Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.34, NO.4, Aug. 2024 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2024.34.4.693

서버 측 브라우저를 활용한 서비스들의 보안 위협 진단 및 안전성 검증

이 민 상,[1†] 최 형 기[2‡]
[1,2]성균관대학교 (대학원생, 교수)

Threat Diagnosis and Security Verification of Services Using Server-Side Browsers

Min-sang Lee,[1†] Hyoung-kee Choi[2‡]
[1,2]Sungkyunkwan University (Graduate student, Professor)

요 약

브라우저는 웹페이지를 렌더링하여 데이터 추출, 형식 변환 그리고 개발 테스트 등의 기능을 수행하는 프로그램에서 활용된다. 브라우저를 활용하는 온라인 서비스는 브라우저 정보가 노출되거나 안전하지 않은 상태로 사용될 때 보안 문제를 야기한다. 본 논문에서는 안전한 브라우저 사용을 위해 취할 수 있는 보안 요구사항을 제시하고, 이를 만족하지 않을 시 발생하는 보안 위협을 설명한다. 실험을 통해 상용 웹 애플리케이션의 안정성을 검증하고, 브라우저가 공격 도구로 악용되는 취약 사례를 분석한다.

ABSTRACT

The browser is utilized to render web pages in programs that perform tasks such as data extraction, format conversion, and development testing on web pages. Online services that utilize browsers can cause security issues if browser information is exposed or used in an unsafe manner. This paper presents security requirements for the safe use of browsers and explains the security threats that arise if these requirements are not met. Through evaluation, the security verification of commercial web applications is conducted, and the vulnerabilities that allow browsers to be exploited as attack tools are analyzed.

Keywords: SSB(Server-side Browsers), SSRF, SSR(Server-side Request), Web Security

I. 서 론

웹크롤러는 웹에서 데이터를 수집하는 자동화된 도구이다. 크롤링은 웹페이지의 렌더링 결과를 기반으로 수집하기 때문에 브라우저의 기능을 이용한다. 브라우저를 사용하는 다른 예로는 웹사이트에 대한 아카이브, 웹 애플리케이션의 테스트 자동화, 그리고 웹페이지에 대한 보안 점검 등이 있다. 웹페이지를 스크린 캡처하고 PDF로 변환하는 온라인 서비스도 활용한다.

웹 애플리케이션 내부에서 브라우저는 한 번 이상 요청을 수행한다. 브라우저는 자바스크립트를 실행하고 콘텐츠를 가져오는 과정에서 사용자로부터 제공받은 URL을 통해 요청한다. 이 과정에서 보안 위협이 발생할 수 있으며, 브라우저의 사용을 식별한 공격자는 URL을 조작하여 전달하거나 악의적인 스크립트를 삽입하여 웹서버를 침해할 수 있다.

서버에서 동작하는 브라우저를 악용하는 공격은 브라우저의 취약점이나 안전하지 않은 환경에서 발생한다. Musch의 연구[1]는 구버전 브라우저의 취약

Received(06. 12. 2024), Modified(07. 25. 2024), Accepted(07. 25. 2024)
† 주저자, ogianto@g.skku.edu
‡ 교신저자, meosery@g.skku.edu(Corresponding author)

점을 이용하는 공격과 브라우저의 버전을 식별하는 방법을 제시한다. 반면, 본 연구는 안전하지 않은 환경에서 브라우저를 실행할 때 발생하는 공격을 다루며, 요청 URL의 철저한 검증과 브라우저 보안 기능 활성화의 중요성을 강조한다.

본 논문은 온라인 서비스에서 브라우저가 유발하는 보안 위협을 설명하고, 조치해야 할 보안 요구사항을 제시한다. 상용 웹 애플리케이션을 대상으로 보안 요구사항 만족 여부를 실험을 통해 확인하고 취약 사례를 분석했다. 실험을 위해 구축한 공격자 서버는 브라우저의 기본 정보 수집, 페이로드 전달, 수신 데이터 분석에 활용된다.

실험 결과, 32개의 애플리케이션 중 14개가 보안 요구사항을 만족시키지 못해 내부의 민감한 정보를 노출했다. 특히, 웹 접근성을 평가하는 Wave[2]와 PDF 관련 도구를 제공하는 PDF24[3]도 취약했다. 이 인기 사이트는 Tranco 순위[4]에서 트래픽 발생 기준 상위 1.6% 내에 위치한다. 취약한 애플리케이션 중 9개는 공격자가 브라우저를 악용해 지정한 서버로 임의 요청을 보낼 수 있는 환경이었다. 브라우저를 격리된 환경에서 동작하게 하는 보안 대책은 브라우저 기능을 이용하는 서비스에 적합한 대안이 될 수 있음을 제안한다.

II. 배경지식

서비스에 사용되는 브라우저와 활용 과정에서 발생하는 요청에 대해 설명하고 부적절한 요청에 따른 보안 위협을 제시한다.

2.1 서버 측 요청의 등장 배경

문서편집기와 메신저 등 데스크톱 기반의 애플리케이션들이 웹 기반으로 변경되어 가는 추세이다. 웹의 구조가 급격히 확산되고 서비스 종류의 다양화로 브라우저와 웹서버 간의 전통적인 클라이언트-서버 방식의 서비스 모델에도 변화를 가져왔다. 서비스 향상과 보안성 유지 등의 다양한 목적으로 웹서버들은 요청한 페이지 내에 포함된 외부 서버의 콘텐츠들을 직접 다운로드해서 렌더링한 후 사용자에게 전달하는 방식을 채택하고 있다. 외부에서 유입되는 자바스크립트와 액티브콘텐츠 등의 보안 안정성을 사전에 점검이 가능한 장점이 있다. 렌더링이 완성된 페이지는 검색 로봇에게 제시함으로써 검색엔진 최적화의 완성도를 높일 수 있다.

외부 콘텐츠 관리는 서버 내에 헤드리스 브라우저가 담당하는데 외부 서버에 직접 요청을 보내 다운로드하고 렌더링한다. 이런 종류의 브라우저를 사용자 측 브라우저와 구분하기 위하여 서버 측 브라우저(Server-side browsers, SSBs)라 부르고 현재의 서버에서 다른 서버에 보내는 요청을 서버 측 요청(Server-side request, SSR)이라 부른다. 요청한 외부 서버의 자원을 문자열로 단순하게 처리하는 wget과 curl 등의 초기 기능에서 앞서 언급한 이유로 렌더링하여 웹페이지를 바로 제공하는 기능으로 발전하였다. 그래픽 인터페이스 기능이 없는 것을 제외하고는 일반 사용자 측 브라우저와 역할이 동일한 서버 측 브라우저는 관리 측면에서 차이를 보인다. 발견된 취약점들을 사용자 측 브라우저는 자동으로 즉시 업데이트되는 반면에 서버 측 브라우저는 관리자에 의해서 주기적으로 업데이트된다. 연동하는 서비스들과 호환성 문제로 업데이트가 제대로 이루어지지 않는 경우도 있다. 업데이트가 지연되면 취약점에 노출되어 서비스 보안에 위협이 된다.

2.2 보안 위협

서버 측 요청의 보안 위협을 두 가지로 정리한다.

2.2.1 서버 측 요청 위조(Server-Side Request Forgery, SSRF)

요청한 페이지 또는 URL이 내부 서버의 자원을 가리키면 웹서버는 서버 측 요청을 내부 서버로 보내 자원을 확보한 후 사용자에게 전달한다. 허술한 서버 측 요청을 악용해서 공격자가 의도하는 요청으로 변경할 수 있다. 방화벽으로 막혀 외부에서는 접근이 불가한 특정 사이트 내부 네트워크에 공격자는 서버 측 요청의 서버 주소를 변경하고 URL을 조작해서 비정상적인 동작을 유도할 수 있다.

Fig. 1.은 example.com 웹서버가 특정한 이미지를 내부의 이미지 서버에 요청해서 사용자에게 전달하는 서비스를 보여준다. 사용자가 http://example.com/proc?url=htts://10.2.2.22로 요청한다(Fig. 1. ①). example.com의 웹서버는 URL을 파싱한 후 이미지 서버 10.2.2.22에게 별도의 서버 측 요청을 보낸다(Fig. 1. ②). 이미지 서버로부터 응답을 받아서 사용자에게 이미지를 전달하는 서비스

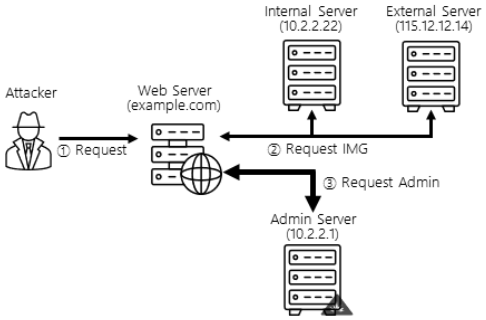

Fig. 1. Example.com is an application that interacts with internal and external resources using server-side request. If example.com is vulnerable to SSRF, an attacker can manipulate SSR to gain access to the management module.

이다. URL에 포함된 이미지 서버의 주소를 10.2.2.1로 위조해서 요청하면 허술한 웹 애플리케이션은 외부에서 접근이 불허된 내부 서버에 요청을 보내서 비정상적인 동작을 유도할 수 있다(Fig. 1. ③). 이런 종류의 공격은 특별히 서비스별로 서버를 독립적으로 분리한 내부 네트워크에서 또는 사용자가 전달한 데이터를 서버가 특정한 목적으로 가공해 주는 서비스에서 빈번하게 발생한다.

이런 취약성으로 SSRF는 2021년 OWASP[5]가 발표한 10대 웹 애플리케이션 취약점에 처음으로 소개되었다. 이를 방어하기 위해서 서버 측 요청 URL에 대한 철저한 검증이 요구된다. 서비스 프로토콜, 내부 서버들의 주소에 대한 화이트 또는 블랙리스트로 유지하는 방법으로 방어가 가능하다.

2.2.2 서버 측 브라우저 공격

사용자의 요청을 웹서버가 별도의 서버에 서버 측 요청으로 전달해서 처리하는 과정에서 발생하는 공격으로 SSRF와 유사하지만, 허술한 웹 애플리케이션이 SSRF의 원인 제공인 반면 서버 측 브라우저 공격은 서버 측 브라우저의 취약점이 주된 원인이다.

서버 측 브라우저는 최선 버전으로 유지하여 알려진 취약점을 제거하고 브라우저가 제공하는 보안 정책에 기반하여 동작해야 한다. 공격자는 브라우저 기본 정보와 실행 환경을 수집하여 두 가지 공격을 수행할 수 있다. 하나는 오래된 버전의 서버 측 브라우저가 가진 공개된 보안 취약점을 악용하는 방법이고, 다른 하나는 보안이 허술한 실행 환경에서 검증되지 않은 URL로 동작하는 서버 측 브라우저를 공격자의 도구로 악용하는 공격이다.

동일 출처 정책은 브라우저가 실행하는 기본 보안 정책으로, 브라우저 버전에 따라 구현의 엄격성이나 방식에 차이가 있을 수 있다. 버전 간 구현 차이로 인해 도메인 간 자원 요청의 실행 결과가 달라질 수 있으므로, 서버 측 브라우저는 일관된 서비스 결과물을 얻기 위해 브라우저의 기본 보안 기능을 해제하는 전략을 사용할 수 있다. 서비스에 중점을 둔 전략으로 공격자는 스크립트를 전달해서 서버 측 브라우저를 통해 악의적인 코드를 실행시킬 수 있다.

Fig. 2.에서 example.com의 웹서버는 사용자가 요청한 임의의 HTML 형식의 웹페이지를 PDF 형식으로 변경하는 서비스를 보여준다. 사용자가 http://example.com/pdf?url=http://google.com을 요청하면(Fig. 2. ①) example.com 웹서버의 헤드리스 브라우저를 이용해서 google.com 페이지를 다운받아(Fig. 2. ②, ③) PDF로 변경해서(Fig. 2. ④, ⑤) 사용자에게 전달한다. 헤드리스 브라우저는 서비스 페이지를 다운받고 페이지 내에 자바스크립트를 실행해서 렌더링을 마친 후에 PDF로 변경한다. 주로 제3의 사이트를 번역, 이미지 추출, 문서 형식 변경 서비스에 많이 사용되고 웹크롤러에서도 자주 사용된다. 이러한 서비스를 제공하기 위해 서버 측 브라우저를 활용하는 애플리케이션을 서버 측 브라우저 애플리케이션(이하 SSB 애플리케이션)이라고 한다.

공격자는 URL 내의 목적지 google.com을 공격

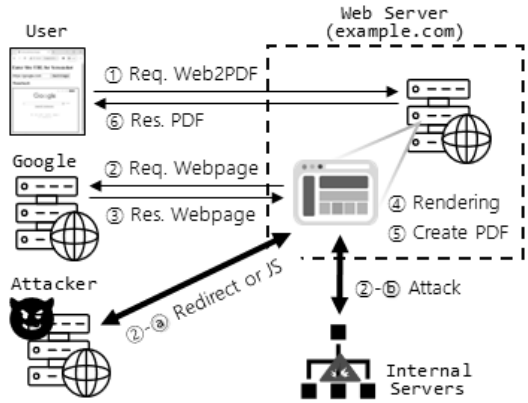

Fig. 2. Example.com is an application that converts web pages into PDF files. When a user provides google.com as input, example.com converts Google's web page into a PDF and provides the PDF to the user.

자가 지정하는 서버의 주소 attacker.com으로 변경한 후 example.com에 요청해서 공격을 개시한다(Fig. 2. ②-ⓐ). 헤드리스 브라우저의 취약점을 알고 있는 공격자는 공격코드를 attacker.com에 사전에 설치해 두고 example.com의 헤드리스 브라우저가 이를 실행해서 내부 네트워크 자원으로 공격하는 방식이다(Fig. 2. ②-ⓑ).

서버 측 브라우저 공격은 방화벽 내부의 서버 측 브라우저에서 시작되며, 서버 측 브라우저가 발생시키는 다량의 악의적인 요청을 방화벽이 차단하지 못한다. 반면, SSRF 공격은 웹서버의 방화벽 외부에서 시작되며, 단일 요청에 기반하여 진행된다. 서버 모니터링을 통해 악의적인 요청을 식별하고, 공격의 출처를 방화벽으로 차단함으로써 최소화할 수 있다.

III. 보안 요구사항

서버 측 브라우저를 활용하는 서비스 플랫폼들은 서버 측 요청을 위조하는 공격에 취약하다. 앞서 살펴본 서버 측 요청 공격을 예방하기 위해, 서버 측 요청을 유발하는 URL에 대한 확인과 점검은 필수적이다. 최소한 세 번에 걸쳐 점검이 수행되어야 한다. 1) 사용자로부터 서버 측 요청 URL을 제공받는 시점, 2) 서버 측 브라우저가 제공받은 URL의 웹페이지를 요청하는 시점, 3) 서버 측 브라우저가 웹페이지를 렌더링하는 시점이다. 점검 시점별로 가능한 공격을 설명하고 예방을 위해 고려해야 할 보안 요구사항을 알아본다.

3.1 SSRF 공격 예방

웹페이지를 PDF로 변환해 주는 SSB 애플리케이션은 변환할 웹페이지를 가져오고 렌더링하는 데 서버 측 브라우저를 이용한다. 서버 측 브라우저가 특정 웹페이지를 요청하기 위해 사용하는 URL을 서비스대상 웹페이지 URL이라 한다. Fig. 2.에서 서비스대상 웹페이지 URL은 google.com이고, 사용자는 SSB 애플리케이션의 웹페이지 입력 폼에 입력하여 전달한다.

서비스대상 웹페이지 URL은 사용자로부터 제공받는 시점에 검증해서 유효한 요청만 서버 측 브라우저가 처리하도록 해야 한다. Fig. 2.에서 서비스대상 웹페이지 URL은 웹페이지를 PDF로 변환해달라는 사용자의 HTTP 요청에 포함되어 전송된다. SSB 애플리케이션은 Fig. 2.의 ①에서 수신한 PDF 변환 요청을 파싱하여 서비스대상 웹페이지 URL을 추출한다. 서버 측 브라우저가 허용되지 않는 URL로 요청하지 않도록 추출한 URL을 검증해야 한다. 서비스대상 웹페이지 URL이 127.0.0.1 등의 내부 시스템을 가리키거나 피싱 사이트 같은 신뢰할 수 없는 외부 도메인을 가리키는 경우, 즉각적으로 서비스를 중단해야 한다. PDF 변환 서비스에서 웹페이지와 콘텐츠의 수집은 HTTP 기반으로 동작하기 때문에 이외 프로토콜을 차단하고 내부의 자원을 가리키는 주소도 필터링해야 한다. 예를 들어, 사용자가 지정한 특정 웹페이지를 PDF로 변환해 주는 서비스 플랫폼에서 파일 프로토콜을 허용하는 경우, 공격자가 이를 악용하면 서버 내부의 패스워드 파일 등 중요한 내용이 PDF로 변환되어 노출될 수 있다. OWASP는 SSRF 예방을 위한 체크리스트[6]를 제공한다.

3.2 리디렉션 공격 예방

공격자는 Fig. 3.의 공격자 서버를 이용해 서버 측 브라우저의 요청을 리디렉션할 수 있다. 리디렉션 공격은 허용하지 않는 URL을 필터링하는 보호 대책을 우회하여 내부 서버를 가리키는 URL을 전달하는 수단으로 사용될 수 있다.

서버 측 브라우저는 서비스대상 웹페이지 URL에서 안전하다고 판단된 URL을 기반으로 웹페이지를 요청한다. 공격자 서버가 신뢰할 만한 도메인 네임을 사용하는 경우, Fig. 2의 ②-ⓐ처럼 서버 측 브라우저는 공격자 서버로 요청을 보내고, 해당 서버는 내부 자원으로 리디렉션을 유도하는 응답을 반환한다. 결국, 서버 측 브라우저는 Fig. 2의 ②-ⓑ처럼 최종 목적지인 내부 자원으로 요청을 보내게 되면서 URL을 제공받는 시점의 검증을 우회할 수 있다. 응

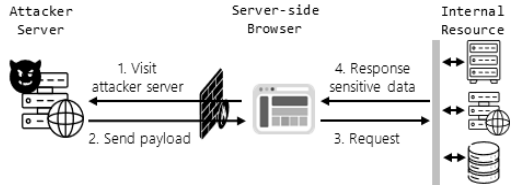

Fig. 3. The attacker's server hosts JavaScript that collects server information, responds with redirection to internal resources, and hosts HTML that triggers subresources.

답으로 받은 리디렉션 URL을 검증하지 않으면, 서버 측 브라우저는 내부 자원의 정보를 렌더링하여 서비스 결과물로 생성하게 된다.

리디렉션 공격을 예방하기 위해서는 서버 측 브라우저가 서비스대상 웹페이지를 요청하는 시점에 다시 필터링해야 한다. 리디렉션 응답 자체를 차단하는 방법을 적용할 수 있지만, 사용자가 클라이언트 브라우저에서 경험하는 웹페이지와 동일하게 렌더링해야 하는 SSB 애플리케이션에는 적용하기 어렵다.

3.3 서버 측 브라우저 공격 예방

공격자는 악의적인 웹페이지를 전달하여 서버 측 브라우저가 내부의 자원을 서브리소스로 요청하도록 지시할 수 있다. 이는 서브리소스 URL을 충분히 검증하지 않는 점을 악용하는 서버 측 브라우저 공격으로, 브라우저 자체의 취약점을 이용하지는 않는다. 서브리소스 요청은 iframe, img 등 HTML 태그를 사용하는 정적인 방법과 자바스크립트를 이용하는 동적인 방법으로 발생한다. 공격자는 Fig. 3의 공격자 서버에 내부 자원을 서브리소스로 요청하는 웹페이지를 준비해 두고, 서버 측 브라우저의 요청에 이를 응답으로 전달하면서 공격을 시작한다. 예를 들어, http://127.0.0.1/admin 이 내부 서버의 관리용 웹페이지 URL인 경우, 해당 URL을 iframe의 src 속성으로 지정한 웹페이지를 서버 측 브라우저에 전달한다. 전달받은 웹페이지 내의 서브리소스 URL을 검증하지 않으면, 서버 측 브라우저는 iframe에 서버 관리 정보를 표시하게 된다. 공격자는 서브리소스 요청에 대해서도 리디렉션시켜 보호 대책을 우회할 수 있기 때문에 이를 고려하여 변경된 최종 목적지의 URL을 대상으로 필터링해야 한다.

서버 측 브라우저 공격에서 서브리소스 URL을 검증하지 않는 취약점은 공격자가 내부 서버를 목표로 삼을 수 있게 하며, 악의적인 자바스크립트와 함께 이를 악용할 수 있다. 서비스대상 웹페이지 URL을 조작하는 방법은 매번 한 번씩의 공격 기회만 주어지지만, 서브리소스 URL을 이용하는 방법은 한 번에 여러 차례 공격할 수 있어 더 효율적이다. 공격자는 다수의 서브리소스를 요청하는 웹페이지를 전달하여 서버 측 브라우저가 특정 내부 서버의 자원을 소모하게 만들 수 있다. 내부 자원은 공격의 출처를 서버 측 브라우저로 인식하기 때문에, 공격자는 자신을 숨길 수 있다.

3.4 서버 측 브라우저의 보안 기능 활성화

렌더링 화면은 브라우저와 그 버전에 따라 차이를 보일 수 있다. 예전 브라우저에서 정상적으로 작동하던 웹페이지도 최신 브라우저의 보안 정책에 의해 다르게 나타나거나 일부 기능이 동작하지 않을 수 있다. 다양한 브라우저 환경에서 일관된 사용자 경험을 제공하기 위해 SSB 애플리케이션은 서버 측 브라우저의 기본 보안 정책을 완화하는 방법을 사용할 수 있다. 보안 기능은 브라우저의 실행 옵션으로 조절한다. 낮은 보안 수준으로 동작하는 브라우저는 동일 출처 정책을 우회하여 크로스 도메인과 상호작용할 수 있고 로컬 파일의 내용도 렌더링할 수 있다. 보호 기능이 해제된 브라우저는 악성 스크립트에 의해 위협받을 수 있으므로 개발 테스트 등 제한된 목적에만 사용해야 한다.

공격자는 서버를 이용하여 서버 측 브라우저와 상호작용함으로써 서버 측 브라우저의 보안 상태를 파악할 수 있다. 보안 상태는 의도적으로 크로스 도메인 콘텐츠를 가져오게 하는 자바스크립트를 전달하여 서버 측 브라우저의 처리 여부를 관찰함으로써 확인한다. 만약 서버 측 브라우저가 크로스 도메인 콘텐츠를 가져와 공격자 서버로 전송한다면, 이는 서버 측 브라우저의 보안 기능이 해제되어 있음을 의미한다. 서버 측 브라우저의 기본 정보는 자바스크립트 실행으로 서비스 결과물에 포함된 사용자-에이전트 값을 통해 알 수 있다. HTTP 요청 헤더에도 동일 정보가 있어 공격자 서버에 기록된 수신 로그를 통해서도 파악할 수 있다. 서버 측 브라우저 버전 정보의 노출은 공격자에게 공개된 취약점을 악용할 기회를 제공한다.

서버 측 브라우저의 보안 수준과 버전 정보를 노출하는 SSB 애플리케이션은 서버 측 브라우저를 공격자의 도구로 활용할 수 있게 하는 서버 측 브라우저 공격에 취약하다. 이를 예방하기 위해 불필요한 정보의 노출은 제거하고 서버 측 브라우저를 지속적으로 최신 버전으로 유지해야 한다.

IV. 서버 측 브라우저 활용의 안전성 실험

실험은 상용 SSB 애플리케이션을 대상으로 서버 측 브라우저의 보안 환경을 평가하고, 원인을 분석한다. 내부 자원이 갖춘 보안 대책은 취약한 애플리케이션을 이용하여 점검했다. 평가 대상으로 선정된 총

32개의 애플리케이션 중 약 44%의 14개에서 서버 측 요청 공격에 취약했다. Tranco는 최근 한 달 동안 발생한 웹 트래픽 데이터를 기반으로 웹사이트의 순위를 책정하는 서비스이다. Tranco 순위에서 상위 1.4%에 위치하는 Wave와 상위 1.6%에 위치하는 PDF24를 각각 SSRF와 서버 측 브라우저 공격의 취약 사례로 소개한다.

4.1 분석 대상 선정

사용자가 서비스대상 웹페이지의 URL을 직접 제공하고 결과를 즉시 확인할 수 있는 온라인 서비스, 특히 스크린 캡처 및 PDF 변환 기능의 SSB 애플리케이션을 대상으로 보안 점검을 수행했다.

분석 대상은 2023년 11월 6일 기준으로 수집한 Tranco 100k 목록 중 10k 웹사이트로 한정했다. 서버 측 브라우저를 활용하는 웹서비스를 선정하는 과정에는 연구자의 최종 확인이 필요했기 때문에 10k 이상으로 확대하지 못했다. 선정 과정은 세 단계로 진행했다. 먼저, 웹페이지를 이미지로 생성하는 크롤링으로 첫 번째 선별을 하고, 이미지로 생성된 웹페이지 화면을 직접 확인하여 두 번째 선별을 진행했다. 이후 자바스크립트 실행 여부로 서버 측 브라우저의 활용을 확인하여 최종 선정했다. 크롤링의 선별 기준은 입력 폼을 보유하면서 'screenshot', 'convert', 'capture' 등의 단어를 포함하는 웹페이지를 대상으로 한다. 추가로, 유사 사이트 검색[7] 및 Google 검색을 통해 평가 대상 사이트를 확장하였으며, 연구의 편의성을 위해 회원 가입 없이 데모 기능을 이용할 수 있는 사이트에 한정하였다.

보안 점검 도구로 사용하는 URLScan[8]은 점검 결과를 스크린 캡처 이미지로 제공하고, 접근성 평가 도구로 사용하는 Wave는 분석 결과를 화면에 나타내기 때문에 평가 대상 사이트에 포함하였다. 선정된 총 32개의 애플리케이션 리스트는 Table 1.에 명시하였다.

선정된 애플리케이션을 호스팅하는 웹서버 중 Apache 8개, Nginx 8개, Cloudflare 9개가 전체 78%를 차지했으며 2개는 서버 식별이 불가능했다. 서비스에 활용되는 서버 측 브라우저는 2개를 제외하면 30개의 애플리케이션이 크롬 기반 브라우저를 사용했다. 2023년 11월부터 2024년 1월까지 약 3개월간 진행된 실험 기간에 크롬의 마지막 업데이트 버전은 120으로 119 버전 이상의 최신 버전을 유지하는 애플리케이션은 10개뿐이었다.

Table 1. List of commercial SSB applications subjected to security inspection for research (FF: Firefox, HC: Headless Chrome, C:Chrome / Number: Version)

Application	Server	OS	Browser
A App	Apache	Linux	HC119
B App	Apache	Linux	C83
C App	Vercel	Linux	HC92
D App	Cloudflare	Win	C49
Avepdf	MS-IIS	Win	HC112
Browshot	Nginx	Linux	FF56
Domsignal	Cloudflare	Linux	C62
Ettvi	Nginx	Linux	C119
GrabzIt	-	Win	C120
Onlineconvertfree	Nginx	Linux	HC107
PagePeeker	Cloudflare	Linux	C77
PDF24 Tools	Apache	Linux	HC120
PDFCrowd	Nginx	Linux	C84
PDFmyURL	Apache	Linux	C119
Pikwy	Nginx	Linux	C114
Screenshot Machine	Cloudflare	Linux	C119
Screenshot.guru	Google Frontend	Linux	HC93
ScreenshotLayer	Cloudflare	Linux	HC60
ScreenshotsCloud	Cloudflare	Win	C112
Sejda HTML to PDF	Cloudflare	Linux	C120
Simpletools.nl	Cloudflare	Mac	C111
Site-Shot	Nginx	Linux	C120
S-shot.ru	Nginx	Linux	C120
T.ly	Cloudflare	Linux	HC118
TestLocally	Apache	Linux	C113
Thumbnail.ws	Apache	Linux	FF81
URL2PNG	Cowboy	Linux	C116
URLScan	Nginx	Win	C120
Urltoscreenshot	Apache	Linux	C61
Vivoldi	-	-	C111
Wondershare	Tengine	Win	HC113
Wave	Apache	Linux	C116

4.2 공격 환경 구축

공격자 서버는 서버 측 요청을 유발하는 URL을 전달하는 용도로 활용되며, 다음과 같은 조건을 만족한다. 1) 웹서버로 동작하고, 2) 신뢰하는 도메인을 사용하며, 3) 수신한 HTTP 요청과 데이터를 확인할 수 있어야 한다. SSB 애플리케이션이 공격자 서버의 URL을 신뢰하지 않는 도메인으로 필터링하지 않도록, 공개 호스팅 플랫폼의 도메인을 사용한다.

클라우드 기반의 웹 호스팅 플랫폼을 제공하는 Glitch[9]는 Node.js 기반의 웹 애플리케이션 개발 환경을 제공하며, 개발된 애플리케이션을 호스팅한다. 코드를 실시간으로 수정할 수 있고, 변경 사항을 웹 애플리케이션에 즉시 반영할 수 있어 공격자 서버로 적합하다. SSB 애플리케이션은 서비스대상 웹페이지의 URL을 제공받으면 사용자 경험을 향상시키기 위해 서비스 결과물을 일정 시간 동안 캐싱한다. 캐싱된 결과가 응답되지 않도록 하기 위해, 공격자 서버에서 호스팅하는 웹페이지의 URL을 수시로 변경해야 한다. Glitch의 개발 환경은 변경 사항을 자동으로 반영하므로 실험 수행에 효과적이다.

공격자 서버는 SSB 애플리케이션의 서버 측 브라우저와 상호작용하는 Node.js 기반의 웹 애플리케이션을 실행하며, 웹페이지와 자바스크립트 같은 정적 리소스를 호스팅한다. 내부 자원을 가리키는 URL을 서버 측 브라우저에 전달하고, 서버 측 브라우저가 해당 URL을 처리하는 과정에서 유출되는 데이터를 실시간으로 수신하여 콘솔에 출력한다. 공격자 서버 구축에 활용된 코드 및 정적 리소스는 Github[10]에 공개하였다.

4.3 자바스크립트를 이용한 서버 측 브라우저의 정보 수집

자바스크립트의 Navigator 객체는 브라우저의 종류와 버전, 플랫폼, 사용 언어, CPU, 메모리, 배터리 상태, 그리고 사용자의 위치 등 브라우저와 관련된 주요 정보를 포함한다. 브라우저 정보의 정확도는 다른 객체들이 제공하는 정보를 조합해서 향상할 수 있다. 디스플레이 화면에 관한 정보를 가진 Screen 객체가 표준 화면 크기와 다른 정보를 제공하는 경우는 헤드리스 브라우저로 식별한다. 크롬 버전 32부터 지원하는 Promise 객체처럼 브라우저가 특정 객체나 메서드의 지원 여부를 확인함으로써 브라우저의 종류와 버전을 특정할 수도 있다. 실험에서는 주로 Navigator 객체를 이용하여 서버 측 브라우저의 기본 정보를 수집한다.

공격자 서버가 전달한 자바스크립트는 서버 측 브라우저의 기본 정보를 노출시키고, 노출된 정보는 이미지 또는 PDF로 생성된다. 브라우저의 종류와 버전 정보는 공격자 서버에서 수신한 서버 측 브라우저의 요청 헤더를 통해서도 수집이 가능하고, 웹서버 정보를 제공하는 웹사이트[11]를 이용하는 방법도 있다. 세 가지 수집 방법 중 스크립트를 활용한 측정 방식은 자바스크립트를 처리하는 서버 측 브라우저의 활용을 직접 확인할 수 있으며, 서버 측 브라우저의 보안 수준도 점검할 수 있다.

서버 측 브라우저의 사용자-에이전트 및 플랫폼 정보는 임의로 변경할 수 있다. 일부 웹사이트는 크롤링 방지 목적으로 헤드리스 브라우저의 사용자-에이전트를 가진 HTTP 요청을 차단한다. 이러한 차단 메커니즘을 우회하기 위해 일부 SSB 애플리케이션은 사용자-에이전트를 일반 브라우저로 변경하여 서버 측 브라우저를 동작시킨다. 실험에서는 수집된 사용자-에이전트 정보를 점검에 필요한 페이로드를 결정하는 데 활용하기 때문에 원래의 정보는 필요하지 않다.

서버 측 브라우저 실행 환경의 보안 수준은 브라우저의 동일 출처 정책 준수 여부를 통해 평가한다. 실행 옵션 설정을 통해 동일 출처 정책 제한을 완화한 서버 측 브라우저는 크로스 도메인의 콘텐츠를 가져오는 자바스크립트를 정상적으로 처리한다. Fetch API는 서버로부터 데이터를 비동기적으로 가져오는 데 사용된다. 크로스 도메인 콘텐츠를 가져오는 Fetch 요청이 성공하면, 이는 서버 측 브라우저가 동일 출처 정책을 준수하지 않음을 의미한다. 동일 출처 정책은 iframe에도 적용되므로, 자바스크립트를 사용하여 iframe 내부의 크로스 도메인 콘텐츠에 접근할 수 있는지 확인함으로써 보안 수준을 평가한다. 이 방법을 통해 iframe 내부의 내용을 읽을 수 있으면, 자바스크립트는 해당 내용을 공격자 서버로 전송한다.

4.4 서버 측 요청 URL에 대한 필터링 여부 점검

서버 파일과 내부 자원을 가리키는 URL의 허용 여부를 점검한다. 서버 파일의 접근 여부는 파일 프로토콜을 포함하는 URL을 사용하여 확인하며, 서버

측 브라우저가 동작하는 운영체제가 리눅스일 경우 file:///etc/passwd를 사용한다. 기본 보안 정책이 해제된 서버 측 브라우저는 서버 계정 파일에 접근하여 파일 내용을 노출한다. 내부 자원의 접근 여부 점검에는 루프백 주소인 http://127.0.0.1을 사용한다. 루프백 주소의 URL은 일반적으로 웹서버가 제공하는 기본 웹페이지 또는 서비스의 메인 웹페이지를 가리킨다. SSB 애플리케이션이 내부 자원을 가리키는 URL을 차단하지 않을 경우, 공격자는 내부 IP가 포함된 URL을 이용하여 내부 시스템을 스캔하거나 대량의 임의 요청을 발생시켜 DoS 공격을 수행할 수 있다.

서버 측 브라우저는 서브리소스를 포함한 웹페이지를 렌더링하기 위해 두 번의 서버 측 요청을 발생시킨다. 첫 번째 요청은 서비스대상 웹페이지 자체를 가져오기 위해, 두 번째 요청은 웹페이지에 포함된 서브리소스를 가져오기 위해 이루어진다. 서버 측 브라우저의 요청은 리디렉션이 발생할 때마다 추가 요청이 생겨 최종 요청 건수가 증가할 수 있다.

실험은 서비스대상 웹페이지의 URL에 대한 필터링 여부로 SSRF 취약점을 점검하고, 서브리소스의 URL에 대한 필터링 여부로 서버 측 브라우저 취약점을 점검한다. 각 요청의 리디렉션으로 변경된 최종 목적지 URL도 필터링 여부를 확인한다.

Table 2.는 SSB 애플리케이션을 대상으로 SSRF 및 서버 측 브라우저 취약점을 조사한 결과를 나타내며, 전체 32개 중 14개에서 해당 취약점이 발견되었다. SSRF 취약점 점검은 서비스대상 웹페이지의 URL을 SSB 애플리케이션의 입력 폼에 직접 입력하여 수행했다. 서버 측 브라우저 취약점 점검은 내부 자원을 서브리소스로 요청하는 웹페이지를 공격자 서버에 준비해 두고, 해당 웹페이지의 URL을 입력하여 수행했다.

스크린 캡처와 PDF 변환을 메인 서비스로 제공하지 않는 URLScan과 Wave를 제외하면, SSB 애플리케이션은 리디렉션 응답을 처리했다. 리디렉션 URL은 자원의 변경된 경로를 나타내므로, 웹페이지와 콘텐츠의 누락을 방지하기 위해 리디렉션 응답을 차단하지 않는다.

서비스대상 웹페이지의 URL이 제공받는 시점의 검증을 통과한 후 리디렉션되면, 변경된 최종 목적지의 URL을 이전 필터링 기준으로 다시 검증해야 한다. 서버 측 요청 URL에서 내부 IP를 허용하는 9개의 애플리케이션 중 Wave를 제외한 8개가 내부 IP를 가리키는 리디렉션 URL을 허용했다. 특히, B App은 입력 폼으로 전달한 URL 중 127.0.0.1만 필터링하지 못했으나, 리디렉션 URL은 localhost와 localtest.me 주소도 필터링하지 못했다.

Table 2. Applications vulnerable due to inadequate filtering of user-supplied and subresource URLs for server-side request vulnerabilities. (*: Access by poor configuration of the admin module)

Application	Server	SSRF Vulnerability by Service Target URL		SSB Vulnerability by Subresource URL	
		File protocol	Internal IP	File protocol	Internal IP
A App	Apache	O	O*		
B App	Apache		O*	O	O*
C App	Vercel	O(MiTM)		O	
D APP	Cloudflare			O	
Browshot	Nginx				O*
Onlineconvertfree	Nginx	O(MiTM)			
PDF24.org	Apache		O*		O*
Pdfmyurl	Apache		by API		
ScreenshotsCloud	Cloudflare			O	
Site-Shot	Nginx		O		O
S-shot.ru	Nginx		O		O
Thumbnail.ws	Apache		O*		O*
Urltoscreenshot	Apache	O			
Wave	Apache		O*		

4.4.1 SSRF 공격에 취약

서비스대상 웹페이지의 URL에서 파일 프로토콜을 허용하는 애플리케이션은 4개, 내부 IP를 허용하는 애플리케이션은 8개로 나타났다. 파일 프로토콜을 허용하는 C App과 Onlineconvertfree는 중간자 공격(Man-in-The-Middle, MiTM)에 취약했다. HTTP 요청을 가로채서 유효한 서비스대상 웹페이지의 URL을 서버 파일을 가리키는 URL로 변경하면, 파일 내용이 유출된다. 일부 SSB 애플리케이션은 서비스 기능을 API로 지원한다. API URL의 파라미터 값으로 전달되는 서비스대상 웹페이지의 URL도 웹 기반 서비스에 적용된 기준과 동일하게 필터링해야 한다. 다른 기준을 적용한 PDFmyURL은 API URL에서만 내부 IP를 허용했다.

SSB 애플리케이션이 API URL을 공개하고 내부 IP를 허용하는 경우, 공격자는 Fig. 4와 같이 파라미터 값으로 전달된 서비스대상 웹페이지의 URL에서 IP와 Port를 변경하여 내부 시스템을 스캔할 수 있다. S-shot.ru는 스크린 캡처 기능을 제공하는 사이트로, 동일 기능의 API를 무료로 제공한다. 앞서 설명한 스캔 방법으로 일부 Port를 점검한 결과, 웹서버는 동일한 기능의 Site-shot과 다른 용도의 웹 애플리케이션들을 호스팅하고 있었다.

서비스 제공자는 유료 사용자에게 API 키를 발급하여 더 많은 서비스 횟수와 고급 옵션을 제공한다. API 키가 포함된 API는 Xurlfind3r[12] 도구를 통해 인터넷에서 수집할 수 있었으며 일부는 성공적으로 동작했다. 인터넷에 노출된 유효한 API는 공격자에게 더 많은 악용 기회를 줄 수 있으므로, 서비스 제공자는 API 키 관리에 적극적이어야 한다.

Fig. 4. Some applications provide APIs. If internal IPs are allowed, the API URL can be utilized for internal scanning.

4.4.1.1 Wave 사례

Wave는 웹페이지의 웹 접근성을 평가하는 도구로, 평가할 웹페이지의 URL을 입력으로 받는다. 루프백 주소로 서버 모니터링 웹페이지의 URL을 제공한 결과, 웹서버의 현재 상태 정보가 노출되었다. 정보의 노출은 두 가지 문제점에 의해 발생한다.

첫째, Wave는 서비스대상 웹페이지의 URL에서 루프백 주소를 필터링하지 못했다. 127.0.0.1의 루프백 주소는 localhost, lvh.me, localtest.me, 127.0.0.1.xio.io 등으로 대체 표현이 가능하다. 내부에서 사용하는 IP, 도메인 그리고 그 대체 표현을 블랙리스트에 포함시켜 필터링에 활용해야 하지만 Wave는 localtest.me를 포함하지 않았다.

둘째, 내부 자원의 보안 설정이 미흡하다. Wave는 Apache 웹서버에서 호스팅되며, 이 서버에는 서버 상태를 모니터링할 수 있는 관리 모듈이 활성화되어 있다. 이 관리 모듈은 기본적으로 로컬 서버에서의 접근만 허용하는 접근 제어 설정이 적용되어 있는데, 이는 서버 내 관리자가 접근할 수 있도록 하기 위함이다. 이 설정은 동일 웹서버에서 동작하는 다른 애플리케이션들의 접근 가능성을 충분히 고려하지 않았다. 관리 모듈의 접근 조건을 만족하는 Wave의 요청으로 인해 서버 정보가 노출된다.

4.4.1.2 A App 사례

A App는 웹페이지를 이미지 또는 PDF로 변환해 주는 기능을 제공한다. 사용자가 입력 폼으로 제공하는 서비스대상 웹페이지의 URL에서 파일 프로토콜과 내부 IP를 필터링하지 않는 취약점을 가지고 있다. 파일 프로토콜 URL을 사용한 SSRF 공격으로 계정 정보와 서버 설정 내용이 노출되었고, 루프백 주소를 통해 웹서버의 기본페이지가 노출되었다.

서버 모니터링 웹페이지의 URL로 점검한 결과, Wave와 동일하게 웹서버 관리 모듈의 보안 설정이 미흡했다. 노출된 서버 상태 정보를 통해 다수의 웹서비스가 호스팅되고 있음을 확인했다. 동일 웹서버가 호스팅하는 다른 웹 서비스들은 SSRF에 취약한 A App에 의해 주요 설정 정보가 노출되거나 서비스 중단 등의 보안 사고로 이어질 수 있다.

4.4.2 서버 측 브라우저 공격에 취약

서비스대상 웹페이지의 URL은 필터링하지만, 서브리소스의 URL은 필터링하지 않는 애플리케이션이 3개로 나타났다. 이 중 스크린 캡처 서비스를 제공하는 Browshot은 서브리소스의 URL에서만 내부 IP를 허용했고, 이로 인해 웹서버 정보가 노출되

었다. 본 실험에서 Nginx를 웹서버로 사용하는 애플리케이션 중 서버 정보가 노출된 유일한 사례였다.

서버 측 브라우저 공격은 보안 기능이 해제된 서버 측 브라우저를 악용하여 자바스크립트를 실행하거나, 내부 IP를 차단하지 않는 서브리소스 URL을 통해 내부 서버로 요청을 보내는 방식으로 수행된다. 서브리소스 URL에서 필터링이 미흡한 PDF24와 취약한 브라우저 환경의 B App을 사례로 설명한다.

4.4.2.1 PDF24 사례

PDF24는 웹페이지를 PDF로 변환해 주는 기능 외에도 다른 유형의 파일을 PDF로 변환하거나 PDF 파일 간 병합 등 다양한 PDF 도구를 제공한다. Tranco 순위에서 15,083에 위치하는 인기 사이트임에도 불구하고 내부 IP를 허용하고 있으며, 내부 자원에 적용된 보안 대책도 미흡했다.

서비스대상 웹페이지 URL과 서브리소스 URL에서 내부 IP를 허용하는 취약점은 SSRF 공격과 서버 측 브라우저 공격을 가능하게 한다. 서비스대상 웹페이지의 URL을 웹서버 모니터링 웹페이지의 URL로 조작하는 SSRF 공격으로 서버 정보가 유출된다. SSRF 공격은 웹페이지 입력 폼뿐만 아니라 API를 통해서도 가능했다. Xurlfind3r 도구로 수집한 미공개 API는 이미지, 오피스 문서 등 다양한 파일 유형을 PDF로 변환해 주는 기능을 제공한다. API의 SSRF 취약점은 내부 데이터를 유출시키는 공격뿐만 아니라 서버 측 브라우저 공격에도 악용될 수 있다. 공격자는 서버 측 브라우저를 이용해 반복적으로 API를 호출하는 자바스크립트를 실행시켜 자원을 고갈시킬 수 있다. 서버 측 브라우저는 API를 사용하지 않고 서브리소스를 반복 요청하여 특정 내부 서버를 위협할 수도 있다.

PDF24를 호스팅하는 Apache 웹서버의 관리 모듈은 로컬 서버의 요청만을 허용하는 기본 보안 설정을 가지고 있다. 로컬 서버에서 동작하는 PDF24는 서비스대상 웹페이지의 URL과 서브리소스 URL에서 내부 IP를 허용하는 취약점을 가지고 있어, SSRF 공격과 서버 측 브라우저 공격으로 서버 상태 정보를 PDF로 노출했다.

PDF24는 서버 측 브라우저로 최신 버전 120의 헤드리스 크롬을 사용한다. 지속적인 버전 업데이트로 브라우저 자체의 취약점에 대비하고 있지만, 내부 IP 필터링에는 여전히 소홀하다.

4.4.2.2 B App 사례

B App은 웹페이지를 스크린 캡처하는 서비스를 제공한다. 점검한 결과 1) 서버 측 요청 URL에 대한 검증, 2) 내부 자원 자체의 보안성, 3) 브라우저 실행 환경의 안정성에서 문제점이 나타났다.

첫째, 서버 측 요청을 유발하는 URL에서 내부 IP를 차단하지 않는다. 특히, 서브리소스 URL에서는 파일 프로토콜도 차단하지 않아 서버 계정 정보가 노출된다.

둘째, Apache 관리 모듈의 기본 보안 설정으로 인해, 취약한 B App이 웹서버의 상태 정보를 노출한다 이는 Apache를 웹서버로 활용하는 Wave와 PDF24의 문제점과 동일하다.

셋째, 서버 측 브라우저가 웹 보안 정책을 준수하지 않는 심각한 상황이었다. 부모 문서의 스크립트가 iframe에 나타난 크로스 도메인 콘텐츠에 접근할 수 있었는데, 이는 서버 측 브라우저의 보안 기능이 비활성화되었기 때문이다. 보안 기능의 해제 여부는 크로스 도메인 콘텐츠를 가져오는 Fetch API의 성공적인 결과로도 확인할 수 있었다. iframe 내부에 나타난 계정 파일과 웹서버 정보는 브라우저의 낮은 보안 수준으로 인해 공격자 서버로 실시간 유출된다. 웹 보안 정책을 준수하지 않는 서버 측 브라우저와 내부 IP를 허용하는 취약점은 서버 측 브라우저 공격을 가능하게 하여, 내부 시스템에 큰 위협이 된다.

4.5 내부 자원의 잘못된 보안 정책

온프레미스 환경에서 활용도가 높은 Apache와 Nginx는 32개의 SSB 애플리케이션 중 50%를 호스팅하는 인기 있는 웹서버로 나타났다. 해당 웹서버에서 동작하는 SSB 애플리케이션 중 취약한 애플리케이션은 11개이고, 이 중 7개가 Apache 웹서버에서 호스팅되고 있었다. 내부 IP를 허용하는 9개의 SSB 애플리케이션을 통해 각각의 웹서버를 점검한 결과, Apache는 6개 중 5개, Nginx는 3개 중 1개가 미흡한 자체 보안 설정으로 인해 웹서버 정보가 노출되었다.

내부 자원은 외부 사용자의 접근으로부터 보호되어야 하며, 허용되지 않은 내부 애플리케이션에 의해 악용되어서도 안 된다. 내부 자원이 갖춘 보안 대책을 점검하기 위해 웹서버 관리 웹페이지의 URL을 페이로드로 사용했다. Apache와 Nginx 웹서버는

```
<ifmOdule mod_status.c>
  <Location /server-status>
    SetHandler server-status
    Require local
  </Location>
</IfModule>
```

Fig. 5. Access control for Apache's server-status path with default settings.

각각 /server-status, /nginx_status 경로를 통해 서버의 현재 상태를 모니터링할 수 있는 웹페이지를 제공한다. Apache에서는 mod_status, Nginx에서는 ngx_http_stub_status_module을 활성화해야 모니터링 웹페이지에 접속할 수 있다.

Fig. 5.는 Apache 관리 모듈의 기본 보안 설정을 보여준다. 이 설정은 로컬 서버로부터의 요청만을 허용하며, IP 기반의 접근 제어를 통해 허용된 장비의 IP 목록을 관리할 수 있다. 서버에서 모니터링하는 관리자는 모니터링 웹페이지를 통해 서버의 가동 시간, 활성화된 내장 모듈, 클라이언트 요청 정보, 서버 부하 관련 정보 등을 실시간으로 확인할 수 있다.

Apache의 기본 설정은 IP 기반으로 로컬 서버만이 관리 모듈에 접근할 수 있도록 되어 있다. 이 설정은 동일 서버 내에서 동작하는 애플리케이션의 접근 가능성을 고려하지 않는다. 해당 설정만으로는 서버 내부 관리자와 애플리케이션의 요청을 구분할 수 없다. 공격자는 로컬 서버에서 동작하는 취약한 애플리케이션을 악용하여 서버 정보를 노출시킬 수 있다. 예방 방법은 IP 기반의 접근 제어 외에도 사용자 인증 등의 보안 조치를 추가로 적용해야 한다. 웹서버 관리 모듈뿐만 아니라 데이터베이스, 내부 네트워크에서 동작하는 보안 모듈 등 다른 내부 자원에도 필요한 조치다.

V. 논 의

크롤러로 연구를 확장할 가능성과 서버 측 브라우저 악용에 의한 침해를 최소화하는 보호 대책을 제시한다.

5.1 취약한 브라우저 사용의 위험성

취약점이 공개된 구버전 브라우저의 사용은 공격자에게 새로운 공격 표면을 제공한다. 실험에서 서버 측 브라우저를 최신 상태로 유지하지 않는 웹 애플리케이션은 68%에 달하는 20개로 나타났다. 이는 시스템 보안에 대한 심각한 위협이 될 수 있다. Musch의 연구[1]에서 서버 측 브라우저의 오래된 버전 유지는 SSB 애플리케이션의 서비스 연속성, 버전 호환성 문제, 업데이트에 따른 잠재적 오작동 등의 이유로 발생한다고 분석했다. 서비스 제공자는 사용하는 서버 측 브라우저의 보안 취약점에 주의하고, 정기적으로 업데이트를 수행하여 최신 버전을 유지함으로써 보안 위협에 대응해야 한다.

5.2 크롤러의 취약 가능성

URLScan은 사용자가 제공한 URL의 웹페이지에서 위협 요소를 식별하는 데 서버 측 브라우저를 활용한다. 제공한 URL은 URLScan 플랫폼을 이용하는 사용자에게 실시간으로 공유된다. 공격자 서버가 호스팅하는 웹페이지의 URL을 점검 대상 URL로 제출한 결과, 약 한 시간 동안 공격자 서버는 URLScan 플랫폼 사용자로부터의 요청을 수신했다. 수신한 각 HTTP 요청의 사용자-에이전트 값은 데스크톱, 모바일 등 다양한 기기의 브라우저 정보를 나타냈다. 이 중 Fig. 6.에 명시한 사용자-에이전트를 갖는 브라우저의 요청에서 리눅스 계열 운영체제의 사용자 계정 정보가 공격자 서버로 전송되었다. 이는 웹페이지를 통해 전달된 공격 코드가 서버 측 브라우저에서 정상적으로 실행되었음을 나타내며, 보안 취약점이 특정 브라우저 환경에서 발생할 수 있음을 시사한다.

URLScan 플랫폼 사용자가 공격자 서버로부터 다운로드한 웹페이지는 iframe에 노출된 로컬 파일의 내용을 읽어 공격자 서버로 전송하도록 설계되어 있다. 보안에 취약한 환경에서 동작하는 브라우저는 사용자 계정 정보를 공격자 서버로 전송하게 된다. 수신된 정보는 분석을 통해 서버 측 브라우저의 보안 상태를 평가할 수 있다. 이 접근 방법은 PDF와 같은 서비스 결과물로 취약점을 점검할 수 없는 크롤러를 대상으로, 서버 측 브라우저에 대한 보안 취약성 연구를 확대할 수 있는 기반을 제공한다. 인공지능 기술의 발전과 데이터 수집 기법의 중요성이 증가하면서 크롤러의 활용도가 높아지고 있다. 크롤러는 주로 데이터 수집과 같은 특정 목적을 위해 독립적으로 운영되므로, 웹 호스팅을 통해 제공되는 SSB 애플리케이션에 비해 보안 측면에서 더 취약할 수 있다.

Fig. 6. User browser on the URLScan.io platform that exposed the file contents.

5.3 SSB 애플리케이션에 대한 보안 대책

SSB 애플리케이션 중 약 44%에 해당하는 14개가 SSRF와 서버 측 브라우저 공격에 취약한 점은 서버 측 요청 URL에 대한 검증의 중요성을 강조한다. 웹페이지 내에 입력 폼뿐만 아니라 API를 통해서도 서비스를 제공하는데, 다양한 접근 수단에 적용된 보안 검증 기준이 일관되지 않는 사례가 있음을 확인했다. 일관된 보안 검증 프로세스는 최종 목적지로의 리디렉션 응답과 웹페이지 내부에서 발생하는 서브리소스 요청에 대해서도 적용해야 한다. 보안 검증 절차의 일원화는 시스템의 보안성을 강화하고, 관리의 효율성을 높이는 데 기여한다.

외부 사용자의 접근이 차단된 내부 자원의 접근 가능성을 평가하기 위해 웹서버 관리 모듈의 URL을 이용했다. Apache 웹서버는 내부 IP를 허용하는 SSB 애플리케이션에 의해 서버 정보가 노출되었는데, 원인은 웹서버 관리 모듈 자체의 잘못된 보안 설정에 있었다. 내부 정보의 노출을 예방하기 위해 내부 자원은 사설 네트워크 내부의 요청에 대해서도 제로 트러스트 기반의 보안 대책을 적용해야 한다. 요청마다 사용자와 권한을 인증하는 메커니즘을 적용하고, 유효한 요청에만 응답해야 한다.

서버 측 브라우저의 활용으로 인한 보안 위협에 대응하기 위해 브라우저 격리[13] 기술을 채택할 수 있다. 브라우저 격리는 클라이언트 측 브라우저의 보안 대책으로, 통신 중인 웹 트래픽을 신뢰하지 않는 제로 트러스트를 기반으로 한다. 브라우저 격리 기술의 한 예로, 컨테이너 기술을 활용하여 브라우저를 독립된 환경에서 실행할 수 있다. 이를 통해 브라우저의 취약점이 다른 시스템 자원에 미치는 영향을 최소화할 수 있다. 브라우저 격리는 내부 네트워크와 데이터의 보안을 유지하는 효과적인 전략으로, 서버 측 브라우저 사용에 따른 보안 위협을 완화하는데 크게 기여할 수 있다.

5.4 연구 윤리

연구의 윤리적 책임을 다하기 위해, 본 연구에서 취약점이 확인된 14곳의 웹 서비스 제공자들에게 2024년 2월 16일에 해당 취약점을 통보하였다. 서비스 제공자들은 통보를 통해 취약점을 인지하고, 필요한 보안 조치를 취할 수 있다. 이 중 4곳은 게시된 연락처가 없거나 잘못된 정보로 인해 통보할 수 없었으므로, 본 논문에서는 익명 처리하였다. Wave는 우리의 보고에 즉각적으로 응답하고 패치하였다. ScreenshotsCloud는 패치를 통해 파일 프로토콜을 차단했지만, 우리에게 응답은 없었다. 그 외 SSB 애플리케이션은 여전히 취약한 상태다.

VI. 관련 연구

보안 분야에서는 헤드리스 브라우저를 취약점 탐지 도구로 사용하는 연구[14, 15]가 일반적이며, 서버 측 브라우저 사용과 관련된 보안 위협에 대한 연구는 드물다. Musch의 연구[1]는 서버 측에서 사용하는 오래된 버전의 브라우저가 공격 대상이 될 수 있음을 지적하며, 버전별로 상이한 브라우저의 기능을 핑거프린팅하여 서버 측 브라우저의 버전을 결정하는 방법을 제시한다. 이는 서버 측 브라우저 공격의 두 가지 모델 중 구버전 브라우저의 공개된 취약점을 악용하는 공격 모델이다. 본 연구는 브라우저의 잘못된 실행 환경으로 서버 측 브라우저 공격이 발생할 수 있음을 제기하고, 상용 SSB 애플리케이션의 취약 사례로 이를 증명하였다. Contrast Security의 연구원들은 원격 디버깅 기능이 활성화된 헤드리스 브라우저를 악용하여 로컬 장비에서 임의의 파일을 읽고 쓸 수 있는 취약점을 보고했다[16]. 취약한 브라우저는 크롬 94버전으로 서버 측 브라우저로 활용되는 경우 공격자에 의해 악용될 수 있다. Pellegrino의 연구[17]는 서버 측 요청을 활용하는 애플리케이션의 인터프리터 기능을 악용하여 대량의 요청을 발생시키고, 서버 자원을 고갈시키는 DoS

공격 모델을 제시한다. 해당 연구는 본 연구에서 제시한 취약한 환경의 서버 측 브라우저가 DoS 공격에 악용될 수 있음을 나타낸다.

Jabiyev의 연구[18]에서는 SSRF를 예방하기 위해 서버 측 요청을 격리된 서버에서 처리하는 방법을 제안한다. 웹 애플리케이션으로 전달되는 요청에 서버 측 요청을 유발하는 URL이 포함되어 있는지 식별하기 위해 역 프록시 서버를 배치한다. 해당 URL이 존재하면, 사용자 요청을 내부로의 요청이 차단된 격리된 서버로 전송하여 처리한다. 역 프록시 서버는 클라이언트의 단일 요청을 처리하는 데에는 효과적이지만, 내부에 위치한 서버 측 브라우저의 요청을 처리하지는 못한다.

VII. 결 론

서버 측 브라우저는 브라우저 자동화의 중요 도구이고, 렌더링과 자바스크립트 실행이 필요한 서비스에 사용된다. 브라우저가 서버에서 동작하며 새로운 요청을 발생시키고 코드를 실행하는 점에서 여러 보안 위협을 내포한다.

본 논문에서는 서버 측 브라우저를 안전하게 활용하는 데 필요한 보안 요구사항을 규정하고 상용 SSB 애플리케이션의 안전성을 검증했다.

브라우저가 요청하는 데 필요한 URL은 1) 사용자로부터 제공받는 시점, 2) 웹페이지를 요청하는 시점, 3) 웹페이지를 렌더링하는 시점에서 필터링해야 하고, 웹 보안 정책하에 동작하는 최신 브라우저를 활용해야 한다. 이는 내부 시스템에 대한 공격 목표 설정을 차단하고 스크립트 기반 공격을 예방한다. SSB 애플리케이션의 보안 대책과 별도로, 내부 자원들은 제로 트러스트 모델의 보안 대책을 적용하여 임의 접근을 방지해야 한다.

상용 SSB 애플리케이션들이 보안 요구사항을 만족하고 있는지 점검한 결과, SSRF와 서버 측 브라우저 공격에 취약한 사례를 발견했다. 안전성을 평가한 32개 중 절반에 가까운 애플리케이션이 적절한 보안 대책을 갖추지 못했다. 이는 서버 측 요청 URL에 대한 필터링의 부재와 미흡이 원인이었다. 웹 애플리케이션이 내부 IP 주소를 허용하는 경우, 취약점을 통해 내부 자원에 임의로 접근할 수 있었다. 브라우저의 보안 기능을 해제한 채 서버 측 브라우저를 활용하는 애플리케이션의 존재도 확인하였다.

서버 측 요청을 유발하는 URL이 유입될 수 있는 경로를 모두 예측하여 필터링하는 보안 대책은 현실적으로 어렵다. 해결 방안으로, 서버 측 브라우저를 활용할 때 내부 시스템과 분리하여 브라우저가 독립적인 환경에서 실행되도록 브라우저 격리 기술을 적용할 것을 제안하였다.

References

[1] M. Musch, R. Kirchner, M. Boll and M. Johns, "Server-side browsers: exploring the web's hidden attack surface," Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security, pp. 1168 - 1181, May. 2022.

[2] Wave, "Web accessibility evaluation tools," https://wave.webaim.org, Aug. 2024.

[3] PDF24, "Free PDF solutions for all PDF problems," https://tools.pdf24.org, Aug. 2024.

[4] Tranco, "A research-oriented top sites ranking hardened against manipulation - 06 November 2023," https://tranco-list.eu, Nov. 2023.

[5] OWASP, "Owasp top 10," https://owasp.org/www-project-top-ten, Aug. 2024.

[6] OWASP, "Server-side request forgery prevention cheat sheet," https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html, Aug. 2024.

[7] Similarweb, "Check and analyze any website," https://www.similarweb.com, Aug. 2024.

[8] URLScan, "URL and website scanner," https://urlscan.io, Aug. 2024.

[9] Glitch, "The friendly community where everyone builds the web," https://glitch.com, Aug. 2024.

[10] Github, "Sources used in the research," https://github.com/zzyo1/server-side-browsers, Aug. 2024.

[11] AccuWebHosting, "Web server infor-

mation tool," https://www.accuwebhosting.com/resources/show-web-server-detail, Aug. 2024.
〔12〕 Xurlfind3r, "Passive urls discovery utility," https://github.com/hueristiq/xurlfind3r, Aug. 2024.
〔13〕 Cloudflare, "What is browser isolation," https://www.cloudflare.com/learning/access-management/what-is-browser-isolation, Aug. 2024.
〔14〕 H. Choi, S. Hong, S. Cho and Y.-G. Kim, "Hxd: hybrid xss detection by using a headless browser," Proceedings of the 2017 4th International Conference on Computer Applications and Information Processing Technology, pp. 1-4, Aug. 2017.
〔15〕 C. Lv, L. Zhang, F. Zeng and J. Zhang, "Adaptive random testing for xss vulnerability," Proceedings of the 2019 26th Asia-Pacific Software Engineering Conference, pp. 63-69, Dec. 2019.
〔16〕 Chromium, "Clickjacking rce of chrome headless with remote debugging," https://issues.chromium.org/issues/40056642, Jul. 2021.
〔17〕 G. Pellegrino, O. Catakoglu, D. Balzarotti, and C. Rossow, "Uses and abuses of server-side requests," Proceedings of the 2016 International Symposium on Research in Attacks, Intrusions, and Defenses, pp. 393-414, Sep. 2016.
〔18〕 B. Jabiyev, O. Mirzaei, A. Kharraz, and E. Kirda, "Preventing server-side request forgery attacks," Proceedings of the 36th Annual ACM Symposium on Applied Computing, pp. 1626-1635, Mar. 2021.

〈저자소개〉

이 민 상 (Min-sang Lee) 학생회원
2022년 8월~현재: 성균관대학교 소프트웨어학과 석사과정
〈관심분야〉 정보보호, 웹 보안

최 형 기 (Hyoung-kee Choi) 정회원
1992년 2월: 성균관대학교 전자공학과 졸업
1996년 2월: Polytechnic University in Brooklyn, NY 석사
2001년 2월: Georgia Institute of Technology in Atlanta, GA 박사
2001년 1월~2004년 12월: Lancope 근무
2004년 3월~현재: 성균관대학교 소프트웨어대학 교수
〈관심분야〉 네트워크 보안, 리버스 엔지니어링