МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Разработка защищенных программных систем»

Тема: Поиск уязвимости в веб-приложении

Студент гр. 5303 Допира В.Е.

Преподаватель Юшкевич И.А.

Санкт-Петербург
2019

ЗАДАНИЕ

НА КУРСОВОЙ ПРОЕКТ

Студент Допира В.Е.

Группа 5303

Тема проекта: Поиск уязвимости в веб-приложении

Исходные данные:

Веб-приложение и его уязвимости

Содержание пояснительной записки:

«Введение», «Описание разработанного приложения», «Описание уязвимости», «Эксплуатация уязвимости», «Заключение».

Предполагаемый объем пояснительной записки:

Не менее 16 страниц.

Дата выдачи задания: 04.09.2019

Дата сдачи реферата: 23.12.2019

Дата защиты реферата: 23.12.2019

Студент		Допира В.Е.
Преподаватель		Юшкевич И.А.

АННОТАЦИЯ

Небезопасное программное обеспечение подрывает финансовую, медицинскую, оборонную, энергетическую и другие критически важные инфраструктуры. По мере того как программное обеспечение становится все более сложным, также возрастает и сложность обеспечения безопасности приложений. Было разработано веб-приложение с описанием услуг туристической компании, которая проводит полярные экспедиции. Обнаружена уязвимость Server-side template injection. Было похищено содержимое параметра config, а также проведена DOS атака. Предложены рекомендации по защите приложения от найденной уязвимости.

SUMMARY

Insecure software is undermining our financial, healthcare, defense, energy, and other critical infrastructure. As software becomes increasingly complex, and connected, the difficulty of achieving application security increases exponentially. A web application was developed describing the services of a travel company that conducts polar expeditions. Server-side template injection is discovered. The contents of the config parameter have been stolen. DOS attack has happened. Suggested recommendations for protecting applications from vulnerabilities found.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Небезопасное программное обеспечение подрывает финансовую, медицинскую, оборонную, энергетическую и другие критически важные инфраструктуры. По мере того как программное обеспечение становится все более сложным, также возрастает и сложность обеспечения безопасности приложений. При быстрых темпах развития современных технологий разработки программного обеспечения риски становятся более существенными. Поэтому необходимо быстро и точно обнаруживать уязвимости, а также устранять их. Особенно проблемы безопасности актуальны для веб-приложений.

Постановка задачи

Разработать веб-приложение и найти в нем уязвимость.

1. ОПИСАНИЕ РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ

1.1. Краткое описание

Разработанное приложение представляет собой описание услуг туристической компании, которая проводит полярные экспедиции. Сайт доступен только для клиентов, знающих логин и пароль для авторизации. После авторизации пользователи сайта могут узнать более подробную информацию о маршрутах экспедиций и связаться с организаторами через форму обратной связи.

Ссылка на разработанное приложение: https://github.com/valeriefuerte/web_exploit

1.2. Используемые технологии

- Frontend: JQuery, JavaScript, CSS;
- Backend: Python, Flask, SQLite.

Приложение разработано с использованием интегрированной среды разработки PyCharm.

1.3. Внешний вид

На рисунках 1, 2, 3 представлены главная страница, страница с более детальным описанием и страница логина соответственно.

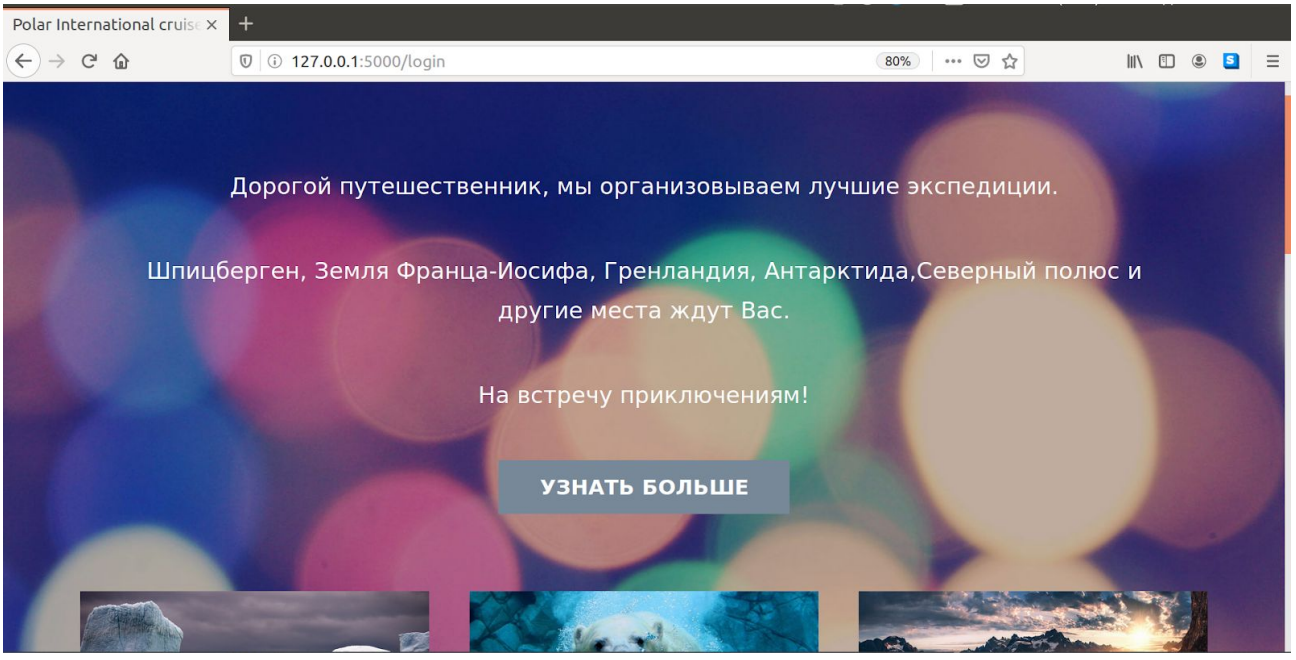

Рисунок 1 - Главная страница

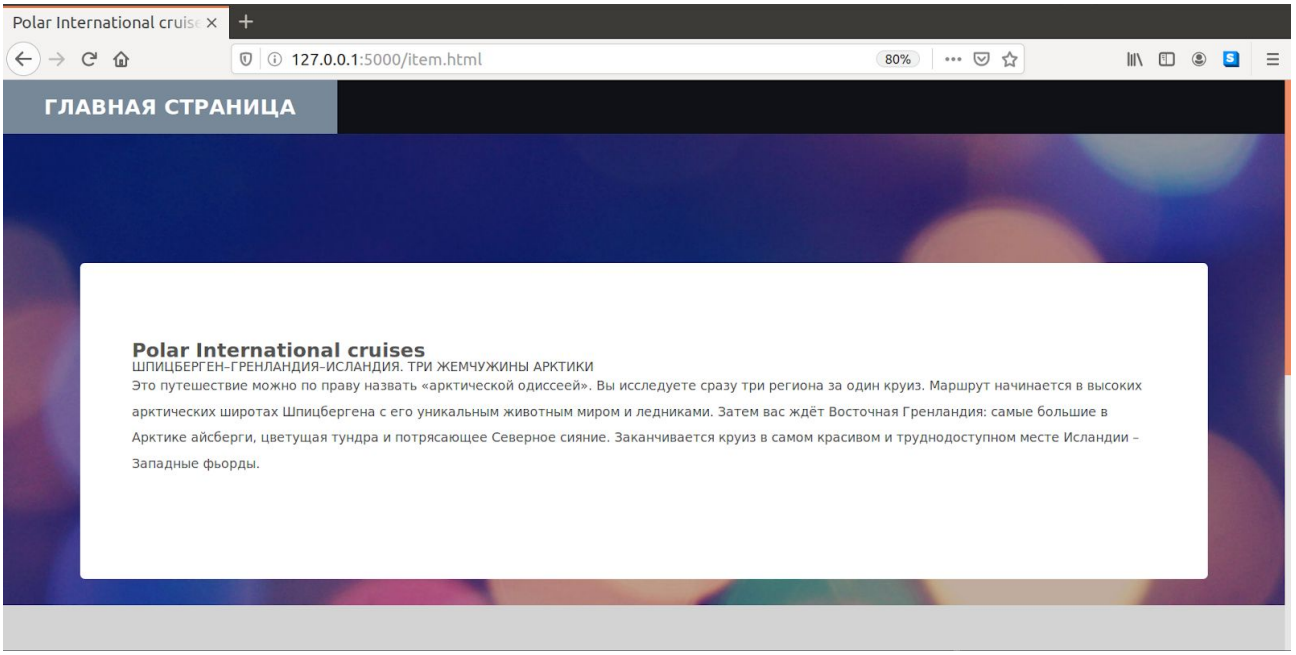

Рисунок 2 - Страница с более подробным описанием

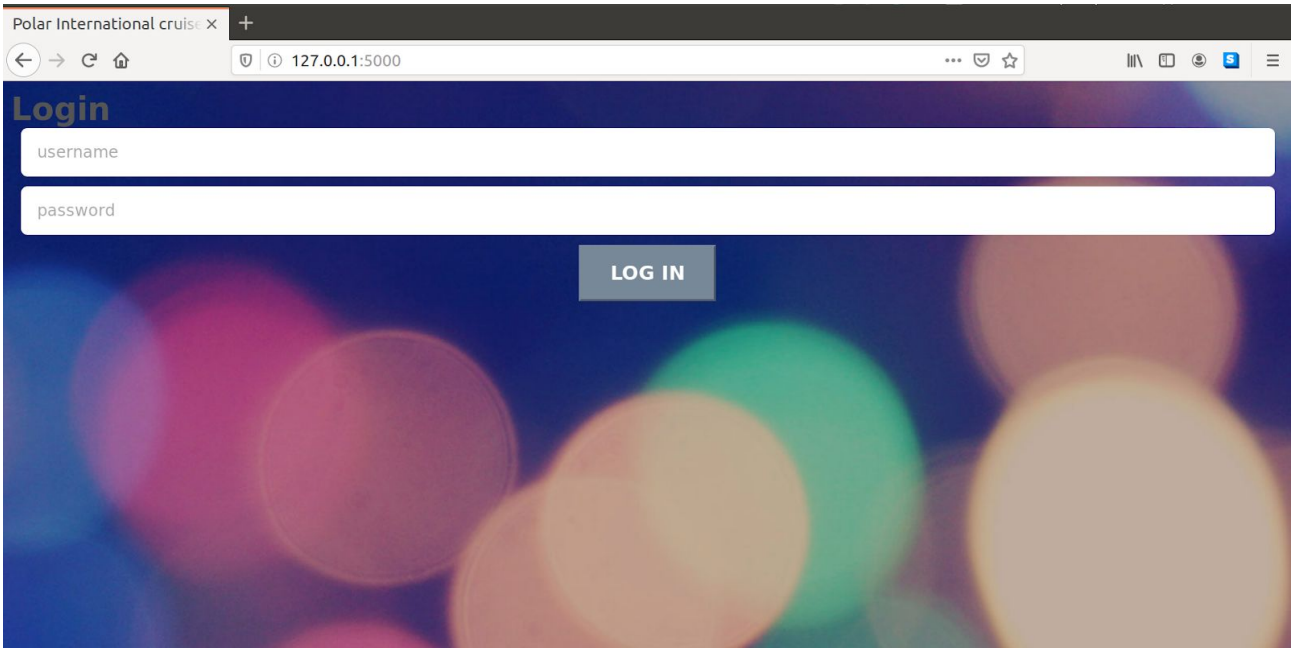

Рисунок 3 - Страница логина

2. ОПИСАНИЕ УЯЗВИМОСТИ

Внедрение шаблона на стороне сервера (Server-side template injection, SSTI) происходит, когда пользовательский ввод небезопасно встроен в шаблон на стороне сервера, что позволяет пользователям вводить директивы шаблона. Используя вредоносные директивы шаблона, злоумышленник может выполнить произвольный код и получить полный контроль над веб-сервером.

Серьезность этой проблемы варьируется в зависимости от типа используемого механизма шаблонов. Шаблонные движки варьируются от тривиальных до почти невозможных для использования. Следующие шаги должны быть использованы при попытке разработки эксплойта:

- Определите тип используемого шаблонизатора.
- Просмотрите его документацию на предмет базового синтаксиса, соображений безопасности, а также встроенных методов и переменных.
- Исследуйте шаблон среды и нанесите на карту поверхность атаки.
- Аудит каждого выставленного объекта и метода.

Уязвимости внедрения шаблонов могут быть очень серьезными и могут привести к полной компрометации данных и функциональности приложения, а также часто сервера, на котором размещается приложение. Также возможно использовать сервер в качестве платформы для дальнейших атак на другие системы. С другой стороны, некоторые уязвимости внедрения шаблонов могут не представлять значительного риска для безопасности.

3. ЭКСПЛУАТАЦИЯ УЯЗВИМОСТИ

3.1. Вход на сайт

Чтобы зайти на сайт, необходимо подобрать логин и пароль. Методом подбора угаданы стандартные: логин - admin, пароль - password (см рис 4).

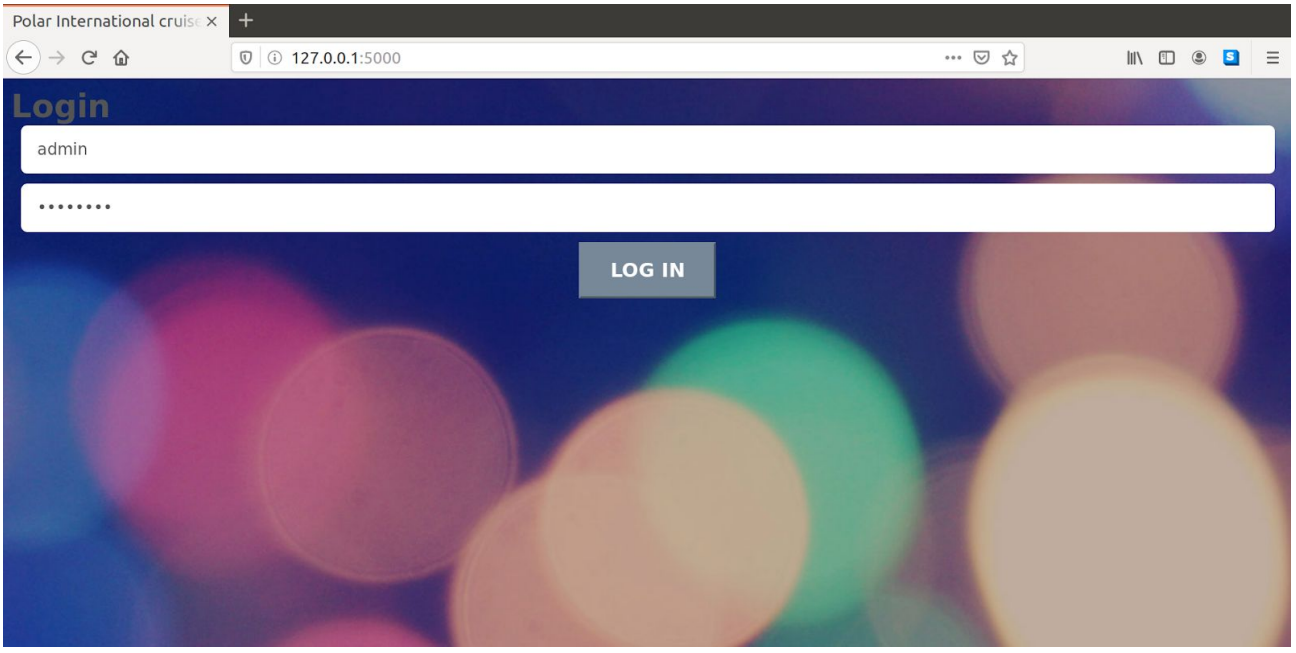

Рисунок 4 - Страница логина

3.2. Определение используемых технологий

Выполнено сканирование хоста 127.0.0.1 (localhost) с помощью nmap. Выполнена команда: “nmap -A 127.0.0.1”. Параметр “-A” используется для обнаружения операционной системы, версий используемых фреймворков, сканирования скриптов и трассировки маршрута.

Получен ответ (см рис 5):

```
valeria@valeria:~$ nmap -A 127.0.0.1

Starting Nmap 7.01 ( https://nmap.org ) at 2019-12-22 16:59 MSK
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000079s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE VERSION
631/tcp  open  ipp     CUPS 2.1
| http-methods:
|_  Potentially risky methods: PUT
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: CUPS/2.1 IPP/2.1
|_http-title: Home - CUPS 2.1.3
5000/tcp open  http    Werkzeug httpd 0.16.0 (Python 3.6.0)
|_http-server-header: Werkzeug/0.16.0 Python/3.6.0
|_http-title: Polar International cruises

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.70 seconds
valeria@valeria:~$
```

Рисунок 5 - Сканирование

На 5000 порту используется Python/3.6.0. Тогда веб-приложение написано с использованием Flask или Django. Предположено, что первый. Необходимо произвести поиск уязвимостей для приложений, разработанных на Flask.

3.3. Поиск уязвимостей

Одна из самых распространенных потенциальных проблем во фреймворке Flask связана с шаблонами на стороне сервера. Поэтому может обнаружиться SSTI (Server-Side Template Injection) уязвимость.

Путем изучения сайта и перехода по разным ссылкам, обнаружено, что разработчик создал свое отображение страницы для ошибки 404 (см рис 6).

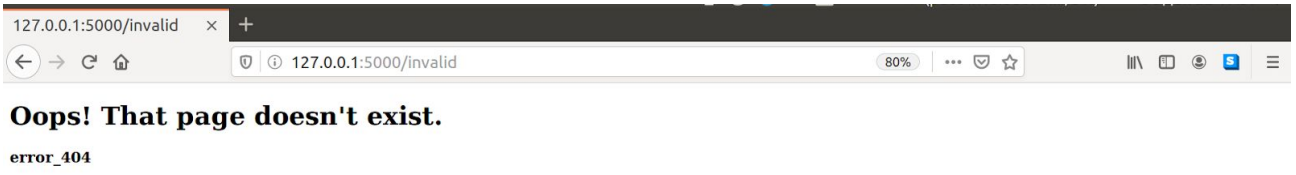

Рисунок 6 - Страница 404

Далее изучена документация по использованию шаблонов во Flask. Предположено, что разработчик решил отказаться от отдельного шаблона под 404 страницу, а создал строку для шаблона внутри функции, отвечающего за

отображение 404 страницы. При возникновении ошибки пользователю возвращается название ошибки: ‘error_404’. Тогда следует попробовать дописать в конец URL математическое выражение, которое должно вычисляться движком шаблона: ‘{{7*7}}’ (см рис 7).

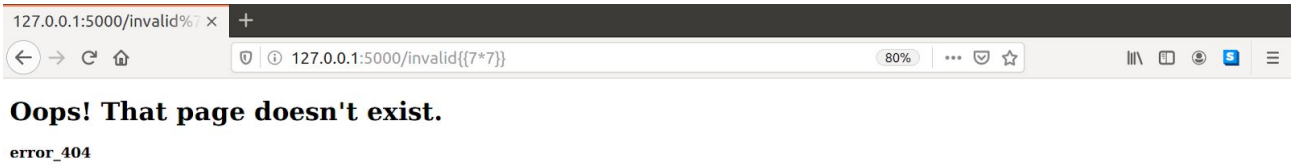

Рисунок 7 - Попытка вычисления выражения

Выражение проигнорировано. Тогда необходимо снова обратиться к документации. Функции шаблонов также могут принимать на вход различные параметры, которые следует перебрать. Если использовать стандартный ‘name’, то будет обнаружена уязвимость (см рис 8, 9).

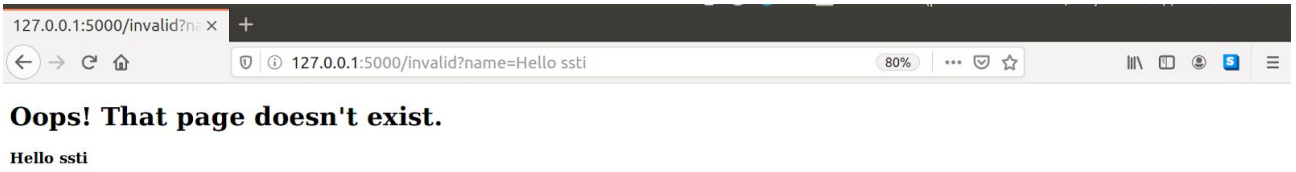

Рисунок 8 - Обнаружение уязвимости

Рисунок 9 - Вычисление выражения

Также наличие XSS может указывать на SSTI-уязвимость. Для проверки в конце строки добавлено: ‘<script>alert(42)</script>’ (см рис 10).

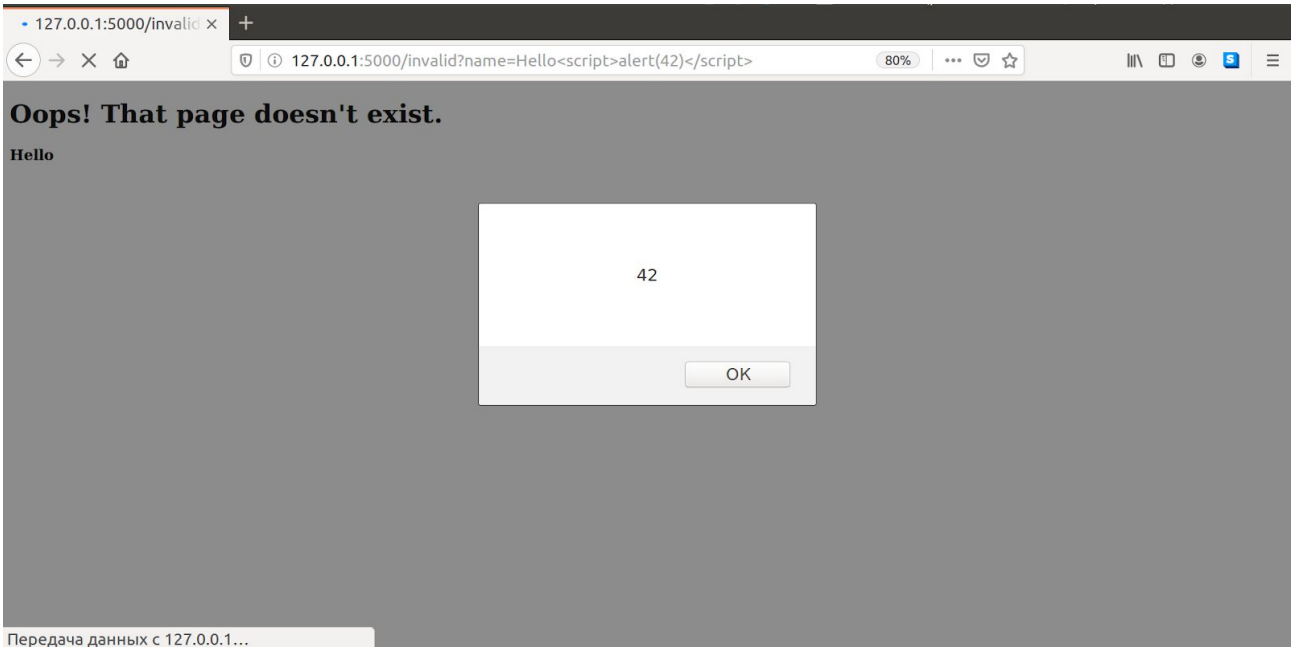

Рисунок 10 - Проверка на XSS

Присутствие XSS-уязвимости является индикатором возможности применения SSTI. Веб-приложение уязвимо.

3.4. Эксплуатация уязвимости

Есть несколько источников, из которых объекты попадают в контекст шаблона: из контекста, глобальные переменные. При этом могут использоваться чувствительные переменные, добавленные разработчиком, что упрощает SSTI. Следующие глобальные переменные доступны в шаблонах по умолчанию:

- config - текущий объект конфигурации
- request - текущий объект запроса
- session - текущий объект сеанса
- g - связанный с запросом объект для глобальных переменных (обычно используется разработчиком для хранения ресурсов во время запроса).

Объект ‘config’ является глобальным в фреймворке Flask (flask.config). Данный объект представляет собой словарь со всеми переменными, связанными с конфигурацией приложения, в том числе строками для подключения к базе данных, учетными записями к сторонним сервисам,

SECRET_KEY и т. д. Просмотр этих переменных осуществляется при помощи выражения: ‘{{config.items()}}’ (см рис 11).

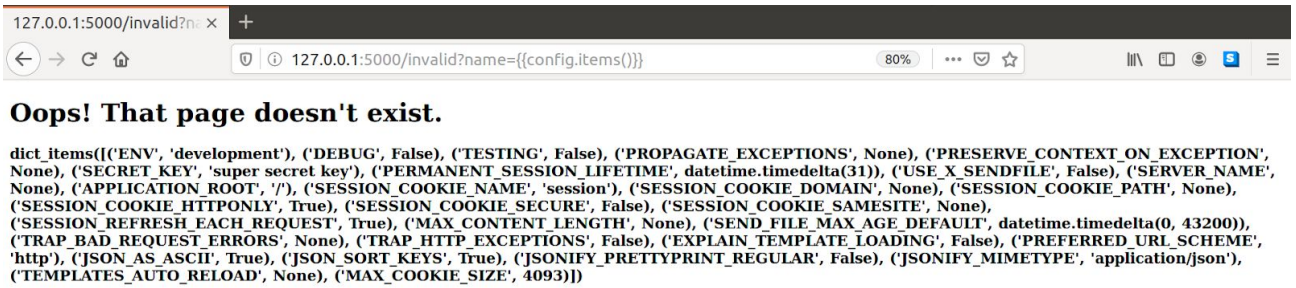

Рисунок 11 - Config

Если бы разработчик перенес эти данные в переменных среды окружения, то приложение все еще было бы уязвимо, так как объект config содержит все переменные, связанные с конфигурацией после обработки фреймворком.

Метод from_object добавляет все атрибуты импортированных модулей, чьи имена переменных полностью находятся в верхнем регистре, в объект config. Атрибуты, добавленные в объект config, поддерживают собственные типы. Добавлены атрибуты библиотеки os с помощью команды:

http://127.0.0.1:5000/invalid?name=123 {{config.from_object('os')}}

Снова просмотрены переменные: ‘{{config.items()}}’ (см рис 12).

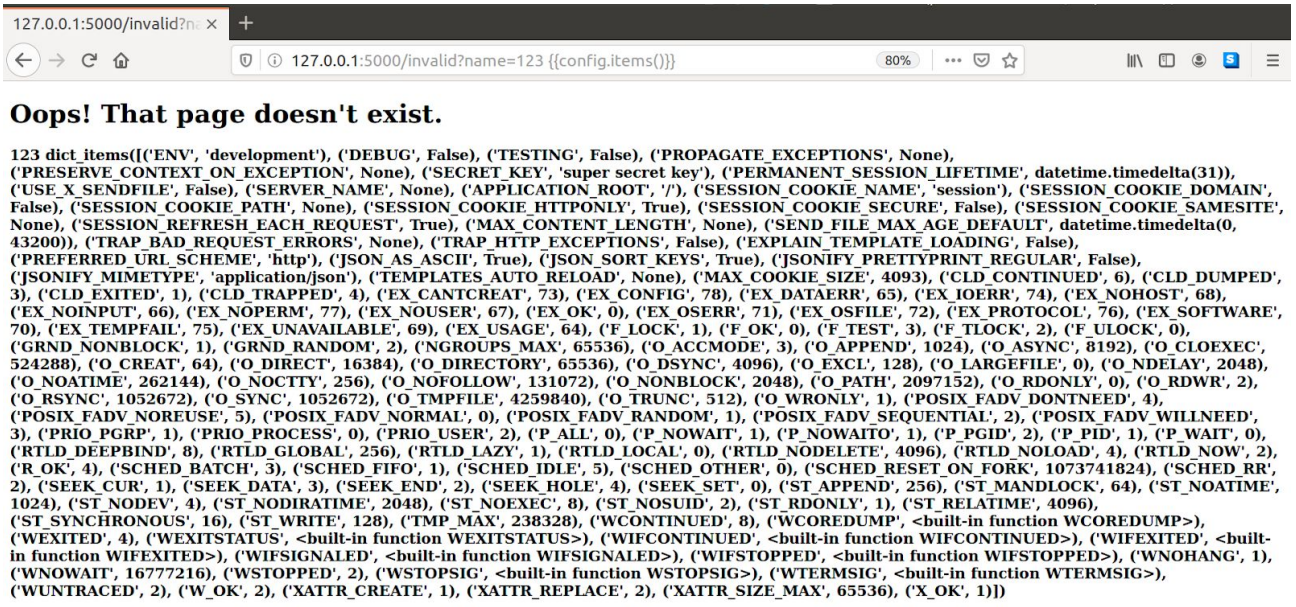

Рисунок 12 - Config после импортирования

Любой объект, добавленный в config, можно вызвать через SSTI - уязвимость, при условии, что сам элемент способен быть вызванным.

Было похищено содержимое config.

Снова необходимо обратиться к документации шаблонов в Flask. Интересен объект ‘request’, который является глобальным в фреймворке Flask (flask.request). Внутри объекта request находится объект environ, который представляет собой словарь объектов, имеющих отношение к серверной части. Один из элементов этого словаря – метод ‘shutdown_server’, который связан с ключом ‘werkzeug.server.shutdown’. Тогда в шаблон инжектировано выражение: {{request.environ['werkzeug.server.shutdown']()}} (см рис 13).

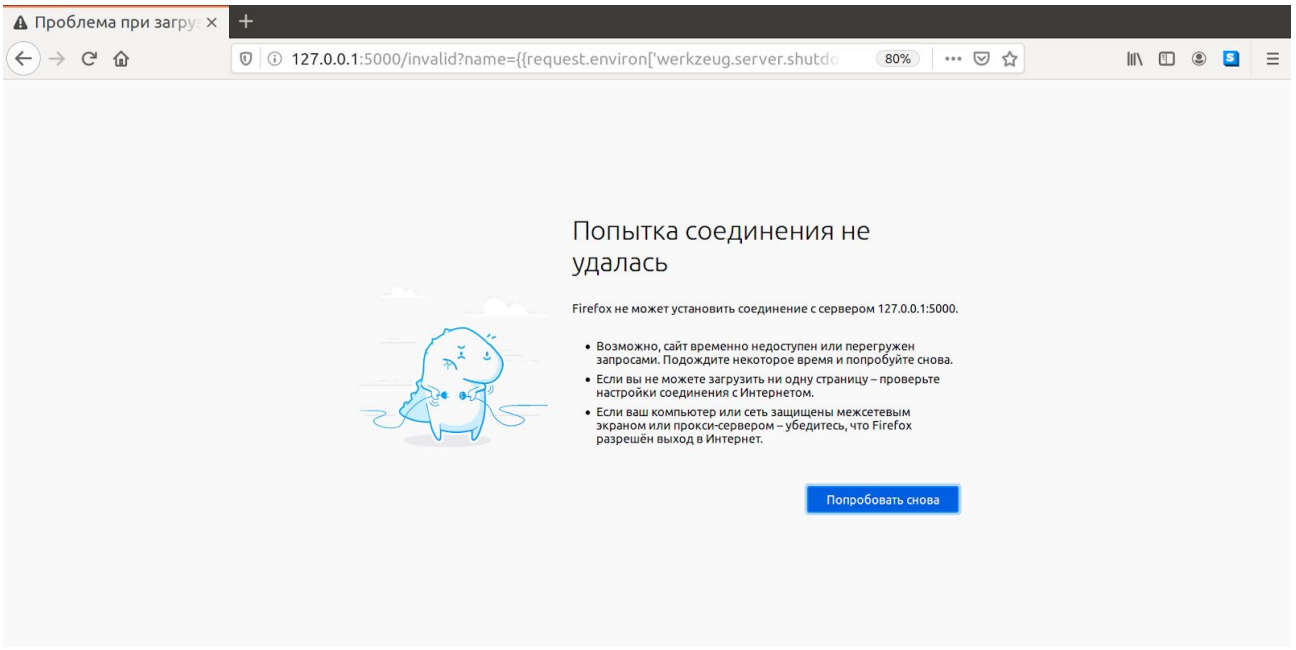

Рисунок 13 - Проверка на XSS

Была спровоцирована DOS-атака.

4. РЕКОМЕНДАЦИИ ПО ЗАЩИТЕ САЙТА

Во-первых, для входа на сайт следует использовать более надежные пароли.

Ниже приведены рекомендации для предотвращения SSTI-уязвимости.

Избегайте создания шаблонов из пользовательского ввода. Передача пользовательского ввода в шаблоны в качестве параметров обычно является безопасной альтернативой.

Если поддержка пользовательских шаблонов является бизнес-требованием, рассмотрите возможность использования простого без логического механизма шаблонов, такого как Mustache, или механизма, предоставляемого родным языком, таким как шаблон Python. Если это не подходит, то просмотрите документацию выбранного механизма шаблонов для повышения безопасности и рассмотрите возможность рендеринга шаблона в изолированной среде выполнения.

ЗАКЛЮЧЕНИЕ

В ходе работы было разработано веб-приложение, а также на нем найдена уязвимость. Она была эксплуатирована. Для защиты необходимо использовать более надежный логин и пароль при входе на сайт, а также не использовать шаблонов из пользовательского ввода.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Python Flask Tutorials. URL: https://pythonspot.com/python-flask-tutorials/
2. Flask Templates. URL: https://flask.palletsprojects.com/en/1.0.x/templating/#standard-context
3. Template Designer Documentation. URL: https://jinja.palletsprojects.com/en/master/templates/#builtin-globals
4. Server-Side Template Injection: RCE for the modern webapp. URL: https://www.blackhat.com/docs/us-15/materials/us-15-Kettle-Server-Side-Template-Injection-RCE-For-The-Modern-Web-App-wp.pdf
5. Server-side template injection. URL: https://portswigger.net/kb/issues/00101080_server-side-template-injection