2017

정보보호 R&D 데이터 챌린지

Team IDK
오동빈, 김예준, 이현, 채진희, 장원희

- Contents -

정보보호 R&D 데이터 챌린지

01 서론

02 방법론

- 함수별 opcode 추출
- 어셈블리 변환
- 유사도 계산

03 프로그램 설명

- IDA 프로그램을 통한 .idb생성
- Opcode를 어셈블리 명령어로 변환
- 유사도 비교

04 실험

- 데이터 추출
- 유사도 비교

05 결론

- 소결 및 한계점
- 결론

다양한 유형의 사이버 침해사고 발생

- 인터넷 나야나
- 워너크라이
- 페트야/낫페트야

…

→

악성코드의 종류와 수도 증가

→

위협적인 2차 직간접적 피해 발생

- 개인정보 유출
- 금융사기
- 사생활 침해

…

사이버 침해사고를 예방하기 위해
지능형 악성코드를 사전에 탐지 및 대응하는 것이 중요

기존의 악성코드 탐지 및 차단과
악성코드 분석가에 의한 수동적인 대응

변종 악성코드의
빠른 생성속도를 따라갈 수 X

폴리모픽, 메타모픽등과 같은
변종 악성코드에 적용되는 기법의 수 ↑

악성코드를 빠르고 효율적으로 분석하기 위해 악성코드의 개별적인 특징을 추출하여 얻은 공통점을 바탕으로 악성코드 유사도 분석 작업이 필요함

이를 바탕으로 악성코드를 분류하여 그룹화하기 위한 접근 방법 제시

정보보호 R&D 데이터 챌린지 데이터 셋

함수별 opcode를 이용한 악성코드 자체의 개별적인 특징 추출

데이터 셋 분석 결과

전체 데이터 셋 300개

295개 IDB 생성 성공
5개 IDB 생성 실패

295개 중 15개의
임의 악성코드 추출

Data

15개 악성코드로부터 3182개의 함수 데이터 추출

나머지 280개 중 100개 를
함수 3182개에 비교

100개 중 패킹 되지 않아
비교된 샘플은 46개

악성코드 선제 대응 알고리즘

악성코드 내부에 존재하는
함수별 opcode 추출

어셈블리 명령어로 변환

특징을 정형화하고
악성코드를 분류

함수별 Opcode 추출

IDA Pro by Ilfak Guilfanov

IDA를 사용하여 Opcode 추출

```
명령 프롬프트
C:₩Users₩hyOOun>"C:₩Program Files₩IDA 7.0₩idat64.exe" -A -B -P+ input_file.tmp
```

▲idat.exe의 autunomous 모드

idat.exe를 사용하여
IDA의 데이터 베이스 파일인 .idb 생성

- 서브함수 주소
- 시작 주소
- 메인 함수의 바이트 코드

.idb 파일 생성이 완료되면
Python idb 모듈을 이용하여
위 항목들을 추출

어셈블리 변환

각각의 함수를 기준으로
Opcode 작성

Opcode → Byte code

추출한 opcode를 바이트 코드로 변환 후,
디스어셈블리 과정을 통해 어셈블리 코드로 변환

이 때, 동적으로 변하는 어셈블리 코드를
제거하여 필터링 수행함

유사도 계산

```
Algorithm 1 Function Matini
  CISC2017 – DataChallenge – MalwareDataset = Malware Binary Data set
Input: Files - converted to .idb by binary
  for all start address of each function do
    if sub function address or start address or main function address then
      while start address < end address in function do
        convert address to opcode
      end while
      convert opcodes to disassembly code
      remove unnecessary value like Call or PUSH instruction and pointer in disassembly code
      similarity = intersection of new input data and stored compare data
      return similarity
```

▲ 유사도 계산 알고리즘(Martini) pseudo code

위 과정에서 저장한
필터링 된 리스트들을 이용

함수별 별 임계값 설정
일정 조건 해당하는 함수만 비교

교집합을 기반으로 유사도 계산식에
대입하여 바이너리를 그룹화

비교할 대상의 바이너리 분석 결과와의
교집합으로 만나는 부분 카운트

개발 환경- Windows 10

IDE- visual studio code

개발 언어- python 3.5

IDA 프로그램을 통해 .idb 생성

- IDA 프로그램을 통해 .idb 생성하기 때문에 IDA 프로그램이 기본적으로 설치되어 있어야 함.
- IDA의 autonomous 모드를 통해 .idb를 생성하고 python의 idb 모듈을 이용해 이를 분석

Opcode를 어셈블리 명령어로 변환

- Idb 모듈을 이용해 추출해낸 opcode는 바이트 코드로 변환
- Distorm3 모듈을 이용하여 바이트 코드를 어셈블리 코드로 변환
- 함수 분기 주소, 포인터 등과 같은 동적인 값은 선택적으로 제거
- 변환된 어셈블리 코드는 데이터베이스에 저장

유사도 비교

- 데이터베이스에 이미 저장한 어셈블리 코드와 분석 대상 악성코드의 각 함수의 사이즈를 임계치를 지정하여 비교
- 함수 사이즈 범위에 해당된다면 동일한 어셈블리 코드를 교집합 과정을 통해 유사도 비교
- 도출된 결과는 csv로 추출

데이터추출

▼KISA-CISC2017-Intelligent-Mal 데이터 셋

임의로 선택한 15개의
악성코드 opcode 추출

추출된 opcode는 어셈블리로 변환

Main, sub 함수, start address 이름을
가지고 opcode가 10개 이상인 함수 추출

이때, ida의 autonomous 모드를 이용해
.idb 파일로 생성

총 3182개의 함수 데이터 추출,
이를 데이터베이스에 저장

나머지 데이터 셋 285개를
대상으로 함수 유사도 비교

유사도비교

위 과정에서 선택된 15개의 악성코드에서 추출한 함수 코드와 분석 대상 바이너리의 각 함수의 사이즈 비교

함수 사이즈가 완벽하게 동일하지 않기 때문에 적절한 임계치 지정

동일한 어셈블리 코드가 얼마나 있는지 리스트화

교집합 과정을 통해 동일한 어셈블리 명령어 코드가 어느정도 있는지 체크

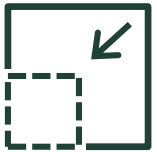
악성코드 유사 비교 시 이름, 오버랩 된 함수들 중 DB내 함수 등의 데이터를 csv로 생성

유사도 알고리즘 적용

악성코드 클러스터링

악성코드 내의
함수 유사도 값 중
최대값으로 대표값 선정

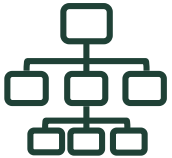
피벗 테이블을 이용한 CSV 재구성



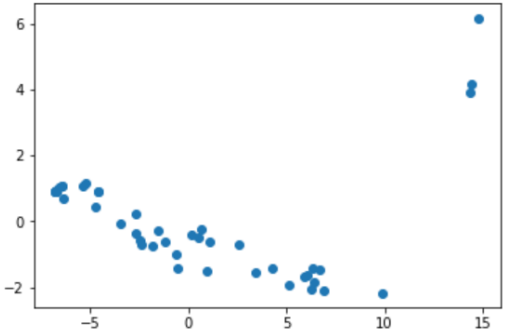
주 성분 분석(PCA)을
이용한 차원 축소

K-means Clustering을 이용한
악성코드 클러스터링

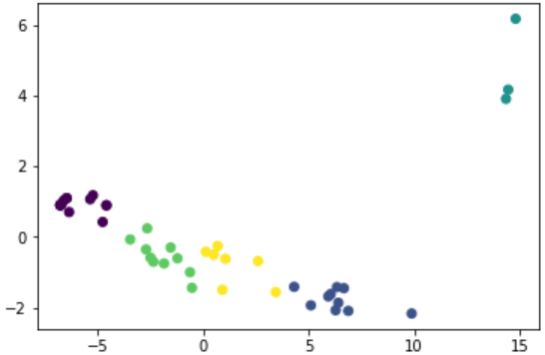
Cluster 값을 변화시키며
최적의 Cluster 수 결정



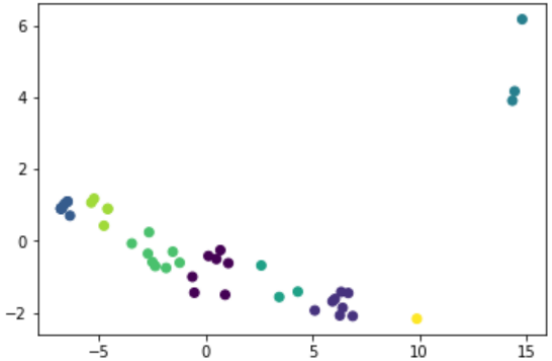
Cluster 별 중앙 값으로
Key 악성코드 판별



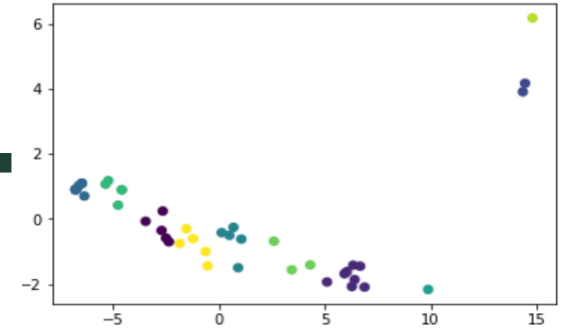
PCA를 이용한 2차원 데이터로의 축소



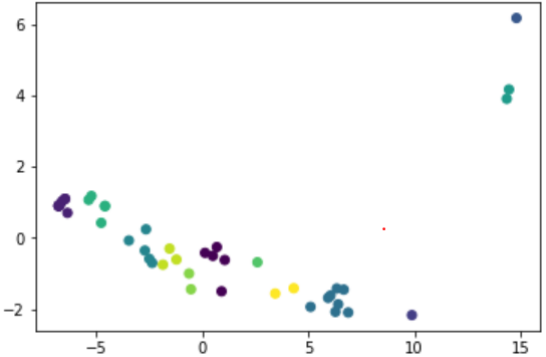
Cluster가 5개일 때의 Clustering



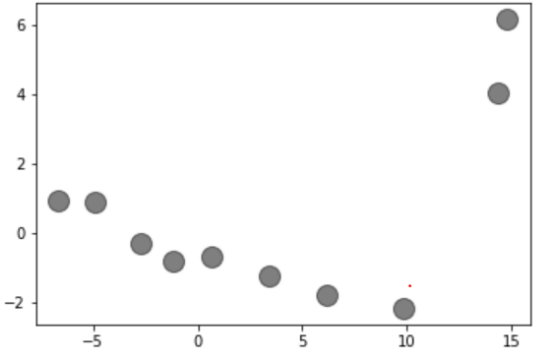
Cluster가 8개일 때의 Clustering



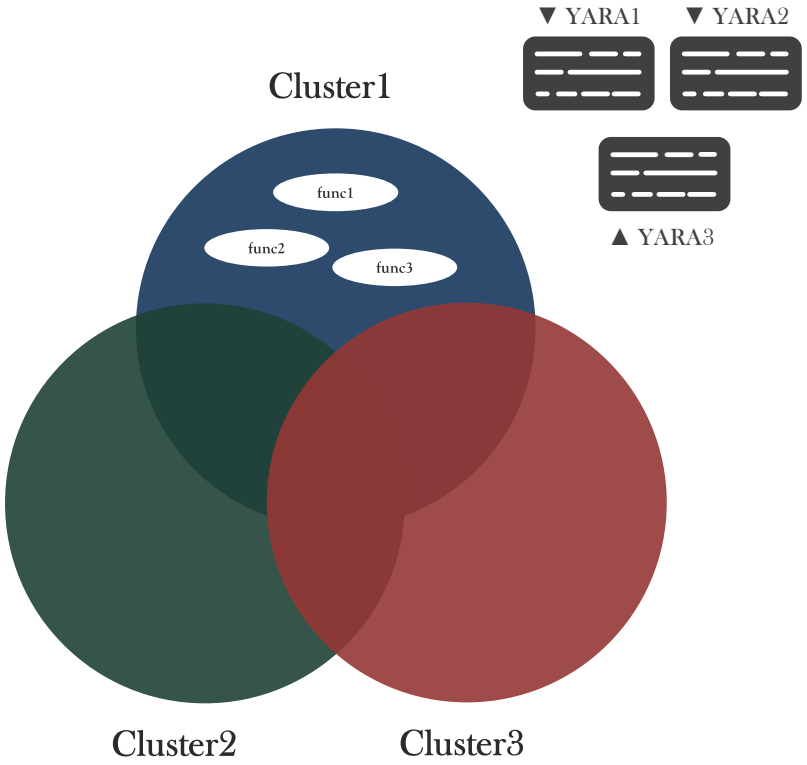
Cluster가 10개일 때의 Clustering



Cluster가 12개일 때의 Clustering



최적의 Cluster 수(10개)에서 Centroid 선정



- 클러스터로부터 악성코드 Family들의 함수들을 추출할 수 있음
- 함수를 기준으로, 클러스터 간 함수 집합을 표현할 수 있음
- 제안하는 방법에 의해 유사한 함수로 분류된 부분을 제외한 함수들로 YARA 자동화 가능

- 악성코드 바이너리 전체의 유사도를 분석하는 것은 한계가 존재
  → 함수 유사 대표값을 이용하여 보완
- K-Means Clustering을 통해 함수 Opcode를 기준으로 Clustering 가능
- Cluster 된 악성코드들은 계산된 함수 유사도에 의해 Family를 구성함
- 악성코드는 Family 별 고유한 함수 Opcode를 가짐
- UPX, Themida 등의 패킹이 걸린 악성코드일 경우, 함수 개수가 0으로 처리됨
- PE 구조가 일부 변형되었을 경우, IDA autonomous에서 인식하지 못함

- 제시하는 알고리즘은 지능형 악성코드들 간의 유사도 측정 및 활용이 목적임
- 악성코드에서 사용하는 특정 루틴 탐지 시, IDA가 함수별로 처리하는 기능을 제공,
  정해진 기준점으로부터 함수별로 비교 가능
- 바이트 코드를 추출하여 비교하므로 중간에 삽입된 더미 코드를 제외하고 변형된 코드도 유사도를 분석 가능
- 유사도 분석을 바탕으로 구성된 Family 별 고유한 함수를 추출해 기존 악성코드 탐지 도구인 Yara 생성 자동화 가능
- 함수 유사도 알고리즘을 이용하여 악성코드 프로파일링에도 도움