Web 기반
Windows Artifact 분석

팀　　명 : 분석해줄게조
지도 교수 : 이병천 교수님
팀　　장 : 지창환
팀　　원 : 김종식
염정현
최예지

2021. 10
중부대학교

목 차

1. 서론

1.1 연구 배경

연구의 배경으로는 디지털 포렌식을 위해 모여서 주제를 선정하던 중, 윈도우 아티팩트에 흥미를 느끼게 되어 이와 관련한 도구 개발을 시작하게 되었다. 또한 최근 침해사고는 단순한 바이러스나 웜 등의 유포가 아닌 개인정보 및 기업기밀 정보를 취득하거나 금전적 이득을 취하기 위한 목적으로 장시간에 걸쳐 지능적이고 다양한 최신의 공격기법이 활용되고 있다. 이에 따라 침해사고의 위협은 점차 증가되고 있으며, 침해 기관의 피해 또한 대형화되어 해당 기관의 존립과도 직결되기도 한다. 디지털 증거는 쉽게 변경될 수 있고, 특정 악성코드는 고도로 지능화된 은닉기법의 사용으로 특정 프로그램을 시스템 파일처럼 위장하고, 관련 설치 파일을 제거하여 흔적을 남기지 않기도 한다. 디지털 증거의 효과적 분석을 위하여 디지털 포렌식 도구가 점차 필수적으로 사용되고 있다.

1.2 연구 목표

이번 연구는 사용자의 불편함을 덜어 줄 수 있도록 각 분석 툴을 다운로드 할 필요 없이 웹 사이트에서 편리하게 파일만 업로드하게 되면 분석을 진행해서 원하는 결과를 받아 볼 수 있도록 계획했다. 또한 현재 다운로드 받아서 사용하는 툴 외에 웹 사이트에서 Artifact를 분석해주는 사이트가 존재하지 않는 것 또한 알게 되었다. 그리고 Artifact의 구조에 대한 설명이 한 곳에 모여 있는 사이트 또한 존재하지 않는다. 결론적으로 본 연구의 초점은 새로운 방법의 시도와 사용자의 편리성에 중점을 두고 진행 하였고, 웹을 통해 수집된 또는 분석이 필요한 윈도우 아티팩트 파일에 대해 분석하고 결과 값을 웹에서 보여주고, CSV 파일로도 제공한다. 또한 윈도우 아티팩트의 개념, 구조와 오픈소스 툴 링크까지 제공한다. 그리고 우리는 LNK, Prefetch, Jumplist, EventLog를 지원하고 있다.

2. 관련연구

2.1 Windows

Microsoft에서 개발한 컴퓨터 운영체제이며, 컴퓨터 역사상 가장 대중적으로 널리 알려지고, 가장 많이 사용되고 있는 운영체제이다.

2.1 Windwos Artifact

Artifact란 포렌식적인 의미에서 생성되는 흔적(증거)라고 볼 수 있다. 또 Windows Artifact는 윈도우가 가지고 있는 특유의 기능들과 그 기능을 구현하는데 필요한 요소들(주로 파일이나 레지스트리)로부터 찾을 수 있는 여러 가지 정보를 말

한다. Artifact의 종류로는 파일 시스템, 웹, 이벤트 로그, 프리패치, 바로가기, 점프리스트, 시스템 복원 지점, 휴지통, 시스템 로그, 임시파일 등이 있고, 해당 하는 정보로 특정 프로그램 실행 흔적, 시간 등 다양한 정보를 찾아 볼 수 있다.

2.2.1 $MFT

$MFT는 Master File Table의 약자로, NTFS File System에 있어 가장 핵심적인 부분이며, 볼륨에 존재하는 모든 파일과 디렉터리에 대한 정보를 가지고 있으며, 1개 이상의 MFT Entry가 존재하고, 파일의 크기가 클 경우 데이터를 여러 개로 나눠서 담으며 각 MFT Entry 값이 존재한다. 또한 MFT Entry의 크기는 모두 동일하며, 파일크기, 생성, 수정, 접근 시간 및 사용 권한, 파일명, 데이터 내용 등 파일에 관한 모든 정보는 $MFT에 저장되어 관리된다.

2.2.2 $LogFile

NTFS 트랜잭션 로그 파일을 말하며, MFT Entry Index 2번에 위치하며 각 볼륨마다 하나 씩 존재한다. 만약 작업 중 갑작스런 파일 손실 시, 복구를 위해 사용된다. 또한 모든 트랜젝션 작업을 레코드 단위로 기록하여, 파일의 생성, 삭제, 수정 등을 기록하며, 해당 정보를 확인 할 수 있다.

2.2.3 $UsnJrnl

NTFS의 메타데이터를 구성하는 파일이며, $UsnJrnl이라는 이름으로 존재하며 파일시스템의 모든 파일 및 디렉터리의 변경 사항을 기록하는 로그이다. 여기에서는 응용 프로그램ㅣ 특정 파일의 변경 여부를 파악하기 위해 사용하고, 파일의 생성, 변경, 추가 등을 파악할 수 있는 중요한 로그이다.

2.2.4 Registry

윈도우 레지스트리는 마이크로소프트 윈도우 32/64비트 버전과 윈도우 모바일 운영 체제의 설정과 선택항목을 담고 있는 데이터베이스트로, 모든 하드웨어, 운영체제 소프트웨어, 대부분의 비운영 체제 소프트웨어, 사용자 PC 선호도 등에 대한 정보와 설정이 들어 있다. 포렌식 관점에서는 운영체제 정보, 사용자계정정보, 시스템정보, 응용프로그램 실행 흔적, 자동실행항목, 저장매체사용흔적 등 다양한 정보를 담고 있다.

2.2.5 Prefetch

프리패치란 Windows 운영체제에서 응용프로그램이 사용하는 시스템 자원을 특정 파일에 미리 저장하여, 프로그램 실행을 빠르게 도와주는 파일이다. 이 파일에는 프로그램 이름, 실행 횟수, 마지막 실행시각, 참조파일목록, 파일시스템 시간 정보 등이 저장된다. 따라서 프리패치 분석을 통해 프로그램의 마지막 실행 시간을 확인

할 수 있고 악성 코드 실행 흔적 또한 발견할 수 있다. 해당 파일은 윈도우 XP 이후의 운영체제부터 제공을 하고 있다.

2.2.6 EventLog

윈도우 이벤트로그란 감사 설정된 시스템의 모든 기록을 담고 있는 데이터라고 할 수 있다. 이러한 데이터에는 성능, 오류, 경고 및 운영 정보 등의 중요 정보가 기록되며, 특별한 형태의 기준에 따라 숫자와 기호 등으로 이루어져 있다. 버전에 따라 차이가 있지만 가징 기본이 되는 세 가지 로그는 응용 프로그램 로그, 보안 로그, 시스템 로그가 있다.

2.2.7 Shortcut(LNK)

바로가기 파일(링크파일) 이라고도 불리며, Windows Shortcut, 공식 명칭은 'Shell Link' 라고 불린다. 윈도우에만 존재하는 기능으로 응용 프로그램, 디렉토리, 파일 등의 객체를 참조하는 파일이며, .lnk 확장자를 가지고 GUI에서만 동작한다. 응용 프로그램 설치 시에 바탕화면이나 시작 메뉴, 빠른 실행 폴더 등에 바로가기가 생성이 가능하며, 사용자 필요에 따라 생성이 가능하다. 포렌식 관점에서는 링크 대상 파일에 관한 생성, 접근, 수정 시간 정보 및 원본 위치 등에 관한 기록을 포함하고 있으므로, 정보 유출에 관한 조사나 시스템 사용에 관한 시간 관계를 정리할 때 유용하게 사용되는 경우가 많다.

2.2.8 JumpList

Windows 7에서 새롭게 추가된 기능으로 응용프로그램 별로 그룹화 하고 최근 실행한 파일에 대한 링크 및 정보를 관리하는 파일이다. 점프리스트의 파일구조는 OLE 형식으로 OLE 구조에서 데이터가 저장되어 있는 Stream 부분이 Link 구조를 갖고있어 OLE 구조와 Link 구조 모두 알고 있어야 한다. 해당 파일에서 얻을 수 있는 정보로는 프로그램 실행 유무 및 자주 사용하는 문서, 최근에 사용한 문서, 프로그램 정보 등이 있다.

2.3 Python

Python은 1991년 프로그래머인 Guido van Rossum이 발표한 고급 프로그래밍 언어로 플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 타이핑 대화형 언어이다. Python은 비영리의 Python 소프트웨어 재단이 관리하는 개방형, 공동체 기반 개발 모델을 가지고 있고, C언어로 구현된 C 파이썬 구현 사실상의 표준이다.

2.4 Django

Python으로 작성된 오픈 소스 웹 프레임워크로, 모델-뷰-컨트롤러(MVC) 패턴을 따르고 있다. 현재는 장고 소프트웨어 재단에 의해 관리되며, 고도의 데이터베이스

기반 웹사이트를 작성하는데 있어서 수고를 더는 것이 장고의 주된 목표이다.

2.5 Linux(Ubuntu)

우분투는 컴퓨터에서 프로그램과 주변기기를 사용할 수 있도록 해주는 운영체제 중 하나이다. 안드로이드 운영체제처럼 리눅스 커널에 기반한 운영체제로 모바일과 데스크톱 PC, 서버에도 우분투 운영체제를 설치해 사용할 수 있다. 리눅스는 리누스 토발즈라는 개발자가 어셈블리어라는 프로그래밍 언어로 유닉스를 모델 삼아 개발한 오픈소스 운영체제이다. 우분투는 리눅스 OS의 배포판 중 하나로 특히 데스크톱 PC에서 사용할 수 있게 특화된 운영체제이다.

2.6 Node.js

Node.js는 Chrome V8 JavaScript 엔진으로 빌드된 JavaScript 런타임이다. Node.js는 이벤트 기반, 논 블로킹 I/O 모델을 사용해 가볍고 효율적이다. Node.js의 패키지 관리 모듈인 npm은 세계에서 가장 큰 오픈소스 라이브러리이며 해당 프로젝트는 npm대신 동일한 역할을 하는 yarn을 사용하였다.

2.7 JavaScript

JavaScript는 웹 브라우저에서 실행하는 스크립트 언어를 작성하는 웹을 동적으로 꾸밀 때 가장 널리 쓰이는 언어이다. 해당 프로젝트의 React는 JavaScript의 라이브러리이다.

2.8 React

React는 JavaScript 라이브러리의 하나로서 facebook에서 제공하는 프론트엔드 라이브러리이다. SPA 기반으로 동작하며 Virtual Dom이라는 기술을 사용하여 페이지를 렌더한다. 비슷한 라이브러리로는 Vue.js Angular.js가 있으며 React가 사용자 수와 오픈 소스가 가장 많으며 Vue, Angular순으로 많이 사용된다.

2.9 Typescript

TypeScript란 MicroSoft에서 개발하고 관리하는 오픈소스 프로그래밍 언어이다. 어떤 브라우저나 호스트, 운영체제에서도 동작하며 JavaScript의 상 집합으로 ECMA의 최신 표준을 충분히 지원한다. 타입이라는 특징으로 JavaScript의 단점을 극복하였으며 ES5 JavaScript의 문법을 그대로 사용할 수 있다.

2.10 Babel

Babel이란 JavaScript 컴파일러이다. Babel의 input은 JavaScript 코드이고 output 또한 JavaScript코드이다. 최신 버전의 JavaScript 문법은 브라우저가 이해하지 못하기에 Babel이 이를 이해할 수 있는 문법으로 변환해주는 핵심적인 역할을 한다.

2.11 Next.js

Next.js는 React SSR을 쉽게 구현할 수 있게 도와주는 프레임워크이다. React 자체가 SSR을 고려하여 설계되었기 때문에 자체적으로도 구현이 가능하지만 개발환경을 만들기 위해서는 복잡한 과정을 거쳐야하고 이러한 문제를 NextJS가 해결해준다. 또한 CSR을 SSR과 같이 혼합하여 사용하기도 쉽게 만들어주는 역할을 한다.

2.12 SWR

SWR은 Vercel에서 만든 CSR을 손쉽게 만들어주는데 도움을 주는 커스텀 hook이다. 이는 Next.js의 CSR생태계에 있어 가장 핵심적인 역할을하며 CSR기능을 구현하는데 꼭 사용하지 않아도 되지만 사용하지 않을 이유가 없을 정도로 압도적인 장점을 가지는 hook이다. CSR의 가장 큰 단점인 매 fetch마다 component의 상태가 변한다는 단점을 극복하는데 도움을 주는 역할을 한다. 또한 웹 개발에 있어 매우 까다로운 작업인 context를 대체할 수 있는 기능또한 가지고 있다.

2.13 TailwindCSS

TailwindCSS는 HTML안에서 CSS스타일을 만들 수 있게 해주는 부트스트랩과 같은 역할을 하는 CSS프레임워크이다. 같은 역할을하지만 성능이나 편의성 모든 면에서 TailwindCSS가 더욱 뛰어난 결과를 보여주며 코드의 생산성면이나 협업적인 면에서는 현재 존재하는 관련 라이브러리중 압도적인 장점을 자랑하고 있다.

2.14 MongoDB

NoSQL기반의 데이터베이스이다. MongoDB는 BSON형태로 데이터를 저장하는 문서지향의 데이터베이스이며 스키마의 제약이 없어 자유로우며 기존 RDMS에서 지원하지 않는 형태로도 저장할 수 있기 때문에 사용하기 쉽고 이해하기도 쉽다. 또한 JSON을 사용하기에 아주 적합한 데이터베이스이므로 Node.js와 호환이 매우 좋다.

2.15 AWS-S3

Amazon Simple Storage Service의 약자로 웹에서 데이터를 저장하거나 검색하기 편하도록 Storage 기능을 제공해주는 역할을 한다. S3는 각종 권한에 대한 설정이나 Cols에 대한 설정 또한 가능하므로 사용자가 원하는대로 Access control을 하여 버킷을 Control할 수 있다.

2.16 AWS-EC2

Amazon Elastic Compute Clout의 약자이며 클라우드 컴퓨팅의 역할을 하는 웹서비스이다. 사용자가 원하는대로 확장이 가능하며 원하는 수의 가상 서버를 구축

하고 보안 및 네트워킹을 구성하며 Storage또한 관리할 수 있다. 이름 그대로 확장이나 축소가 가능하기 때문에 트래픽을 예측할 필요성이 줄어든다는 장점이 있다.

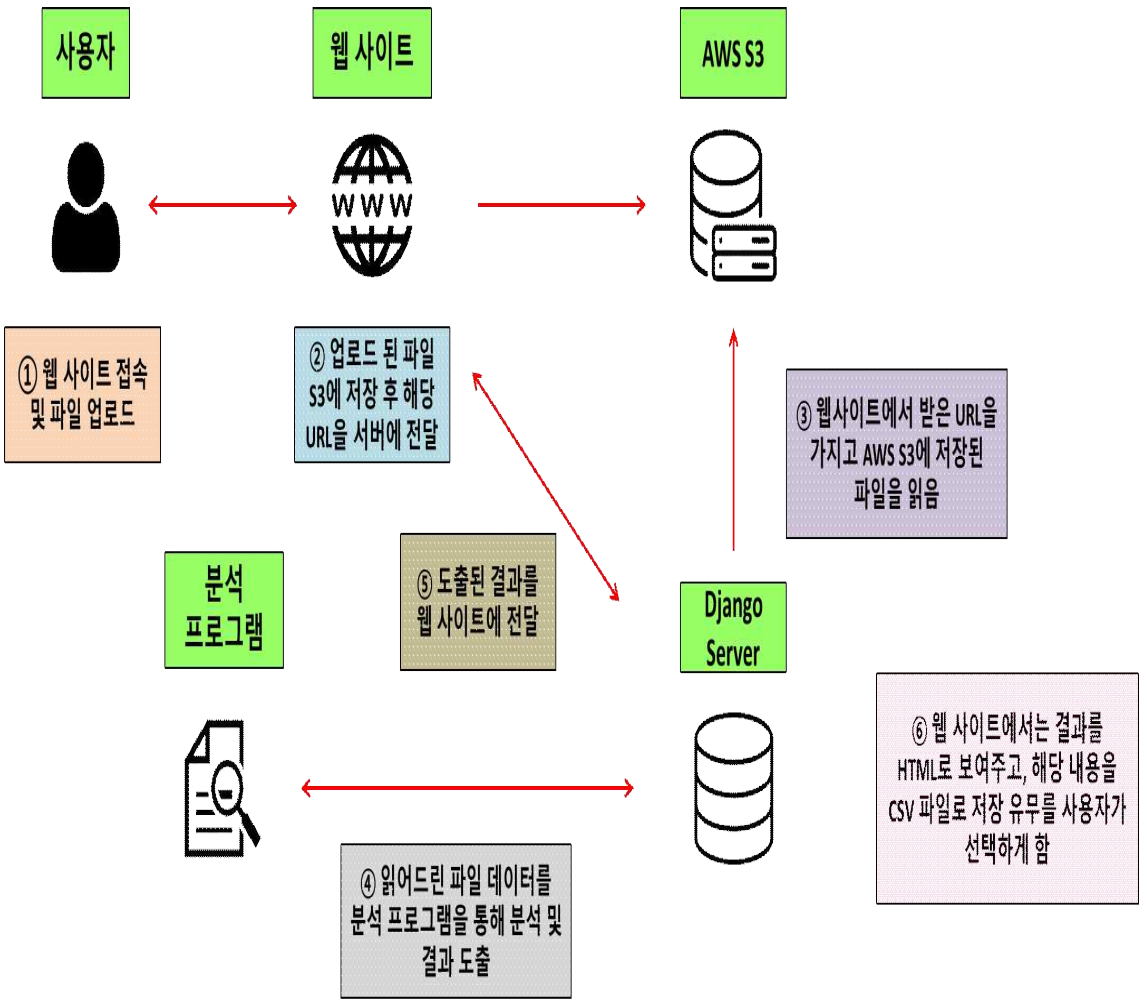
2.17 Vercel

사용자의 Work flow에 완벽하게 맞는 정적 사이트 및 서버리스 기능을 위한 클라우트 플랫폼이다. Vercel은 Configuration, Supervision 없이 즉시 배포할 수 있고 자동으로 확장하며 웹 사이트 및 웹 서비스를 호스팅할 수 있어 웹 배포에 있어서 해결해야하는 각종 까다로운 과정을 겪지 않아도 된다.

3. 본론

3.1 서비스 설계

사용자가 웹을 통해서 본 프로젝트를 배포하여 Desktop, Mobile 환경에서 프로젝트를 체험해볼 수 있도록 제작하였으며, 어느 디바이스에서 시행해도 어색하지 않도록 반응형 웹으로 제작하였다.



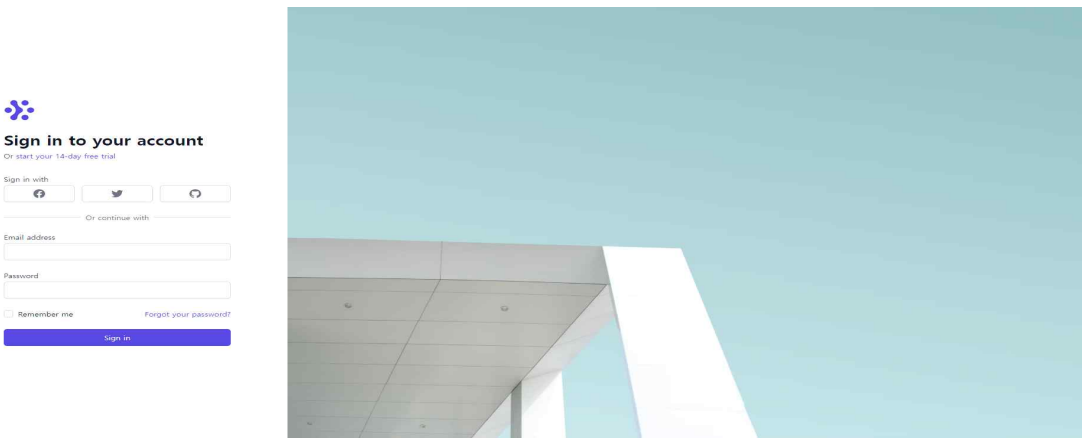
3.2 구현 환경

Web 배포	Web Front-end	Web Back-end	Data Parsing Server
Vercel	Next.js	Next.js	Python
	Node.js	Node.js	Django
	Typescript	MongoDB	AWS-S3
	React	Git oauth	AWS-EC2
	Tailwind CSS	AWS-S3	

3.3 구현 과정 설명

3.3.1 웹 구성

Typescript와 React를 기반으로 Next.js를 사용하여 웹을 제작하였다. Next.js는 Serverless기반이며 Express와 연동하여 사용할 수도 있지만 본 프로젝트에서는 순수 Next.js만을 사용하여 웹 서버를 구성하였으며 Front-end는 React를 사용하였다.
웹의 기본적인 동작은 파싱 서버로 어떠한 파일의 다운로드 Url을 보내면 해당 파일을 분석, 파싱하여 결과 값을 Return하며 그 결과 값을 웹상에서 CSV 파일로 만들어서 유저가 다운받을 수 있도록 구성하였다. 또한 서비스는 기본적으로 Github Oauth를 사용하여 로그인이 된 유저만 사용 가능하도록 구성하였다.

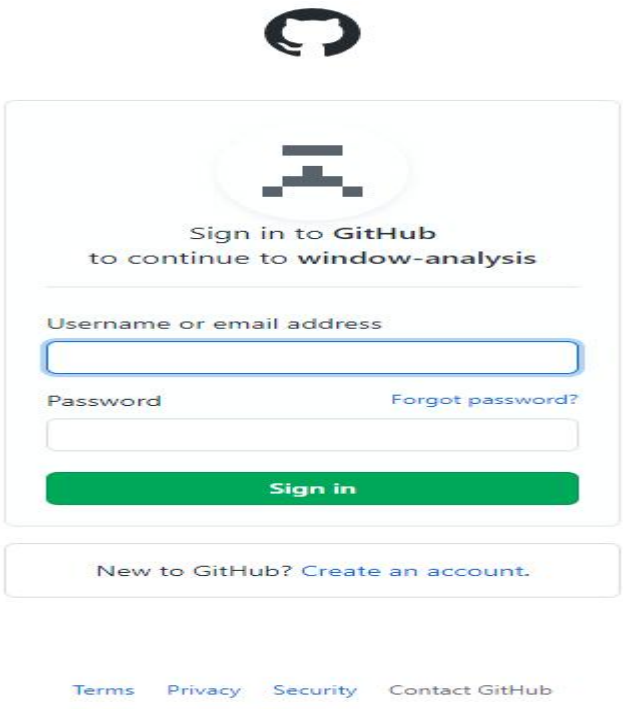


[그림 1. Sign-in 화면]

먼저 웹에 접속하여 로그인 버튼을 누르면 [그림.1] 페이지를 확인할 수 있으며 해당 페이지에서 Git 아이콘을 클릭하면 Github Oauth페이지로 Redirect 하게 된다.

```ts
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);
```

[그림 2. Git Provider 코드]



[그림 3. Git Login Page]

이후 정상적으로 로그인을 진행한다면 Github에 등록해놓은 Redirect Uri를 통하여 해당 웹 프로젝트로 다시 Redirect 시켜준다. 이후 Github에서 받은 정보를 바탕으로 Callback api에서 DB에 User등록 및 Authrization 과정을 거쳐 AccessToken을 Cookie에 등록하며 로그인이 완료된다. 그리고 다시 해당 프로젝트의 메인페이지로 이동하게 된다.

```
      _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;
```

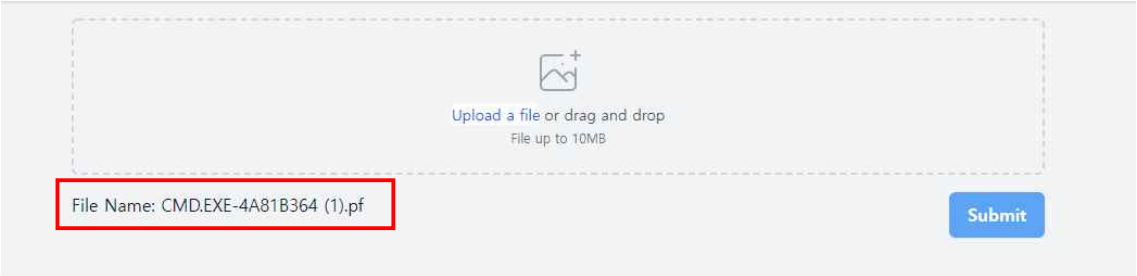
[그림 4. DB 업로드 및 Cookie 등록]

Filter Only show cookies with an issue

Name	Value	Do...	Path	Exp...	Size	Htt...	Sec...	Sa...	Sa...	Prio...
jb.accessToken	eyJhbGciOiJIUzUxMiIsInR5cCI6I...	loc...	/	Ses...	228	✓		Lax		Me...

[그림5. 쿠키가 정상적으로 등록된 사진]

해당 쿠키를 통하여 유저의 로그인 유무를 확인하고 쿠키가 만료되거나 없는 상태라면 index페이지로 유저를 강제 Redirect 시킨다. 이후 분석을 원하는 파일을 Drag&Drop 혹은 직접 클릭하여 업로드 하여준다.



[그림 6.파일 업로드가 완료된 모습]

[그림6]와 같이 업로드를 완료한 뒤 Submit 버튼을 누르면 파싱 서버로 해당 파일을 전송하게 되고 파싱 서버에서 해당 파일을 분석하여 결과 값을 JSON형태로 Return한다.

```
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);
```

[그림 7. 파일 Url을 파싱서버로 전송하고 결과 값을 받는 코드]

또한 해당 파일을 업로드한 후 Submit 버튼을 누르면 자동으로 AWS-S3에 Presigned-Post를 사용하여 파일을 업로드한 뒤 해당 파일의 Read 권한을 Public으로 설정하여 다운로드 받을 수 있는 Url을 파싱 서버로 보내게 된다.

```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);
  }
```

[그림 8. Presigned-Post lib 코드]

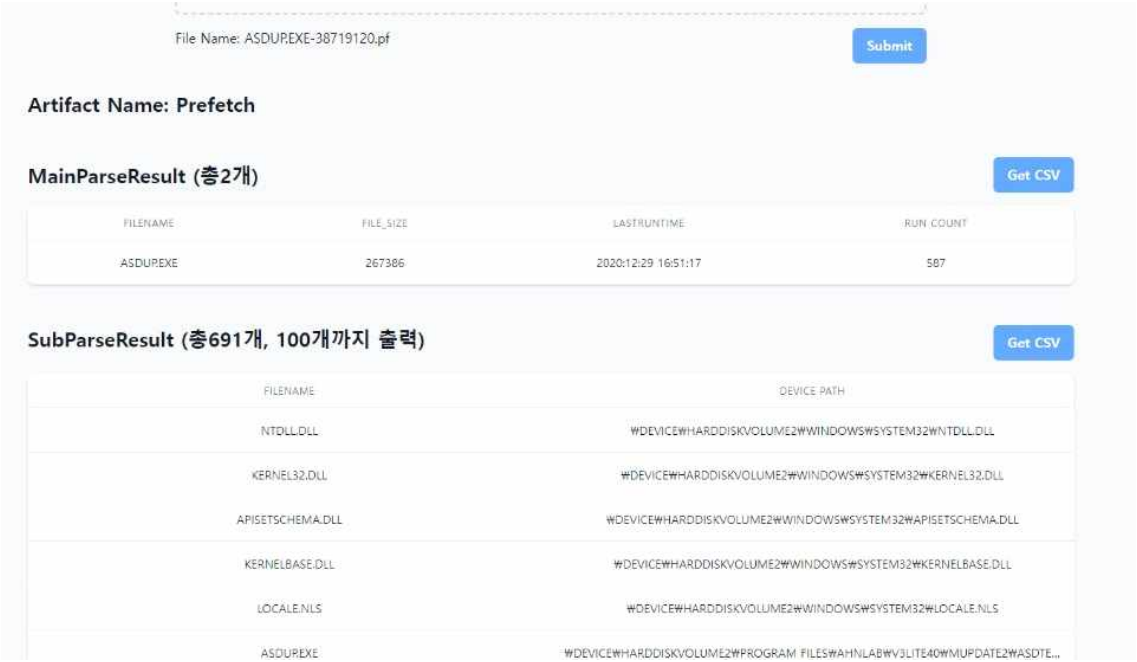
```ts
import { createPresignedPost } from '@aws-sdk/s3-presigned-post';
import Joi from 'joi';

// 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 });
  }
```

[그림 9. Presigned-Post API 코드]

ASDUP.EXE-38719120.pf	pf	October 7, 2021, 14:08:08 (UTC+09:00)	261.1 KB	Standard
CMD.EXE-4A81B364 (1).pf	pf	October 21, 2021, 21:30:59 (UTC+09:00)	6.1 KB	Standard
CMD.EXE-4A81B364.pf	pf	October 21, 2021, 21:31:40 (UTC+09:00)	6.1 KB	Standard
HXD.EXE-0683C898.pf	pf	October 6, 2021, 23:52:28 (UTC+09:00)	29.3 KB	Standard

[그림 10. AWS-S3에 정상적으로 파일이 업로드된 모습]

이후 파싱 서버를 통하여 데이터를 받은 후 해당 데이터를 웹 프로젝트에서 다시 Table 형식으로 Render하게 된다. 만약 파일 분석에 실패한다면 에러 메시지를 출력한다.



[그림11. 분석된 파일 정보가 웹에 출력된 결과]

[그림11]과 같이 분석에 성공한 파일은 Get CSV버튼 클릭 시 CSV파일로 개별로 출력할 수 있다.

```
<CSVLink filename='parse-result.csv' data={[...mainParseResult,[],...subParseResult]}>
  <Button>Get CSV</Button>
</CSVLink>
```

[그림 12. CSV 버튼 코드 (react-csv package 사용)]

FileName	File_Size	LastRunTir	Run count
HXD.EXE	29984	1601:01:01	2

FileName	Device Path
NTDLL.DLL	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩NTDLL.DLL
KERNEL32	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩KERNEL32.DLL
UNICODE.	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩UNICODE.NLS
LOCALE.N	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩LOCALE.NLS
SORTTBLS	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩SORTTBLS.NLS
HXD.EXE	₩DEVICE₩HARDDISKVOLUME1₩PROGRAM FILES₩HXD₩HXD.EXE
OLEAUT32	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩OLEAUT32.DLL
ADVAPI32	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩ADVAPI32.DLL
RPCRT4.DI	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩RPCRT4.DLL
SECUR32.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩SECUR32.DLL
GDI32.DLL	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩GDI32.DLL
USER32.DI	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩USER32.DLL
MSVCRT.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩MSVCRT.DLL
OLE32.DLL	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩OLE32.DLL
VERSION.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩VERSION.DLL
SHELL32.D	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩SHELL32.DLL
SHLWAPI.I	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩SHLWAPI.DLL
COMCTL3	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩WINSXS₩X86_MICROSOFT.WINDOWS.COMMON-CONTROLS_6595B64144CCF1DF_6.0.2600.5512_X-WW_35D4CE83₩COMCTL32.DLL
WININET.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩WININET.DLL
CRYPT32.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩CRYPT32.DLL
MSASN1.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩MSASN1.DLL
COMDLG3	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩COMDLG32.DLL
WINSPOO	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩WINSPOOL.DRV
WINMM.D	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩WINMM.DLL
OLEACC.D	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩OLEACC.DLL
MSVCP60.	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩MSVCP60.DLL
IMM32.DL	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩IMM32.DLL
LPK.DLL	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩LPK.DLL

[그림 13. 저장된 CSV 파일 내부 모습]

3.3.1 분석 서버 구성

Python 언어를 이용해 Django 프레임워크를 사용해서 구성하였다. 서버 배포는 AWS-EC2를 사용하여 진행했고, 본 프로젝트에서는 Django 프레임워크의 모든 기능이 아닌 특정한 부분만을 사용하여 진행이 되었다.

서버의 기본적인 동작으로는 웹에서 Url을 보내오면 해당 Url을 이용하여 AWS-S3에 접근하여 Binary 형태로 읽어오게 된다. Binary 형태의 앞부분 파일의 Signature를 비교하여 해당 하는 분석 프로그램으로 Binary 데이터를 넘겨서 분석을 진행 및 결과 도출을 하고 다시 JSON 형태로 웹에서 Return하게 된다.

```
@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")
```

[그림 1. Url 확인 및 AWS-S3 접근 및 분석 로직으로 데이터 전달]

해당 [그림1]에서 웹에서 보내온 Url을 미리 정의해 놓은 Url과 일치할 경우 GET요청으로 들어온 URL로 AWS-S3에 요청을 보내 해당 파일을 Bytes 단위로 읽는다.

```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_parse(data, filename)
        return result_lnk
    else:
        result_else = None
        return result_else
```

[그림 2. 미리 정의된 파일의 Signature와 비교 후 데이터를 분석 로직에 전달]

[그림2]는 미리 정의된 각 파일의 Signature와 Bytes를 비교해서 해당 파일일 경우 해당 분석 로직으로 전달해 주는 역할을 함

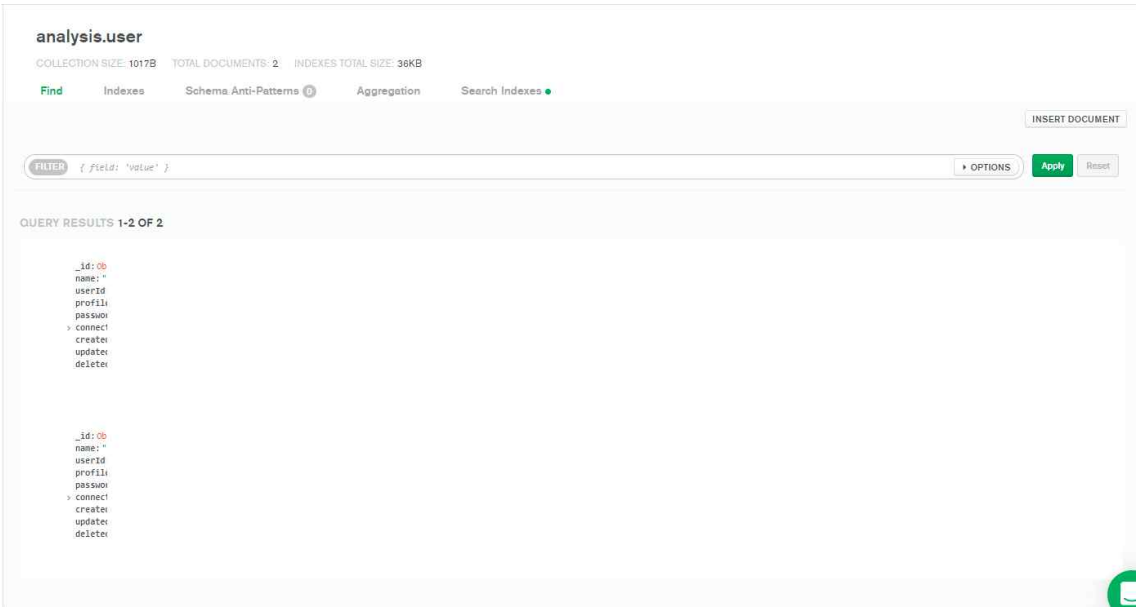
```python
win10_prefetch_sig = bytes(b'\x4D\x41\x4D') # prefetch 맞는지 ?

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_)
    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("\\")
```

[그림 3. Prefetch 분석 로직]

해당 [그림3]은 Prefetch에 대한 분석 로직이다. 각 윈도우 버전별로 다르게 되어 있으며, 분석을 완료한 뒤에는 JSON 형태로 다시 전달해주고 그것을 받은 [그림1]에서는 넘어온 JSON 데이터를 웹에게 Return 해주게 된다.

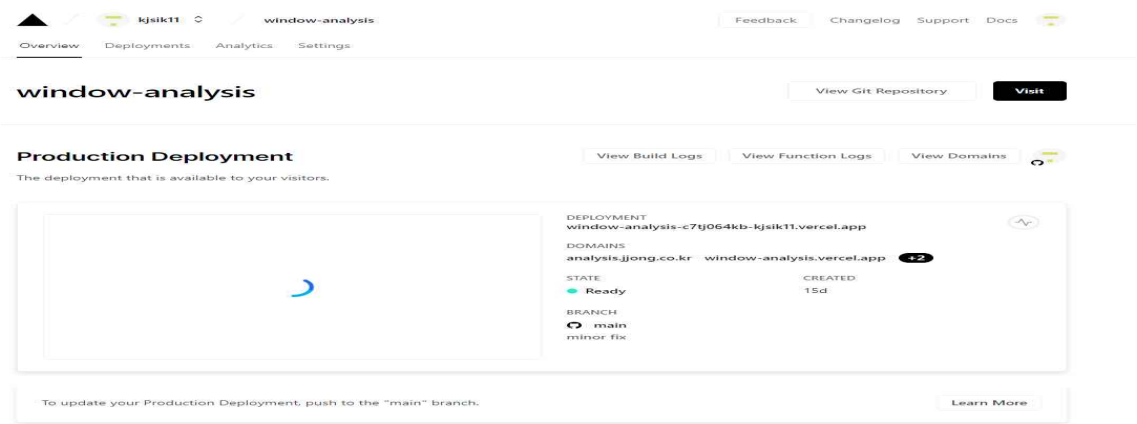
3.3.3 DB 구성 및 웹 호스팅

데이터베이스는 MongoDB를 사용하였으며 이는 Nosql기반의 데이터베이스이다. 해당 데이터베이스에는 현재 유저에 대한 정보만 insert하고 있으며 추후 파일에 대한 정보도 저장할 예정이다.



[그림 14. 데이터베이스 user collection 구조]

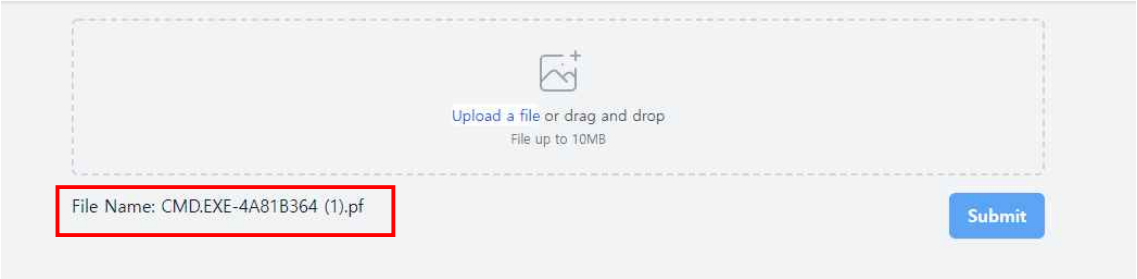
웹 호스팅은 Vercel을 사용하여 배포하고 있으며 Vercel은 별 다른 설정 없이도 서버리스 기능을 사용하여 웹을 손쉽게 배포해주는 역할을 한다.



[그림 15. 해당 프로젝트의 Vercel 메인페이지]

3.4 주요 기능 데모

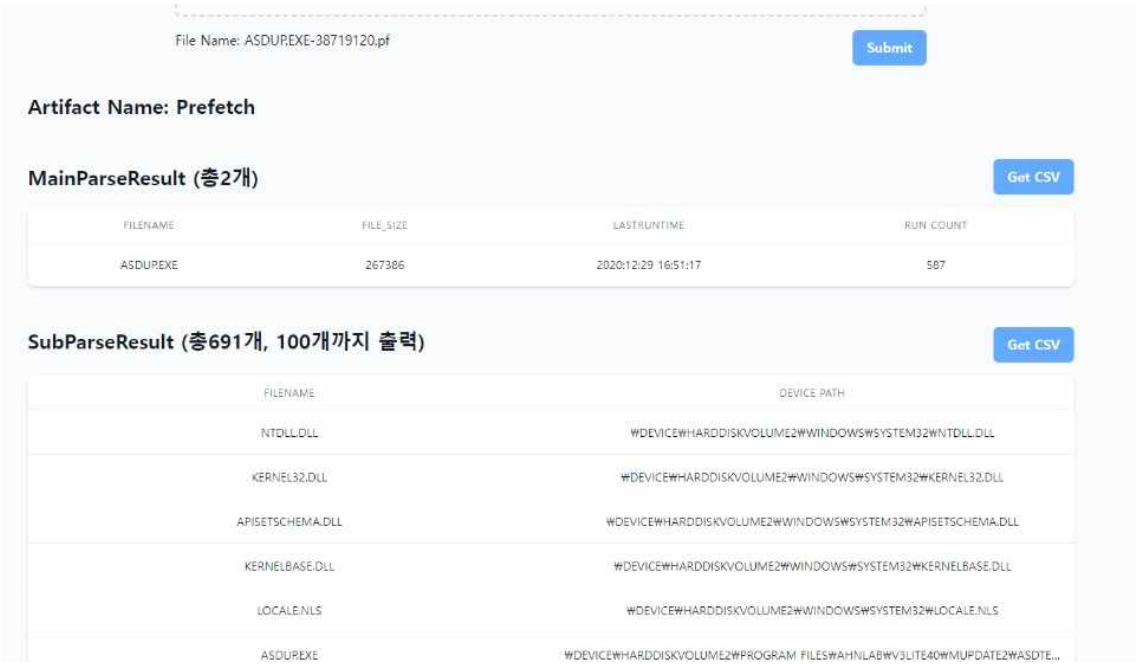
1) 웹 사이트에 파일 업로드



2) AWS-S3에 파일 업로드 및 서버로 AWS-S3에 저장된 파일 URL 전송

ASDUP.EXE-38719120.pf	pf	October 7, 2021, 14:08:08 (UTC+09:00)	261.1 KB	Standard
CMD.EXE-4A81B364 (1).pf	pf	October 21, 2021, 21:30:59 (UTC+09:00)	6.1 KB	Standard
CMD.EXE-4A81B364.pf	pf	October 21, 2021, 21:31:40 (UTC+09:00)	6.1 KB	Standard
HXD.EXE-0683C898.pf	pf	October 6, 2021, 23:52:28 (UTC+09:00)	29.3 KB	Standard

3) 서버에서 분석 및 웹에 분석된 결과 전달 및 웹 사이트에서 확인



4) 웹 사이트에서 파일 분석 결과 확인 및 Get CSV를 통해 CSV 파일 다운로드

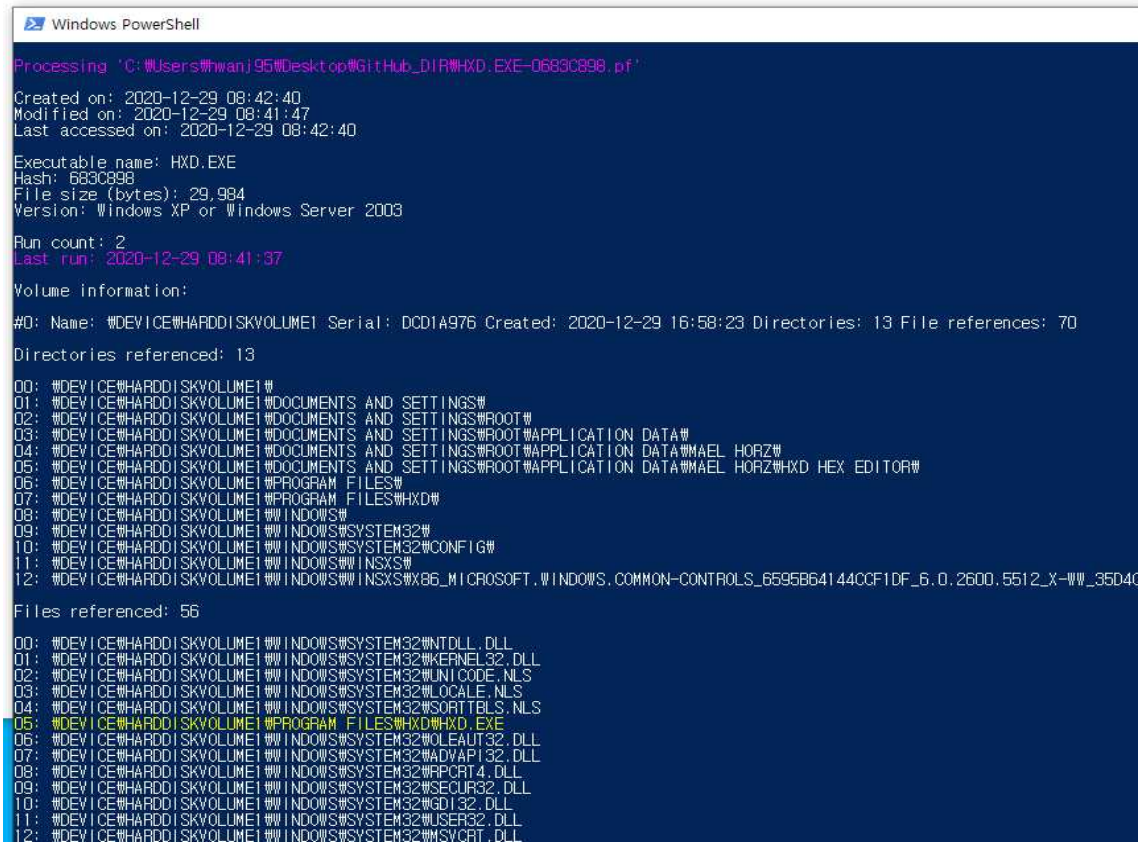
FileName	File_Size	LastRunTir	Run count
HXD.EXE	29984	1601:01:01	2

FileName	Device Path
NTDLL.DLL	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩NTDLL.DLL
KERNEL32	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩KERNEL32.DLL
UNICODE.	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩UNICODE.NLS
LOCALE.N	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩LOCALE.NLS
SORTTBLS	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩SORTTBLS.NLS
HXD.EXE	₩DEVICE₩HARDDISKVOLUME1₩PROGRAM FILES₩HXD₩HXD.EXE
OLEAUT32	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩OLEAUT32.DLL
ADVAPI32	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩ADVAPI32.DLL
RPCRT4.DI	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩RPCRT4.DLL
SECUR32.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩SECUR32.DLL
GDI32.DLL	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩GDI32.DLL
USER32.DI	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩USER32.DLL
MSVCRT.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩MSVCRT.DLL
OLE32.DLL	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩OLE32.DLL
VERSION.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩VERSION.DLL
SHELL32.D	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩SHELL32.DLL
SHLWAPI.I	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩SHLWAPI.DLL
COMCTL3	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩WINSXS₩X86_MICROSOFT.WINDOWS.COMMON-CONTROLS_6595B64144CCF1DF_6.0.2600.5512_X-WW_35D4CE83₩COMCTL32.DLL
WININET.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩WININET.DLL
CRYPT32.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩CRYPT32.DLL
MSASN1.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩MSASN1.DLL
COMDLG3	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩COMDLG32.DLL
WINSPOO	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩WINSPOOL.DRV
WINMM.D	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩WINMM.DLL
OLEACC.D	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩OLEACC.DLL
MSVCP60.	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩MSVCP60.DLL
IMM32.DL	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩IMM32.DLL
LPK.DLL	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩LPK.DLL

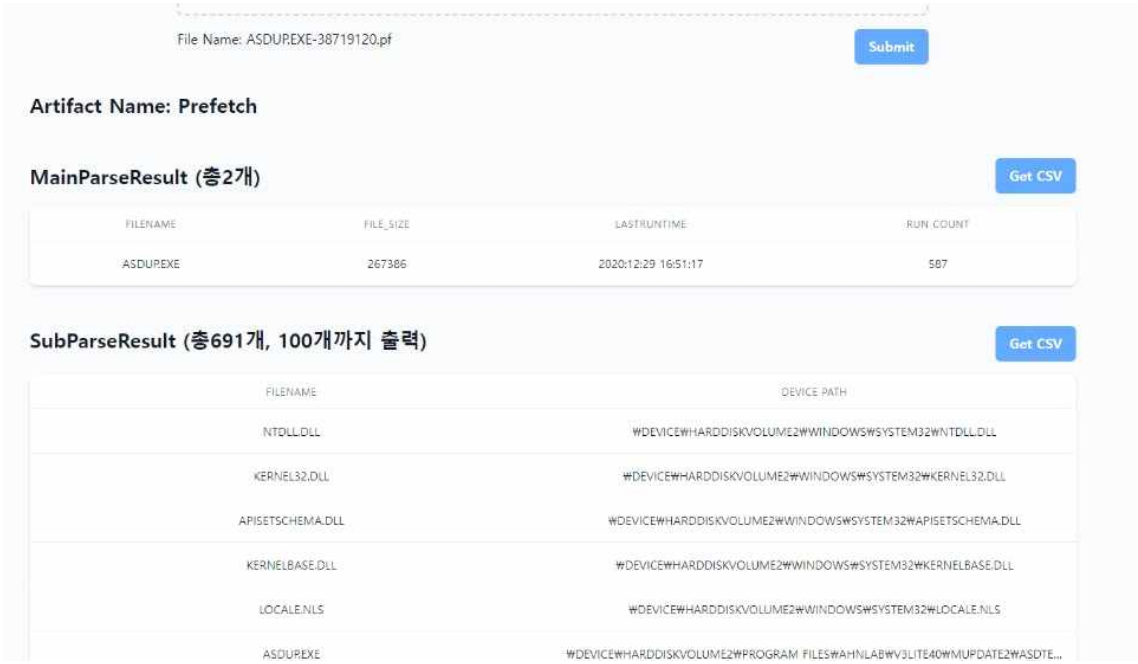
3.5 결과 분석

1) 기존 오픈소스 툴 PECmd(Prefetch)

해당 툴을 이용하면 CSV파일 혹은 PowerShell 창에서 결과를 볼 수 있습니다. 하지만 해당 툴의 단점이라면 아무래도 파일을 다운로드 받아서 CLI 환경으로 실행시키는 것이라고 볼 수 있습니다.



2) 저희가 만든 분석 툴(Prefetch)
웹에 업로드만 하게 되면 웹 상에서 바로 결과를 볼 수 있고 따로 툴을 다운로드 받아서 CLI 환경으로 실행 할 필요가 없습니다. 그로인해 편리성과 시간단축을 모두 챙길 수 있다고 생각합니다.



FileName	File_Size	LastRunTir	Run count
HXD.EXE	29984	1601:01:01	2

FileName	Device Path
NTDLL.DLL	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩NTDLL.DLL
KERNEL32	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩KERNEL32.DLL
UNICODE.	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩UNICODE.NLS
LOCALE.N	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩LOCALE.NLS
SORTTBLS	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩SORTTBLS.NLS
HXD.EXE	₩DEVICE₩HARDDISKVOLUME1₩PROGRAM FILES₩HXD₩HXD.EXE
OLEAUT32	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩OLEAUT32.DLL
ADVAPI32	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩ADVAPI32.DLL
RPCRT4.DI	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩RPCRT4.DLL
SECUR32.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩SECUR32.DLL
GDI32.DLL	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩GDI32.DLL
USER32.DI	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩USER32.DLL
MSVCRT.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩MSVCRT.DLL
OLE32.DLL	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩OLE32.DLL
VERSION.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩VERSION.DLL
SHELL32.D	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩SHELL32.DLL
SHLWAPI.I	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩SHLWAPI.DLL
COMCTL3	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩WINSXS₩X86_MICROSOFT.WINDOWS.COMMON-CONTROLS_6595B64144CCF1DF_6.0.2600.5512_X-WW_35D4CE83₩COMCTL32.DLL
WININET.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩WININET.DLL
CRYPT32.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩CRYPT32.DLL
MSASN1.[	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩MSASN1.DLL
COMDLG3	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩COMDLG32.DLL
WINSPOO	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩WINSPOOL.DRV
WINMM.D	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩WINMM.DLL
OLEACC.D	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩OLEACC.DLL
MSVCP60.	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩MSVCP60.DLL
IMM32.DL	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩IMM32.DLL
LPK.DLL	₩DEVICE₩HARDDISKVOLUME1₩WINDOWS₩SYSTEM32₩LPK.DLL

4. 결론

4.1 결론

- 프로젝트를 진행함으로써 웹 과 서버 분석 로직 결합에 성공 및 Windows Artifact에 대한 추구하던 목표인 편리성과 시간단축을 모두 챙길 수 있도록 제작, 또한 웹 사이

트에서 파일 구조 혹은 파일에 대한 설명을 확인할 수 있음
- 하지만 아직은 지원 가능한 Windows Artifact가 많지 않음
- 현재까지는 CLI 혹은 GUI 분석 툴만 존재하였는데, 처음으로 웹 사이트에서 분석을 시도했다. 하지만 용량이 큰 Artifact의 경우 웹에서 진행하기에는 어려움이 존재한다.

5. 참고자료

5.1 참고문헌

https://github.com/libyal/libscca/blob/main/documentation/Windows%20Prefetch%20File%20(PF)%20format.asciidoc (Prefetch Format)

http://forensic-proof.com/archives/584

https://docs.djangoproject.com/en/3.2/

https://www.django-rest-framework.org/

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-shllink/16cb4ca1-9339-4d0c-a68d-bf1d6cc0f943

https://www.sweetscape.com/010editor/ (Templates 활용 파일 구조 분석)

5.2 Open Source Tool

1) MFTECmd : $MFT 파일을 분석할 수 있는 오픈소스 툴
2) PECmd : Prefetch 파일을 분석할 수 있는 오픈소스 툴
3) LECmd : Lnk 파일을 분석할 수 있는 오픈소스 툴
4) JLECmd : JumpList 파일을 분석할 수 있는 오픈소스 툴

6. 별첨

6.1 소스코드

GitHub : https://github.com/hwanj-95/django

GitHub : https://github.com/kjsik11/window-analysis

6.2 발표자료

웹 기반 Windows Artifact 분석

중부대학교 정보보호학과
지도교수 : 이병천 교수님
분석해줄게조 지창환
김종식
염정현
최예지

목 차

조원 편성

이름	역할
지창환(조장)	서버 구축 및 Artifact 분석 및 개발
김종식	프론트엔드 구축 및 개발
염정현	Artifact 분석 및 개발
최예지	Artifact 분석 및 개발

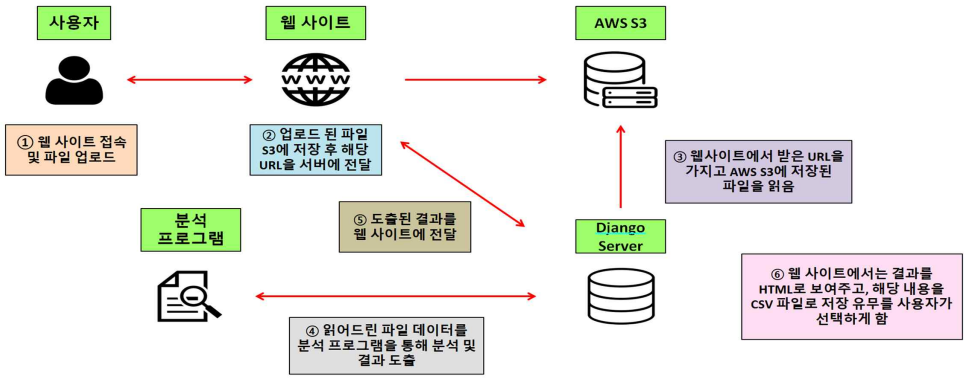
주제 선정 (1/2)

기존 오픈소스 분석 툴을
거의 다 CLI 환경으로
제공되며, 웹 사이트에서
분석해주는 사이트의 부재

주제 선정 (2/2)

시간 단축 및 편리성을 챙기기 위해 웹 사이트로 개발 진행

구 상 도



추진 경과

개발 환경 및 개발 내용 (1/10)

Web Server

Django
AWS-EC2
AWS-S3

Development Language

Python
Typescript

DB

MongoDB

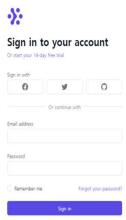
Web Front-end

Next.js
Node.js
Typescript
React
Tailwind CSS

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

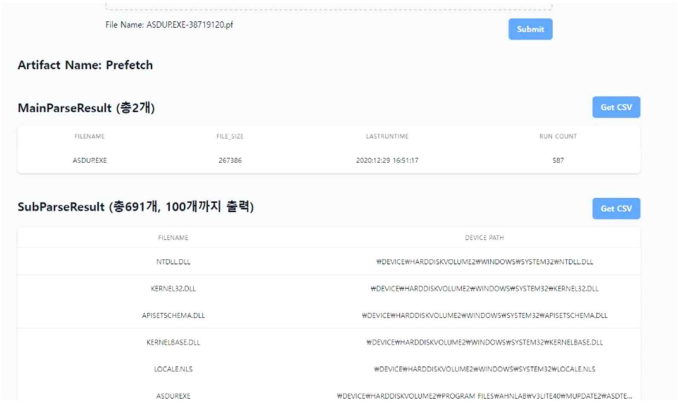
개발 환경 및 개발 내용 (6/10)

```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)
            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)

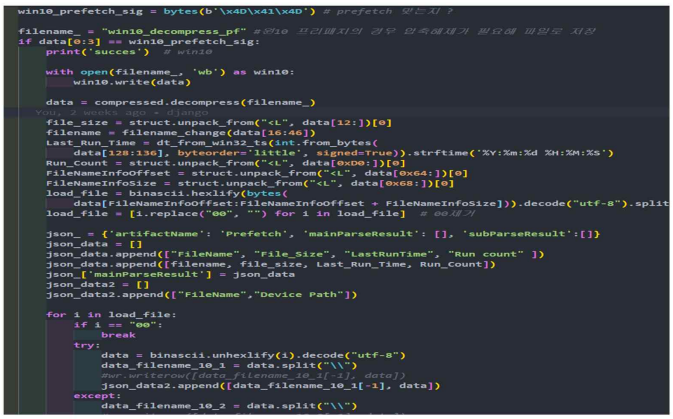
```python
def parse_(data, filename):
    prefetch_sig_1 = bytes(b'\x4D\x41\x4D')
    prefetch_sig_2 = bytes(b'\x53\x43\x43\x41')
    lnk_sig = bytes(b'\x4C\x00\x00\x00')

    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)



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

filename_ = "win10_decompress_pf" #win10 프리패치의 경우 압축해제가 필요해 매핑로 지정
if data[0:3] == win10_prefetch_sig:
    print('succes')  # win10
        win10.write(data)

    data = compressed.decompress(filename_)
    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("\\")
            #sr.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