브라우저 확장 프로그램을 활용한 웹 취약점 분석 방안

정준영*, 구도훈*, 김종민*, 김주원*, 이주명*, 이상현*, 이강석**, 최지헌***

*차세대 보안리더 양성 프로그램 (BoB) 10기 (교육생)

**차세대 보안리더 양성 프로그램 (BoB) 10기 (멘토)

***월간 해킹

Web vulnerability analysis method using browser extension

Joonyoung Jeong*, Dohun Koo*, Jongmin Kim*, Joowon Kim*, Joomyeong Lee*, Sanghyeon Lee*, Gangseok Lee**, Jiheon Choi***

*KITRI Best of the Best 10th (Mentee)

**KITRI Best of the Best 10th (Mentor)

***Monthly Hacking Inc.

요 약

종래의 웹 취약점 분석을 위한 자동화 스캔 도구는 일반적으로 Crawler, Fuzzer를 통한 분석으로 결과를 출력하여 취약점이 있는지 검증한다. 다만 Fuzzing을 통해 과도한 트래픽으로 서버의 가용성을 위협하는 문제를 유발하였고 취약점 오탐으로 수동 분석을 필요로 한다. 자동 스캔 도구의 문제를 해결하기 위해서 가용성에 문제가 되는 Fuzzer 부분을 제거하고 Crawling을 통해 수집한 정보를 기반으로 공격 벡터를 제시한다면 사용자는 정리된 형태의 내용을 바탕으로 수동 분석을 편리하게 진행할 수 있다. 본 논문에서는 수동 분석을 진행해야 하는 상황을 가정하여 브라우저 확장 프로그램을 활용한 웹 취약점 분석 방안을 제안한다.

I. 서론

오늘날 우리는 웹 애플리케이션을 블로그, 소셜 네트워크, 웹 메일, 은행 등 서비스 형태로 개인부터 큰 조직까지 사용하고 있다 [1]. 이러한 웹의 취약점을 분석하기 위해 쓰이는 웹 스캐너는 Crawler*, Fuzzer†, ‡ Analyzer로 이루어져 있다 [2].

종래의 자동화 웹 스캐너의 경우, Fuzzing 기법을 활용해 과도한 트래픽으로 서버의 가용성을 해칠 가능성이 있다. 또한, 제시된 취약점의 오탐 검증이 필요하거나 과도한 트래픽 발생 시 WAF(Web Application Firewall)에서 IP를 차단하는 경우, 수동으로 웹 취약성 분석을 수행해야 한다.

수동 분석 보조 도구인 Watcher [3]가 과거에 있었지만 제시하는 공격 벡터의 개수가 한정적이고, Fuzzing으로 찾을 수 있는 취약점의 경우 대상에서 제외되었다.

이에 따라 수동 분석이 필요한 경우를 가정해 기존의 수동 분석의 반복되는 작업을 줄이고 서비스의 가용성에 영향을 미치지 않는 방법으로 공격 벡터를 제시하여 수동 분석을 효과적으로 도울 웹 취약점 분석 방안을 제안하고자 한다.

II. 관련 연구

2.1 프레임워크 및 라이브러리

다음은 Modern Web으로 활용되고 있는 프레임워크 및 라이브러리를 정리한 표이다.

---

* 웹 애플리케이션 구조 파악을 위한 Crawling을 수행한다.
† 취약점을 분석하기 위해 다양한 입력 값을 대입한다.
‡ Fuzzing 결과를 바탕으로 취약점인지 판단한다.

Name	Info	Type
React.js	React.Component	SSR, CSR
Vue.js	script load	SSR, CSR
Angular	script load	CSR
Nuxt.js	__nuxt__	SSR
Next.js	next_PAGE_	SSR
Node.js	node_modules	-
Flask	Header	-
Django	Header	-

[표 1] 웹 프레임워크 및 라이브러리

Info에 기재된 내용은 프레임워크 및 라이브러리에 해당하는 키워드로, 이를 통해 자바스크립트 코드를 분석하여 웹 페이지에서 사용하는 환경 정보를 확인할 수 있다. SSR(Server-Side Rendering)은 웹 서버에 요청할 때마다 브라우저가 서버에 새로운 페이지에 대해 요청을 하는 방식이며, CSR(Client-Side Rendering)은 브라우저가 서버에 HTML과 자바스크립트 파일이 로드되면 사용자의 상호 작용에 따라 자바스크립트를 이용하여 동적으로 렌더링하는 방식이다. 두 가지 모두 기재된 경우는 CSR 방식으로 렌더링하지만, SSR 방식으로도 활용이 가능하다는 것을 의미한다. React와 Next.js 프레임워크를 통해 SSR을 활용하는데 도움을 받을 수 있고, Vue.js는 Nuxt.js를 통해 진행할 수 있다.

위와 같은 방법으로 웹 취약점 분석을 수행할 때 환경 정보를 파악할 수 있으며, 브라우저 확장 프로그램에서 렌더링 방식에 맞춘 데이터 수집으로 정보를 수집하도록 설계되었다.

2.2 파라미터를 통한 공격 벡터

웹 취약점 분석을 할 때 각각의 사용자 입력이 들어가는 위치에 잠재적인 공격 벡터를 제안해주면 모의해킹을 좀 더 수월하게 진행할 수 있다. 웹 사이트 내에 존재하는 Insert Point 별 공격 벡터를 제안한다. 사용자가 웹 모의해킹을 진행할 때 OWASP Top10를 기준으로 공격 기법과 취약점을 목록화하여 사용자에게 제안한다. 서버로 전달되는 데이터를 별도로 검증하지 않으면 취약할 수 있다. 이 점을 이용하여 HTTP Method 별로 전달하는 값을 통해 맞춤형 공격 벡터를 제시할 수 있다. 예를 들어, 페이지 번호를 파라미터로 전달받는 경우, 사용자에게 할당된 권한 이외의 접근, SQL Injection, XSS, File Inclusion 와 같은 추가적 공격이 가능하다.

Type	Parameter
GET	?param1=id&param2=pw
POST	Input Tag → form-data

[표 2] 파라미터를 통한 공격 벡터

GET Method를 이용하는 경우, Crawling 한 URL 목록을 기반으로 파라미터를 분석하여 공격 벡터로 인식한다. 로그인 서비스의 경우, Input 태그는 일반적으로 검색, 댓글 등 모든 서비스에서 입력 값을 전달하기 위해 사용된다. 본 논문에서는 로그인 양식에 맞춰진 Input 태그로 이름, 전화번호, ID, 비밀번호의 정보를 입력받고 전달하기 위해 작성된 내용이다.

이 과정에서 서버의 요청/응답 패킷을 캡쳐해 데이터를 파싱 후 Key-Value 형식, form-data 형식 등의 다양한 파라미터 전달 방식을 고려하여 공격 벡터로 인식하고 공격방법을 제시한다.

예로 들었던 로그인 서비스가 분석 대상인 경우, 공격 벡터를 사용자에게 나타내어 해당 벡터에서 발생할 수 있는 공격방법(SQL Injection, Server side template injection 등)을 제공한다.

2.3 쿠키를 통한 공격 벡터

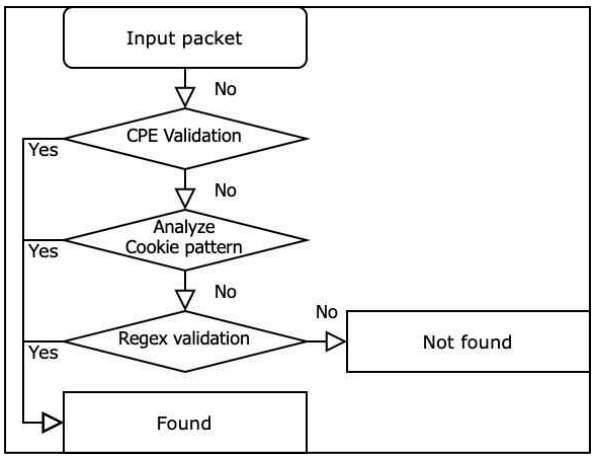
Cookie란 웹 서버에서 전송한 데이터를 클라이언트에 저장하는 값으로, 사용자가 조작할 수 있으므로 공격 벡터로 정의한다. 쿠키 값을 이용해 공격할 수 있는 방법이 다수 존재한다. 이에 대한 예시로 None Algorithm Attack 및 HMAC Algorithm Attack 등의 논리적 오류로 인해 발생하는 JWT 관련 취약점, XSS 등으로 인해 발생하는 세션 탈취, 쿠키 미검증으로 인한 데이터 처리 오류 발생 등을 들 수 있다.

위와 같이 쿠키 값을 이용한 공격 벡터 무수히 많으므로 Cookie를 공격 벡터로 인식하여 웹 사이트에 구현된 기능에 따라 사용자에게 공격방법을 제공할 수 있다.

III. 방법론

3.1 대상 사이트 구성요소 분석

아래에 표현된 대상 사이트 구성요소 분석 기능의 동작 구조는 대상 서비스에 연결하여 패킷 정보가 확보되어 있다는 전제 분석을 수행할 수 있도록 설계되어 있다.



[그림 1] 구성요소 분석 기능 동작 구조

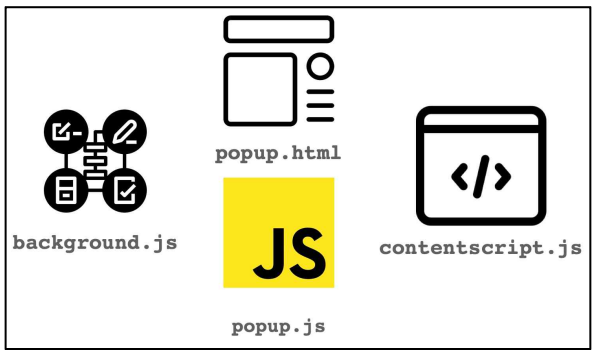
패킷이 분석 시스템 내에 들어오면 가장 먼저 MITRE에서 발표한 CPE(Common Platform Enumeration)[4]를 기반으로 1차적인 필터링을 거친다. 다른 과정을 수행하기 전에 CPE 기반 필터링을 수행함으로써 분석의 정확도를 높일 수 있다. 만약 해당 과정에서 확인되지 않는다면 다음 단계의 검사를 수행한다.

두 번째 단계로는 쿠키를 분석하여 해당하는 프레임워크가 존재하는지 확인한다. 예시로, PHP는 세션을 설정하기 위해 쿠키에 'PHPSESSID'를 설정한다. 특정 프레임워크에서 사용하는 쿠키 이름 등을 확인하는 과정을 통해서 충분히 식별 가능하다.

마지막으로는 보안 설정 오류로 인해 패킷에 함께 전송되어오는 Server나 X-Powered-By 등의 헤더 정보를 사전에 정의된 정규식을 기반으로 검증하는 검사를 수행한다. 이 단계에서도 요소 식별에 실패한다면 해당 패킷에서 데이터를 추출하는 데 실패한 것으로 처리한다.

3.2 Chrome Extension을 이용한 취약점 분석

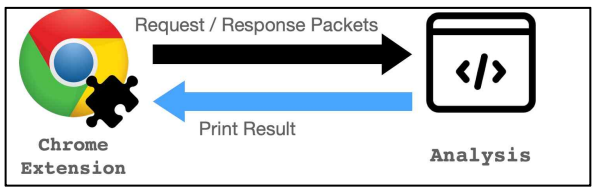
CRX(Chrome Extension)을 이용하여 사용자의 Chrome Browser에 기능을 추가할 수 있다. 이를 통해 사용자는 Chrome API를 사용하여 현재 탭에 대한 정보를 얻어 내거나, DOM에 접근 및 조작할 수 있다. CRX는 대표적으로 다음과 같은 구조로 만들어진다.



[그림 2] CRX 구조도

그림 2의 popup.html과 popup.js는 CRX에서 사용자에게 보여지는 작은 창을 디자인하고, 사용자의 동작에 반응할 수 있도록 하는 역할을 수행한다. contentscript.js는 현재 접속한 페이지의 DOM을 읽고, 수정하는 역할을 수행하는 코드 파일이다. 마지막으로, background.js는 브라우저에서 발생하는 Event listener가 내장되어 있어 사용자가 행하는 특정 행위에 대해 대응할 수 있도록 하는 이벤트 핸들링 역할을 수행한다.

CRX를 Chrome에 적용하여 취약점 분석을 수행하고 편의 기능을 제공한다. 앞선 구성 요소 분석 단계에서는 사용자가 특정 이벤트를 발생시켜야만 동작하는 기능, 로그인을 해야 접근할 수 있는 페이지 등 접근하기 어려운 부분들이 있다. 취약점 분석 기능에서는 구성 요소 분석 단계에서 기술적으로 접근하기 어려운 부분을 추가로 분석하고 결과를 도출한다.



[그림 3] CRX의 분석 진행 과정

먼저, 사용자가 취약점 분석을 하며 발생한 요청 및 응답 패킷을 수집하고, 분석 결과를 CRX

에 반환한다. 만약 해당 분석 결과가 유용한 정보일 경우, DOM에 접근하여 정보를 출력한다.

그림 3을 통해 사용자가 취약점 분석을 하는 일련의 과정을 줄일 수 있다. 예를 들어, 요청 및 응답 패킷을 분석해 찾는 과정 등을 앞서 설명한 CRX로 정보를 빠르게 수집, 분석할 수 있다. 사용자의 네트워크 패킷을 캡쳐하기 위해서는 그림 4의 예시 코드처럼 Chrome Extension API를 이용하여 요청 및 응답 패킷을 캡쳐할 수 있다.

```
chrome.devtools.network.onRequestFinished.addListener(
    function(request) {
        console.log(request.request.url);
    }
);
```

[그림 4] CRX 응답 패킷 캡쳐

IV. 결론

본 논문에서는 현재까지 널리 사용되고 있는 웹 스캐너에 대해 분석하고, 대상 서비스에 대한 Fuzzing 기법 활용을 최소화하여 가용성을 침해하지 않으면서 웹 취약점 분석을 진행하기 위한 방안을 제시하였다.

3.1 에서 제시된 구조를 통한 분석 진행 시 Crawling을 통해 대상 서비스에 대한 환경 정보가 존재한다면 대부분의 웹 프레임워크 및 라이브러리 정보를, 웹 페이지 구조, 페이지 내에서 redirect 하기 위한 URL 등 일반적으로 사용자가 직접적으로 접근해야 볼 수 있는 정보들을 빠르게 가이드라인으로 제시한다. 또한, 수집된 정보를 바탕으로 HTTP Method 별 데이터 전달 방식을 고려해 공격 벡터를 제시한다. 3.2 에서 제시한 취약점 분석 기능은 CRX를 사용해 수작업 보조를 위해 진행할 수 있게 설계되었다. 앞서 Crawling으로 받아 올 수 없는 요청, 응답 패킷 등 다양한 정보를 가져올 수 있어 Burp Suite 도구와 같은 Proxy 기능을 지원하는 타 도구를 중복으로 사용하지 않아도 된다.

웹 애플리케이션에 대한 모의해킹, 버그 바운티 등의 상황에서 실질적으로 Dummy data나 Fuzzing 행위를 수행할 수 없을 때 수작업으로 취약점 분석을 진행하기 위해서 대상에 대한 사전 조사를 진행한다. 따라서 본 논문에서 제시된 분석 방안을 통해 시간적 비용을 최소화하고, 서버 자원의 불필요한 소모를 막을 수 있다. 활용할 수 있는 공격 벡터, 서버 환경 정보 등 모의해킹에 필요한 정보를 정의하고 가이드라인으로 제시해 웹 애플리케이션의 취약점 분석에서 기존보다 효율적으로 개선할 수 있을 것으로 기대한다.

[참고문헌]

[1] S. Patil, N. Marathe and P. Padiya, "Design of efficient web vulnerability scanner," 2016 International Conference on Inventive Computation Technologies (ICICT), 2016, pp. 1-6, doi: 10.1109/INVENTIVE.2016.7824873.

[2] A. Al Anhar and Y. Suryanto, "Evaluation of Web Application Vulnerability Scanner for Modern Web Application," 2021 International Conference on Artificial Intelligence and Computer Science Technology (ICAICST), 2021, pp. 200-204, doi: 10.1109/ICAICST53116.2021.9497831.

[3] "Watcher for Web app testing", Casaba Security, https://www.casaba.com/products/watcher/

[4] Buttner, Andrew, and Neal Ziring. "Common Platform Enumeration (CPE)-Specification.", http://cpe.mitre.org (2009).