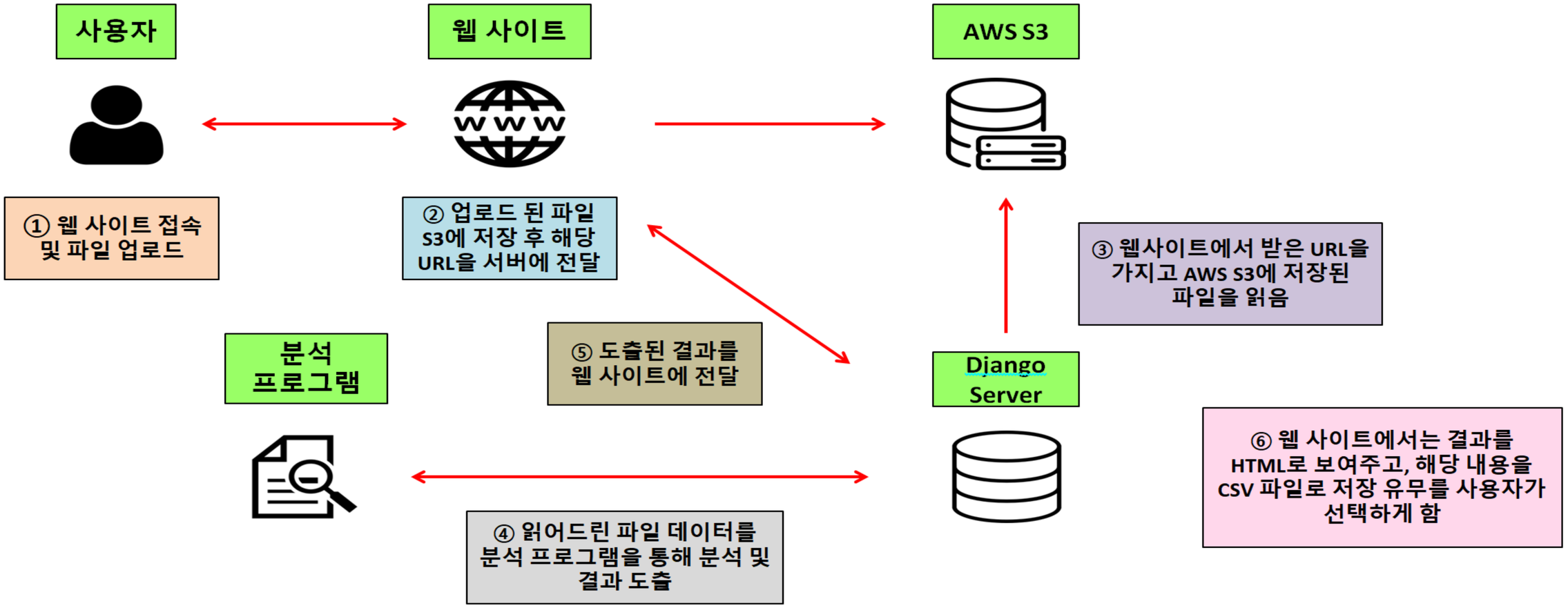
웹 기반 Windows Artifact 분석

중부대학교 정보보호학과
지도교수 : 이병천 교수님
분석해줄게조 지창환
김종식
염정현
최예지

목 차

구 상 도


사용자
웹 사이트
AWS S3
① 웹 사이트 접속 및 파일 업로드
② 업로드 된 파일 S3에 저장 후 해당 URL을 서버에 전달
③ 웹사이트에서 받은 URL을 가지고 AWS S3에 저장된 파일을 읽음
분석 프로그램
⑤ 도출된 결과를 웹 사이트에 전달
Django Server
⑥ 웹 사이트에서는 결과를 HTML로 보여주고, 해당 내용을 CSV 파일로 저장 유무를 사용자가 선택하게 함
④ 읽어드린 파일 데이터를 분석 프로그램을 통해 분석 및 결과 도출

추진 경과

수행업무 \ 추진기간	3월	4월	5월	6월	7월	8월	9월	10월
Artifact 분석 및 개발								
Web 개발								
서버 개발 및 Rest API 적용								
테스트 및 보완								




개발 환경 및 개발 내용 (1/10)

Web Server

Django
AWS-EC2
AWS-S3

DB

MongoDB

Web Front-end

Next.js
Node.js
Typescript
React
Tailwind CSS

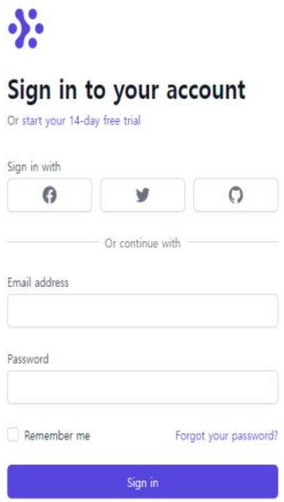
Development Language

Python
Typescript

Web Back-end

Next.js
Node.js
MongoDB
Git Oauth

개발 환경 및 개발 내용 (2/10)




```
import qs from 'qs';

import { signToken } from '@utils/jsonwebtoken';
import { withErrorHandler } from '@utils/with-error-handler';

import type { NextApiRequest, NextApiResponse } from 'next';

const client_id = process.env.GITHUB_ID;
if (!client_id) throw new Error('Missing GITHUB_ID');

const jwtSecret = process.env.JWT_SECRET;
if (!jwtSecret) throw new Error('Missing JWT_SECRET');

const SERVER_URL = process.env.SERVER_URL;
if (!SERVER_URL) throw new Error('Missing SERVER_URL');

const handler = async (req: NextApiRequest, res: NextApiResponse) => {
  if (req.method === 'GET') {
    const ourStateToken = signToken({}, { expiresIn: '1m' });

    const requestOptions = {
      client_id,
      redirect_uri: `${SERVER_URL}/api/oauth/callback/github`,
      state: ourStateToken,
    };

    res.redirect(
      `https://github.com/login/oauth/authorize?user:email&${qs.stringify(requestOptions)}`,
    );
    return;
  }
};

export default withErrorHandler(handler);
```

로그인 방식

저희가 제공하는 웹 사이트는 Github 로그인(회원가입)만을 제공하고 있음

개발 환경 및 개발 내용 (3/10)

```
      _id: exUser._id,
      connectedAccounts: { $elemMatch: { provider: { $eq: 'github' } } },
    },
    {
      $set: {
        'connectedAccounts.$.accessToken': access_token,
        'connectedAccounts.$.accessTokenExpires': new Date(Date.now() + expires_in),
        'connectedAccounts.$.refreshToken': refresh_token,
        'connectedAccounts.$.refreshTokenExpires': new Date(
          Date.now() + refresh_token_expires_in,
        ),
        'connectedAccounts.$.updatedAt': new Date(),
      },
    },
  );
} else {
  await db.collection<User>('user').updateOne(
    {
      _id: exUser._id,
    },
    {
      $push: {
        connectedAccounts: {
          provider: 'github',
          providerAccountId: id,
          accessToken: access_token,
          accessTokenExpires: new Date(Date.now() + expires_in),
          refreshToken: refresh_token,
          refreshTokenExpires: new Date(Date.now() + refresh_token_expires_in),
          createdAt: new Date(),
          updatedAt: new Date(),
        },
      },
    },
  );
}

const accessToken = signToken(
  { userId: encodeId(exUser._id) },
  { expiresIn: ACCESS_TOKEN_EXPIRES_IN },
);
res.setHeader('Set-Cookie', [
  serialize(COOKIE_KEY_ACCESS_TOKEN, accessToken, defaultCookieOptions),
]);

res.redirect(req.cookies[COOKIE_KEY_REDIRECT_URL] || '/home');
return;
```

DB 정보저장

GitHub에서 받은 정보를 바탕으로 DB에 등록 및 Cookie 등록

개발 환경 및 개발 내용 (4/10)

```ts
import { createPresignedPost } from '@aws-sdk/s3-presigned-post';
import Joi from 'joi';

// utils

import { verifySession } from '@lib/server/verify-session';

import { s3Client } from '@utils/aws/s3';
import { withErrorHandler } from '@utils/with-error-handler';

// types & defines

import type { NextApiRequest, NextApiResponse } from 'next';

const Bucket = process.env.AWS_PUBLIC_BUCKET_NAME;
if (!Bucket) throw new Error('Missing AWS_PUBLIC_BUCKET_NAME');

const awsPublicUrl = process.env.AWS_PUBLIC_URL;
if (!awsPublicUrl) throw new Error('Missing awsPublicUrl');

const expiresIn = 300;

const path = 'test/files';

const handler = async (req: NextApiRequest, res: NextApiResponse) => {
  verifySession(req, res);

  if (req.method === 'GET') {
    const querySchema = Joi.object({
      key: Joi.string().label('key').max(100).required(),
    });

    const { key } = (await querySchema.validateAsync(req.query)) as { key: string };

    const { url, fields } = await createPresignedPost(s3Client, {
      Bucket,
      Key: `${path}/${key}`,
      Conditions: [{ Bucket }, ['content-length-range', 1, 50 * 1024 * 1024]],
      Fields: { acl: 'public-read' },
      Expires: expiresIn,
    });

    return res.status(201).json({ url, fields });
  }
```

WS-S3에 업로드

사용자가 웹 사이트에 파일을 업로드 하게 되면, 웹에서 AWS-S3에 해당 파일을 업로드

개발 환경 및 개발 내용 (5/10)

```ts
import { fetcher } from './fetcher';

export default async function uploadFileAWS(file: File) {
  try {
    const { url, fields } = await fetcher
      .get('/api/aws/presigned-post', { searchParams: { key: file.name } })
      .json<{ url: string; fields: { [key: string]: string } }>();

    const formData = new FormData();
    Object.entries({ ...fields }).forEach(([key, value]) => {
      formData.append(key, value);
    });

    formData.append('file', file, file.name);

    const response = await fetch(url, { method: 'POST', body: formData });
    if (!response.ok) {
      throw new Error(await response.text());
    }

    return `${url}/test/files/${file.name}`;
  } catch (err) {
    console.log('[uploadFileAWS error]', err);
    throw new Error(err);
  }
```

WS-S3에 업로드

사용자가 웹 사이트에 파일을 업로드 하게 되면, 웹에서 AWS-S3에 해당 파일을 업로드

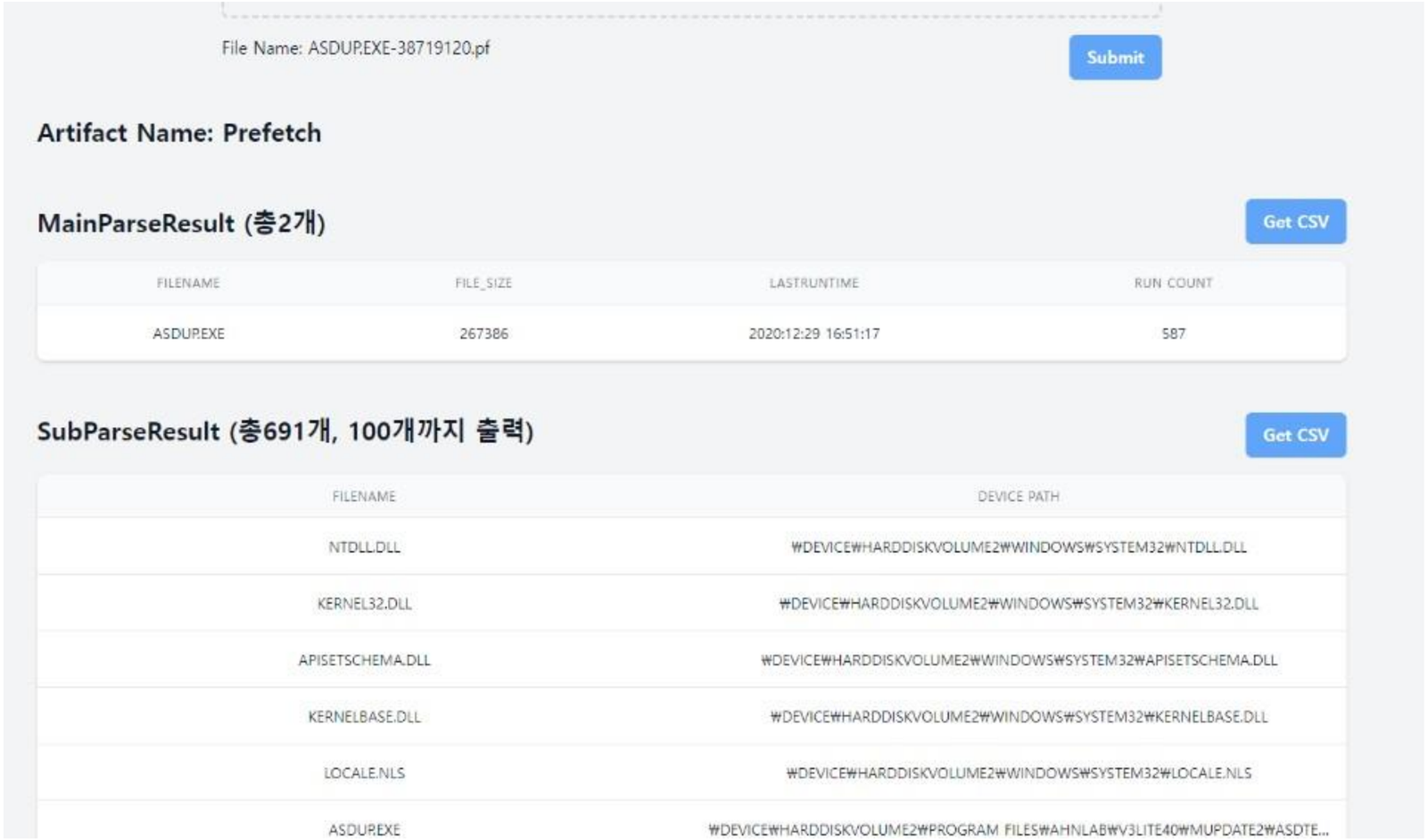
개발 환경 및 개발 내용 (6/10)

```ts
import got from 'got';
import Joi from 'joi';

import { withErrorHandler } from '@utils/with-error-handler';

import type { NextApiRequest, NextApiResponse } from 'next';

const parseUrl = process.env.PARSE_SERVER_URL;

if (!parseUrl) throw new Error('No such url');

const handler = async (req: NextApiRequest, res: NextApiResponse) => {
  const querySchema = Joi.object({
    url: Joi.string().label('url').required(),
  });

  const { url } = (await querySchema.validateAsync(req.query)) as { url: string };

  if (!url) throw new Error('Error!!');
  if (req.method === 'POST') {
    const data = await got.get(parseUrl, { searchParams: { urllink: url } }).json();

    return res.json({ data });
  }
};

export default withErrorHandler(handler);
```

서버로 URL 전송

AWS-S3에 저장된 파일의 Url을 서버로 전송 및 분석 결과를 JSON 형태로 받음

개발 환경 및 개발 내용 (7/10)



분석된 결과

분석된 결과를 웹 사이트에서 바로 확인할 수 있고, CSV 파일로 다운 받을 수 도 있다.

개발 환경 및 개발 내용 (8/10)

```python
@csrf_exempt
def s3_url(request):
    if request.method == 'GET':
        s3_url = request.GET['urllink']

        fixed_s3_url = ['https://jongsik-exam.s3.ap-northeast-2.amazonaws.com/', 'https://hwans

        if fixed_s3_url[0] in s3_url or fixed_s3_url[1] in s3_url:

            a = s3_url
            b = a.split("/")
            file_name = b[-1]
            s3_req = requests.get(s3_url, stream=True)      You, seconds ago • Uncommitted cha
            data = bytes()
            data = s3_req.raw.read()

            Result_data = parser.parse_(data, file_name)

            if Result_data != None:
                return JsonResponse(Result_data)
            else: return HttpResponse("don't support format")


        else: return HttpResponse("URL Error")
```

웹에서 전달 받은 URL

전달받은 Url을 미리 정의된 url과 비교 후 AWS-S3에 접근하여 파일을 읽어옴

개발 환경 및 개발 내용 (9/10)

```python
def parse_(data, filename):
    prefetch_sig_1 = bytes(b'\x4D\x41\x4D')
    prefetch_sig_2 = bytes(b'\x53\x43\x43\x41')
    lnk_sig = bytes(b'\x4C\x00\x00\x00')

    if data[0:3] == prefetch_sig_1 or data[4:8] == prefetch_sig_2:
        result_pf = pf.prefetch_(data, filename)
        return result_pf
    elif data[0:4] == lnk_sig:
        result_lnk = lnk.lnk_par
        return result_lnk
    else:
        result_else = None
        return result_else
```

```python
Result_data = parser.parse_(data, file_name)

if Result_data != None:
    return JsonResponse(Result_data)
else: return HttpResponse("don't support format")
```

파일 Signature 비교

미리 정의된 Signature를 비교해서 해당 하는 분석 로직으로 데이터를 다시 한번 전달

개발 환경 및 개발 내용 (10/10)

```python
win10_prefetch_sig = bytes(b'\x4D\x41\x4D') # prefetch 맞는지 ?

filename_ = "win10_decompress_pf" #윈10 프리패치의 경우 압축해제가 필요해 파일로 저장
if data[0:3] == win10_prefetch_sig:
    print('succes')  # win10

    with open(filename_, 'wb') as win10:
        win10.write(data)

    data = compressed.decompress(filename_)
    You, 2 weeks ago • django
    file_size = struct.unpack_from("<L", data[12:])[0]
    filename = filename_change(data[16:46])
    Last_Run_Time = dt_from_win32_ts(int.from_bytes(
        data[128:136], byteorder='little', signed=True)).strftime('%Y:%m:%d %H:%M:%S')
    Run_Count = struct.unpack_from("<L", data[0xD0:])[0]
    FileNameInfoOffset = struct.unpack_from("<L", data[0x64:])[0]
    FileNameInfoSize = struct.unpack_from("<L", data[0x68:])[0]
    load_file = binascii.hexlify(bytes(
        data[FileNameInfoOffset:FileNameInfoOffset + FileNameInfoSize])).decode("utf-8").split
    load_file = [i.replace("00", "") for i in load_file]  # 00제거

    json_ = {'artifactName': 'Prefetch', 'mainParseResult': [], 'subParseResult':[]}
    json_data = []
    json_data.append(["FileName", "File_Size", "LastRunTime", "Run count" ])
    json_data.append([filename, file_size, Last_Run_Time, Run_Count])
    json_['mainParseResult'] = json_data
    json_data2 = []
    json_data2.append(["FileName","Device Path"])

    for i in load_file:
        if i == "00":
            break
        try:
            data = binascii.unhexlify(i).decode("utf-8")
            data_filename_10_1 = data.split("\\")
            #wr.writerow([data_filename_10_1[-1], data])
            json_data2.append([data_filename_10_1[-1], data])
        except:
            data_filename_10_2 = data.split("\\")
```

파일 분석 로직

Signature를 확인 후 데이터를 전달 받은 분석 로직에서 Byte 데이터를 각 파일 구조에 맞춰 분석 및 변환해서 의미있는 데이터 값으로 변경 및 JSON 형태로 Return

결론 및 기대 효과

결　　론

- Django 서버 및 웹 서버를 연동하여 사용자가 파일 업로드 하면 분석해주는 웹 사이트를 개발/구축하는데 성공

- 파일들에 대한 구조 및 설명을 웹 사이트 내에서 제공

기대효과

- 프로젝트 완성을 통해 각각의 툴을 다운로드 하여 CLI 환경으로 파일 분석을 할 필요 없이 편리성 및 시간단축을 시킴

Q & A

감사합니다