오픈 소스 취약점 분석과 대응 방안

유승민
고려대학교 컴퓨터정보통신 대학원
소프트웨어 보안 학과
e-mail : ryusm@korea.ac.kr

Open Source Vulnerabilities Analysis and Countermeasures

Seung-Min Ryu
Software Security,
Korea University Graduate School of Computer & Information Technology

요 약

오픈소스 활용이 증가함에 따라 같이 증가하는 보안 위험성에 대한 문제점을 제시하고자 한다. 오픈 소스 활용의 생산성 향상과 비용 절감 대비 보안 취약점이 따르는 문제를 분석 하고자 한다. 본 논문에서는 널리 알려진 오픈소스의 취약점과 이를 해결할 방법에 대해서 소개하고자 한다. 오픈소스의 취약점 공격 방법과 해결 방안을 제시하고 이를 해결할 분석 도구를 소개하는 것을 목표로 한다.

1. 서론

2017 Coverity Scan Report에 따르면 현재까지 4600가지의 Open Source Software(이하 OSS) 프로젝트를 진행했고, 총 60만 개 이상의 결함과 함께 누적으로 110만 개의 결함을 확인 했다고 한다.

앞으로도 접근이 용이하고 비용 절감에 기여할 수 있는 OSS 활용은 더욱 커질 것으로 전망된다. OSS 취약점 분석이 중요한 이유는 복제, 수정, 배포가 자유롭기 때문에 취약점도 함께 전파되는 것이다. 따라서 단일 프로젝트보다 더 큰 위협에 노출되어 있음을 의미한다.

따라서 지속적인 보안 업데이트와 OSS 뿐만 아니라 업데이트 접근이 어려운 경우 수동으로 조치할 수 있는 방법까지 안내하는 것이 바람직하다.

2절에서는 알려진 OSS 취약점에 대한 사례를 설명하고 3절에서는 2절에서 언급한 취약점에 대한 공격 방법, 4절에서는 3절에서 소개한 취약점을 어떤 분석도구로 대처하는 것이 효과적일지 제안하려고 한다.

2. OSS 취약점 사례

2-1. 워드프레스

WordPress는 널리 알려진 오픈 소스 저작물 관리 시스템이다. WordPress 기반 Website는 전세계 점유율 30%를 넘는다.[1] 한국에서는 서울특별시 홈페이지가 활용될 정도로 한국에서도 잘 알려져 있다.

그림1은 WordPress의 취약점을 공유하는 취약점 DB WebSite에서 최근까지의 취약점을 보여주고 있다.[2] OWASP Top 10에서 정의하고 있는 취약점 리스트를 포함한 최근 7개의 리스트 중에는 Cross-Site Scripting, Authenticated 관련 취약점, PHP Object Injection 취약점을 포함하고 있다.[3][4]

Latest WordPress Vulnerabilities

2019-03-13	WordPress 3.9-5.1 - Comment Cross-Site Scripting (XSS)
2019-02-19	WordPress 3.7-5.0 (except 4.9.9) - Authenticated Code Execution
2018-12-13	WordPress <= 5.0 - Authenticated File Delete
2018-12-13	WordPress <= 5.0 - Authenticated Post Type Bypass
2018-12-13	WordPress <= 5.0 - PHP Object Injection via Meta Data
2018-12-13	WordPress <= 5.0 - Authenticated Cross-Site Scripting (XSS)
2018-12-13	WordPress <= 5.0 - Cross-Site Scripting (XSS) that could affect plugins

그림 1 WordPress Vulnerabilities[1]

또한 OSS에서 사용하는 플러그인에서 취약점이 지속적으로 확인되고 있다. Easy WP SMTP 이란 플로그인에서 Exploit 가능한 문제가 발견되기도 했다.[5]

2-2. Google Chrome

Google 크롬의 V8에서 69.0.3497.81 이전까지 UAF(Use-After-Free)를 유발 한 Javascript 문제로 원격 공격자가 HTML 페이지를 통해 샌드 박스 내부에서 임의의 코드를 실행할 수 있다.

Impact

CVSS v3.0 Severity and Metrics:	CVSS v2.0 Severity and Metrics:
Base Score: 8.8 HIGH	Base Score: 6.8 MEDIUM
Vector: AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H (V3 legend)	Vector: (AV:N/AC:M/Au:N/C:P/I:P/A:P) (V2 legend)
Impact Score: 5.9	Impact Subscore: 6.4
Exploitability Score: 2.8	Exploitability Subscore: 8.6
Attack Vector (AV): Network	Access Vector (AV): Network
Attack Complexity (AC): Low	Access Complexity (AC): Medium
Privileges Required (PR): None	Authentication (AU): None
User Interaction (UI): Required	Confidentiality (C): Partial
Scope (S): Unchanged	Integrity (I): Partial
Confidentiality (C): High	Availability (A): Partial
Integrity (I): High	Additional Information:
Availability (A): High	Victim must voluntarily interact with attack mechanism
	Allows unauthorized disclosure of information
	Allows unauthorized modification
	Allows disruption of service

그림 2 Impact Score[6]

Chrome Browser 같은 경우 점유율이 60%[7]를 초과하고 있기 때문에 하나의 취약점이 발견되더라도 모든 사용자가 영향을 받을 수 있기 때문에 그 문제는 더욱 심각하다.

3. 취약점 분석

2-1 워드프레스에서의 취약점은 Cross-Site Scripting, Authenticated, PHP Object Injection 크게 세 가지로 정리할 수 있다. 2-2 Google Chrome의 경우는 Use-After-Free 이란 Heap 공간에서 Free 하고 Reuse할 때 일어날 수 있는 취약점이다. 본 논문에서는 위 취약점 중 OWASP Top 10 에 지속적으로 언급되는 Cross-Site Scripting과 Injection 취약점에 대한 내용을 서술하고자 한다.

3-1. Cross-Site Scripting(XSS)

XSS 공격은 일반적으로,

1. 사용자가 웹 페이지를 로드하고 악성 코드가 사용자의 쿠키 복사
2. 요청한 페이지를 도난당한 쿠키와 함께 HTTP 요청을 웹 서버로 전송
3. 공격자는 해당 쿠키를 활용하여 사용자를 가장함

프로세스로 이뤄 진다.

아래 그림 3은 공격에 대한 플로우를 보여 준다.

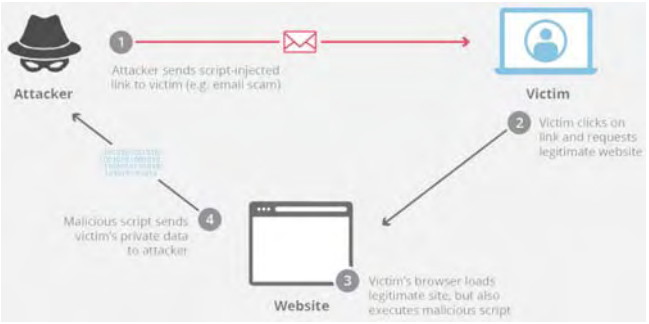

그림 3 XSS Attack[8]

3-2. PHP Object Injection

해당 취약점을 악용하기 위해서는 두 가지 조건이 충족되어야 한다.

1. POP chain을 구현하는데 사용할 수 있는 PHP magic method(_wakeup 이나 _destruct)와 같은 구문을 포함한 클래스가 존재해야함
2. 공격 중에 사용되는 모든 클래스는 객체 자동 로드가 지원되거나 unserialize() 호출 시에 선언 되어야 한다.

아래 예제는 _wakeup Method를 통한 악용 사례를 보여준다.

```
class Example2
{
   private $hook;

   function __construct()
   {
      // some PHP code...
   }

   function __wakeup()
   {
      if (isset($this->hook)) eval($this->hook);
   }
}

// some PHP code...

$user_data = unserialize($_COOKIE['data']);

// some PHP code...
```

그림 4 Exploit - wakeup method[4]

탈취한 쿠키 정보를 가지고 호스트로 전송하여 사용자로 위장, 변조가 가능 해진다.

4. 분석 도구

본 절에서는 KISA에서 소개된 6가지의 분석도구 중 2가지를 임의로 선택했다.[9]

4-1. Yasca

Yasca에서 제공하는 모듈을 활용하여 3-1에서 설명한 XSS를 탐지할 수 있는 방법을 포함하고 있기에 필자는 해당 정적 분석 도구를 활용하는 것이 효과적이라 판단했다. grep 플러그-인을 통하여

<%=request.getParameter("foo")%>,<?=$_REQUEST["foo"]?> 스크립트를 삽입하여 XSS 공격에 대응할 수 있다.

또한 Yasca는 애플리케이션 소스 코드의 보안 취약점을 탐지하기 위한 OSS 정적 분석도구다.[10] Secure Coding과 관련하여 플러그인을 지원 한다. 각종 Java, C, C++, PHP, JSP 언어를 지원 하고 간단한 코드 패턴 검사까지도 가능 하다. 굉장히 유연하고 확장하기 쉽도록 설계되어 있으며 사용하기 쉽다는 점이 가장 큰 강점이다.

4-2. LAPSE+

OWASP에서 진행하는 프로젝트로 Java EE 응용 프로그램을 위한 취약점 스캐너이다.[11] 코드 정적 분석을 기반으로한 도구이며 Eclipse 기반 플러그인으로 개발 됐다. 지원하는 기능은 각종 Injection 취약점, XSS, 파라미터 변조, URL 변조 등의 취약점 분석이 가능하다.

3-2 절에 대한 대처로 LAPSE+를 꼽은 이유는 기본적으로 매개변수 조작, URL 조작, 헤더 조작에 대한 탐지를

제공하고 있으며, 웹 쿠키를 통한 주입 공격 또한 탐지 가능하다.

5. 결론

본 논문에서는 OSS 취약점에 대한 최근 사례와 공격 방법과 그를 해결한 분석 도구를 소개하였다.

OSS를 사용함에 따라 서론에서 언급하였듯 복제, 수정, 배포가 쉽기 때문에 충분히 검증되지 않았음에도 불구하고 소스 코드가 쉽게 취약점에 노출되기 쉬울 것이다. 사용자는 OSS 활용 시에 보안 문제에 대한 경각심을 가지고 제시한 분석 도구를 통하여 취약점을 보완할 것을 제시한다. 근본적으로 OSS 개발 단계에서 보안적인 관점에서 충분히 검토되어 배포하는 것이 필수적으로 수행돼야 할 것이다. 4절에서 소개된 분석 도구 외에도 NIST에서 제안하는 분석 도구들을 활용하여 크로스 체크하는 것을 권장 한다.[12]

하나의 정적 분석 도구로 모든 취약점을 발견해낼 수 있다는 것은 불가능할 것이기 때문에 향후 연구에서는 실제 사용되는 OSS에서 각 분석도구를 활용하여 가장 적합한 취약점 분석 방법을 연구하는 것을 목표로 한다.

참고문헌

[1] WordPress - https://wordpress.org/
[2]WPScan Vulnerability Database - https://wpvulndb.com/
[3]Cross-site Scripting (XSS) - https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
[4] PHP Object Injection - https://www.owasp.org/index.php/PHP_Object_Injection
[5]https://sucuri.net/2019/03/0day-vulnerability-in-easy-wp-smtp-affects-thousands-of-sites.html
[6]CVE-2018-16065 Detail - https://nvd.nist.gov/vuln/detail/CVE-2018-16065
[7]http://gs.statcounter.com/browser-market-share
[8] https://www.cloudflare.com/learning/security/threats/cross-site-scripting/
[9]공개용 소스코드 보안약점 분석도구 개발 동향 - http://www.kisa.or.kr/uploadfile/201406/20140616095754317 4.pdf
[10] Yasca - https://www.scovetta.com/yasca/, https://www.owasp.org/images/1/1f/NYPHP-Yasca.pdf
[11] LAPSE+ - https://www.owasp.org/index.php/OWASP_LAPSE_Project
[12] NIST - SAMATE https://samate.nist.gov/index.php/Source_Code_Security_Analyzers.html