오픈소스 기반 APT 공격 예방 Chrome extension 개발

[1] 김희은, [2*]손태식, [3] 김두원, [4] 한광석, [5] 성지훈

Development of an open source-based APT attack prevention Chrome extension

[1]Heeeun Kim, [2*]Taeshik Shon, [3]Duwon Kim and [4]Gwangseok Han, [5]JiHoon Seong

요 약

APT(advanced persistent threat) 공격이란 잠행적이고 지속적인 컴퓨터 해킹 프로세스들의 집합으로 특정 실체를 목표로 행해지는 공격이다[1]. 이러한 APT 공격은 대개 스팸 메일과 위장된 배너 광고 등 다양한 방식을 통해서 이뤄진다. 대부분 송장, 선적 서류(Shipment Document), 구매 주문서(P.O.-Purchase Order) 등으로 위장한 스팸 메일을 통해 유포되기 때문에 파일 이름도 동일하게 위와 같은 이름이 사용된다. 그리고 이러한 정보탈취형(Infostealer) 공격이 가장 2021년 2월 첫째 주 가장 많이 발견된 악성 코드였다[2]. Content Disarm & Reconstruction(이하 CDR)은 백신, 샌드박스에서 막아내지 못한 보안 위협에 대하여 파일 내 잠재적 보안 위협 요소를 원천 제거 후 안전한 파일로 재조합하여 악성코드 감염 위험을 사전에 방지할 수 있는 '콘텐츠 무해화 & 재조합' 기술이다. 글로벌 IT 자문기관 '가트너(Gartner)'에서는 첨부파일 형태의 공격에 대한 솔루션으로 CDR을 추천하고 있다. Open source로 공개된 CDR 기법을 사용하는 프로그램으로 'Dangerzone'이 있다. 해당 프로그램은 대부분의 문서 파일의 확장자를 지원하지만, 한국에서 많이 사용되는 HWP 파일의 확장자를 지원하지 않고 있다. 그리고 Gmail은 악성 URL을 1차적으로 차단해주지만 Naver, Daum 등의 메일 시스템에서는 악성 URL을 차단하지 않아 손쉽게 악성 URL을 유포할 수 있다. 이러한 문제점에서 착안하여 APT 공격을 예방하기 위한 HWP 확장자를 지원하는 'Dangerzone' 프로그램, Naver, Daum 메일 내 URL 검사, 배너형 광고 차단의 기능을 수행하는 Chrome extension을 개발하는 프로젝트를 진행했다.

Abstract

Advanced persistent threat (APT) attacks are attacks aimed at a particular entity as a set of latent and persistent computer hacking processes. These APT attacks are usually carried out through various methods, including spam mail and disguised banner advertising. The same name is also used for files, since most of them are distributed via spam mail disguised as invoices, shipment documents, and purchase orders. In addition, such Infostealer attacks were the most frequently discovered malicious code in the first week of February 2021. CDR is a 'Content Disarm & Reconstruction' technology that can prevent the risk of malware infection by removing potential security threats from files and recombining them into safe files. Gartner, a global IT advisory organization, recommends CDR as a solution to attacks in the form of attachments. There is a program using CDR techniques released as open source is called 'Dangerzone'. The program supports the extension of most document files, but does not support the extension of HWP files that are widely used in Korea. In addition, Gmail blocks malicious URLs first, but it does not block malicious URLs in mail systems such as Naver and Daum, so malicious URLs can be easily distributed. Based on this problem, we developed a 'Dangerzone' program that supports the HWP extension to prevent APT attacks, and a Chrome extension that performs URL checking in Naver and Daum mail and blocking banner ads.

Keywords: *APT, Dangerzone, malware, Ad block, Virus total, open source, Chrome extension*

[1] *아주대학교, 사이버보안학과 학사과정(gmldms324@ajou.ac.kr)*
[2*] *교신저자 아주대학교, 사이버보안학과, 교수 (tsshon@ajou.ac.kr)*
[3] *아주대학교, 사이버보안학과 학사과정 (r01596@ajou.ac.kr)*
[4] *아주대학교, 사이버보안학과 학사과정 (hanson119@ajou.ac.kr)*
[5] *아주대학교, 사이버보안학과 학사과정 (battleb123@ajou.ac.kr)*

Received: June 13, 2021, Revised: Aug. 05, 2021, Accepted: Aug. 31, 2021

I. 서론

현재 한국뿐만 아니라 세계적으로 스팸메일을 사용한 APT 공격이 점점 많아지고 있다. 스팸메일에는 악성 링크 또는 악성 파일 등 사용자의 개인 정보를 탈취하기 위한 공격이 존재한다. 이를 예방하기 위한 프로그램들이 존재하지만 사용자들은 불편함, 사용료 등에서 발생하는 비용 때문에 APT 공격을 예방하는 프로그램 사용에 개인적으로 큰 관심을 기울이지 않고 있다.

본 논문에서는 APT 공격으로 사용될 수 있는 악성 URL 과 악성 파일을 예방하는 Chrome extension 을 개발한다. 해당 Chrome extension 은 크게 2 가지 기능을 가지고 있는데, 첫 번째 기능으로는 CDR 기법을 활용한 악성 파일 변환이고 두 번째 기능은 메일 내에 존재하는 링크의 URL 을 검사하는 것이다 ‘Dangerzone’ open source 를 활용해 악성 파일을 안전한 PDF 파일로 변환하고 VirusTotal API[3]를 활용해 Naver, Daum 메일을 통해 유포되는 URL 을 검사하고 알림을 제공한다[3] [4].

Chrome extension 으로 무료로 위와 같은 기능을 가진 프로그램을 제공함으로써 APT 공격을 예방하는 데 사용자가 불편함을 느끼지 않게 하고, 더 나아가 ‘Dangerzone’ 프로젝트에 기여함으로써 사이버 보안에 이바지할 수 있도록 하는 것이 목표이다.

논문의 구성은 다음과 같다. 2 장에서 기술 동향 및 개발 환경에 대해 서술한다. 3 장에서는 프로그램 구조를 분석한다. 4 장에서는 알고리즘을 기술하며 5 장에서는 구현 결과를 보여준다. 끝으로 6 장에서 결론으로 마무리한다.

II. 기술 동향 및 개발 환경

2.1 관련 제품

지능화된 악성 위협이 증가함에 따라 샌드박스 회피 및 우회하는 기술이 증가하고 있는 세로 샌드박스만으로는 지능화된 위협에 대응하는데 한계를 나타내고 있다. 이에 가트너(Gartner)가 발표한 회피성이 높은 공격으로부터 보호하기 위한 5 가지 핵심 보안 패턴 중 하나로 CDR 기법이 있다. APT 공격을 방지하기 위해 많은 백신 프로그램이 존재하지만 CDR 기능을 지원하고 메일을 통해 유포되는 URL 을 검사하는 기능을 지원하는 경우는 드물다. 현재, 실제로 운영되고 있는 CDR 기법을 지원하는 프로그램과 URL 검사를 지원하는 프로그램에 대한 분석은 다음과 같다.

2.1.1 지란지교 시큐리티: SaniTox

국내에서 CDR 기법을 지원하는 대표적인 프로그램은 지란지교 시큐리티의 SaniTOX 이다[5]. 지란지교 시큐리티의 SaniTOX 는 파일 내 실행 가능한 액티브 콘텐츠를 원천 제거 후 안전한 파일로 재조합해 알려지지 않은 보안 위협에 대응할 수 있는 콘텐츠 악성코드 무해화 솔루션이다. Web 형식으로 이용이 편리하고 HWP, MS Office, PNG 등 많은 확장자를 지원한다. 그렇지만, linux 에서 사용되는 odt 등의 확장자를 지원하지 않고 모든 기능을 사용하기 위해선 요금을 지불해야 한다

2.1.2 Dangerzone

현재 github 에 공개되어 있는 CDR 기법을 적용한 프로그램으로 ‘Dangerzone’이 있다. 해당 프로그램은 pixel 렌더링을 통해 안전한 flat PDF 파일로 바꾸는 것으로 무료로 사용이 가능하다[6]. 하지만 사용하기 위해서는 docker 와 ‘Dangerzone’을 local machine 에 설치해야 되고 한국에서 많이 사용되는 HWP 확장자를 지원하지 않는다

2.1.3 McAfee WebAdvisor

McAfee WebAdvisor 는 피싱 사이트에 들어가거나 악성 파일 링크를 잘못 클릭했을 때 block 한다[7]. 또한, 파일을 내려받을 때 악성코드 유무 검사를 진행해 준다. 그리고 Chrome

extension 으로 사용해 편리함도 제공해 준다. 그렇지만 메일 내 URL 을 검사해 주는 기능이 없어 Naver, Daum 메일 시스템을 통해 악성 URL 유포하는 것이 비교적 수월해질 수 있다

2.2 차별성

CDR 기법을 사용하는 'Dangerzone' 은 Docker 와 'Dangerzone' 다운로드가 필요해 사용에 불편함이 있고, 한국에서 많이 사용되는 HWP 확장자를 지원하지 않으며 해당 파일이 악성코드를 가졌는지 알려주지 않는다.

SaniTox 는 web, cloud 형태로 제공되어 편리하지만 유료라는 점에서 결제해야 하는 번거로움이 있고 linux 환경에서 사용되는 Open Document 확장자(odt, ods, odp, odg) 를 지원하지 않는다.

McAfee WebAdvisor 는 Chrome extension 이고 무료로 편리하게 사용이 가능하나 웹 브라우저를 통해 연결된 URL 의 악성 유무를 판단해 줄 뿐 mail 을 통해 전달되는 URL 을 검사하지는 않는다. Gmail 은 자체적으로 피싱 사이트를 차단하지만 Naver, Daum 에서는 차단을 하지 않는다.

이에 본 프로그램에서는 앞의 단점들을 보완하여 HWP 기능을 지원하고 무료로 편리하게 CDR 기법을 사용할 수 있는 것과 메일 내 URL 을 검사할 수 있는 것에 차별성을 두었다. 분석한 기존 제품과의 차별성에 대한 정리는 다음과 같다[표 1].

Table 1. Differentiation from the existing products
표 1. 기존 제품과의 차별성

product	HWP	Open Document Format	Secure file conversion	Malicious code	Convenience	Mail URL scan
Dangerzone	X	O	O	X	X	
niTox	O	X	O	O	X	
Mc Afee Web advisor				O	O	X
Our Project	O	O	O	O	O	O

2.3 개발 환경

본 프로그램에선 프로그램을 개발하기 위해 파이썬으로 Dangerzone 에 한글 확장자를 추가하였고 URL 분석을 진행했다. Chrome extension 으로 동작하는 Front-end 는 HTML 과 JavaScript 를 사용해 개발되었고, Back-end 서버로 장고를 사용했다. 광고 차단과 사이트 위험도 검사에는 Javascript 를 사용했다. 문서 파일을 안전한 파일로 바꾸는 서버로 Ubuntu server 18.04 버전을 사용하였으며, 서버를 계속 동작시키기 위해 Naver Cloud platform 을 사용한다. 개발 환경에 대한 정리는 다음과 같다[표 2].

Table 2. Development environment
표 2. 개발 환경

Python	Add a hwp extension to the open source, Dangerzone, using Python.
	Use Python to detect malicious files and URL risk using Virus Total API.
Ubuntu Server 18.04.5 LTS	When a user uploads a file, Ubuntu Server is used as a server that converts the file into a secure PDF
Chrome Extension	Use Javascript and HTML to develop Chrome Extension
Naver Cloud platform	Use the Naver Cloud platform to maintain the server
Django	Use it as a server back-end framework

III. 시스템 구조

3.1 구성 요소 별 세부 구조와 기능

3.1.1 전체 구조

본 프로젝트의 구조는 Chrome extension, 클라우드 서버로 구성되어 있다. Chrome extension 으로 동작하는 Front-end 는 HTML 과 JavaScript 를 사용해 개발되었고, Back-end 서버로 장고를 사용했다.

Chrome extension 은 기본적으로 인터넷 활동 중 사용자에게 노출될 수 있는 배너형 광고를 차단하고 서버로 의심 URL 과 파일을 전송한다. 차단할 배너형 광고 Domain 은 Chrome extension API 와 EasyList 를 통해 수집되며 주기적으로 업데이트된다.

서버를 계속 동작시키기 위해 Naver Cloud platform 을 사용한다. 서버의 OS 는 Ubuntu server 18.04 LTS 이고 스펙은 vCPU 1 개, RAM 1GB, DISK 50GB 이다. 서버는 의심 URL, 파일을 검사해 사용자에게 그 결과를 알려주고 독립된 환경에서 의심 파일을 안전한 파일로 변경해 사용자에게 전송한다. 의심 파일을 안전한 파일로 변환하고, 의심 파일 또는 URL 을 검사해 그 결과를 보고서로 반환하기 위해 HWP 확장자가 추가된 Dangerzone 과 VirusTotal API 을 사용한다.

서버는 사용자에게 파일 또는 URL 검사 결과를 전달해 줘야 한다. 사용자가 의심 URL 을 입력했을 경우 서버는 VirusTotal API 를 사용해 도출된 결과와 작성된 알고리즘을 바탕으로 사용자에게 URL 위험도를 알려준다. 그리고 의심 파일인 경우 VirusTotal API 를 통해 나온 결과와 안전한 파일을 사용자에게 전송해 준다. 사용자에게 전달된 보고서 및 파일은 Chrome 에서 새로운 탭을 통해 시각화된다[그림 1].

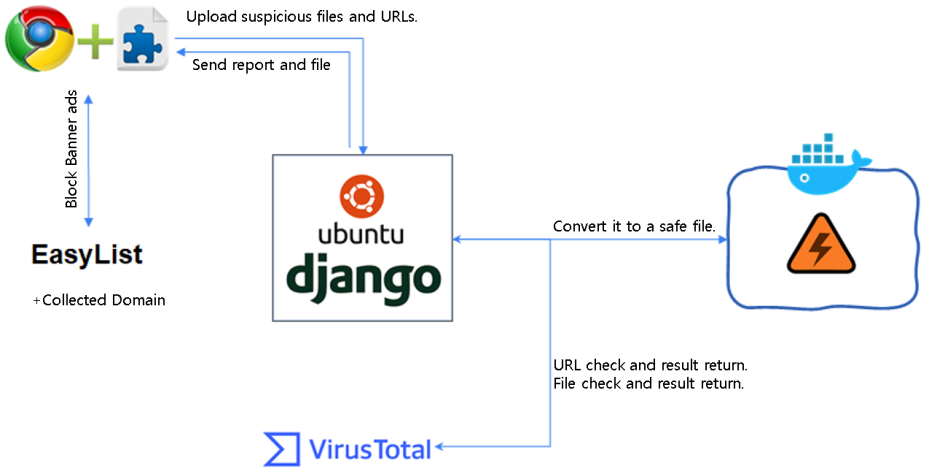

Figure 1. Structure
그림 1. 전체 구조

3.1.2 Dangerzone

파일 내 실행 가능한 Active Content 를 제거하거나 비활성화하기 위해 오픈소스인 'Dangerzone'을 활용한다. 'Dangerzone'에서 지원하는 파일 확장자 'PDF, DOCX, DOC, DOM, XLSX, XLS, PPTX, PPT, ODT, OPS, ODP, ODG, JPG, JPEG, GIF, PNG, TIF, TIFF'에 HWP 확장자를 추가해 CDR 기능을 구현했다. 먼저 파일이 서버로 전송되면 모두 PDF 파일로 변환한다.

'DOCX, DOC, DOM, XLSX, XLS, PPTX, PPT, ODT, OPS, ODP, ODG' 파일은 libreoffice 툴을 통해 PDF 파일로 변환되고, 'JPG, JPEG, GIF, PNG, TIF, TIFF' 파일은 GraphicsMagick 툴을 통해 PDF 파일로 변환된다. 마지막으로 HWP 파일일 경우 pyhwp 모듈과 wkhtmltopdf 모듈을 통해

PDF 파일로 변환된다[8][9]. 변환된 PDF 파일은 페이지 별로 분할한 뒤 pdftocairo 툴을 통해 PNG 파일로 변경하고, gm 툴을 통해 PNG 파일에서 RGB pixel 데이터를 추출한다.

추출된 RGB pixel 데이터는 다시 gm 툴을 통해 PDF 파일로 변환되고 pdfunite 툴을 사용해 분할된 PDF 파일들을 하나의 PDF 파일로 만든다. 만약 사용자가 flat PDF 가 아니라 searchable PDF 파일을 원한다면 OCR 기능을 수행하는 tesseract 툴을 사용한다.

3.1.3 URL Scan

URL Scan 기능은 기본적으로 네이버, 다음 등의 메일에 포함된 악성 URL 을 탐지하기 위해서 만들었지만 메일 이외에도 URL Scan 이 가능하다. 사용자가 URL Scan 을 시도하면 Chrome Extension 은 해당 URL 을 받아 서버로 전달하고, 서버에서 해당 URL 을 VirusTotal API 를 사용하여 분석한다. 분석 결과를 다시 Chrome Extension 에 전달받아 위험도 별로 사이트를 4 가지 타입으로 분류한다.

타입은 안전/ 주의/ 위험/ 매우 위험의 4 가지 타입이 존재하며 아무런 위협이 존재하지 않을 시 안전, 위험한 악성코드는 포함되지 않았지만 광고 등의 사용자에게 불편함을 제공하는 요소가 포함된 경우에는 주의, 적지만 사용자에게 위협이 되는 Malware 가 하나라도 포함되어 있는 경우 위험, 사용자가에게 큰 위험을 줄 수 있는 URL 의 경우 매우 위험으로 분류하였다.

3.1.4 광고 차단

사이트 접속 시 Chrome Extension API 를 활용해 광고 배너를 차단한다[10]. 기본적으로 웹 Request 를 기준으로 해당 URL 이 EasyList 의 목록에 존재하거나 광고와 관련된 특정 키워드를 포함할 시 해당 배너는 사용자에게 보이지 않도록 차단한다.

Chrome Extension API 중 webRequest 는 사용자가 사이트에 접속하면 전달되는 웹 요청을 기준으로 해당 페이지가 실행되기 이전에 Chrome Extension 이 미리 URL 을 판별한다. EasyList 에서 제공하는 사이트를 저장해 둔 blocked_sites 에 존재하는 URL 이라면 해당 배너를 차단한다.

3.2 DFD

전체적인 Data Flow 로 사용자는 chrome 을 사용하면서 기본적인 AdBlock 기능을 사용 가능하고 의심 url 을 검사할 수 있다. 또한 의심가는 파일이 있다면 서버로 업로드하여 안전한 파일로 변환이 가능하다.

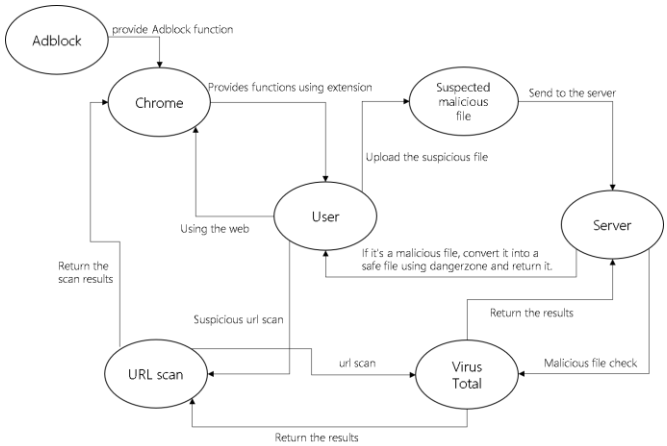

Figure 2. Data Flow Diagram
그림 2. 데이터 흐름도

IV. 알고리즘

4.1 광고 차단

광고 사이트 및 배너 차단의 경우 Chrome 에서 제공하는 WebRequest API 를 사용해 구현하였다. blocked_sites.js 에는 EasyList 에서 제공하는 광고 사이트의 URL 및 광고 배너에 포함된 키워드가 약 80000 개가 넘게 저장되어 있다. 해당 스크립트에 포함된 URL 의 경우 Chrome 의 백그라운드에서 해당 URL 에 접속하지 못하도록 redirect 한다. [그림 3]

```
chrome.webRequest.onBeforeRequest.addListener(
    function(details) {
        if(!enabled){
            return { cancel: false };
        }
        return {redirectUrl: chrome.extension.getURL("page_easy.html") };
    },
    {urls:  [blocked_sites]},
    ["blocking"]
)
function r(tableId){
    chrome.tabs.update(tabId,{
        "url": redirectUrl
    });
}
chrome.extension.onRequest.addListener(function (request, sender, sendResponse) {

    if (request.redirect) {
            chrome.windows.getCurrent(function(w){
                chrome.tabs.query({windowId : w.id}, function(t){
                    r(t.id);
                });
            });
    }
    sendResponse({
        redirected: redirectUrl
    });
})
```

Figure 3. background.js
그림 3. background.js

4.2 URL 안전도 검사

사용자가 Chrome 에서 URL 을 우클릭할 시, Chrome 의 백그라운드에서 해당 URL 을 클라이언트에서 서버로 전송한다. 전송된 URL 은 VirusTotal API 로 검사하고 그 결과는 json 형태로 제공된다. 탐지요소 별 위험도는 안전 / 주의 / 위험 / 매우 위험으로 4 가지 타입이 존재하며 분류 기준은 아래와 같다. [그림 4]

```
안전 : positive_Json 0개, unrated site = 0;
주의 : positive_Json 0개, unrated site = 1;
위험 : positive_Json 1~3개 and Type : malware 2개 미만,phishing 미포함, malicious만 포함
매우위험 : positive_Json 4이상 or malware 3개이상 or phishing 포함
```

Figure 4. Risk classification by detection element
그림 4. 탐지요소 별 위험도 분류 기준

Chrome 에서 URL 마우스 우클릭을 통해 URL 전달 이벤트를 실행하면 Chrome 의 백그라운드에서 해당 URL 을 클라이언트에서 서버로 전송한다. 그림 5 는 해당 이벤트를 처리하는 코드이다.

```
chrome.contextMenus.onClicked.addListener(function (info, tab) {
    url_link = info.linkUrl;
    formData = new FormData()
            formData.append('testurl', url_link)
            fetch("http://49.50.166.66:8000/url/", {method: 'POST', body: formData})
            .then(resp => resp.text())
            .then(data => {
                chrome.tabs.create({url: "http://49.50.166.66:8000/url/" + data})
            })
});
```

Figure 5. url_upload.js
그림 5. url_upload.js

서버에서 동작하는 URL 검사 python script 는 전달받은 url 을 문자열로 저장한 뒤, Virustotal API 를 사용하여 Scan 하고 위험요소가 탐지될 경우 detect 는 true 를 return 한다. Detect 는 false 이지만 url 에 인증되지 않은 요소가 포함되어 있는 경우 unrated site 가 return 된다. 그리고 detect 를 true 로 탐지한 백신 엔진과 탐지된 요소를 각각 detectTrueProg, detectTrueType 에 List 형태로 저장하여 사용자에게 제공한다. [그림 6]

```
params = {'apikey': apikey, 'resource':line}
response = requests.post('https://www.virustotal.com/vtapi/v2/url/report', params=params, headers=headers)
json_response = response.json()
positives_Json = json_response.get("positives")
response_Json = json_response.get("response_code")
scanr = json_response.get("scans")
print(json_response)
for i in range(0,88):
    programs.append(list(scanr.keys())[i])
```

```
if (positives_Json == 0 and unrated == 0):
    return render(request, "page1.html", context= {'url': url_real, 'detectTrueProg': detectTrueProg, 'detectTrueType' : detectTrueType, 'unrated': unrated})
elif (positives_Json == 0 and unrated == 1):
    return render(request, "page2.html", context= {'url': url_real, 'detectTrueProg': detectTrueProg, 'detectTrueType' : detectTrueType, 'unrated': unrated})
elif (positives_Json <=3 and malwcount <= 1 and phicount == 0):
    return render(request, "page3.html", context= {'url': url_real, 'detectTrueProg': detectTrueProg, 'detectTrueType' : detectTrueType, 'unrated': unrated})
else:
    return render(request, "page4.html", context= {'url': url_real, 'detectTrueProg' : detectTrueProg, 'detectTrueType' : detectTrueType, 'data': data, 'unrated': unrated}
```

Figure 6. URL scan script
그림 6. URL scan script

4.3 파일 검사 및 변환

사용자는 Chrome extension 을 통해 파일을 서버로 업로드 한다. 서버는 사용자별로 전달받은 파일을 변환한 결과와 검사한 결과를 웹 형식으로 반환해야 하는데, 이는 사용자 id 와 파일의 hash 값을 통해 이뤄진다. 그림 7 은 파일을 서버로 업로드하고 서버로부터 전달받은 결과를 사용자에게 보여주는 코드이다.

서버는 사용자로부터 요청을 받았을 때 URL 인지 파일인지 판별한 후, 파일인 경우 먼저 VirusTotal API 를 통해 파일 검사를 실행하고 HWP 파일이 추가된 docker image 로 docker container 를 생성해 안전한 파일로 변환한다.

```
document.addEventListener('DOMContentLoaded', function() {
    document.getElementById('import-button').onclick = function() {
        var fileChooser = document.createElement('input');
        fileChooser.type = 'file';
        fileChooser.accept = '.hwp,.pdf,.docx,.doc,.xlsx,.xls,.pptx,.ppt,.odt,.ods,.odp,.odg,.jpg,.jpeg,.gif,.png,.tif,.tiff';

        fileChooser.addEventListener('change', function() {
            console.log("file change");
            var file = fileChooser.files[0];

            formData = new FormData()
            formData.append('file', file)
            fetch("http://49.50.166.66:8000/upload/", { method: 'POST', body: formData })
                .then(resp => resp.text())
                .then(data => {
                    var split = data.split("\n")
                    var id = split[1]
                    var hash = split[0]
                    chrome.tabs.create({ url: "http://49.50.166.66:8000/upload/" + hash + "/" + id })
                })
                .catch(error => alert(error));

            // <-- Resets the input so we do get a `change` event,
            //     even if the user chooses the same file
        });
```

Figure 7. file_upload.js
그림 7. file_upload.js

```
def file_upload(orgfile, timeout=None, proxies=None):
    if not os.path.isfile(orgfile):
        raise Exception('File not found.')
    base_url = 'https://www.virustotal.com/api/v3/files'
    file_size = os.path.getsize(orgfile)
    headers = {
        'x-apikey': API_KEY,
    }
    # 32 MB 기준
    if file_size >= 33554432:
        with open(orgfile, 'rb') as f:
            data = {'file': f.read()}
            try:
                response = requests.get(base_url + '/upload_url',
                                        headers=headers,
                                        proxies=proxies,
                                        timeout=timeout)

                if response.status_code != 200:
                    raise Exception(response)

                upload_url = response.json()['data']
                response = requests.post(upload_url,
                                         headers=headers,
                                         files=data,
                                         proxies=proxies,
                                         timeout=timeout)
                if response.status_code != 200:
                    raise Exception(response)

                return response.json()['data']['id']

            except Exception as e:
                print(e)
                exit(1)
    else:
        with open(orgfile, 'rb') as f:
            data = {'file': f.read()}
            try:
                response = requests.post(base_url,
                                         headers=headers,
                                         files=data,
                                         proxies=proxies,
                                         timeout=timeout)

                if response.status_code != 200:
                    raise Exception(response)

                return response.json()['data']['id']

            except Exception as e:
                print(e)
                exit(1)
```

Figure 8. file upload to VirusTotal Server
그림 8. file upload to VirusTotal Server

그림 8 은 VirusTotal API 를 활용해 파일을 검사하는 코드이다. VirusTotal API 는 버전 2 와 3 이 존재하는데 둘의 차이는 정보의 양과 시간이다. 본 프로젝트에서는 좀 더 빠른 시간에 더욱 많은 정보를 얻어와 사용자에게 알려주기 위해 버전 3 을 사용하였다. VirusTotal API 버전 3 은 파일의 용량 32MB 를 기준으로 32MB 보다 크면 GET 방식을 통해 파일을 전달하고 32MB 보다 작으면 POST 방식을 통해 파일을 전달한다.

```python
def file_report(file_id):
    headers = {
        'x-apikey': API_KEY,
        'Accept': 'application/json',
    }
    response = requests.get(
        'https://www.virustotal.com/api/v3/analyses/{}'.format(file_id), headers=headers)
    return response.json()
```

Figure 9. receive scan result as json
그림 9. receive scan result as json

파일을 서버에 전달하면 해당 파일에 대한 id 를 전달받아, id 를 조회함으로써 파일의 검사 결과를 얻을 수 있다. 얻은 결과인 json 파일은 업로드 된 문서형 파일의 sha-256 hash 값을 이름으로 가지는 폴더 안에 저장되어 안전한 파일과 함께 사용자에게 전달된다. 그림 9 는 전달받은 파일 id 로 검사 결과를 json 형식으로 받는 코드이다.

```python
def vtchart(request, hash, pk):
    print('-'*10 + 'VirusTotal Malicious Report' + '-'*10)
    print(pk)
    file_path = BASE_DIR + "/media/" + hash + "/report.json"
    with open(file_path, "r") as f:
        file_json = json.load(f)
    file_data = file_json['data']['attributes']
    status = file_json['data']['attributes']['status']
    if file_data['stats']['malicious'] > 0 or file_data['stats']['suspicious'] > 0:
        flag = True
    else:
        flag = False

    # 0: malicious, 1: suspicious, 2: undetected, 3: type-unsupported, 4: etc
    re_data = [[] for _ in range(5)]
    for e in file_data['results']:
        base = file_data['results'][e]
        if base['category'] == 'malicious':
            re_data[0].append((base['engine_name'], base['result']))
        elif base['category'] == 'suspicious':
            re_data[1].append((base['engine_name'], base['result']))
        elif base['category'] == 'undetected':
            re_data[2].append((base['engine_name'], 'undetected'))
        elif base['category'] == 'type-unsupported':
            re_data[3].append((base['engine_name'], 'type-unsupported'))
        else:
            re_data[4].append((base['engine_name'], base['category']))

    return render(request, 'report.html', context={'check': status, 'data': re_data, 'flag': flag, 'hash': hash})
```

Figure 10. Preprocessing result of file scan
그림 10. Preprocessing result of file scan

그림 10 은 결과로 전달된 json 파일을 전처리한 뒤, report.html 에 rendering 하여 클라이언트에게 검사 결과를 보여주는 코드이다. 검사 결과는 ‘malicious’, ‘suspicious’, ‘undetected’, 그 외 순으로 list 에 정렬되며, 클라이언트에게 보여진다.

```
<button type="button" class="btn btn-dark" onclick="location.href='/download/{{hash}}'">안전한 파일 다운로드</button>
    <br>
    <br>
<table class="table table-striped table-dark" width="100%">
    <thead>
        <tr>
            <th scope="col">Name</th>
            <th scope="col">Detection</th>
        </tr>
    </thead>
    <tbody>
        {% for d in data%} {% for name,detection in d%}
        <tr>
            <td>{{ name }}</td>
            <td>{{ detection}}</td>
        </tr>
        {% endfor %} {% endfor %}
    </tbody>
</table>
```

Figure 11. Key point of 'report.html'
그림 11. report.html 핵심

그림 11 은 report.html 에서 다운로드 버튼과 백신 엔진과 탐지 결과 table 에 해당하는 코드이다. 버튼 클릭 시 hash 값을 참조하여 변환된 파일을 다운로드 받고, data 에 저장된 백신 엔진과 탐지 결과를 table 로 만든다.

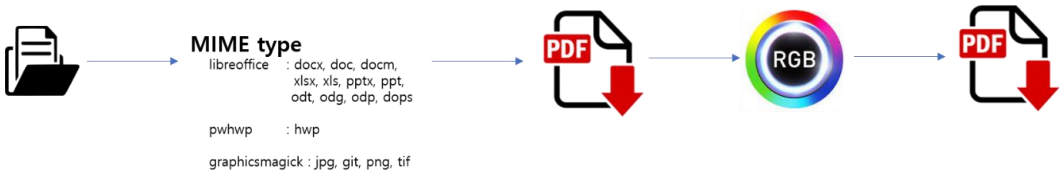

Figure 12. Docker image algorithm
그림 12. Docker 이미지 알고리즘

개발된 docker 이미지 알고리즘은 그림 12 와 같다. magic 모듈을 이용해 파일의 mime type 을 파악하고 mime type 에 따라 libreofiice, pwhwp, graphicmagick 모듈을 사용해 PDF 파일로 변경한다. 변경된 PDF 파일을 페이지 별로 분리한 다음, pdftocario 모듈을 이용해 PNG 파일로 변경한다. PNG 파일로부터 graphicmagick 모듈을 사용해 각 페이지 별 RGB 데이터를 뽑아내고, 다시 graphicmagic 모듈을 사용해 PDF 파일로 변환한다. 바로 이 과정에서 악성코드 무해화가 일어난다고 할 수 있다. 마지막으로 각 페이지 별 안전한 PDF 로 변환된 파일들을 하나의 PDF 통합하여 안전한 파일로 변환하는 과정을 마무리한다.

만약 searchable PDF 를 원한다면 tesseract 모듈을 통해 OCR 과정을 거쳐 이미지로만 이뤄진 PDF 가 아니라 텍스트로 검색 가능한 PDF 파일을 얻을 수 있다.

Dangerzone 에서 파일을 변환을 수행할 때 제일 먼저 magic 모듈을 이용해 파일의 mime type 을 파악한다. 지원하는 확장자는 그림 4 와 같으며 각 확장자 별 mime type 을 확인하여 분기점을 통해 파일 변환이 일어난다. HWP 파일의 mime type 은 'application/x-hwp', 'application/octet-stream' 2 가지가 있다. 확장자가 HWP 인 파일이면, 해당하는 분기점을 실행해 HWP 파일을 html 파일로 변환하고, html 파일을 PDF 파일로 변환한다. HWP mime type 을 추가한 코드는 그림 13 과 같고 이에 해당하는 분기점 코드는 그림 14 와 같다.

```
def main():
    conversions = {
    #hwp
    "application/x-hwp": {"type": "pyhwp"},
    "application/octet-stream": {"type": "pyhwp"},
 }
```

Figure 13. Added HWP time type
그림 13. 추가된 HWP mime type

```
elif conversion["type"] == "pyhwp":
    print_flush(f"Converting to xhtml using pyhwp")
    args = [
        "hwp5html",
        "--output",
        "/tmp/xhtml",
        "/tmp/input_file",
    ]
    try:
        p = subprocess.run(args, timeout=60)
    except subprocess.TimeoutExpired:
        print_flush(
            "Error converting document to xhtml, pyhwp timed out after 60 seconds"
        )
        sys.exit(1)
    if p.returncode != 0:
        print_flush(f"Converting to xhtml failed: {p.stdout}")
        sys.exit(1)
    print_flush(f"Converting to PDF using wkhtmltopdf")
    args = [
        "xvfb-run",
        "wkhtmltopdf",
        "/tmp/xhtml/index.xhtml",
        "/tmp/input_file.pdf",
    ]
    try:
        p = subprocess.run(args, timeout=60)
    except subprocess.TimeoutExpired:
        print_flush(
            "Error converting document to PDF, wkhtmltopdf timed out after 60 seconds"
        )
        sys.exit(1)
    if p.returncode != 0:
        print_flush(f"Converting to PDF failed: {p.stdout}")
        sys.exit(1)
    pdf_filename = "/tmp/input_file.pdf"
else:
    print_flush("Invalid conversion type")
    sys.exit(1)
```

Figure 14. Run module 'pyhwp, wkhtmltopdf'
그림 14. pyhwp, wkhtmltopdf 모듈 실행

Docker 이미지에서 HWP 파일 확장자를 지원하면서 pyhwp 과 wkhtmltopdf, 한글 언어팩을 추가로 설치해야 한다. Docker 이미지를 만들고 배포하기 위한 Docker file 은 그림 15 와 같다.

```
FROM ubuntu:20.04

ENV LANG="en_US.UTF-8":"ko_KR.UTF-8"
ENV LANGUAGE = "ko_KR:ko:en_GB:en"

RUN apt-get update && \
    apt-get install -y --no-install-recommends sudo python3 python3-magic python3-pil poppler-utils graphicsmagick ghostscript tesseract-ocr libreoffice tesseract-ocr-all && \
    apt-get install -y --no-install-recommends python3-pip xvfb xfonts-100dpi xfonts-75dpi xfonts-scalable xfonts-cyrillic wkhtmltopdf flashplugin-nonfree language-pack-ko && \
    pip3 install --pre pyhwp six && \
    locale-gen ko_KR.UTF-8 && \
    apt-get install -y --no-install-recommends fonts-nanum fonts-nanum-coding fonts-nanum-extra fcitx-hangul && \
    apt-get upgrade -y && \
    rm -rf /var/lib/apt/lists/* && \
    useradd -ms /bin/bash user

COPY ./script/* /usr/local/bin/
```

Figure 15. Docker file

그림 15. Docker file

V. 구현 결과

5.1 Dangerzone

Python script 와 Shell script, Docker 를 활용해 독립적인 공간에서 의심 파일을 안전한 파일로 변환하는 도커 이미지 파일을 서버에 설치하여 해당 기능을 구현하였다. 먼저 front-end 개발이 끝난 확장프로그램을 로드한다. 그 후 UI 에서 Browse and Upload 버튼을 통해 의심 파일을 업로드하여 서버로 전송한다. [그림 16].

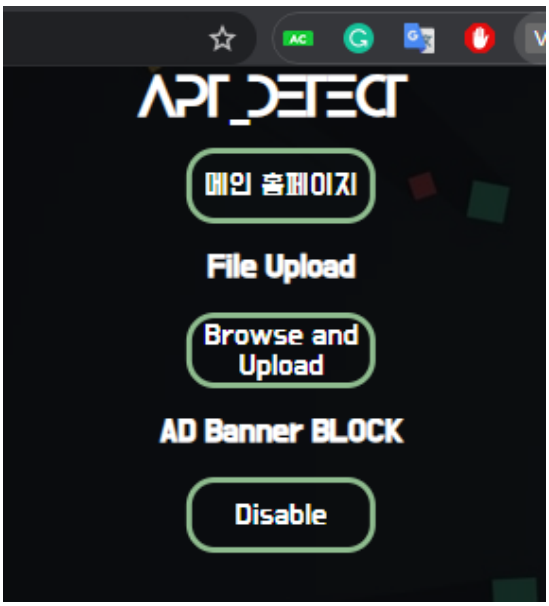

Figure 16. Chrome extension launch screen
그림 16. Chrome extension 실행화면

전송된 파일은 HWP 확장자가 추가된 Dangerzone 도커 이미지 파일을 통해 안전한 파일로 변환된다. 그 후 VirusTotal API 를 활용해 의심 파일을 검사하고 그 결과를 사용자에게 전송한다. 사용자는 악성코드 리스트 보고서를 볼 수 있고 안전한 파일을 다운로드할 수 있다[그림 17].

Figure 17. Malicious Detection Report Results Screen
그림 17. 악성 탐지 보고서 결과화면

5.2 URL 안전도 검사

Javascript 와 Python 을 활용하여 프로그램을 구성하였다. 사용자가 사이트에 그림 18 과 같이 우클릭하면, Chrome Extension 은 해당 URL 을 받아 서버로 전달하고, 서버에서 해당 URL 을 VirusTotal API 를 사용하여 분석한다. 결과화면으로는 안전/ 주의/ 위험/ 매우 위험 4 가지가 존재한다. 그림 19 는 검사 결과가 ‘위험’일 때의 결과 화면이다.

Figure 18. Program Right-Click Screen
그림 18. 프로그램 우클릭 실행화면

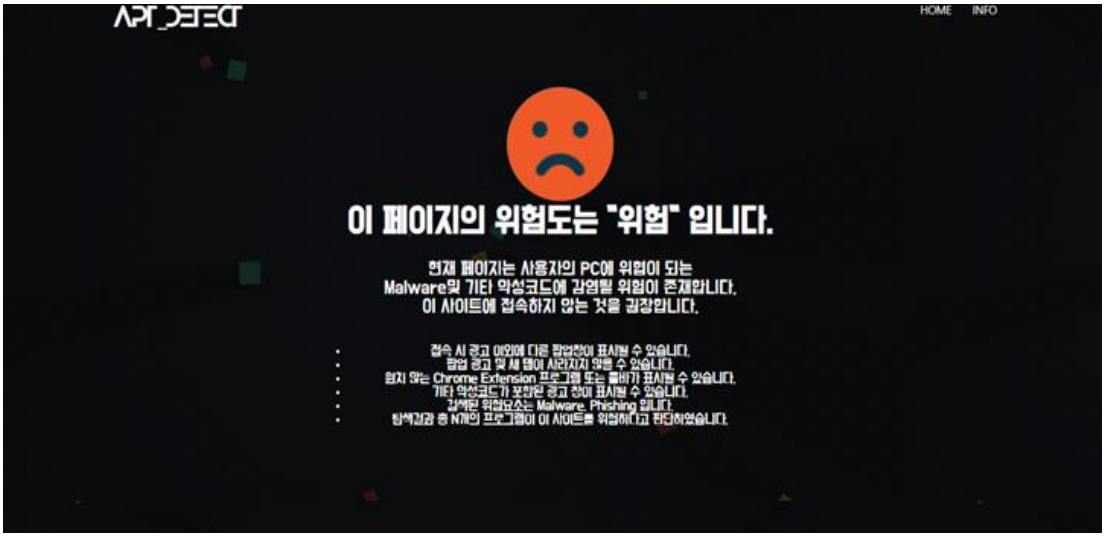

Figure 19. screen when URL check result is 'Risk'
그림 19. URL 검사 결과가 ‘위험’일 때의 실행 화면

5.3 광고 차단

Javascript 와 Python 을 활용하여 프로그램을 구성하였다. 사용자가 사이트에 접속 시 Chrome Extension API 를 활용해 광고 배너를 차단하며 차단 기준은 EasyList 에서 제공하는 사이트를 저장해 둔 blocked_sites 에서 URL 의 존재여부로 결정한다[그림 20]

Figure 20. When Activating Ad Blocking
그림 20. 광고 차단 시 실행화면

VI. 결론

이 프로젝트는 악성코드를 지닌 문서 내부 데이터를 확인할 수 있다는 점, HWP, Ms office 등 다양한 확장자 변환 기능을 제공한다는 점, 악성사이트 차단 및 사이트별 위험도 확인 기능을 제공한다는 점에서 사용자들이 APT 공격을 예방하는 데 도움을 줄 수 있을 것이라 기대한다.

또한 Chrome extension 상용으로 사용자에게 높은 접근성과 편의성을 제공하고 오픈소스로 무료로 사용할 수 있으며 편의에 따라 개량도 가능하여 다양한 사용자들이 쉽게 사용할 수 있을거라 기대된다.

VII. 참고문헌

[1] VirusTotal API, https://www.virustotal.com/
[2] Chrome extension API, https://developer.chrome.com/docs/extensions/reference
[3] Dangerzone github, https://github.com/firstlookmedia/dangerzone-converter
[4] Pyhwp module, https://pythonhosted.org/pyhwp/ko/
[5] wkhtmltopdf module, https://wkhtmltopdf.org/
[6] Wikipedia, “APT” https://ko.wikipedia.org/wiki/%EC%A7%80%EB%8A%A5%ED%98%95_%EC%A7%80%EC%86%8D_%EA%B3%B5%EA%B2%A9
[7] Boannews "The most frequently discovered malicious code in the first week of February, No. 1 information-stealing type 'Agent Tesla'", https://www.boannews.com/media/view.asp?idx=94881&page=1&kind=1
[8] Jiransecurity SaniTOX, https://www.jiransecurity.com/products/sanitox
[9] McAfee Webadvisor, https://www.mcafee.com/en-us/safe-browser/mcafee-webadvisor.html
[10] Development of an open source-based document-type malware blocking program, 2020, Seo Minjeong, Ko Heesoo, Yang Hyungji, Kang Nimjoo, Kim Kwanyoung.
[11] Park Eungyong, “Jump to django”, easy publishing, 2021
[12] Yong Chanho, “Go ahead! Docker/Kubernetes: Managed containers easily understand with kind explanations”, Wikibooks, 2020

김희은*(Heeeun Kim)*

2018 년 3 월 ~ 현재: 아주대학교 사이버보안학과 학사과정

관심분야 : 디지털 포렌식, 악성코드 분석, 모의해킹 등

손태식(Taeshik Shon)

2004-2005 University of Minnesota Research Scholar
2005-2011 삼성전자 통신연구소/DMC 연구소 책임연구원
현재: 아주대학교 사이버보안학과 교수

관심분야 : Digital Forensics, Vehicle Security, ICS

김두원*(Duwon Kim)*

2016 년 3 월 ~ 현재: 아주대학교 사이버보안학과 학사과정

관심분야 : AI, 시스템 개발 등

한광석*(Gwangseok Han)*

2016 년 3 월 ~ 현재: 아주대학교 사이버보안학과 학사과정

관심분야 : AI, XR, 웹 개발 등

성지훈*(Jihoon Seong)*

2016 년 3 월 ~ 현재: 아주대학교 사이버보안학과 학사과정

관심분야 : 안드로이드, AI, 웹 개발 등