HIVE 软件自动化逆向分析环境

李卷孺

GoSSIP @ LoCCS

密码与计算机安全实验室 · LoCCS
上海交通大学计算机系

HIVE：自动化程序分析环境

1. **程序逆向分析**
   a) 动态调试
   b) 静态反汇编、反编译
   c) 程序模拟执行
   d) 程序分析工具开发
2. **恶意软件分析**
   a) 虚拟化执行环境
   b) 代码反保护、反混淆
3. **程序黑盒分析**
   a) 网络协议分析
   b) 系统调用监控

概览

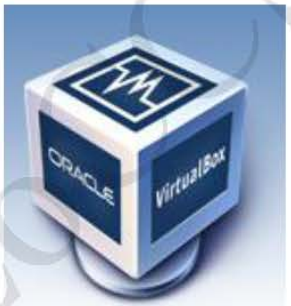
- 环境搭建
- 程序插桩
- 算法分析

A Virtual Environment for Program Analysis

环境搭建

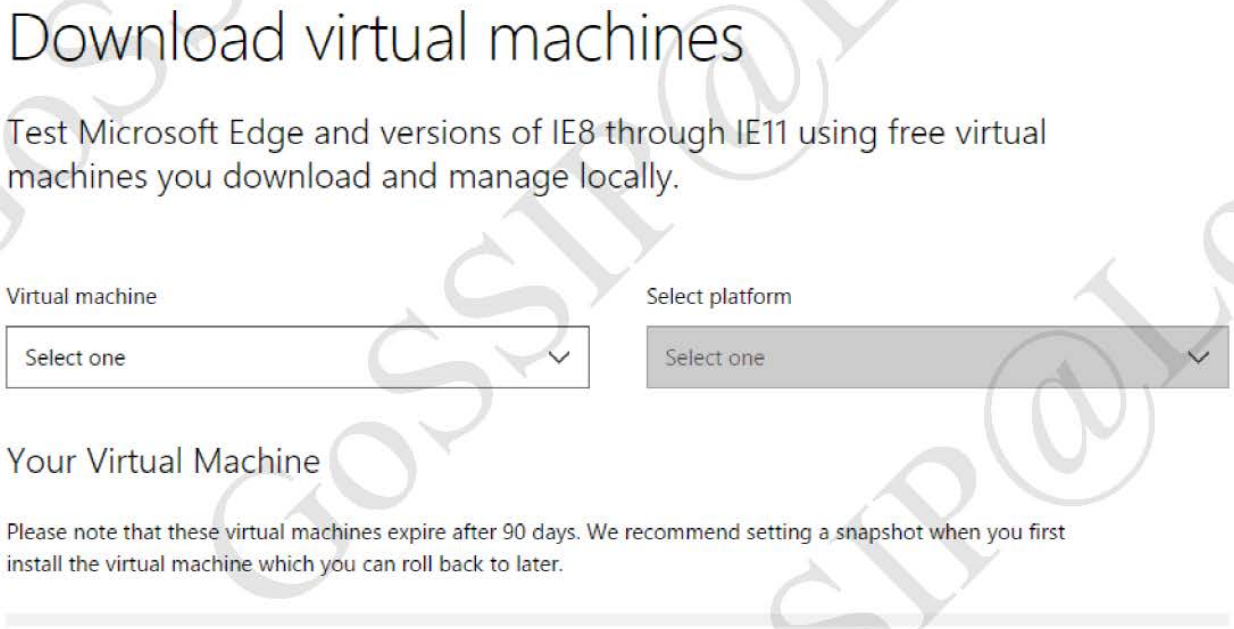
安装VirtualBox

- https://www.virtualbox.org/



安装试用版Windows虚拟机

- https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/



试用版Windows虚拟机

- Windows XP 32bit

安装调试工具

- **IDA Demo**
  - **v5.0**
- **Ollydbg**
  - **v1.10 with Plugins**
- **Intel Pin Instrumentation Framework**
  - **pin-2.14-71313 (msvc9: Visual Studio 2008)**

安装开发工具

- Python 2.7
- Visual Studio Express 2008
- Notepad2
- Wireshark or Microsoft Message Analyzer
- HEdit

搭建Kaleidoscope环境

- **Python Executor + Pin Engine**
- **Three basic program analysis pintools**
  - **Profiler**
  - **MemInspector**
  - **Kscope**
- **A pintool development environment**
  - **Based on Visual Studio express**

Program Instrumentation

程序插桩

Binary Code Instrumentation

- Concept
  - https://www.utdallas.edu/~zxl111930/spring2012/public/lec4.pdf
- Tools
  - QEMU
  - PIN
  - Valgrind
  - DynamoRio

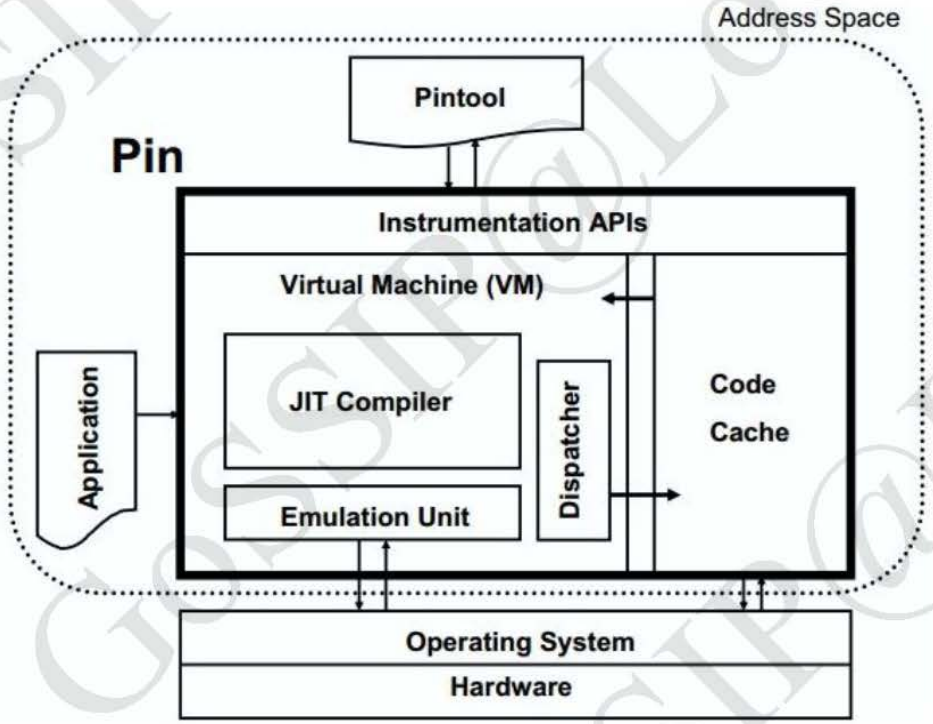
Pin Framework

- Pin是由Intel开发的一个支持多种平台的二进制代码插桩框架

- Pin在运行时对于其所监控的程序来说是完全透明的（上帝视角），使用动态编译（JIT）的方法，结合内联、寄存器重分配、活跃性分析以及插桩指令的一些优化性的调度等策略，对程序进行动态监控，且相对于传统的工具DynamoRIO和Valgrind来说，更加高效。

Pin Framework

- 从高层来看，PIN由VM, 代码缓存（code cache）,以及一个由PINTOOL调用的插庄API组成



Pin Framework

- 在Pin的开发过程中，将所有的程序分为3类：Pin, Pintool, APP。
  - Pin即INTEL官方提供的一个已经编译好的可执行程序
  - Pintool是开发者基于PIN官方的各种API开发出来的辅助工具，也即我们的主要debug战场，以及各种链接进来的需要与pin进行通信的库。
  - APP即为我们要用PIN来监控的程序或进程（可在pin参数中指定某个进程的pid，通过attach的方式来监控进程）

Pin Framework

- Pin拦截APP（进程）的第一条指令（当前指令），并生成从这条指令起的线性代码序列，然后将控制转到生成的序列上去，当遇到分支跳转时，pin重新获得控制权限

- 转换和插桩后的代码被放在一个代码缓存中，以便以后再次执行时提高效率

```
/* ===================================================================== */
/* Main                                                                  */
/* ===================================================================== */
int main(int argc, char *argv[])
{
    // Initialize pin & symbol manager
    PIN_InitSymbols();
    if( PIN_Init(argc,argv) )
    {
        return Usage();
    }
    // Write to a file since cout and cerr maybe closed by the application
    TraceFile.open(KnobOutputFile.Value().c_str());
    TraceFile << hex;
    TraceFile.setf(ios::showbase);

    // Register Image to be called to instrument functions.
    IMG_AddInstrumentFunction(Image, 0);
    PIN_AddFiniFunction(Fini, 0);

    // Never returns
    PIN_StartProgram();

    return 0;
}
```

Pin Framework

- 教学
  - Intel Pin 1 ：如何使用Pin进行插桩 | Star
  - https://huirong.github.io/2015/12/30/Intel-Pin-introduction/#Pin
- 官方文档
  - http://software.intel.com/sites/landingpage/pintool/docs/58423/Pin/html//
  - http://software.intel.com/sites/default/files/article/256675/cgo2013.pdf
- 论文
  - http://www.cs.virginia.edu/kim/courses/cs851/papers/luk05pin.pdf
  - http://scale.eecs.berkeley.edu/papers/pin-wbia.pdf
  - http://www.ckluk.org/ck/papers/pin_ieeecomputer10.pdf

Pintools

- Pin仅仅提供了插桩分析的引擎，接下来需要做的工作包括：
  - 确定在哪里插入什么代码
  - 插入的代码是什么？
- 我们使用一个动态链接库（DLL或so）来完成上述工作，将其称为Pintool
- Pintools为自定义程序分析提供了无限可能

Diff Slicing

- Differential Slicing
  - 通过两次运行程序，分别执行不同的功能，求指令的差集
  - 有效排除不相关指令

Profiling

- Instruction Profiling
  - 统计程序运行过程中，每条指令的运行次数，从而分析出程序哪些部分是执行关键
  - 另一方面，获取程序运行过程中执行到指令，可以减少分析的内容
  - 定量分析，可以多次分析，进行比例量化

Mem Analysis

- Memory Input and Output Analysis
  - 通过编写相关的Pintools，对程序运行时输入输出数据分析
  - 注意：本工具仅仅针对基本块，否则开销会非常大！

Tracing

- Kaleidoscope：a slim Tracer
  - Tracing：不仅记录程序运行时的指令，还要记录执行时的context
  - 全面的记录每一条执行的指令及相关信息，将导致巨大的运行时开销和存储开销，因此我们需要精简记录的内容
  - Kaleidoscope采取了两方面的简化措施
    - 只记录部分指令的内容（通过KsAddrFilter.cfg指定要记录的范围）
    - 每条指令只记录有限次（如果一条指令循环执行多次，我们并不需要关心它整个执行过程，通常情况下64次即可分析出相关信息）

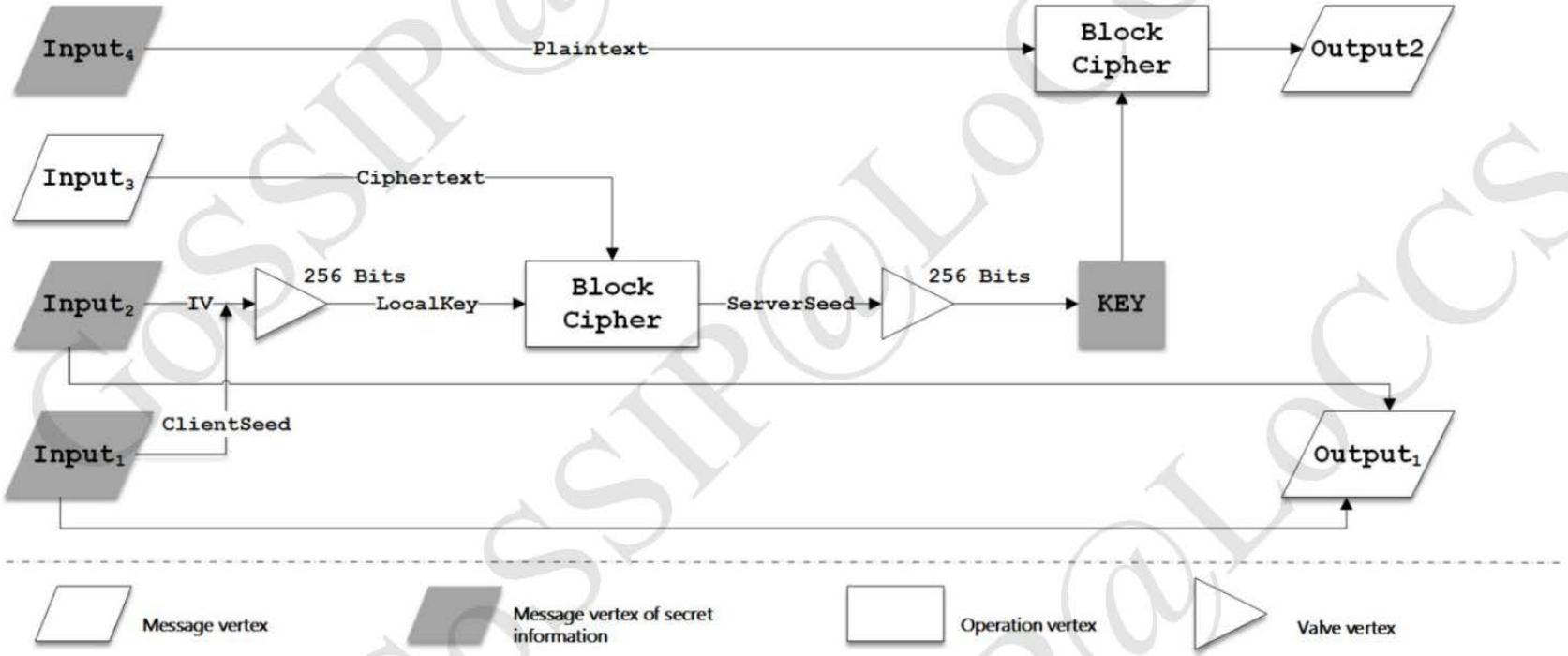
SiNan

密码软件分析

密码软件的自动化分析

- 司南分析系统
  - 二进制代码动态运行
  - 密码元素识别
  - 模型重建
  - 攻击模拟和漏洞发掘

Crypto Workflow Graph



算法识别

- 充分利用输入数据
  - 输入数据可以被归类为groups
  - 输入/输出数据和加密算法的执行呈线性相关关系
  - 输入数据可以作为information flow的线索来定位加密算法
  - 检测数据的Entropy变化
- 密钥提取
  - 密钥的运算特殊性：xor或者乘法模幂运算
- 二进制代码相似性检测
  - 利用已有的密码学算法库知识
  - 自动化匹配现有库

安全分析

- 敌手能力评估
  - 根据CWG，分析得出敌手拥有的知识量
  - 根据敌手拥有的知识量，推演敌手是否能够获得秘密信息（Security Goal）
- 可利用基础
  - 前向信息流分析
  - 后向规则推导： CWG定义的规则

Case Study

- A file encryption tool – Challenger
  - 首先通过Diff Slicing和Profiling分析程序的关键指令
  - 将关键指令按照基本块划分为不同的关键指令集
  - 然后通过内存运行分析监控各个关键指令集
  - 通过Tracing观察关键指令的运行情况，找到加密过程
  - 提取密钥


GoSSIP @ LoCCS