문서형
악성코드/바이러스 탐지 프로그램
(Document-type malware/virus detection program)

팀　　명 : 캐치버그핑
지도교수 : 이병천 교수님
팀　　장 : 신명진
팀　　원 : 김태현
최경은
임혜준
김호준

2024. 10

중부대학교 정보보호학과

목 차

1. 서론

1.1 연구 배경 및 필요성

- 개인정보 유출이 빈번해짐에 따라, 한국인터넷진흥원에 따르면 개인정보 침해신고센터에 접수된 개인정보 침해 상담·신고 건수는 2016년 9만 8,210건에서 지난해 17만 7,457건으로 4년 사이 80% 넘게 급증했습니다.
- 이와 같이 지속적으로 증가하고 있는 개인정보 유출 사고는 해마다 급증하고 있으며, 앞으로도 계속 증가할 것으로 보입니다. 이러한 사례들은 단순 실수나 관리 미흡으로 발생하며, 노출된 정보는 보이스피싱 등 금융 범죄에 악용되어 더 큰 피해를 일으킬 수 있습니다. 이러한 사고를 예방하기 위해서는 관리자가 파일의 보안에 대해 경각심을 가지고 있어야 하며, 주기적인 취약점 진단은 필수입니다.

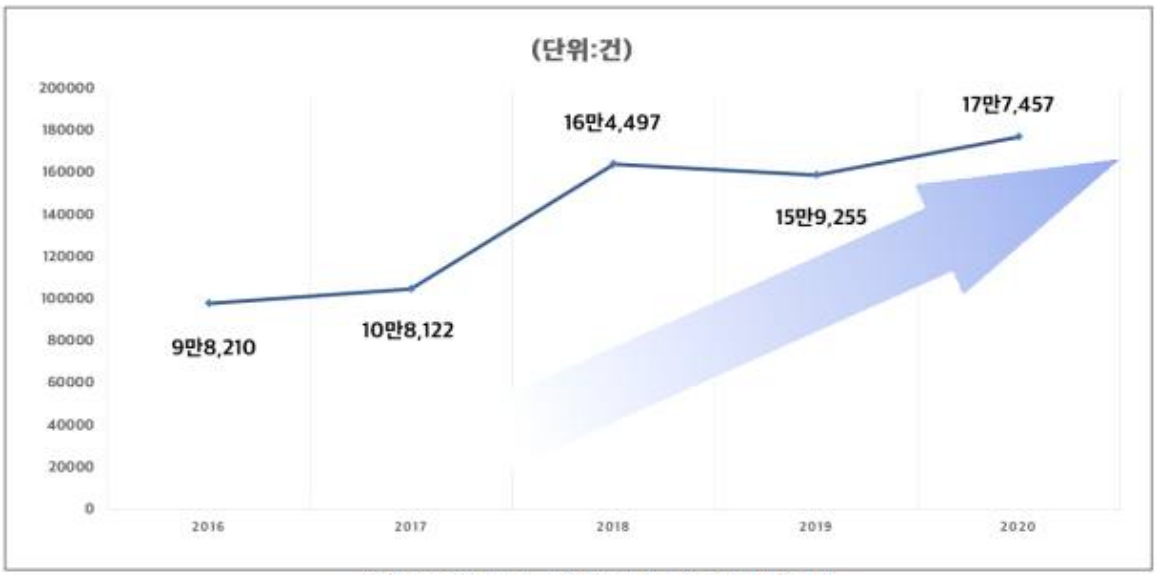


그림 1 개인정보 침해 신고•상담건수 (출처 : 한국인터넷진흥원)

1.2 연구 목적 및 주제 선정

- 최근 웹 취약점을 이용한 공격들이 지속적으로 발생하고 증가하고 있습니다. 웹서버나 웹사이트의 체계적인 보안 관리가 제대로 이루어지지 않은 회사가 다수이고, 웹사이트는 사용자들에게 항상 노출되어 있으며 접근 방법이 매우 쉬워 보안 사고가 끊임없이 발생하고 있습니다.
- 전체 개인정보 유출 피해의 75%가 외부 해킹이 원인이며, 가장 취약한 부분은 웹에서부터 시작됩니다. 이처럼 잦은 보안 사고로 인해 개인정보보호법, 정보통신망법, 전자금융거래법 시행령, 신용정보법 등에서 보안 의무와 처벌이 강화되는 추세입니다.
- 본 연구는 개인이나 기업이 간단하게 웹 진단을 할 수 있는 웹 기반 파일 취약점 자동 진단 도구를 제작하고자 합니다. 이 도구는 취약점 진단 시간을 단축하여 효율적인 진단을 가능하게 하고, 사용자에게 친절한 진단 보고서와 조치 방안을 제공합니다.
- 따라서 이를 통해 웹 기반 파일 취약점 진단 도구를 제작하게 되었습니다.

2. 관련 연구

2.1 언어 / DB

2.1.1 NodeJS

Node 공식 사이트에서는 Node.js를 다음과 같이 설명하고 있습니다.

"Node.js는 Chrome V8 JavaScript 엔진으로 빌드된 JavaScript 런타임입니다."

즉, Node.js를 통해 다양한 자바스크립트 애플리케이션을 실행할 수 있으며, 서버를 실행하는 데 제일 많이 사용됩니다.
- Node.js는 JavaScript를 서버에서도 사용할 수 있도록 만든 프로그램입니다.
- Node.js는 V8이라는 JavaScript 엔진 위에서 동작하는 자바스크립트 런타임(환경)입니다.
- Node.js는 서버사이드 스크립트 언어가 아니라 프로그램(환경)입니다.
- Node.js는 웹 서버와 같은 확장성 있는 네트워크 프로그램을 제작하기 위해 만들어졌습니다.

Node.js는 확장성 있는 네트워크 애플리케이션 개발에 사용되는 소프트웨어 플랫폼입니다. 특히 서버사이드에서 많이 사용되고 있습니다. 사용하는 언어로는 자바스크립트(Javascript)를 활용하며, Non-blocking I/O와 단일 스레드 이벤트 루프를 통한 높은 처리 성능을 가지고 있는 것이 특징입니다.

Node.js는 내장 HTTP 서버 라이브러리를 포함하고 있어, 웹 서버에서 아파치 등의 별도 소프트웨어 없이 동작이 가능하며, 이를 통해 웹 서버의 동작에 있어 더 많은 통제에서 벗어나 여러 가지 기능을 가능하게 합니다.

즉, Node.js를 통해 웹 애플리케이션이 더욱 발전하였으며, 정적인 홈페이지뿐만 아니라 쇼핑몰, 티켓 예매 사이트, 블로그 등 데이터가 변하는 사이트를 만들 수 있습니다. 또한 여러 개발자가 만든 프로그램과 게임을 웹상에서 구동시켜 안드로이드폰, 아이폰, 윈도우PC, 맥 등 플랫폼의 제약에서 벗어나 어디서든 실행할 수 있게 해줍니다.

물론 단순히 웹에서 실행 가능한 게임을 만들려면 JavaScript만으로도 가능하지만, 실시간 온라인 채팅, 실시간 온라인 게임 등 실시간 기능을 넣거나, 로그인 기능을 통해 유저를 관리하고 점수를 관리하는 데이터베이스 기능을 Node.js를 통해 만들 수 있습니다.

Node.js 사용 이유

- Node.js를 사용하려면 먼저 JavaScript를 배워야 합니다.
- Node.js는 JavaScript를 사용하기 위해 만들어졌기 때문입니다.

- JavaScript는 C/C++, Java와 같은 프로그래밍 언어입니다.
하지만 이름에서 알 수 있듯 JavaScript는 독립적인 언어가 아닌 스크립트 언어입니다.
스크립트 언어는 특정한 프로그램 안에서 동작하는 프로그램이기 때문에 웹 브라우저 프로그램 안에서만 동작합니다.

즉, 웹 브라우저(크롬, 사파리, 익스플로러, 파이어폭스 등)가 없으면 사용할 수 없는 프로그램입니다.
여기서 Node.js가 중요한 역할을 합니다.

Node.js는 JavaScript를 웹 브라우저에서 독립시킨 것이며, Node.js를 설치하면 터미널 프로그램(윈도의 cmd, 맥의 terminal 등)에서 Node.js를 입력해 브라우저 없이 바로 실행할 수 있습니다.
문법은 JavaScript와 동일합니다.

Node.js를 이용하여 웹 브라우저와 무관한 프로그램을 만들 수 있습니다.
중요한 것은 Node.js를 이용해 서버를 만들 수 있다는 점입니다.

그 이유는, 이전에는 Server-Client 웹사이트를 만들 때 웹에서 표시되는 부분은 JavaScript로, 서버는 Ruby, Java 등 다른 언어로 만들어야 했지만, 이제는 하나의 언어로 전체 웹페이지를 만들 수 있기 때문입니다.

2.1.2 mongoDB

MongoDB 데이터베이스는 플랫폼 간 오픈 소스 No SQL 데이터베이스 관리 시스템입니다. 유연하고 확장 가능한 방식으로 대량의 데이터를 저장하고 관리하도록 설계되었습니다. MongoDB의 주요 기능 중 하나는 선택적 스키마가 있는 JSON과 같은 문서 형식으로 데이터를 저장하는 문서 지향 데이터 모델입니다. 이를 통해 비용이 많이 드는 데이터 마이그레이션 없이 스키마를 쉽게 수정할 수 있으므로 유연성이 향상되고 개발 시간이 단축됩니다.

기술적 세부 사항 측면에서 MongoDB는 데이터가 여러 서버 또는 샤드에 분할되는 분산 아키텍처를 사용합니다. 이를 통해 수평적 확장이 가능합니다. 즉, 데이터베이스는 값비싼 하드웨어 업그레이드 없이 증가하는 데이터양과 읽기 및 쓰기 워크로드를 처리할 수 있습니다. MongoDB에는 자동 장애 조치 및 복제 세트와 같은 고가용성을 위한 몇 가지 기본 제공 기능도 포함되어 있습니다. 이렇게 하면 하드웨어 장애 또는 기타 중단 중에도 데이터베이스를 계속 사용할 수 있고 액세스할 수 있습니다.

MongoDB는 최신 데이터 기반 애플리케이션에서 대량의 데이터를 저장하고 관리하기 위한 강력하고 유연한 도구입니다. 문서 지향 데이터 모델과 분산 아키텍처는 실시간 분석 및 콘텐츠 관리에서 IoT 및 전자 상거래 애플리케이션에 이르기까지 다양한 사용 사례에 매우 적합합니다.

어떻게 작동합니까?

MongoDB는 JSON 개체와 유사한 문서 형식으로 데이터를 저장하여 작동합니다. 이러한 문서는 기존 관계형 데이터베이스의 테이블과 유사한 컬렉션으로 구성됩니다. 컬렉션의 각 문서는 다른 구조를 가질 수 있으므로 저장할 수 있는 데이터 유형에 더 큰 유연성을 허용합니다. 예를 들어, 한 문서에는 이름, 이메일, 주소 등 사용자에 대한 정보가 포함될 수 있고 다른 문서에는 이름, 가격, 설명 등 제품 관련 정보가 포함될 수 있습니다.

MongoDB에서 데이터를 저장하고 검색하기 위해 개발자는 SQL과 유사한 MongoDB 쿼리 언어를 사용할 수 있습니다. 쿼리를 사용하여 컬렉션 내의 특정 문서를 검색하고, 문서를 업데이트 또는 삭제하고, 기타 데이터 관리 작업을 수행할 수 있습니다.

MongoDB에는 성능을 개선하고 고가용성을 보장하기 위해 데이터를 인덱싱, 샤딩 및 복제하기 위한 다양한 내장 기능도 포함되어 있습니다. 인덱스를 사용하여 컬렉션 내에서 특정 문서를 빠르게 찾을 수 있으며 샤딩을 사용하면 데이터를 여러 서버에 분할하여 수평적 확장이 가능합니다. 반면 복제는 데이터를 항상 사용할 수 있고 오류 발생 시 복구할 수 있도록 여러 서버에 걸쳐 데이터 복사본을 만듭니다.

몽고DB 기능

MongoDB에는 현대적인 데이터 기반 애플리케이션을 구축하는 개발자들 사이에서 널리 선택되는 몇 가지 기능이 있습니다. 주요 기능 중 일부는 다음과 같습니다.

- 문서 지향 데이터 모델 : MongoDB는 데이터를 JSON과 유사한 문서 형식으로 저장하므로 유연성이 향상되고 개발 시간이 단축됩니다.
- 확장성 : MongoDB는 분산 아키텍처를 사용하고 수평적 확장을 지원합니다. 즉, 값비싼 하드웨어 업그레이드 없이 증가하는 데이터양과 읽기 및 쓰기 워크로드를 처리할 수 있습니다.
- 고가용성 : MongoDB에는 자동 장애 조치 및 복제본 세트를 위한 기본 제공 기능이 포함되어 있어 하드웨어 오류 또는 기타 중단 중에 데이터베이스를 계속 사용할 수 있고 액세스할 수 있습니다.

- 인덱싱 : MongoDB는 인덱싱을 지원하여 쿼리 및 검색 성능을 향상시켜 컬렉션 내에서 특정 문서를 더 빠르고 쉽게 찾을 수 있도록 합니다.
- 집계 : MongoDB는 데이터 집계를 위한 기본 제공 도구를 제공하므로 개발자가 대량의 데이터를 쉽게 분석하고 조작할 수 있습니다.
- 광범위한 언어 지원 : MongoDB는 JavaScript, Python, Java, C++ 등을 포함한 많은 프로그래밍 언어와 함께 사용할 수 있습니다.

이러한 기능을 통해 MongoDB는 최신 데이터 기반 애플리케이션에서 대량의 데이터를 저장하고 관리하기 위한 강력하고 유연한 도구가 됩니다.

2.2 매크로 및 악성코드 탐지

문서 내 악성 요소 탐지(VBA 매크로 탐지)

1. VBA란 MS 오피스 문서에서 사용자가 원하는 기능을 직접 소스 코드로 작성하는 기능으로, 번거로운 작업을 자동화하여 사용자의 편의를 돕는 도구입니다. 그러나 이 기능은 파일 생성, 삭제, 실행뿐만 아니라 레지스트리와 같은 시스템 파일에 대한 접근도 가능하여 악의적인 목적으로 악용될 수 있습니다.

2. VBA 매크로 탐지 - VBA 프로젝트에 삽입된 매크로를 탐지합니다.
비정상적인 레코드 탐지 - 엑셀의 workbook 스트림 내부에 삽입된 비정상적인 바이너리 데이터를 탐지합니다.

3. 외부 객체 참조 영역 내 비정상 요소 탐지 - MS 오피스 문서는 다른 문서 파일, PDF, 이미지, Adobe Flash 등 32가지 외부 파일을 삽입할 수 있으며, 이 기능을 악용해 악성 요소를 삽입하는 경우를 탐지합니다.

4. 미사용 영역 데이터 탐지 - MS 문서는 복합 파일 이진 구조를 가지고 있으며, 이 구조에는 비할당 영역이 존재할 수 있습니다. 이 영역에 쉘코드나 악성 실행 파일과 같은 바이너리 데이터가 저장될 수 있어 이를 탐지합니다.

5. 스트림과 스토리지 구조 변조 확인 - 복합 이진 파일 구조에서 스트림은 문서 내용을 담고, 스토리지는 파일을 담는 폴더와 같은 개념입니다. 공격자는 임의로 스토리지나 스트림을 생성하거나 변조하여 악성코드를 심을 수 있습니다.

> 5-1. 알 수 없는 스트림이나 스토리지가 존재하는지 여부를 판단하여 파일의 비정상 요소를 탐지합니다.
>
> 5-2. 버퍼 오버플로우나 응용 프로그램의 라이브러리 취약점을 악용한 문서 삽입형 악성코드의 경우, 필수적인 스토리지와 스트림이 누락될 수 있습니다. 따라서, 파일 내 필수 스트림과 스토리지의 존재 여부를 확인하여 손상된 파일이나 악성 요소를 파악합니다.
>
> 5-3 문서 파일 내부의 각 스트림이 정상적인 부모 스토리지 하위에 위치하고 있는지에 대한 검증도 수행함으로써 비정상 요소를 확인합니다

2.2 매크로 및 악성코드 탐지

문서 내 악성 요소 탐지(VirusTotal)

- VirusTotal은 파일이나 URL을 다수의 바이러스 백신 엔진과 웹사이트 검사 도구를 사용하여 분석하고, 잠재적인 악성코드 여부를 탐지하는 온라인 서비스입니다. 다양한 바이러스 탐지 도구를 하나의 플랫폼에서 제공하여 파일의 악성 여부를 판별할 수 있는 강력한 도구로 활용되고 있습니다. 이를 통해 각기 다른 백신 엔진이 탐지한 결과를 종합적으로 보여주어, 악성코드를 탐지할 때 더 높은 신뢰성을 제공합니다.

VirusTotal의 주요 기능:

1. 파일 검사: 사용자가 파일을 업로드하면, VirusTotal은 여러 백신 엔진을 사용해 파일을 검사합니다. 각 백신 엔진의 탐지 결과를 비교함으로써, 악성코드가 포함된 파일을 신속하게 판별할 수 있습니다. 이를 통해 다양한 형태의 바이러스, 악성코드, 트로이 목마 등의 위협을 탐지할 수 있습니다.
2. URL 검사: 악성 웹사이트를 탐지하기 위해 URL을 제출하면, VirusTotal은 이를 여러 웹 검사 도구로 분석합니다. 이를 통해 피싱 사이트, 악성 스크립트 또는 의심스러운 사이트를 사전에 차단하는 데 도움이 됩니다.

3. 다중 백신 엔진 통합: VirusTotal은 여러 백신 엔진과의 협력을 통해 파일을 동시에 여러 엔진으로 검사합니다. 엔진마다 탐지 방식이나 데이터베이스가 다르기 때문에 다중 엔진의 결과를 종합하는 방식은 악성코드 탐지의 신뢰도를 높입니다.

4. 행위 기반 분석: VirusTotal은 정적 분석뿐만 아니라 파일의 동적 행위를 분석하는 기능도 제공합니다. 예를 들어, 의심스러운 파일이 실행 중 시스템 리소스나 네트워크에 어떤 변화를 일으키는지 추적하여, 잠재적 악성 행위를 발견합니다.

5. API 지원: VirusTotal은 API를 제공하여 다양한 개발자가 자신의 프로그램에서 VirusTotal의 파일 검사 기능을 활용할 수 있게 지원합니다. 이를 통해 파일 업로드 및 분석을 자동화할 수 있으며, 보안 시스템이나 애플리케이션에서 악성코드 탐지 기능을 통합할 수 있습니다.

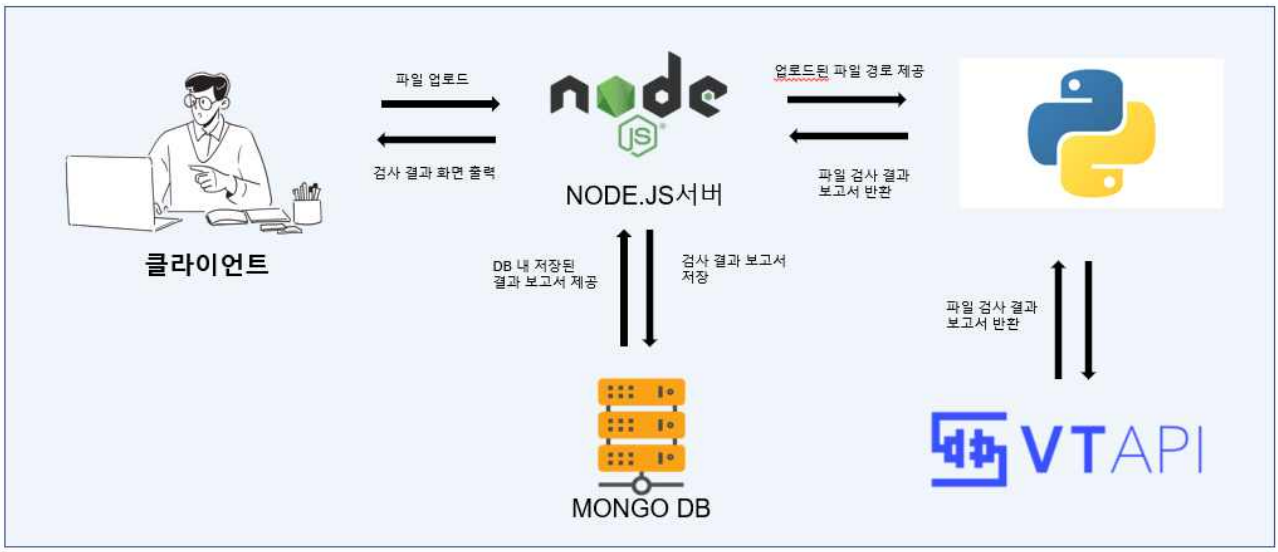
바이러스 탐지의 의의

VirusTotal의 바이러스 탐지 기능은 단일 백신 소프트웨어로는 발견하기 어려운 위협을 여러 백신 엔진을 통해 종합적으로 분석할 수 있다는 점에서 유리합니다. 이를 통해 악성코드를 탐지하고 신속하게 대응할 수 있으며, 다양한 위협 요소에 대한 인텔리전스를 제공하여 보안의 전반적인 수준을 향상시키는 역할을 합니다. 특히 기업이나 개발자들이 VirusTotal API를 통해 자동화된 악성코드 분석 시스템을 구축함으로써, 보안 프로세스를 효율화하고 위협에 대한 빠른 대응이 가능합니다.

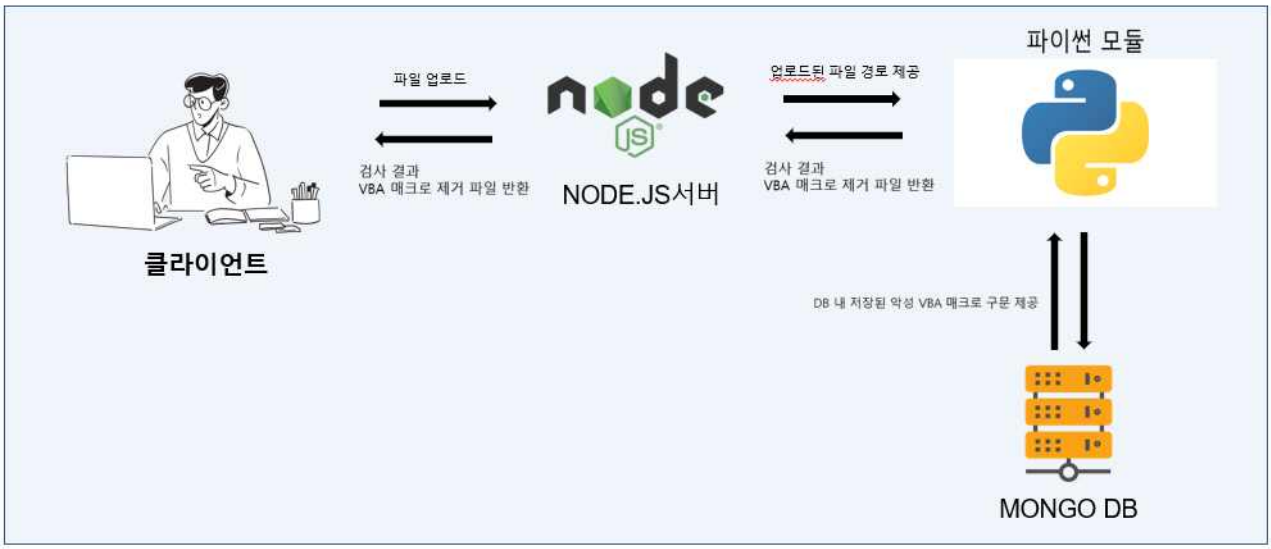
3. 프로젝트 내용

3.1 시스템 구성도

1. 바이러스 토탈 API 활용한 탐지

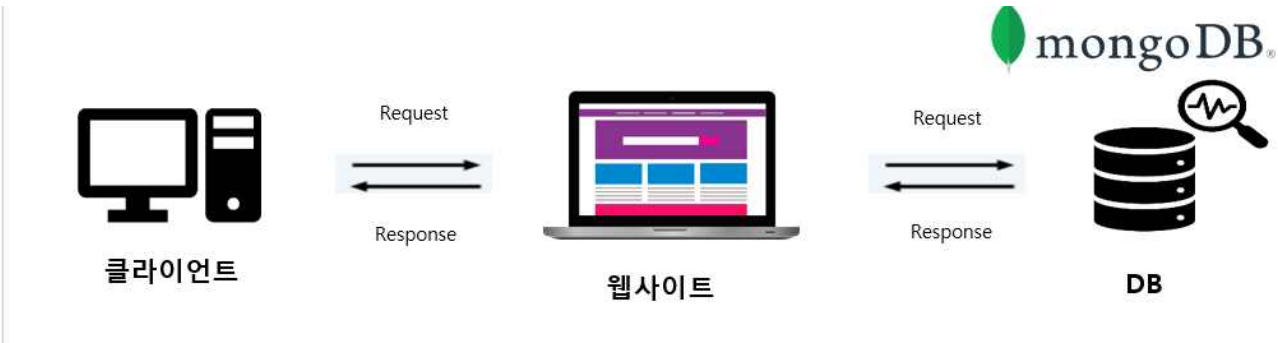


2. VBA 매크로 탐지&제거

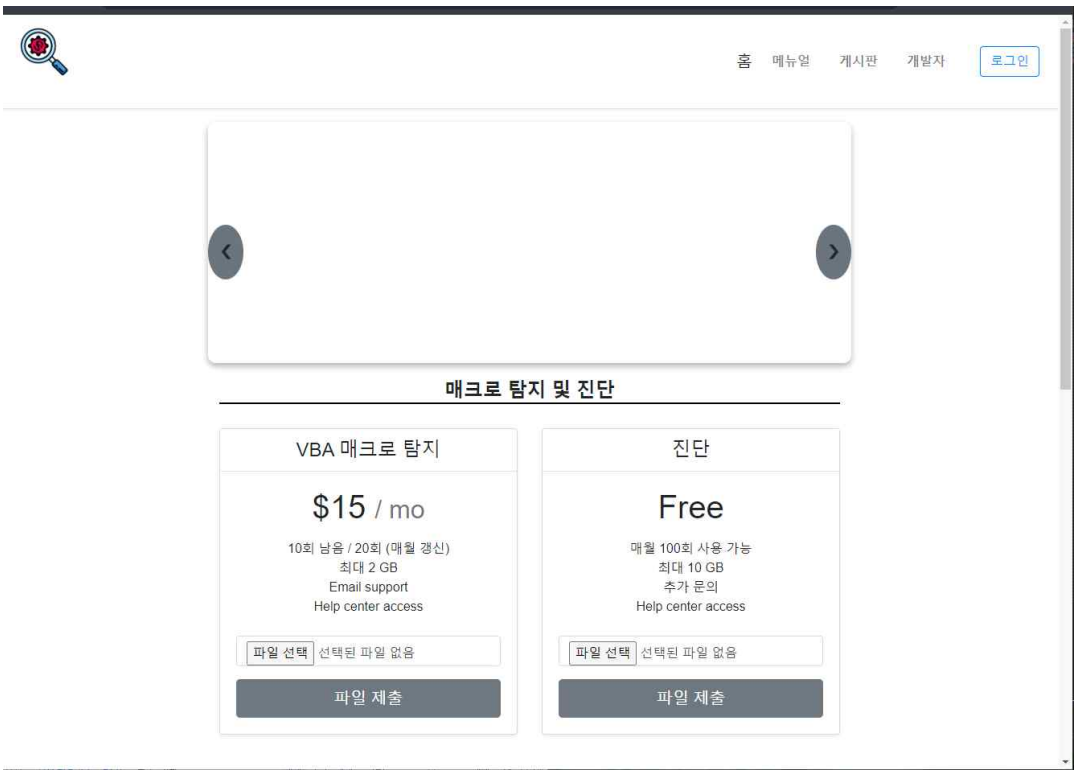


3.2 웹사이트 제작

- Node.js 와 mongoDB를 활용한 웹사이트 제작

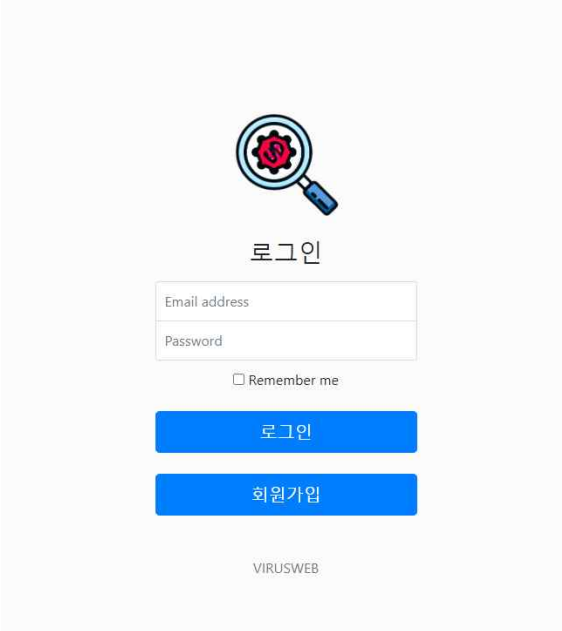


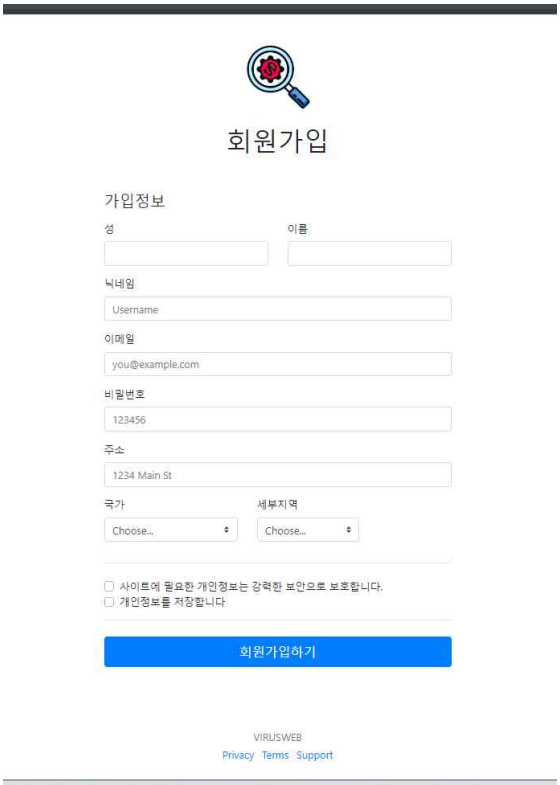
다양한 기능이 있는 웹사이트를 제작하였습니다.



주요기능

- 회원가입/로그인
- 마이페이지
- 게시판
- 피드백페이지


로그인
Email address
Password
Remember me
로그인
회원가입
VIRUSWEB


회원가입
가입정보
성
이름
닉네임
Username
이메일
you@example.com
비밀번호
123456
주소
1234 Main St
국가
세부지역
Choose...
Choose...
사이트에 필요한 개인정보는 강력한 보안으로 보호합니다.
개인정보를 저장합니다
회원가입하기
VIRUSWEB
Privacy Terms Support


공지사항
번호
제목
글쓴이
작성일
조회
1
2
3
4
5

3.3 파일 업로드& 바이러스 검사(바이러스 진단)

```
// 파일 저장 디렉토리 설정
const storage = multer.diskStorage({
  destination: function (req, file, callback) {
    callback(null, 'upload/')
  },
  filename: function (req, file, callback) {
    callback(null, file.originalname)
  },
})

// 파일 업로드 미들웨어 생성
const upload = multer({ storage: storage })

app.post('/upload', upload.single('file'), (req, res) => {
  // single 메서드를 이용하여 하나의 파일만 업로드하도록 설정
  // 'file'은 프론트엔드에서 업로드할 때 사용한 name 속성 값입니다.
  const filePath = path.resolve(req.file.path)
  console.log('업로드 완료!')
  uploadFilePath = filePath

  // /scan 으로 redirect 합니다.
  res.redirect(`/scan`)
})
```

```
pythonProcess.stdout.on('end', () => {
  console.log('Python process ended')
  console.log(pythonResult)
  // const resultJson = JSON.parse(pythonResult)
  // console.log(resultJson)
  res.send(pythonResult)
})

pythonProcess.stderr.on('data', (data) => {
  if (data.includes('에러')) {
    console.error(`stderr: ${data}`)
    res.status(500).send('Internal Server Error')
  }
})

pythonProcess.on('close', (code) => {
  console.log(`child process exited with code ${code}`)
})
)
```

```
with open(file_path, "rb") as f:
    file_content = f.read()
uploadurl = "https://www.virustotal.com/api/v3/files"

files={"file": file_content}

headers = {
    "accept": "application/json",
    "x-apikey": os.environ.get('API_KEY')
```

- node.js 상에서 파이썬 프로그램을 사용하기 위해서 child_process라는 모듈를 사용하면 파이썬 프로그램을 호출하고 인자값을 전달할 수 있습니다. 파이썬을 호출 후 서버에 저장된 파일의 경로를 전달, 바이러스토탈 파일 업로드 api까지 접목해서 코드 추가, 여기서 해시값 사용 용도는 업로드 파일의 검사 결과를 받아오는데 쓰입니다.

- 결과값을 받고 웹페이지에 결과값을 출력, 오류 발생 시 오류 처리

서버에 업로드 받은 파일 검사 이후 삭제 기능

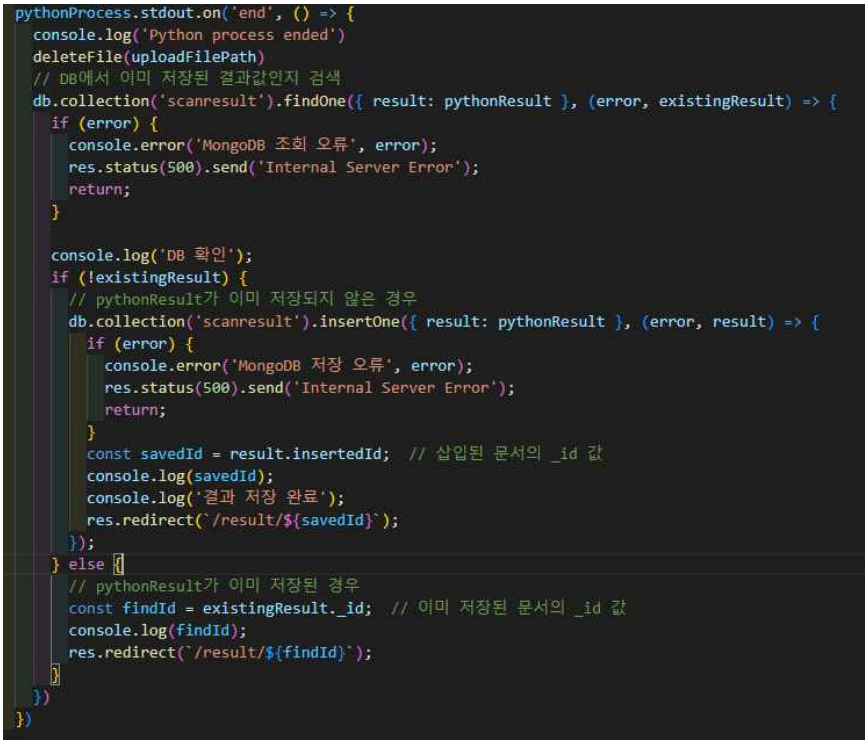
- 업로드 받은 파일을 서버 저장소에 계속 잔재 시 이후 용량 문제 및 악성파일이 이후에 문제를 일으킬 수 있으므로 검사 이후 결과 반환 받은 뒤 해당 파일을 삭제하는 기능을 추가

```
//파일 삭제 처리 함수
const deleteFile = filePath =>{
  fs.unlink(filePath,(err)=>{
    if (err) {
      console.error('파일 삭제 오류', error)
    } else {
      console.log('파일삭제 완료')
    }
  })
}
```

```
console.log('Python process ended')
deleteFile(uploadFilePath)
```

DB에 중복 결과값 저장 방지 기능 추가

- 파일 업로드 시 이미 결과값이 저장되었던 파일이면 저장하지 않고 이미 저장되어있는 파일의 결과값을 반환해주는 방식으로 구현하였습니다.



```
pythonProcess.stdout.on('end', () => {
  console.log('Python process ended')
  deleteFile(uploadFilePath)
  // DB에서 이미 저장된 결과값인자 검색
  db.collection('scanresult').findOne({ result: pythonResult }, (error, existingResult) => {
    if (error) {
      console.error('MongoDB 조회 오류', error);
      res.status(500).send('Internal Server Error');
      return;
    }

    console.log('DB 확인');
    if (!existingResult) {
      // pythonResult가 이미 저장되지 않은 경우
      db.collection('scanresult').insertOne({ result: pythonResult }, (error, result) => {
        if (error) {
          console.error('MongoDB 저장 오류', error);
          res.status(500).send('Internal Server Error');
          return;
        }
        const savedId = result.insertedId;  // 삽입된 문서의 _id 값
        console.log(savedId);
        console.log('결과 저장 완료');
        res.redirect(`/result/${savedId}`);
      });
    } else {
      // pythonResult가 이미 저장된 경우
      const findId = existingResult._id;  // 이미 저장된 문서의 _id 값
      console.log(findId);
      res.redirect(`/result/${findId}`);
    }
  })
})
```

3.4 매크로 탐지, 제거 기능 제작

3.4.1 파일 업로드 취약점 보안

- 파일 업로드 취약점은 파일 업로드 기능을 악용하여 시스템 권한을 획득할 수 있는 취약점으로 웹서버에 악성 스크립트을 업로드하고 실행하여 시스템의 권한을 장악합니다.

(1) 파일 업로드 가능 확장자 제한
- 웹 서버에서 실행 가능한 확장자는 다음과 같습니다.

언어	확장자
asp, aspx	asp, aspx, htm, html, asa
php	phtml, php, php3, php4, php5, inc, htm, html
jsp, java	jsp, jspx, jsw, jsv, jspf, htm, html
perl	pl, pm, cgi, lib, htm, html
coldfusion	cfm, cfml, cfc, dbm, htm, html

파일 업로드 취약점을 보안하는 가장 기초적인 방법은 파일 업로드 가능 확장자를 제한하는 것.
본 사이트는 문서 파일 검사만을 취급하기 때문에 문서 확장자로만 제한했습니다.

```
<input
type="file"
name="file"
accept=".doc,.docx,.ppt,.pptx,.xls,.xlsx,.xlsm, .pdf"
```

(2) 확장자 우회 대응
- 확장자 우회는 파일 업로드 시 파일에 이름에 특수문자를 넣어서 진짜 확장자를 숨기는 거나 업로드 시 서버 요청 헤더의 Content-type 속성을 조작하여 우회해서 허용되지 않은 확장자의 파일을 업로드하는 것입니다..
- 파일명에 특수문자를 삽입하는 방식에 공격에 대응하기 위해 특수문자가 포함된 파일명은 업로드가 제한됩니다.

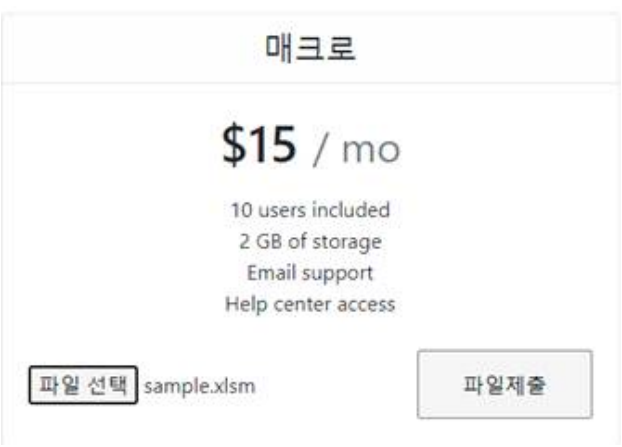
```
<script>
  function validateFileName() {
    const fileInput = document.querySelector('input[name="file"]');
    const fileName = fileInput.value;
    const disallowedCharacters = /[!@#$%^&*(),?"{}|<>]/; // 특수 기호 거부
    console.log("파일 이름:", fileName)

    if (disallowedCharacters.test(fileName)) {
      alert("파일명에 특수 기호가 포함되어 있습니다.");
      return false; // 파일 업로드를 중단하기 위해 false를 반환합니다.
    }

    return true; // 파일 업로드를 진행할 수 있음을 나타내기 위해 true를 반환합니다.
  }
</script>
```

- 업로드 요청 파일의 확장자를 업로드를 처리 중 파일의 정보를 다시 한번 확인하여 허용되지 않은 파일은 삭제합니다.

```
const allowedExtensions = [
  '.doc',
  '.docx',
  '.ppt',
  '.pptx',
  '.xls',
  '.xlsx',
  '.xlsm',
  '.pdf',
]
```

```
app.post('/upload', upload.single('file'), (req, res) => {
  if (!req.file) {
    res.status(400).send('파일이 업로드되지 않았습니다.')
    return
  }
  const filePath = path.resolve(req.file.path)
  uploadFilePath = filePath
  const fileExtension = path.extname(req.file.originalname).toLowerCase()
  if (!allowedExtensions.includes(fileExtension)) {
    deleteFile(filePath)
    res.status(400).send('허용되지 않는 파일입니다.')
    return
  }

  console.log('파일 유형:', fileExtension)
  console.log('업로드 완료!')
  res.redirect('/scan')
})
```

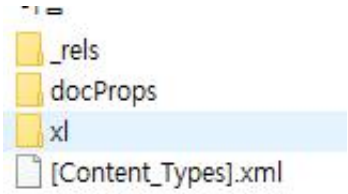
3.4.2 ms문서 파일의 VBA 매크로 검사



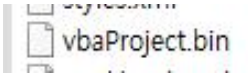
- 해당 폼으로 파일 제출 시 DB에 저장된 매크로 구문과 비교하여 어떤 VBA 매크로 구문이 어떤 식으로 작성되었는지 확인할 수 있습니다.
- oletools를 이용하면 ole 구조의 문서 파일 내 저장된 매크로를 쉽게 찾아낼 수 있습니다.

3.4.3 VBA 매크로 제거

- 문서 파일을 업로드 받을 시 문서 파일을 zip파일로 처리하여 압축을 해제합니다.



- zip 파일로 처리하면 위와 같은 파일들을 확인할 수 있는데
- xl 폴더 아래

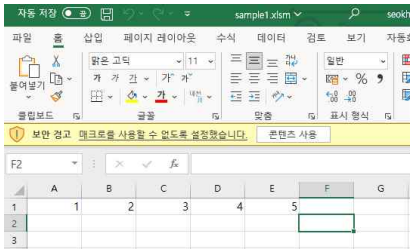


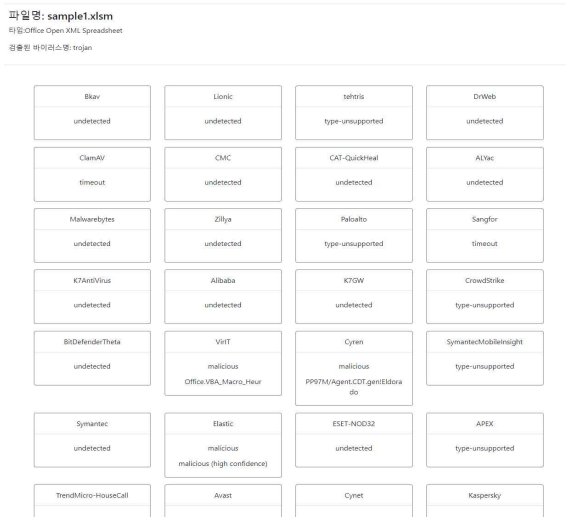
라는 파일을 확인할 수 있습니다.
해당 바이너리 파일에 vba 매크로 스크립트가 저장되게 됩니다.
해당 바이너리 파일의 바이너리 데이터를 읽어드린 뒤 DB에 저장된 매크로 구문을 바이트 데이터로 인코딩 후 바이너리 데이터에 해당 매크로 구문이 존재 시 삭제하고 vbaProject1.bin 이라는 새로운 바이너리 파일로 저장합니다.

```
with open("vbaProject.bin", "rb") as f:
    while True:
        data = f.read()
        if not data:
            break

        def remove_macros(input_data, macros):
            output = input_data
            for macro in macros:
                macro_bytes = macro["macro"].encode()
                output = output.replace(macro_bytes, b"")
            return output

        result = remove_macros(data, macros)

        save_path = "vbaProject1.bin"
        with open(save_path, "wb") as file:
            file.write(result)
```

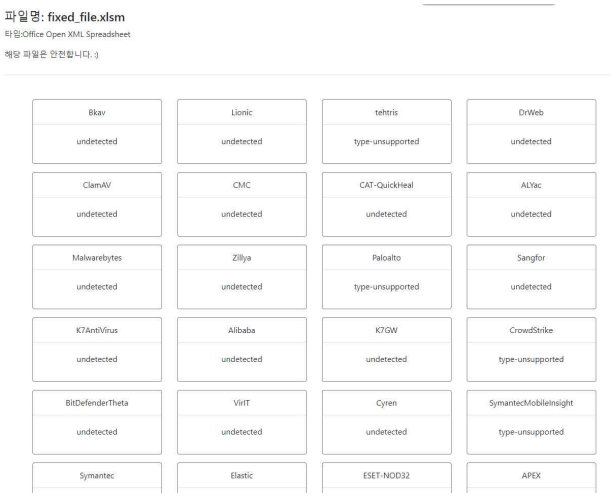
```
def zip():
    current_directory = os.getcwd()
    fixfile = os.path.join(current_directory, new_filename)
    rel_directory = os.path.join(current_directory, "_rels")
    docprops_directory = os.path.join(current_directory, "docProps")
    xl_directory = os.path.join(current_directory, "xl")
    content_types_xml = os.path.join(current_directory, "[Content_Types].xml")

    with zipfile.ZipFile(fixfile, "w", compression=zipfile.ZIP_DEFLATED) as fix_xlsm:
        for root, dirs, files in os.walk(rel_directory):
            for file in files:
                file_path = os.path.join(root, file)
                fix_xlsm.write(file_path, os.path.relpath(file_path, current_directory))

        for root, dirs, files in os.walk(docprops_directory):
            for file in files:
                file_path = os.path.join(root, file)
                fix_xlsm.write(file_path, os.path.relpath(file_path, current_directory))

        for root, dirs, files in os.walk(xl_directory):
            for file in files:
                if file != "vbaProject.bin":  # fix.xlsm 파일에 포함시키지 않을 파일명 지정
                    file_path = os.path.join(root, file)
                    fix_xlsm.write(file_path, os.path.relpath(file_path, current_directory))
```

이후 기존 vbaProject.bin을 제외하고 새로 만든 파일을 포함시켜서 재압축한다. 그리고 서버에서 웹페이지로 파일을 반환하여 클라이언트에게 제공합니다.
번거롭게 압축해제, 파일 생성, 재압축 과정을 거치는 이유는 기존 파일에서 매크로를 제거시 파일 구조에서 데이터 블록에 손상을 입히게 하기 때문입니다.

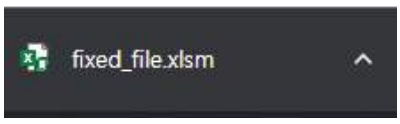


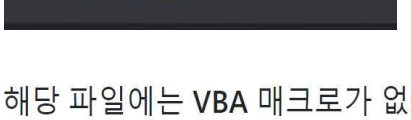


- 자체 제작한 vba매크로가 존재하며 내용이 존재하는 sample1.xlsm를 이용하여 실험해보았습니다. 파일 검사 시 VBA 매크로 등이 발견되는 것을 볼 수 있습니다.

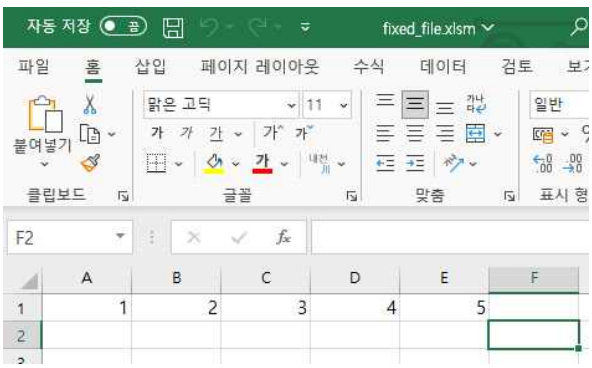


파일 제출 시 해당 파일로 반환된다.





파일 내용이 정상적으로 존재
매크로가 제거된 것을 확인할 수 있습니다.



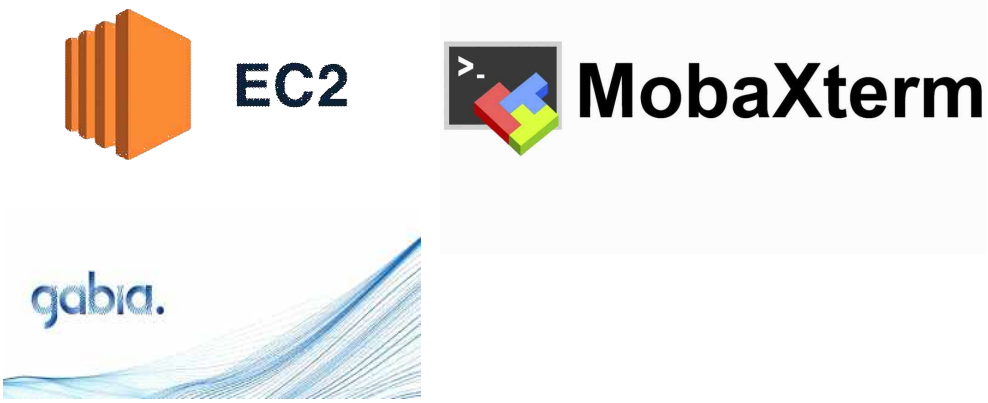
3.5 배포 환경 구성 및 연결

본 연구에서는 배포 프로그램으로 MobaXterm을 사용하여 AWS EC2 Public Instance와 연결하였습니다. AWS EC2는 확장성, 안정성, 그리고 유연한 비용 구조를 제공하기 때문에 대규모 웹 애플리케이션을 호스팅하기에 적합합니다. 특히 AWS는 다양한 리소스 모니터링 및 관리 도구를 통해 서버의 성능을 최적화할 수 있으며, 필요에 따라 인스턴스의 크기를 유동적으로 조정할 수 있어 배포 환경에서 탁월한 선택입니다.

우선 Ubuntu 기반의 EC2 서버를 생성하고, 퍼블릭 IP 주소를 사용해 외부에서 접근 가능하도록 설정하였습니다. 이후, 가비아에서 구매한 도메인과 AWS EC2 인스턴스를 연결하기 위해 AWS Route 53 서비스를 활용해 도메인 네임 서버(DNS) 설정을 진행하였습니다. 이를 통해 EC2 인스턴스에 도메인을 연결하여 사용자가 도메인을 통해 서버에 접근할 수 있도록 하였습니다.

MobaXterm은 직관적인 사용자 인터페이스와 **다양한 원격 프로토콜 지원(SFTP, SSH 등)**을 제공하는 점에서 선택되었습니다. 특히 SSH 프로토콜을 통해 EC2 인스턴스에 원격으로 접속해 서버 관리와 애플리케이션 배포를 효율적으로 수행할 수 있었습니다. MobaXterm은 여러 탭을 사용하여 동시에 여러 서버에 연결하거나 파일 전송을 쉽게 관리할 수 있어 개발 및 운영 과정에서 유용합니다.

이와 같은 배포 환경은 AWS의 클라우드 인프라와 MobaXterm의 편리한 관리 도구를 결합함으로써 서버 배포와 유지 관리의 효율성을 극대화하였습니다. 이는 확장 가능한 클라우드 인프라와 직관적인 원격 관리 도구를 활용한 안정적이고 효율적인 애플리케이션 배포 환경을 구축하는 데 기여하였습니다.



4. 결론

4.1 결론 및 기대효과

<파일 검사>
사용자가 악성코드가 포함된 파일을 제출하면, VirusTotal API를 활용하여 파일의 검사 결과를 사용자에게 출력합니다. 검사 결과에서 악성코드가 발견될 경우, 사용자는 해당 악성코드 카드를 클릭하면 미리 연결된 ChatGPT로부터 악성코드에 대한 실시간 설명을 제공받을 수 있습니다(예정).

<치료 과정(매크로 제거)>
악성 매크로에 주로 사용되는 구문으로는 String 형식의 자동 실행 트리거인 AutoOpen, Document_Open, DocumentOpen이 있으며, 파일 시스템이나 메모리 데이터를 수정할 수 있는 메소드인 Write, Put, Output, Print가 있습니다. 또한, 외부에서 파일을 실행하거나 다운로드하는 메소드로는 Powershell, URLDownloadToFileA, Shell, Wscript.shell, run 등이 있습니다.

현재 프로젝트에서는 샘플 파일에 Powershell 코드를 임의로 작성한 후, 매크로가 저장되는 바이너리 파일을 텍스트 형식으로 추출하여 Powershell 코드가 포함된 경우 이를 악성 파일로 간주하고 해당 구문을 삭제합니다. 악성 매크로가 제거된 안전한 문서 파일을 사용자에게 제공하는 방식으로 작동합니다.

- 악성코드 점검 효율성 증대

- 자동 진단 도구 활용으로 인한 악성코드 진단 시간 단축

- 빠른 취약점 위치 식별 잠재적 취약점 발생 위치 파악 및 대처

- 학생들의 보안 역량 강화 모의 점검 웹사이트를 직접 제작하고 연구함으로써 학생들의 실습 경험 제공

4.2 향후 계획

- 추가적인 보안대책 강구
- 향후 사이트 UI 개선
- 매크로 필터 추가
- 백신 프로그램 개발

5. 별첨

5.1 팀원 소개

5.2 발표 PPT (별첨)

5.3 소스 코드

https://github.com/MJ174/virusweb

5.4 YouTube 시연 영상

https://www.youtube.com/watch?v=C_pyD3gZwiQ

5.5 웹서비스 주소

http://virusweb.site:8000