Abusing HTTP Status Codes to Expose Private Information

OP: https://grepular.com/Abusing_HTTP_Status_Codes_to_Expose_Private_Information

When you visit my website, I can automatically and silently determine if you're logged into Facebook,Twitter, GMail and Digg. There are almost certainly thousands of other sites with this issue too, but I picked a few vulnerable well known ones to get your attention. You may not care that I can tell you're logged into GMail, but would you care if I could tell you're logged into one or more porn or warez sites? Perhaps http://oppressive-regime.example.org/ would like to collect a list of their users who are logged into http://controversial-website.example.com/?

Ignoring the privacy implications for a second, as a website developer, you might like to know if your visitors are logged into GMail; you could use that information to automatically fill the email fields in your forms with "@gmail.com"... Perhaps you might want to make your Facebook "like" buttons more prominent if you can tell your visitor is logged into Facebook at the moment? Here's how I achieve this:
First of all. Lets check if you're logged into GMail right now (not including Google Apps)... (**Yes, you are logged in**). Now, how did I get that information? Really, really, easily... I generated a hidden image in my HTML similar to this:

```
<img style="display:none;"
     onload="logged_in_to_gmail()"
     onerror="not_logged_in_to_gmail()"
     src="https://mail.google.com/mail/photos/static/AD34hIhNx1pdsCxEpo6LavSR8d
YSmSi0KTM1pGxAjRio47pofmE9RH7bxPwelO8tlvpX3sbYkNfXT7HDAZJM_uf5qU2cvDJzlAWxu7-
jaBPbDXAjVL8YGpI"
/>
```

I generated the URL in the "src" attribute by logging into my own GMail account, then going into the general settings and uploading a picture in the "My Picture" section. I then ticked the "Visible to everyone" checkbox, and right clicked the uploaded image to get the image location. Fetching the content at that URL does two different things depending on whether or not you're logged into GMail. If you are logged into GMail, it returns an image. If you're not logged into GMail, it redirects to a HTML page. This is why the img tag in my example above works. "onload" is triggered if an image is returned, but "onerror" is triggered otherwise.

I tested this technique in Firefox, Safari, Chrome, Opera and various versions of Internet Explorer and it worked in them all. I reported it to Google and they described it as "expected

Abusing HTTP Status Codes to Expose Private Information

behaviour" and ignored it.

At this point, you might be wondering why I have "Status Codes" in the title; the method I use for attacking Facebook, Twitter and Digg is slightly different and works because various URLs provide different HTTP status codes depending on your logged in status. Unfortunately, this attack doesn't seem to work in Internet Explorer or Opera, but does work in Firefox, Chrome and Safari. If you're using a non-IE, non-Opera browser, here are tests for Twitter and Facebook:

```
Are you logged into Twitter ? (Yes, you are logged in)
Are you logged into Facebook? (Yes, you are logged in)
```

If you have JavaScript disabled on twitter.com and facebook.com, the above tests wont work. Here is how they work when you have JavaScript enabled:

```
<script type="text/javascript"
        src="https://twitter.com/account/use_phx?setting=false&amp;format=text"
        onload="not_logged_in_to_twitter()"
        onerror="logged_in_to_twitter()"
        async="async"
></script>

<script type="text/javascript"
        src="https://www.facebook.com/imike3"
        onload="logged_in_to_facebook()"
        onerror="not_logged_in_to_facebook()"
        async="async"
></script>
```

In Firefox, Safari and Chrome, the <script/> tags fire onload if a 200 HTTP status code is returned, even if there was no valid JavaScript and the Content-Type was text/html. But if the status code was one of 404, 403, 406 or 500, then onerror is triggered instead. In the above examples, the Twitter URL returns an error code if you're logged in, but redirects to the login form with a success status code if you're not logged in. The Facebook one works because my profile is only visible to people who are logged in and returns a 404 if you're not. There is a similar problem with Digg. http://digg.com/settings returns a 403 status code if you're not logged in, but a 200 if you are.

This can be an awkward problem to avoid if you're developing a website. Some of these

Abusing HTTP Status Codes to Expose Private Information

requests could be stopped by doing referrer checks; reject all external referrers for content only accessible when logged in. You want your status codes and responses to image requests to be relevant, but that can leak information. Firefox users could defend from this problem by using the Request Policy addon. I've never used it myself because it looks like a pain to manage, but it sounds like it would do the job.

And finally, this isn't just an issue of detecting whether or not a user is logged in. The question could technically be anything, if a HTTP response results in an image or html depending on the answer, or results in a success/error status code depending on the answer.

For the web developers out there who are familiar with jQuery, as a demonstration of the usefulness of this technique. The following chunk of code will detect if a user is logged into GMail, and if they are will replace all the mailto: links on your webpage with links to the GMail compose window (automatically filling in the To field):

```
$('<img/>').hide()
   .attr('src','https://mail.google.com/mail/photos/static/AD34hIhNx1pdsCxEpo6
LavSR8dYSmSi0KTM1pGxAjRio47pofmE9RH7bxPwelO8tlvpX3sbYkNfXT7HDAZJM_uf5qU2cvDJzl
AWxu7-jaBPbDXAjVL8YGpI')
   .load(function(){
      $('a[href^="mailto:"]').each(function(){
         var email = $(this).attr('href').replace(/^mailto:/,'');
         $(this).attr('href','https://mail.google.com/mail/?
view=cm&fs=1&tf=0&to='+escape(email));
      });
   })
   .appendTo('body');
```