KIPS Trans. Comp. and Comm. Sys.
Vol.8, No.5 pp.119~126 pISSN: 2287-5891

https://doi.org/10.3745/KTCCS.2019.8.5.119

A DOM-Based Fuzzing Method for Analyzing Seogwang Document Processing System in North Korea

Chanju Park† · Dongsu Kang††

ABSTRACT

Typical software developed and used by North Korea is Red Star and internal application software. However, most of the existing research on the North Korean software is the software installation method and general execution screen analysis. One of the ways to identify software vulnerabilities is file fuzzing, which is a typical method for identifying security vulnerabilities. In this paper, we use file fuzzing to analyze the security vulnerability of the software used in North Korea's Seogwang Document Processing System. At this time, we propose the analysis of open document text (ODT) file produced by Seogwang Document Processing System, extraction of node based on Document Object Mode (DOM) to determine test target, and generation of mutation file through insertion and substitution, this increases the number of crash detections at the same testing time.

Keywords : Fuzzing, Document Object Model(DOM), Open Document Text(ODT), Seogwang Document Processing System

북한 서광문서처리체계 분석을 위한 Document Object Model(DOM) 기반 퍼징 기법

박 찬 주† · 강 동 수††

요 약

자체 개발하여 사용하고 있는 대표적인 소프트웨어는 붉은별(Red Star)과 내부 응용 소프트웨어이다. 하지만 이러한 북한 소프트웨어에 대한 기존 연구는 소프트웨어 설치방법 및 일반적인 실행화면 분석이 대부분이다. 소프트웨어 보안 취약점을 확인하는 방법 중 하나인 파일 퍼징은 보안 취약점을 식별하는 대표적인 방법이며, 본 연구에서는 북한에서 개발하여 사용 중인 소프트웨어 중 서광문서처리체계에 대한 보안 취약점을 분석하기 위해 파일 퍼징을 사용한다. 이때 서광문서처리체계에서 생산되는 Open Document Text(ODT) 파일 분석 및 테스팅 대상을 정하기 위한 Document Object Model(DOM) 기반 노드 추출, 그리고 삽입과 대체를 통한 변이 파일 생성을 제안하며, 이를 통해 동일한 테스팅 시간에 크래시 발견 횟수를 증가시킨다.

키워드 : 퍼징, Document Object Model(DOM), Open Document Text(ODT), 서광문서처리체계

1. 서 론

북한은 6,800여 명의 사이버전 인력을 양성하는 등 열악한 환경을 군사적 우위로 전환하기 위해 사이버공격능력 강화에 힘쓰고 있으며[1-3], 이러한 강화된 사이버능력을 통해 2009년 7.7 DDoS 공격을 시작으로 2017년 워너크라이 공격 등의 사이버위협을 확대하였다. 북한은 여타의 재래식 무기와 핵무기 외에도 사이버공격과 같은 강력한 대남 도발 능력을 갖추고 있는 것으로 평가되며, 사이버 공격 전술은 핵실험과 달리 비가시적이기 때문에 북한 사이버 능력에 대한 정보 획득 및 분석이 지속적으로 필요하다[2].

북한은 소프트웨어 개발기관인 조선콤퓨터센터(KCC)를 설립하고 자체 소프트웨어를 개발하여 사용하고 있으며, 리눅스를 기반으로 개발한 붉은별(Red Star)을 개발하였다. 붉은별은 2006년에 개발을 시작하여 2012년에 2.0버전, 그리고 2014년에 3.0버전이 공개되었으며 4.0버전이 나온 것으로 추정된다. 2.0 버전부터는 내부 응용소프트웨어로 한컴 오피스의 한글과 비슷한 서광문서처리체계가 개발되어 사용되고 있다[4]. 북한이 자체 개발하였다고 알려진 붉은별(Red Star)과 내부 응용프로그램에 대한 연구는 소프트웨어 설치방법과 일반적인 실행화면 분석이 대부분이다.

† 준 회 원 : 해군사관학교 전산과학과 전산학교관
†† 종신회원 : 국방대학교 컴퓨터공학전공/사이버전과정 교수
Manuscript Received : November 26, 2018
First Revision : January 14, 2019
Accepted : January 28, 2019
* Corresponding Author : Dongsu Kang(greatkoko@kndu.ac.kr)

소프트웨어 보안 취약점을 확인하는 방법인 퍼징은 유효하지 않은 데이터를 대상 소프트웨어에 입력해 크래시를 발생시켜 취약점 유무를 확인하는 소프트웨어 보안 테스트 기법이다[5]. 이는 프로그래머가 예상할 수 없는 부분에서의 취약점을 탐지할 수 있는 장점이 있으며, 파일 구조 및 변이 방법 등 여러 가지 요소들에 따라 파일 퍼징의 효율성이 달라질 수 있다. 따라서 본 논문에서는 서광문서처리체계의 입력파일인 Open Document Text(ODT) 파일에 적용 가능한 Document Object Model(DOM) 구조 기반 퍼징을 제안하고 붉은별 3.0에 탑재된 서광문서처리체계를 실험한다.

본 논문의 구성은 2장에서 붉은별 운영체계 및 서광문서처리체계, DOM, 퍼징과 관련된 연구들을 설명하고, 3장에서 ODT 파일의 DOM 구조 기반 퍼징 기법을 제안한다. 4장에서는 적용사례 및 평가로, 제안 기법으로 구현한 결과를 비교분석하며, 5장에서 결론을 맺는다.

2. 관련 연구

2.1 서광문서처리체계 및 ODT 파일

서광문서처리체계는 붉은별(Red Star) 운영체계에서 구동되는 Office 프로그램이다. 서광 문서처리체계는 우리나라의 한컴 Office와 유사하며, 실제 실행화면은 Fig. 1과 같다. 서광문서처리 프로그램은 평양인쇄공업대학에서 개발한 Office 패키지 프로그램으로 한컴 Office 내에 한글, 한쇼, 한셀 등이 있는 것처럼 서광사무처리 3.0 내에도 각 업무들을 통합적으로 처리 및 관리하기 위하여 문서처리체계, 표처리체계, 연시체계로 구성되어 있다.

서광문서처리체계에서 사용하고 있는 확장자는 개방형 문서표준 포맷(Open Document Format for Office Application)을 사용하고 있다. 이 표준 포맷은 사용자가 다른 문서 작성 소프트웨어를 쓰더라도 공동으로 문서를 작성하고 열어볼 수 있도록 하기 위해 마련된 표준으로, Organization for the Advancement of Structured Information Standards (OASIS)에서 표준화하였으며, 2006년에 ISO/IEC 26300:2006으로 발표되었다[6].

Fig. 1. Red star's Seogwang Document Processing System 3.0

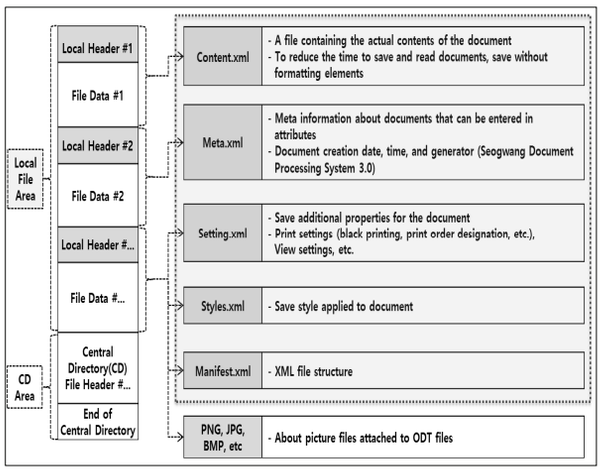


Fig. 2. ODT File Structure

ODT 파일의 구조는 ZIP 파일의 구조와 유사하다. Fig. 2에서 볼 수 있듯이, 파일은 크게 2개의 영역으로 구분할 수 있으며, 첫 번째 영역은 센트럴 디렉터리(Central Directory) 영역이고, 두 번째 영역은 로컬 파일(Local File) 영역이다. 로컬 파일 영역은 1개 이상의 로컬 파일로 XML 파일로 구성되어 있다[7].

이러한 XML 파일은 총 5개로, Content.xml은 문서의 실제 내용을 담고 있는 파일이며, 문서를 저장하고 여는 시간을 줄이기 위해 서식요소를 빼고 저장되며, Meta.xml은 속성에서 입력할 수 있는 문서에 관한 메타 정보(문서 생성/수정 날짜, 시간 및 서광문서처리체계 3.0)에 대한 정보가 담겨져 있고, Setting.xml은 문서에 대한 추가 속성(인쇄 설정, View 설정 등)이 저장되어 있다. 계속해서 Styles.xml은 Content.xml에 없었던, 문서에 적용된 스타일이 저장되어 있으며, 마지막 Manifest.xml은 파일들의 Full-Path와 같은 XML 구조에 대해 저장되어 있다[8].

2.2 Document Object Model(DOM)

XML은 IEEE에서 정의된 표준화된 문법에 따라 모든 데이터를 태그로 저장하여 사용되며, 텍스트를 추출하거나 태그를 분리해 내는 과정에서는 주로 Document Object Model(DOM)를 활용하여 문서를 접근한다. DOM은 XML 문서를 계층화된 구조로 메모리에 표현하여 문서에 포함된 정보를 공유할 수 있도록 하는 표준화된 방법이다. 즉, DOM은 외부 프로그램에서 XML 문서의 요소(Element), 속성(Attribute), 속성 값(Attribute Value), 텍스트 등을 추출하기 위한 접근 방법 설명과 인터페이스를 제공한다. DOM에서는 문서내의 모든 것이 노드(Node)라는 계층적 단위에 저장된다. 본 논문에서는 이러한 구조 중 일부를 변이시키더라도 소프트웨어가 파일을 입력받을시 영향을 받지 않을 것으로 판단되는 문서노드/요소노드/속성 노드와 같은 내부노드에 집중해서 퍼징을 수행한다[9].

2.3 퍼징(Fuzzing)

소프트웨어 보안 취약점을 미리 줄이는 방법 중 하나가 바

로 소프트웨어 보안 테스팅[10]이다. 퍼징은 블랙박스 보안 테스팅의 한 형태로 소프트웨어에 입력되는 값을 특정 규칙에 따라 변이·주입시켜 예상치 못한 값을 입력했을 때 발생하는 오작동을 탐지하고 취약점을 발견하는 것이다[11]. 소프트웨어에 변이시킨 데이터를 입력하고 그 결과로 소프트웨어 개발자가 예상치 못한 에러가 발생될 경우 보안 취약점이 존재할 가능성이 높다는 것이다.

퍼징의 종류에는 입력 데이터 주입 형태, 입력 데이터 처리 방법, 대상 소프트웨어 및 입력에 대한 분석 정도에 따라 Table 1과 같이 구분된다[12]. 특히, 퍼징 데이터 처리방법, 포맷 구조 이해 유무에 따라 대상 소프트웨어에 대한 취약점 발견의 효율성이 크게 달라질 수 있다. 데이터 주입 형태와 퍼즈 테스트의 데이터 처리 방법 측면에서 살펴보면, 일반적으로 서광문서처리체계와 같은 워드 프로세서는 해당 소프트웨어 자체 취약점도 있을 수 있으나, 소프트웨어에서 입력으로 사용하는 문서 파일에서 발생되는 경우가 많다.

특히 서광문서처리체계는 소스코드가 공개되어 있지 않기 때문에 화이트박스 테스트는 제한된다. 대신 입력으로 ODT라는 문서 파일의 구조를 분석할 수 있으며 이에 따라 서광문서처리체계 자체를 분석하는 것이 아닌 입력 파일을 분석, 변이, 주입하여 테스트를 수행함으로서 대상 소프트웨어 퍼징을 효율적으로 수행할 수 있을 것이다. 이때 어떤 규칙에 따라 여러 변이 기법이 있을 수 있으며, 대표적인 변이 기법으로 Swap 기법과 Bytemut 기법, Wave 기법이 있다[13].

Swap 기법은 Fig. 3과 같이 입력파일의 시작 바이트부터 마지막 바이트까지 인접 바이트와 값을 바꾸며 변이 파일을 생성한다. Bytemut 기법은 Fig. 4와 같이 입력 파일의 시작 바이트부터 마지막 바이트까지 1바이트씩 가능한 범위인 0~255의 값으로 변경하며 변이 파일을 생성한다. 마지막 Wave 기법은 Fig. 5와 같이 두 번째 바이트가 첫 번째 바이트로, 세 번째 바이트가 두 번째 바이트 순으로 N+1번 바이트가 N 바이트로 순차적으로 이동하면서 변이 파일을 생성한다. 이러한 기법은 입력 파일의 크기가 길면 길수록 테스트 케이스 생성에 소요되는 시간도 증가하는 단점이 있다.

Table 1. Types of Fuzzing

Type	Category	Contents
Data injection type	Simple	- Change the value the user enters on the keyboard
	File	- Change the file that the software enters as input
	Network	- Change the network data packet to send and receive
Fuzzing data processing method	Generation	- Create completely new input
	Mutation	- Change some data from existing input
Understand format structure	Dumb	- Random assignment to trigger vulnerability
	Smart	- Understanding the target and putting the appropriate input values

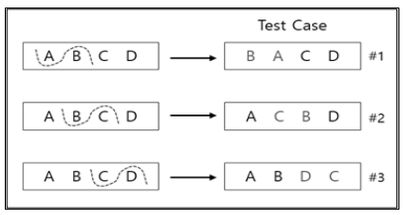


Fig. 3. Swap

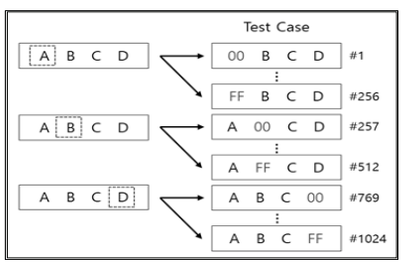


Fig. 4. Bytemut

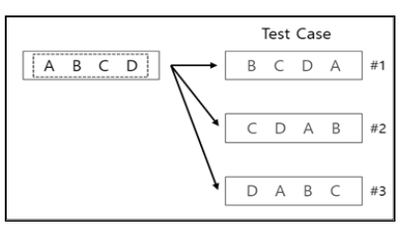


Fig 5. Wave

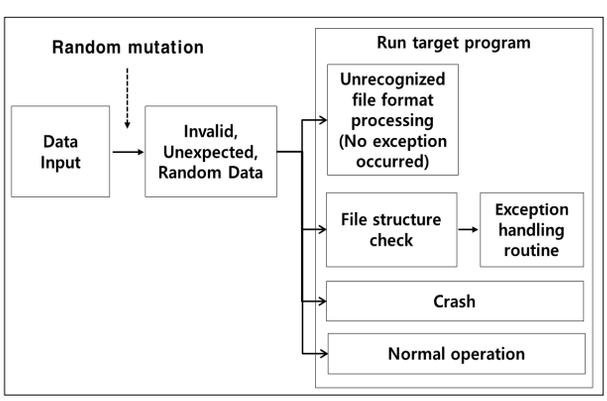


Fig. 6. Result of Execution of Target Program by Dump Fuzzing

두 번째, 포맷 구조 이해 측면에서 살펴보면 Fig. 6과 같이 파일 구조를 전혀 고려하지 않은 상태에서 무작위로 데이터를 생성하는 방법인 덤 퍼징의 경우, 한컴 Office나 서광문서처리체계와 같이 복잡한 구조를 가지고 있는 소프트웨어에서는 인식할 수 없는 파일 포맷으로 처리하기 때문에 예외가 발생되지 않을 가능성이 높다.

또한 인식할 수 없는 파일 포맷으로 처리되지 않더라도 파일 구조 검사에 의해 쉽게 지정된 예외처리 루틴으로 처리가 된다. 즉, 파일 구조가 깨지므로 취약점이 발견되지 않고 손상된 파일이라는 메시지 출력 가능성이 높다. 따라서 서광문서처리체계처럼 복잡한 구조의 소프트웨어를 대상으로 퍼징 수행 시, 대상을 이해하고, 유의미한 변이파일을 생성하여 입력값으로 넣어주는 방식이 무작위로 대입하여 취약점을 유발

시키는 방식보다 퍼징 효율성을 높일 수 있다.

이러한 방향으로 퍼징의 효율성을 높이는 연구가 진행된 사례가 있다. 먼저 멀티미디어 분야에서 2007년 버클리 대학은 3가지 방식(무작위 데이터인 Random file, 무작위 데이터에 파일 구조를 적용한 Structured random file, 기존 실제 파일을 변이한 Randomized real file)으로 실험하였으며[14], 2009년 한국 인터넷 진흥원(Korea Internet & Security Agency)은 버클리대학의 3가지 방식과 더불어 파일 구조를 파악 후에 원하는 특정 영역을 변이시키는 구조적 변이를 추가하여 실험하였다[15]. 2017년 국방대에서는 버클리대학과 한국 인터넷 진흥원에서의 결과를 보완하기 위해 파일구조를 파악 후 기존 취약점을 통해 선정된 취약영역을 무작위 변이 방법을 제안하였다[16].

다음으로 워드프로세서 분야에서는 2014년 성균관대에서 한컴 Office에 대해 분석하였다. 한글문서는 한컴 Office에서 지정한 데이터 저장 표현방법인 레코드 구조로 연속적으로 저장이 되는데, 이러한 레코드 구조를 깨지 않으면서 퍼징을 수행하기 위해 2가지 방식(레코드 위치변경, 레코드 내 데이터 위치변경)을 통해 변이파일을 생성하였다[17].

위 연구들을 종합해보면, 공통적으로 임의 데이터로 테스트 케이스를 생성할 경우는 소프트웨어에서 인식이 불가능하였다. 또한 기존 파일을 무작위로 변이시켰을 경우에는 많은 취약점을 발견할 수 있었지만 치명적인 취약점을 발견하기는 어려웠다. 따라서 서광문서처리체계의 퍼징 효율성을 높이기 위해서 ODT 파일 구조를 분석하여 DOM 구조화 후 노드라는 특정 영역을 식별하고, 두 가지의 변이 방법을 사용한다.

3. ODT 파일 퍼징 기법

서광문서처리체계와 같은 워드프로세서의 취약점은 해당 소프트웨어 자체 취약점도 있을 수 있지만 대개의 경우, 소프트웨어에서 입력으로 사용하는 문서 파일, 즉 ODT나 HWP 등의 파일에서 발생되는 경우가 많다. 그래서 파일 퍼징을 사용하여 취약점을 찾고자 하는데, 기존의 일반적인 파일 퍼징 기법은 파일의 오프셋을 처음부터 마지막까지 Bytemut나 SWAP 방식 등을 사용하여 변이한다. 그렇기 때문에 수많은 테스트 케이스가 만들어지면서 반복(Iteration) 횟수가 많아지며, 이로 인해 크래시가 발생했을 경우 파일의 어느 위치에서 크래시가 생겼는지 확인하기가 어렵고 시간이 오래 걸리게 된다. 또한 기존 각각의 변이 기법에 따라 변이되는 영역 크기, 변이 값 등이 모두 다르기 때문에 발생시킬 수 있는 크래시도 다르다. 이를 해결하기 위해 ODT 파일의 파일 구조를 확인하고 특정 영역을 선택한 뒤 크래시를 발생시킬 확률이 높도록 2가지의 변이 방법을 사용하여 퍼징 기법을 설계한다. 제안하는 ODT 파일 퍼징은 3단계이며, Fig. 7과 같다.

전체적인 프로세스는 변이기반 파일 퍼징 기법을 수행한다. 먼저, 테스팅 영역 추출 준비 과정이다. 이 과정에서는 대상 소프트웨어를 확인하고 변이를 일으킬 여러 개의 입력 파일을 수집하며, 반복(Iteration)을 1번 돌기 위해서 1개의 입력

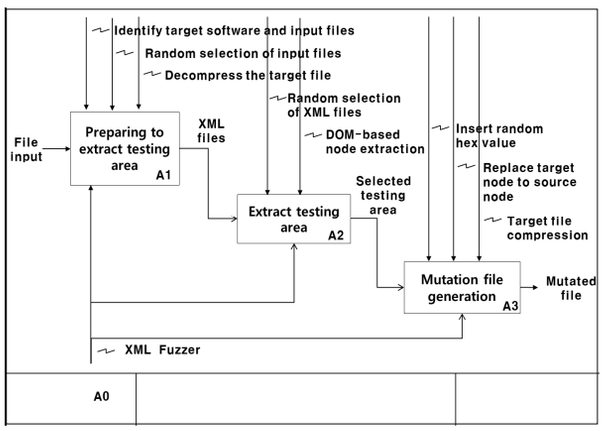


Fig. 7. Proposed ODT File Fuzzing Process

파일을 무작위 선택한다. ODT 파일은 5개의 XML 파일과 문서 내 저장되는 그림 파일(PNG, JPG 등)이 ZIP 파일 구조로 압축된 파일이므로, XML 파일을 추출하기 위해서 ZIP 파일 구조의 ODT 파일을 압축해제해서 XML 파일들을 추출한다.

두 번째 단계는 테스팅 영역 추출 단계로 이전 단계에서 추출된 XML 파일들 중 하나를 무작위 선택한다. 선택된 XML 파일에서 DOM 구조의 노드를 선택한다. 이는 기존의 바이너리 수준에서 이루어지는 기법과는 달리 불필요한 테스트 케이스 생성을 방지하여 테스팅의 효율성을 높이는 핵심적인 역할을 수행한다.

세 번째, 변이파일 생성단계에서는 2단계에서 추출된 노드 뒤에 무작위 Hex 값을 삽입하고, 다른 노드 위치로 대체시킨 후 ODT 파일로 재압축시켜서 변이파일을 만든다. 이러한 변이 기법은 대상 워드프로세스가 순차적으로 파싱하여 문서 데이터를 읽어들일 때, 정상적인 노드가 아닌 비정상 노드를 파싱함으로써 예상치 못한 예외사항을 발생시킬 수 있다.

3.1 테스팅 영역 추출 준비

테스팅 영역 추출 준비 단계는 변이파일을 생성하기 위한 가장 기초적인 단계로, 서광문서처리체계와 같은 대상 소프트웨어의 입력파일, 즉 ODT 파일 수집 및 이러한 ODT 파일을 압축해제 후 XML 파일들로 만들어주는 단계이다. 세부적

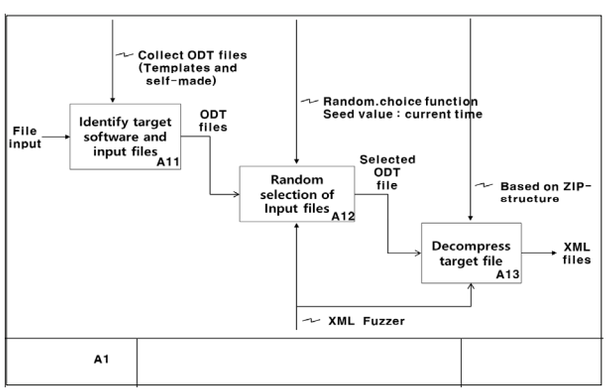


Fig. 8. Preparing to Extract Testing Area

인 과정은 Fig. 8과 같이 대상 소프트웨어 확인, 입력 파일 무작위 선택, 대상 ODT 파일 압축해제 순이다.

이때, 수집하는 파일은 서광문서처리체계에서 기본적으로 제공하는 템플릿 파일 78개가 있으며, 추가적으로 퍼징 테스트를 수행하는 테스터가 그림 및 표, 수식, 스타일 등을 추가하여 자체 제작하거나, 인터넷 상의 ODT 파일들을 모아서 테스팅에 활용한다. 실제 파일을 변이시키기 위하여 다양한 형식과 크기의 입력 파일을 수집할 필요가 있다.

3.2 테스팅 영역 추출

테스팅 영역 추출 단계는 전 단계인 테스팅 영역 추출 준비 단계에서 만들어진 XML 파일들 중 하나를 무작위 선택한 후, DOM 구조의 노드를 추출하기 위한 단계로, Fig. 9와 같은 단계로 진행된다.

하나의 ODT 파일을 압축 해제할 경우 총 5개의 XML 파일을 확인할 수 있으며, 이 중 하나의 파일을 무작위 선택한 후 DOM 구조의 노드를 추출한다. 이때 노드는 루트노드, 요소노드, 속성노드만을 추출하며 이러한 과정을 거쳐 선택된 영역은 변이의 대상이 되고 세 번째 단계인 변이 파일 생성 과정을 통해 테스트 케이스로 생성이 된다. 이때 선택된 영역은 DOM 구조에 기반 하여 기존 파일 전체를 대상으로 테스팅하는 것보다 반복(Iteration) 횟수가 감소한다.

3.3 변이 파일 생성

3번째 단계인 변이 파일 생성 단계는, DOM 기반 노드들의 테스팅 대상 영역을 추출하여 테스팅 반복(Iteration)을 줄이고 효율성을 증대시키는 1단계 테스팅 영역 추출 준비와 2단계

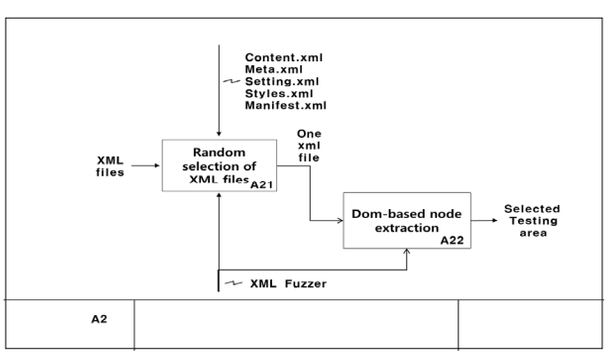


Fig. 9. Extract Testing Area

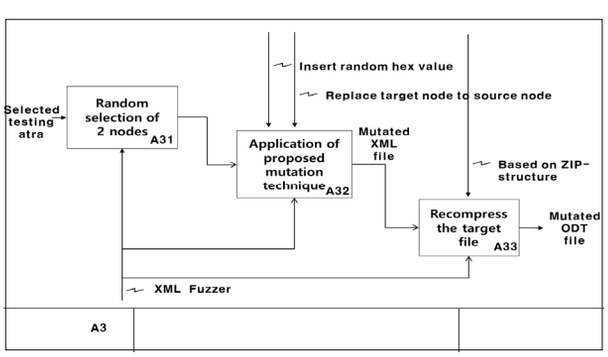


Fig. 10. Mutation of the Area to be Tested

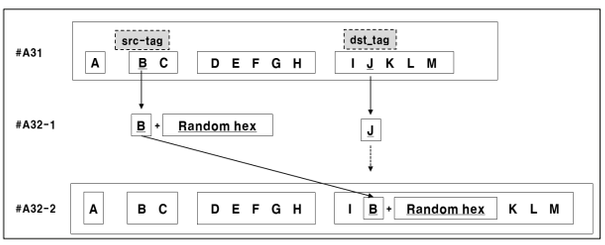


Fig. 11. Detailed Mutation of A31 and A32

테스팅 영역 추출 단계와는 다르게, 테스팅 대상 영역에 대해 기존 2가지 퍼징 방법을 통해 테스팅 정확성과 효과성을 높이는 것이 목적이다. 2가지 단계를 통해 선택된 퍼징 영역에 대하여 삽입(Insert)과 대체(Replace)를 사용하여 보다 정밀한 취약점 식별이 가능하게 한다. Fig. 9는 변이 파일 생성을 위해 본 논문에서 제안하는 변이 방법을 나타내며, Fig. 10은 Fig. 9의 A21와 A22의 세부 변이 방법이다.

먼저 2단계를 거쳐 추출된 노드들 중에 2개의 노드(Source_tag, Destination_tag)를 무작위 선택하고, src-tag 뒤에 무작위 Hex 값을 삽입한다. 그 후 dst_tag 위치에 stc-tag + hex value를 넣어 대체한다. 이는 서광문서처리체계에서 XML 파일 정보를 순차적으로 파싱하여 데이터를 읽어올 때 바뀐 src-tag + hex 위치에서 원래의 노드가 아닌 다른 노드를 파싱함으로써 예상치 못한 예외사항을 발생시킬 수 있다.

4. 적용 사례 및 평가

본 장에서는 제안한 퍼징 기법을 적용하고 평가하기 위해 붉은별 운영체계에 퍼징 도구인 Basic Fuzzing Framework(BFF)와 제안된 기법으로 작성된 퍼징 도구를 설치하고 대상파일 및 서광문서처리체계를 사용한다.

4.1 실험 및 평가

실험에 사용한 컴퓨터의 사양과 대상 소프트웨어, 자동화 도구는 Table 2와 같다. Windows를 사용하는 개인용 컴퓨터를 사용하고, 대상 소프트웨어를 Windows 기반에서 사용하기 위해 VMware Workstation 14 player를 사용하였다. 기존 일반적인 방식으로 퍼징하기 위해 Carnegie Mellon 대학 CERT의 Basic Fuzzing Framework(BFF) Ver 2.5를 선정하였다. 실험 환경 구축 후, 이어서 제안한 변이 기법을 ODT 파일에 적용 및 퍼징을 수행하였으며 발생한 크래시를 확인한다. 이후 기존 기법 2가지와 제안기법에 대하여 시간에 따른 반복(Iteration)과 크래시 수를 비교한다.

Fig. 12는 3장에서 제안한 방법을 구현하기 위해 Python 2.7을 이용하여 작성된 퍼징 도구의 프로세스이다.

해당 퍼징 도구를 실행하면 ① 해당 퍼징 도구는 먼저 ODT 파일을 무작위 선택하여 Target_seed_file에 입력하고, ② Target_seed_file의 ODT 파일을 압축해제(Decompress)한다. ③ 압축해제 한 ODT 파일 내 XML 파일 중 하나를

Target_xml로 지정하고 ④ 3장의 A32단계 기법으로 XML 파일을 변이시킨다. ⑤ 이후 다시 ODT 파일로 재압축 하여 해당 소프트웨어에 주입한다. ⑥ 주입 후 해당 프로세스에 예상치 못한 이벤트가 발생시 Crash로 나타내고 해당 변이된 파일은 따로 저장한다. ⑦ 변이된 파일이 정상적으로 열리거나, 예외처리 루틴 등으로 서광문서처리체계에서 에러 메시지를 나타내 줄 경우 해당 프로세스는 강제 종료되고 다시 ①번으로 돌아가서 위 과정을 반복 수행한다.

실제 공개된 기존 취약점인 CVE-2012-2665은 인증되지 않은 원격 공격자가 잘못된 부모 노드 내의 자식 노드 또는 중복 노드로 이루어진 조작된 ODT 파일을 열도록 유도하여 취약점을 유발한다. 실제 크래시(Crash)가 난 파일을 열어보면, Fig. 13에서 볼 수 있듯이 <office:automatic-styles> 태그가 <office:automatic-stylesxA3xDC#1xDENAKc>로 변

Table 2. Experiment Environment

Division		Contents
Equipment	CPU	Intel(R) CoreTM i7-4700HQ CPU @ 2.4GHz
	RAM	8.00GB
	SSD	256GB
	OS	Windows 10 Home
Target SW	Seogwang	Ver 3.0
Tool	VMware	Workstation 14 Player Memory: 2GB Storage capacity: 20GB Processors: 2 OS: Red Star 3.0
	BFF	Ver 2.5
	Python	Ver 2.7
	HxD	Ver 1.7.7
	Notepad++	Ver 7.5.8

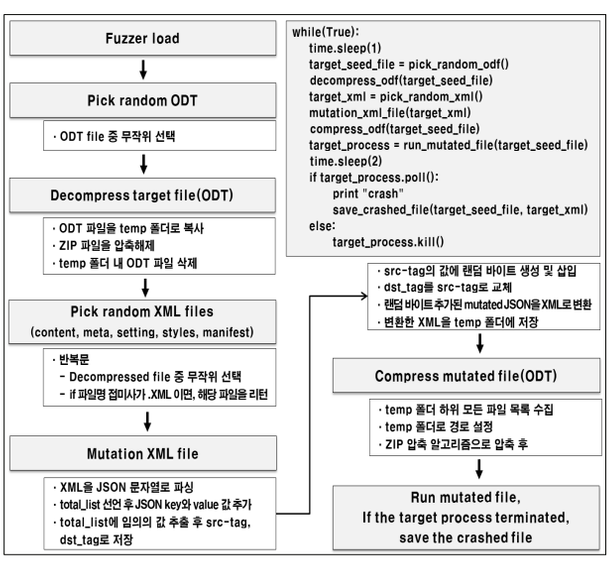


Fig. 12. The Basic Process of a Fuzzing Tool Written in Python

```
    svg:font-family="'KP CheonRiMa'" style:font-family-generic="system"/>
    <style:font-face style:font-pitch="variable" style:name="Luxi Sans"
    svg:font-family="'Luxi Sans'" style:font-family-generic="system"/>
</office:font-face-decls>
<office:automatic-stylesxA3xDC#a1xDENAKc>
    <style:page-layout style:name="pm1">
        <style:footer-style>
            <style:header-footer-properties fo:margin-left="0cm" fo:margin-right
            ="0cm" fo:min-height="0.75cm" fo:margin-top="0.25cm"/>
        </style:footer-style>
        <style:header-style>
            <style:header-footer-properties fo:margin-left="0cm" fo:margin-right
            ="0cm" fo:min-height="0.499cm" fo:margin-bottom="0cm"/>
```

Fig 13. The Mutated Part of the Styles.xml, where the Crash is a Mutation File

경된 것을 확인할 수 있다. 이러한 변이된 파일이 처리될 때 힙 기반 버퍼 오버 플로우(Heap-Based Buffer Overflow)가 발생할 수 있으며, 공격자는 이 메모리 손상(Memory Corruption)을 이용하여 해당 소프트웨어가 예기치 않게 종료되거나 시스템에서 임의의 코드가 실행되고 서비스 거부(Denial of Service, DoS)공격을 유발할 수 있다[18, 19].

4.2 기존 방법과의 비교

본 논문에서 제안하는 변이 파일 생성 기법을 입증하기 위해 기존의 변이 파일 생성 방법 2가지와 반복(Iteration)과 크래시(Crash) 횟수를 비교 분석하였다. 파일 크기가 같다면 Wave 기법은 Test case 생성 수가 Swap과 비교하여 동일하므로, Test case 수가 서로 같아 Wave 기법과의 비교는 생략한다. 기존 변이 파일 생성 기법 2가지는 실제 파일의 영역 전체를 대상으로 Bytemut와 Swap 기법을 사용한다. 제안 기법은 실제 파일의 DOM 내부노드만을 대상으로 Hex 값 삽입(Insert)와 대체(Replace)를 순차적으로 적용한다. 실제파일은 서광문서처리체계에서 기본 제공하는 78개의 템플릿 파일과 링크된 이미지와 표, 여러 글자속성들을 가진 자체 제작한 파일이다.

84시간 기준으로 테스팅 반복 횟수를 비교한 결과, Table 3과 Fig. 13에서와 같이 Swap과 Bytemut 대비 각각 51,516번 그리고 54,996번 감소하였으며, 이는 각각 74.4%와 75.7%가 감소하였음을 알 수 있다.

또한 Table 4와 Fig. 15에서 볼 수 있듯이, 84시간 기준으로 비교한 결과 크래시 발견 횟수는 기존 기법에 비해 20개, 14개가 증가되었다. 이것은 기존 방법인 Bytemut와 제안 기법을 비교하면, 제안 기법은 변이파일을 54,996개를 적게 만들고도 크래시(Crash) 발견 횟수는 14개로 더 많은 크래시를 발생시켰으며, Swap 방식과 비교를 하면 제안기법은 변이파일을 51,556개를 적게 생성하면서 크래시(Crash) 발견 횟수는 20개 발견으로, 두 경우 모두 유사한 결과를 확인할 수 있다. 즉, 위 동일 시간대에 제안기법은 기존 2가지 기법보다 크래시 수의 발견 확률이 높다고 말할 수 있다.

이를 통해 ODT 파일을 입력값으로 하는 서광문서처리체계의 보안 취약점을 확인할 시 제안기법으로 수행 시 변이파일을 적게 생성하더라도 즉, 보안취약점을 확인하려는 테스터가 적은 노력으로도 기존 기법 대비 더 많은 크래시(Crash)를 발견할 수 있으며, 이 결과로 볼 때 퍼징에 대한 효율성을 증가시킬 수 있다.

Table 3. Iteration Cumulative Counts for Three Methods

Time (Hour)	Swap	Byte mut	Proposed Method	Compare	
				Swap	Bytemut
12	8357	8956	3110	(-) 5247 (62.8%)	(-) 5846 (65.3%)
24	17319	18007	6210	(-) 11109 (64.1%)	(-) 11797 (65.5.%)
36	25995	27333	8521	(-) 17474 (67.2%)	(-) 18812 (68.8%)
48	35752	38167	10812	(-) 24940 (79.8%)	(-) 27355 (71.7.%)
60	45771	48996	13792	(-) 31979 (69.9%)	(-) 35204 (71.9%)
72	55656	59772	15752	(-) 39904 (71.7%)	(-) 44020 (73.6%)
84	69198	72648	17682	(-) 51516 (74.4%)	(-) 54966 (75.7%)

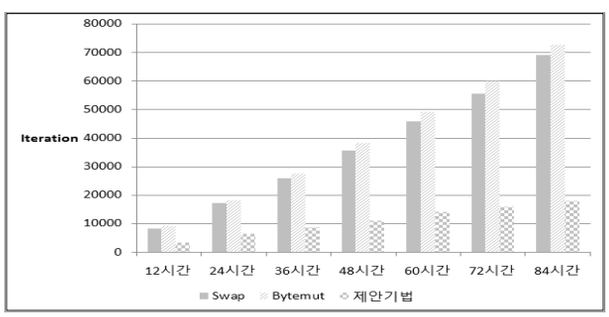


Fig. 14. Iteration Cumulative Comparisons in Three Methods

Table 4. Crash Cumulative Counts for Three Methods

Time (Hour)	Swap	Byte mut	Proposed Method	Compare	
				Swap	Bytemut
12	0	0	4	(+) 4	(+) 4
24	0	2	7	(+) 7	(+) 5
36	1	3	9	(+) 8	(+) 6
48	1	5	15	(+) 14	(+) 10
60	3	6	21	(+) 18	(+) 15
72	3	8	23	(+) 20	(+) 15
84	3	9	23	(+) 20	(+) 14

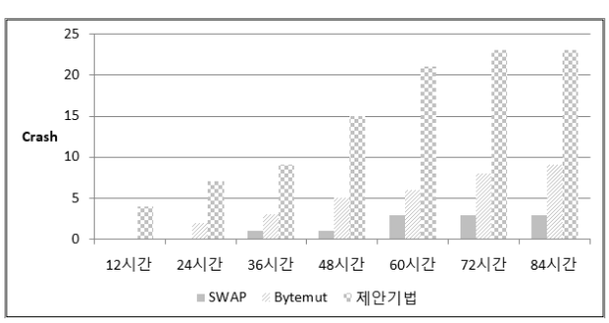


Fig. 15. Crash Cumulative Comparisons in Three Methods

5. 결 론

본 논문에서는 ODT 파일을 입력파일로 가지는 북한 서광문서처리체계에 대한 퍼즈 테스트를 위해 DOM 구조 기반의 퍼즈 테스트 기법을 제안하고, 적용 가능성 분석과 기존 기법과의 비교를 통해 평가하였다. 제안한 ODT 변이파일 생성은 3단계로 구성되어 있다. 먼저 파일의 구조를 파악하여 XML 파일들로 압축해제하고, 테스팅 효율을 위해 DOM 기반으로 특정 영역을 선택한 뒤 크래시를 발생시킬 확률이 높도록 삽입과 대체라는 기존 변이 방법 두 가지를 사용하여 변이 파일을 생성한다.

새로운 퍼징 기법을 수행하기 위한 도구를 제작하였으며, 제안한 퍼징 기법을 증명하기 위해 기존의 대표적인 퍼징 기법인 Swap, Bytemut와 비교하였다. 실험 대상 소프트웨어는 북한의 서광문서처리체계이고 입력파일은 ODT 파일이며, 제안 기법의 단계에 따라 ODT 파일들을 XML DOM 노드로 추출하여 이러한 노드를 바탕으로 무작위 Hex 값 삽입과 노드 대체를 통해 퍼징을 진행하였다.

그 결과 기존 파일 퍼즈 테스트 기반 기법과 비교하여 같은 시간대를 살펴보면, 반복 횟수는 각각 74.4%와 75.7%가 감소하였으며, 크래시 발견 횟수는 각각 20개, 14개가 증가하였다. 이를 통해 제안된 기법은 퍼즈 테스트 수행 시 적은 노력으로 더 많은 크래시(Crash)를 확인할 수 있었다. 향후 제안 기법의 구현을 통해 발견한 크래시를 분석한 후 서광문서처리체계의 실제 취약점을 발견하는 연구를 통해 제안 기법을 보완해 나갈 예정이며, 실제 취약점을 발견할시 유사 소프트웨어의 보안 취약점에 대한 공격을 방어하는데 도움이 될 것으로 보인다. 또한 연구에서 제안된 도구는 ODT 파일을 지원하는 한컴 Office와 MS Word에도 동일하게 테스트 가능할 것으로 기대하고, 알려진 취약점을 판단할 수 있는 취약점 스캐너로서 활용이 될 것으로 기대한다.

References

[1] Ministry of National Defense, “2016 Defense white paper,” pp.20-25, 2016.

[2] Guyeon Jeong and Gitae Lee, “Science technology development and new threats from North Korea : Cyber threat and UAV Penetration,” *KINU Research Series 16-04*, pp.69-72, 2016.

[3] Kihun Park and Dongsu Kang, “A security vulnerability analysis of North Korea OS Red Star,” in *Proceedings of Korea Software Congress*, pp.146-148, 2017.

[4] Jongseon Kim and Lee Choongeun, “Analysis and cooperation of North Korea's IT technology in uniform preparation,” *Science and Technology Policy Institute*, 2014.

[5] P. Oehlert, “Violating assumptions with fuzzing,” in *Proc. the IEEE Security & Privacy(S&P)*, Vol.3, No.2, pp.58-62, 2005.

[6] ISO/IEC 26300:2006 Information technology - Open Document Format for Office Applications [Internet], https://www.iso.

org/standard/43485.html.

[7] Byungjoon Jung, Jaehyeok Han, and Sangjin Lee, “A method of recovery for damaged ZIP files,” *Journal of The Korea Institute of Information Security & Crypto logy*, Vol.27, No.5, pp.1099-1106, 2017.

[8] Chanju Park and Dongsu Kang, “Analysis of file structure about Red Star's SeoKwang Document Processing System for security vulnerability analysis,” in *Proceedings of the Korea Information Processing Society*, Vol.25, No.1, pp.110-112, 2018.

[9] G. Wang, “Improving data transmission in web applications via the translation between xml and json,” *Communications and Mobile Computing(CMC) 2011 Third International Conference*, pp.182-185, 2011.

[10] R.shirey, “RFC 2828-Internet Security Glossary,” 2007.

[11] Sangsu Kim and Dongsu Kang, “Software Vulnerability Analysis using File Fuzzing,” in *Proceedings of the Korean Society of Computer Information Conference*, Vol.25, No.2, pp.29-32, 2017.

[12] Michael Sutton, “FUZZING: Brute Force Vulnerability Discovery,” United States of America: Addison-Wesley, 2007.

[13] Jaeseo Lee, Jongmyung Kim, Suyong Kim, Youngtae Yun, Yongmin Kim and Bongnam Noh, “A length-based file fuzzing test suite reduction algorithm for evaluation of software vulnerability,” *Journal of the Korea Institute of Information Security & Cryptology*, Vol.23, No.2, pp.231-242, 2013.

[14] Colleen Lewis, Barret Rhoden and Cynthia Sturton, “Using structured random data to precisely fuzz media players,” *Project Report*, 2007.

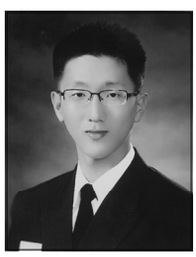
[15] Hanyang University, “Study on systematic approach for finding vulnerabilities in multimedia data and players for Microsoft Windows systems,” *KISA*, 2009.

[16] Sangsu Kim and Dongsu Kang, “Fuzzing-based test case generation technique for multimedia file vulnerability analysis,” *Journal of Security Engineering*, Vol.14, No.6, pp.441-458, 2017.

[17] Sunghwan Ahn, “A novel fuzzing approach for discovering potential vulnerabilities in Hangul Word Processor,” Thesis, Sungkyunkwan University, Seoul, Korea, 2014.

[18] CVE Details, Vulnerability Details : CVE-2012-2665 [Internet], https://www.cvedetails.com/cve/CVE-2012-2665.

[19] CISCO, Multiple Products XML Manifest Encryption Handling Arbitrary Code Execution Vulnerability [Internet], https://tools.cisco.com/security/center/viewAlert.x?alertId=26540.



박 찬 주

https://orcid.org/0000-0002-2231-7469

e-mail : myvexation@hanmail.net

2009년 해군사관학교 전산과학과(학사)

2019년 국방대학교 컴퓨터공학전공(석사)

2019년～현 재 해군사관학교 전산과학과 전산학교관

관심분야 : SW Engineering, Software Security Testing

강 동 수

https://orcid.org/0000-0001-6481-5071

e-mail : greatkoko@kndu.ac.kr

2011년 고려대학교 컴퓨터공학과(박사)

2015년～현 재 국방대학교 컴퓨터공학전공/사이버전과정 부교수

관심분야 : Weapon System Software, Software Security Testing, Defense Acquisition