Introduction to Cross-Site Request Forgery and Defense Method

PLASSE 김성진

Hanyang University

August 30, 2010

4th ROSAEC Workshop(2010.8.25 - 8.28), 설악 대명콘도

CSRF란 무엇인가?

- Cross-Site Request Forgery의 약자로 웹 어플리케이션에 특화된 공격방법
  - 공격자가 제 3자의 브라우저를 통해 해당 사용자로 위장하여 승인된 작업을 수행가능
  - 캐시된 브라우저에서 제공된 자격증명을 이용하여 웹어플리케이션의 신뢰를 악용
  - 2010년 OWASP TOP 10에서 5번째 순위로 선정
- 성공적인 CSRF 공격을 위해서 다음 조건이 요구
  - 대상이 되는 웹사이트에서 쿠키나 HTTP인증 등의 암시적인 인증을 사용해야함
  - 피해자가 대상 웹 어플리케이션에 이미 인증을 받은 상태여야하며, 사용자의 브라우저는 인증정보를 캐시
  - 대상 웹 어플리케이션에 HTTP 요청을 수행하기 위해 피해자 브라우저의 사용권한 획득이 요구됨

CSRF공격이란?

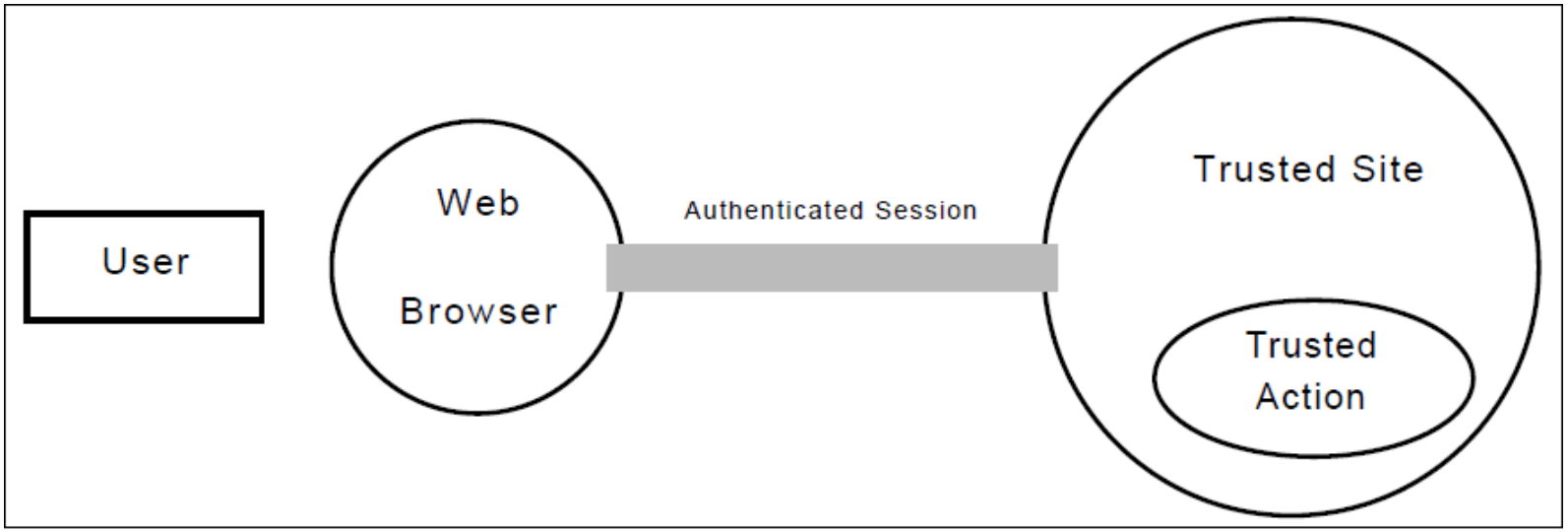

Figure 1 : Initial state

CSRF공격이란?

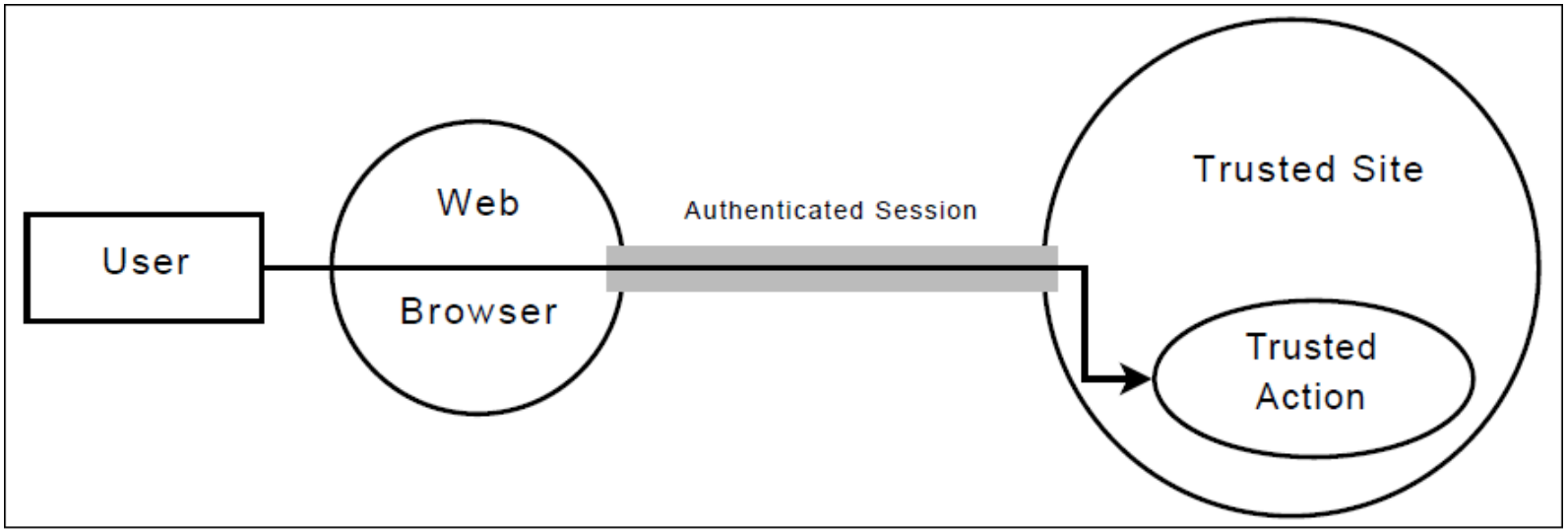

Figure 2 : A valid request

CSRF공격이란?

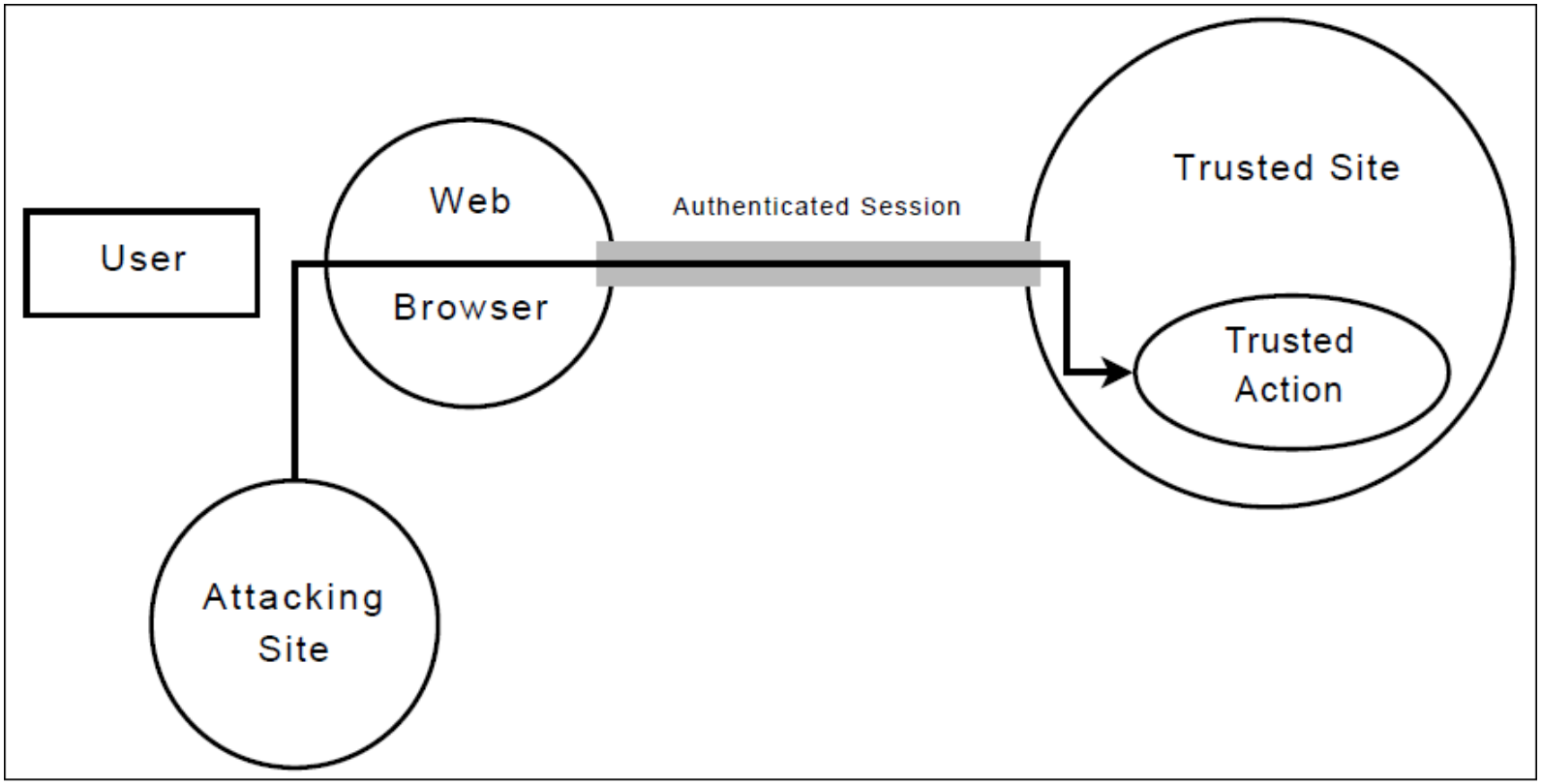

Figure 3 : A CSRF attack

XSS와 CSRF의 비교

	XSS	CSRF
스크립트의 필수여부	O	X
XSS에 취약한 사이트	공격 가능	공격 가능
XSS를 방어하는 사이트	공격 불가능	공격 가능성 있음

CSRF의 공격 예

- 어떤 특정한 사이트에서 CSRF 공격이 가능한 공격지점 탐지
- 정상적인 요청을 변조하여 이를 <img> 태그 등에 은닉
- 목표 대상 혹은 불특정한 피해자에게 쪽지, e-mail 등으로 전송
- 피해자가 해당 내용을 열람할 때 실제적인 피해발생

CSRF의 실제 공격사례 - Mahalo.com

- 트위터나 마이스페이스와 비슷한 종류의 의견교류 사이트
- 질문을 등록하는 폼과 쇼핑관련 모듈에 취약점 존재
  - 어떠한 상품을 주문할 경우에 배송지 정보가 암호화과정 없이 원문 그대로 노출 -> 조작 가능성 내포
  - 질문을 등록하는 폼에 스크립트나 태그를 포함해서 등록가능
- 공격자는 이것을 이용하여 특정 상품을 공격자가 지정한 장소로 배송되도록 악성요청 작성가능
- 질문을 읽은 불특정 사용자는 자신도 모르게 악성요청이 실행되어 피해발생
- 이 취약점을 사이트측에 통보한 결과 현재는 수정이 완료된 상태

Mahalo.com의 공격과정

- 정상적인 배송정보를 입력

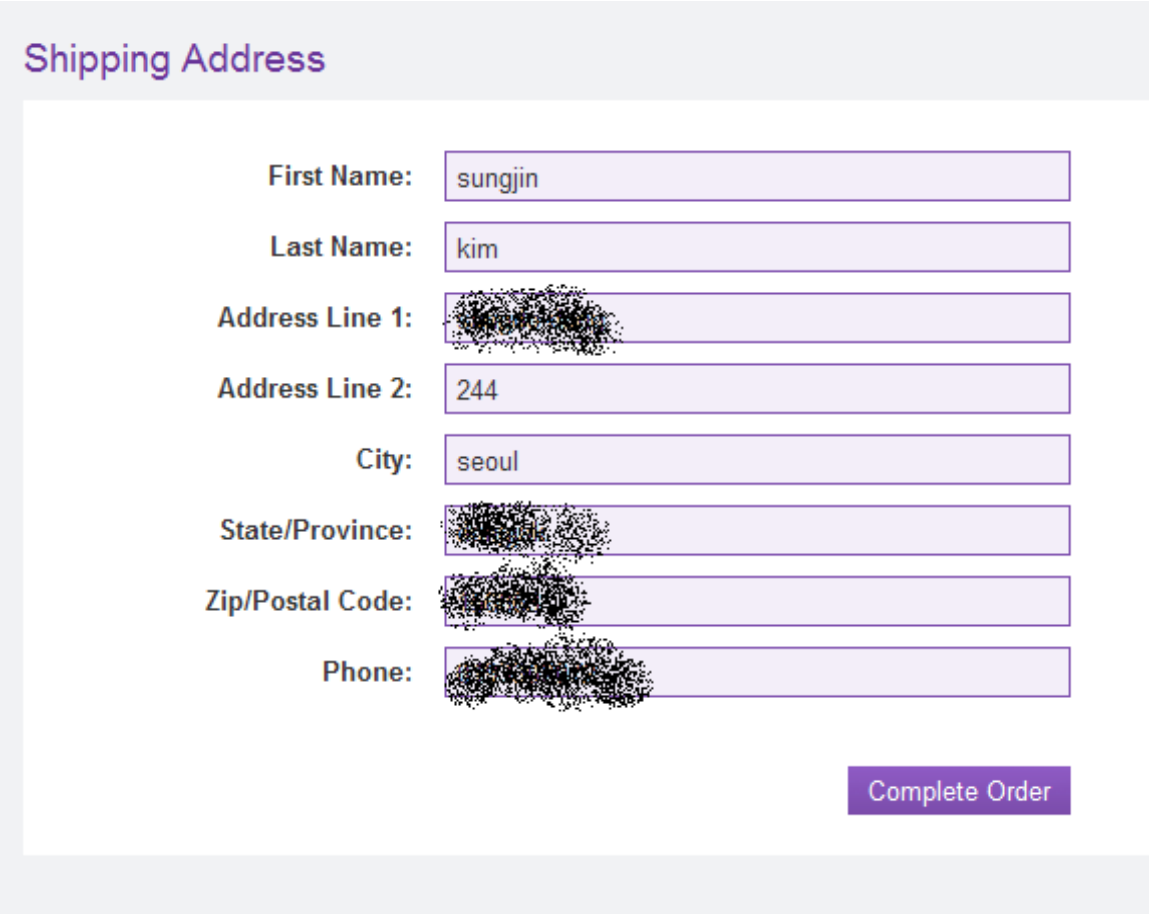

Figure 1 : Input normal delivery information

Mahalo.com의 공격과정

- proxy 툴을 이용하여 배송정보 전달시의 요청을 intercept

Parsed | Raw

```
POST http://www.mahalo.com:80/api/store/complete_order/ HTTP/1.1
x-requested-with: XMLHttpRequest
Accept-Language: ko
Referer: http://www.mahalo.com/store/checkout
Accept: application/json, text/javascript, */*
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; InfoPath.2)
Host: www.mahalo.com
Content-length: 363
Proxy-Connection: Keep-Alive
Pragma: no-cache
Cookie: sessionid=344d0ed950e96095cde3d257eaeaf869; _chartbeat=zu3rsxt9st30jjk2; __qca=P0-1455203616-1281015039187; __utma=192772534.263341927.1281187949.1281450054.1281613377.8; __utmz=

first_name=sungjin&last_name=kim&address_1=[illegible]address_2=244&city=seoul&state=[illegible]&zip=[illegible]phone=[illegible]&key_owner=mahalo&access_key=a72949203c050dc9ad53b30ff164f4
```

Figure 2 : Intercept request using proxy tool

Mahalo.com의 공격과정

- 요청조작의 성공여부를 알기 위해 모의로 조작된 요청이 포함된 질문 등록
- 조작된 요청이 실제로 적용되었는지를 확인하기 위해 쇼핑센터 방문

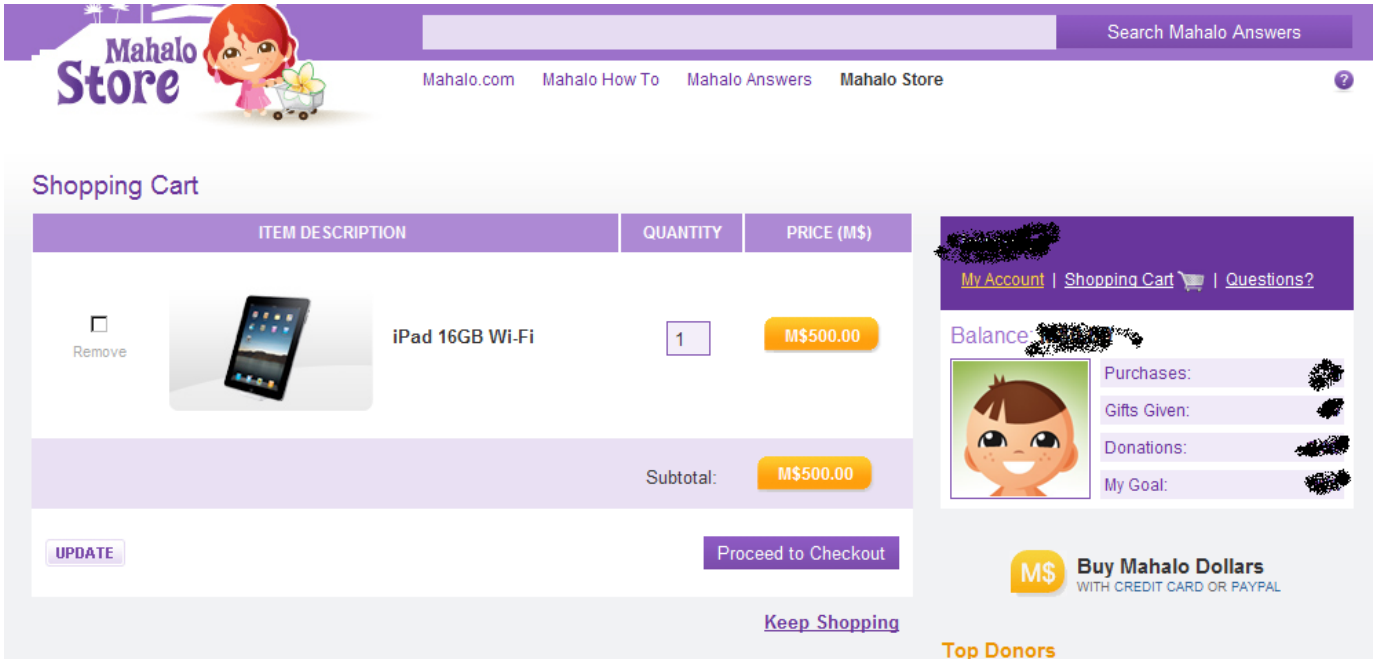

Figure 3 : Check shopping cart

Mahalo.com으로의 통보

- 위의 두 가지 취약점을 Mahalo.com으로 통보(8/13)

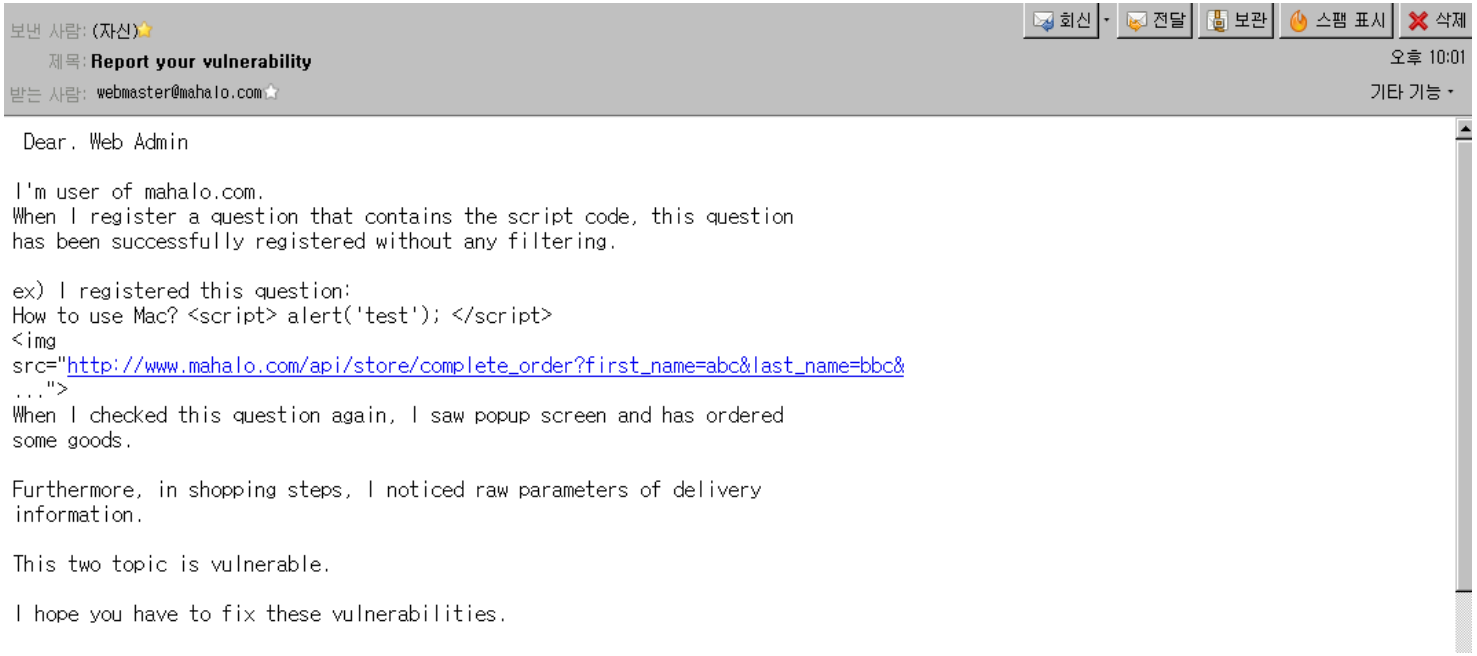

Figure 4 : Report vulnerability to Mahalo.com

Mahalo.com의 답변

- Mahalo.com에서의 답변

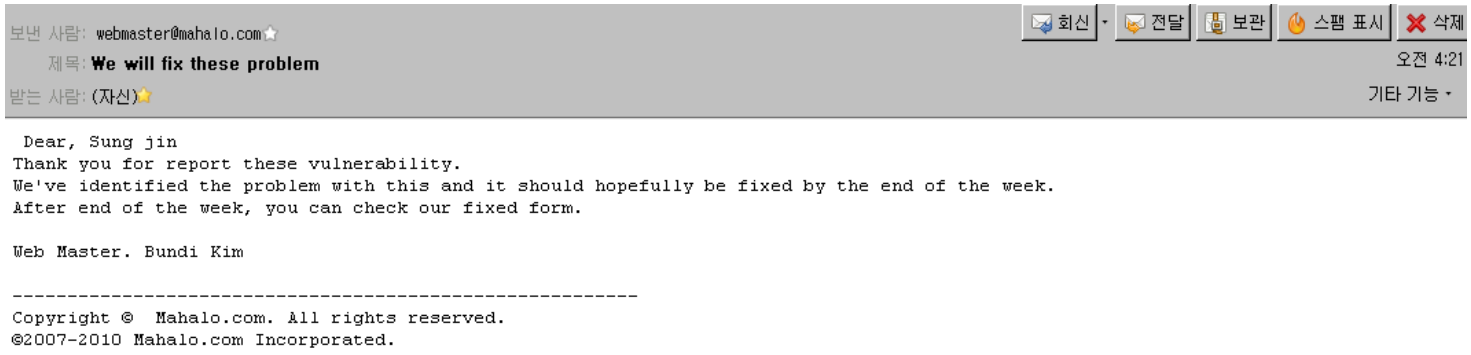

Figure 5 : Response from Mahalo.com

수정된 사항 확인

- 질문을 등록하는 폼에 스크립트나 비정상 태그 사용금지
- 배송관련 요청은 아직 암호화모듈이 적용되지 않음

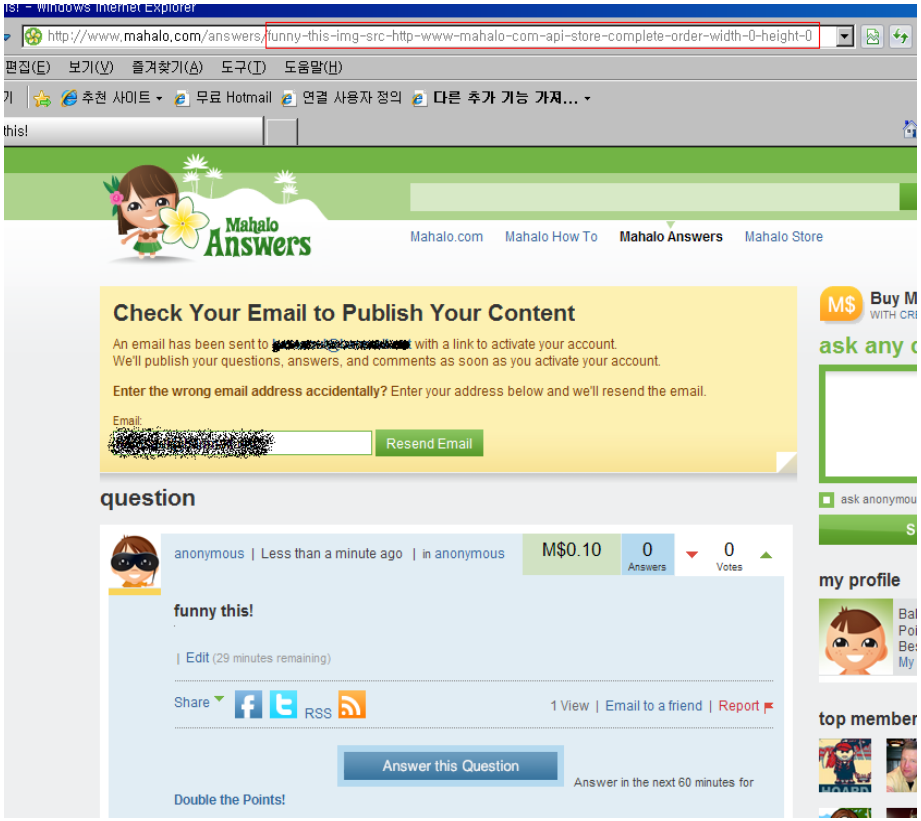

Figure 6 : Check improved question form

현재의 CSRF 방어방법

- 서버측 방어방법
  - 비밀 인증토큰
  - 참조헤더
- 클라이언트측 방어방법
  - SOP(Same Origin Policy)
  - 정책엔진을 통한 요청검사

서버 - 비밀 인증토큰

- 모든 HTTP 요청에 적절한 인증토큰을 끼워넣어 인증된 곳에서의 요청인지의 여부를 결정
- 인증토큰은 다음 네가지로 분류
  - Session Identifier : 모든 요청 절차에서 세션정보를 확인
  - Session-Independent Nonce : 세션에 독립적인 1회성 임시번호 이용
  - Session-Dependent Nonce : 세션에 의존적인 1회성 임시번호 이용
  - HMAC of Session Identifier : 세션정보를 해시기반의 HMAC으로 변환

서버 - 참조헤더

- 참조헤더의 내용을 살펴보아 요청의 근원지가 어디인지를 판별
- 그 인증 강도에 따라서 두 가지로 분류
  - lenient Referer validation : 참조헤더가 없는 ftp나 data URL에 대해서는 인증절차 생략
  - strict Referer validation : 참조헤더가 없는 경우에도 내용을 분석하여 적절히 차단

클라이언트 - SOP

- SOP란 Same-Origin Policy의 줄임말로 같은 곳에서의 요청인지를 검사하는 정책
- 제 3의 사이트에서 온 데이터를 읽어들이는 것을 방지
- 그러나 제 3의 사이트에서 온 요청은 방어하지 못함

클라이언트 - 정책엔진 모델

- 아래 그림과 같이 클라이언트에서 만들어진 요청을 중간에 가로채서 정책엔진으로 검사

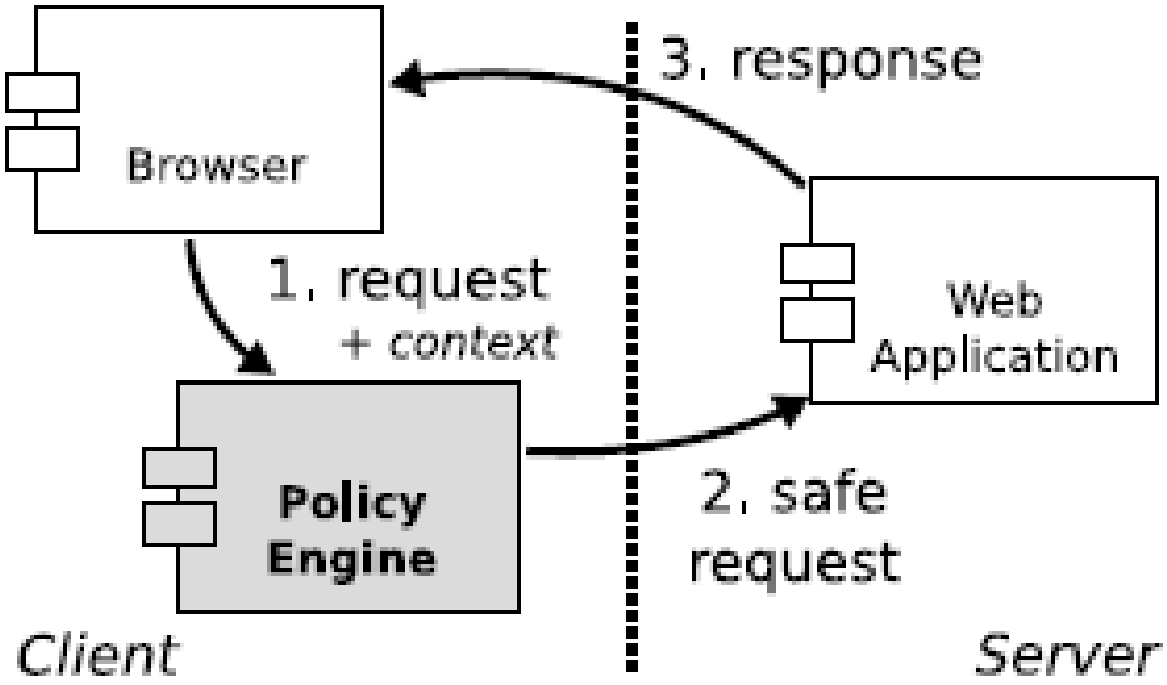

Figure 1 : Request interception.

기존방법의 한계점

- 현재의 방어방법은 모두 한계점이 존재
  - 서버측 방어방법 : 인증토큰과 참조헤더 모두 조작가능성 내포
  - 클라이언트측 방어방법 : SOP정책의 한계점과 정책엔진의 완벽하지 못한 신뢰도
- 이미 만들어진 요청을 검사하는 단계를 뛰어넘어 악성 요청자체가 만들어지지 못하게 소스코드 단계에서의 방법이 필요함

제안사항 - 소스코드의 강화

- input form에 입력되는 내용들을 검사하는 모듈 구현
  - 일반적인 input form의 경우에는 html 태그를 사용하지 못하도록 제한
  - 게시판이나 이메일의 내용을 입력받는 form의 경우에는 html 태그를 허용하되, 제한된 범위로 사용허가
- CSRF공격에 주로 이용되는 태그의 처리방법
  - img 태그 : src내부의 내용이 반드시 이미지확장자로 종결되는지 검사
  - iframe 태그 : src내부의 내용을 검사하여 parameter가 없는 경우에만 허용

결론

- CSRF는 아직 잘 알려져있지 않은 취약점이지만 충분한 위험성 내포
- CSRF를 방어할 수 있는 몇 가지 방법이 있지만 근본적인 방법은 아님
- 소스 자체의 방어필터 강화로 CSRF 공격으로부터 근본적인 안전성을 보장
- 더 나아가 웹 기술의 복잡도가 증가하는 만큼 새로운 패턴의 공격방법을 예측하고 방어할 준비가 필요함

참고문헌

1. W. Zeller and E. W. Felten. Cross-Site Request Forgeries : Exploitation and Prevention. Oct 2008.
2. A. Barth, C. Jackson, and J. C. Mitchell. Rubust Defenses for Cross-Site Request Forgery. In CCS, 2008.
3. W. Maes, T. Heyman, L. Desmet, and W. Joosen. Browser Protection against Cross-Site Request Forgery. In SecuCode, 2009.