문서형 악성코드 탐지를 위한 HWP 포맷 취약점 분석*

최민지○ 신강식 정동재
KAIST 사이버보안연구센터
chlalswl0928@kaist.ac.kr, ksshin90@kaist.ac.kr, jjp1018@kaist.ac.kr

HWP Format Vulnerability Analysis For Document-Type Malware Detection

MinJi Choe○, KangSik Shin, DongJae Jung
KAIST Cyber Security Research Center

요 약

최근 지능화된 악성코드를 이용한 사이버 공격이 지속적으로 발생되고 있다. 2010년 이후부터 한글 문서를 이용한 공격이 본격적으로 등장하였고 최근까지도 다양한 제목 및 내용을 이용한 문서로 인해 피해가 발생되고 있다. 한글 문서를 이용한 공격은 특정 기관이나 사용자에게 사회공학 기법인 스피어피싱(Spear Phishing)을 이용하여 이메일의 첨부파일에 악성코드를 삽입하여 유포되고 있다. 이러한 문서를 이용한 악성코드 공격을 탐지하고 방어하기 위해 문서형 악성코드에 대해 분석해보고자 한다. 문서를 이용한 악성코드는 실행 파일 구조인 일반 악성코드와 다르게 악성 스크립트 및 악성코드가 오브젝트에 삽입되어 존재하기 때문에 문서의 포맷에 대한 이해가 필요하다. 본 논문에서는 한글 문서 포맷 및 악성행위를 식별할 수 있는 오브젝트를 바탕으로 한글 문서에서 사용되는 취약점 및 공격 방식에 대해 조사하여 분류하였다.

1. 서 론

최근 고도화 및 지능화 악성코드를 이용한 사이버 공격이 발생하고 있으며, 그 중 공격 대상은 불특정 다수가 아닌 특정 대상을 타겟으로 하는 스피어피싱(Spear Phishing) 기법을 이용하여 이메일에 악성코드를 첨부한 문서형 악성코드를 유포하고 있다. 문서형 악성코드는 사회적 이슈 또는 일반인이 관심을 둘만한 제목 및 내용으로 이메일을 통해 유포되고 있으며, 문서 파일을 클릭함과 동시에 사용자 자신도 모르게 PC에 악성코드가 드롭되어 실행된다. 이전에는 공격자들이 이메일에 윈도우 운영체제에서 사용 가능한 실행 파일 포맷인 PE(Portable Executable) 파일을 첨부하였으나 최근에는 문서 파일의 정상 기능 및 취약점을 이용하는 공격이 증가하고 있는 추세이다.

지난 2010년 이후로 한글 문서를 이용한 공격이 본격화 되었고, 2017년부터 최근까지 그래픽 파일 포맷으로 각종 고화질 벡터 이미지를 표현하기 위한 EPS (Encapsulated PostScript)파일 처리와 관련된 동일한 취약점을 노린 공격이 계속해서 발생되고 있다. 이에 따라 한글 문서를 악용한 문서형 악성코드를 탐지하고 분석하기 위해서는 문서 파일의 포맷을 이해하는 것이 필요하다. 본 논문에서는 국내를 타겟으로 하는 한글(HWP) 문서 포맷 분석을 통해 악성행위를 식별하여 지금까지 알려진 취약점 및 공격에 식별된 악성행위들이 어떤 방식으로 이용되는지 조사하고 분류하였다.

2. 문서형 악성코드 특징

일반적인 악성코드는 실행 가능한 PE(Portable Executable) 파일 포맷 구조를 지닌다. PE 파일이란 윈도우 운영체제에서 사용 가능한 실행 파일, DLL, 객체 코드, 폰트 파일 등을 위한 파일 형식이다. 하지만, 문서형 악성코드는 문서가 실행되는 과정에서 응용 프로그램을 통해 실행되기 때문에 PE 파일 포맷 구조인 일반 악성코드와 달라서 가장 먼저 문서 종류를 식별하는 과정이 선행되어야 한다. 문서형 악성코드는 실제 악성 행위를 수행하는 코드가 압축 및 난독화되어 오브젝트에 존재하기 때문에 기존 보안 프로그램에 탐지되지 않을 가능성이 높다는 특징이 있다. 따라서 일반 악성코드 분석에 사용되는 도구로 문서 파일에 오브젝트로 존재하는 악성코드를 추출하여 분석하기에는 한계점이 존재한다.

이러한 특징으로 인해 문서형 악성코드를 분석하기 위해서는 분석하고자 하는 문서의 포맷 구조를 이해해야 한다. 문서 포맷을 분석해 주는 도구를 통해 어느 스트림 오브젝트에 어떠한 형태로 악성코가 삽입되었는지 파일 구조를 확인해야 한다. 이후 악성코드, 악성 쉘코드 및 스크립트 코드 등의 압축 여부를 확인하고 압축이 되어있을 경우 압축 해제 후 코드 등을 추출하여 분석을 진행할 수 있다.

2.1 HWP 포맷

문서를 이용한 악성코드는 응용 프로그램을 통해 실행되는 파일로 실행 파일 자체가 아니기 때문에 백신 프로그램을 쉽게 우회할 수 있다. 이에 따라 문서형 악성코드를 탐지하기 위해서는 문서의 포맷 구조를 분석해야만 한다. 본 논문에서 설명하고자 하는 HWP 문서는 대한민국의 공공기관 및 기업, 일반 사용자들이 주로 사용하는 문서 작성 소프트웨어로 자바스크립트를 사용한 매크로 작성 및 편집, EPS 파일 삽입 및 보기, 객체 연결 삽입(OLE), 배포용 문서와

* 본 논문은 과학기술정보통신부 글로벌사이버보안기술연구(과제고유번호: 1711125408) 사업의 지원을 받아 수행된 연구임.

같은 다양한 기능을 제공하는데 악성코드는 이를 악용한다. HWP 문서는 스토리지(Storage)와 스트림(Stream)으로 구성된 복합 파일(Compound File) 구조로 표 1과 같이 FileHeader, DocInfo, BodyText, Summary Information, BinData, PrvText, PrvImage, DocOptions, Scripts, XML Template, DocHistory 스토리지로 구성되며 하나의 스트림에는 바이너리 및 데이터가 저장된다. 여기서 설명하는 스토리지는 폴더, 스트림은 파일의 관계와 유사하며, 한글 문서의 미사용 영역에는 비정상적인 바이너리 데이터 및 스트림을 삽입할 수 있다.

표 1. 한글 파일 포맷 구조

설명	구별 이름	압축/암호화
파일 인식 정보	FileHeader	
문서 정보	DocInfo	가능
본문	BodyText	가능
문서 요약	HwpSummaryInformation	
바이너리 데이터	BinData	가능
미리보기 텍스트	PrvText	
미리보기 이미지	PrvImage	
문서 옵션	DocOptions	
스크립트	Scripts	
XML 템플릿	XMLTemplate	
문서 이력 관리	DocHistory	가능

한글 포맷은 Root Entry를 참조하여 데이터를 얻고 각 스토리지 포맷의 특성에 따라 레코드 구조 여부나 압축 여부가 결정되며, zlib 라이브러리를 사용하여 문서 파일을 압축한다. 한글 문서임을 증명하는 FileHeader 스트림의 압축 플래그를 통해 압축 여부를 확인할 수 있고 압축 파일인 경우 압축을 해제해야 한다. DocInfo 스트림에는 문서에 사용되는 세부 정보(글꼴, 속성, 탭 등)들이 저장되고, DocOptions 스트림에는 연결 문서, 배포용 문서, 전자 서명 관련 정보들이, DocHistory 스트림에는 문서 이력 정보 데이터가 저장된다. 미리보기 텍스트인 PrvText 스트림에는 텍스트가 유니코드 문자열로 저장되고, 미리보기 이미지인 PrvImage 스트림에는 이미지들이 BMP 및 GIF 형식으로 저장되는 포맷을 사용한다.

2.2 악성 오브젝트 식별

한글 문서를 탐지하고 분석하기 위해서는 한글 문서의 어떤 스토리지에 악성코드 및 악성 스크립트 등이 삽입될 수 있는지 식별해야 한다. 한글 문서에는 바이너리 데이터들이 저장되는 'BinData' 라는 스토리지가 존재하며, BinData 스토리지 하위에는 EPS 파일 및 OLE 개체, 포스트스크립트 등의 스트림들이 저장되는 위치이다. BinData에서는 저장된 스트림 중 비정상적인 스트림을 확인할 수 있다. 즉, 공격자가 악의적인 의도로 삽입한 악성 EPS 파일 및 OLE 개체, 포스트스크립트 등의 스트림들이 zlib으로 압축되어 *.EPS, *.OLE, *.PS 형식으로 존재한다.

한글 문서는 자바 스크립트를 지원하며 'Scripts' 스토리지에는 스크립트 코드(JScriptVersion, DefaultJScript)들이 저장되는 위치이다. 'JScriptVersion' 스트림에는 스크립트 버전이 저장되고 'DefaultJScript' 스트림에는 스크립트 헤더, Pre 및 Post 소스 등이 저장된다. 두 개의 스트림은 모두 압축 및 암호화를 지원하지 않으며 'Scripts' 스토리지에는 비정상적인 자바스크립트 코드가 삽입될 수 있다.

표 2. 악성 오브젝트 분류

설명	구별 이름
바이너리 데이터 (BinData)	BinData - BinaryData0 - BinaryData1 - ...
스크립트 (Scripts)	Scripts - JScriptVersion - DefaultJScript - ...
본문 (BodyText)	BodyText - Section0 - Section1 - ...

한글 문서의 본문에서 사용되는 문단, 표, 그리기 개체 등의 내용들은 'BodyText' 스토리지에 저장된다. BodyText 스토리지는 본문의 구역에 따라 표1과 같이 'Section[숫자]' 로 구분되며 숫자는 각 구역의 번호를 의미한다. Section 부분의 문단 텍스트를 의미하는 'HWPTAG_PARA_TEXT' 태그에 비정상적인 값이 삽입될 수 있으며 이를 토대로 실제 악의적인 행위를 수행하는 쉘코드가 삽입된 것으로 추측할 수 있다. 결론적으로 한글 문서의 BinData, Scripts, BodyText 스토리지에 악성코드 및 악성 스크립트가 삽입될 수 있으며, 공격자는 실질적인 악성 행위를 수행하는 스크립트 코드 등을 XOR 인코딩 및 난독화 등의 방식을 사용하여 코드가 노출되지 않도록 한다.

3. HWP 포맷 취약점 분석

앞에서 한글 문서에서 악성행위가 존재할 수 있는 오브젝트들에 대해 살펴보았으며, 3장에서는 식별한 오브젝트를 바탕으로 어떠한 취약점 및 방법으로 공격에 이용되는지 분류한다. 공격자는 표 3과 같은 매크로(JavaScript, VBA 등), EPS 파일, 객체 연결 삽입(OLE), 배포용 문서들을 활용하여 공격한다.

첫 번째 방법은 자바스크립트를 이용한 공격이다. 한글 문서에서는 자바스크립트(JavaScript) 언어를 사용하여 매크로를 작성하거나 편집할 수 있다. 과거에는 매크로 기능을 악용한 악성 한글 문서가 자주 발생하였으며, 매크로를 이용한 공격 방법은 다른 악성코드를 다운받을 수 있는 웹 주소 또는 실제로 사용자 PC에 감염시킬 실행 파일 등을 매크로 코드 내에 임베디드되어 동작된다. 해당 공격은 과거에 자주 쓰이던 방식으로 최근에는 자주 사용되고 있지 않다.

두 번째 방법은 EPS를 이용한 공격으로 한글 문서에서는 각종 고화질 벡터 이미지를 표현할 수 있는 EPS 파일을 삽입하거나 볼 수 있는 기능을 제공한다. EPS 포스트스크립트(PostScript) 언어를 이용하여 그래픽 이미지를 표현한 파일로 사용자 눈에 보이지 않을 정도의 아주 작은 크기로 삽입되어 존재한다. 한글 문서는 EPS 파일을 화면에 표현하기

위해 고스트스크립트(Ghostscript) 인터프리터를 사용하며, 고스트스크립트(GS, 9.21 및 이하버전)에 의해 EPS 파일이 처리되는 과정에서 악성행위가 삽입된 페이로드로 인해 악성

```
/ar <4b080506010855445552472222222244000102444b08
def /limit {ar length -1 add}
def /len {ar length}
def /str len string
def ar 0 1 limit {
    2 copy get 100 xor put ar
}for
pop str 0 1 limit {
    dup ar exch get put str
}for cvx exec exec
```

그림 1. 악성 EPS 내부 코드 압축 해제

코드가 실행된다. 그림 1은 압축 해제 후 확인한 내부 코드로 꺽쇄 안에 존재하는 문자열은 쉘코드로 XOR 연산을 사용하는 것을 확인할 수 있으며, 해당 포스트스크립트를 실행하면 복호화키를 통해 XOR 연산으로 암호화된 페이도르를 복호화하여 실행된다.

세 번째 방법은 객체 연결 삽입을 이용한 공격으로 OLE(Object Linking and Embedding)란 MS에서 개발한 기술로 문서와 기타 개체에 대한 연결과 삽입을 도와주는 연결규약이다. OLE 개체를 이용한 악성코드의 경우 별다른 특이점이 눈에 나타나지는 않는다. 투명한 OLE 개체가 문서 전체 영역으로 설정될 수 있으며, 사용자가 문서를 편집하기 위해 정상적인 OLE 이미지 파일로 위장한 묶음 개체를 직접 클릭하면 OLE 객체에 삽입된 악성코드가 실행된다. 이러한 문제점을 예방하기 위해 최신 버전의 한글 문서는 기본적으로 문서 열람시 연결 문서의 링크 실행을 방지하기 위해 보안 위험 안내 메시지를 출력하여 한글 문서의 보안을 유지하고 있다.

또 다른 방법으로는 배포용 문서가 있으며 배포용 문서는 일반 한글 문서와 다르게 수정 및 편집이 불가능한 문서로 주로 공공기관에서 공문을 보낼 때 사용하며 공격자는 이러한 특징을 이용하여 마치 기관에서 제작한 문서인 것처럼 위장하여 유포한다. 배포용 문서는 'ViewText' 스토리지 하위에 본문 스트림을 암호화하여 저장하며 'ViewText/Section', 'Scripts', 'DocHistory' 영역을 자체적인 암호화 기능을 통해 암호화할 수 있다. 공격자는 암호화 기능을 악용하여 배포용 문서에 악성 쉘코드를 삽입할 수 있으며, 복호화 및 zlib 디컴프레스 과정을 통해 악성행위를 수행하는 쉘코드가 동작하게 된다.

4. 결론

최근 문서를 이용한 악성코드의 유포가 증가하고 있으며, 문서형 악성코드는 문서의 취약점을 이용하여 코드를 삽입하기 때문에 백신 프로그램에서 탐지되지 않고 우회하기 쉬워 공격자들이 많이 사용하고 있다. 본 논문에서는 악성코드 중 특정 타겟을 목표로 한 스피어피싱(Spear Phishing)을 통해 유포되는 한글 문서형 악성코드를 대상으로 한글 파일 포맷 및 악성행위를 식별할 수 있는 오브젝트 뿐만 아니라 한글 포맷 취약점에 대해 알아보았다. 과거에는 문서 내 실행 파일을 삽입하는 방법과 자바스크립트를 이용한 공격이 가장 많았지만 최근 문서 파일의 취약점 및 한글 문서의 정상 기능을 악용하는 추세로 변하고 있으며, EPS 파일을 악용한 공격이 꾸준히 발생되고 있고 2020년 하반기에 들어서면서 OLE 기능을 활용하여 추가 명령을 실행하는 방식으로 공격이 다변화되고 있다.

향후, 한글 문서의 포맷 구조를 바탕으로 악성 행위를 수행할 수 있는 스트림 오브젝트 추출 및 분석 단계의 효율성을 높이고자 자동화 도구를 개발할 예정이다.

5. 참고 문헌

[1] Financial Security Institute(금융보안원), "한글 문서를 이용하는 악성코드 프로파일링", 2018 사이버 위협 인텔리전스 보고서, 2018.

[2] Jeong, Y. S., Woo, J., & Kang, A. R., "Malware Detection on Byte Streams of Hangul Word Processor Files", Applied Sciences, 9(23), 5178, 2019.

[3] Kang, A. R., Jeong, Y. S., Kim, S. L., Kim, J., Woo, J., & Choi, S., "Detection of malicious pdf based on document structure features and stream objects". Journal of The Korea Society of Computer and Information, 23(11), 85-93, 2018.

[4] 윤채은, 정혜현, & 서창진, "딥러닝과 PDF 객체분석을 이용한 문서형 악성코드 탐지", 전기학회논문지 P, 70(1), 44-49, 2021.

[5] KISA, "2020년 2분기 사이버 위협 동향 보고서", 2020, Jul.

[6] Financial Security Institute(금융보안원), "코로나19 금융부문 사이버 위협동향", 2020, May.