 사용할 회원 정보를 모두 입력 후 회원가입 버튼을 누르면 된다.

[표 1] 회원가입 입력값

항목	설명	유효값
아이디	SOS finder에서 사용할 아이디	
이름	가입하는 사용자의 성명	

 국립대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24

비밀번호	SOS finder에서 설정할 비밀번호
비밀번호 확인	설정한 비밀번호 재입력

2) 메인 홈페이지



[그림 19] 주메뉴

[표 2] 주 메뉴 설명

구분	설명
About	SOS finder에 대한 서비스 소개 및 SOFA 팀원 소개
SOS finder	사용자의 IoT 기기에 대한 실행파일을 업로드 및 탐지된 취약점 가시화
Board	취약점 DB 구성의 확장성을 위한 취약점 코드 게시판
User Infor	사용자 비밀번호 변경 및 취약점 코드 건의 개수에 따라 부여된 등급 확인
Logout	SOS finder 웹 서비스에서 로그아웃



[그림 20] 홈 헤더

SOS finder의 자세한 내용을 알고 싶다면 Learn More 버튼을 누르면 서비스 소개 메뉴로 넘어간다.



[그림 21] SOS finder 키워드 설명

		결과보고서		
국민대학교 컴퓨터공학부 캡스톤 디자인 I		프로젝트 명	SOSfinder	
팀 명		SOFA		
Confidential Restricted		Version 0.1		2017-5-24

SOS finder의 차별화

'B'사의 'P'제품

- 'B'사의 'P'제품은 소프트웨어 속 오픈소스를 탐지하는 서비스를 제공하고 있습니다. 하지만 취약점 탐지가 아닌 라이선스에 관한 법적 문제를 해결하는 데에 중점을 두고 있습니다.

'K'대 'I'서비스

- 'K'대 'I'서비스는 바이너리 기반이 아닌 소스 코드를 분석해 IoT 기기의 취약점을 검사를 진행 하긴 하나, 사용자 입장에서 소스코드를 인지하지 못하는 경우가 대다이며, 소프트웨어를 구성하는 소스코드를 구하는 것은 쉽지 않은 일입니다.

SOS finder의 차별화

- SOS finder는 IoT기기의 펌웨어를 이용하는 보안 기관 같은 곳이나, 파워유저를 중심 대상으로 서비스를 시작합니다. 더 나아가 일반 사용자를 대상으로는 매뉴얼을 제공하여 누구나 쉽게 자신의 기기를 점검하고 취약한 정도를 다시 보드를 통해 가시화하여 보여줍니다. 그리하여 자신의 기기에 맞는 펌웨어를 업데이트하는 등 적절한 조치를 취할 수 있도록 권고합니다.

The screenshot shows a comparison between 'B'사의 'P'제품 and 'K'대 'I'서비스. It highlights that SOS finder uses source code analysis for IoT devices, unlike B's product which focuses on license issues. SOS finder also provides a user-friendly interface for end-users to check their own devices and receive recommendations for updates.

[그림 22] SOS finder의 차별화 설명

SOS finder와 유사한 타사 제품 및 서비스에 대해 장단점 비교를 통해 본 프로젝트에 대한 차별화된 전략을 설명하고 있다.

3) About 메뉴

About 메뉴에는 서비스 소개와 팀원 소개 두 가지로 되어있다. 좌측에 사이드바 메뉴를 클릭해서 볼 수 있다. 서비스 소개에서는 서비스 추진 배경, 목표, 차별화, 기대효과를 설명하고 있다. 팀원 소개에는 SOFA 팀원의 역할 분담을 확인할 수 있다.

Service 서비스 소개

Home / Service

서비스 소개

팀원 소개

서비스 추진 배경

지금은 제 4차 산업혁명의 시대

제 4차 산업혁명의 시대에 주목해야 할 기술로 단연 IoT를 꼽을 수 있을 것입니다. 따라서 저희 SOFA는 사용자가 자신의 IoT 기기의 보안 상태에 대해 경각심을 가질 수 있도록 기기의 취약성을 진단하는 서비스를 제공하려 합니다.

IoT기기 사용 현황 관련 설문조사

본 서비스를 개발 하기위해 앞서 사람들의 IoT 사용 현황에 대해 설문조사를 진행했습니다. SNS를 통해 설문조사를 실시하였고, 대상자는 IT 지식이 있는 사람부터 관련 지식이 전혀 없는 사람까지 포괄적이었습니다. 5일 동안 총 302명이 응답하였습니다.

Survey One Survey Two Survey Three Survey Four Survey Five

최근 사물인터넷(IoT)의 부작용에 대해 기사 등을 전해보거나 들어본 적이

[그림 23] 서비스 소개

 국립대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24

Our Team 팀원 소개

Home / Our Team

서비스 소개
팀원 소개

Our Team



송창현

Song

- 전체 프로젝트 총괄 및 설계
- 유사도 비교 알고리즘 구현

750×450

차진원

Cha

- 유사도 비교 알고리즘 설계
- 알고리즘 성능 테스트

750×450

정성민

Jung

- 웹 서비스 구현 (Software 업로드 및 알고리즘 연동)
- 가시화 구현 (Dashboard)

[그림 24] SOFA 팀원 소개

4) SOS finder 메뉴

SOS finder

Home / SOS finder

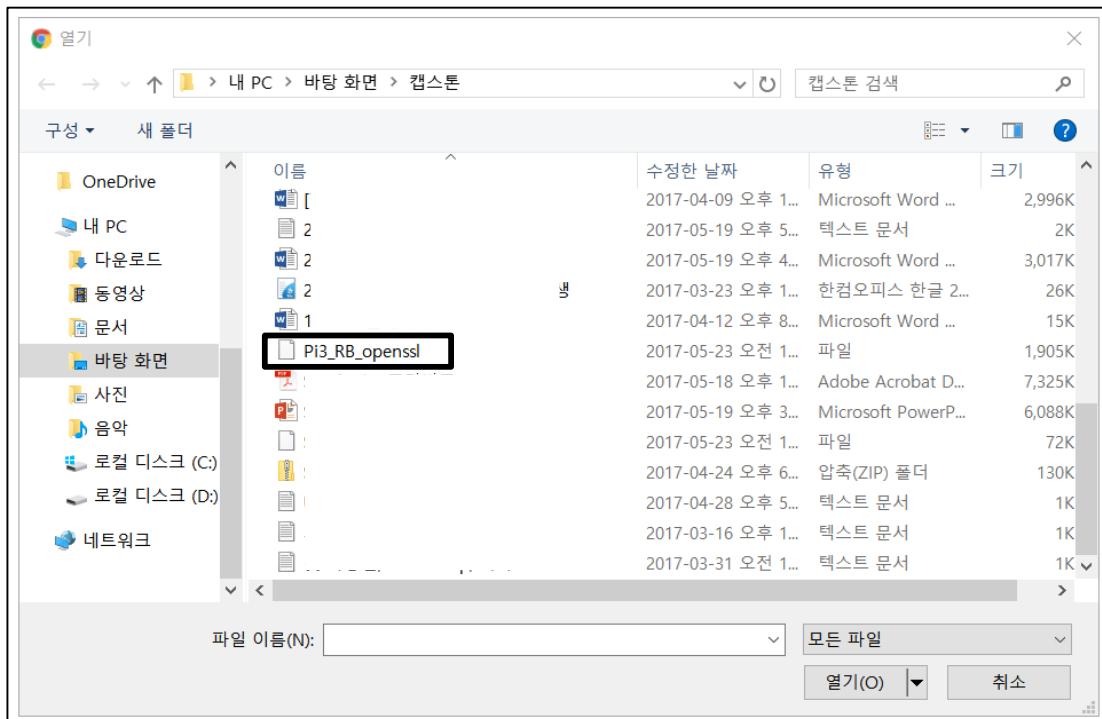
파일 선택 선택된 파일 없음

UPLOAD

[그림 25] 파일 업로드 페이지

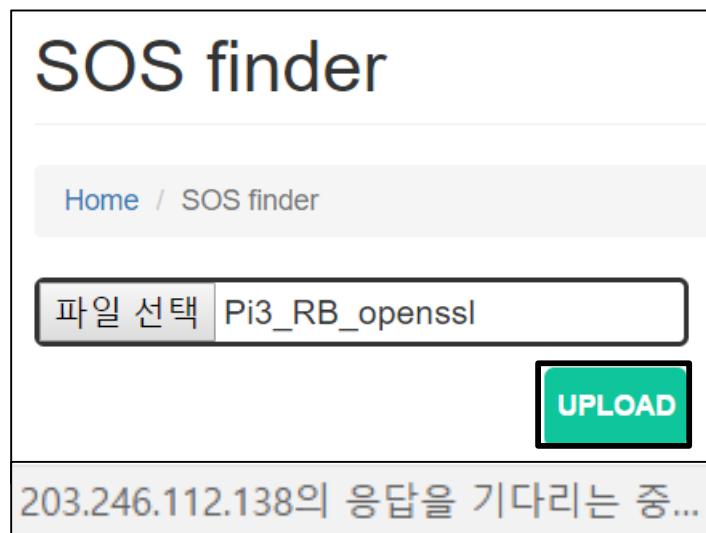
사용자는 자신이 점검하고자 하는 IoT 기기에 대한 펌웨어(실행파일)를 업로드 하기 위해, 파일 선택 버튼을 누른다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24



[그림 26] 파일 선택 화면

위 그림에서 업로드 할 실행 파일은 Raspberry PI 3 기기의 Raspbian 운영체제를 사용하는 환경에서 openSSL 취약점을 가진 테스트 모듈 파일이다.

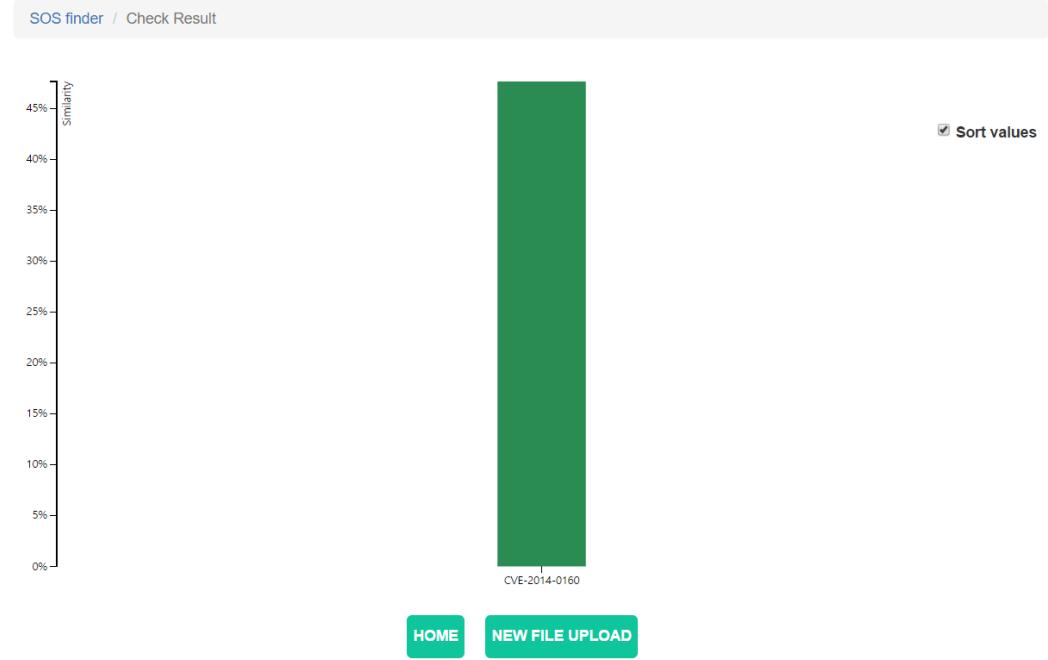


[그림 27] 업로드 진행 중

위와 같이 파일을 선택한 후 업로드 버튼을 누르면 하단에 주소 상태 표시줄에서 나타나는 것과 같이, 서버로부터 모듈에 대한 응답을 기다린다.

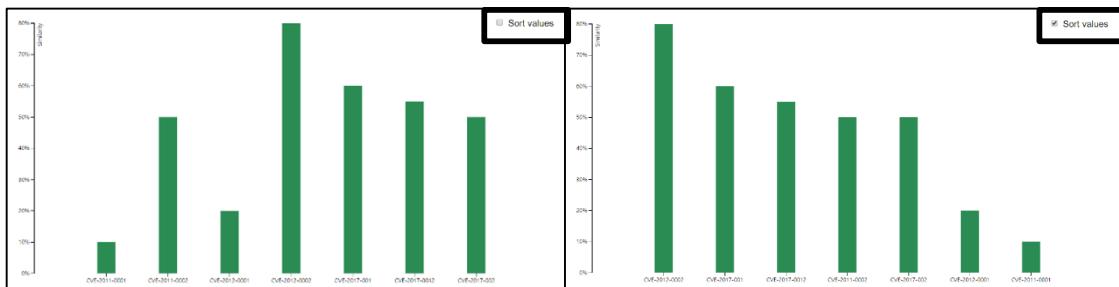
 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24

Check Result



[그림 28] 점검 결과 가시화

SOSfinder의 진단 모듈로 실행파일을 진단한 결과를 그래프를 통해 사용자들이 쉽게 볼 수 있도록 가시화 한다. 이로써 사용자는 CVE 번호를 확인하고 유사도 정도를 확인할 수 있다.



[그림 29] 그래프 정렬

그래프 왼쪽 상단에 보면 CVE 번호순이나 가장 취약한 정도 순으로 정렬할 수 있는 체크박스 라벨이 있다.

- 🔴 100~51% : 매우 취약
- 🟡 50~6% : 취약
- 🟢 5~0% : 안전

유사도에 따른 취약 정도

저희 SOFA는 사용자가 자신의 IoT 기기의 보안 상태에 대해 경각심을 가질 수 있도록 기기의 취약성을 진단하는 서비스를 제공하려 합니다. CVE 번호와 유사도를 확인하십시오.

[그림 30] 유사도에 따른 취약 정도

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24

5) User Infor 메뉴

User Profile

Home / Profile

아이디	test
이름	test
Good 게시물 갯수	1

* Good 게시물이란, 커뮤니티 게시판에 올린 사용자의 글 중 관리자가 선택한 게시물을 의미합니다.

[MODIFY PASSWORD](#) [LOGOUT](#)

[그림 31] 사용자 정보

User Infor 메뉴에서 사용자가 회원가입 시 입력한 본인의 회원 정보가 등록되어 나타난다. 게시판에 건의한 취약 코드가 관리자로부터 승인 받았을 때, 'Good 게시물 개수'가 올라간다. 이것으로 사용자 등급을 나누게 된다. 그리고 비밀 번호 변경 버튼과 로그아웃 버튼이 있다. 로그아웃은 상단에 주 메뉴 우측 끝에도 있다.

비밀번호 수정

비밀번호	<input type="password"/>
비밀번호 확인	<input type="password"/>

정보 수정

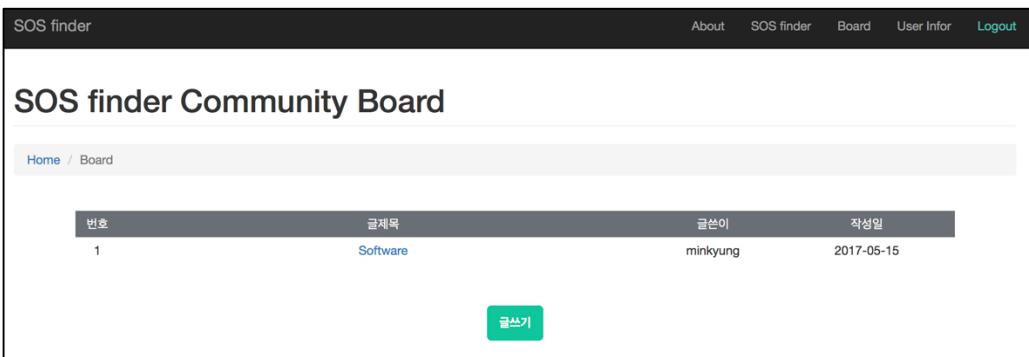
[그림 32] 비밀번호 변경



[그림 33] 로그아웃 확인 메시지

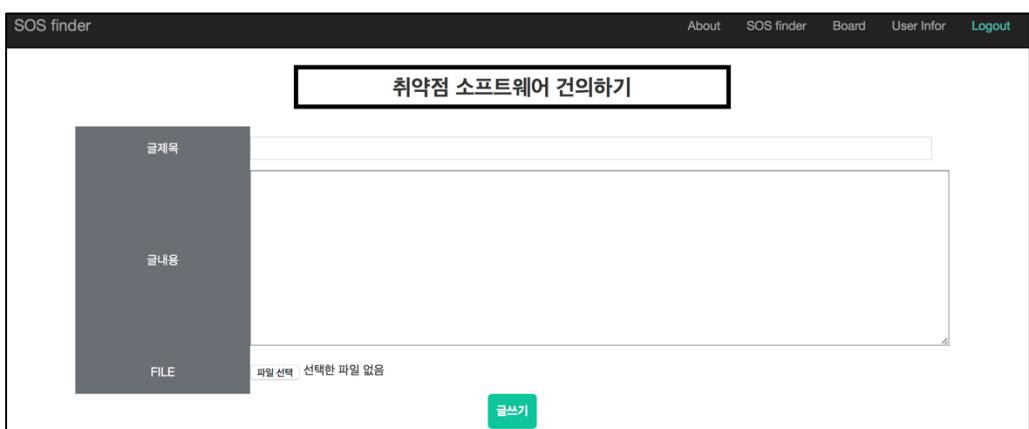
 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24

2.2.1.5 웹 서비스 개발 (취약점 커뮤니티 게시판 기능)



[그림 34] 취약점 건의 게시판 목록 인터페이스

Board 메뉴이다. 사용자는 자신이 진단하고 싶은 오픈소스 취약점에 대해서 건의할 수 있고, 다른 사용자가 건의한 글에 대해 확인할 수 있다. 건의한 글이 적합할 시, 관리자는 게시물을 관리자 데이터베이스에 따로 저장하고, 해당 게시물을 'Good 게시글'로 취급한다. 'Good 게시글'의 수가 많을 수록 사용자의 신용이 올라간다.



[그림 35] 취약점 건의 게시판 글쓰기 인터페이스

위 그림은 취약점 소프트웨어를 건의하기 위한 글쓰기 인터페이스이다. 글의 제목과 내용을 입력하고, 건의할 취약점 실행파일을 업로드한 후 글쓰기 버튼을 누르면 글 등록이 완료된다.

[표 3] 취약점 게시판 글쓰기 입력값

항목	설명
글제목	글의 제목
글내용	건의할 취약점 소프트웨어의 설명
FILE	건의할 취약점 소프트웨어 업로드

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24

2.2.2 시스템 기능 요구사항

프로그램 유형	내용	진행 상황
요구사항	취약점 점검 알고리즘 모듈 Target device의 binary code를 추출할 수 있어야 한다.	완료
	유사도 비교 알고리즘으로 Target device의 정확한 취약점을 찾아내야 한다.	완료

프로그램 유형	내용	진행 상태
요구사항	사용자가 펌웨어 파일을 업로드 할 수 있어야 한다.	완료
	사용자의 펌웨어 파일을 알고리즘을 사용하여 취약점을 점검 할 수 있어야 한다.	완료
	사용자의 기기의 취약점 점검 결과를 사용자에게 알기 쉽게 보여줄 수 있어야 한다.	완료
	권한을 가진 사용자가 직접 취약하다고 판단되는 소프트웨어 를 게시판에 업로드 할 수 있어야 한다.	완료

2.2.3 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
사용성	본 프로젝트의 목적은 IoT 기기 사용자를	완료

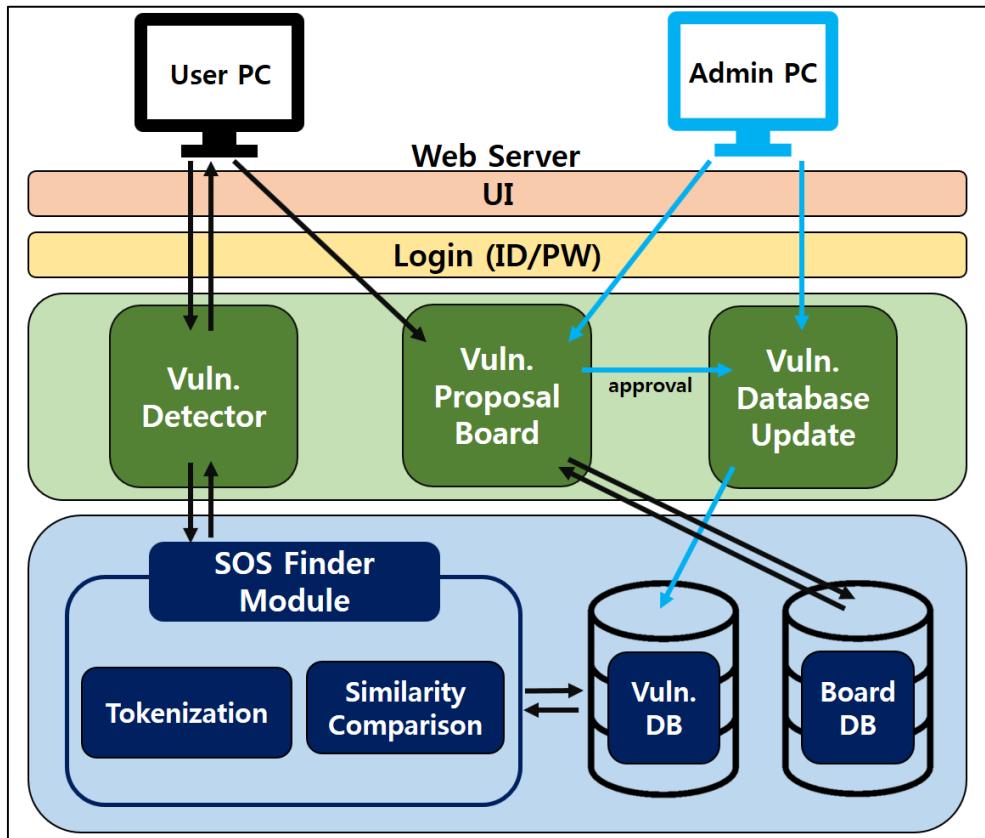


	<p>대상으로 해당 기기의 소프트웨어 속 취약점 존재 여부를 판단하는 것이다. 프로젝트의 사용성은 사용자가 자신의 기기의 취약점에 대해 쉽게 파악할 수 있는 지로 구분할 수 있다. 이를 위해, 사용자가 기기의 실행 파일을 쉽게 업로드 할 수 있게 하고, 게시판을 두어 사용자와 관리자간 취약한 소프트웨어에 대한 의사소통을 가능하도록 한다.</p>	<p>사용성을 위하여 취약점의 결과를 가시화 할 때, 그래프를 사용하여 쉽게 파악할 수 있도록 하였다. 또한, 그래프 좌측에 퍼센트에 따른 취약성 정도를 기록하였다. 사용자와 관리자간의 의사소통을 위해 게시판 메뉴에서 취약점에 대해 건의할 수 있도록 하였다.</p>
재사용 가능성	<p>본 프로젝트의 목적은 IoT기기의 취약점을 탐지하는 것으로, 프로젝트 개발 기간의 제약 상 자카르드 인덱스를 통해 비교를 진행 하나 비교의 정확도를 높이기 위해 다른 알고리즘을 적용할 수 있도록 모듈화한다. 또한 웹 서비스의 구현과 취약점 점검 모듈을 분리하여 구현한다.</p>	<p>완료</p> <p>웹 서비스에서 서버에 존재하는 점검 모듈을 통해 점검을 하고, 결과를 받아오는 형태로 구현하였다. 비교의 정확도와 속도 개선을 위해 사용한 알고리즘에 대해서는 '2.2.1.2 취약점 진단 알고리즘 모듈 개발'에 서술되어 있다.</p>



프로젝트 명	SOSfinder	
팀 명	SOFA	
Confidential Restricted	Version 0.1	2017-5-24

2.2.4 시스템 구조 및 설계도



[그림 36] 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.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	SOSfinder	
	팀 명	SOFA	
	Confidential Restricted	Version 0.1	2017-5-24

2.2.5 활용/개발된 기술

1) IoT 보안 웹 서비스

- 본 프로젝트의 어플리케이션은 Web 으로 설계하였으며 사용자가 사용하기 쉬운 인터페이스를 제공한다.
- 사용자는 로그인을 통해 서비스의 권한을 얻고, 메인 화면에 있는 파일 업로드 기능을 통해 검사 받기를 원하는 파일을 쉽게 업로드한다.
- 업로드 이후 검사 버튼을 누르면 서버의 진단 모듈에서 취약점을 진단하여 결과를 받아온다.
- 그래프를 통해 사용자는 자신이 업로드한 파일이 어느 오픈소스에 의해 취약한지 정보를 얻는다.
- 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

- TargetTokenize: 타겟 파일의 어셈블리 코드를 mnemonic, memory address, register, offset 으로 분리하여 토큰화한다.
- GetSignature: database 에서 미리 저장된 시그니처 리스트를 가져와 비교 대상을 찾는다.
- CheckLength: 타겟 파일의 토큰화된 어셈블리의 길이를 확인하고 선택된 signature 의 길이를 비교하여 window size 를 정한다.
- 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	
	운영체제	Ubuntu, Windows7
	컴파일	



프로젝트 명

SOSfinder

팀 명

SOFA

Confidential Restricted

Version 0.1

2017-5-24

결과물을 확인하는 환경	개발언어	
	언어의 문법적 요구사항	



2.2.6 현실적 제한 요소 및 그 해결 방안

1) 취약점 데이터베이스 구성의 제한

- 취약점에 대한 데이터베이스의 구성을 위해 사용하는 소프트웨어의 다양성이 제한될 수 있다. 본 프로젝트의 시간적 제약 때문이다. 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.



2.2.7 결과물 목록

대분류	소분류	기능	형식	진행 상태	수정 사항
IoT 보안 웹 서비스	레이아웃	사용자에게 편리한 User Interface 를 제공한다.		완료	
	파일 업로드	사용자 기기의 소프트웨어를 업로드한다.	모듈	완료	
	DB 연동, 알고리즘 적용	업로드한 소프트웨어의 정보를 서버에 전달하고, DB 에 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, offset 으로 토큰화한다	모듈	완료	
	GetSignature	Database 에서 signature list 를 가져와 비교 대상을 찾는다.	모듈	완료	
	CheckLength	Target 의 토큰화된 어셈블리의 길이를 확인하고 선택된 signature 의 길이를 비교하여 window size 를 정한다	모듈	완료	
	CmpSigTarget	Signature 와 Target 을 비교하여 유사도를 확인하고 기준치를 넘는 값들의 이름을 가져온다	모듈	완료	



2.3 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 IoT 기기의 실행 파일을 업로드 할 수 있는 지식을 가진 파워 유저를 대상으로 한다. 일반 사용자의 경우, IT 분야의 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 검사항으로써, 자신의 개인정보 유출 위험이 있는지 파악할 수 있다. 따라서 본 서비스는 사용자의 안전한 디바이스 사용을 돋는다.

3) 취약점 관련 커뮤니티 형성

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



3 자기평가

최종 결과물	IoT 보안 웹 서비스
주요 평가 기준	IoT 기기의 들어있는 오픈소스의 취약점을 정확하게 검사할 수 있는가?
자기평가	<p>본 프로젝트는 IoT 기기에 속한 취약점 중 OpenSSL-1.0.1f 의 'dtls1_process_heartbeat' 함수, Freetype-2.5.2 의 'cf2_hintmap_build' 함수, OpenSSH-6.4 의 'hash_buffer' 함수에 대해 정확하게 찾아내고, 취약성 정도를 상세하게 진단한다. 각 취약점에 있어서, 다음과 같은 플랫폼에 대해 진단할 수 있다. 3 개의 라즈베리 파이(Pi1, Pi2, Pi3) 기기, 3 개의 운영체제(Raspbian, Arch Linux, Kali Linux), 4.9.2 버전의 arm-linux-gcc 컴파일러, 그리고 6 개의 최적화 옵션(O0, O1, O2, O3, Ofast, Os)이다.</p> <p>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.</p> <p>결과적으로, 본 SOSfinder 서비스는 IoT 기기의 오픈소스 취약점에 대해 정확하고 상세하게 검사할 수 있고, 앞으로 더 많은 취약점에 대해 점검할 수 있는 확장성을 갖추고 있다.</p>

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	결과보고서				
	프로젝트 명	SOSfinder			
	팀 명	SOFA			
	Confidential Restricted	Version 0.1		2017-5-24	

4 참고 문헌

번호	종류	제목	출처	발행년도	저자
1	기사	http://www.yonhapnews.co.kr/bulletin/2016/07/12/0200000000AKR2016071210930002.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
7	논문	ReDeBug: Finding Unpatched Code Clones in Entire OS Distributions	2012 IEEE Symposium on Security and Privacy	2012.5	Jiyong Jang, Abeer Agrawal, and David Brumley



4.1 테스트 케이스

대분류	소분류	기능	테스트 방법	기대 결과	테스트 결과
웹 서비스	기능		2.1.1.4 웹 서비스 개발(전체 기능) 및 2.1.1.5 웹 서비스 개발 (취약점 커뮤니티 게시판 기능) 참고		성공