악성코드/바이러스 탐지 웹어플리케이션

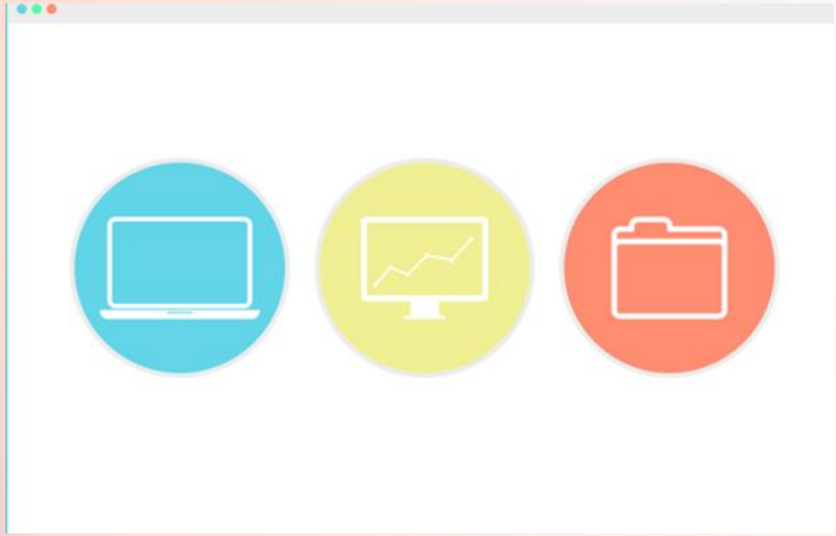
캐치버그핑
팀원 : 신명진 김태현 김호준 최경은 임혜준

목차

프로젝트 배경 및 필요성

문서 혹은 메일로 위장한 악성코드 유포 사례가 발생

사전에 진단하고 점검하는 프로그램의 중요성 인식



프로젝트 배경 및 필요성

-문서형 악성 코드의 경우는 불가피하게 문서를 열어봐야 하는 기자나 특수성을 가진 직업의 경우 보안에 매우 취약하다.

-메일형 악성 코드의 경우에는 하나라도 감염되는 순간 개인 pc를 넘어 서버까지 순식간에 감염시킬 수 있기에 임직원 단 한 명의 방심으로 사내 인프라가 영향을 받을 수 있는 보안 위험이 있다.

-이처럼 악성 코드의 형태가 매우 다양해지고 있는 만큼 대처하기 위한 방안과 관련 프로그램의 필요성도 증가하고 있다.

-대부분의 해킹 대응 방식이 감염 기기에 직접 접근해 악성코드를 제거하는 방식이기 때문에 악성 코드의 탐지가 매우 중요하다.

팀원 소개 및 역할 분담

신명진(팀장)

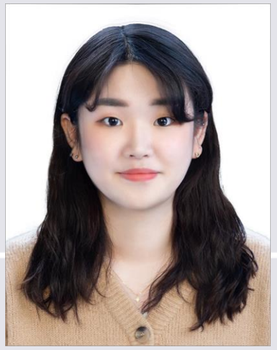
DB 개발
웹 페이지 기능 개발, 구현

김호준

웹 페이지 구상, 구현
PPT 제작

김태현

파이썬 파트 개발, 발표



최경은

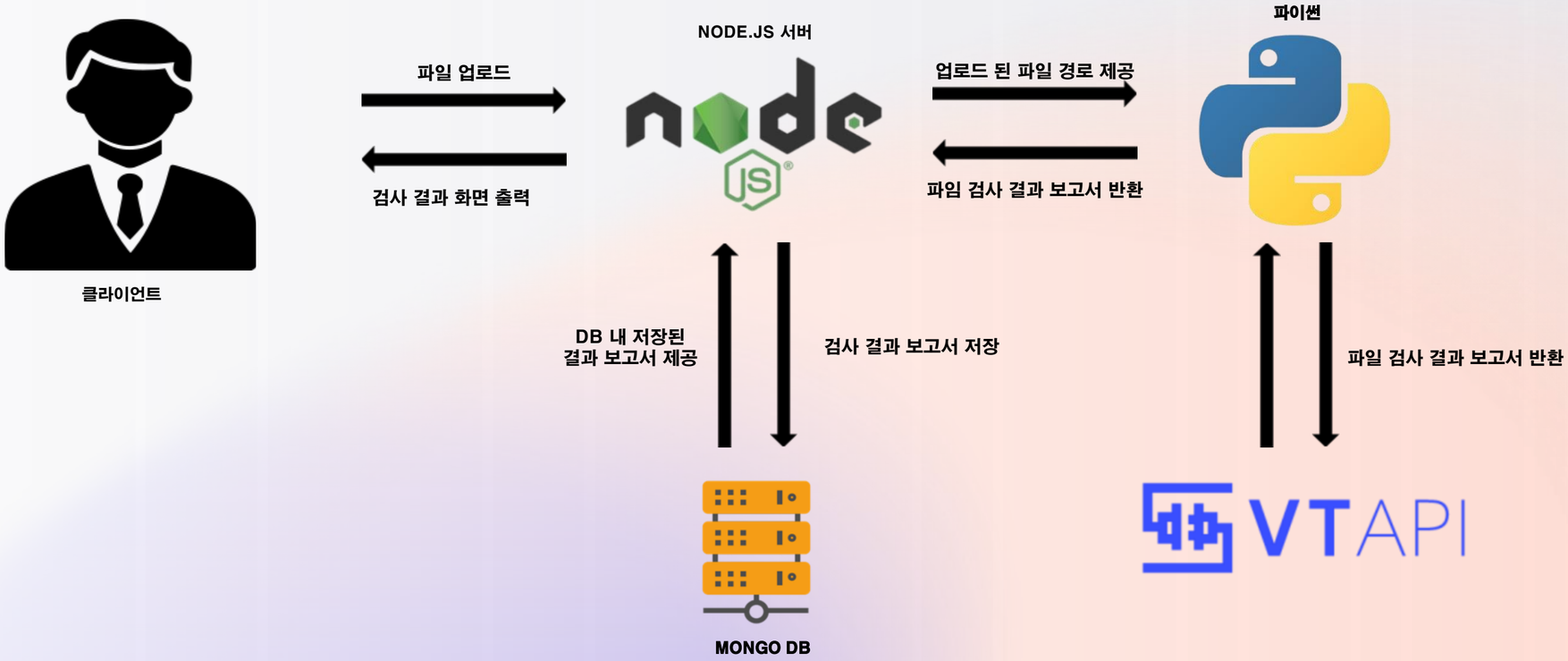
웹 페이지 구상, 개발
PPT 제작

임혜준

악성코드 분석, 메크로 제작

프로젝트 구성도

바이러스 토탈 API 활용한 탐지



프로젝트 구성도

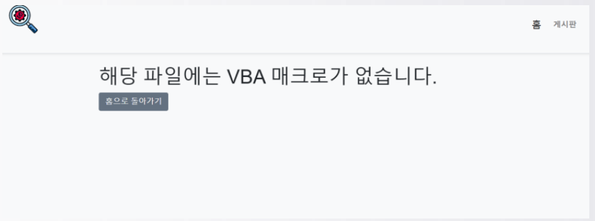
VBA 매크로 탐지 & 제거

프로젝트 진행 방법

웹사이트 제작

파일 검사 기능 제작

매크로 탐지 & 제거 기능 제작



웹 사이트 제작

Node.js와 mongoDB를 활용한 웹사이트 제작

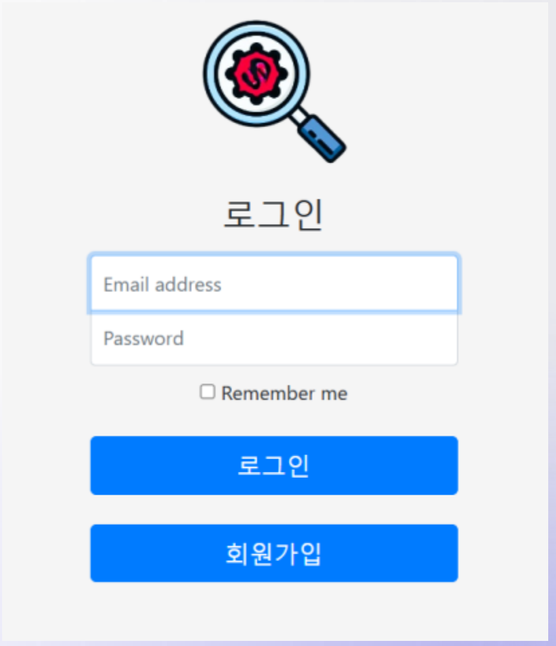
웹 사이트 제작

주요 기능


protect your computer
from viruses
매크로 탐지 및 진단
VBA 매크로 탐지
$15 / mo
10회 남음 / 20회 (매월 갱신)
최대 2 GB
Email support
Help center access
파일 선택 선택된 파일 없음
파일 제출
진단
Free
매월 100회 사용 가능
최대 10 GB
추가 문의
Help center access
파일 선택 선택된 파일 없음
파일 제출
NOTICE
[매크로검사] 파일 검사하러 가기 2024.05.09
[공지사항] 바이러스 웹 사용방법 2024.04.12

웹 사이트 기능

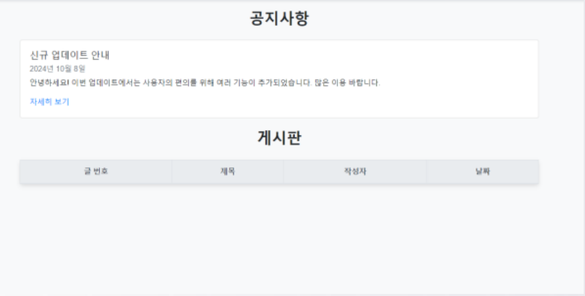
로그인



회원가입

검사 결과 카드

공지사항



파일 업로드 & 검사 제작

파일 업로드 & 검사

```
// 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}`)
})
)
```

```
import hashlib
import requests
import json
import sys
from dotenv import load_dotenv
import os
from time import sleep


def main():

    load_dotenv()

    file_path = sys.argv[1]

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

    files= {"file" : (file_path, open(file_path,'rb')) }

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

    uploadresponse = requests.post(uploadurl,files=files, headers=headers)

    sleep(60)

    hash_value = hashlib.sha256(file_content).hexdigest()

    url = "https://www.virustotal.com/api/v3/files/{}".format(hash_value)

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

    response = requests.get(url, headers=headers)
    result = json.dumps(response.json())
    print(result)
    return result
if __name__=='__main__':
    main()
```

node.js 상에서 파이썬 프로그램을 사용하기 위해서 child_process라는 모듈를 사용하면 파이썬 프로그램을 호출하고 인자값을 전달할 수 있다.

파이썬을 호출 후 서버에 저장된 파일의 경로를 파이썬 프로그램에 전달

바이러스토탈에서 제공하는 api를 통해 바이러스 토탈에 파일을 업로드하고 바이러스 토탈에서 검사 후 작성되는 결과 보고서를 파일의 sha-256 해시값을 이용해 받아오고자 한다. 받은 결과 보고서의 정보를 토대로 웹페이지에 출력하도록 한다.

파일 업로드 & 검사 제작

서버에 업로드 받은 파일 검사 이후 삭제 기능, DB에 중복 결과값 저장 방지 기능

```
//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)
```

```
pythonProcess.stdout.on('end', () => {
  console.log('Python process ended')
  deleteFile(uploadFilePath)
  // DB에서 이미 저장된 결과값인지 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}`);
    }
  })
})
```

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 추가하였다.
결과값을 DB에 저장 시 이미 저장되었던 파일의 결과값이면 저장하지 않고 이미 저장되어 있는 파일의 결과값을 반환해주는 방식으로 구현하였다.

파일 업로드 & 검사 제작

파일 업로드 취약점 보안

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

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

```
<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>
```

```
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')
})
```

본사이트는 문서 파일 검사만을 취급하기 때문에 문서 확장자로만 제한했다.
업로드 요청 파일의 확장자를 업로드를 처리 중 파일의 정보를 다시 한번 확인하여 허용되지 않은 파일은 삭제

매크로 탐지, 제거 기능 제작

매크로 및 악성 코드 탐지 정리

- VBA란 ms 오피스 문서들에서는 사용자가 원하는 기능을 직접 소스 코드로 작성하는 기능으로 번거로운 작업을 소스코드로 구현하여 사용자의 편의를 위한 기능이다.

- 하지만 해당 기능은 일반적인 파일에 대한 생성과 삭제, 실행 등이 가능하며, 레지스트리와 같은 시스템 파일에 대한 접근도 가능하기 때문에 악의적인 목적으로도 충분히 사용할 수 있다.

MS 문서 내 악성 요소 탐지 기법

1. VBA 매크로 탐지 - VBA 프로젝트에 삽입되는 매크로를 탐지

2. 비정상적인 레코드 탐지 - 엑셀에 존재하는 workbook 스트림 내부의 바이너리 데이터를 삽입하는 방식을 탐지

3. 외부 객체 참조 영역 내 비정상 요소 탐지 - MS 오피스는 해당 문서 파일 내부에 다른 문서 파일이나 PDF, 이미지 파일, Adobe Flash 등
32가지의 파일을 삽입할 수 있는데 해당 기능을 이용해 악성 기능을 삽입

4. 미사용 영역 데이터 탐지-ms 문서들은 복합 파일 이진 구조를 띄고 있는데 이 구조 특성상 비할당영역과 같은 미사용 영역이 존재할 수 있다. 해당 영역에는 쉘코드 혹은 악의적인 실행 파일과 같은 바이너리 데이터가 저장 될 수 있다.

매크로 탐지, 제거 기능 제작

매크로 및 악성 코드 탐지 정리

5. 스트림과 스토리지 구조 변조 확인

- 복합 이진 파일 구조에서는 스트림과 스토리지라는 개념을 갖는데 스트림은 문서 내용을 담고 있는 파일, 스토리지는 파일을 담는 폴더와 같은 개념이다.
- 공격자가 임의로 스토리지나 스트림을 생성하거나 변조하여 악성 코드를 심을 수 있다.

5-1. 알 수 없는 스트림이나 스토리지가 존재하는지 여부를 판단함으로써 해당 파일의 비정상 요소 존재 여부를 파악

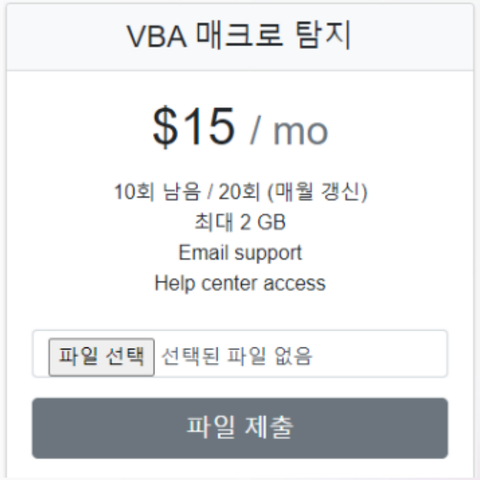
5-2. 한, 버퍼오버플로우나 응용프로그램이 사용하는 라이브러리 취약점 등을 사용한 문서 삽입형 악성코드의 경우 해당 파일의 필수
스토리지와 스트림이 존재하지 않을 수 있다. 따라서 파일 내에 필수 스트림과 스토리지의 존재 여부를 살펴봄으로써 손상된 파일이나
악성 코드와 같은 비정상요소를 파악

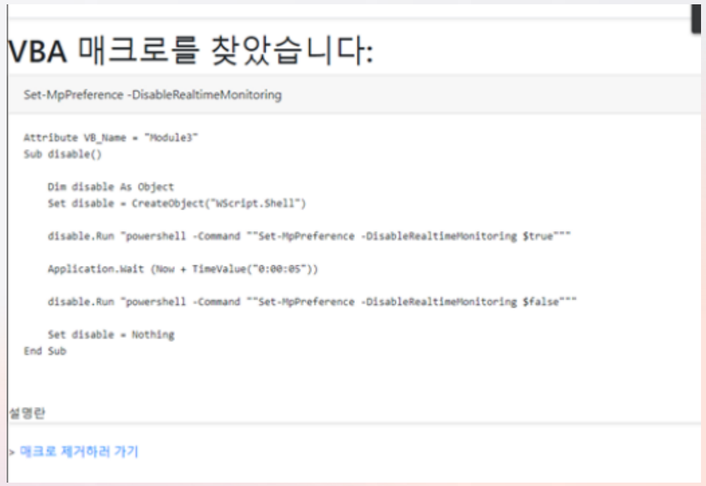
5-3 문서 파일 내부의 각 스트림이 정상적인 부모 스토리지 하위에 위치하고 있는지에 대한 검증도 수행함으로써 비정상 요소를 확인

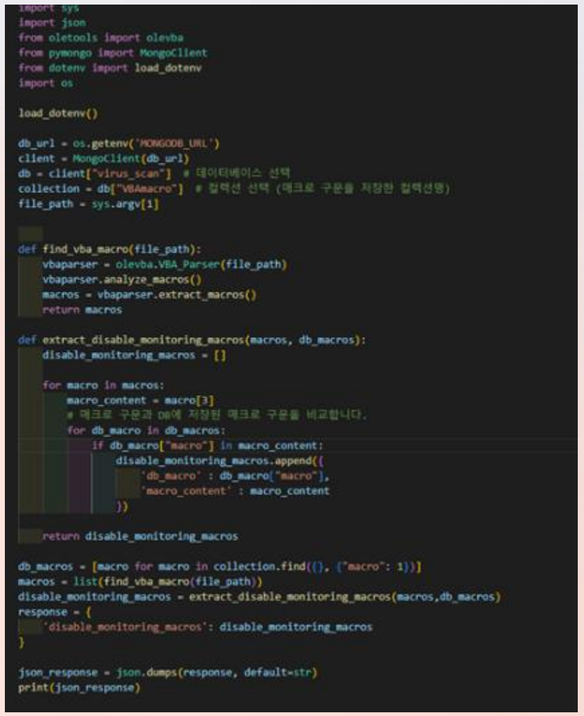
여기서 우리는 VBA 매크로 탐지를 우선적으로 구현하고자 한다.

매크로 탐지, 제거 기능 제작

ms문서 파일의 VBA 매크로 검사





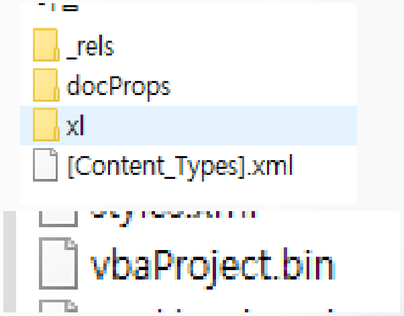


DB에 저장된 매크로 구문과 비교하여 어떤 VBA 매크로 구문이 어떤 식으로 작성되었는지 확인할 수 있다.
또한 oletools를 이용하면 ole 구조의 문서 파일 내 저장된 매크로를 쉽게 찾아낼 수 있다.

매크로 탐지, 제거 기능 제작

VBA 매크로 제거

```
zip1 = zipfile.ZipFile(file_path)
zip1.extractall(extract_path)
zip1.close()
```




```
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))
```

```
const result = JSON.parse(data) // 파이썬에서 반환된 JSON 파싱
const fixFilePath = result.file_path // file_path 값을 추출
const fileStream = fs.createReadStream(fixFilePath)
const originalExtension = path.extname(filePath)
const fileName = `fixed_file${originalExtension}` // 원본 확장자를 유지한 파일명 설정
res.setHeader('Content-Disposition', `attachment; filename="${fileName}"`) // 다운로드 시 파일명 설
fileStream.pipe(res)
```

문서 파일을 업로드 받을 시 문서 파일을 zip파일로 처리하여 압축을 해제한다.

기존 vbaProject.bin을 제외하고 새로 만든 파일을 포함시켜서 재압축한다.

이후 서버에서 웹페이지로 파일을 반환하여 클라이언트에게 제공한다.

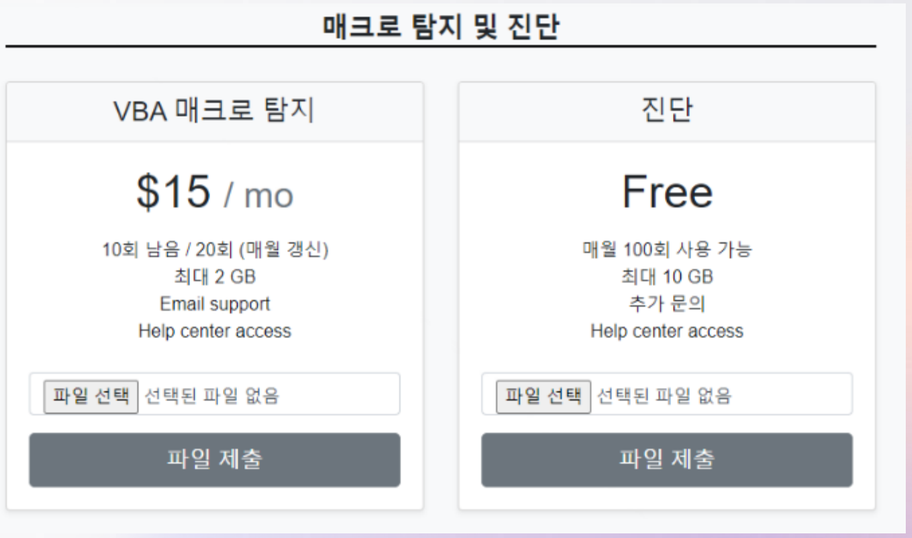
배포 환경 구성 및 연결

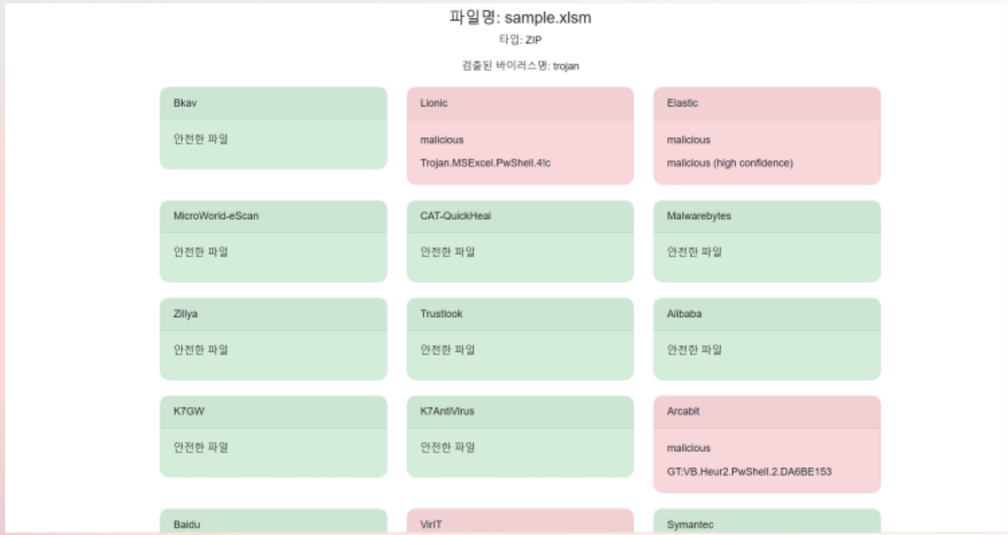
AWS EC2 인스턴스를 사용해 확장성과 안정성을 고려한 배포 환경을 구축하고, 가비아에서 구매한 도메인과 연결하였다.

MobaXterm은 직관적인 인터페이스와 SSH를 통한 원격 서버 관리 기능을 제공하여 배포 및 유지 관리를 용이하게 하였다. 이를 통해 클라우드 인프라와 원격 관리 도구를 결합한 효율적인 애플리케이션 배포 환경을 마련하였다.

프로젝트 결과

프로젝트 결과 화면



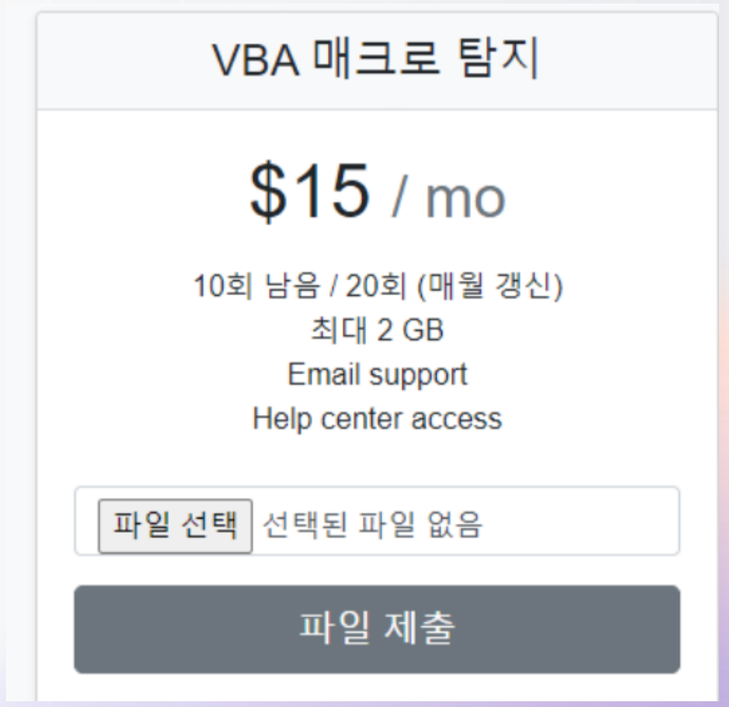


<파일 선택> 후
<파일 제출> 클릭

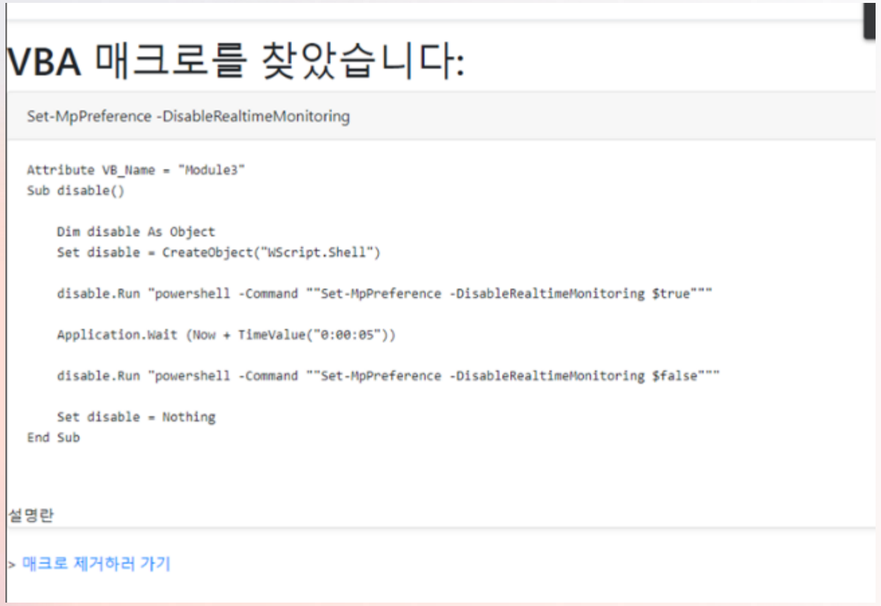
파일 검사 결과 확인

프로젝트 결과

프로젝트 결과 화면



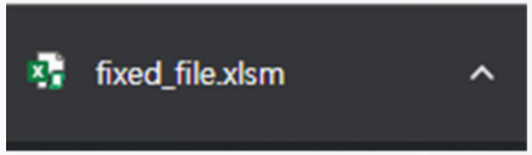
ms문서 파일의
VBA 매크로 검사

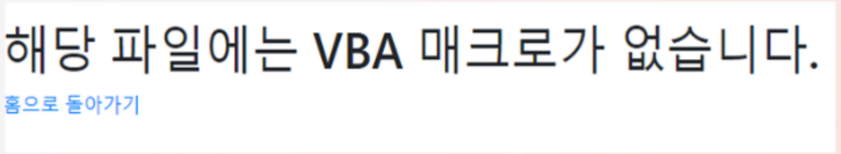


DB에 저장된 매크로 구문과 비교하여
어떤 VBA 매크로 구문이 어떤 식으로
작성되었는지 확인가능

프로젝트 결과

프로젝트 결과 화면

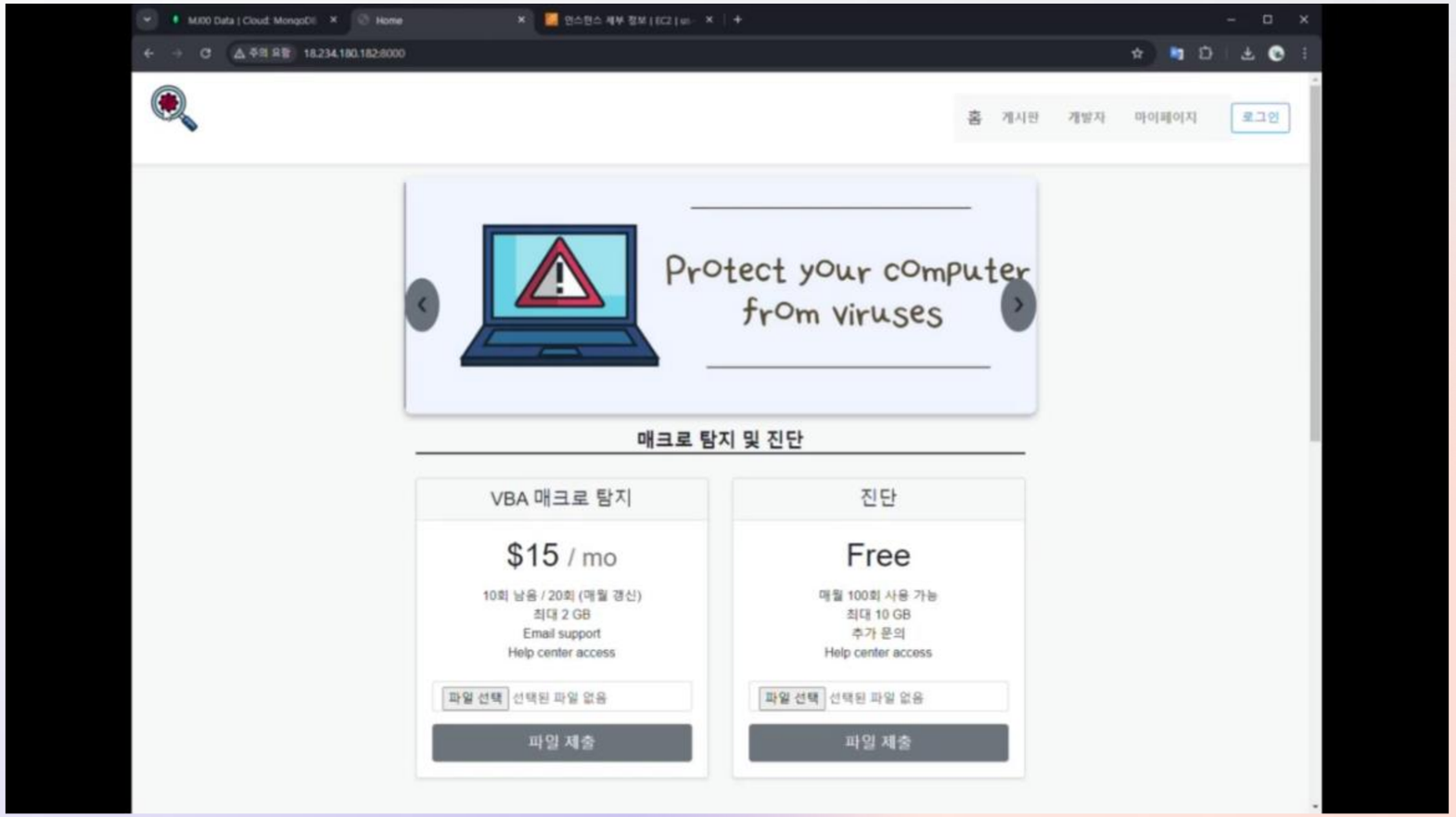




VBA 매크로 제거

프로젝트 결과

시연영상


18.234.180.182:8000
홈
게시판
개발자
마이페이지
로그인
Protect your computer from viruses
매크로 탐지 및 진단
VBA 매크로 탐지
$15 / mo
10회 남음 / 20회 (매월 갱신)
최대 2 GB
Email support
Help center access
파일 선택
선택된 파일 없음
파일 제출
진단
Free
매월 100회 사용 가능
최대 10 GB
추가 문의
Help center access
파일 선택
선택된 파일 없음
파일 제출

프로젝트 결과

유튜브 링크 & QR 코드

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

프로젝트 결과

기대효과

Point 01

악성코드 점검
효율성 증대

Point 02

자동 진단 도구 활용으로 인한
악성코드 진단 시간 단축

Point 03

빠른 취약점 위치 식별
잠재적 취약점 발생 위치
파악 및 대처

Point 04

학생들의
악성코드 점검 웹사이트를
직접 제작하고 운영함으로써
학생들의 실습 경험 제공

감사합니다

캐치버그핑
팀원 : 신명진 김태현 김호준 최경은 임혜준