함수 수준 특징정보 기반의 오픈소스 소프트웨어 모듈 탐지

Journal of KIISE 2015. 6
김동진, 조성제

김영철

2016. 3. 11.

서론

- OSS 라이선스 위반과 관련 분쟁 빈번
- 기존의 소스코드 기반 OSS 모듈 탐지 도구
  ➔ 외부 라이브러리의 OSS 모듈 탐지의 어려움
  ➔ 바이너리 기반의 OSS 모듈 탐지 기술 필요

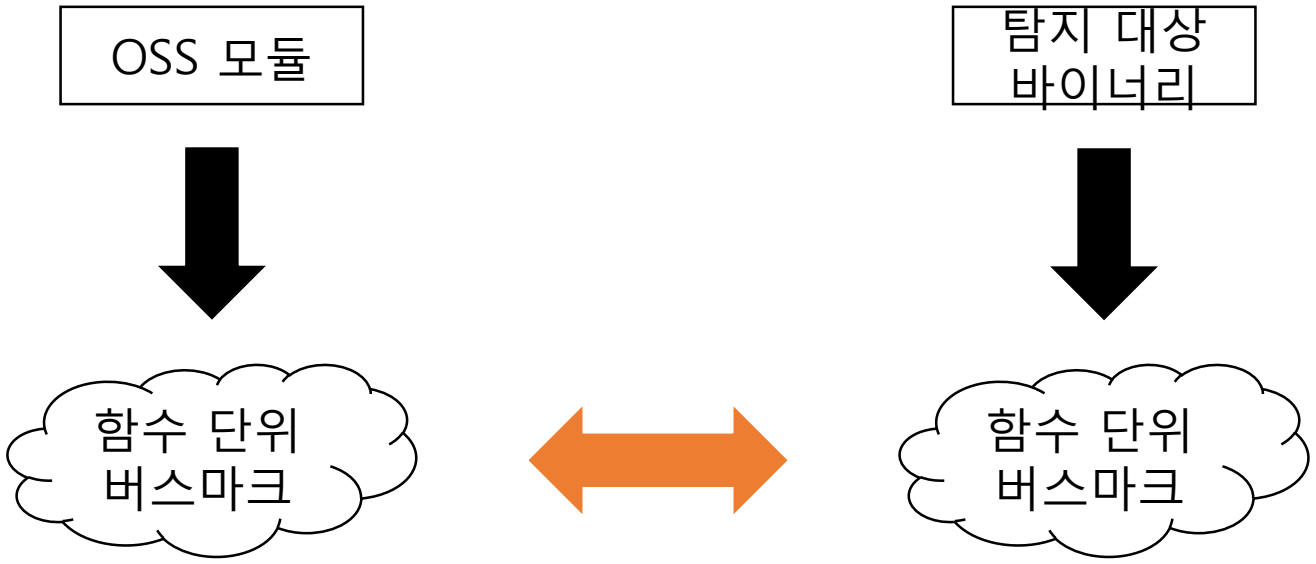
- 소프트웨어 버스마크 기법
  ➔ 명령어, API, 콜 그래프 등 프로그램의 특징 정보 사용
  ➔ 프로그램 전체 간의 유사성 분석 목적

서론

- OSS 모듈 탐지 방법
  R1. 일부 함수를 사용하는 경우에도 탐지
  R2. 특징정보 양이 적고 탐지속도가 빨라야 함
  R3. 난독화 및 최적화에 대한 강인성이 높아야 함

서론

- PE에서 소프트웨어 버스마크 기반의 OSS 탐지 기법 제안



***함수 단위 정적 버스마크 : 실행명령어, CFG, 함수 수준 구조적 특징정보

관련 연구

- OSS 탐지 연구
  (1) Black Duck Software - Protext
  -코드 비교, 문자열 검색 등을 통한 OSS 사용 탐지

  (2) HP – Fossology
  -주석으로 기재된 라이선스 정보를 바탕으로 탐지

  (3) Binary Analysis Tool
  -바이너리 수준에서 OSS 포함 여부 탐지
  -심볼 및 문자열 테이블만을 소스코드와 비교

  ➔ 외부 라이브러리 같은 경우 탐지 불가

관련 연구

- 소프트웨어 버스마크
  (1) mnemonic에 k-gram 기법을 적용, 유사성 비교
  ➔ 난독화 및 최적화에 약함

  (2) API 기반 동적 버스마크
  (3) API 기반 정적 버스마크
  ➔ OSS 모듈에 API 사용량이 적음

관련 연구

- 소프트웨어 버스마크
  (4) WPPB(Whole Program Path Birthmark) – 동적 제어흐름 그래프 기반 버스마크

  (5) CDG(Component Dependence Graph) 기반 버스마크
  ➔ 일부 모듈 탐지에 부적합

  (6) mnemonic 서열 특징정보를 사용한 탐지 기법
  (7) 함수 인자, 지역 변수의 크기를 이용한 탐지 기법
  ➔ 명령어나 함수의 재배치 된 경우 오탐률 상승

함수 수준 정적 특징정보 기반 OSS 탐지

- 함수 수준 OSS 모듈 탐지 기법 제안
  (1) OSS 모듈과 대상 프로그램의 코드영역 역어셈블
  (2) 함수 단위로 분리 및 함수들의 명령어 분석
  (3) OSS 모듈과 대상 프로그램 함수의 유사성 비교

  ➔ 명령어, CFG, 함수 수준 구조적 특징 사용

함수 수준 정적 특징정보 기반 OSS 탐지

- 명령어 기반 특징정보
  역어셈블 된 실행명령어의 mnemonic 서열 비교
  ➔ 기존 연구에서는 k-gram 기법 사용
    ➔ 컴파일 과정에서 mnemonic 변하는 경우 좋지 않음
  ➔ LCS(Longest Common Sequence) 기법 사용

$$Similarity(P, Q) = \frac{|LCS(P, Q)|}{min(|P|, |Q|)}$$

<LCS값을 이용한 유사도 수식>

함수 수준 정적 특징정보 기반 OSS 탐지

- 제어흐름 그래프
  악성코드 패밀리 분류, 악성코드 변종 탐지에 활용
  ➔ 난독화, 코드 변조에 강한 특징정보를 의미

  그래프 유사성 비교
  ➔ 함수 단위 그래프 정보가 적기 때문에 특징정보로 활용 가능

  X. Hu가 제안한 그래프 비교 기법 적용
  ➔ Neighborhood Cost 기반 Graph edit distance 기법 사용

함수 수준 정적 특징정보 기반 OSS 탐지

- 제어흐름 그래프
  Graph edit distance 기법

$$a_{ij} = relabeling\,cost + \left(|\mathrm{ON}_i| + |\mathrm{ON}_j| - 2\times|\mathrm{ON}_i \cap \mathrm{ON}_j|\right) + \left(|\mathrm{IN}_i| + |\mathrm{IN}_j| - 2\times|\mathrm{IN}_i \cap \mathrm{IN}_j|\right)$$

a_{ij} : 노드간의 매칭 비용
ON_i : a_i의 나가는 엣지로 연결된 노드 집합
IN_i : a_i의 들어오는 엣지로 연결된 노드 집합

➔ 먼저 mnemonic 서열의 MD5 해시값을 비교
➔ 같지 않을 경우, LCS 값이 임계값을 넘으면 동일한 노드로 판단
➔ 완성된 행렬에서 최소 수정 비용 선택

함수 수준 정적 특징정보 기반 OSS 탐지

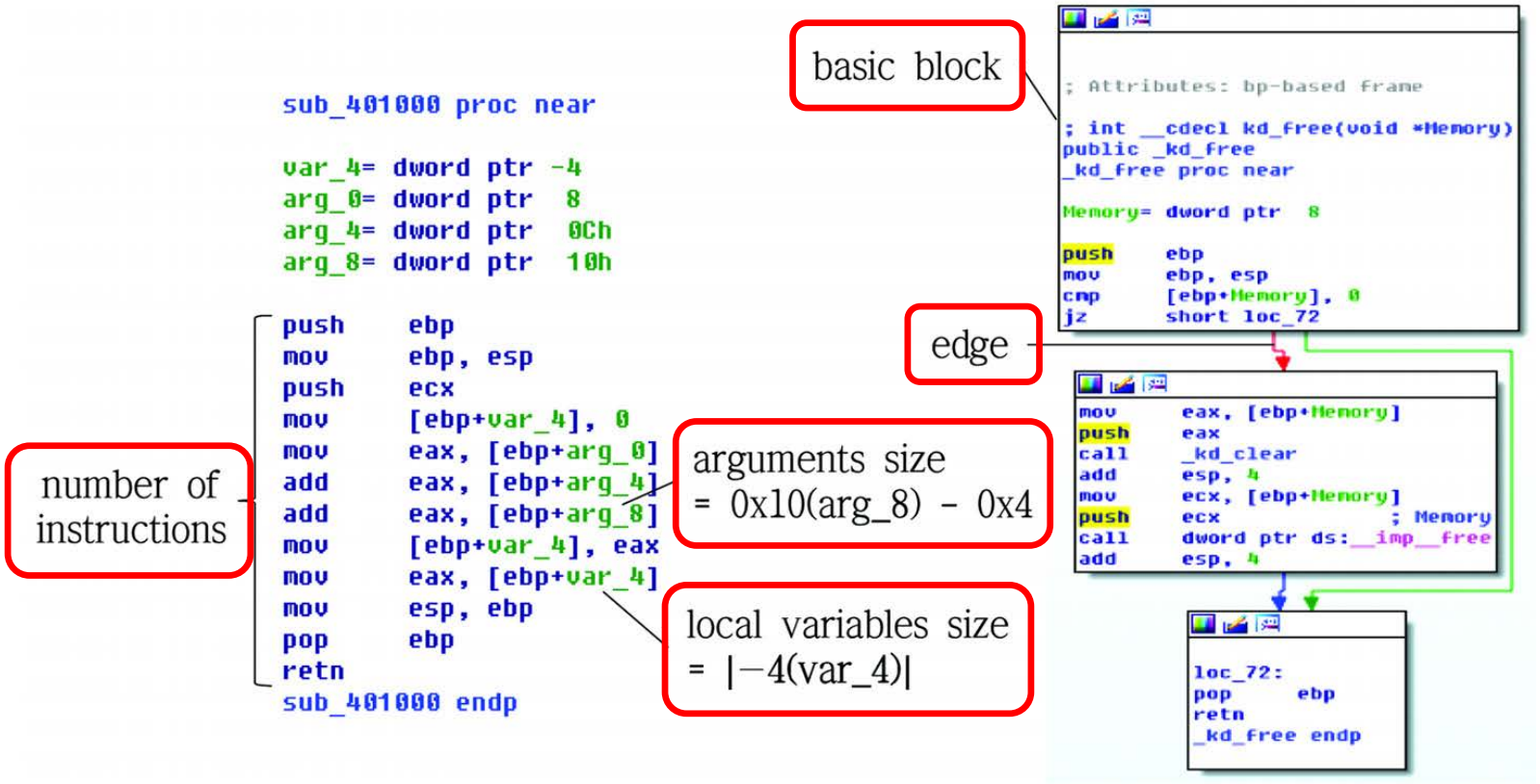
- 제어흐름 그래프

$$Similarity(G_1, G_2) = 1 - \frac{minimum\ \ edit\ distance}{|V_1|+|E_1|+|V_2|+|E_3|}$$

V1 : G1의 노드 집합
V2 : G2의 노드 집합
E1 : G1의 엣지 집합
E2 : G2의 엣지 집합

함수 수준 정적 특징정보 기반 OSS 탐지

- 개선된 함수 수준 구조적 특징정보



함수 수준 정적 특징정보 기반 OSS 탐지

- 개선된 함수 수준 구조적 특징정보
  5개의 구조적 정보를 벡터 형태로 변환하여 코사인 유사(Cosine similarity)도 기법을 사용하여 유사도 분석

$$Similarity(P, Q) = \frac{p_1 q_1 + p_2 q_2 + \cdots + p_n q_n}{\sqrt{p_1^2 + p_2^2 + \ldots + p_n^2}\ \sqrt{q_1^2 + q_2^2 + \ldots + q_n^2}}$$

<코사인 유사도 수식>