동적 바이너리 분석 툴 비교 분석: Binary Instrumentation

최영현*, 장성수*, 임헌정*, 엄정호*, 정태명**
*성균관대학교 전자전기컴퓨터공학
**성균관대학교 정보통신공학
e-mail:{yhchoi, ssjang, hjlim99, jheom, tmchung}@imtl.skku.ac.kr

A Comparison of tools for Dynamic Analysis: Binary Instrumentation

Young-Hyun Choi*, Seongsoo Jang*, Hun-Jung Lim*, Jung-Ho Eom*, Tai-Myoung Chung**
*Dept of Electrical and Computer Engineering, Sungkyunkwan University
**Dept of Information and Communication Engineering, Sungkyunkwan University

요 약

본 논문에서는 동적 인스트러멘테이션을 적용한 동적 바이너리 분석 툴들에 대해 비교 분석을 수행하였다. 비교 분석은 각 툴들에서 공통의 항목에 맞는 특성 값들을 도출하여 비교함으로써 동일한 상황에서 툴들의 특장점을 확인할 수 있고, 각 특징에 따른 기술적인 배경을 뒷받침하여 더 나은 동적 분석 툴을 만들 수 있는 발판을 마련하였다. 이를 위해 DynamoRIO, DynInst, Pin, Valgrind의 4가지 동적 분석 툴을 지원 가능 플랫폼, 실행 메카니즘의 컨셉, 인스트러멘테이션 가능 범위, 성능, 라이선스와 관련된 입수 가능성의 5가지 주요 항목으로 비교 분석을 수행하였다.

1. 서론

IT 기술이 발전하고 다른 사업들과 융합이 되어 성장함에 따라 IT는 실생활에 필수불가결한 요소로 자리잡고 있다. 컴퓨터뿐만 아니라 생활 곳곳에서 사용하는 기기들에 프로그램이 사용되고 있다. 이러한 프로그램에서 사고가 발생하게 되면 큰 피해를 불러일으키게 된다. 프로그램에 존재하는 보안 취약점은 개발자의 실수나 코드 상에 존재하는 취약성에 의하여 발생한다. 이러한 보안 취약점이 보안 사고를 일으킬 수 있는 환경을 마련해주는 것이다. 따라서 각 프로그램들에 대한 보안 취약점을 분석하여 이러한 보안 사고를 방지하는 것이 중요하다.

사용하는 프로그램들은 대부분 소스코드를 확인할 수 없는 상태로 배포되어 사용된다. 이는 개발자 혹은 개발사 측의 지적재산권 및 자산의 보호를 위해 바이너리 상태로 배포[1]하게 된다. 때문에 소스코드가 없는 실행 프로그램의 보안 결함을 발견하기 위한 동적 분석 툴을 사용하게 된다.

여러 동적 분석 툴 중 본 논문에서 비교할 대상은 뛰어난 검출 능력을 가지고 있는 동적 인스트러멘테이션 기법을 적용한 것들이다. 동적 인스트러멘테이션을 사용한 동적 분석 툴들을 비교 분석하여 각각의 특장점과 기술적 배경들을 도출하는 것을 본 논문의 목표로 한다.

본 논문은 중소기업청에서 지원하는 2010년도 산학연공동기술개발사업(No. 00044301)의 연구수행으로 인한 결과물임을 밝힙니다.

2장에서느 바이너리 인스트러멘테이션 기법에 대한 내용을 소개하고, 3장에서는 바이너리 인스트러멘테이션이 적용된 동적 바이너리 분석 툴들 중 4개를 선정하여 5가지의 항목에 맞춰 비교 분석을 수행한다. 마지막으로 4장에서 결론을 맺는다.

2. 바이너리 인스트러멘테이션

동적 분석 툴들에서 사용자가 원하는 코드를 원래의 코드에 삽입하여 수행할 수 있는 기능을 제공하기위해 여러 가지 기법을 사용한다.

바이너리 파일은 클라이언트 프로세스와 함께 프로그램에 의해 추가하거나 외부 프로세스에 의해 추가할 수 있다. 만약 프로그램이 동적으로 링크된 코드를 사용할 때는 분석 코드는 동적 링크 뒤에 더해지게 된다.

동적 바이너리 인스트러멘테이션 (instrumentation)은 두 가지의 큰 장점을 가지고 있다. 첫 번째는 클라이언트 프로그램 필요없이 사용이 가능하고, 이는 사용자들의 편의를 제공하게 된다. 두 번째로는 모든 클라이언트 코드를 처리 가능하다. 기존의 기술들로는 코드 및 데이터가 혼합되어 있는 경우에는 정적으로 분석하기 힘들 수 있어서 다른 모듈이 사용되거나 만약 클라이언트가 동적으로 생성된 코드를 사용하는 경우는 분석이 불가능한 경우가 종종 발생하게 된다. 이 특징은 모든 코드를 인스트러멘테이션 하기 위한 라이브러리의 정확하고 완전한 처리를 위해 매우 중요하다. 동적 바이너리 인스트러멘테이션의 이러한

이점들은 많은 동적 분석 툴들을 최고의 성능으로 이끌기 위해 다양하게 사용된다.

일반적으로 바이너리 인스트러멘테이션을 통해 삽입되는 코드들은 메모리의 성능이나 코드의 흐름, 보안 테스트 등을 행하여 바이너리 파일의 동작을 분석하는 데 도움을 준다.

3. 동적 분석 툴

본 논문에서 살펴볼 동적 분석 툴은 가상 머신과 같은 시뮬레이션 환경을 제공하여 실행타임에 동적 바이너리 인스트러멘테이션을 수행하여 동적 바이너리 분석을 행한다.

3장에서는 동적 분석 툴들을 각 특징에 따라 비교해본다. 세부 사항들은 논문, 웹사이트, 소스코드, 매뉴얼 등을 참고하여 작성하였다.

기본적으로 비교는 다음과 같은 형식을 따른다.

- (Plat.): 툴이 실행 가능한 플랫폼
- (Exec.): 실행 메카니즘의 컨셉
- (Inst.): 인스트러멘테이션 가능 범위
- (Perf.): 인스트러멘테이션이 없는 환경을 위한 성능. 보통의 환경을 1.0이라고 가정.
- (Avail.): 라이선스와 관련된 입수가능성

3.1. DynamoRIO

DynamoRIO[1-4]는 Dynamo에서 시작되었으며 동적 바이너리 최적화 및 인스트러멘테이션 툴이다. 2001년 처음으로 릴리즈 되었고, 인스트러멘테이션은 2002년부터 사용하게 되었다.

- Plat. x86/Win32와 x86/Linux
- Exec. DynamoRIO는 동적 바이너리 compilation과 caching을 사용하였다. 이를 통하여 원래의 x86 코드는 거의 대부분 직접 해결 가능하게 되었다.
- Inst. 분석 코드는 in-line을 추가할 수 있고, C 함수를 부를 수 있는 기능을 제공한다.
- Perf. 프로그램 분석 코드에 작은 틈이 생기더라도 보통의 속도보다는 프로그램이 빨리 동작한다.
- Avail. 타인에게 양도하지 않는다는 조건으로 바이너리 파일의 라이선스는 무료이다.

3.2. DynInst

DynInst[5,6]는 다른 동적 바이너리 인스트러멘테이션을 사용하는 툴들과는 다른 점이 있다. 장기간 실행되는 과학적 프로그램의 프로파일링을 특정 대상으로 하고 있다. 이 프로그램은 런타임 도중에 코드를 삽입하거나 제거하는 때의 성능에 강점을 가지고 있다.

- Plat. x86/Win32와 x86/Linux, IA64/Linux, SPARC/Solaris, PowerPC/AIX, Alpha/Tru64, MIPS/IRIX.
- Exec. In-line과 Trampolines를 사용하였다. 이를 통하여 선택한 인스트럭션이 replacement jump보다 작을 경우 가능하다면 근처의 인스트럭션이 공간을 제공한다.
- Inst. 분석 코드를 변수와 표현들로부터 구문 그래프를 뽑아 내는 것은 구조에 맞춰 자르는 독립적인 방법을 사용한다.
- Perf. tampling 기술을 사용자의 의도대로 탈착할 수 있어서 목적에 따라 툴의 속도를 향상시킬 수 있다.
- Avail. 타인에게 양도하지 않는다는 조건을 가지고 연구 목적으로 사용할 시에는 무료로 사용할 수 있으나 상업적 목적으로 사용할 때는 사용료를 지불해야한다.

3.3. Pin

Pin[7-8]은 active analysis code를 제공하게 설계되었다. 임베디드 시스템에서도 동작할 수 있도록 지원한 것이 특징이라고 할 수 있다.

- Plat. IA64/Linux, x86/Linux, x86-64bit/Linux, ARM/Linux
- Exec. 동적 바이너리 compilation과 caching을 사용한다. Pin은 성능 향상을 위해 Ispike 포스트-링크 최적화 도구(post-link optimizer)를 사용하였다.
- Inst. Pin은 분석 루틴에서 C나 어셈블리 코드로 작성된 calls를 추가할 수 있다. 단순한 분석 루틴은 Pin에 의해 inline될 수도 있다. 함수의 시작/끝 포인트를 인스트러멘트 할 수 있다. 가상 레지스터를 이용하여 분석 루틴 내 정보들의 패스 정보를 알 수 있다.
- Perf. IA64 환경에서는 보통을 1이라고 기준하였을 때, 1.4 정도의 속도를 가지고, x86 환경에서는 2.0 정도의 속도를 나타낸다.
- Avail. Pin은 기본적으로 무료로 이용이 가능하다. 하지만 인스트러멘테이션 엔진은 바이너리로만 배포하며, 라이브러리와 예제 툴은 BSD-style의 라이선스를 따른다.

3.4. Valgrind

Valgrind[9-11]는 active analysis code를 제공하게 설계되었다. Valgrind의 첫 릴리즈는 2002년에 시작되었다.

- Plat. x86/Linux
- Exec. 동적 바이너리 compilation과 caching을 사용한다.
- Inst. 기본적인 x86 코드는 UCode라고 불리는 RISC 형식의 IR로 변형시킬 수 있다. Valgrind에서의 인스트러멘테이션은 UCode-레벨에서 이루어지게 된다. inline 추가를 할 경우에도 UCode로 추가하게 된다. 선택적으로 함수를 중지시킬 수 있다.
- Perf. 보통의 경우에 2.1의 속도를 가지고, 코드 확장의 케이스에는 4.5 정도의 속도를 나타낸다.
- Avail. GNU GPL 라이선스를 따르며 소스 이용은 무료이다.

<표 1> 동적 바이너리 분석 툴 비교

Tools	Plat.	Exec.	Inst.	Perf.	Avail.
DynamoRIO	x86/Win32 x86/Linux	Caching Optimization	C function In-line Active-code	High	Binary-Free
DynInst	x86/Linux, x86/win32, SPARC, MIPS, etc.	In-line Trampolines	C function Function-entry Active-code	Medium	Source-Free
Pin	IA64/Linux	Caching	C function In-line Active-code Function-entry	Medium	Source-Free Binary-Free
Valgrind	x86/Linux	Caching	C function In-line Active-code Function-entry	High	Source-Free

4. 결론 및 향후 연구

본 논문에서는 뛰어난 성능을 보이는 동적 바이너리 인스트러멘테이션 기법을 사용하여 코드를 삽입하는 동적 분석 툴들을 비교/분석하는 것을 목적으로 하였다. 이를 통하여 개발을 하거나 검증을 위하여 실제 사용할 동적 분석 툴을 선정하기 위해 자료를 수집하거나 현재 사용하고 있는 동적 분석 툴보다 나은 자신에게 맞는 동적 분석 툴을 찾는데 도움을 줄 수 있다. 또한 동적 분석 툴들이 가지고 있는 여러 가지 특징들을 나타내어 각 툴들이 가지고 있는 장점이 나오게 된 기술적 배경을 유추할 수 있고 더 나은 동적 분석 툴을 만들기 위해 고려할 수 있는 특징들을 채택할 수 있는 발판을 마련하고 있다.

향후 더 많은 동적 분석 툴들과 비교 특징들을 도출하여 분석하고 이를 통해 향상된 성능을 보이는 효율적인 동적 분석 툴을 개발하는 기틀을 마련한다.

참고문헌

[1] Derek. B., DynamoRIO.: http://dynamorio.org/, http://www.cag.lcs.mitedu/dynamorio/

[2] Derek. B., Timothy. G. and Saman. A. "An infrastructure for adaptive dynamic optimization.", *In proceeding of the International Symposiu,* on Code Generation and Optimization (CGO '03), pp. 265-276. 2003.

[3] Derek. B., et al.. "An infrastructure for adaptive dynamic optimization", *Code Generation and Optimization, 2003. CGO 2003. International Symposium* , pp. 265-275, 2003.

[4] Derek. B. and Saman A., "Maintaining Consistency and Bounding Capacity of Software Code Caches", *Code Generation and Optimization,* pp. 74-85, 2005.

[5] Bryan. B. and Jeffrey K.H.: An API for runtime code patching.: Journal of High Performance Computing Applications.:, No. 14, vol. 4 , pp. 317-329. (2000)

[6] B. Buck and J. K.H. "An API for runtime code patching,", *The International Journal of High Performance Computing Applications*, vol. 14, pp. 317--329, 2000.

[7] Robert. C.: Instrumentation of Intel Itanium Linux program with Pin.: Tutorial at International Symposiu, on Code Generation and Optimization (CGO '04). (2004)

[8] Chi-Keung Luk, et al., "Pin: building customized program analysis tools with dynamic instrumentation", *Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation,* pp. 190-200, 2005

[9] Valgrind.: http://valgrind.org

[10] Nicholas N. and Julian S., "Valgrind: A Program Supervision Framework", *Electronic Notes in Theoretical Computer Science,* Vol. 89, No. 2, pp. 44-66, 2003.

[11] Nicholas N. and Julian S., "Valgrind: a framework for heavyweight dynamic binary instrumentation", *Proceedings of the 2007 PLDI conference,* Vol. 42, No. 6, pp. 89-100, 2007.