Web Application Vulnerabilities and Static Detection of XSS Vulnerabilities

SELAB
이길주
2009.02.04

Reference

- G. Wassermann, Z. Su, “Static Detection of Cross-Site Scripting Vulnerabilities”, Proc. of the 30th international conference on Software engineering, 2008
- Y. Huang, F. Yu, C. Hang, C. Tsai, D. Lee, S. Kuo, “Securing Web Application Code by Static Analysis and Runtime Protection”, Proc. of the 13th international conference on World Wide Web, 2004
- A. Klein, “DOM Based Cross Site Scripting or XSS of the Third Kind: A look at an overlooked flavor of XSS”, Web Application Security Consortium, 2005
- Wikipedia : Cross-site scripting http://en.wikipedia.org/wiki/Cross-site_scripting
- ㈜안철수연구소 보안컨설팅사업부, “웹 애플리케이션 개발 보안 코딩 가이드”, ㈜안철수연구소, 2008

웹 애플리케이션의 취약점

- 가장 흔하게 악용되는 취약점
  - script injection
- Cross-Site Scripting(XSS)
- SQL Injection

Cross-Site Scripting(XSS)

- 웹 애플리케이션 취약점의 가장 흔한 형태
- 화면 처리를 위한 입력값을 검증하지 않고 그대로 사용할 경우 나타난다

```
$month=$_GET['month']; $year=$_GET['year'];
$day=$_GET['day'];
echo "<a href=\"day.php?year=$year&amp;"
echo "month=$month&amp;day=$day\">";
```

An XSS vulnerability found in *SquirrelMail*.

```
<a href= "day.php?year=><script>malicious_script();</script>
```

Compromised HTML output.

XSS 취약점 공격 형태

- Type 1: Non-Persistent(Reflected) attack
  사용자가 제공한 정보를 즉시 받아온다
- 공격 시나리오

1. 김군은 박군이 운영하는 특정 웹사이트를 종종 방문한다. 이 때 아이디/비밀번호 및 결제 관련 민감한 정보들을 기록으로 남긴다.
2. 공격자는 박군의 웹에서 XSS type 1 취약점이 있는지 확인한다.
3. 공격자는 김군에게 이메일 등으로 클릭할 수 있는 URL을 만들어 보낸다.
4. 김군은 박군의 웹사이트에 로긴한 상태로 공격자의 URL 방문한다.
5. URL에 포함된 악성 스크립트가 김군의 브라우저에서 마치 박군 서버에서 온 것처럼 꾸며서 실행된다. 이 스크립트는 김군의 세션 쿠키 등을 공격자에게 발송할 수 있다. 이를 통해 공격자는 김군의 민감한 정보들을 훔친다.

XSS 취약점 공격 형태

- Type 2: Persistent(Stored) attack
  사용자가 제공한 정보를 수집 저장하여 적당한 시기에 악용한다
- 공격 시나리오

1. 박군은 웹사이트를 운영하는데 다른 멤버들이 볼 수 있는 메시지 및 컨텐츠를 포스팅하는 기능을 제공하고 있다.
2. 공격자는 박군의 웹사이트가 Type 2에 취약한지 확인한다.
3. 공격자는 사이트의 되도록 많은 유저들이 보도록 이슈가 될만한 내용을 포스팅한다.
4. 단순히 해당 메시지를 보기만 하면 당사자 모르게 유저의 사이트 세션 쿠키나 기타 신용정보가 습득되어 공격자의 웹서버로 발송된다.
5. 나중에, 공격자는 수집된 세션 등의 정보를 악용할 수 있다.

XSS 취약점 공격 형태

- Type 3: DOM-based attack
  HTML 보다는 DOM을 기반으로 하는 Javascript 등이 사용되는 사이트를 대상으로 한다. 페이지의 요소들(로그인 타이틀 등)을 조작하거나 추가하여 사용자의 정보를 취득할 수 있다.
- 공격 시나리오

1. 공격자는 이메일 등으로 악성 웹페이지 URL을 김군에게 보낸다.
2. 김군이 링크를 클릭한다.
3. 악성 페이지의 자바스크립트가 김군의 로컬 컴퓨터에서 type 3에 취약한 HTML을 연다.
4. 취약한 HTML 페이지는 김군의 로컬 컴퓨터에서 실행되는 자바스크립트를 포함한다.
5. 공격자의 악성 스크립트는 이제 김군의 컴퓨터 권한을 취득하여 주어진 명령을 실행한다.

SQL Injection

- 데이터베이스에서 쿼리를 이용하여 데이터를 처리할 때 프로그램에서 쿼리를 조합하여 요청되도록 만든 사이트에서 발생
- DB와 연결되어 있는 입력값에 임의의 SQL문을 삽입하여 개발자가 의도한 바와 달리 오작동하게 하는 공격
- 인증 우회, 데이터 획득, DBMS의 시스템 명령어 실행을 수행하는 등의 정보 권한을 획득

SQL injection 발전 경향

- Automated SQL Injection – 수백가지 조합을 수동으로 DB정보를 빼지 않고, 특정 DB에 맞추어서 특정 쿼리들을 미리 준비해 놓은 이후 클릭 한번으로 진행
- Mass SQL Injection – 앞의 공격 대상이 단일 사이트가 아니라 수십개 이상의 사이트에 대해 일시적으로 발생

XSS 취약점 탐지 방향

- Unchecked untrusted data
- Insufficiently-checked untrusted data
- An adapted string analysis to track untrusted substring values
- A check for un trusted scripts based on formal language techniques

기존의 탐지 접근

- Testing 또는 Static Taint Analysis
- 자동화된 테스팅은 입력 인증 코드에서 에러를 찾는데 잘 안맞음. 대부분의 악성 입력을 찾아낸다 해도 특정 부분의 취약점을 파고드는 시도가 반드시 나타나기 때문
- 오염 분석은 오염을 제거하기 위해 지정된 함수 목록을 입력받지만, 이들로 분석이 수행되지 않음. 따라서 약한 입력 유효성에 의한 에러를 잡을 수 없을 것

Main contributions

- Weak input validation으로 인한 XSS 취약점 탐색 접근
- untrusted script에 대해 생성된 HTML 문서를 체크하는 레이아웃 엔진의 행동에 기반한 알고리즘
- Evaluates the approach on several real-world PHP web application

알고리즘 분석

- String-taint Analysis

1. echo
ass
2.
3.
dat
4.
5.

$$
\begin{aligned}
REQUESTmodule^{T} &\rightarrow \Diamond \\
data1 &\rightarrow REQUESTmodule \\
data2 &\rightarrow data1; \\
module &\rightarrow data2 \\
output1 &\rightarrow \texttt{
} \\
hiddenfields &\rightarrow \texttt{<input value='}module\texttt{'/>} \\
output2 &\rightarrow \texttt{</div>}\ hiddenfields \\
output3 &\rightarrow output1 \mid output2
\end{aligned}
$$

d

한다.

Figure 5: CFG with untrusted substrings summarized; see Section 3.2.4 for an explanation of '◇.'

```
555        return $data;
556    }
```

알고리즘 분석

- Preventing Untrusted Script
- 어떤 untrusted input도 브라우저의 자바스크립트 해석기를 불러오지 못하도록 하자
- W3C 권고문을 참조하여 HTML에 스크립트 포함하는 방식 등을 연구

검증

- Minamide's PHP string analyzer 를 확장하여 구현
- 실제 대규모의 웹 애플리케이션에서 얼마나 잘 체크할 수 있는가?
- 수동 입력 유효성 코드에 대해 얼마나 잘 체크할 수 있는가?

결론

- 가장 널리 퍼진 웹 애플리케이션 취약점
  - XSS, SQL Injection
- Taint analysis – Untrusted input가 client의 자바스크립트 interpreter를 부르는지 여부를 체크
- 큰 규모의 코드에도 적용 가능
- 수동으로 쓰여진 입력 유효성 루틴을 이용해 알려지지 않은 XSS 취약점도 탐지 가능