Journal of The Korea Society of Computer and Information
Vol. 23 No. 11, pp. 85-93, November 2018

www.ksci.re.kr
https://doi.org/10.9708/jksci.2018.23.11.085

# Detection of Malicious PDF based on Document Structure Features and Stream Objects

Ah Reum Kang*, Young-Seob Jeong*, Se Lyeong Kim**, Jonghyun Kim***, Jiyoung Woo*, Sunoh Choi***

## Abstract

In recent years, there has been an increasing number of ways to distribute document-based malicious code using vulnerabilities in document files. Because document type malware is not an executable file itself, it is easy to bypass existing security programs, so research on a model to detect it is necessary. In this study, we extract main features from the document structure and the JavaScript contained in the stream object In addition, when JavaScript is inserted, keywords with high occurrence frequency in malicious code such as function name, reserved word and the readable string in the script are extracted. Then, we generate a machine learning model that can distinguish between normal and malicious. In order to make it difficult to bypass, we try to achieve good performance in a black box type algorithm. For an experiment, a large amount of documents compared to previous studies is analyzed. Experimental results show 98.9% detection rate from three different type algorithms. SVM, which is a black box type algorithm and makes obfuscation difficult, shows much higher performance than in previous studies.

▸Keyword: malware, PDF, machine learning, java script, detection

## I. Introduction

문서형 악성코드는 주로 이메일이나 웹사이트의 첨부 문서 파일을 통해 유포되며 문서 파일을 클릭하여 확인함과 동시에 사용자 모르게 악성코드가 드롭되거나 다운로드 된다. 때문에 사용자는 자신이 악성코드에 감염되었다는 사실을 알지 못하게 된다.

최근에는 공격자들이 스피어피싱(Spear phishing, 피싱의 진화된 공격 형태로 불특정 다수가 아닌 특정인을 타겟으로 하는 공격 방법), 워터링 홀(Watering hole, 사자가 먹이를 습격하기 위해 물 웅덩이 근처에서 매복하고 있는 형상을 빗댄 것으로 정보 유출 등을 목적으로 타켓화된 사이버 첩보 활동에 주로 쓰이는 공격 방법) 기법을 이용하여 공격 대상 기업의 담당자들에게 이메일로 악성코드를 보내거나 자주 방문하는 웹사이트를 공격하여 악성코드를 유포한 뒤 감염시켜 목표 기업을 공격한다. 이전에는 이메일에 윈도우 운영 체제에서 사용되는 실행 파일 포맷인 PE(portable executable) 파일을 첨부하였으나 최근에는 문서 파일의 취약점을 이용하여 문서형 악성코드를 유포하고 있다.

• First Author: Ah Reum Kang, Corresponding Author: Jiyoung Woo
*Ah Reum Kang (armk@arkang.net), Dept. of Big Data Engineering, Soonchunhyang University
*Young-Seob Jeong (bytecell@sch.ac.kr). Dept. of Big Data Engineering, Soonchunhyang University
**Se Lyeong Kim (srkim@kisa.or.kr). Korea Internet & Security Agency(KISA)
***Jonghyun Kim (jhk@etri.re.kr). Electronics and Telecommunication Research Institute (ETRI)
*Jiyoung Woo (jywoo@sch.ac.kr), Dept. of Big Data Engineering, Soonchunhyang University
***Sunoh Choi (suno@etri.re.kr). Electronics and Telecommunication Research Institute (ETRI)
• Received: 2018. 10. 01, Revised: 2018. 10. 30, Accepted: 2018. 11. 01.
• This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.2016-0-00078, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning) and also supported by Soonchunhyang University Research Fund (No: 20180121).

2016년 8월부터 2017년 3월까지 진행된 골든타임 캠페인은 조직 내 특정 업무 담당자를 감염시키고자 집중적으로 악성코드를 유포한 공격이다. 이 캠페인에는 서로 다른 종류의 악성 파일 문서가 첨부된바 있다. 이중 한 개 이메일은 연세대학교 한반도국제포럼 학술대회 담당자를 사칭하여 표적으로 삼은 담당자를 실제 존재하지 않는 통일 북한 학술대회의 패널로 합류하도록 유도하였다. 다른 이메일은 자신을 분단 전 한국의 강원도 지역인 문천 출신 인물로 소개하며 도움을 호소하는 내용이었다. 두 이메일에 첨부된 악성코드는 동일한 취약점(CVE-2013-0808)을 활용하였고, 이 취약점은 encapsulated postscript(EPS, 고품질의 인쇄를 목적으로 하는 파일 포맷 방식으로 파일 용량이 크고 비트맵 방식과 벡터 방식의 이미지를 동시에 저장할 수 있음) 형식을 대상으로 하였다.

2016년 11월부터 2017년 1월까지 진행된 사악한 새해 캠페인은 2017년 북 신년사 분석이라는 제목을 사용하고 통일부에서 작성된 것처럼 가장한 악성 문서를 첨부한 이메일을 유포한 사례이다. 첨부된 문서 내에 있는 북한의 새해 활동에 관한 세부 사항 링크를 누르는 경우, 악성 OLE(Object Linking and Embedding) 개체 파일이 실행되고 악성코드를 가져올 수 있는 C&C(Command & Control) 서버와 통신을 하게 된다. C&C 서버란 해커가 원하는 공격을 수행하도록 원격에서 명령을 내리거나 악성코드를 제어하는 서버를 말한다.

2017년 2월에는 중국인 이력서로 위장한 악성 파일 문서가 이메일을 통해 유포되는 사례가 공개되었다. 사용자가 문서를 열람하면 PC가 정보 탈취 악성코드에 감염되는 방식이다. 악성코드는 취약점을 이용해 정상 프로그램을 실행하고, 그림 파일로 위장한 정보 탈취 악성코드를 내려 받아 메모리에서만 동작하게 한다. 파일을 생성하지 않으므로 파일 기반 진단을 수행하는 보안 프로그램은 이를 탐지하기 어렵다.

문서형 악성코드는 실행 파일 자체가 아니기 때문에 기존 보안 프로그램을 우회하기 쉽다. 이를 탐지하는 모델 개발을 위해서는 문서 파일의 구조를 이해하는 것이 필요하다. PDF 문서의 구조를 파악하여 악성 문서의 특질을 추출하는 연구가 진행되어 왔다. 과거의 연구는 문서의 구조로부터 도출된 특질을 이용하고 있으며, 그 수가 한정적인데 반해, 본 연구에서는 문서의 구조뿐만 아니라 삽입된 악성 스크립트로부터 텍스트 키워드를 추출하여 보다 정교한 탐지 모델을 제시하고자 한다.

## II. Related Works

문서형 악성 파일은 Microsoft Office, Adobe PDF, 한글과 같은 프로그램의 취약점이나 문서 파일 구조의 취약점을 이용하여 정상적인 문서 파일인 것처럼 속이고 백그라운드로 악성코드를 실행시킨다. Microsoft Office 파일의 경우, 복합 파일 이진 구조 형식으로 미사용 영역 내에 악성코드를 삽입할 수 있는 가능성이 존재하며 Microsoft Office 프로그램에서 제공하는 기능인 매크로를 삽입하여 악성 행위를 추가할 수 있다. 한글 HWP 파일의 경우, 미사용 영역에 비정상 바이너리 데이터를 삽입하거나 HWP에서 사용하지 않는 스트림을 추가할 수 있다. 이에 따라 문서형 악성 파일을 탐지하고, 실행을 차단하기 위한 여러 가지 연구가 진행되고 있다.

문서형 악성코드의 악성 행위 정보를 파일 및 레지스트리, 네트워크, 프로세스의 관점에서 분석하여 해당 행위 정보를 기반으로 탐지하는 연구, 문서의 취약점을 분석하고 내부 구조 및 알려진 취약점을 보여주는 연구, 악성코드 바이너리 실행파일에 포함된 API에 대한 호출 빈도수 및 문자열의 유사도를 비교하여 악성 여부를 판별하는 연구 등이 있다.

아래와 같이 PDF 문서형 악성코드의 주요 연구를 정리하였다.

Laskov와 Srndic [1]의 연구는 문서에 삽입된 자바스크립트를 추출하여 코드를 토큰화하여 특질로 이용하였다.

Smutz와 Stavrou [2]의 연구는 메타데이터와 문서의 구조에 대한 특질을 이용하였다. 메타데이터로는 Stream: 개체의 수, 제목에 있는 소문자 수, 메타 데이터 속성 - 각 필드의 문자 수, 상자 및 이미지 - 각각의 크기 및 위치, 데이터 인코딩 방법, 개체 유형 - 암호화 개체 수를 이용하였다. 구조적 특질은 Count_font, count_javascript,count_js: “/Font” “/JavaScript”, “/JS” 표시의 인스턴스 수, Count_stream_diff: “stream”과 “endstream” 표시의 인스턴스 차이, Pos_box_max: 마지막 상자 표시의 상대 위치, Image_totalpx: 모든 이미지의 모든 픽셀의 합, Producer_len: 메타데이터 개체의 문자 수, Count_obj: “obj” 표시의 인스턴스 수, PDFid0_mismatch: PDFid0 값의 고유 인스턴스 수(일반적으로 항상 1), 전체 문서의 크기에 대한 페이지 수 비율을 이용하였다. 이 연구는 파일에서 추출 가능한 특질과 이를 가공한 특질(count-stream_diff) 등을 사용하였는데, 본 연구에서는 구조적 특징에 대한 통계 정보, 예를 들면 오브젝트의 수, 문서의 크기, stream 발생 빈도 등을 이용하였고, 더 나아가 포함된 스크립트를 해독하여 이로부터 특질을 추출하여, 기존 연구보다 보다 많은 고급 특질이 도출되도록 하였다.

Šrndic와 Laskov [3]의 연구는 PDF parser 오픈 소스 Poppler를 이용하여 자바스크립트를 추출하고, Mozilla 사에서 개발한 SpiderMonkey를 사용, Token화하여 이를 특질로 사용하였다. 이 연구는 자바스크립트를 포함하지 않은 악성 문서를 탐지하지 못하는 한계를 가진다.

Lu 외 [4]의 연구는 동적 분석을 통해 자바스크립트를 실행시키고, 그 안의 opcode를 추출하여 특질로 사용하였다.

Corona 외 [5]의 연구는 악성 자바스크립트를 탐지하는 비교적 간단한 방법을 제시하고자, 자바스크립트 기반의 PDF 악성코드와 일반 PDF는 API 참조 패턴이 다르다는 점에 착안하였다. API 참조를 기반으로 자바스크립트 특성화하는 방법을 이용하여 머신러닝의 입력 데이터로 사용하였다.

Šrndic와 Laskov [6]의 연구는 PDF의 태그의 조합을 특질로 사용하였다. 예를 들면 /OpenAction/JS 등의 태그의 조합을 구하였다.

Li 외 [7]의 연구는 PDF 헤더, 오브젝트 정보, 상호 참조, 트레일러 정보를 추출하고, 난독화 여부, 자바스크립트의 루프 형태, 자바스크립트 상의 쓸모없는 정보, 트레일러의 특정 패턴 등을 특질로 이용하였다. 난독화된 부분을 해독하고, 자바스크립트의 내용을 심층 분석한 면에서 의의를 가진다.

Khitan 외 [8]는 YARA 규칙을 정의하고, 바이트 시퀀스와 문자열을 검색하여 PDF 파일의 malware를 찾아내는 방법을 제안하였다. 여기서 사용한 방법은 header 부분 이상, 참조 부분 없음, 자바스크립트 포함, 수상한 open action, 파일 삽입으로 이루어졌다. 구조 분석을 통해 키워드를 추출하고 이를 규칙화한 결과와 기계 학습을 적용한 결과를 비교하였다.

Cuan 외 [9]는 malware의 99.7%를 탐지할 수 있는 SVM 분류기를 설정하였다. 그러나 이 분류기는 악의적인 PDF 파일을 정상 파일로 속이는 것이 쉽다. 이 SVM을 피하기 위하여 그래디언트 디센트 공격을 구현하였고, 이를 탐지할 수 있는 모델을 만들었다.

Zhang [10]의 연구는 PDF 기반 악성코드 탐지를 위해 MLPdf라 불리는 MLP 신경망 모델을 기반으로 한 새로운 접근법을 제안하였다. MLPdf 모델은 모델 업데이트를 위한 확률적 그래디언트 디센트 검색을 갖는 역전파 알고리즘을 사용한다. 약 105,000개의 양성 및 악의적인 PDF 문서로 구성된 두 개의 실제 데이터 셋에서 고품질 특질을 추출하였다. 평가 결과는 0.08%의 false positive와 95.12%의 true positive 비율을 보였다.

Torres와 Santos [11]의 연구는 자바스크립트가 내장된 PDF 문서에서 악성코드 탐지에 대한 기계 학습 기술을 사용하여 바이러스 백신, 샌드 박스 등과 같은 기존 솔루션을 효과적으로 보완할 수 있는지 여부를 확인하는 것을 목표로 하였다. 또한, malware 탐지에서 여러 가지 감시 대상 컴퓨터 학습 알고리즘 간의 비교 결과와 분류 체계에 대한 전반적인 설명을 제공하였다.

Matiorca 외 [12]는 PDF 파일의 구조와 관련이 있는 특질 추출기 모듈과 효과적인 분류자를 결합한 기법을 제시하였다. 이 시스템은 대부분의 상용 바이러스 백신뿐만 아니라 악의적인 PDF 검색을 위한 연구 도구보다 효과적임을 입증하였다.

Liu 외 [13]은 PDF에서 악의적인 자바스크립트를 탐지하고 제한하기 위한 컨텍스트-어웨어 방식 접근법을 제안하였다. 정적인 기능을 정적으로 추출하고 컨텍스트 모니터링 코드를 문서에 삽입하였다. 이 문서가 열리면 컨테스트 모니터링 코드가 자바스크립트 실행 컨텍스트에서 잠재적 감염 시도를 탐지하기 위해 런타임 모니터와 협조한다. 탐지기는 정적 및 런타임 기능을 모두 사용하여 악성 문서를 식별할 수 있다. 실험은 18,623개의 정상 PDF 샘플과 7,370개의 악성 샘플을 사용하였다. 평가 결과는 PDF에서 악의적인 자바스크립트를 정확하게 탐지하고 제한할 수 있음을 보여주었다.

본 연구는 기존의 연구의 한계점을 극복하고자 자바스크립트의 포함여부와 관계없이 다양한 특질을 이용하여 우회가 어려운 탐지 알고리즘을 제시하고자 한다.

# III. Proposed Model

## 1.1 PDF structure analysis

PDF는 일반 사용자가 사용하는 기능 이외의 다양한 기능을 제공하는데, 이 중 하나가 자바스크립트를 포함한 오브젝트를 추가할 수 있는 기능이고, 악성코드는 이를 악용하는 경우가 많다.

PDF는 다음과 같은 구성 요소를 가진다.

- Object: PDF는 데이터 오브젝트들로 구성됨
- File Structure: 파일 구조는 오브젝트 저장, 접근, 업데이트정보를 포함하고 있음
- Document Structure: 문서 구조는 여러 오브젝트 구성이 문서를 구성하고 배치하는지에 대한 정보를 담고 있음
- Content Streams: 콘텐트 스트림, 문서의 외관과 그래픽 요소 설명하는 코드를 담고 있음

PDF 파일 구조를 살펴보면, 그림 1에서처럼 Header, Body, Cross-reference table, Trailer로 구성되어 있다.

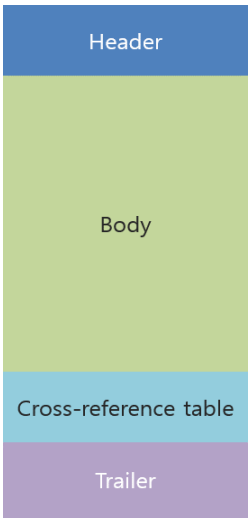


Fig. 1. PDF structure

- Header
  - PDF 명세 버전을 알리는 한 줄의 헤더
- Body
  - PDF 문서를 구성하는 오브젝트를 포함하는 body
  - 문서의 실질적인 내용을 담고 있는 간접 오브젝트들로 구성됨
  - 문서의 내용, 폰트, 페이지, 이미지 요소들을 나타냄

- Cross-reference (xref) table
  - 오브젝트들을 참조할 때 사용되는 테이블
  - 간접 오브젝트에 관한 정보를 가지고 있는 전역 참조 테이블
  - 각 간접 오브젝트의 위치 정보를 가지고 있으며 특정 오브젝트의 랜덤 액세스를 가능하게 해줌
  - 각 엔트리는 20바이트 길이, end-of-line 마커를 포함
- Trailer
  - Body 영역에 존재하는 오브젝트 중에서 Root 오브젝트가 무엇인지, Cross-reference table 위치는 어디인지 표시함
  - trailer라는 지시어로 시작하여 %EOF로 끝남
  - << >> 사이에 현재 오브젝트 속성을 표시해 주는데, /Size는 Cross-reference table 항목 수를 나타내며, /Root는 Root 오브젝트의 번호를 나타냄
  - startxref는 Cross-reference table의 위치를 알려줌

```
98  trailer
99  <<
100     /Root 1 0 R
101     /Size 5
102 >>
103 startxref
104 7507
105 %%EOF
```

Fig. 2. trailer

그림 2는 trailer의 예로 파일 끝에 위치하며 Root 오브젝트와 Cross-reference table의 위치를 표시한다. 붉은 색 박스 안에 있는 내용은 1번째 오브젝트의 오프셋 0을 참조함을 의미한다. R은 reference의 약자이다. 그림 2는 Root 오브젝트는 1번, Cross-reference table 항목 수는 5개 , Cross-reference table의 시작 오프셋 값이 7507임을 나타낸다.

```
91  xref
92  0 5
93  0000000000 65535 f
94  0000000010 00000 n
95  0000000120 00000 n
96  0000000189 00000 n
97  0000000408 00000 n
```

Fig. 3. cross-reference table

그림 3은 cross-reference table의 예로 PDF Body 부분에 존재하는 오브젝트들의 위치 값과 사용 여부를 나타낸다. 테이블의 각 항목은 공백을 포함하여 20 바이트의 크기를 가진다. xref 키워드는 Cross-reference table의 시작을 표시한다. f는 free entry라는 의미로 사용 안함을 뜻하고, n은 In-use entry라는 의미로 사용함을 뜻한다. 92번 라인의 숫자 0은 오브젝트 시작 번호를 뒤의 숫자 5는 엔트리 수를 나타낸다. 93번 라인은 오브젝트 0번, 오프셋 0, 사용 안함을 나타내며, 94번 라인은 오브젝트 1번, 오프셋 10, 사용함을 나타낸다. 95번 라인은 오브젝트 2번, 오프셋 120, 사용함, 96번 라인은 오브젝트 3번, 오프셋 189, 사용함, 97번 라인은 오브젝트 4번, 오프셋 408, 사용함을 나타낸다.

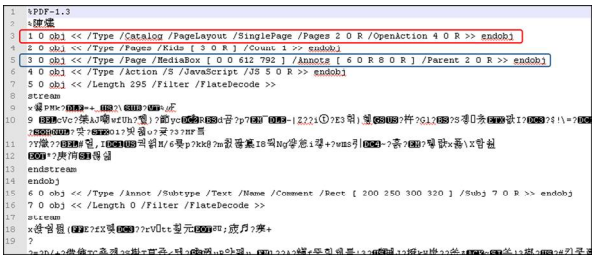


Fig. 4. body

그림 4는 body의 예시이다. 붉은색 박스의 오브젝트 1번은 타입은 catalog, 페이지 레이아웃은 싱글 페이지, 페이지 내용은 2번 오브젝트를 참고, 문서 열람 시 액션은 4번 오브젝트를 참고한다는 의미이다. 파란색 박스의 오브젝트 3번은 타입은 페이지, 미디어 컨텐츠를 포함하는 사각형 그림 있음(좌표 표시), 6번, 8번 오브젝트를 추가로 참고하며, 부모 오브젝트는 2번이라는 것을 나타낸다.

오브젝트는 여러 종류를 가지는데, 그 중 스트림(stream) 오브젝트는 연속적인 바이트의 집단(이진 데이터)으로, 길이에 제약이 없어 크기가 큰 이미지 파일이나 페이지 구성하는 오브젝트이다. 이 오브젝트는 Dictionary 오브젝트와 그 뒤를 따라서 오는 키워드 stream과 endstream 사이의 여러 바이트들로 구성된다.

- Boolean values: TRUE, FALSE, 배열의 요소, dictionary의 엔트리 값
- Integer and real numbers: 정수형, 실수형
- Strings: 문자 스트링, 16진수 스트링, 길이에 제약이 있음
- Names: 하나의 유일한 문자열, 대소문자 구별
- Arrays: 배열에 다양한 형태의 오브젝트 조합을 가질 수 있음
- Dictionaries: 키와 값으로 구성
- Streams: 연속적인 바이트의 집단(이진 데이터)
- 간접 오브젝트: 레이블된 오브젝트

PDF 파일에서 레이블된 오브젝트는 오브젝트 고유의 식별자를 갖게 되며 다른 오브젝트는 이를 이용하여 해당 오브젝트를 간접 참조할 수 있다. 간접 오브젝트는 오브젝트의 수, 생성 수, 키워드 obj와 endobj로 둘러싸여진 오브젝트 값으로 구성된다.

악성 PDF는 보통 이 스트림 오브젝트에 악성코드를 삽입하게 되는데, 다른 타입은 길이의 제약을 가지므로 stream 오브젝트를 사용한다.

그림 5는 정상 파일과 악성 파일을 PDFStreamDumper 도구를 이용하여 비교한 것이다. 정상 문서의 경우에는 구성하는 오브젝트의 수가 많은데 비해 악성 파일은 오브젝트의 수가 적다. 악성 문서를 분석해보면 자바스크립트가 포함되어 있는 경우가 대부분인데, stream 오브젝트 안에 PDF가 지원하는 인코딩 방식으로

인코딩 되어 있거나, PDF가 지원하는 특수 문자 구별 코드(#)를 사용하여 난독화되어 있다. 가장 일반적인 인코딩 형식은 FlateEncode이며, ASCIIHexEncode와 ASCII85Encode가 중첩된 형태나 FlateEncode와 ASCII85Encode가 중첩된 형태 등도 존재한다.

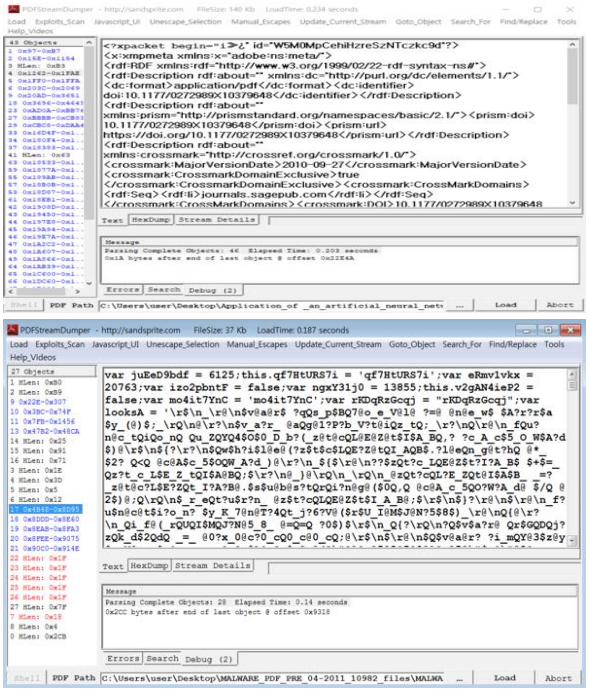

Fig. 5. benign (above) vs. malicious (below) file

그림 5의 아래 그림은 stream object에서 인코딩되어있는 부분을 PDFStreamDumper를 이용하여 디코딩을 한 화면이다.

### 1.2 PDF feature extraction

PDF의 구조 분석을 통해 문서형 악성코드를 탐지를 위한 126개의 feature를 추출하였는데, 다음과 같이 구분된다.

- PDF 파일 기본 정보 (2개)
  - 사이즈, 버전
- PDF 파일 인코딩 및 자바스크립트 삽입 정보 (4개)
  - ASCIIHex, Flate, ASCII85, JAVASCRIPT
- PDF 파일 내의 주요 키워드 정보 (43개)
  - obj, endobj, stream, endstream, xref, trailer, startxref, /Page, /Encrypt, /ObjStm, /JS, /JavaScript, /AA, /OpenAction, /AcroForm, /JBIG2Decode, /RichMedia, /Launch, /EmbeddedFIle, /XFA, /Colors > 2^24, /Type, /Catalog, /Version, /Pages, /OpenAction, /rovers, /abdula, /Kids, /Count, /S, /Filter, /FlateDecode, /Length, /ASCIIHexDecode, /ASCII85Decode, /LZWDecode, /RunLengthDecode, /CCITTFaxDecode, /JBIG2Decode, /DCTDecode, /JPXDecode, /Crypt
- PDF 파일에 인코딩되어 삽입된 스크립트 내 텍스트 정보 (77개)
  - 메소드명: charCodeAt, fromCharCode, charAt, IndexOf, replace, substring, split, join, push, unescape
  - 개체명, 속성명: Array, String, Date, Number, Object, Number, Object, length
  - 연산자, 상수, 타입, 예약어: var, if, this, function, return, for, null, false, new, true, while, try, catch
  - 가독 스트링: app, fnc, plugIns, buf, sum, arr, num, doc, nPage, getAnnots, subject, syncAnnotScan, proc, out, func, info, Function, viewerVersion, array, STARTSCRIPT, mail, author, pageNum, getPageNumWords, viewerType, creationdate, moddate, numPages

PDF가 지원하는 인코딩 방식 중 ASCIIHexDecode 필터는 ASCII 16진수 형식으로 인코딩된 데이터를 디코딩한다. FlateDecode 필터는 ZIP 알고리즘을 기반으로 하여 인코딩된 데이터를 디코딩한다. ASCII86Decode 필터는 ASCII 기본 85 인코딩으로 인코딩된 데이터를 디코딩하고 바이너리 데이터를 생성한다.

아래에 PDF 파일 내에 존재하는 정상 문서와 악성 문서의 특징이 구별되는 주요한 키워드의 의미를 정리하였다.

- obj, endobj: 오브젝트의 시작과 끝을 알리는 태그
- stream, endstream: 스트림의 시작과 끝을 알라는 태그
- /Page: PDF 내에 존재하는 페이지 수 정보
- /JS, /JavaScript: 자바스크립트 시작을 알리는 태그
- /AA, /OpenAction: 자동 실행
- /Type: 오브젝트의 타입 정보

본 연구에서는 위와 같은 PDF 파일의 기본 정보인 파일 size와 version 정보, PDF 파일 인코딩 및 자바스크립트 삽입 정보 4개, PDF 파일 내의 주요 키워드 43개의 발생 건수를 특질로 사용하였다. 또한, 자바스크립트가 삽입된 경우에는 스크립트 내 메소드명, 개체명, 속성명, 연산자, 상수, 타입, 예약어 및 가독 스트링 중 악성코드 내에서의 발생 빈도가 높은 77개를 추출하여, 발생 건수를 특질로 사용하였다.

## Ⅳ. Experiment Results

본 연구에서는 2012년 2월부터 2017년 3월까지 F-secure, Kaseperskey, FireEye, Bitdefender, Crysys 등의 주요 보안 회사에서 수집한 20,093개의 문서를 분석하였다. 기존 연구에서 분석한 샘플의 수에 비해 본 연구에서는 대용량의 문서를 분석하였다. 실험 데이터는 악성 문서는 11,093개, 9,000개의 정상 문서로 구성되어 있다.

### 1.1 Feature statistics

특질 그룹별 정상과 악성 문서의 분포를 살펴보면 다음과 같다.

Table 1. PDF Version

| | malware | clean |
|---|---|---|
| %PDF-0.9 | 2 | 0 |
| %PDF-1. | 1,770 | 1,147 |
| %PDF-1.0 | 576 | 3 |
| %PDF-1.1 | 65 | 144 |
| %PDF-1.2 | 71 | 536 |
| %PDF-1.3 | 5,971 | 1,142 |
| %PDF-1.4 | 921 | 2,610 |
| %PDF-1.5 | 68 | 322 |
| %PDF-1.6 | 945 | 1,001 |
| %PDF-1.7 | 31 | 2,095 |
| etc. (%PDF-, -aaa, II*) | 673 | 0 |
| total | 11,093 | 9,000 |

Table 2. PDF Feature Statistics

| | malware | clean |
|---|---|---|
| ASCIIHexDecode, # | 303 | 0 |
| FlateDecode # | 3769 | 0 |
| ASCII85Decode # | 128 | 0 |
| JavaScript # | 5362 | 0 |
| size, mean | 23,310 | 95,080 |
| /Page, mean of # | 0.75 | 2 |
| /Type,mean of # | 3 | 5.6 |
| obj,mean of # | 11 | 90 |
| stream, mean of # | 3.4 | 30 |
| /FlateDecode, mean of # | 1.4 | 4.6 |
| /Length, mean of # | 3 | 13 |

### 1.2 Training algorithm

추출한 126개의 특질을 이용하여 정상과 악성을 구분할 수 있는 학습 모델을 생성하였다. 학습 알고리즘은 동작 방식이 상이한 기계 학습 알고리즘의 분류 중 대표적인 알고리즘을 선택하였다. 사용한 알고리즘은 Naive Bayes, decision tree 방식의 random forest, support vector machine을 이용하였다.

Naive Bayes는 특질의 독립성을 가정하고, 학습 데이터로부터 두 부류에서 발생 확률을 특질별로 계산하고, 새로운 데이터의 특질에 대해 앞서 구한 발생 확률의 곱을 구해 분류하는 알고리즘이다. Support vector machine(SVM)은 두 부류를 나누는 linear hyper plane을 찾는데, 두 부류의 데이터에서 가장 먼 의사결정 평면을 찾는 알고리즘이다. SVM은 분류나 예측 문제에 동시에 사용할 수 있으며 예측의 정확도가 높다. 또한, 많은 양의 feature가 있어도 잘 다룰 수 있는 능력이 있는 것으로 알려져 있다. 신경망 기법에 비해 과적합 정도가 덜하다. 그러나 모형 구축 시간이 오래 걸리는 단점이 있다. Random forest 알고리즘은 여러 개의 결정 트리들을 임의적으로 학습하는 방식의 앙상블 방법이다. CART 방식을 이용해 트레이닝 데이터의 일부로 여러 개의 decision tree들을 학습시킨다. 분류하는 동안 모든 tree의 투표를 실시하고, 많은 표를 받은 하나의 값이 결과로 도출된다. Random forest 알고리즘은 뛰어난 일반화 능력과 데이터 노이즈에 강한 장점이 있다.

Smutz와 Stavrou[2]의 기존 연구가 본 연구에서 사용한 특질과 비슷한데, 본 연구에서는 인코딩 방법, 스크립트내의 주요 키워드를 추출하여 특질로 사용한 방법이 다르다. 이들의 연구에서는 random forest의 방법을 제외하고는 탐지율 60% 수준의 낮은 성능을 보였는데, 본 연구에서는 추가 특질을 사용하였을 때 정확도가 향상되는 정도를 보고자 한다. Random forest와 같은 의사결정나무 모델은 주요 특질이 파악가능하고, 이들은 우회할 수 있는데, 예를 들면, [2]의 연구에서처럼 악성 문서를 의도적으로 수정하여 일부 기능을 “정상화”하고 포함된 악성 콘텐츠를 그대로 유지하면서 양성 문서와 유사하게 만드는 것이 가능하다. 문서 내 폰트의 수를 악성 문서에서 작위적으로 높게 설정하게 되면 해당 방법을 우회할 수 있다. 우회를 어렵게 하기 위해서는 트리 형태의 모델보다는 black box 형태의 알고리즘에서 좋은 성능을 내는 것이 중요하다.

학습 결과는 다음과 같다. 표 3은 random forest를 제외하고 10 교차 검증으로 검증한 결과를 나타낸다. Random forest는 앙상블 트리를 형성하면서 일부 데이터를 이용하여 트리를 형성하는 교차 검증 방식을 적용하므로 따로 교차 검증을 실시할 필요가 없다. 알고리즘의 성능은 정상/악성 클래스별로 정밀도(precision)와 재현율(recall), 그리고 이 둘의 조화 평균인 F-measure로 평가하였다.

Table 3. Algorithm performance

| | | precision | recall | F-measure |
|---|---|---|---|---|
| Naive Bayes | benign | 0.898 | 0.956 | 0.926 |
| | malware | 0.963 | 0.912 | 0.936 |
| | weighted avg. | 0.933 | 0.932 | 0.932 |
| SVM | benign | 0.976 | 0.999 | 0.987 |
| | malware | 0.999 | 0.98 | 0.99 |
| | weighted avg. | 0.989 | 0.989 | 0.989 |
| Random Forest | benign | 1 | 1 | 1 |
| | malware | 1 | 1 | 1 |
| | weighted avg. | 1 | 1 | 1 |

결과에서 보듯이 black box 형태의 알고리즘인 SVM에서도 98.9%의 성능을 나타내어, 문서 구조의 특질을 이용한 과거의 연구에서보다 훨씬 높은 성능을 보였다.

### 1.3 Feature importance

Random forest 학습에 주요하게 작동한 변수를 살펴보면 다음과 같다. Random forest에서 사용한 변수 중요도는 지니(GINI) 지수로 구했는데, 이는 특정 변수로 트리를 분기했을 때 감소하는 노드의 불순도를 측정한다. 지니 지수는 분기 시에 특정 클래스의 비율과 나머지 클래스의 비율의 곱을 모든 클래스에 대해 합한 값으로 구한다.

그림 3에 변수의 중요도를 시각화하였다. 가장 중요한 변수는 size로 악성 문서는 정상 문서에 비해 크기가 작은 특징을 가진다. 참조 오브젝트의 시작을 알리는 startxref, 오브젝트의 끝을 나타내는 endobj, 그 다음으로는 자바스크립트 포함 건수가 중요한 변수로 나타난다.

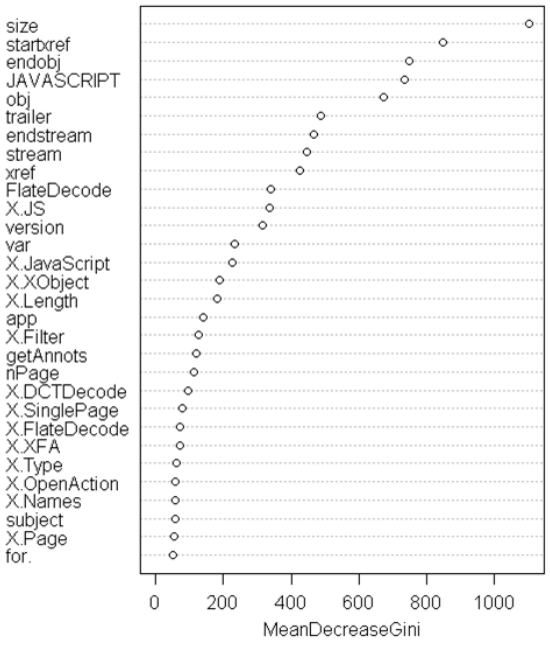


Fig. 3. Feature importance

추출한 126개의 특질을 이용하여 정상과 악성을 구분할 수 있는 학습 모델을 생성하였다. 학습 알고리즘은 동작방식이 상이한 기계학습 알고리즘의 분류중 대표적인 알고리즘을 선택하였다. 사용한 알고리즘은 Naive Bayes, decision tree 방식의 random forest, support vector machine을 이용하였다. Random forest와 같은 의사결정나무 모델은 주요 특질이 파악 가능하고, 이를 우회할 수 있다. 우회를 어렵게 하기 위해서는 트리 형태의 모델보다는 black box 형태의 알고리즘에서 좋은 성능을 내는 것이 필요하다.

실험 결과 세 가지 알고리즘에서 98.9%의 탐지율을 보였으며, black box 형태의 알고리즘인 SVM에서도 과거의 연구에서보다 훨씬 높은 성능을 보였다. 본 연구는 악성 문서 파일 11,093개, 정상 문서 파일 9,000개를 분석함으로써 기존 연구에 비하여 대량의 문서를 대상으로 하였다.

향후 연구로는 딥러닝을 이용한 PDF 문서형 악성코드 탐지를 계획하고 있다. PDF 문서형 악성코드의 대부분이 자바스크립트를 이용하여 취약점을 익스플로잇(exploit)하는 방법을 사용한다. 자바스크립트는 PDF에서 지원하는 여러 방식으로 인코딩 되어있으며, 길이에 제약이 없는 stream 타입의 오브젝트에 존재하는데, 이에 착안하여 stream 타입의 오브젝트의 raw data를 입력 값으로 사용하여 딥러닝 모델에 적용할 것이다. 또한, 그 결과를 본 연구에서 제안한 방식과 비교 평가할 예정이다.

## IV. Conclusions

악성코드는 이메일에 실행 파일을 첨부하는 기존의 형태에서 더 나아가 문서 파일의 취약점을 이용하여 악성코드를 유포하는 방식으로 진화하고 있다. 문서형 악성코드는 실행 파일 자체가 아니므로 기존의 보안 프로그램들을 우회하기 쉽고, 오탐의 위험이 높다. 때문에 이를 비교적 정확히 탐지하는 모델에 대한 연구가 필요하다. 악성 문서 파일을 탐지하기 위해서는 문서 구조를 이해하여야 하며, 본 연구에서는 PDF 문서를 대상으로 하였기에 PDF 구조 분석을 통해 과거의 연구보다 좀 더 다양한 특질을 추출하는데 중점을 두었다.

분석 결과 악성 PDF는 이들 오브젝트에 악성코드를 삽입하게 되는데, 다른 타입은 길이의 제약을 가지므로 stream 오브젝트를 사용하게 된다. 긴 바이트를 분석해보면 자바스크립트가 포함되어 있는 경우가 대부분인데, 자바스크립트가 바 로 보이게 되면 탐지될 우려가 높기 때문에 여러 가지 형태로 인코딩되어 있음을 알 수 있었다.

본 연구에서는 문서의 구조적 특징을 파악하고 스트림 오브젝트에 포함된 자바스크립트를 해독하여 이로부터 PDF 파일 기본 정보 2개, PDF 파일 인코딩 및 자바스크립트 삽입 정보 4개, PDF 파일 내의 주요 키워드 정보 43개의 발생 건수를 특질로 추출하였다. 이외에도 자바스크립트가 삽입된 경우 스크립트 내 메소드명, 개체명, 속성명, 연산자, 상수, 타입, 예약어 및 가독 스트링 중 악성코드 내에서의 발생 빈도가 높은 77개를 추출하여, 발생 건수를 특질로 사용하였다.

## REFERENCES

[1] P. Laskov and N. Srndic, “Static Detection of Malicious JavaScript-Bearing PDF Documents,” Proceedings of the Annual Computer Security Applications Conference (ACSAC), pp.373-382, 2011.

[2] C. Smutz and A. Stavrou, “Malicious PDF Detection using Metadata and Structural Features,” Proceedings of the 28th Annual Computer Security Applications Conference, pp.239-248, 2012.

[3] N. Šrndic and P. Laskov, “Detection of Malicious PDF Files Based on Hierarchical Document Structure,” Proceedings of the 20th Annual Network & Distributed System Security Symposium, pp.1-16, 2013.

[4] X. Lu, J. Zhuge, R. Wang, Y. Cao, and Y. Chen, “De-obfuscation and Detection of Malicious PDF Files with High Accuracy,” Proceedings of the 46th Hawaii International Conference on System Sciences (HICSS), pp.4890-4899, 2013.

[5] I. Corona, D. Maiorca, D. Ariu and G. Giacinto, “Lux0r: Detection of Malicious PDF-embedded Javascript Code through Discriminant Analysis of API References,” Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop, pp.47-57, 2014.

[6] N. Šrndić and P. Laskov, "Hidost: a static machine-learning-based detector of malicious files," EURASIP Journal on Information Security, vol.2016, no.1, pp.22, 2016, 9.
[7] M Li, Y Liu, M Yu, G Li, and Y Wang, "FEPDF: A Robust Feature Extractor for Malicious PDF Detection," Proceedings of BigDataSE/ICESS 2017, pp.218-224, 2017.
[8] S. Khitan, A. Hadi and J. Atoum, "PDF Forensic Analysis System using YARA," International Journal of Computer Science and Network Security, vol.17, no.5, pp.77-85, 2017, 5.
[9] B. Cuan, A. Damien, C. Delaplace, and M. Valois, "Malware Detection in PDF Files Using Machine Learning," SECRYPT 2018 - 15th International Conference on Security and Cryptography, pp.8, 2018, 7.
[10] J. Zhang, "MLPdf: An Effective Machine Learning Based Approach for PDF Malware Detection," arXiv:1808.06991v1, 2018, 8.
[11] J. Torres and S. D. L. Santos, "Malicious PDF Documents Detection using Machine Learning Techniques," Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018), pp.337-344, 2018.
[12] D. Maiorca, G. Giacinto, and I. Corona, "A Pattern Recognition System for Malicious PDF Files Detection," Perner, P. (ed.) MLDM 2012, LNCS(LNAI), vol.7326, pp.510-524, 2012.
[13] D. Liu, H. Wang, and A. Stavrou, "Detecting Malicious Javascript in PDF through Document Instrumentation," Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2014, 6.

## Authors

Ah Reum Kang received the M.S. and Ph.D degree in Information Security from in Seoul, South Korea in 2012 and 2016, respectively. She is a currently research professor at SCH Convergence Science Laboratory, Soonchunhyang University. From 2016 to 2018, she was a researcher at the Department of Computer Science and Engineering at the University at Buffalo, State University of New York, USA. Her research interest includes computer security, privacy-preserving data mining and data science in general.

Young-Seob Jeong received a BS in Computer Science from Hanyang University, Korea, in 2012, an MSc in Computer Science from KAIST, Korea, and in 2016, a PhD in School of Computing from KAIST, Korea. He joined the faculty of the Department of Big Data Engineering at Soonchunhyang University, Asan city, Korea, in 2017. His current research topics are text mining, information extraction, action recognition, and dialog systems, where his favorite techniques are topic modeling and deep learning.

Se Lyeong Kim received B.S degree in computer science from Seoul Women's University in 2005, and received M.S degree in Information Security from Korea University in 2012. She is a Deputy Researcher of Korea Internet & Security Agency (KISA).

Jonghyun Kim received the M.S. degree and the Ph.D degree in computer science from the University of Oklahoma, USA, in 2000 and 2005, respectively. He was a research with the Samsung Electronics in 1995-1997 and the Samsung SDS in 2000. He joined the Electronics and Telecommunications Research Institute (ETRI) in 2005. He is also involved in standardization activities as an association rapporteur of ITU-TSG17/Q.4 (cybersecurity). His research interests include Information Security, Cyber Security, Network Forensics and Network Security Function Virtualization.

Jiyoung Woo received the B.S., M.S., Ph.D degree in industrial engineering from KAIST in 2000, 2002, and 2006. She is a currently professor at Big data engineering department, Soonchunhyang University. From 2008 to 2010, she was a researcher with AI lab of Arizona University, USA. She was a research professor at Graduate School of Information Security, Korea University from 2010 to 2016. Her research interest includes data mining and business intelligence.

Sunoh Choi received the B.S. and M.S. degree in Computer Science from Korea University in 2005 and 2008, and received Ph.D degree in Electrical and Computer Engineering from Purdue University in 2014. He is a senior researcher at Information Security Division, Electronics and Telecommunication Research Institute (ETRI) since 2014. His research interest includes data security, network security, and artificial intelligence.