MS 워드프로세서의 입력 파일에 대한 유효성 테스팅 방법에 관한 연구

윤 영 민[†] · 조 성 제[††] · 최 종 천[†††] · 유 해 영[††††]

요 약

본 논문에서는 MS 워드프로세서의 입력 파일에 대한 유효성을 검사하여 MS 워드프로세서의 보안 취약성을 분석하는 방법을 제안하였다. 즉, 워드프로세서에 적용되는 입력 파일의 헤더 정보를 분석함으로써, 그 입력 파일에 존재하는 취약성을 검출하였다. 본 논문에서 수행한 입력 파일 유효성 테스팅은 기존 결함 주입 도구인 Holodeck 및 CANVAS 등으로는 테스트할 수 없다. 제안한 방법은 헤더를 가진 데이터 파일을 입력으로 사용하는 한글, MS 엑셀 등의 입력 파일에 대한 취약성 검출에도 적용될 수 있다. 또한, 대상 소프트웨어들의 결함 허용성 평가 및 신뢰성 평가에도 사용될 수 있다.

키워드 : 입력 파일 유효성 테스팅, 워드프로세서, 취약성

A Study on Validation Testing for Input Files of MS Word-Processor

Young-Min Yun[†] · Seong-Je Cho[††] · Jong-Cheon Choi[†††] · Hae-Young Yoo[††††]

ABSTRACT

In this paper, we propose a method to analyze security vulnerabilities of MS word-processor by checking the validation of its input files. That is, this study is to detect some vulnerabilities in the input file of the word processor by analyzing the header information of its input file. This validation test can not be conducted by the existing software fault injection tools including Holodeck and CANVAS. The proposed method can be also applied to identify the input file vulnerabilities of Hangul and Microsoft Excel which handle a data file with a header as an input. Moreover, our method can provide a means for assessing the fault tolerance and trustworthiness of the target software.

Key Words : Input File Validation Checking, Word-Processor, Vulnerability

1. 서 론

취약성(vulnerability)이란 한 시스템의 보안 절차(security procedures), 설계, 구현, 또는 운용 및 관리 제어(management control) 등에 내재되어 있는 결함(fault) 또는 결점(weakness)이다. 공격자는 보안 침해(breach)나 보안 정책의 위반(violation)을 유발시키는데 취약성을 악용할 수 있으므로, 취약성은 보안 사고의 원인이 될 수 있다. IT 관련 기술의 발달로 소프트웨어의 개발 규모가 커지고 복잡해짐에 따라 점차 소프트웨어 보안 취약성이 증가하고 있다[1]. 또한 소프트웨어 취약성 생명 주기에 의하면 소프트웨어 보안 사고는 보안 결함이 수정되어 발표되기 전까지 증가하는 것으로 나타났다[2].

정보 보호의 주요 과정은 소프트웨어에 대한 보안 취약성을 분석·검출하여 취약성을 악용하는 공격을 방어하는 것이라 볼 수 있다. 현재의 프로그래밍 기술 수준으로는 소프트웨어에 존재하는 모든 취약성을 제거하는 것이 불가능하므로 잘 작동하는 것처럼 보이는 소프트웨어에도 발견되지 않은 보안 결함이 내재되어 있을 수 있다. 소프트웨어 개발자들이 주의를 기울여 소프트웨어를 개발하기만을 바라는 것은 한계가 있으므로 소프트웨어에 존재하는 취약성을 분석하여 조기에 결함을 발견할 수 있는 기법에 대한 연구가 필요하다.

소프트웨어 취약성 테스팅(vulnerability testing)은 소프트웨어에 존재하는 취약성을 분석·검출하는 기술로 소프트웨어의 신뢰성과 품질 검증에 관련이 있다. 현재 소프트웨어 취약성 분석은 소프트웨어 개발자와 보안 전문가의 지적인 능력에 주로 의존하고 있고, 많은 시간과 비용을 요구하므로 효

※ 이 논문은 2006년도 국가보안기술연구소의 위탁연구결과물임(06040).
† 준 회 원 : 단국대학교 전산통계학과 박사수료 (주저자)
†† 종신회원 : 단국대학교 정보컴퓨터과학 부교수 (교신저자)
††† 준 회 원 : 단국대학교 정보컴퓨터과학과 박사과정
†††† 정 회 원 : 단국대학교 정보컴퓨터과학 교수
논문접수 : 2006년 11월 8일, 심사완료 : 2007년 8월 16일

율적으로 취약성을 분석하고 검출하는 데 한계가 있다. 또한 상용(COTS : commercial off the shelf) 소프트웨어의 취약성 악용에 의한 보안 사고는 그 원인을 규명하는 일이 매우 복잡하므로 정확한 원인을 밝히는 것은 쉽지 않다. 따라서 소프트웨어 보안 취약성에 의한 피해를 줄이기 위해서는 취약성을 체계적이고 효율적으로 검출할 수 있는 방법에 대한 연구가 필요하다.

보안 취약성을 검출하기 위한 기존 방법으로는 정형 기법(formal method)과 모의 해킹(penetration testing) 등이 있다. 정형 기법은 주로 소스 코드(source code)가 제공되는 규모가 크지 않은 소프트웨어를 대상으로 적용되며, 모의 해킹은 이미 알려진 취약성의 존재 여부를 파악하는데 사용된다. 따라서 소스 코드가 없이 바이너리 코드(binary code)만 주어지는 상용 소프트웨어의 알려지지 않은 취약성을 분석하고 검출하기에는 적절하지 못하다.

취약성을 검출하기 위해 소프트웨어 결함 주입 방법(SFI : software fault injection)이 이용되기도 한다. Holodeck, CANVAS, Impact 등은 소프트웨어의 내부나 수행 환경에 결함을 주입하고, 주입된 결함으로 인해 소프트웨어가 어떻게 동작하는지를 검사하는 소프트웨어 결함 주입 도구이다. 소프트웨어 결함 주입 방법은 소프트웨어의 안정성(stability)을 측정하는 자동화된 방법이기도 하지만, 보안 취약성을 검출하는 방향으로도 연구되고 있다. 즉, 대상 소프트웨어에 보안 결함을 유발하는 입력의 조합을 찾아내는 연구로도 활발하게 진행되고 있다[3].

본 논문에서는 MS 윈도우즈(Microsoft Windows) 상에서 워드프로세서의 입력 파일에 대한 유효성을 테스트하여, 보안 사고의 발생 가능성을 미연에 방지할 수 있는 방법을 제안한다. 제안한 방법에서는 워드프로세서의 입력 파일들이 고유한 형식을 가지고 있으므로, 그 파일을 처리하기 전에 파일의 형식(특히, 헤더)이 유효한지 아닌지를 검사한다. 유효하지 않은 입력 파일의 경우, 워드프로세서의 취약성을 악용하여 오동작을 유발시킬 수 있다. 제안한 방법은 워드프로세서 입력 파일의 특정 부분을 변경·악용하여 버퍼 오버플로우(buffer overflow)를 유발시키는 공격, 또는 워드프로세서의 정상 동작을 방해하는 서비스 거부(denial of service) 공격의 원인을 파악할 수 있게 하여 준다.

본 논문의 구성은 다음과 같다. 2장에서는 취약성 검출 방법에 관한 기존의 연구들에 설명하고, 3장에서는 기존의 취약성 분석 도구들의 한계에 대해 설명한다. 4장에서는 입력 파일에 내재되어 있을지 모르는 취약성을 검출하기 위한 방법으로 입력 파일 유효성 검사 모델을 제안하며, 5장에서는 4장에서 제안한 모델을 이용하여 DOC 파일에 포함된 취약성 검출의 예를 보인다. 끝으로 6장에서는 논문의 결론 및 향후 연구 방향에 대해 언급한다.

2. 관련 연구

소프트웨어 테스팅은 소프트웨어 개발 과정에서 발생할 수 있는 오류를 찾기 위해 효과적인 테스트 케이스(test case)를 선정하고, 그 데이터를 이용하여 프로그램을 실행시키는 과정이다. 소프트웨어 테스팅 방법은 프로그램의 소스 코드를 대상으로 테스팅을 수행하는 정적 테스팅(static testing)과 프로그램을 실행시켜 테스팅을 수행하는 동적 테스팅(dynamic testing)으로 나누어진다[3].

동적 테스팅은 화이트 박스 테스팅(white-box testing)과 블랙 박스 테스팅(black-box testing)으로 구분된다. 화이트 박스 테스팅은 소프트웨어 소스 코드의 실행 경로를 기반으로 수행되므로 확실하고 객관적인 테스트를 수행할 수 있지만, 기능의 오류를 발견하기 어렵고, 인터페이스나 인터럽트 등의 동적인 부분에 대한 테스트가 불가능하다. 따라서 소스 코드의 규모가 크고 실행 경로의 수가 무수히 많은 소프트웨어의 보안 테스트에는 블랙 박스 테스팅이 주로 적용된다. 특히, 상용 소프트웨어는 사용자와 개발자가 동일하지 않으며, 사용자가 소스 코드를 획득할 수 없는 경우가 대부분이므로 취약성 검출에 블랙 박스 테스팅이 사용된다.

2.1 정형 기법

정형 기법은 소프트웨어 시스템이 지니는 제반 문제들을 해결하기 위한 기법으로, 정형 명세(formal specification)와 정형 검증(formal verification)으로 구분된다. 정형 명세는 소프트웨어 개발 초기 단계에서 개발자로 하여금 모든 요구사항들을 정확하게 명세하게 함으로써 소프트웨어의 강건성(robustness)을 향상시키는 방법이다. 정형 검증은 정형 명세를 기반으로 하여 모델 검사(model checking)나 이론 증명(theorem proving) 등을 통해서 정형 명세된 소프트웨어의 안정성을 증명하는 것이다. 정형 기법에 의한 취약성 검출 방법은 요구 사항 명세에 기술된 내용이 소스 코드에 정확하게 구현되었는지를 수학적으로 검증하기 때문에 정확도가 높다는 장점이 있지만, 보안 요구 사항을 정확히 명세하기 어렵다는 단점이 있다[4, 5].

2.2 모의 해킹

모의 해킹을 이용한 테스트 방법은 보안 팀이 알려진 보안 결함을 이용하여 실제 시스템을 공격하는 방법으로 보안 결함을 검출할 수 있다는 장점이 있지만, 알려지지 않은 보안 결함은 모의 해킹을 통해 검출될 수 없다는 단점이 있다.

Pfleeger 등에 의하면 모의 해킹은 테스트를 하기 위한 테스트 케이스 확인 절차가 복잡하고, 언제 모의 해킹을 중단해야 하는지에 대한 측정 기준이 없으며, 모의 해킹의 대상이 되는 시스템에 대해 잘 알고 있어야 하며, 경험이 풍부한 전문가를 필요로 하므로 테스트 계획을 수립하기가 어렵다고 지적하였다[6].

2.3 소프트웨어 결함 주입 방법

소프트웨어 결함 주입 방법은 일종의 블랙 박스 테스팅으로, 결함 주입 후의 분석을 통해 소프트웨어에 존재하는 결함을 제거하거나 결함을 감내할 수 있는 다양한 방법들을 추가 또는 보완하여 더욱 강건한 소프트웨어를 작성하는데

도움을 준다. 소프트웨어 결함 주입 방법은 저수준 처리기 및 메모리 고장(low level processor and memory failures), 프로세스 메모리 이미지 손상(corruption of a process's memory image), SAN 기반 결함 주입(SAN based fault injection), 인터페이스 결함 주입(interface fault injection), 시스템 혼동(system perturbations) 등으로 결함의 종류와 주입 방법에 따라 구분할 수 있으며, Xception, HYBRID, FERRARI, Janus, ORCHESTRA, FTAPE, Ballista, Fuzz, Holodeck, Wrapping, CANVAS 등의 도구가 있다[3].

Whittaker 등은 소프트웨어 결함 주입 방법을 이용하여 소프트웨어의 보안 결함을 발견하기 위한 방법으로, 소프트웨어의 종속성 공격(attacking software dependencies), 사용자 인터페이스에 대한 공격(attacking user interface), 설계 오류 공격(attacking design), 구현 오류 공격(attacking implementation) 등을 제안하였다[7, 8, 9]. Whittaker 등이 제안한 공격 방법은 소프트웨어 취약성 검출에 효율적이고 유용한 방법으로, 소프트웨어의 플랫폼(platform)과 개발 언어의 종류에 관계없이 응용이 가능하다.

2.4 워드프로세서 취약성 사례

보안 취약성 및 정보 보안 관련 정보 리스트인 CVE (common vulnerabilities and exposures)에 의하면, 문자열 입력 부분에 긴 문자열을 입력하여 버퍼의 메모리 용량을 초과하는 버퍼 오버플로우 공격이나 입력되는 파일의 특정 부분을 변경하여 소프트웨어가 정상적으로 동작하지 못하게 하는 서비스 거부 공격이 발생하는 것으로 나타났다[10]. <표 1>은 CVE에 보고된 취약성 중 MS 워드의 입력 파일에 대한 취약성을 정리한 것이다.

〈표 1〉 CVE에 보고된 MS 워드의 입력 파일에 대한 취약성

Name	Description
CVE-2000-0073	Buffer overflow in Microsoft Rich Text Format (RTF) reader allows attackers to cause a denial of service via a malformed control word.
CVE-2005-0558	Buffer overflow in Microsoft Word 2000, Word 2002, and Word 2003 allows remote attackers to execute arbitrary code via a crafted document.
CVE-2006-0935	Microsoft Word 2003 allows remote attackers to cause a denial of service (application crash) via a crafted file, as demonstrated by 101_filefuzz.
CVE-2006-5994	Unspecified vulnerability in Microsoft Word 2000 and 2002, Office Word and Word Viewer 2003, Word 2004 and 2004 v. X for Mac, and Works 2004, 2005, and 2006 allows remote attackers to execute arbitrary code via a Word document with a malformed string that triggers memory corruption.
CVE-2006-6561	Unspecified vulnerability in Microsoft Word 2000, 2002, and Word Viewer 2003 allows user-assisted remote attackers to execute arbitrary code via a crafted DOC file that triggers memory corruption, as demonstrated via the 12122006-djtest.doc file.

```
Bugtraq ID : 8761
Vulnerable Verson : Microsoft Word 98 Japanese Version
                    Microsoft Word 98
                    Microsoft Word 97 SR2
                    Microsoft Word 97 SR1
                    Microsoft Word 97
                    Microsoft Word 2002 SP2
                    Microsoft Word 2002 SP1
                    Microsoft Word 2002
                    Microsoft Word 2000 SR1
                    Microsoft Word 2000 SP3
                    Microsoft Word 2000 SP2
                    Microsoft Word 2000

1. Open Microsoft Word
2. Save .doc file
3. Modify .doc file by using binary editor as follows
   00 00 00 00 00 a3 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 40 01
   00 00 00 00 00 b4 01 00 00 20 00 00 00 9c 01 00 00 00 00 00 00 9c
   01 00 00 00 00 00 9c 01 00 00 00 00 00 00 9c 01 00 00 00 00 00 00

   00 00 00 00 00 a3 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 40 01
   00 00 62 62 62 62 b4 01 00 00 20 00 00 00 9c 01 00 00 00 00 00 00 9c
   01 00 00 00 00 00 9c 01 00 00 00 00 00 00 9c 01 00 00 00 00 00 00
4. Open Modified .doc file
5. Microsoft Word will crashes
```

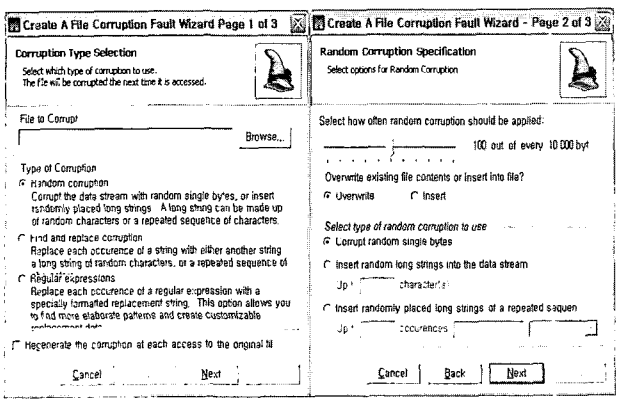
(그림 1) MS 워드의 취약성 악용 사례

(그림 1)은 MS 워드에서 DOC 파일의 특정 부분에 결함을 주입하고, 결함이 주입된 파일을 실행하는 경우, MS 워드가 정상적으로 동작하지 않고 서비스 거부를 유발하는 취약성 악용 사례를 보여준다[11]. MS 워드, 엑셀, MS 파워포인트와 RTF(rich text format) 파일 형식을 이용하는 워드프로세서 등에서 발생한 침해 사례들 대부분이 (그림 1)과 같이 특정 부분에 결함이 주입된 파일을 공격의 매체로 이용한 버퍼 오버플로우나 서비스 거부 공격과 관련이 있다.

3. 기존 취약성 분석 도구들의 한계

소프트웨어 보안 테스팅은 동적 분석의 경우 디버거(debugger)와 같은 역공학(reverse engineering) 도구들을 사용하기도 하지만, 최근에는 소프트웨어에 결함을 주입하여 취약성을 검출하기 위한 자동화 기법들에 대해 연구가 이루어지고 있다.

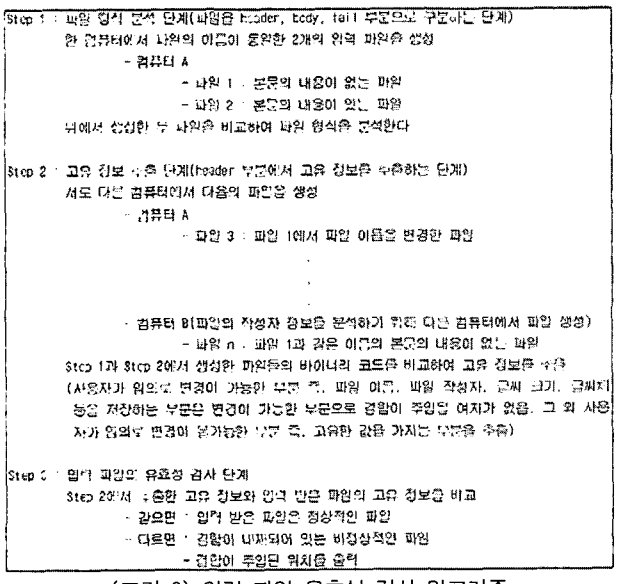
Holodeck은 소프트웨어에 결함을 주입하여 소프트웨어의 강건성을 시뮬레이션할 수 있는 도구로, 테스터와 개발자가 메모리 부족(insufficient memory), 손상된 파일(corrupt file), 손상된 레지스트리 데이터(corrupt registry data), 네트워크 패킷 결함(network packet fault) 등을 대상 소프트웨어에 주입하여 소프트웨어의 보안성을 테스트한다[12]. Holodeck은 (그림 2)와 같이 파일 손상 결함 마법사(file corruption fault wizard)를 이용하여 파일에 결함을 주입하고, 파일에 주입된 결함이 소프트웨어에 어떠한 영향을 초래하는지 테스트하는 도구로는 유용하지만, 결함이 포함된 파일에서 결함의 원인을 검출할 수 없다. 즉, 파일에 결함을



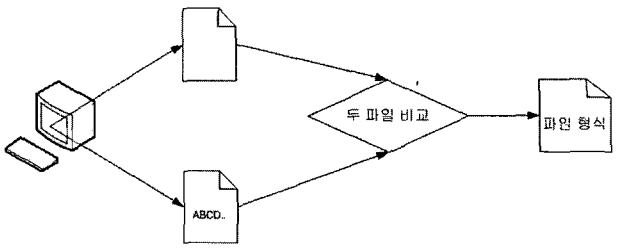
(그림 2) Holodeck을 이용한 결함 주입 예

주입(임의로 변경)하고, 결함이 주입된 파일이 버퍼 오버플로우나 서비스 거부 공격을 유발하는지에 대한 테스트는 가능하지만, 비정상적인 파일에서 결함의 위치를 분석하지는 못한다.

CANVAS는 세계적인 모의 해킹 테스터와 보안 전문가로 이루어진 Immunity사에서 개발된 도구로, 수백 개의 테스트 케이스를 가지고 있는 자동화된 재연 시스템으로, 포괄적으로 보안 테스트를 수행할 수 있다[13]. Impact는 Core Security사에서 개발된 도구로, 시스템에 접근할 수 있는 정보 수집부터 최종적인 보고 단계까지 자동으로 수행할 수 있어 네트워크의 보안 정도를 개선하는데 비용과 시간을 절약할 수 있다[14]. CANVAS와 Impact는 웹 서비스(web service), 데이터베이스(database), 원격 프로시저 호출(RPC : remote procedure call) 등 주로 네트워크와 관련된 취약성을 테스트하는 도구로는 유용하지만 테스트 케이스에 포함되어 있지 않은 취약성은 검출할 수 없다. 즉, (그림 1)과 같이 입력 파일의 취약성 검사에 대한 테스트 케이스가 제공되지 않아 비정상적인 입력 파일에 대한 취약성을 검출할 수 없다. 이 외에도 수많은 취약성 분석 도구들이 존재하지



(그림 3) 입력 파일 유효성 검사 알고리즘



(그림 4) 입력 파일의 형식 분석

만 <표 1>와 같이 비정상적인 파일에 내재되어 있는 취약성은 검출이 불가능하다.

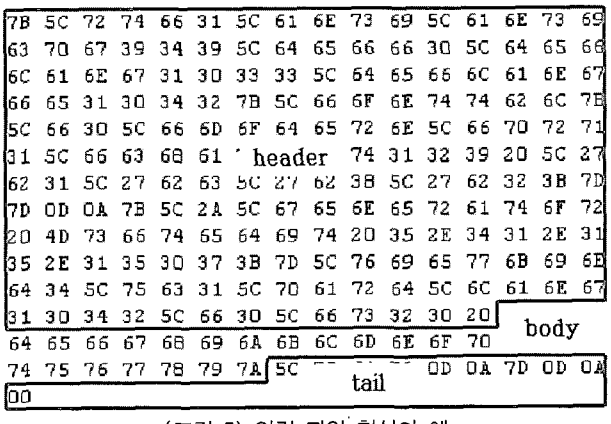
4. 입력 파일 유효성 검사 모델

본 장에서는 입력 파일에 내재되어 있는 취약성을 검출하기 위한 방법으로 입력 파일의 유효성 검사 모델을 제안한다. 제안하는 입력 파일의 유효성 검사 모델은 경험적 방법을 이용하여 3단계(파일 형식 분석, 고유 정보 추출, 입력 파일의 유효성 검사)로 구성되며 알고리즘은 (그림 3)과 같다.

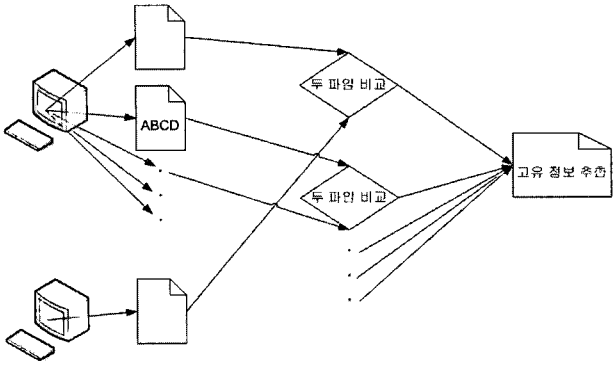
4.1 입력 파일의 형식 분석

MS 윈도우즈 기반 워드프로세서들은 RTF, DOC, HWP 등의 확장자를 가진 파일을 입력으로 사용하며, 고유한 파일 형식을 가지고 있으므로 파일의 유효성을 검사하기 위해서는, 입력 파일의 형식을 분석해야 한다. 입력 파일의 형식을 분석하기 위해, (그림 4)와 같이 동일한 컴퓨터에서 파일의 이름이 동일한 2개의 파일(본문의 내용이 있는 파일과 본문의 내용이 없는 파일)을 생성한다. 생성한 2개 파일을 비교하여 파일의 형식(header, body, tail)을 분석한다.

MS 윈도우즈 기반 워드프로세서들의 입력 파일은 (그림 5)와 같이 파일의 본문이 저장되는 부분인 body, 파일의 기본 정보를 저장하는 부분인 header와 tail 이렇게 3개의 부분으로 구분할 수 있다. 본문의 내용이 없는 파일의 경우, body 부분은 존재하지 않으며, 본문의 내용이 있는 파일의 경우 body 부분은 header와 tail 부분 사이에 존재한다. header와 tail 부분의 크기는 모든 파일이 동일하다.



(그림 5) 입력 파일 형식의 예



(그림 6) 보안 문제를 일으킬 여지가 있는 고유 정보 추출

〈표 2〉 고유 정보 추출을 위해 생성한 파일 리스트

컴퓨터 이름	파일	내용
A	파일 1	본문의 내용이 없는 파일
	파일 2	파일 1과 동일한 이름의 본문의 내용이 있는 파일
	파일 3	파일 1에서 다른 이름으로 저장한 파일
	파일 4	파일 2에서 글자 크기를 변경하여 저장한 파일
	파일 5	파일 2에서 글자 모양을 변경하여 저장한 파일
	파일 6	파일 2에서 글자 스타일을 변경하여 저장한 파일
	파일 7	파일 2에서 글자 색을 변경하여 저장한 파일
	파일 8	파일 2에서 글자 간격을 변경하여 저장한 파일
	파일 9	파일 2에서 문단 맞춤을 변경하여 저장한 파일
	파일 10	파일 2에서 문단 간격을 변경하여 저장한 파일
	파일 11	파일 2에서 다단으로 변경하여 저장한 파일
B	파일 12	파일 1과 동일한 이름의 본문의 내용이 없는 파일

4.2 고유 정보 추출

파일의 고유 정보는 header와 tail 부분 중에서 파일 작성자, 파일 이름 등의 정보를 포함한 부분을 제외한 나머지 부분이다. 파일 작성자, 파일 이름 등의 정보는 사용자가 임의로 변경이 가능한 정보이므로 고유 정보가 될 수 없다.

header와 tail에서 파일 작성자, 파일 이름 등의 정보를 제외한 부분과 body는 즉, 고유 정보가 아닌 부분은 결함이 주입(바이너리 코드가 변경)되어도 버퍼 오버플로우나 서비스 거부 공격과 같은 보안 문제를 유발하지 않는다. 즉, 고유 정보가 아닌 부분은 사용자가 임의로 변경을 할 수 있기 때문에 유동적인 바이너리 코드를 가지게 되므로 결함 주입이 불가능하다.

고유 정보를 추출하기 위해 (그림 6)과 같이 2대의 컴퓨터에서 다수의 파일을 생성한다. 생성한 파일을 비교하여 header와 tail 부분에서 사용자가 임의로 변경할 수 없는 고유 정보를 추출한다.

본 논문에서는 고유 정보를 추출하기 위해 <표 2>와 같이 12개의 파일을 생성하였다. 고유 정보를 추출하기 위해 생성한 12개의 파일은 MS 윈도우즈 기반 워드프로세서를 이용하여 문서를 작성하는 일반적인 방법을 고려하였으므로, 추출한 고유 정보에는 보안 문제를 일으키지 않는 부분도 포함되어 있다.

<표 2>의 파일들 중 기본 정보가 하나만 다른 파일 2개를 짝을 지어 비교하여 서로 다른 부분을 제외하면 고유 정보를 추출할 수 있다. 예를 들어, 파일 1과 파일 12의 서로 다른 부분은 파일을 작성한 파일 작성자 정보가 저장되는 부분이며, 파일 1과 파일 3의 서로 다른 부분은 파일 이름이 저장되는 부분이다.

4.3 입력 파일의 유효성 검사

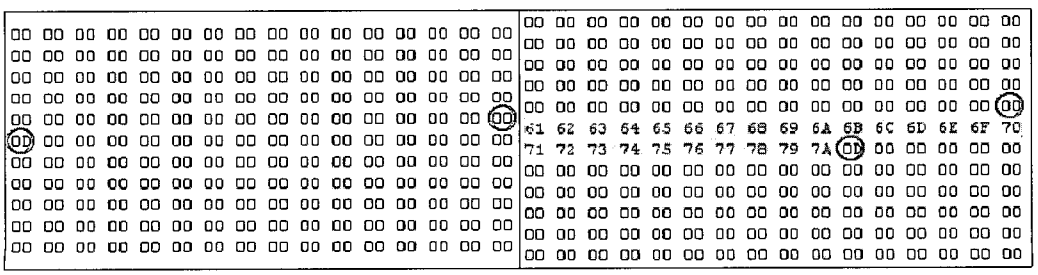
입력 파일의 유효성 검사는 추출한 고유 정보를 이용한다. 고유 정보는 파일의 특정한 위치에 특정한 값으로 존재하므로 파일에서 고유 정보가 저장되는 위치의 바이너리 코드를 비교하여, 같으면 정상적인 파일이고, 다르면 취약성이 내재된 비정상적인 파일이 된다. 비정상적인 파일로 분석되면 결함이 주입된 위치를 출력한다.

5. 입력 파일 구조 분석 및 유효성 검증 사례

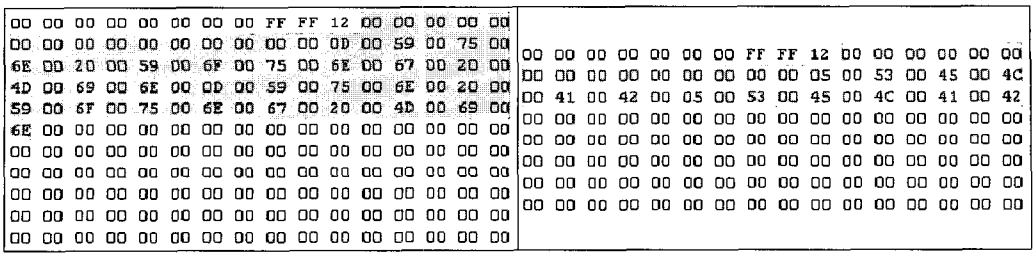
본 장에서는 DOC 파일에 결함을 주입하고 4장에서 제안한 입력 파일의 유효성 검사 모델을 이용하여 결함을 검출하는 사례를 보인다.

5.1 입력 파일 유효성 검사 모델을 이용한 DOC 파일에서의 취약성 검출

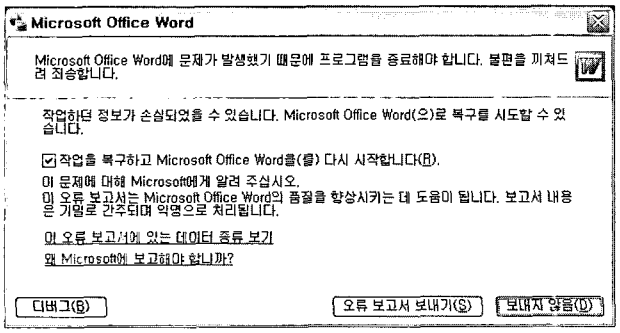
DOC 파일의 형식을 분석하기 위해 2개의 파일(본문의 내용이 있는 파일과 본문의 내용이 없는 파일)을 생성한다. (그림 7)의 왼쪽 파일은 본문의 내용이 없는 파일이고, (그림 7)의 오른쪽 파일은 본문의 내용이 있는 파일이다. (그림



(그림 7) DOC 파일 형식에서 본문이 저장되는 위치



(그림 8) 서로 다른 컴퓨터에서 생성한 DOC 파일



(그림 9) 서비스 거부 결함을 주입한 DOC 파일 실행 결과

7)와 같이 DOC 파일은 본문의 내용이 있는 파일의 경우 header와 tail 부분 사이에 body 부분이 존재하며, 본문의 내용이 없는 파일의 경우 body 부분이 존재하지 않는다.

고유 정보를 추출하기 위해 서로 다른 컴퓨터에서 <표 2>와 같이 다수의 파일을 생성한다. (그림 8)은 서로 다른 컴퓨터에서 작성한 2개의 파일(파일을 작성자가 다른 본문의 내용이 없는 파일)을 비교한 것으로, 파일 작성자에 대한 정보를 저장하는 부분이 서로 다른 것을 알 수 있다.

DOC 파일의 경우 파일 작성자, 파일 이름 등에 관한 정보가 저장되므로, header와 tail 부분에서 파일 작성자, 파일 이름 등의 정보가 저장된 부분을 제외한 부분이 버퍼 오버플로우나 서비스 거부 공격과 같은 보안 문제를 일으킬 여지가 있는 부분이 된다.

바이너리 편집기 또는 취약성 분석 도구인 Holodeck을 이용하여 정상적인 파일에 결함을 주입한다. (그림 9)는 결함이 주입된 파일을 실행한 결과이다.

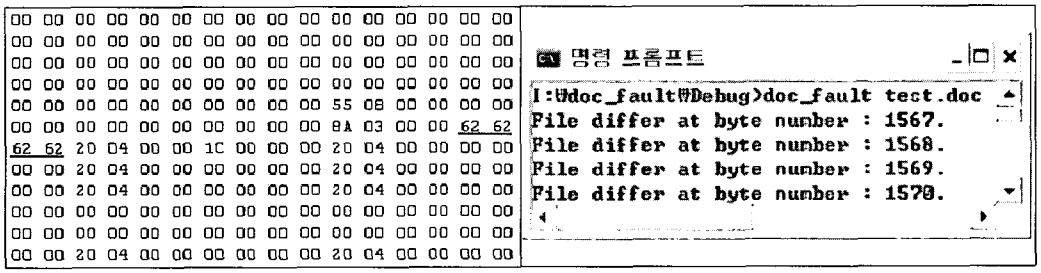
(그림 10)은 DOC 파일의 유효성을 검사하는 프로그램을 작성하여, 결함이 주입된 입력 파일을 테스팅하여 결함의 위치를 출력한 결과를 보여준다. 작성한 프로그램은 고유 정보 부분만을 비교하여 <표 1>과 같이 MS 워드의 입력 파일에 버퍼 오버플로우나 서비스 거부 등의 공격을 유발하는 결함이 주입되어 있다면 검출이 가능하다.

5.2 제안 기법의 분석

기존의 소프트웨어 결함 주입 도구들은 "어떤 결함을, 어디에, 어떻게 주입할 것인가?"하는 결함 주입 정책 및 방법에 관한 사항을 결정하는데 주안점을 두고 있다. 또한 취약성 분석 도구들은 시스템이 예기치 못한 상황에 대해 어느 정도 적절하게 대응하는지를 평가하는 강건성 테스트 도구와, 이미 알려진 취약성을 사용하여 시스템을 공격하고 이에 대한 시스템의 안정성을 평가하는 취약성 악용(exploit) 테스트 도구로 분류될 수 있다. 따라서 기존의 취약성 분석 도구들을 사용하여 (그림 1)과 같은 공격 유형의 결함을 생성하고 주입하는 것은 가능하지만 검출은 불가능하다.

본 논문에서는 결함이 주입되어 있는 입력 파일로 인해 MS 워드프로세서의 취약성이 악용되어 보안 문제가 발생할 수 있다는 것을 보이고, 그 취약성에 대한 공격을 미리 차단하는 방법을 제안하였다. 제안한 방법을 검증하기 위해, 결함을 주입할 위치를 정상적인 입력 파일에서 파악하고 분석한 다음, 그 위치에 결함을 주입하고 워드프로세서의 반응 결과(취약성 악용 결과)를 확인하였다. 또한, MS 워드용 결함 검출 프로그램을 작성하여, 결함이 있는 입력 파일이 워드프로세서에 의해 처리되기 전에 문제의 발생 가능성을 미연에 차단할 수 있도록 입력 파일에 존재하는 보안 결함을 검출하였다.

실험 분석 결과, DOC 파일은 (그림 5)와 같은 형식으로 header 및 tail 부분에서 파일 이름, 파일 작성자 등의 정보를 저장하는 부분을 제외한 나머지 부분이 결함 주입의 대상이 될 수 있다. DOC 파일의 경우 파일 형식이 매우 복잡하고, 공격을 유발하는 결함을 주입할 수 있는 부분도 광범위하여 DOC 파일에 포함된 결함을 검출하기 위해서는 많은



(그림 10) 결함이 주입된 DOC 파일에서 취약성 검출

경우의 수를 고려해야 한다. 즉, DOC 파일에 포함된 모든 결함을 검출할 수 있는 방법은 없어 보이나, 본 논문에서는 DOC 파일의 결함을 검출할 수 있는 가능성을 보였다는 점에서 연구의 의의가 있다.

6. 결론 및 향후 연구

본 논문에서는 보안 테스팅 중의 하나인 결함 주입 방법을 이용하여 MS 윈도우즈 기반 워드프로세서들의 입력 파일의 유효성을 검사하는 방법을 제안하였다. 제안한 방법은 공격에 악용될 수 있는 결함을 데이터 파일에 주입하고, 워드프로세서가 파일을 입력으로 사용하기 전에 데이터 파일의 유효성 검증을 통하여 보안 사고를 미연에 예방하는 것이다. 제안한 방법을 이용하여, 결함을 검출할 수 있는지 확인하기 위해 DOC 파일에 버퍼 오버플로우나 서비스 거부와 같은 공격을 유발하는 결함을 주입하고, 그 결함을 검출할 수 있는 프로그램을 개발함으로써, 본 논문의 유효성을 보였다.

본 연구의 결과는 소스 코드가 주어지지 않는 소프트웨어에 내재한 보안 취약성을 분석 및 검출할 수 있는 방법의 토대를 제공하는데 사용될 수 있다. 또한 소프트웨어 보안 취약성의 악용을 방지할 수 있는 기반 조성, 소프트웨어의 안정성을 평가하는 기준 마련, 보안 취약성 검출 방법의 향상 및 도구 개발 촉진에 기여할 수 있을 것이다.

참 고 문 헌

[1] http://www.cert.org.

[2] W. Arbaugh, W. Fithen and J. Mchugh, "Windows of vulnerability : A case study analysis," IEEE Computer, Vol.33, No.12, pp.52-59, 2000.

[3] P. Broadwell and E. Ong, "A comparison of static analysis and fault injection techniques for developing robust system services," Technical report, Computer Science Division, University of California, Berkeley, 2002.

[4] D. Peled, "Software reliability methods," Springer, 2001.

[5] E. Clarked and J. Wing, "Formal methods : State of the art and future directions," ACM Computing Surveys, Vol.28, No.4, pp.626-643, 1996.

[6] C. Pfleeger, S. Pfleeger and M. Theofanos, "A methodology for penetration testing," Computers and Security, Vol.8, No.2, pp.613-620, 1990.

[7] James A. Whittaker, "How to break software," Addison-Wesley, 2002.

[8] James A. Whittaker and Herbert H. Thompson, "How to break software security," Addison-Wesley, 2003.

[9] H. Thompson, J. Whittaker and F. Moatty "Software Security Vulnerability Testing In Hostile Environments," In Proceedings of the 17th ACM Software Applications Conference, pp.260-264, 2002.

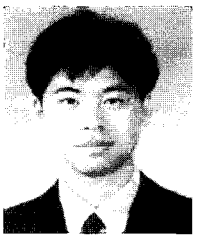
[10] http://cve.mitre.org.

[11] http://www.securityfocus.com.

[12] http://www.securityinnonation.com.

[13] http://www.immunitysec.com.

[14] http://www.coresecurity.com.



윤 영 민

e-mail : maverick@dankook.ac.kr

1999년 단국대학교 전산통계학과 졸업 (학사)

2001년 단국대학교 대학원 전산통계학과 (이학석사)

2004년 단국대학교 대학원 전산통계학과 (박사수료)

2001년~2007년 단국대학교 정보컴퓨터학부 컴퓨터과학전공 시간강사

2007년~현재 파수닷컴 선임컨설턴트

관심분야 : 컴퓨터 보안, 소프트웨어 취약성, 소프트웨어 테스팅, DRM 등

조 성 제

e-mail : sjcho@dankook.ac.kr

1989년 서울대학교 컴퓨터공학과(학사)

1991년 서울대학교 대학원 컴퓨터공학과 (공학석사)

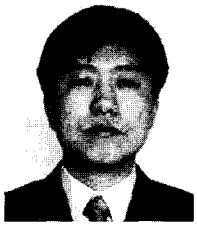
1996년 서울대학교 대학원 컴퓨터공학과 (공학박사)

1996년~1997년 서울대학교 컴퓨터신기술연구소 연구원

2001년~2002년 미국 University of California, Irvine 객원연구원

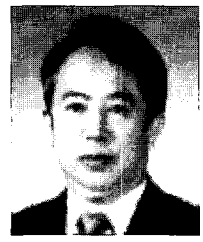
1997년~현재 단국대학교 정보컴퓨터학부 컴퓨터과학전공 부교수

관심분야 : 컴퓨터 보안, 시스템 소프트웨어, 실시간 시스템, 임베디드 시스템 등



최 종 천

e-mail : godofslp@dankook.ac.kr
1995년 강릉대학교 통계학과 졸업(학사)
2005년 단국대학교 대학원 컴퓨터과학 및 통계학과(이학석사)
1997년~현재 단국대학교 대학원 정보컴퓨터과학과 박사과정
관심분야 : 컴퓨터 보안, DRM, 소프트웨어 취약성, 임베디드 시스템 등



유 해 영

e-mail : yoohy@dankook.ac.kr
1979년 단국대학교 수학과 졸업(학사)
1982년 단국대학교 대학원 수학과 (이학석사)
1994년 아주대학교 대학원 컴퓨터공학과 (공학박사)
1983년~현재 단국대학교 정보컴퓨터학부 컴퓨터과학전공 교수
관심분야 : 소프트웨어 테스팅, 시스템 프로그래밍 등