



국민대학교
소프트웨어융합대학
소프트웨어학부


캡스톤 디자인 I

종합설계 프로젝트

| | |
|--------|---|
| 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) |
| 팀 명 | <i>Do it!</i> |
| 문서 제목 | 중간보고서 |

| | |
|---------|------------|
| Version | 1.4 |
| Date | 2019-04-18 |

| | |
|------|---------|
| 팀원 | 문다민(팀장) |
| | 김기환 |
| | 김현석 |
| | 정혜리 |
| | 방유한 |
| 지도교수 | 윤명근 교수 |

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 **DREAM**(Detecting in Real-time mAlicious document using Machine Learning)를 수행하는 팀 **Do it!** 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 **Do it!**의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역

| | |
|-----------------|-------------------------|
| Filename | 8조_중간보고서_최종.docx |
| 원안작성자 | 문다민, 김현석 |
| 수정작업자 | 문다민, 김현석, 정혜리, 김기환, 방유한 |

| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
|------------|-------|----------|----------|----------------|
| 2019-04-13 | 문다민 | 1.0 | 최초 작성 | |
| 2019-04-14 | 김현석 | 1.1 | 내용 추가 | 수정 사항 내용 추가 |
| 2019-04-15 | 정혜리 | 1.2 | 내용 추가 | 수행 사항 내용 추가 |
| 2019-04-16 | 김기환 | 1.3 | 내용 추가 | 향후 계획 추가 |
| 2019-04-17 | 방유한 | 1.3.1 | 내용 수정 | 수정 사항 수정 |
| 2019-04-18 | 문다민 | 1.4 | 내용 수정 | 최종 검토 및 오타자 수정 |
| | | | | |

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

목 차

| | | |
|----------|------------------------------------|----|
| 1 | 프로젝트 목표 | 4 |
| 2 | 수행 내용 및 중간결과 | 5 |
| 2.1 | 계획서 상의 연구내용 | 5 |
| 2.1.1 | 데이터 라벨링(Data Labeling) | 5 |
| 2.1.2 | 특징 추출(Feature Extraction) | 6 |
| 2.1.3 | 기계 학습(Machine Learning) | 6 |
| 2.1.4 | 웹 서버(Web server) | 6 |
| 2.2 | 수행내용 | 8 |
| 2.2.1 | 데이터 수집 | 8 |
| 2.2.2 | 데이터 라벨링 | 9 |
| 2.2.3 | 특징 추출 | 10 |
| 2.2.3.1. | PDF | 10 |
| 2.2.3.2. | MS Word | 13 |
| 2.2.4 | 기계 학습 | 14 |
| 2.2.4.1. | PDF | 15 |
| 2.2.4.2. | MS Word | 16 |
| 2.2.4.3. | 결과 분석 | 17 |
| 2.2.5 | 엔진 | 18 |
| 2.2.6 | 웹 서버 | 20 |
| 3 | 수정된 연구내용 및 추진 방향 | 21 |
| 3.1 | 수정사항 | 21 |
| 3.1.1 | 웹 | 21 |
| 3.1.2 | 데이터 라벨링 | 22 |
| 4 | 향후 추진계획 | 23 |
| 4.1 | 향후 계획의 세부 내용 | 23 |
| 4.1.1 | 특징 추출 | 23 |
| 4.1.1.1. | MS Word | 23 |
| 4.1.1.2. | PDF | 24 |
| 4.1.2 | 기계 학습 | 24 |
| 4.1.2.1. | 학습 알고리즘 실험 | 24 |
| 4.1.2.2. | 하이퍼 파라미터(Hyper parameter) 실험 | 24 |
| 4.1.3 | 바이러스 공유 채널 | 25 |
| 4.1.4 | 오픈소스 커뮤니티 | 25 |
| 5 | 고충 및 건의사항 | 26 |

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

1 프로젝트 목표

의심스러운 파일 및 URL을 분석하고 모든 종류의 악성 코드를 탐지하는 서비스인 바이러스 토탈(VirusTotal)에 따르면 PDF, MS Word, MS Excel 등 우리가 자주 사용하는 문서형 파일이 악성 코드로 유입되고 있음을 알 수 있다.

그리고 문서형 악성코드로 인한 사회적 피해가 전 세계에 걸쳐 지속해서 발생하고 있지만, 문서형 악성코드를 전문적으로 탐지하는 안티바이러스는 많지 않다. 이는 문서형 악성코드가 쉽게 유포되어 사용자들의 PC에 감염될 수 있고 사회적 문제가 발생할 수 있다.

본 프로젝트에서는 문서형 파일 중 의심스러운 문서형 파일을 탐지하는 엔진을 제작하여 위에서 언급한 유형의 악성 코드의 유포와 그로 인한 피해가 생기는 것을 막고자 한다.

● 세부 목표

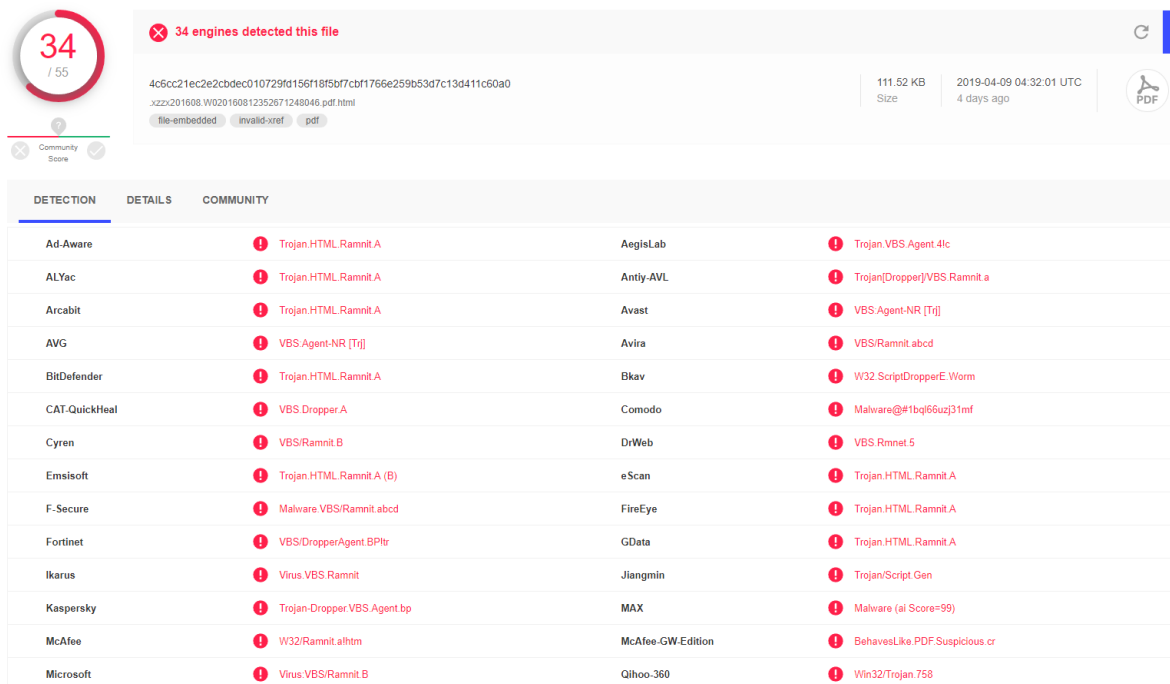
- 사용자가 엔진에 전송한 파일을 분석하여 악성 여부 확률 정보를 제공한다.
- 엔진을 웹 서버, 메일 서버 등 다양한 서비스에 확장할 수 있도록 개발한다.
- 사용자들의 피드백을 반영하기 위해 오픈 소스 커뮤니티를 운영하여 엔진을 유지 보수한다.
- AWS로 클라우드(Cloud) 서버를 운영하여 주기적으로 문서형 악성코드를 탐지하는 모델을 재 학습시키고 사용자들에게 배포한다.
- 엔진 API 문서를 작성한다.
- 여러 사용자와 데이터를 공유할 수 있는 웹 사이트를 개발한다.

2 수행 내용 및 중간결과

2.1 계획서 상의 연구내용


2.1.1 데이터 라벨링(Data Labeling)

바이러스토탈은 약 60여 개의 안티바이러스가 파일에 대해서 탐지 결과 및 진단명을 제공하는 서비스이다. 바이러스토탈에 등록된 안티바이러스 중 글로벌 안티바이러스 테스트 기관인 'AV-TEST'의 성능 지표를 종합한 상위 5개의 안티바이러스의 탐지 결과로 독자적인 라벨을 만든다. 파일에 대한 탐지 결과를 얻기 위해 바이러스토탈의 기능을 사용할 수 있는 API를 사용한다.



| DETECTION | DETAILS | COMMUNITY |
|---------------|-----------------------------|-------------------|
| Ad-Aware | Trojan.HTML.Ramnit.A | AegisLab |
| ALYac | Trojan.HTML.Ramnit.A | Antiy-AVL |
| Arcabit | Trojan.HTML.Ramnit.A | Avast |
| AVG | VBS.Agent-NR [Tij] | Avira |
| BitDefender | Trojan.HTML.Ramnit.A | Bkav |
| CAT-QuickHeal | VBS.Dropper.A | Comodo |
| Cyren | VBS/Ramnit.B | DrWeb |
| Emsisoft | Trojan.HTML.Ramnit.A (B) | eScan |
| F-Secure | Malware.VBS/Ramnit.abcd | FireEye |
| Fortinet | VBS/DropperAgent.BPTr | GData |
| Ikarus | Virus.VBS.Ramnit | Jiangmin |
| Kaspersky | Trojan-Dropper.VBS.Agent.bp | MAX |
| McAfee | W32/Ramnit.alhtm | McAfee-GW-Edition |
| Microsoft | Virus.VBS/Ramnit.B | Qihoo-360 |

<그림 1> 바이러스토탈에서 제공하는 여러 안티바이러스의 탐지 결과 화면
출처 = www.virustotal.com

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

2.1.2 특징 추출(Feature Extraction)

PDF와 MS Word 파일에서 악성과 정상을 구분할 수 있는 특징을 추출한다. 본 프로젝트는 PDF나 MS Word 문서의 구조적 정보를 특징으로 한다. 문서의 구조적 정보를 추출할 수 있는 파서(Parser)를 사용하여 문서 구조를 추출하고, 추출한 문서 구조를 원핫 인코딩(One hot encoding) 등의 방식으로 벡터화(Vectorization)하여 기계 학습에 사용한다.

2.1.3 기계 학습(Machine Learning)


본 프로젝트는 기계 학습으로 악성 PDF와 악성 MS Word를 탐지하는 모델을 제작하고자 한다. 기계 학습 방법으로 지도 학습을 사용한다.

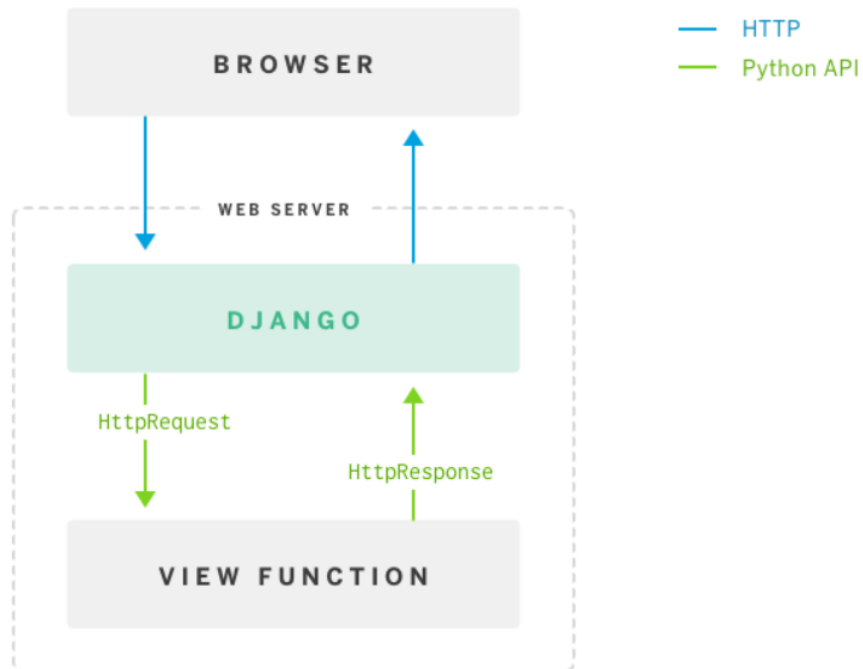
수집한 데이터를 라벨링하여 정상 파일과 악성 파일로 분류하고, 특징 추출 과정으로 데이터를 벡터화를 한 뒤 기계 학습에 사용한다. 기계 학습 알고리즘으로 SVM(Support Vector Machine), 랜덤 포레스트(Random Forest), 그라디언트 부스팅 결정 트리(Gradient Boosting Decision Tree), DNN(Deep Neural Network) 등 다양한 알고리즘을 사용하여 실험을 진행해서 가장 좋은 성능의 알고리즘을 사용하고자 한다.

2.1.4 웹 서버(Web server)

시나리오를 위한 기본적인 웹 서버를 구현한다.

본 프로젝트에서 개발한 엔진을 웹 서버에 확장한 상황을 가정한다. 웹 서버는 파일 업로드가 가능한 게시판 형태로 구현한다. 파일이 업로드 되었을 때 정상 파일이라면 정상적으로 업로드 되었음을 클라이언트에 전달하여야 한다. 또한 업로드 된 파일을 저장하기 위해 DB 연동이 되어야 하며 장고(Django)를 사용하여 웹 서버와 DB를 연동할 계획이다.


| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |



<그림 2> 웹 서비스 구조

(출처 =

https://blog.heroku.com/in_deep_with_django_channels_the_future_of_real_time_apps_in_django)

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |


2.2 수행내용

2.2.1 데이터 수집

기계 학습 기반의 문서형 악성코드 탐지 모델을 제작하기 위해 문서형 악성코드 데이터가 필요하다. 필요에 따라 문서형 악성코드 데이터를 공유하는 사이트에서 데이터를 수집하는 크롤러를 제작했다. 데이터를 수집한 사이트 목록은 다음 표와 같다.

<표 1> 데이터를 제공하는 사이트 목록
출처 = <https://chogar.blog.me/80212372093>

| 사이트 이름 | 사이트 주소 |
|--|---|
| Contagiodump | http://contagiodump.blogspot.kr |
| Kernelmode | http://www.kernelmode.info/forum/viewforum.php?f=16 |
| Malshare | http://malshare.com |
| AVCaesar | http://avcaesar.malware.lu |
| Malwareblacklist | http://www.malwareblacklist.com/showMDL.php |
| Malwr | https://malwr.com |
| Minotaur | http://minotauranalysis.com/exetweet |
| Openmalware | http://openmalware.org |
| Secuboxlabs | http://secuboxlabs.fr |
| Virusign | http://www.virusign.com |
| Virusshare | http://virusshare.com |
| Google | http://www.google.com |

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

2.2.2 데이터 라벨링


2018년 AV-TEST 성능 평가에서 수상한 3개의 안티바이러스를 선정하여 해당 안티바이러스의 탐지 결과를 데이터 라벨링에 사용하였다. 선정한 3개의 안티바이러스는 에프시큐어(F-Secure), 카스퍼스키(Kaspersky), 시만텍(Symantec)이다. 이 3개의 안티바이러스는 AV-TEST 성능 평가에서 일반 사용자 부문, 기업용 부문 모두에서 수상한 안티바이러스다.



<그림 3> AV-TEST 성능 부문 평가에서 수상한 안티바이러스

출처 = <https://www.av-test.org/en/news/av-test-awards-this-is-the-elite-class-of-it-security-2018/>

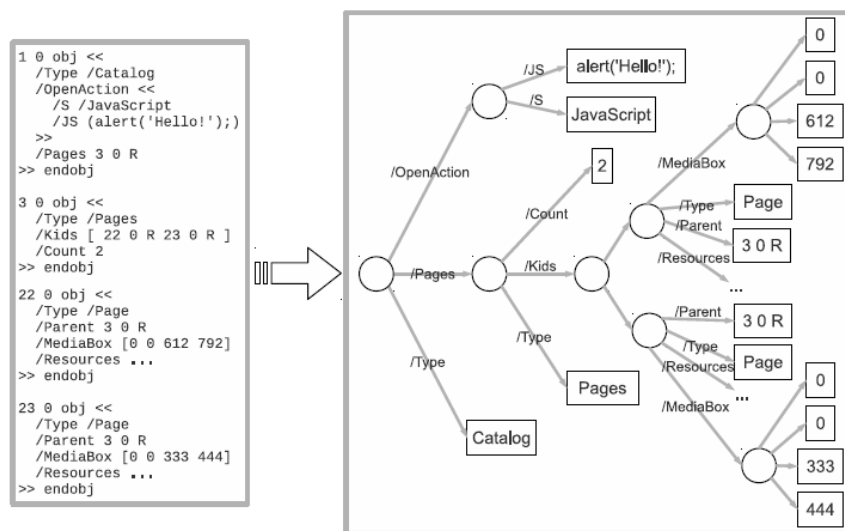
바이러스토탈에서 라벨링 할 파일의 결과 리포트를 바이러스토탈 API를 사용해서 받은 다음, 위에서 선정한 3개의 안티바이러스 중 하나 이상의 안티바이러스가 악성이라고 한다면 악성으로, 바이러스토탈에 등록된 모든 안티바이러스가 정상으로 라벨링 하였다.

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time Malicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

2.2.3 특징 추출

PDF와 MS Word 파일의 특징을 추출하여 고정된 크기의 벡터를 생성하고, 생성한 벡터를 기계 학습에 사용한다.

2.2.3.1. PDF



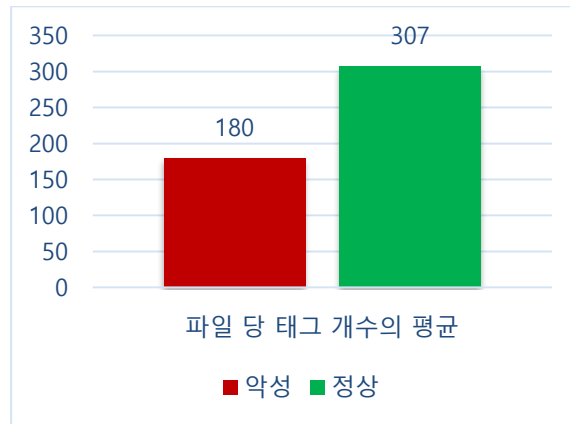
<그림 4> PDF 구조; 물리적인 구조(왼쪽), 논리적인 구조(오른쪽)

출처 = Nedim Šrncić and Pavel Laskov. Detection of Malicious PDF Files Based on Hierarchical Document Structure. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2013*

PDF 구조는 <그림 4> 과 같이, /Type, /Page 같은 다양한 태그(Tag)를 갖는 형태이다. 태그 내부에는 태그와 관련된 내용이 있다.

우리는 악성 PDF가 갖는 태그 정보와 정상 PDF가 갖는 태그 정보가 다르다는 것을 확인하였는데, <그래프 1> 과 같이 악성 PDF 에서 등장한 태그 종류는 평균 180개, 정상 PDF 에서 등장한 태그 수는 평균 307개가 등장한다는 것을 확인했다.

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |




<그래프 1> 파일 당 등장하는 태그 개수의 평균

그리고 <표 2>를 통해서 악성 PDF와 정상 PDF에서 다른 특징을 확인 할 수 있다. <표 2>는 자주 등장하는 태그의 상위 16개를 나타낸 표이다.

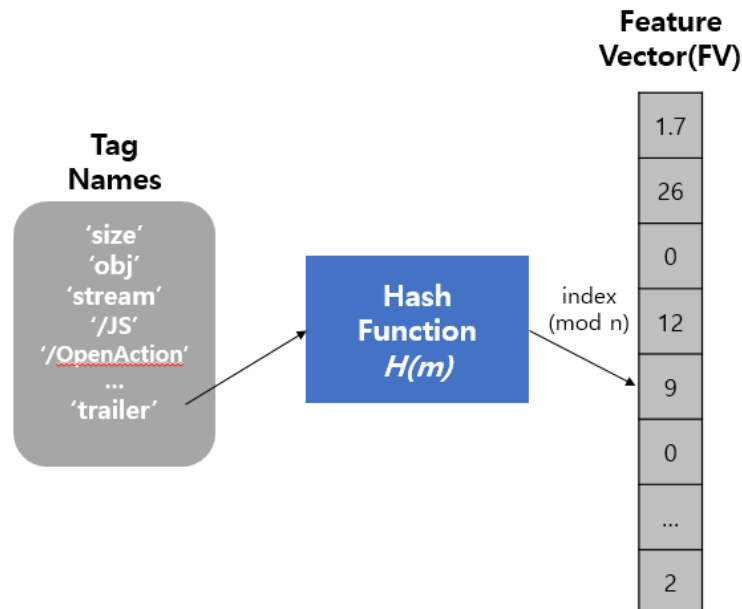
<표 2> 악성 PDF 태그 정보(왼쪽), 정상 PDF 태그 정보(오른쪽)

| Tag Name | Count | Tag Name | Count |
|-----------|-----------|--------------|------------|
| /URI | 8,204,328 | obj | 16,470,754 |
| endobj | 5,930,353 | endobj | 16,462,865 |
| obj | 5,226,287 | /Type | 7,882,414 |
| /Type | 5,054,637 | /Length | 6,351,320 |
| /www | 4,407,113 | stream | 6,342,069 |
| /A | 4,405,322 | endstream | 6,280,863 |
| /Subtype | 4,355,685 | /Filter | 5,687,772 |
| /S | 4,355,685 | /FlateDecode | 5,091,152 |
| /Contents | 4,348,565 | /Subtype | 4,090,938 |
| /I | 4,112,098 | /xmpG | 3,645,833 |
| /M | 4,111,108 | /S | 3,579,119 |
| /P | 4,109,853 | /rdf | 3,579,119 |
| /H | 4,109,502 | /XObject | 2,680,936 |
| /F | 4,109,445 | /Font | 2,370,243 |
| /Rect | 4,102,544 | /P | 2,243,650 |
| /Border | 4,102,524 | /stEvt | 2,206,310 |

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

이처럼 악성 PDF에서 등장하는 태그 종류와 개수가 정상 PDF와 다른 양상을 보이기 때문에 태그 정보로 해싱 트릭(Hashing Trick)을 사용하면 악성 PDF에서 생성한 특징 벡터(Feature Vector)와 정상 PDF에서 생성한 특징 벡터가 다른 양상을 가질 것이라 판단하였다. 따라서 PDF는 태그 정보를 해싱 트릭으로 특징 벡터를 생성하였다.


해싱 트릭이란 고차원의 특징을 저차원으로 투영시킬 수 있는 방법으로, <그림 5>와 같이 해시 함수를 사용하여 한 특징의 해시 값을 구하고 그 값을 특징 벡터 크기로 나눈 값을 인덱스(Index)로 하여 그 인덱스에 값을 누계하는 방법이다.



<그림 5> 해싱 트릭

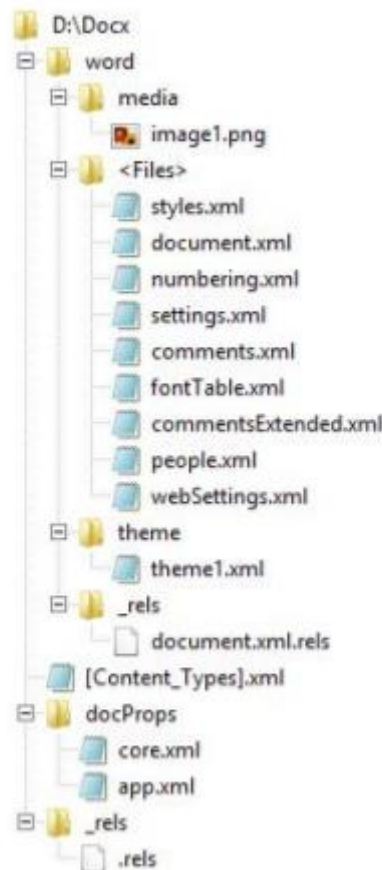
$$\text{FEATURE_VECTOR}[h(\text{tag name}) \bmod \text{SIZE_OF_FV}] += \text{Num_of_tag_name}$$

출처 = Kilian Weinberger KILIAN, Anirban Dasgupta ANIRBAN, John Langford et.al. *Feature Hashing for Large Scale Multitask Learning*. Proc. ICML 2009.

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |


2.2.3.2. MS Word

MS Word의 구조는 <그림 6>와 같다. <참고 논문 ALDOCX>에 따르면 저자는 MS Word 파일의 구조 중 내부 파일 경로를 사용하여 악성 파일과 정상 파일을 구분할 수 있다고 주장한다. 그 이유는 악성코드 제작자가 문서에 악성 매크로나 콘텐츠를 추가하면 새로운 경로나 파일이 추가되어 정상 문서의 구조와 차이가 생기기 때문이라고 한다. 따라서 본 프로젝트는 이 특징을 특징(Feature)로 사용하여 MS Word 파일의 악성 여부를 탐지해보고자 했다.



<그림 6> DOCX 내부 구조

출처=NissimN,CohenA,EloviciY. *ALDOCX: Detection of Unknown Malicious Microsoft Office Documents Using Designated Active Learning Methods Based on New Structural Feature Extraction Methodology*. IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 12, NO. 3, MARCH 2017,

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

$$DF = \frac{\text{단어가 나타난 문서 수}}{\text{전체 문서 수}}$$

특징 추출은 MS Word 파일의 내부 파일 경로를 경로 구분자로 토큰화(tokenization) 하고 DF(Document Frequency)를 사용하여 토큰별 DF 값을 구한 다음, 특징 추출을 진행하여 특징 벡터를 생성하였다. 예를 들어, 파일 내부 경로로 "word\media\image1.png" 가 주어진다면 ["word", "word\media", "word\media\image1.png", ...] 로 토큰화하고 각 토큰의 DF 값을 구하여 그 값을 특징 벡터 요소로 사용하였다.

그리고 논문에서 사용한 특징에 내부 파일의 엔트로피(Entropy), 내부 파일의 크기를 추가해서 특징 벡터를 생성했다. <그림 6>을 보면 DOCX 파일 내에 PNG, XML 등 여러 파일로 구성된 것을 확인할 수 있다. 그래서 각 파일의 엔트로피를 측정하여 엔트로피의 최댓값, 최솟값, 평균값을 특징 벡터에 추가하였다. 마찬가지로 파일 크기의 최댓값, 최솟값, 평균값을 특징 벡터에 추가했다. 수정한 특징 벡터로 학습을 진행해 본 결과 기존 방법의 특징 벡터로 학습했던 것보다 검증 성능이 향상된 것을 확인하였다.

2.2.4 기계 학습


특징 추출 과정에서 생성한 특징 벡터를 Scikit learn, LightGBM 라이브러리를 사용하여 기계 학습을 진행했다. 사용한 모델은 LR(Linear Regression), DT(Decision Tree), SVM, NB(Naïve Bayes), KNN(k-Nearest Neighbor Algorithm), RF(Random Forest), LightGBM 이다.

여러 알고리즘으로 기계 학습을 진행하고 테스트를 진행하여 알고리즘마다의 성능을 확인해 보았다. 테스트 결과에 사용한 성능 지표는 다음과 같다.

1. 정확도(Accuracy)

계산된 값이 실제 값과 얼마나 가까운지 나타내는 척도이다.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}}$$

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

2. F1-Score

F1-Score 는 Recall 과 Precision을 이용하여 조화 평균을 이용한 척도이다.

$$F1 \text{ Score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

F1 Score 는 악성 데이터와 정상 데이터 간의 불균형을 이루었을 때 성능을 확인할 수 있는 지표이다.

2.2.4.1. PDF

데이터는 2013년부터 2018년의 데이터로 구성했다. 악성 PDF 75,000개, 정상 PDF 120,000개를 학습에 사용했다. 그 중 학습으로 사용한 데이터는 악성 PDF 65,000개, 정상 PDF 110,000개를 사용하였고 검증 데이터는 악성 PDF 10,000개, 정상 PDF 10,000개로 사용했다.

<표 3> PDF의 모델 별 탐지 성능 지표


| | LR | DT | SVM | NB | KNN | RF | LGB |
|----------|-------|-------|-------|-------|-------|-------|-------|
| Acc | 0.639 | 0.983 | 0.974 | 0.838 | 0.946 | 0.976 | 0.993 |
| F1-score | 0.39 | 0.982 | 0.972 | 0.833 | 0.939 | 0.974 | 0.992 |

본 프로젝트 엔진과 유사한 클램에이브이(ClamAV) 와 악성 PDF 탐지 성능을 비교해보았다. 테스트 샘플(Sample)은 2018년 악성 PDF 500개를 대상으로 하였다. 결과는 <그림 7> 와 같이 클램에이브이는 500개 중 16개를 탐지했고, 드림은 500개 중 438개를 탐지했다.



[2018년도 문서형 악성 코드 500개]

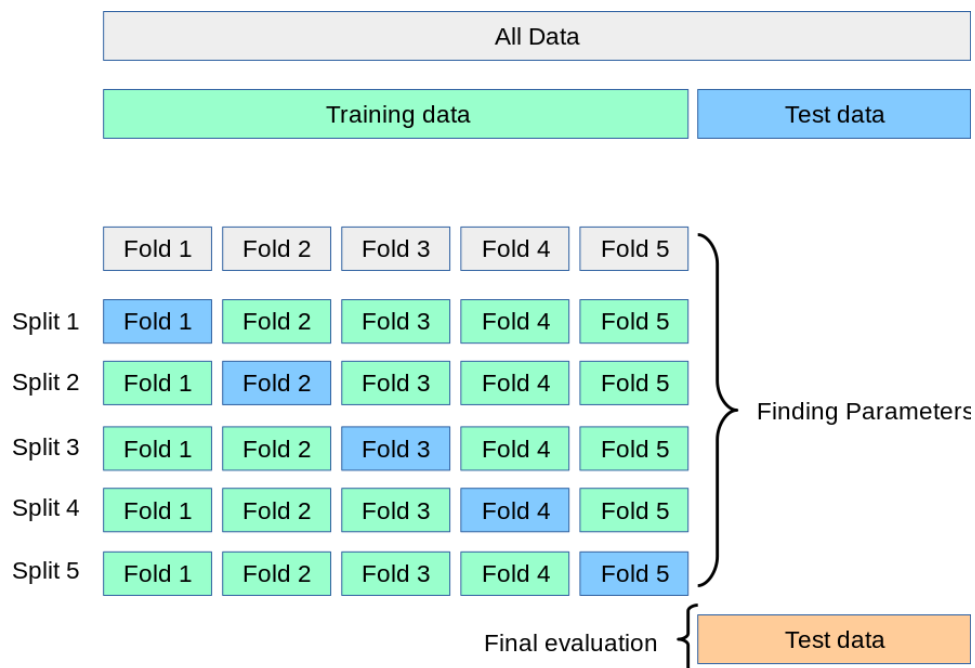
<그림 7> 클램에이브이와 탐지 성능 비교

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

2.2.4.2. MS Word

학습에 사용한 데이터는 악성 MS Word 3,428개, 정상 MS Word 300개이다. MS Word는 10 폴드 교차 검증(10-Fold Cross Validation)으로 성능을 확인했다.

K 폴드 교차 검증이란 K개의 폴드를 만들어서 진행하는 교차 검증이다. 데이터 셋이 적은 경우 테스트 셋에 대한 성능 평가의 신뢰성이 떨어지기 때문에 데이터를 K개의 폴드로 나누어 학습과 검증을 진행하는 방식이다.



<그림 8> 5 폴드 교차 검증

출처 = https://scikit-learn.org/stable/modules/cross_validation.html

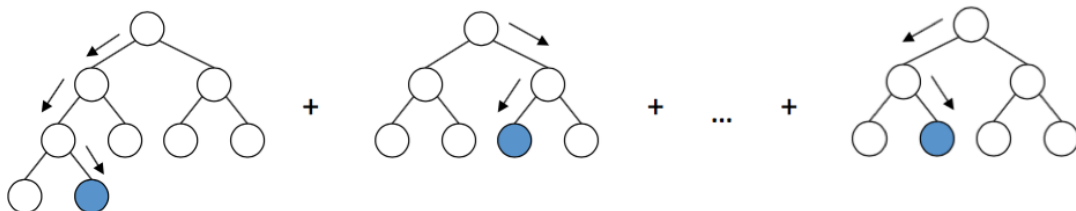
<표 4> MS Word의 모델 별 교차 검증 점수

| | LR | DT | SVM | NB | KNN | RF | LGB |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 기존 방법 | 0.919 | 0.921 | 0.613 | 0.613 | 0.902 | 0.92 | 0.861 |
| 제안 방법 | 0.925 | 0.957 | 0.928 | 0.331 | 0.948 | 0.963 | 0.984 |

<표 4>는 논문에서 제시한 방법으로 특징 벡터를 생성하여 검증한 결과와 본 프로젝트에서 연구 중인 방법으로 특징 벡터를 생성하여 검증을 진행한 결과이다. 기존 방법보다 본 프로젝트에서 사용한 방법이 대부분의 학습 모델에서 향상된 결과를 보였다.

2.2.4.3. 결과 분석


PDF, MS Word 결과에서 가장 좋은 성능을 보이는 알고리즘은 LightGBM이다. LightGBM은 그라디언트 부스팅 결정 트리(Gradient Boosting Decision Tree, GBDT) 기반 알고리즘이다. LightGBM은 결정 트리를 순차적으로 훈련하는 앙상블(Ensemble) 모형으로, 반복마다 음의 기울기(잔차 오차)에 적합하여 의사 결정 트리를 학습한다.

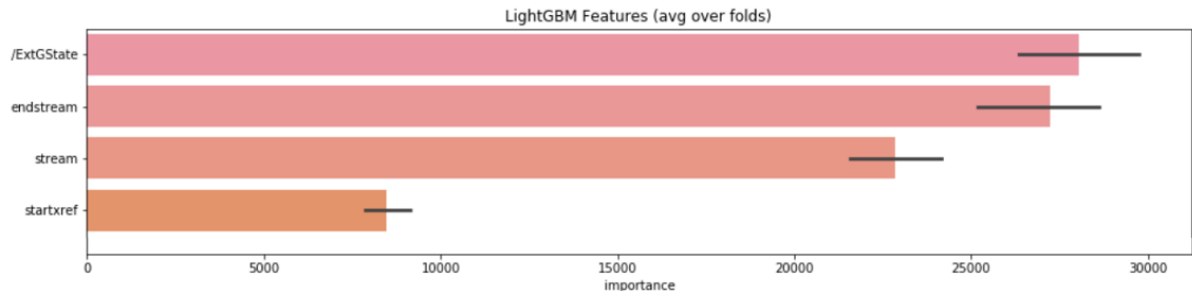


<그림 9> LightGBM 구조

출처 = http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html

<그림 9>는 LightGBM의 모델 구조이다. 최종적으로 산출된 악성일 확률값을 특정 임계값(threshold)을 정하여 악성과 정상을 구분한다. LightGBM은 데이터의 수가 적어도 어느 정도 성능을 보장하고 각 특징별 결과 기여도를 볼 수 있는 장점이 있다. 특징 기여도는 <그림 10>와 같다. (출처 = Guolin Ke, Qi Meng, Thomas Finley et.al, LightGBM: A Highly Efficient Gradient Boosting Decision Tree)

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |



<그림 10> LightGBM 에서 확인할 수 있는 특징 기여도 예제(PDF 특징 상위 4개)

2.2.5 엔진

악성 PDF와 MS Word를 탐지할 수 있는 엔진을 제작했다. 사용자는 엔진에서 제공하는 API를 사용하여 엔진을 사용하거나 애플리케이션과 엔진을 연동하여 사용할 수 있다.

엔진 서버는 Flask를 사용하여 제작했고 uWSGI 와 Nginx를 추가하여 다른 애플리케이션과의 연결을 용이하게 하였다 uWSGI와 Nginx를 추가한 이유는 다음과 같다.

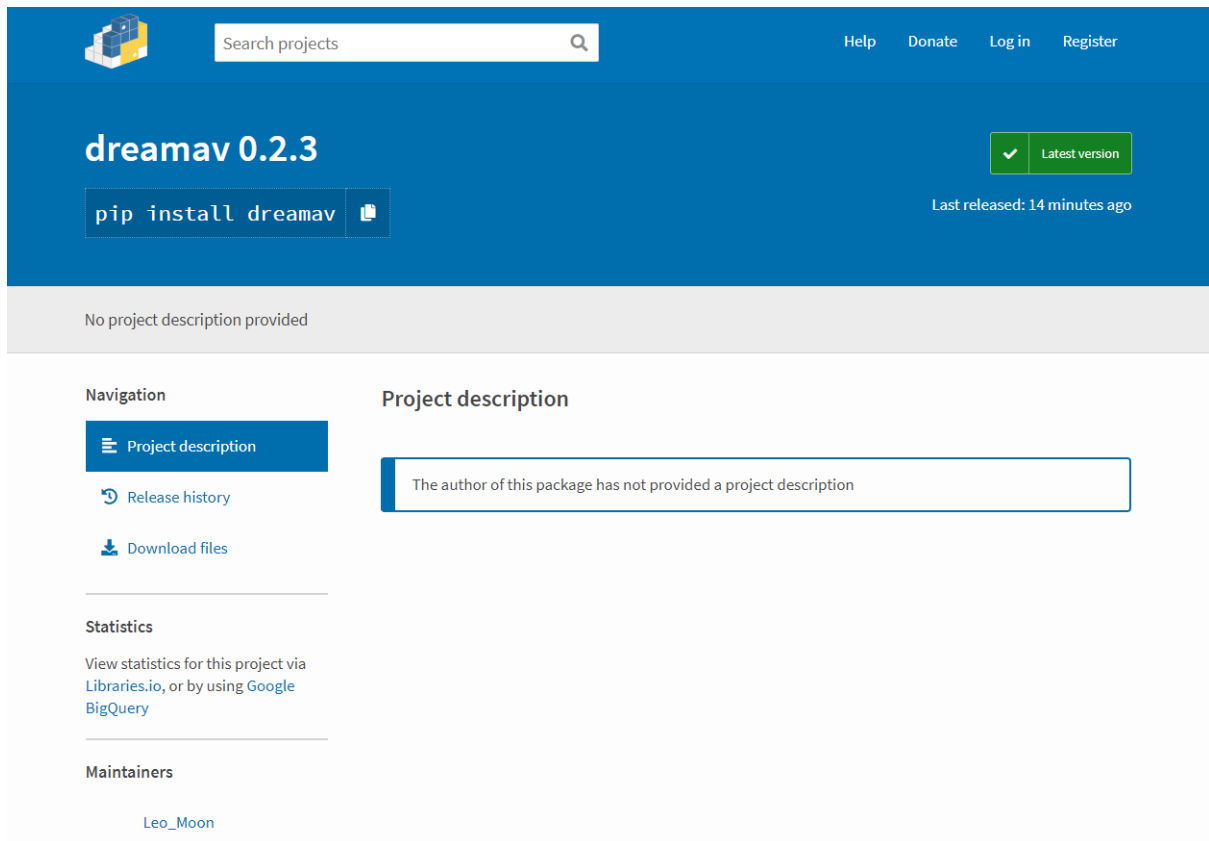
1. 가상 호스트(Virtual Host) 관련 문제 해결에 용이하다.
2. 설정 파일(Configuration)을 관리하는 데 편하다.
3. 정적 파일 서빙(Static File Serving)과 유저별 설정 파일 상세 설정이 용이하다.
4. 트래픽이 많아졌을 경우 필요에 따라 업 스케일링이 쉽다.

현재 엔진에서 사용할 수 있는 API는 다음과 같다.


1. Submit : PDF나 MS Word 파일이 악성 파일인지 아닌지 검사한다.

|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
|--|-------------------------|---|-------------|
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

엔진 패키지를 PyPI에 등록하여 여러 사용자들이 사용할 수 있도록 배포하고 있다.



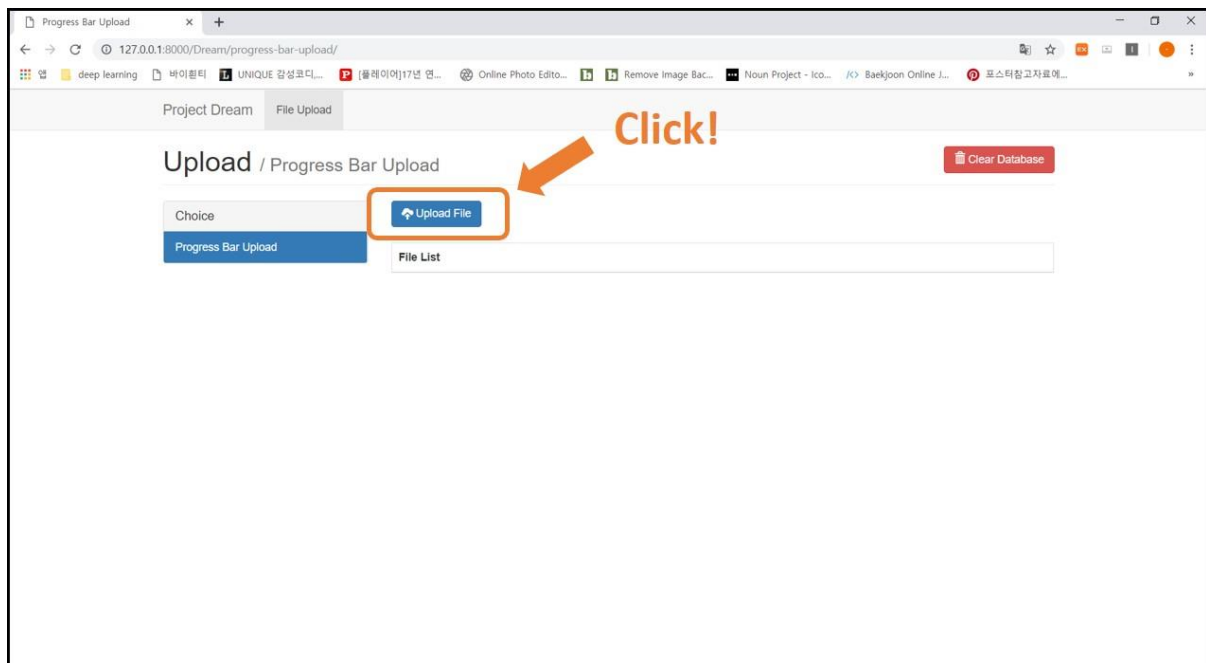
<그림 11> PyPI 에 등록된 엔진 패키지 화면
<https://pypi.org/project/dreamav/#description>

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time MAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

2.2.6 웹 서버

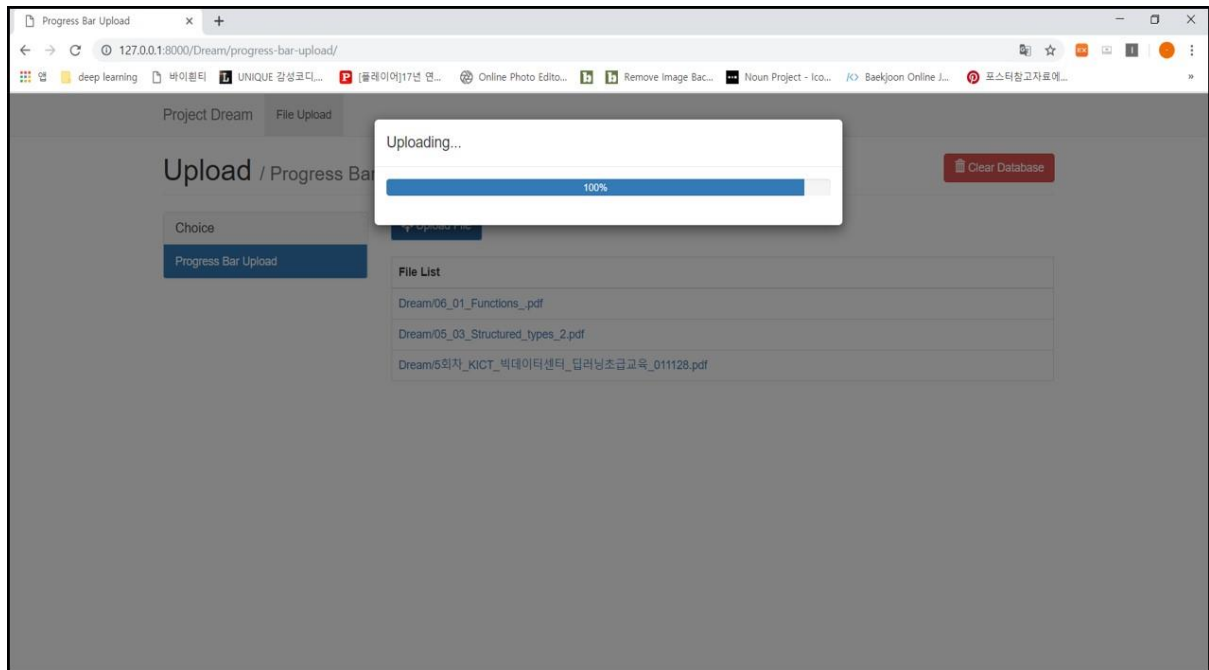
시나리오를 위한 기본적인 웹 서버를 구현했다.

본 프로젝트에서 개발한 엔진을 웹 서버에 확장한 상황을 가정한다. 웹 서버는 파일 업로드가 가능한 형태로 구현했다. <그림 12>과 같이 파일을 업로드 할 수 있는 버튼을 구현하였으며 파일을 업로드 할 시 <그림 13>과 같은 상태 창이 나타난다. 파일이 업로드 되었을 때 정상 파일이라면 정상적으로 업로드가 되었음을 클라이언트에 전달 한다. 전송이 완료된 후 파일 목록에 정상 업로드 된 파일 이름이 추가된다. 업로드 된 파일을 저장하기 위해 장고의 기본 DB를 사용하였다.



<그림 12> 웹 서버 화면 1

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |



<그림 13> 웹 서버 화면 2

3 수정된 연구내용 및 추진 방향

3.1 수정사항

3.1.1 웹

엔진 사용 시나리오를 위해 웹 게시판에서 게시글을 올릴 때 문서형 악성코드를 첨부하여 게시글을 작성할 때, 게시글 작성을 제한하는 형태를 구현하고자 하였다.

본 프로젝트의 목적은 웹 서버, 메일 서버, 백신 등 다양한 환경에서 사용 가능한 확장성 있는 문서형 악성코드 탐지 엔진을 구현하는 것이다. 이에 따라 게시판과 같은 구조보다는 파일 업로드를 할 수 있는 기능이 있는 웹으로 수정하였고, 문서형 파일이 업로드 될 때 해당 파일을 엔진에 전송한 뒤, 엔진에서 도출한 결과를 받아 악성 여부를 판단하여 해당 파일의 업로드를 제한하는 것으로 수정하였다.

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

3.1.2 데이터 라벨링

기존에 선정했던 기준은 AV-TEST 탐지 성능 부문 2019년 2월 평가에 해당하는 기준으로 일반 사용자가 사용할 수 있는 안티바이러스만 나타낸 것이다. 이 때 몇 가지 문제점이 발생할 수 있다. 첫 번째는 특정 기간에 한정될 수 있다는 문제점이 있다. 두 번째는 기업을 대상으로 하는 안티바이러스는 제외되어 객관적인 평가가 부족할 수 있다는 문제점이다. 마지막으로 선정한 5개의 안티바이러스 모두가 악성이라고 한다면 악성이라고 라벨링했으나, 이 경우 한 악성 파일에 대해서 한 안티바이러스가 악성이라 하여도 나머지가 정상이라 한다면 미탐이 된다. 따라서 이 문제점들을 해결하고자 2018년 AV-TEST 탐지 성능 평가 중 기업용과 일반 사용자용 안티바이러스 모두에서 수상한 카스퍼스키, 시만텍, 에프시큐어의 탐지 결과를 사용하여 하나 이상의 안티바이러스가 악성이라 한다면 악성이라고 라벨링 하는 것으로 수정했다.

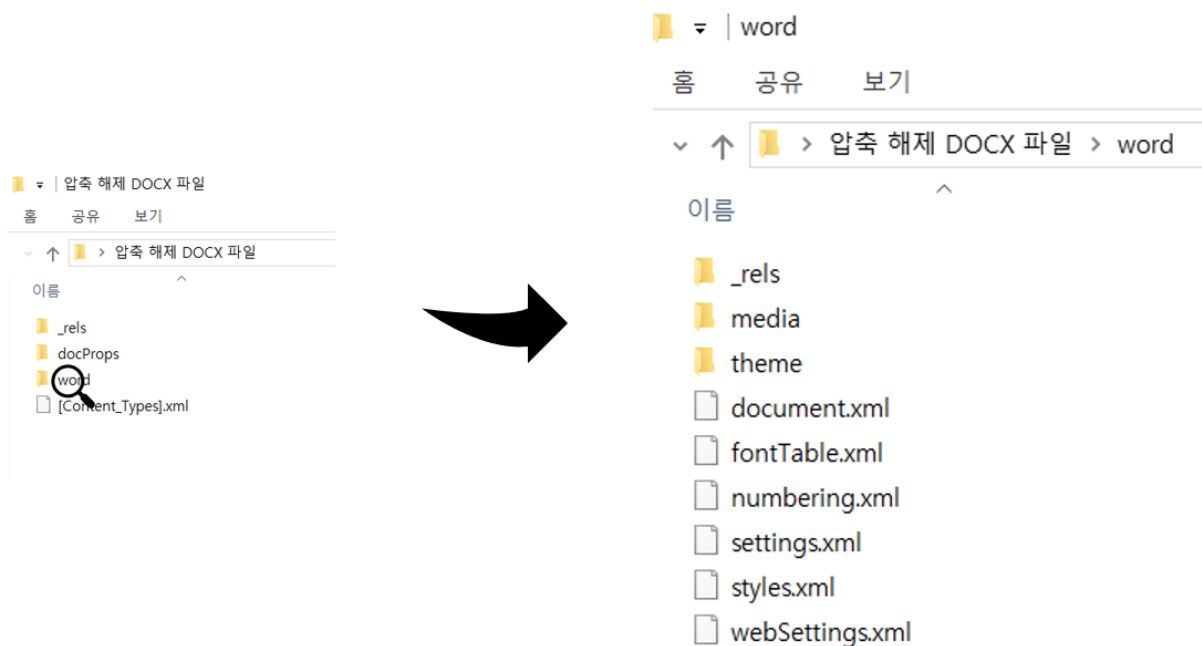
| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

4 향후 추진계획

4.1 향후 계획의 세부 내용


4.1.1 특징 추출

4.1.1.1. MS Word



<그림 14> DOCX 파일의 내부 구조

현재는 MS Word 파일을 압축 해제하고 내부 파일의 경로를 특징으로 사용하고 있다. 이 경우 내용을 확인해서 탐지해야하는 매크로 악성 파일을 미탐하는 경우를 확인했다. 따라서 MS Word의 매크로에서 비주얼 베이직 코드를 추출해서 추출한 코드를 벡터화하여 기계 학습을 진행한 결과를 확인해보려 한다.

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

4.1.1.2. PDF

현재 특징 벡터를 생성하는 방식은 해싱 트리를 사용하여 특징을 생성한다. 기계 학습에서는 특징들의 위치뿐만 아니라 해당하는 특징 값이 어떤 값을 갖는지도 중요하다. 따라서 단순히 해당 원소에 누계만 하는 방식이 아닌 다른 방법을 추가하여 벡터화한다면 일반화 성능이 향상될 것으로 기대한다.

또는 특징 벡터 크기를 변화시키며 해싱 트리로 특징 벡터를 생성하고 기계 학습을 진행한 결과를 확인하여 최적의 특징 벡터 크기를 설정하고자 한다.

4.1.2 기계 학습

4.1.2.1. 학습 알고리즘 실험

현재까지는 기계 학습 알고리즘으로 실험을 진행하였다. 다음으로는 딥 러닝(Deep learning)으로 실험을 진행한 결과를 확인해보고자 한다. 그래서 기계 학습 알고리즘과 딥 러닝 중 높은 성능을 보이는 알고리즘을 사용하거나 높은 성능의 알고리즘을 앙상블 하여 최종 문서형 악성코드 탐지 분류기로 사용하고자 한다.

4.1.2.2. 하이퍼 파라미터(Hyper parameter) 실험

하이퍼 파라미터란 학습 중 자동으로 변하는 변수가 아닌 사람이 직접 튜닝해줘야 하는 변수를 말한다. 하이퍼 파라미터의 예시로는 학습률(learning rate), 비용 함수(cost function), 규제(regularization), 파라미터(parameter), 미니배치(mini-batch), 학습 반복 횟수, 은닉 개체(hidden unit) 개수, 가중치 초기화(weight initialization) 등이 있다.

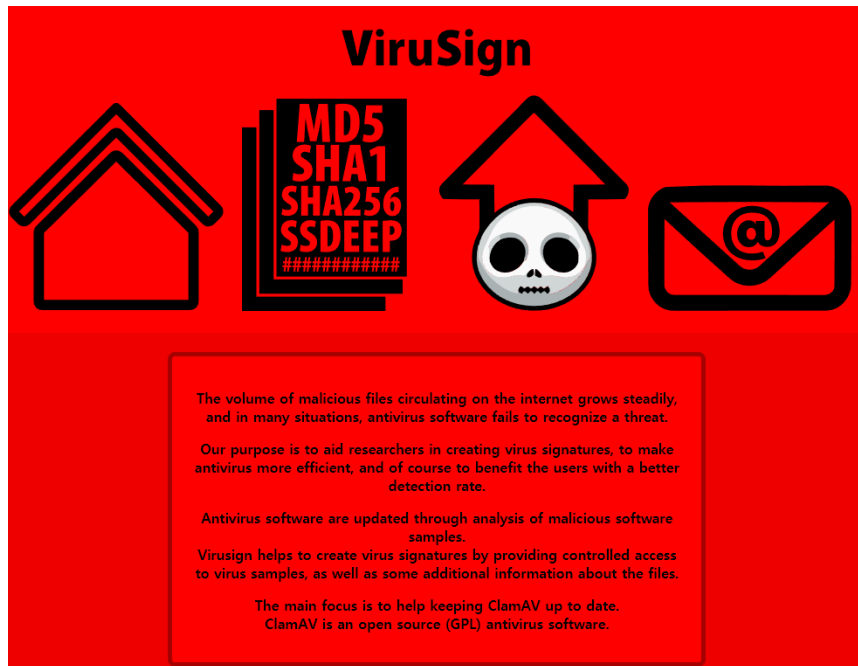
하이퍼 파라미터에 의해 기계 학습 모델의 성능이 크게 변할 수 있다. 따라서 가장 높은 성능을 보이는 알고리즘을 선택한 뒤, 반복적인 실험을 통해 최적의 하이퍼 파라미터를 구하고자 한다.

하이퍼 파라미터를 설정하는 방법으로는 그리드 탐색(Grid Search), 무작위 탐색(Random Search) 이 있다. 두 방법을 사용하여 하이퍼 파라미터를 수정한 뒤, K 폴드 교차 검증을 하여 하이퍼 파라미터별 모델 성능 평가를 하고자 한다.

|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
|--|-------------------------|---|-------------|
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

4.1.3 바이러스 공유 채널

악성코드 샘플을 제공하는 채널인 바이러사인(VirusSign)은 클램에이브이의 엔진을 최신 상태로 유지하는데 기여한다.




<그림 15> 바이러사인 화면

출처 = <https://www.virusign.com/about.php>

본 프로젝트 또한 문서형 악성코드를 공유하는 채널을 개설하여 엔진을 최신 상태로 유지하고, 사용자와 연구자들에게 공유하여 문서형 악성코드와 관련된 연구에 도움을 주고자 한다.

4.1.4 오픈소스 커뮤니티

엔진 사용자와 네트워크를 형성하여 엔진 발전에 기여하고자 슬랙을 운영하여 사용자와 아이디어를 공유한다.

| | | | |
|--|-------------------------|---|-------------|
|  국민대학교 소프트웨어학부 캡스톤 디자인 I | 중간보고서 | | |
| | 프로젝트 명 | DREAM(Detecting in Real-time mAlicious document using Machine Learning) | |
| | 팀 명 | Do it! | |
| | Confidential Restricted | Version 1.4 | 2019-APR-18 |

5 고충 및 건의사항

1. 학습에 필요한 데이터

- A. 학습을 위해 양질의 데이터가 필요했으나 현실적 요소로 데이터를 쉽게 구하지 못하였다.
- B. 본 프로젝트는 HWP 파일에 대해서도 진행해보려 했으나 HWP 파일은 국내에서만 사용하기 때문에 악성 HWP 파일 수집 자체가 어려웠다.

2. 관련 연구의 부족

- A. 문서형 악성코드 탐지와 관련된 최근 연구가 적어 참고할 수 있는 자료가 적었다. 특히 악성 HWP 탐지와 관련된 논문은 전무하다.