# 가상 머신 기반으로 난독화된 실행파일의 구조 및 원본의미 추출 동적 방법

이성호, 한태숙

Journal of KIISE. Vol 41. pp 859-869. 2014 10

김영철

2016. 1. 8.

# 실행파일의 가상 머신 구조 추출 기법

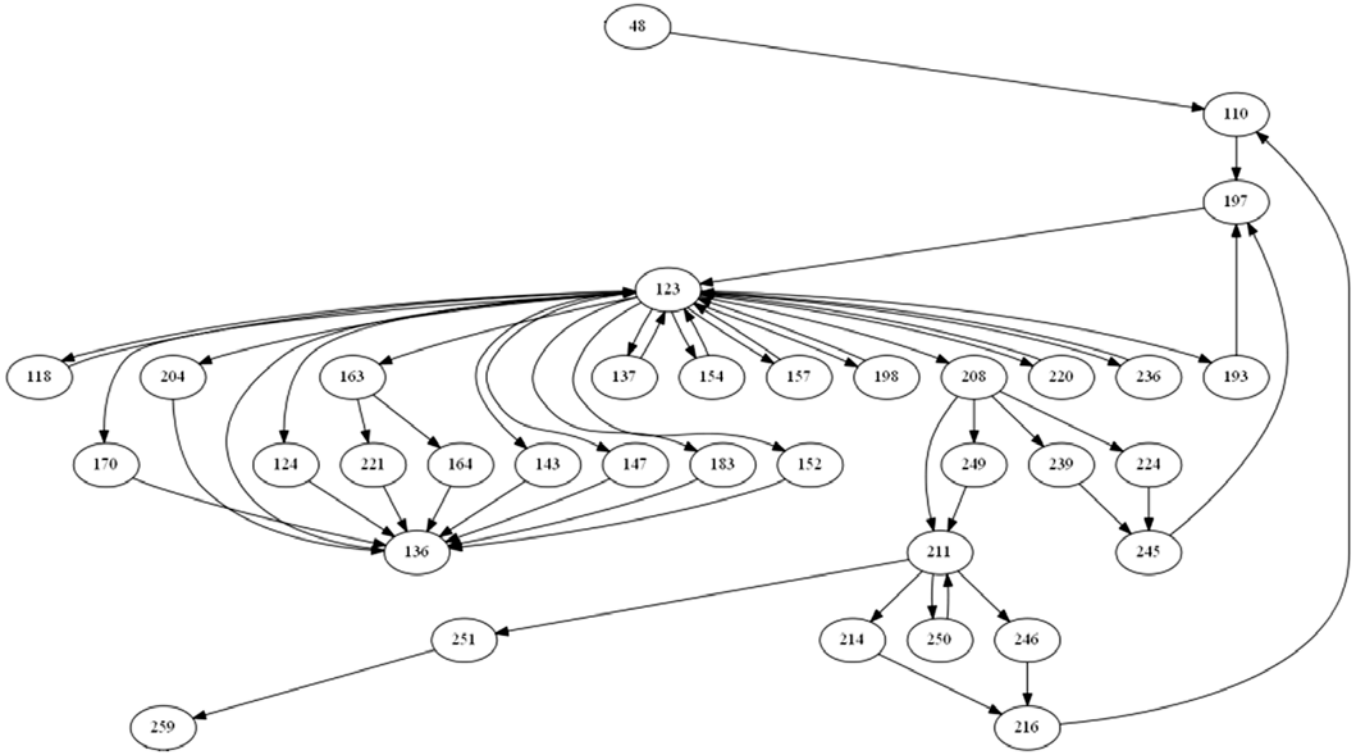
- 가상 머신 구조
  a. 인터프리터
  b. 가상 프로그램 카운터
  c. 인터프리터 초기화 루틴
  d. 바이트 코드 핸들러

# 실행파일의 가상 머신 구조 추출 기법

- 제어 흐름 그래프 구축을 위한 규칙
  R1. 베이직 블록 정의
  R2. 베이직 블록 분화
  R3. 제어 흐름 보정

# 실행파일의 가상 머신 구조 추출 기법

- 가상 머신 구조 추출




<제어 흐름 그래프>

# 실행파일의 가상 머신 구조 추출 기법

- 인터프리터 추출
  : 제어 흐름 그래프에서 가장 많음 후임 노드를 가진 노드

# 실행파일의 가상 머신 구조 추출 기법

- 가상 프로그램 카운터 추출
  : 6개의 규칙을 정의하여 역방향 슬라이싱 수행

**B3:** $r \leftarrow c$

if $r \in R_{i+1}$ then

$R_i \leftarrow R_{i+1} - \{r\}$

$M_i \leftarrow M_{i+1}$

fi

**B2:** $r_1 \leftarrow [r_2]$

if $r_1 \in R_{i+1}$ then

if IsStackAddress(value($r_2$)) then

$R_i \leftarrow R_{i+1} - \{r_1\}$

$M_i \leftarrow M_{i+1} \cup \{value(r_2)\}$

else then

$R_i \leftarrow (R_{i+1} - \{r_1\}) \cup \{ r_2\}$

$M_i \leftarrow M_{i+1}$

fi

fi

**B6:** $[r] \leftarrow c$

if value($r_1$) $\in M_{i+1}$ then

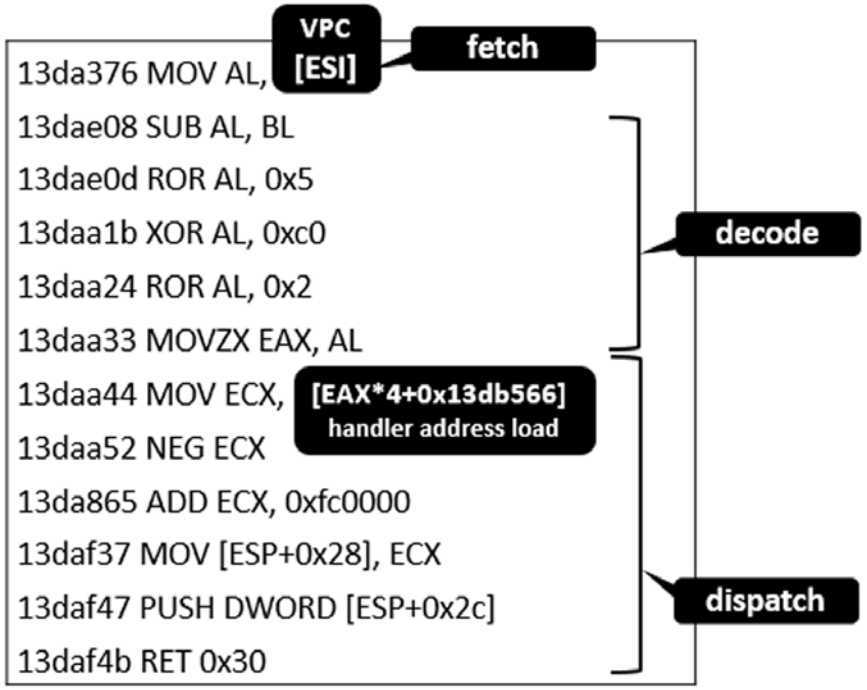
$R_i \leftarrow R_{i+1}$

$M_i \leftarrow M_{i+1} - \{value(r)\}$

fi

# 실행파일의 가상 머신 구조 추출 기법

- 가상 프로그램 카운터 추출

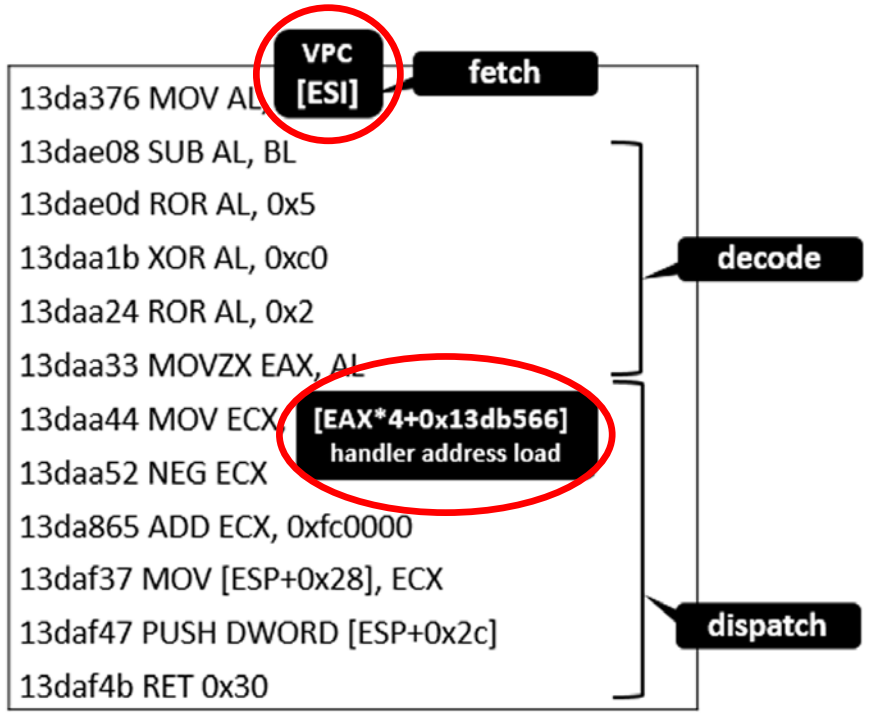



<인터프리터 베이직 블록>

fetch : 바이트 코드 로드
decode : VPC 변경
dispatch : 바이트 코드 핸들러 주소 로드

# 실행파일의 가상 머신 구조 추출 기법

- 가상 프로그램 카운터 추출




<인터프리터 베이직 블록>

```
vpc = {}
register = {}
index = Inst_list.size-1

while(index >= 0)
  inst = Inst_list[index]
  if(inst.type == AssignStatement)
    if(inst.RightHandSide.type == LoadStatement)
      if(inst.RightHandSide.address != StackAddress)
        if(register.size == 0)
          register = register ∪
                {inst.RightHandSide.base_registers}
        else if(inst.LeftHandSide.register ∈ register)
          vpc = vpc ∪ {inst.RightHandSide.base_registers}
        fi
      fi
    else if(inst.LeftHandSide.register ∈ register)
      register = register - {inst.LeftHandSide.register} ∪
              {inst.RightHandSide.registers}
    fi
  fi
  index--
elihw
```

<VPC 추출 알고리즘>

# 실행파일의 가상 머신 구조 추출 기법

- 인터프리터 초기화 루틴 추출
  : 인터프리터 이전에 실행된 베이직 블록들에 대해 슬라이싱
  : 집합 R과 M이 공집합이 되거나 최초 베이직 블록에 대한 슬라이싱이 완료될 때까지 수행

# 실행파일의 가상 머신 구조 추출 기법

- 인터프리터 초기화 루틴 추출

```
13dcab9   PUSH DWORD 0xcdcd541a
13dc33b   MOV DWORD [ESP+0x4], 0xfc0000
13dc5d8   MOV ESI, [ESP+0x30]
13dbe56   XOR ESI, 0xa0934b1c
13da993   ROL ESI, 0xc
13db364   SUB ESI, 0xe1ae980d
13da77d   MOV EBX, ESI
13da789   ADD ESI, [EBP+0x0]
```

<인터프리터 초기화 루틴>

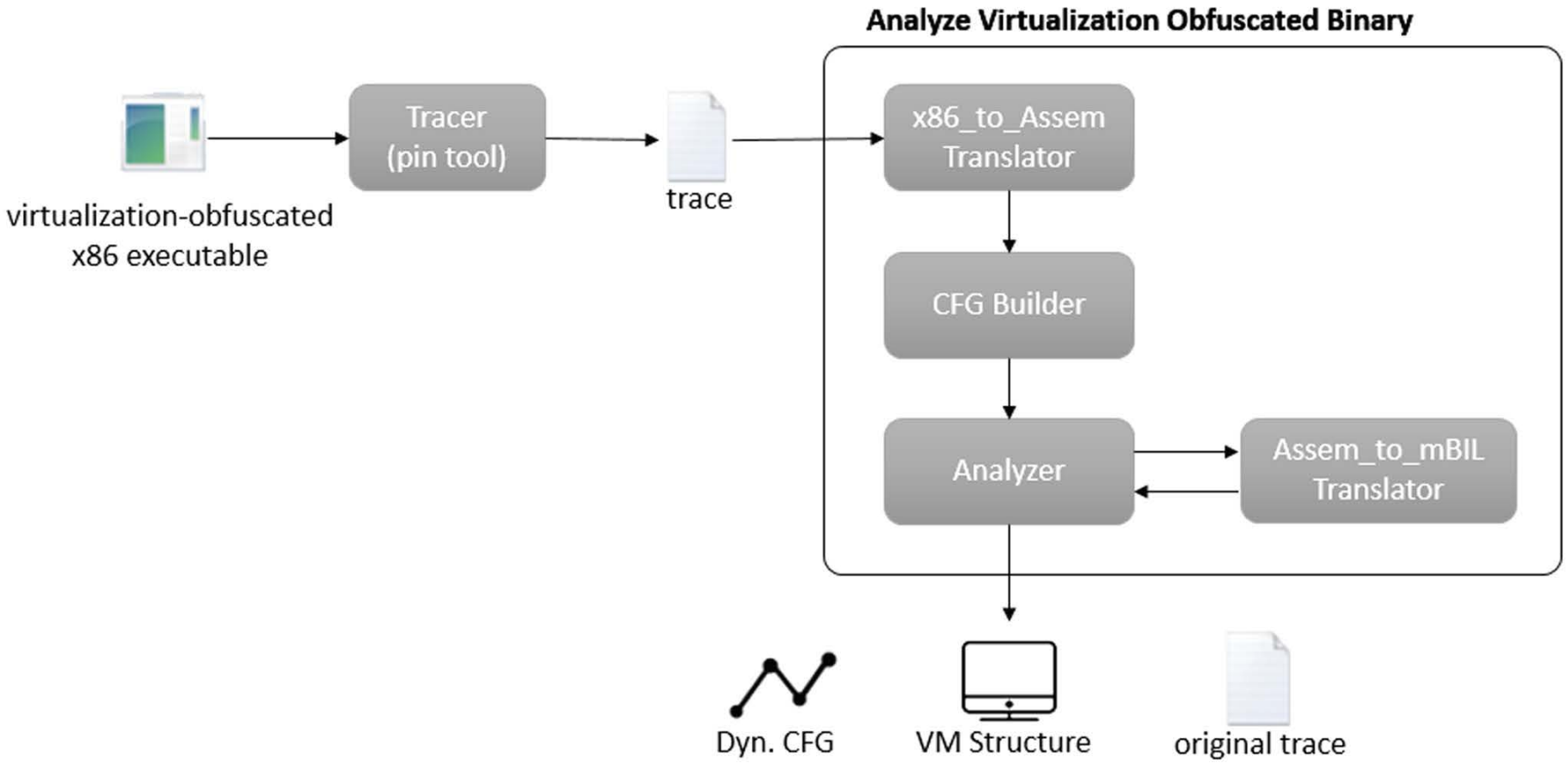
# 실행파일의 가상 머신 구조 추출 기법

- 바이트 코드 핸들러 추출
  : 제어 흐름 그래프에서 인터프리터와 사이클을 이루는 루틴
  : 베이직 블록 하나 or 베이직 블록 리스트로 구성

# 원본 프로그램 의미 추출

- 제어 흐름 그래프 + 동적 트레이스
- 원본 프로그램의 의미를 포함한 동적 트레이스
- 바이트 코드 로드 시점의 메모리를 순서대로 저장
  ➔ 실행되는 바이트 코드 주소의 순서 파악
- 원본 제어 흐름 그래프 복원
  ➔ 바이트 주소를 베이직 블록으로 선정
- 원본 동적 트레이스 복원
  ➔ 실행된 핸들러를 순서대로 나열

# 도구 시스템 구성


Analyze Virtualization Obfuscated Binary
virtualization-obfuscated
x86 executable
Tracer
(pin tool)
trace
x86_to_Assem
Translator
CFG Builder
Analyzer
Assem_to_mBIL
Translator
Dyn. CFG
VM Structure
original trace

# 샘플 및 실험 결과

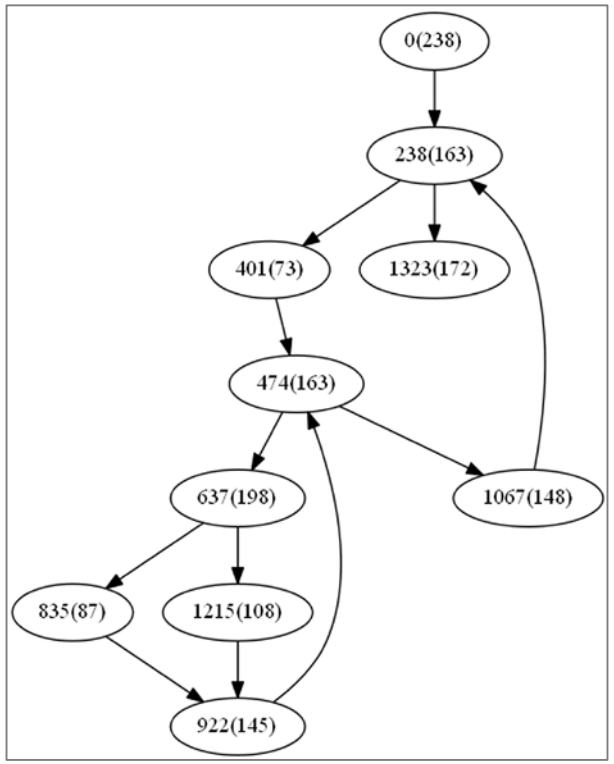
- 샘플은 C언어로 작성
- 난독화 도구
  ➔ VMProtect v2.12.2
  ➔ Code Virtualizer v1.3.9.10

# 샘플 및 실험 결과

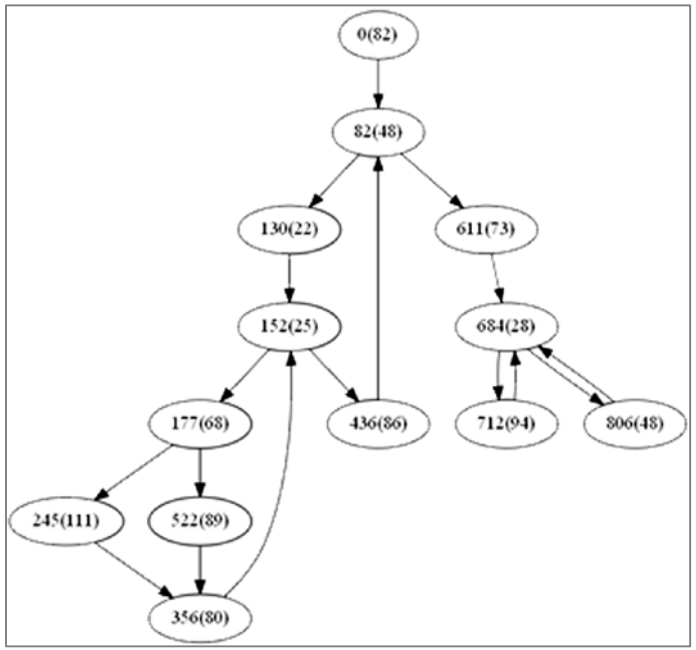
```
#include <stdio.h>

void main(){
   int result = 0;
   for(int i=1; i<11; i++){
      for(int j=1; j<11; j++){
         if(i%2 == 0)
            result += i;
         else
            result -= i;
      }
   }
   printf("%d\n",result);
}
```

<이중 반복문 내 분기문>




<VMProtect CFG>




<Code Virtualizer CFG>

# 샘플 및 실험 결과

Tori : 원본 인스트럭션 수
Tob : 난독화 인스트럭션 수
Tre : 복원된 인스트럭션 수

| | $T_{ori}$ | $T_{ob}$ | | $T_{re}$ | | $Del\%$ | |
|---|---|---|---|---|---|---|---|
| | | VMProtect | Code Virtualizer | VMProtect | Code Virtualizer | VMProtect | Code Virtualizer |
| Sample1 | 2,124 | 384,714 | 190,613 | 170,156 | 71,533 | 55.8% | 62.5% |
| Sample2 | 2,181 | 575,649 | 311,305 | 271,451 | 121,383 | 52.8% | 63.4% |
| Sample3 | 3.589 | 5,544,662 | 3,662,938 | 2,563,034 | 1,163,414 | 53.8% | 68.2% |
| Sample4 | 2,333 | 1,052,416 | 538,564 | 518,302 | 212,353 | 50.8% | 60.6% |

<난독화된 샘플의 복원>

# 결론

- 가상 머신 기반 난독화 기법에 대한 분석 방법 제안, 도구 개발
  ➔ 가상 머신 구조 추출
  ➔ 원본 의미 추출
- 원본에 비해 많은 인스트럭션을 가짐
- 최적화 기법을 적용하여 원본과 유사하게 복원 기대