전사적 정적 정보 기반 악성코드 탐지

- S&D 팀 -

2018. 12. 1

호서대학교

1. Approach

- 기본적으로 정적 정보를 활용한 다양한 방법론들의 연계로 악성코드 탐지 정확도 상승 추구
- 한정된 데이터 환경에서 전체 데이터 구성에 영향을 덜 받는 k-NN 알고리즘을 악성코드 분류에 우선으로 가용하며 유사도 또한 3가지 방법으로 측정
  - ssdeep : 기본적인 유사도 측정 기법
  - tlsh : 일반적으로 ssdeep보다 coverage가 큰 유사도 측정 기법
  - dhash : packing 파일에 robust한 유사도 측정 기법

  ※ k-NN 기반 분류에서 시간 복잡도를 낮추기 위한 fast k-NN 적용
- 정적 분석에서 추출할 수 있는 다양한 feature들에 대한 통계적 해석
  - structure : PE Header에서 다수 파일에 대한 통계적 분석으로 feature 선별
  - dll/api : feature hashing을 통한 feature 선별
  - section/entropy : packing 유형 별 feature 선별

  ※ string 및 image feature는 현재의 feature 처리 정책으로는 성능이 좋지 않아 제외
- 다양한 feature들과 머신러닝 알고리즘의 연계
  - DNN, Ensemble(Bagging Tree), SVM, k-NN 알고리즘의 조합으로 최종 분류하는 정책으로 과적합 완화 및 탐지 성능 개선 추구
  - 모델 학습에는 앞서 분석한 다양한 feature들을 사용

2. k-NN - SSDEEP

- SSDEEP은 Fuzzy Hash를 사용하여 파일간의 유사도를 측정 가능
- 시험 결과, 두 파일 간의 SSDEEP 유사도 80 이상일 때 k-NN(k=1) 방법으로 악성코드를 분류하면 95% 이상의 정확도를 나타냄

Testset-1 대상 결과			
트레이닝		테스트	
7,000개		3,000개	
탐지율	오탐율	정확도	커버리지
99.51%	4.98%	98.77%	40.50%

Testset-2 대상 결과			
트레이닝		테스트	
5500만여개		10,000개	
탐지율	오탐율	정확도	커버리지
98.93%	14.52%	95.67%	43.18%

2. k-NN - TLSH

- TLSH는 Trend Micro 사에서 공개한 LSH(Locality Sensitive Hashing) 기법으로 파일간의 유사도를 측정 가능
- 시험 결과, 두 파일 간의 TLSH 유사도 40 이하일 때 k-NN(k=1) 방법으로 악성코드를 분류하면 97% 이상의 정확도를 나타냄

Testset-1 대상 결과			
트레이닝		테스트	
7,000개		3,000개	
탐지율	오탐율	정확도	커버리지
99.88%	6.61%	98.50%	35.50%

Testset-2 대상 결과			
트레이닝		테스트	
24만여개		10,000개	
탐지율	오탐율	정확도	커버리지
98.14%	4.53%	97.43%	45.48%

2. k-NN - DHASH

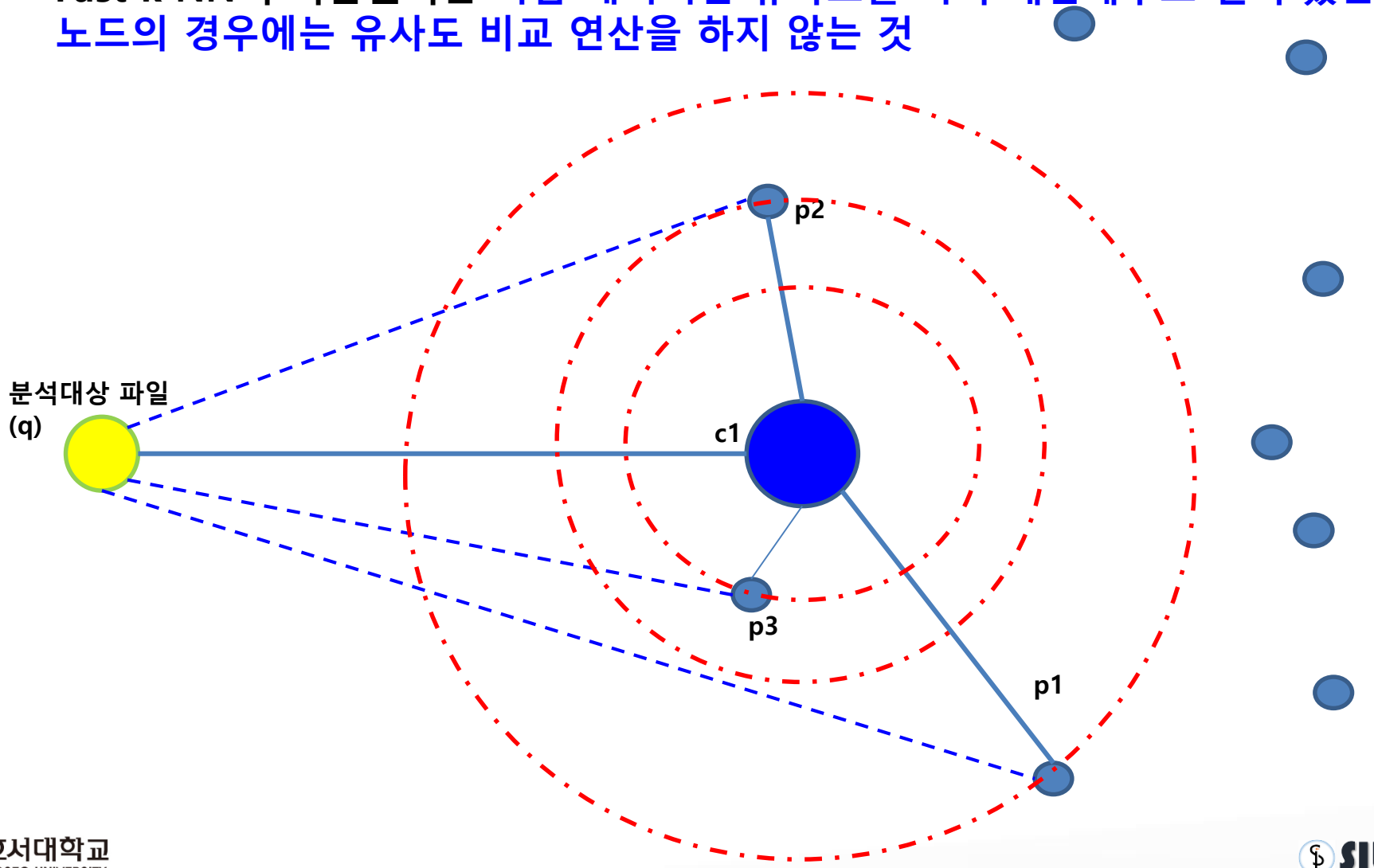
- DHASH는 이미지 파일을 LSH로 표현하는 기법으로 Hamming Distance를 사용하여 DHASH로 이미지 파일 간 유사도 측정 가능
- 시험 결과, 두 파일 간의 DHASH의 Hamming Distance가 1이하일 때 k-NN(k=1) 방법으로 악성코드를 분류하면 97% 이상의 정확도를 나타냄

Testset-3 대상 결과			
트레이닝		테스트	
152,963개		7,500개	
탐지율	오탐율	정확도	커버리지
99.94%	0%	99.95%	28.41%

Testset-2 대상 결과			
트레이닝		테스트	
282,420개		10,000개	
탐지율	오탐율	정확도	커버리지
96.26%	0.41%	97.15%	27.04%

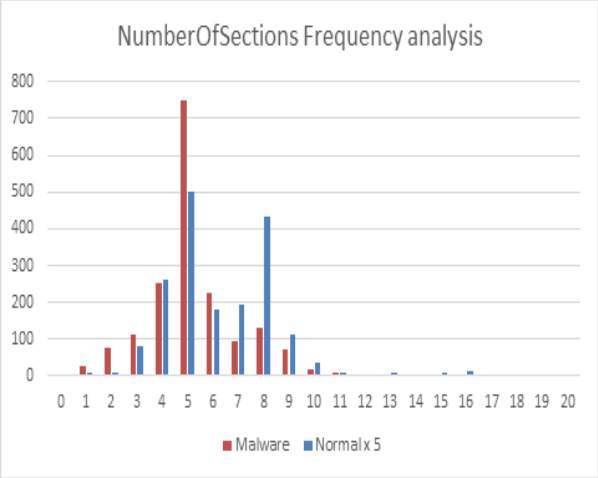
2. k-NN - Fast k-NN

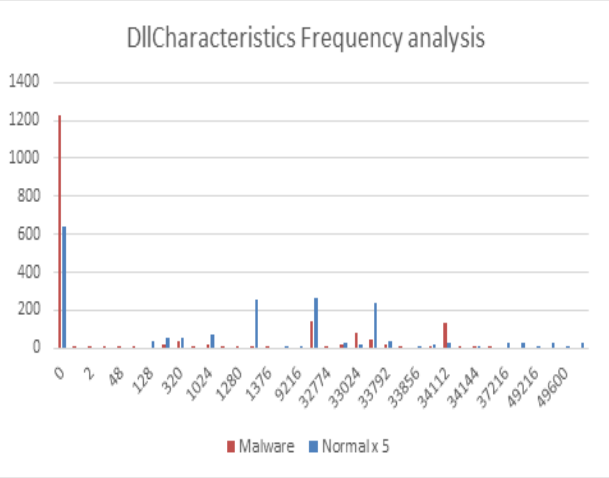
- Fast k-NN의 기본원리는 학습 데이터간 유사도를 미리 계산해두고 멀리 있는 노드의 경우에는 유사도 비교 연산을 하지 않는 것

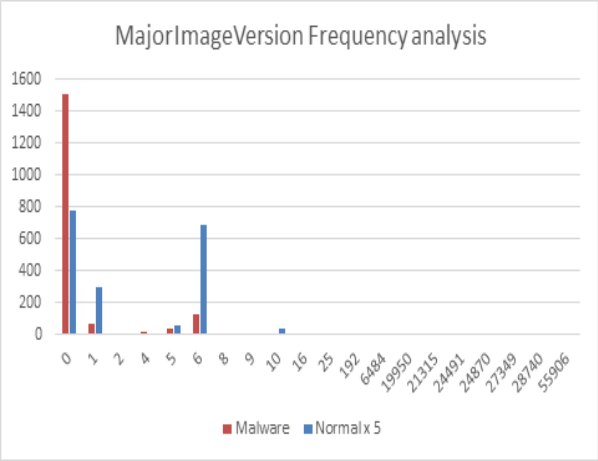


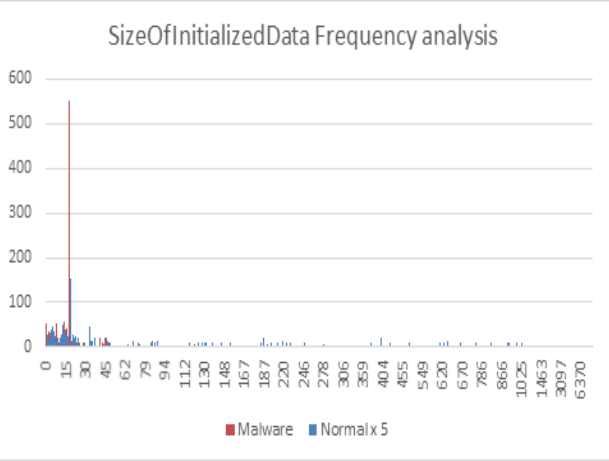
3. Features - Structure

- PE Header에서 다수 파일에 대한 통계적 분석으로 feature 선별
- 작년 대회 데이터 및 VXHeaven, Malshare 등에서 수집한 데이터 대상 분석



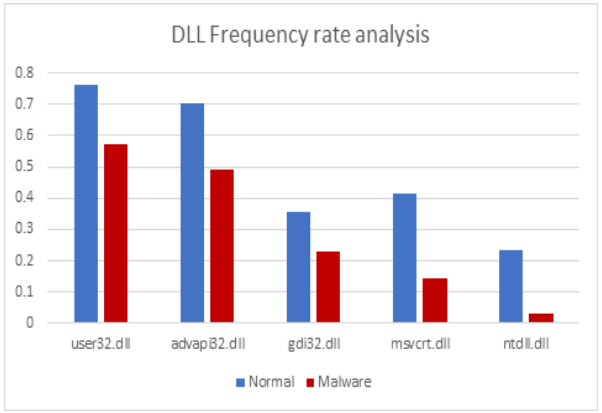


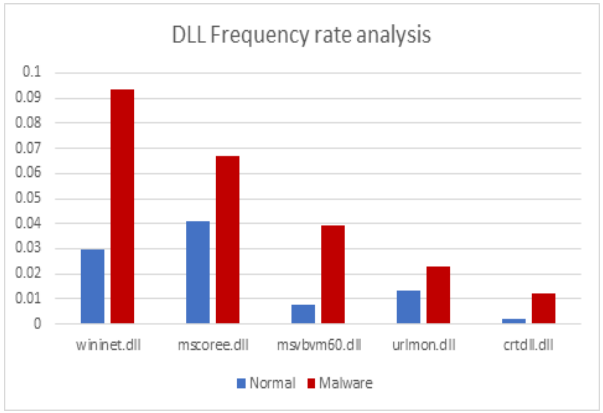


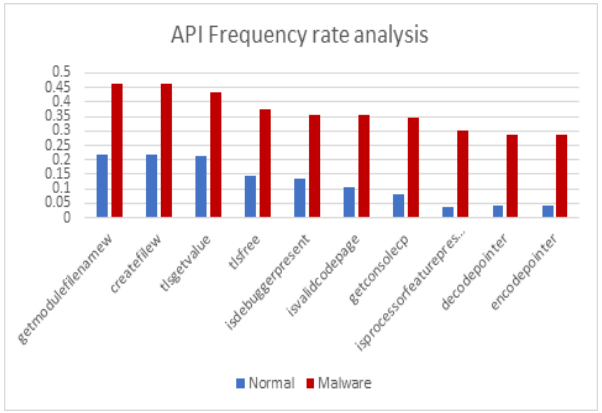


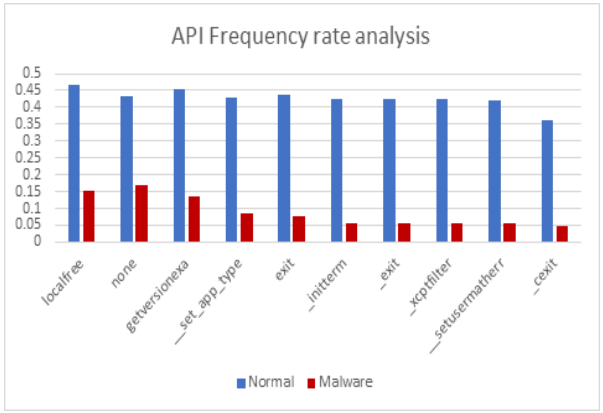
3. Features - DLL/API

- **Feature hashing을 통한 feature 선별**
- 작년 대회 데이터 및 VXHeaven, Malshare 등에서 수집한 데이터 대상 분석









3. Features - Section/Entropy(1/2)

- 파일을 같은 packer로 packing 하면 packer 내부의 정형화된 로직에 의해 section과 entropy 경향이 유사하게 나타나는 점을 고려하여 **packing 유형 별 feature 선별**
- **ASPack ➔ 기존파일이 가지고 있는 section에 .aspack_Section, .adata_Section 추가됨**

FileName	Packer	.text	.rdata	.data	.rsrc	.plugins	.aspack	.adata
Backdoor.Win32.BO2K.10		6.351372	3.467346	6.158735		5.305741		
Backdoor.Win32.BO2K.10.packed.ASPack.1	ASPack	7.991341	7.719718	7.916377		7.997089	5.875983	0

FileName	Packer	.text	.rdata	.data	.rsrc	.plugins	.aspack	.adata
Backdoor.Win32.BO2K.11.d		6.424001	4.879616	2.925431	6.307741			
Backdoor.Win32.BO2K.11.d.packed.ASPack.1	ASPack	7.996416	7.890525	7.859854	7.884233		4.385715	0

- **nspack ➔ 기존파일이 가지고 있는 section이 사라지고 .nsp0_Section, .nsp1_Section, .nsp2_Section으로 대체됨**

FileName	Packer	.text	.rdata	.data	.rsrc	.plugins	.nsp0	.nsp1	.nsp2
Backdoor.Win32.BO2K.10		6.351372	3.467346	6.158735		5.305741			
Backdoor.Win32.BO2K.10.packed.nspack.1	nspack						0	7.996616	0

FileName	Packer	.text	.rdata	.data	.rsrc	.plugins	.nsp0	.nsp1	.nsp2
Backdoor.Win32.BO2K.11.d		6.424001	4.879616	2.925431	6.307741				
Backdoor.Win32.BO2K.11.d.packed.nspack.1	nspack						0	7.944774	0

3. Features - Section/Entropy(2/2)

- **Pecompact ➔ 기존파일이 가지고 있는 section이 .text_Section, .rsrc_Section, .reloc_Section 으로 대체됨**

FileName	Packer	.text	.rdata	.data	.rsrc
Backdoor.Win32.BO2K.10		6.351372	3.467346	6.158735	
Backdoor.Win32.BO2K.10.packed.pecompact.1	pecompact	7.998372			7.6604936

FileName	Packer	.text	.rdata	.data	.rsrc	.reloc
Hoax.Win32.Renos.bz		6.104848	4.862032	0.999689	4.4068188	4.207680207
Hoax.Win32.Renos.bz.packed.pecompact.1	pecompact	7.965111			4.8947508	0.341311914

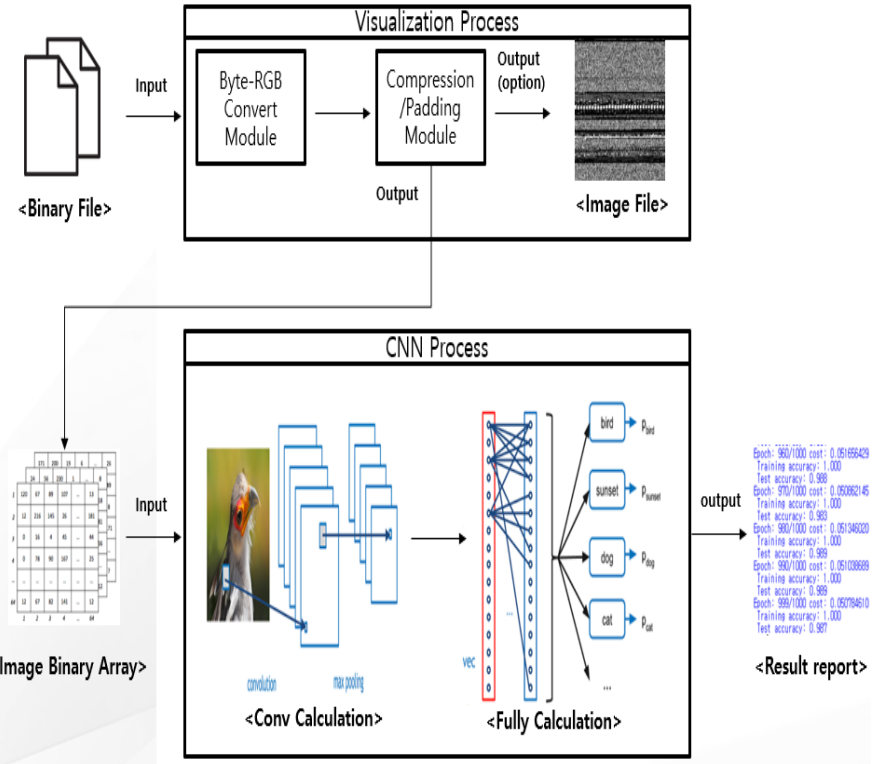
- **Petite ➔ 기존파일이 가지고 있는 section이 사라지고 .petite _Section, Nameless Section으로 대체됨**

FileName	Packer	.text	.rdata	.data	.petite	Nameless_
Backdoor.Win32.BO2K.10		6.351372	3.467346	6.158735		
Backdoor.Win32.BO2K.10.packed.petite.1	petite				4.306773	7.977052

FileName	Packer	.text	.rdata	.data	.petite	Nameless_
Trojan-Downloader.Win32.IstBar.ai		6.17151	4.437987	5.15677		
Trojan-Downloader.Win32.IstBar.ai.packed.petite.1	petite				0	4.731973

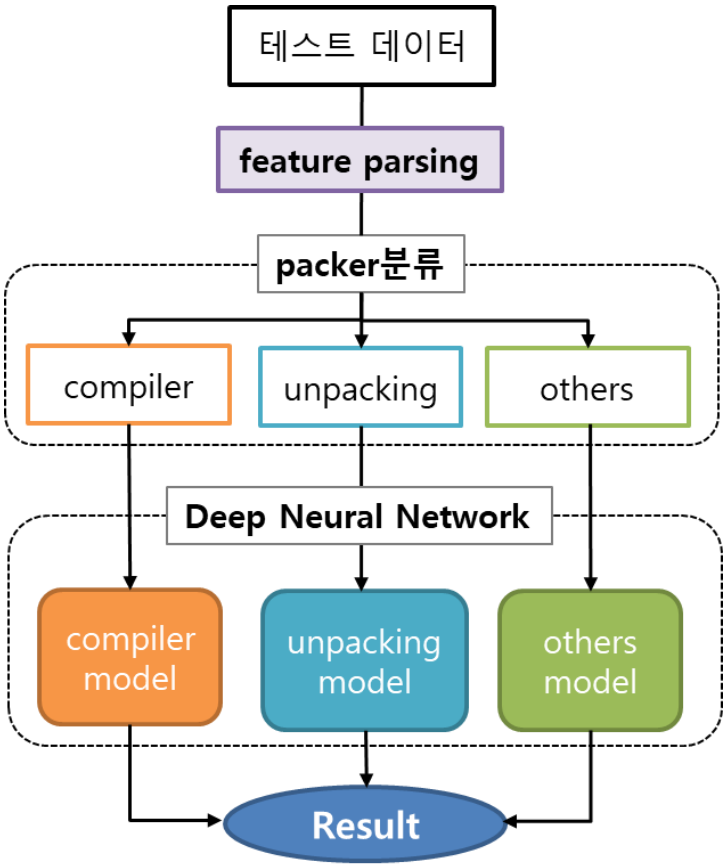
3. Features - String/Image

- String은 파일 내의 Symbol 문자나 문자열의 통계적 해석으로 feature 선별
- 바이너리를 image로 치환하고 CNN(Convolutional Neural Network)로 악성 코드 분류 가능
- 성능 이슈로 인해 대회에는 미반영



Symbol	Range	Malware Frequency	Benign Frequency
!	0	0.22%	3.20%
	1 ~ 30	10.56%	37.52%
	31 ~ 100	13.54%	20.44%
	101 ~ 800	50.62%	17.92%
	801 ~ 1000	4.64%	2.36%
	1000 ~	20.42%	18.56%
#	0	0.16%	7.52%
	1 ~ 30	6.34%	35.64%
	31 ~ 100	17.06%	19.04%
	101 ~ 800	51.42%	16.64%
	801 ~ 1000	5.14%	2.56%
	1000 ~	19.88%	18.60%
[	0	0.38%	2.28%
	1 ~ 30	5.02%	19.88%
	31 ~ 100	10.82%	21.64%
	101 ~ 800	54.34%	30.56%
	801 ~ 1000	5.18%	3.12%
	1000 ~	24.26%	22.52%
{	0	0.66%	12.68%
	1 ~ 30	0.36%	37.00%
	31 ~ 100	12.34%	15.32%
	101 ~ 800	13.12%	14.80%
	801 ~ 1000	48.98%	2.44%
	1000 ~	5.96%	17.76%

4. Machine Learning 알고리즘 – Deep Neural Network



- **DNN(Deep Neural Network)**
  - 학습데이터를 compiler, unpacking, others 로 분류하여 3가지 모델로 학습

→ 데이터 특징에 따라 별도의 학습모델을 생성하기 때문에 반대 성향의 데이터가 섞여 노이즈를 발생시키는 것을 방지

  - 549개 Feature와 40만여개의 학습데이터를 사용한 모델, 670개 Feature와 30만여개의 학습데이터를 사용한 모델 2종류 생성

→ 40만여개 모델 : 패킹된 파일 중점

→ 30만여개 모델 : 오탐율 하향 중점

5. 결론

- S&D 팀에서 진행한 전사적 정적 정보 및 AI 기반 분류의 핵심 특징 7가지

1. 데이터 경향성과 모델 과적합을 감안하여 k-NN 방법 적용
2. SSDEEP, TLSH 뿐 아니라, packing에 강하다고 알려진 DHASH 방법론 적용
3. k-NN 전수비교 성능한계를 개선한 fast k-NN 기술도입
4. 최적의 structure feature를 선별하기 위해 수만개의 파일에서 통계적 분석수행
5. packing의 특징이 feature에 반영되도록 하는 section/entropy feature 선별
6. 다수의 머신러닝 방법론(DNN, Ensemble, SVM, k-NN) 복합사용
7. 머신러닝 모델에서 compiler, unpacker, others(packer)별 분류학습을 통해 정적 정보의 난독화 의존성 완화

감사합니다