CS 6431 - Security and Privacy Technologies
Fall 2014

Homework #2

Due: 7:30pm EDT, October 22, 2014

NO LATE SUBMISSIONS WILL BE ACCEPTED

YOUR NAME: _____

Collaboration policy

No collaboration is permitted on this assignment. Any cheating (*e.g.*, submitting another person's work as your own, or permitting your work to be copied) will automatically result in a failing grade.

Homework #2 (50 points)

Problem 1

Web browsers' same origin policy (SOP) for DOM access is based on the (protocol, host, port) triple. The SOP for sending cookies to websites involves domain and path, but cookies marked "secure" are sent over HTTPS only.

In Safari before version 3.0, the SOP for DOM access was defined using host and port only (*i.e.*, it did not include the protocol).

Problem 1a (5 points)

Explain how a network attacker could steal secure `google.com` cookies. (Hint: consider a user who logs into Gmail using HTTPS, but then receives a `google.com` page served over HTTP.)

Problem 1b (5 points)

Under the same assumptions, is it possible for a Web attacker to steal secure `google.com` cookies? Describe an attack or explain why you believe none exists. Recall that a Web attacker can set up a malicious website (at some domain other than `google.com`) and trick the user into visiting this site, but cannot intercept or forge network packets.

Problem 2 (6 points)

Suppose you want to design a client-side protection tool against XSRF attacks using an HTTP proxy or a browser plugin. What would be the main issues to solve and what security checks might your tool perform? It is important that your defense not prevent "normal" webpages from rendering correctly in the browser.

Problem 3

Consider the following PHP script for logging into a website:

```
$username = $_GET[user];
$password = $_GET[pwd];
$sql = "SELECT * FROM usertable
        WHERE username= '$username' AND password = '$password' ";
$result = $db->query($sql);
if ($result->num_rows > 0) { /* successful login */ }
else                       { /* login failed */ }
```

Problem 3a (4 points)

Give an example of a username that will successfully subvert the above authentication code.

Problem 3b (4 points)

The PHP function `addslashes` adds a slash before every quote. For example, `addslashes(x'y)` outputs the string `x\'y`.

Suppose user's input is sanitized as follows:

```
$username = addslashes($_GET[user]);
$password = addslashes($_GET[pwd]);
```

In the Chinese, Korean, and Japanese unicode character sets, some characters are encoded as single bytes, while others are double bytes. For example, the database interprets `0x5C` as \, `0x27` as ', `0x5C27` as \', but `0xBF5C` is interpreted as a single Chinese character.

Give an example of a username that will successfully subvert the above authentication code even if the input is sanitized using `addslashes`.

Problem 3c (4 points)

How should `addslashes` be implemented to prevent SQL injection attacks?

Problem 4 (6 points)

Consider extending the technique for detecting SQL injection vulnerabilities described in the PLDI 2007 paper by Wassermann and Su ("Sound and Precise Analysis of Web Applications for Injection Vulnerabilities") to detect cross-site scripting vulnerabilities.

What do you think would be the main difficulties?

Problem 5 (5 points)

In their "Attacks on WebView in the Android System" paper, Luo et al. describe several attacks exploiting vulnerabilities in the WebView security model. Which of these attacks can and cannot be considered instances of permission re-delegation? Explain.

Problem 6 (5 points)

Tatebayashi, Matsuzaki, and Newman (TMN) proposed the following protocol, which enables Alice and Bob to establish a shared symmetric key K with the help of a trusted server S. Both Alice and Bob know the server's public key K_S. Alice randomly generates a temporary secret K_A, while Bob randomly generates the new key K to be shared with Alice. The protocol then proceeds as follows:

Alice	$\rightarrow$	*Server*	$enc_{K_S}(K_A)$
Bob	$\rightarrow$	*Server*	$enc_{K_S}(K)$
Server	$\rightarrow$	*Alice*	$K \oplus K_A$
			Alice recovers key K as $K_A \oplus (K \oplus K_A)$

To summarize, Alice sends her secret to the Server encrypted with the Server's public key, while Bob sends the newly generated key, also under encryption. The Server XORs the two values together and sends the result to Alice. As a result, both Alice and Bob know K.

Suppose an active network attacker observes all messages between Bob and the server. How can he extract the key K that Alice and Bob are using to protect their communications?

Problem 7 (6 points)

In their "How to Shop for Free Online" paper, Wang et al. describe vulnerabilities in several cashier-as-a-service protocols. For each of these protocols, explain how its design violates one or more of the principles listed by Abadi and Needham in their paper on "Prudent Engineering Practice for Cryptographic Protocols."