Packer Identification using Byte plot and Markov plot

Journal of Computer Virology and Hacking Techniques, Volume 12, 2016. 2
Kesav Kancherla, John Donahue, Srlnlvas Mukkamala

김영철

2017. 4. 5.

Introduction

- 많은 malware 프로그램이 분석을 막기 위해서 packing 된 상태로 만들어지고 있음.
  - 특히 static analysis가 많이 방지됨.

- packing 된 프로그램을 unpacking 하는 도구가 존재함.
  - PolyUnpack, Renovo 등 : 프로그램의 instruction을 일일이 모니터링
    ➔ 성능 오버헤드가 커짐
  - 특정 packer(UPX, PECompact, ASPack, Molebox)의 unpacker
    ➔ 위 4가지 이외의 packer를 사용하였을 경우, unpack 불가

  즉, 무의미한 unpacking 시도를 줄이기 위해 packer identification 필요

Introduction

- packer identification에 대한 연구
  - PEiD : signature 기반 식별 도구, 미리 정의된 signature를 이용하여 exact match를 수행

    ➔ signature에 매우 의존적이고 새로운 packer에 대한 지식이 필요
  - Machine Learning (ML)을 이용한 방법들
    - 초기의 연구들은 packer detection을 목적으로 진행됨.
    - Sun et al. propose static-analysis-based scheme
      - 여러가지 classifier를 사용하여 높은 정확도로 packer를 식별함.

    ➔ packer의 수를 늘려서 실험을 하게 되면 성능이 떨어지게 됨.

Introduction

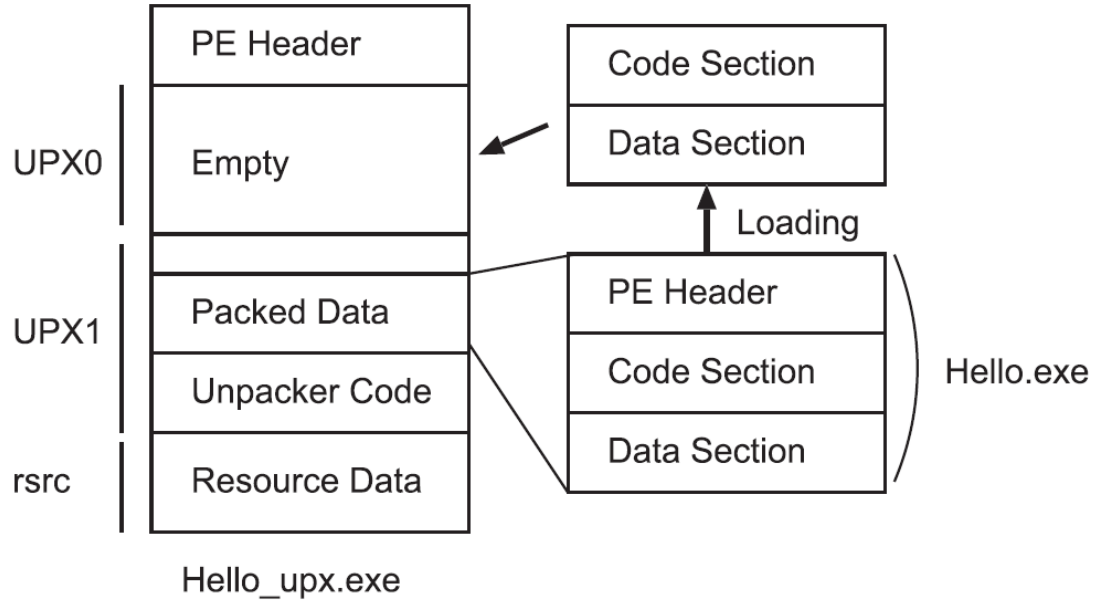
- signature와 ML의 장점을 이용하기 위해 edit distance kernel을 이용한 SVM을 소개
  - edit-distance : 두 프로그램 사이의 dissimilarity metric
  - SVM : classifier의 한 종류

Related Work

- packer에 대한 기본 배경 지식
  - packer : executable file을 압축, 압축해제 루틴과 함께 하나의 executable file로 만들어주는 도구
  - 원래는 최적화 목적으로 사용되었으나, malware 제작자들이 난독화를 위한 도구로 사용하고 있음.

Related Work

- UPX로 패킹된 파일의 구조



PE Header : 실행이 시작되는 entry point를 가짐
(unpacker code를 entry point로 지정해 줌)

Packed Data : 원래 파일의 압축 데이터
Unpacker Code : 압축 해제 코드
Resource Data : Packed Data에 대한 리소스

Related Work

- Signature-Based Packer Identification Tool
  - PEiD
    - normal mode : entry point로부터 시작하는 바이트 시퀀스를 스캔
    - deep mode : entry point를 포함하는 section의 여러 바이트 시퀀스를 스캔
    - hardcore mode : 파일 전체 중에서 여러 바이트 시퀀스를 스캔

    normal mode의 경우에 가장 낮은 false positive를 보임.

    ➔ entry point를 포함하는 section일수록 packer와 관련된 정보를 가짐.

UPX 2.90 signature : *60 BE ?? ?? ?? ?? 8D BE … 11 C0 01 DB*
"?" : wild card
"…" : 생략된 값

<signature example>

Related Work

- Signature-Based Packer Identification Tool
  - PEiD의 3가지 문제점
    - 새로운 packer에 대한 정보를 자동으로 수집하기 어려움.
    - exact matching을 사용하기 때문에 틀릴 확률이 높음.
    - 위의 두 가지와 관련된 signature를 수정하거나 업데이트하는 방법의 어려움.

Related Work

- Machine-Learning-Based Approach
  - Sun et al.의 연구
    - packer와 관련된 특징 정보를 추출하는 방법

      step1) 프로그램의 모든 바이트 값에 대한 byte-frequency histogram 생성
      histogram에 대한 Huffman tree 빌드

      step2) 프로그램을 고정된 길이 w의 코드 세그먼트로 분할
      각 코드 세그먼트에 포함된 바이트 값들은 Huffman encoding

      step3) step2의 encoding lengths가 classifier의 input vector로 입력
    - 처음 N개 마지막 N개의 encoding lengths가 가장 분별력 있는 packer 정보를 전달한다. ➔ input file은 2N의 벡터로 표현된다. (?)
    - pattern classification algorithm을 사용하여 9개 packer로 생성된 바이너리들에 대해 좋은 결과를 얻음.

Proposed Method

- Similarity Metric
  - Levenshtein Distance (edit distance)

    하나의 string이 다른 string으로 바뀌기 위해서 수행 되는 edit operation의 수와 같은 score

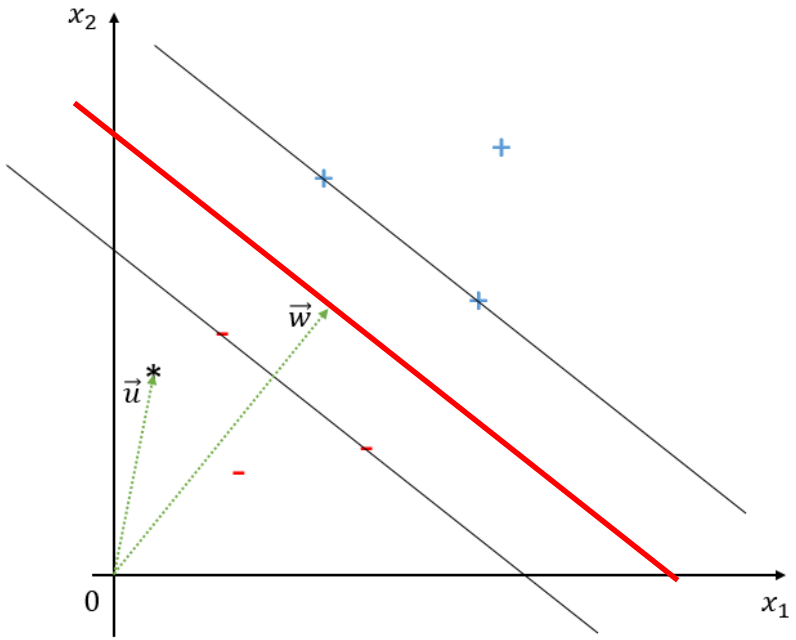
    edit operations : insertion, deletion, substitute operation
    edit path : string x를 string y로 바꾸는 edit operation의 sequence
    function c : edit operation에 가중치를 할당하는 함수
    cost : edit path에 포함된 각각의 가중치가 적용된 edit operation의 합
    ➔ symmetric metric을 위해 insertion과 deletion의 가중치는 같게 적용해야 함.

Proposed Method

- Similarity Metric
  - 앞서 정의된 LD가 주어졌을 때, 여러가지 classification 알고리즘을 적용할 수 있다.
  - Sun et al.의 연구에서는 *k*NN 알고리즘이 적용됨.
  - 본 논문에서는 Support Vector Machine (SVM)을 이용하여 *k*NN 알고리즘을 이용한 머신 러닝 기법보다 좋은 결과를 도출함.

Proposed Method

- Support Vector Machine



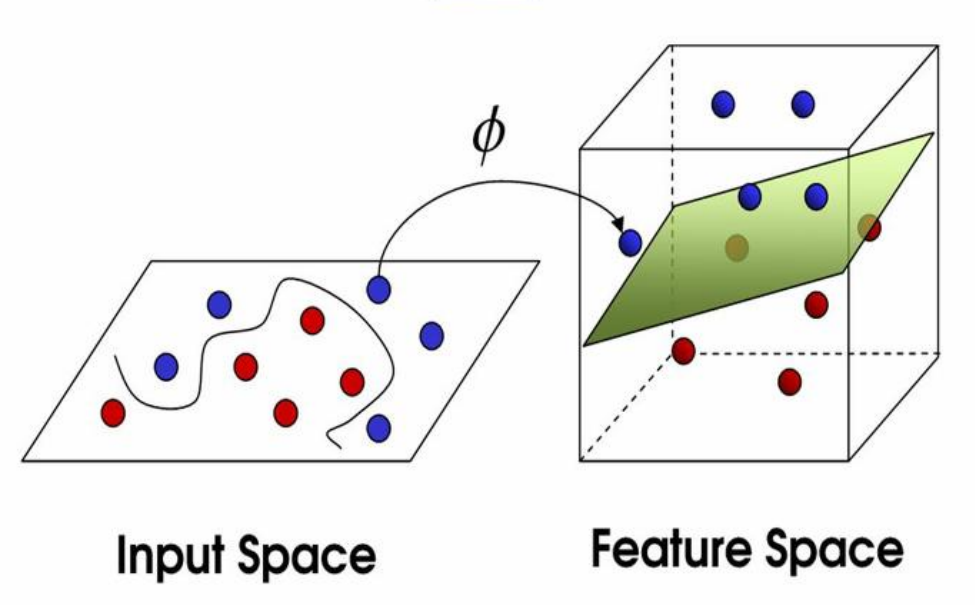
새로운 입력 벡터 u에 대해서 두 개의 클래스 중에서 어떤 클래스에 속할 것인지 판단하는 모델 (이진 선형 분류 모델)

초평면(빨간 실선)을 구함으로써 두 클래스를 나누는 방법을 학습

초평면과 가장 가까운 벡터를 Support Vector라고 함.

Proposed Method

- Support Vector Machine



비선형 분류 데이터들을 분류하기 위해서 주어진 데이터를 고차원 특징 공간으로 바꾸는 작업이 필요하다. (커널 트릭)

논문에서는 RBF (Radial Basis Function) Kernel을 사용

Proposed Method

- Input to SVM
  - Edit distance는 시간 복잡도, 공간 복잡도 모두 좋지 않음.
  - PEiD 도구를 통해 entry point의 시작 부분에 packer 특징 정보가 있는 것을 알아냄.
  - 분류기의 입력으로 entry point에서 시작하는 첫 L바이트를 취함.
    - 적절한 L은 분류기의 정확도와 트레이닝 및 테스트 시간을 줄일 수 있음.

Evaluation

- Dataset
  - “unpacked” 중에서 219 개의 프로그램을 베이스 파일로 지정
    - 26 개의 packer에 입력으로 주고 생성된 파일 중 non-executable은 제외
    - packer의 default 옵션으로 packed file 생성
  - unpacked 432 programs (64bit), 126 programs (.net & 32bit)
  - 총 3902 개의 PE 파일을 training set으로 주어짐.

Evaluation

- Parameter Tuning
  - 각 파라미터 조합에 대해 10-fold cross validation을 수행
  - best cross validation 결과와 해당 파라미터 변수 설정이 보고됨.

PARA.	CLASSIFIER	PHYSICAL MEANING	GRID VALUES
K	SUN_kNN, LD_kNN	Number of nearest neighbors for kNN	$\{1, 2, \cdots, 11\}$
W	SUN_kNN, SUN_SVM	Code segment length for Sun et al.'s feature extraction	$\{8, 16, 32, 64\}$
N	SUN_kNN, SUN_SVM	Prune size for Sun et al.'s feature extraction	$\{10, 20, \cdots, 250\}$
C	SUN_SVM, LD_SVM	Penalty parameter for SVM	$\{2^0, 2^1, \cdots, 2^{14}\}$
L	LD_kNN, LD_SVM	Input code length starting from entry point	$\{5, 10, 15, 20, 25, 30\}$
γ_1	SUN_SVM	Width parameter for RBF kernel	$\{10^{-3}, 10^{-2}, 10^{-1}, 10^0\}$
γ_2	LD_SVM	Width parameter for edit distance kernel	$\{2^{-10}, 2^{-9}, \cdots, 2^0\}$

CLASSIFIER	ACCURACY	PARAMETERS
SUN_kNN	92.21	$K = 1$, $W = 32$, $N = 160$
SUN_SVM	94.05	$C = 128$, $\gamma = 0.1$, $W = 32$, $N = 160$
LD_kNN	99.10	$K = 1$, $W = 32$, $L = 15$
LD_SVM	99.46	$C = 16$, $\gamma = 2^{-5}$, $L = 15$
PEiD	66.53	normal scan
PEiD	61.48	deep scan
PEiD	61.48	hardcore scan

Evaluation

- Evaluation Metrics
  - True Positive(TP) : 올바른 Positive 예측
  - False Positive(FP) : 잘못된 Positive 예측
  - True Negative(TN) : 올바른 Negative 예측
  - False Negative(FN) : 잘못된 Negative 예측

$$Accuracy = \frac{TP + TN}{n}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$FalsePositiveRate = \frac{FP}{FP + TN}$$

Evaluation

- Numerical Results

No.	Class	#	PEiD			SUN_SVM			LD_SVM		
			P(%)	R(%)	FPR(%)	P(%)	R(%)	FPR(%)	P(%)	R(%)	FPR(%)
1	Unpacked (64bit)	432	NaN	0.00	0.00	95.28	98.15	0.61	100.00	99.77	0.00
2	Armadillo 4.00.0053	203	100.00	99.01	0.00	91.87	94.58	0.46	98.54	100.00	0.08
3	MoleboxPro 2.6.4	203	100.00	99.01	0.00	96.19	99.51	0.22	100.00	100.00	0.00
4	Themida 1.8.5.5	195	100.00	98.97	0.00	88.52	94.87	0.65	100.00	100.00	0.00
5	PESpin 1.33	194	100.00	98.97	0.00	92.55	89.69	0.38	100.00	100.00	0.00
6	obsidium 1.3.5.4	191	NaN	0.00	0.00	99.48	99.48	0.03	100.00	100.00	0.00
7	ASProtect 2.1	188	100.00	98.94	0.00	97.35	97.87	0.13	100.00	100.00	0.00
8	yoda's protector 1.02	185	100.00	98.92	0.00	99.46	99.46	0.03	100.00	100.00	0.00
9	Unpacked (32bit)	170	98.06	89.41	0.08	76.50	82.35	1.15	95.27	94.71	0.21
10	obsidium 1.4.5	165	NaN	0.00	0.00	100.00	99.39	0.00	100.00	100.00	0.00
11	nPack 1.1.300	143	100.00	11.19	0.00	100.00	99.30	0.00	100.00	100.00	0.00
12	eXPressor 1.5.0.1	141	NaN	0.00	0.00	94.59	99.29	0.21	100.00	100.00	0.00
13	ASPack 2.12	139	100.00	98.56	0.00	83.69	84.89	0.61	100.00	100.00	0.00
14	PECompact 2.64	127	100.00	98.43	0.00	96.90	98.43	0.11	100.00	100.00	0.00
15	Unpacked (32bit&.net)	126	100.00	95.24	0.00	97.56	95.24	0.08	100.00	99.21	0.00
16	NsPack 3.7	122	100.00	98.36	0.00	98.31	95.08	0.05	100.00	100.00	0.00
17	RLPack 1.20	116	NaN	0.00	0.00	99.14	99.14	0.03	100.00	100.00	0.00
18	FSG 2.0	116	100.00	99.14	0.00	100.00	98.28	0.00	100.00	100.00	0.00
19	UPX 3.08	111	97.35	99.10	0.08	97.32	98.20	0.08	98.21	99.10	0.05
20	ASPack 2.11	99	100.00	97.98	0.00	80.43	74.75	0.47	100.00	100.00	0.00
21	Mew11SE 1.2	99	100.00	97.98	0.00	98.00	98.99	0.05	100.00	100.00	0.00
22	ACProtect pro 1.32	88	98.72	87.50	0.03	87.06	84.09	0.29	95.60	98.86	0.10
23	WWPack32 1.20	75	100.00	98.67	0.00	97.14	90.67	0.05	100.00	100.00	0.00
24	AntiCrack 1.32Pro	64	NaN	0.00	0.00	85.96	76.56	0.21	100.00	95.31	0.00
25	PETITE 2.2	63	100.00	95.24	0.00	93.33	88.89	0.10	96.83	96.83	0.05
26	exe32pack 1.4.2	62	100.00	96.77	0.00	96.23	82.26	0.05	100.00	98.39	0.00
27	tElock 0.98	55	100.00	98.18	0.00	95.83	83.64	0.05	100.00	100.00	0.00
28	PKLITE 32	20	100.00	85.00	0.00	63.64	35.00	0.10	90.00	90.00	0.05
29	Upack 0.39	10	100.00	90.00	0.00	90.00	90.00	0.03	100.00	100.00	0.00