

动态二进制检测框架Pin简介及API Log Tool

本文原创作者：比尔.盖茨

Pin工具的介绍

Pin 是一种动态二进制检测框架，适用于x86,x64架构，一般用于程序动态分析用，是多个平台都支持，windows、linux以及OSX。该工具原本适用于计算机架构分析用的，但是由于丰富的api，与强大的功能现在运用的地方很多。比如计算机安全，环境模拟器，并行计算。

Pin相当于一个JIT ("just in time ") 编译器，并且有相当数量的实例代码(这一点真不错)，对于新手来说上手是很快的。但是貌似在国内安全界很少被公开提及，但我知道还是用的。对于这样的一个好工具，我觉得再不介绍给大家真不好意思了。

Pin大体上有三种插装代码模式(指令级，rtn级，bbl级)，

指令级的执行过程大体上如下图：

```
movzx ecx, [rax+0x2]
call 0x77ef7870
cmp rax, rdx
jz 0x77f1eac9
```



```
..some code
..some code
movzx ecx, [rax+0x2]
..some code
..some code
call 0x77ef7870
..some code
..some code
cmp rax, rdx
..some code
..some code
jz 0x77f1eac9
```

REEBUF



下面讲讲pin 工具的功能：

1. 替换原有程序函数
2. 探测程序任意指令，在自己设定的插入位置插入自己的代码并执行
3. 记录程序调用，包括syscall(检测改变参数)

4. 记录程序线程活动情况
5. 监测进程树
6. 模拟api调用（这里可以思维宽广点。。。）

Pin的三种插入（代码）模式：

1 . Instruction level (Ins)

即在每一条原始程序指令前，后插入自己的代码，即调用这个函数，这里我不详细讲，因为pin的文档很全面，自己去就行。但是有一点这个级别执行可想而知，会执行的很慢，很耗时间。（因为在每条指令上加入代码，即使自己的代码只包含一行(ps: nop)，代码就至少膨胀5倍以上）

2. Function level(RTN)

Pin通过符号表信息来找到这些需要插入的位置，要使用这种模式的代码插入，首先得调用pin内置的初始化符号表函数，即PIN_InitSymbols()

3. Basic block level(BBL)

即基本调用块级别,插入模式，只在trace时候可用。

下图为pin的执行过程：

0. 启动自己（pin.exe），以debugger的模式，附加要分析程序的进程
1. 注入pinvm.dll到宿主程序中
2. 注入自己的工具代码(即pintool.dll)
3. 激活主程序进程

然后的过程就是程序执行与pin代码执行的过程了。

下图为其各个部分的具体功能：





大灰狼:

```

0x402d3b*Sleep(0)=>>0x12c234
0x402d3f*Sleep(0)=>>0x12c234
0x402d91*LoadLibraryA("WININET.dll")=>>0x1
0x402d98*GetProcAddress(0x76260000,"InternetOpenUrlA")=>>0x76312caa
0x402e23*Sleep(0)=>>0x12c9b4
0x402e27*Sleep(0)=>>0x12c9b4
0x402e65*LoadLibraryA("KERNEL32.dll")=>>0x1
0x402e6c*GetProcAddress(0x77880000,"CreateFileA")=>>0x778cea61
0x402e81*CreateFileA("F:\Program Files\AppPatch\mysql.d.dll",0x40000000,0,0,0x2,0,0)=>>0x12c964
0x40346f*Sleep(0)=>>0x12ddb4
0x40347b*Sleep(0)=>>0x12ddb4
0x40347f*Sleep(0)=>>0x12ddb4
0x403483*Sleep(0)=>>0x12ddb4
0x403487*Sleep(0)=>>0x12ddb4
0x40348b*Sleep(0x14)=>>0x12ddb4
0x402610*LoadLibraryA("KERNEL32.dll")=>>0x1
0x402619*GetProcAddress(0x77880000,"CreateFileA")=>>0x778cea61
0x402638*CreateFileA("C:\Program Files\AppPatch\mysql.d.dll",0x80000000,0x1,0,0x3,0,0)=>>0x12d514
0x4026b5*LoadLibraryA("KERNEL32.dll")=>>0x1
0x4026bc*GetProcAddress(0x77880000,"CreateFileA")=>>0x778cea61
0x402707*CreateFileA("C:\Program Files\AppPatch\mysql.d.dll",0x40000000,0,0,0x2,0,0)=>>0x12cd64
0x402610*LoadLibraryA("KERNEL32.dll")=>>0x1
0x402619*GetProcAddress(0x77880000,"CreateFileA")=>>0x778cea61
0x402638*CreateFileA("D:\Program Files\AppPatch\mysql.d.dll",0x80000000,0x1,0,0x3,0,0)=>>0x12d514

```

```

0x4026b5*LoadLibraryA("KERNEL32.dll")=>>0x1
0x4026bc*GetProcAddress(0x77800000,"CreateFileA")=>>0x778cea61
0x402707*CreateFileA("D:\Program Files\AppPatch\mysqld.dll",0x40000000,0,0,0x2,0,0)=>>0x12cd64
0x402610*LoadLibraryA("KERNEL32.dll")=>>0x1
0x402619*GetProcAddress(0x77800000,"CreateFileA")=>>0x778cea61
0x402638*CreateFileA("E:\Program Files\AppPatch\mysqld.dll",0x80000000,0x1,0,0x3,0,0)=>>0x12d514
0x4026b5*LoadLibraryA("KERNEL32.dll")=>>0x1
0x4026bc*GetProcAddress(0x77800000,"CreateFileA")=>>0x778cea61
0x402707*CreateFileA("E:\Program Files\AppPatch\mysqld.dll",0x40000000,0,0,0x2,0,0)=>>0x12cd64
0x402610*LoadLibraryA("KERNEL32.dll")=>>0x1
0x402619*GetProcAddress(0x77800000,"CreateFileA")=>>0x778cea61
0x402638*CreateFileA("F:\Program Files\AppPatch\mysqld.dll",0x80000000,0x1,0,0x3,0,0)=>>0x12d514
0x4026b5*LoadLibraryA("KERNEL32.dll")=>>0x1
0x4026bc*GetProcAddress(0x77800000,"CreateFileA")=>>0x778cea61
0x402707*CreateFileA("F:\Program Files\AppPatch\mysqld.dll",0x40000000,0,0,0x2,0,0)=>>0x12cd64
0x402abc*Sleep(0)=>>0x12c9b4
0x402ac0*Sleep(0)=>>0x12c9b4
0x402b01*LoadLibraryA("KERNEL32.dll")=>>0x1
0x402b0a*GetProcAddress(0x77800000,"CloseHandle")=>>0x778ce868
0x402b55*LoadLibraryA("WININET.dll")=>>0x1
0x402b58*GetProcAddress(0x76260000,"InternetReadFile")=>>0x762859b8
0x402b9c*LoadLibraryA("KERNEL32.dll")=>>0x1
0x402ba3*GetProcAddress(0x77800000,"WriteFile")=>>0x778d53ee
0x402bb3*Sleep(0)=>>0x12d164
0x402bb7*Sleep(0)=>>0x12d164
0x402c0a*Sleep(0)=>>0x12c234
0x402c0e*Sleep(0)=>>0x12c234
0x402c54*LoadLibraryA("WININET.dll")=>>0x1
0x402c5b*GetProcAddress(0x76260000,"InternetOpenA")=>>0x7628ba6d
0x402d3b*Sleep(0)=>>0x12c234
0x402d3f*Sleep(0)=>>0x12c234
0x402d91*LoadLibraryA("WININET.dll")=>>0x1
0x402d98*GetProcAddress(0x76260000,"InternetOpenUrlA")=>>0x76312caa

```

Zbot dump出来的文件

..	Up
1.dump	177737
2.dump	10
3.dump	10
4.dump	5
5.dump	5

其中1.dump为一个pe文件

```

08 D:\Sample\vmwaresample\zbot\1.dump
00000000: 40 5A 90 00-03 00 00 00-04 00 00 00-FF FF 00 00  2?  []  []
00000010: 88 00 00 00-00 00 00 00-40 00 00 00-00 00 00 00   ?  []  e
00000020: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000030: 00 00 00 00-00 00 00 00-00 00 00 00-C8 00 00 00  ?
00000040: 0E 1F BA 0E-00 B4 09 CD-21 B8 01 4C-CD 21 54 68  []v[] ?[]?[]!Th
00000050: 69 73 20 70-72 6F 67 72-61 6D 20 63-61 6E 6E 6F  is program canno
00000060: 74 20 62 65-20 72 75 6E-20 69 6E 20-44 4F 53 20  t be run in DOS
00000070: 6D 6F 64 65-2E 0D 0D 0D-0A 24 00 00-00 00 00 00  mode.[] [] []$
00000080: 00 53 E1 76-DA 17 80 18-89 17 80 18-89 17 80 18  S?v?g?g?g?g?g?
00000090: 89 99 9F 08-89 34 80 18-89 EB A0 0D-0A 89 16 80  ??? ? 74????[]?
000000A0: 18 89 D0 86-1E 89 16 80-18 89 52 69-63 68 17 80  ??? ?  ?Richg
000000B0: 18 89 00 00-00 00 00 00-00 00 00 00-00 00 00 00  ??
000000C0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  PE L[]
000000D0: 04 00 0C 49-DC 4B 00 00-00 00 00 00-00 00 E0 00  []  ?I?K  ?
000000E0: 0F 01 0B 01-05 00 00 0C-00 00 00 7C-00 00 00 00  []  ?[]  ?  I
000000F0: 00 00 2C 15-00 00 00 10-00 00 00 20-00 00 00 00  , $  ?  ?
00000100: 40 00 00 10-00 00 00 02-00 00 04 00-00 00 00 00  e  ?  []  []
00000110: 00 00 04 00-00 00 00 00-00 00 00 70-02 00 00 04  []  ?  p[]  []
00000120: 00 00 00 00-00 00 02 00-00 00 00 00-10 00 00 10  []  ?  ?  ?
00000130: 00 00 00 00-10 00 00 10-00 00 00 00-00 00 10 00  ?  ?  ?  ?
00000140: 00 00 00 00-00 00 00 00-00 00 08 21-00 00 64 00  ?  ?  !  ?
00000150: 00 00 00 00-02 00 A9 68-00 00 00 00-00 00 00 00  []  ?h
00000160: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000170: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000180: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000190: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000001A0: 00 00 30 20-00 00 D8 00-00 00 00 00-00 00 00 00  0  ?
000001B0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00

```



```
000001C0: 00 00 2E 74-65 78 74 00-00 00 EC 0B-00 00 00 10 .text ?  
000001D0: 00 00 00 0C-00 00 00 04-00 00 00 00-00 00 00 00 ?  
000001E0: 00 00 00 00-00 00 20 00-00 60 2E 72-64 61 74 61 .rdata  
000001F0: 00 00 CE 05-00 00 00 20-00 00 00 06-00 00 00 10 ?  
00000200: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 40 00 @  
00000210: 00 60 2E 64-61 74 61 00-00 00 00 00-01 00 00 30 .data ?  
00000220: 00 00 00 0C-00 00 00 16-00 00 00 00-00 00 00 00 ?
```

有了这些api log，所以很容易处理批量已知恶意代码。对于未知的文件，则可以制定一些特征规则，然后去匹配运行后的api log与内存dump文件特征，则能较快筛选出恶意与非恶意文件，还有可以更进一步给出样本分类。

对于特殊的样本则可以先用此工具运行，找到感兴趣的api或位置，根据前面的api返回地址跟踪过去，则会很容易进行逆向分析过程。对于加壳或者混淆过的恶意代码分析能够提供参考分析与帮助。

工具还有不完善的地方和可以优化和怎加功能的地方，如果有时间我会再完善的，如果大家有好的想法，可以给我提建议。

工具下载地址

链接：<http://pan.baidu.com/s/1ntKmzKd>

密码：x6kn

1. 该工具只能运行在win7 x86, x64以上环境，且目标程序只支持x86(如果有需要我可以搞一个x64版本的)
2. 该工具在执行目标程序的时候是真实跑起来的，所以如果运行病毒请在虚拟机中运行，如果目标文件不是恶意代码则无所谓

压缩包md5:

f30fcb9fad3f509a8cb1bd9614c24720

压缩包sha1:

78cf69cc4119589e808a5247020540853cf18c55

*** 作者：比尔.盖茨（[微博](#)），属FreeBuf原创奖励计划文章，未经许可禁止转载**