석사학위논문

파라미터 탐지와 뮤테이션을 이용한 웹 브라우저 퍼징 도구

제주대학교 교육대학원

컴퓨터교육 전공

안 지 민

2011년 8월

파라미터 탐지와 뮤테이션을 이용한 웹 브라우저 퍼징 도구

지도교수 김 한 일

안 지 민

이 논문을 교육학 석사학위 논문으로 제출함

2011년 6월

안지민의 교육학 석사학위 논문을 인준함

심사위원장 김 철 민 ㊞

위 원 김 성 백 ㊞

위 원 김 한 일 ㊞

제주대학교 교육대학원

2011년 7월

<국문초록>

파라미터 탐지와 뮤테이션을 이용한 웹 브라우저 퍼징 도구

안 지 민

제주대학교 교육대학원 컴퓨터교육 전공

지도교수 김 한 일

인터넷 기술의 발전으로 인해 컴퓨터와 인터넷 사용이 일반화되었으며 기술의 고도화로 결제 서비스, 금융 거래 등 정보보안을 요구하는 서비스도 활발히 이용되고 있다. 하지만 인터넷 이용률과 의존도가 높아짐에 따라 역기능 또한 다양해져서 해킹피해, 개인정보유출, DDoS 등과 같은 보안 사고가 연일 끊이지 않는다.

컴퓨터 환경에서 발생하는 보안 사고는 바이러스, 웜, 스파이웨어와 같은 악성코드들로 인해 발생하는 경우가 많으며, 홈페이지나 소프트웨어 취약점은 악성코드의 전파경로로 이용될 수 있다. 소프트웨어에 취약점이 존재할 경우 악의적인 사용자는 이를 이용하여 허용된 권한 이상의 동작을 수행하고 중요한 정보를 열람하여 개인정보를 유출하거나 심각한 보안 사고를 일으킬 수 있다. 따라서 소프트웨어의 취약점을 신속히 파악하여 제거하는 것이 보안 사고를 줄일 수 있는 근본적인 방안 중 하나이다.

소프트웨어 취약점 분석 방법은 화이트 박스 테스트, 블랙 박스 테스트 등이 있다. 이 중 블랙 박스 테스트는 취약점 점검 대상 소프트웨어에 데이터를 입력하여 얻어지는 결과를 바탕으로 테스트를 진행하는 방식으로, 전문적인 지식이나 내부

※ 본 논문은 2011년 8월 제주대학교 교육대학원 위원회에 제출된 교육학 석사 논문임

로직에 대한 이해 없이도 비교적 쉽게 테스트를 수행할 수 있다는 장점을 가진다.

본 연구에서는 블랙박스 테스트 기법의 하나로 소프트웨어에 비정상적인 값을 주입하고 모니터링하여 소프트웨어의 결함을 발견하는 퍼징 기법을 이용하여 웹 브라우저의 취약점을 탐지하는 퍼징 도구를 설계, 구현하였다.

제안된 퍼징 도구는 웹 페이지를 수집하여 템플릿을 구성하고, 템플릿의 소스코드를 분석하여 파라미터를 탐지한다. 파라미터 부분에 퍼징 오라클에 정의된 비정상 값을 입력하여 테스트케이스를 생성하며, 테스트 케이스를 브라우저에 주입한 후 그 실행결과를 모니터링하여 브라우저 취약점 존재 여부를 판단한다.

이 도구는 특정 웹 언어에 대한 의존도를 낮추어 다양한 언어와 버전에 대해 퍼징을 수행할 수 있으며, 웹 페이지에 포함된 플러그인에 대해서도 취약점 존재 여부를 점검할 수 있다는 장점을 지닌다.

웹 브라우저는 웹 환경에서 가장 많이 사용하는 애플리케이션 중 하나로 취약점이 악용될 경우 그 영향도 매우 크기 때문에 본 퍼징 도구가 취약점 탐지 분야에서 광범위하게 이용될 수 있을 것으로 기대한다.

<차 례>

<표 차례>

<그림 차례>

Ⅰ. 서론

1. 연구의 배경 및 필요성

인터넷 기술의 발전과 사용의 보편화로 인해 컴퓨터와 인터넷은 우리의 생활 전반에 걸쳐 널리 이용되고 있다. 우리나라의 인터넷 이용현황을 살펴보면 2010년 인터넷 이용자가 3,701만 명으로 전체 국민의 77.8%에 이르고 있으며, 10대~30대의 인터넷 이용률은 99%, 40대의 인터넷 이용률도 87.3%에 달한다.[1] 인터넷 이용률이 높아짐에 따라 인터넷을 통해서 제공되는 서비스도 매우 다양해져서 메일, 멀티미디어, 블로그와 같은 정보공유, 커뮤니케이션의 용도뿐 아니라 결제 서비스, 금융거래 등 고도의 정보보안을 요구하는 서비스도 인터넷 상에서 활발하게 이용되고 있는 실정이다.

하지만, 일상생활에서 정보통신에 대한 의존도가 높아짐에 따라 그 역기능 또한 다양해졌으며 연일 DDoS, 해킹피해, 개인정보유출과 같은 보안 사고가 끊이지 않고 있다. 이는 바이러스, 웜, 스파이웨어와 같은 악성코드들로 인해 발생하는 경우가 많으며, 이러한 악성코드는 홈페이지나 소프트웨어 취약점을 악용하여 전파된다.

소프트웨어 취약점은 악의적인 사용자에게 허용된 권한 이상의 동작을 수행하거나 정보열람을 가능하게 하기 때문에 공격자는 보안상의 취약점을 악용하여 의도한 행동을 수행하거나 중요 정보를 탈취할 수 있게 된다. 따라서 취약점을 사전에 파악하여 제거하는 것이 보안 사고를 줄일 수 있는 가장 근본적인 방안 중 하나라 할 수 있다.[2]

소프트웨어 취약점 분석 방법은 화이트 박스 테스트와 블랙 박스 테스트로 구분할 수 있다. 화이트박스 테스트는 소스코드 등의 소프트웨어 리소스를 직접 분석하여 오류 사항을 탐지하는 테스트 기법이다. 정확한 취약점 탐지를 위해서는 소스코드에 대한 분석이 필요하지만 테스트에 소요되는 시간이 길어질 수 있으

며 현실적인 테스팅 환경에서는 리소스에 대한 모든 권한을 갖기 어렵기 때문에 화이트 박스 테스트를 적용하기 어렵다는 단점이 있다.

블랙 박스 테스트는 해당 어플리케이션에 데이터를 입력하여 얻어지는 결과물을 바탕으로 테스트를 진행하는 방식이다. 화면상에서 테스트를 수행할 수 있으므로 테스트와 개발 산출물에 대한 전문 지식이 없는 테스터라도 정해진 루틴에 따라 비교적 쉽게 테스트를 수행할 수 있다는 장점이 있다.[3] 블랙박스 테스트 기법의 하나인 퍼징(Fuzzing)은 비정상적인 데이터를 어플리케이션의 입력으로 주입하여 에러를 유도하는 방법으로 에러를 통해 어플리케이션의 취약점을 판단할 수 있다.

본 연구에서는 어플리케이션 중 웹 브라우저의 취약점을 점검하는 퍼징 도구를 제안한다. 인터넷의 사용이 대중화되면서 웹 브라우저는 웹 환경에서 가장 많이 사용하는 애플리케이션이 되었다. 따라서 웹브라우저를 공격 대상으로 할 경우, 다수의 불특정 사용자들을 공격할 수 있으며 그 영향도 매우 광범위하다.[4] 따라서 웹 브라우저는 비정상적인 입력에 대해서도 크래쉬 없이 처리할 수 있어야하며 상용화되기 이전에 충분한 테스트를 거쳐야 한다. 크래쉬란 웹 브라우저가 비정상적으로 종료되거나 멈추어 정상적으로 요청을 처리하지 못하는 현상을 말한다.

본 연구에서 제안하는 웹 브라우저 퍼징 도구는 샘플 웹 페이지의 소스코드에서 파라미터를 탐지한 후 비정상 값들을 삽입하여 테스트 케이스를 생성한다. 또한, 테스트하려는 웹 브라우저에서 테스트 페이지들을 순차적으로 실행하고 크래쉬 발생 여부를 모니터링 한다. 본 논문의 2장에서는 기존 연구로 소프트웨어의 테스트 방법을 소개하고 퍼징과 웹 브라우저 퍼징 도구를 소개한다. 제 3장은 연구에서 제안하는 브라우저 퍼징 도구를 설계하고, 4장에서는 구현을 보인다. 마지막으로 5장에서는 본 연구를 정리하고 향후 연구방향에 대하여 논하고자 한다.

Ⅱ. 관련 연구 및 동향

1. 취약점 분석 방법

일반적으로 어플리케이션의 취약점을 찾기 위한 방법으로는 화이트 박스 테스트, 블랙 박스 테스트 등이 존재한다. 이는 취약점 점검자가 테스트 대상에 대해 어느 정도의 접근 권한을 가지고 있는가에 따라 구분된다.

화이트 박스 테스트 방식은 소스코드 등의 소프트웨어 리소스에 접근하여 오류 사항을 탐지하는 테스트 기법이다. 따라서 화이트 박스 테스트를 위해서는 점검하고자 하는 대상의 리소스(프로그램 소스코드, 설계서, 개발자 등)에 대한 접근 권한을 가지고 있어야만 한다.

반면 블랙 박스 테스트는 취약점 점검 대상에 대한 어떤 정보도 가지고 있지 않으며, 리소스가 전혀 공개되지 않은 상태에서 테스트를 수행 한다. 만일 테스트가 접근 가능한 리소스의 범위가 어느 정도 제한된 환경에서 수행된다면 이는 그레이 박스 테스트로 분류한다. 컴파일 된 바이너리나 매뉴얼과 같은 기본적인 문서에 대한 접근이 가능한 상황을 예로 들 수 있다.[5]

1) 화이트 박스 테스트

화이트박스 테스트는 소스코드 리뷰 방식으로 대표될 수 있다. 소스코드 리뷰 시에는 매뉴얼 방식을 이용하거나 자동화 도구의 도움을 받아 프로그램의 소스코드를 확인할 수 있는데, 매뉴얼 방식의 경우 소스코드를 일일이 사람이 읽어 분석해야하므로 취약점 분석 대상 프로그램의 소스코드의 양이 방대할 경우 코드 분석에 매우 비실용적이다. 따라서 테스터의 작업량을 감소시키고 시간을 절약할 수 있는 자동화 도구를 많이 이용한다. 자동화 도구를 이용한 화이트 박스 테스트는 소스코드에 대한 접근 권한을 가지는 만큼 발견된 취약점에 대해 명확성을 제공하지만, 상대적으로 오탐률이 높아 발견된 취약점에 대해 테스터가 수

동으로 확인 작업을 반드시 거쳐야 한다는 문제점이 있다. 또한, 복잡한 프로그램일수록 취약점 분석 리포트는 방대해지기 때문에 테스터가 수행해야 하는 검증에도 상당한 시간이 소요된다. 무엇보다도 일반적인 테스트 환경에서는 상용프로그램의 소스코드에 접근할 수 없는 경우가 많기 때문에 화이트 박스 테스트가 원천적으로 불가능하다는 단점이 있다.

2) 블랙 박스 테스트

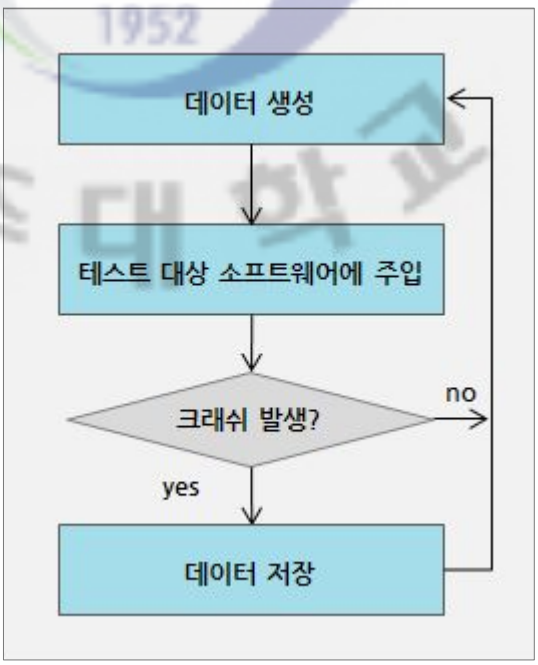
블랙 박스 테스트는 테스터가 눈으로 확인 가능한 부분만을 바탕으로 테스트하는 방법이다. 즉, 일반 사용자가 소프트웨어를 사용하는 것처럼 취약점 점검 대상 프로그램에 데이터를 입력하여 얻어지는 결과를 바탕으로 테스트를 진행한다. 화면상에서 테스트를 수행할 수 있으므로 테스터는 테스트와 개발 산출물에 대한 전문 지식이나 내부적인 로직에 대한 이해 없이도 비교적 쉽게 테스트를 수행할 수 있다. 예를 들어, 웹 어플리케이션이나 웹 서비스를 대상으로 블랙 박스 테스트를 진행하는 경우 테스터는 HTML이나 XML 등으로 작성된 문서의 입력을 조작하여 요청을 보내고 반환되는 결과를 관찰하지만 웹서버 내부에서 진행되는 처리과정과 방법에 대해서는 전혀 알지 못한다.

이처럼 블랙 박스 테스트는 프로그램 내부 구조에 대한 이해 없이 화면상으로 보여지는 출력만으로 테스트 성공 여부를 판정한다. 이 때문에 실제로 테스트가 수행된 부분과 결함이 발견된 부분을 파악하기 매우 힘들며 테스트의 종료시점을 결정하거나, 테스트가 얼마나 효과적인지 명확하게 확인할 수 없다. 또한, 개별적인 입력 벡터에 데이터를 입력하고 결과 값을 확인하는 방식으로 취약점을 찾아내는 단순한 구조인 탓에 복잡한 로직으로 인한 취약점이 발견되지 않을 수 있다는 단점을 가진다.[3]

하지만 대상 자체만으로 실행이 가능하기 때문에 테스트를 위한 추가적인 코드를 생성하는데 소요되는 시간과 비용을 절약할 수 있다. 즉, 실행 가능한 규모의 개발물을 하나의 테스트 단위로 하여 화면 중심의 블랙박스 테스트를 수행하는 것이 소프트웨어 개발 시 택할 수 있는 매우 적합한 웹 어플리케이션 테스트 방법이라 할 수 있다.[3]

2. 퍼징(Fuzzing)

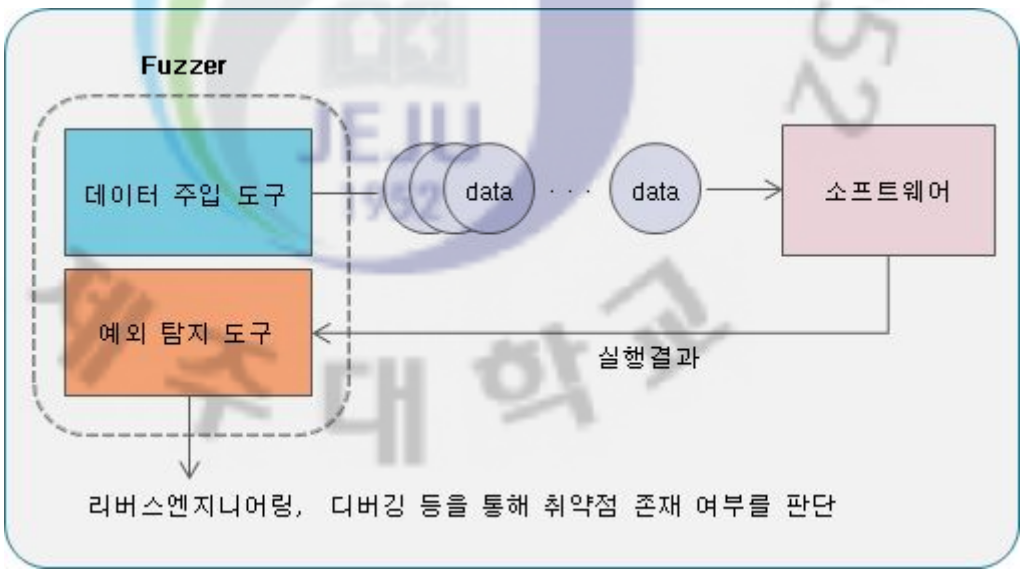
퍼징(Fuzzing)은 블랙 박스 테스팅 기법의 하나로 소프트웨어에 비정상 입력 값을 주입하고 모니터링하여 소프트웨어의 결함을 발견하는 과정이다. 퍼징이라는 용어는 1989년 위스콘신-메디슨 대학의 연구 프로젝트에서 가장 처음 사용되었다. 당시 연구 중에 있던 유닉스 응용프로그램의 견고함을 테스트하기 위해 퍼징 도구인 퍼저(Fuzzer)를 최초로 개발하여 사용했다. 이후 많은 연구를 통해 네트워크 프로토콜 분석, 파일 처리 분석 등을 위한 다양한 종류의 퍼저들이 개발되었고 현재 사용되고 있다.[5][6]



(그림 1) 퍼징 흐름도

퍼징의 동작 방식은 (그림 1)과 같다. 퍼징 대상인 목표 소프트웨어에 입력 할 데이터를 생성하고 데이터를 주입한다. 만일 입력을 처리하던 소프트웨어에 크래쉬가 발생하면 입력된 데이터와 발생된 크래쉬에 대한 관련 정보를 저장한 후 다시 데이터를 생성하고 주입하는 과정을 반복한다.[7]

퍼징의 반복적인 작업을 자동화 한 퍼징 도구를 퍼저(fuzzer)라 한다. 퍼저는 데이터 주입 도구와 예외 탐지도구로 구성된다. 데이터 주입 도구는 퍼징 대상 소프트웨어에 주입 할 입력 데이터를 생성해내고 이를 소프트웨어에 주입한다. 예외탐지도구는 소프트웨어가 입력을 처리한 결과를 모니터링 하고 크래쉬 발생 시 이를 처리하는 역할을 수행한다. 퍼저를 통해 발견된 크래쉬는 리버스 엔지니어링, 디버깅 등의 분석 방법을 이용하여 소프트웨어의 취약점으로 발전될 수 있는 오류인지 여부를 판단한다.



(그림 2) 퍼저의 구조

퍼징은 퍼징 대상의 이해 유무에 따라 덤(dumb), 스마트(smart)로 분류하고, 데이터 처리방법에 따라 생성(generation), 변이(mutation)로 나눌 수 있다.

- 덤(dumb) 퍼징

  덤 퍼징은 퍼징을 하고자 하는 대상에 대한 어떠한 기반 지식도 없이 입력을 주어 테스팅을 수행하는 기법이다. 예를 들면, 워드 어플리케이션에 임의의 입력 데이터 혹은 파일들을 무작위로 입력하는 방법을 말한다.

- 스마트(smart) 퍼징

  스마트 퍼징은 테스팅을 하고자 하는 대상에 대해 입력 데이터를 처리하기 위한 방법 혹은 올바른 데이터 구조에 대한 정보를 어느 정도 알고 있는

상태에서 테스팅을 수행하는 기법을 말한다. 예를 들면, 워드 문서 포맷과 올바른 처리 과정을 알고 있는 상황에서 오프셋이나 길이 필드에 올바르지 않은 정보를 입력하는 방법을 말한다.

- 생성(Generation)

  테스트를 위한 입력 데이터를 생성하여 처리하는 기법이다. 예를 들면, 도움말 파일 포맷을 알고 있는 상태에서 도움말 보기 어플리케이션을 위해 .chm의 확장자를 갖는 새로운 도움말 파일을 구성하는 방법이 있다.

- 변이(Mutation)

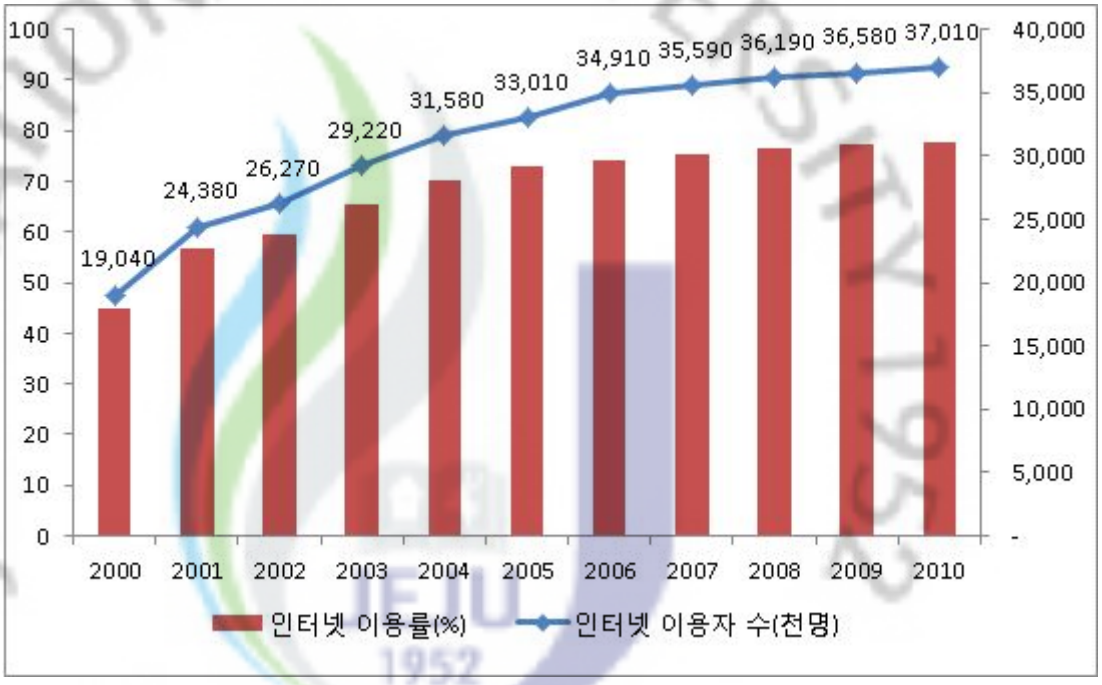
  기존 올바른 데이터에 대한 샘플들을 토대로 그 일부를 변형시켜 입력 데이터를 처리하는 기법을 말한다. 예를 들면, 네트워크 outbound 패킷의 몇 바이트를 실시간으로 변경하는 방법이 있다.[8]

3. 브라우저 퍼저(Browser Fuzzer)

한국인터넷진흥원(KISA)에 따르면 우리나라 만 3세 이상 국민 중 인터넷 이용자 수는 37,010천명으로 인터넷 이용률이 77.8%에 달하는 것으로 나타났다.[1] 이처럼 많은 사람들이 인터넷을 통해 다양한 활동을 수행하고 있으며 웹 브라우저는 인터넷이용을 위해 가장 많이 사용하게 되는 어플리케이션 중 하나라 할 수 있다.

만일 어플리케이션에 취약점이 존재하면 이는 어플리케이션을 비정상적으로 동작하게 하거나 시스템의 권한을 획득하기 위한 도구로 악용될 수 있으며, 공격자가 해당 시스템의 권한을 획득하게 되면 공격자는 웜이나 바이러스 같은 형태의 악성 코드를 사용자의 시스템에서 실행하거나 개인 정보를 획득할 수 있다.

현재 취약점 공격 코드의 대부분은 클라이언트 애플리케이션을 대상으로 하고 있으며, 이 중 특히 웹 브라우저가 주요 공격 대상이 되고 있다. 그 이유는 다음과 같다.



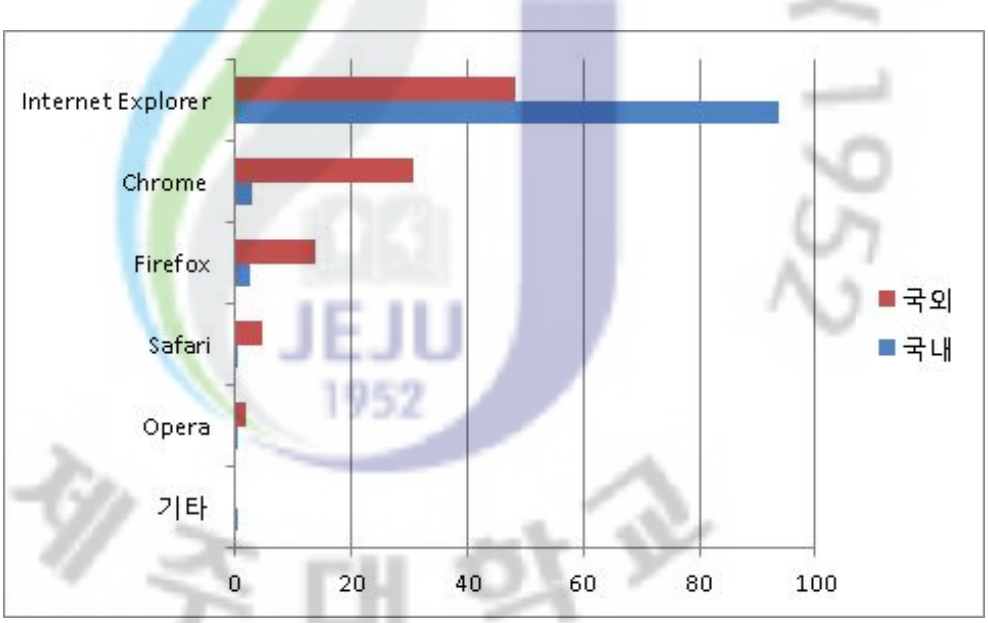
(그림 3) 우리나라 인터넷 이용률 및 이용자 수

첫째, 가장 많은 다수의 사용자를 공격대상으로 할 수 있기 때문이다. 인터넷의 사용이 대중화되면서, 웹브라우저는 일반 클라이언트 사용자가 가장 많이 사용하는 애플리케이션이 되었다. 따라서 웹브라우저를 공격대상으로 할 경우, 다수의 불특정 사용자들을 공격할 수 있으며 그 영향도 매우 광범위하다.

둘째, 다양한 애플리케이션을 공격할 수 있다. 웹 브라우저는 운영체제에서 사용하는 파일 및 애플리케이션을 로드할 수 있다. 따라서 웹 브라우저를 공격대상으로 하는 공격코드는 브라우저 내에 존재하는 취약점뿐만 아니라 취약점이 존재하는 다른 애플리케이션을 공격대상으로 할 수 있다. 예를 들어, 취약점이 존재하는 ActiveX 애플리케이션의 공격을 위해 인터넷 익스플로러를 사용할 수 있다.[4] 따라서 브라우저에 대한 취약점을 탐지하고 발견된 취약점을 제거하는 것은 보안 측면에서 매우 중요하다.

현재 다양한 웹 브라우저들이 개발되어 사용되고 있다. 브라우저 별 사용률을 살펴보면 전 세계적으로는 Internet Explorer가 48%, firefox 30.6%, chrome 13.9%, safari 4.66% 등의 순으로 나타나고 있다. 국내에서는 Internet Explorer의 점유율이 무려 93.5%로 압도적으로 우위를 차지하고 있으며, 구글 chrome이 2.8%, firefox 2.6%,

safari 0.65%로 나타났다.[9] ActiveX의 사용이 많은 국내의 인터넷 환경으로 인해 Internet Explorer의 사용률이 현저히 높은 것을 확인할 수 있다. 따라서 브라우저 자체의 취약점 뿐 아니라 ActiveX와 같은 브라우징 요소에 대한 취약점도 보안에서는 중요한 이슈가 될 수 있다고 하겠다.



(그림 4) 브라우저 별 사용률

소프트웨어 취약점 수집 사이트인 Exploit DB에 보고 된 브라우저 별 취약점을 살펴보면 <표 1>과 같이 Internet Explorer에서 2008년 이후 총 58건의 취약점이 발견되었음을 알 수 있다. firefox가 51건, 구글 chrome이 23건, safari가 39건으로 보고되는 등 브라우저 출시 이후에도 많은 취약점이 발견되고 있다.

<표 1> 브라우저 별 취약점 개수(2008년 이후)

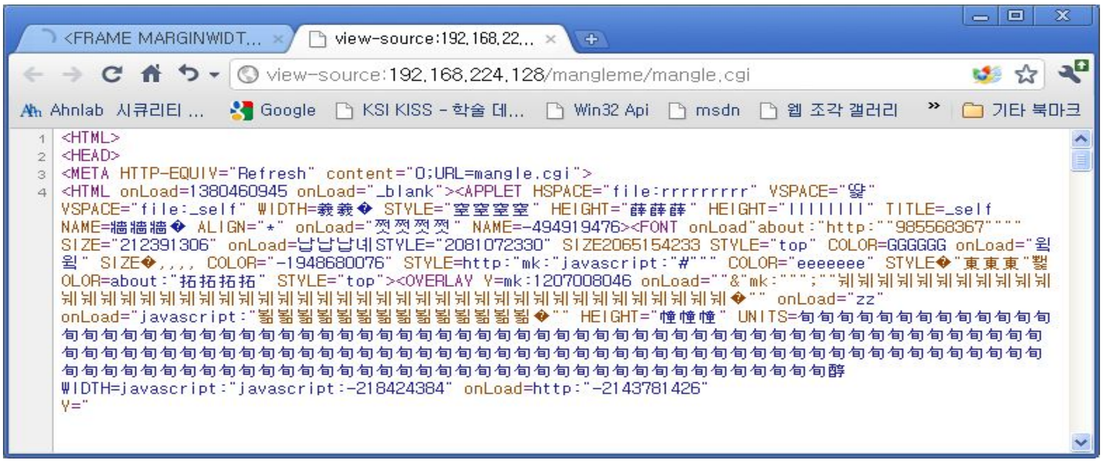
브라우저	Internet explorer	chrome	firefox	safari
취약점 수	58	23	51	39

발견된 취약점을 공개하지 않고 비밀리에 유통하거나 개발자에게 보고하는 경우를 고려할 때 실제 웹 브라우저에 존재하는 취약점은 <표 1>에서 보이는 것보다 훨씬 더 많을 것으로 예상된다. 따라서 취약점을 이용한 악성코드의 확산을 미연에 방지하기 위해서는 웹 브라우저 취약점을 탐지하여 제거하기 위한 노력이 필요하다. 웹 브라우저 취약점을 탐지하기 위한 퍼징 도구로는 mangleMe, BF3(Browser Fuzzer 3) 등이 사용되고 있다.

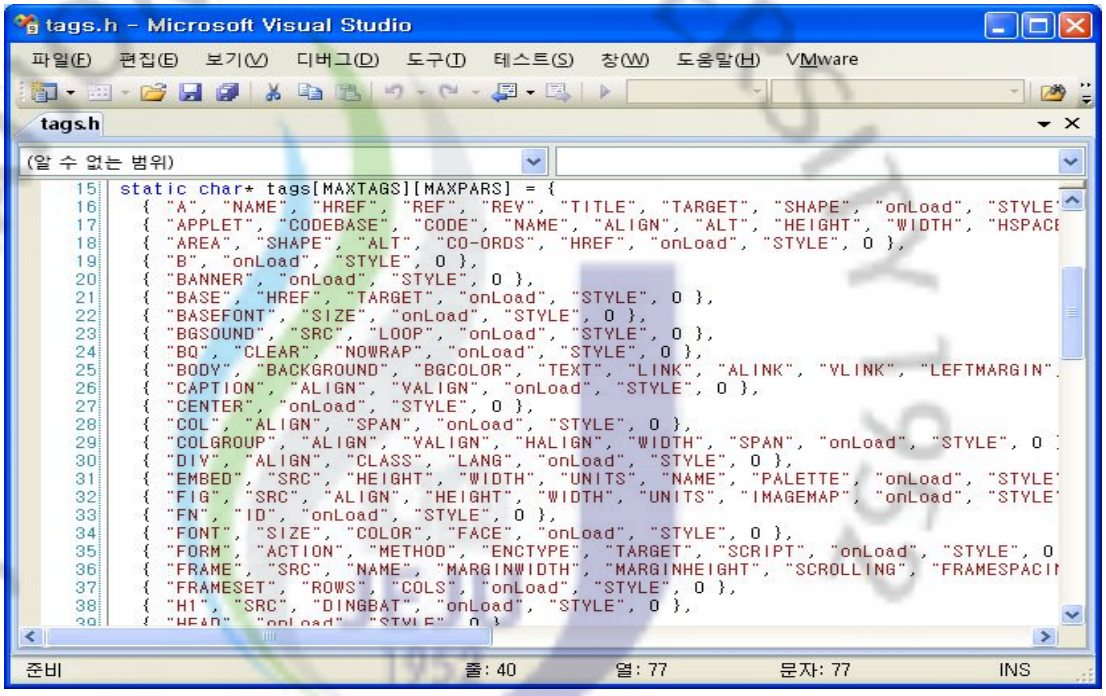
1) mangleMe

mangleMe는 최초의 웹 브라우저 퍼징 툴로 헤더 파일에 정의된 HTML 태그를 임의로 조합하여 테스트 케이스로 사용될 웹 페이지를 생성하고 웹 브라우저에서 실행시킨다. 브라우저에서 mangleMe를 실행하면 페이지가 리프레쉬 되면서 다음 테스트 케이스를 연쇄적으로 실행하며, 이는 웹 브라우저에 크래쉬가 발생할 때 까지 반복된다. [10]

mangleMe.cgi가 설치된 경로로 접속하여 mangleMe를 실행시키면 웹 브라우저에서는 퍼저가 생성한 HTML 파일이 실행된다. (그림 5)는 mangleMe에서 자동으로 생성된 HTML코드를 보인다. 소스코드의 첫 부분에 <META HTTP-EQUIV="Refresh" content="0;URL=mangle.cgi">가 포함되어 크래쉬가 발생할 때 까지 새로운 테스트 케이스가 만들어지고 실행되는 과정이 반복된다.



(그림 5) mangleMe에서 생성된 HTML코드의 예



(그림 6) mangleMe의 태그 그룹

MangleMe는 (그림 6)과 같이 헤더파일에서 HTML 태그 집합을 정의하고 있으며, 웹 브라우저에 주입 할 HTML 소스 생성 시 헤더파일의 태그를 랜덤하게 조합하고 파라미터에도 임의의 값을 입력한다.

mangleMe는 자동화 된 웹 브라우저 퍼징이 가능하나, 퍼저의 헤더파일에서 정의하고 있는 언어의 태그로 조합된 소스파일만을 생성하고 테스트할 수 있다는 단점을 가진다. 브라우징 언어가 업그레이드되어 새로운 태그가 추가되거나, 퍼저가 정의하지 않는 언어를 웹 브라우저가 정상적으로 실행하는지 여부를 테스트 할 경우 퍼징 도구의 소스코드 수정이 불가피하다.

mangleMe에서는 태그의 파라미터로 입력되는 값의 경계를 구분짓지 않고 랜덤한 값을 입력한다. 이 경우 정상적인 입력이 포함되어 크래쉬 발생 확률을 낮춤으로써 테스트의 효율성을 떨어트릴 수 있다는 단점을 가진다. 또한 크래쉬가 발생했을 경우 이력을 남기는 기능을 제공하지 않기 때문에 분석의 편의가 떨어진다.

2) Browser Fuzzer 3(BF3)

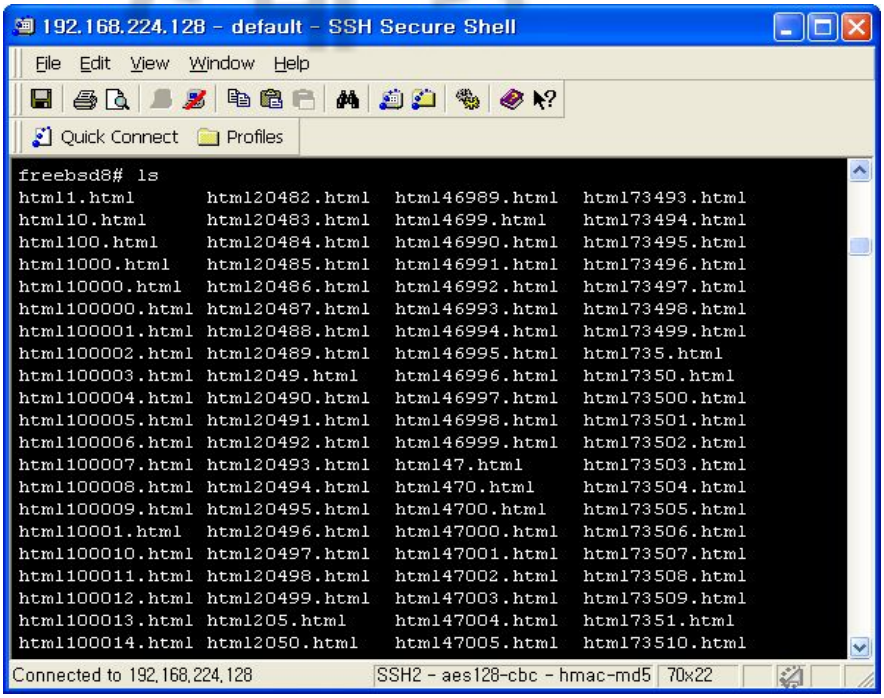
Browser Fuzzer 3는 대표적인 브라우저 퍼저로 CSS, DOM, HTML,

JavaScript, XML등으로 구성된 웹 페이지들을 테스트 케이스로 생성할 수 있는 웹 브라우저 퍼징 도구이다.[11]

mangleMe가 소스코드의 파라미터 생성 시 단순히 랜덤 값을 주입하는데 반해 bf3는 랜덤 값뿐만 아니라 내장하고 있는 퍼징 오라클을 이용하여 웹 페이지를 생성함으로써 퍼징의 효율을 높인다. 또한 샘플 페이지를 변형하는 뮤테이션 퍼징 엔진도 탑재하고 있어서 지정 된 샘플 웹 페이지의 내용을 변경해가면서 테스트하는 퍼징도 가능하게 한다.

bf3는 유닉스 기반의 퍼저로 커맨드 라인을 통해 퍼저 수행에 필요한 인자를 받아들인 후 그에 맞는 웹 페이지를 지정된 경로에 생성한다. bf3에서 생성된 웹 페이지들은 mangleMe와 같이 리프레쉬 코드를 가지고 있어서 테스터는 첫 번째 페이지를 실행시키기만 하면 생성된 웹 페이지들이 연쇄적으로 웹 브라우저를 통해 테스트된다.

웹 브라우저에 크래쉬가 발생하면 실행한 서버의 웹 로그를 통해 크래쉬 관련 정보를 분석하여 해당 오류가 웹 브라우저의 취약점으로 인한 것인지 확인할 수 있다.



(그림 7) bf3로 생성된 테스트 케이스

bf3는 HTML, CSS, JavaScript 등으로 구성된 웹 페이지를 테스트 케이스로 생성하고 웹 브라우저에서 크래쉬 발생 여부를 확인할 수 있는 종합적인 웹 브라우저 퍼징 도구이다. 퍼징 오라클, attend mode 등을 제공하여 퍼징의 효율을 높이고 사용자의 편의를 고려하였지만 퍼저에서 정의하고 있는 언어로만 테스트가 가능하며 새로운 언어나 추가된 기능에 대해 웹 브라우저 동작을 테스트하고자 하는 경우 퍼저를 수정해야만 한다. 또한, bf3에서 생성된 테스트 케이스를 테스터가 수동으로 실행시켜야 하며, 자체적인 이력 관리 기능이 없어 웹 서버의 에러 로그를 참조해야 한다는 불편함이 있다.

이처럼 기존 웹 브라우저 퍼징 도구는 퍼저 자체에서 정의한 언어에 대해서만 퍼징이 가능한 특징을 보인다. 또한, ActiveX와 같은 플러그인에 대해서는 지원을 하지 않아 플러그인에 대한 퍼징을 수행하려면 별도의 플러그인 전용 퍼저를 이용해야 하는 등 여러가지 단점이 존재한다. 따라서 본 연구에서는 기존 웹 브라우저 퍼저를 보완하여 웹 언어에 독립적이며 웹 페이지에 삽입된 플러그인에도 적용이 가능한 웹 브라우저 퍼징 도구를 설계하고 구현한다.

Ⅲ. 설계

1. 퍼징 도구의 개요

기존 웹 브라우저 퍼저들은 자신이 정의하고 있는 특정 언어만을 테스트 할 수 있으며, ActiveX와 같이 소스코드에 포함된 플러그인에 대한 취약점 탐지는 불가능하다는 한계를 가진다. 즉, HTML5와 같이 새로운 태그와 속성이 추가됐을 때 웹 브라우저가 새로운 사항들을 오류 없이 처리하는지 확인하기 위해 취약점 탐지가 신속히 이루어져야 한다. 그러나 기존 퍼저에서 새로운 버전의 웹 언어를 퍼징하기 위해서는 퍼저 자체의 소스코드 수정이 불가피하다. 또한, 웹 브라우저가 웹 페이지에 삽입된 플러그인을 크래쉬 없이 실행하는지 확인하기 위해서 별도의 전용 퍼저(ActiveX 퍼저 등)를 사용해야 한다는 문제점이 존재한다.

따라서 본 연구에서는 기존 연구의 단점을 보완할 수 있는 퍼징 도구를 제안하고자 한다. 본 연구에서 설계, 구현하고자 하는 웹 브라우저 퍼징 도구의 요구사항은 아래와 같다.

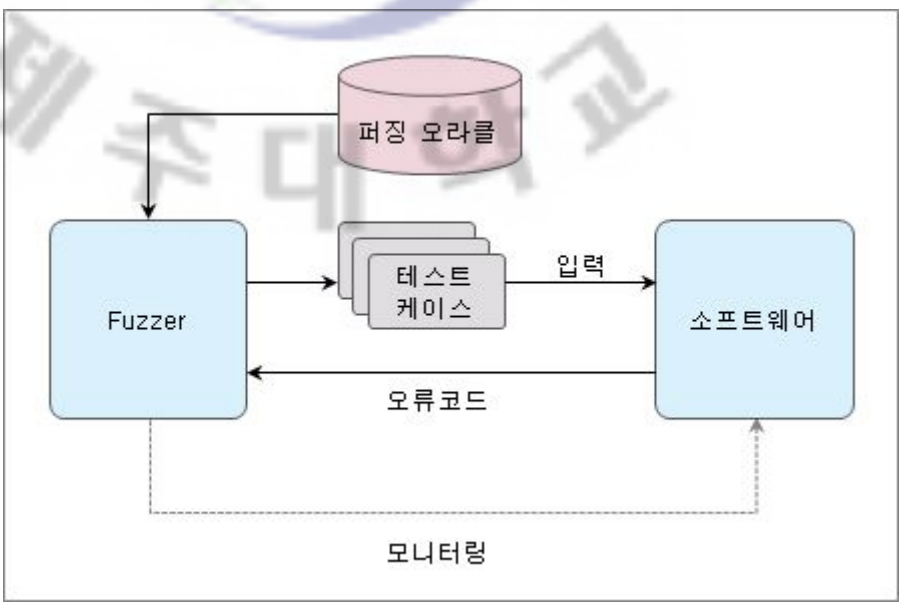
첫째, 테스트 케이스를 생성하는 웹 언어에 대한 의존도를 낮출 수 있어야 한다. 퍼징 도구 자체의 수정 없이도 다양한 웹 언어와 버전에 대한 테스트 케이스를 생성하여 웹 브라우저를 테스트할 수 있어야 한다.

둘째, ActiveX와 같은 웹 브라우저 플러그인에 적용 가능해야 한다. 웹 페이지에는 다양한 플러그인이 탑재되나, 웹 브라우저 퍼징 도구에서 동시에 테스트하지 못해 별도의 퍼징 툴을 적용하는 불편함이 존재한다. 따라서 웹 브라우저 퍼징 시 플러그인에 대한 테스트를 수행할 수 있는 방안이 필요하다.[12]

본 논문에서는 위 요구사항을 충족하는 파라미터 분석을 통한 언어 독립적 웹 브라우저 퍼징 도구를 설계・구현하였다.

2. 퍼징 도구의 설계

본 논문에서 제안하는 웹 브라우저 퍼징 도구는 샘플로 주어진 웹 페이지인 템플릿의 소스코드를 분석하여 파라미터를 탐지하고 그 자리에 퍼징 오라클에서 정의하는 임의의 값을 입력하여 테스트 케이스를 생성하는 뮤테이션 방식을 기반으로 동작한다. 일반적으로 퍼징 오라클을 이용하는 퍼저의 구성도는 (그림 8)과 같다. 퍼징 오라클은 웹 페이지의 파라미터로 입력되었을 때 웹 브라우저 실행 시 오류를 유발 할 수 있는 값의 집합으로, 다수의 연구에서 경험적으로 누적된 데이터이다.



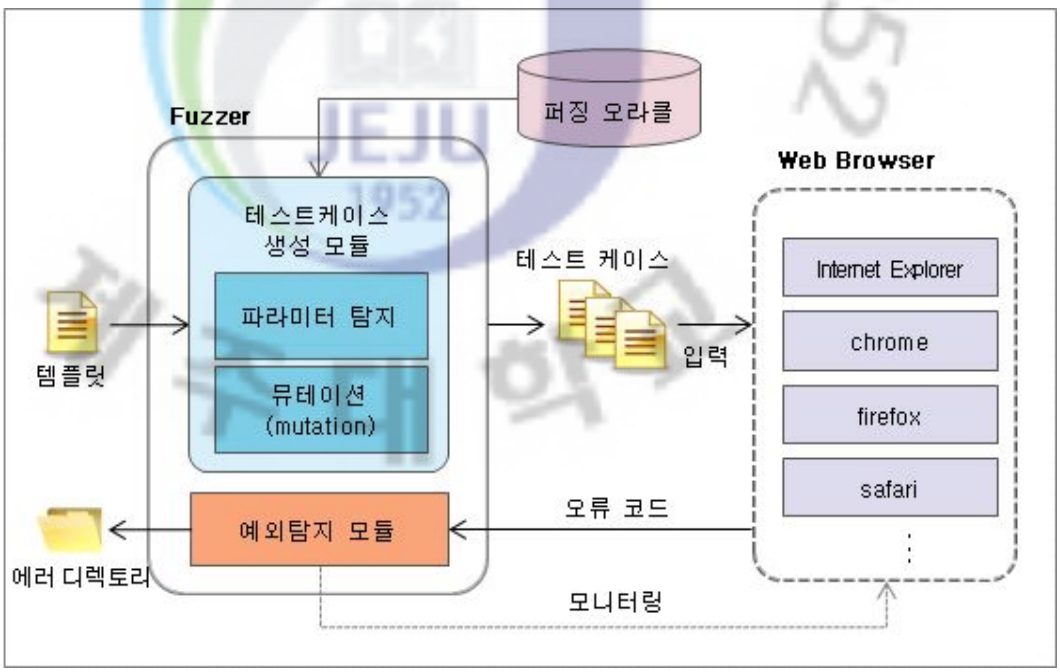
(그림 8) 퍼징 오라클을 이용한 퍼저의 개념도

본 논문에서 설계한 웹 브라우저 퍼징 도구는 (그림 9)와 같이 테스트 케이스 생성 모듈과 예외탐지 모듈로 구성된다.

테스트 케이스 생성 모듈은 테스트 대상인 웹 브라우저의 입력으로 사용 할 웹 페이지를 생성하는 역할을 수행한다. 주어진 템플릿의 소스코드를 분석하여

파라미터를 구분해내는 파라미터 탐지 모듈과 파라미터에 퍼징 오라클을 입력하여 테스트 케이스를 만들어내는 뮤테이션(mutation) 모듈 등 두 개의 세부 모듈을 가진다. 테스트 케이스 생성 모듈에서 만들어진 웹 페이지들은 테스트 대상 웹 브라우저에서 실행된다.

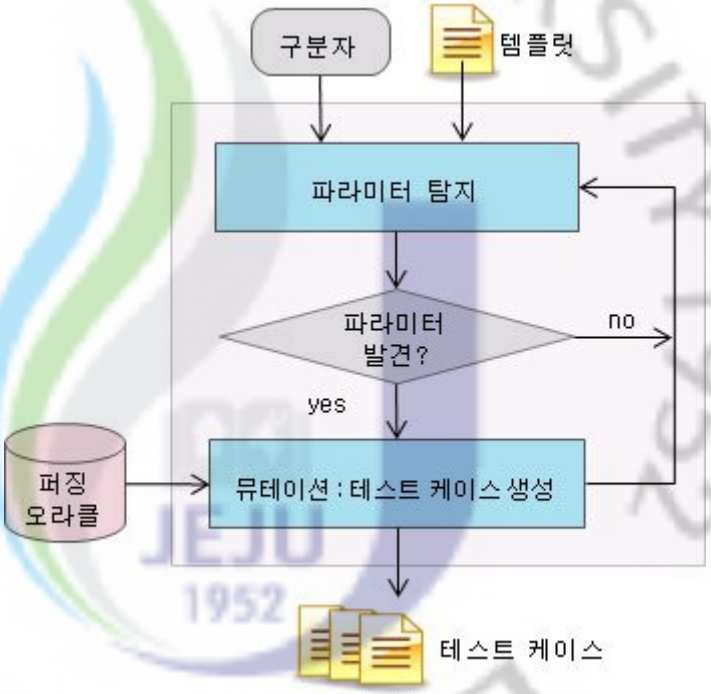
예외 탐지 모듈은 웹 브라우저에서 크래쉬가 발생했을 때 해당 웹 페이지를 오류 디렉토리로 복사하고 로그를 기록한 후 다음 테스트 웹 페이지가 웹브라우저에서 실행될 수 있도록 제어하는 기능을 수행한다.



(그림 9) 퍼징 도구 구성도

1) 테스트 케이스 생성 모듈

테스트 케이스 생성 모듈은 웹 브라우저 퍼징 시 입력으로 사용할 비정상 웹 페이지를 생성한 후 이를 웹 브라우저에서 실행시키는 역할을 수행한다. 테스트 케이스 생성 모듈은 파라미터 탐지와 뮤테이션의 두 가지 세부 모듈로 구성되어 있으며, 템플릿 및 퍼징 오라클을 입력으로 이를 가공하여 테스트 케이스 생성 작업을 수행한다. 테스트 케이스 생성 모듈에서 생성된 테스트 웹 페이지들은 테스트 대상 웹 브라우저에서 실행되며 웹 브라우저의 크래쉬를 유도한다.



(그림 10) 테스트 케이스 생성 모듈 흐름도

앞서 언급한 바와 같이 기존 웹 브라우저 퍼저는 테스트 케이스 생성 시 헤더 파일에서 정의하고 있는 특정 웹 언어의 태그와 엘리먼트들을 조합한다. 때문에 퍼저가 정의하지 않는 웹 언어이거나 정의하고 있더라도 새로운 버전인 경우에는 퍼징을 수행할 수 없으며 플러그인과 같은 기타 브라우징 요소에도 퍼징을 적용할 수 없는 한계를 가짐을 확인할 수 있었다. 따라서 본 연구의 테스트 케이스 생성 모듈에서는 실제 인터넷 환경에서 템플릿 파일을 수집하고, 템플릿의 소스코드를 분석하여 파라미터 부분을 탐지한 후, 식별된 파라미터에 퍼징 오라클에서 정의하는 값을 입력함으로써 테스트 웹 페이지를 생성하여 기존 웹 브라우저 퍼저를 보완한다.

템플릿은 테스트 케이스 생성 모듈에서 테스트 웹 페이지를 생성하기 위한 밑바탕이 되는 소스파일이다. 테스터는 퍼저 수행 시 템플릿을 지정하는데 wget 등의 도구를 이용하여 실제 인터넷 웹 페이지를 다운받을 수도 있다. wget은 수집하고자 하는 파일의 경로를 인자로 받아 해당 경로에 존재하는 파일을 로컬

시스템으로 다운로드 하는 동작을 수행한다. 특정 사이트를 퍼징의 템플릿으로 지정하고자 한다면 wget의 '-r'(recursive) 옵션을 이용하여 해당 페이지에 링크된 모든 웹 페이지를 수집하여 퍼징에 이용할 수 있다. 제주대학교 홈페이지(http://jejunu.ac.kr)는 HTML, CSS, JavaScript 등 다양한 요소로 이루어져 있어 이를 본 논문의 템플릿 예제로 이용하고자 한다.

<표 2> 퍼저의 템플릿 예시(제주대학교 웹 페이지)

```
1  <html>
2  <head>
3  <meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
4  <title>국립제주대학교 - Jeju National University</title>
5  <!--title>국립제주대학교 - Jeju National University</title-->
6  <script language="javascript" type="text/javascript"
7  src="/_html/include/javascript.js"></script>
8  <script language="javascript" type="text/javascript"
9  src="/_html/include/FlashActiveX.js"></script>
10 <script language="javascript" type="text/javascript"
11 src="/_html/include/FlashMenuLink.js"></script>
12 <script language="javascript" type="text/javascript" src="/_html/include/zoom.js"></script>
13 <link href="/_html/include/style.css" rel="stylesheet" type="text/css">
14 <script type="text/javascript">
15 function change_img(img_name,img_src) {
16         document[img_name].src=img_src;
17 }
18 function juyuLayerDisplay(data) {
19         document.getElementByld("juyuLayer").style.visibility = data;
20 }
21 </script>
22 </head>
23 <body background="images/new_main/main_bg.gif" style="background-repeat:repeat-x">
24 <style type="text/css">
25 <!--
26 #onBar {
27         position:absolute;
```

```
28              left:10px;
29              top:0px;
30              width:708px;
31              height:336px;
32              z-index:300;
33              visibility:  hidden;
34  }
35  #qr {
36              position:absolute;
37              left:185px;
38              top:338px;
39              width:214px;
40              height:87px;
41              z-index:300;
42  }
43  -->
44  </style>

    … 이하 생략
```

(1) 파라미터 탐지 모듈

파라미터 탐지 모듈에서는 템플릿의 소스코드에서 특정 구분자(delimiter)의 패턴을 검사하여 파라미터를 탐지 한다.

웹 언어의 코드에서 파라미터 사용 패턴을 살펴보면 HTML 언어의 경우 <표 2>의 3열, 23열에서와 같이 대부분 '="'뒤에 파라미터를 입력함을 확인할 수 있다. <표 2>의 26~40열에서는 CSS코드가 포함되어 있는데 코드의 내용을 살펴보면 ':'의 다음에 파라미터를 입력하는 패턴을 보인다. 또한 <표 2>의 6~22열에서 사용된 javascript에서는 ' (" ' 뒤에 파라미터가 위치한다. 따라서 파라미터 탐지 모듈에서는 각 언어에서 파라미터를 구분하는 패턴을 구분자(delimiter)로 등록하고, 템플릿의 소스코드에 해당 구분자가 존재하는지 검사함으로써 템플릿의 파라미터를 탐지한다. 파라미터의 구분자는 퍼저에서 기본으로 제공하는 패턴 외에 사용자가 직접 추가할 수 있도록 하여 퍼저의 확장

성을 보장한다.

(2) 뮤테이션(mutation) 모듈

파라미터 탐지 모듈에서 템플릿 파일의 파라미터를 탐지하면 그 위치를 뮤테이션 모듈로 전달한다. 뮤테이션 모듈에서는 전달받은 위치에 퍼징 오라클에서 정의하는 값을 주입하여 테스트 케이스로 사용 될 웹 페이지들을 생성한다. 이 과정은 템플릿의 소스코드를 처음부터 끝까지 읽어가면서 파라미터를 발견할 때마다 계속 반복한다.

퍼징 오라클은 웹 페이지의 파라미터로 입력되었을 때 웹 브라우저 실행 오류를 유발 할 수 있는 값의 집합으로, 다수의 연구에서 경험적으로 누적된 데이터이다. 본 연구에서는 기존 웹 브라우저 퍼저인 Browser Fuzzer 3(BF3)에서 사용하는 퍼징 오라클을 퍼저의 기본 오라클로 사용한다.

<표 3> 퍼징 오라클

<table>
<tr><th>구분</th><th>오라클 예</th></tr>
<tr><td>정수 언더플로우 및 오버플로우</td><td>"2147483647", "-2147483647", "2147483648", "-2147483648",
"4294967294", "4294967295", "4294967296", "357913942",
“-357913942"</td></tr>
<tr><td>문자열 오버플로우</td><td>"A x 550", "A x 1100", "A x 2100", "A x 4200", "A x 8400",
"A x 16500", "A x 33000", "A x 65800", "A x 131200",
"A x 262400", "A x 525000", "A x 1050000"</td></tr>
<tr><td>포맷 스트링</td><td>"%n%n%n%n%n", "%p%p%p%p%p", "%s%s%s%s%s",
"%d%d%d%d%d", "%x%x%x%x%x", "%s%p%x%d", "%.1024d",
"%.1025d", “%.2048d", "%.2049d", "%.4096d", "%.4097d",
"%99999999999s",
"%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s
%s%s%s%s%s"</td></tr>
<tr><td>잘못된 구분자</td><td>“test|touch”, “/tmp/FU_ZZ_ED|test”, “test;touch”,
“/tmp/FU_ZZ_ED;test”, “|/bin/sh|”
“test`C:/WINDOWS/system32/calc.exe`test”</td></tr>
<tr><td>연산수행오류</td><td>“-=_+”, “[]\{}”, “|;\\':”</td></tr>
</table>

<표 4> ActiveX 사용 소스코드의 예

```
(생략) …
1  <div class="event" style="display:none" id="swf_area">
2  <object width="445" height="271"
3  codebase="https://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.
4  cab#version=7,0,0,0" classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000">
5  <param name="allowScriptAccess" value="sameDomain" />
6  <param name="movie" value="/apps/site/default/flash/main.swf" />
   … (생략)
```

<표 5> 생성된 테스트 케이스의 예

```
1  <html>
2  <head>
3  <meta http-equiv="-2147483647Content-Type" content="text/html; charset=euc-kr">
4  <title>국립제주대학교 - Jeju National University</title>
5  <!--title>국립제주대학교 - Jeju National University</title-->
6  <script language="javascript" type="text/javascript"
7  src="/_html/include/javascript.js"></script>
   … (생략)
```

퍼징 오라클은 <표 3>에서 보이는 바와 같이 오버플로우를 발생시킬 수 있는 문자열 및 정수, 메모리상의 데이터를 확인 및 변조할 수 있게 하는 포맷 스트링 등으로 구성되어 있다. 퍼징 오라클에 정의되어 있는 값 외에 사용자가 추가하고자 하는 값이 있는 경우 이를 반영할 수 있도록 설계하였다.

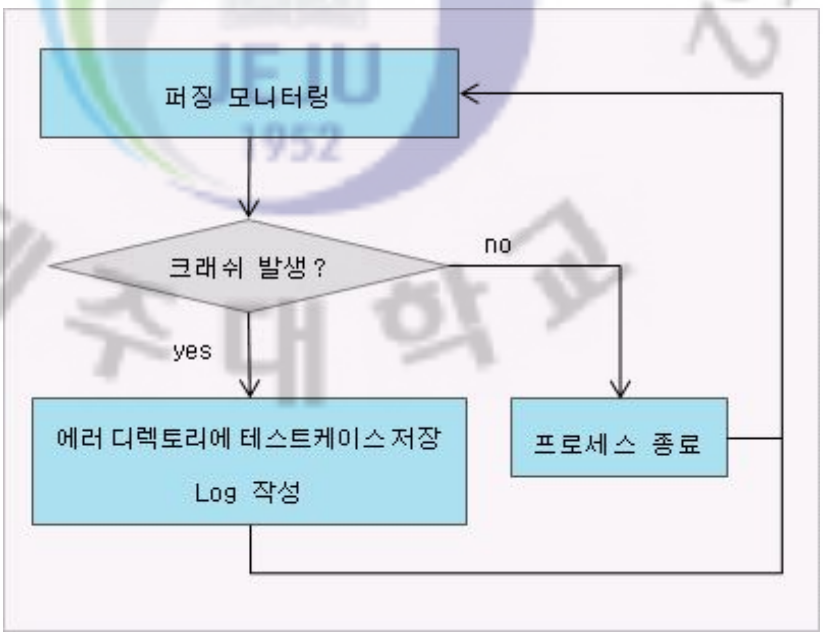
템플릿의 파라미터에 오라클 데이터를 입력하여 생성된 웹 페이지는 생성과 동시에 퍼저에서 지정한 웹 브라우저에서 실행된다. 설계한 퍼저에서는 웹 브라우저 실행파일 경로를 입력받아 테스트하는 웹 브라우저를 유연하게 변경할 수 있도록 하였다.

<표 4>는 ActiveX가 사용된 웹 페이지의 소스코드이다. ActiveX 코드가 삽입될 때 파라미터를 입력받으므로 같은 방법으로 파라미터를 탐지하고 그 자리에 퍼징 오라클 값을 넣어 웹 브라우저에서 퍼징 할 수 있다. 이처럼 본 연구에서 제

안하는 퍼저는 플러그인에 대한 퍼징이 가능하도록 설계되었다. <표 5>는 생성된 테스트 케이스의 예를 보인다.

2) 예외탐지 모듈

퍼저의 테스트케이스 생성 모듈에서 생성된 테스트 웹페이지는 지정된 웹 브라우저를 통해 실행되며, 본 연구에서 설계하는 퍼징 도구의 예외 탐지 모듈에서는 웹 브라우저의 동작을 모니터링하여 크래쉬 발생 시 이를 처리하는 기능을 수행한다.



(그림 11) 예외탐지 흐름도

예외 탐지 모듈은 퍼징 모니터링과 예외 처리의 세부 모듈로 구성된다.

(1) 퍼징 모니터링 모듈

앞에서 설계한 바와 같이 테스트케이스 생성 모듈에서 생성된 테스트 웹 페이지는 웹 브라우저에 의해서 실행된다. 퍼징 모니터링 모듈에서는 웹 브라우저에서 수행된 결과를 모니터링하여 크래쉬 여부를 판단하고 예외 처리 모듈에 데이터를 전송하는 역할을 수행한다.

웹 브라우저가 테스트 웹 페이지를 실행하여 크래쉬가 발생하면 웹 브라우저 실행 프로세스는 비정상 종료 코드를 리턴하고 종료한다. 이 경우 모니터링 모듈은 해당 웹 페이지의 정보를 예외처리 모듈에 전달한다. 반면 웹 브라우저가 테스트 웹 페이지를 크래쉬 없이 정상 수행했다면 웹 브라우저 실행 프로세스는 종료하지 않고 시스템 상에 남아있게 된다. 모니터링 모듈은 정상적인 웹 페이지를 실행한 프로세스를 처리하기 위해 일정 시간동안 프로세스의 응답을 기다리고 응답이 없는 프로세스를 정상 실행 완료된 것으로 판단하여 강제 종료한다.

(2) 예외 처리 모듈

예외 처리 모듈에서는 웹 브라우저에서 크래쉬가 발생했을 때 관련정보를 저장하여 추후 웹 브라우저 취약점 분석에 사용될 수 있도록 하는 역할을 한다.

크래쉬가 발생하여 퍼징 모니터링 모듈에서 실행된 웹 페이지 정보를 전달받으면 예외 처리 모듈에서는 해당 웹 페이지를 정해진 경로에 저장한다. 또한 사용자에게 오류가 발생한 웹 페이지의 파일명, 오류 코드 등의 관련 정보를 제공하기 위한 로그를 작성한다.

앞서 설명하였듯이 크래쉬가 발생하면 웹 브라우저 실행 프로세스는 오류코드를 리턴하고 종료된다. 이 때, 윈도우에서는 발생한 오류를 처리하기위해 디버거를 자동 실행시킨다. 이를 JIT(Just-in-Time) 디버거라고 하며 윈도우에서는 기본적으로 Dr.Watson 이라는 디버거를 사용한다. 만일 해당 디버거가 활성화 됐을 경우 본 연구에서 설계한 퍼저를 이용하여 테스트 중 크래쉬가 발생했을 때 JIT가 실행되어 퍼징의 연속성이 방해를 받게 된다. 따라서 본 퍼저 수행 시 별도의 디버깅 작업을 수행하지 않도록 하는 디버거를 윈도우 운영체제의 기본 디버거로 설정하도록 구성한다.

본 연구에서 제안하는 퍼저는 템플릿의 소스코드에서 파라미터를 식별한 후 퍼징 오라클에서 정의된 값을 파라미터로 주입한 웹 페이지를 생성하여 퍼징에 사용하도록 설계되었다. 때문에 퍼저 자체에서는 테스트 웹 페이지 조합을 위해 사용될 웹 언어를 정의하지 않는다. 따라서 기존 웹 브라우저 퍼저가 지니고 있던 특정 웹 언어에 대한 의존도가 높은 문제점을 보완할 수 있다. 웹 언어에 대한 의존도를 낮춤으로써 HTML5와 같이 웹 언어의 새로운 버전에 태그나 규약

이 추가된 경우에도 HTML5로 구현된 웹 페이지를 템플릿으로 이용하면 웹 브라우저에 대한 퍼징을 수행할 수 있다. 또한 ActivX와 같은 플러그인에 대한 퍼징도 동시에 수행할 수 있다.

본 연구에서 제안하는 퍼징 도구는 뮤테이션 방식으로 테스트에서 사용하는 템플릿의 파라미터 부분을 변경하여 테스트 케이스를 생성하도록 설계되었다. 따라서 템플릿을 특정 웹 사이트의 페이지들로 구성한다면 사이트 단위의 퍼징이 가능하여 다양한 웹 브라우저에서 해당 사이트가 오류 없이 실행되는지 점검하는 용도로도 사용될 수 있는 장점을 지닌다.

Ⅳ. 구현

본 장에서는 앞에서 제안하고 설계한 파라미터 탐지와 뮤테이션을 이용한 웹 브라우저 퍼징 도구를 구현한다. 연구에서 제안하는 퍼저의 구현 환경과 퍼저의 구현된 모습을 보인다.

1. 구현환경

본 연구에서 제안한 퍼징 도구의 구현에 이용한 시스템 환경은 <표 6>과 같다.

<표 6> 시스템 구현 환경

운영체제	Windows XP SP3
구현언어	visual C++
하드웨어	Intel(R) Dual(TM) CPU 2.0GHz

2. 퍼징 도구의 구현

본 연구에서 구현한 퍼저의 실행을 위한 구성요소로는 퍼저를 수행하기 위한 실행파일인 fuzzila.exe, TEMP, MUTE, FAULT 디렉토리와 FuzzLog.txt 파일 등이 있다.

- fuzzila.exe : 퍼저를 실행하기 위한 실행파일이다.

- TEMP : wget으로 수집한 템플릿 페이지들이 저장되는 디렉토리로 퍼징 시 테스트 케이스 생성에 이용된다.
- FAULT : 퍼저 실행 중 크래쉬가 발생한 테스트 케이스들을 저장하는 디렉토리이다.
- MUTE : 생성된 테스트 웹 페이지들을 저장하는 공간이다. 생성된 모든 테스트 케이스를 저장하면 퍼징 성능이 저하 될 우려가 있으므로 필요시에만 테스트 케이스를 저장하도록 구현하였다.
- FuzzLog.txt : 로그파일로 퍼저 실행 중 발생한 에러에 대한 로그를 기록한다.

<표 7>은 파라미터 탐지모듈의 구현된 모습을 보인다. 파라미터 탐지 모듈에서는 템플릿의 소스코드에서 파라미터의 위치를 찾아내는 역할을 수행한다. 변수 'key'에 퍼저에서 정의한 구분자를 정의하고, 템플릿의 소스코드를 순차적으로 읽어가면서 'key'와 매칭되는 부분을 탐색한다. 파라미터 부분을 발견하면 gen_File() 함수를 호출한다.

<표 7> 파라미터 탐지모듈의 구현

```
1  char *findkey_Buffer(int current_pos, char *Buffer, const char *key, char* FileName)
2  {
3    char *FindStr = NULL;
4    FindStr = strstr(Buffer, key);
5    if(FindStr != NULL) {
6      current_pos += (FindStr-Buffer) + strlen(key);
7      gen_File(FileName, key, current_pos, FileName);
8      char *next_buffer = FindStr+(strlen(key));
9      findkey_Buffer(current_pos, next_buffer, key, FileName);
10   }
11   return FindStr
12 }
```

<표 8>은 뮤테이션 모듈을 구현한 코드이다. 파라미터 탐지 모듈에서 확인된 파라미터 부분에 오라클에 정의되어 있는 비정상 값을 입력하여 새로운 테스트 케이스를 생성하는 과정을 수행한다. 퍼징 오라클은 Oracle[] 배열에 정의되어 있다.

<표 8> 뮤테이션 모듈의 구현

```
1  bool gen_File(const char *srcName, const char *key, int pos, char *FileName)
2  { (생략)…
3
4   while( (ch=fgetc(src)) != EOF )
5    {
6       if(pos == n) {
7       char szOraBuffer[256];
8       if(oracle[nOraOffset] !=NULL){
9           memset(szOraBuffer, 0x00, sizeof(szOraBuffer));
10          sprintf(szOraBuffer, "%s ", oracle[nOraOffset]);
11          printf(szOraBuffer);
12          fputs(szOraBuffer, dest);
13      }
14    }
15    if( fputc(ch, dest) == EOF ) break;
16    n++;
17  }
18    fclose(src);
19  fclose(dest);
20  printf(destName);
21  ExecProc(APP_PATH, destName);
22  return true
23 }
```

<표 9> 예외탐지 모듈의 구현

```
1  int ExecProc(char *szApplication, char *szFuzzFile){
2  (생략)…
3
4   if(CreateProcess(NULL,szCommand,NULL,NULL,FALSE,0,NULL,NULL, &si, &pi))
5   {
6    if (WAIT_OBJECT_0 == WaitForSingleObject(pi.hProcess, 3 * 1000))
7    {
8     if(GetExitCodeProcess(pi.hProcess, &dwExitCode))
9     {
10      if(dwExitCode != 0){
11         LogMsg("Taint File : %s\n", szFuzzFile);
12         LogMsg("Exception Code: %x\n", dwExitCode);
13
14         if(szFuzzFile != NULL){
15            NsFileDisassemble szFileDis;
16            memset(szCopyPath, 0x00, MAX_PATH - 1);
17            _splitpath(szFuzzFile, szFileDis.szDrive, szFileDis.szDir,
   szFileDis.szFName, szFileDis.szExt);
18            sprintf(szCopyPath, "%s%s%s%s", COPY_PATH, szFileDis.szFName, ".",
19 szFileDis.szExt);
20
21            if(!CopyFile(szFuzzFile, szCopyPath, FALSE)){
22               LogMsg("Error : Failed to COPY\n");
23            }
24
25            else{
26               LogMsg("Complete : Successed to COPY\n");
27            }
28         }
29 …(생략)
```

예외탐지 모듈에서는 웹 브라우저에 주입된 테스트 케이스의 실행을 모니터링하고 제어하는 역할을 수행하며 그 구현된 내용은 <표 9>와 같다. 테스트 케이스 생성모듈에서 만들어진 테스트 케이스를 프로세스가 실행하며, 크래쉬가 발생하지 않았다면 종료코드가 리턴되지 않는다. 예외탐지 모듈에서는 특정 시간동안 응답이 없는 프로세스를 강제로 종료시킨다. 연구에서는 <표 9>의 6행과 같이 응답 대기 시간을 3초로 설정하였다. 반면 테스트 케이스를 실행하던 중 크래쉬가 발생하였다면 FuzzLog.txt 파일에 테스트 케이스의 파일명과 예외 코드를 입력한다. 또한 FAULT 디렉토리에 테스트 케이스를 저장하는 작업을 수행 한 뒤 다음 테스트 케이스를 실행한다.

Ⅴ. 결론

본 논문에서는 파라미터 탐지와 뮤테이션을 이용한 웹 브라우저 퍼징 도구를 설계·구현하였다.

연구에서 제안한 퍼징 도구는 테스트 케이스 생성을 위해 인터넷 상에서 웹 페이지를 수집하여 템플릿으로 이용한다. 웹 언어의 파라미터를 식별할 수 있는 구분자를 정의한 후 이를 이용하여 템플릿의 소스코드에서 파라미터를 탐지하며, 탐지한 파라미터에 퍼징 오라클의 데이터를 입력하여 테스트 케이스를 생성한다.

생성된 테스트케이스를 취약점 점검 대상 웹 브라우저에 주입하여 실행시킴으로서 웹 브라우저의 크래쉬 발생 여부를 모니터링 한다. 브라우저에서 크래쉬가 발생했을 때 퍼저는 크래쉬를 유발한 테스트 케이스를 별도의 디렉토리에 저장하고, 로그를 남긴다.

제안한 웹 브라우저 퍼징 도구는 기존 웹 브라우저와는 다르게 퍼저 내부에서 웹 언어의 태그를 정의하지 않는다. 대신 웹 페이지를 수집한 템플릿에 오라클을 입력하여 테스트 케이스를 생성하는 뮤테이션 방식을 사용하기 때문에 기존 웹 브라우저 퍼저가 가지고 있던 단점 중 하나인 특정 웹 언어에 대한 의존도가 높다는 문제를 보완할 수 있다. 뿐만 아니라, 웹 언어의 새로운 버전에서 태그나 엘리먼트가 추가되더라도 이에 구속되지 않고 퍼징을 수행할 수 있다는 장점을 지닌다. 또한 뮤테이션 방식을 사용함으로써 웹 페이지에 포함되어 있는 플러그인에 대해서도 퍼징이 가능하여 별도의 플러그인용 퍼저를 사용하지 않아도 웹 브라우저에서 플러그인이 크래쉬를 유발하는지 여부를 테스트할 수 있다.

본 연구에서 제안하는 퍼징 도구는 wget 도구를 이용하여 템플릿을 수집한다. 따라서 웹 사이트 단위의 퍼징이 가능하므로, 웹 사이트가 다양한 브라우저에서 크래쉬 없이 잘 동작하는지 테스트하는 용도로 사용 될 수 있을 것으로 기대한다.

향후 퍼징 옵션을 지정할 수 있도록 인터페이스를 보완하여 테스트 상황에 맞는 맞춤형 퍼징이 가능하도록 개선이 요구된다. 또한, 퍼징에 소요되는 시간을

줄이고 크래쉬 발생 확률을 높이기 위한 퍼징 오라클의 구성과 테스트 케이스 생성 방법에 대한 논의가 필요하다. 그리고 본 퍼징 도구가 스마트 환경에서 동작할 수 있도록 응용·발전시켜 스마트 기기용 브라우저 퍼징 도구 개발에 대한 연구가 필요할 것으로 보인다.

참 고 문 헌

[1] 한국인터넷진흥원(KISA) 2010 인터넷 이용 실태 조사(최종보고서)

[2] 박용수, 조성제 외 (2009. 9) 윈도우 멀티미디어 취약점 분석 방법론 연구. 한국인터넷진흥원 연구보고서

[3] 임정희, 이시현, 장진아, 최병주, 황상철 (2009). 사용자 화면 중심의 블랙 박스 테스트와 웹 인터페이스 테스트 커버리지를 통한 웹 어플리케이션 테스트 방법. 정보과학회논문지 : 소프트웨어 및 응용, 36(9)

[4] 김민성, 정덕영 (2008. 6.) 클라이언트 애플리케이션에서의 해킹 위협. 정보보호학회지, 18(3)

[5] Michael Sutton (2007. 6) Fuzzing : Brute Force Vulnerability Discovery

[6] 김연재 외 (2010.10) 취약점 분석을 위한 퍼징(Fuzzing). 부경대학교 연구보고서

[7] 이동현, 김수영, 최대식, 오형근 (2008.11.) 필드 정보와 결함 주입 규칙을 사용하는 파일 퍼징 시스템. 보안공학연구논문지, 5(4)

[8] 양진석, 김태균, 김형천, 홍순좌 (2007.4) ROAD(RPC Object vulnerability Automatic Detector) 도구의 설계 및 구현. 정보보호학회논문지, 17(2)

[9] StatCounter(http://gs.statcounter.com)

[10] http://lcamtuf.coredump.cx/

[11] http://www.krakowlabs.com/dev.html

[12] 남현우, 김수현 (2010. 6.) CPU 독립적인 웹브라우저 플러그인의 설계. 한국인터넷정보학회 학술발표대회 논문집, (207-211)

<Abstract>

Web Browser Fuzzing Tool using Parameter Detection and Mutation Method

Ahn, Ji-min

Graduate School of Education in Jeju National University

(Majoring in Computer Education)

Supervised by Professor Kim, Han-il

Rapid advances in the information and computer technologies have brought us the wide spread of the computers and the Internet's World Wide Web in our daily life. Such kinds of changes even include services that require high-level information security like banking or many other types of financial transactions. However, being too much dependent upon web-based transactions have shown severe adverse effects, for example, security mishaps, with startling varieties; which includes being a victim of hacking, data spill and DDoS (Distributed Denial of Service) attacks.

Computer security mishaps are typically caused by malicious codes, such as computer virus, worm, or spyware, and they rapidly spread to other vulnerable web pages and software packages as well. Any vulnerability in software can allow a malicious user to perform an unauthorized action and to

disclose something important that should be kept secure. Furthermore, it can result in the leak of personal information or other serious security accidents. Therefore, an early detection and elimination of any vulnerability in software is one of fundamental solutions to reduce any possible security accidents.

Software vulnerability analysis can be done using either a white box testing technique or a black box testing technique. Among those two types of techniques, black box testing is a type of testing that we enter a set of test data into the target software and observe the outputs. Using this type of testing technique is useful when we want to conduct our testing without much knowledge and understanding of the software internal logic.

Fuzzing is one of automated black-box testing techniques, which finds any implementation defects in software by providing invalid, unexpected, or random data in an automated fashion and monitoring the result. In this study, we design and implement a fuzzing tool that discovers the vulnerabilities of web browsers. Web browsers arethe most widely used software packages in web environment. Therefore, a vulnerability abuse in a web browser can cause much more extensive impact than any other web applications.

A web browser vulnerability detection in a fuzzing tool we propose is performed as follows: 1) collects web pages, 2) compromises templates, 3) detects parameters by analyzing the codes of template, 4) generates test cases by providing invalid input data defined in a fuzzing oraclefor parameters of templates, 5) injects test cases to web browser, and 6) verifies whether any vulnerability exists or not by monitoring the execution result.

Our approach has the following benefits compared to others: using templates in the test case generation allows the tool to be applied to a variety of languages and versions, and it can also be used to verify the vulnerabilities of plug-ins included in a web browser.