Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.3, Jun. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.3.535

웹 브라우저 취약성 검증을 위한 이벤트 및 커맨드 기반 퍼징 방법

박 성 빈,[1†] 김 민 수,[2] 노 봉 남[1‡]
[1]전남대학교 시스템보안연구센터, [2]목포대학교

Event and Command based Fuzzing Method for Verification of Web Browser Vulnerabilities

Seongbin Park,[1†] Minsoo Kim,[2] Bong-Nam Noh[1‡]
[1]Chonnam National University System Security Research Center,
[2]Mokpo National University

요 약

최근 소프트웨어 산업의 발전과 더불어 소프트웨어 취약점을 이용한 공격이 사회적으로 많은 이슈가 되고 있다. 특히, 웹 브라우저 취약점을 이용한 공격은 윈도우 보호메커니즘을 우회할 수 있기 때문에 주요 공격 대상이 될 수 있다. 이러한 보안위협으로부터 웹 브라우저를 보호하기 위한 퍼징 연구가 지속적으로 수행되어 왔다. 하지만 기존 웹 브라우저 퍼징 도구에서는 대부분 DOM 트리를 무작위로 변형시키는 단순한 형태의 퍼징 방법을 사용하고 있다. 본 논문에서는 알려진 취약점에 대한 패턴 분석과 기존 웹 브라우저 퍼징 도구의 분석을 통해 최신 웹 브라우저 취약점을 보다 효과적으로 탐지하기 위한 이벤트 및 커맨드 기반 퍼징 방법을 제안한다. 기존 퍼징 도구 3종과 제안하는 방법을 비교한 결과 이벤트 및 커맨드 기반의 퍼징 도구가 더욱 효과적이라는 것을 볼 수 있었다.

ABSTRACT

As the software industry has developed, the attacks making use of software vulnerability has become a big issue in society. In particular, because the attacks using the vulnerability of web browsers bypass Windows protection mechanism, web browsers can readily be attacked. To protect web browsers against security threat, research on fuzzing has constantly been conducted. However, most existing web browser fuzzing tools use a simple fuzzing technique which randomly mutates DOM tree. Therefore, this paper analyzed existing web browser fuzzing tools and the patterns of their already-known vulnerability to propose an event and command based fuzzing tool which can detect the latest web browser vulnerability more effectively. Three kinds of existing fuzzing tools were compared with the proposed tool. As a result, it was found that the event and command based fuzzing tool proposed was more effective.

Keywords: Fuzzing, Event, Command, Vulnerability, Web browser

I. 서 론

전 세계적으로 소프트웨어 산업과 네트워크 인프라가 발전함에 따라 다양한 분야에서 소프트웨어의 역할이 매우 중요해지고 있다. 이에 따라 악의적인 목적을 가진 사용자들은 소프트웨어에 존재하는 취약점을 이용한 공격을 통해 사회적으로 심각한 문제를 일으키고 있다. 특히, 웹 브라우저는 인터넷 사용에 꼭 필요한 소프트웨어로서 가장 널리 사용되고 있기 때문에 사이버 공격 또한 웹 브라우저의 취약점이나 웹 브라우저

접수일(2014년 5월 12일), 수정일(2014년 6월 16일), 게재확정일(2014년 6월 16일)
† 주저자, still2day@gmail.com
‡ 교신저자, bbong@jnu.ac.kr(Corresponding author)

플러그인의 취약점을 기반으로 이루어지고 있다[1].

IBM의 2013년 보고서[2]에 따르면 2012년 한 해 동안 웹 브라우저를 대상으로 한 공격 도구의 개발이 급증하였으며, 특히 자바(Java) 취약점을 이용한 공격 기능을 지원하고 있다고 보고되었다. 한편 파이어아이(FireEye)에서 예측한 2014년 보안 위협 동향[3]에서는 2013년 2월 이후로 새로운 자바 취약점 공격의 발생 빈도는 급감하였으며, 향후 웹 브라우저 기반의 공격이 증가할 것이라고 전망하였다. 그 이유는 웹 브라우저 취약점 기반의 공격이 웹 브라우저 상의 특정 프로세스가 사용하는 메모리 주소를 무작위로 부여하여 익스플로잇을 방어하기 위한 ASLR (Address Space Layout Randomization) 기법을 우회할 수 있기 때문이다. ASLR 우회 취약점은 임의의 코드를 실행할 수 있는 원격 코드 실행 취약점과 함께 사용될 수 있으므로 악성 해커들의 주요 공격 수단 중 하나로 사용될 수 있다.

이러한 웹 브라우저의 보안 취약점을 이용한 공격에 대응하기 위한 많은 연구들이 진행되어 왔다. 마이크로소프트사의 Godefroid[4]에 의해 처음 웹 브라우저에서의 퍼징에 관한 연구가 시작 되었다. 이는 문법 기반의 화이트박스 퍼징에 관한 내용으로 문법적으로 유효한 자원들을 대상으로 최대한의 커버리지(coverage)를 갖는 형태의 퍼징 방법을 제안함으로서 블랙박스 퍼징보다 효과적임을 보였다. 또한 이를 확장한 형태인 LangFuzz[5]는 언어에 독립적이면서 원본 테스트 케이스로부터 다양한 코드 조각을 통해 랜덤하게 조합하고, 변형하는 퍼징 방식을 제안하였다. LangFuzz는 문법 표준을 충족시키지 못하는 테스트 케이스의 경우, 브라우저 인터프리터에 의해 폐기되는 문제점이 존재한다. 최근 LangFuzz의 문제점을 해결하기 위해 Guo[6] 등에 의해 GramFuzz가 제안되었다. GramFuzz는 웹을 통해 수집한 샘플에서 HTML, CSS, 자바스크립트 내의 소스 코드 분석을 통해 문법 트리와 코드 조각에 대한 라이브러리를 구축하고, 테스트 케이스를 구조적으로 변형함으로서 퍼징을 수행한다. GramFuzz에서 제안하는 퍼징 방법은 최근 웹 브라우저 취약점의 주를 이루고 있는 이벤트 및 커맨드 관련 취약점을 탐지하기가 어렵다. 또한 퍼징 도구의 성능 평가 방법에 있어서 단순히 다른 종류의 크래시 발생 빈도를 평가의 척도로 사용하기는 무리가 있다.

본 논문에서는 기존의 퍼징 방법으로 발견하기 어려운 최신 웹 브라우저 취약점 탐지를 위해 이벤트 및 커맨드 기반의 웹 브라우저 퍼징 방법을 제안한다. 제안하는 방법은 기본적으로 DOM 이벤트 디스패치 메커니즘의 원리를 이용한다. 퍼징을 통해 변형된 DOM 트리에서 이벤트를 처리하는 원리를 이용하면 이미 메모리 해제된 객체에 접근하여 발생하는 Use-After-Free 형태의 오류에 대해서도 점검이 가능하다. 또한 이벤트 디스패치 메커니즘에 추가적인 커맨드를 실행하는 방법으로 퍼징의 복잡성을 높여 보다 발견하기 어려운 오류에 대해서도 점검이 가능하다.

II. 관련 연구

2.1 최신 웹 브라우저 취약점 패턴 분석

본 절에서는 최근에 발표되었던 웹 브라우저에 대한 취약점 패턴을 분석함으로서 최신 동향을 파악하고, 이러한 최신 취약점이 기존 웹 브라우저 퍼징 도구를 이용해 발견될 수 있는지에 대한 가능성을 살펴본다. 본 논문에서 분석한 최신 웹 브라우저 취약점은 Table 1과 같다.

Table 1. Recent Web Browser Vulnerabilities

CVE	Browser	Vulnerability
2013-3893	IE 6-11	SetMouseCapture
2013-3897	IE 8	CDisplayPointer

2.1.1 SetMouseCapture Use-After-Free[7]

이 취약점은 마이크로소프트사의 인터넷 익스플로러에서 특정 객체에 대한 마우스 캡처를 잃어버렸을 때 발생하는 'onLoseCapture' 이벤트가 처리되는 동안 SetMouseCapture 함수에서 이미 메모리 해제된 레퍼런스에 접근함으로서 발생하는 취약점이다. 해당 취약점에 대한 전체 메커니즘은 Fig.1과 같다.

먼저, 두 개의 엘리먼트(element)를 생성하여 body 객체의 자식 노드에 차례로 추가한다. 다음으로 'onLoseCapture' 이벤트를 설치하고, 최하위에 추가된 엘리먼트의 'outerText' 속성에 널(null) 값을 입력함으로서 추가되었던 두 개의 엘리먼트를 body 객체로부터 제거한다. 이 때, 각 엘리먼트에 대해 setCapture() 함수를 순서대로 호출하는데 두 번째 객체에 대한 setCapture() 함수가 호출되는 순간,

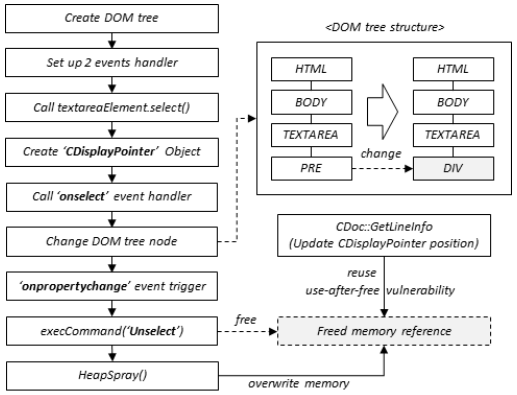


Fig.1. MS IE CDisplayPointer Use-After-Free Vulnerability Mechanism(CVE-2013-3897)

'onLoseCapture' 이벤트가 트리거(trigger) 된다. 이 이벤트는 해당 객체에 대한 마우스 캡처를 잃어버렸을 때 발생한다. 연결된 이벤트 핸들러는 document.write() 함수를 이용하여 임의의 메모리를 해제한다. 메모리가 해제된 레퍼런스는 계속해서 남아있게 되고, 이후에 CTreeNode::GetInterface 함수에 의해 재사용 됨으로서 Use-After-Free 오류가 발생하게 된다.

2.1.2 CDisplayPointer Use-After-Free[8]

이 취약점은 마이크로소프트사의 인터넷 익스플로러에서 객체 속성이 변경될 때 발생하는 'onPropertyChange' 이벤트가 처리되는 동안 CDisplayPointer 객체에서 이미 메모리 해제된 레퍼런스에 접근함으로서 발생하는 취약점이다. 해당 취약점에 대한 전체 메커니즘은 Fig.2와 같다.

CVE-2013-3893 취약점과 마찬가지로 먼저, 두 개의 엘리먼트를 생성하여 body 객체의 자식 노드에 차례로 추가한다. 다음으로 'onSelect' 이벤트와 'onPropertyChange' 이벤트를 설치하고, 첫 번째 자식 객체에 대한 select() 함수를 호출한다. select() 함수가 호출되면, textarea에 대한 CDisplayPointer 객체가 생성되고, 'onSelect' 이벤트 핸들러가 호출된다. 이 핸들러에서는 DOM 트리의 최하위 엘리먼트인 'pre'를 'div'로 변경한다. 결국 'onPropertyChange' 이벤트가 트리거 되어 해당 핸들러가 호출된다. 이 핸들러에서는 document.execCommand('Unselect') 함수를 호출함으로서 임의의 메모리를 해제한다. 하지만 메모

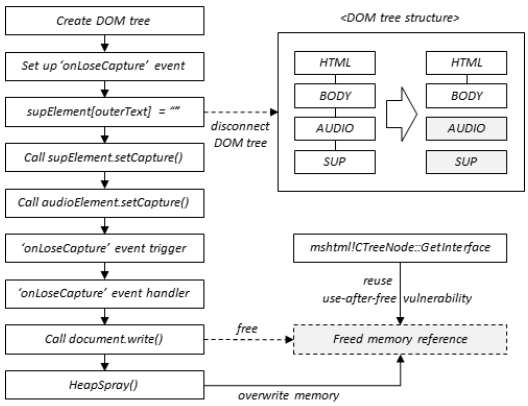


Fig.2. MS IE SetMouseCapture Use-After-Free Vulnerability Mechanism(CVE-2013-3893)

리가 해제된 레퍼런스는 계속해서 남아있게 되고, 이후에 CDisplayPointer 객체의 위치를 업데이트하기 위한 목적으로 CDoc::GetLineInfo 함수에 의해 재사용됨으로서 Use-After-Free 오류가 발생하게 된다.

2.2 기존 웹 브라우저 퍼징 도구

대표적인 기존 웹 브라우저 퍼징 도구는 퍼징 방법에 따라서 Table 2와 같이 분류할 수 있다. 퍼징 방법은 크게 생성 기반 퍼징 방법(generation based fuzzing method)과 변형 기반 퍼징 방법(mutation based fuzzing method), 그리고 두 방식을 결합한 하이브리드 기반 퍼징 방법(hybrid based fuzzing method)으로 나눌 수 있다.

생성 기반 퍼징 방법은 퍼징 대상에 대한 정보를 알고, 정확한 목표를 선정하여 적절한 퍼징을 수행하기 때문에 일반적으로 변형 기반 퍼징 방법에 비해 높은 커버리지를 갖는다. 하지만 입력 데이터에 대한 포맷 정보를 미리 알고 있어야하기 때문에 해당 정보를 모를 경우에는 퍼징이 불가능하다. 반면, 변형 기반 퍼징 방법은 퍼징 입력 데이터의 정보가 없더라도 퍼징이 가능하며, 생성 기반 퍼징 방법에 비해 손쉽게 퍼

Table 2. Existing Web Browser Fuzzing Tools

Fuzzing tools	Rele-ased	Fuzzing method
Hamachi	2006	Generation based
Ref_fuzz	2008	Mutation based
Nduja	2013	Hybrid based

징이 가능하다. 하지만 변형 방식의 정확도나 다양성을 충족하기 쉽지 않기 때문에 생성 기반 퍼징 방법에 비해 커버리지가 낮다는 단점이 있다. 이러한 두 가지 퍼징 방법의 장점을 결합한 하이브리드 기반 퍼징 방법은 비교적 높은 커버리지를 가지며, 무작위적인 변형을 통해 예측하기 어려운 상황들을 다양하게 이끌어 낼 수 있다. 따라서 본 논문에서는 이러한 세 가지 퍼징 방법을 이용한 대표적인 퍼징 도구인 Hamachi, Ref_fuzz, Nduja를 각각 선정하였다.

2.2.1 Hamachi

Hamachi[9]는 메타스플로잇(metasploit)의 창시자인 H.D.Moore와 Aviv Raff에 의해 2006년에 만들어진 웹 브라우저 퍼징 도구이다. 이 도구에서는 크래시를 유발할 가능성이 있는 값들을 미리 정의하여 엘리먼트(element) 메소드의 인자와 속성 값에 주입함으로서 일반적인 DHTML 구현에 대한 취약점을 찾아낸다. Hamachi의 특징은 최초에 브라우저를 탐지하고, 생성 가능한 모든 객체를 DOM 트리에 연결하여 퍼징에 대한 커버리지를 높인다는 점이다. 전체적인 퍼징 메커니즘은 Fig.3과 같다.

Hamachi에서의 퍼징은 초기 설정 단계와 실제 퍼징 단계로 나누어진다. 초기 설정 단계에서는 먼저 브라우저를 식별함으로서 해당 브라우저에 대한 Div 엘리먼트 레퍼런스를 설정한다. 다음으로 임의의 문자열 및 정수 값에 대한 배열을 생성 및 저장하며, DHTML 태그 리스트와 메소드, 그리고 속성들에 대한 리스트를 배열에 저장한다. 실제 퍼징 단계에서는 미리 정의된 DHTML 태그 엘리먼트 배열을 이용하여 DOM 트리를 생성하고, 생성된 DOM 트리의 각 노드 엘리먼트들에 대한 메소드와 속성에 임의의 값을 입력한다. 마지막으로 설정한 메소드를 호출하거나 속성에 새로운 값을 다시 입력함으로서 해당 페이지에 대한 오류 발생 여부를 확인한다. 이러한 작업은 생성된 DOM 트리의 모든 노드를 순회할 때까지 반복적으로 수행되며, 모든 노드에 대한 순회가 끝나면 퍼징이 완료된다.

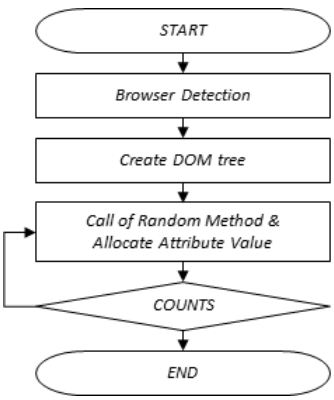


Fig.3. Hamachi Fuzzing Mechanism

2.2.2 Ref_fuzz

Ref_fuzz[10]는 구글의 보안 전문가인 Michal Zalewski에 의해 2008년에 만들어진 웹 브라우저 퍼징 도구이다. 이 도구는 DOM 바인딩을 이용하여 자바스크립트로 작성되었으며, DOM 객체에 대한 레퍼런스를 이용하여 퍼징을 수행한다. 퍼징 방식은 DOM 트리를 탐색하여 레퍼런스를 수집하고, 다양한 방법을 이용하여 객체의 메모리를 해제한 다음, 수집된 함수 타입의 레퍼런스를 호출함으로서 Use-After-Free 오류에 대한 취약점을 찾아낸다. 전체적인 퍼징 메커니즘은 Fig.4와 같다.

Ref_fuzz에서의 퍼징은 재귀적으로 레퍼런스를 수집하고 이를 재사용하는 것이 핵심이다. 최초에 DOM 트리를 생성하고, 레퍼런스를 수집한다. 레퍼런스 수집은 해당 객체에 대한 속성 타입과 함수 타입으로 나누어지며, 함수 타입의 레퍼런스가 수집될 경우, 이를 강제적으로 호출한다. 레퍼런스의 수집이 완료되면 객체의 'innerHTML' 속성이나, 'delete' 키워드 등을 이용하여 메모리 해제 작업을 수행한다. 이

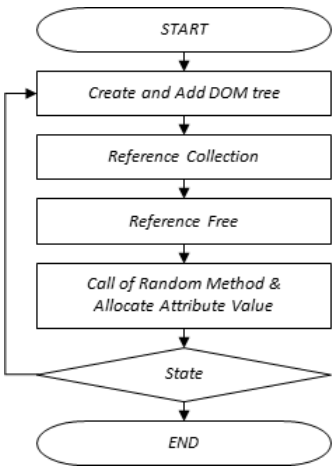


Fig.4. Ref_fuzz Fuzzing Mechanism

후, 이미 메모리 해제된 객체를 재사용하도록 유도하기 위해 수집한 레퍼런스를 이용하며, 속성에는 임의의 값을 입력하고, 함수 타입의 레퍼런스는 강제로 호출함으로서 크래시를 유발시킨다. 이러한 과정은 사용자가 지정한 횟수만큼 반복하여 수행되며, 반복 과정에서 지속적으로 새로운 객체를 생성함으로서 DOM 트리에 추가적인 퍼징이 이루어진다.

2.2.3 Nduja

Nduja[11]는 Rosario Valotta에 의해 DeepSec 2012에서 발표된 Grinder[12] 기반의 웹 브라우저 퍼징 도구이다. 기존의 대부분 웹 브라우저 퍼징 도구에서는 DOM 레벨 1 또는 2에 대한 API를 이용한 퍼징을 수행한 반면, Nduja에서는 DOM 레벨 3에 해당하는 이벤트 요소를 추가로 활용하여 퍼징 도구를 구현하였다. Nduja의 특징은 DOM 레벨 3까지 활용함으로서 좀 더 복잡하고, 새로운 형태의 취약점을 발견하고자 한다. 전체적인 퍼징 메커니즘은 Fig.5와 같다.

Nduja의 주요 퍼징 알고리즘은 기본적으로 초기화 단계와 크롤링 및 DOM 트리 변형 단계, 이벤트 트리거 단계 등으로 이루어진다. 초기화 단계에서는 DOM 트리를 생성하고, 각 노드에 대한 속성에 임의의 값을 입력한다. 그리고 DOM 레벨 1의 노드 이터레이터(iterator), 레벨 2의 트리워커(tree walker), 레벨 3의 이벤트 리스너(listener)를 각각 설치한다. 크롤링 단계에서는 생성된 DOM 트리를 순회하며 무작위로 변형을 수행한다. 만약 설치한 이벤트가 발생할 경우, 해당 이벤트에 대한 핸들러에서는 다양한 변형 작업이 이루어짐으로서 크래시를 발생시키도록 한다. 이러한 퍼징 과정은 사용자가 임의로 종료할 때까지 페이지 새로 고침을 통해 반복적으로 수행된다.

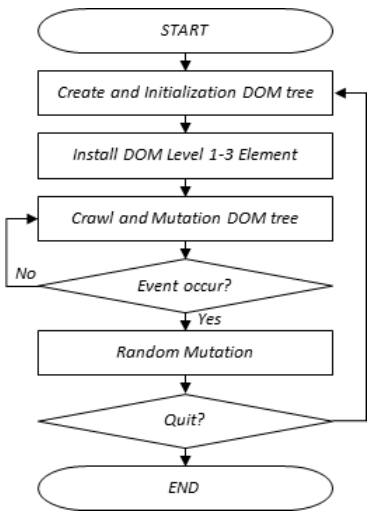


Fig.5. Nduja Fuzzing Mechanism

2.3 기존 도구의 문제점 및 한계점

최근에는 앞에서 분석한 최신 웹 브라우저 취약점과 같은 웹 브라우저에서 사용되는 다양한 이벤트를 이용한 방식의 Use-After-Free 취약점이 주로 발견되고 있다[13]. 하지만 기존 웹 브라우저 퍼징 도구에서는 최근에 발견되고 있는 웹 브라우저 취약점형태를 점검하는 데 몇 가지 한계점과 문제점들이 존재한다.

Hamachi는 이미 오래전에 나온 퍼징 도구로서 단순히 DOM의 구조적인 문제로 발생할 수 있는 취약점 형태만을 점검하기 때문에 최근 발견되고 있는 이벤트를 기반한 취약점과 같은 복잡한 취약점에 대해서는 점검이 불가능하다. Ref_fuzz는 Hamachi 보다는 발전된 형태의 퍼징 도구로서 레퍼런스를 수집하여 재사용하는 방식을 이용하지만, 재사용을 유도하는 과정에서 DOM 트리의 복잡하고 임의적인 변형이 부족하기 때문에 마찬가지로 복잡한 취약점에 대해서 점검이 불가능한 한계점이 존재한다. 마지막으로 Nduja는 이벤트를 이용한 퍼징 방식을 도입하여 복잡성을 높였으나 이는 이벤트가 발생한 경우에 처리하는 과정에 대해서만 복잡성을 높인 것으로 이벤트 발생 자체의 과정이 단조롭다.

이에 본 논문에서 제안하는 퍼징 도구 또한 기존 퍼징 도구의 문제점을 해결하고, 최근에 발견되고 있는 취약점 형태에 대해서도 점검할 수 있는 최적화된 방식의 퍼징 방법을 제안한다.

III. 제안하는 웹 브라우저 퍼징 방법

본 논문에서는 이벤트 및 커맨드 기반의 웹 브라우저 퍼징 방법을 이용한 EnC_fuzz를 제안한다.

3.1 DOM 이벤트 디스패치 메커니즘

DOM(Document Object Model)[14]은 객체지향 모델로서 구조화된 문서를 표현하는 방식이다. 이러한 DOM은 HTML 문서의 요소를 제어하기 위

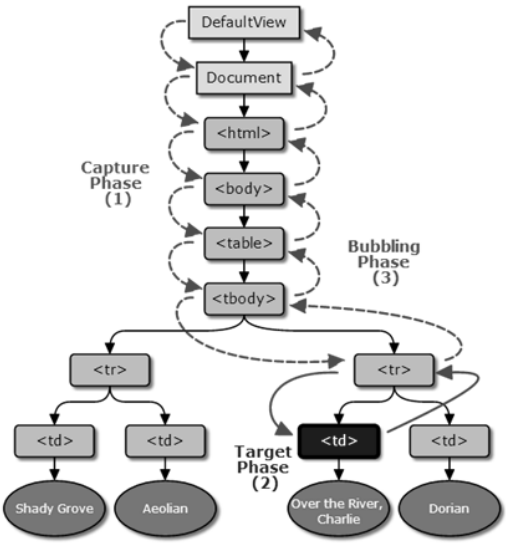


Fig.6. DOM Level 3 Event Dispatch Mechanism[15]

해 웹 브라우저에서 처음으로 지원되었으며, 동적으로 문서의 내용, 구조, 스타일 등에 접근하거나 변경하기 위한 수단으로 사용된다. 이러한 DOM의 규격은 1단계부터 3단계까지 존재하며, 3단계에서는 확장된 형태의 이벤트 기능을 제공한다. 이러한 DOM에서의 이벤트 디스패치 메커니즘은 Fig.6과 같다.

DOM 레벨 3 이벤트 디스패치 메커니즘은 캡처(capture), 타겟(target), 버블(bubble) 단계로 이루어진다. 첫 번째, 캡처 단계에서는 DOM 트리의 루트 노드로부터 이벤트가 설치된 대상 노드까지 탐색해 나가는 과정이다. 두 번째, 타겟 단계는 대상 노드에 도달한 순간을 의미하며, 마지막, 버블 단계는 대상 노드에서 루트 노드까지 다시 돌아가는 과정이다. 이러한 순회(travel) 과정은 DOM 트리에서 이미 계산된 경로를 통해 전파되며, 만약 각 단계에서 이벤트가 설치된 노드를 만나게 되면, 그에 해당하는 이벤트 핸들러가 호출된다. 각 단계에서 특정 이벤트에 대한 핸들러가 호출되는 과정은 Fig.7과 같다.

임의의 DOM 트리에는 이벤트를 탐색하고, 그에 대한 핸들러를 호출하기 위한 이벤트 리스너 맵(event listener map)이 존재한다. 이러한 이벤트 리스너 맵은 대상 노드에 도달하기 전인 캡처 단계에서는 임의로 변경이 가능하다. 하지만 대상 노드에 도달하게 되는 타겟 단계에서는 변경이 불가능하다. 만약 이때, DOM 트리의 순회 경로에 존재하는 노드가 이동하거나 제거되어도 순회 경로는 변경되지 않고 그

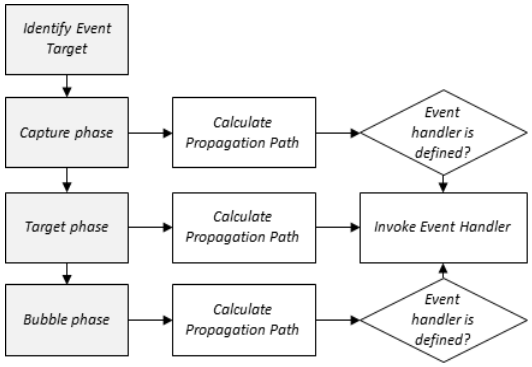


Fig.7. DOM Level 3 Event Handler Algorithm[11]

대로이다.

따라서 한 번 결정된 이벤트 순회 경로 내에 존재하는 노드를 이동시키거나 제거하는 등의 변형을 하게 된다면 해당 DOM 트리를 렌더링(rendering)하는 브라우저 엔진에서는 충분히 문제가 발생할 수 있을 것이다. 또한 이러한 변형은 이벤트 핸들러가 호출되어 처리하는 도중에 다른 이벤트가 발생할 수 있기 때문에 예측하기 어려운 문제들이 발생할 가능성이 있다.

3.2 이벤트 및 커맨드 기반 퍼징 메커니즘

본 논문에서 제안하는 이벤트 및 커맨드 기반 퍼징 메커니즘의 전체 동작 과정은 Fig.8과 같다.

이벤트 및 커맨드 기반 퍼징 메커니즘의 원리는 기본적으로 DOM 이벤트 디스패치 메커니즘의 원리를 기반한다. DOM 트리에서 특정 노드의 엘리먼트에 설치된 이벤트가 발생하면 해당 노드에 이벤트가 디스패치되는 과정은 3.1절에서 설명한 바와 같이 세 가지로 구성된다. 제안하는 이벤트 및 커맨드 기반 퍼징 메커니즘은 이벤트 디스패치 과정에서 복잡성을 높여 예외를 강제적으로 발생시킨다. 만약, 특정 노드에서 이벤트 핸들러가 처리되는 동안 DOM 트리가 변형되어 임의의 노드에 대한 객체가 해제된다면 이벤트 디스패치 과정의 버블링(bubbling) 단계에서는 이미 메모리가 해제된 객체에 접근하게 될 가능성이 있으며 이 경우 Use-After-Free 오류가 발생할 수 있다. 또한 이벤트 디스패치 과정에서 커맨드의 실행은 보다 복잡한 상황을 만든다.

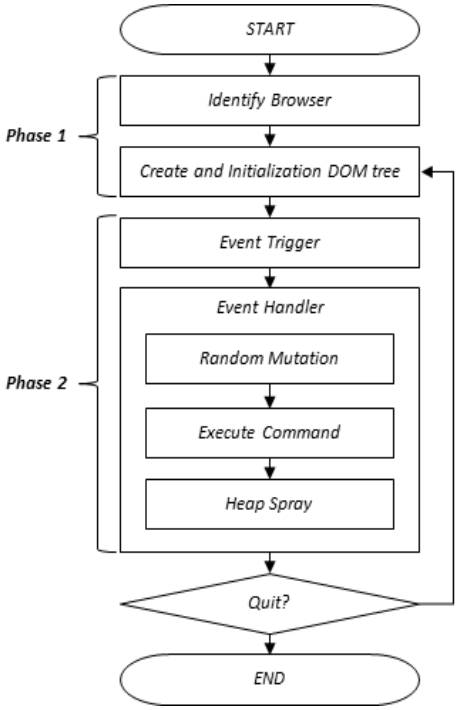


Fig.8. Event and Command based Fuzzing Mechanism

3.2.1 Phase 1 - 브라우저 식별 및 DOM 트리 초기화

제안하는 퍼징 도구는 첫 번째 단계에서 퍼징을 수행하고자 하는 웹 브라우저의 식별과 DOM 트리의 초기화 과정을 수행한다. 웹 브라우저의 식별은 실제로 퍼징 과정에서 사용되는 자바스크립트 메소드가 문법 오류를 발생시켜 퍼징을 중단시키는 문제를 사전에 방지한다. DOM 트리 초기화 과정은 기존 웹 브라우저 퍼징 도구인 Nduja에서는 기본적으로 주어진 DOM 트리에서 임의의 엘리먼트를 생성하여 가장 최상위 노드인 〈html〉부터 최하위 노드까지 무작위로 선택하여 생성된 엘리먼트를 해당 노드의 자식 노드로 추가하였다. 이러한 DOM 트리 생성 방식은 종종 구조적인 문제로 인해 퍼징이 중단되는 문제점이 존재한다.

본 논문에서 제안하는 도구에서는 이러한 문제를 해결하기 위해 〈body〉 노드를 탐색함으로서 해당 노드의 하위에서만 랜덤하게 DOM 트리를 구성하도록 설계하였다. 또한 기본적인 엘리먼트 뿐만 아니라 스타일시트(style-sheet)도 퍼징 요소로 사용하였으며, 최종적으로 생성된 DOM 트리의 각 노드에 연결된 엘리먼트의 속성에는 버퍼 오버플로(buffer overflow)나 기타 오류를 일으킬 가능성이 있는 임의의 값을 입력한다. 마지막으로 각 엘리먼트에 임의의 이벤트를 설치함으로서 DOM 트리의 초기화 단계가 완료된다.

3.2.2 Phase 2 - 이벤트 트리거 및 커맨드 실행

두 번째 단계에서는 DOM 트리의 각 노드에 설치된 이벤트를 강제로 발생시키며, 발생한 이벤트에 대한 핸들러를 처리하는 동안 임의의 커맨드를 호출한다. 또한 초기 DOM 트리를 다양한 방법을 통해 변형함으로써 트리 구조에 연결된 다양한 DOM 객체들의 생성 및 제거 과정에서 예기치 못한 상황이 발생하도록 만든다. 변형 방법에는 이미 생성된 DOM 트리 객체에 대한 레퍼런스를 이용하여 임의의 값으로 변경하는 방법, DOM 트리에 자식 노드를 추가하거나 제거하는 방법, 특정 노드의 하위 객체를 모두 해제하는 방법, 범위(range) 객체를 이용한 변형 방법을 사용하고 있다. 초기 DOM 트리에 대한 변형을 모두 수행한 뒤에는 execCommand 메소드를 이용하여 임의의 커맨드를 실행함으로서 예외 상황이 발생하도록 유도한다. 이 과정에서 새로운 이벤트가 발생하여 기존의 이벤트 핸들러에 대한 처리 과정이 잠시 중단될 수도 있다.

마지막으로 앞에서 변형된 DOM 트리에 의해 메모리가 해제된 이후 다시 사용되는 객체가 존재할 경우, 메모리 접근 오류가 발생하게 될 것이다. 만약 이렇게 재사용되는 메모리가 발견된다면, 일련의 분석을 통해 원하는 메모리 영역으로 실행 흐름을 변경할 수 있는 Use-After-Free 취약점으로 발전시킬 수 있을 것이다. 이 때, 원하는 코드를 삽입하기 위해 힙 스프레이(heap-spray)[16] 기법을 사용하고, 실제로 힙 스프레이를 통해 많은 데이터를 주입했을 때, 브라우저의 크래시가 더 많이 발생한다는 것을 실험을 통해 확인할 수 있었다.

3.3 EnC_fuzz 구현 환경 및 방법

EnC_fuzz는 Nduja와 마찬가지로 Grinder를 기반으로 구현하였으며, 전체 구성은 Fig.9와 같다. Grinder[12]는 노드(node)와 서버(server)로 구성된 루비(ruby) 기반의 웹 브라우저 퍼징 프레임워크이다. 노드의 역할은 퍼징 대상 웹 브라우저를 실행하고, 브라우저 프로세스에 로깅 모듈을 주입함으로서

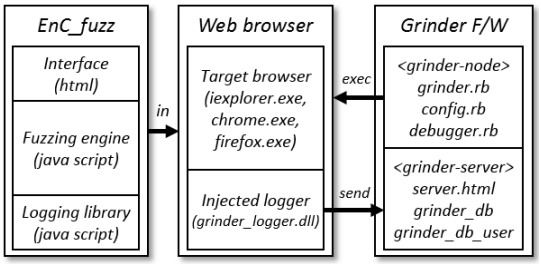


Fig.9. Enc_fuzz's Implementation Environments and Whole Components

퍼징 과정에서 발생하는 로그 정보를 전송한다. 설정 파일을 통해서는 로그 수신을 위한 서버 정보, 로그 저장 경로, 브라우저 실행 경로 등의 퍼징에 필요한 설정 정보를 정의하며, 디버거 모듈은 대상 브라우저 프로세스를 후킹하고, WinDbg API를 사용하여 퍼징 과정에서 크래시 발생 여부를 판단한다. 서버는 실제 퍼징을 위해 입력으로 주어진 HTML 문서를 로드하고, 퍼징 중에 발생한 로그 정보를 데이터베이스에 저장하는 역할을 수행한다.

EnC_fuzz는 Grinder 서버에서 퍼징을 수행하기 위한 HTML 문서와 자바스크립트로 작성된 퍼징 엔진, 로그 생성을 위한 로깅 모듈로 구성된다. 퍼징 엔진에서는 실제 퍼징 작업을 위한 코드도 중요하지만, 크래시 발생 시 이를 재연하기 위한 구문을 로그로 남겨주는 작업 또한 매우 중요하다. 예를 들어, 동적으로 생성되는 DOM 객체의 종류나 이벤트 및 커맨드의 종류 등을 정확하게 로깅하는 작업 등이 이에 해당된다.

IV. 실험 및 결과

본 장에서는 제안한 EnC_fuzz 퍼징 도구와 기존 퍼징 도구 3종을 이용하여 실제 퍼징 실험을 수행하고, 그 결과에 대한 내용을 분석한다.

4.1 웹 브라우저 퍼징 실험 방법

웹 브라우저 퍼징 실험 방법으로 먼저 퍼징 환경을 구축하고, 대상 웹 브라우저의 선정 및 평가 계획에 따라 각 퍼징 도구에 대한 실험을 수행하였다. 실제 퍼징에 활용한 시스템 사양은 Table 3과 같고, 퍼징 실험 환경의 전체 구성은 Fig.10과 같다.

전체적인 퍼징 실험 환경은 각 도구에 대한 호스트 머신과 퍼징 대상 웹 브라우저 4종에 대한 가상 머신

Table 4. System Specification of Experimental Environments

	Host	Guest VM
OS	Win 7(64bit)	Win XP/7(32bit)
S/W	VMware	Grinder
CPU	Intel Core i7	Intel Core i7
RAM	8.00GB	512MB/1.00GB

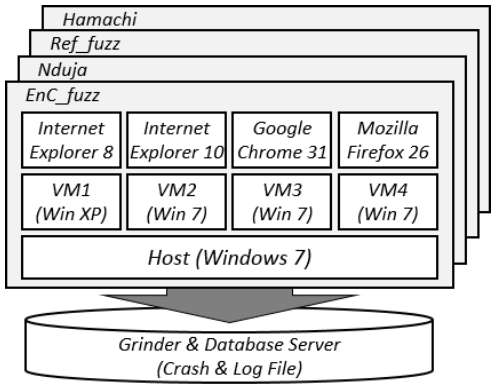


Fig.10. Whole Components of Experimental Environments

들로 이루어지며, 퍼징 수행 결과로 발생하는 크래시 및 로그 파일 정보를 저장하기 위한 Grinder 및 데이터베이스 서버가 존재한다.

퍼징 대상 웹 브라우저는 최근 가장 많이 사용되고 있는 인터넷 익스플로러 8, 10, 구글 크롬 31, 모질라 파이어폭스 26 버전을 선정하였으며, 해당 버전들은 실험 당시 가장 최신 버전을 대상으로 하였다. 인터넷 익스플로러 8의 경우 윈도우 XP에서 사용 가능한 가장 높은 버전으로서 최근까지도 윈도우 XP 환경에서의 취약점이 지속적으로 발표되고 있다[17]. 또한 최근 윈도우 XP에 대한 지원이 종료됐음에도 불구하고, 글로벌IT시장조사업체인 스탯카운터 자료에 따르면, 지난해 5월까지 30% 이상이 윈도우 XP를 사용하고 있는 것으로 나타났다[18]. 따라서 본 연구에서는 윈도우 XP 환경에서의 인터넷 익스플로러 8 버전을 퍼징 대상으로 추가하였다.

구축한 각 시스템에서 퍼징 도구별로 2주간 퍼징을 수행하였고, 이 때 발생한 크래시의 개수를 평가의 척도로 이용하였다. 더 자세하게 크래시는 취약성 관점에서 단순한 메모리 접근 오류와 프로그램 카운터 손상으로 인해 실행흐름이 변경될 수도 있는 심각한 오류로 나눌 수 있다. 본 논문에서는 후자의 심각한

Table 5. 2 Weeks of the Experimental Results Using EnC_fuzz with Three Kinds of Existing Fuzzing Tools

Guest OS	Windows XP	Windows 7			Total
Fuzzing Tools	Internet Explorer 8	Internet Explorer 10	Google Chrome 31.0	Mozilla Firefox 26.0	
Hamachi	0	0	0	0	0
Ref_fuzz	32	12	6	0	50
Nduja	74	224	27	207	532
EnC_fuzz	368	32	646	318	1,364

오류로 분류될 수 있는 크래시에 한해서만 다루도록 한다. 그 이유는 실행흐름의 변경 가능성이 존재한다는 것은 실제로 공격자가 힙 스프레이 공격 등을 통해 원하는 코드를 주입한 후, 해당 코드로 실행흐름을 변경할 수 있는 가능성이 높기 때문이다.

4.2 웹 브라우저 퍼징 실험 결과

퍼징 도구별 퍼징 실험 결과는 Table 4와 같다. Hamachi 실험에서는 단순히 DOM 트리 구조의 결함만을 찾는 퍼징 도구로서 최신 웹 브라우저에서는 한 개의 크래시도 발생하지 않았다. Ref_fuzz 실험에서는 인터넷 익스플로러 8.0에서 32개의 크래시가 발생하였지만, 파이어폭스에서는 한 개의 크래시도 발생하지 않았다. Nduja 실험에서는 인터넷 익스플로러 10에서 가장 많은 224개의 크래시가 발생하였으며, 크롬에서는 가장 적은 27개의 크래시가 발생하였다. 그리고 본 논문에서 제안한 이벤트 및 커맨드 기반의 퍼징 도구인 EnC_fuzz 실험에서는 인터넷 익스플로러 10을 제외한 모든 웹 브라우저에서 가장 많은 크래시가 발생하였으며, 총 크래시 개수에서 기존 퍼징 도구와 비교하여 최소 2.5배에서 최대 27배에 달하는 크래시가 발생한 것을 확인할 수 있었다. 인터넷 익스플로러 10에서는 Nduja의 DOM 레벨 1과 2에 대한 추가적인 퍼징 요소로 인해 EnC_fuzz보다 많은 수의 크래시가 발견된 것으로 보인다. 또한 인터넷 익스플로러 10에서 사용하는 트라이던트(trident) 기반 렌더링 엔진과 크롬 28 이후부터 사용하는 블링크(blink) 기반 렌더링 엔진의 구조적 차이가 Nduja와 EnC_fuzz의 퍼징 실험 결과에 영향을 미치지 않았는지 추정해 본다. 하지만 이러한 Nduja의 높은 퍼징 복잡도는 퍼징 수행 결과 로그에 대한 분석에 상당한 어려움이 따를 수 있으며, 심각하게는 크래시를 발견했음에도 불구하고 해당 취약점에 대한 검증이 불가능할 수 있다는 단점이 존재한다.

EnC_fuzz는 Nduja 이전의 퍼저들에 비해 퍼징의 복잡도를 높여 크래시 발생율을 높인 반면, 이벤트 및 커맨드 기능에 선택과 집중을 한 결과, Nduja에 비해서는 보다 심플한 결과를 얻어 취약점 검증에 드는 시간적 비용을 줄일 수 있다는 장점이 존재한다. 물론 인터넷 익스플로러 10에서의 실험과 같은 경우에 퍼징의 다양성이 아직은 조금 미흡하다는 단점 또한 존재한다.

V. 결론 및 향후 연구

본 논문에서는 최신 웹 브라우저에 대한 취약점을 탐지하기 위해 이벤트 및 커맨드 기반의 웹 브라우저 퍼징 방법을 제안하였다. 관련 연구로는 최근 발생한 웹 브라우저 취약점 2종에 대한 패턴 분석과 기존에 연구되었던 3종의 웹 브라우저 퍼징 도구들을 분석하였다. 그 결과 기존 웹 브라우저 퍼징 도구들에 대한 문제점과 한계점을 발견할 수 있었다.

본 논문에서는 이러한 문제점과 한계점을 극복하기 위해 DOM 이벤트 디스패치 메커니즘의 특징과 커맨드 기능을 이용한 특화된 형태의 웹 브라우저 퍼징 도구인 EnC_fuzz를 제안하였다. 또한 이 퍼징 도구는 웹 브라우저 식별 기능을 통해 각 브라우저 엔진에 대한 DOM 메소드를 호출함으로서 범용성을 갖추었다. 이렇게 구현된 퍼징 도구는 실제로 기존 퍼징 도구들과의 비교 실험을 통해 보다 좋은 결과를 얻을 수 있었다. 하지만 인터넷 익스플로러 10에서 기존 퍼징 도구인 Nduja보다 좋지 않은 결과를 보임으로서 제안한 방식의 퍼징 도구에도 한계가 존재함을 알 수 있었다.

따라서 향후에는 이러한 한계를 극복하기 위해서 이벤트 및 커맨드 기반의 퍼징 방식이 인터넷 익스플로러 10에서 다른 브라우저보다 크래시 발생률이 저조한 원인을 파악하기 위한 분석이 필요할 것으로 보인다. 그 결과 더욱 향상된 성능의 웹 브라우저 퍼징 도구를 구현할 수 있을 것으로 기대한다.

References

〔1〕 Minho Kim, Seongbin Park, Jino Yoon, Minsoo Kim, and Bong-Nam Noh, "File Analysis Data Auto-Creation Model For Peach Fuzzing", Journal of the KITS. vol. 24, no. 2, pp. 327-333, Apr. 2014.

〔2〕 IBM, "IBM X-Force 2012 Trend and Risk Report", http://www.ibm.com/ibm/files/I218646H25649F77/Risk_Report.pdf, IBM Security Systems, Mar. 2013.

〔3〕 FireEye, "FireEye, 2014 Forecast Report of Security Threat", FireEye Korea, http://www.fireeye.com/kr/ko/news-events/press-releases/read/fireeye-2014-secyrity-threat-forecase-report, Nov. 2013.

〔4〕 Godefroid, Patrice, Adam Kiezun, and Michael Y. Levin. "Grammar-based whitebox fuzzing." ACM SIGPLAN Notices. Vol. 43. No. 6. ACM, 2008.

〔5〕 Holler, Christian. "Grammar-based interpreter fuzz testing." Master's Thesis Dissertation, Department of Computer Science, Saarland University, Jun. 2011.

〔6〕 Tao Guo, Puhan Zhang, Xin Wang, and Qiang Wei, "GramFuzz: Fuzzing Testing of Web Browsers Based on Grammar Analysis and Structural Mutation", ICIA 2013 Second International Conference, pp. 212-215, IEEE, Sep. 2013.

〔7〕 Exploit-DB, "MS13-080 Microsoft Internet Explorer CDisplayPointer Use-After-Free", http://www.exploit-db.com/exploits/28974/, Oct. 2013.

〔8〕 Exploit-DB, "Micorosft Internet Explorer SetMouseCapture Use-After-Free", http://www.exploit-db.com/exploits/28682/, Oct. 2013.

〔9〕 Bugzilla, "Bug 332602 - crashes found by hamachi fuzzer at metasploit", https://bugzilla.mozilla.org/show_bug.cgi?id=332602, 〔Attachment〕 https://bug332602.bugzilla.mozilla.org/attachment.cgi?id=217061, Apr. 2006.

〔10〕 Lcamtuf, "Announcing ref_fuzz, a 2 year old fuzzer", Lcamtuf's blog, http://lcamtuf.blogspot.kr/2010/06/announcing-reffuzz-2yo-fuzzer.html, Jun. 2010.

〔11〕 Rosario Valotta, "Taking Browsers Fuzzing To The Next (DOM) Level", DeepSec 2012.

〔12〕 Grinder, https://github.com/stephenfewer/grindjer.

〔13〕 Danielle Anne Veluz, "Zero-day Exploit Hits All Versions of Internet Explorer", Trend Micro, Last visited 2014-02-15.

〔14〕 Wikipedia, "Document Object Model", http://en.wikipedia.org/wiki/Document_Object_Model, Apr. 2013.

〔15〕 W3C, "DOM Event Architecture", http://www.w3.org/TR/DOM-Level-3-Events/#dom-event-architecture, Nov. 2013.

〔16〕 Wikipedia, "Heap spraying", http://en.wikipedia.org/wiki/Heap_spraying, Jan. 2014.

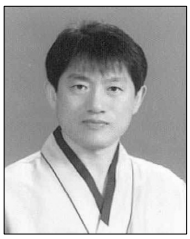
〔17〕 CVEDetails, "Microsoft Internet Explorer : List of security vulnerabilities", http://www.cvedetails.com/vulnerability-list/vendor_id-26/product_id-9900/Microsoft-Internet-Explorer.html, Jun. 2014.

〔18〕 StatCounter, "StatCounter Global Stats - Top 7 Desktop OSs in South Korea from Jan 2012 to May 2014", http://gs.statcounter.com/#desktop-os-KR-monthly-201201-201405, Jun. 2014.

〈저 자 소 개〉

박 성 빈 (Seongbin Park) 학생회원
2011년 2월: 전남대학교 전자컴퓨터공학부 공학사
2013년 2월: 전남대학교 정보보안협동과정 이학석사
2013년 3월~현재: 전남대학교 정보보안협동과정 박사과정
〈관심분야〉 취약점 분석, 악성코드 탐지, 시스템 보안

김 민 수 (Minsoo Kim) 종신회원
1993년: 전남대학교 전산통계학과 (이학사)
1995년: 전남대학교 전산통계학과 (이학석사)
2000년: 전남대학교 전산통계학과 (이학박사)
2000년~2001년: 한국인터넷진흥원 선임연구원
2001년~2004년: 전남대학교 연구교수
2005년~현재: 목포대학교 정보보호학과 부교수
〈관심분야〉 침입탐지, 디지털 포렌식, 데이터마이닝, 악성코드 분석



노 봉 남 (Bongnam Noh) 종신회원
1987년: 전남대학교 수학교육과 (이학사)
1982년: KAIST 전산학과 (이학석사)
1994년: 전북대학교 전산과 (이학박사)
1983년~현재: 전남대학교 전자컴퓨터공학부 교수
2000년~현재: 시스템보안연구센터 소장
〈관심분야〉 디지털 포렌식, 시스템 및 네트워크 보안, 정보사회와 사이버 윤리