Program Vulnerability Analysis Using DBI

CodeEngn Co-Administrator
DDeok9@gmail.com
2011.7.2

Outline

- What is DBI ?
- Before that
- How ?
- A simple example
- Demo !

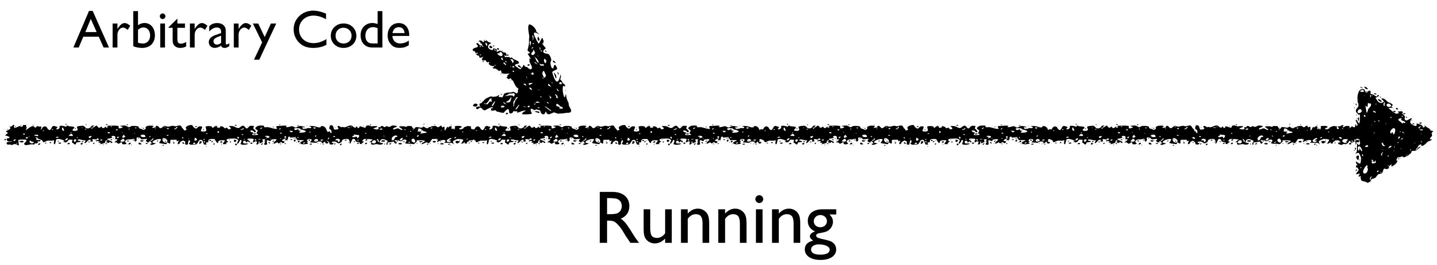
What is DBI ?

- Instrumentation

  Keyword : To gather information, insert code

- Dynamic Binary Instrumentation

  Keyword : Running program, special purpose, insert code



Arbitrary Code

Running

Static Analysis

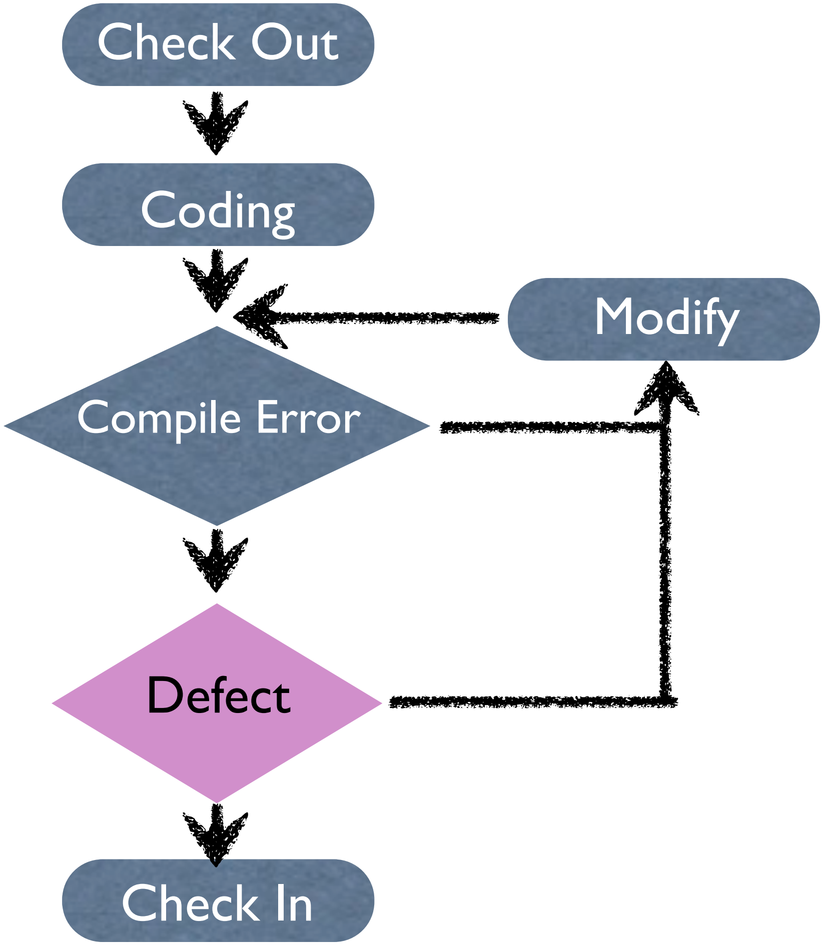
- Summary

  - Without running

  - Considering all execution paths in a program

  - Tools : Sonar, cppcheck, Prevent, KlockWork

Static Analysis



Dynamic Analysis

- Summary

  - Running

  - Considering single execution path

  - Input dependency

Winner

- Dynamic Analysis

  More precise →

  Because > works with real values in the run-time

- if (you think Ollydbg & IDA Disassembler)

  Easy to understand

Source Analysis

- Source Analysis

  - Language dependency

  - Access high-level information

  - Tools : Source insight

Binary Analysis

- Binary Analysis

  - Platform dependency

  - Access low-level information ex) register

  - Complexity, Lack of Higher-level semantics, Code Obfuscation

DRAW

- Binary Analysis

  Original source code is not needed

- Source Analysis

  Just you look at source

영어 -> 한국어 번역

그냥 소스 좀 봐

geunyang soseu jom bwa

New! 대체 번역을 보려면 위 단어를 클릭합니다. 무시

SBI

- Static Binary Instrumentation

  - Before the program is run

  - Rewrites object code or executable code

  - Disassemble -> instrumentation

DBI

- Dynamic Binary Instrumentation

  - Run-time

  - By external process, grafted onto the client process

Winner

- DBI

1. Client program doesn’t require to be prepared
2. Naturally covers all client code

Usefulness of DBI

- Do not need Recompiling and Relinking
- Find the specific code during execution
- Handle dynamically generated code
- Analyzing running process

Use

- Trace procedure generating
- Fault tolerance studies
- Emulating new instructions
- Code coverage -> t / all * 100
- Memory-leak detection
- Thread profiling
- And so on . . .

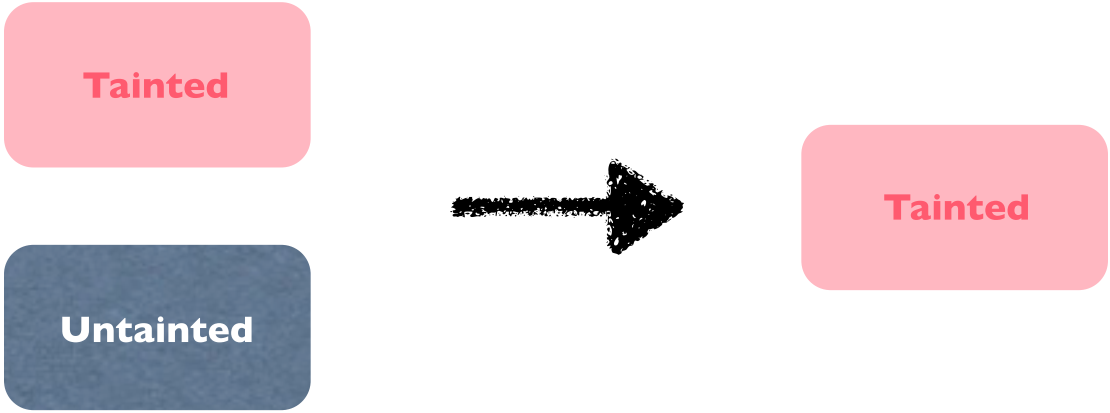
Before that

- Taint Analysis

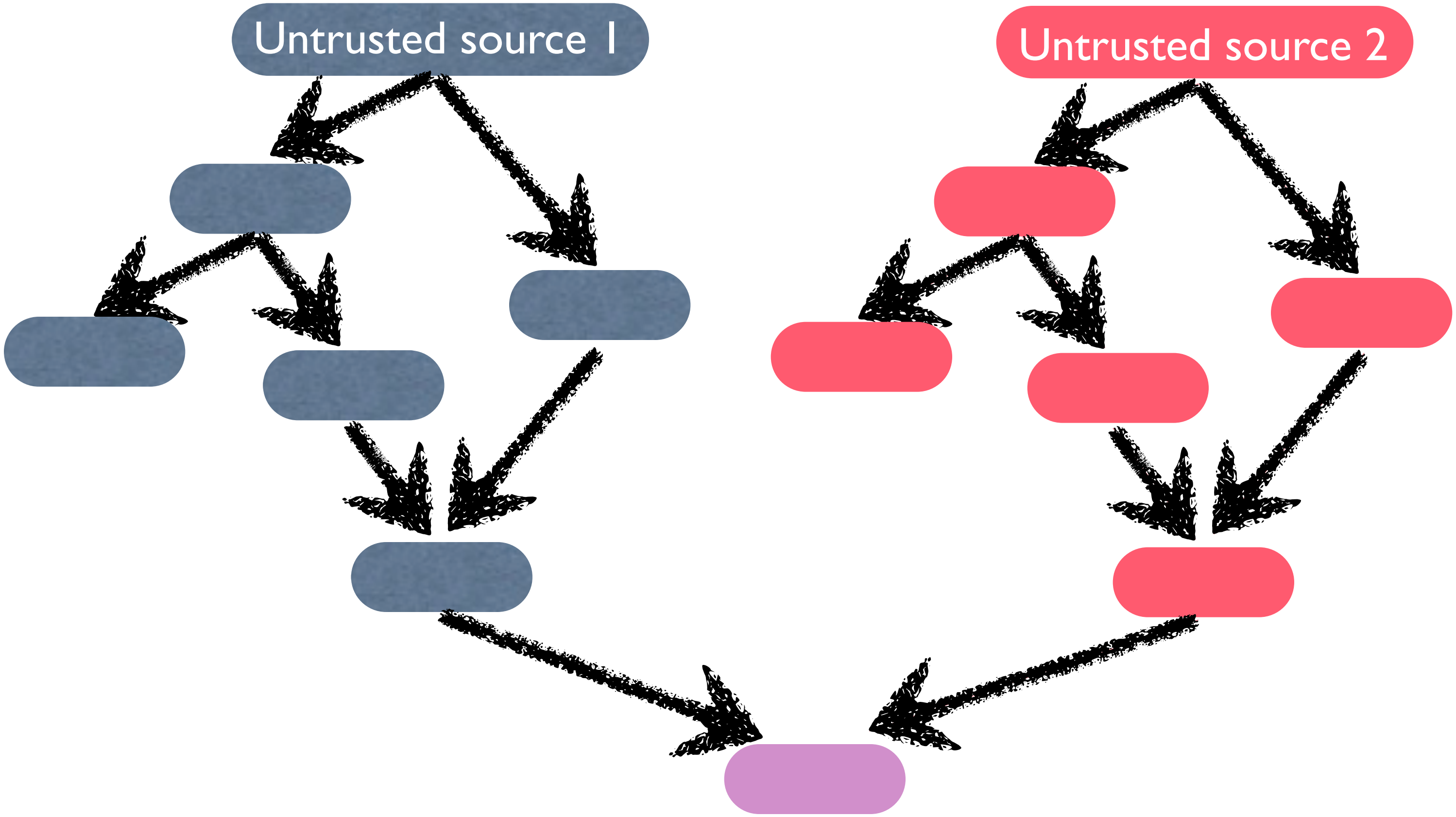
  Kind of information flow

  To see the flow from the external input effect

Taint propagation



Taint propagation



Use

- Detecting flaws

```
if ( tracking user data == available )
  I see where untrusted code swimming
```

- Data Lifetime Analysis

How ?

- Dynamic Binary Instrumentation Tools

Pin : Win & Linux & MAC, Intermediate Language

DynamoRIO : Win & Linux & MAC

TEMU : Win & Linux, QEMU based

Valgrind : Linux

How ?

- Use PIN Tool

  Windows, Linux, MAC OSX

  Custom Code (C or C++)

  Attach the running file

  Extensive API

  Pinheads

Pin ?

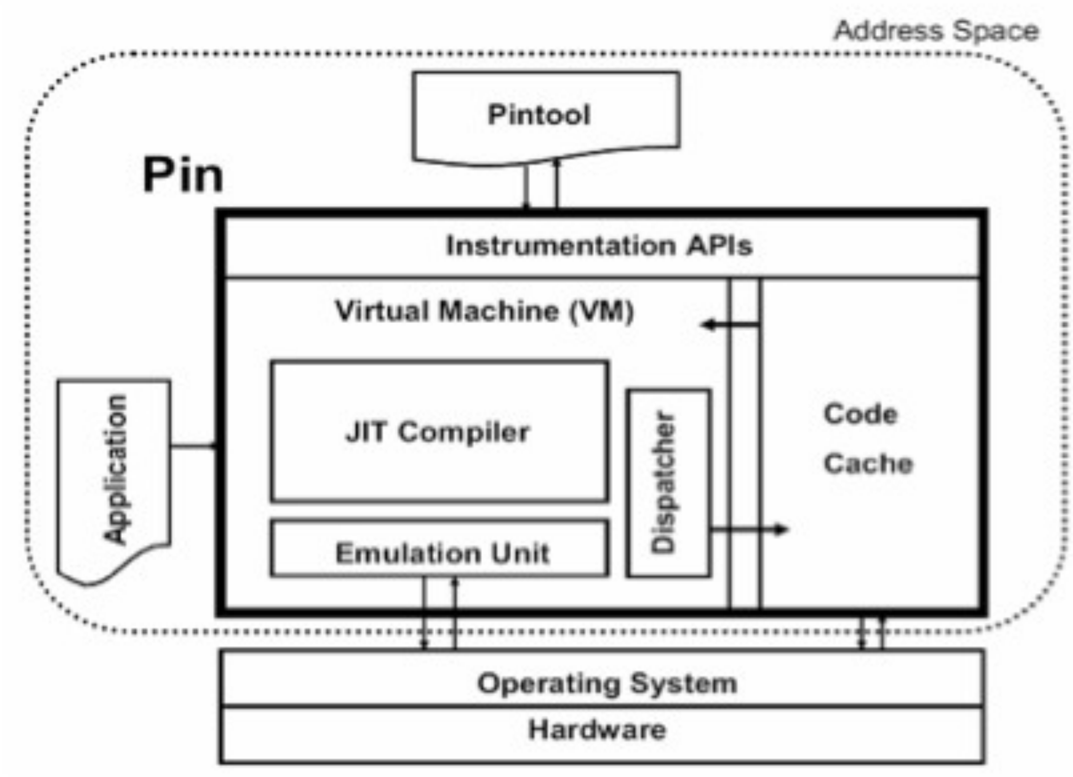
- http://pintool.org

  One of JIT (Just In Time) compiler

  Not input bytecode, but a regular executable

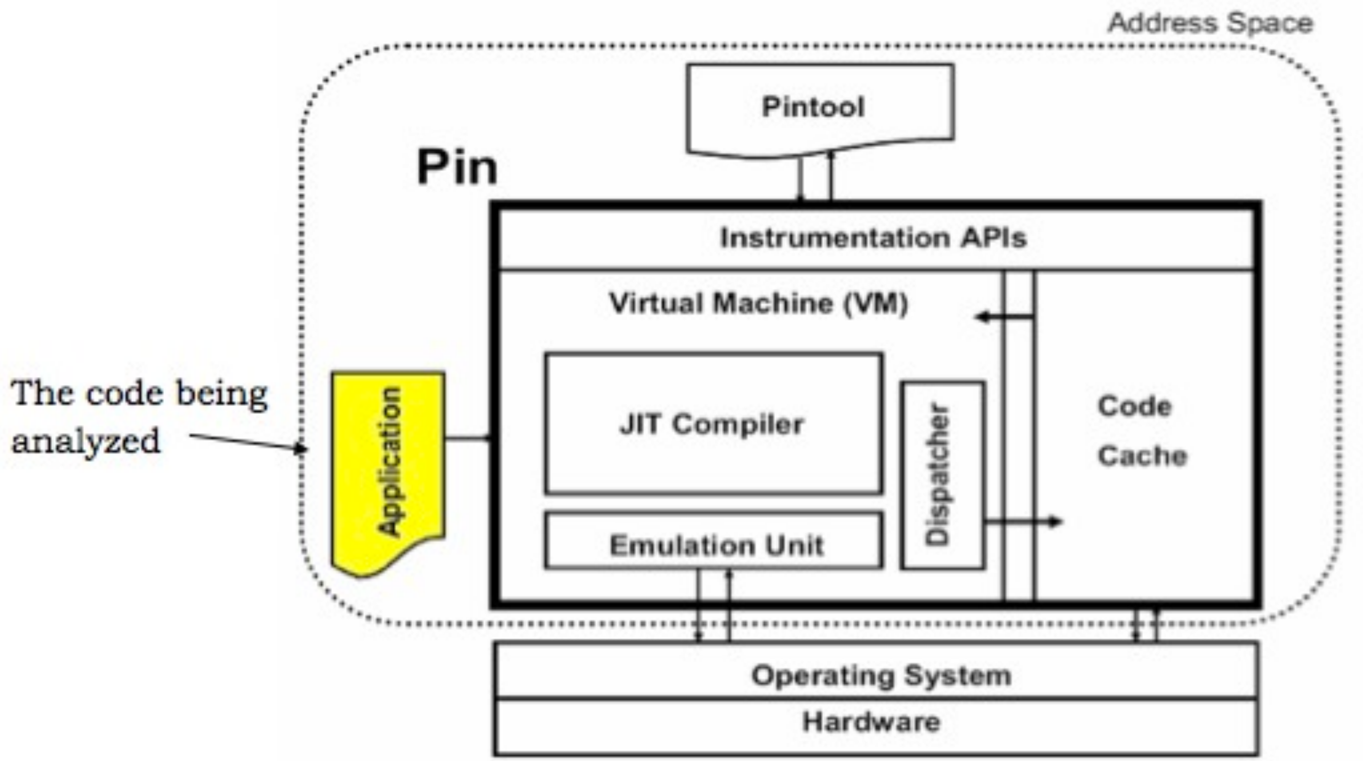
  Intercept instruction and generates more code and execute

Pin ?

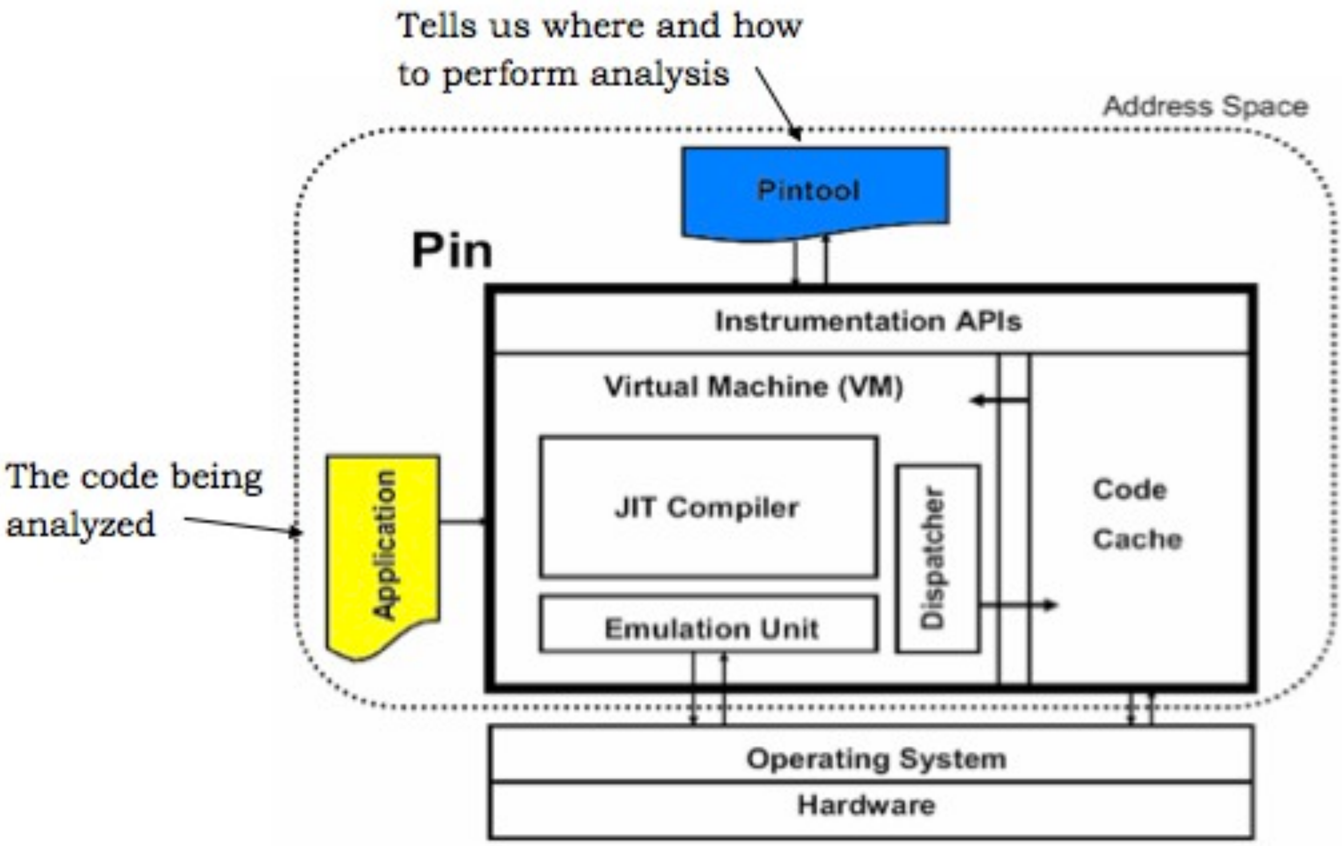


Pin : Instrumentation Engine
Pintool : Instrumentation Tool
Application : Target Program or Process

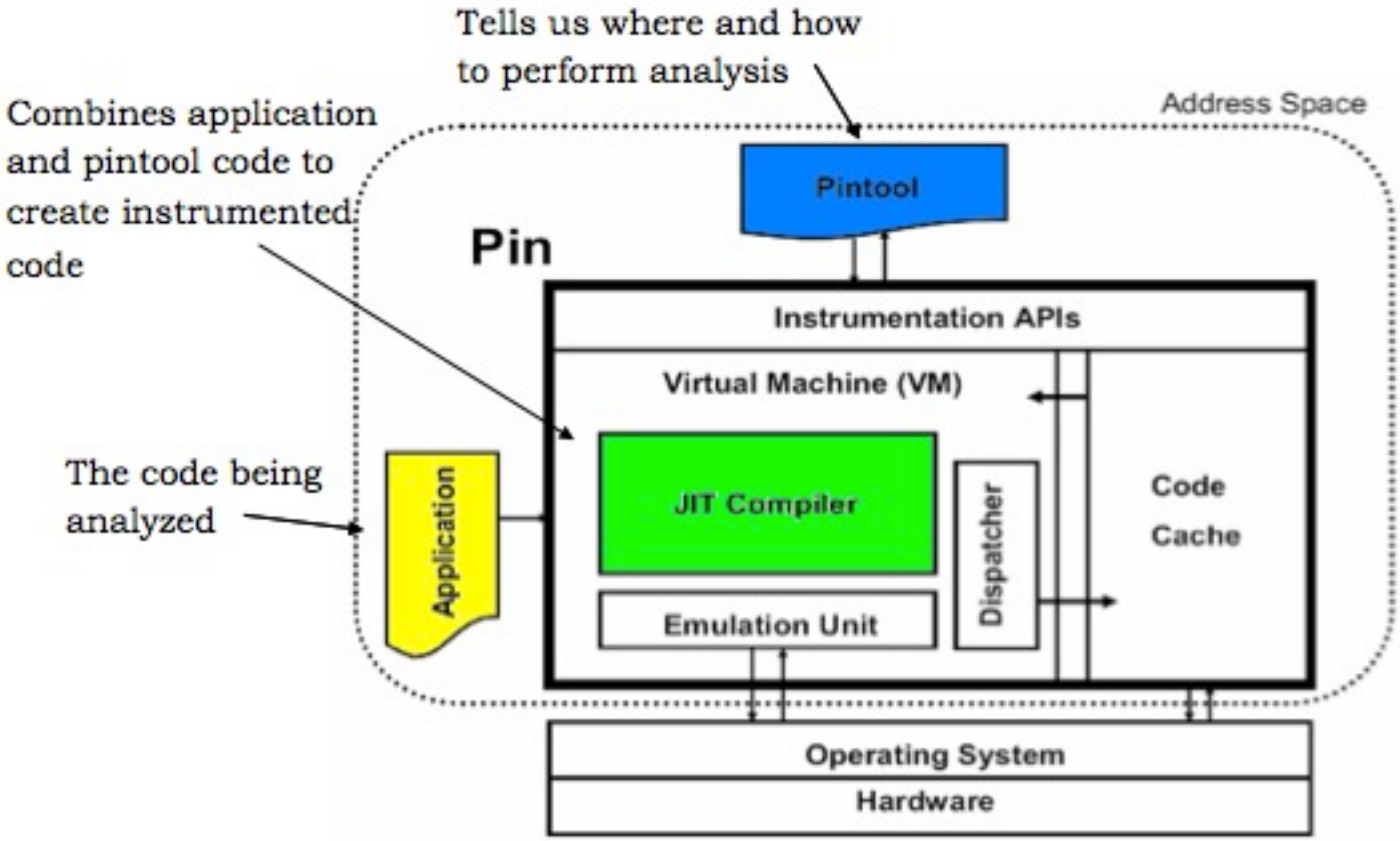
Pin ?


Address Space
Pintool
Pin
Instrumentation APIs
Virtual Machine (VM)
JIT Compiler
Emulation Unit
Dispatcher
Code Cache
Application
The code being analyzed
Operating System
Hardware

Pin ?


Tells us where and how to perform analysis
Address Space
Pintool
Pin
Instrumentation APIs
Virtual Machine (VM)
JIT Compiler
Emulation Unit
Dispatcher
Code Cache
The code being analyzed
Application
Operating System
Hardware

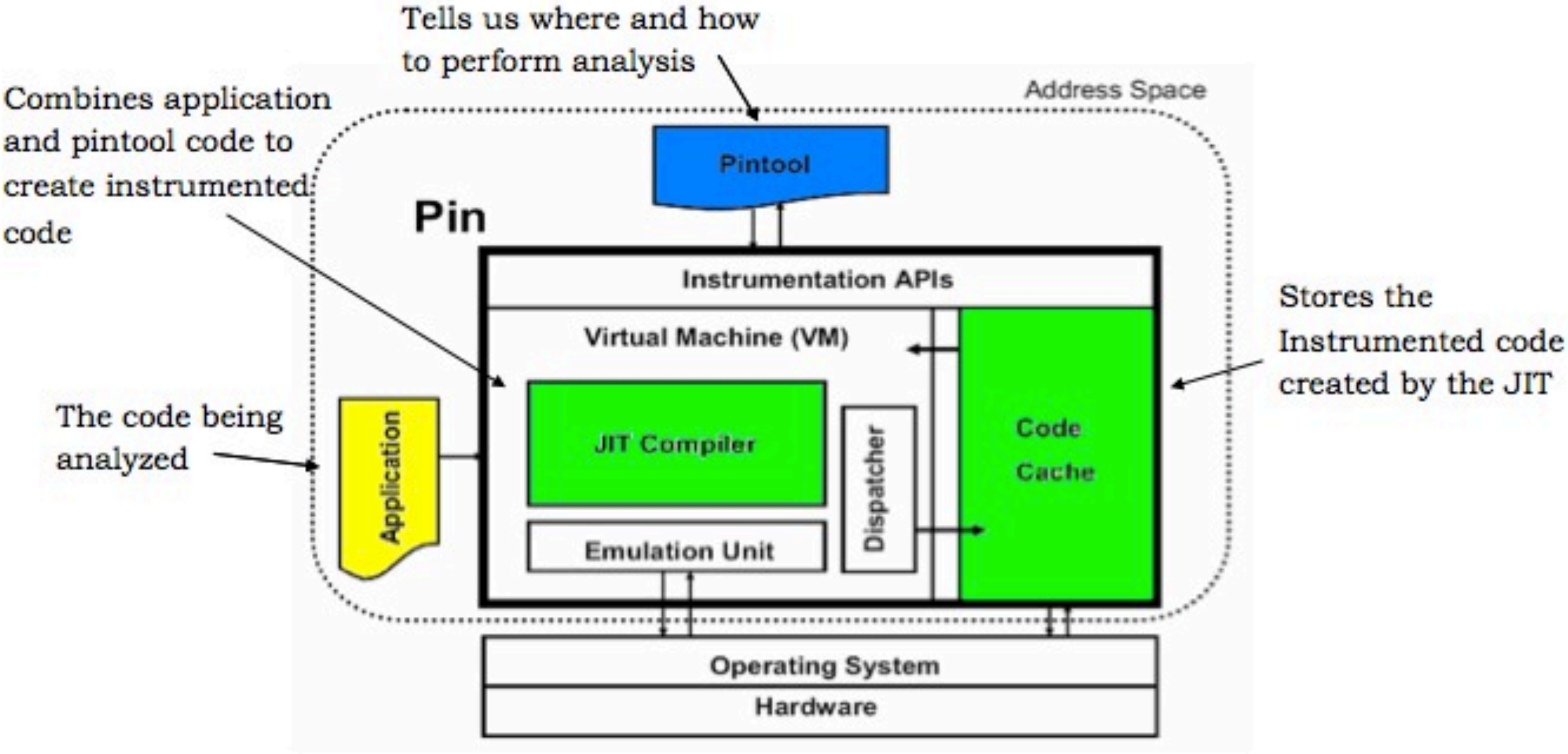
Pin ?


Tells us where and how to perform analysis

Combines application and pintool code to create instrumented code

The code being analyzed

Pin ?



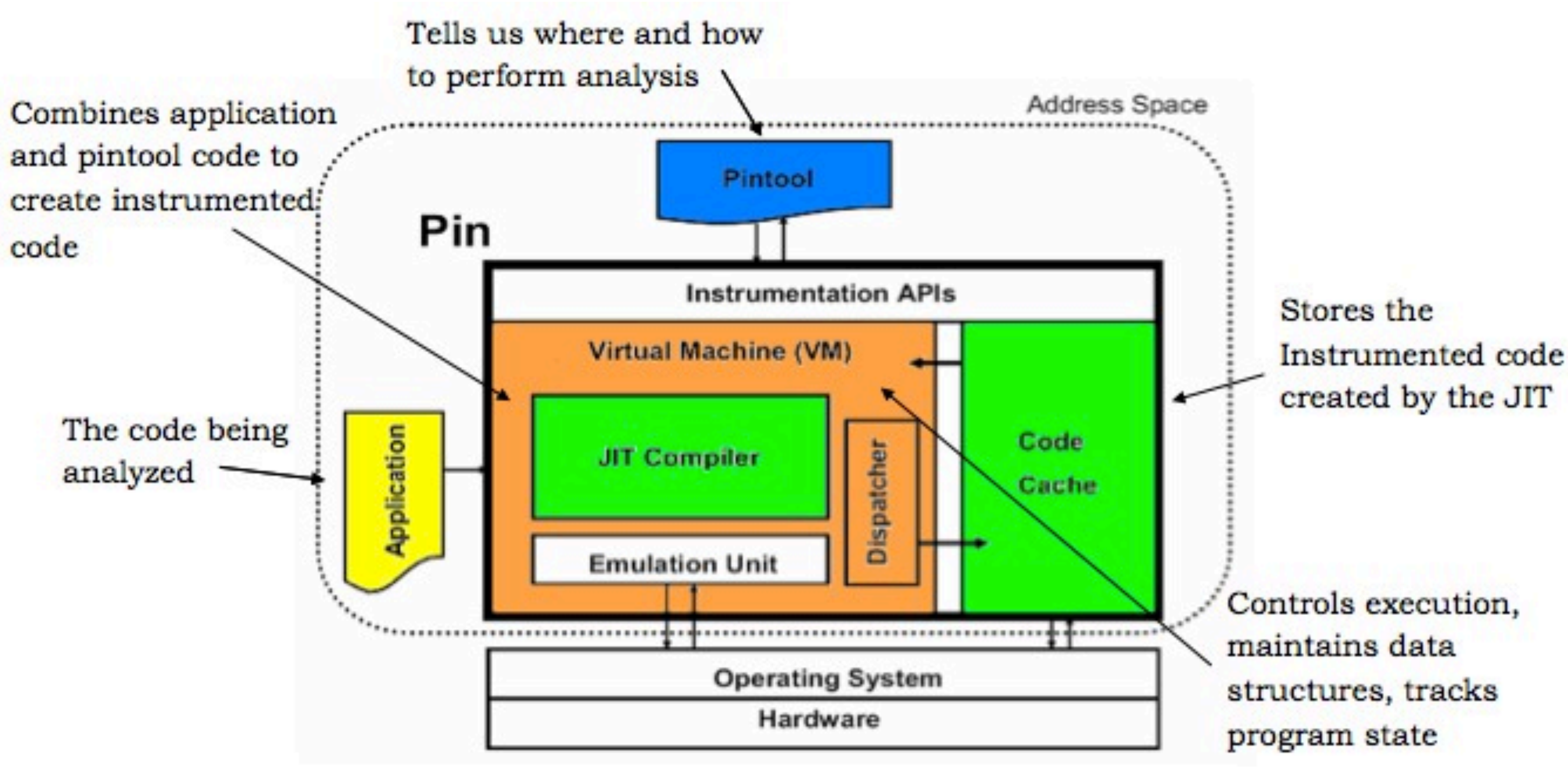
Tells us where and how to perform analysis

Combines application and pintool code to create instrumented code

The code being analyzed

Stores the Instrumented code created by the JIT

Pin ?



Tells us where and how to perform analysis

Combines application and pintool code to create instrumented code

Stores the Instrumented code created by the JIT

The code being analyzed

Controls execution, maintains data structures, tracks program state

Install

- if (Install window)

  you need to visual c++

- else if (install linux)

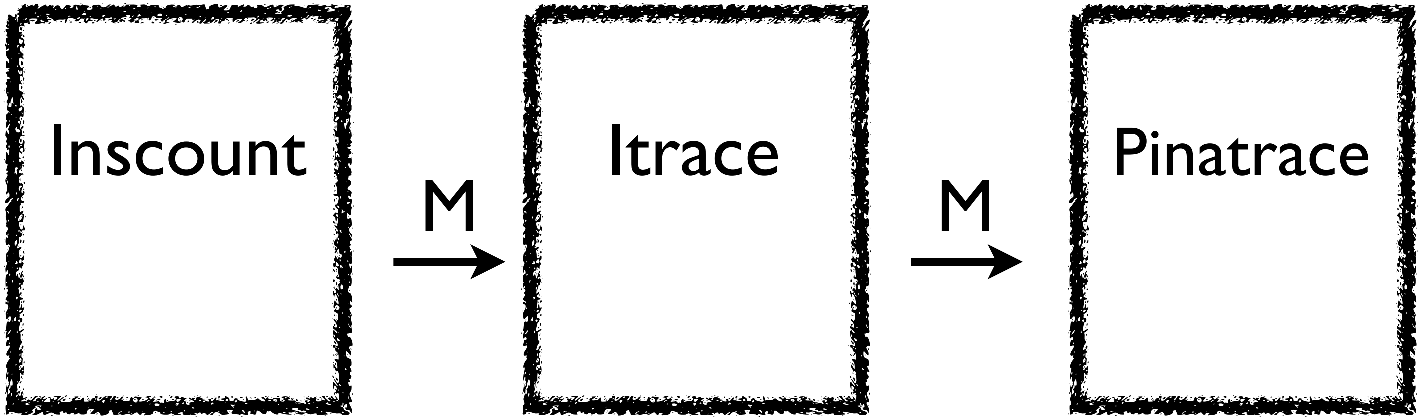
  you need to gcc-c++

- else if (install mac 64bit)

  not available

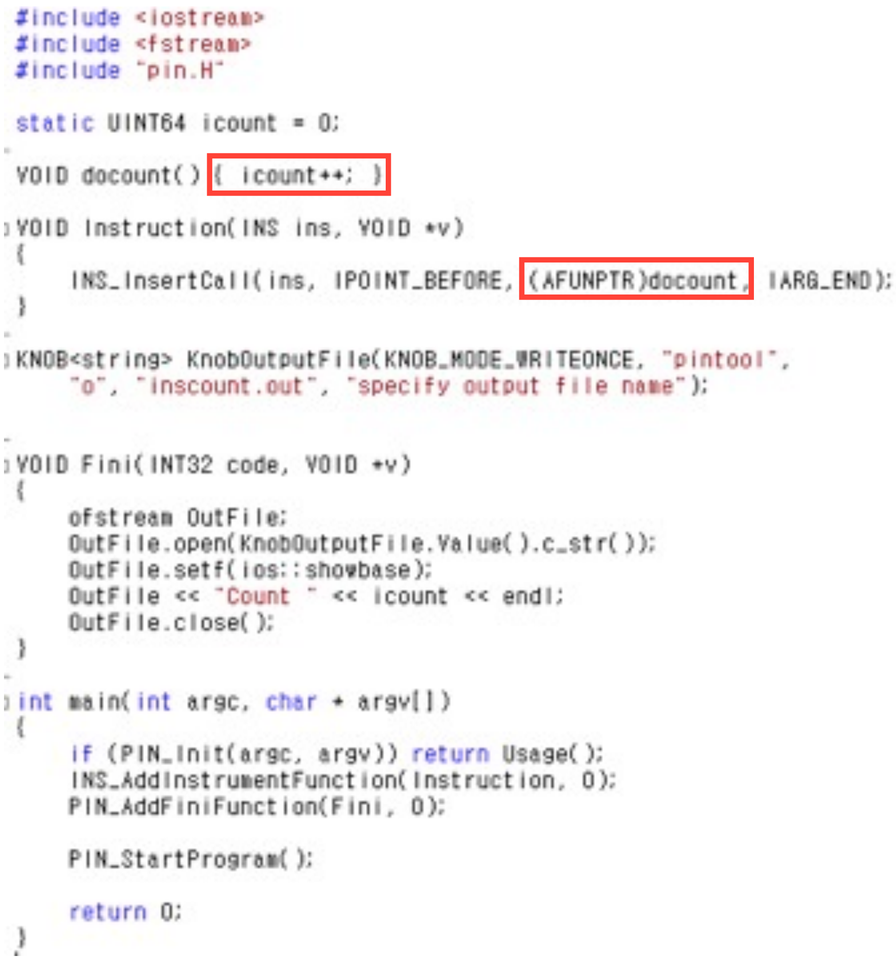
A Simple Example

- Inscount & Itrace & Pinatrace
- Step by modify code



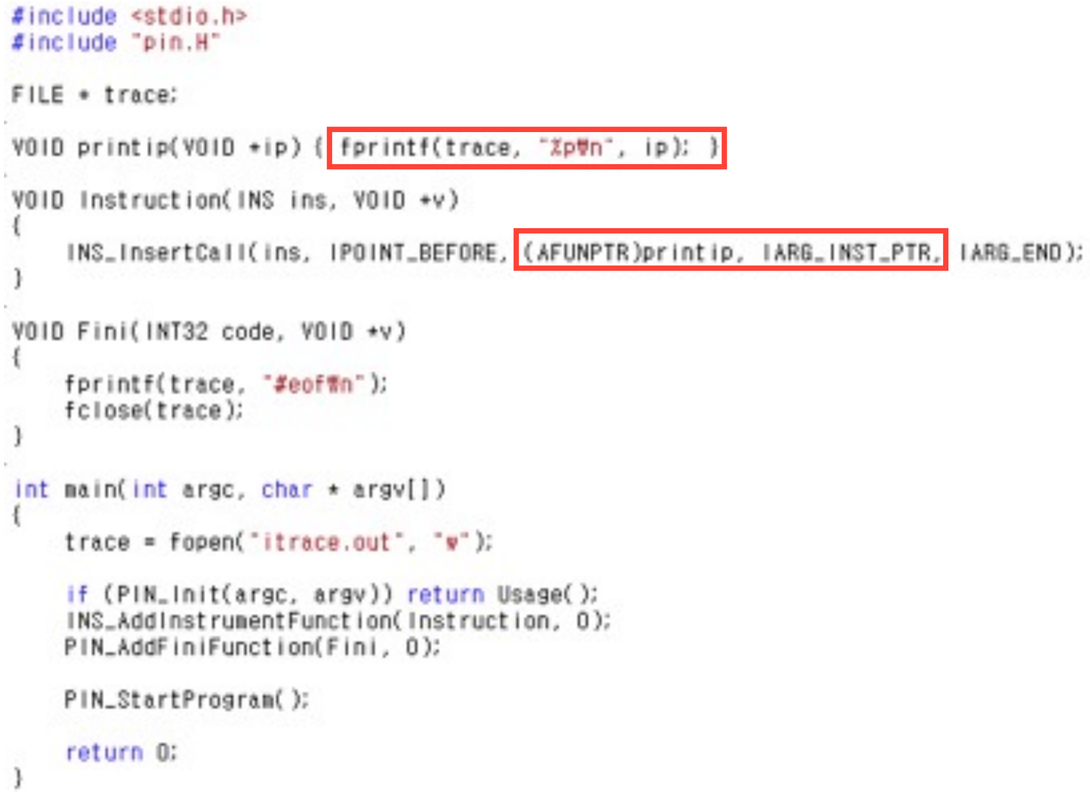
Inscount

- count the total number of instructions executed



```
#include <iostream>
#include <fstream>
#include "pin.H"

static UINT64 icount = 0;

VOID docount() { icount++; }

VOID Instruction(INS ins, VOID *v)
{
        INS_InsertCall(ins, IPOINT_BEFORE, (AFUNPTR)docount, IARG_END);
}

KNOB<string> KnobOutputFile(KNOB_MODE_WRITEONCE, "pintool",
        "o", "inscount.out", "specify output file name");


VOID Fini(INT32 code, VOID *v)
{
        ofstream OutFile;
        OutFile.open(KnobOutputFile.Value().c_str());
        OutFile.setf(ios::showbase);
        OutFile << "Count " << icount << endl;
        OutFile.close();
}

int main(int argc, char * argv[])
{
        if (PIN_Init(argc, argv)) return Usage();
        INS_AddInstrumentFunction(Instruction, 0);
        PIN_AddFiniFunction(Fini, 0);

        PIN_StartProgram();

        return 0;
}
```

Modify Inscount



```
#include <stdio.h>
#include "pin.H"

FILE * trace;

VOID printip(VOID *ip) { fprintf(trace, "%p₩n", ip); }

VOID Instruction(INS ins, VOID *v)
{
    INS_InsertCall(ins, IPOINT_BEFORE, (AFUNPTR)printip, IARG_INST_PTR, IARG_END);
}

VOID Fini(INT32 code, VOID *v)
{
    fprintf(trace, "#eof₩n");
    fclose(trace);
}

int main(int argc, char * argv[])
{
    trace = fopen("itrace.out", "w");

    if (PIN_Init(argc, argv)) return Usage();
    INS_AddInstrumentFunction(Instruction, 0);
    PIN_AddFiniFunction(Fini, 0);

    PIN_StartProgram();

    return 0;
}
```

Itrace

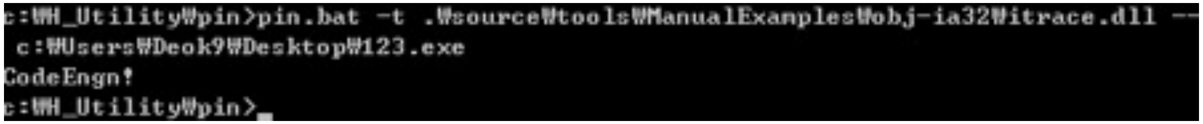
- Itrace

  Instruction Address Trace

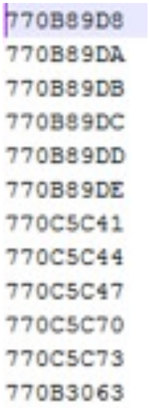
  How to pass arguments

  Useful understanding the control flow of a program for debugging

Itrace

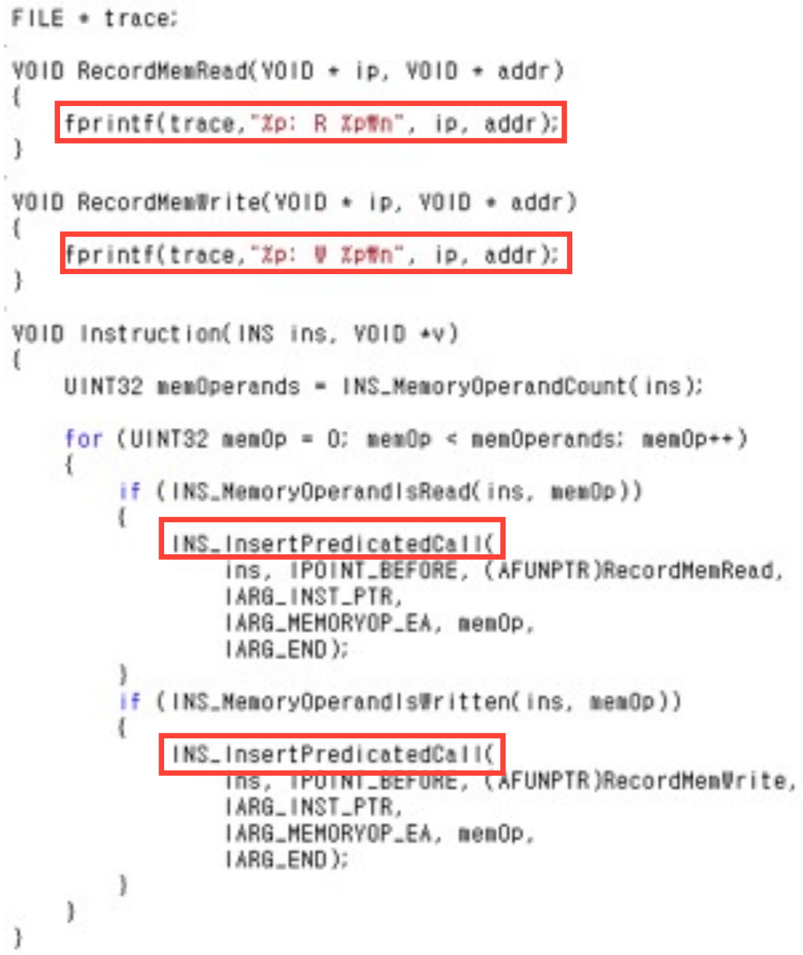


```
c:₩H_Utility₩pin>pin.bat -t .₩source₩tools₩ManualExamples₩obj-ia32₩itrace.dll --
 c:₩Users₩Deok9₩Desktop₩123.exe
CodeEngn!
c:₩H_Utility₩pin>_
```



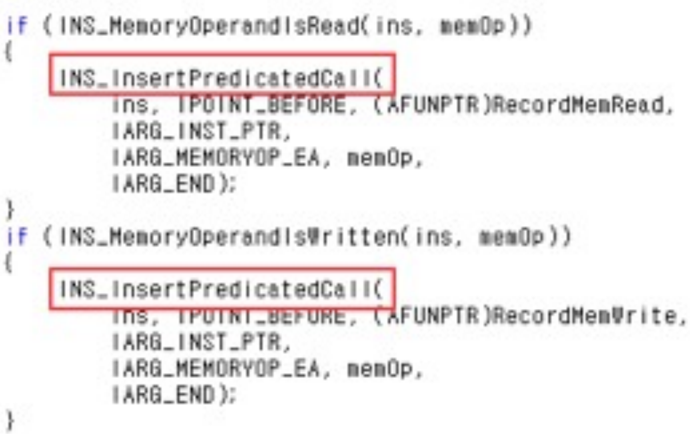
```
770B89D8
770B89DA
770B89DB
770B89DC
770B89DD
770B89DE
770C5C41
770C5C44
770C5C47
770C5C70
770C5C73
770B3063
```

Modify Itrace



```
FILE * trace;

VOID RecordMemRead(VOID * ip, VOID * addr)
{
    fprintf(trace,"%p: R %p₩n", ip, addr);
}

VOID RecordMemWrite(VOID * ip, VOID * addr)
{
    fprintf(trace,"%p: W %p₩n", ip, addr);
}

VOID Instruction(INS ins, VOID *v)
{
    UINT32 memOperands = INS_MemoryOperandCount(ins);

    for (UINT32 memOp = 0; memOp < memOperands; memOp++)
    {
        if (INS_MemoryOperandIsRead(ins, memOp))
        {
            INS_InsertPredicatedCall(
                ins, IPOINT_BEFORE, (AFUNPTR)RecordMemRead,
                IARG_INST_PTR,
                IARG_MEMORYOP_EA, memOp,
                IARG_END);
        }
        if (INS_MemoryOperandIsWritten(ins, memOp))
        {
            INS_InsertPredicatedCall(
                ins, IPOINT_BEFORE, (AFUNPTR)RecordMemWrite,
                IARG_INST_PTR,
                IARG_MEMORYOP_EA, memOp,
                IARG_END);
        }
    }
}
```

insertPredicatedCall ?



```
if ( INS_MemoryOperandIsRead( ins, memOp))
{
    INS_InsertPredicatedCall(
        ins, IPOINT_BEFORE, (AFUNPTR)RecordMemRead,
        IARG_INST_PTR,
        IARG_MEMORYOP_EA, memOp,
        IARG_END);
}
if ( INS_MemoryOperandIsWritten( ins, memOp))
{
    INS_InsertPredicatedCall(
        ins, IPOINT_BEFORE, (AFUNPTR)RecordMemWrite,
        IARG_INST_PTR,
        IARG_MEMORYOP_EA, memOp,
        IARG_END);
}
```

- To avoid generating references to instructions that are predicated when the predicate is false

- Predication is a general architectural feature of the IA-64

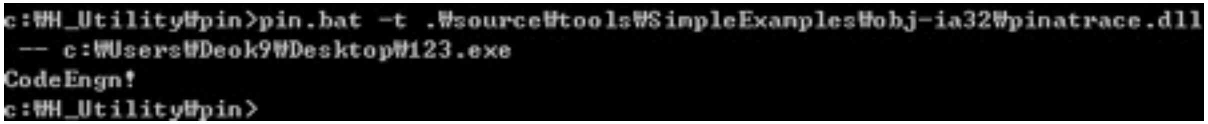
Pinatrace

- Pinatrace

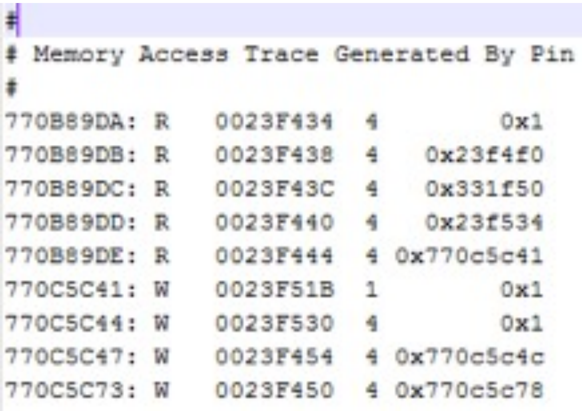
  Memory Reference Trace

  Useful debugging and for simulating a data cache in processor

Pinatrace



```
c:₩H_Utility₩pin>pin.bat -t .₩source₩tools₩SimpleExamples₩obj-ia32₩pinatrace.dll
 -- c:₩Users₩Deok9₩Desktop₩123.exe
CodeEngn!
c:₩H_Utility₩pin>
```



```
#
# Memory Access Trace Generated By Pin
#
770B89DA: R   0023F434  4         0x1
770B89DB: R   0023F438  4    0x23f4f0
770B89DC: R   0023F43C  4    0x331f50
770B89DD: R   0023F440  4    0x23f534
770B89DE: R   0023F444  4  0x770c5c41
770C5C41: W   0023F51B  1         0x1
770C5C44: W   0023F530  4         0x1
770C5C47: W   0023F454  4  0x770c5c4c
770C5C73: W   0023F450  4  0x770c5c78
```

770B89DA : Instrumentation Points

R/W : Access Type

0023F434 : &Address

4 : R/W Size

0x01 : *Address

Vera

- Use vera !

  Shmoocon 2011 Danny Quist

  Visualizing Executables for Reversing & Analysis

  Better OEP detection & IDA Pro Plugin

Demo !

- if (Use DBI with Vera)

  you will see the memory flow (easily)

- And

  you will see the pattern of vulnerable program and patched program

Demo !

Zero-day !

1. Hook Vulnerability Function

   strcpy, strcat, sprintf, scanf, fscanf, strstr, strchr

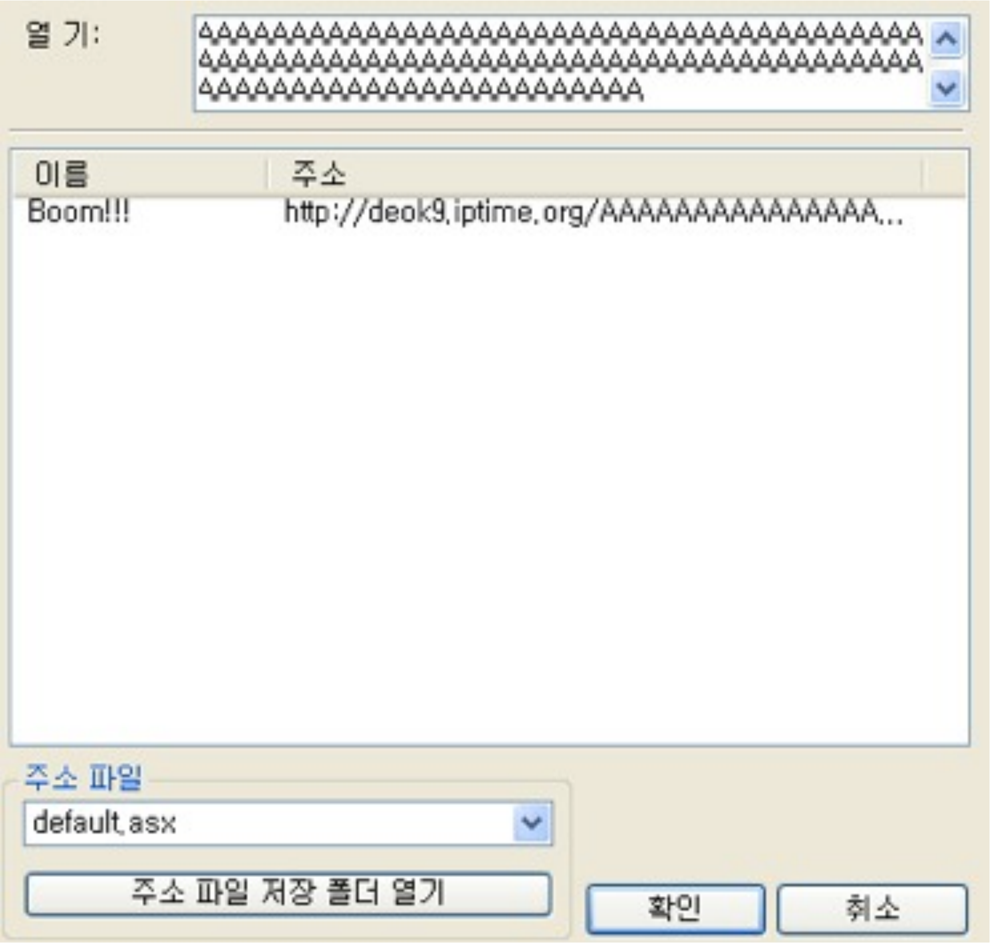
2. And

   monitoring ESI

3. Olleh!

   It's possible to modify the parameters

Zero-day !

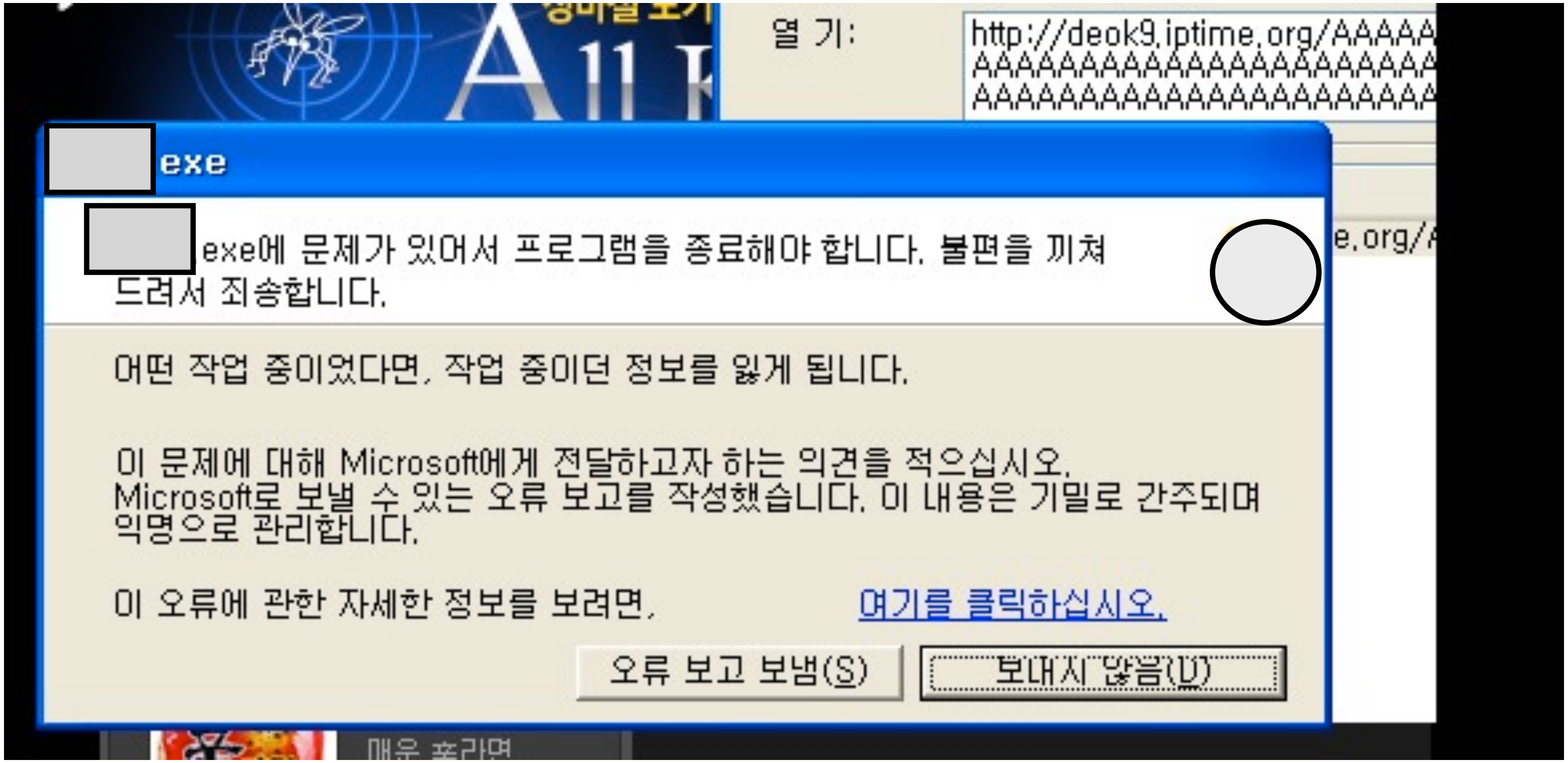


열 기: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA

이름	주소
Boom!!!	http://deok9.iptime.org/AAAAAAAAAAAAAAA...

주소 파일: default.asx

Zero-day !



열 기: http://deok9.iptime.org/AAAAA
AAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAA

[redacted] exe

[redacted] exe에 문제가 있어서 프로그램을 종료해야 합니다. 불편을 끼쳐 드려서 죄송합니다.

어떤 작업 중이었다면, 작업 중이던 정보를 잃게 됩니다.

이 문제에 대해 Microsoft에게 전달하고자 하는 의견을 적으십시오. Microsoft로 보낼 수 있는 오류 보고를 작성했습니다. 이 내용은 기밀로 간주되며 익명으로 관리합니다.

이 오류에 관한 자세한 정보를 보려면, 여기를 클릭하십시오.

reference

- http://translate.google.co.kr/?hl=ko&tab=wT
- http://www.pintool.org/
- http://www.youtube.com/watch?v=9nlWbDdxKjw

Q & A

Quiz

OR, XOR 연산에서

A 가 Taint 된 값(1) 이라고 가정했을 때

B 의 값이 무엇일 때 “Taint 되었다”

라고 할까요 ??

답과 간단한 이유를 말해주세용

hint) AND 연산일때 B 가 1일때 Taint 되었다.