情 報 保 護 學 會 誌
第 16 卷 第 5 號, 2006. 10

결함 주입을 이용한 소프트웨어 보안 테스팅

김 기 범*, 최 영 한*, 양 진 석*, 홍 순 좌*

요 약

사이버 공격이 급증함에 따라 이를 사전에 효과적으로 예방할 수 있는 소프트웨어 보안 테스팅의 필요성이 점차 증대되고 있다. 결함 주입 기법은 사이버 공격과 마찬가지로 프로그램의 외부에 노출된 인터페이스에 고의적으로 결함을 주입하는 방법으로 보안 취약점을 찾는 우수한 방법이다. 이 논문에서는 결함 주입 기법의 기본적인 개념을 살펴본다. 또한, 결함 주입을 수행하는 절차인 결함 주입 대상 외부 인터페이스 선택, 결함 유발 가능 입력 데이터 생성, 결함 주입에 따른 이상현상 탐지 각각에 대한 개요 및 최근 동향을 기술한다.

Ⅰ. 서 론

최근 이미지 파일 및 오피스 파일에 의한 버퍼 오버플로우 취약점이 발표되고, 많은 공격이 이루어짐에 따라 해당 파일을 처리하는 프로그램의 보안 취약점을 사전에 찾아내기 위한 연구가 활발히 진행 중이다. 이들 취약점은 프로그램에서 전혀 예상치 못한 형태로 데이터를 변형하여 해당 파일을 처리하는 프로그램에 입력함으로써 버퍼 오버플로우를 유발시키는 특징을 갖고 있다.

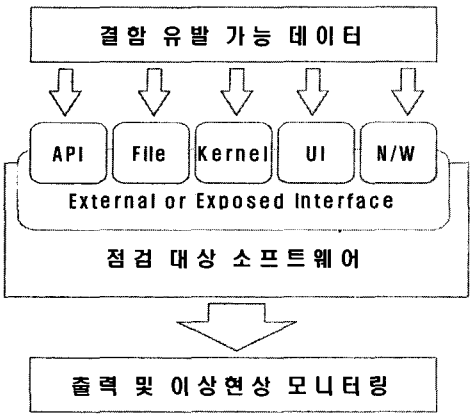
소프트웨어의 취약점 분석을 위하여 소스 코드 검사(audit), 정적 분석(static analysis), 역공학(reverse engineering), 결함 주입(fault injection) 기법 등을 널리 이용하고 있다. 근본적으로 소프트웨어에 내포하고 있는 모든 취약점을 식별하는 것은 불가능하기 때문에 이들 기법들은 서로 상호 보완적인 성격을 갖고 있으며 서로 발견할 수 있는 결함이 다르다. 예를 들어, 이미지 파일 및 오피스 파일 처리 잘못에 기인한 버퍼 오버플로우 취약점은 소스 코드 검사 및 정적 분석을 통하여 발견하기 매우 어렵고, 결함 주입 기법을 통하여 비교적 용이하게 찾을 수 있다.

결함 주입 기법은 소프트웨어 테스팅 중 블랙 박스 테스팅의 일종으로 주로 COTS(Commercial-Off-The-Shelf: 상용제품)를 대상으로 수행한다. 블랙박스 테스팅 방법은 프로그램의 내부 구조 및 코드에 대한 사전 지식 없이 테스트를 수행하는 방법이다. 특히, COTS는 프로그램의 소스 코드 및 내부 스펙이 공개된 경우가 거의 없기 때문에 해당 제품의 취약점을 검증하기 위해서는 결함 주입과 같은 새로운 접근 시도가 필요하다.

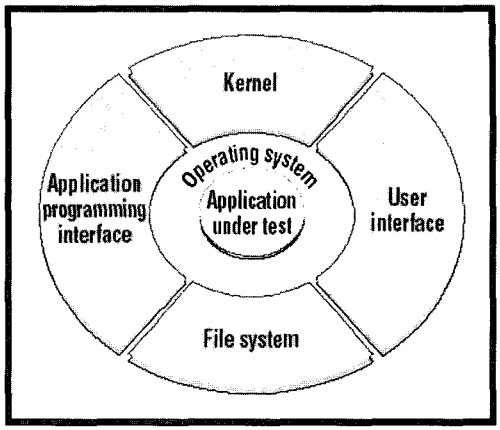
결함 주입 기법은 소프트웨어 시스템의 외부에 노출된 API(Application Programming Interface), 네트워크, 파일, 유저 인터페이스 등에 대하여 프로그램이 예상치 못한 다양한 값을 입력하여 취약점을 점검하는 방법이다. 결함 주입 기법은 퍼징(fuzzing) 혹은 강건성 테스팅(robustness testing) 등의 용어로도 혼용하여 사용하고 있다. 결함 주입 대상을 네트워크에 초점을 맞춘 연구자들의 경우 주로 퍼징이라는 용어를 사용한다. 또한 소프트웨어 테스팅 관점에서 예기치 못한 입력에 대하여 자연스럽게 처리할 수 있는가 여부를 점검하는 것을 강건성 테스팅이라는 용어로 사용하고 있다.

결함 주입에 대한 개념은 초창기 하드웨어의 안전성 및 신뢰성을 점검하기 위한 테스팅 방법으로 사용되었고, 1990년 위스콘신대학교에서 Unix 운영체제에 대한 검증 도구인 FUZZ[1]를 발표하면서 소프트웨어 테스팅 방법으로 사용되고 있다. 최근에 소프트웨어의 품질을 높이고 외부 공격으로부터 보다 안전한 시스템을 개발하기 위하여 많이 사용되는 방법이다. 이는 결함 주입과 사이버 공격이 주로 외부의 악의적인 입력에 기인하는 공통된 특징에 따라 소프트웨어의 취약점 분석에 활용도를 높이고 있다.

* 국가보안기술연구소 (kibom@etri.re.kr, yhch@etri.re.kr, jsyang@etri.re.kr, hongsj@etri.re.kr)



〔그림 1〕 결함 주입의 절차



〔그림 2〕 시스템 결함 주입 포인트

결함 주입 기법의 절차는 우선 결함을 주입할 대상 외부 인터페이스를 선택하고, 선택된 외부 인터페이스에 결함을 유발할 수 있는 데이터를 점검 대상 프로그램에 주입하여 결과 값 및 이상현상을 모니터링 하는 것이다. 〔그림 1〕은 결함 주입 기법의 일반적인 과정과 특징을 나타내고 있다.

이 논문에서는 결함 주입 기법의 절차에 따라 개요 및 연구 동향을 기술한다. 2장에서는 결함 주입 대상 외부 인터페이스에 대하여 살펴보고, 외부 인터페이스별 연구 동향을 기술한다. 3장에서는 결함을 유발할 수 있는 입력 데이터의 생성 방법에 대하여 살펴본다. 4장에서는 주입된 결함에 대한 결과 값 및 이상현상을 어떻게 모니터링 할 것인가를 기술하고, 5장에서 결론을 맺는다.

II. 결함 주입 대상 외부 인터페이스

결함 주입은 프로그램의 외부에 노출된 인터페이스를 대상으로 한다. 결함 주입의 연구들은 결함을 유발할 수 있는 데이터를 어디에 주입할 것인가에 따라 상이하게 발전하여 왔다. 즉, 범용적인 결함 주입 방법이 존재하는 것이 아니라, 외부 인터페이스별로 결함 주입 방법 및 이에 따른 모니터링 방법이 발전하고 있다. 결함을 주입할 대상으로 보통 파일, 레지스트리, API, UI(User Interface), 네트워크 인터페이스, 데이베이스 인터페이스, 명령행 입력인자(command line argument)가 있다〔2〕.

결함 주입 포인트를 시스템 자체만으로 한정하여 James A. Whittaker는 커널, API, UI, 파일, 운영체제 등으로 나누었으며 〔그림 2〕와 같다〔3〕. 즉, 어플리케이션의 결함을 유발할 수 있는 요인으로 자체의 어플리케이션 뿐만 아니라 주위의 환경 즉 운영체제, 각종 외부 인터페이스에도 영향을 많이 받음을 언급하고 있으며 이것을 Invisible User라고 지칭하였다. 이와 같은 이론을 기반으로 James A. Whittaker는 Holodeck〔4〕이라는 결함 주입 도구를 구현하여 시판하고 있다.

이 장에서는 결함 주입에 주로 사용되는 외부 인터페이스인 API, 파일, 커널, UI, 네트워크 각각의 연구 동향에 대하여 기술한다.

1. API

API를 대상으로 결함 주입을 수행하는 것은 소프트웨어가 사용하는 API의 파라미터에 비정상적인 값을 삽입하여 대상 소프트웨어의 이상현상 발생 여부를 테스팅하는 방법이다.

대표적인 프로젝트로 Balistar〔5,6〕가 있다. Balisar는 COTS 프로그램에 대해 강건성을 테스트하기 위해 API에 대해 다양한 값들은 삽입하여 소프트웨어를 테스팅한다. 데이터가 자체 에러처리 메커니즘에 의해 에러로 간주되는 것을 방지하기 위해 함수 파라미터의 데이터 타입을 고려하여 입력을 삽입한다. Ballistar의 경우 입력에 대해 발생할 수 있는 이상현상을 5개로 나눈 CRASH 스케일로 검증하였고, 이를 4장에서 보다 자세히 기술한다.

2. 파 일

비정상적인 데이터를 가진 파일을 이용하여 결함 주

입을 수행하는 방법이다. 대부분의 소프트웨어는 설정 파일이든 이전의 작업을 저장한 파일이든 파일을 읽는 메커니즘이 구현되어 있다. 해당 모듈의 취약점을 검증하기 위하여, 파일에 결함 유발 데이터를 삽입하고 이것을 대상 소프트웨어가 읽게 하여 이상현상을 유발시키는 방법이다.

파일을 이용하여 결함 주입을 수행하는 대표적인 프로그램으로 FileFuzz[7]와 SPIKEfile[8]이 있다. FileFuzz는 정상적인 파일에 대해 임의의 위치에 임의의 값을 삽입하는 프로그램이다. 비정상으로 생성한 파일을 자동으로 실행을 시킴으로써 이상 현상을 발생시킨다. SPIKEfile은 네트워크 퍼징을 수행하기 위해 구현한 SPIKE을 변형하여 파일에 초점을 맞추어 퍼징을 수행할 수 있도록 구현한 프로그램이다. 사용방법은 SPIKE와 유사하며 파일 포맷에 대해 직접 바이너리 값들을 입력으로 사용해야 한다.

3. 커 널

커널에 대해 결함 주입을 수행하는 방법은 사용자 레벨에서 직접적으로 커널 영역으로 접근할 수 없기 때문에 커널에 접근할 수 있는 시스템 콜을 이용한다. 결함 주입을 할 때 커널에 접근을 할 수 없는 영역이 있으면 그곳은 안전하다고 할 수 있다. 그 이유는 해당 영역에 접근을 할 수 없으므로 결함을 유발할 수 없기 때문이다.

sysfuzz[9]는 시스템 콜과 해당 파라미터의 값을 임의의 값으로 변경시킴으로서 이상현상을 유발시키는 프로그램이다. 시스템 콜을 통해 전달된 값들이 커널 영역에서 사용되므로 이상이 발생하였을 경우 커널 패닉(kernel panic)에 빠지게 된다.

[10]은 유닉스의 ptrace 함수를 이용하여 결함을 삽입하는 프로그램을 구현하였다. ptrace 함수를 이용하는 경우 대상 프로그램에 접근을 쉽게 할 수 있으므로 다양한 결함을 삽입할 수 있는 장점이 존재한다. ptrace 함수의 경우는 소프트웨어 테스팅의 과정 중 이상 현상을 감시하기 위한 디버거로 사용되는 경우도 많이 있다.

4. UI

UI는 사용자의 입력을 받아들이는 부분으로 다양한 값들이 설정될 수 있으며 위험이 가장 많이 노출된 부분이다. GUI(Graphic User Interface)인 경우에는 문자열 입력보다는 설정에 관련된 부분이 많으며 설정 옵션들의 조합을 통해 소프트웨어 테스팅을 수행한다. 문자열을 입력으로 받아들이는 부분이 있으면 다양한 길이의 다양한 문자 조합을 입력으로 삽입한다.

[11]은 Windows의 GUI에 대해 결함 주입을 수행한다. GUI에 대해 자동으로 결함 삽입을 수행하기 위해 GUI가 수행될 때 호출되는 다양한 메시지 관련 함수에 대해 메시지 값과 여타의 다른 값들을 변경시킴으로써 취약성을 분석한다.

5. 네트워크

네트워크 인터페이스에 대한 결함 주입은 네트워크 포트 및 원격에서 호출 가능한 함수에 대하여 부적당한 데이터를 의도적으로 주입하여 잘못된 코드 실행 및 취약점을 찾아내는 방법이다. 결함 주입을 수행하는 외부 인터페이스 중 가장 활발히 연구가 진행되는 분야로 주로 퍼징이라는 용어로 많이 불린다. 최근 네트워크에 대한 결함 주입은 프로그램의 실행을 기록하였다가 데이터 포맷 및 내용을 변경하여 자동으로 데이터를 주입할 수 있는 일반화된 결함 주입 도구들이 발표되고 있다.

네트워크 인터페이스를 통한 결함 주입 프로젝트 중 가장 유명한 것으로 Protos가 있다[12]. Protos 프로젝트는 핀란드 Oulu 대학에서 진행하고 있으며, 프로토콜 구현상의 보안취약성을 평가할 수 있는 방법론 및 도구를 개발하는 프로젝트로서 SNMPv1, ISAKMP, SIP 프로토콜 등의 보안점검 도구가 공개되어 있다. 2006년 DefCon에서 발표된 최신 네트워크 퍼저인 GPF(General Purpose Fuzzer)는 응용 프로그램 수준의 기록 및 재생(Record/ Playback) 기능을 지원한다[13]. 이를 통하여, 포맷 스트링 공격 및 버퍼 오버플로우 공격을 테스트할 수 있다.

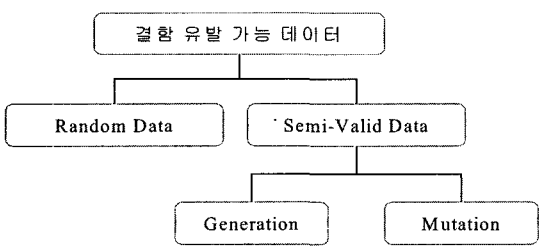
상기 프로젝트 이외에도 네트워크를 대상으로 결함 주입을 수행하는 도구는 [표 1]과 같다.

Ⅲ. 결함 유발 가능 입력 데이터

결함 주입 기법에서 가장 많은 연구가 이루어진 분야로서, 결함 유발을 위하여 주입할 데이터를 어떻게 생성할 것인가에 초점을 맞추고 있다. 결함을 유발할 수 있는 데이터를 생성하는 방법은 [그림 3]과 같다.

〔표 1〕 네트워크 결함 주입 도구

도구 이름	설 명
Scratch	- 간단한 패킷으로 다양한 취약점을 찾는 도구 - SSL, SMB의 퍼징을 위한 프레임워크를 수립
Anti-parser	- 여러 가지 데이터 타입을 갖는 파일 포맷이나 네트워크 프로토콜의 API에 대한 퍼징을 목적으로 함
Peach	- COM/ActiveX, SQL, Shared library, 네트워크 애플리케이션을 퍼징 - 짧은 개발주기, 소스코드 재사용, 사용의 용이성, 유연성이 목표
SMU-DGE	- FTP, SMTP, IMAP, POP3, HTTP, RTSP, MSSQL, DCE-RPC, SMB 뿐만 아니라 IE와 같은 로컬 애플리케이션까지도 퍼징
SPIKE	- block based 퍼징 도구로써 C 언어로 구현 되어 있음 - HTTP, DCE-RPC, SMB 등의 프로토콜 퍼징 가능
Fuzz-ball2	- TCP/IP의 옵션을 퍼징하는 도구
ISIC	- IP Stack Integrity Checker - TCP, UDP, ICMP 등의 프로토콜 안전성 테스팅 도구
IP6sic	- IPv6 프로토콜 스택을 퍼징하는 도구 - ISIC와 비슷한 도구임
Efuzz	- 알려지지 않은 버퍼 오버플로우를 찾아내기 위한 Win32 TCP/UDP 프로토콜 퍼징 도구
BSS	- Blue Tooth Stack Smasher - BSS는 L2CAP계층 퍼징 도구
Radius Fuzzer	- RADIUS를 퍼징할 수 있는 도구
Hydra	- 응용 프로그램 수준의 기록 및 재생 - 클라이언트/서버 간의 실시간 통신 패킷을 가로채어 변경 후 송신
Auto-dafe	- 응용 프로그램 수준의 기록 및 재생 - 대상 컴퓨터에 Tracer라 불리는 디버거를 연결하여 효율성 제고 방안을 제안한 도구



〔그림 3〕 데이터 생성 방법

결함 주입에서 사용되는 입력 데이터는 크게 랜덤(Random) 데이터와 부분 유효(Semi-Valid) 데이터로 나눌 수 있고, 부분 유효 데이터는 Generation 방법과 Mutation 방법에 의하여 생성할 수 있다.

1. 랜덤 데이터

랜덤 데이터는 결함 주입 대상 소프트웨어와 무관하게 임의의 데이터를 입력하는 것으로 Unix 운영체제 커널 및 유틸리티 프로그램을 점검하기 위하여 개발된 FUZZ가 유명하다. 1990년의 FUZZ를 이용한 Unix 운영체제의 유틸리티 프로그램에 대한 점검 결과 25%에서 33%가 Crash 되는 문제점을 발견하였다〔1〕.

랜덤 데이터는 소프트웨어와는 상관없이 임의의 값을 입력하는 것으로 데이터 타입의 최대 값과 최소 값, 경계 값 등이 많이 사용한다. 또한, 스트링 값인 경우 스트링의 끝에 NULL 값을 대체하는 이상한 값, 매우 긴 스트링 등을 입력으로 많이 사용한다.

랜덤 데이터는 모든 소프트웨어의 결함 주입에 기본적으로 사용되는 데이터인 반면, 소프트웨어의 최소한의 특성조차 고려하지 않음으로 인하여 입력 단계에서 바로 폐기되는 문제가 발생한다. 예를 들어, HTTP 프로토콜을 구현한 웹서비스 소프트웨어에 대한 검증을 수행하기 위해서는 최소한 TCP/IP 프로토콜 계층을 통과할 수 있는 데이터가 필요하다. 이는 대부분의 랜덤 데이터가 웹서비스 소프트웨어 검증 단계 이전인 하위 계층에서 폐기되는 문제점이 발생하기 때문이다.

2. 부분 유효 데이터

부분 유효 데이터는 대상 소프트웨어에 대한 분석을 통하여 해당 소프트웨어에서 최소한의 처리가 될 수 있도록 하는 데이터이다. 부분 유효 데이터를 만드는 방법에는 Generation과 Mutation이 있다.

Generation은 소프트웨어에서 사용하는 데이터 타입 및 프로토콜에 대한 철저한 분석을 바탕으로 테스트를 위한 입력 데이터를 새롭게 만드는 기법이다. 대표적으로 Protos 프로젝트는 네트워크 프로토콜에 대한 철저한 분석을 바탕으로 해당 프로토콜을 구현한 소프트웨어를 검증하기 위하여 필요한 네트워크 패킷 데이터를 생성한다〔12〕. 또한, 도움말 파일 포맷을 알고 있는 상태에서 도움말 보기 어플리케이션을 위해 .chm의 확장자를 갖는 새로운 도움말 파일을 구성하는 방법이 이에 해당한다.

Mutation은 기존에 올바른 데이터에 대한 샘플들을 얻어와 그 일부를 변형시켜 입력 데이터를 생성한다. 예를 들어, 오피스 프로그램에 대한 취약점 검증은 오피스 프로그램에서 우선 샘플 파일을 생성 한 후 생성된 파일에 대한 부분적인 변형을 취한 파일을 오피스 프로그램에 입력으로 사용하여 결함을 관찰한다. 또한, 네트워크 프로그램에 대한 검증은 해당 프로그램의 송·수신 패킷을 캡처한 후 해당 패킷을 변경하여 입력 데이터로 활용한다. Mutation 방법은 비교적 손쉽게 결함 주입을 수행할 수 있고, 상용 제품인 경우 정확한 프로토콜 및 파일 포맷을 파악하기 어렵기 때문에 가장 많이 활용되는 방법이다. Mutation을 이용하여 결함 주입을 수행하는 대표적인 도구로서 Autodafe[14], GPF[13], FileFuzz[7], SPIKEFile[8] 등이 있다.

Ⅳ. 출력 및 이상현상 모니터링

결함이 있는 데이터를 삽입한 후 소프트웨어의 이상현상을 감시해야 한다. [15]에서는 입력으로 결함을 삽입하는 것보다 더 중요한 것은 입력에 대해 출력이 어떻게 나오는가를 감시하는 것이라고 하였다. 이는 결함 주입 기법이 입력을 하기 위해 소프트웨어 테스팅을 수행하는 것이 아니라 출력으로 어떤 이상현상이 발생하는지 검사하는 소프트웨어 테스팅 방법이기 때문이다.

1. 이상현상의 유형

이상현상에 대한 유형을 구분하는 대표적인 방법으로 CRASH 스케일[6]과 STRIDE[16]가 있다. CRASH 스케일은 이상현상이 발생하였을 때 시스템에 미치는 영향에 초점을 맞추고, STRIDE는 CRASH 스케일로 발견한 현상이 시스템에 보안 관점에서 미치는 위협에 초점을 맞추고 있다.

이상현상의 종류를 구분하는 방법으로 대표적인 것이 Balistar에서 사용하는 CRASH 스케일이 있다. CRASH 스케일은 결함이 삽입되었을 때 발생할 수 있는 시스템 현상을 5가지로 구분하였고, 이는 [표 2]와 같다.

Balistar의 경우 Catastrophic, Restart, Abort의 세 가지 경우만 찾는다. Silent와 Hindering의 경우는 자동으로 찾기는 힘들고 사람의 개입이 들어가 해당 상황을 분석해야 가능하다. CRASH 스케일을 통해 이상현상을 발견하였을 때 해당 취약점이 시스템에 어떤 영향을 미칠 것인지 분석해야 한다. 이를 위하여 Michael Howard 등에 의하여 제안된 방법으로 [표 3]와 같은 STRIDE 유형이 있다.

STRIDE 유형은 CRASH 스케일에서 찾은 이상현상을 보안관점에서 어떤 영향을 미칠 것인지 판단하

[표 2] CRASH 스케일

Scale	설 명
Catastrophic	결함 데이터에 의해 시스템에 심각한 영향을 미치는 상황이다. OS는 해당 상황을 해결하기 위해 재부팅 시킨다.
Restart	해당 상황은 소프트웨어가 hang 상태에 들어가 더 이상의 진행을 하지 않는 경우이다. 이러한 상황에서는 복구를 위해 해당 소프트웨어를 재실행 시켜야 한다.
Abort	결함 데이터에 의해 비정상적으로 소프트웨어가 종료하는 상황이다. 해당 상황인 경우에는 시스템에서 예외를 발생시킨 후 소프트웨어를 종료한다.
Silent	해당 상황은 에러가 발생하였으나 내부적으로 처리가 되어 에러를 반환하지 않는 경우이다.
Hindering	에러를 발생했으나 부적절한 에러 코드를 반환하는 경우이다.

[표 3] STRIDE

	설 명
S	- Spoofing identity - 공격자가 다른 사용자인 것처럼 위장하거나, 가짜 서버를 진짜 서버처럼 위장하는 것이다.
T	- Tempering with data - 악의를 가지고 데이터를 변조하는 것을 말한다.
R	- Repudiation - 상대방이 증명할 방법이 없는 상황에서 자신이 한 조작이나 행위를 부인하는 것이다.
I	- Information disclosure - 정보의 취득이 허가되지 않은 사람에 정보가 유출되는 것이다.
D	- Denial of service - 정당한 사용자에 대한 서비스를 못하게 만드는 위협이다.
E	- Elevation of privilege - 자신의 권한을 상승시켜 관리자 권한을 획득함으로써, 시스템을 손상시키거나 완전히 파괴하는 것이 가능하게 되는 위협을 말한다.

는 방법으로 이상 현상의 위험성을 판단할 수 있다.

2. 이상현상 모니터링 방법

최근까지 결함 주입에 따른 이상 현상을 모니터링 하는 방법에 대한 구체적인 연구는 많이 이루어지고 있지 않다. 주로 프로그램에 결함을 주입한 후 시스템 다운, 재시작, 프로그램 종료 등을 관찰하여 입력한 데이터가 시스템에 이상을 야기함을 파악한다. 보다 구체적인 이상현상의 원인을 찾을 수 있는 방법은 크게 3가지로 분류할 수 있다.

- 디버거로 소프트웨어의 동작을 감시하는 방법 : 디버거로 이상현상을 관찰하는 방법으로 대부분의 결함 주입 기법은 이 방법을 사용하여 이상현상을 찾는다. 디버거로 윈도우즈에서는 WinDBG[17]를 리눅스에는 GDB[18]를 많이 사용한다. 기존의 디버거를 사용하여 이상현상을 발견할 수 있으며 직접 디버거를 구현하여 이상현상을 감시할 수도 있다. 윈도우즈 경우에는 Debug API[18]가 지원이 되어 디버거를 구현할 수 있도록 하고 있다. 리눅스의 경우에는 2가지 방법이 존재한다[20]. ptrace 함수를 이용하는 방법과 /proc 파일 시스템을 이용하는 방법이다. ptrace 함수를 사용하는 방법은 윈도우즈에서 Debug API를 사용하는 방법과 유사하다.
- 이상현상이 발생했을 때의 정보를 저장하는 방법 : 이상현상이 발생했을 때의 메모리의 데이터를 파일로 저장하는 방법이다. 리눅스 계열인 경우에는 예외가 발생할 때 메모리 정보를 core dump 파일로 저장한다. 추후 이 파일을 읽어와 이상현상을 분석한다.
- 함수의 반환 값을 보고 판단하는 방법 : 함수가 처리가 된 후 처리 결과에 대한 상태를 반환한다. 해당 방법은 함수의 반환 값을 감시하여 이상현상을 발견하는 방법이다.

위의 방법 중 디버거로 이상현상을 감시하는 방법이 많이 사용되고 있고, FileFuzz[7]의 경우에는 Debug API를 이용하여 예외가 발생했을 때의 레지스터 정보를 출력하도록 하고 있다. Balistar[6]의 경우는 API를 대상으로 결함 주입을 수행하며 함수의 반환 값을 통해 이상 현상을 감시한다.

V. 결 론

결함 주입 기법은 소프트웨어 테스팅의 보조적인 방법에서 보안 취약점을 찾기 위한 필수적인 테스팅 기법으로 발전되어 왔다. 결함 주입이 가능한 다양한 외부에 노출된 인터페이스에 대하여 활발하게 연구가 진행되고 있다. 이들 연구는 주로 결함 유발 데이터 생성에 초점을 맞추고 있고 최근에는 Mutation을 통한 결함 데이터 생성이 대세를 이루고 있다. 아직까지 주입된 결함에 대한 결과를 측정할 수 있는 체계화된 방법에 대한 연구는 미진한 상태이다. 이는 프로그램의 소스 코드 및 내부 명세를 모르는 상황에서 갖는 기본적인 한계에 기인한다. 그럼에도 불구하고, 결함 주입 기법은 예기치 못한 입력에 대하여 이상현상을 유발할 수 있는 포인트를 찾는 가장 효과적인 방법이다. 향후 화이트박스 테스팅 기법과의 접목을 통하여 결함 주입 기법은 활발한 연구가 진행될 것이다.

참 고 문 헌

[1] Barton P. Miller, Lars Fredriksen, Bryan So, "An empirical study of the reliability of Unix Utilities", *Communications of the ACM*, 33(12):pp.32-44, December 1990

[2] Peter Oehlert, "Violating Assumptions with Fuzzing", *IEEE Security & Privacy*, pp.58-62, March/April 2005

[3] James A. Whittaker, "Software's Invisible Users", *IEEE Software*, 18(31):pp.84-88, 2002

[4] http://www.securityinnovation.com/holodeck/index.shtml

[5] http://www.ece.cmu.edu/~koopman/ballista/

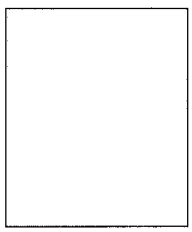
[6] Nathan P. Kropp, Philip J. Koopman, Daniel P. Siewiorek. "Automated Robustness Testing of Off-the-Shelf Software Components", *Proceedings of the 28th Fault Tolerant Computing Symposium*, pp.230-239, June 1998

[7] Michael Sutton, Adam Greene, "The Art of File Format Fuzzing", Blackhat 2005

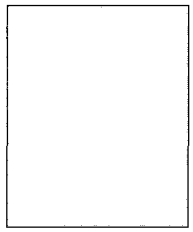
[8] http://labs.idefense.com/software/fuzzing.php#more_spikefile

[9] Ilja van Sprundel, "Unix Kernel Auditing"
[10] Volkmar SIEH, "Fault-Injector using UNIX ptrace Interface", *Internal Report No.:11/93*, Universit at Erlangen-Nurnberg, 1993
[11] Justin E. Forrester, Barton P. Miller, "An Empirical Study of the Robustness of Windows NT Applications Using Random Testing", *4th USENIX Windows System Symposium*, 2000
[12] PROTOS Project, http://www.ee.oulu.fi/research/ouspg/protos/
[13] GPF project, http://www.appliedsec.com
[14] Autodafe, http://autodafe.sourceforge.net
[15] Jeffrey M. Voas, *Software Fault Injection Inoculating Programs Against Errors*, John Wieley & Sons, 1997
[16] Michael Howard, David LeBlanc, *Writing Secure Code*, 2nd Edition, 2002, Microsoft Press
[17] http://www.microsoft.com/whdc/devtools/debugging/default.mspx
[18] http://msdn.microsoft.com/library
[19] http://www.gnu.org/software/gdb/
[20] Diomidis Spinellis, "The Design and Implementation of a Two Process Prolog Debugger", 1989

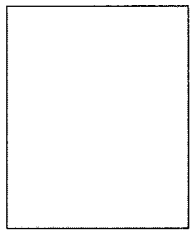
〈著者紹介〉



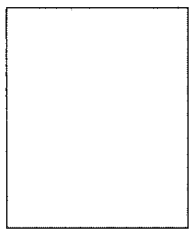
김 기 범 (Kibom Kim)
정회원
1994년 2월 : 제주대학교 정보공학과 졸업
1996년 8월 : 고려대학교 전산과학과 석사
2001년 2월 : 고려대학교 전산과학과 박사
2001년 1월~2004년 7월 : (주)이씨오 개발부장
2004년 8월~현재 : 국가보안기술연구소 연구원
관심분야 : 정보보호, 유비쿼터스컴퓨팅, 분산시스템



최 영 한 (Younghan Choi)
정회원
2002년 2월 : 한양대학교 전자전기학과 졸업
2004년 2월 : 한국과학기술원 전자전산학과 석사
2004년 2월~현재 : 국가보안기술연구소 연구원
관심분야 : 운영체제, 네트워크, 정보보호



양 진 석 (Jinseok Yang)
정회원
2003년 2월 : 성균관대학교 정보공학과 졸업
2005년 2월 : 성균관대학교 컴퓨터공학과 석사
2005년 3월~현재 : 국가보안기술연구소 연구원
관심분야 : 정보보호, 네트워크 보안



홍 순 좌 (Soonjwa Hong)
정회원
1989년 2월 : 숭실대학교 전산학과 졸업
1991년 2월 : 숭실대학교 전산학과 석사
2005년 8월 : 충남대학교 컴퓨터과학과 박사
1991년 2월 ~ 2000년 1월 : 국방과학연구소 선임연구원
2000년 2월~현재 : 국가보안기술연구소 팀장
관심분야 : 정보보호, 시스템 보안