ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

OTACUS : 간편URL기법을 이용한 파라미터변조 공격 방지기법

OTACUS: Parameter-Tampering Prevention Techniques using Clean URL

김 귀 석[1] 김 승 주[1*]
Guiseok Kim Seungjoo Kim

요 약

웹 애플리케이션에서 클라이언트와 서버간의 정보전달의 핵심요소인 URL 파라미터는 F/W이나 IPS등의 네트워크 보안장비를 별다른 제약없이 통과하여 웹서버에 전달된다. 공격자는 이렇게 전달되는 파라미터를 변조하여 조작된 URL을 요청하는 것만으로도 인가받지 않은 기밀정보를 유출하거나 전자상거래를 통하여 금전적 이익을 취할 수 있다. 이러한 파라미터변조 취약점은 해당 애플리케이션의 논리적 판단에 의해서만 조작여부를 확인할 수 있어 웹 방화벽에서 차단할 수 없다. 이에 본 논문에서는 기존 방지기법의 취약점을 점검하고 이를 보완하는 OTACUS (One-Time Access Control URL System)기법을 제시한다. OTACUS는 파라미터가 포함된 복잡한 URL을 단순화 하는 간편URL기법을 이용하여 공격자에게 URL노출을 막음으로써 POST이나 GET방식으로 서버로 전달되는 파라미터의 변조를 효과적으로 차단할 수 있다. 실제 구현된 OTACUS의 성능 실험결과 3%이내의 부하가 증가함을 보여 안정적인 운영이 가능함을 증명한다.

☞ 주제어 : 웹서비스 보안, URL파라미터, OWASP

ABSTRACT

In a Web application, you can pass without restrictions special network security devices such as IPS and F/W, URL parameter, which is an important element of communication between the client and the server, is forwarded to the Web server. Parameters are modulated by an attacker requests a URL, disclose confidential information or through e-commerce, can take financial gain. Vulnerability parameter manipulation thereof cannot be able to determine whether to operate in only determined logical application, blocked with Web Application Firewall. In this paper, I will present a technique OTACUS(One-Time Access Control URL System) to complement the shortcomings of the measures existing approaches. OTACUS can be effectively blocked the modulation of the POST or GET method parameters passed to the server by preventing the exposure of the URL to the attacker by using clean URL technique simplifies complex URL that contains the parameter. Performance test results of the actual implementation OTACUS proves that it is possible to show a stable operation of less than 3% increase in the load

☞ keyword : Web service Security, URL Parameter, OWASP

1. 서 론

인터넷서비스의 대중화로 인하여 PC 뿐만 아니라 스마트폰이나 태블릿같은 미디어에서도 단순한 정보검색을 넘어 인터넷뱅킹이나 각종 민원처리까지 대부분의 업무들이 웹 애플리케이션을 통하여 서비스 하는 등 웹은 오늘날 가장 널리 사용되는 플랫폼 중 하나이다. 이렇게 웹 애플리케이션이 다루는 영역이 넓어지고, 중요한 가치의 서비스를 제공하면서 공격자에게는 공격의 난이도에 비하여 높은 효과를 얻을 수 있는 영역이 되고 있다. 기업들이 네트워크 방화벽이나 IDS/IPS와 같은 네트워크 layer의 보안장비를 구비하면서 공격자의 시스템 직접 접근은 어려워진 반면, 웹 애플리케이션의 환경은 서버와 클라이언트로 구분되고 클라이언트에 해당하는 웹 브라우저가 웹서버의 통제권 밖에 존재하면서 공격자가 웹서버로 전달되는 값을 임의로 변조할 수 있다는 구조로 인하여 다양한 취약점이 발견되고 있다. XIAOWEI LI는 웹 애플리케이션의 취약점을 1)입력값 검증 취약점 2)세션 관리 취약점 및 3)애플리케이션 논리 취약점으로 구분하였다[1]. 그 중 애플리케이션 논리 취약점은 웹 애플리케이션의 기능에 따라 다양하게 존재하는데, 일반적인 유형은 공격자가 예측 가능한 URL을 통하여 기밀정보 및 작업에 액세스하거나 비즈니스 로직의 처리순서를 의도

[1] CIST(Center for Information Security Technologies), Korea University, Seoul, 136-713, Korea
* Corresponding author (skim71@korea.ac.kr)
[Received 12 June 2014, Reviewed 10 July 2014, Accepted 1 October 2014]

적으로 회피할 수 있다. 예를 들어, 취약한 전자 상거래 웹 사이트 는 공격자가 가격을 줄이기 위해 동일한 쿠폰을 여러 번 적용하거나 결제 처리 절차 동안 비용이나 세금 계산 단계를 우회 할 수 있다. 이러한 취약점은 클라이언트로부터 전송되는 파라미터 등의 상태 값을 변조함으로써 발생하며 OWASP 2013 TOP 10의 A7.기능수준의 접근통제누락, A4.취약한 직접객체 참조, A10.검증되지 않은 리다이렉트 및 포워드 등이 논리취약점에 기인한다고 정의하였다[1]. 최근 전문보안업체와 학계의 많은 연구 성과로 개발된 웹 방화벽의 도입은 SQL Injection이나 XSS 같은 입력값 검증 취약점이나 세션관리 취약점에 대한 공격을 효과적으로 차단할 수 있으나, 파라미터 변조공격으로 유발되는 애플리케이션 논리취약점은 각 애플리케이션의 사용시점이나 용도에 따라 전달되는 파라미터의 논리적 검증 메커니즘이 서로 다르기 때문에 웹 방화벽에서의 논리적 판단이 불가능하므로 자동적인 차단이 어렵다. 이 취약점을 개선하려면 우선 해당 애플리케이션의 논리적 처리 절차에 대한 충분한 이해가 선행되어야 하므로 다른 취약점에 비하여 상대적으로 최근에서야 그 위험성이 주목받고 있다.

파라미터 변조공격의 심각성에 대하여 예를 들어보면, 게시판 id가 26인 사내기밀 게시판과 id가 36인 고객Q&A 게시판이 존재한다고 가정할 때 공격자는 Q&A게시판을 접근하여 취득한 URL b_list?tab=36에서 tab 파라미터 값을 26으로 변조하면 내부정보가 저장된 사내기밀 게시판에 접근할 수 있다. Marco Balduzzi는 PAPAS를 통하여 5,000개 이상의 유명사이트를 점검한 결과 약 30%의 사이트에서 취약한 파라미터가 존재하고 그 중 적어도 14%가 파라미터 변조공격에 취약한 것으로 나타났으며, 시만텍, 구글, VM-ware및 Microsoft 등의 잘 알려진 전문 웹 사이트 조차 파라미터 변조공격의 취약점을 가진 것으로 조사되었다[3]. 이러한 취약점은 CWE이나 OWASP에서도 변조공격의 심각성을 강조하며 방지기법에 대한 활발한 연구가 진행되고 있다[4, 5].

실제 알려진 사례로 2014.3월 적발한 KT 홈페이지 고객정보 유출 사건은 URL에 포함되어 있는 고객고유번호를 무작위로 생성하여 KT서버에 요청한 뒤 정상적으로 반환되는 고객정보를 수집하는 수법으로 1200만명의 개인정보를 유출하였으며[6], 2011.6월 발생한 CITIBANK 해킹사례에서 공격자는 고객고유번호가 URL에 포함되었음을 간파하고 URL내의 고객고유번호를 임의의 번호로 변경함으로써 제3의 고객 계좌를 조회하는 등 20여만건 이상의 신용카드번호를 비롯한 개인정보를 유출하는 사고가 발생하였다[7]. 이러한 악의적 유출공격 이외에도 파라미터를 통한 개인정보 유출로 판단할 수 있는 A6.민감 데이터 노출[2]의 경우도 암호화하지 않는 파라미터가 URL에 포함되면서 노출되는 경우가 많다. Andrew G. West는 공개 및 비공개포럼 이용자들에 의해 배포된 8.9억개의 서버전송 URL을 분석한 결과 170만개 이상의 URL에서 e-mail주소 등의 개인정보를 포함하고 있었으며 심지어 사용자 계정의 ID와 비밀번호가 평문 상태로 유출된 것을 확인하였다[8].

본 논문에서는 파라미터 변조방지를 위한 선행연구를 분석하고 각 연구의 단점을 효과적으로 보완할 수 있는 OTACUS를 제안한다. OTACUS는 보안이 필요한 URL이나 파라미터에 대하여 1회용 token을 부여하고 해당 token을 가진 link의 요청이 들어오는 경우 token에 대한 전체URL을 서버 측에서 재구성함으로써 사용자에게 실제 URL이나 파라미터를 숨기는 기법으로 파라미터 변조를 원천적으로 차단한다. 또한 부가기능으로 link별 접근제어 기능을 부여하여 기간이나 횟수에 따라 접근을 제한을 하거나 별도 인증절차를 거치도록 하는 등 접근가능여부를 제어할 수 있는 다목적 접근제어 솔루션으로 구성되어 있다.

본 논문은 다음과 같은 구성을 가진다. 2장에서는 파라미터변조를 이용한 공격을 분류하고 3장에서는 기존 방지기법의 장단점을 분석한다. 4장에서는 OTACUS 를 제안하면서 기존 방지기법과 제안기법을 비교분석하여 OTACUS의 안정성을 검증한다. 5장에서는 실제로 구현된 OTACUS를 통해 효율성을 검증하여 적용 가능한 수준인지를 보여주며 마지막으로 6장에서는 결론을 짓는다.

2. 파라미터변조공격

파라미터 변조 공격은 사용자의 자격 및 권한을 상승시키거나 전자상거래에서의 가격, 수량 등 주문 정보 등의 데이터를 조작하기 위해 클라이언트와 서버 간에 교환 되는 파라미터의 조작을 기반으로 하는 공격으로 논리취약점의 대표적인 공격방법이다[4]. 서버와 클라이언트간의 파라미터 전송은 GET방식과 POST방식으로 서버에 전달되는데 전송되는 파라미터는 이름/값의 쌍으로 구성된 하나 이상의 필드들로 구성되며 클라이언트에서 전송된 파라미터는 서버에 파라미터 배열형태로 전달되어 애플리케이션의 논리적 처리를 진행하게 된다. 이러

한 구조를 이용하여 다양한 형태의 파라미터변조공격이 존재한다.

GET방식

domain.com/proc?N_1=V_1&N_2=V_2…&N_n=V_n

POST방식

<form name=list action=proc>
<input type=hidden name=N_1 value=V_1 >
<input type=text name=N_2 value=V_2 > …
<input type=radio name=N_n value=V_n >
</form>

서버에서의 파라미터

P = {(N_1,V_1), (N_2,V_2),...,(N_n,V_n)}

2.1 FORM기반 파라미터변조공격

변조공격을 이해하기 위해 전자상거래 사이트를 가정하자. 해외구매대행업체인 buynote.com은 S노트북(상품코드 S1)을 $500, T노트북(상품코드 S2)을 $300에 판매하고 있으며 주문에 대한 배송비 $50와 구매대행수수료 $50를 별도로 부담하여야 한다. 단골고객에게는 구매대행수수료를 50%할인하여 받고 있으며, 고객홍보용 $10 할인쿠폰을 발행하였다. 이 쇼핑몰에서 S노트북, T노트북 각 1대를 구입하는 일반고객의 구매 처리절차는 다음과 같다.

1) 구매상품 및 수량 선택 P = {s1=1, s2=1}
2) 부대비용 합산 및 배송지 입력
   P={s=1,s2=1,deli=50,fee=50,addr=Seoul}
3) 구매내용확인 및 최종결제 요청

여기서 악의적인 공격자가 1)수량선택 form내용을 s1=1, s2=-1로 변조하는 경우 서버 측 구매비용 계산단계에서 s2상품의 금액이 $300에서 -$300으로 변경되어 실제 총금액을 $800 에서 $200으로 변경할 수 있다. 또한 2)부대비용 합산단계에서 가산되는 배송료 deli=50을 deli=-100으로 변조하면 총금액은 $100로 변경된다**(기본파라미터 변조공격)**. 구매대행수수료 할인정책은 단골고객에게만 적용되는 것으로 일반사용자에게는 적용되지 않는다. 이를 위해 단골고객의 구매인 경우 숨겨진 파라미터값으로 vip=y 값을 추가하여 적용하는데, 공격자는 임의로 해당 파라미터를 추가하여 할인혜택을 받을 수 있다**(비인가 파라미터추가공격)**. 웹 애플리케이션의 프로그래밍 언어에 따라 전달되는 파라미터명의 중복처리 방법이 표1과 같은 차이가 있는데[3], 만약 소유한 할인쿠폰 적용을 위한 파라미터가 cup_id=a1일 때 cup_id=a1, cup_id=a1 으로 반복하여 전송하는 경우 서버의 처리방법에 따라 중복하여 할인 될 수 있다**(중복파라미터공격)**.

(표 1) 중복된 파라미터명 처리 우선순위
(Table 1) Parameter Precedence of duplicate name

Technology	Parameter Precedence
ASP/IIS	All (comma delimited string)
PHP/APACHE	Last
JSP/Tomcat	First
Perl/Apache	First
Python/Apache	All(List)
Oracle Webtoolkit / Apache	All (comma delimited string)

2.2 링크기반 파라미터변조공격

링크기반 파라미터 변조공격을 이해하기 위해 가상의 H기업 사이트를 가정한다. 해당 사이트 주소는 comph.com으로 사내업무와 대고객 서비스업무를 하나의 도메인에서 처리하고 있으며 게시판 id가 26인 사내기밀게시판과 id가 36인 고객Q&A게시판을 운영한다. 사원의 로그인 페이지는 VPN으로 구성하였으며 접속주소 vpn.comph.com은 사원에게만 공개하였다. 여기서 공격자A는 먼저 Q&A 게시판에 접근하여 URL b_list? tab=36를 취득하여 tab파라미터 값을 36에서 임의로 대입 해보니 26으로 변조할 때 사내기밀게시판에 접근함을 알아냈다. 게시물 조회 URL b_view?seq=3490 에서 b_view?seq=2150으로 파라미터를 변조하여 요청함으로써 핵심 사업정보가 등록된 게시물에 접근하였다**(기본파라미터변조공격)**. 심지어 게시물 수정 URL인 b_edit?seq=2150 으로 접근하여 해당게시물의 수정 또는 삭제 처리까지 가능하다**(접근경로 변조공격)**. 또한 고객공지사항의 게시물 등록정보를 통하여 사원ID를 취득하고 보편적으로 예상되는 VPN 접근 경로인 vpn.comph.com을 통하여 사내망에 접속할 수 있다**(예측가능한 접근경로)**.

3. 파라미터변조 방지기법

위와 같은 다양한 파라미터변조공격에 대응하는 방지기법은 크게 post방식과 get방식 파라미터변조 방지기법으로 구분할 수 있으며 방지기법의 핵심은 서버로부터 전송받은 form이나 파라미터영역의 구조를 변조할 수 없

도록 하는 것으로 공격방식에 따라 차이를 보인다.

링크기반 변조공격에 대응하는 get방식 파라미터 변조 방지기법에는 URL에 포함된 파라미터영역(querystring)을 암호화 하는 방법과 해당 파라미터에 대한 해시를 추가하는 방법, 해당파라미터에 대한 token을 발급하는 방법[9]까지 다양하다. form기반 변조공격에 대응하는 post방식 파라미터 변조 방지기법의 핵심은 서버에서 전송할 form의 구조에 대한 token을 발급하고 해당 token을 form에 숨겨진 필드로 추가하여 클라이언트에서 여러 선택값을 포함한 form을 전송 받을 때 해당 token을 비교 검증하여 값의 변조여부와 허용된 입력범위 이내 여부를 검증한다.

3.1 TamperProof

post방식에 대응하는 대표적인 방지기법[10]으로 서버와 클라이언트 사이 proxy서버형태로 존재하여 클라이언트로 전송되는 모든 form을 가로채어 각 필드의 명칭, 순서, 필드별 입력값의 허용범위와 숨겨진 필드의 내용을 DB에 기록하고 그에 해당하는 임의의 patchID를 부여한 뒤 해당 patchID를 form에 추가하여 클라이언트에 전달한다. 검증단계는 사용자의 입력값을 javascript에서 검증 후 서버로 전송되는 시점에서 form에 포함된 patchID의 구성을 확인하여 저장내용과 일치하는 경우에만 정상 입력값으로 인정하여 실제 운영서버로 전송하는 기법으로 구성도는 그림1과 같다.

patchID의 생성절차는 서버로부터 생성된 HTML에서 form을 추출하여 field의 control 형태별로 고정값 및 제약범위을 설정[11]하고 각 필드별 제약범위를 추출, patches 배열에 추가한다. 예제에서의 노트북 구매절차form에 대하여 제약범위를 추출하면 다음과 같다.

```
s1 > 0 ∧ s2 > 0
deli = 50
(vip=Y, null)
fee ∈ (50 | 25)
```

그 뒤 전체 제약조건에 대한 전체 patchID에 해당하는 key를 생성하여 form에 추가하는데 생성절차는 다음과 같다.

```
Algorithm1 TamperProof-To-User(html)
1: html := add-patchids(html)
2: fork(analyzeClient, html)
3: return html
```

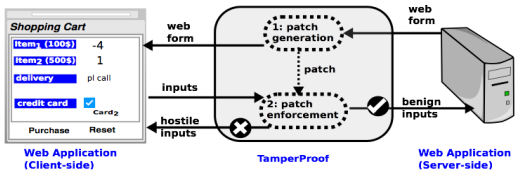
```
Algorithm2 AnalyzeClient(html)
1: for all forms f in html do
2:    js := javascript-signature(html)
3:    id := find-patchID (html)
4:    〈url, fields, constraints〉:= codeAnalysis(html, js)
5:    patches[id] = 〈hurl, fields, constraints〉
```

사용자가 form의 내용을 완성하고 서버로 전송할 때 TamperProof의 검증절차는 다음과 같다.

```
Algorithm3 TamperProof-from-User(request)

1: if request.url ∉ Entries then
2: id := request.data['patchID']
3: if id ∉ patches.keys() then return error
4: wait until patches[id] is non-empty
5: 〈url, fields, constraints〉:= patches[id]
6: if request.data.keys() ∉ fields then return error
7: if !sat(request.data,constraints) then return error
8:    patches.delete(id)
9: return request
```

TamperProof는 form기반 파라미터 변조방지기법으로 proxy서버를 사용하여 기존 애플리케이션의 별도 수정이 필요없도록 구성하였으며 입력값 검증절차를 통하여 논리적 오류취약점을 최소화 하고 있다. 또한 whitelist를 적용하여 patchID를 적용하여야 하는 URL인 경우 patchID가 없으면 접근을 차단함으로써 post-get 방식 전환공격도 효과적으로 차단하고 있다. 다만, field에 대한 입력값 제약범위 설정단계에서 개발자의 의사결정없이 애플리케이션의 논리적 판단을 자동으로 처리할 수 없다. 예를 들어 s1값이 수량을 의미하는 경우 s1>0 이 true가 되지만, ±5개이내의 증감수량을 입력받는 경우 5≧s1≧-5 의 수치 입력도 필요하기 때문이다. 또한 hidden field의 경우 field명과 부여된 값이 그대로 노출되어 공격자는 애플리케이션의 구조를 쉽게 분석할 수 있으며 노출된 field를 분석하여 form이 없는 get방식의 link를 호출하는 경우 중요정보가 노출되는 취약점이 있다. 예컨대 주문단계에서 노출된 cust_id 나 cart_id 로 주문내역을 표시하는 URL을 호출하면 타인의 고객정보나 주문내역이 유출될 수 있다.



(그림 1) TamperProof 구성도
(Figure 1) Overview of TamperProof

3.2 인증토큰 추가기법

get기반파라미터공격의 가장 단순한 방지기법으로 querystring영역을 해시함수로 메세지다이제스트하여 생성한 token을 기존 정보에 추가하여 전송하고, 사용자의 link 요청시 파라미터에 대한 해시 값을 비교하여 변조여부를 검증한다[12]. 최근 취약한 MD5나 SHA-1에서 MACJER-320 해시기법을 사용하여 보다 빠르게 처리하는 기법이 발표되었으나[13], 파라미터 내용이 전부 노출되어 접근경로변경공격에 취약하다.

```
board_view?tab=26&seq=2150
digest = HASH(querystring, secret salt key)
→ board_view?tab=26&seq=2150&digest=BT9fAJh9x2
```

3.3 Link-e-Param

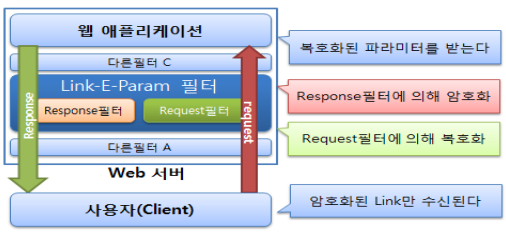
Link-e-Param 파라미터영역 암호화 기법은 get기반 파라미터공격에 대응하는 기법으로 URL호출에서 구분자 ? 를 기준으로 접근경로와 querystring으로 나뉘는 HTTP의 구조를 활용하여 querystring 부분을 전체 암호화 하는 방식으로 변조공격을 차단한다.

다음과 같이 파라미터영역이 모두 암호화되므로 공격자는 파라미터명 tab와 seq의 존재여부를 인지할 수 없어 변조공격을 할 수 없다[14].

```
board_view?tab=26&seq=2150
==> board_view?__E_PARAM=uyGs60sdtqnlSsWm3PGmSle6Q
```

이러한 암호화처리는 servlet필터방식으로 처리되므로 기존 소스의 별도 수정없이 반영 할 수 있도록 구성되어 있다.

그러나 get방식의 link에 대해서만 암호화 처리되므로 form형태로 전달되는 파라미터는 차단할 수 없다. 또한 동일한 파라미터에 대하여 다수의 페이지에 반영하는 접근경로 변경공격에 취약하다. 예를 들어 게시판관련 페이지의 구성이 다음과 같이 구분된다고 할 때 읽기 권한만 부여받은 공격자가 board_view 페이지에서 얻은 암호화된 파라미터를 board_input 파일에 전송하는 경우 수정권한을 얻을 수 있다.



(그림 2) Link-E-Param 구성도
(Figure 2) Overview of TamperProof

```
board_view   게시물 읽기
board_input  게시물 등록/수정

board_view?__E_PARAM=uyGs60sdn8myZ.fZVRjdKtqnlSsWm3PGmSle6Q
    ↓
board_input?__E_PARAM=uyGs60sdn8myZ.fZVRjdKtqnlSsWm3PGmSle6Q
```

또한 암호화된 영역에 대하여 복호화 되므로 임의로 추가되는 파라미터는 무방비로 노출되어 있다. 다음을 보면 암호화된 파라미터에 &seq=2200 추가하는 경우 복호화 이후에도 추가된 파라미터가 그대로 남게된다.

```
board_view?__E_PARAM=uyGs60sdn8myZ.fZVRjdKtqnlSsWm3PGmSle6
Q&seq=2200
→ board_view?tab=26&seq=2150&seq=2200
```

이렇게 변환된 URL은 파라미터 중복처리 우선순위에 따라 권한없는 seq=2200의 파라미터가 적용되는 취약점에 노출되어 있다. MS에서 공개한 암호화된 token기반 인증기법에서는 Link-e-Param과 같이 querystring 영역을 암호화 하는데, 암호화 과정에서 Session-id와 timeout를 추가함으로써 동일파라미터에 동일한 암호문이 나오는 단점을 보완하였으나[15], 접근경로는 여전히 그대로 노출되고 파라미터 추가공격에 대한 취약점이 존재한다. 또한 예제에서의 VPN로그인 페이지와 같이 파라미터를 포함하지 않는 URL의 경우 별도의 접근제어를 두지 않으면 아무런 제재 없이 해당페이지를 접근 할 수 있다.

3.4 SecretURL 기법

get방식 파라미터변조 방지기법으로 파일공유서비스(File Sharering Services, FSS), SNS 등에서 컨텐츠를 공유하거나 e-mail을 통한 사용자 인증 등에서 주로 사용되며 FSS에서 접근하는 컨텐츠의 URL에 대한 고유의 token을 발급하고 생성된 token을 URL로 대체하거나 포함하여 제공함으로써 실제 컨텐츠의 접근경로를 숨길 수 있다[9]. 사용자인증의 경우 인증에 필요한URL에 대하여 발급한 token을 포함하는 URL을 e-mail로 전송하고 해당 e-mail의 열람여부를 검증함으로써 해당 token의 소유자는 e-mail계정의 사용 권한이 있는 것으로 판단 할 수 있으며, 이 방식은 다채널 인증으로 많이 사용되고 있다.

그러나 이 기법은 컨텐츠 공유를 위한 고정값 암호키를 사용함으로써 token의 유효기간에 대한 제한이 없어 열거형공격에 취약할 수 있으며 실제로 Nick Nikiforakis는 FSS에 대한 접근link를 검증 한 결과 많은 FSS에서 개인정보를 포함하는 민감한 정보들이 노출됨을 증명하였다[16]. Secret URL의 유사개념으로 IIS 등 web server의 rewrite모듈을 이용한 Canary URLs 솔루션은 개인화된 URL을 제공하며 그 구조는 아래와 같다[17].

```
domain.com/{token}/list.jsp?param=value
```

token으로는 세션값을 사용하는데, 이 기법은 크로스 도메인 XDR공격[18]을 효과적으로 방어할 수 있으나 동일세션에서의 파라미터 변조공격은 방어할 수 없다.

4. OTACUS

위에서 설명한 방지기법의 취약점은 근본적으로 URL의 경로가 공개되어 있다는 것에 기인한다. 이러한 취약점은 URL을 숨김으로써 해결할 수 있으며 OTACUS에서는 효과적으로 URL을 숨기기 위해 Secret URL기법을 사용한다. 사용자는 전체URL과 전혀 유사성 없이 발급된 token으로 구성된 URL을 호출하면 OTACUS는 URL DB에서 요청token에 해당하는 실제URL을 찾아 접근권한을 확인하고 파라미터를 정리하여 해당URL에 전달 처리한다. 이 때 사용되는 token은 1회사용을 위한 token과 장기간/외부 공유용 token으로 나뉘는데, 1회사용을 위한 token은 가독성이 무의미하므로 해시함수를 사용하고 장기간 사용하는 token의 경우 가독성이나 공유 효율성을 위해 12byte 이내의 단축URL로 구성한다.

4.1 URL 생성

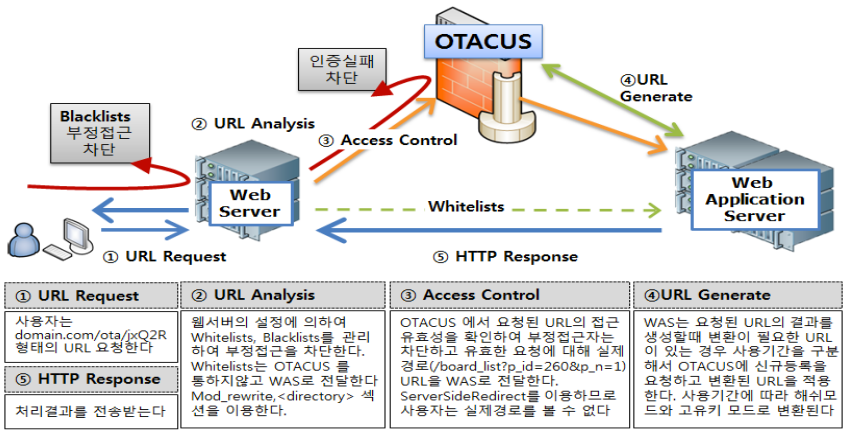
OTACUS는 전체URL 및 접근제어 요건을 DB에 기록하고 이에 대응하는 token을 생성하는데 운영모드에 따라 다음과 같이 생성한다. 1회용 URL로 사용될 token은 SHA-1 해시함수를 이용하며, 공식은 다음과 같다.

```
token = Hash( url + sessionid + timestamp, salt )
```

생성된 token은 해시함수 자체의 충돌저항성으로 인하여 고유한 Key로 사용할 수 있다. 공유용 URL은 20byte 이내의 62가지 문자의 조합으로 이루어 지며 열거형 공격에 대응하기 위하여 SHRT URL의 생성방식[19]을 응용하여 회원ID나 IP주소를 활용한 예약어, 요청페이지에 따른 1차 인덱스, timestamp, 요청횟수에 의한 2차 인덱스로 구분하였다. OTACUS를 적용하기 위해 웹 애플리케이션의 코드를 일부 수정하여야 하는데, get방식 link의 수정방법은 다음과 같다.

```
적용전 <a href="list.jsp?id=secret&cnt=100">
수정내용 <a href="<% ota_gen(url=>'list.jsp',
          param=>'id=secret&cnt=100', mode=>'hash') %>">
적용결과 <a href="/ota/380b9f314c48aa7b604e961fcbe174cbe65e3cf1">
```

공유용 단축URL을 생성하는 경우 1회용URL 생성에 추가정보를 적용할 수 있으며 다음과 같다.



(그림 3) OTACUS 구성도
(Figure 3) Overview of OTACUS

```
ota_gen(url=>"list.jsp",
        param=>"id=secret&cnt=100",
        mode=>'share', access_count=>15,
        access_limit_day=>15, access_check=>false,
        access_key=>null)
적용결과 <a href="/ota/xmhY2g5C3j">
```

post방식의 form에 대응하기 위하여 다음과 같은 수정 절차가 필요하며, hidden field 가 있는 경우 해당 내용을 모두 등록하고, 이를 DB에 기록한다. 이 때 개발언어의 post전달방식 특성상 수신 페이지 p의 노출이 불가피하나 해당페이지에 대한 접근제어를 반영하거나 다른 언어로 구현하는 경우 노출을 피할 수 있다.

```
적용전
<form action="board_edit">
  <input type="hiidden" name="seq" value="°2100">
  <input type="hiidden" name="tab" value="26">
수정내용
 ota_gen(url=>'board_edit', mode=>'formGen',
          param=>'seq=2100&tab=26') //hidden field
적용결과
<form action="/ota/lp?e=be174cbe65e3cf1380b9f314c48aa7b604e9">
  <input type="hidden"
      name="be174cbe65e3cf1380b9f314c48aa7b604e9" value="ae76c4de2">
  <input type="hidden"
      name="be174cbe65e3cf1380b9f314c48aa7b604e9" value="1917abee6">
```

4.2 호출에 대한 검증/ 접근제어

먼저 요청된 token에 대하여 DB상에 존재여부를 확인하고 존재하지 않는 경우 열거형 공격과 같은 비정상 접근으로 간주하고 반복 접근시 해당 IP를 차단하여 공격자의 비정상 접근을 방지한다. DB에 존재하는 경우 해당 접근에 대한 접근제어를 하게 되는데, 해시모드인 경우 동일세션인지, 제한시간이내 접근인지 확인하고, 공유모드인 경우 공유횟수, 공유기한, 비밀번호 일치여부등을 검증하여 실제경로와 파라미터 집합을 얻어 이를 포함하여 해당URL로 전달한다. 이 과정에서 웹서버로 유입되는 URL에 대해 whitelists를 제외한 모든 URL을 필터링함으로써 실수로 노출된 경로에 대해서도 접근을 차단할 수 있다. /images, /js 와 같은 정적 컨텐츠나 파라미터 전송이 없는 범용접근경로에 대해서 허용하고 그 외 OTACUS로 접근가능한 /ota만 허용하면 실제 애플리케이션 URL이 노출된다 하더라도 웹서버에서 자동으로 차단되므로 혹시 모를 URL의 노출에 의한 비정상 접근을 효과적으로 차단 할 수 있다.

4.3 OTACUS의 안전성

OTACUS는 외부로 노출된 영역이 인식 가능한 파라미터를 포함하지 않는 1회용 token으로 구성되어 있어 기본변조공격에 대응할 수 있으며, 한번 발급한 token은 세션 유효기간 이내에서 1회만 사용할 수 있어 이미 사용한 token의 재접근을 허용하지 않아 반복공격이나 XDR 공격을 효과적으로 차단한다[18]. 또한 발급한 token에 접근경로까지 포함하고 있어 경로를 바꿔 호출하는 접근경로 변경공격에도 원천적으로 안전하며, 해당token에 파라미터를 추가하여 호출하는 공격에도 token 외 전송되는 정보는 폐기하는 시스템 구조상 추가된 파라미터는 즉시 폐기되므로 파라미터 추가공격에도 안전하다. 또한, 대부분의 요청이 OTACUS를 통과하므로 인증 실패 요청이 일정시간 반복되는 경우 해당IP의 접근을 차단함으로써 임의의 문자대입으로 반복적 접근을 시도하는 열거형 공격도 효과적으로 대처할 수 있다.

5. OTACUS의 효율성

OTACUS는 생성되는 link나 form에 대한 정보를 DB에 기록하고 조회하는 DB Access가 필요하여, 이 과정의 처리 소요시간은 OTACUS 적용의 가장 중요한 결정요소로 작용한다. 시스템 적용을 위한 효율성 검증을 위해 다음과 같은 실험을 실시하였다. 테스트 환경의 서버는 SUN의 Ent.6800으로 900MHz 16코어, Oracle DB와 Web-Logic으로 구성하였으며 개발언어는 OHS, PL- SQL Web-ToolKit, 부하장비는 2.5GHz급의 업무용PC에서 HP LoadRunner ver12를 사용하여 진행하였고, 객관적 평가를 위하여 기존기법 중 유사 실험 결과가 존재하는 Link-E-Param 기법의 결과와 비교하였다.

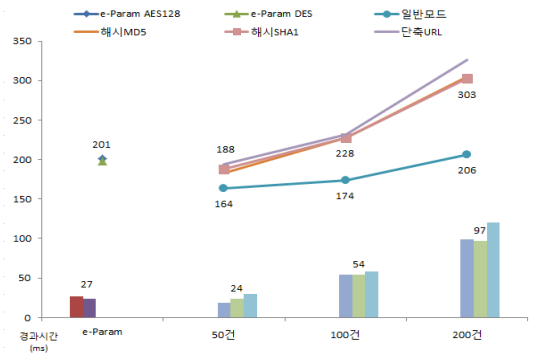
5.1 시나리오 1

이 실험은 OTACUS를 적용 전과 적용 후의 소요시간을 비교하여 단순 token 발급과정의 부하 정도를 확인하는 실험이다. 별도의 정적 컨텐츠 접근을 배제한 각각 50, 100, 200개의 링크를 포함하는 게시판 목록을 표시하는 페이지이며, 해당 페이지를 동시에 10명의 사용자가 매 500회를 요청한 뒤 이에 대한 소요시간을 측정하고 그 평균값을 적용전인 일반모드와 OTACUS의 각 운영모드별 시간 차이를 비교하면 token 발급과정에서 발생하는 부하 정도를 측정할 수 있다. 실험은 일반모드와

(표2) 방지기법의 비교분석
(Table 2) Summary Analysis of prevention techniques

방지기법	대응방식	구현방법	특징	취약점
TamperProof	POST	Proxy server + javascript	form입력값 검증, proxy방식으로 웹 소스 변경 불필요, javascript를통한 서버처리분산효과	GET 방식 미지원, 파라미터명/값 노출, 논리적 판단 부족
인증토큰 추가기법	GET/POST	GATEWAY	빠른 처리속도, GET/POST 지원, 소스코드 수정필요	경로, 파라미터명/값 노출로 인한 접근경로변경 공격
Link-E-Param	GET	fillter	JSP 필터사용으로 소스코드변경 불필요, 다양한 암호화방식지원	POST방식 미지원, 중복파라미터공격, 접근경로변경공격
Secure URL	GET	GATEWAY	link경로에 대응하는 token 발급, 배포에 따른 재사용 허용, 소스변경필요	POST방식 미지원, 장기간 노출된 token으로 인한 열거형공격
OTACUS	GET/POST	GATEWAY	link/form에 대응하는 1회용 token발급, 배포용 token 발급, token별 접근제어 가능, 소스코드 변경필요	-

OTACUS의 해시모드(MD5, SHA1), 단축URL 모드로 구분하여 동일한 환경에서 실시하였으며 결과는 그림4와 같다.



(그림 4) 결과1 운영모드별 소요시간 및 시간차(ms)
(Figure 4) Result1 Elapsed time by operationg mode

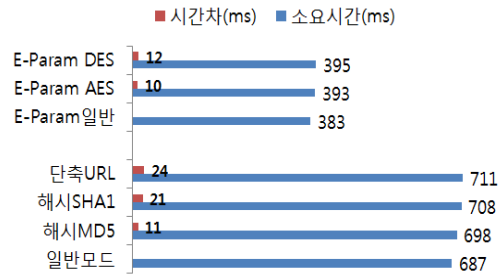
실험결과에서 표시되는 게시물의 건수가 증가할수록 DB의 처리량도 증가되는데, 일반모드에서의 증가폭은 완만한 상승선을 이루지만 OTACUS를 적용하면 더 큰폭으로 상승함을 볼 수 있다. 이는 OTACUS가 token을 발급하고 DB에 기록하는 과정에서 발생하는 시간으로 50건의 URL이 존재하는 페이지에서 해시로 적용하였을 경우 일반모드에 비하여 24ms의 소요시간이 증가되었다.

또한 100건, 200건으로 변환대상 링크의 수가 늘어날수록 54ms, 97ms으로 추가 소요시간도 증가함을 확인 할 수 있다. 이는 URL을 생성하고, DB에 기록하는 절차가 Link-e-Param의 필터 변환절차보다 더 많은 시간이 소요되나, 보편적인 웹 어플리케이션의 단일페이지 기준 링크의 개수가 30~50여개 이내임을 고려할 때 실제 체감할 수 있는 속도의 지연은 아니라고 판단된다. 또한, 필터를 통하여 페이지의 모든 데이터에 대하여 암호화 검증을 하여야 하는 Link-e- Param의 특성상 긴 내용의 페이지에 대한 처리시간까지 감안할 때 OTACUS가 더 효율적이라고 판단 할 수 있다. 또한, 단축 URL방식의 소요시간이 해시방식에 비해 더 큰 폭으로 증가하는데, 이는 짧은 길이의 token을 생성하는 과정에서 기존 token의 중복확인에 필요한 시간으로 일반적인 운영 환경에서는 실험처럼 페이지 내의 모든 링크를 변환하는 것이 아니라 사용자의 요청에 의해 1~2개 이내의 단축URL만을 생성하므로 실제 적용단계에서의 부하는 고려할 필요가 없다.

5.2 시나리오 2

이 실험은 보편적인 게시판 애플리케이션 환경에서 일반 사용자와 동일한 환경에서 OTACUS를 적용하는 경우 token의 생성 및 검증과정에서 발생되는 부하로 인하여 지연되는 소요시간을 확인하는 실험으로 일반적인 게시판 목록에서 표시하는 20개의 링크를 포함하고 있으며

10여개의 이미지파일과 js파일이 포함되어 있다. 이 환경에서 10명의 동시사용자가 200회 이내에서 목록페이지를 요청하고, 요청하는 횟수의 30%내외에서 무작위로 반환된 URL을 통해 본문을 호출함으로써 token의 생성과 접근에 대한 검증, 접근제어 등 OTACUS를 적용한 게시판 애플리케이션 운영 전반에 대한 실험을 실시하였으며 결과는 그림5와 같다.



(그림 5) 결과2 운영모드별 소요시간(ms)
(Fig. 5) Result2 Elapsed time by operationg mode

실험결과 실제 환경에서의 OTACUS 적용은 2~3%이내의 추가 소요시간이 발생함을 알 수 있다. 이는 Link-e-Param방식의 생성단계에서 발생하는 경과시간에 비교하여 큰 차이가 없는 것이며, 검증단계까지 적용된 Link-e-Param기법에 비하여 더 좋은 효율성을 보인 것이다. 이를 통하여 일반 사용 환경에서 OTACUS를 통한 방지기법의 적용이 사용자가 인식 가능한 부하를 유발하지 않으면서 기존 기법의 취약점을 보완하는 효과적인 파라미터변조 방지기능을 수행함을 확인 할 수 있다.

6. 결 론

본 논문에서는 단순한 파라미터 변조공격의 심각성을 인식하고 기존의 파라미터 변조방지기법의 취약점 개선에 목적을 두었다. 최근 KT의 홈페이지 파라미터 조작으로 인한 고객정보 유출 사례에서 볼 수 있듯이 고성능의 다양한 기능을 가진 해킹 툴이 개발되고 있는 현실에서 간단한 파라미터 조작만으로 심각한 정보유출이 발생하고 있으며 이는 웹서비스가 HTTP를 사용하는 구조적인 문제로 현재의 URL구조가 유지되는 한 파라미터 변조공격의 방어가 어렵지만, OTACUS기법으로 token기반의 1회용 접근URL을 사용한다면 해당 공격을 효과적으로 방어할 수 있다. 비록 웹 애플리케이션의 소스 중 일부를 수정하여야 하는 번거로움이 있지만 파라미터를 포함하는 URL노출을 피함으로써 얻을 수 있는 보안효과는 매우 높다고 판단된다.

참 고 문 헌 (Reference)

[1] Xiaowei Li and Yuan Xue. 2014. A survey on server-side approaches to securing web applications. ACM Comput. Surv. 46, 4, Article 54 (March 2014)

[2] OWASP top10 2013 https://www.owasp.org

[3] Balduzzi, Marco, et al. "Automated Discovery of Parameter Pollution Vulnerabilities in Web Applications." NDSS. 2011.

[4] Web Parameter Tampering, https://www.owasp.org/index.php/Web_Parameter_Tampering

[5] CWE-472 External Control of AssumedImmutable Web Parameter, http://cwe.mitre.org/data/definitions/472.html

[6] Administrative penalties for Privacy violations of KT, Korea Communications Commission,2014.6.26

[7] Newyork Times June. 13, 2011. "Thieves Found Citigroup Site an Easy Entry", http://www.nytimes.com/2011/06/14/technology/14security.html

[8] Andrew G. West and Adam J. Aviv, "On the Privacy Concerns of URL Query Strings." In W2SP'14: Proceedings of the 8th Workshop on Web 2.0 Security and Privacy. San Jose, CA, USA. May 2014

[9] Amazon, System and method for providing secureURL based access to private resources, US 6360254 B1

[10] Nazari Skrupsky, Prithvi Bisht, Timothy Hinrichs, V. N. Venkatakrishnan, and Lenore Zuck. 2013. "TamperProof: a server-agnosticdefense for parameter tampering attacks on web applications." In Proceedings of the third ACM conference on Data and application security and privacy (CODASPY '13). ACM, New York, NY, USA, 129-140.

[11] Bisht, P., Hinrichs, T., Skrupsky, N.,Bobrowicz, R., and Venkatakrishnan, V. "NoTamper: Automatic Blackbox Detection of Parameter Tampering Opportunities in Web Applications." In CCS'10: Proceedings of the 17th ACM Conference on Computer and Communications Security (Chicago, IL, USA, 2010).

[12] Scott Mitchell "Passing Tamper-Proof QueryString Parameters" http://www.4guysfromrolla.com/articles/083105-1.aspx

[13] Asish Kumar Dalai, Saroj Kumar Panigrahy, Sanjay Kumar Jena, A Novel Approach for Message Authentication to Prevent Parameter Tampering Attackin Web Applications, Procedia Engineering, Volume 38, 2012, Pages 1495-1500, ISSN 1877-7058

[14] Deok-Byung Lim and JunCheol Park, "Link-E-Param: A URL Parameter Encryption Technique for Improving Web Application Security", J-KICS, 11-09 Vol.36 No.9

[15] Microsoft Corporation, "Token-based authentication using middle tier" US 20120084561 A1

[16] Nick Nikiforakis, OWASP AppSecDev Research 2010 - On the privacy of file sharing services

[17] Brayn Sullivan, MSDN Magazine 2009 March, Protect Your Site With URL Rewriting

[18] SangHo Lee, YoungJae Maeng, DaeHun Nyang and KyungHee Lee "Possibility of Disclosure of User Information in Internet Explorer", J-KICS '13-12 Vol.38 No.12

[19] Soojin Yoon, Jeongeun Park, Changkuk Choi and Seungjoo Kim, "SHRT : New Method of URL Shortening including Relative Word of Target URL ", J-KICS 13-05 Vol.38 No.6B

[20] Dafydd Stuttard and Marcus Pinto, "The Web application Hacker's Handbook: Discovering and Exploiting security flaws" 2008.11.13

◐ 저 자 소 개 ◑

김 귀 석 (Guiseok Kim)
2008년 2월 한국사이버대학교 컴퓨터정보통신학과 학사
2013년 3월~현재 고려대학교 정보보호대학원 석사과정
1992년 1월~현재 KB국민은행 인재개발부 차장
관심분야: Web Application, e-Learning, Usable Security
E-mail : kgssks@gmail.com

김 승 주 (Seungjoo Kim)
1994년~1999년: 성균관대학교 정보공학과 (학사, 석사, 박사)
1998년 12월~2004년 2월: KISA(舊한국정보보호진흥원) 팀장
2002년~현재: 한국정보통신기술협회(TTA) IT 국제표준화전문가
2004년 3월~2011년 2월: 성균관대학교 정보통신공학부 조교수, 부교수
2011년 3월~현재: 고려대학교 사이버국방학과/정보보호대학원 정교수
2004년~현재: 한국정보보호학회 이사
2005년~2006년: 교육인적자원부 유해정보 차단 자문위원
2007년: 국가정보원장 국가사이버안전업무 유공자 표창
2007년~2009년: 전자 정부 서비스 보안 위원회 사이버 침해사고대응 실무위원회 위원
2010년: 방송통신위원회 정보통신망 침해사고 민관합동조사단 위원
2012년 3월~2012년 6월: 선관위 디도스 특별검사팀 자문위원
2013년 4월~2013년 12월: IT보안인증사무국 자문위원
2013년 9월~현재:중앙선거관리위원회 자문위원
2014년 3월~현재: 헌법재판소 자문위원
관심분야: 보안공학, 암호이론, 정보보증, 정보보호제품 보안성 평가, IoT 보안, HCI Security, Cyber-Physical Security
E-mail : skim71@korea.ac.kr