# Краткие выводы:

По результатам сканирования были обнаружены следующие проблемы:

| № | Критичность | Название | Приложение |
| --- | --- | --- | --- |
| 1 | Высокая | Уязвимости Denial of Service, Command Injection ([A05:2021-Security Misconfiguration](https://owasp.org/Top10/A05_2021-Security_Misconfiguration/), [A03:2021-Injection](https://owasp.org/Top10/A03_2021-Injection/)) | Оба приложения |
| 2 | Высокая | Уязвимость [Path Traversal](https://owasp.org/www-community/attacks/Path_Traversal) ([A01:2021-Broken Access Control](https://owasp.org/Top10/A01_2021-Broken_Access_Control/)) | Оба приложения |
| 3 | Высокая | Уязвимость [Unrestricted File Upload](https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload). ([A03:2021-Injection](https://owasp.org/Top10/A03_2021-Injection/)) | Оба приложения |
| 4 | Высокая | Уязвимость [SQL Injection](https://owasp.org/www-community/attacks/SQL_Injection) ([A03:2021-Injection](https://owasp.org/Top10/A03_2021-Injection/)) | Оба приложения |
| 5 | Высокая | Слабый пароль администратора ([A07:2021-Identification and Authentication Failures](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/)) | NetologyVulnApp |
| 6 | Средняя | Использование чужой сессии. ([A07:2021-Identification and Authentication Failures](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/)) | NetologyVulnApp |
| 7 | Средняя | Уязвимость к XSS атакам ([A03:2021-Injection](https://owasp.org/Top10/A03_2021-Injection/), [Stored XSS](https://owasp.org/www-community/attacks/xss/#stored-xss-attacks)) | Оба приложения |
| 8 | Средняя | Уязвимость к BruteForce атакам. ([A07:2021-Identification and Authentication Failures](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/)) | Оба приложения |
| 9 | Средняя | Отсутствие защиты от атак типа Сlickjacking, XSRF. ([A01:2021-Broken Access Control](https://owasp.org/Top10/A01_2021-Broken_Access_Control/)) | Оба приложения |

В первую очередь рекомендованы к устранению уязвимости с Высоким уровнем критичности.  
Для быстрого снижения и уменьшения уровня уязвимостей, а также устранения уязвимостей и профилактики появления новых уязвимостей рекомендуется:

* Оценить уровень компании по OWASP SAMM
* Правильно настроить конфигурационный файл веб-сервера, сервера приложений, балансировщика нагрузки.
* Правильно настроить уровень права доступа пользователей и приложений
* Не создавайте динамические SQL-запросы, используя простую конкатенацию строк
* Не используйте «exec» или эквивалентные функции
* Установить ограничение попыток ввода пароля
* Проверить процессы разработки ПО по ГОСТ Р ИСО/МЭК 12207-2010
* Внедрить практики безопасной разработки ПО, такие как статический анализ кода, динамический анализ кода, анализ зависимостей ПО. За основу можно взять ГОСТ Р 56939-2016 Защита информации. Разработка безопасного программного обеспечения. Общие требования.
* Внедрить систему менеджемта ИБ (ГОСТ Р ИСО/МЭК 27001-2021)

Для сбора (сканирования) информации методом черного ящика предоставлен ip-адрес 92.51.39.106. Работа будет состоять из трех этапов – OSINT, SCANNING, TESTING.

# OSINT

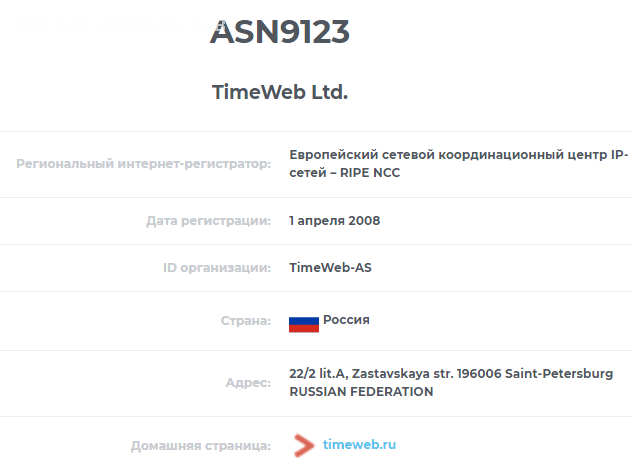
Дан ip-адрес 92.51.39.106

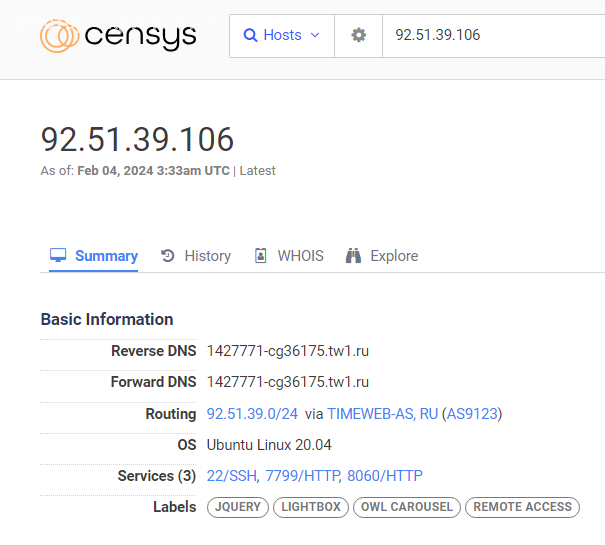
Для сбора информации о адресе использованы следующие открытые сервисы:

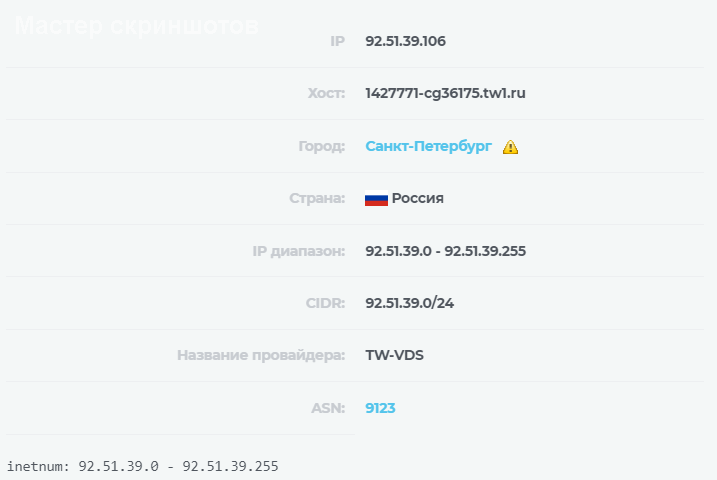
[2ip.ru](https://2ip.ru/whois/#result-anchor) [criminalip.io](https://www.criminalip.io/asset/report/92.51.39.106) [shodan.io](https://www.shodan.io/host/92.51.39.106) [censys.io](https://search.censys.io/hosts/92.51.39.106)

## ip-адрес 92.51.39.106 общие данные

Из открытых данных видно, что ip-адрес 92.51.39.106 принадлежит провайдеру TW-VDS, который зарегистрирован в России, 196006 г. Санкт-Петербург, ул. Заставская 22/2 лит.А. ID организации – TimeWeb-AS.





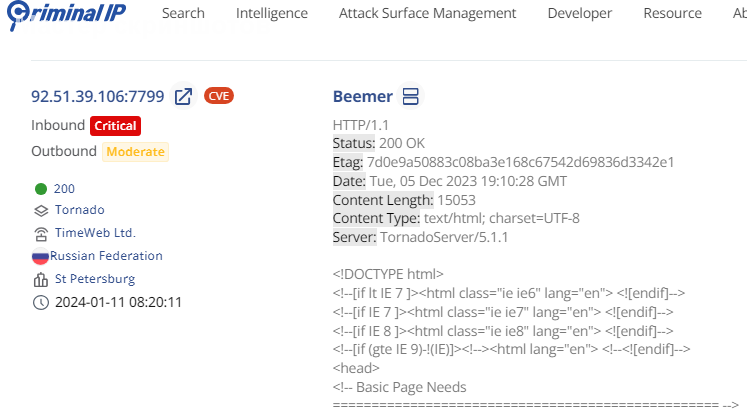


Диапазон ip-адресов от 92.51.39.0 до 92.51.39.255. Количество доменов в данном диапазоне 97 (на 23.01.2024). Данный ip адрес находится в маске подсети /24. Hostname - 1427771-cg36175.tw1.ru

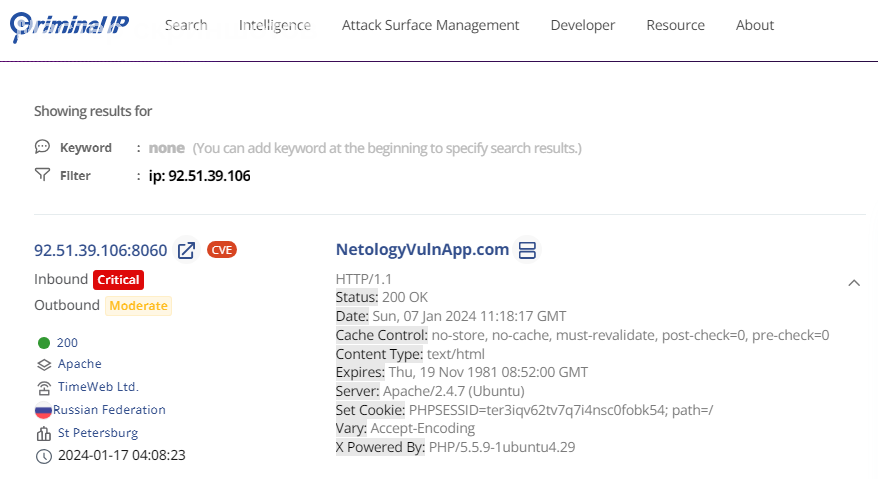


## Порт «7799/HTTP»

Используется незащищенный протокол HTTP/1.1. Используется server: Tornado Server версии 5.1.1



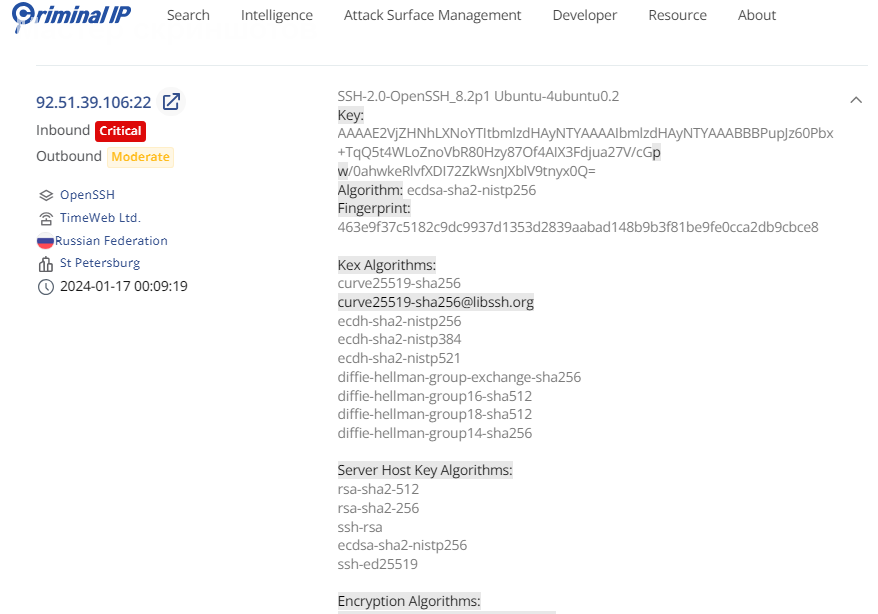
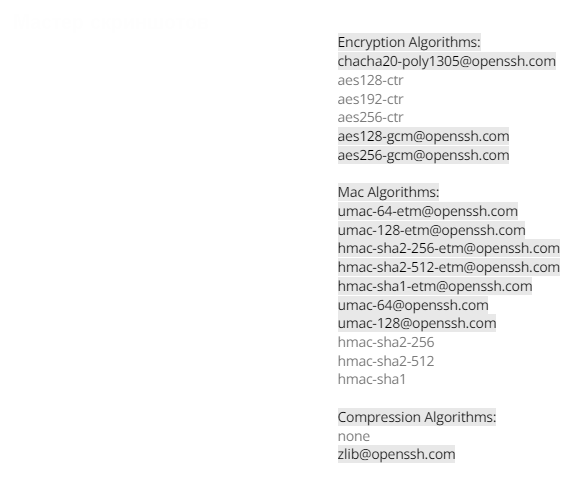
## Порт «8060/HTTP»

Используется незащищенный протокол HTTP/1.1. И­­спользуется Server: Apache версии 2.4.7 (Ubuntu). Работает на php версия 5.5.9-1ubuntu4.29

Скриншот с сайта censys.io

## Порт «22» OpenSSH

На порту «22» используется сервис SSH-2.0-OpenSSH версия 8.2p1. Используется ОС Ubuntu 20.04



Скриншот с censys.io

## Вывод

Ip-адрес 92.51.39.106 зарегистрирован у провайдера TW-VDS, который зарегистрирован в России, в г. Санкт-Петербурге.

Данный ip-адрес находится в маске подсети /24. Работает на Ubuntu Linux 20.04. На данном ip-адресе (92.51.39.106) открыты порты:

* «22» SSH (OpenBSD OpenSSH 8.2)
* «7799/HTTP» (веб-сайт Beemer, веб-сервер TornadoWeb Tornado 5.1.1)
* «8060/HTTP» (веб-сайт NetologyVulnApp.com, веб-сервер Apache HTTPD 2.4.7, язык PHP 5.5.9 )

Существует 86 уязвимостей, от средних до критичных. (по системе оценки CVSS v3).

# SCANNING

Для сканирования хоста использовались инструменты:

Nmap

Zenmap

DirSearch

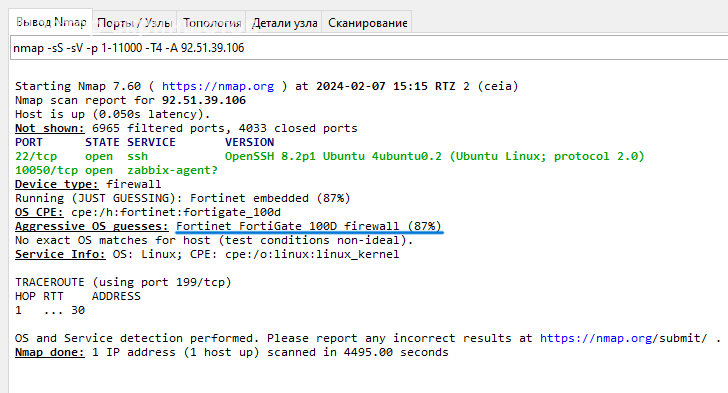
OWASP ZAP

Kali linux

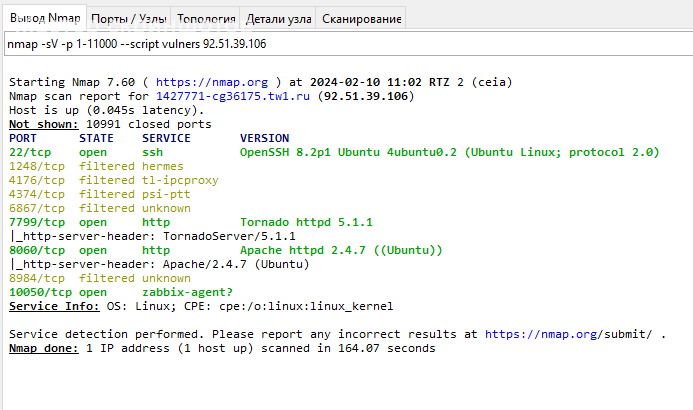
## Итоги сканирования:

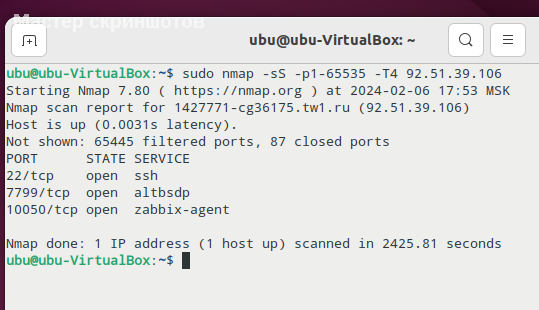
В ходе сканирования хоста, обнаружено 4 открытых порта:

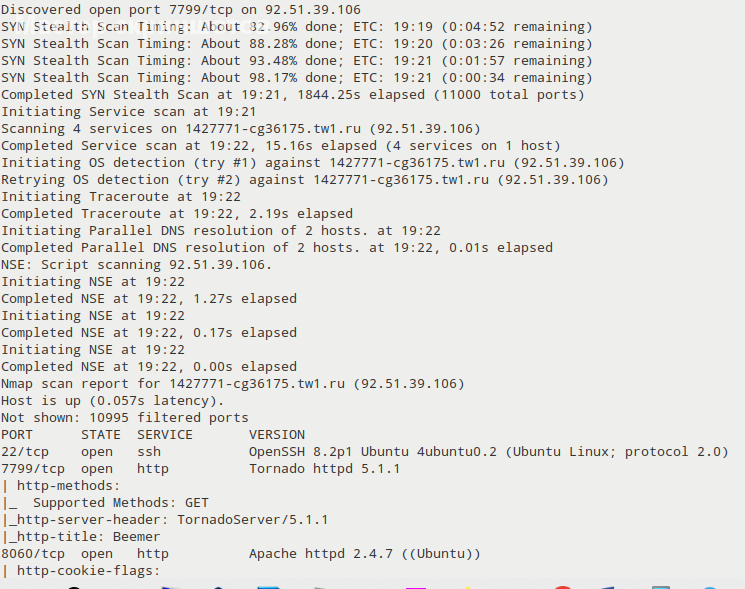
1. 22/tcp ssh (SSH-2.0-OpenSSH версия 8.2p1. Используется ОС Ubuntu 20.04)
2. 8060/tcp http (Apache версии 2.4.7 (Ubuntu))
3. 7799/tcp http (Tornado Server версии 5.1.1)
4. 10050/tcp zabbix-agent

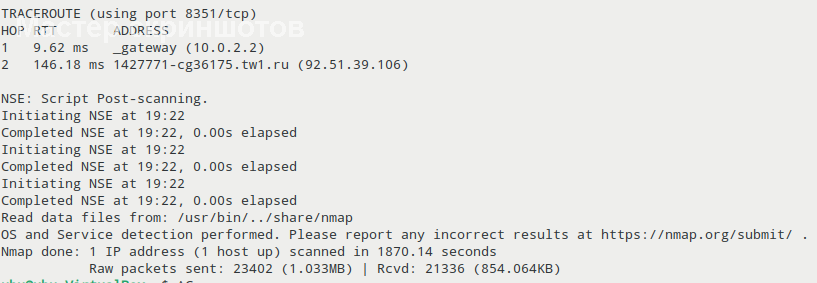
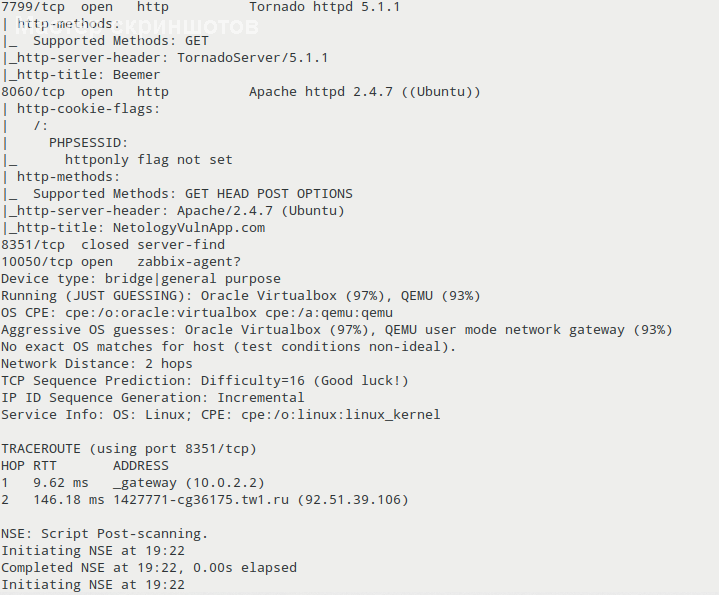
В ходе сканирования хоста удалось выяснить, что на хосте установлен Firewall Fortinet FortiGate 100D

## Режимы сканирования.

Для определения открытых, портов на хосте, использовались разные режимы сканирования используемых инструментов.

При использовании nmap и флага сканирования SYN (-sS) обнаружить удалось три открытых порта.

Предполагая, что на хосте расположен firewall, добавил флаг –Pn (не производить пинг хоста). Также, для получения детальной информации добавил флаг –sV обнаружение версии сервисов, расположенных на хосте. Обнаружить удалось четыре открытых порта.



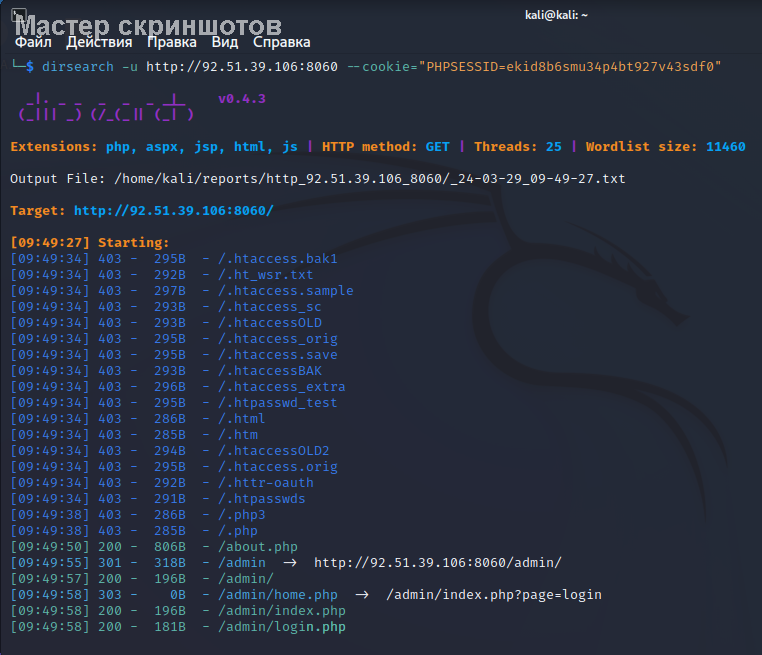
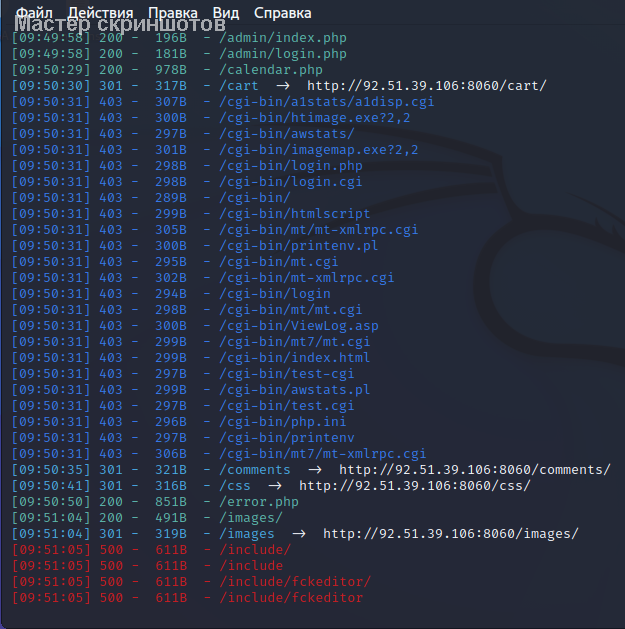
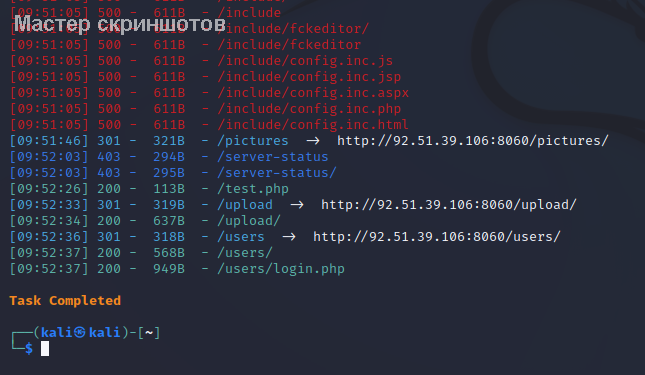
Dirsearch

С помощью Dirsearch проводим сканирование веб-серверов на наличие скрытых директорий и файлов.

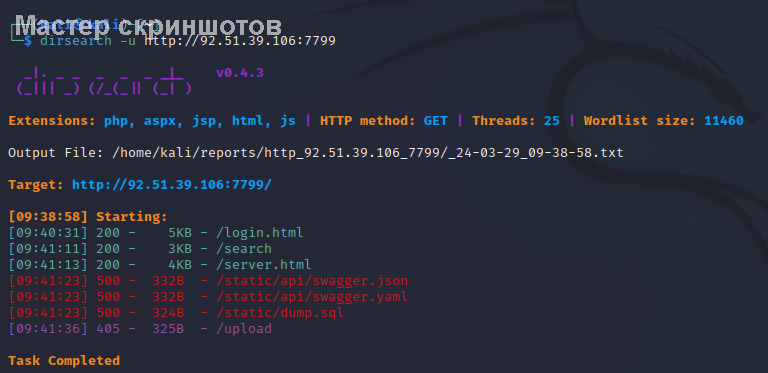
По результатам сканирования видно, что включена индексация директорий, что дает возможность:

* + Изучить файлы приложения
  + Найти файлы конфигурации и пароли
  + Попробовать внедрить shell-скрипты
  + Попробовать использовать уязвимость Path Traversal

Сканирование порта 8060



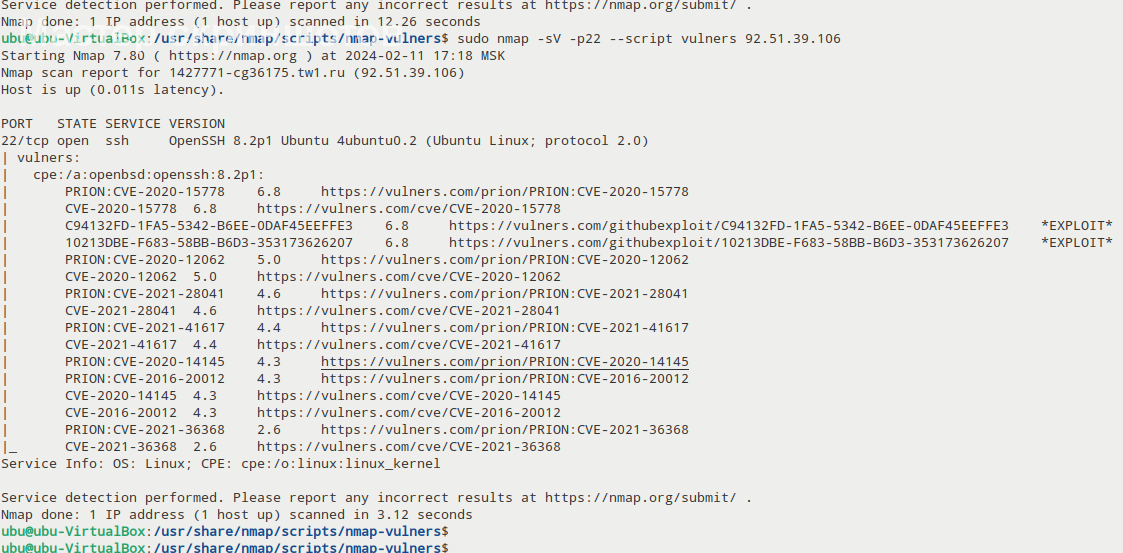
Сканирование порта 7799



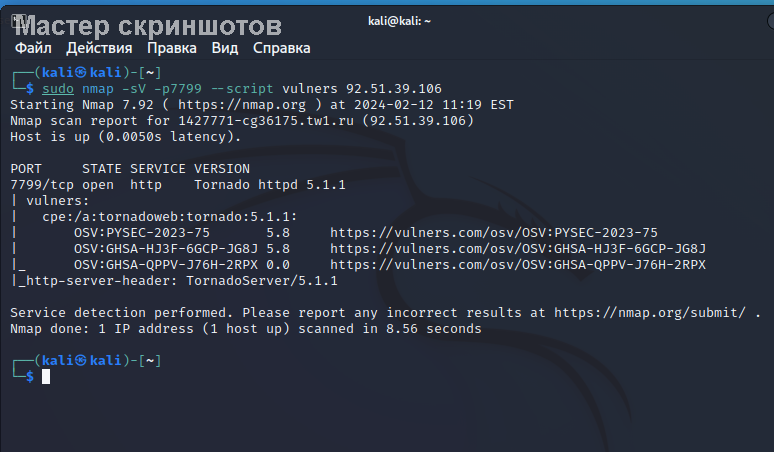
## Уязвимости сервисов

Для определения уязвимостей для сервисов, которые находятся на открытых портах, был применен nmap со скриптом.

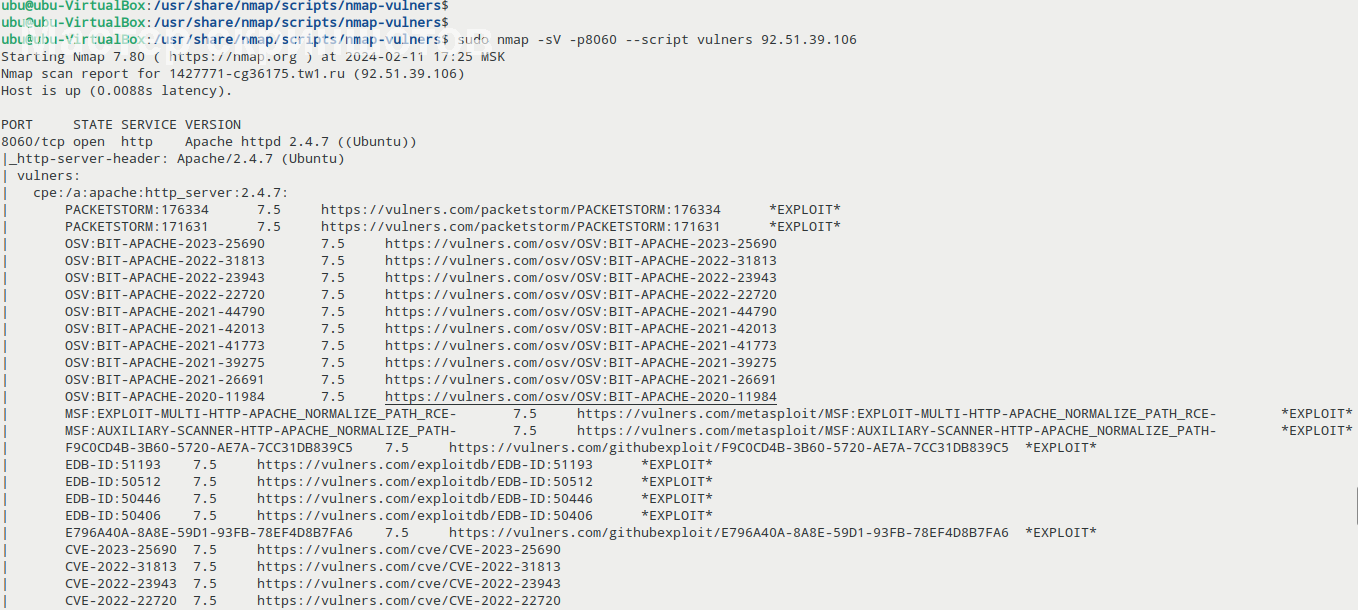
### 22 порт уязвимости.

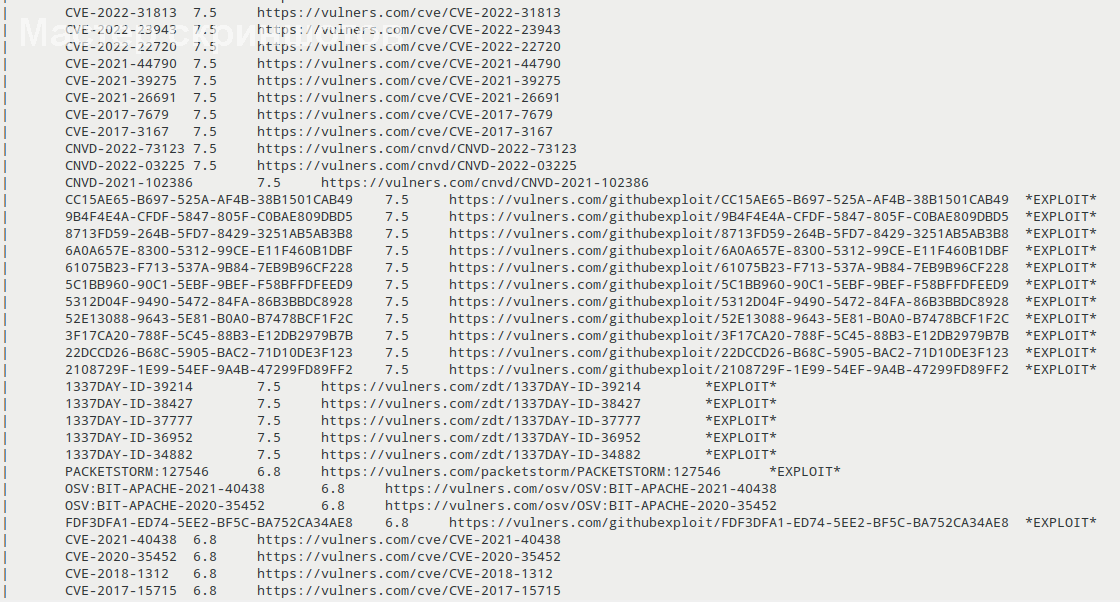


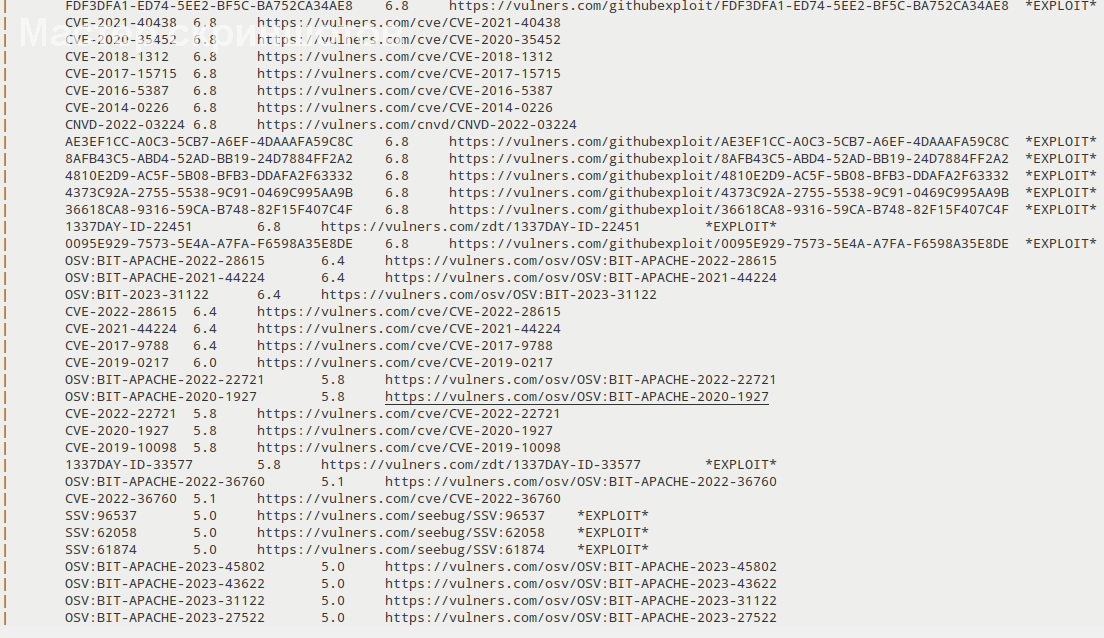
**7799 порт уязвимости.**

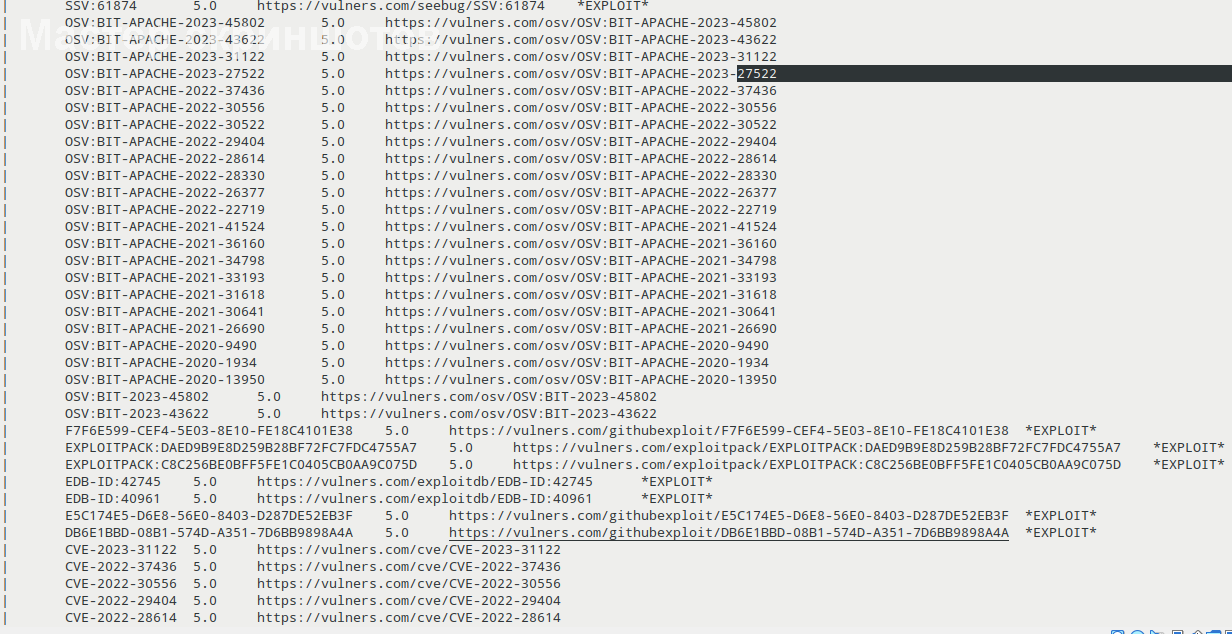


### 8060 порт уязвимости.

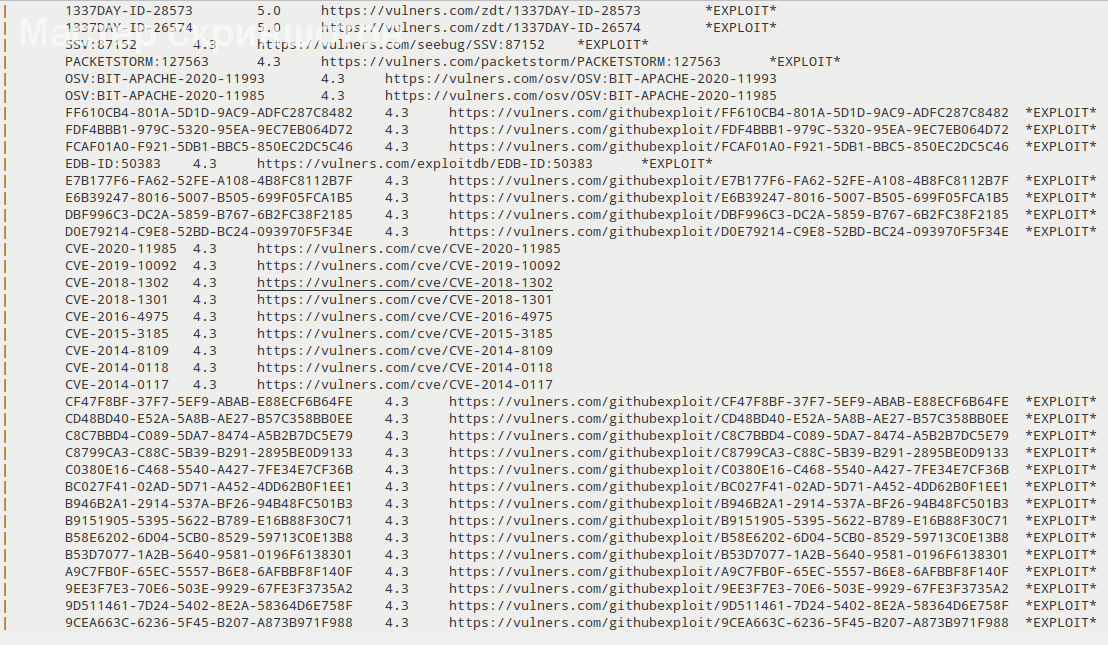


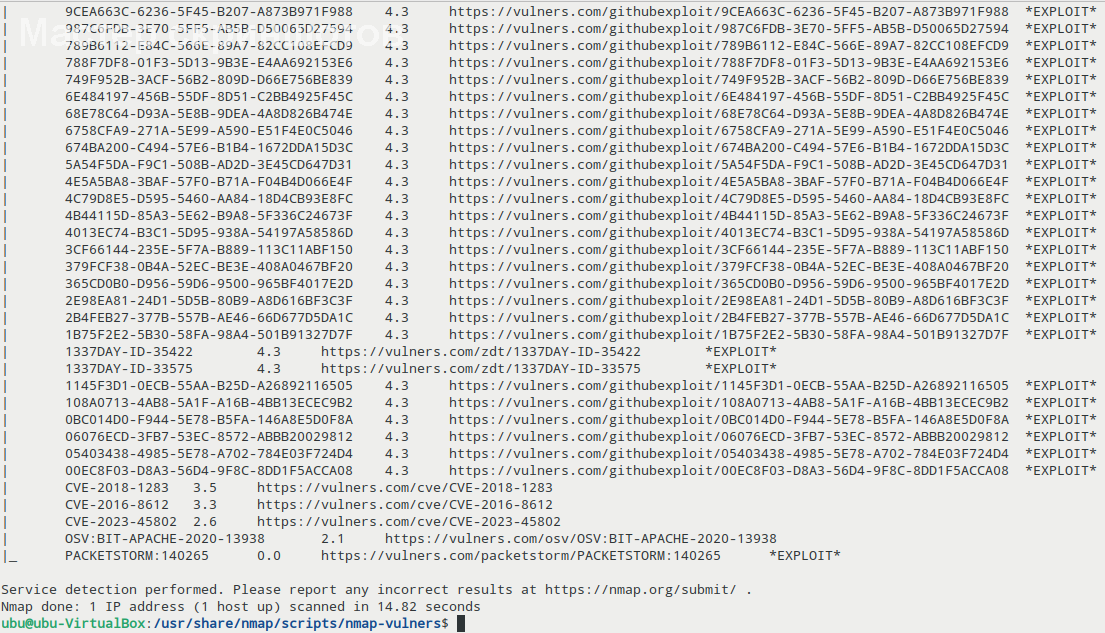












# TESTING

## NetologyVulnApp (port: 8060)

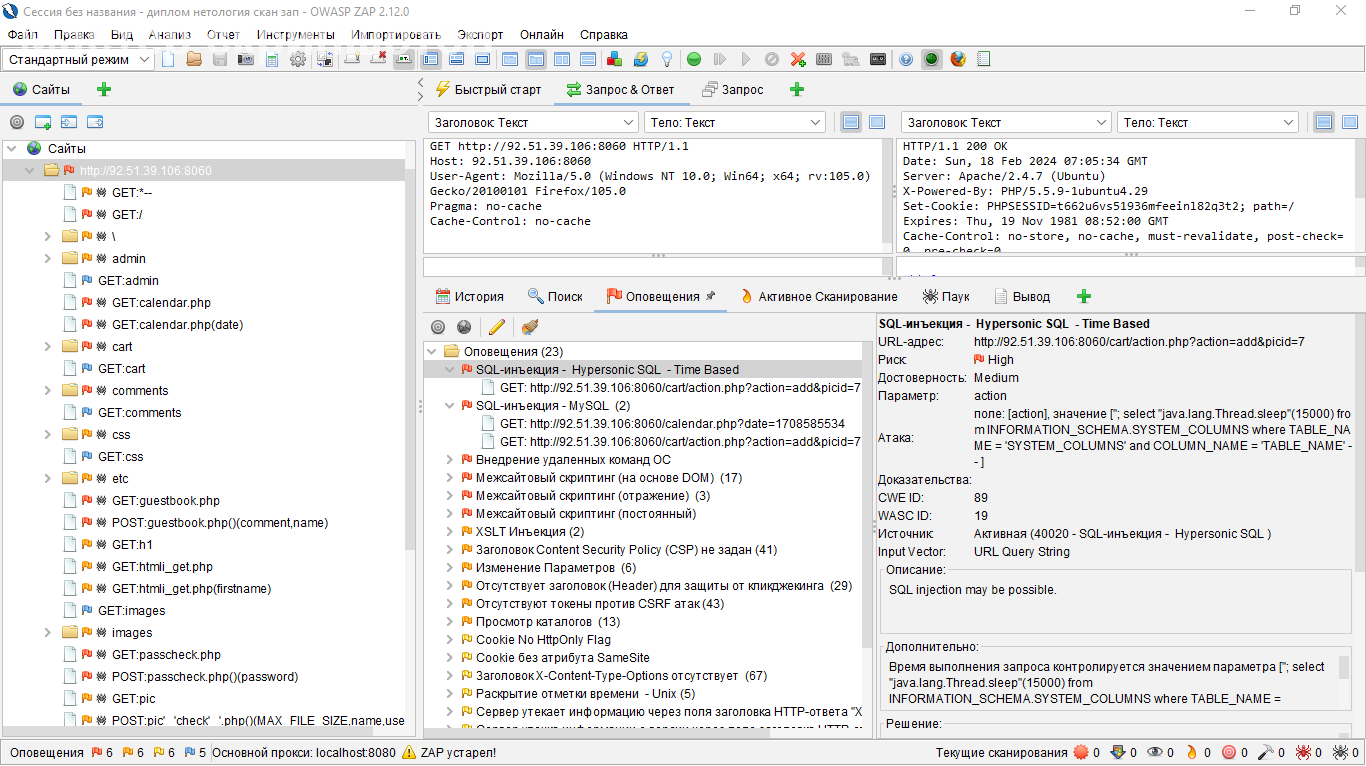
Тестирование приложения NetologyVulnApp проводилось средствами

* DirSearch
* OWASP ZAP
* Kali linux
* hydra

В результате тестирования были найдены следующие проблемы:

* SQL injection
* Внедрение удаленных команед ОС
* Межсайтовый скриптинг (на основе DOM-based)
* Межсайтовый скриптинг (Отраженный - Reflected)
* Межсайтовый скриптинг (Постоянный)
* Отсутствует заголовок (Header) для защиты от кликджекинга «ClickJacking»
* Отсутствуют токены против CSRF атак
* Просмотр каталогов
* Cookie без атрибута SameSite, HttpOnly, Strict-Transport-Security
* Отсутствует Заголовок X-Content-Type-Options
* Сервер предоставляет информацию о версии через поле заголовка HTTP-ответа “Server”
* Сервер предоставляет информацию на чем работает через поле заголовка HTTP-ответ “X-Powered-By”

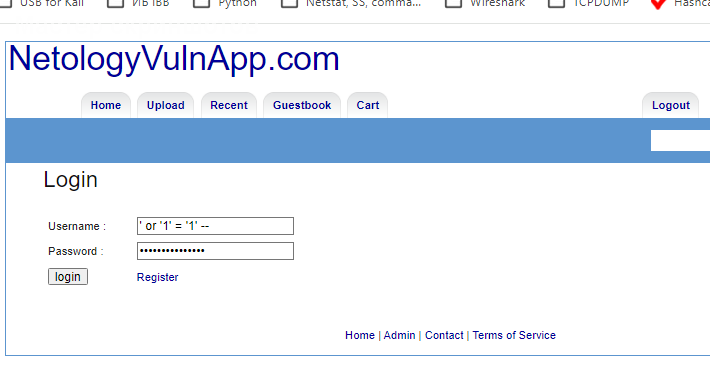
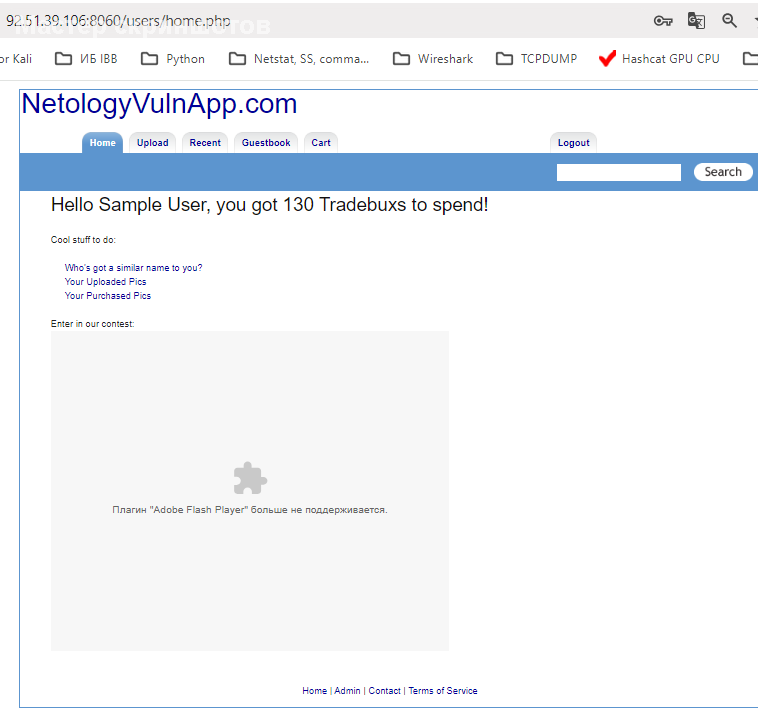
### **OWASP ZAP сканирование NetologyVulnApp**.



### Результаты тестирования:

### ****1. Уязвимость****[SQL Injection](https://owasp.org/www-community/attacks/SQL_Injection)****(****[A03:2021-Injection](https://owasp.org/Top10/A03_2021-Injection/)****)****

**Критичность:** Высокая  
**Страница:** http://92.51.39.106:8060/users/login.php



**Описание:**

Через форму авторизации пользователей есть возможность внедрить sql скрипт в поле логина. Были использованы два варианта кода sql: (' or '1' = '1' -- -), (' or 1 -- -). В результате чего удалось пройти авторизацию под пользователем “Sample User”

Существует возможность выполнения следующих действий:

* Добавление, изменение данных в таблицах
* Удаление данных из таблиц
* Нарушение схемы БД

**Предложения по исправлению:**

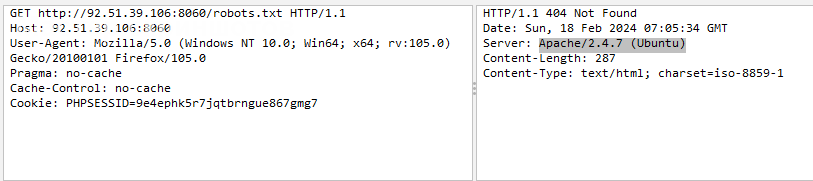
* Добавить валидацию, санитизацию входных данных с формы.

### 2. Уязвимость Security\_Misconfiguration (A05\_2021)

( <https://owasp.org/Top10/A05_2021-Security_Misconfiguration/> )

**Критичность:** Низкая

**Страница:**<http://92.51.39.106:8060/robots.txt>



**Описание**.

Доступ к такой информации может облегчить злоумышленникам определение других уязвимостей, которым подвержен ваш веб-сервер / сервер приложений.

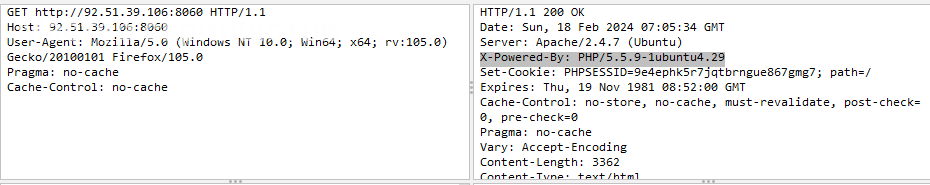
**Предложения по исправлению**.

Настроить и убедиться, что ваш веб-сервер, сервер приложений, балансировщик нагрузки и т.д. настроен на подавление заголовка «Server» при предоставлении общих сведений.

### 3. Уязвимость Broken\_Access\_Control (A01:2021)

( <https://owasp.org/Top10/A01_2021-Broken_Access_Control/> )

**Критичность:** Низкая

**Страница:** <http://92.51.39.106:8060/>

**Описание.**

Веб-сервер / сервер приложений пропускает информацию через один или несколько заголовков HTTP-ответа «X-Powered-By». Доступ к такой информации может облегчить злоумышленникам идентификацию других фреймворков / компонентов, от которых зависит ваше веб-приложение, и уязвимостей, которым могут быть подвержены такие компоненты.

**Решение.**

Убедитесь, что ваш веб-сервер, сервер приложений, балансировщик нагрузки и т. д. настроены на подавление заголовков «X-Powered-By».

### 4. ****Уязвимость****[Unrestricted File Upload](https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload)****. (****[A03:2021-Injection](https://owasp.org/Top10/A03_2021-Injection/)****)****

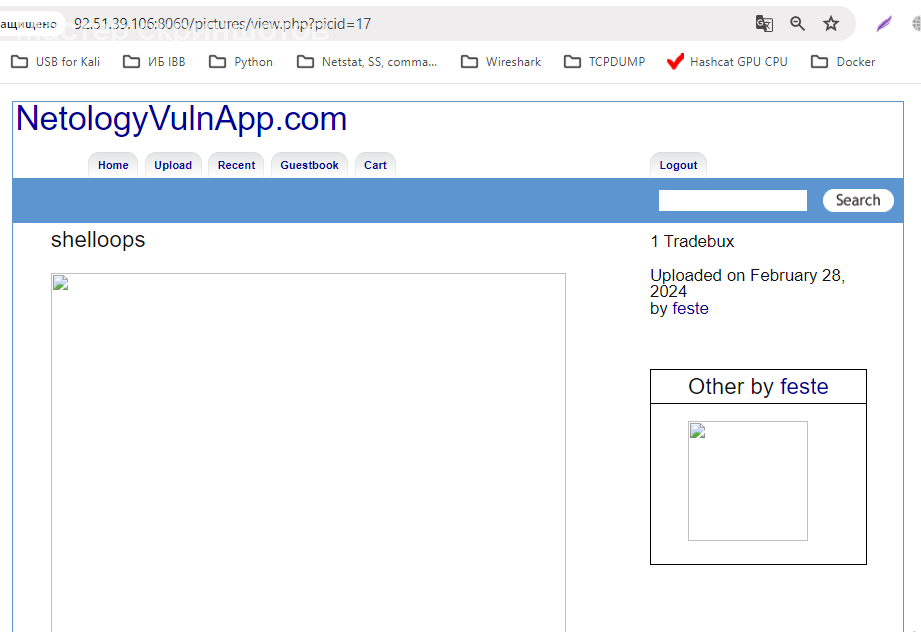
**Критичность:** Высокая

С**тран**ица**:** <http://92.51.39.106:8060/pictures/upload.php>

**Описание**: Уязвимость позволяет загрузить произвольный файл, отличный от картинки. Существует возможность выполнения следующих действий:

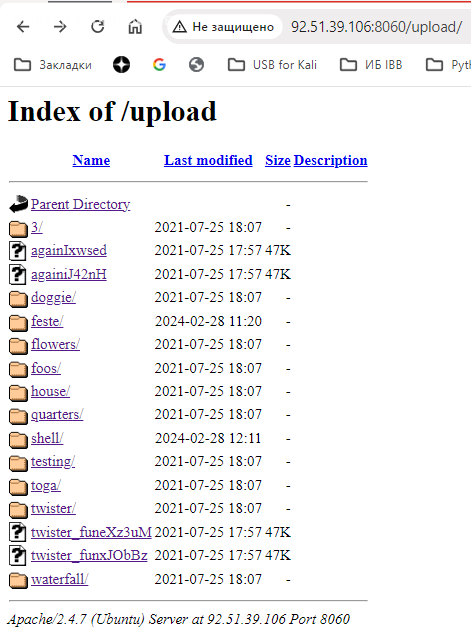
* Загрузить PHP-Shell файл

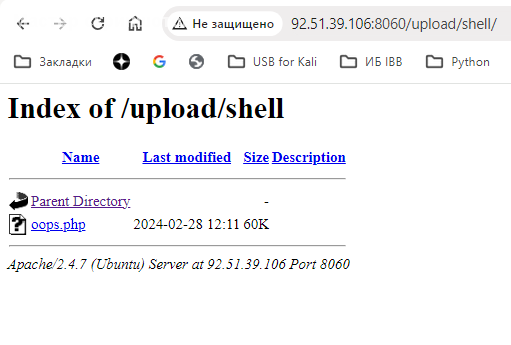
**Предложения по исправлению**:

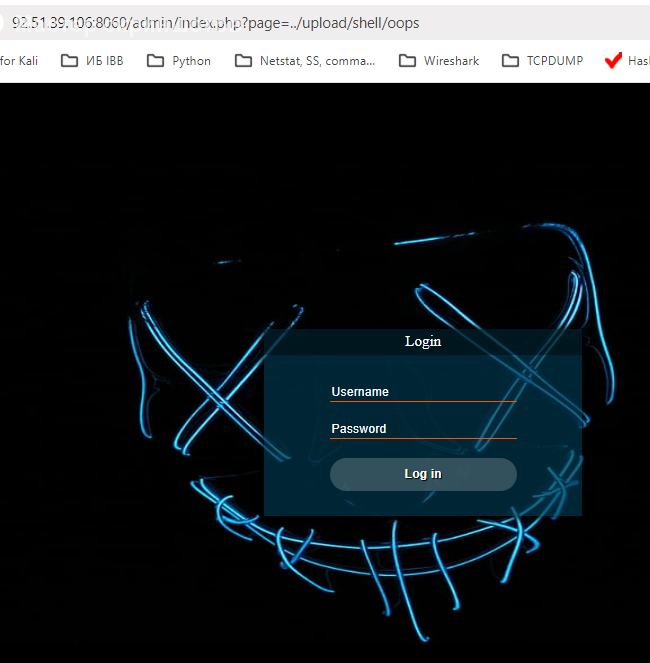
* Добавить валидацию входного по типу содержимого. Например, можно использовать сигнатурный анализ файла и сравнивать первые байты файлов с известными сигнатурами. Например, сигнатура для файлов формата JPEG будет выглядеть следующим образом: FF D8 FF E0, формата PNG будет выглядеть следующим образом: 89 50 4E 47.
* Разрешить загрузку картинок определённых расширений. Например, GPEG, PNG, BMP.
* Запускать приложение под пользователем с минимальными правами. Пользователь не должен иметь прав на чтение системных файлов, тем более на их модификацию или удаление.

Скриншот загрузки файла, отличного от картинки, на сервер.

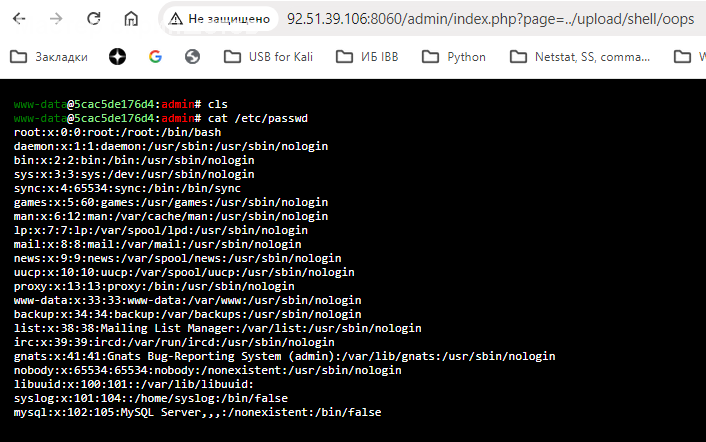
Воспользовавшись также уязвимостью «Неправильная конфигурация безопасности», мы можем просмотреть директории и содержимое папки upload. Находим путь, к нашему shell-файлу.





****Используя уязвимость Path Traversal, переходим на страницу <http://92.51.39.106:8060/admin/index.php?page=../upload/shell/oops> и видим окно нашего shell-приложения.

Заходим в приложение и выполняем команду на сервере для отображения содержимого файла: /etc/passwd. Сервер отправляет в ответ содержимое запрошенного файла.

****

### 5. ****Использование чужой сессии. (****[A07:2021-Identification and Authentication Failures](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/)****)****

**Критичность: Средняя**

**Описание**:

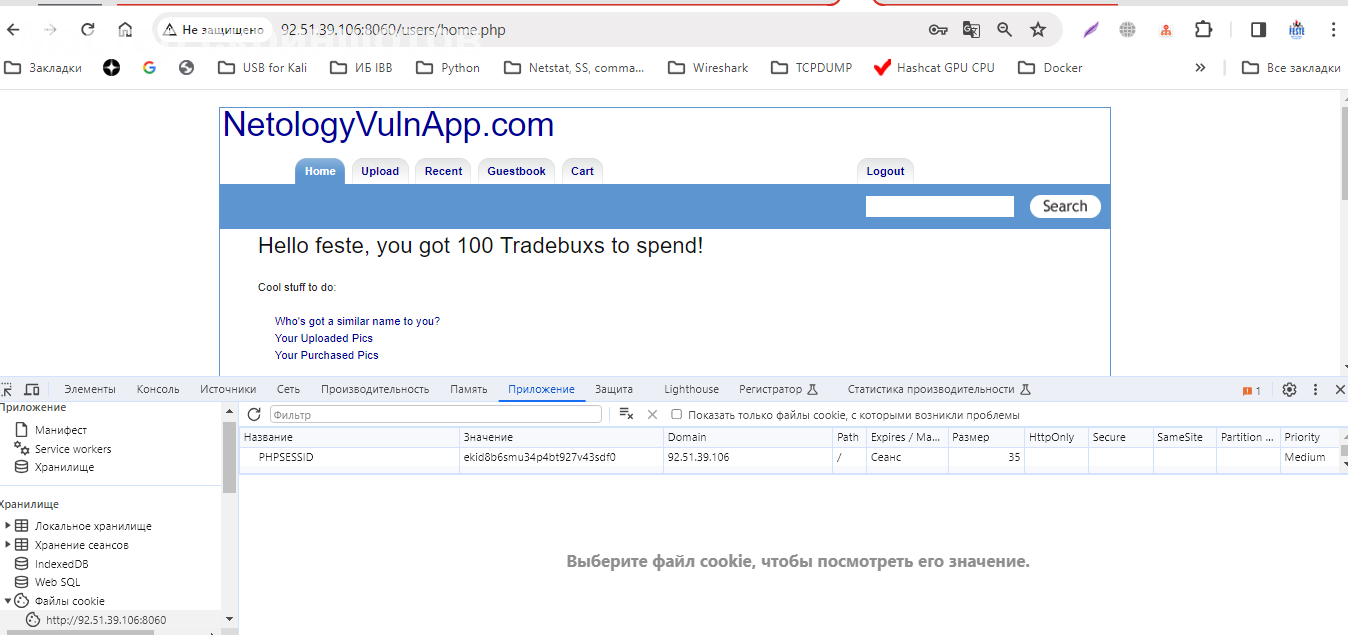
Есть возможность скопировать сессионную куку пользователя из одного браузера в другой и продолжить работать в обоих браузерах.  
Существует возможность выполнения следующих действий:

* Кражи пользовательской куки
* Реализация XSS атаки

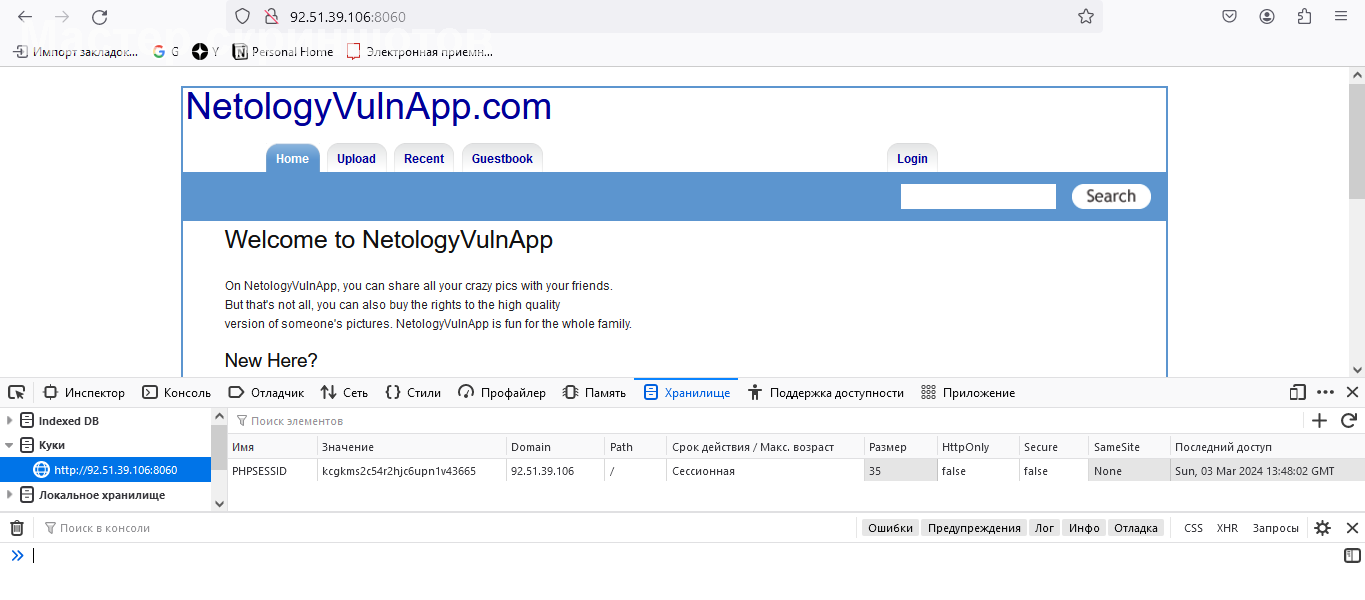
**Предложения по исправлению**:

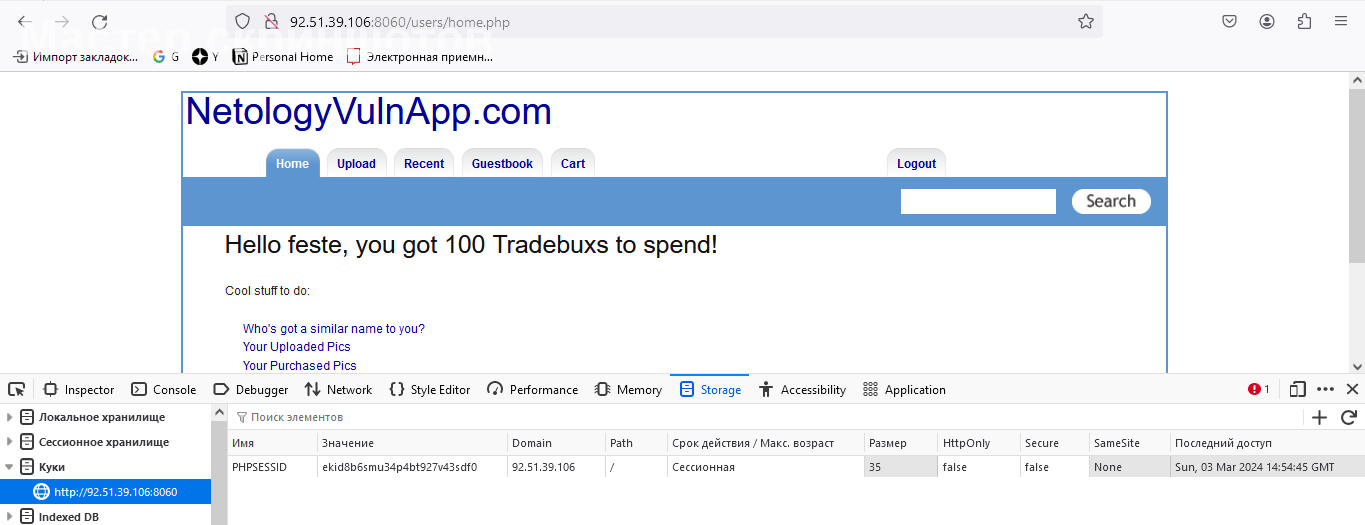
* Сделать привязку сессионной куки пользователя к устройству(браузеру)
* Установить время жизни сессии пользователя в период бездействия

**Реализация**

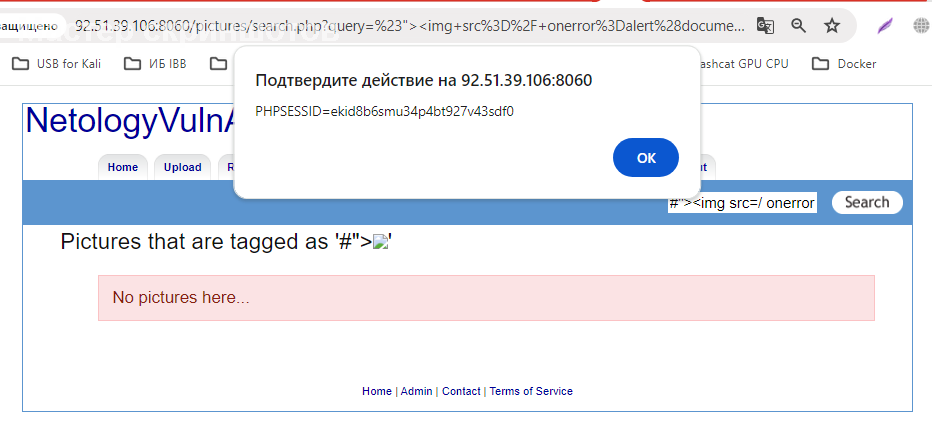
 Производим регистрацию нового пользователя. Заходим в инструменты разработчика и переходим в раздел cookie. В значение видим сессионный cookie пользователя.

Открываем приложение в новом браузере и в инструментах разработчика меняем текущий cookie на cookie нашего пользователя. Также можно установить нужный cookie через консоль разработчика, следующей командой: document.cookie="PHPSESSID=ekid8b6smu34p4bt927v43sdf0", в нашем примере.





Также сессионный cookie можно получить и скопировать, реализовав XSS, через уязвимое поле ввода, использовав: #"><img src=/ onerror=alert(document.cookie)>



### 6. ****Уязвимость к XSS атакам (****[A03:2021-Injection](https://owasp.org/Top10/A03_2021-Injection/)****,****[Stored XSS](https://owasp.org/www-community/attacks/xss/#stored-xss-attacks)****)****

**Критичность:** Средняя

**Страницы**:

* <http://92.51.39.106:8060/>
* <http://92.51.39.106:8060/guestbook.php>
* <http://92.51.39.106:8060/piccheck.php>
* <http://92.51.39.106:8060/pictures/search.php?query>=

**Payload**:

#"><img src=/ onerror=alert(document.cookie)>

<script>alert(document.cookie)</script>

<script>alert("XSS IS DONE ON THE PAGE")</script>

**Описание**: На страницах приложения происходит добавление пользовательского ввода на страницу без санитизации и экранирования вводимых данных.

Существует возможность выполнения следующих действий:

* Кража сессионной куки
* Перенаправление пользователей на сторонние сайты
* Выполнение XSRF атак на другие сайты с этой страницы

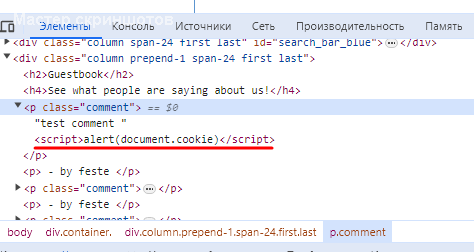
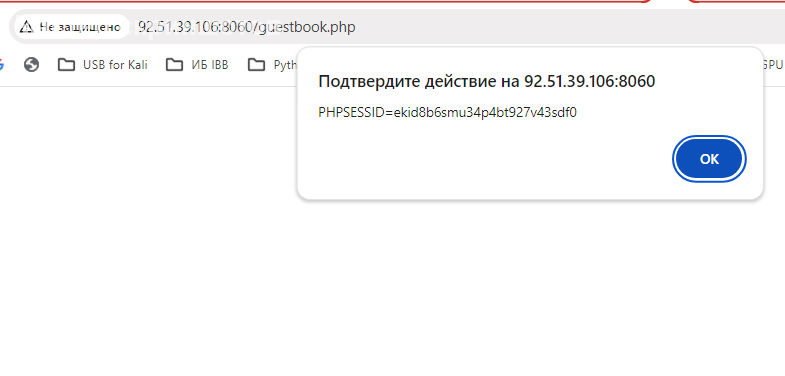
**Предложения по исправлению**:

* Добавить валидацию/санитизацию пользовательского ввода

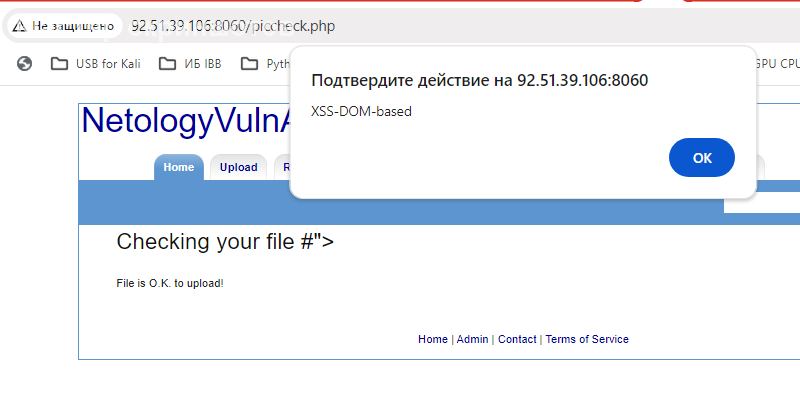
Реализация XSS:

На странице <http://92.51.39.106:8060/guestbook.php> в поле комментария размещаем скрипт для вектора атаки stored-XSS <script>alert(document.cookie)</script>

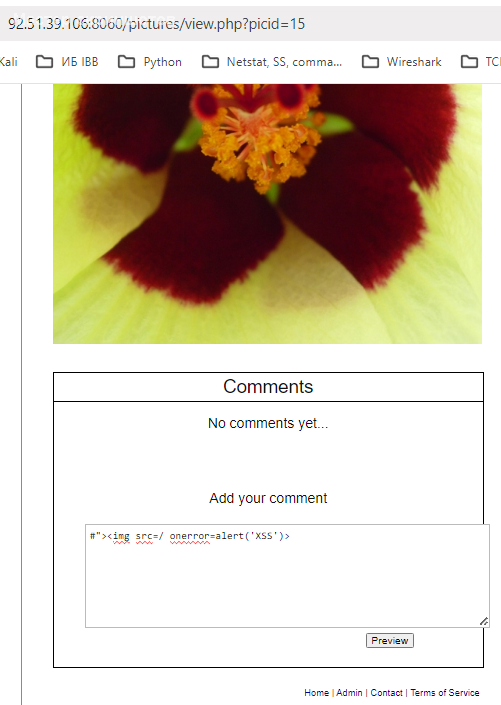
В дальнейшем, при заходе на данную страниц, будет срабатывать XSS, в данном случае предоставление сессионного cookie пользователя.



Далее заходим на главную страницу http://92.51.39.106:8060/ и в поле отправить файл пишем вектор атаки DOM-based XSS: #"><img src=/ onerror=alert(document.cookie)>



Заходим на любую страницу, где есть текстовое поле и применяем вектор атаки XSS, например:



### 7.  ****Уязвимость к BruteForce атакам. (****[A07:2021-Identification and Authentication Failures](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/)****)****

**Критичность:**Средняя

**Страница:** <http://92.51.39.106:8060/users/login.php>

**Описание:** При авторизации в приложении нет ограничений на количество попыток ввода паролей пользователей, что открывает возможность к перебору пароля от известного пользователя или подбору комбинации логина и пароля.

Существует возможность выполнения следующих действий:

* Подбор пароля методом "грубой силы"

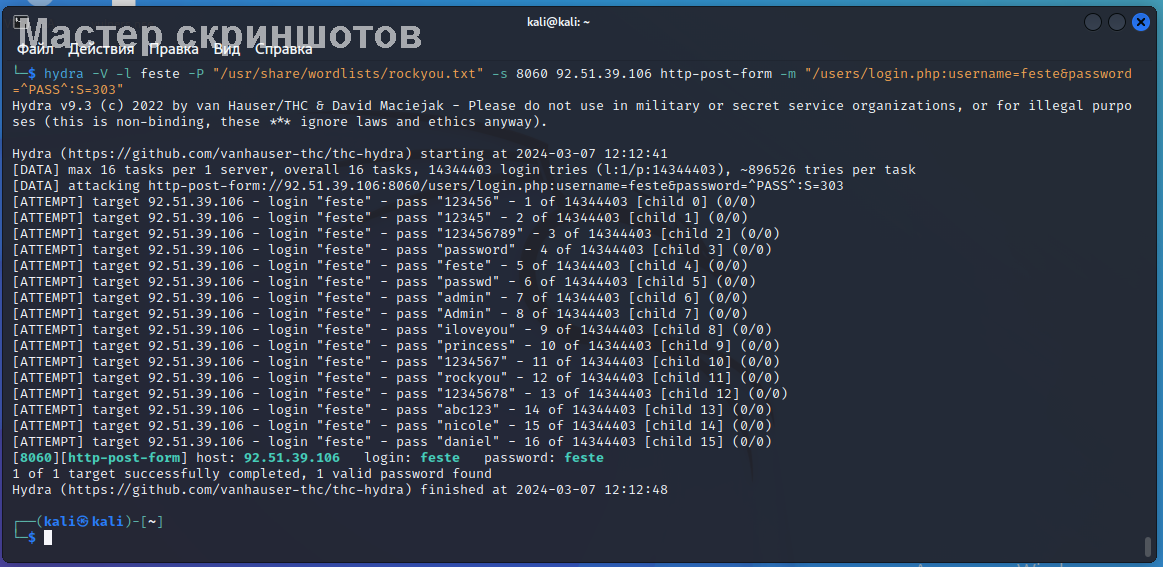
**Предложения по исправлению:**

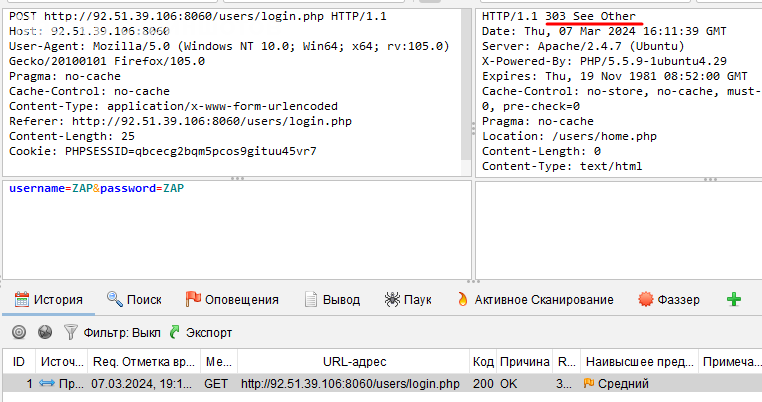
* Установить ограничение попыток ввода пароля
* Установить ограничение попыток авторизации по IP-адресу

Реализация:

Адрес страницы, на которой нам надо подобрать логин/пароль (в форме авторизации) используем в инструменте Gydra. Прописываем следующую команду, которая позволит подобрать пароль к известному нам пользователю.

**$ hydra** -V -l feste -P "/usr/share/wordlists/rockyou.txt" -s 8060 92.51.39.106 http-post-form -m "/users/login.php:username=feste&password=^PASS^:S=303"

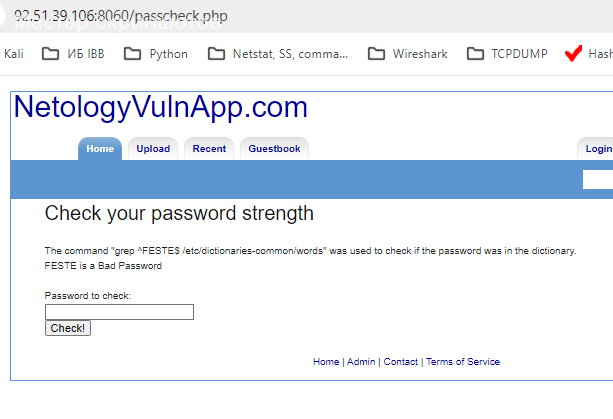


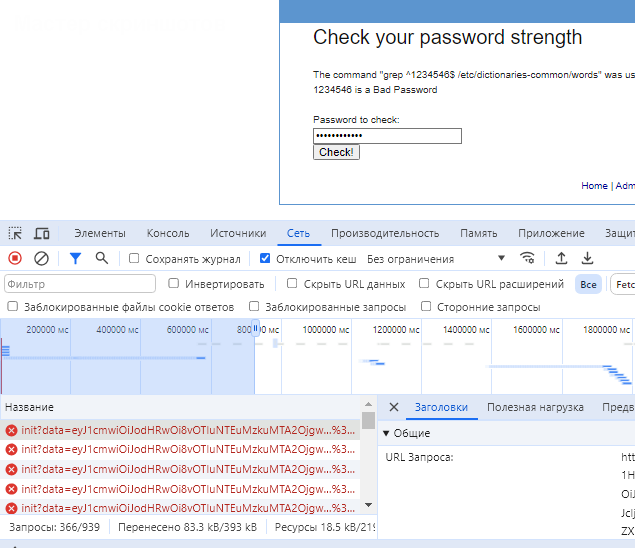
Чтобы прописать условие, которое подтвердит правильность подобранного пароля, смотрим, какой ответ приходит от сервера. В нашем случае это код ответа 303. Его и прописываем в условия: S=303 .

### 8. ****Уязвимости Denial of Service, Command Injection (****[A05:2021-Security Misconfiguration](https://owasp.org/Top10/A05_2021-Security_Misconfiguration/)****,****[A03:2021-Injection](https://owasp.org/Top10/A03_2021-Injection/)****)****

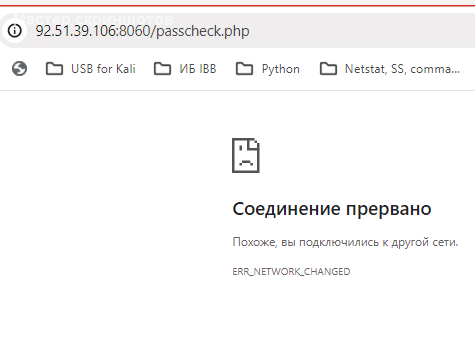
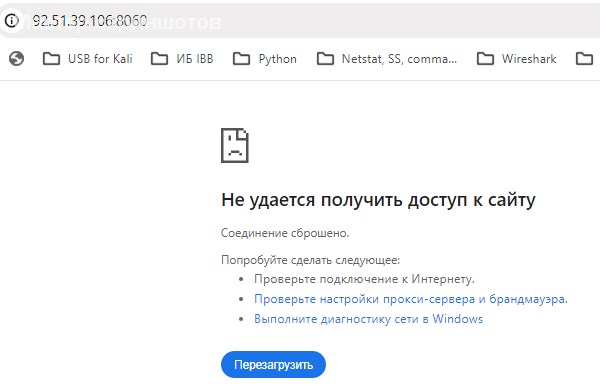
**Критичность:** Высокая

**Страница**: <http://92.51.39.106:8060/passcheck.php>

  
**Описание**: На странице есть возможность проверить надежность пароля. При этом пароль проверяется по файлу паролей в операционной системе. Для чтения файла используется метод PHP exec с использованием пользовательского ввода в сыром ввиде.

Попробуем повлиять на команду и введем один из спец. символов &, &&, |, || для образования pipeline (конвейера) команд. test | pwd

После отправки запроса на сервер сайт будет недоступен какое-то время, что является отказом в обслуживании. (Denial of Service)



## Beemer (port: 7799)

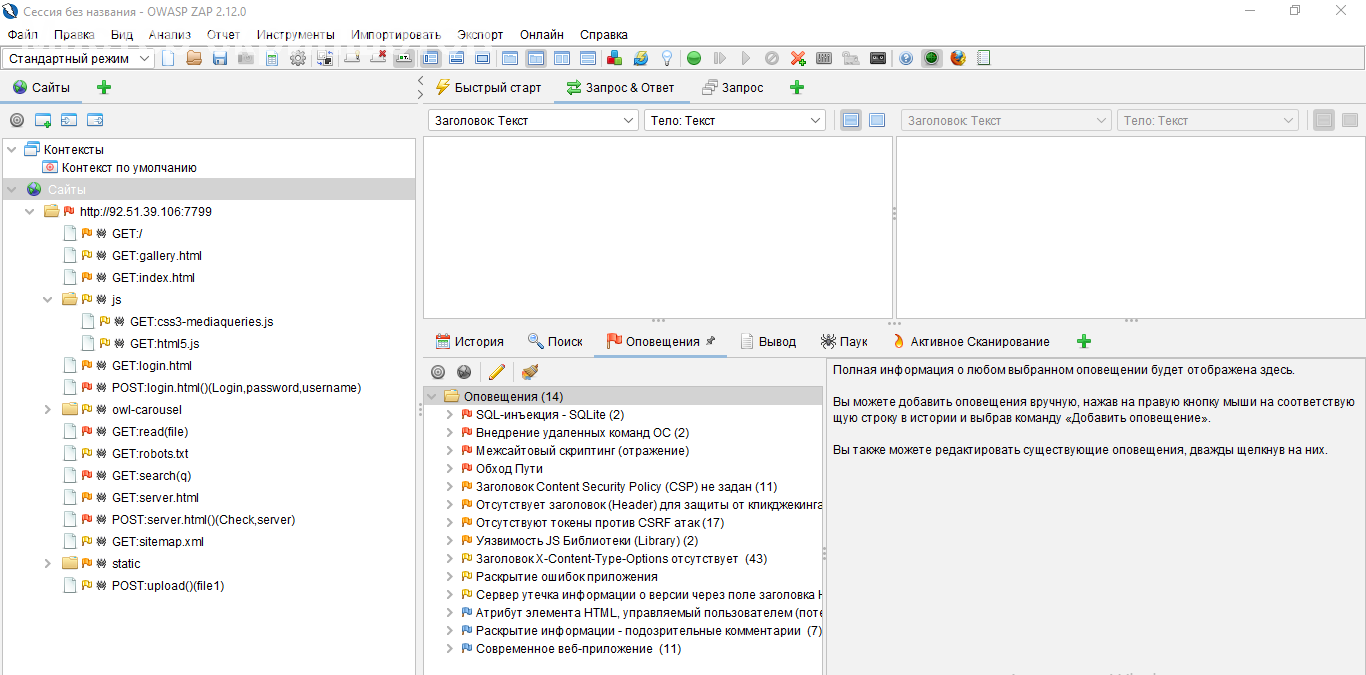
Тестирование приложения Beemer проводилось средствами

* DirSearch
* OWASP ZAP
* Kali linux
* hydra

В результате тестирования были найдены следующие проблемы:

* SQL-инъекция - SQLite
* Внедрение удаленных команд ОС
* Межсайтовый скриптинг (отражение)
* Обход Пути
* Заголовок Content Security Policy (CSP) не задан
* Отсутствует заголовок (Header) для защиты от кликджекинга
* Отсутствуют токены против CSRF атак
* Уязвимость JS Библиотеки (Library)
* Заголовок X-Content-Type-Options отсутствует
* Раскрытие ошибок приложения
* Сервер утечка информации о версии через поле заголовка HTTP-ответа «Server»

### OWASP ZAP сканирование Beemer.



### 1. ****Уязвимость****[SQL Injection](https://owasp.org/www-community/attacks/SQL_Injection)****(****[A03:2021-Injection](https://owasp.org/Top10/A03_2021-Injection/)****)****

**Критичность:** Высокая  
**Страница:** <http://92.51.39.106:7799/login.html>  
**Описание:** Через форму авторизации пользователей есть возможность внедрить sql скрипт в поле логина.

Существует возможность выполнения следующих действий:

* Добавление, изменение данных в таблицах
* Удаление данных из таблиц
* Нарушение схемы БД

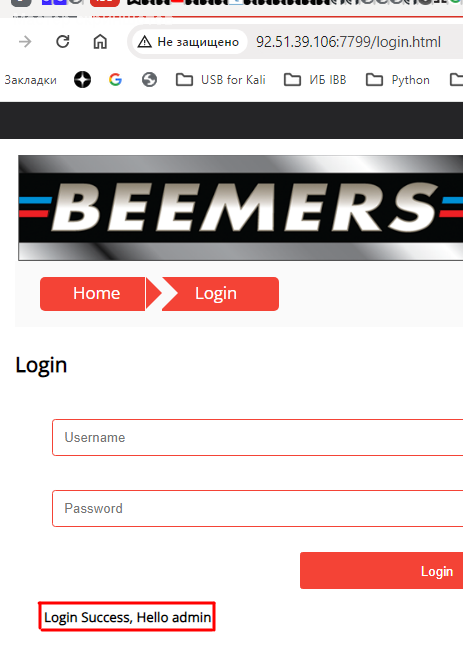
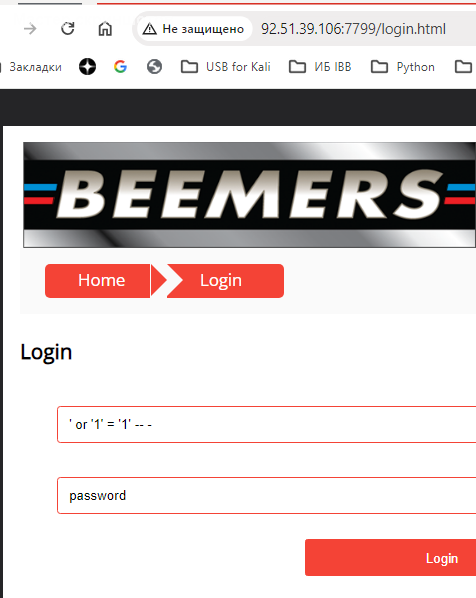
**Предложения по исправлению:**

* Добавить валидацию, санитизацию входных данных с формы
* Экранировать все данные, полученные от клиента
* Примените «список разрешенных» разрешенных символов или «список запрещенных» символов при вводе пользователем
* Примените принцип наименьших привилегий, используя минимально возможного пользователя базы данных
* не используйте «exec» или эквивалентные функции
* Предоставьте минимальный доступ к базе данных, необходимый приложению
* Не создавайте динамические SQL-запросы, используя простую конкатенацию строк

**Реализация:**

На странице <http://92.51.39.106:7799/login.html> , в поле логин были использованы два варианта кода sql:

[' or '1' = '1' -- -] - В результате чего удалось пройти авторизацию под пользователем “admin”



[ ' ( ] – В результате чего сервер выдал ошибку «syntax error», подтверждающую возможность проведения sqli.

### 2. ****Уязвимость к XSS атакам (****[A03:2021-Injection](https://owasp.org/Top10/A03_2021-Injection/)****,****[Stored XSS](https://owasp.org/www-community/attacks/xss/#stored-xss-attacks)****)****

**Критичность:** Средняя

**Страницы**: http://92.51.39.106:7799/search?q=

**Payload**:

#"><img src=/ onerror=alert(document.cookie)>

<script>alert("XSS")</script>

**Описание**: На нескольких страницах происходит добавление пользовательского ввода на страницу без санитизации и экранирования существует возможность выполнения следующих действий:

* Кража сессионной куки
* Перенаправление пользователей на сторонние сайты
* Выполнение XSRF атак на другие сайты в этой страницы

**Предложения по исправлению**:

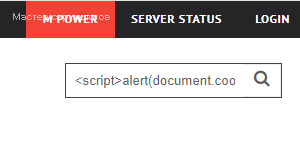
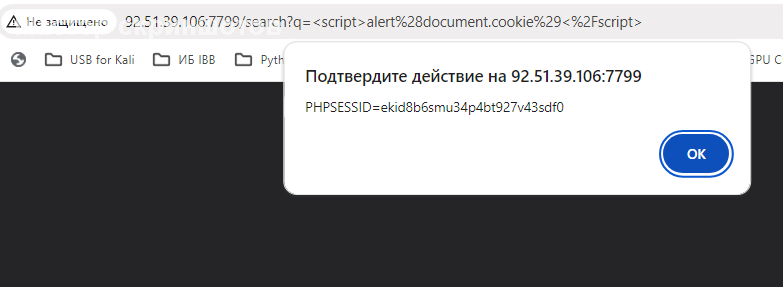
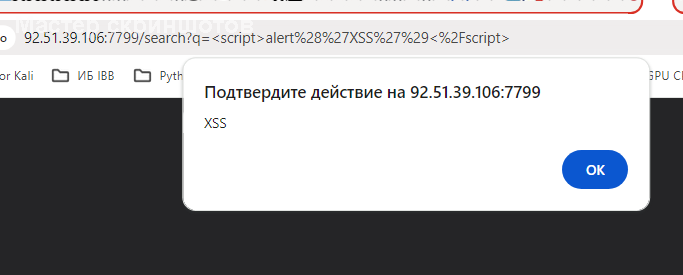
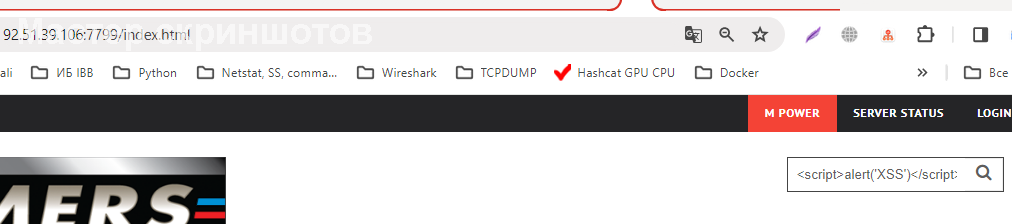
* Добавить валидацию/санитизацию пользовательского ввода.
* Установите для файла cookie сеанса значение HttpOnly (этот атрибут может предотвратить доступ к cookie сеанса пользователя для вредоносных клиентских скриптов, использующих document.cookie. Это не полное решение, поскольку HttpOnly поддерживается не всеми браузерами. Что еще более важно, XMLHTTPRequest и другие мощные браузерные технологии предоставляют доступ для чтения к заголовкам HTTP, включая заголовок Set-Cookie, в котором установлен флаг HttpOnly.)
* Используйте разрешенный список допустимых входных данных, которые строго соответствуют спецификациям
* Если возможно, используйте структурированные механизмы, которые автоматически обеспечивают разделение данных и кода
* Используйте проверенную библиотеку или платформу которая не позволяет этому недостатку возникать или предоставляет конструкции которые позволяют обойти эту угрозу. (Примеры библиотек и фреймворков, которые упрощают создание правильно закодированного вывода, включают библиотеку Microsoft Anti-XSS, модуль кодирования OWASP ESAPI и Apache Wicket).

Заходим на главную страницу http://92.51.39.106:7799/index.html, заполняем поисковое поле и нажать кнопку Enter.

Используем при этом вектор атаки для тестирования XSS

* #"><img src=/ onerror=alert(document.cookie)>
* <script>alert("XSS")</script>

Далее мы будем перенаправлены на уязвимую страницу http://92.51.39.106:7799/search?q=, где в диалоговом окне увидим выполненную XSS атаку.



### 3. Уязвимости Command Injection ([A05:2021-Security Misconfiguration](https://owasp.org/Top10/A05_2021-Security_Misconfiguration/), [A03:2021-Injection](https://owasp.org/Top10/A03_2021-Injection/)****)****

**Broken access control**

**Критичность:** Высокая

**Страница**: http://92.51.39.106:7799/server.html

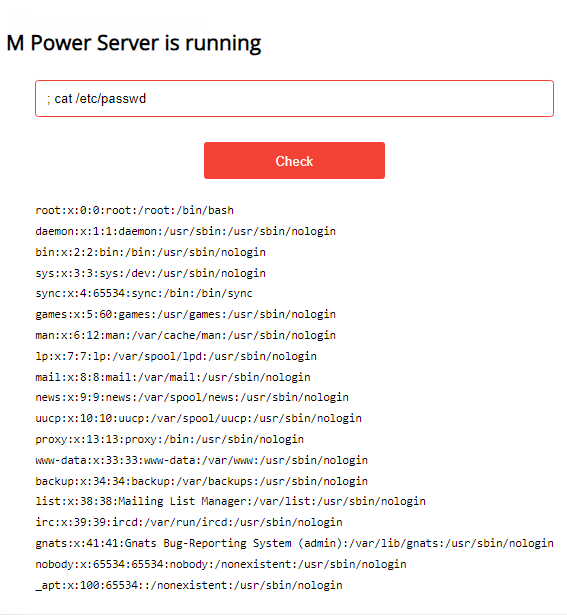
**Описание**: На странице есть функционал проверки доступности сервера. Сервис проверяет отправляет ping запрос из консоли для введенного ip-адреса. При этом пользовательский ввод никак не проверяется. Есть возможность отправить shell-код, который будет выполнен на сервере и результат выполнения будет возвращен в UI.

**Предложения по исправлению**:

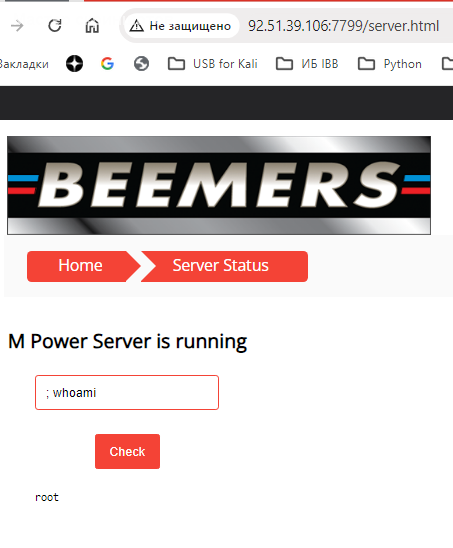
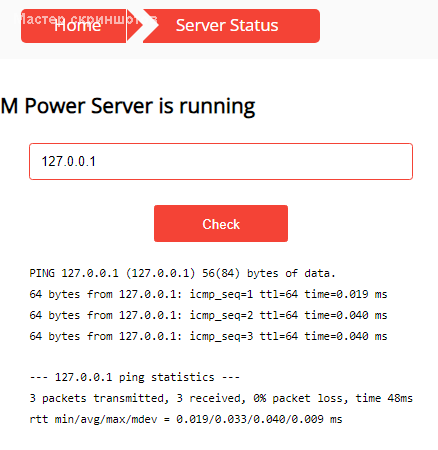
* Использовать санитизацию и экранирование пользовательского ввода.

**Реализация**

Заходим на исследуемую страницу и вводим в поле ввода следующие команды:

 [127.0.0.1] [;cat /etc/passwd], [;whoami] которые выведут пинг, покажут файл пользователей, и покажут под каким пользователем работаем. После выполнения команд, на сервере будет отображено содержимое:

В ответе сервера получаем, что приложение работает под пользователем root, а значит наши права на сервере неограничены.



### 4. Уязвимость Path Traversal (A01:2021-Broken Access Control)

**Критичность:** Высокая

**Страница:** http://92.51.39.106:7799/read?file=

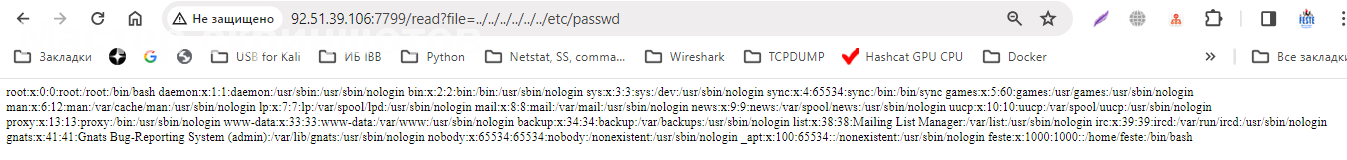
**Описание:** Существует возможность обращения к файловой системе через параметр GET запроса.

**Предложения по исправлению:**

Валидация значений параметров запросов.

**Реализация**

Перейти на страницу http://92.51.39.106:7799/read?file=../../../../../../etc/passwd, в ответ сервер отобразит содержимое файла /etc/passwd



### ****5. Уязвимость к BruteForce атакам. (****[A07:2021-Identification and Authentication Failures](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/)****)****

**Критичность:** Средняя  
**Страница:** http://92.51.39.106:7799/login.html  
**Описание:** При авторизации в приложении нет ограничений на количество попыток ввода паролей пользователей, что открывает возможность к перебору пароля от известного пользователя или подбору комбинации логина и пароля.  
Существует возможность выполнения следующих действий:

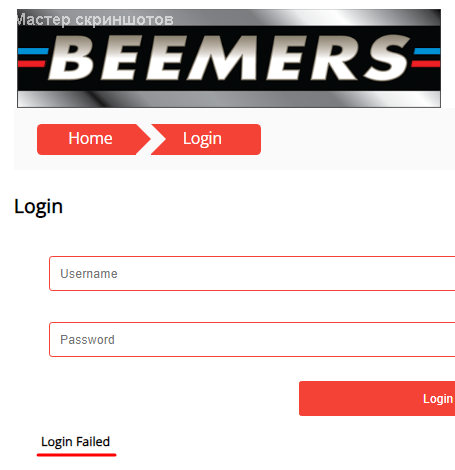
* Подбор пароля методом "грубой силы"

**Предложения по исправлению:**

* Установить ограничение попыток ввода пароля
* Установить ограничение попыток авторизации по IP-адресу

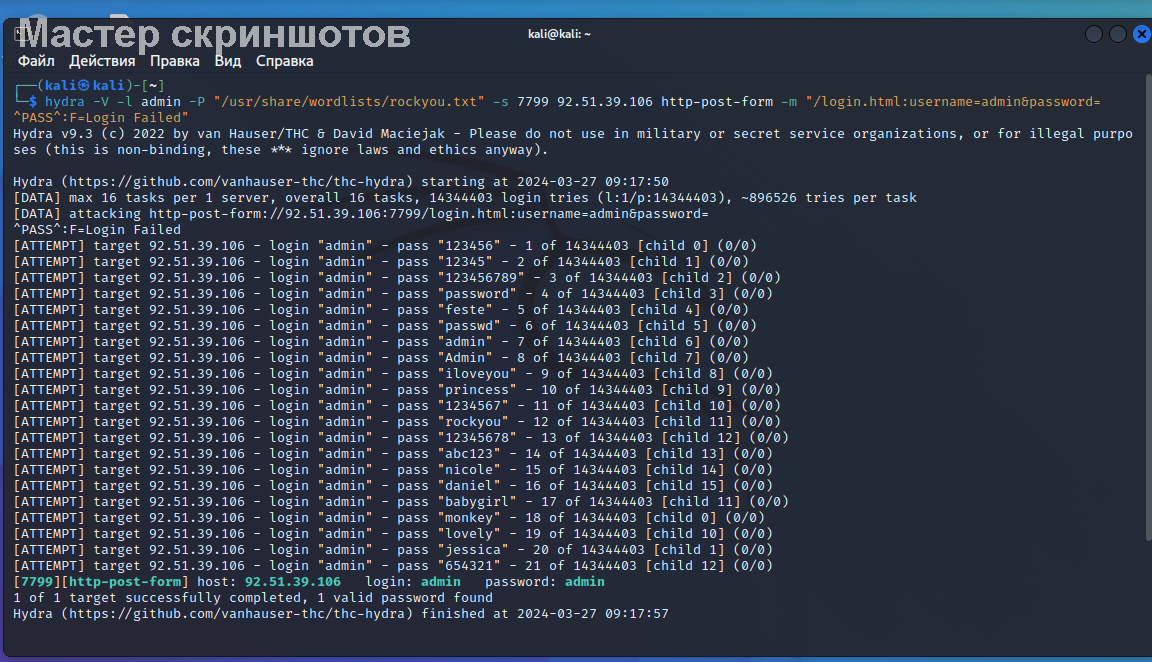
**Реализация**

Адрес страницы, на которой нам надо подобрать логин/пароль (в форме авторизации) используем в инструменте Gydra.

 Чтобы прописать условие, которое подтвердит правильность подобранного пароля, смотрим, какой ответ приходит от сервера при неправильном вводе пароля. Его и прописываем в условие: F= Login Failed

Прописываем следующую команду, которая позволит подобрать пароль к известному нам пользователю.

$hydra -V -l admin -P "/usr/share/wordlists/rockyou.txt" -s 7799 92.51.39.106 http-post-form -m "/login.html:username=admin&password=^PASS^:F=Login Failed"



### 6. ****Уязвимость****[Unrestricted File Upload](https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload)****. (****[A03:2021-Injection](https://owasp.org/Top10/A03_2021-Injection/)****)****

**Критичность:** **Высокая**

**Страница**: http://92.51.39.106:7799/index.html

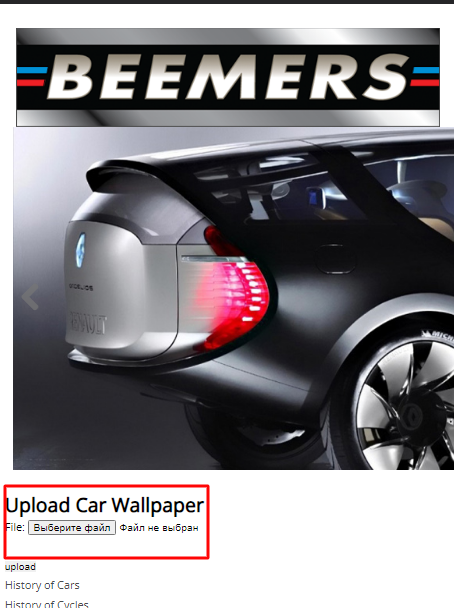
**Описание**: Уязвимость позволяет загрузить произвольный файл, отличный от картинки. Существует возможность выполнения следующих действий:

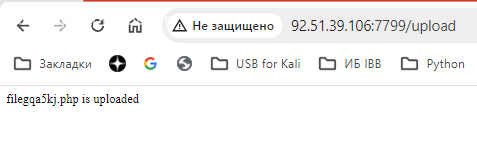
* Загрузить PHP-Shell файл

**Предложения по исправлению**:

* Добавить валидацию входного по типу содержимого. Например, можно использовать сигнатурный анализ файла и сравнивать первые байты файлов с известными сигнатурами. Например, сигнатура для файлов формата JPEG будет выглядеть следующим образом: FF D8 FF E0, формата PNG будет выглядеть следующим образом: 89 50 4E 47.
* Разрешить загрузку картинок определённых расширений. Например, GPEG, PNG, BMP.
* Запускать приложение под пользователем с минимальными правами. Пользователь не должен иметь прав на чтение системных файлов, тем более на их модификацию или удаление.

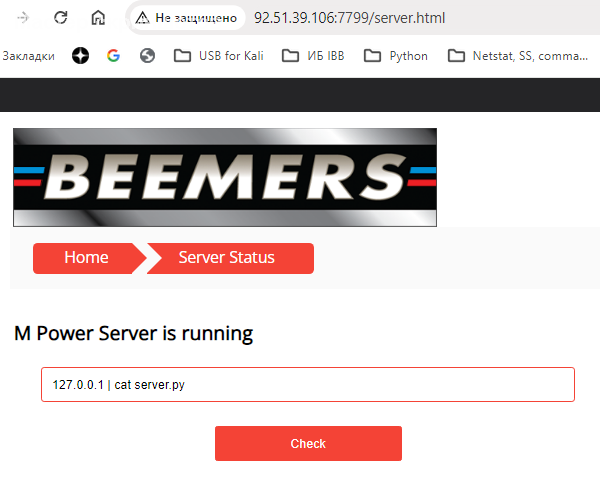
**Реализация**

Переходим на страницу загрузки файла и выбираем специальный [php-shell](https://github.com/VitaliySid/sib-Diplom-Track-Penetration-Testing/blob/main/assets/shell/php-shell.php)` файл.

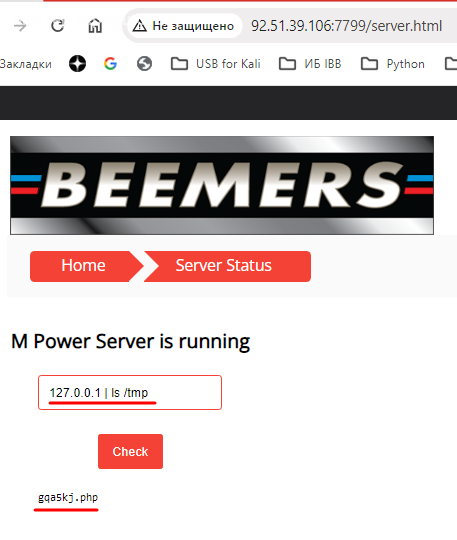
После загрузки файла открывается страница с сообщением, что файл загружен filegqa5kj.php is uploaded

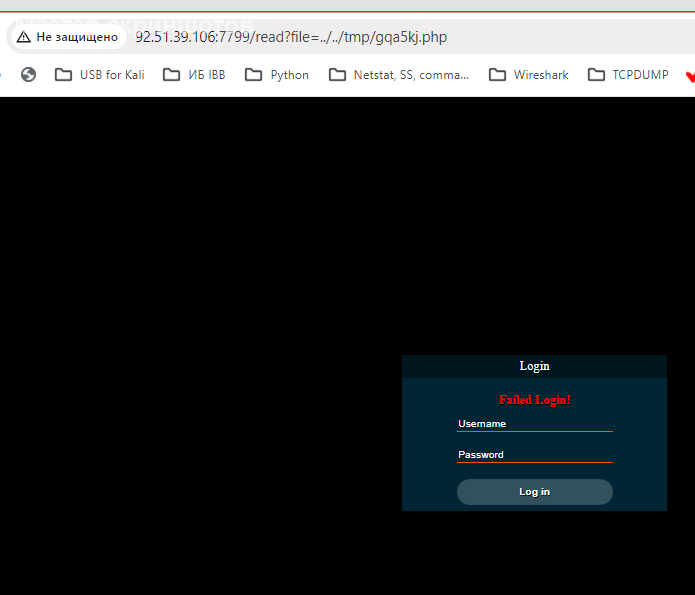
Используя уязвимость сервера, которая позволяет просматривать список файлов сервера - просматриваем код текущего приложения.

На странице http://92.51.39.106:7799/server.html используем следующую команду - 127.0.0.1 | cat server.py.



Находим раздел upload и смотрим, куда происходит загрузка файлов. По коду сервера понимаем, что пользовательские файлы попадают в папку /tmp/ а имя файла присваивается рандомно, длинной 6 символов.

Проверяем наличие нашего файла в директории /tmp/. Для этого на той же странице http://92.51.39.106:7799/server.html вводим команду 127.0.0.1 | ls /tmp. Находим имя нашего файла, оно будет без приставки file.

Используем уязвимость Path Traversal, переходим на страницу http://92.51.39.106:7799/read?file=../../tmp/gqa5kj.php и видим окно нашего shell-приложения.

# Итоги

В результате анализа тестируемого IP-адреса было обнаружено множество уязвимостей различных уровней, начиная от низкого и заканчивая критическим. Проведенные сканирование и тестирование выявили уязвимые места, которые могут быть использованы злоумышленниками для атак и получения конфиденциальной информации.

Система информационной безопасности сайта требует полной перестройки, так как текущее состояние не соответствует стандартам безопасности и представляет угрозу для пользователей. Рекомендуется немедленно приступить к обновлению системы безопасности

При реконструкции системы безопасности рекомендуется ориентироваться на стандарты безопасности информационных технологий, такие как ISO/IEC 27001:2022, ФЗ №152 (регулирующим деятельность по обработке персональных данных), COBIT, NIST Cybersecurity Framework,

При добавлении платежных систем, рекомендуется ориентироваться также на стандарты PCI DSS (стандарт безопасности платёжных систем.), GDPR (общий регламент по защите данных. Этот норматив определяет порядок сбора, обработки, хранения и распространения персональных данных в странах Евросоюза и с участием его граждан). Приказ ЦБ РФ №3832-У "О бухгалтерском учете сведений об операциях с платежными картами".

Соблюдение этих стандартов поможет обеспечить безопасность данных пользователей платежных систем и снизить риск финансовых мошенничеств, поможет обеспечить надежную защиту информации, уменьшить риск инцидентов безопасности и повысить доверие пользователей к сайту.