(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년11월07일
(11) 등록번호 10-1672791
(24) 등록일자 2016년10월31일

(51) 국제특허분류(Int. Cl.)
G06F 21/57 (2013.01) *G06F 21/12* (2013.01)
G06F 21/56 (2013.01)
(52) CPC특허분류
G06F 21/577 (2013.01)
G06F 21/128 (2013.01)
(21) 출원번호 10-2015-0148744
(22) 출원일자 2015년10월26일
심사청구일자 2015년10월26일
(56) 선행기술조사문헌
KR101543237 B1*
KR1020140113013 A*
KR101526500 B1
KR1020060062882 A
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
김휘강
서울특별시 서초구 태봉로2길 65 404동 302호 (우면동,네이처힐4단지)
한미란
서울특별시 은평구 백련산로 160, 102동 1702호(응암동, 녹번역센트레빌)
(뒷면에 계속)
(74) 대리인
김등용, 김홍석

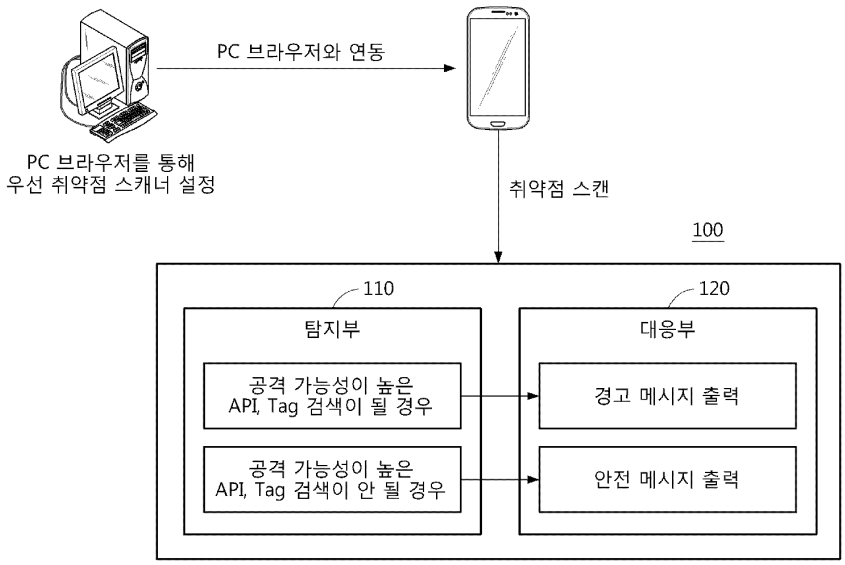
전체 청구항 수 : 총 17 항 심사관 : 구대성

(54) 발명의 명칭 **모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법 및 시스템**

(57) 요 약

본 발명은 모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법에 관한 것으로서, 일실시예에 따른 취약점 탐지 방법은 스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계, 및 상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함한다.

대 표 도



(52) CPC특허분류
G06F 21/563 (2013.01)
(72) 발명자
곽병일
경기도 고양시 일산서구 킨텍스로 456 103동 501호 (일산동,후곡마을1단지아파트)
김하나
경기도 의왕시 약수터1길 31 102호 (내손동)

이 발명을 지원한 국가연구개발사업
과제고유번호 B0101150230000400 3
부처명 미래창조과학부
연구관리전문기관 정보통신기술진흥센터
연구사업명 정보통신방송연구개발사업
연구과제명 스크립트 기반 사이버 공격 사전 예방 및 대응 기술 개발
기 여 율 1/1
주관기관 고려대학교 산학협력단
연구기간 2015.03.01 ~ 2016.02.29

명 세 서

청구범위

청구항 1

삭제

청구항 2

삭제

청구항 3

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계; 및

상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함하고,

상기 취약점을 스캔하는 단계는,

블랙리스트로 설정된 유알엘(URL, uniform resource locator) 여부를 확인하여 취약점을 스캔하는 단계를 포함하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법.

청구항 4

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계; 및

상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함하고,

상기 취약점을 스캔하는 단계는,

DDoS(Distributed Denial of Service) 공격 취약점 탐지 알고리즘을 이용하여, 브라우저 상에 보이는 웹 페이지에 대하여 반복문 내부에서 특정 함수에 대한 사용여부, 기능 중에서 적어도 하나를 확인하는 단계를 포함하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법.

청구항 5

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계; 및

상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함하고,

상기 취약점을 스캔하는 단계는,

현재 방문 중인 웹 페이지의 HTML 소스코드에서 쿠키 값으로의 접근 여부를 검색하는 단계; 및

쿠키에 접근하면서 특정 함수를 보내는 소수가 존재하는지 여부를 확인하는 단계를 포함하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법.

청구항 6

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계; 및

상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함하고,

상기 취약점을 스캔하는 단계는,

웹 쉘(Web shell)을 이용한 공격 탐지 알고리즘을 이용하여, 클라이언트의 웹 페이지에서 파일 업로드 시 파일 확장자를 검사하고, 상기 검사 결과에 따라 웹 쉘(Web shell)의 서버접근을 방지하는 단계;

웹 페이지의 HTML 소스코드에서 파일 업로드 기능을 사용하고 있는지 확인하는 단계; 및

상기 확인 결과 파일 업로드를 사용하는 경우 HTML5의 억셉트(accept) 코드를 사용하여 파일 업로드 시 확장자를 제한하는지 여부를 확인하는 단계를 포함하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법.

청구항 7

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계; 및

상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함하고,

상기 취약점을 스캔하는 단계는,

프로토콜/스킴(Protocol/Schema) 취약점 탐지 알고리즘을 이용하여,

브라우저 상에 보이는 웹 페이지에 대하여 registerProtocolHandler 기능을 사용하여 유알아이(URI, Uniform Resource Identifier)에 스킴(scheme)을 연결하는 단계; 및

상기 연결에 기초하여 스킴(scheme)이 사용되었는지 검사하는 단계를 포함하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법.

청구항 8

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계; 및

상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함하고,

상기 취약점을 스캔하는 단계는,

로컬 데이터베이스(Local DB) SQL 취약점 탐지 알고리즘을 이용하여,

브라우저 상에 보이는 웹 페이지에 대하여 openDatabase와 executeSql 기능을 사용하여 로컬 데이터베이스에 연결하고, 상기 연결에 기초하여 로컬 데이터베이스에 접근하였는지 검사하는 단계를 포함하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법.

청구항 9

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계; 및

상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함하고,

상기 취약점을 스캔하는 단계는,

파일 다운로드 취약점 탐지 알고리즘을 이용하여,

파일 다운로드 시의 유알엘(URL, Uniform Resource Locator) 주소를 확인하여 파일 다운로드 스크립트를 이용하는지 여부를 확인하는 단계; 및

상기 파일 다운로드 시의 유알엘(URL, Uniform Resource Locator) 주소와 관련한 첨부파일 하이퍼링크 부분을 확인하는 단계를 포함하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법.

청구항 10

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계; 및

상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함하고,

상기 취약점을 스캔하는 단계는,

현재 방문 중인 웹 페이지의 HTML 소스코드에서 입력 값에 대한 검증을 하고 있는지를 확인하는 단계; 및

상기 확인 결과, 입력 값에 대한 검증을 수행하는 경우에 그룹화 되어 있거나 반복적인 값을 입력 받는 패턴이 있는지를 확인하는 단계를 포함하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법.

청구항 11

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계; 및

상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함하고,

상기 취약점을 스캔하는 단계는,

CSRF(Cross Site Request forgery) 공격 탐지 알고리즘에 기반하는 속성 확인을 통해 CSRF(Cross Site Request forgery) 관련 악성 스크립트가 삽입되어 있는지 확인하는 단계를 포함하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법.

청구항 12

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계; 및

상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함하고,

상기 취약점을 스캔하는 단계는,

웹 소켓(Web socket)을 이용한 공격 탐지 알고리즘에 기반하여 원격 시스템의 포트 상태 및 필터링 유무에 따른 값의 지속시간을 확인하는 단계를 포함하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법.

청구항 13

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계; 및

상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함하고,

상기 취약점을 스캔하는 단계는,

CORS(Cross-Origin Resource Sharing) 공격 탐지 알고리즘을 이용하여, 현재 방문 중인 웹 페이지의 HTML 소스코드에서 CORS(Cross-Origin Resource Sharing) 기능을 사용하는지에 대해 검사하는 단계를 포함하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법.

청구항 14

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계; 및

상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함하고,

상기 취약점을 스캔하는 단계는,

브라우저 상에 보이는 웹 페이지에 대하여 파일 API기반의 클릭 재킹(Click jacking) 사용 여부를 검출하는 단계를 포함하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법.

청구항 15

삭제

청구항 16

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계; 및

상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함하고,

상기 취약점을 스캔하는 단계는,

현재 방문 중인 웹 페이지의 HTML 소스코드에서 웹 워커(Web Worker)가 사용하고 있는지 확인하는 단계;

상기 확인 결과, 웹 워커(Web Worker)를 사용하고 있을 경우 XMLHttpRequest와 관련된 함수 호출을 사용여부를 검색하는 단계를 포함하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법.

청구항 17

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계; 및

상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함하고,

상기 취약점을 스캔하는 단계는,

현재 방문 중인 웹 페이지의 HTML 소스코드를 검색하여 위치 정보를 취득하는 함수가 존재하는지 검색하는 단계;

상기 검색 결과, 다크 웹 사이트(Dark Web site)에 억세스되었는지 여부를 확인하는 단계를 포함하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법.

청구항 18

삭제

청구항 19

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계; 및

상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함하고,

상기 취약점을 스캔하는 단계는,

현재 방문 중인 웹 페이지의 HTML 소스코드로부터 HTML5의 크로스 문서 메시징(Cross Document Messaging) 기능과 관련된 단어 정보를 습득하고, 스크립트에서 오리진 검사 유무와 위치를 체크하는 단계를 포함하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법.

청구항 20

제3항 내지 제14항, 제16항, 제17항, 및 제19항 중 어느 하나의 항에 기재된 방법을 수행하기 위한 프로그램이 기록된 컴퓨터로 판독 가능한 기록 매체.

청구항 21

삭제

청구항 22

스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격에 대해, 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 탐지부; 및

상기 스캔 결과에 따라 경고 메시지 및 안전 메시지 중에서 적어도 하나를 출력하는 대응부를 포함하고,

상기 탐지부는,

DDoS 공격 취약점 탐지 알고리즘, Cookie sniffing 공격 탐지 알고리즘, Web shell을 이용한 공격 탐지 알고리즘, Protocol/Schema 취약점 탐지 알고리즘, Local DB SQL 취약점 탐지 알고리즘, 파일 다운로드 취약점 탐지 알고리즘, ReDos (Regular Expressions Denial of Service) 공격 탐지 알고리즘, CSRF 공격 탐지 알고리즘, Web socket을 이용한 공격 탐지 알고리즘, CORS 공격 탐지 알고리즘, File API 취약점 탐지 알고리즘, Web storage를 이용한 공격 탐지 알고리즘, Web worker를 이용한 공격 탐지 알고리즘, Geolocation 기능을 악용한 공격 탐지 알고리즘, 신규 속성 및 태그를 악용한 XSS 공격 탐지 알고리즘, 및 Cross Document messaging 공격 탐지 알고리즘 중에서 적어도 하나를 이용하여 상기 취약점을 스캔하는,

모바일 웹 애플리케이션 환경에서의 취약점 탐지 시스템.

청구항 23

삭제

발명의 설명

기 술 분 야

[0001] 본 발명은 사용 폭이 확대되고 있는 HTML5 모바일 웹 애플리케이션(Web Application) 환경에서 발생 가능한 HTML5 취약점을 사용자 측면에서 탐지하기 위해, 스마트 기기 상에서 HTML5 기반의 웹 애플리케이션의 HTML 코드를 체크한 후 비정상적인 자바스크립트 코드를 검사하는 기술적 사상에 관한 것이다.

배 경 기 술

[0003] 액티브 엑스와 같은 비표준 기술에 기반을 둔 인터넷 환경을 개선하기 위해 국제 웹 표준 활용기술인 HTML5의 전환이 진행되고 있다.

[0004] 웹 페이지를 만들기 위한 기본 프로그래밍 언어인 HTML5(Hyper Text Markup Language 5)는 HTML4보다 보안을 고려하여 설계되었지만, 새롭게 추가된 신기술들로 인해 웹 기반 취약점에 대한 공격 범위가 넓어졌으며, HTML4 환경에서 발생하던 다양한 보안 위협들을 그대로 상속하고 있다.

[0005] iOS(애플), 안드로이드(구글), 블랙베리 OS(RIM), 심비안(노키아) 등 서로 다른 환경에서 동작하는 애플리케이션을 제작하기 위해 플랫폼 별로 따로따로 제작하는 애플리케이션 방식보다 최근에는 HTML5 기반 모바일 웹(Web) 방식을 '행정안전부 고시 제2010-40호'에 따라 표준으로 권고하고 있다.

[0006] 그러나 HTML5 기반 웹 플랫폼의 발달은 XMLHttpRequest Level2 API, Thick Features, DOM API 등과 같이 다양

해진 기술만큼 그에 따른 취약점을 이용한 공격들도 많아졌다.

[0007] 이러한 공격들의 예로, CORS, CSRF, Web Messaging and Web Worker injections, Web Sockets Attacks 등이 있다.

[0008] 이에, HTML5 기반 웹 애플리케이션(Web Application) 사용 시, 악의적인 행위에 노출되기 전에 미리 예방할 수 있는 기술이 필요한 실정이다.

선행기술문헌

특허문헌

[0010] (특허문헌 0001) 대한민국 특허등록공보 제10-1525519호

(특허문헌 0002) 대한민국 특허등록공보 제10-1519381호

발명의 내용

해결하려는 과제

[0011] 본 발명은 스마트 기기 상에서 HTML5 기반의 웹 애플리케이션의 HTML 코드를 체크한 후 비정상적인 자바스크립트 코드를 검사하는 것을 목적으로 한다.

[0012] 본 발명은 크롬 확장 프로그램을 통하여 PC 브라우저에서 먼저 연동시킨 후 모바일 브라우저에서 사용을 하는 방식이기 때문에, 모바일 상에서 앱 다운 시 유사한 악성 앱 다운로드로 인한 피해를 줄이는 것을 목적으로 한다.

[0013] 본 발명은 스크립트 기반 공격과 HTML5 기능 악용 스크립트 공격에 대한 취약점 점검이 구분되어 있어 선택적, 통합적 스캔이 모두 가능하여, 악의적인 행위에 노출을 미리 예방하는 것이다.

과제의 해결 수단

[0015] 일실시예에 따른 취약점 탐지 방법은 스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격을 유형별로 분류하는 단계, 및 상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 단계를 포함한다.

[0016] 일실시예에 따른 취약점 탐지 시스템은 스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격에 대해, 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 탐지부, 및 상기 스캔 결과에 따라 경고 메시지 및 안전 메시지 중에서 적어도 하나를 출력하는 대응부를 포함한다.

[0017] 일실시예에 따른 프로그램은 스크립트 기반 공격과 HTML5의 기능을 악용한 악성 자바스크립트 공격을 유형별로 분류하는 명령어 세트, 및 상기 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔하는 명령어 세트를 포함한다.

발명의 효과

[0019] 실시예들에 따르면, 스마트 기기 상에서 HTML5 기반의 웹 애플리케이션의 HTML 코드를 체크한 후 비정상적인 자바스크립트 코드를 검사할 수 있다.

[0020] 본 발명에 따르면, 크롬 확장 프로그램을 통하여 PC 브라우저에서 먼저 연동시킨 후 모바일 브라우저에서 사용을 하는 방식이기 때문에, 모바일 상에서 앱 다운 시 유사한 악성 앱 다운로드로 인한 피해를 줄일 수 있다.

[0021] 본 발명에 따르면, 스크립트 기반 공격과 HTML5 기능 악용 스크립트 공격에 대한 취약점 점검이 구분되어 있어 선택적, 통합적 스캔이 모두 가능하여, 악의적인 행위에 노출을 미리 예방할 수 있다.

도면의 간단한 설명

[0023] 도 1a는 일실시예에 따른 취약점 탐지 시스템을 설명하는 도면이다.

도 1b는 본 발명의 일실시예에 따른 모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법을 설명하는

도면이다.

도 2a 내지 2c는 스크립트 기반 공격의 탐지 기술 중 DDoS 공격 취약점 탐지 알고리즘을 설명하는 도면이다.

도 3a 및 3b는 스크립트 기반 공격의 탐지 기술 중 Cookie sniffing 공격 탐지 알고리즘을 설명하는 도면이다.

도 4a 내지 4c는 스크립트 기반 공격의 탐지 기술 중 Web shell을 이용한 공격 탐지 알고리즘을 설명하는 도면이다.

도 5a 및 5b는 스크립트 기반 공격의 탐지 기술 중 Protocol/Schema 취약점 탐지 알고리즘을 설명하는 도면이다.

도 6a 내지 6c는 스크립트 기반 공격의 탐지 기술 중 Local DB SQL 취약점 탐지 알고리즘을 설명하는 도면이다.

도 7a 내지 7c는 스크립트 기반 공격의 탐지 기술 중 파일 다운로드 취약점 탐지 알고리즘을 설명하는 도면이다.

도 8a 및 8b는 스크립트 기반 공격의 탐지 기술 중 ReDos (Regular Expressions Denial of Service) 공격 탐지 알고리즘을 설명하는 도면이다.

도 9a 및 9b는 스크립트 기반 공격의 탐지 기술 중 CSRF 공격 탐지 알고리즘을 설명하는 도면이다.

도 10a 내지 10c는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 Web socket을 이용한 공격 탐지 알고리즘을 설명하는 도면이다.

도 11a 및 11b는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 CORS 공격 탐지 알고리즘을 설명하는 도면이다.

도 12a 내지 12c는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 File API 취약점 탐지 알고리즘을 설명하는 도면이다.

도 13a 및 13b는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 Web storage를 이용한 공격 탐지 알고리즘을 설명하는 도면이다.

도 14a 및 14b는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 Web worker를 이용한 공격 탐지 알고리즘을 설명하는 도면이다.

도 15a 및 15b는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 Geolocation 기능을 악용한 공격 탐지 알고리즘을 설명하는 도면이다.

도 16a 및 16b는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 신규 속성 및 태그를 악용한 XSS 공격 탐지 알고리즘을 설명하는 도면이다.

도 17a 내지 17c는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 Cross Document messaging 공격 탐지 알고리즘을 설명하는 도면이다.

발명을 실시하기 위한 구체적인 내용

[0024] 본 명세서에 개시되어 있는 본 발명의 개념에 따른 실시예들에 대해서 특정한 구조적 또는 기능적 설명들은 단지 본 발명의 개념에 따른 실시예들을 설명하기 위한 목적으로 예시된 것으로서, 본 발명의 개념에 따른 실시예들은 다양한 형태로 실시될 수 있으며 본 명세서에 설명된 실시예들에 한정되지 않는다.

[0025] 본 발명의 개념에 따른 실시예들은 다양한 변경들을 가할 수 있고 여러 가지 형태들을 가질 수 있으므로 실시예들을 도면에 예시하고 본 명세서에 상세하게 설명하고자 한다. 그러나, 이는 본 발명의 개념에 따른 실시예들을 특정한 개시형태들에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 변경, 균등물, 또는 대체물을 포함한다.

[0026] 제1 또는 제2 등의 용어를 다양한 구성요소들을 설명하는데 사용될 수 있지만, 상기 구성요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만, 예를 들어 본 발명의 개념에 따른 권리 범위로부터 이탈되지 않은 채, 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소는 제1 구성요소로도 명명될 수 있다.

[0027] 어떤 구성요소가 다른 구성요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성요소에

직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성요소가 다른 구성요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는, 중간에 다른 구성요소가 존재하지 않는 것으로 이해되어야 할 것이다. 구성요소들 간의 관계를 설명하는 표현들, 예를 들어 "~사이에"와 "바로~사이에" 또는 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.

[0028] 본 명세서에서 사용한 용어는 단지 특정한 실시예들을 설명하기 위해 사용된 것으로, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 설시된 특징, 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것이 존재함으로 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0029] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 갖는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0031] 이하, 실시예들을 첨부된 도면을 참조하여 상세하게 설명한다. 그러나, 특허출원의 범위가 이러한 실시예들에 의해 제한되거나 한정되는 것은 아니다. 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타낸다.

[0032] 도 1a는 일실시예에 따른 취약점 탐지 시스템(100)을 설명하는 도면이다.

[0033] HTML5는 다양한 서비스를 지원하는 플랫폼으로 많은 기능을 포함하고 있지만, 기존 모바일 애플리케이션에서 발생하는 문제들을 그대로 상속 받을 수 있다. 이에, 일실시예에 따른 취약점 탐지 시스템(100)은 최근 사용 폭이 확대되고 있는 HTML5 모바일 웹 애플리케이션(Web Application) 환경에서 발생 가능한 HTML5 취약점을 사용자 측면에서 탐지할 수 있다.

[0034] 사용자는 개별 계정으로 로그인한 후 구글 크롬 확장 프로그램을 설정하여 PC상에 우선적으로 브라우저에 연동시킨다. 또한, 스마트 기기 상에서 HTML5 기반의 웹 애플리케이션의 HTML 코드를 체크한 후 비정상적인 자바스크립트 코드를 검사할 수 있다.

[0035] 이를 위해, 취약점 탐지 시스템(100)은 탐지부(110)와 대응부(120)를 포함할 수 있다.

[0036] 먼저, 탐지부(110)는 스크립트 기반 공격과 HTML5의 기능과 관련한 악성 자바스크립트 공격에 대해, 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔한다. 예를 들어, 탐지부(110)는 스크립트 기반 공격과 HTML5 기능 악용 스크립트 공격에 대한 취약한 API, Tag를 웹 애플리케이션의 HTML 코드를 체크한 후 자바스크립트 코드를 검사하는 방식으로 취약점을 스캔할 수 있다.

[0037] 구체적인 예로, 탐지부(110)는 DDoS 공격 취약점 탐지 알고리즘, Cookie sniffing 공격 탐지 알고리즘, Web shell을 이용한 공격 탐지 알고리즘, Protocol/Schema 취약점 탐지 알고리즘, Local DB SQL 취약점 탐지 알고리즘, 파일 다운로드 취약점 탐지 알고리즘, ReDos (Regular Expressions Denial of Service) 공격 탐지 알고리즘, CSRF 공격 탐지 알고리즘, Web socket을 이용한 공격 탐지 알고리즘, CORS 공격 탐지 알고리즘, File API 취약점 탐지 알고리즘, Web storage를 이용한 공격 탐지 알고리즘, Web worker를 이용한 공격 탐지 알고리즘, Geolocation 기능을 악용한 공격 탐지 알고리즘, 신규 속성 및 태그를 악용한 XSS 공격 탐지 알고리즘, 및 Cross Document messaging 공격 탐지 알고리즘 중에서 적어도 하나를 이용하여 상기 취약점을 스캔할 수 있다.

[0038] 한편, 대응부(120)는 스캔 결과에 따라 경고 메시지 및 안전 메시지 중에서 적어도 하나를 출력할 수 있다. 예를 들어, HTML 소스 내 자바스크립트 코드를 검사하여 악성 스크립트가 삽입되어 있을 경우 경고 메시지를 출력할 수 있다.

[0039] 취약점 탐지 시스템(100)을 이용하면, 스마트 기기 상에서 HTML5 기반의 웹 애플리케이션의 HTML 코드를 체크한 후 비정상적인 자바스크립트 코드를 검사할 수 있다. 또한, 크롬 확장 프로그램을 통하여 PC 브라우저에서 먼저 연동시킨 후 모바일 브라우저에서 사용을 하는 방식이기 때문에, 모바일 상에서 앱 다운 시 유사한 악성 앱 다운로드로 인한 피해를 줄일 수 있고, 스크립트 기반 공격과 HTML5 기능 악용 스크립트 공격에 대한 취약점 점검이 구분되어 있어 선택적, 통합적 스캔이 모두 가능하여, 악의적인 행위에 노출을 미리 예방할 수 있다.

[0040] 도 1b는 일실시예에 따른 취약점 탐지 방법을 설명하는 도면이다.

[0041] 본 발명의 일실시예에 따른 모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법은 스크립트 기반 공격과 HTML5의 기능을 악용한 악성 자바스크립트 공격을 유형별로 분류한다(단계 101).

[0042] 유형별로 분류된 악성 자바스크립트 공격은 [표 1]과 같다.

[0044] [표 1]

스크립트 기반 공격	HTML5 기능 악용 스크립트 공격
DDoS 공격 취약점	Web socket을 이용한 공격 취약점
Cookie sniffing 취약점	CORS 공격 취약점
Web shell을 이용한 공격 취약점	File API 취약점
Protocol/Schema 취약점	Web storage를 이용한 공격 취약점
Local DB SQL 취약점	Web worker를 이용한 공격 취약점
파일 다운로드 취약점	Geolocation 기능을 악용한 공격 취약점
ReDos 공격 취약점	신규 속성 및 태그를 악용한 XSS 공격 취약점
CSRF 공격 취약점	Cross Document messaging 공격 취약점

[0045]

[0047] 본 발명의 일실시예에 따른 모바일 웹 애플리케이션 환경에서의 취약점 탐지 방법은 분류된 각각의 공격 유형별로 공격 가능성을 고려하여 취약점을 스캔한다(단계 102).

[0048] 본 발명에 따른 취약점 탐지 방법은 각각의 공격 유형별로 공격 가능성이 높은 API, Tag를 우선적으로 자바스크립트 코드 내에서 검색하거나 블랙리스트 설정된 유알엘(URL, uniform resource locator) 여부를 확인하는 방법으로 취약점을 스캔할 수 있다.

[0049] 예를 들어, 본 발명에 따른 취약점 탐지 방법은 개별 계정으로 로그인한 후 구글 크롬 확장 프로그램을 설정하여, PC상에 우선적으로 브라우저에 연동시킬 수 있다. 그리고, 스마트 기기 상에서 HTML5 기반의 웹 애플리케이션의 HTML 코드를 체크한 후 비정상적인 자바스크립트 코드를 검사할 수 있다.

[0050] 이하, 도 2 내지 도 17을 통해 취약점 스캔을 위한 다양한 알고리즘에 따른 구체적인 실시예를 설명한다.

[0051] 도 2a 내지 2c는 스크립트 기반 공격의 탐지 기술 중 DDoS 공격 취약점 탐지 알고리즘을 설명하는 도면이다.

[0052] 먼저, 도 2a에서 보는 바와 같이, 본 발명에 따른 취약점 탐지 방법은 스캔모듈의 클릭 입력에 응답하여 웹 페이지의 HTML 스크립트를 검색할 수 있다(단계 201).

[0053] 다음으로, 본 발명에 따른 취약점 탐지 방법은 DDoS(Distributed Denial of Service) 공격 취약점 탐지 알고리즘을 이용하여, 브라우저 상에 보이는 웹 페이지에 대하여 반복문 내부에서 특정 함수에 대한 사용여부, 기능 중에서 적어도 하나를 확인한다(단계 202).

[0054] 다음으로, 본 발명에 따른 취약점 탐지 방법은 반복문 내부에서 특정 함수를 포함하는 경우 미리 지정된 경고 메시지를 출력하고(단계 204), 특정 함수를 포함하지 않는 경우에는 안전 메시지를 출력할 수 있다(단계 203).

[0055] 도 2b를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 브라우저 상에 보이는 웹 페이지에 대하여 반복문, 예를 들어 for문 내부에서 XMLHttpRequest가 사용되었는지 검사하여 취약점으로 인한 DDoS 공격이 발생할 수 있음을 사용자에게 알려줄 수 있다.

[0056] 본 발명에 따른 취약점 탐지 방법은 현재 방문 중인 웹 페이지의 HTML 소스코드에서 반복문 속에서 XMLHttpRequest가 사용되었는지 검사한 후 사용자에게 해당 웹 페이지가 반복문 안에서 XMLHttpRequest의 어떤 기능을 사용하는지 알려준다.

[0057] 본 발명에 따른 취약점 탐지 방법은 정상적인 웹페이지 소스코드의 반복문 안에서 XML Request 기능이 사용되지 않았거나 오탐을 방지하기 위해 XML Request 기능이 사용되었더라도 반복문이 아니라면 탐지되지 않도록 한다.

[0058] 도 2c를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 악성 스크립트가 삽입된 웹 페이지 소스코드 반복문 안에서 XML Request 기능들이 사용된 경우 경고 메시지를 출력할 수 있다.

[0059] 도 3a 및 3b는 스크립트 기반 공격의 탐지 기술 중 Cookie sniffing 공격 탐지 알고리즘을 설명하는 도면이다.

[0060] 도 3a를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 스캔모듈의 클릭 입력에 응답하여 웹 페이지의 HTML 스크립트를 검색할 수 있다(단계 301).

[0061] 다음으로, 본 발명에 따른 취약점 탐지 방법은 Cookie sniffing 공격 탐지 알고리즘을 이용하여, 현재 방문 중인 웹 페이지의 HTML 소스코드에서 쿠키 값으로의 접근 여부를 검색하고, 쿠키에 접근하면서 특정 함수를 보내는 소수가 존재하는지 여부를 확인한다(단계 302). 예를 들어, Document.Cookie와 네트워크 패킷 전송과 관련된 Get.Post가 설정되어 있는지 여부를 확인할 수 있다.

[0062] 만약, 본 발명에 따른 취약점 탐지 방법은 Get.Post가 설정되어 있다면 경고 메시지를 출력하고(단계 304), 특정 함수를 포함하지 않는 경우에는 안전 메시지를 출력할 수 있다(단계 303).

[0063] 도 3b를 살펴보면, Cookie 파일은 주로 웹 기반의 환경에서 사용되며, 이용자가 본 내용, 상품 구매 내역, 신용카드 번호, 아이디(ID), 비밀번호, IP 주소 등의 중요한 정보를 가지고 있기 때문에, 유출 시 사생활 침해나 개인정보 유출이라는 심각한 피해를 초래할 수 있다. 본 발명에 따른 취약점 탐지 방법은 현재 방문 중인 웹 페이지의 HTML 소스코드에서 Cookie 값에 접근 여부를 검색하고, Cookie 에 접근하면서 Get, Post 등을 통해서 보내는 소스가 존재할 경우 경고 메시지를 출력한다.

[0064] 도 4a 내지 4c는 스크립트 기반 공격의 탐지 기술 중 Web shell을 이용한 공격 탐지 알고리즘을 설명하는 도면이다.

[0065] 도 4a를 참고하면, 본 발명에 따른 취약점 탐지 방법은 스캔모듈의 클릭 입력에 응답하여 웹 페이지의 HTML 스크립트를 검색할 수 있다(단계 401).

[0066] 다음으로, 본 발명에 따른 취약점 탐지 방법은 확장자 제한을 하는 코드가 설정되어 있는지 여부를 판단할 수 있다(단계 402).

[0067] 구체적으로, 본 발명에 따른 취약점 탐지 방법은 취약점을 스캔하기 위해, 웹 쉘(Web shell)을 이용한 공격 탐지 알고리즘을 이용하여, 클라이언트의 웹 페이지에서 파일 업로드 시 파일 확장자를 검사하고, 상기 검사 결과에 따라 웹 쉘(Web shell)의 서버접근을 방지하고, 웹 페이지의 HTML 소스코드에서 파일 업로드 기능을 사용하고 있는지 확인하며, 확인 결과 파일 업로드를 사용하는 경우 HTML5의 억셉트(accept) 코드를 사용하여 파일 업로드 시 확장자를 제한하는지 여부를 확인할 수 있다.

[0068] 만약, 본 발명에 따른 취약점 탐지 방법은 Get.Post가 설정되어 있다면 경고 메시지를 출력하고(단계 404), 특정 함수를 포함하지 않는 경우에는 안전 메시지를 출력할 수 있다(단계 403).

[0069] 도 4b를 살펴보면, 서버단에서 동작하는 Web Shell 스크립트는 파일 확장자를 체크하지 않는 게시판의 첨부파일 업로드를 통해 침투가 가능하다. 따라서 파일 업로드 시 php, cgi, asp, jsp 확장자가 업로드 되지 않도록 1차적으로 클라이언트의 웹 페이지에서 파일 업로드 시 파일 확장자를 검사하고, Web Shell의 서버접근을 방지한다.

[0070] 본 발명에 따른 취약점 탐지 방법은 현재 방문 중인 웹 페이지의 HTML 소스코드에서 파일 업로드 기능을 사용하고 있는지 확인하고, 파일 업로드를 사용하는 경우 html5의 accept 코드를 사용하여 파일 업로드 시 확장자를 제한하는 지 확인한다. 또한, 파일 업로드를 사용하나 억셉트(accept)를 사용하여 파일 업로드 확장자를 제한하지 않는 경우 취약한 웹 페이지로 진단하고 경고 메시지를 출력한다. 정상적인 웹 페이지인 경우 파일 업로드 시 html5에서 제공하는 억셉트(accept) 속성을 사용하여 특정 확장자만 업로드 할 수 있도록 제한된다.

[0071] 도 4c에서 보는 바와 같이, 본 발명에 따른 취약점 탐지 방법은 취약한 웹 페이지인 경우 파일 업로드 시 파일 확장자 제한이 없어 Web Shell 업로드가 가능하다.

[0072] 도 5a 및 5b는 스크립트 기반 공격의 탐지 기술 중 Protocol/Schema 취약점 탐지 알고리즘을 설명하는 도면이다.

[0073] 도 5a를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 스캔모듈의 클릭 입력에 응답하여 웹 페이지의 HTML 스크립트를 검색할 수 있다(단계 501).

[0074] 다음으로, 본 발명에 따른 취약점 탐지 방법은 registerProtocolHandler 기능을 사용하는지 여부를 판단할 수 있다(단계 502).

[0075] 구체적으로, 본 발명에 따른 취약점 탐지 방법은 프로토콜/스킴(Protocol/Schema) 취약점 탐지 알고리즘을 이용하여 브라우저 상에 보이는 웹 페이지에 대하여 registerProtocolHandler 기능을 사용하여 유알아이(URI, Uniform Resource Identifier)에 스킴(scheme)을 연결하고, 연결에 기초하여 스킴(scheme)이 사용되었는지 검사할 수 있다.

[0076] 본 발명에 따른 취약점 탐지 방법은 방법은 registerProtocolHandler 기능을 사용하는 경우 경고 메시지를 출력하고(단계 504), registerProtocolHandler 기능을 사용하지 않는 다면 안전 메시지를 출력할 수 있다(단계 503).

[0077] 도 5b를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 브라우저 상에 보이는 웹 페이지에 대하여 registerProtocolHandler 기능을 사용하여 URI에 scheme을 연결하고 scheme을 사용되었는지 검사하여 취약점으로 인한 Protocol/Schema 공격이 발생할 수 있음을 사용자에게 알려준다.

[0078] 도 6a 내지 6c는 스크립트 기반 공격의 탐지 기술 중 Local DB SQL 취약점 탐지 알고리즘을 설명하는 도면이다.

[0079] 도 6a는 본 발명에 따른 취약점 탐지 방법은 스캔모듈의 클릭 입력에 응답하여 웹 페이지의 HTML 스크립트를 검색할 수 있다(단계 601).

[0080] 다음으로, 본 발명에 따른 취약점 탐지 방법은 openDatabase와 executeSql 기능을 사용하여 Local DB에 접근하는지 여부를 판단할 수 있다(단계 602).

[0081] 본 발명에 따른 취약점 탐지 방법은 openDatabase와 executeSql 기능을 사용하여 Local DB에 접근하는 경우 경고 메시지를 출력하고(단계 504), Local DB에 접근하지 않는 경우에는 안전 메시지를 출력할 수 있다(단계 603).

[0082] 도 6b를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 브라우저 상에 보이는 웹 페이지에 대하여 openDatabase와 executeSql 기능을 사용하여 Local DB에 연결하고 DB에 접근하였는지 검사하여 취약점으로 인한 Local DB SQL 공격이 발생할 수 있음을 사용자에게 알려줄 수 있다.

[0083] 또한, 도 6c에서 보는 바와 같이, 본 발명에 따른 취약점 탐지 방법은 Local DB SQL 취약점 스캔 후 경고 메시지를 출력한다.

[0084] 도 7a 내지 7c는 스크립트 기반 공격의 탐지 기술 중 파일 다운로드 취약점 탐지 알고리즘을 설명하는 도면이다.

[0085] 도 7a를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 스캔모듈의 클릭 입력에 응답하여 웹 페이지의 HTML 스크립트를 검색할 수 있다(단계 701).

[0086] 본 발명에 따른 취약점 탐지 방법은 파일 다운로드 취약점 탐지 알고리즘을 이용하여, 파일 다운로드 시의 유알엘(URL, Uniform Resource Locator) 주소를 확인하여 파일 다운로드 스크립트를 이용하는지 여부를 확인하고, 파일 다운로드 시의 유알엘(URL, Uniform Resource Locator) 주소와 관련한 첨부파일 하이퍼링크 부분을 확인할 수 있다(단계 702).

[0087] 본 발명에 따른 취약점 탐지 방법은 파일 다운로드 시의 유알엘(URL, Uniform Resource Locator) 주소를 확인하여 파일 다운로드 스크립트를 이용하는 경우 경고 메시지를 출력하고(단계 704), 파일 다운로드 스크립트를 이용하지 않는 경우 안전 메시지를 출력할 수 있다(단계 703).

[0088] 도 7b를 살펴보면, 파일 다운로드 기능을 가진 웹 페이지가 접근할 수 있는 디렉토리에 대한 제한과 파일에 대한 접근 제한이 이루어지지 않게 되면 발생하게 되는 공격으로, 공격자는 다운로드 스크립트의 매개변수를 변경하면서 주요 파일 다운로드를 시도하여 웹 서버 내의 비공개 자료를 다운로드 받을 수 있다.

[0089] 파일 다운로드 취약점을 탐지할 수 있는 방안은 파일 다운로드 시의 URL 주소를 확인하여 파일 다운로드 스크립트를 이용하는지 여부를 확인하고, 파일 다운로드 URL 주소를 확인한 후에 첨부파일 하이퍼링크 부분을 확인한다.

[0090] 또한, 본 발명에 따른 취약점 탐지 방법은 7c에서 보는 바와 같이 첨부파일이 저장되어 있는 특정 디렉토리에 있는 파일만을 다운 받을 수 있게 다운로드 스크립트를 수정할 수 있다. 다운받기 위한 파일 이름에 "..", "/", "₩"와 같은 문자열이 들어오게 되면 필터링을 통해서 페이지를 불러올 수 없게 하는 방식을 취할 수 있다.

[0091] 도 8a 및 8b는 스크립트 기반 공격의 탐지 기술 중 ReDos (Regular Expressions Denial of Service) 공격 탐지

알고리즘을 설명하는 도면이다.

[0092] 도 8a를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 스캔모듈의 클릭 입력에 응답하여 웹 페이지의 HTML 스크립트를 검색할 수 있다(단계 801).

[0093] 다음으로, 본 발명에 따른 취약점 탐지 방법은 ReDos (Regular Expressions Denial of Service) 공격 탐지 알고리즘을 이용하여, 현재 방문 중인 웹 페이지의 HTML 소스코드에서 입력 값에 대한 검증을 하고 있는지를 확인하고(단계 802), 단계 802의 확인 결과, 입력 값에 대한 검증을 수행하는 경우에 그룹화 되어 있거나 반복적인 값을 입력 받는 패턴이 있는지를 확인할 수 있다(단계 804).

[0094] 만약, 단계 802의 확인 결과, 입력 값에 대한 검증을 하지 않는 경우거나, 단계 804의 확인 결과 입력 받는 패턴이 존재하지 않는 경우라면 안전 메시지를 출력할 수 있다(단계 803).

[0095] 본 발명에 따른 취약점 탐지 방법은 단계 804의 확인 결과, 입력 받는 패턴이 존재하는 경우라면 경고 메시지를 출력할 수 있다(단계 803).

[0096] 도 8b를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 클라이언트 단 또는 서버 단에서 입력 값에 대한 체크를 수행할 때, 악성 정규 표현식(그룹화되고 반복적인 패턴)을 사용하는 경우 클라이언트 화면 또는 서버가 Dos 공격을 받은 것과 같은 현상이 발생하므로 이를 탐지할 수 있다. 또한, 본 발명에 따른 취약점 탐지 방법은 현재 방문 중인 웹 페이지의 HTML 소스코드에서 new regexp 함수를 사용하여 입력 값에 대한 검증을 하고 있는 지 확인하고, new regexp를 사용하여 입력 값에 대한 검증을 수행하는 경우, 그룹화 되어 있거나 반복적인 값을 입력 받는 패턴이 있는 지 확인한다. 만약, 그룹화 되어 있거나, 반복적인 값을 제한 없이 사용하는 경우, 취약점 탐지 방법은 취약한 웹 페이지로 진단하고 경고 메시지를 출력한다.

[0097] 도 9a 및 9b는 스크립트 기반 공격의 탐지 기술 중 CSRF 공격 탐지 알고리즘을 설명하는 도면이다.

[0098] 도 9a를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 스캔모듈의 클릭 입력에 응답하여 웹 페이지의 HTML 스크립트를 검색할 수 있다(단계 801).

[0099] 다음으로, 본 발명에 따른 취약점 탐지 방법은 CSRF(Cross Site Request forgery) 공격 탐지 알고리즘에 기반하는 속성 확인을 통해 CSRF(Cross Site Request forgery) 관련 악성 스크립트가 삽입되어 있는지를 판단할 수 있다(단계 902).

[0100] 단계 902의 판단 결과, 악성 스크립트가 삽입되어 있다면 경고 메시지를 출력하고(단계 904), 악성 스크립트가 삽입되어 있지 않다면 안전 메시지를 출력할 수 있다(단계 903).

[0101] 도 9b를 살펴보면, 특정 웹사이트가 사용자의 웹 브라우저를 신용하는 상태를 이용하여, 로그인한 상태에서 사이트간 요청 위조 공격 코드가 삽입된 페이지를 열면, 공격 대상이 되는 웹사이트는 위조된 공격 명령이 믿을 수 있는 사용자로부터 발송된 것으로 판단하게 되어 공격에 노출될 수 있다. 본 발명에 따른 취약점 탐지 방법은 이러한 공격이 가능한 게시판 등을 스캔하여 경고 메시지 출력한다. 이를 위해, 본 발명에 따른 취약점 탐지 방법은 Iframe hidden 속성 확인, "image" 링크, From 태그 csrf_token 속성 확인을 통해 CSRF 관련 악성 스크립트가 삽입되어 있는지 확인할 수 있다.

[0102] 도 10a 내지 10c는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 Web socket을 이용한 공격 탐지 알고리즘을 설명하는 도면이다.

[0103] 도 10a를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 현재 웹 페이지 포트를 스캔하고(단계 1001), 특정 IP 입력 후 포트를 스캔한다(단계 1002).

[0104] 다음으로, 취약점 탐지 방법은 각 포트 별로 Open/Closed/Blocked의 상태를 확인할 수 있다(단계 1003). 예를 들어, 웹 소켓(Web socket)을 이용한 공격 탐지 알고리즘에 기반하여 각 포트 별로 원격 시스템의 포트 상태 및 필터링 유무에 따른 값의 지속시간을 확인하여 Open/Closed/Blocked의 상태를 확인할 수 있다.

[0105] 취약점 탐지 방법은, 확인 결과 포트가 Open되어 있는지를 판단하고(단계 1004), 포트가 Open되어 있다면 경고 메시지를 출력하고(단계 1004), 포트가 Open되어 있지 않다면 안전 메시지를 출력한다(단계 1005).

[0106] Web Socket을 이용한 취약점은 사용자의 허가 없이 다른 도메인으로 연결을 요청 가능하고 요청이 발생했음을 사용자에게 알려주지 않는다. 또한 사용자 모르게 브라우저를 통해 다른 도메인에 있는 시스템과 통신 채널을 형성 가능하다.

[0107] Web Socket은 연결하려는 객체의 상태정보를 의미하는 세 가지 값이 존재하는데, 최초 Web Socket이 생성되면 Connecting(0)이지만, 스캐닝 대상에 따라 Open(1) 혹은 Closed(2)로 변화한다. 아래 [표 2]와 같이, ReadyState의 값에 따라 일정 포트 또는 IP의 상태를 파악할 수 있으며, ReadState 값의 지속 시간에 따른 포트 상태 여부를 알 수 있다.

[0109] [표 2]

원격 시스템 포트 상태 및 필터링 유무	0값의 지속시간
원격 시스템의 포트가 열려있는 경우(open)	<100ms (0.1초 미만)
원격 시스템의 포트가 닫혀있는 경우(Closed)	~1000ms (1초)
패킷이 필터링 되었을 경우	>30000ms (30초 초과)

[0110]

[0111] 주요 포트에 해당하는 포트들에 대해서 웹소켓 기능을 이용한 포트 스캔을 진행한다. 웹 소켓을 이용한 포트스캔 결과를 각 포트 별로 Open/Closed/Blocked 상태를 사용자에게 리포트 한다. Open 포트를 발견하면 웹소켓을 이용한 포트스캔에 대한 취약점이 있다고 알려주고, 미 발견 시 취약점이 없다고 결과를 보여준다.

[0112] 취약점 검사 시 스캔 하는 주요 포트 리스트는 아래 [표 3]과 같다.

[0114] [표 3]

포트 번호	서비스	서비스 내용
69(UDP)	TFTP	인증고정 없이 단순하게 파일을 전송
80(TCP)	HTTP	웹 서비스
137(UDP)		NetBIOS 이름 해석 서비스
143	IMAP	메일 서버로 전송된 메일을 읽을 때 사용
161	SNMP	네트워크 관리와 모니터링
445(TCP/UDP)		Direct Host(SMB/CIFS over TCP)
514(UDP)		Syslog 서비스
587(TCP)	SMTP	메일을 보낼 때 사용하는 서비스(유동IP 사용시)
993(TCP)		SSL위의 IMAP4(암호화 전송)

[0115]

[0116] 도 10b에서는 현재 방문 중인 웹 페이지를 스캔하는 경우를 나타내고, 도 10c에서는 특정 IP를 입력한 후 스캔하는 경우를 나타낸다.

[0117] 도 10b에서 보는 바와 같이, 스캔 결과 80번 오픈 포트 발견으로 웹 소켓을 이용한 포트스캔 취약점이 존재함을 알 수 있다.

[0118] 도 10c에서는 스캔 결과 Open 포트의 발견으로 인해 취약점이 존재함을 확인할 수 있다.

[0119] 도 11a 및 11b는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 CORS 공격 탐지 알고리즘을 설명하는 도면이다.

[0120] 도 11a를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 스캔모듈의 클릭 입력에 응답하여 웹 페이지의 HTML 스크립트를 검색할 수 있다(단계 1101).

[0121] 다음으로, 취약점 탐지 방법은 CORS(Cross-Origin Resource Sharing) 공격 탐지 알고리즘을 이용하여, 현재 방문 중인 웹 페이지의 HTML 소스코드에서 CORS(Cross-Origin Resource Sharing) 기능을 사용하는지에 대해 검사할 수 있다(단계 1102).

[0122] 예를 들어, withCredentials="true"의 속성을 사용하는 경우 HTML 소스코드에서 CORS(Cross-Origin Resource Sharing) 기능을 사용한다고 판단할 수 있다.

[0123] withCredentials="true"의 속성을 사용한다면, 취약점 탐지 방법은 경고 메시지를 출력할 수 있고(단계 1104), withCredentials="true"의 속성을 사용하지 않는 다면 안전 메시지를 출력할 수 있다(단계 1103).

[0124] 도 11b를 살펴보면, CORS(Cross-Origin Resource Sharing)를 사용하기 위해서는 도메인간의 신뢰관계가 형성이 되어야 하는데 XHR Level2를 사용한 Request에는 COR을 발생시킨 도메인의 정보를 포함하는 Origin 헤더가 포함되어 있으며, 해당 Request의 Response에는 Access-Control-Allow-Origin 헤더를 포함하여 허용 여부를 확인하게 한다.

[0125] 쿠키정보가 포함된 CORS(Cross-Origin Resource Sharing)에 대한 응답은 Access-Control-Allow-Credentials: true가 포함되어야 한다. 공격자는 CORS(Cross-Origin Resource Sharing)를 사용하기 위해 맺은 신뢰관계를 이용하여 타겟 서버에 공격을 수행한다. Access-Control-Allow-Origin의 값을 '모두'로 허용할 경우 모든 도메인의 자원 공유 요청을 허용하는 것임으로 개인정보 탈취가 가능하다. 취약점 탐지 방법은 현재 방문 중인 웹 페이지의 HTML 소스코드에서 CORS(Cross-Origin Resource Sharing) 기능을 사용하는지에 대해 검사하고, CORS(Cross-Origin Resource Sharing)에 대하여 태그 범위 내에 withCredentials 속성을 이용하여 웹 페이지 탐색 결과를 사용자에게 알려줄 수 있다. withCredentials(옵션, 쿠키값을 남기기 위해 사용하는 기능)가 탐색되었을 경우에는 확실히 CORS(Cross-Origin Resource Sharing)가 사용됨을 알 수 있다.

[0126] 도 12a 내지 12c는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 File API 취약점 탐지 알고리즘을 설명하는 도면이다.

[0127] 도 12a를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 스캔모듈의 클릭 입력에 응답하여 웹 페이지의 HTML 스크립트를 검색할 수 있다(단계 1201).

[0128] 다음으로, 취약점 탐지 방법은 File API 취약점 탐지 알고리즘을 이용하여 FileReader 객체 사용과 투명도를 조절하는 Opacity가 0으로 설정되었는지 여부를 판단할 수 있다(단계 1202). 즉, 취약점 탐지 방법은 브라우저 상에 보이는 웹 페이지에 대하여 파일 API기반의 클릭 재킹(Click jacking) 사용 여부를 검출함으로써, 취약점을 탐지할 수 있다.

[0129] 만약, Opacity가 0으로 설정되었다면, API기반의 클릭 재킹(Click jacking) 사용이 확인되는 것이므로, 취약점 탐지 방법은 경고 메시지를 출력할 수 있다(단계 1204). 또한, Opacity가 0으로 설정되지 않았다면 취약점 탐지 방법은 안전 메시지를 출력할 수 있다(단계 1203).

[0130] 도 12b를 살펴보면, 취약점 탐지 방법은 브라우저 상에 보이는 웹 페이지에 대하여 File API기반의 Click jacking 사용 여부를 검출하고, FileReader 객체 사용과 Opacity (투명도 조절) 가 0으로 설정되어 있을 경우 악성 사이트로 판단하여 경고 메시지를 출력해준다. 이때, HTML 소스코드에서 FileReader 가 사용되지 않고, Opacity 설정이 "0"이 아니라면 정상으로 판단한다.

[0131] 한편, 도 12c에서 보는 바와 같이, HTML 소스코드에서 FileReader 가 사용되고, Opacity 설정이 "0"이면 비정상으로 판단할 수 있다.

[0132] 도 13a 및 13b는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 Web storage를 이용한 공격 탐지 알고리즘을 설명하는 도면이다.

[0133] 도 13a를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 스캔모듈의 클릭 입력에 응답하여 웹 페이지의 HTML 스크립트를 검색할 수 있다(단계 1301).

[0134] 다음으로, 취약점 탐지 방법은 Web storage를 이용한 공격 탐지 알고리즘을 이용하여 현재 방문 중인 웹 페이지의 HTML 소스코드에서 웹 스토리지(Web Storage)를 사용하기 위한 복수의 API들의 사용여부를 판단할 수 있다(단계 1302). 예를 들어, 취약점 탐지 방법은 스토리 API 중에서 Length, Key, getItem, setItem, removeItm, clear 중에서 어느 하나가 존재하는지를 판단하여 웹 스토리지(Web Storage)를 사용하기 위한 복수의 API들의 사용여부를 판단할 수 있다.

[0135] 만약, 위 6개의 API가 존재한다면 취약점 탐지 방법은 경고 메시지를 출력하고(단계 1304), 존재하지 않는 다면 안전 메시지를 출력할 수 있다(단계 1303).

[0136] 도 13b를 살펴보면, 취약점 탐지 방법은 브라우저 상에 보이는 웹 페이지에 대하여 Web Storage 사용여부를 검사하여 취약점으로 인한 개인정보 유출 가능성을 사용자에게 알려준다. 현재 방문 중인 웹 페이지의 HTML 소스코드에서 Web Storage를 사용하기 위한 API 6종의 사용여부를 검사하고, 6종의 API에 대하여 각각 사용 여부를 보여주어 사용자에게 해당 웹 페이지가 Web Storage의 어떤 기능을 사용하는지 알려준다. Web Storage를 사용하기 위한 API 6종은 storage.Length, storage.Key, storage.getItem, storage.setItem, storage.removeItem,

storage.clear이고, 악성 웹 페이지 내 데이터 추출을 위해 사용된 함수의 사용 여부를 알려준다.

[0137] 도 14a 및 14b는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 Web worker를 이용한 공격 탐지 알고리즘을 설명하는 도면이다.

[0138] 도 14a를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 스캔모듈의 클릭 입력에 응답하여 웹 페이지의 HTML 스크립트를 검색할 수 있다(단계 1401).

[0139] 다음으로, 취약점 탐지 방법은 Web worker를 이용한 공격 탐지 알고리즘을 이용하여 현재 방문 중인 웹 페이지의 HTML 소스코드에서 웹 워커(Web Worker)가 사용하고 있는지를 확인하고, 상기 확인 결과, 웹 워커(Web Worker)를 사용하고 있을 경우 XMLHttpRequest와 관련된 함수 호출을 사용여부를 검색한다(단계 1402).

[0140] 만약, 관련된 함수 호출을 사용하는 경우라면, 취약점 탐지 방법은 경고 메시지를 출력하고(단계 1404), 관련된 함수 호출을 사용하지 않는 경우라면, 안전 메시지를 출력할 수 있다(단계 1403).

[0141] 도 14b를 살펴보면, 취약점 탐지 방법은 현재 방문 중인 웹 페이지의 HTML 소스코드에서 Web Worker가 사용하고 있는지 확인한 후에, Web Worker를 사용 하고 있을 경우 XMLHttpRequest와 관련된 함수 호출을 사용여부를 검색하고 그 결과를 사용자에게 알려줄 수 있다. XMLHttpRequest와 관련된 함수는 web worker(), XDomainRequest(), open(), send(), XMLHttpRequest()를 포함하며, 사용했을 경우 사용한 함수에 대해서 '사용함'으로 표시되고, 사용하고 있지 않을 경우 해당 함수들을 '사용하지 않음'으로 표시될 수 있다.

[0142] 도 15a 및 15b는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 Geolocation 기능을 악용한 공격 탐지 알고리즘을 설명하는 도면이다.

[0143] 도 15a를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 스캔모듈의 클릭 입력에 응답하여 웹 페이지의 HTML 스크립트를 검색할 수 있다(단계 1501).

[0144] 다음으로, 취약점 탐지 방법은 Geolocation 기능을 악용한 공격 탐지 알고리즘을 이용하여 위치 정보를 취득할 수 있는 함수가 존재하는지 여부를 판단할 수 있다(단계 1502). 즉, 취약점 탐지 방법은 현재 방문 중인 웹 페이지의 HTML 소스코드를 검색하여 위치 정보를 취득하는 함수가 존재하는지 검색하고, 다크 웹 사이트(Dark Web site)에 억세스되었는지 여부를 확인할 수 있다.

[0145] 단계 1502의 판단 결과, 위치 정보를 취득할 수 있는 함수가 존재하는 경우 단계 1504와 같이 경고 메시지를 출력할 수 있고, 함수가 존재하지 않는 경우 단계 1503과 같이 안전 메시지를 출력할 수 있다.

[0146] 도 15b를 살펴보면, 특정 목적을 위하여 공개하지 않은 이상 사용자 동의 없이 웹 브라우저 사용자 시 GPS 정보가 노출되는지 확인하여 경고 메시지 출력한다. GPS Device와 인터넷 연결이 되지 않을 경우 Cache된 과거 위치 정보가 사용되어 웹 브라우저 사용자의 정보를 노출될 가능성이 있다. 이에, 취약점 탐지 방법은 현재 방문 중인 웹 페이지의 HTML 소스코드를 검색하여 위치 정보를 취득할 수 있는 함수 getCurrentPosition(), watchPosition(), cords, maximumAge가 존재하는지 검색하고, Dark Web site (피싱사이트, 도박사이트, 적대국 관련 사이트 등)에 Access가 되었을 경우 경고 메시지 출력한다. 만약, 화이트리스트가 있을 경우에는 리스트 내용과 비교할 수 있다.

[0147] 도 16a 및 16b는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 신규 속성 및 태그를 악용한 XSS 공격 탐지 알고리즘을 설명하는 도면이다.

[0148] 도 16a를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 스캔모듈의 클릭 입력에 응답하여 웹 페이지의 HTML 스크립트를 검색할 수 있다(단계 1601).

[0149] 다음으로, 취약점 탐지 방법은 HTML5 신규태그 중 audio, video, sorce를 사용하는지 여부를 판단할 수 있다(단계 1602).

[0150] 만약, HTML5 신규태그 중 audio, video, sorce를 사용한다면 취약점 탐지 방법은 경고 메시지를 출력하고(단계 1604), HTML5 신규태그 중 audio, video, sorce를 사용하지 않는다면 안전 메시지를 출력할 수 있다(단계 1603).

[0151] 도 16b에서 보는 바와 같이, 취약점 탐지 방법은 브라우저 상에 보이는 웹 페이지에 내 각각의 HTML5의 신규태그에 대하여 태그 범위 내에 <script> 또는 Javascript:를 이용한 스크립트 사용 여부를 검사하여 사용자에게 해당 웹 페이지가 HTML5 신규태그 위험 여부를 알려준다.

[0152] 도 17a 내지 17c는 HTML5 기능 악용 스크립트 공격의 탐지 기술 중 Cross Document messaging 공격 탐지 알고리즘을 설명하는 도면이다.

[0153] 도 17a를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 Cross Document Messaging과 관련된 단어 정보를 습득한다(단계 1701). 즉, 입력된 댓글 내용을 임의대로 변경하는 정보를 습득할 수 있다.

[0154] 다음으로, 본 발명에 따른 취약점 탐지 방법은 스캔모듈의 클릭 입력에 응답하여 웹 페이지의 HTML 스크립트를 검색할 수 있다(단계 1702).

[0155] 다음으로, 취약점 탐지 방법은 Origin 검사와 위치 검사를 하는지 여부를 확인할 수 있다(단계 1703).

[0156] 만약, Origin 검사와 위치 검사를 하는 경우 취약점 탐지 방법은 경고 메시지를 출력하고(단계 1705), Origin 검사와 위치 검사를 하지 않는 경우 안전 메시지를 출력할 수 있다(단계 1704).

[0157] 도 17b와 17c를 살펴보면, 본 발명에 따른 취약점 탐지 방법은 HTML5에서 제공하는 Cross Document Messaging 기능을 사용할 때 Origin검사를 하지 않으면 악용 가능성이 높기 때문에 이런 취약점이 있는지 조사하고 결과를 알려준다. 현재 방문 중인 웹 페이지의 HTML 소스코드를 읽어와 HTML5의 Cross Document Messaging 기능과 관련된 단어 정보를 습득하고, 스크립트에서 Origin검사 유무와 위치를 체크하여 취약한지 아닌지를 판단, 그 결과를 바탕으로 취약점 결과를 사용자에게 알려줄 수 있다.

[0158] 결국, 본 발명을 이용하면, 스마트 기기 상에서 HTML5 기반의 웹 애플리케이션의 HTML 코드를 체크한 후 비정상적인 자바스크립트 코드를 검사할 수 있다. 또한, 크롬 확장 프로그램을 통하여 PC 브라우저에서 먼저 연동시킨 후 모바일 브라우저에서 사용을 하는 방식이기 때문에, 모바일 상에서 앱 다운 시 유사한 악성 앱 다운로드로 인한 피해를 줄일 수 있고, 스크립트 기반 공격과 HTML5 기능 악용 스크립트 공격에 대한 취약점 점검이 구분되어 있어 선택적, 통합적 스캔이 모두 가능하여, 악의적인 행위에 노출을 미리 예방할 수 있다.

[0159] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0160] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(signal wave)에 영구적으로, 또는 일시적으로 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0161] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은

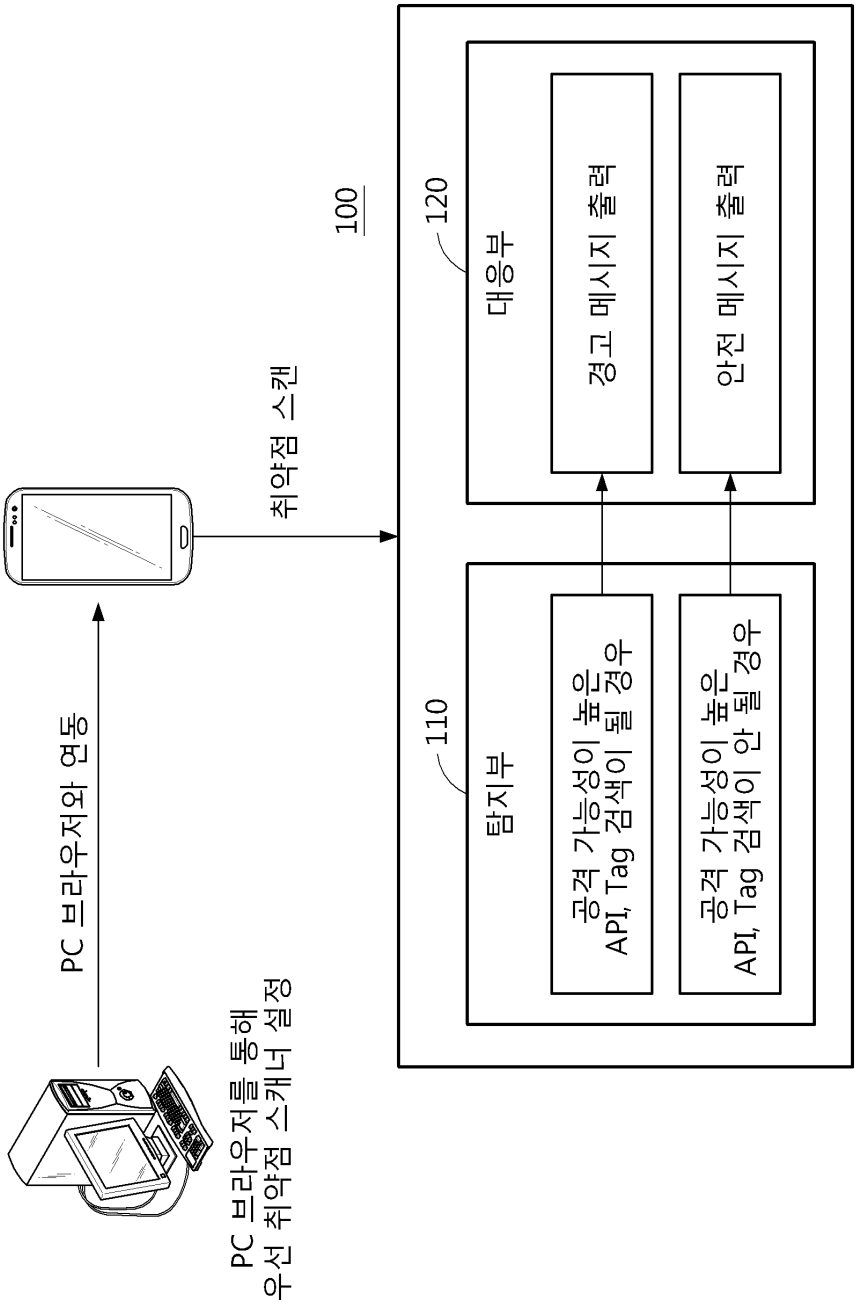
기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0162] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

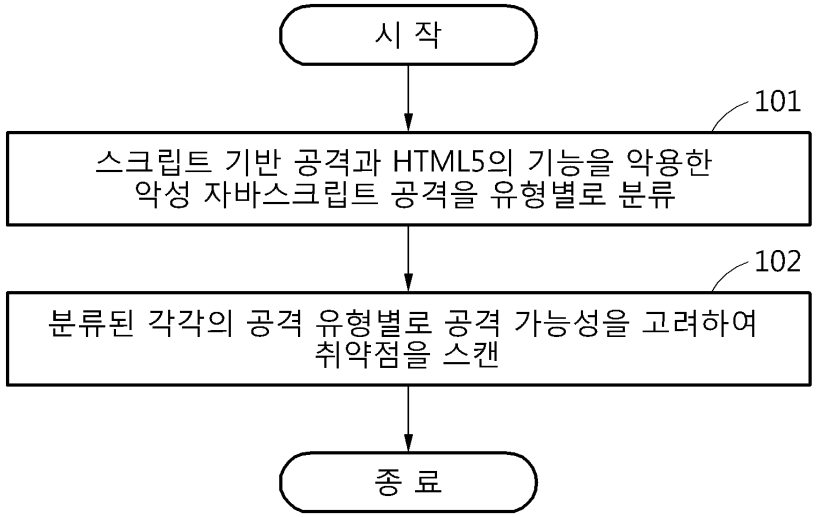
[0163] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

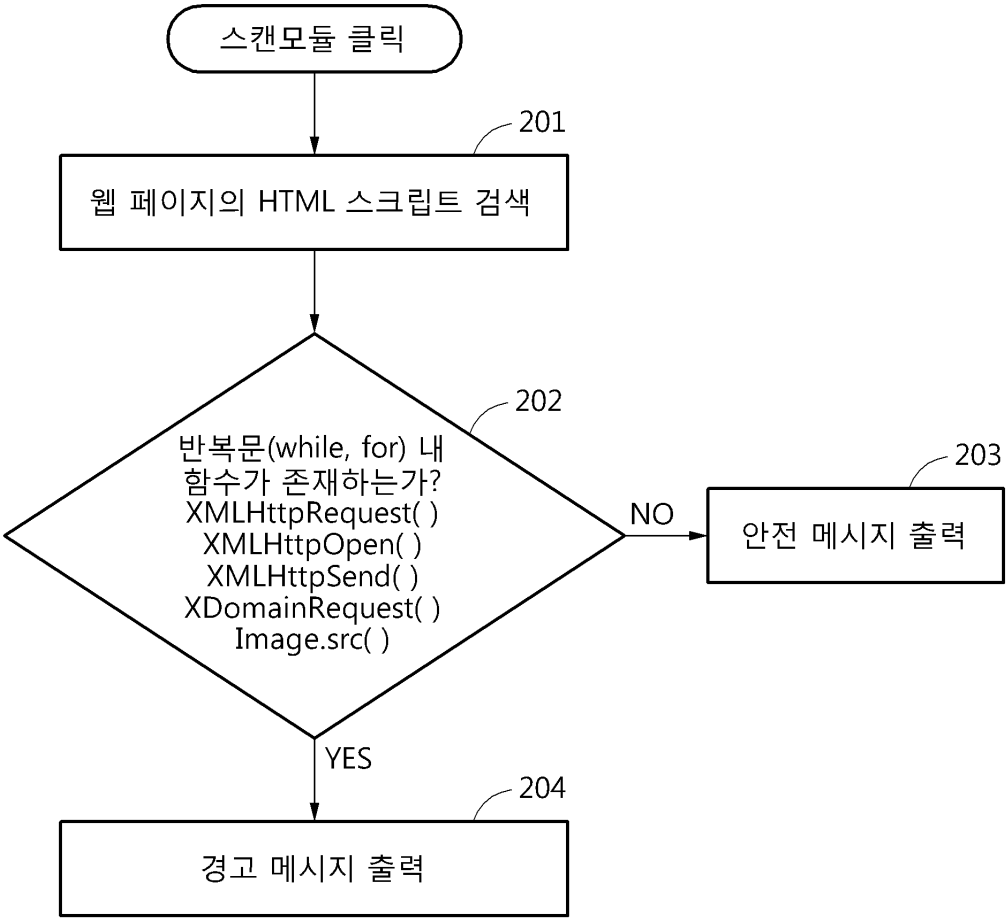
도면1a



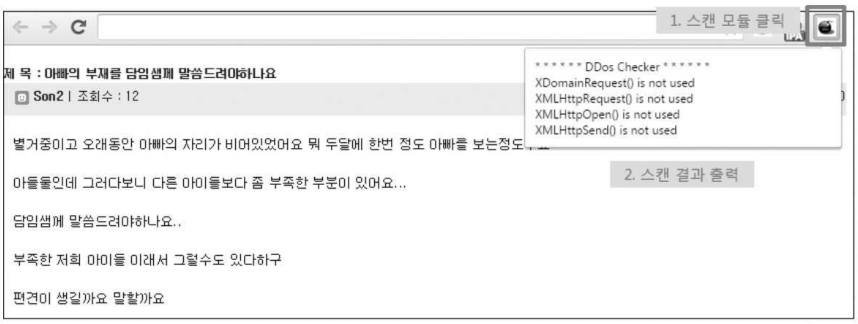
도면1b


시 작
101
스크립트 기반 공격과 HTML5의 기능을 악용한
악성 자바스크립트 공격을 유형별로 분류
102
분류된 각각의 공격 유형별로 공격 가능성을 고려하여
취약점을 스캔
종 료

도면2a


스캔모듈 클릭
201
웹 페이지의 HTML 스크립트 검색
202
반복문(while, for) 내
함수가 존재하는가?
XMLHttpRequest( )
XMLHttpOpen( )
XMLHttpSend( )
XDomainRequest( )
Image.src( )
NO
203
안전 메시지 출력
YES
204
경고 메시지 출력

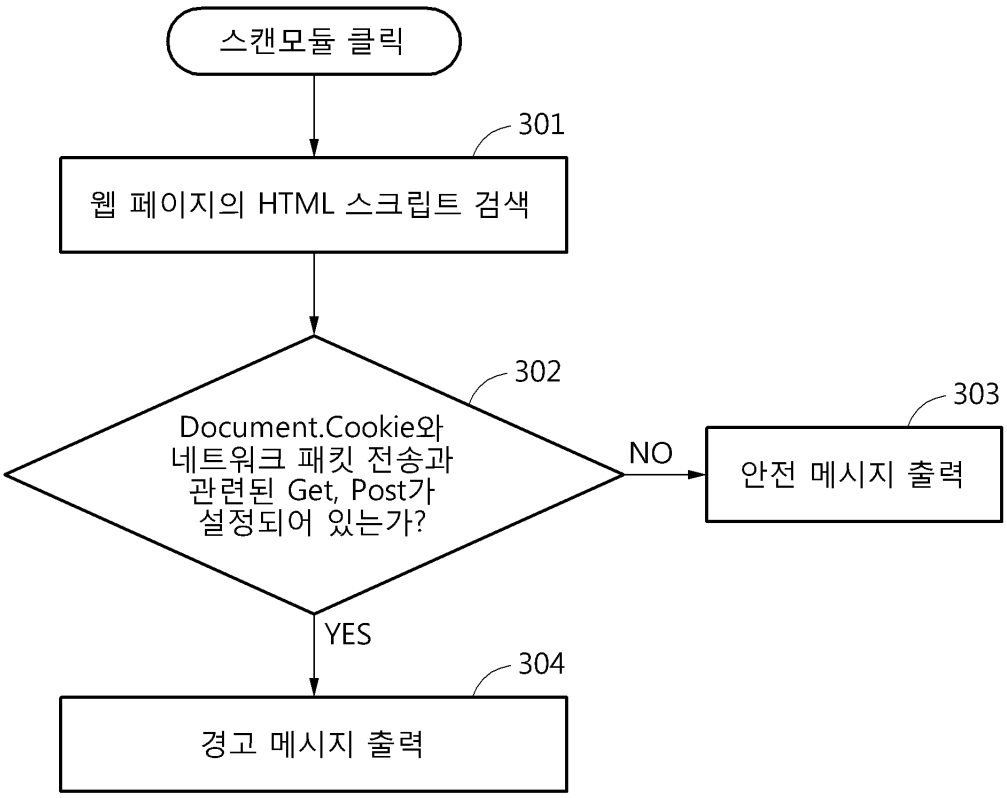
도면2b



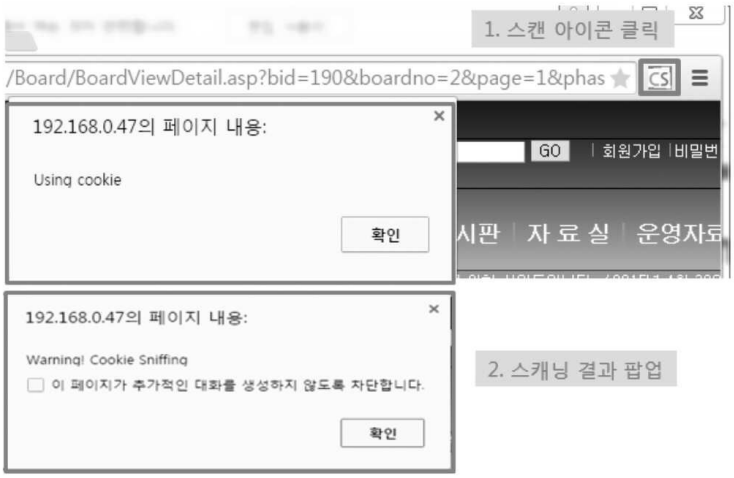
도면2c



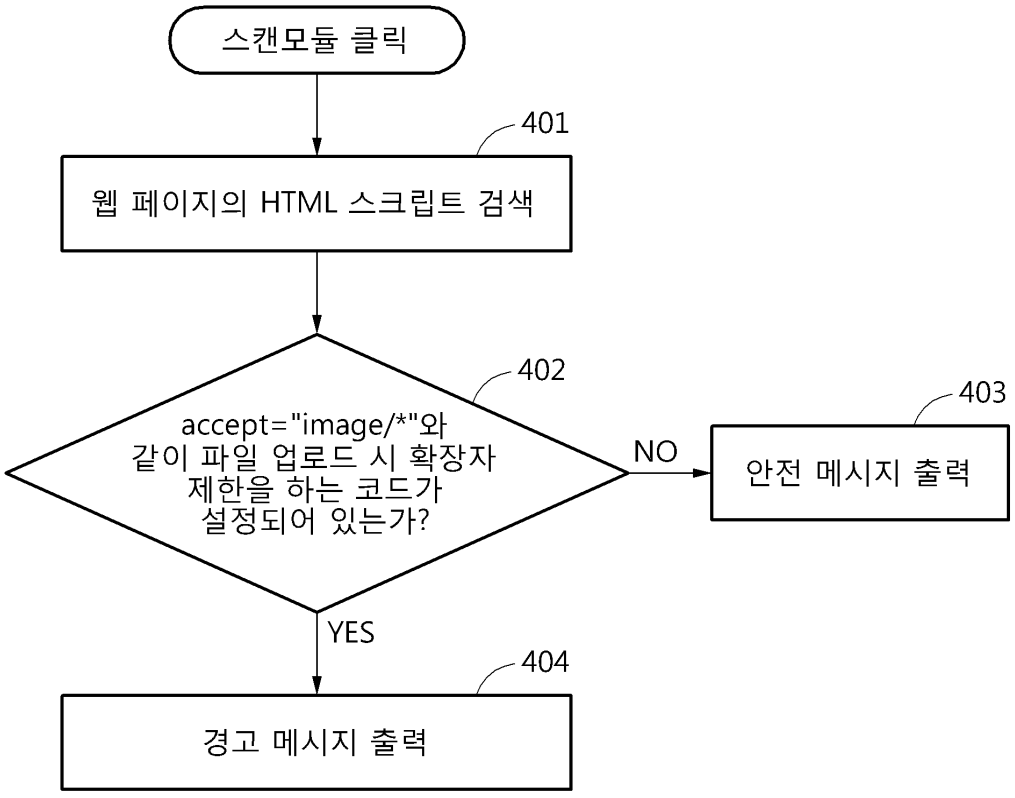
도면3a



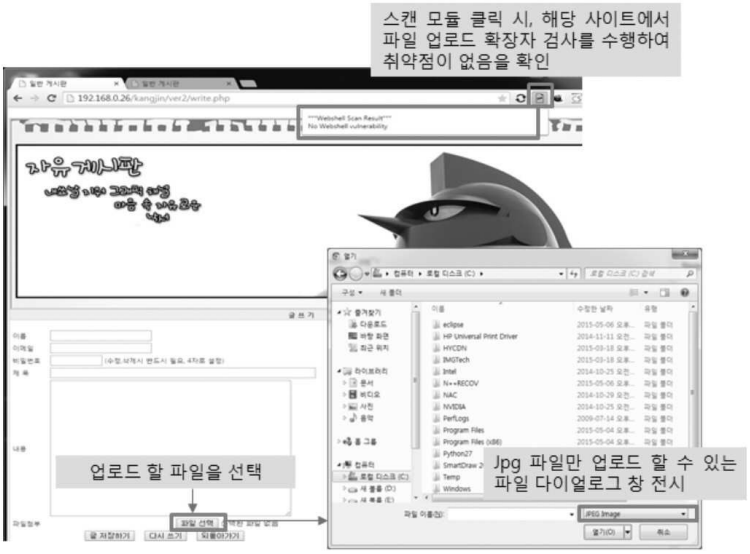
도면3b


1. 스캔 아이콘 클릭
/Board/BoardViewDetail.asp?bid=190&boardno=2&page=1&phas
192.168.0.47의 페이지 내용:
Using cookie
확인
GO
회원가입
비밀번
시판
자 료 실
운영자료
192.168.0.47의 페이지 내용:
Warning! Cookie Sniffing
이 페이지가 추가적인 대화를 생성하지 않도록 차단합니다.
확인
2. 스캐닝 결과 팝업

도면4a


스캔모듈 클릭
401
웹 페이지의 HTML 스크립트 검색
402
accept="image/*"와
같이 파일 업로드 시 확장자
제한을 하는 코드가
설정되어 있는가?
NO
403
안전 메시지 출력
YES
404
경고 메시지 출력

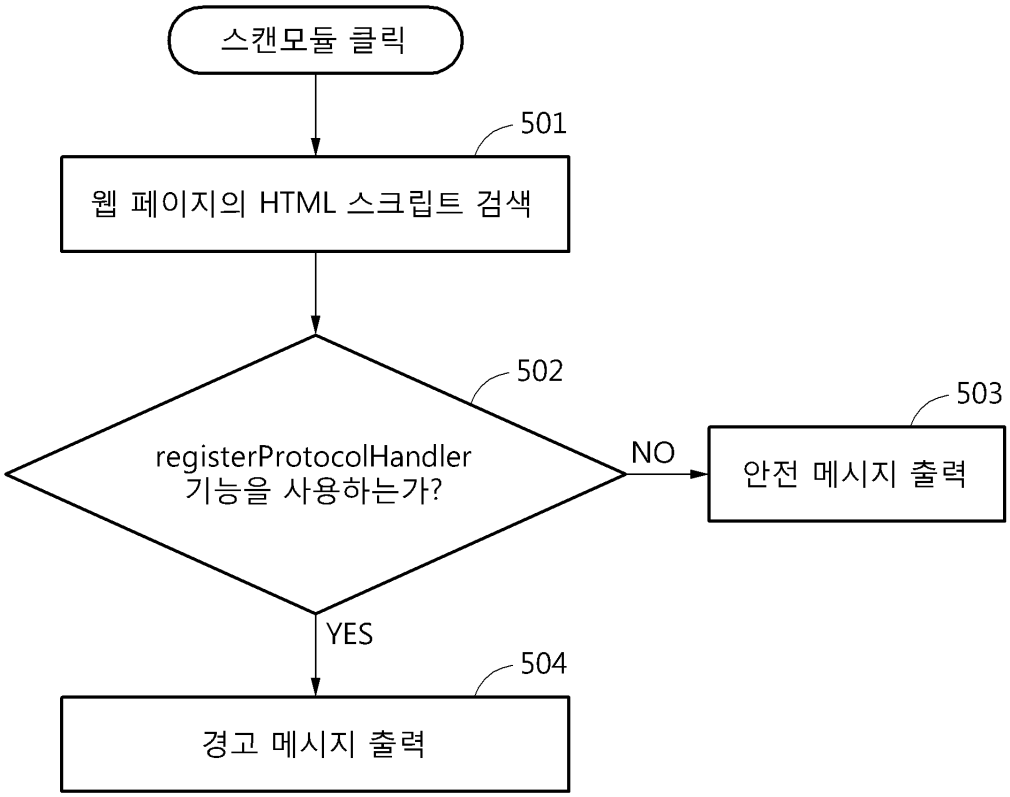
도면4b


스캔 모듈 클릭 시, 해당 사이트에서 파일 업로드 확장자 검사를 수행하여 취약점이 없음을 확인
업로드 할 파일을 선택
Jpg 파일만 업로드 할 수 있는 파일 다이얼로그 창 전시

도면4c


스캔 모듈 클릭 시, 해당 사이트에서 파일 업로드 확장자 검사를 수행하지 않아 취약점이 존재함
HACKER
업로드 할 파일을 선택
확장자 제한 없이 모든 파일을 업로드 할 수 있는 파일 다이얼로그 전시

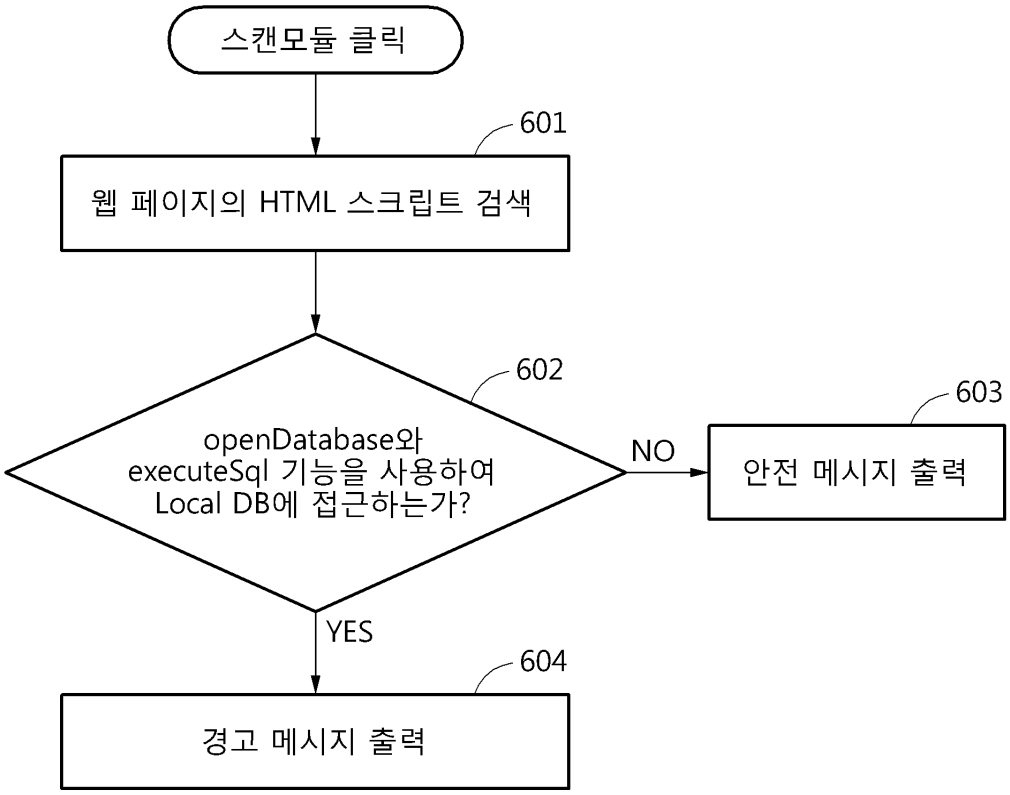
도면5a


스캔모듈 클릭
501
웹 페이지의 HTML 스크립트 검색
502
registerProtocolHandler
기능을 사용하는가?
NO
503
안전 메시지 출력
YES
504
경고 메시지 출력

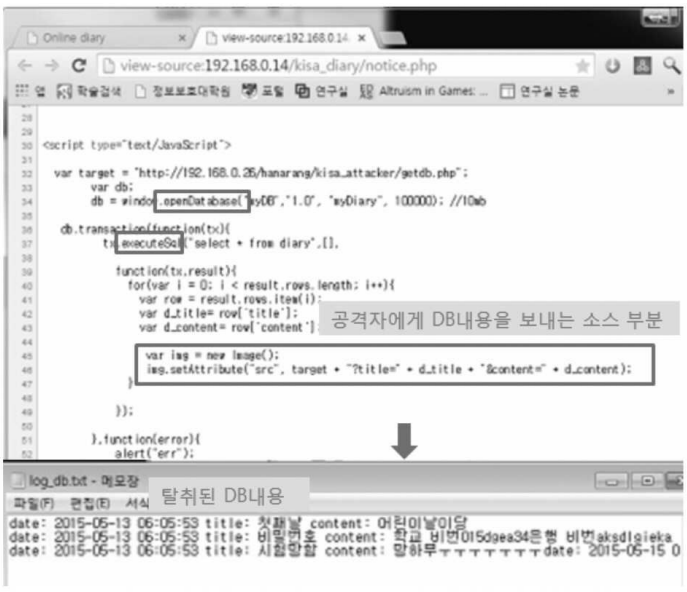
도면5b


CNBLUE
192.168.0.14/kisa_nav/jyh_m.html
1. 스캔 아이콘 클릭
This Browser is protocol register method use
Scheme : 'chem'
2. 스캐닝 결과 팝업
. CNBLUE
팬 여러분의 목소리를 듣습니다.
3월 22일, #정용화 #?OneFineDay in 상하이
자리를 빛내주신 팬 여러분, 감사합니다 smile 이모티콘

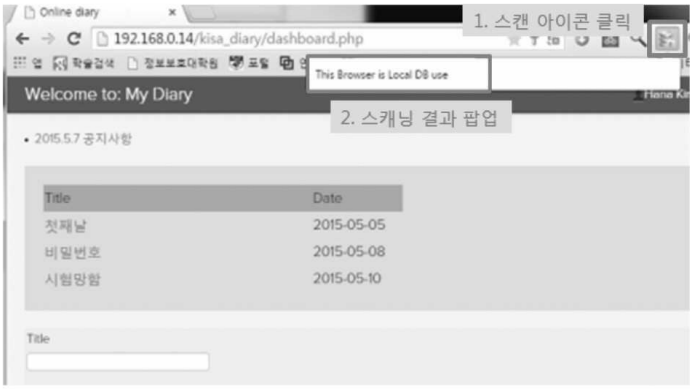
도면6a


스캔모듈 클릭
601
웹 페이지의 HTML 스크립트 검색
602
openDatabase와
executeSql 기능을 사용하여
Local DB에 접근하는가?
NO
603
안전 메시지 출력
YES
604
경고 메시지 출력

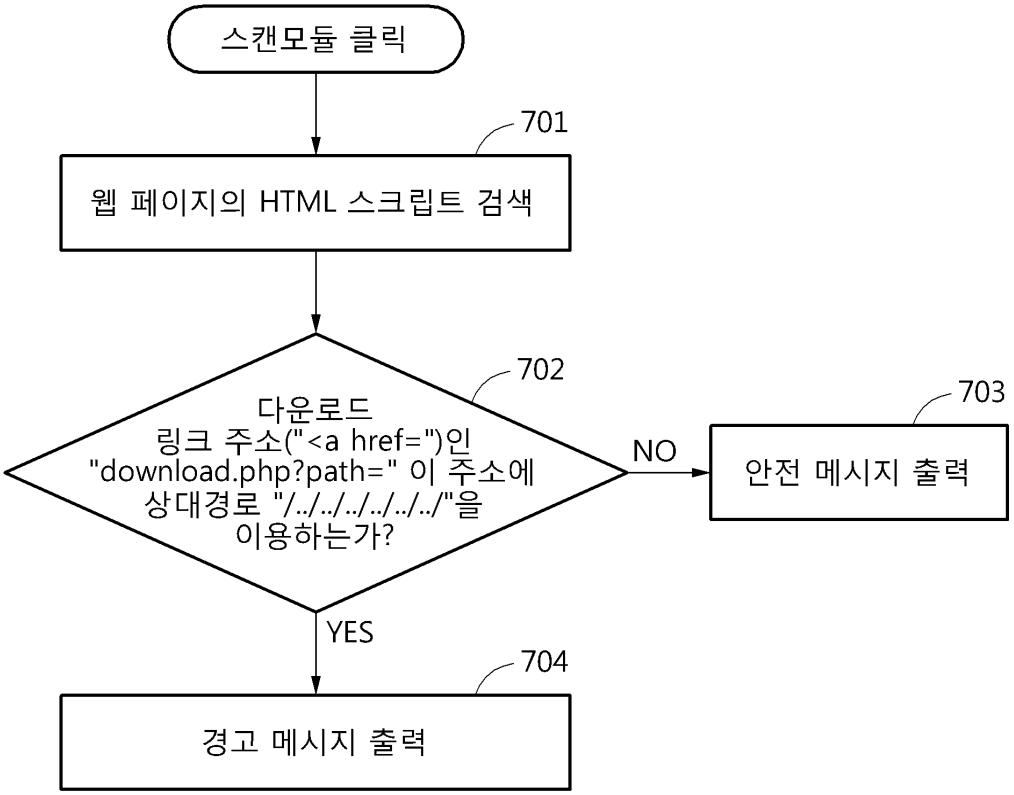
도면6b


view-source:192.168.0.14/kisa_diary/notice.php
공격자에게 DB내용을 보내는 소스 부분
탈취된 DB내용

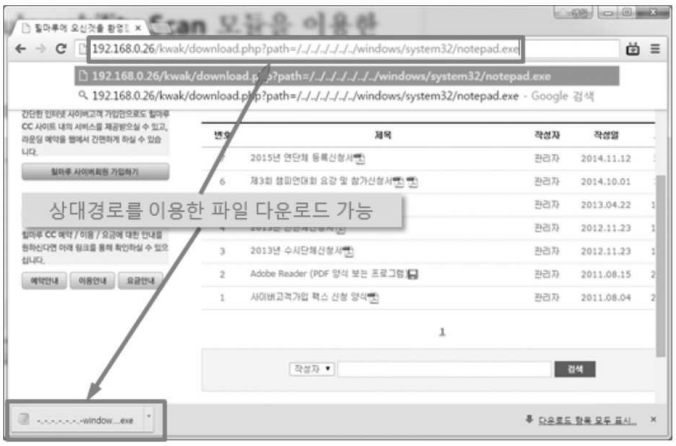
도면6c


Online diary
192.168.0.14/kisa_diary/dashboard.php
1. 스캔 아이콘 클릭
This Browser is Local DB use
Welcome to: My Diary
2. 스캐닝 결과 팝업
• 2015.5.7 공지사항
Title
Date
첫째날
2015-05-05
비밀번호
2015-05-08
시험망함
2015-05-10
Title

도면7a


스캔모듈 클릭
701
웹 페이지의 HTML 스크립트 검색
702
다운로드
링크 주소("<a href=")인
"download.php?path=" 이 주소에
상대경로 "/../../../../../../"을
이용하는가?
NO
703
안전 메시지 출력
YES
704
경고 메시지 출력

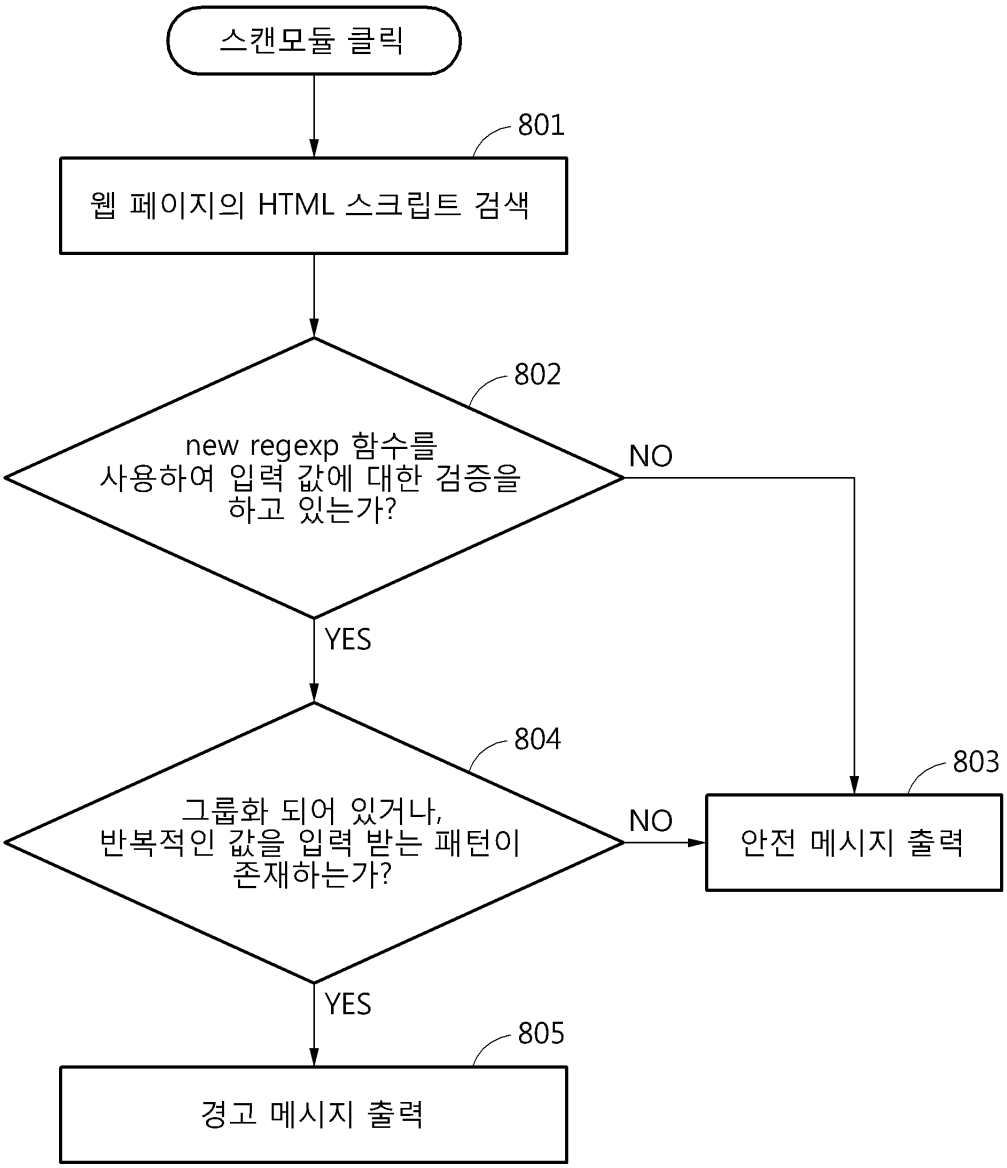
도면7b


192.168.0.26/kwak/download.php?path=/../../../../../windows/system32/notepad.exe
192.168.0.26/kwak/download.php?path=/../../../../../windows/system32/notepad.exe - Google 검색
상대경로를 이용한 파일 다운로드 가능

도면7c


파일 다운로드 링크 주소를 파악하여 File download 취약점을 수행 가능
192.168.0.26/kwak/download.php?path=abc.pdf

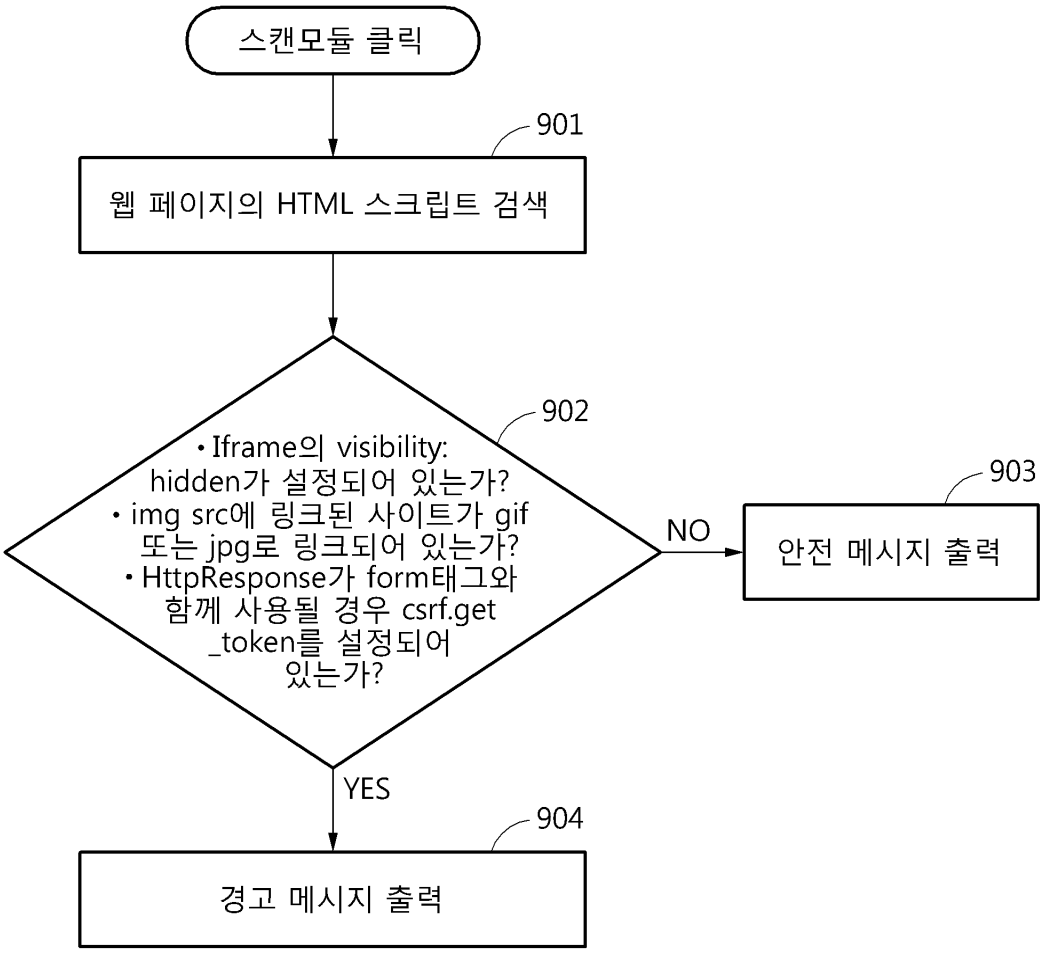
도면8a


스캔모듈 클릭
801
웹 페이지의 HTML 스크립트 검색
802
new regexp 함수를
사용하여 입력 값에 대한 검증을
하고 있는가?
NO
YES
804
그룹화 되어 있거나,
반복적인 값을 입력 받는 패턴이
존재하는가?
NO
803
안전 메시지 출력
YES
805
경고 메시지 출력

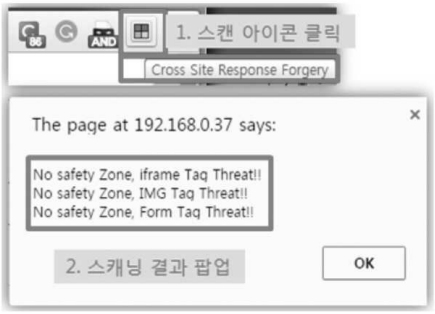
도면8b


1. 스캔 아이콘 클릭
192.168.0.26/kangjin/ver2/write.php
ReDos Scan Result
Repeated regexp is not found
No ReDos vulnerability
2. 스캐닝 결과 팝업
자유게시판

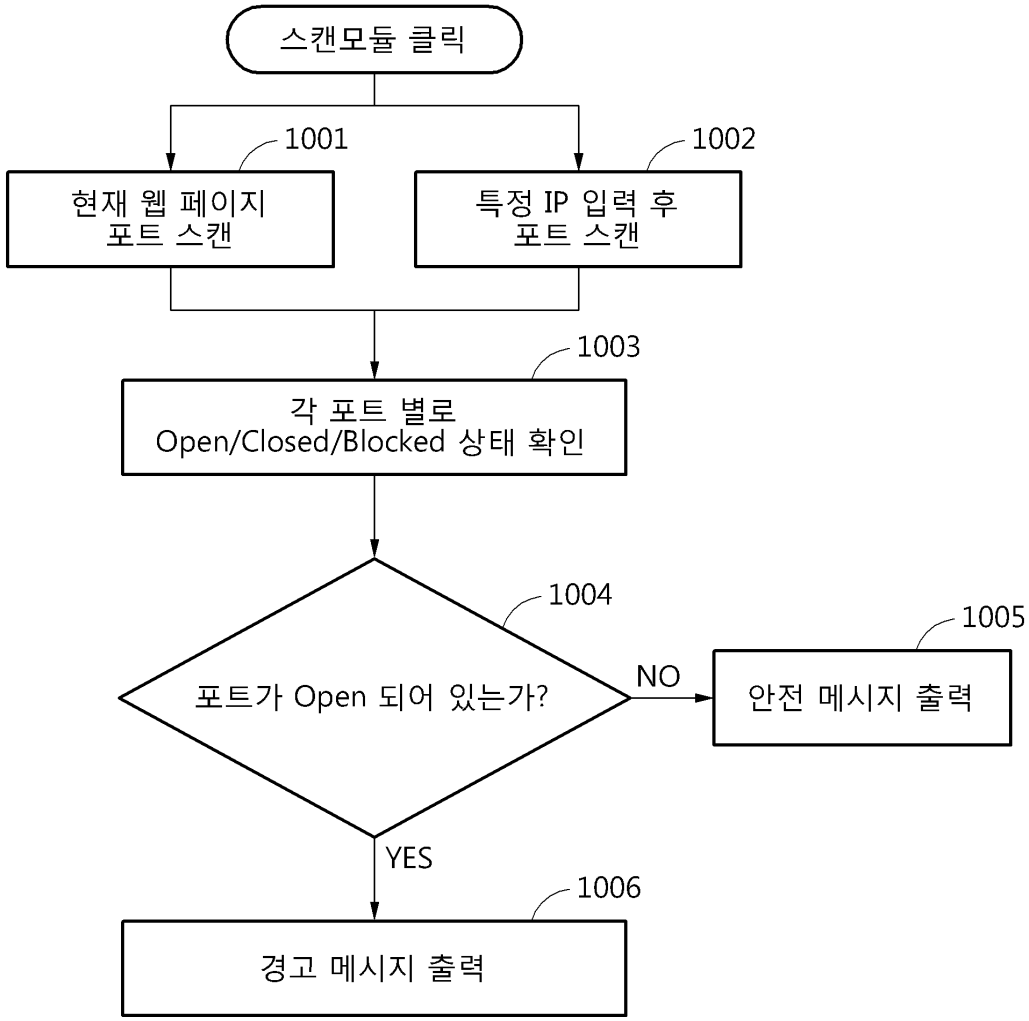
도면9a


스캔모듈 클릭
901
웹 페이지의 HTML 스크립트 검색
902
• Iframe의 visibility: hidden가 설정되어 있는가?
• img src에 링크된 사이트가 gif 또는 jpg로 링크되어 있는가?
• HttpResponse가 form태그와 함께 사용될 경우 csrf.get _token를 설정되어 있는가?
NO
903
안전 메시지 출력
YES
904
경고 메시지 출력

도면9b


1. 스캔 아이콘 클릭
Cross Site Response Forgery
The page at 192.168.0.37 says:
No safety Zone, iframe Tag Threat!!
No safety Zone, IMG Tag Threat!!
No safety Zone, Form Tag Threat!!
2. 스캐닝 결과 팝업
OK

도면10a


스캔모듈 클릭
1001
현재 웹 페이지
포트 스캔
1002
특정 IP 입력 후
포트 스캔
1003
각 포트 별로
Open/Closed/Blocked 상태 확인
1004
포트가 Open 되어 있는가?
NO
1005
안전 메시지 출력
YES
1006
경고 메시지 출력

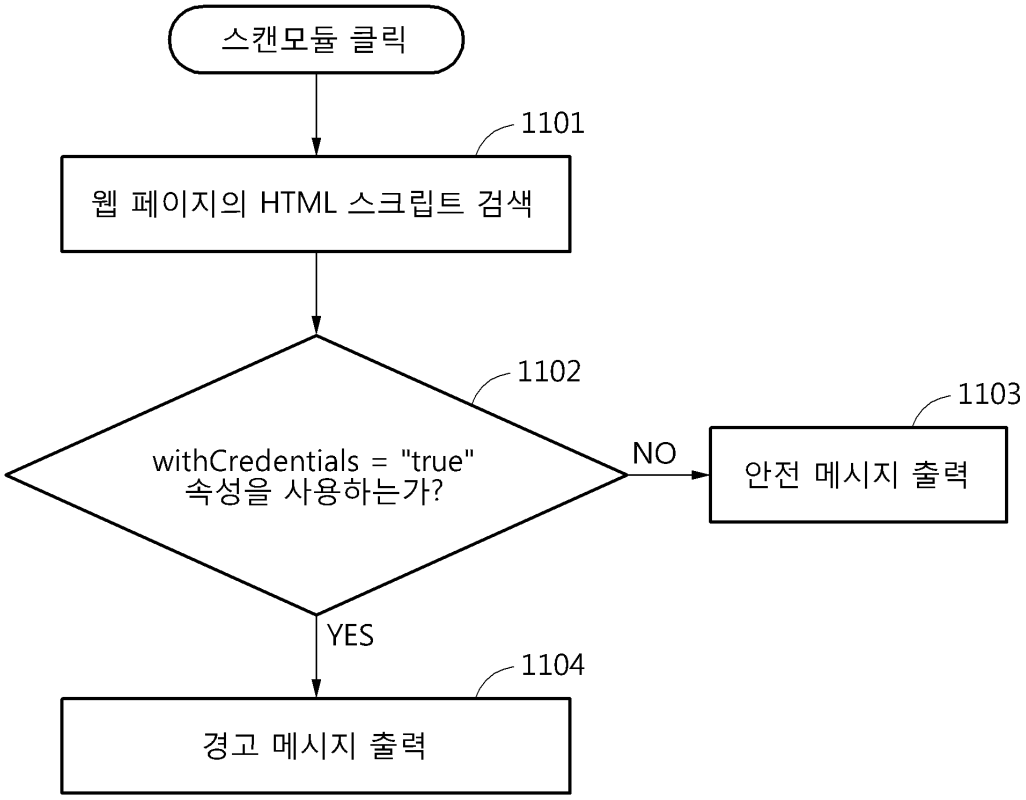
도면10b


1. 스캐너 아이콘 클릭
192.168.0.26/kangjin/daum.html
Web Socket
Vulnerability Scanner
IP Address:
Entered IP Scan
Current Page Scan
2. 현재 웹 페이지 스캔 버튼 클릭
Web Socket Vulnerability Scanner
- Vulnerability Report -
IP:192.168.0.26
Open Ports:80
Closed/Blocked Ports:69
Filtered/Application Type 3&4 Ports:
137,161,445
Scan Log:
69(TFTP) - closed
80(HTTP) - open
137(UDP) - time exceeded
143(IMAP) - blocked port
161(SNMP) - time exceeded
445(TCP/UDP) - time exceeded
514(UDP) - blocked port
587(SMTP) - blocked port
993(TCP) - blocked port
Result : Vulnerability exists
3. 80번 오픈 포트 발견으로 웹 소켓을 이용한 포트스캔 취약점 존재함

도면10c


1. 스캐너 아이콘 클릭
Web Socket Vulnerability Scanner
192.168.0.26/kangjin/daum.html
Web Socket Vulnerability Scanner
IP Address:
168.152.127.56
Entered IP Scan
2. IP 입력 후 스캔버튼 클릭
- Vulnerability Report -
IP:168.152.127.56
Open Ports:
Closed/Blocked Ports:
Filtered/Application Type 3&4 Ports:
69,80,137,161,445
Scan Log:
69(TFTP) - time exceeded
80(HTTP) - time exceeded
137(UDP) - time exceeded
143(IMAP) - blocked port
161(SNMP) - time exceeded
445(TCP/UDP) - time exceeded
514(UDP) - blocked port
587(SMTP) - blocked port
993(TCP) - blocked port
Result : None
3. Open 포트 발견으로 취약점 존재

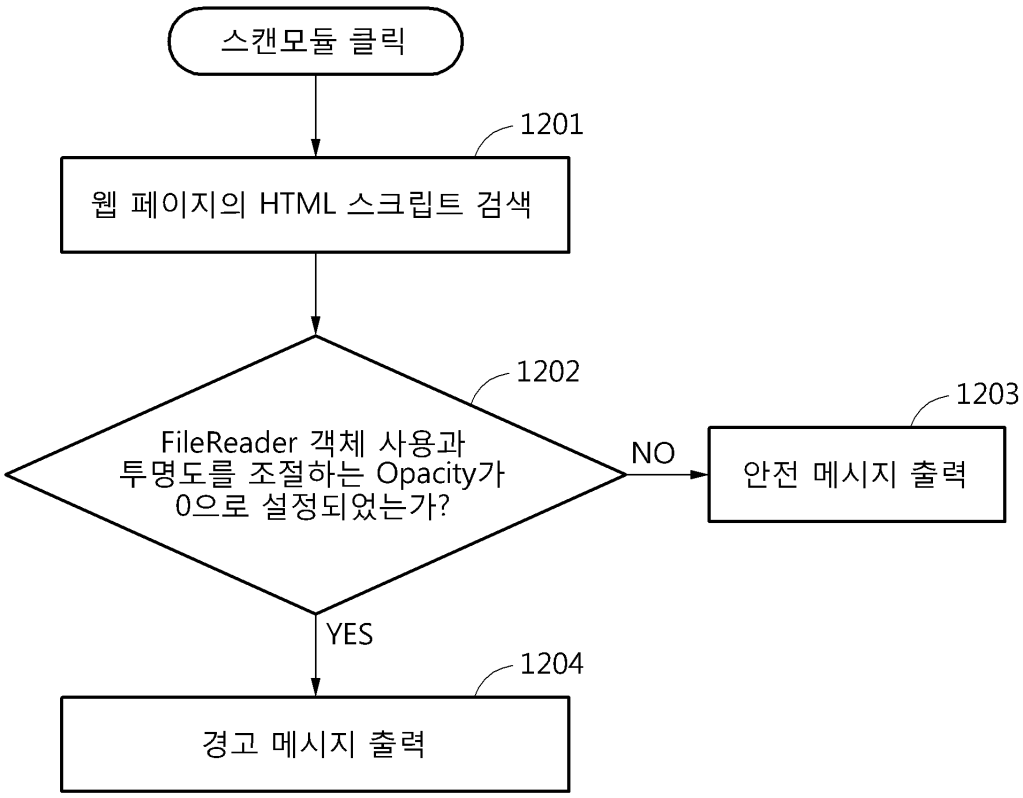
도면11a


스캔모듈 클릭
1101
웹 페이지의 HTML 스크립트 검색
1102
withCredentials = "true"
속성을 사용하는가?
NO
1103
안전 메시지 출력
YES
1104
경고 메시지 출력

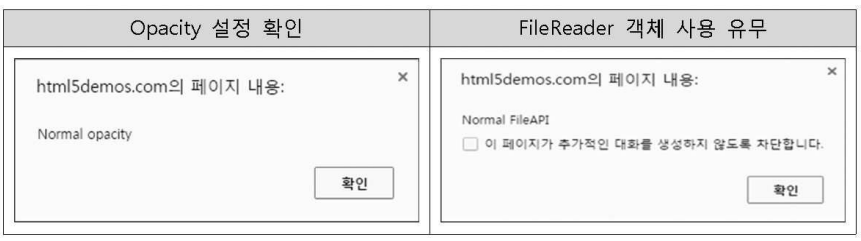
도면11b



도면12a



도면12b

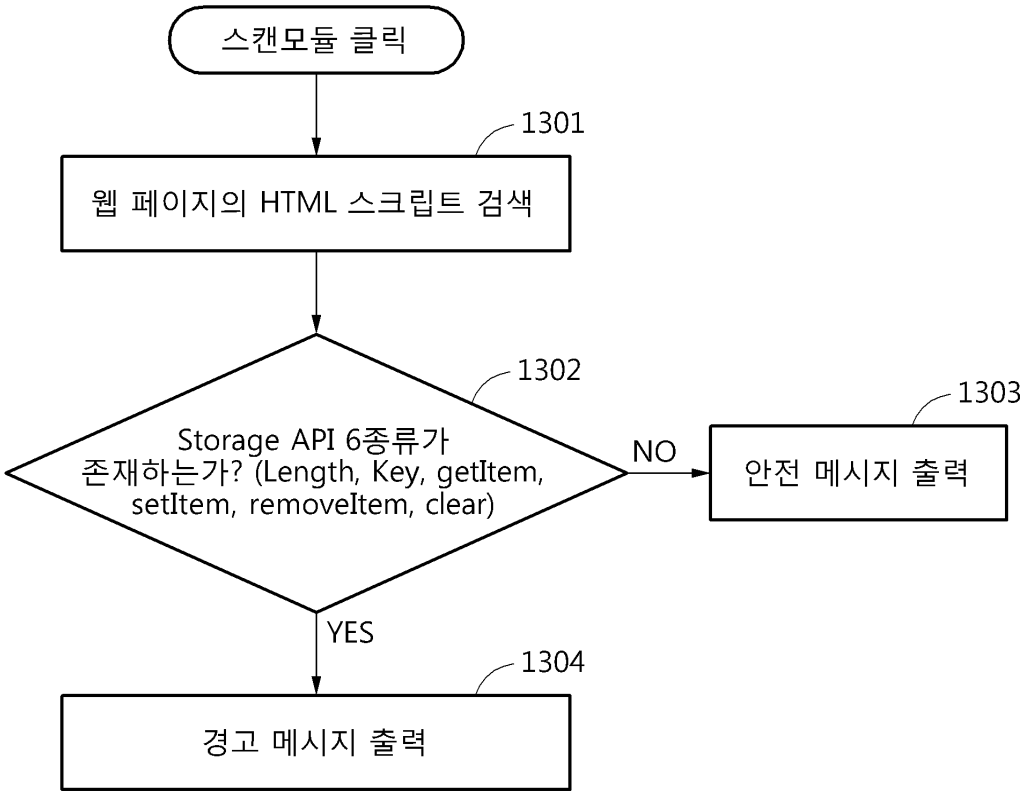


도면12c

Opacity 설정 확인	FileReader 객체 사용 유무

<body>
</body>
<div style="position:absolute;left:750p
<input type="button" style="opacity:0;
 	
	var file = document.getElementById("file").files[0];
	var reader = new FileReader();
	reader.onload = function() {
	xhr = new XMLHttpRequest();
	xhr.open("post","http://192.168.0.26/jaehyuk/TestSit
	xhr.setRequestHeader("Content-Type","application/x-w

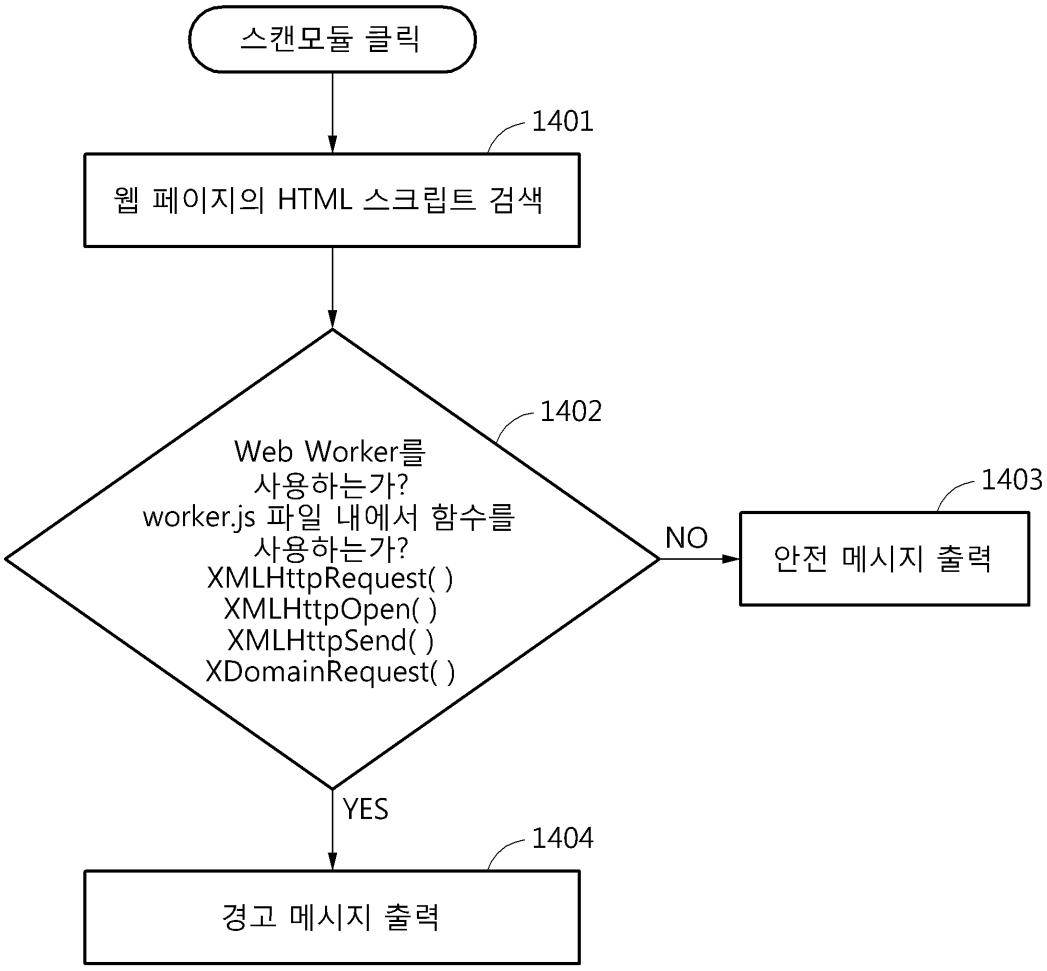
도면13a



도면13b



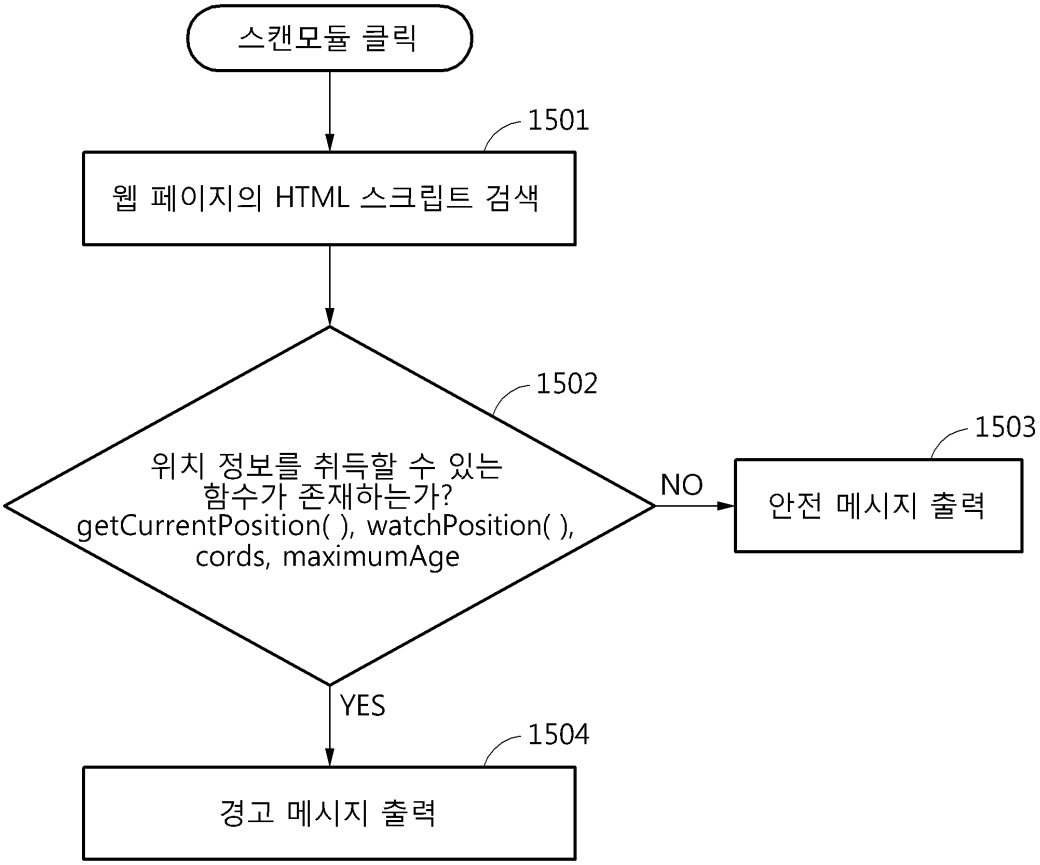
도면14a


스캔모듈 클릭
1401
웹 페이지의 HTML 스크립트 검색
1402
Web Worker를
사용하는가?
worker.js 파일 내에서 함수를
사용하는가?
XMLHttpRequest( )
XMLHttpOpen( )
XMLHttpSend( )
XDomainRequest( )
NO
1403
안전 메시지 출력
YES
1404
경고 메시지 출력

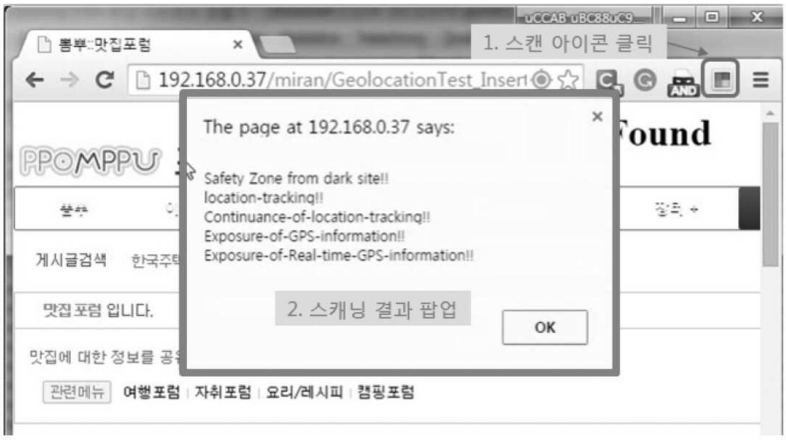
도면14b


XMLHttpRequest와 관련된 함수 사용
XMLHttpRequest와 관련된 함수 미사용
자바스크립트 알림
web worker() 사용함
XDomainRequest() 사용함
open() 사용함
send() 사용함
XMLHttpRequest() 사용함
확인
https://www.qoogle.co.kr의 페이지 내용:
web worker() 사용하지 않음
send() 사용하지 않음
open() 사용하지 않음
XMLHttpRequest() 사용하지 않음
XDomainRequest() 사용하지 않음
확인

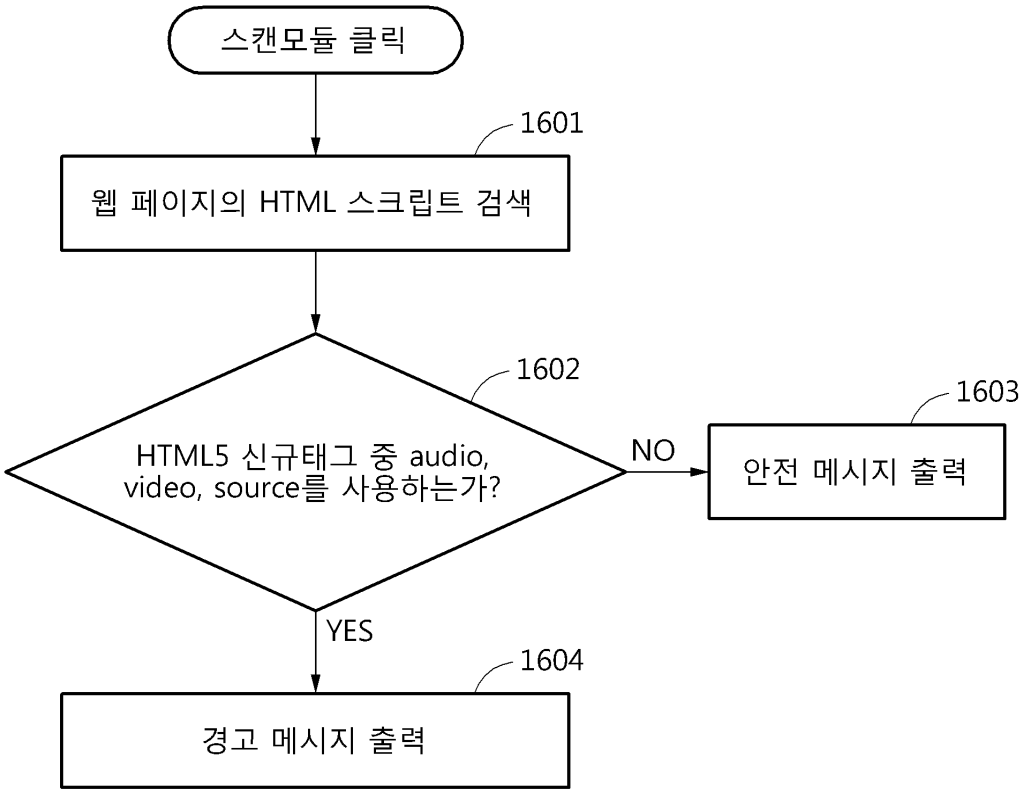
도면15a


스캔모듈 클릭
1501
웹 페이지의 HTML 스크립트 검색
1502
위치 정보를 취득할 수 있는
함수가 존재하는가?
getCurrentPosition( ), watchPosition( ),
cords, maximumAge
NO
1503
안전 메시지 출력
YES
1504
경고 메시지 출력

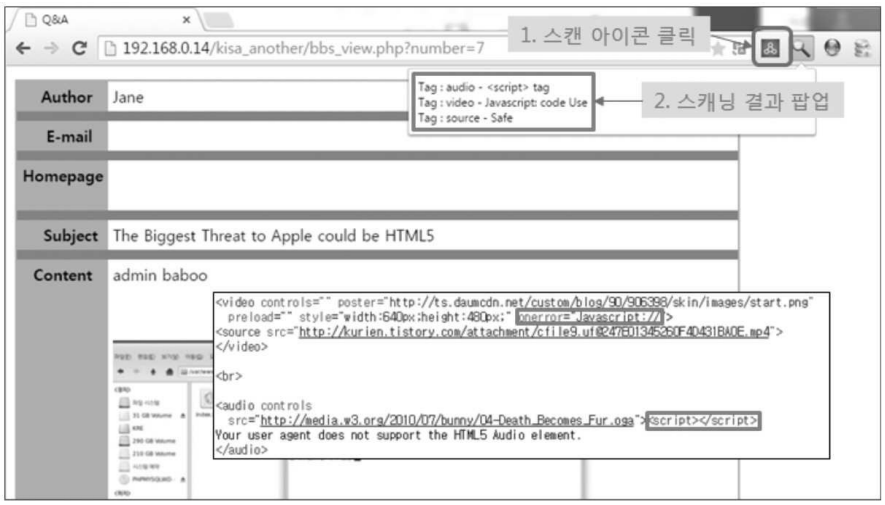
도면15b


1. 스캔 아이콘 클릭
192.168.0.37/miran/GeolocationTest_Insert
The page at 192.168.0.37 says:
Safety Zone from dark site!!
location-tracking!!
Continuance-of-location-tracking!!
Exposure-of-GPS-information!!
Exposure-of-Real-time-GPS-information!!
2. 스캐닝 결과 팝업
OK
맛집 포럼 입니다.
여행포럼 | 자취포럼 | 요리/레시피 | 캠핑포럼

도면16a

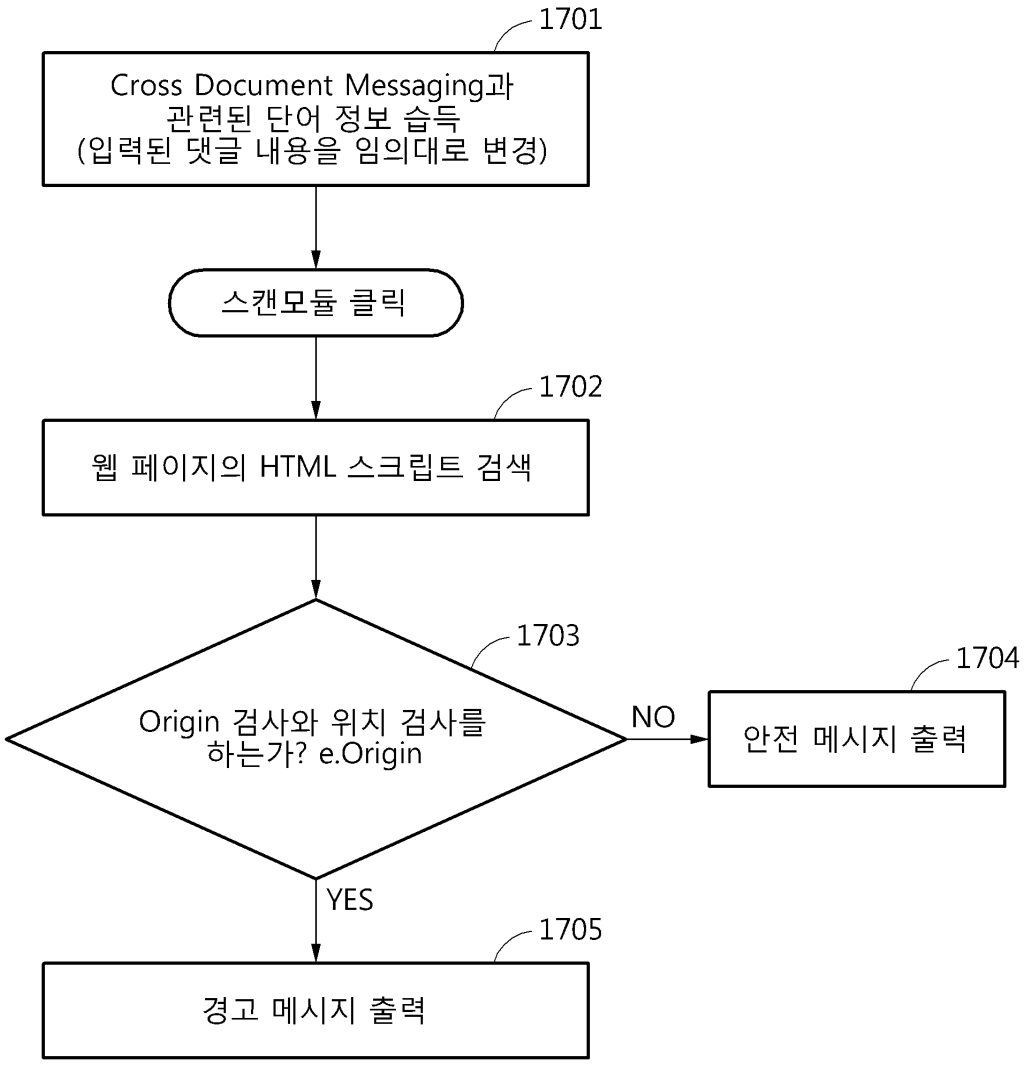

스캔모듈 클릭
1601
웹 페이지의 HTML 스크립트 검색
1602
HTML5 신규태그 중 audio, video, source를 사용하는가?
NO
1603
안전 메시지 출력
YES
1604
경고 메시지 출력

도면16b


Q&A
192.168.0.14/kisa_another/bbs_view.php?number=7
1. 스캔 아이콘 클릭
Tag : audio - <script> tag
Tag : video - Javascript: code Use
Tag : source - Safe
2. 스캐닝 결과 팝업
Author
Jane
E-mail
Homepage
Subject
The Biggest Threat to Apple could be HTML5
Content
admin baboo
<video controls="" poster="http://ts.daumcdn.net/custom/blog/90/906398/skin/images/start.png"
preload="" style="width:640px;height:480px;" onerror="Javascript://">
<source src="http://kurien.tistory.com/attachment/cfile9.uf@247E013452560F4043IBA0E.mp4">
</video>

<audio controls
src="http://media.w3.org/2010/07/bunny/04-Death_Becomes_Fur.oga"><script></script>
Your user agent does not support the HTML5 Audio element.
</audio>

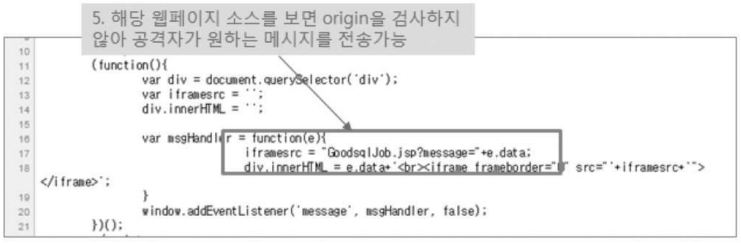
도면17a


1701
Cross Document Messaging과
관련된 단어 정보 습득
(입력된 댓글 내용을 임의대로 변경)
스캔모듈 클릭
1702
웹 페이지의 HTML 스크립트 검색
1703
Origin 검사와 위치 검사를
하는가? e.Origin
NO
1704
안전 메시지 출력
YES
1705
경고 메시지 출력

도면17b


3. 스캐너 아이콘 클릭
192.168.0.26:8081/KISA/postmessageAttacker.html
Cross Document Messaging vulnerability exists.
Please check the origin codes
. Attacker Modified
팬 여러분의 목소리를 듣습니다.
4. 스캐너 결과 취약점이
존재함을 알 수 있음
3월 22일, #정용화 #OneFineDay in 상하이
자리를 빛내주신 팬 여러분, 감사합니다 smile 이모티콘
#JungYongHwa 'One Fine Day' in Shanghai on Mar 22th
Dear Fans, Thanks for attending the concert
1. 댓글에 좋아요 버튼 클릭
댓글달기 good~
제출
댓글내용 :
your webpage is hacked!!!!!!
2. 입력 값(good)은 변경할 수 없지만
공격자가 원하는 값을 입력함

도면17c


5. 해당 웹페이지 소스를 보면 origin을 검사하지 않아 공격자가 원하는 메시지를 전송가능
(function(){
var div = document.querySelector('div');
var iframesrc = '';
div.innerHTML = '';
var msgHandler = function(e){
iframesrc = "GoodsqlJob.jsp?message="+e.data;
div.innerHTML = e.data+'
<iframe frameborder="0" src="'+iframesrc+'">
</iframe>';
}
window.addEventListener('message', msgHandler, false);
})();