http://dx.doi.org/10.3745/KTCCS.2013.2.9.399

A Study on MS Crash Analyzer

Noh Myoung Sun† · Na Jong Bae†† · Jung Gwang Un††† · Ryou Jae Cheol†††† · Noh Bong Nam†††††

ABSTRACT

MSEC(Microsoft Security Engineering Center) performed fuzz testing Windows Vista with 350 million test cases for 14 months before launching it. They analyzed crashes resulted from the testing and developed crash analyzer *!exploitable* based on the data used to determine exploitability. In this paper, we describe how MS crash analyzer determines exploitability of crashes. Besides, we suggest an improvement to overcome the limitations found in the MS crash analyzer during the analysis.

Keywords : MS Crash Analyzer, Crash Analysis, Vulnerability, Taint Analysis

MS 크래시 분석도구에 관한 연구

노 명 선† · 나 종 배†† · 정 광 운††† · 류 재 철†††† · 노 봉 남†††††

요 약

마이크로소프트사의 보안공학센터인 MSEC(Microsoft Security Engineering Center)에서 윈도우즈 비스타 운영체제 출시 전 14개월 동안 약 3억 5천만 개의 퍼징 테스트를 수행하였고, 이를 통해 수집된 크래시들을 분석하여 발생 유형별로 분류하고 이들의 위험도를 판별한 데이터를 기반으로 *!exploitable* 이라는 크래시 분석도구를 개발하였다. 본 논문에서는 MS 크래시 분석도구의 크래시 위험도 판단방법에 대해 분석한 결과를 기술하며, 분석을 통해 발견된 도구의 문제점을 알아보고 개선방안을 제시한다.

키워드 : MS 크래시 분석도구, 크래시 분석, 취약점, 테인트 분석

1. 서 론

크래시(crash)란 프로그램 내 처리 되지 않은 예외(exception)에 의해 결함이 발생하여 프로세스가 비정상적으로 종료되는 것을 의미하며, 프로그램 작성 시 예외상황에 대한 적절한 예외 처리기(exception handler)를 등록하지 않았을 경우 나타나게 된다[1]. 크래시는 단순 결함과 공격자에 의해 프로그램의 실행 흐름이 조작되어 공격코드가 실행될 수 있는 위험한 결함으로 분류될 수 있다. 공격코드 실행가능 여부를 확인하기 위해서는 크래시 정보를 바탕으로 분석을 수행해야 한다. 그러나 발생된 모든 크래시를 대상으로 원인을 분석하고 문제를 해결하는 것은 경우에 따라 상당한 시간이 소모된다. 효율적인 결함 제거를 위해서는 위험도가 높은 결함을 파악하여 처리에 대한 우선순위를 정할 수 있어야 한다. 이를 위해 2009년 MSEC(Microsoft Security Engineering Center)에서 크래시의 위험도를 판단할 수 있는 분석도구를 개발하였다[2].

!exploitable로 불리는 MS의 크래시 분석도구는 윈도우즈 비스타 출시 전 14개월 동안 비스타 운영체제를 대상으로 약 3억 5천만 개의 퍼징 테스트를 수행하였고, 이를 통해 수집된 크래시들을 분석하여 발생 유형별로 분류하고 이들의 위험도를 판별한 데이터를 기반으로 분석정책을 만들었다[3].

크래시 분석도구는 윈도우즈용 디버거 WinDbg의 확장모듈 형태로 제공되며, 발생된 크래시의 위험도에 따라 Exploitable, Probably Exploitable, Probably Not Exploitable, Unknown으로 구분하고 분석한 결과를 제공한다. 또한 위험도 판단에 사용된 프로세스 정보, 세부예외정보 등과 함께 기본적인 설명을 나타내어 간단한 수행만으로 크래시의 위험도와 기타 정보를 좀 더 쉽게 확인 할 수 있도록 한다[2].

본 논문에서는 오픈소스 형태로 제공되는 MS 크래시 분석도구의 크래시 위험도 판단방법을 분석한 결과를 기술하며 분석을 통해 발견된 도구의 문제점과 개선방안을 다룬다.

※ 본 연구는 미래부가 지원한 2013년 정보통신·방송(ICT) 연구개발사업의 연구결과로 수행되었음.
† 정 회 원 : 전남대학교 정보보안협동과정 박사과정
†† 비 회 원 : 전남대학교 정보보안협동과정 박사과정
††† 준 회 원 : 충남대학교 컴퓨터공학과 석사과정
†††† 종신회원 : 충남대학교 컴퓨터공학과 교수
††††† 종신회원 : 전남대학교 전자컴퓨터공학부 교수
논문접수 : 2013년 4월 18일
수 정 일 : 1차 2013년 5월 27일
심사완료 : 2013년 5월 27일
* Corresponding Author : Noh Bong Nam(bongnam@jnu.ac.kr)

2. MS 크래시 분석도구의 크래시 분석방법

MS 크래시 분석도구는 크래시 발생 시 운영체제로부터 제공되는 예외 정보를 바탕으로 Table 1의 데이터를 판별하고 이를 통해 크래시 분류 및 위험도를 판단한다.

Table 1. Data used for analyzing crashes

Data	Description
PROCESSOR_MODE	프로세서 접근모드
EXCEPTION_ADDRESS_RANGE	예외발생주소 범위
EXCEPTION_TYPE	예외 타입
EXCEPTION_SUB_TYPE	예외 세부 타입
EXCEPTION_LEVEL	예외 탐지 단계
ANALYZE_FUNCTION	추가 분석함수

프로세서모드(PROCESSOR_MODE)는 크래시 발생 시 프로세서의 접근모드를 나타내는 것으로 커널모드와 유저모드로 구분된다. 예외주소범위(EXCEPTION_ADDRESS_RANGE)는 크래시 발생원인 주소(faulting address)에 기반한다. 크래시 발생원인 주소는 일반적으로 예외가 발생된 명령어의 주소, 즉 명령어 포인터 레지스터(EIP) 값을 나타내지만, 예외적으로 접근위반(access violation)으로 인해 발생한 크래시의 경우 예외를 유발하는 참조 메모리의 주소를 나타낸다. 예외주소범위 값은 프로세서 모드와 크래시 발생원인 주소에 따라 Table 2와 같이 구분된다. 예를 들어, 커널모드에서 크래시 발생 시 크래시 발생원인 주소가 0x80000000 보다 작을 경우 커널영역 메모리로 판단하고, 유저모드에서 접근위반으로 인한 크래시 발생 시 참조하는 메모리 주소가 0-65,535 내에 속할 경우 널(NULL) 주소 인접 메모리로 간주한다.

Table 2. Value of memory address range

Mode	Memory range	Value
kernel (K)	kernel memory	>= 0x80000000
	user memory	< 0x80000000
user (U)	not near NULL	>= 65,536
	near NULL	< 65,536

예외타입(EXCEPTION_TYPE)은 크래시가 어떠한 예외로 인하여 발생하였는지를 나타내는 데이터로 윈도우즈 운영체제에서 발생 가능한 예외는 총 20개이다[4]. 하지만 MSEC에서 윈도우즈 비스타를 대상으로 퍼징 테스트를 통해 수집된 크래시에서는 일부 예외만이 발견되었으며, 그 결과 분석정책에서 등록되어 처리하는 예외는 Table 3과 같이 10개로 정의되어 있다.

예외세부타입(EXCEPTION_SUB_TYPE)은 접근위반으로 인해 예외가 발생한 경우 어떠한 유형의 접근위반인지를 나타내기 위해 사용된다. Table 4와 같이 실행, 쓰기, 읽기 접근위반으로 분류된다.

Table 3. Exception list

Exception type	Description
GUARD_PAGE_VIOLATION	보호페이지에 대한 접근 위반
ACCESS_VIOLATION	허용되지 않은 메모리에 대한 접근 위반
BREAKPOINT	소프트웨어 브레이크 포인트 수행
IN_PAGE_ERROR	페이지 할당 및 접근 오류
ILLEGAL_INSTRUCTION	잘못된 형식의 명령어 수행
INTEGER_OVERFLOW	정수형 데이터의 허용 데이터 범위 초과
INTEGER_DIVIDE_BY_ZERO	정수형 데이터를 0으로 나누기 수행
FLOAT_DIVIDE_BY_ZERO	실수형 데이터를 0으로 나누기 수행
PRIVILEGED_INSTRUCTION	현재 모드에서 권한 없는 명령어 수행
STACK_BUFFER_OVERRUN	버퍼 보안 체크 위반

Table 4. Exception subtype list

Exception sub type	Description
ACCESS_VIOLATION_DEP	허용되지 않은 메모리에 대한 실행 접근위반
ACCESS_VIOLATION_WRITE	허용되지 않은 메모리에 대한 쓰기 접근위반
ACCESS_VIOLATION_READ	허용되지 않은 메모리에 대한 읽기 접근위반

예외단계(EXCEPTION_LEVEL)는 1차 예외(first chance)와 2차 예외(second chance)로 구분된다. 예외가 최초로 발생한 시점을 1차 예외라 하며, 해당 예외에 대한 적절한 예외 처리기가 존재하지 않아 예외를 처리하지 못하고 다시 예외가 발생한 경우를 2차 예외라 한다. 2차 예외가 발생할 경우 운영체제에 존재하는 기본 예외 처리기에 의해 예외가 처리되며, Fig. 1과 같은 오류 메시지를 나타내고 해당 프로세스를 종료한다[1].

마지막으로 분석함수(ANALYZE_FUNCTION)는 예외 발생 상황의 데이터만으로 크래시의 위험도 판단이 어려울 경우 추가적인 분석을 수행하기 위해 이용된다. 분석함수는 레지스터 및 명령어 분석함수와 테인트 분석(taint analysis) 함수로 분류된다.

레지스터 및 명령어 분석함수는 크래시를 유발하는 명령어와 주요 레지스터에 저장된 데이터 등을 판단하는 것으로 함수의 기능은 Table 5와 같다.

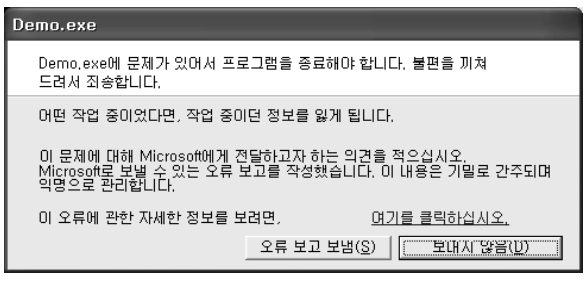


Fig. 1. Error message by default exception handler of operating system

Table 5. Functions for analyzing registers and instructions

Function	Description
IsFaultingAddressInstructionPointer	크래시 발생원인 주소가 예외 발생 시점의 수행 명령어의 주소인지 확인
IsFaultingInstructionOnStack	예외 발생 시점의 수행 명령어가 스택 영역에 존재하는지 확인
IsFaultingInstructionControlFlow	예외 발생 시점의 수행 명령어가 분기(branch) 명령어 또는 복귀(return) 명령어인지 확인
IsFaultingInstructionBlockDataMove	예외 발생 시점의 수행 명령어가 블록 데이터 이동(move) 명령어인지 확인
DoesStackTraceContainUnknownFunctions	스택추적에 알려지지 않은 심볼 정보를 가진 명령어 주소가 포함되어 있는지 확인

테인트 분석함수에 사용되는 테인트 분석기법은 입력 데이터가 프로그램에서 어떻게 변형되고 프로그램에 어떠한 영향을 주는지를 분석하는 것으로[5], MS 크래시 분석도구는 크래시 상황의 정보만으로 분석이 어려운 경우 이러한 테인트 분석기법을 이용하여 추가적인 분석을 수행한다.

MS 크래시 분석도구에서는 발생된 예외가 읽기 접근위반(ACCESS_VIOLATION_READ)인 경우에 크래시 발생 명령어의 목적 레지스터의 값을 입력 데이터로 정의하고, 해당 레지스터의 데이터가 특정 명령어 수행에 필요한 데이터에 영향을 주는지를 분석하여 위험도를 판단한다. MS 크래시 분석도구가 제공하는 테인트 분석함수는 Table 6과 같다.

MS 크래시 분석도구는 위에 설명한 6가지 데이터를 기반으로 분석정책에 따라 크래시의 위험도를 판별한다. 크래시 분석정책의 세부 내용은 다음 장에서 설명한다.

Table 6. Taint analysis functions

Function	Description
IsTaintedDataUsedToDetermineBranchSelection	기본블록 마지막 명령어가 분기 명령어일 경우 예외가 발생한 읽기 명령어의 대상 레지스터의 값이 분기의 조건에 영향을 주는지 확인
IsTaintedDataUsedAsAFunctionRetVal	기본블록 마지막 명령어가 복귀 명령어일 경우 예외가 발생한 읽기 명령어의 대상 레지스터의 값이 반환 값에 영향을 주는지 확인
IsTaintedDataUsedAsAFunctionArgument	기본블록 마지막 명령어가 함수 호출 명령어일 경우 예외가 발생한 읽기 명령어의 대상 레지스터의 값이 함수 호출의 인자에 영향을 주는지 확인
IsTaintedDataUsedAsSourceForBlockDataMove	예외가 발생한 읽기 명령어의 대상 레지스터의 값이 이후 블록 데이터 이동 명령어의 원본데이터 주소에 영향을 주는지 확인
IsTaintedDataUsedInALaterWrite	예외가 발생한 읽기 명령어의 대상 레지스터의 값이 이후 메모리 쓰기 명령어의 대상메모리 주소에 영향을 주는지 확인
IsTaintedDataUsedToDetermineBranchTarget	기본 블록 마지막 명령어가 분기 명령어일 경우 예외가 발생한 읽기 명령어의 대상 레지스터의 값이 분기의 목적지 주소에 영향을 주는지 확인

3. MS 크래시 분석도구의 크래시 분석정책

MS 크래시 분석도구는 총 39개의 크래시 분석정책을 가지고 있으며, 위험도에 따라 15개의 Exploitable, 9개의 Probably Exploitable, 5개의 Probably Not Exploitable 그리고 10개의 Unknown 정책으로 분류된다. 본 장에서는 크래시 분석도구에 사용되는 크래시 분석정책에 대해 알아본다. 지면 관계상 각 위험도를 E, PE, PNE, U로 나타낸다.

3.1 명령어 포인터 변조

크래시 상황에서 다음으로 실행할 명령어의 주소를 저장하고 있는 명령어 포인터의 주소 값 또는 명령어 자체에 대한 적합성을 검증한다. 명령어 포인터를 직접적으로 변조시킬 수 있다는 것은 실행흐름을 조작하여 공격코드를 수행시킬 수 있다는 것을 의미하기 때문에 Table 7과 같이 위험한 문제점으로 분류된다. 따라서 MS 크래시 분석도구는 명령어 포인터 변조 문제점이 존재하는지를 우선적으로 판단한다.

Table 7. Exceptions caused by invalid modification of instruction pointer

Exception type	Analysis function	Exploitability	Description
-	IsFaultingInstructionOnStack	E	스택 영역에 존재하는 명령어 수행 과정에서 예외 발생
Illegal Instruction	-	E	잘못된 형식의 명령어 수행
Privileged Instruction	-	E	권한 없는 명령어 수행

3.2 스택 버퍼 오버플로우

크래시 발생 원인이 스택 버퍼 오버플로우(stack buffer overflow)에 의해 다른 데이터가 변조되어 발생한 것인지를 검사한다. 데이터 검증 보다는 주로 스택 보호 메커니즘에 의해 발생된 예외인지 검사한다. 스택 버퍼 오버플로우가 발생하게 되면 스택에 저장된 복귀주소(return address)를 포함한 주요 데이터를 변조하여 실행흐름을 조작하는 것이 가능하므로 Table 8에서와 같이 위험한 문제점으로 분류된다[6].

Table 8. Exceptions caused by stack overflow

Exception type	Exploitability	Description
Guard Page Violation	E	보호 페이지 접근위반
Stack Buffer Overrun	E	버퍼 보안 체크위반

3.3 힙 손상

크래시 발생 원인이 힙 구조 손상에 의한 것인지 검사한다. 주로 동적 메모리 할당 및 해제 시 힙 매니저가 손상된 힙 구조를 참조할 때 발생한다[7]. 힙 손상은 다른 힙 블록 내 데이터를 변조하여 실행흐름을 조작하는 것이 가능하기 때문에 Table 9와 같이 위험한 문제점으로 분류된다.

Table 9. Exception caused by corruption of heap memory

Exception type	Exploitability	Description
HEAP_CORRUPTION	E	힙 메모리 구조 손상

3.4 실행 접근위반

크래시의 원인이 실행 방지 보호가 설정되어 있는 메모리의 명령어를 실행하여 발생한 것인지를 확인한다. 해당 문제점은 명령어 포인터 변조와 마찬가지로 실행흐름이 비정상적으로 조작되는 것을 의미하므로 위험한 문제점으로 분류된다. 하지만 유저모드에서 접근위반 예외 발생 시 참조하는 메모리 주소가 널(NULL)에 인접한 메모리 주소일 경우 조작 가능성이 적다고 간주하며[8], 상대적으로 낮은 위험성을 가진 문제점으로 분류한다. 예외가 발생하면 프로세서 모드와 크래시 발생원인 주소를 확인하여 Table 10과 같이 위험도를 부여한다.

Table 10. Exceptions caused by execution access violation

Mode	Memory	Exploitability	Description
K	-	E	실행 접근 위반
U	not near NULL	E	널 비인접 주소 메모리 내 명령어 실행 접근위반
U	near NULL	PE	널 인접 주소 메모리 내 명령어 실행 접근위반

Table 11. Exceptions caused by write access violation

Mode	Memory	Exception level	Exploit ability	Description
U	not near NULL	-	E	널 주소 비인접 메모리로의 쓰기 접근위반
U	near NULL	-	PE	널 주소 인접 메모리로의 쓰기 접근위반
K	kernel	-	E	커널영역 메모리로의 쓰기 접근위반
K	-	second chance	E	2차 예외단계의 쓰기 접근위반
K	user	frist chance	PNE	1차 예외단계의 유저영역 메모리로의 쓰기 접근위반

3.5 쓰기 접근위반

크래시 발생 원인이 권한 없는 메모리에 대한 쓰기 명령어 수행에 의한 것인지 검사한다. 해당 크래시의 문제점은 쓰기 대상이 되는 주소가 임의로 조작이 가능한 경우 실행 흐름을 변경할 수 있어 위험한 문제점으로 분류된다. 실행 접근위반 예외에서와 마찬가지로 크래시 발생원인 주소를 기반으로 위험도를 달리하며, 커널모드일 경우 추가적으로 예외 발생단계도 고려하여 Table 11과 같이 위험도를 부여한다.

Table 12. Exceptions caused by read access violation

Mode	Memory	Exception level	Analysis function	Exploit ability	Description
K	-	-	IsFaultingAddressInstructionPointer	E	명령어 포인터로의 읽기 접근위반
U	not near NULL	-		E	널 비인접 주소의 메모리로부터 명령어 포인터로의 읽기 접근위반
U	near NULL	-		PE	널 인접 주소의 메모리로부터 명령어 포인터로의 읽기 접근위반
K	-	-	IsFaultingInstructionControlFlow	E	제어흐름 변경 명령어 수행과정에서 읽기 접근위반
U	not near null	-		E	제어흐름 변경 명령어 수행과정에서 널 비인접 주소 메모리로부터의 읽기 접근위반
U	near NULL	-		PE	제어흐름 변경 명령어 수행과정에서 널 인접 주소 메모리로부터의 읽기 접근위반
U	-	-	IsFaultingInstructionBlockDataMove	PE	블록데이터 이동 명령어 수행과정에서의 읽기 접근위반
K	kernel	-		PE	커널 영역 메모리로부터 블록 데이터 이동 명령어 수행 과정에서의 읽기 접근위반
K	user	second chance		PE	2차 예외단계에서 유저 영역 메모리로부터 블록데이터 이동 명령어 수행과정에서의 읽기 접근위반
-	-	-	IsTaintedDataUsedToDetermineBrachTarget	PE	읽기 접근위반 명령어의 대상 레지스터 값이 이후 분기명령어의 타겟에 영향을 줌
-	-	-	IsTaintedDataUsedInALaterWrite	PE	읽기 접근위반 명령어의 대상 레지스터 값이 이후에 쓰기명령의 원본데이터 주소에 영향을 줌
-	-	-	IsTaintedDataUsedAsSourceForBlockDataMove	U	읽기 접근위반 명령어의 대상 레지스터의 값이 이후 블록데이터 이동 명령어의 원본데이터에 영향을 줌
-	-	-	IsTaintedDataUsedAsAFunctionArgument	U	읽기 접근위반 명령어의 대상 레지스터의 값이 이후 함수 호출의 인자에 영향을 줌
-	-	-	IsTaintedDataUsedAsAFunctionRetVal	U	읽기 접근위반 명령어의 대상 레지스터의 값이 이후 함수의 반환 값에 영향을 줌

-	-	-	IsTaintedDataUsedToDetermineBranchSelection	U	읽기 접근위반 명령어의 대상 레지스터 값이 이후 분기문의 조건에 영향을 줌
K	user	first chance	IsFaultingInstructionBlockDataMove	U	1차 예외단계에서 블록 데이터 이동 명령어 수행 과정에서 유저영역 메모리로부터의 읽기 접근위반
K	near NULL	-	-	U	널 인접 메모리로부터의 읽기 접근위반
K	user	first chance	-	PNE	1차 예외단계에서 유저영역 메모리로부터의 읽기 접근위반
U	near NULL	-	-	PNE	널 인접 메모리로부터의 읽기 접근위반

Table 13. Additional exceptions

Exception type	Mode	Analysis function	Exploitability	Description
Integer Divide by Zero	U	-	PNE	정수를 0으로 나누기 수행
Float Divide by Zero	U	-	PNE	실수를 0으로 나누기 수행
WX86 breakpoint	-	-	U	브레이크 포인트 명령어 수행
breakpoint	-	-	U	브레이크 포인트 명령어 수행
BugCheck	K	WasBugCheckDetected	U	심각성에 대한 판별이 불가능한 예외
-	U	DoesStackTraceContainUnknownFunctions	U	스택추적에 알려지지 않은 심볼정보가 포함

3.6 읽기 접근위반

크래시 발생 원인이 권한 없는 메모리에 대한 읽기 명령어에 의한 것인지 검사한다. 해당 크래시는 접근 불가능한 메모리에서 데이터를 읽어오는 것이기 때문에 크래시 발생 시점의 문제점만으로는 위험도를 판단하기 어렵다. 하지만 읽어온 데이터가 이후에 사용되는 방향에 따라 실행흐름이 조작되는 문제점이 유발 될 가능성이 있다. 예외가 발생하면 프로세서 모드와 참조 메모리 주소 및 추가 분석 함수를 통해 Table 12와 같이 위험도를 부여한다.

3.7 기타 예외 판단정책

기타 예외 판단정책으로는 Table 13과 같이 '0'으로 나누기, 브레이크 포인트 명령어 수행 등이 있다.

4. MS 크래시 분석도구 문제점 및 개선방안

MS 크래시 분석도구는 읽기 접근위반으로 인해 발생한 크래시의 경우 위험도 판단에 한계점을 가지고 있다. 읽기 접근위반으로 인한 크래시는 예외가 발생한 상황의 데이터만으로는 위험도 판단이 어려우며, 예외발생지점 이후의 명령어를 확인하여 예외를 유발하는 데이터에 영향을 받는 명령어가 존재하는지를 검사해야 한다. 이를 위해 해당 유형의 예외에 대해서는 테인트 분석을 수행하지만 예외를 유발하는 명령어가 속한 기본 블록 내에서만 분석이 이루어지기 때문에 위험도 판단 결과의 신뢰도가 낮을 수밖에 없다.

2012년 일부 소프트웨어를 대상으로 수행된 취약점 발굴 연구 결과에 따르면 6개월간의 퍼징을 통해 발생한 크래시를 MS 크래시 분석도구로 확인한 결과, Table 11과 같이 크래시의 상당 부분이 읽기 접근위반에 의해 발생하였음이 확인되었다[9].

Table 14. The result of fuzzing domestic S/W for 6 months [9]

Target program	Exception type	The number of exception
Gom player	read access violation	5041
	write access violation	1567
	etc	1
	total	6609
Hangeul 2005	read access violation	334
	write access violation	101
	etc	13
	total	448

발생빈도가 높은 읽기 접근위반 크래시의 분석 결과에 높은 신뢰도를 보장하기 위해서는 기본 블록 이후의 명령어에 대해서도 분석이 이루어져야한다. 이를 위해서는 크래시 유발 명령어가 속한 기본 블록뿐만 아니라 이후 수행될 기본 블록들을 도출하고, 도출된 기본 블록들 간의 관계를 고려하여 실행흐름 변경이 가능한 명령어의 존재 여부를 확인해야 한다.

크래시 유발 명령어 수행에 의해 변경되는 데이터를 입력 데이터로 정의하고, 레지스터와 메모리 내 데이터를 기반으로 해당 데이터가 소멸될 때까지의 데이터 흐름을 분석하여 영향 받는 명령어들을 조사한다면 보다 확실한 위험도 판단이 가능하다.

5. 결 론

MS 크래시 분석도구는 크래시 분석 시 프로세서 모드, 크래시 발생원인 주소, 예외타입, 예외세부타입, 예외탐지단계 정보 등과 같이 다양한 정보를 기반으로 위험도 등급을 부여하며, 예외적으로 읽기 접근위반에 의해 발생한 크래시

는 테인트 분석기법을 이용하여 추가분석을 수행한다. 그러나 읽기 접근위반 크래시의 경우 크래시 발생지점이 속한 기본 블록 내에서만 테인트 분석이 이루어진다는 한계점을 확인하였으며, 기본 블록을 도출하고 블록간의 관계를 고려하여 크래시 발생지점 이후의 명령어 확인이 이루어져야 위험도 판단 결과의 신뢰도를 높일 수 있음을 나타내었다. 제한적으로 수행되는 테인트 분석을 확장하여 크래시 발생지점 이후의 기본 블록들을 분석할 수 있도록 개선한다면 크래시 분석도구의 활용가치는 더욱 커질 것으로 기대된다.

참 고 문 헌

[1] Microsoft, “First and Second chance exception handling,” 2013년 4월 확인.

[2] MSEC, “!exploitable Crash Analyzer - MSEC Debugger Extensions,” Mar., 2009.

[3] Microsoft Security Engineering Center, “The History of the !exploitable Crash Analyzer,” Apr., 2009.

[4] Microsoft, “EXCEPTION_RECORD structure,” 2012.10, http://msdn.microsoft.com/en-us/library/windows/desktop/aa363082(v=vs.85).aspx

[5] J. Newsome and D. Song. “Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software,” NDSS ’05, Feb., 2005.

[6] C. Cowan et al., “StackGuard: automatic adaptive detection and prevention of buffer-overflow attacks,” 7th USENIX, Jan., 1998.

[7] Alexander Sotirov, “Heap Feng Shui in JavaScript,” BlackHat, 2007.

[8] Adel Abdouchaev et al., “Analyze Crashes to Find Security Vulnerabilities in Your Apps,” Nov., 2007.

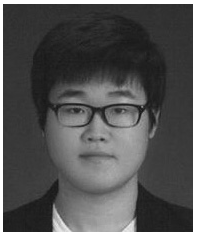
[9] Jaecheol Ryou et al.,, “A Study on Major Domestic S/W Vulnerability Discovery and Analysis Method,” Oct., 2012.

노 명 선

e-mail : nmsnms@kisa.or.kr
1990년 건국대학교 전자공학과(학사)
2011년 광운대학교 경영대학원(석사)
현 재 전남대학교 정보보안협동과정 박사과정
2004년~2013년 한국인터넷진흥원 단장
2013년~현 재 국립외교원 글로벌리더십 과정 파견
관심분야 : 정보보호, 네트워크 및 시스템 보안

나 종 배

e-mail : jbnah123@gmail.com
2010년 광운대학교 경영대학원(석사)
현 재 전남대학교 정보보안협동과정 박사과정
현 재 Vice President of Kinetic Technologies in Sillicon Vally CA
현 재 GNK Network Korea 대표
관심분야 : 정보보호, 네트워크 보안



정 광 운

e-mail : exso@home.cnu.ac.kr
2012년 충남대학교 컴퓨터공학과(학사)
현 재 충남대학교대학원 컴퓨터공학과 석사과정
관심분야 : 정보보호, 시스템 보안

류 재 철

e-mail : jcryou@home.cnu.ac.kr
1985년 한양대학교 산업공학과(학사)
1988년 Iowa Sate University 전산학(석사)
1990년 Northwestern University 전산학(박사)
1991년~현 재 충남대학교 컴퓨터공학과 교수
관심분야 : 정보보호, 네트워크보안, 암호학, 보안프로토콜

노 봉 남

e-mail : bongnam@jnu.ac.kr
1978년 전남대학교 수학교육과(학사)
1982년 KAIST 대학원 전산학과(석사)
1994년 전북대학교 대학원 전산과(박사)
1983년~현 재 전남대학교 전자컴퓨터 공학부 교수
2000년~현 재 전남대학교 시스템보안연구센터 소장
관심분야 : 정보보안, 시스템 및 네트워크 보안