Pin을 이용한 악성코드 분석 방법

Malware Analysis Method Using Pin

저자
(Authors)

김성호, 지성배, 박용수
Sungho Kim, Sungbae Ji, Yongsu Park

출처
(Source)

한국정보과학회 학술발표논문집 38(2C), 2011.11, 187-190 (4 pages)

발행처
(Publisher)

한국정보과학회
KOREA INFORMATION SCIENCE SOCIETY

URL

http://www.dbpia.co.kr/Article/NODE01746415

APA Style

김성호, 지성배, 박용수 (2011). Pin을 이용한 악성코드 분석 방법. 한국정보과학회 학술발표논문집, 38(2C), 187-190.

Pin을 이용한 악성코드 분석 방법

김성호[1O], 지성배[2], 박용수[1]
[1] 한양대학교, [2] 한국인터넷진흥원
mptstory@gmail.com, sbji@kisa.or.kr, yongsu@hanyang.ac.kr

Malware Analysis Method Using Pin

Sungho Kim[1O], Sungbae Ji [2], Yongsu Park[1]
[1] Hanyang University, [2] Korea Internet & Security Agency

요 약

최근 다양한 종류의 악성코드들이 급격히 증가하고 있는 추세이다. 이와 같은 악성 코드들은 정보유출, 서비스 거부 등 다양한 유형을 지니고 있다. 급격히 늘어나고 있는 악성코드 및 변종으로 인해 이를 분석하고 조치를 취하는 데에 한계가 있다. 따라서 최근 다양한 방법으로 자동화 된 악성코드 분석에 대한 연구가 활발히 이루어 지고 있다. 본 논문에서는 Intel에서 제공하는 Pin을 이용하여 악성코드를 분석하는 방법을 소개한다. Pin은 동적 이진 Instrumentation 도구로 분석을 위한 다양한 API와 툴을 제공을 한다. 이를 통해 다양한 유형의 악성코드 분석 방법을 소개한다.

1. 서 론

최근 10년 동안 7.7DDoS, 3.4 DDoS, 슬래머 웜 등 대규모 사고를 포함한 악성코드들이 증가하고 있는 실정이다. 공격 기법, 분석 방해 등의 기술들이 점점 더 발전이 되고 있고 피해범위가 개인 뿐만 아니라 국가에까지 확장이 되었다. 또한 다양한 악성코드 제작 툴들이 보급됨으로 변종 악성코드들이 증가를 하고 있는 실정이다. 이와 같은 악성코드들은 Anti-VM, 디버거, 실행 압축 등 다양한 분석 회피 기술을 지고 있어 분석하는 데에 많은 시간과 기술을 요구하고 있다. 하지만 이에 대응 하는 기술을 악성코드의 진화속도에 비해 뒤쳐져 있는 실정이다.

대부분의 악성코드가 분석 방해 기법을 적용하고 있기 때문에 이러한 분석 방해 기법을 우회 또는 무력화해야 분석이 가능하다. 따라서 악성코드 자체 파일을 정적 분석 하는 것 보다는 행위를 분석하기 위한 동적 분석 기법에 대하여 많은 연구가 진행되어 왔다. 대표적인 기술로는 Anubis[1], CWSandBox[2], Norman Sandbox[3]등이 있다. 이와 같은 분석 시스템은 악성코드를 실행시켜 악성코드의 프로세스, 파일, 네트워크, 레지스트리 등과 관련된 행위를 모니터링 하여 분석을 한다.[4] 하지만 이와 같은 방법 또한 VM 탐지, Time Check, Process Check등으로 인하여 분석을 방해 받는 경우가 존재한다. 따라서 정적 분석과 동적 분석을 같이 수행하여야지만 악성코드를 조금 더 세밀하게 분석을 할 수 있다.

따라서 본 논문에서는 정적 분석에 도움을 줄 수 있는 Pin[5]을 소개를 한다. Pin은 동적 이진 분석 도구로 타겟 프로그램의 런타임에 다양한 정보를 얻을 수 있다. 이와 같은 정보는 정적 분석에도 융합을 하여 악성코드 분석에 큰 도움을 줄 수 있다. 2장에서는 악성코드 분석에 사용되는 Pin을 소개하고, 3장에서는 악성코드를 분석하기 위한 툴을 소개한다. 4장에서는 핀을 통해 실제 악성코드를 분석한 결과를 보여주고 5장에서는 결론 및 발전사항에 대해 제시한다.

2. Pin의 소개

PIN은 intel에서 제공하는 동적 이진 분석 도구(Dynamic Binary Analyzer)로써 C혹은 C++로 작성된 임의의 코드를 실행코드의 한 부분으로 삽입을 시킬 수 있다. 소스코드의 재 컴파일이 필요가 없으며 동적으로 코드를 재 컴파일 하는 프로그램이다. PIN은 일종의 JIT(Just In Time) 컴파일러로 하나의 명령어가 실행할 때마다 그 흐름을 가로채어 PIN 내부적으로 새로 컴파일 한 코드를 실행하며 이때 사용자가 원하는 코드를 Instrumentation을 할 수 있다.

PIN의 다양한 API등을 이용하여 보안과 시스템 에뮬레이션 등을 동적으로 수행할 수 있다. PIN은 라이브러리 로드와 시스템 콜 예외처리 그리고 쓰레드 생성 등 여러 가지 확장된 API들을 제공하여 동적 분석을 한다.

2.1 Instrumentation

Instrumentation 이란 런타임 정보를 얻기 위해 타겟 프로그램에 추가적으로 코드를 삽입하여 분석을 하는 기술이다. [6]

```
    counter++;
sub $0xff, %edx
    counter++;
cmp %esi, %edx
    counter++;
jle <L1>
```

그림 1 Pin의 instruction 카운트 예제[6]

그림 1 와 같이 instruction의 실행 개수를 파악하기 위해 각 instruction 이전에 카운팅을 할 수 있는 코드를 삽입을 한다.

그림 2의 예제 코드는 타겟 프로그램의 실행되는 instruction의 개수를 카운트 하는 예제이다. Pin tool의 코드에는 그림2과 같이 핵심적인 두 가지 루틴이 포함이 되어 있다.

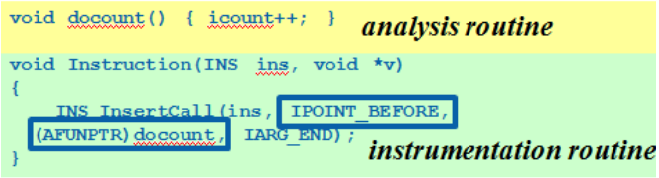

그림 2 Pin의 주요 루틴[6]

2.2 Instrumentation Routines

Instrumentation 코드를 어디에 삽입을 할지 정하는 루틴이다. <그림3> 와 같이 Instruction 바로 이전에 instrumentation을 하여 instruction이 실행되기 전에 카운팅 함수를 위치 시켜 instruction의 개수를 파악한다. [5]

2.2 Analysis Routines

Instrumentation 코드가 활성화 될 때 어떠한 행위를 할지 정의하는 루틴이다. 그림 3과 같이 Instruction이 실행이 될 때마다 docount루틴이 호출이 되어 카운팅 변수의 값을 증가시킨다. [6]

3. 악성코드 분석을 위한 도구

3.1 Instruction Trace

Instruction 트레이스는 타겟 프로그램의 실행되는 Instruction을 트레이스 한다. 이는 악성코드에서 수행이 되는 Instruction의 주소를 로그로 남겨 악성코드의 흐름 분석에 도움을 줄 수 있다. 이와 같은 정보를 이용하여 안티디버깅, 타임 체크 등의 특정 조건에 의해 분기 되는 코드 패턴을 분석하는 데에 도움을 준다. 본 도구는 각 instruction의 이전에 instrumentation을 하여 코드를 분석한다.

3.2 메모리 Trace

메모리 트레이스는 메모리에 읽기 또는 쓰기 하는 내역을 로그로 남긴다. 이는 악성코드에서 메모리에 읽기, 쓰기 하는 주소 값, 문자 등을 확인 할 수 있어 악성코드에서 사용되는 다양한 행위들을 예측할 수 있다. 본 도구는 인스트럭션이 메모리에 읽기 혹은 쓰기 하는 인스트럭션인지 판별하고 이를 이용하여 로그를 남긴다.

3.3 API 인자 값 Trace

함수의 인자 값 트레이스는 악성코드가 사용하는 API의 인자 값을 확인하여 분석을 할 수 있다. 함수의 인자 값을 분석을 통해 악성코드가 어떠한 행위를 하고 있는지를 확인 할 수 있다. 이와 같은 값을 통해 악성코드가 어떠한 행위를 하는 지 확인 할 수 있고 또한 어떠한 위치에서 API가 실행이 되고 어떠한 인자 값을 사용하는지 확인 할 수 있어 정적 분석에도 도움을 줄 수 있다. 본 도구는 지정된 System call API의 시작지점과 끝나는 지점에 Instrumentation을 하여 분석을 한다. 시작 점에는 API의 인자 값을 Log로 남기는 분석코드를 삽입하고, 종료 시에는 리턴되는 값을 Log로 남기는 분석 코드를 삽입한다.

4. Malware Analysis using Pin tool

4.1 Instruction Trace

그림4는 Pin Tool을 인스트럭션 트레이스 결과와 IDA pro에서의 instruction trace 스크립트의 결과이다. 결과를 보면 네모 부분 40d9dc에서부터 프로그램의 흐름이 변하는 것을 확인 할 수 있다.

40d9cf	40d9cf
40d9d0	40d9d0
40d9d3	40d9d3
40d9d8	40d9d8
40d9dc	40d9dc
40d8f8	40d9e2
40d8f9	40d4e1

그림 3 Pin을 이용한 결과(좌), IDA를 이용한 결과 (우)

위와 같은 위치에서 프로그램의 흐름이 변경이 된다는 것을 확인 할 수 있고 이와 같이 흐름이 변경되는 위치를 따라가보면 다음과 같은 코드가 있음을 확인 할 수 있다. 오른쪽 로그를 따라가 보면 쓰레드 종료 API가 있는 코드로 이동을 하고 좌측 PIN으로 분석한 로그를 따라가보면 실제 악성행위를 하는 코드로 이동하는 것을 확인 할 수 있다. 이를 이용하여 분기가 발생하는 40d9dc의 위치를 따라가보면 그림 3과 같이 분기가 발생하는 위치를

찾을 수 있다.

```
:0040B03F __GetPEB_ProcessHeap_Flags proc near     ;
:0040B03F                nop
:0040B040                stc                       ;
:0040B041                mov     eax, [eax+18h]    ;
:0040B044                cmc                       ;
:0040B045                cmp     ah, ah            ;
:0040B047                nop
:0040B048                push    dword ptr [esp+8]
:0040B04B                mov     eax, [eax+0Ch]    ;
:0040B04E                jmp       DebuqqedCheck
```

그림 4 안티디버깅 루틴[7]

그림 4와 같이 분기가 발생하는 위치에서 역추적하면 PEB(Process Environment Block)의 ProcessHeap.Flag값을 이용하여 디버거를 체크 하는 것을 확인 할 수 있다.

4.2 Memory Trace

메모리 트레이스는 악성코드가 메모리에 쓰기 또는 읽기 하는 사항을 저장한다. 이를 통해 다양한 악성행위를 예측을 할 수 있다. 다음은 본 도구를 이용한 결과 이다. (Win-Trojan/Downloader.38912.DG (AhnLab V3))

```
8962  004013B7: W 01D349A5 1 q
8963  004013B7: W 01D349A6 1 e
8964  004013B7: W 01D349A7 1 m
8965  004013B7: W 01D349A8 1 u
8966  004013B7: W 01D349AA 1 !
8967  004013B7: W 01D349AC 1 v
8968  004013B7: W 01D349AD 1 i
8969  004013B7: W 01D349AE 1 r
8970  004013B7: W 01D349AF 1 t
8971  004013B7: W 01D349B0 1 u
8972  004013B7: W 01D349B1 1 a
8973  004013B7: W 01D349B2 1 l
8974  004013B7: W 01D349B5 1 v
8975  004013B7: W 01D349B6 1 m
8976  004013B7: W 01D349B7 1 w
8977  004013B7: W 01D349B8 1 a
8978  004013B7: W 01D349B9 1 r
8979  004013B7: W 01D349BA 1 e
```

그림 5 가상머신-탐지 관련 문자열 탐지

위의 결과는 다음과 같이 구성이 되어 있다.

표 1 메모리 트레이스 결과 값 구성

EIP:	Access (R or W)	&Addr (R or W)	*Addr

Instumentation이 된 위치, 메모리 Read, Write여부, 참조한 메모리 주소, 참조 메모리 주소의 값으로 구성이 되어 있다.

이와 같은 결과를 이용하여 다음과 같이 악성코드의 자가 수정관련 코드와 수정이 되는 위치 또한 찾을 수 있다. (Win-Trojan/Atraps.25211584(Ahnlab V3))

다음은 위의 도구를 수정하여 얻은 결과 값이다. (기존의 결과에서 동일한 주소에 읽기, 쓰기를 수행 부분만을 추출한다.)

```
16  R 0x40d549->0x401000:0xe85f4a47
17  W 0x40d555->00401000:0x60286856
18  R 0x40d87c->0x401004:0x9d882251
19  W 0x40d3d9->00401004:0x15ff0040
20  R 0x40d87c->0x401008:0x88377241
21  W 0x40d3d9->00401008:0x405050
22  R 0x40d87c->0x40100c:0x7ef2d29a
23  W 0x40d3d9->0040100C:0xf685f08b
```

그림 6 수정된 도구를 이용한 결과 값

위 결과 값은 쓰기 또는 읽기, EIP, 접근 메모리 주소, 값으로 구성이 되어 있다. 이와 같은 결과를 통해 다음과 같은 자가 수정 코드를 탐지할 수 있다. (단, 동일한 주소에서 값을 읽고, 쓰기를 한 경우에만 분석 가능하다.)

```
00401000  47              INC EDI
00401001  4A              DEC EDX
00401002  5F              POP EDI
00401003  E8 5122889D     CALL 9DC83259
00401008  41              INC ECX
```

그림 7 자가 수정이 되는 코드 (실행 전)

```
00401000  56              PUSH ESI
00401001  68 28604000     PUSH 225dox.00406028
00401006  FF15 50504000   CALL DWORD PTR DS:[405050]
0040100C  8BF0            MOV ESI,EAX
0040100E  85F6            TEST ESI,ESI
00401010  74 19           JE SHORT 225dox.0040102B
```

그림 8 자가 수정이 되는 코드 (실행 후)

또한 본 도구를 이용하여 악성코드 내의 특정 디코딩 모듈을 분석할 수 있다. 특정 디코딩 모듈은 메모리에서 문자를 읽고 디코딩 후 다시 메모리에 쓰는 패턴을 지니고 있다. 다음은 본 도구를 이용한 분석 결과이다. (Trojan/Onlinegamehack.85372-Ahnlab V3)

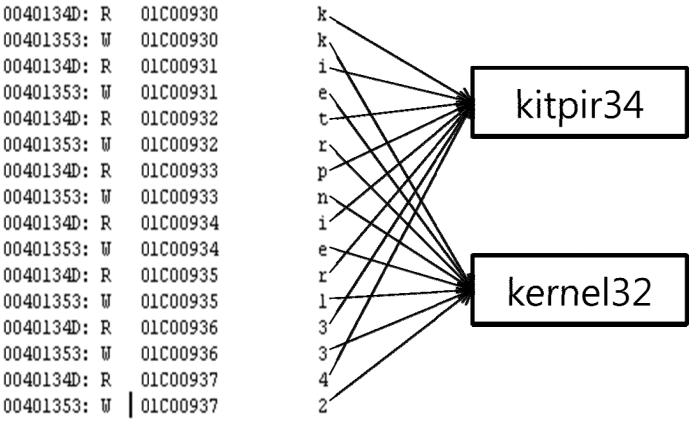

그림 9 특정 디코딩 패턴 분석

이와 같은 결과를 통해 다음과 같이 디코딩 모듈을 찾을 수 있다.

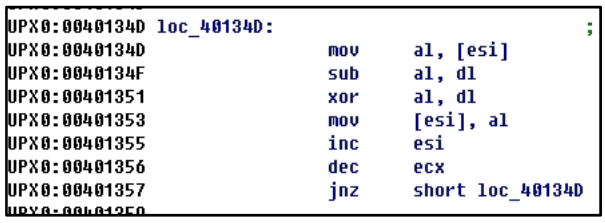

그림 10 악성코드의 디코딩 모듈

4.3 System call API 인자 값 Trace

본 도구는 동적 이진 instrumentation 도구이므로 실행압축에 대한 영향을 받지 않는다. 다음은 UPX로 실행 압축이 되어 있는 악성코드(Win-Adware/WideLink.215040)의 로그 값의 일부이다. 다음과 같이 악성코드가 사용하는 URL 정보를 얻을 수 있다.

```
socket 0x4572ce 0x2 0x1 0
  returns 0
gethostbyname 0x456e5b www.widelink.kr
inet_addr 0x76ed3569 www.widelink.kr
  returns 0xffffffff
  returns 0x2ba5ec0
```

그림 11 악성코드 분석 결과 값

또한 다음과 같이 프로세스를 탐지하는 것을 확인 할 수 있다. 본 악성코드(Trojan/Onlinegamehack.85372-Ahnlab V3)에서는 다음과 같이 현재 프로세스 리스트를 구하고 이를 통해 백신 관련 문자열을 찾는 것을 확인 할 수 있다.

```
CreateToolhelp32Snapshot 0x4023cd 0x2 0
  returns 0x774
LoadLibraryA 0x402452 kernel32.dll
  returns 0x7c800000
Process32First 0x4024fd 0x774 0x12f6e0
  returns 0x1
lstrcmpi 0x4044bb ALYac.aye [System Process]
  returns 0x1
```

그림 12 악성코드 분석 결과 값

위의 결과는 다음과 같이 구성이 되어 있다.

표 2 API 인자 값 트레이스 도구 결과 구성

API 이름	리턴 주소	첫 번째 인자	두 번째 인자	…

실행이 된 API명 API의 인자 값, 그리고 API가 수행이 된 후 리턴하는 address로 구성이 되어 있다.

5. 결 론

동적 분석도구인 Pin을 사용하면 이와 같이 다양한 방법으로 악성코드를 분석할 수 있다. 이외에도 PIN에서 제공하는 다양한 API를 이용하여 악성코드의 분석에 큰 도움이 될 것이다. 이처럼 PIN을 이용하여 악성코드를 자동으로 분석하는 시스템을 만들 수 있으며 다양한 디버거 또는 정적 분석 도구들과 융합을 하여 더욱 세밀한 분석을 가능하게 한다. 현재 악성코드 분석을 수행한 도구는 너무 많은 양의 로그를 남기기 때문에 자세한 분석을 위해서는 시간이 걸린다. 따라서 필요한 정보만을 추출하고 또한 이를 정적으로 분석할 수 있는 연구가 계속되어야 할 것이라 예상한다.

Acknowledgment

본 연구는 2011 년 한국인터넷진흥원 위탁과제 “악성코드 내부코드 자동 역공학 분석 방법 연구”의 지원을 받아 작성되었음.

참고문헌

[1] Anubis, [anubis.iseclab.org]
[2] CWSandbox, [www.sunbeltsandbox.com]
[3] Norman Sandbox, [www.norman.com/technology/norman_sandbox/]
[4] 임채태 외 2, 최신 악성코드 기술동향 및 분석 방안 연구, 정보과학회지 2011
[5] Pin, [www.pintool.org]
[6] Robert Cohn, Pin Tutorial, Academia Sinica 2009
[7] Mark Vincent Yason, The Art of Unpacking, IBM 2007