2022 사이버보안트랙 캡스톤 디자인 최종보고서

악성 위협 감지 시스템

지도교수 : 최원석 교수님

2조 1971086 이채은
1971100 이은서
1771378 송승민
1971236 서아름

1. 프로젝트 개요

그림 2 <2022년 6월 첫째 주 악성코드 통계[자료=안랩 ASCE 분석팀]>

2022년 6월 첫째 주, 가장 많이 발견된 악성코드인 'Form book'은 Info stealer(정보 탈취형) 악성코드로 33.7%를 나타내고 있다. 다른 인포스틸러 악성코드들과 동일하게 스팸메일을 통해 유포된 것으로 확인된다.[1]

이러한 스팸메일은 APT 공격의 주요 타겟이다. APT 공격은 스팸메일과 악성 첨부파일을 통해 수일~수년 단위 지속되는 공격을 말한다.

Content Disarm & Reconstruction(이하 CDR[2])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' 기술로 외부 보안 위협으로부터 최신 대응 기술로 손꼽히고 있다.

Open source 프로그램 중 CDR 기법을 사용하는 'Dangerzone'은 한국에서 많이 사용되는 문서 타입 HWP 확장자를 지원하지 않는다. 또한, 사람들이 많이 사용하는 이메일 시스템에서 자체적으로 유해 URL을 차단하지 못해 악성 URL에 접속할 우려가 크다.

이러한 악성 위협으로부터 안전하도록 기존 문서 확장자들의 pdf 변환도 가능하며, 추가로 HWP 파일도 pdf로 변환을 지원하는 'Dangerzone' 프로그램, URL 검사와 File 감염 여부 검사가 가능한 Web과 광고배너 차단 기능을 가진 Chrome extension을 개발하였다.

2. 주제 선정 배경

가. 이메일을 이용한 지능형 APT 공격 사례[3]

1) 사회적 배경

코로나19로 인한 원격·재택 근무 증가와 클라우드 환경으로 업무 환경이 전환되어 업무 전산화가 가속화됨에 따라 사이버 공격 또한 증가하게 되었다. 그중 이메일을 이용한 공격이 사이버 피해의 시발점이 되고 있으며, 언론에서도 많은 해킹사고의 근원이 이메일 공격에서 시작 되었다고 지속 보도 하고 있다.

2) E-mail이 사이버 공격에 이용되는 이유

해커가 이메일을 공격에 이용하는 이유는 여러 가지가 있다.

가) 보안 인식이 높아지면서 개인은 물론 기업은 당연하게 방화벽과 같은 보안 제품을 도입하였다. 이에 보안담당자들은 웹, 메일, DNS 등 이용자와 고객에게 최소한의 서비스만 외부에 노출시켜 외부에서 침투하기 어렵도록 하였다. 또한, 코로나19로 비대면 업무가 적극적으로 활성화되며 메일과 웹에 대한 의존도는 더욱 높아지게 되었다. 외부에 최소 서비스만 공개하더라도, 웹이나 메일은 공개할 수밖에 없기 때문이다. 해커는 이러한 공개 서비스를 대상으로 한 웹 해킹, 메일을 통한 악성코드나 악성 URL 유포로 감염·정보 탈취 시도가 증가하게 된 것이다.

나) 기업의 내·외부 망 분리로 내부 정보 유출이 어려워져 메일 계정 탈취를 통해 메일 내 업무 정보를 유출하고, 피해계정을 이용한 2차 공격 시도 등으로 악용하고 있다.

다) 해커는 가장 손쉬운 정보 유출 피해 공격 방법으로 이메일을 이용한다. 지인 또는 신뢰 기관·사람으로 사칭하거나 업무 관련 메일로 위장하고, 사회적 이슈를 이용하는 등 사람의 심리를 이용하는 동시에 지능적·지속적 APT 공격[1]을 한다.

3) E-mail을 이용한 공격 사례[4]

이메일 공격 유형은 크게 사용자 계정 정보를 탈취하기 위한 피싱 메일, end-Point 감염을 목적으로 하는 악성파일 첨부, 특정 대상을 감염시키기 위한 스피어 피싱으로 구분할 수 있다.

가) 포털 운영진·고객센터를 사칭한 피싱 메일

해커는 개인정보 탈취를 위한 피싱 메일 공격을 시도할 때, 포털 고객센터, 보안센터 등 포털 운영진을 사칭해 로그인 시도 알림, 비밀번호 유출, 아이디 충돌 등으로 비밀번호 변경을 유도하거나 본인 확인을 위한 로그인으로 유도한다. 최근에는 각 기관이나 기업들의 계정정보 탈취를 위해 정교하게 로그인 페이지로 위장하여 계정 탈취를 시도하고 있다. 메일 본문 내 링크를 클릭하고 계정정보를 입력하면 해커에게 정보가 전송되어 정보 유출, 사칭 등 추가 피해가 발생 될 수 있다.

---

1) APT(Advanced Persistent Threats, **지능형 지속 위협**)

피싱 로그인 페이지를 분별하는 방법은 계정정보 입력 전, 연결된 페이지의 도메인 주소(URL)를 확인해야 한다. 피싱 페이지는 공식 포털 도메인 주소가 아닌 공격자의 도메인 주소를 사용하므로 정상 로그인 페이지와 구별을 할 수 있다.

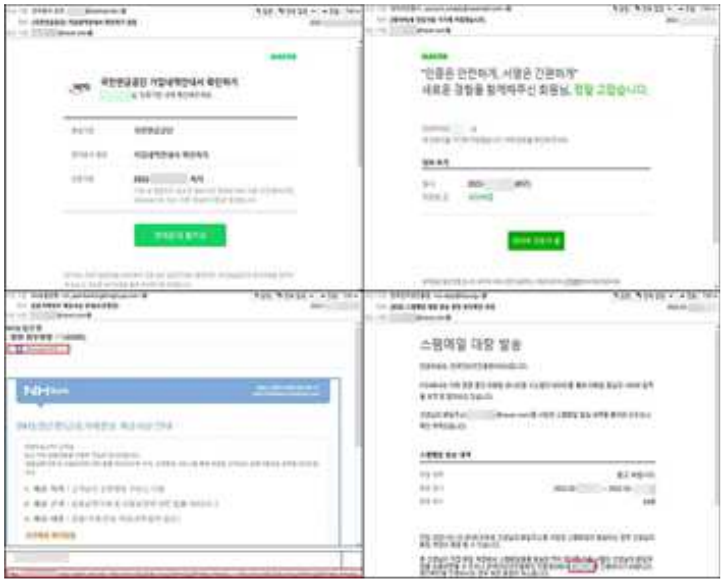


그림 3 <최근 피싱 메일 유포 사례>

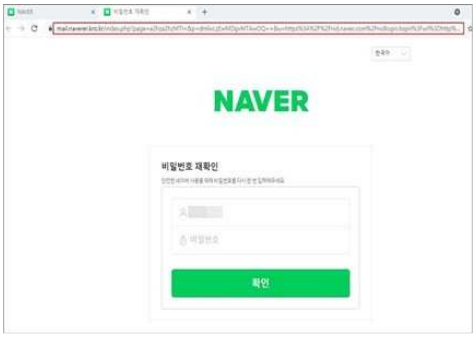


그림 4 < 피싱 NAVER 도메인 >

그림 5 < 정상 NAVER 도메인 >

나) 파일 다운로드를 유도하는 피싱 로그인 페이지

해커는 자신의 로그인 페이지로 메일 수신자를 유도해 개인정보를 입력하도록 유도할 때 첨부파일을 위장한 링크를 포함해 메일을 전송한다. 피싱 메일의 첨부파일은 실제 보이는 확장자와 다른 실행파일로, HTML 이미지 태그 등으로 구성되어 있다. 수신자가 파일 다운로드를 위해 클릭하면 피싱 페이지로 연결하여 계정정보와 같은 개인정보를 입력하도록 유도한다. 정보를 입력 후 전송하면 정상 파일이 다운로드 되는 것처럼 보이지만, 실제로 계정정보가 해커에게 전송되고 수신자는 계정정보 유출 사실조차 인지하기 어렵다.

다음은 최근 피싱 메일에 사용된 제목들이다.

• 차단한 지역에서 로그인이 시도되었습니다.
• 회원님의 비밀번호가 유출되었습니다.
• 회원님의 계정을 시급히 보호하세요.
• 해외 지역에서 접속 시도가 차단되었습니다.
• 로그인 시도 안내
• 새로운 기기에서 로그인이 시도되었습니다.
• [긴급] 회원님의 계정이 정지상태로 전환되었습니다.
• [긴급] 회원님의 아이디에 대한 중복요청이 접수되었습니다.
• [긴급] 고객님의 비밀번호찾기 요청이 20회 이상 감지되었습니다.
• 계정 아이디가 충돌하였습니다.
• 회원님의 연락처 휴대전화 번호가 변경되었습니다.
• 계정 복구 코드가 추가되었습니다.
• 고객님의 계정에서 비정상적인 활동이 감지되었습니다.
• 회원님의 계정이 이용제한 되었습니다.
• 고객님의 네이버 인증서가 발급되었습니다.
• [네이버] 새 인증서를 기기에 저장했습니다.
• [중요 알림] 메일함 백업 요청이 접수되었습니다.
• [네이버 전자문서] 회원님께 중요한 전자문서가 도착했습니다.

그림 6 < 포털 운영진・고객센터 사칭 메일 제목 사례 >

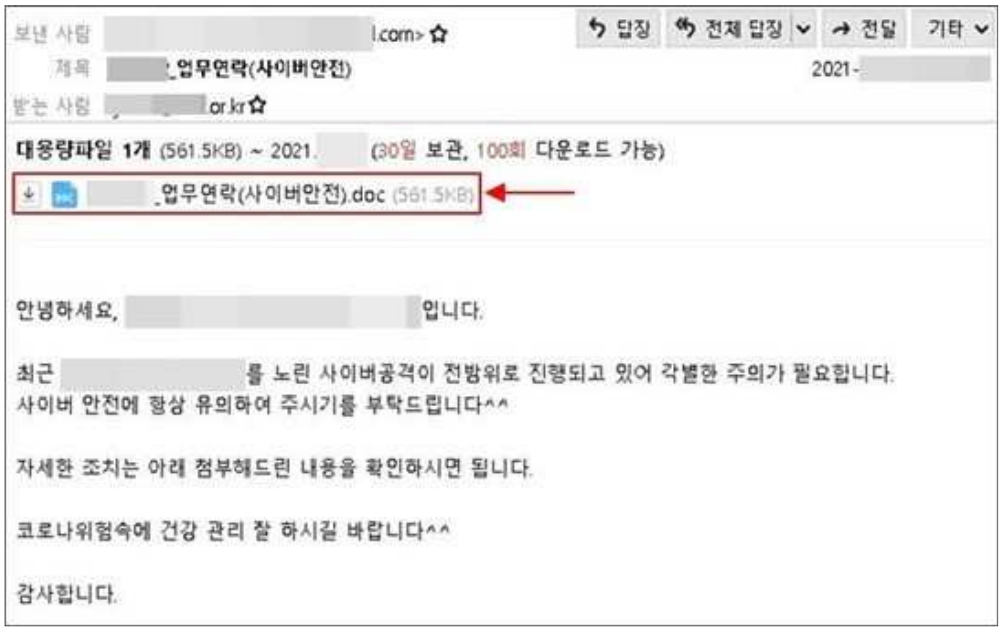


그림 7 <다운로드 파일로 위장한 피싱 메일 예시>

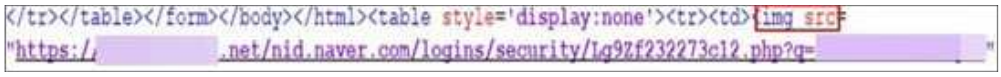


그림 8 <피싱 URL 링크>

또한 End-Point 악성코드 감염을 목적으로 하는 악성파일 첨부 이메일 공격은 MS Office, 한글과 컴퓨터 문서 정상 기능을 악용하거나 악성 실행파일을 첨부 방식을 이용한다.

다) 문서의 정상 기능을 악용한 악성파일 첨부

해커는 공격 벡터 중 MS Office나 한글과 컴퓨터 문서의 정상 기능을 악용하여 End-Point 기기에 악성코드를 감염시키기도 한다. 이는 이메일을 주로 업무적으로 활용하는 사용자들을 타겟으로 공격하며, 대표적인 사례로 MS Office의 macro기능

과 한글의 개체 연결삽입(OLE) 기능이 있다.

MS Office의 매크로[5]는 반복된 작업을 자동화하기 위해 사용할 수 있는 여러 개의 명령을 그룹화한 것으로 Word, Excel, PowerPoint 등의 부가 기능으로 사용되고 있다. 또한 국내에서 사용하는 한글과 컴퓨터 문서는 독립적인 자료를 하나로 연결하는 개체로 지원하는데 OLE 개체 삽입으로 실행파일을 링크 방식으로 삽입할 수 있다. 해커는 문서의 정상 기능을 악용하여 악성 매크로·실행 파일을 사용자가 직접 실행하도록 유도하여 악성코드 감염을 시도하고 있다.

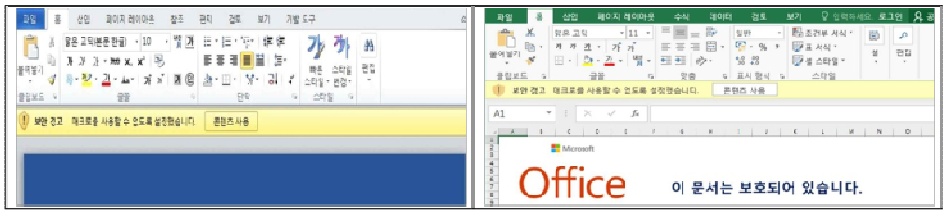


그림 9 < MS오피스 악성 매크로 활성화 유도 화면 >

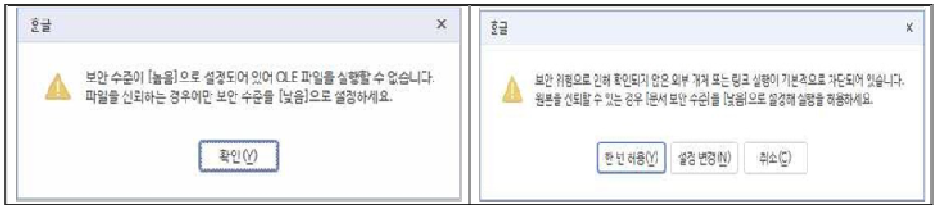


그림 10 < 한글 OLE 개체 기능 활성화 유도 화면 >

이러한 기능은 문서 프로그램에서 제공하는 정상적인 기능으로 최신 버전 업데이트 및 보안 패치 적용과 상관없이 실행할 수 있다. 때문에, 해커는 악성 문서 파일을 주변 지인 사칭 또는 업무 관련 메일 등으로 위장하여 공격을 시도한다. 만약 메일 수신자가 의심하지 않고 첨부파일을 실행한 후 팝업창을 클릭할 경우, 육안상으로는 정상적 문서가 보이나 악성 매크로가 자동 실행되어 수신자 PC는 악성코드에 감염되거나 정보가 유출된다. 악성 매크로는 주로 내부 변수들을 난독화하여 백신 프로그램을 우회하기 때문에 탐지가 어렵다.

그림 11 <문서 내 악성 매크로 일부>

한글 문서 악성코드는 과거에 자바스크립트나 포스트스크립트에 악성코드를 삽입하여 유포하는 사례가 많았지만, 최근에는 OLE 개체 실행 취약점을 많이 악용하는 추세

이다. 문서 실행 시 알림창으로 '확인' 또는 '허용'을 클릭하면 링크 파일이 실행되어 악성코드에 감염된다.

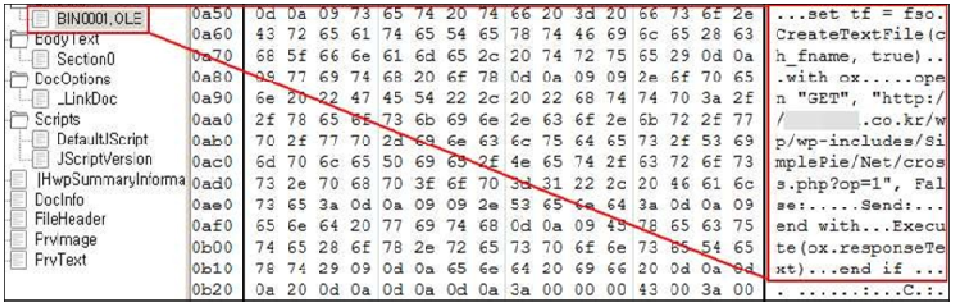


그림 12 <문서 내 OLE 개체 삽입 일부>

해커는 간혹 이메일에 악성 실행파일을 문서 아이콘으로 위장하여 수신자 클릭을 유도하기도 하는데, 첨부된 파일을 문서로 인식하고 열람 및 실행 시 사용자 화면에는 정상 문서가 실행되는 것처럼 보이지만 은밀하게 정보를 탈취하며 추가적 공격을 시도한다.

구분	확장자 예시
실행파일	exe, MSI, bat, scr, pif, vbs, wsf 등
문서 아이콘 위장 이중 확장자	pdf(공백).exe, pdf(공백).scr, doxs(공백).exe, xlsx(공백).exe 등

표 1 <악성파일 확장자 정보>

나. 이메일을 이용한 지능형 APT 공격 대응 방안[6]

가) 해커의 메일을 이용한 공격은 지인을 사칭하여 수신자의 심리적 의심을 없애거나 사회적 이슈, 업무를 메인으로 하여 발신해 호기심을 자극하는 등 사람의 심리를 이용하고 있다. 사람의 심리를 이용한 사회 공학적 기법은 정보시스템을 해킹하는 것이 아니라 사람을 해킹하는 것이라고 한다. 공격의 구체적인 방법으로는 수신자가 피싱 로그인 페이지에 수신자의 정보를 입력하게 하거나 악성 첨부파일을 실행하여 악성코드를 설치하도록 유도하고 있다.

나) 기업들은 지속적인 보안 교육을 통해 직원들이 경각심을 가지도록 해야 하며, 모의훈련을 통해 메일로 인한 사이버 위협에 대한 대응력을 높일 수 있다. 메일 수신자는 의심스러운 메일은 발신자에게 전화, 문자 등으로 확인하도록 하고 메일 본문 내 링크를 클릭한 경우 계정 및 비밀번호 등 개인정보 입력에 주의해야 한다. 마지막으로 첨부파일의 확장명을 확인하고 문서파일로 위장한 실행파일을 실행시키지 않도록 한다.

다) 다음은 '메일 이용 시 지켜야 할 보안 수칙'이다.

(1) 출처가 불분명한 의심스러운 메일 내 링크 클릭 및 첨부파일 열람 주의

(2) 메일 본문 내 링크를 클릭하여 계정정보 입력 주의

(3) 메일 첨부파일 실행 시 파일 확장명 확인

(4) 메일 첨부 문서 열람 시 매크로, 개체 연결삽입(OLE) 허용 주의

(5) 백신 프로그램 최신 버전 업데이트 및 실시간 감시 기능 사용

(6) 매일 로그인 시 2단계 인증 적용

(7) 트위터, 인스타그램 등 SNS에 개인정보 노출 주의

라) 최근에는 금융기관 명세서를 위장하거나 보안 프로그램의 제작사를 사칭한 사례도 있다. 첨부파일은 파일 압축이나 문서 비밀번호 설정으로 보안 기능을 피해 수신자에게 전달되도록 하기 때문에 주의 깊게 살피지 않으면 피해가 우려된다.

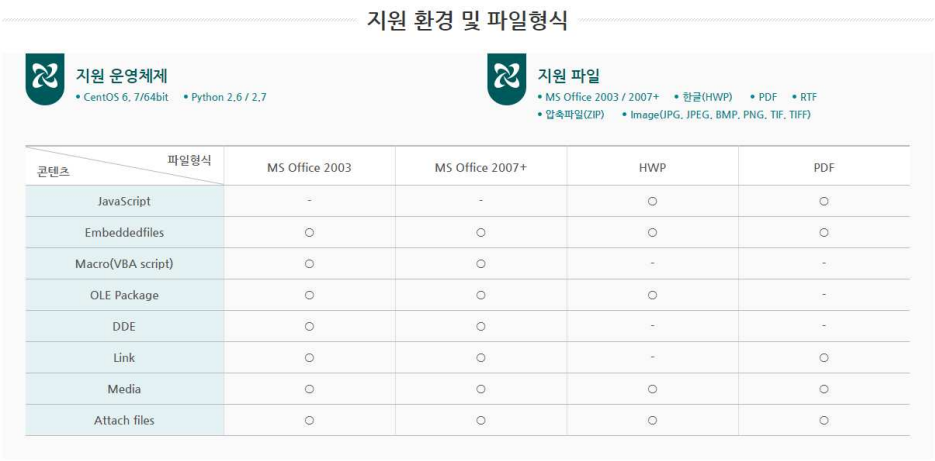
마) 위 '(다) 메일 이용 시 지켜야 할 보안 수칙'을 준수하는 개발물을 본 캡스톤 디자인에서 개발하고자 하였다.

3. 유사 프로젝트와의 차별성

가. 관련 제품

지능화된 악성 위협이 증가하면서 샌드박스 회피 및 우회 기술이 발전됨에 따라 샌드박스만으로는 지능화된 APT 공격 대응에 한계를 보인다. 글로벌 IT 자문기관'Gartner'[7]에서는 차세대 APT 대응 방안 5가지 핵심 보안 패턴 중 하나로 CDR 사용을 권고하였다. APT 공격을 방어하기 위해 많은 백신 프로그램이 존재하지만, CDR 기능을 지원하고 URL을 검사하는 기능을 지원하는 경우는 드물다. 현재, 실제 운영되는 CDR 기법을 지원하는 프로그램과 URL 검사를 지원하는 프로그램에 대한 분석을 제시한다.

1) 지란지교시큐리티:SaniTox



콘텐츠 \ 파일형식	MS Office 2003	MS Office 2007+	HWP	PDF
JavaScript	-	-	○	○
Embeddedfiles	○	○	○	○
Macro(VBA script)	○	○	-	-
OLE Package	○	○	○	-
DDE	○	○	-	-
Link	○	○	-	○
Media	○	○	○	○
Attach files	○	○	○	○

그림 13 <지란지교 시큐리티 SaniTOX 지원 환경 및 파일 형식>

국내에서 CDR 기법을 지원하는 대표적 프로그램은 지란지교 시큐리티의 SaniTOX이다. SaniTOX[8]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. 그

렇지만, Linux에서 사용되는 Open Document Format 확장자들은 지원되지 않는다. 또한 모든 기능은 유료로 제공된다.

2) DangerZone

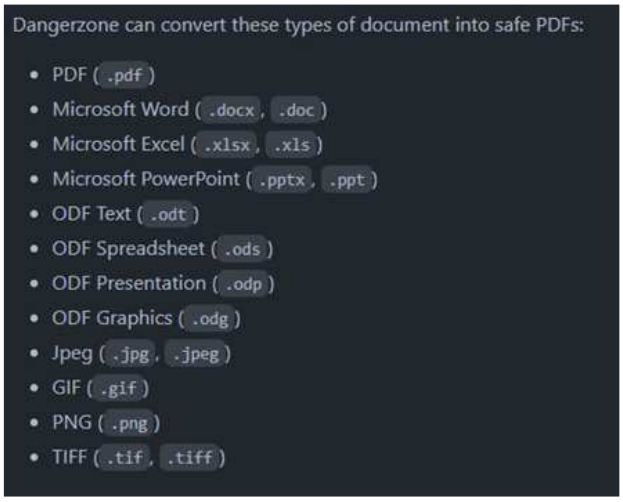


그림 14 <Dangerzone 지원하는 확장자>

현재 github에 공개되어 있는 CDR 기법을 적용한 프로그램으로 'Dangerzone'이 있다. 해당 프로그램은 Pixel 렌더링을 통해 안전한 flat PDF 파일로 바꾸는 것이 가능하며, 무료로 사용할 수 있다. 그러나 사용을 위해서는 사용자가 docker와 'Dangerzone'을 local machine에 설치해야 하는 불편함이 존재하며, 한국에서 많이 사용되는 한글 문서의 확장자 hwp는 지원되지 않는다.

3) McAfee Web Advisor

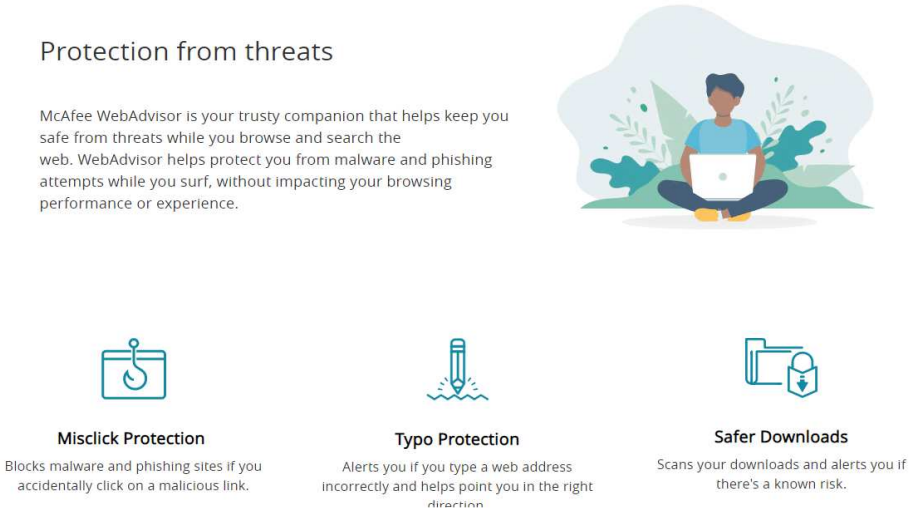


그림 15 <McAfee Web Advisor 기능>

McAfee Web Advisor는 피싱 사이트에 들어가거나 악성파일 링크를 잘못 클릭한 경우 block 한다. 또한, 파일을 내려받을 때 악성코드 유무 검사 진행이 가능하다. Chrome

Extension으로 지원되어 편리함도 제공한다. 그러나 URL 검사 기능이 없어 악성 URL 유포에 수월하게 작용한다.

나. 본 프로젝트의 차별성

앞서 설명한 기존 제품들과의 차별성을 비교하면 다음과 같다.

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

SaniTox는 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 Web Advisor는 Chrome extension이고 무료로 편리하게 사용이 가능하나 웹 브라우저를 통해 연결된 URL의 약성 여부를 판단해 줄 뿐 mail을 통해 전달되는 URL을 검사하지는 않는다.

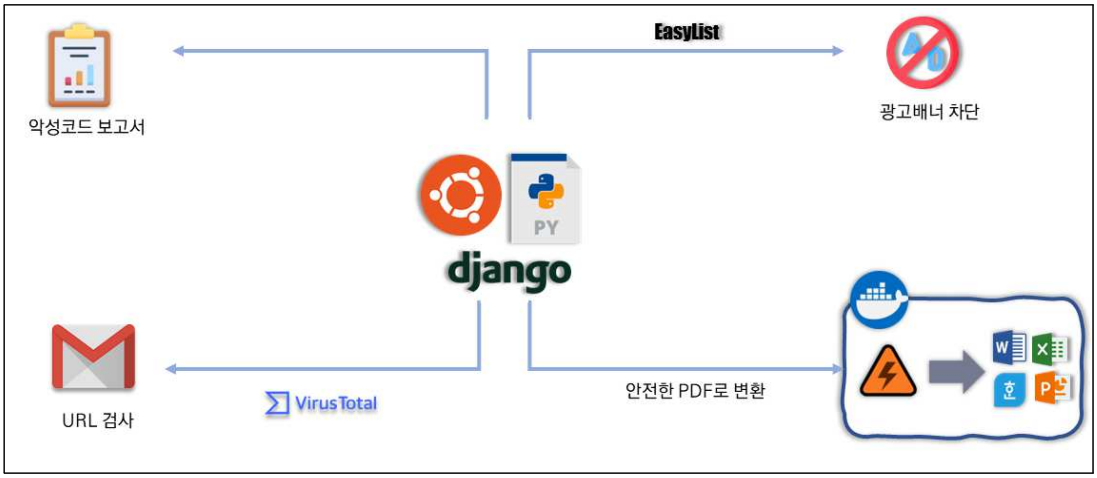
이에 본 프로젝트에서는 앞의 단점들을 보완하여 HWP 기능을 지원하고 무료로 편리하게 CDR 기법을 사용할 수 있는 것과 메일 내 URL을 검사할 수 있는 것에 차별성을 두었다.

제품	HWP 지원	Open Document Format	안전한 파일 변환	악성코드 유무	편리성	URL 검사
DangerZone	X	O	O	X	X	
SaniTox	O	X	O	O	X	
MC Afee Web Advisor				O	O	X
본 개발물	**O**	**O**	**O**	**O**	**O**	**O**

표 2 <관련 유사 프로젝트 조사 및 차이점 분석>

4. 주요 기능 및 시스템 구성

가. 전체 시스템 구성



Ubuntu 환경에서 Python과 django 을 이용해 구현된 web-server 환경에서 크게 4가지 기능을 제시한다. 4가지 기능은 다음과 같다.

나. 핵심 기능

1) URL 검사

가) 사용자가 브라우저 또는 이메일로 수신한 URL에 접속하기 이전 해당 URL에 대해 위험도를 확인할 수 있음

나) 악성 URL일 경우, 탐지된 malware 확인 가능

2) File 검사

가) 사용자가 파일을 검사하면, 악성 위협으로부터 안전한 파일인지, 안전하지 않은 파일인지 확인할 수 있음

나) 악성 File일 경우, 어떤 malware가 포함되어 있는지 확인 가능

3) DangerZone[9]

가) 한글 파일 HWP 변환 시, 악성 매크로·OLE 등 악성 위협을 가진 부분을 제외(삭제)하고 글 및 그림 등 본문 내용을 무손상 상태로 사용자에게 안전한 형태로 제공됨

나) 기존 Dangerzone에서 제공하는 문서 확장자도 동일한 서비스를 제공함

4) AD Block

가) 사이트 내 무분별한 광고 배너를 차단해 쾌적하게 사이트를 이용할 수 있도록 함

5. 개발 시 요구된 기술 및 소프트웨어/하드웨어 소개

본 캡스톤 디자인에서 악성 위협 감지 시스템을 개발하기 위해 python으로 Dangerzone에 한글 확장자를 지원하도록 구현하고 URL 분석과 파일 분석을 구현하였다. Chrome

Extension으로 동작하는 광고 배너 차단 프로그램 또한 구현하였다.
개발환경은 및 사용한 기술은 아래와 같다.

1) Python

가) Opensource Dangerzone에 hwp 확장자 추가

나) Virus Total API를 이용해 악성파일 및 URL 위험 탐지

다) Easy List API를 이용하여 광고 차단

라) Django를 이용한 Web - Server 개발

2) Ubuntu 20.04 LTS

가) 사용자의 파일을 업로드 시, 우분투 서버로 전달

3) 파일 검사 및 URL 검사

가) Virus Total OpenAPI [10]

4) 악성파일 변환

가) CDR 기법을 적용한 DangerZone →기존 Dangerzone 기능을 원활 사용 가능

나) Docker로 악성파일 감염 방지를 위해 네트워크가 분리된 독립적 공간 사용

5) 광고 배너 차단

가) Chrome Extension API (onBeforeRequest)[11]

나) Easy List API

다) Membranes pattern

라) Shadow DOM bypassing

6. 구현 결과 상세 소개 (사진 포함)

가. URL Scan 기능

1) 알고리즘

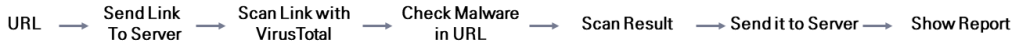


그림 15 <URL Scan 기능 DFD>

사용자가 원하는 URL을 사이트에 입력하면 모든 URL에 대하여 검사할 수 있다. 사용자가 URL scan을 시도하면 Web server에서 해당 URL을 받아 서버로 전달하고, 서버에서 해당 URL을 Virus Total API로 전달하고, 분석한다. 분석 결과를 서버에서 전달받아 다시 Web Site에 표시한다.

URL에서 탐지된 malware 개수에 따라 safe/warning/danger 3가지 경우로 분류한다.

```python
def url_upload(request):
    f = open('mal_test.txt','w')
    if request.method == 'POST':
        form = MalUrlForm(request.POST)

        if form.is_valid():
            geturl = request.POST.get("url", "")
            if is_valid_url(geturl)==False:
                return render(request, 'urlerror.html')
            cmd = "curl -s --request GET --url 'https://www.virustotal.com/vtapi/v2/url/report?apikey="+apikey+"&resource="+geturl+"'"
            resp = os.popen(cmd)
            output = resp.read()
            #changing start
            my_apikey = "6af39ae43ef096dbac787cdc0208f7669dc604821d417399a379fcd2759151ea"
            my_url = geturl
            url_scan = 'https://www.virustotal.com/vtapi/v2/url/scan'
            scan_params = {'apikey': my_apikey, 'url': my_url}
            scan_response = requests.post(url_scan, data=scan_params)

            print('Virustotal URL SCAN START (60 Seconds Later) : ', my_url, '\n')

            #time.sleep(60)

            url_report = 'https://www.virustotal.com/vtapi/v2/url/report'
            report_params = {'apikey': my_apikey, 'resource': my_url}
            report_response = requests.get(url_report, params=report_params)
            report = report_response.json()
            report_scan_date = report.get('scan_date')
            report_scan_result = report.get('scans')
```

그림 16 <views.py: URL scan Script>

2) 시연 URL

시연에 malware를 포함한 사이트 주소를 이용하기 위하여 Open Phish 사이트를 이용하였다. 본 사이트는 전 세계 네트워크의 필터링 되지 않은 수백만 개의 URL을 수신하여 타겟 브랜드별로(ex. Google, Instagram….) 피싱 URL을 선별하여 네트워크 및 지리적 위치, 피싱 키트 및 드롭 계정을 포함하는 메타데이터를 추출한다. 추출한 데이터를 기반으로 실시간으로 7일간 많은 피싱 작업이 이루어진 순위로 URL을 dashboard에 나타낸다.

그림 17 <피싱 사이트 목록을 실시간으로 업데이트하는 사이트>

본 시연에는 github, Instagram을 타겟으로하여 피싱을 수행하는 URL을 사용하였다.

가) http://melissacinta.github.io/itskillscenter-instagram-login

나) http://varnikarathee.github.io/vtf_netflix_clone.github.io

본 기능 시연에 사용할 web page이다.

그림 18 <개발된 URL Scan page>

3) 악성 URL

위에서 언급한 악성 URL을 넣고 검사한 결과이다. 탐지된 malware 개수에 따라 위험도를 나눈다.

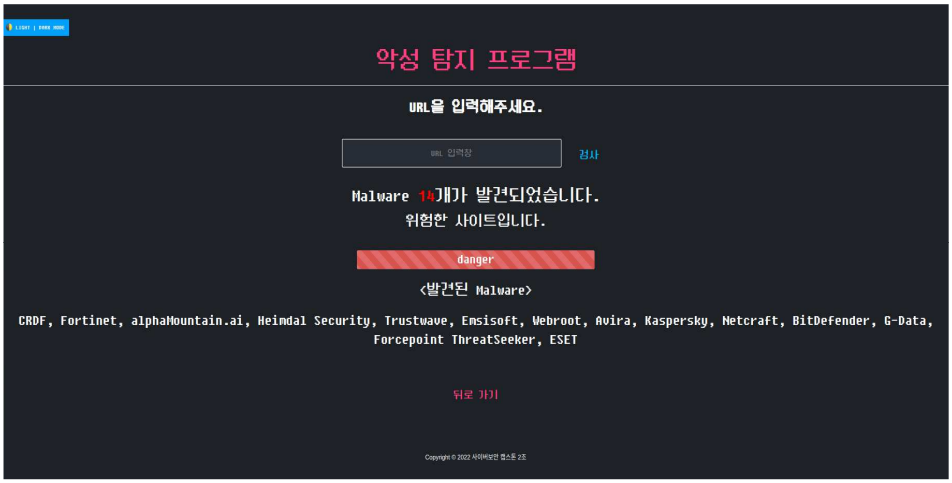


그림 19 <악성 URL 검사 결과 화면>

4) 주의 URL

위에서 언급한 악성 URL을 넣고 검사한 결과로, 탐지된 malware 개수에 따라 위험도를 나눈다. 주의 URL일 경우, 바로 해당 사이트로 접속할 수 있다.

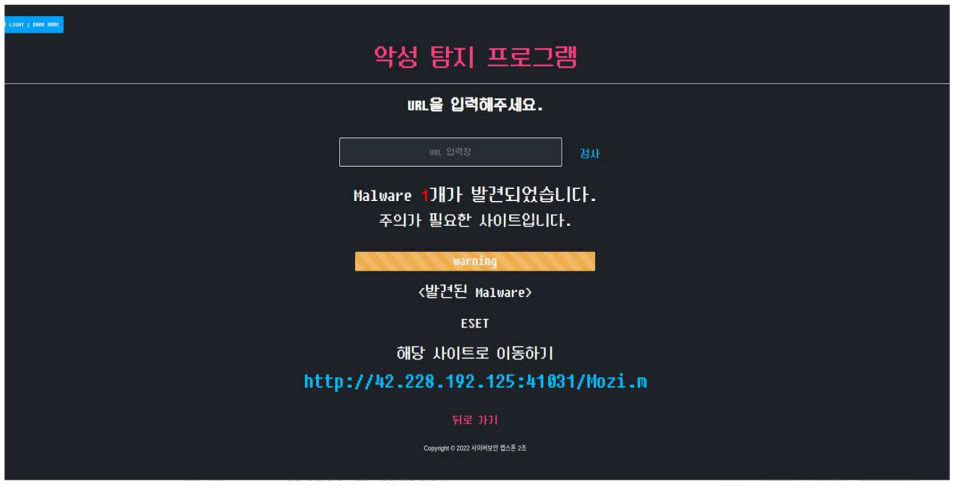


그림 20 <악성 URL 검사 결과 화면>

5) 안전 URL

한성대학교의 URL을 넣고 검사한 결과이다. 탐지된 malware 개수에 따라 위험도를 나눈다. 안전 URL일 경우, 바로 해당 사이트로 접속할 수 있다.

그림 21 <안전 URL 검사 결과 화면>

나. File Scan 기능

1) 알고리즘

File → Send File To Server → Scan File with VirusTotal → Check Malware in File → Scan Result → Send it to Server → Show Report

그림 22 <File Scan 기능 DFD>

사용자가 의심되는 File을 사이트에 업로드하면 검사가 가능하다. 사용자가 File scan을 시도하면 Web server에서 해당 File을 받아 서버로 전달하고, 서버에서 해당 File을 Virus Total API로 전달하고, 분석한다. 분석 결과를 서버에서 전달 받아 다시 Web Site에 표시한다.

```python
def upload(request):
    url_check= '123'
    context = {}
    f2 = open('mal_file.txt','w')
    if request.method == 'POST':
        uploaded_file = request.FILES['document']
        fs = FileSystemStorage()
        name = fs.save(uploaded_file.name, uploaded_file)
        context['url'] = fs.url(name)
        file = uploaded_file.name
        files = {'file': (file, open(file, 'rb'))}
        url_scan = 'https://www.virustotal.com/vtapi/v2/file/scan'
        url_scan_params = {'apikey': apikey}
        response_scan = requests.post(url_scan, files=files, params=url_scan_params)
        result_scan = response_scan.json()
        scan_resource = result_scan['resource']
        print('Virustotal FILE SCAN START (60 Seconds Later) : ', file, '\n')
        url_report = 'https://www.virustotal.com/vtapi/v2/file/report'
        url_report_params = {'apikey': apikey, 'resource': scan_resource}
        response_report = requests.get(url_report, params=url_report_params)
        report = response_report.json()
        report_scan_date = report.get('scan_date')
        report_scan_sha256 = report.get('sha256')
        report_scan_md5 = report.get('md5')
        report_scan_result = report.get('scans')
        report_scan_vendors = list(report['scans'].keys())
        report_scan_vendors_cnt = len(report_scan_vendors)
        url_check=file
        num = 1
        print(report.get('verbose_msg'), '\n')
```

그림 23 <views.py: file scan Script>

그림 24 <개발된 File Scan page>

File에서 탐지된 malware 개수에 따라 safe/warning/danger 3가지 경우로 분류한다.

2) 시연 File

시연에 사용할 파일은 '북한의 회색지대 전략과 대응 방안.hwp'로 EPS 취약점 스크립트를 포함한 악성 문서이다. 다양한 case에 대해 테스트하기 위하여 Malware Bazaar Database 사이트에서 ppt, doc 파일도 내려받아 시연에 사용하였다.

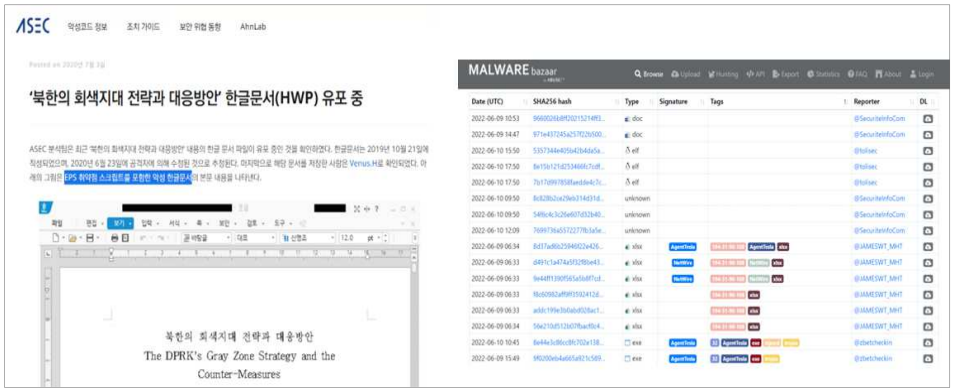


그림 25 <시연에 사용한 malware file 출처>

북한의 회색지대 한글 문서를 로컬 PC에 내려받으면, 백신에서 파일을 차단한다.

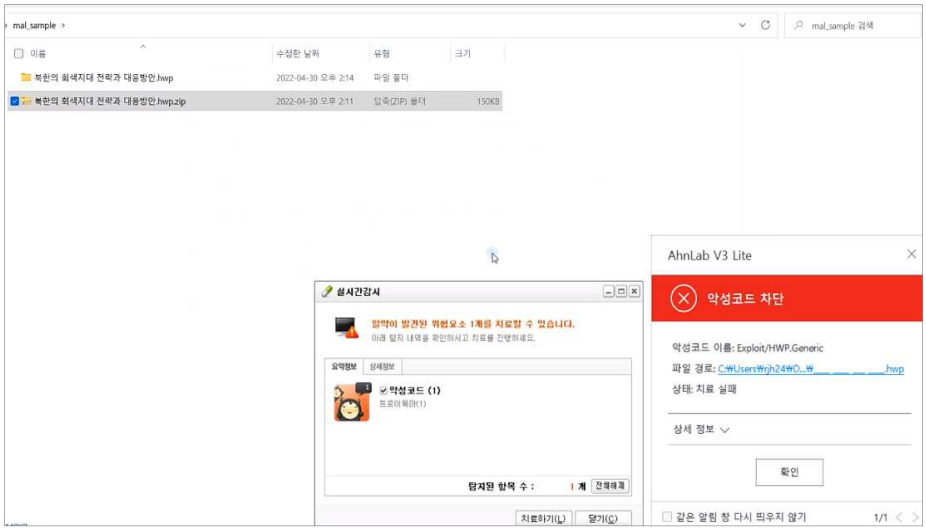


그림 26 <Local PC 백신에서 malware 탐지>

3) 악성 File 검사

언급한 악성 File을 넣고 검사한 결과로, 탐지된 malware 개수에 따라 위험도를 나눈다.

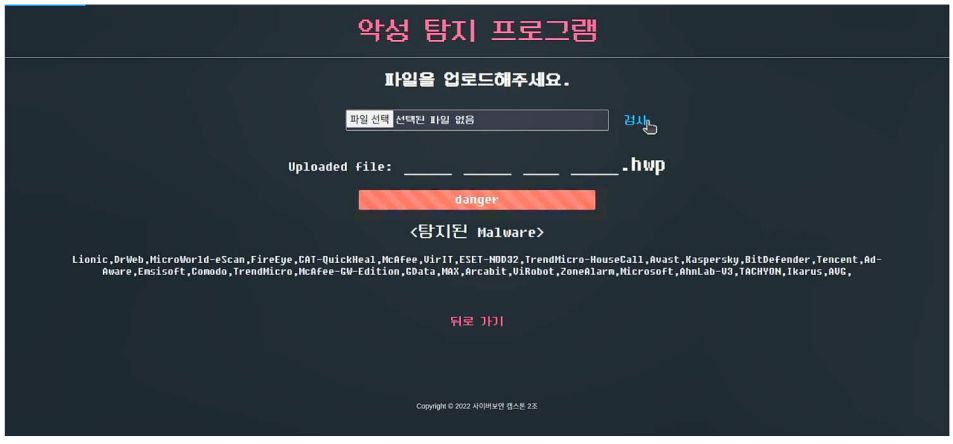


그림 27 <북한의 회색지대 전략과 대응 방안.hwp 검사 결과>

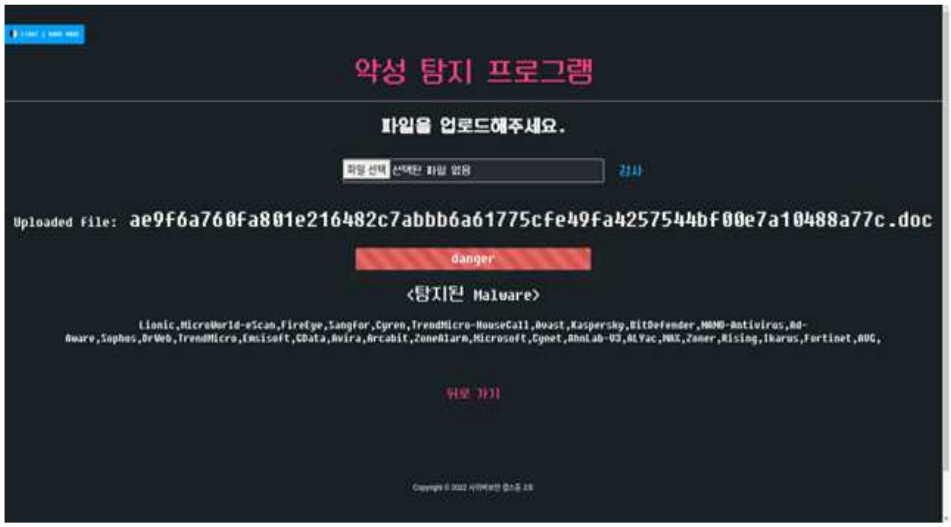


그림 28 <Malware Bazaar Site의 doc 파일 검사 결과>

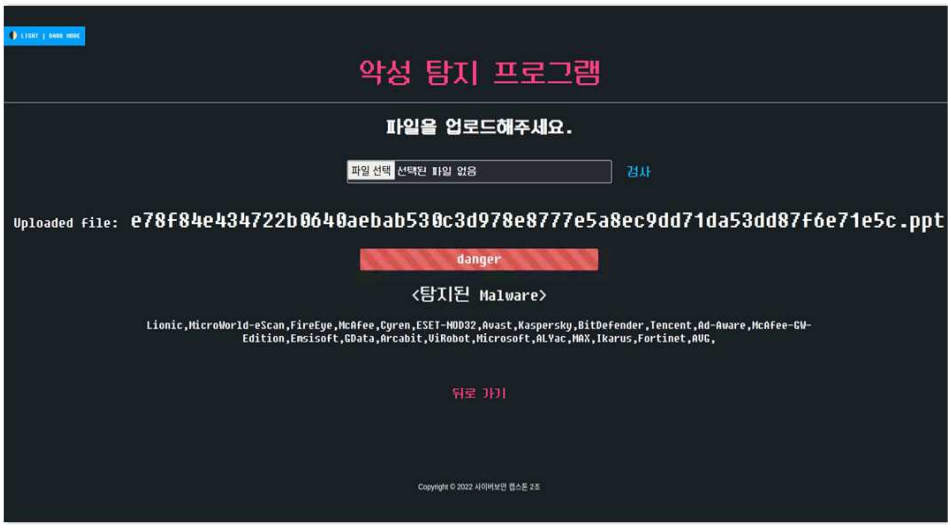


그림 29 <Malware Bazaar Site의 ppt 파일 검사 결과>

4) 안전 File 검사

한성대학교 홈페이지에서 내려받은 hwp 파일을 검사한 결과이다. 탐지된 malware 개수에 따라 위험도를 나눈다.

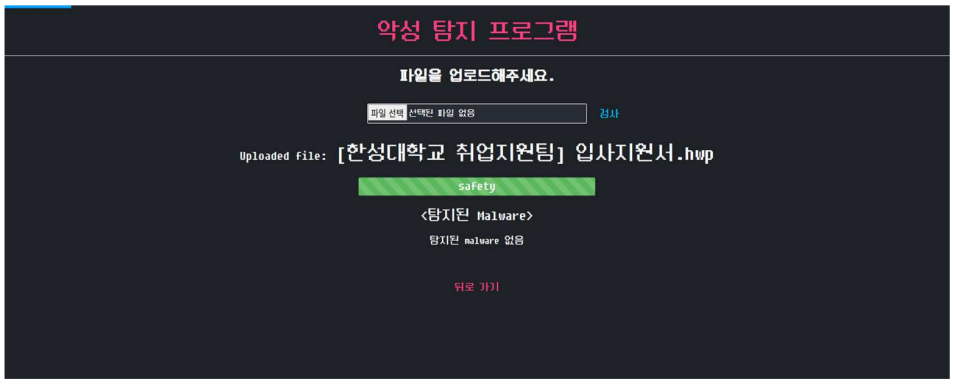


그림 30 <한성대에서 내려받은 hwp 파일 검사 결과>

다. Dangerzone

1) 알고리즘

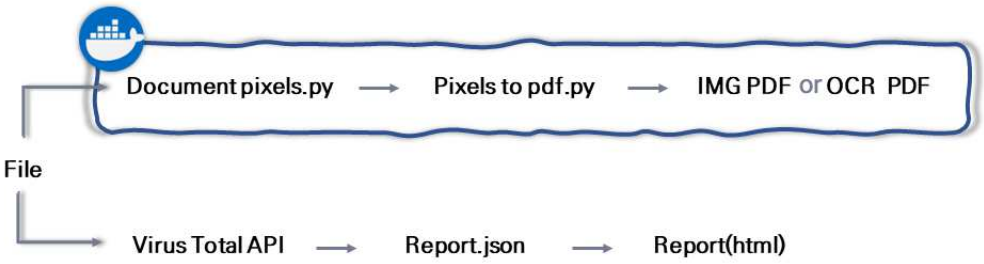


그림 31 <Dangerzone hwp 변환 기능 DFD>

```python
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)
```

그림 32 <File을 Virus Total API로 전달>

사용자가 웹에 변환할 파일을 업로드하면, Virus Total API를 통해 감염된 파일인지 검사한다. Virus Total API에서는 파일을 전달받으면서 user_id와 file hash 값을 전달받아 검사를 진행하고 서버로 검사 결과를 리턴한다. 이후, 파일을 docker IMG로 docker container를 생성하여 안전한 파일로 변환한다.

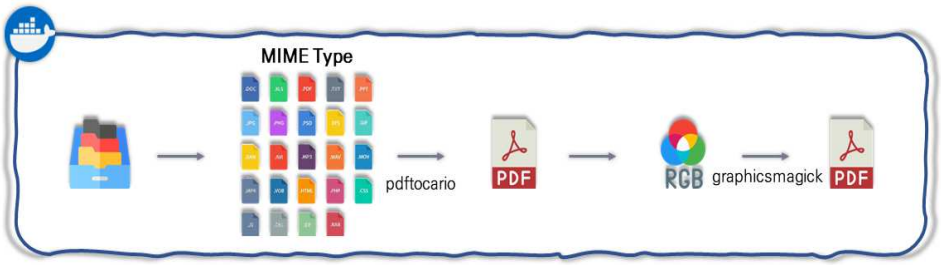


그림 33 <Docker 內 DFD>

개발한 docker에서 pdf를 생성하는 과정은 위와 같다. 처음 python의 magic lib을 이용해 파일 mime type을 파악한다. 파악된 mime type에 따라 libreofiice, pwhwp, graphicmagick lib을 사용해 pdf로 변환한다. 이후, 변환된 pdf를 페이지별로 분류해 각 페이지별로 RGB data를 추출하고 다시 pdf로 변환한다. 이 과정에서 악성코드가 사라지고 문서 내용만 남게 된다. 마지막으로 페이지별로 pdf로 변환된 파일을 하나의 pdf로 합쳐 파일 변환 과정을 마무리한다.

```
def excute_ocr(path, hash,fileName):
    print('-'*10 + 'Excute Ocr' + '-' * 10)
    file_rm = BASE_DIR+'/media/' + hash + fileName
    if os.path.isfile(file_rm):
        os.remove(file_rm)

    uploadpath = BASE_DIR + '/media/' + hash
    args = [
        "docker",
        "run",
        "--network",
        "none",
        "-v",
        f"{path}:/tmp/input_file",
        "-v",
        f"{uploadpath}:/safezone",
        "-e",
        "OCR=1",
        "-e",
        "OCR_LANGUAGE=Hangul",
        "c0natus/cap:0.0",
        "document-to-pdf.sh"
    ]
    try:
        p = subprocess.run(args, timeout=480)
    except subprocess.TimeoutExpired:
        print("Error converting document to PDF, LibreOffice timed out after 60 seconds",flush=True)
        sys.exit(1)
```

그림 34 <개발된 OCR PDF 변환 Script 1>

```python
def uploadOcr(request):
    context = {}
    print("Receive File or URL")
    file_sha="a"
    if request.method == 'POST':
        uploaded_file = request.FILES['dd_2']
        file_name = uploaded_file.name
        media_dir = 'media/ocr_files'
        file_rm = BASE_DIR + '/' + media_dir + '/' + file_name
        if os.path.isfile(file_rm):
            os.remove(file_rm)
        fs = FileSystemStorage(location=media_dir)
        name = fs.save(file_name, uploaded_file)
        file_sha='b'

        fileName = file_name.split('.')[0] + '_OCR.pdf'
        excute_ocr(BASE_DIR+'/'+media_dir+'/'+file_name, file_sha,fileName)
        old_file = os.path.join(BASE_DIR + '/media/'+str(file_sha),'safe-output.pdf')

        new_file = os.path.join(BASE_DIR + '/media/'+str(file_sha),fileName)
        os.rename(old_file,new_file)
        #context['request']=file_sha
        context['url'] = BASE_DIR + '/media/'+str(file_sha)+'/'+fileName
        context['hash'] = fileName
```

그림 35 <개발된 OCR PDF 변환 Script 2>

OCR (Searchable) PDF도 크게 다르지 않으며, 변환 과정 마지막에 tesseract lib을 통해 OCR 과정을 거쳐 텍스트로 검색 가능한 PDF 파일을 얻을 수 있다.
본 알고리즘을 수행할 때, file의 mime type을 먼저 파악하고 이후 작업을 수행한다. 지원하는 확장자는 3의(가) danger zone에 첨부한 그림과 같다. 여기에 hwp 확장자를 추가하기 위하여 hwp 파일의 mime type ‘application/x-hwp’을 추가한다. mime type이 ‘application/x-hwp’이면 해당 조건문을 실행해 hwp 파일을 html 파일로 변환 후, pdf 파일로 변환한다.

```python
def main():
    conversions = {

        #hwp
        "application/x-hwp": {"type": "pyhwp"},


        # .pdf
        "application/pdf": {"type": None},
        # .docx
        "application/vnd.openxmlformats-officedocument.wordprocessingml.document": {
            "type": "libreoffice",
```

그림 36 <hwp 확장자의 mime type 추가>

```
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)
```

그림 37 <pyhwp, wkhtmltopdf 모듈 실행>

```
def excute_dangerzone(path, hash,fileName):
    print('-'*10 + 'Excute Dangerzone' + '-' * 10)
    file_rm = BASE_DIR+'/media/' + hash + fileName
    if os.path.isfile(file_rm):
        os.remove(file_rm)

    uploadpath = BASE_DIR + '/media/' + hash
    args = [
        "docker",
        "run",
        "--network",
        "none",
        "-v",
        f"{path}:/tmp/input_file",
        "-v",
        f"{uploadpath}:/safezone",
        "c0natus/cap:0.0",
        "document-to-pdf.sh"
    ]
    try:
        p = subprocess.run(args, timeout=480)
    except subprocess.TimeoutExpired:
        print("Error converting document to PDF, LibreOffice timed out after 60 seconds",flush=True)
        sys.exit(1)

    if p.returncode != 0:
        print(f"Conversion to PDF failed: {p.stdout}",flush=True)
        sys.exit(1)
```

그림 38 <개발된 PDF 변환 Script>

2) IMG PDF 변환

IMG PDF 변환을 위해 개발된 Web page이다.

기능 수행 후 변환된 결과는 다음과 같다.

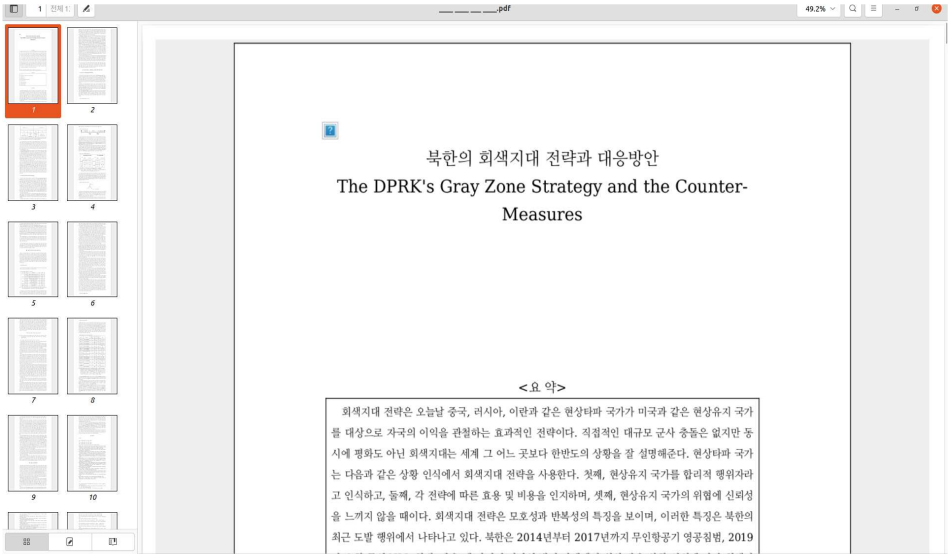


그림 40 <IMG PDF 변환 결과>

3) OCR (Searchable) PDF 변환

OCR (Searchable) PDF 변환을 위해 개발된 Web page이다.

그림 41 <개발된 OCR PDF 변환 page>

기능 수행 후 변환된 결과는 다음과 같다. 앞의 IMG PDF와 다르게 letter Search가 가능함을 확인할 수 있다.

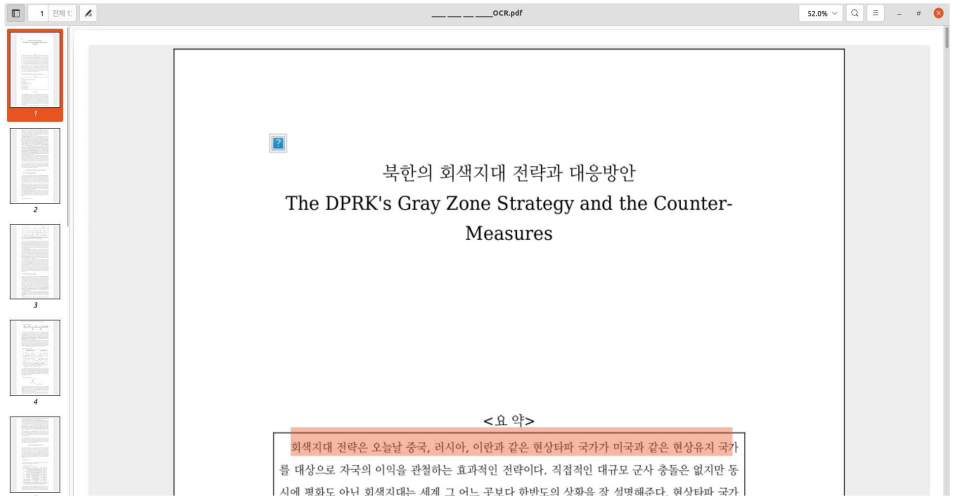


그림 42 <OCR (Searchable) PDF 변환 결과>

4) 변환된 PDF 감염 여부 확인

문서에 첨부되어 있던 malware가 제대로 제거되었는지 확인하기 위해 virus Total의 file 검사 기능을 사용하였다.

가) IMG PDF

IMG PDF로 변환한 파일의 검사 결과이다.

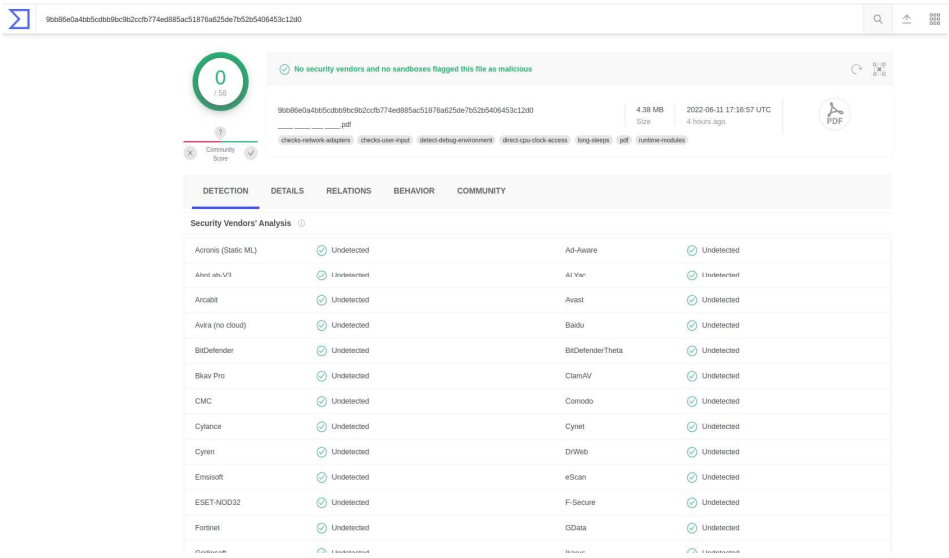


그림 43 <IMG PDF 검사 결과>

5) OCR (Searchable) PDF

OCR (Searchable) PDF로 변환한 파일의 검사 결과이다.

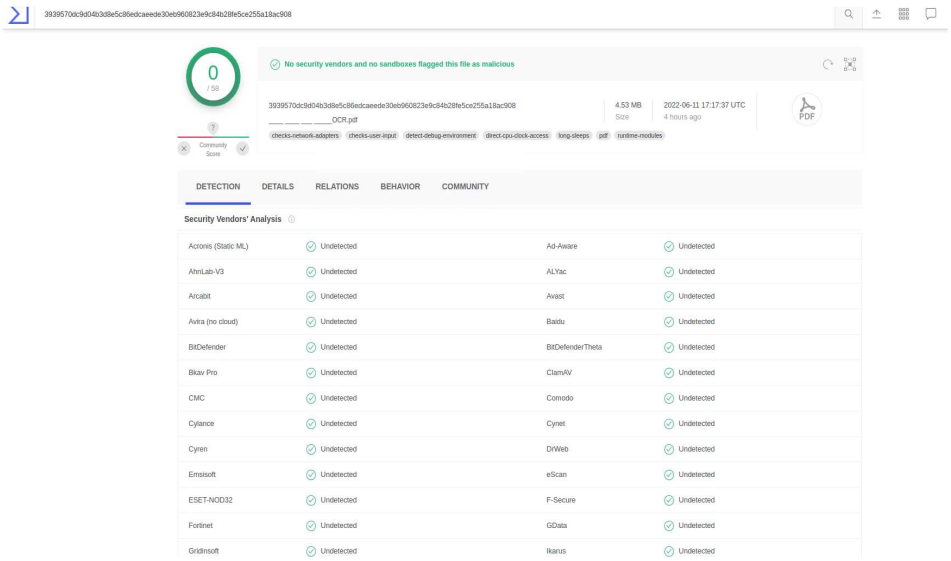


그림 44 < OCR (Searchable) PDF 검사 결과>

라. Adblock 차단 기능

1) 알고리즘

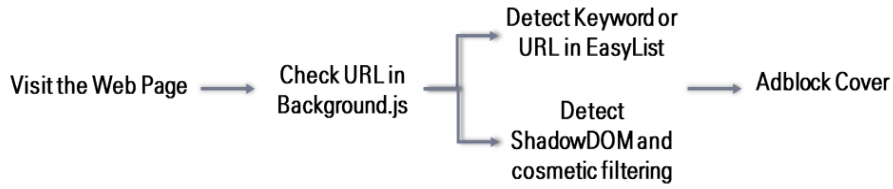


그림 45 <Adblock DFD>

JavaScript와 Python을 이용하여 Chrome extention 형태로 개발하였으며, 사용자가 사이트에 접속 시, Easy List API 기반으로 광고 배너를 차단한다. Webpage Request를 기준으로 하여 해당 URL이 Easy List API에 존재하거나, 광고와 관련된 특정 키워드를 포함하는 경우 해당 배너가 표시되지 않도록 한다.

Chrome Extension API의 Web Request는 사용자가 사이트에 접속하면 전달되는 웹 요청을 기준으로 하여 해당 페이지가 실행되기 전, URL을 받아 분석하여 광고 URL 여부를 판단한다.

아래 코드에서 Easy List API를 호출하여 광고 배너에 삽입되는 특정 키워드와 스크립트 URL 약 85,000개에 해당하면 해당 URL에 접속하지 못하도록 redirect 한다.

```
background.js > ...

chrome.webRequest.onBeforeRequest.addListener(

    function(details) {
        if(!enabled){
            return { cancel: false };
        }

        console.log("I am going to block:", details.url)
        return {cancel: true};
    },
    {urls: blocked_sites},
    ["blocking"]


)

// css 기반 광고 차단
chrome.runtime.onMessage.addListener((b,a)=>
{chrome.tabs.insertCSS(a.tab.id,
    {code:":is(img,line):not(:root *):not([*="+b+"]):not([*="+b+"] *):not([*="+b+"] ~ * *){clip-path:polygon(0% 0%,0% 100%,0.00001% 100%,0.00001% 0.00001%,99.99
    cssOrigin:"user",frameId:a.frameId,runAt:"document_start"})});
```

그림 46 <Adblock main Algorithm>

2) 기존 Site

뉴스를 보기 위해 접속한 Site에 상단과 좌우 광고 배너들이 많다.

그림 47 <Adblock 적용 전 : 뉴스 Site 광고 배너 모습>

3) Adblock 시연

괘적한 Site 이용을 위하여 광고 배너를 제거한 모습이다.

그림 48 <Adblock 적용 후 : 뉴스 Site 광고 배너가 사라진 모습>

4) 차별성

Web site 운영 측에서 광고 차단 솔루션에 대응하여, 광고를 차단하는 사용자를 감지하고 사이트 이용을 가로막거나, 고의로 사이트 기능을 정상적으로 수행할 수 없도록 하는 등 광고 배너를 차단한 사용자를 차단하는 Anti Adblock Web Site가 등장했을 뿐 아니라, 광고 차단 솔루션이 광고 배너를 차단할 수 없도록 URL Obfuscation(난독화), DOM obfuscation 등을 수행해 광고블럭을 삽입하는 ad-reinsertion(광고 차단 우회)을 적용하는 매체가 증가하여 위의 개발물이 특정 사이트에서 기능 수행을 못 하는 문제를 발견하였다.

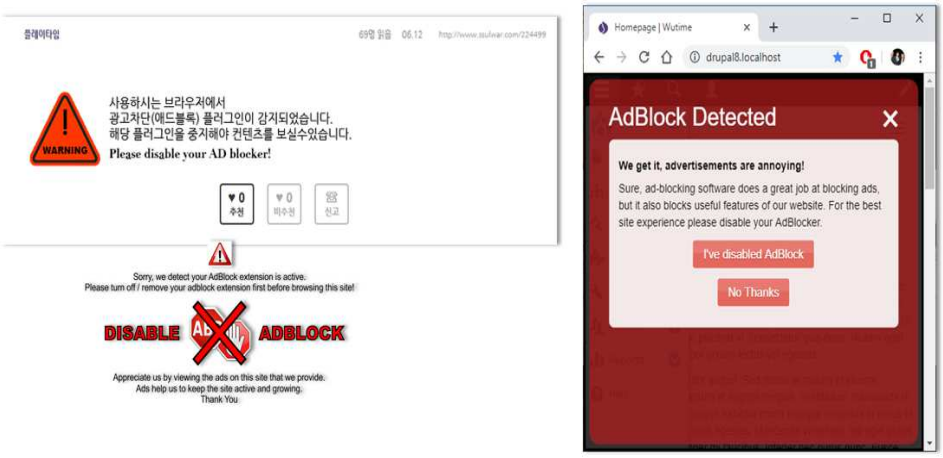


그림 49 <Adblock 차단 프로그램으로 이용이 불가한 사례>

위에서 설명한 바와 같이 일반적인 광고 차단 프로그램으로 차단이 어려운 악성 광고를 사용자에게 보이지 않도록 하면서 웹 사이트에서 차단을 감지할 수 없도록 추가 구현함. 악성 광고에서 사용하는 기술 "Shadow DOM"을 차단할 수 있고, cosmetic filtering CSS 기반 차단이 가능하다.

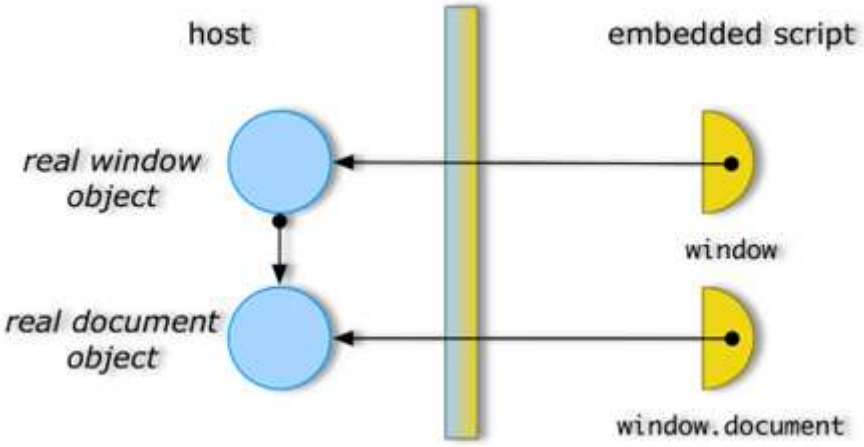


그림 50 <Membrane Pattern Model>

Membrane 패턴을 이용하여 광고 차단 여부 감지를 수행하는 browser API에 hook을 설치하여 웹 사이트에서 사용자의 현재 페이지 상태를 API로 유추할 때 광고가 차단되지 않았다는 결과를 거짓으로 리턴하여 antiadblock이 불가능하게 하였다.
또한, Shadow DOM-bypassing을 적용하여, ad-reinsertion에서 자주 사용되는 Shadow DOM을 이용한 DOM 난독화가 적용된 광고 요소도 정확하게 타겟할 수 있도록 하였다.

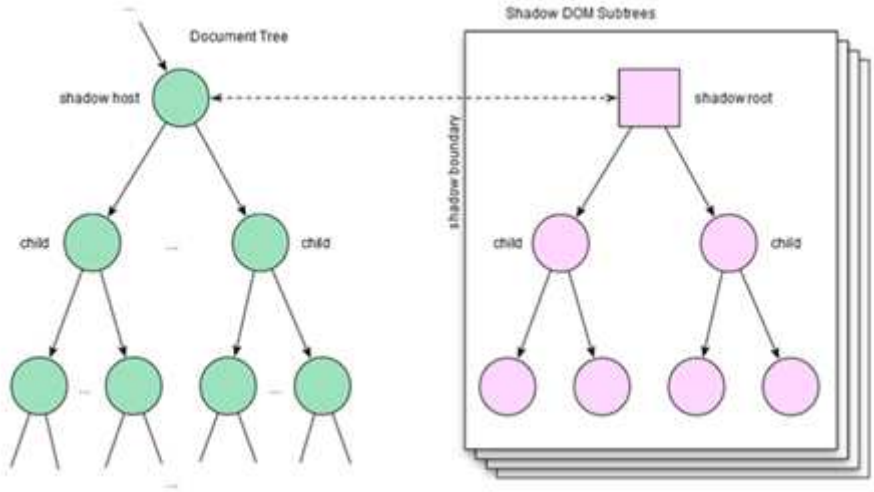


그림 51 <左 Basic DOM 右 Shadow DOM>

위와 같은 구현으로 차단할 수 없던 광고 배너들도 차단할 수 있으며, 간단한 브라우저 확장프로그램 레벨에서 anti Adblock을 불가능하게, 웹 사이트가 광고 차단 여부를 알 수 없도록 하였다.

5) **구현 전 차단 불가 광고 배너**

아래의 두 사이트는 antiadblock과 ad-reinsertion을 적용한 것으로 확인되는 사이트로 많은 사용자가 광고로 불편을 호소하였다.

그림 52 <ppss.kr>

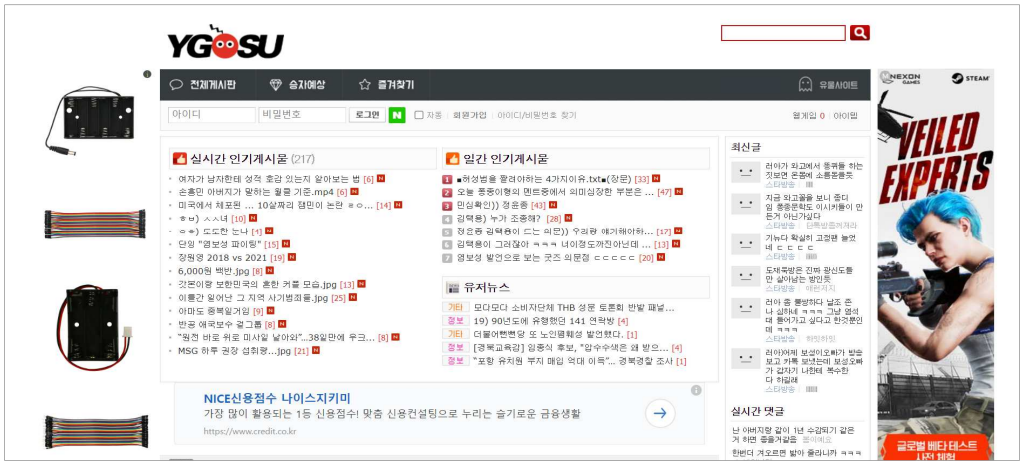


그림 53 <ygosu.com>

6) 추가 구현 후 차단 된 모습

모두 Adblock 적용 전 차단되지 않아 사이트 이용에 불편을 주었던 광고 배너들이 가려진 것을 볼 수 있다.

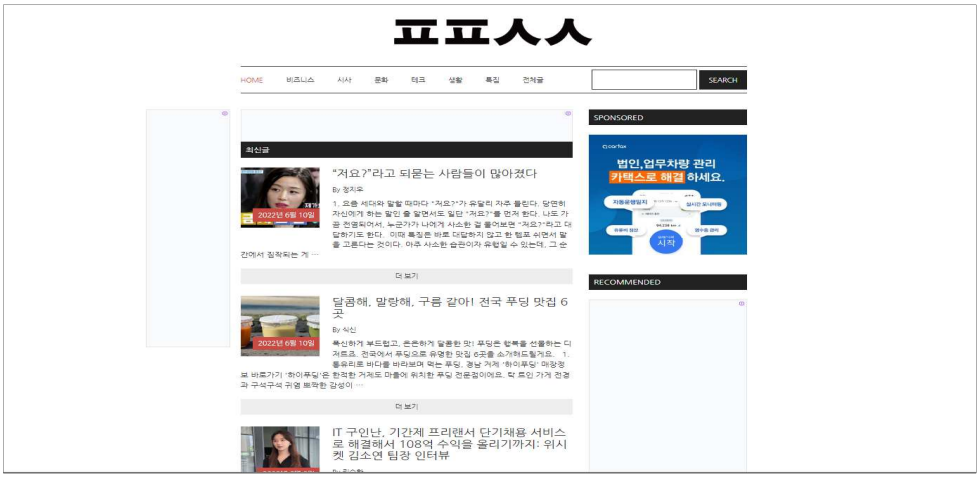


그림 54 <Adblock 사용 후 ppss.kr>

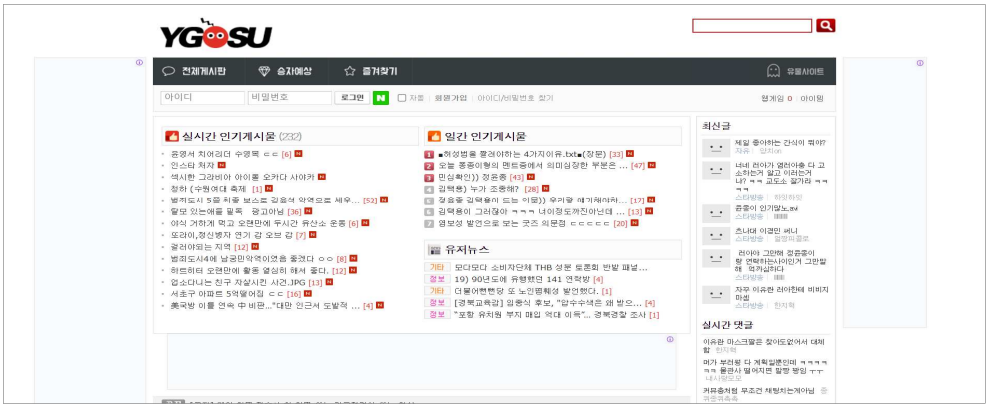


그림 55 <Adblock 사용 후 ygosu.com>

7. 기대효과

인포스틸러 악성코드 Formbook, AgentTesla, Lokibot 1~3위 차지
안랩 ASEC 분석팀에서 집계한 6월 첫째 주 악성코드 통계 분석 결과

[보안뉴스 권 준 기자] 2022년 6월 첫째 주에는 개인정보 및 계정정보 탈취 목적의 인포스틸러 악성코드가 국내에 유포된 전체 악성코드의 대부분을 차지한 것으로 분석됐다.

보안전문 업체 안랩의 ASEC 분석팀이 ASEC 자동 분석 시스템 'RAPIT'를 활용해 2022년 5월 30일부터 6월 5일까지 한 주간에 수집된 악성코드 통계에서 대분류 상으로는 인포스틸러가 89.9%로 1위를 차지했다. 그 다음으로는 RAT(Remote Administration Tool) 악성코드가 8.5%, 랜섬웨어, 다운로더, 뱅킹 악성코드가 각각 0.5%로 집계됐다.

그림 50 <보안뉴스 최근 악성코드 유형>

본 프로젝트로 인한 기대효과는 3가지 측면에서 생각해 볼 수 있다. 먼저 기술적 측면에서 악성코드를 지닌 문서 내부 데이터를 확인할 수 있게 됨과 동시에 HWP, MS Office 등 다양한 확장자 변환 기능을 제공한다는 점, 악성 사이트 차단 및 사이트(URL) 위험도 확인 기능과 파일 감염 여부 확인 기능을 제공하는 장점이 있다. 경제적 측면으로는 Chrome Extension 방식을 사용한 광고 배너 차단 기능은 사용자에게 높은 접근성과 편의성을 제공하여 많은 사용자를 인터넷상의 유해 광고들로부터 보호할 수 있다. 무엇보다 오픈 소스로 사용자의 편의에 따라 변경할 수 있다. 마지막으로 사회적 측면에서 APT 공격 예방이 가능하다.

8. 개발 일정 및 팀원별 담당 업무

개발 일정은 아래와 같이 진행함.

이채은	서아름
1. 프로젝트 총괄 및 발표 자료·보고서 준비 2. Web Base 개발 3. URL 스캔 기능 구현 4. 광고 배너 Chrome Extension 개발 5. 광고 배너 기능 구현 6. 기존 Adblock 개발 가. Membrane pattern 구현 나. Shaw DOM bypass 구현 7. Dangerzone Docker 환경 구축	1. 발표 자료 준비 및 발표 2. 광고 배너 기능 구현 가. Membrane pattern 적용 나. Shaw DOM bypass 적용 3. Dangerzone 서버 업로드 구현 4. Web UI 개발
이은서	**송승민**
1. 발표 자료·보고서 준비 2. Django 환경 구축 3. File 스캔 기능 구현 4. 광고 배너 Chrome Extension UI 개발 5. Dangerzone 서버 업로드 구현 6. Web UI 개발	1. Dangerzone 서버 업로드 구현 2. Dangerzone 서버 다운로드 구현

9. 지원금 집행 내역

웹 개발 결과물로, 추가 하드웨어나 외형 등 지원금이 불필요하였음. 회의비로 1건 지출함.

10. 프로젝트 수행 후기

- 이채은

학부 생활을 하는 4년 동안 배운 전공지식을 활용해 프로젝트의 개발 프로세스를 경험할 수 있는 본 과목을 통해 많은 것을 배웠다. 한 학기 동안 프로젝트를 수행하면서 사용해 보지 않았던 여러 기술과 플랫폼을 사용하게 되면서 많은 어려움과 개발 이슈가 있었다. 결과적으로 학기 초 계획한 모든 기능 구현을 완벽하게 한 점은 만족스럽다. 그러나, 프로젝트 수행 기간 동안 일부 몇몇이 개발 작업이나 프로젝트 관련 발표 준비나 서류작성 등 업무 및 개발을 본인 몫 이상으로 많이 하게 된 점이 아쉽다.
마지막으로 프로젝트 수행 중간중간 개발물에 대한 피드백으로 올바른 방향으로 나아가게 해주신 최원석 교수님께 감사드립니다.

- 이은서

처음 프로젝트를 기획했을 때 계획했던 결과물을 잘 구현해낼 수 있을지 걱정이 되었지만, 팀원들과의 협업을 통해 만족스러운 결과물이 나온 거 같아서 뜻깊은 시간이 된 거 같습니다. 나중에도 기회가 된다면 다른 프로젝트를 수행해보고 싶습니다.

- 서아름

이번 프로젝트를 진행하면서 이전 수업과는 다른 경험을 할 수 있었고, 앞으로 기회가 된다면 보안과 관련된 또 다른 프로젝트를 진행해보고 싶습니다.

- 송승민

프로젝트를 진행하면서 부족했던 부분들을 알 수 있었고 앞으로 이러한 부분들을 보완하여 또 다른 프로젝트를 진행해보고 싶습니다.

<부록> 프로그램 코드

https://github.com/chaeuny/Malicious-Threat-Detection-System.git

```
2022_SECURE_ Capstone
├─── dangerzone //댄저존 파일 변환하는 폴더
|          ├─── build_context
|               ├──  script
|               |      ├─── document-to-pdf.sh // 아래 두 py 파일 실행할
|               |                                  때 사용되는 쉘 스크립트
|               |      ├─── document-to-pixels.py //픽셀 단위로 변환해주는 파일
|               |       |        ├─── print_flush // 버퍼에 있는 모든 것을 터미널
|               |       |        |       에 기록
|               |       |        └─── main //문서를 픽셀 단위로 변환해주는 함수
|               |      └─── pixels-to-pdf.py //pdf로 변환해주는 파일
|               |                ├─── print_flush // 버퍼에 있는 모든 것을 터미널
|               |                |                       에 기록하는 함수
|               |                └─── main //픽셀을 pdf로 변환해주는 함수
|               └─── Dockerfile //Dockerfile 사용한 서버구축 및 구성관리 방법
 ├─── maldetect // 프로젝트 폴더
|         ├── __init__.py //디렉토리를 패키지처럼 다루도록 알려주는 빈 파일
|         ├── asgi.py // 비동기와 관련된 파일
|         ├── settings.py //프로젝트 전반에 걸친 각종 설정을 위한 파일
|         ├─── urls.py //웹 페이지를 어떤 주소에 연결 시킬지 정의한 파일
|         └─── wsgi.py // 웹 사이트 실행 프로세스와 관련하여 사용된 파일
 ├─── templates //필요한 html파일을 모아둔 폴더
|          ├─── adblock.html //광고 배너 설명 및 다운로드 하는 파일
|          ├─── id.html //객체 출력을 하기위한 파일
|          ├─── index.html //localhost:8000 접속 시 보이는 화면을 구성 파일
|          ├─── upload_f.html //댄저존 실행 가능한 웹 페이지 구성 파일
|          ├─── upload.html //파일 검사를 하는 웹 페이지 구성 파일
|          ├─── uploadOcr.html //ocr 변환 가능한 웹 페이지 구성 파일
|          ├─── url_upload.html //URL 검사를 하는 웹 페이지 구성 파일
|          └─── urlerror.html //URL형태가 아닌 다른 형태 입력 시 오류 출력 웹
|                                  페이지 구성 파일
├─── webapp //웹 페이지를 띄우기 위한 모듈을 구성한 장고 앱 폴더
|         ├─── __pycache__ //캐쉬파일
```

```
|       ├── migrations //db 파일을 보관하는 디렉터리
|       ├── __init__.py //디렉토리를 패키지처럼 다루도록 알려주는 빈 파일
|       ├── admin.py //장고 관리자 웹 구성에 필요한 파일
|       ├── apps.py //장고 웹앱에 대한 설정을 위한 파일
|       ├── forms.py //폼 관리 파일
|       ├── models.py //db에 데이터 모델 생성을 위한 파일
|       ├── tests.py //단위 테스트 실행 가능한 파일
|       ├── urls.py //웹 페이지를 어떤 주소에 연결 시킬지 정의한 파일
|       └── views.py //웹 페이지나 웹 요청 등 처리하는 코드를 작성한 파일
|               ├── error_500 //500에러 발생 관련 함수
|               ├── error_404 //404에러 발생 관련 함수
|               ├── is_valid_url //타당한 URL 형태인지 검사                    |
|               ├── malurl_form //index.html로 리턴해주는 함수
|               ├── url_upload //바이러스 토탈 api를 이용하여 URL 검사하는 함수
|               ├── upload //바이러스 토탈 api를 이용하여 파일 검사하는 함수
|               ├── uploadFile //댄저존 실행된 파일 받는 함수
|               ├── file_upload //댄저존 실행 시 파일 검사하는 함수
|               ├── file_report //바이러스 토탈 api 분석을 위한 함수
|               ├── vtchart //바이러스 토탈 api를 이용해 검사한 결과 출력하는 함수
|               ├── excute_dangerzone //댄저존 실행 함수
|               ├── adblock //광고 배너 웹 페이지로 리턴해주는 함수
|               ├── excute_ocr //ocr 실행 함수
|               └── uploadOcr //ocr로 변환된 파일 받는 함수
 └── manage.py
       └── main //Django 프로젝트와 상호작용하는 커맨드라인 유틸리티
```

Block AD Banners

```
├── background.js
|       ├── chrome.webRequest.onBeforeRequest.addListener//easylist기반
|                                                          광고차단
|       ├── chrome.runtime.onMessage.addListener//css 기반 광고차단
├── blockedsites.js //easylist API
├── domaingsblockraw.txt // easylist API
├── getcurrentURL.js
|       ├── getCurrentTabURL //접속한 페이지 url callback
|       ├── renderURL //광고 차단 후 page reload
|       ├── document.addEventListener // DOMContentLoaded callback
```

```
├─── inject.js
|        ├─── pageInfo // 접속한 page의 정보 load
├─── menifest.json //확장프로그램의 기본적인 정보와 기능을 정의
├─── onoff.js //광고배너 on, off 버튼 처리
|        ├─── onload
├─── parsedomains.py
└─── popup.html // 확장프로그램 화면
```

참고문헌

[1] 보안뉴스, https://m.boannews.com/html/detail.html? tab_type=1&idx=107419
[2] 지란지교 시큐리티, https://www.jiransecurity.com/products/sanitox
[3] EQST insight, https://m.blog.naver.com/sk_shieldus/222678742084
[4] 보안뉴스, https://www.boannews.com/media/view.asp?idx=100453&kind=
[5]MSoffice,https://support.microsoft.com/ko-kr/office/%ed%8c%8c%ec%9d%bc%ec%97%90%ec%84%9c-office-%eb%a7%a4%ed%81%ac%eb%a1%9c-12b036fd-d140-4e74-b45e-16fed1a7e5c6?ui=ko-kr&rs=ko-kr&ad=kr
[6] EQST insight, https://m.blog.naver.com/sk_shieldus/222678742084
[7] Tera Information Security, http://terais.co.kr/nurilab-cdr/
[8] JIRANFAMILY, https://www.jiran.com/main/index
[9] Dangerzone github, https://github.com/firstlookmedia/dangerzone-converter
[10] Virus Total API, https://www.virustotal.com/
[11] Chrome extension API, https://developer.chrome.com/docs/extensions/reference