Open Web Application Security Project (OWASP)

Response to Draft W3C Best Practices for Mobile Web Applications

Summary

The Open Web Application Security Project (OWASP) supports new and improved standards and guidance. We are pleased to contribute to the development of this draft W3C Recommendation.

Applications, and especially web applications and web services, are increasingly being targeted to spread malware and to access sensitive data. In our experience, many of the flaws that affect mobile web applications are identical to flaws that affect traditional web applications. OWASP has a large corpus of widely-referenced[1] open and free guidance which helps software developers and security engineers build better software. Thus, we believe that our organization is uniquely qualified to comment on the sections of this document which are security related.

We were concerned when we read section 3.2 of the current draft of the recommendation and found that the only security recommendation is to advise against executing unescaped JSON data. This is certainly a good recommendation, but it is hardly the most significant security risk facing mobile web applications today.

We would like to draw your attention to our flagship document, the OWASP Top 10, which was updated earlier this year and represents what we believe are the Top 10 Most Critical Security Risks:

http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[1] http://www.owasp.org/index.php/Industry:Citations

Note that these apply to both mobile web applications as well as traditional web applications. Other specific issues we have discovered which are somewhat unique to mobile web applications are as follows:

- "presence on carrier network" being considered equivalent to authentication and authorization
- implementation of client side security controls which are not enforced server side.

In order to provide the best guidance for W3C's constituency, we would like to see OWASP materials referenced in the forthcoming W3C Recommendation. The Top 10 is obviously not an exhaustive list of security considerations for developers of mobile apps. Thus we would also encourage the W3C to include references to our more in-depth resources for developers, such as our "Development Guide"[2] and our "Testing Guide"[3].

Apart from including references to these other documents, we believe further detail and/or reference to more sources of information would be a great benefit to readers of the final document. This may be using additional footnotes or by the inclusion of a bibliography/further guidance section. In particular, we would like to recommend three of OWASP's key projects—the Development Guide, the Testing Guide and the Application Security Verification Standard (ASVS)[4]—containing detailed information on good development, testing and verification practices respectively. These include significant guidance on building, testing and verifying web applications, including web applications.

Our detailed point-by-point response follows.

---

[2] http://www.owasp.org/index.php/Guide_Table_of_Contents

[3] http://www.owasp.org/index.php/OWASP_Testing_Guide_v3_Table_of_Contents

[4] http://www.owasp.org/index.php/ASVS

Detailed Response

We suggest the following additions to section 3.2:

3.2.2 Validate Input

3.2.2.1 What it means

When accepting user input, be sure to validate the user input against a centralized whitelist of known good responses. For example, if accepting user input from a form parameter called "FirstName", and valid characters are only A-Z, a-z, and (space), the web application should check to ensure that no invalid characters have been submitted before passing the form input on for further processing.

Failing to perform input validation can lead to a variety of problems, including (but not limited to) injection flaws (such as command injection, SQL injection, LDAP injection, XPATH injection, etc.), as well as cross site scripting and other problems.

3.2.2.2 How to do it

In the example mentioned above, a suitable regex for validating the input would be [A-Za-z].

The OWASP ESAPI[5] is an open source security framework available for a variety of languages (including J2EE, .NET, PHP, Javascript, Coldfusion and more) and can be used to create a central input validation facility for your application. An example call to ESAPI's "Validator" interface is as follows (in Java):

```
String validatedFirstName = ESAPI.validator().getValidInput("FirstName",
      myForm.getFirstName(), "FirstNameRegex", 255, false, errorList);
```

3.2.3 Encode Output

3.2.3.1 What it means

Before rendering dynamic content to the user, ensure that content is encoded for the appropriate context. For example, if rendering HTML content for display to the user, HTML encoding must be applied, for example transforming the "<" character into "<". Properly encoding output is a strong defense against Cross Site Scripting (XSS).

3.2.3.2 How to do it

An example requiring both input validation and output encoding is a mobile web application which permits a user to make a comment on a blog post.

---

5 http://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API

The OWASP ESAPI can easily be used to perform both of these functions. An example follows:

```
//performing input validation

String cleanComment = ESAPI.validator().getValidInput("comment",
       request.getParameter("comment"), "CommentRegex", 300, false, errorList);

//performing output encoding for the HTML context

String safeOutput = ESAPI.encoder().encodeForHTML( cleanComment );
```

3.2.4 Set "HttpOnly" and "Secure" Flag on Session Cookies

3.2.4.1 What it means

If cookies are used as session variables to track the logged in state of a user in the application, ensure that these cookies are set with the HttpOnly and secure flags. The HttpOnly flag, when appended to a Set-Cookie response header, will instruct supporting browsers to disallow client-side script to access the cookie. This mounts an effective defense against XSS attacks designed to steal session cookies.

The secure flag, when appended to a Set-Cookie response header, prevents supporting browsers from providing the session cookie when connecting to the same origin but without transport layer security (SSL). This mounts an effective attack against SSL stripping attacks[6].

3.2.4.2 How to do it

Java developers can set the flags manually as follows

```
String sessionid = request.getSession().getId();

response.setHeader("SET-COOKIE", "JSESSIONID=" + sessionid + "; HttpOnly" + "; secure");
```

Various web frameworks allow these cookies to be set automatically via configuration files. See http://www.owasp.org/index.php/HttpOnly for more information.

---

6 http://www.thoughtcrime.org/software/sslstrip/

3.2.5 Prevent Cross Site Request Forgery (CSRF)

3.2.5.1 What it means

Cross Site Request Forgery (CSRF) is possible when a site does not validate that a request it receives was sent intentionally.

Consider the following HTML:

```
<html>

<p>this is a website for nothing in particular

<img src=images/some_image.jpg>

<img src=http://1.2.3.4/admin.cgi?uiViewUserName=admin&uiViewPassword=admin
&setDNS1=5.6.7.8>

</html>
```

A user who visits this page may unintentionally set the default DNS server of their wireless router to an attacker controlled malicious server. This is due to the fact that the admin.cgi in the example does require that the user prove that they intended to send the request. See http://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29 for more information about CSRF attacks.

3.2.5.2 How to do it

To properly mitigate CSRF, a developer of a mobile web application must ensure that with every form that is rendered to the user, a unique random value (token or nonce) must be included. This token should be submitted as a hidden variable in the form, and should be validated for correctness prior to performing processing on the form input.

Detailed information about using the OWASP ESAPI to implement robust anti-CSRF protections is available at http://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29#How_to_Prevent_CSRF_Vulnerabilites.

About OWASP

This response is submitted on behalf of the Open Web Application Security Project (OWASP) by the OWASP Global Industry Committee. OWASP is a worldwide free and open community focused on improving the security of application software. Our mission is to make application security "visible," so that people and organizations can make informed decisions about application security risks. Everyone is free to participate in OWASP and all of our materials are available under a free and open software license. The OWASP Foundation is a U.S. recognized 501(c)(3) not-for-profit charitable organization, that ensures the ongoing availability and support for our work at OWASP. Further information:

- OWASP Foundation
  http://www.owasp.org/index.php/OWASP_Foundation
- About The Open Web Application Security Project
  http://www.owasp.org/index.php/About_OWASP
- The Open Web Application Security Project
  http://www.owasp.org/
- OWASP Global Industry Committee
  http://www.owasp.org/index.php/Global_Industry_Committee
- Legislation, standards, guidelines, etc referencing OWASP
  http://www.owasp.org/index.php/Industry:Citations

Contact for this response

This response is submitted by the OWASP Global Industry Committee. If you have any queries, please contact:

- David Campbell
  dcampbell 'at' owasp.org