Security Analysis of Github

Sean Smith, Kyle Holzinger, Amalia Safer

April 29, 2015

1 Introduction

This is time for all good men to come to the aid of their party!

Outline The remainder of this article is organized as follows. Section 2 gives account of previous work. Our new and exciting results are described in Section 3. Finally, Section 4 gives the conclusions.

2 Cookies

A session cookie called `user_session is stored which contains a seemingly random nonce. When a get request is made to https://github.com, the cookie is sent and the database is queried to see if the cookie is valid. If the cookie is valid it will return user data as if the user is logged in. There are two more cookies of importance logged_in which is a yes/no value and dotcom_user which is the user's username. To impersonate a user, only the user_session is needed, the logged_in cookie will always be yes and the dotcom_user will be filled by the server if the user_session cookie is valid. When the user logs out, the cookie is invalided by the server in a post request to https://github.com/logout that contains the content type and an authenticity token. The authenticity token is a random nonce that github uses to prevent against CSRF. If an attacker is using the user's cookie and the user logs out, the attacker's cookie will be invalidated.`

When the user logs in, a post request is sent to https://github.com/session with the username, password and authenticity token.

One attack is to guess a random cookie and query to see if it's valid. There are approximately 8 million active github users at a time so roughly 8 million valid cookies. Since you don't need the logged_in cookie to be set correctly, you can construct a random cookie and check if it's valid. The length of the cookie is 80 characters and each character is from the universe (a-z, A-Z, 0-9, -, _) which has a size of 64. Say the set of correct cookies S has size $|S| = 8,000,000$, the universe U has a size of $|U| = 64^{80}$. The probability of getting a correct cookie is so low that it's not a reasonable attack.

$$Pr[] \approx$$

3 Results

In this section we describe the results.

4 Conclusions

We worked hard, and achieved very little.