ДОКУМЕНТАЦИЯ

на курсова работа

ПО ДИСЦИПЛИНА

ПРОГРАМИРАНЕ ЗА РАЗПРЕДЕЛЕНИ СРЕДИ

Тема: Billion Laughs Attack

Факултет: Компютърни системи и технологии
Специалност: Компютърно и софтуерно инженерство

Изготвил: Георги Александров Стефанов
Фак. номер: 121220015
Група: 43

Асистент: ас. Петър Маринов

Съдържание

Задание

Обобщено задание:

1. Реализирайте функционалност, специфична за XML формата, която няма директен аналог в JSON (на среда по избор)
2. Реализирайте WEB услуга, която изпраща обработения в т.1 файл с реализирана идентификация и/или автентификация (на среда по избор)

(изпълняват се и двете точки)

Детайлно задание:

1. Намерете функционалност свързана с XML, която няма директен аналог в JSON
    a. не че не може да се направи с JSON, но не може под същата форма
    b. Attributes не се зачита, защото се разглежда на упражнения
2. Намерете готов XML файл или създайте подходящ XML файл с достатъчно данни, върху който може да се демонстрира функционалността (от т.1).
3. Създайте програма (на среда по избор), която да изпълнява функционалността (от т.1) върху демонстрационния файл (от т.2)
4. Реализирайте WEB услуга, която изпраща обработения файл (от т.3) на отсрещната страна
5. Реализирайте WEB услугата (от т.4) така че: да няма достъп до нея произволен потребител (идентификация/автентификация)
6. Документирате предните точки.

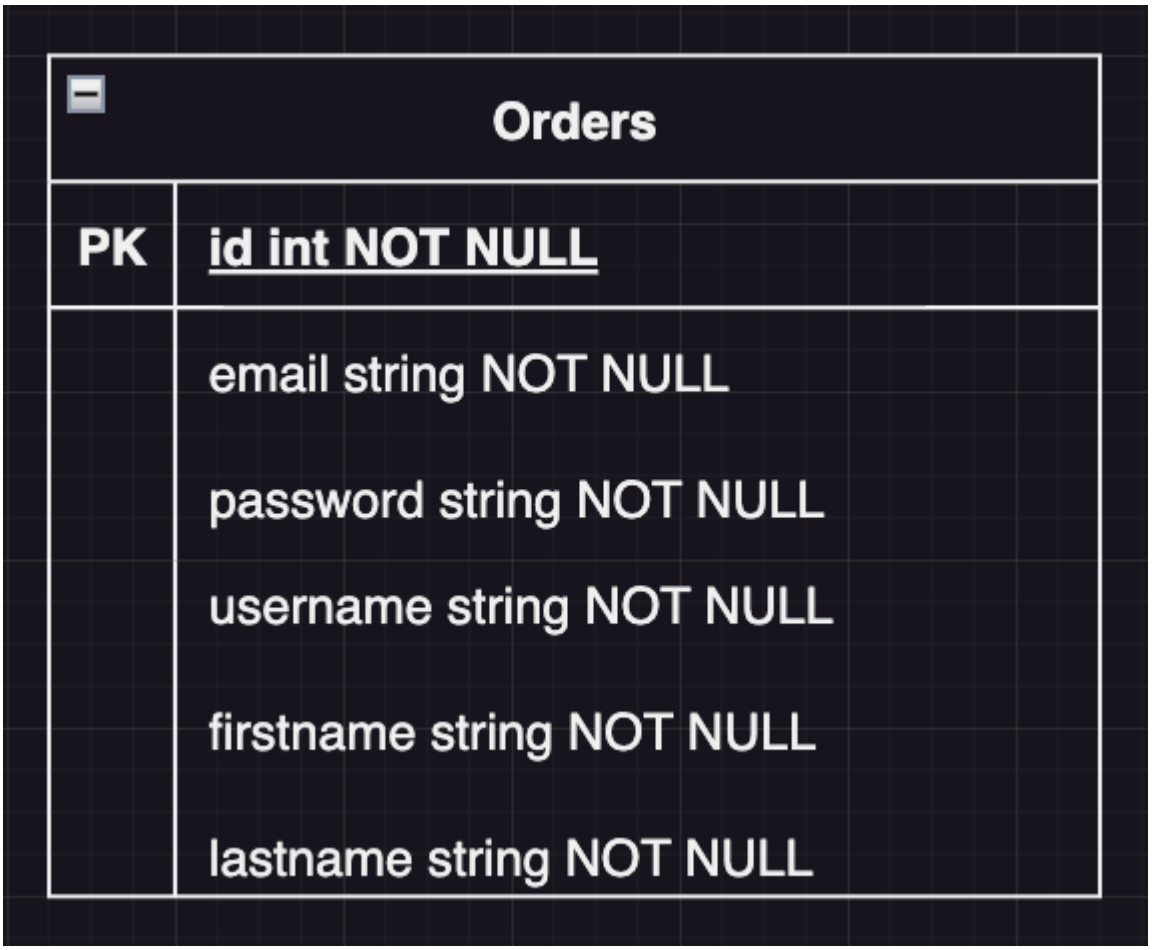
Въведение в приложението

Проектът представлява уеб приложение, което използва софтуерната рамка “Flask” и реализира връзка с база данни посредством Flash Alchemy. Само по себе си

предоставя възможност за потребителска регистрация и вход, запомняне на потребителската сесия и съхранение на потребителите в база данни. Приложението демонстрира на потребителите т.нар. Billion Laughs атака - уязвимост при XML, която има за цел претовари сървъра и да доведе до Denial of Service - DoS. Атаката се състои от дефиниране на 10 обекта (ENTITY), всеки дефиниран като състоящ се от 10 от предишния обект, като документът се състои от единичен екземпляр на най-големия обект, който се разширява до един милиард копия на първия обект.

Спецификация на базата данни

На следната фигура е представена Entity Relationship (ER) диаграмата на базата данни, която показва наличната таблица в базата данни. В базата се съхраняват регистрираните потребители.



Entity Relationship диаграма

Спецификация на кода

models.py

Съдържа модела на потребителя като описва неговите полета. Класът се използва за работа с потребители като създаването им е code-first:

- class User(db.Model, UserMixin) - представлява потребител в системата. Има следните полета:
  - id - уникален идентификатор в базата данни;
  - email - имейл адрес на потребителя;
  - password - парола на потребителя, която се пази в криптиран вид в базата данни;
  - username - уникално потребителско име;
  - firstname - първо име на потребителя;
  - lastname - фамилно име на потребителя.

main.py

Съдържа двата главни метода, които се представят пред потребителя. Те са следните:

- index() - изобразява главния изглед на приложението и съответства на “/” пътя;
- profile() - изобразява изгледа за профил на потребителя като този изглед е наличен само когато потребителят е влязъл в профила си. Това се постига чрез @login_required декоратора.

auth.py

Съдържа методи, свързани с автентикацията, регистрирането и влизането на потребители. Методите са следните:

- login() - изобразява изгледа за влизане в приложението;
- login_post() - POST метод за влизане на потребител в приложението. Използвайки въведените потребителско име и парола, методът ги валидира и се опитва да вкара потребителя в системата. При невалидни данни, чрез flash() се предава съобщение на изгледа, което се визуализира в червен цвят, уведомявайки потребителят за грешно въведените си данни. Ако данните са коректни, приложението препраща потребителят към страницата за неговия профил;
- signup() - изобразява изгледа за регистрация в приложението;

- signup_post() - POST метод за регистриране на потребител в приложението. Методът извлича данните от формата и прави проверка за вече съществуващ потребител с тези креденциали. При намерен такъв, отново чрез flash(), потребителят бива уповествен. Ако потребителят не съществува, той се създава и записва в базата данни, като преди това паролата се криптира. При успешна регистрация, потребителят бива препратен към страницата за влизане;
- logout() - метод, който изкарва потребителят от приложението, използвайки logout_user() от flask_login библиотеката, след което препраща потребителят към главната страница. Този изглед е наличен само за потребители, които се влезли в системата, което се постига чрез @login_required декоратора.

attack.py

Съдържа методи, свързани с генерирането на уязвимия XML файл и неговото представяне. Методите са следните:

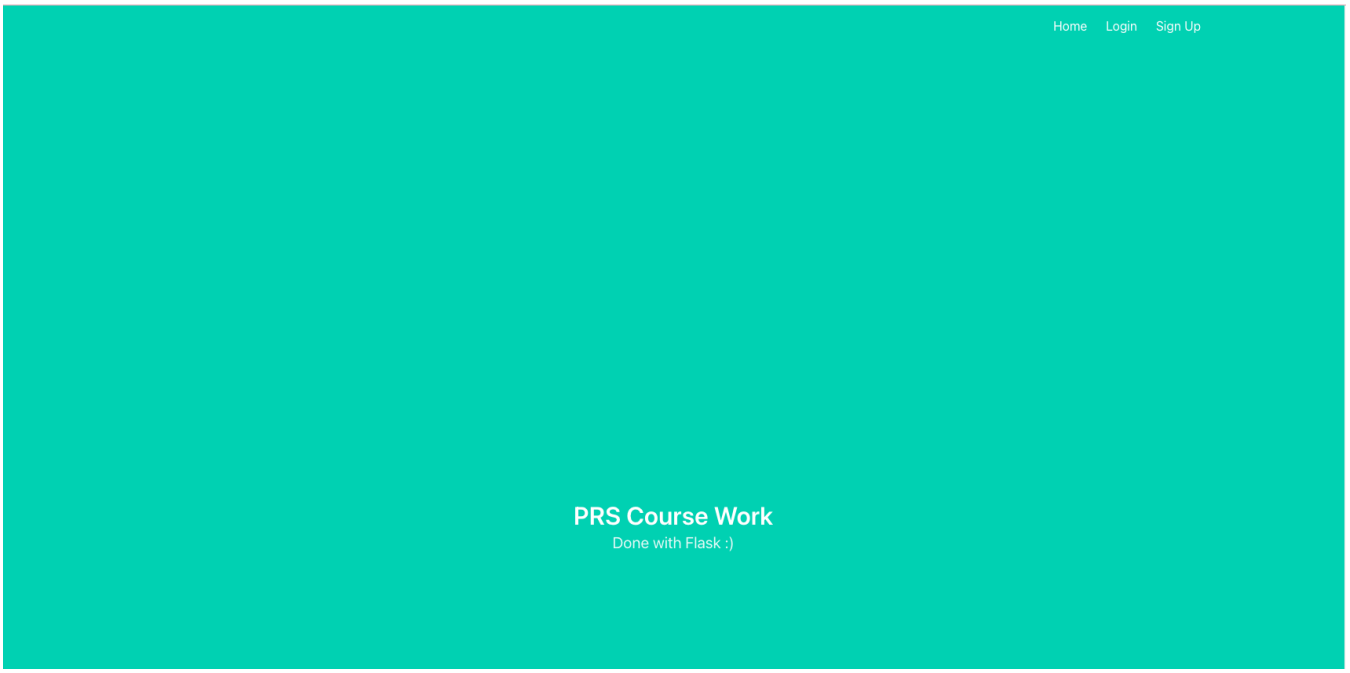
- generate_xml() - генерира уязвим XML файл под папката static в директорията на проекта. Използва базова XML структура, в която вкарва повтарящите се ENTIT, които правят референция едно към друго. Когато съдържанието на файла е генерирано, то се записва чрез write() функцията в XML файл;
- billion_laughs() - изобразява изгледа за демонстрация на уязвимостта. Извиква генерирането на файла, след което се опитва да извърши parse() операция чрез xml.etree.ElementTree. В по-стари версии на Python това довежда до сриване на сървъра и DoS. В текущата версия, подобна уязвимост във файла се засича и парсърът повдига ParseError грешка, която е обработена в метода. Това демонстрира изпълнението и засичането на атаката.

templates/ папка

Съдържа HTML файлове, които представляват изгледите, съответстващи на горе-описаните функции. В тези изгледи се използва шаблонизиране чрез jinja2, което позволява динамично представяне на съдържание на потребителите, както и вкарване на допълнителна логика за визуализиране и изключване на дадени елементи от потребителския интерфейс. В изгледите, в които има възможност за повдигане на съобщение чрез flash(), има обособен блок, който прихваща съобщението и го визуализира на потребителя с правилното форматиране.

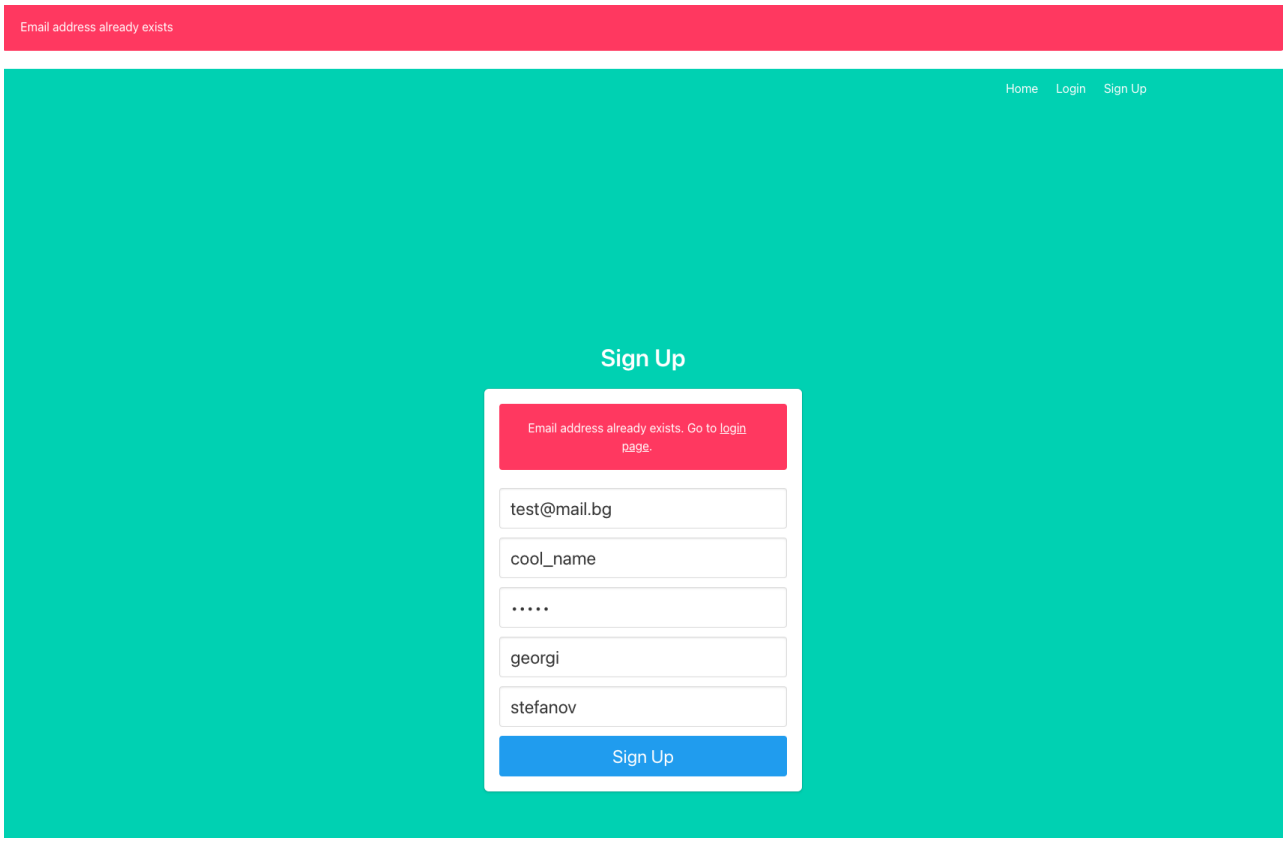
Демонстрация на работоспособността

На фигурите по-долу е демонстрирана работата на проекта. На *фиг. 1* е показано най-първоначалното стартиране на програмата, при което се визуализира главният изглед, който приветства потребителят. Както се вижда, без да е влязъл в системата, пред потребителят стоят 2 опции - Login и Sign Up.



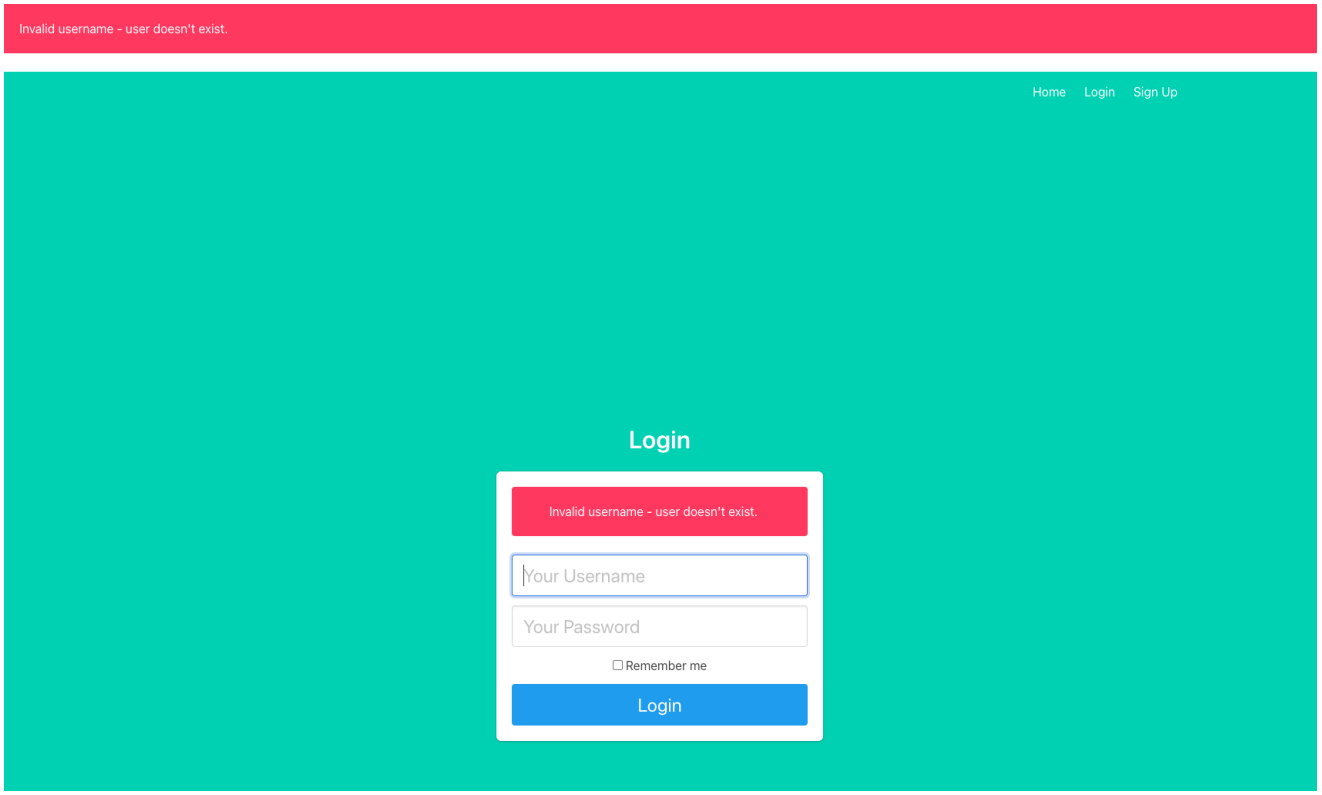
Фиг. 1. Главен изглед на програмата

На *фиг. 2* е показан изгледът за регистриране на потребителят. Както се вижда, потребителят трябва да въведе данните си като имейла и потребителското име не трябва да съществуват. Ако съществуват, на екрана се визуализира грешката в червено, която се вижда на фигурата.



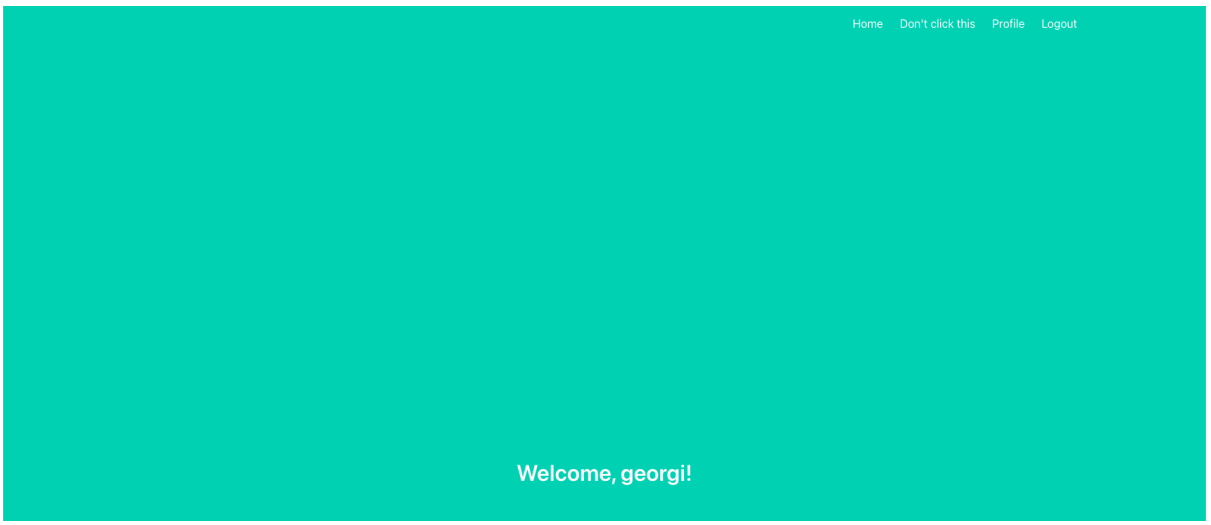
Фиг. 2. Страница за регистриране

При правилно въведени данни, потребителя бива регистриран и изпратен към изгледа за влизане в профила си, което е представено на *фиг. 3*. Фигурата отново показва грешката при въвеждане на несъществуващ потребител. Тази за грешна парола е аналогична. Потребителят има опцията да избере да бъде запомнен, за да не трябва да влиза в профила си отново при презареждане на страницата.



Фиг. 3. Страница за потребителски вход

На *фиг. 4* представя изгледа, който се визуализира при успешен потребителски вход. Приложението динамично разпознава името на потребителя.



Фиг. 4. Страница на потребителя след успешен потребителски вход

Фиг. 5 резултатът от натискането на менюто “Don’t click this”. Това препраща потребителя към /attack пътя, което изпълнява генерирането и parse-ването на уязвимия XML файл. Както се вижда, в тази версия на python, грешката е известна и се прихваща, както се очаква.

Фиг. 5. Резултат от опит за parse на уязвимия от billion laughs attack файл