http://dx.doi.org/10.3745/KIPSTC.2012.19C.1.047

원격코드검증을 통한 웹컨텐츠의 악성스크립트 탐지

최 재 영[†] · 김 성 기[††] · 이 혁 준[†††] · 민 병 준[††††]

요 약

최근 웹사이트는 매쉬업, 소셜 서비스 등으로 다양한 출처의 리소스를 상호 참조하는 형태로 변화하면서 해킹 시도도 사이트를 직접 공격하기보다 서비스 주체와 연계 서비스, 클라이언트가 상호 작용하는 접점에 악성스크립트를 삽입하는 공격이 증가하고 있다. 본 논문에서는 웹사이트 이용 시 신뢰관계에 있는 여러 출처로부터 다운받은 웹컨텐츠의 HTML 코드와 자바스크립트 코드가 클라이언트 브라우저에서 구동 시 삽입된 악성스크립트를 원격의 검증시스템으로부터 탐지하는 모델을 제안한다. 서비스 주체의 구현코드 정보를 활용하여 요청 출처에 따라 검증 항목을 분류하고 웹컨텐츠의 검증 요소를 추출하여 검증 평가결과를 화이트, 그레이, 블랙 리스트로 데이터베이스에 저장하였다. 실험평가를 통해 제안한 시스템이 악성스크립트를 효율적으로 탐지하여 클라이언트의 보안이 향상됨을 확인하였다.

키워드 : 악성스크립트 탐지, 원격 코드 검증, 화이트 리스트, 블랙 리스트

Detecting Malicious Scripts in Web Contents through Remote Code Verification

Jae-Yeong Choi[†] · Sung-Ki Kim[††] · Hyuk-Jun Lee[†††] · Byoung-Joon Min[††††]

ABSTRACT

Sharing cross-site resources has been adopted by many recent websites in the forms of service-mashup and social network services. In this change, exploitation of the new vulnerabilities increases, which includes inserting malicious codes into the interaction points between clients and services instead of attacking the websites directly. In this paper, we present a system model to identify malicious script codes in the web contents by means of a remote verification while the web contents downloaded from multiple trusted origins are executed in a client's browser space. Our system classifies verification items according to the origin of request based on the information on the service code implementation and stores the verification results into three databases composed of white, gray, and black lists. Through the experimental evaluations, we have confirmed that our system provides clients with increased security by effectively detecting malicious scripts in the mashup web environment.

Keywords : Detection of Malicious Scripts, Remote Code Verification, White List, Black List

1. 서 론

최근 웹은 사용자 경험의 개선, 서비스간 상호 작용과 협업이 가능한 매쉬업 서비스, 인간간 상호 작용이 가능한 소셜 서비스 등 웹 트랜드가 변화하면서 다양한 출처의 리소스를 상호 참조하는 형태로 변화되고 있다. 사이트간 연계를 통한 서비스 확장을 위해 서비스 상호간 신뢰(trust) 관계 구축이 증가하고 인간간 신뢰 관계를 이용하여 추천 등의 방식으로 외부링크로 접속 유도가 증가하고 있다. 이에 따라 단일 사이트에 초점을 둔 기존 웹보안 방식에 한계가 있다.

사이트간 연계로 사용자의 브라우저에 다양한 소스로부터 가져온 웹컨텐츠가 뒤섞이어 한 화면을 구성하게 된다. 웹컨텐츠간 안전을 보장하기 위해 브라우저는 기본 보안 모델로 동일출처정책(SOP, Same Origin Policy)[1]을 수립하여 동일출처가 아닌 컨텐츠간 접근을 방지한다. 그러나 단순히 출처에 대한 신뢰검증만으로는 보안을 확신할 수 없다. 올바른 출처에서 온 코드가 안전하다고 보장할 수 없기 때문이다.

사용자 경험 개선과 매쉬업 API, 전송 데이터 가공 등을 위해 자바스크립트 기반의 서드 파티 라이브러리를 이용 시, 웹페이지에 외부 스크립트로 삽입된 자바스크립트는 삽입된

※ 본 연구는 인천대학교 자체연구비 지원에 의한 것임.
† 준 회 원 : 인천대학교 컴퓨터공학과 박사과정
†† 정 회 원 : 선문대학교 IT교육학부 전임강사
††† 준 회 원 : 인천대학교 컴퓨터공학과 석사과정
†††† 종신회원 : 인천대학교 컴퓨터공학과 교수(교신저자)
논문접수 : 2011년 10월 17일
수 정 일 : 1차 2011년 12월 1일
심사완료 : 2011년 12월 7일

페이지와 동일출처 권한을 갖게 된다. 악성스크립트가 삽입된 외부 스크립트 참조 시, 스크립트가 삽입된 페이지와 동일한 SOP 권한을 갖게 되어 브라우저 내의 세션, 쿠키 정보와 동일출처 서버의 웹 어플리케이션은 무방비 상태가 된다. 웹페이지에 iframe이 있는 경우 iframe에 호출된 페이지에 대해 동일출처가 아니면 접근 권한이 없지만 크로스 도메인 요청이 가능하여 CSRF 공격에 이용된다.

동일출처정책이 불완전하지만 안전하고 유용한 보안 정책이다. 그런데 동일출처정책이 매쉬업 서비스에 장애 요소로 작용하기 때문에 일부 타도메인을 동일출처로 설정하여 신뢰관계를 확장하여 CORS[2], postMessage API[3], Cross-domain policy file specification [4] 등의 새로운 방안이 이용되고 있다. 신뢰관계 구축이 어려운 경우는 동일출처 내 서버를 프록시로 이용하여 접근이 가능하도록 우회하기도 한다[5]. 그러나 이는 서로간에 안전하다는 전제에서 연계 서비스간 권한을 상승시키는 것으로 그에 따른 위험도 증가하게 된다. 이에 따라 신뢰관계를 부여한 도메인에 대한 지속적인 감시와 위험 발생 시 신속한 차단 방법이 필요하다.

본 논문에서는 웹사이트 이용 시 신뢰관계에 있는 여러 출처로부터 다운받은 웹컨텐츠의 HTML 코드와 자바스크립트 코드가 클라이언트 브라우저에서 구동 시 삽입된 악성스크립트를 원격의 검증시스템으로부터 탐지하는 모델을 제안한다. 서비스 주체의 구현코드 정보를 활용하여 요청 출처에 따라 검증 항목을 분류하고 웹컨텐츠의 검증 요소를 추출하여 검증 평가결과를 화이트 리스트, 그레이 리스트, 블랙 리스트로 데이터베이스화하여 사전 검증평가에 따른 검증 효율을 확인한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구를 바탕으로 악성스크립트 대응 기법의 문제점을 분석하고 3장에서는 악성스크립트 검출을 위한 원격코드검증 방안을 제안하고 4장에서 제안한 기법의 평가 및 분석을 기술한다. 5장에서 결론 및 향후 연구방향을 제시한다.

2. 관련 연구

본 장에서는 인터넷 이용자가 웹사이트 접속 시, 삽입된 악성스크립트의 실행이나 피싱 사이트로의 접속 유도로 발생하는 피해를 막기 위한 기존의 다양한 대응 기법과 한계점을 파악하였다.

2.1 블랙리스트 기반 탐지

구글의 안전 브라우징은 방문하려는 사이트가 피싱의 위험이 있거나 악성스크립트가 포함된 것으로 의심되는 사이트인지를 악성 사이트 목록과 비교하여 위험성이 있는 사이트인지 여부를 확인한 후 경고 메시지를 표시한다[6][7]. 모질라의 파이어폭스3, 구글의 크롬4, 사파리4 등이 구글의 안전 브라우징 서비스를 이용하고 있다. MS의 IE8도 스마트 스크린 필터를 이용해 접속한 웹사이트에 대한 보안경고를 표시한다[8]. 브라우저에서 자체 지원되는 피싱사이트나 악성스크립트 유포 웹사이트 접속을 차단하는 기술은 웹사이트 크롤링을 통해 수집한 방대한 웹 페이지 정보를 바탕으로 검증을 통해 탐지한 대상 URL을 블랙리스트로 DB화하고 사용자의 접속 사이트와 블랙리스트를 비교하여 차단 여부를 결정한다. 이러한 블랙리스트를 이용한 대응은 블랙리스트 업데이트 지연에 따른 신규 유해 사이트에 대한 대응이 늦다. 또한 악성스크립트가 검출 시 해당 웹사이트 전체에 경고가 발생하며, 잦은 경고는 브라우징 경험을 저하시키고 이용을 방해하며, 사용자가 오탐지로 생각하여 보안경고를 무시하거나 기능을 제거할 소지가 있다. 또한 악성스크립트 탐지 실패로 블랙리스트에 추가되지 않으면 무용지물이 된다.

2.2 크로스 도메인간 정책 부여

크로스 사이트간 자원 공유 Cross-Origin Resource Sharing(CORS)[2] 는 브라우저의 Same-Origin Policy(SOP)의 제약을 극복하기 위해 만든 W3C 표준으로, 리소스를 제공하는 측에서 어떤 도메인에서의 접근을 허용할 지 'Access-Control-Allow-Origin' 헤더에 명시하여 클라이언트 브라우저에서 명시한 도메인간에 SOP와 같은 권한을 부여하는 정책이다. 이는 SOP의 권한을 느슨하게 하여 위험요소가 증가하게 된다. 연계 서비스 도메인에 내포되는 악성스크립트의 권한도 상승하게 된다.

[9]에서는 클라이언트에 정책 엔진을 두고 서버에서 제공하는 정책에 따라 크로스 도메인 요청을 허용, 거부, 수정하는 별도의 정책 모델을 제시했다. 그러나 이 연구방안은 서버와 클라이언트가 보안상 안전하다는 전제에서 서버에서 제공하는 모든 서비스에 대한 정책을 수립해야지 효과적으로 이용이 가능하다.

Subspace[10], Smash[11]는 매쉬업 웹환경에서 크로스 도메인간 안전한 통신 방안을 제시하였다. 그러나 이 방안을 적용하기 위해서는 브라우저와 기존 서비스, 프로토콜의 수정이 필요하다. 아울러 통신 방식이 안전하다고 해서 연계 서비스 자체가 안전하다는 것을 보장하지 못한다.

2.3 요청, 응답 조작

BrowserShield[12]는 HTML과 자바스크립트의 코드를 재작성하여 실행 시 웹컨텐츠의 변경이나 알려진 취약점 요소를 검사한다. 코드 변경에 따른 유연성 문제가 발생할 수 있다. 또한 스크립트 기반의 검사로 검사 요청이 브라우저에서 스크립트가 동작하여 이루어지기 때문에 스크립트 조작으로 검사를 우회할 수 있다.

SpyProxy[13]는 가상머신 기반의 웹 프록시로 서버와 클라이언트간 주고 받는 웹트래픽을 수집하여 행위 기반의 검출을 수행한다. 서버측 대응 방안은 서버로 들어오는 요청과 응답에 대해서만 검사하여 서드 파티 리소스에 의해 발생하는 요청이나, 외부 서버로 전달되는 요청을 감지하지 못한다. 또한 외부 페이지를 클라이언트 브라우저가 다운받아 실행하는 경우 탐지가 어렵다. 발생하는 요청, 응답 트래픽을 분석하여 민감한 정보의 유출을 차단하는 방안은 연계

서비스간 유연성을 침해할 수 있으며 유출 정보를 가공할 경우 차단이 어렵다.

3. 제안모델

본 장에서는 웹사이트 이용 시 신뢰관계에 있는 여러 출처로부터 다운받은 웹컨텐츠의 HTML 코드와 자바스크립트 코드가 클라이언트 브라우저에서 구동 시 삽입된 악성스크립트를 원격의 검증서버로부터 탐지하는 모델을 제안한다.

3.1 제안 모델의 요구사항

2장에서 논의한 한계를 해결하고 보다 안전한 인터넷 환경을 제공하기 위해 제안모델의 요구사항은 다음과 같다.

(1) 서비스에 이용되는 전체 서비스 범위에 대한 검증

최근의 웹사이트는 단일 서버에서 전체 서비스를 제공하는 형태의 폐쇄형에서 매쉬업과 사용자 참여 등 웹2.0으로 대표되는 구현기술과 서비스특성의 변화에 따라 다양한 출처의 리소스를 이용하여 서비스를 구성하는 오픈형, 연계형으로 변화되고 있다. 이에 따라 검증의 범위를 서비스 주체에 국한 시키기보다 연계 서비스까지 확장하여야 한다.

(2) 웹브라우저 관점에서의 검증

공격 시도가 과거에는 서버의 취약점이나 서버 어플리케이션의 논리적 결함 등에 집중 되었으나 보안기술의 발전과 관리강화에 따라 상대적으로 취약한 클라이언트에 대한 공격이 증가하고 있다. 이에 따라 브라우저에 다운로드 되어 실행되는 코드와, 코드 실행을 통해 발생하는 HTTP 요청에 대한 검증이 필요하다.

(3) 서버 측 정보를 최대한 활용

CSRF[14] 공격은 피해자의 인지 없이 피해자의 권한으로 정상적인 요청을 생성하는 공격으로 정상적인 트래픽과 구분이 어렵다. 세션, 쿠키 등의 인증 정보나 민감한 개인정보의 유출이 의심되는 요청을 단순히 차단하는 것은 서비스 이용에 불편을 초래하게 된다. 요청간 상관관계를 분석하여 의도된 요청인지를 서버측 구현코드를 분석하여 검증의 정확성을 높여야 한다.

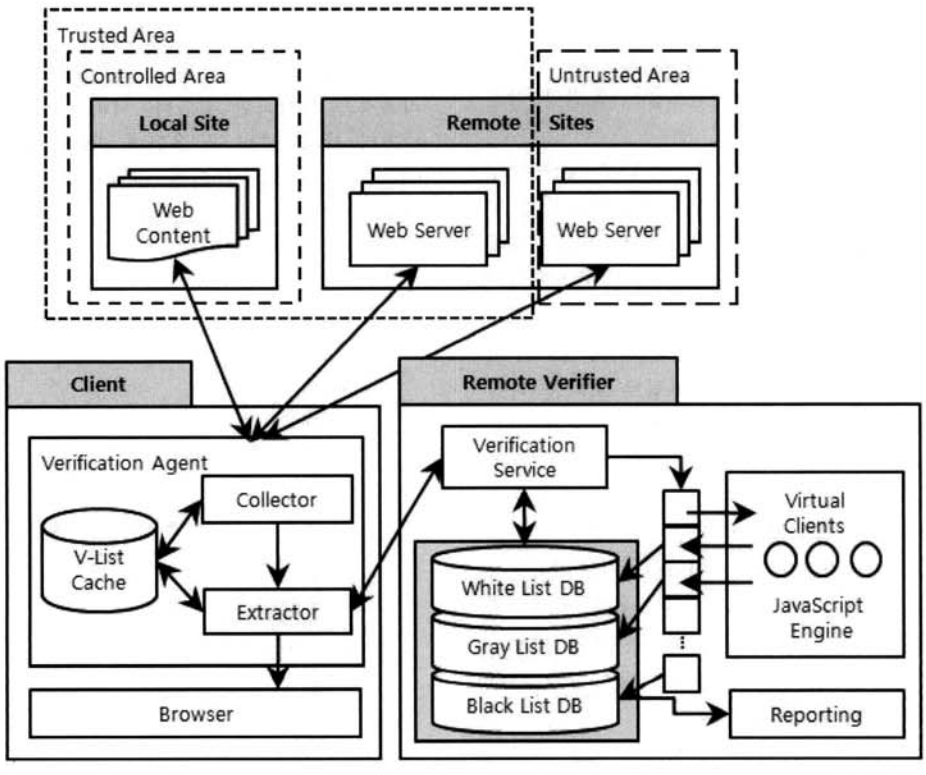
(4) 악성스크립트 발견 즉시 차단하고, 제거 및 복구를 위한 정보 제공

악성스크립트의 발견만큼이나 발견한 악성스크립트에 대한 대응도 중요하다. 발견 즉시 차단하고 차단에 따른 서비스 영향을 최소화하여야 한다. 또한 빠른 복구를 위한 정보를 제공하여야 한다.

3.2 검증 구조

원격코드검증 시스템은 (그림 1)과 같이 서버, 클라이언트, 원격검증서버 3가지 요소로 구성된다.

서버는 특성에 따라 제어영역(Control Area), 신뢰영역(Trust Area), 비신뢰영역(Untrust Area)으로 분류된다.



(그림 1) 원격코드검증 시스템 구조

제어영역은 직접 서비스하는 로컬 사이트로 서비스 제공과 변경에 대한 관리와 책임을 갖는다. 제어영역은 악성스크립트 발견 시 즉각적인 대응과 수정이 가능한 영역으로 구현코드에 대한 접근권한이 있어 서비스 전 사전 분석을 통한 검증에 필요한 정보 확보가 가능하다.

신뢰영역은 로컬 사이트가 서비스 제공 시 의도적으로 상호참조하며 이용하는 서비스 영역이다. 신뢰영역 내에서 크로스 도메인간 요청이 허용되도록 서비스가 구현되며, 서버측 로컬 사이트나 클라이언트 브라우저에서 서비스가 연계된다. 서버측에서 서비스 연계가 이루어지는 경우, AJAX 보안 샌드박스 제약을 극복하기 위해 로컬 사이트가 매쉬업을 위해 서비스 프록시[5]로 동작한다. 로컬 사이트로 원격 사이트의 컨텐츠를 가져와서 로컬 사이트의 컨텐츠와 함께 클라이언트로 전송한다. 클라이언트 브라우저는 출처는 다르지만 로컬 사이트로 컨텐츠를 모아서 응답을 받았기 때문에 동일출처로 보아 SOP 권한으로 원격 사이트 이용이 가능해진다. 브라우저에서의 서비스 연계는 SOP를 우회하여 크로스 도메인간 접근을 통해 이루어진다. 제어영역을 제외한 신뢰영역은 정해진 정책에 따라 자유롭게 서비스를 이용하지만 통제권한은 없는 영역으로 신뢰영역 내에 악성스크립트 발견 시에는 로컬 서버와 클라이언트 브라우저가 접근 못하도록 차단하지만, 악성스크립트 제거와 같은 문제 해결은 원격 사이트의 관리자에게 요청하고 조치되도록 기다릴 수밖에 없다.

비신뢰영역은 서비스 구현 시 의도하지 않은 연결로 주로 UCC나 SNS 등을 통해 이용자에 의해 유도된다. 비신뢰영역은 서비스 연계로 보기는 어렵고 단순한 링크나 참조가 대부분이다. 비신뢰영역으로의 접속 시 접속을 생성하거나 유도하는 부분에서 민감한 데이터에 대한 접근과 외부로의 전달 여부를 확인하고 클라이언트가 컨텐츠 검증 요소를 전달하거나 검증시스템이 해당 URL 컨텐츠를 다운받아 악성행위 여부를 검토한다.

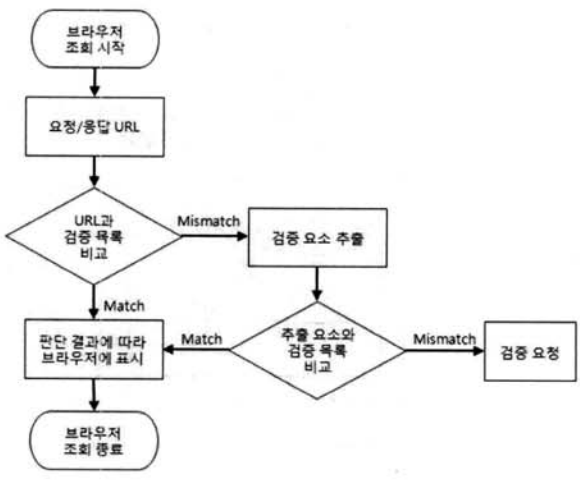
클라이언트는 브라우저와 검증 에이전트(V-Agent)로 구성된다. 검증 에이전트는 브라우저와 서버간 전달되는 요청과 응답을 필터링하여 검증리스트와 비교하며, 비교결과를 브라우저에 반영한다. 수집기(collector 필터)는 브라우저가 서버로 보내는 요청과 응답을 검증캐시(verifying cache)의 블랙 리스트와 비교한다. 정보추출기(Extractor)는 브라우저로 전송되는 컨텐츠의 HTML과 자바스크립트 코드를 파싱하여 요청이 발생하는 URL 정보를 추출한다. 추출된 정보는 3.2절의 검증 요소에 해당한다. 추출 정보는 검증캐시 정보와 비교하고 일치하지 않는 정보는 원격검증 시스템으로 검증 요청을 전송한다. 클라이언트와 검증 시스템간 통신은 SSL(Secure Socket Layer)로 안전한 데이터 교환이 이루어지도록 한다. 검증캐시(V-List Cache)는 원격검증 시스템에서 검증하여 확정된 화이트 리스트와 블랙 리스트를 저장한다. 클라이언트의 응답 속도 향상과 검증 시스템과의 통신 절약을 위해 최신 정보를 유지한다.

원격검증 시스템(Remote Verifier)은 웹사이트의 구현코드와 연계 서비스 정보를 사전 분석하여 URL 정보와 페이지간 상관관계, 구현코드에 대한 서명코드(Signed Code) 정보를 사전 데이터로 보유한다. 분석된 결과 구현코드에 의해 의도된 요청과 서명된 코드 블록을 화이트 리스트로 구축하여 이후 검증 시간을 단축한다. 검증 서비스(Verification Service)는 클라이언트로부터 받은 검증요청에 따라 검증목록DB를 조회한다. 일치하는 정보가 없는 경우 가상 클라이언트에서 위험성을 검증한다. 가상 클라이언트(Virtual Clients)는 검증요청 URL의 웹컨텐츠를 수집하여 자바스크립트 엔진으로 자바스크립트 코드 블록을 실행하여 발생하는 DOM 변경과 URL 요청을 분석하여 악성스크립트 삽입여부를 검증한다. 리포팅(Reporting)은 블랙 리스트로 누적된 악성스크립트 정보를 제공하여 탐지 후 대응을 위한 정보로 활용한다.

3.3 검증 과정

검증은 클라이언트로부터 검증 요청을 받은 URL과 스크립트 코드를 분석하여 사전 구축한 검증 목록과 비교한다. 악성스크립트 삽입 여부가 결정되면 결과에 따라 화이트 리스트나 블랙 리스트로 저장되어 유지 관리된다. 결정이 보류된 경우 자바스크립트 엔진을 이용하여 검증과정을 거치게 된다. 검증 목록과의 비교 검증은 일정한 시간이 보장되며, 클라이언트 에이전트에서 캐시로 보유하기 때문에 실시간 검증이 가능하다. 검증 목록에서 찾을 수 없는 항목은 원격코드검증 과정을 거치게 되며 검증 소요 시간을 예측할 수 없다. 검증이 끝날 때까지 클라이언트가 무한정 기다릴 수 없기 때문에 사용성과 보안성의 트레이드오프를 고려하여 서비스 특성에 따라 적절히 절충한다. 일반적인 경우 접속을 허용하고 검증 결과를 이후 반영하여 클라이언트의 영향을 최소화한다.

(그림 2)는 검증 과정 중 브라우저 내에서 이루어지는 처리과정을 도식화한 것이다. 서비스 이용을 위해 요청을 보내거나 서버로부터 온 응답에 의해 발생하는 컨텐츠 자원의 URL을 원격검증 목록과 비교한다. 검증 목록과 일치하는 경우 화이트/그레이/블랙 리스트 여부에 따라 브라우저 화



(그림 2) 브라우저 내에서의 검증과정

면에 표시한다. 블랙리스트는 경고와 함께 접속을 차단한다. 검증 결과가 없는 경우 컨텐츠를 구성하는 데이터 중에서 악성스크립트 삽입으로 이용될 수 있는 요소들을 검증을 위해 추출한다. 추출한 각 요소에 대해 검증목록과 비교한다. 비교 결과 안전성이 확인되지 않은 요소는 원격검증서버에서 별도의 검증 과정을 거치게 된다.

3.4 요청 분류

해커가 삽입한 악성스크립트에 의해 신뢰 사이트간 개인정보의 유출, 사용자를 가장한 요청 변조 등을 방지하기 위해 브라우저에서 요청이 생성되는 요소에 대해 검사한다. 연계 서비스와 원격지 리소스는 HTML 태그와 자바스크립트 코드를 제외한 서버 사이드 구현은 접근할 수 없다. 로컬 서버는 서버 사이드 구현 코드의 접근이 가능하므로 로컬 리소스의 구현 코드를 분석하여 단순 조회 페이지와 이용자의 세션 정보에 따라 개인정보 조회 페이지, 서버나 사용자 상태정보 변경 페이지로 구분하여 <표 1>과 같이 검증을 차별화 한다.

〈표 1〉 로컬 리소스의 인증 요구에 따른 페이지 분류

페이지 분류	인증 요구	위험 여부
단순 조회	인증 정보 불필요	안전
개인정보 조회	인증 정보 필요	정보 유출 우려
상태 변경	인증 정보 필요	요청 변조

HTTP 요청이 발생할 때 브라우저에 적용되는 보안정책에 따라 브라우저에 로딩된 HTML, 자바스크립트 변수와 함수, 세션, 쿠키 등 데이터에 접근 가능한 범위가 달라진다. 발생하는 요청과 접근 영역간의 관계에 따라 요청을 분류하고 로딩된 페이지와 요청간 상관관계와 요청에 이용되는 데이터를 검증한다.

요청이 발생하는 요청 출처 페이지와 요청URL간 상관관계를 분석하여 요청의 위험성을 판단한다. 동일출처에서 발생하는 요청(Same Origin Request)은 SOP에 의해 안전한 것으로 간주하나 해커에 의해 요청 변조가 발생할 수 있다. 이에 출처(Origin) URL과 목적지(Destination) URL간 요청이 서비스에 의해 의도된 요청인지 악성스크립트에 의해 임의로 발생한 요청인지를 판단한다. 크로스 도메인간 발생하는 요청(Cross Domain Request)은 서비스 연계를 위해 의도된 요청인지를 검사한다. 신뢰 관계가 없는 사이트로의 요청(Un-trusted Site Request)은 단순 링크인지 정보유출인지 판단하기 위해 요청 파라미터 데이터를 검증하여 민감한 정보가 담긴 요청의 발생을 차단한다.

〈표 2〉 요청 출처에 따른 요청 분류

Types of Request	Inspection Items
Same Origin Request	Intended Flow
Cross Domain Request	Intended Interaction
Un-trusted Site Request	Request Parameter

3.5 검증 요소

브라우저가 서비스 이용을 위해 서버로부터 다운로드한 웹컨텐츠에서 사용자의 동작없이 페이지로딩만으로 요청이 생성되거나 페이지 원본의 DOM을 수정하여 새로운 요청을 유도하는 요소에 대해 검증한다. (그림 3)의 사례와 같이 HTML 태그 속성에 따라 GET 요청이 자동으로 발생하기 때문에 이러한 경우를 검증하기 위해 태그 속성이 URL 형태인 속성을 검출하여 출처 페이지에서 의도한 요청인지, 요청에 의해 전달되는 파라미터 값이 안전한지 등을 검사한다.

```
〈META http-equiv="refresh" content="0;url=
http://hacker/hackpage"〉
〈BODY { background: url('http://hacker/hackpage') }〉
〈IMG src="http://hacker/hack.gif?x=session&y=cookie" /〉
```

(그림 3) GET 요청 생성 태그 사례

HTML 문서 안에 자바스크립트는 아래 (그림 4)와 같이 다양한 형태로 삽입이 된다. 페이지에 삽입된 인라인 스크립트 코드 블록과 원격 참조 스크립트 파일의 변경 여부를 확인하기 위해 페이지 내 모든 스크립트 코드를 추출하고 코드 전체에 대한 해시코드 값을 생성하여 코드 값으로 비교 검증한다. 코드 값이 다른 경우 악성스크립트에 이용되는 DOM 객체를 변경하거나 링크정보, 코드 동적 생성이 가능한 자바스크립트 함수 코드를 추출한다. 검출을 어렵게 하기 위해 코드를 난독화하고 인코딩하는 경우 함수 추출이 불가능하다. 이러한 스크립트는 검증 서버의 자바스크립트 엔진을 통해 모의 실행하여 동작을 확인한다.

```
〈DIV STYLE="width: expression(hack());"〉
〈TABLE〉〈TD BACKGROUND="javascript:hack()"〉
〈STYLE type="text/css"〉BODY{background:
url("javascript:hack()")}〈/STYLE〉
〈SCRIPT type="text/javascript"〉code_block〈/script〉
```

(그림 4) 스크립트 코드 삽입 사례

3.6 검증 목록

요청 분류와 검증 요소를 기초로 검증 결과를 목록으로 구축하여 관리한다. 검증 목록 판별을 위해 먼저 알려진 악성스크립트에 대한 시그니처(Malicious Signature) 스캐닝을 수행한다. 시그니처란 악성스크립트를 분석해 내부의 특정 코드를 수집하고 데이터베이스화한 것으로 알려진 악성스크립트에 대한 탐지에 이용된다. 일반적으로 웹방화벽이나 웹스캐너를 통해서 악성스크립트를 검출할 때 이용된다. 두번째로 검증요소의 인증요구를 확인한다. 인증이 필요한 리소스에 접근 시 웹사이트 자체에 구현된 인증 메커니즘을 이용하여 접근 여부를 판단한다. 인증이 유효하면 정상적인 요청으로 간주하나 CSRF[14], XSS[15], 인증우회 공격 등의 위

험이 존재한다. 세번째로 서명코드(Signed Code)를 비교하여 스크립트 코드의 무결성을 확인한다. 서명코드는 스크립트 코드 전체에 대한 해시코드 값으로 비교하여 의도하지 않은 실행코드의 삽입이나 변경을 간단한 연산으로 검증한다. 마지막으로 요청출처와 목적지(Destination)간 요청이 서비스에 의해 의도된 요청(Intended Interaction)인지 확인한다.

검증목록은 검증 항목의 평가 결과에 따라 <표 3>과 같이 화이트 리스트(White List), 그레이 리스트(Gray List), 블랙 리스트(Black List)로 분류된다. 화이트 리스트는 서비스 이용에 안전이 확인된 웹컨텐츠 정보를 기초로 구축된 목록으로 웹사이트의 안전성과 구현코드의 무결성, 페이지간 동작 과정의 검증이 입증된 대상이다. 화이트 리스트는 로컬사이트의 검사와 분석을 통해 사전 구축이 가능하며 구현 코드 변경 시 마다 화이트 리스트를 갱신하여 검증과정의 성능과 정확성을 높일 수 있다. 초기 화이트 리스트 생성을 위해 검증 대상 사이트의 서버측 소스코드를 분석하여 인증에 따른 로직의 분기 여부에 따라 페이지 목록을 생성하고 호출 가능한 페이지간 연관성을 파악한다. 페이지에 포함된 자바스크립트 코드블록과 외부참조 리소스에 대한 해시코드를 생성하여 저장한다. 분석 대상의 안전성 확인을 위해 웹스캐너로 시그니처 검사를 수행한다.

〈표 3〉 초기 화이트/그레이/블랙 리스트 판별 기준

	Malicious Signature	Authentication	Signed Code	Intended Interaction
White List	No	Yes	Yes	Yes
	No	No	Yes	Yes
Gray List	No	Yes	Yes	No
	No	Yes	No	Yes
	No	Yes	No	No
Black List	Yes	Yes	No	No

블랙 리스트는 악성 사이트로의 접속을 유도하거나 악성 행위를 수행하는 코드의 삽입이 확인된 웹컨텐츠 목록이다. 블랙 리스트가 발생 시 사이트 내에 악성스크립트가 발견된 것으로 블랙 리스트에 접속하지 못하도록 로컬서버와 브라우저 양측에서 요청 시도를 차단한다. 사이트의 위변조나 악성스크립트가 발견 시 1차적으로 시스템에 의해 자동 차단되지만 실질적인 조치는 블랙 리스트 정보를 이용해 수동으로 사이트 복구 및 악성스크립트 제거를 수행한다. 악성스크립트가 제거되면 페이지 내 요소 검증을 통해 블랙 리스트 항목이 자동으로 감지되기 때문에 이후 검증 시 화이트 리스트나 그레이 리스트로 처리된다.

그레이 리스트는 화이트 리스트나 블랙 리스트로의 판별이 보류된 목록이다. 악성스크립트 탐지가 진행 중이거나 탐지 결과 위험성 판단이 보류된 경우에 해당한다. 오탐율을 감안하여 악성스크립트 여부에 대한 확정이 어려운 경우는 판단을 보류하고 별도의 보고과정을 거쳐 수동으로 코드 검토(code inspection)를 통해 최종 확정한다.

3.7 탐지 시 대응

악성스크립트 탐지 시 단순 차단으로 인한 서비스 영향을 최소화하면서 효과적인 차단을 위해 정보량과 제어권한에 따라 대응 방안을 차별화한다. 서버 측에서 제어, 신뢰, 비신뢰 영역에 따라 다음과 같이 대응한다.

제어영역은 악성스크립트 발견 시 즉각적인 대응이 가능한 영역으로 제어영역의 로컬 사이트의 구현 코드 중 악성스크립트가 존재하는 웹컨텐츠의 코드블록을 차단하고 발견된 웹컨텐츠의 URL과 코드블록 위치를 리포팅한다. 제어영역은 리포팅 정보를 이용하여 발견 즉시 수정이 가능하다.

제어영역을 제외한 신뢰영역은 통제권한이 없기 때문에 악성스크립트가 삽입된 URL과의 연계를 차단하고 연계 서비스 제공자에게 악성스크립트 관련 정보를 제공하여 빠른 조치를 요청한다. 그러나 신뢰영역에 대한 통제권한이 없기 때문에 악성스크립트 제거가 늦어질 경우 서비스에 영향을 받게 된다. 연계 서비스 차단이 장기화될 경우 원활한 서비스를 위해 로컬 사이트 구현을 변경할 필요가 있다.

비신뢰영역은 서비스 제공에 직접적인 영향을 미치지 않는 영역으로 주로 이용자에 의한 추천 링크가 이에 속한다. 비신뢰영역에서의 악성스크립트는 차단 및 제거가 가능하다. 재발방지를 위해 리포팅하고 로컬 사이트에서 링크정보를 제거한다.

4. 실험 및 분석

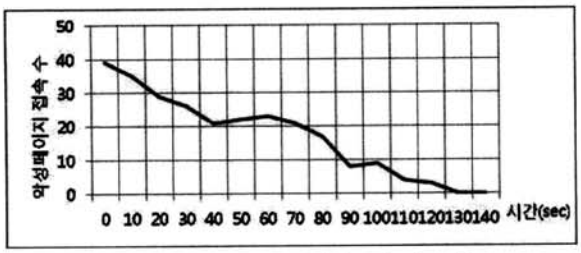
4.1 실험 환경/시나리오

본 논문에서 제안한 원격코드검증을 통한 웹컨텐츠의 악성스크립트 탐지 모델의 효과를 확인하기 위하여 실험을 하였다. 아파치 웹서버를 이용하여 실험을 위한 사이트를 구축하고 악성스크립트를 삽입하였다. 악성스크립트는 [16]에서 제시한 악성스크립트 코드를 응용하고 확장하여 실험에 사용하였다. [16]에서는 [17]에서 제시한 대표적인 웹사이트 침해유형을 실험을 통해 연구해 볼 수 있는 샘플 악성스크립트 코드를 제시하고 있다.

클라이언트의 검증 에이전트의 역할을 수행할 프로그램을 자바 쓰레드로 구동하여 다수의 클라이언트가 동시에 웹서버를 이용하는 환경을 구축하여 악성스크립트가 삽입된 페이지로의 요청과 검증 시스템의 처리에 따른 변화를 확인하였다.

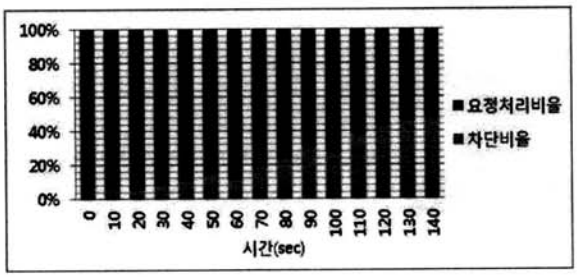
4.2 결과 분석

(그림 5)는 시간에 따른 클라이언트의 악성스크립트 삽입 페이지의 접속 회수를 나타낸다. 블랙 리스트 판별 전에는 악성스크립트 삽입 페이지에 대한 접속 차단이 이루어지지 않는다. 그러나 검증 시스템에서 각 페이지에 대한 검증 결과가 누적됨에 따라 악성 페이지로의 접속이 차단됨을 알 수 있다. 모든 페이지에 대한 검증이 끝난 이후에는 악성 페이지로의 접속이 0이 되었다. 이는 실제 서비스 시 신규로 생성되거나 링크되는 페이지에 대해서 검증 과정에서 일시적으로 클라이언트가 접속할 수 있으나 검증과정을 통해 악성스크립트를 발견하여 접속률이 0.0로 수렴된다.



(그림 5) 시간대별 악성페이지 접속 수

실험에서는 클라이언트가 일정 시간 간격으로 페이지를 요청하게 하였다. 페이지 중 블랙리스트로 확정된 경우 클라이언트에서 요청을 차단한다. (그림 6)은 클라이언트가 서버로 요청을 보내기 전에 검증 에이전트에서 요청을 차단한 회수를 합산한 결과다. 시간이 경과함에 따라 페이지의 악성스크립트 삽입 여부가 확인되어 악성페이지로의 접속 차단이 증가하였다.



(그림 6) 시간대별 악성페이지 접속 차단

4.3 제안 모델의 평가

본 논문의 제안 모델은 3.1의 요구사항에 따라 연계 서비스 형태의 웹사이트 안전성 확보를 위해 제어영역과 신뢰영역, 비신뢰영역으로 서비스를 구분하고 영역간 웹요청을 차별화하여 분석하는 방안을 제시하였다. 서버의 응답으로 브라우저에 다운로드 되어 실행되는 코드와 발생하는 웹트래픽을 웹브라우저 관점에서 수집하여 서비스 동작에 대한 상세한 정보를 보유하고 있는 원격의 검증 시스템에 안전성을 확인하도록 하였다. 악성스크립트가 탐지되는 경우 제어영역의 코드 블록을 차단하고 신뢰영역과 비신뢰영역으로의 연결을 차단하는 방안을 제시하였다. <표 4>는 관련 연구와의 비교 결과다.

블랙리스트 기반 탐지[6,7,8]는 전세계 도메인을 대상으로 인터넷 사이트 전체를 처리하기에 시간과 비용이 많이 소요된다. 도메인 기반 블랙리스트 정보는 피싱 사이트의 주기가 짧아지고 검출하는데 시간이 소요되기 때문에 최신 업데이트가 지연되는 단점이 있다. 반면 제안 모델은 제공 서비스에 중점을 두고 능동적으로 서비스를 방어하기 위한 방안으로 탐지 범위가 한정적이며 상대적으로 적은 비용으로 적용이 가능하다.

크로스 도메인간 정책부여[2,9,10,11]는 크로스 도메인 요청 시 정책에 따른 권한을 부여하는 모델로서 연계 서비스간 상호 정책 수립이 필수이며, 정책에 의한 신뢰관계를 맺은 사이트가 내포한 취약점이 전이될 수 있다. 제안 모델에서는 코드 내용을 검증 대상으로 하여 요청 출처에 동일출처, 크로스 도메인, 비신뢰 출처로 구분하여 상세한 검증을 수행한다.

〈표 4〉 관련 연구와의 비교

비교	블랙리스트 기반 탐지	크로스 도메인 정책부여	요청 응답 조작	제안 모델
서비스 영역 탐지	X	X	X	O
연계 서비스 분석	X	X	X	O
외부 사이트 검증	O	X	X	O
실시간 검증	X	O	O	O
코드 검증	X	X	O	O
정책 부여	X	O	X	O

요청 응답 조작[12,13]은 사이트 내부에 국한되며 외부 페이지를 클라이언트 브라우저가 다운받아 실행하는 경우 검증이 어렵다. 요청 파라미터를 조작하거나 응답 페이지를 변경하는 경우 서비스 이용에 지장을 초래한다. 제안 모델을 이용 시 코드 변경없이 클라이언트 브라우저와 협업을 통해 검증을 수행할 수 있다.

5. 결 론

브라우저는 기본 보안 모델인 동일출처정책(SOP)을 통해 동일출처내에서 컨텐츠간 접근을 허용하였으나 개방, 참여, 공유로 대표되는 웹2.0으로의 서비스 환경이 변화되면서 다양한 출처의 리소스를 상호 참조한다. 사용자측에서는 서비스 이용 시 제공받는 컨텐츠가 어떤 출처로부터 온것인지 인지하기 어렵고, 악성스크립트의 삽입 위치가 분산되어 탐지 및 차단이 어렵다.

본 논문에서는 신뢰관계에 있는 여러 출처로부터 클라이언트 브라우저로 다운받은 웹컨텐츠를 원격검증시스템에게 검증 요청하여 악성스크립트의 삽입 여부를 탐지하는 모델을 제안하였다. 서비스의 내부 동작에 대한 정보가 부족한 클라이언트에서는 컨텐츠의 악성행위 여부를 판단하기 어렵지만 원격검증시스템은 서비스 구현코드 정보를 활용하여 요청 출처에 따라 검증 항목을 분류하고 검증 요소를 추출하여 검증 평가결과를 화이트 리스트, 그레이 리스트, 블랙리스트로 데이터베이스화하여 검증 효율을 높일 수 있음을 확인하였다.

본 논문의 결과는 [17]에서 제시한 웹사이트 침해유형 중 현재 해결이 어려운 매쉬업 기반의 웹환경의 보안문제를 해결할 수 있음을 보여준다.

향후 실제 서비스되고 있는 웹사이트의 서비스 유형을 분류하고 구현 특성을 분석하여 제안모델을 적용 시 비용과 효과를 정량화하고 비용을 줄이며 효과를 높일 수 있도록 제안모델을 발전시키는 보완연구를 수행할 것이다.

참 고 문 헌

[1] Wikipedia, "Same origin policy", http://en.wikipedia.org/wiki/Same_origin_policy.

[2] A. van Kesteren, "Cross-Origin Resource Sharing", http://www.w3.org/TR/cors/, W3C Working Draft, 2010.

[3] S. Hanna, E. Chul, R. Shin, D. Akhawe, A. Boehm, P. Saxena, and D. Song, "The emperor's new APIs: On the (in)secure usage of new client-side primitives", Web2.0 Security and Privacy Conference (W2SP), 2010.

[4] Adobe, "Cross-domain policy file specification", http://www.adobe.com/devnet/articles/crossdomain_policy_file_spec.html, 2010.

[5] Yahoo Developer Network's Javascript Developer Center, "JavaScript: Use a Web Proxy for Cross-Domain XML HttpRequest Calls", http://developer.yahoo.com/javascript/howto-proxy.html.

[6] Google, "Google safe browsing", http://code.google.com/apis/safebrowsing/.

[7] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu. "The ghost in the browser analysis of web-based malware", Proc. Of the USENIX Workshop on Hot Topics in Understanding Botnets (HotBots), 2007.

[8] Microsoft, "SmartScreen Filter | Internet Explorer 8 Security.", http://www.microsoft.com/security/filters/smartscreen.aspx.

[9] W. Maes, T. Heyman, L. Desmet, and W. Joosen. "Browser Protection against Cross-Site Request Forgery". In Workshop on Secure Execution of Untrusted Code (SecuCode), 2009.

[10] C. Jackson and H. J. Wang. "Subspace: Secure Cross-Domain Communication for Web Mashups". In Proceedings of the 16th International World Wide Web Conference (WWW), 2007.

[11] F. D. Keukelaere, S. Bhola, M. Steiner, S. Chari, and S. Yoshihama. "Smash: secure component model for cross-domain mashups on unmodified browsers". In Proceeding of the 17th international conference on World Wide Web (WWW), 2008.

[12] C. Reis, J. Dunagan, H. J. Wang, O. Dubrovsky, and S. Esmeir. "BrowserShield: Vulnerability-Driven Filtering of Dynamic HTML", In Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2006.

[13] A. Moshchuk, T. Bragin, D. Deville, S. D. Gribble, and H. M. Levy. "SpyProxy: Execution-based Detection of Malicious Web Content", In Proceedings of the 16th USENIX Security Symposium, 2007.

[14] OWASP Foundation, "Cross-Site Request Forgery(CSRF)". https://www.owasp.org/index.php/Cross-Site_Request_Forgery, 2010.

[15] OWASP Foundation, "Cross-site Scripting". https://www.owasp.org/index.php/Cross-site_scripting, 2010.

[16] Attack & Defense Lab, "Cross Origin Requests Security", http://www.andlabs.org/html5.html

[17] 민병준, 김성기, 최재영 외, "모바일 접속환경을 위한 웹사이트 침해예방 연구", 한국인터넷진흥원, 2010.09.

최 재 영

e-mail : jero@incheon.ac.kr
1999년 인천대학교 컴퓨터공학과(학사)
2001년 인천대학교 컴퓨터공학과(석사)
2010년~현 재 인천대학교 컴퓨터공학과 박사과정
관심분야 : Computer & Network Security, Web Security

김 성 기

e-mail : skim@sunmoon.ac.kr
1996년 인천대학교 전자계산학과(학사)
1998년 인천대학교 컴퓨터공학과(석사)
2006년 인천대학교 컴퓨터공학과(박사)
2006년~2009년 인천대학교 초빙교수
2009년~현 재 선문대학교 IT교육학부 전임강사
관심분야 : Computer & Network Security, Distributed Systems, Ubiquitous Computing, Smart Grid, IT교육

이 혁 준

e-mail : hjlee@incheon.ac.kr
2010년 인천대학교 컴퓨터공학과(학사)
2011년~현 재 인천대학교 컴퓨터공학과 석사과정
관심분야 : Computer & Network Security

민 병 준

e-mail : bjmin@incheon.ac.kr
1983년 연세대학교 전자공학과(학사)
1985년 연세대학교 전자공학과(석사)
1991년 미국캘리포니아대학교(UC Irvine) 전기및컴퓨터공학과(박사)
1995년~현 재 인천대학교 컴퓨터공학과 교수
관심분야 : Computer & Network Security, Distributed Systems