ECE 568 Homework 3

Kuan Wang (kw300)
Weihan Zhang (wz125)

Homework 1 Code Review

I. Code Quality

A. Abstraction

- The python files are put in an organized manner under the Django web framework.
- They make good use of the *forms* file to create forms.
- Every piece of code has its functionality. No obvious repetition is found.
- It is a good practice to write a new file for the *view* of drivers. Building on this, it would be better to make a separate file for the *view* of users.
- One app is set up to realize all the functionalities. I suggest creating two apps for users and drives separately.

B. Formatting

- Function names and variable names are straightforward to understand.
- Classes and functions are written within a reasonable size.
- There is clear logic in classes and functions, so readers can easily understand how things are handled.
- Some codes are too long in a line, which reduces the readability.
- There is a small number of comments that help readers better understand how the code work, but more comments should be added for readability and future development.
- Leaving unused codes commented can be confusing to readers. My suggestion is to delete the unused codes.

II. Security

A. Initial Security Analysis

By reviewing the code, we could figure out some pros and cons related to security aspects.

Pros

- They use *LoginRequiredMixin* to authenticate the user. Hence, a user who has not logged in would be redirected to the login page, if she directly enters a URL of a user's page.
- They require users to enter a strong enough password at the signup page by using *UserCreationForm*.
- By using the Django ORM interface, SQL injection is prevented.
- Orders are shown differently according to their status, so owners can only modify their orders when it is open and not shared.

Cons

- There is no password for the database. All the information may be attacked by evil users or attackers.
- The confirmation page is not redirected correctly. So, an error page will occur when a driver clicks the “confirm” button.
- *LoginRequiredMixin* is not applied to the driver registration page. Hence, a user can go directly to this page without login. Further operation on this page may lead to an error.

Then, we would like to demonstrate the pros and cons of security problems by implementing the program.

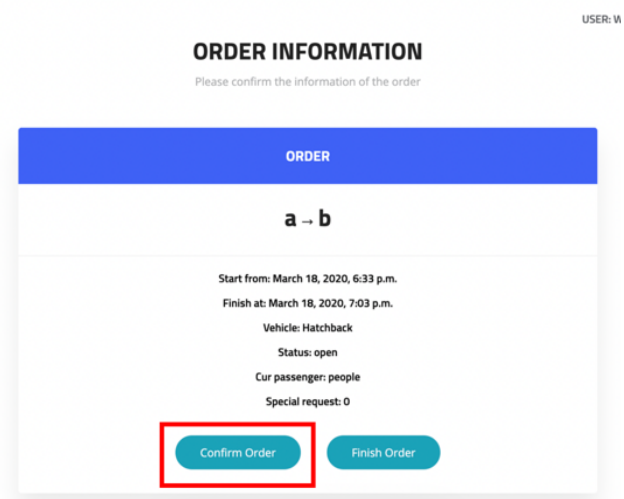
B. Vulnerable Parts

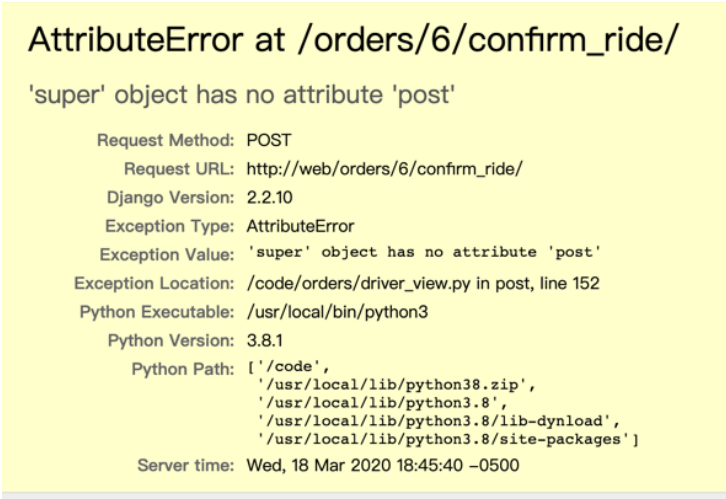
In this part, we exploit some attacks related to the availability of the ride-sharing website.

- **Functionality Problem**

As a Driver:

-When the driver wants to confirm the order, the website would not work well. The following is the problem that occurred when we click that confirm button.





- The driver can finish the order by clicking the finish order button directly without confirming it firstly.
- The drivers cannot modify their personal information (vehicle type, plate number) and they can only view their information unless they confirm an order.

As a Ride Owner:

- The ride owner cannot update the current passenger number correctly.

As a Ride Sharer:

- The sharers could not withdraw their own joined orders.

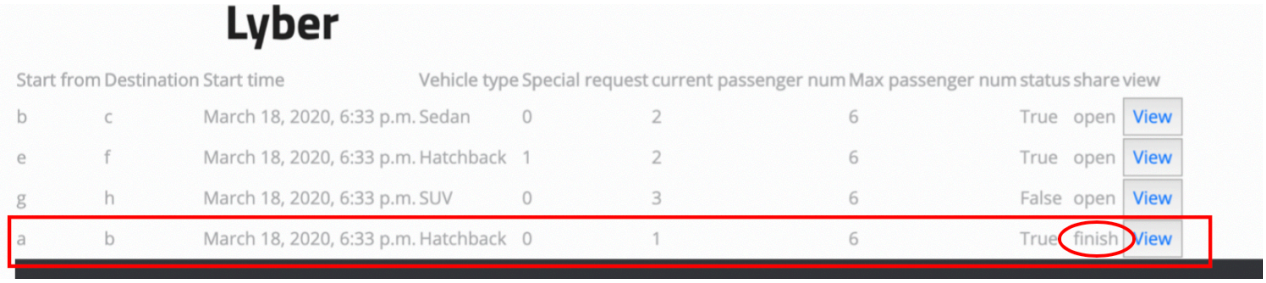
- **Malicious Attack**

1. The finished order can also be modified by the ride owner. If we click on the modify my rides button, we could see the finished rides are still on the list. By clicking the view button, we will jump to a page where we could update all the information we provided for this order.

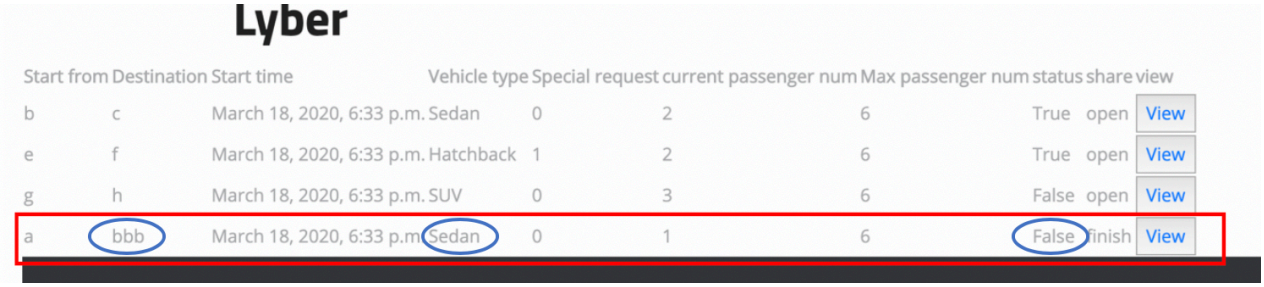
Example: here we provide two screenshots to demonstrate where the problem occurred:

Another point here is that the attribute of the list is ambiguous. The status means the order status (open, confirm, finish) and the value (True or False) of the share attribute shows whether the current order is allowed to share with others or not.

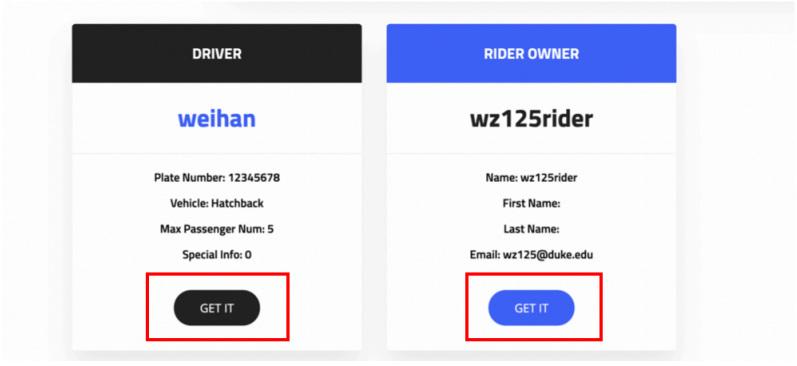
The following screenshot shows that the order has the finish status.



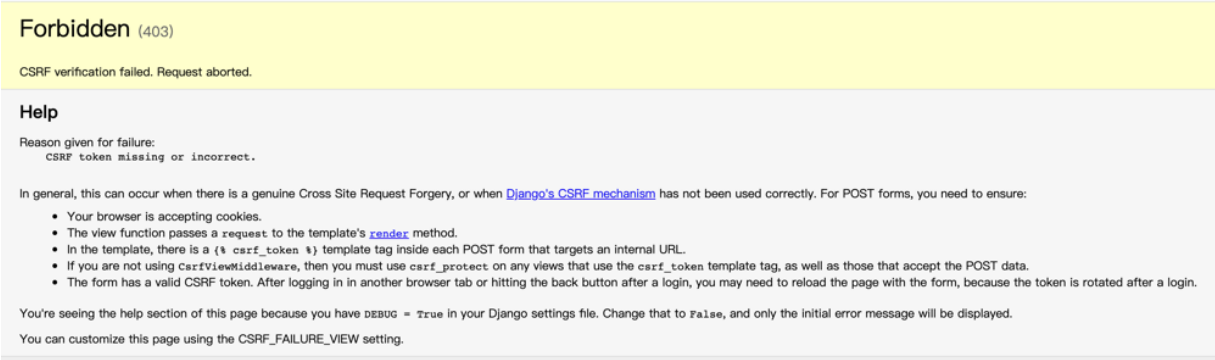
However, if we click on the view button, we could change all the information and the result was shown in the next screenshot. The blue circle shows where we updated.



2. The “GET IT” button cannot work normally.

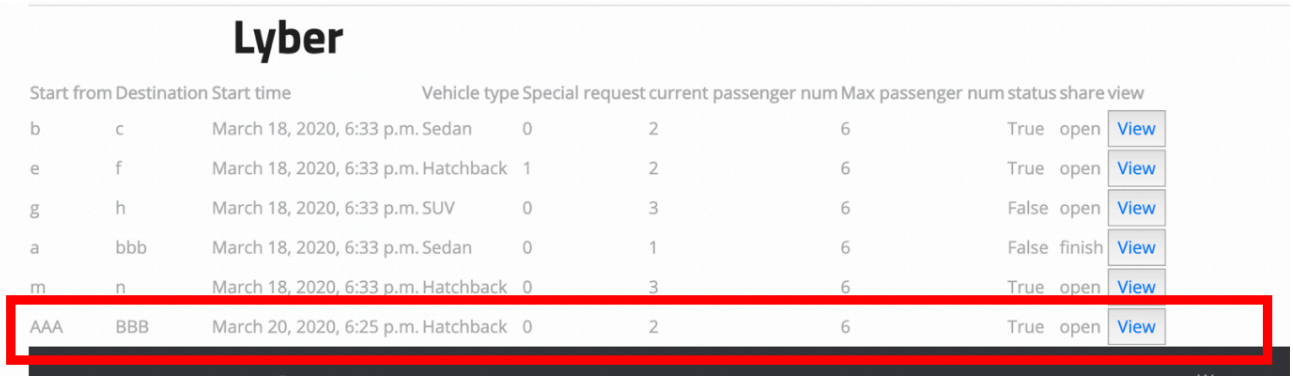


It will lead to an error message:



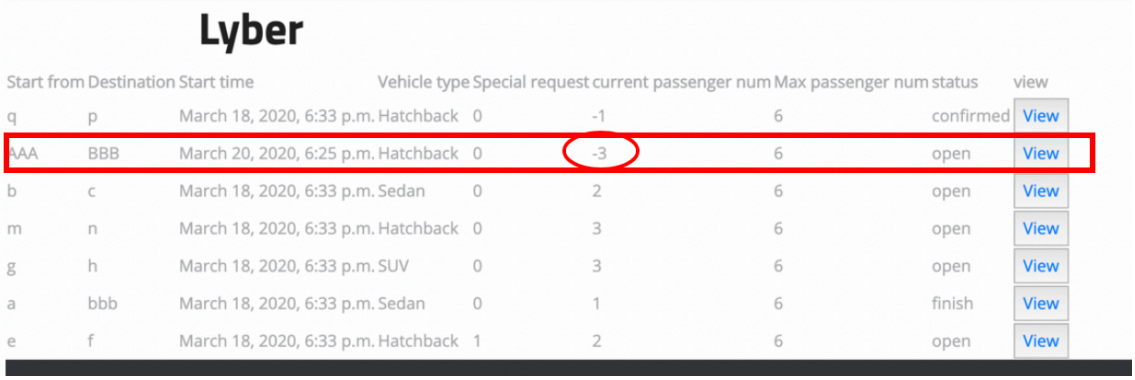
3. If we act as a ride sharer by clicking the "Join a ride" button, we can input the detailed information. The unexpected thing is that the website allows us to choose the number of sharers could be negative or zero.

Example: We would provide the following procedures to demonstrate this illegal error.
First, we login as a ride owner to request a ride.

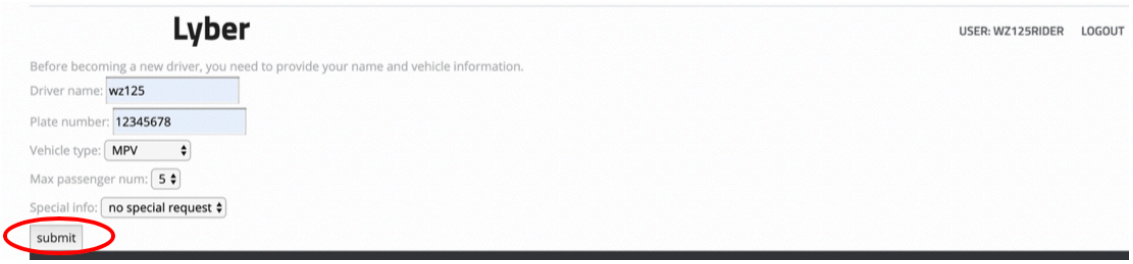


Then, we login as ride sharer to click on the "Join a ride" button to share that ride we request.

Finally, we return to a list of the orders web page, we could find that the order we requested has a negative number of current passengers.



4. Type in the URL (/driver/newDriver) directly in the browser address bar could open that page without login.



Then we click on the “submit” button. This would lead to IntegrityError.

5. The users could pick up their rides if they become a driver and request a ride.

C. Immune Parts

- If we directly type in the URL of some webpages to the browser address bar, we could not go to that page. Because they use the *LoginMixinRequired* except for one page we clarified clearly in the malicious attack in the former vulnerable parts.

D. Interface Design

Except for some designs that were obviously not completed due to time constraints, the rest of the front end of the website looks very nice.

Homework 2 Code Review

I. Code Quality

A. Abstraction

- Functions are written with clear logic and a reasonable size.
- They did not make use of Object-oriented programming. They just use the C format but create the files naming *.cpp, since C++ needs to create all the methods and fields inside of classes.
- They can make better use of object-oriented design to create more classes, such as HTTP requests, HTTP responses and so on.

B. Formatting

- Some long functions are well commented in each part, while the others are not adequately commented.
- They use good variable names and function names, which facilitates understanding.
- Test cases are provided to demonstrate the functionality of the proxy server. We suggest providing more comprehensive test cases.
- Some unused files are left in the main directory and they may confuse the readers. We can understand that the project has not been fully debugged due to the time limit. A good point is that they write clarification in README. A better way is to create a sub-directory for these unused files or write more guidance for reference.

II. Security

A. Initial Security Analysis

Pros

- They make good use of exceptions throughout the project.
- The proxy would reply with a 400-error code when it receives a malformed request.
- They try to add a lock to the cache to prevent race conditions between multiple threads, so the data in the cache can be consistent.

Cons

- By searching through the entire code, we did not find they handle the situation where the proxy contacts the destination web server and receives a corrupted response. The proxy did not reply to the client with a 502-error code.

- RAII is not implemented in this project.
- They have not considered memory overflow, while it can occur with a large number of HTTP requests.
- We could find that because of the constraints of time, they did not call the functions related to cache parts in reality.

- In all the cache part cpp files, we did not find they handle the revalidation procedure (Eg. Use Etag and LastModified fields and add the string “If-None-Match” and “If-Modified-Since”).

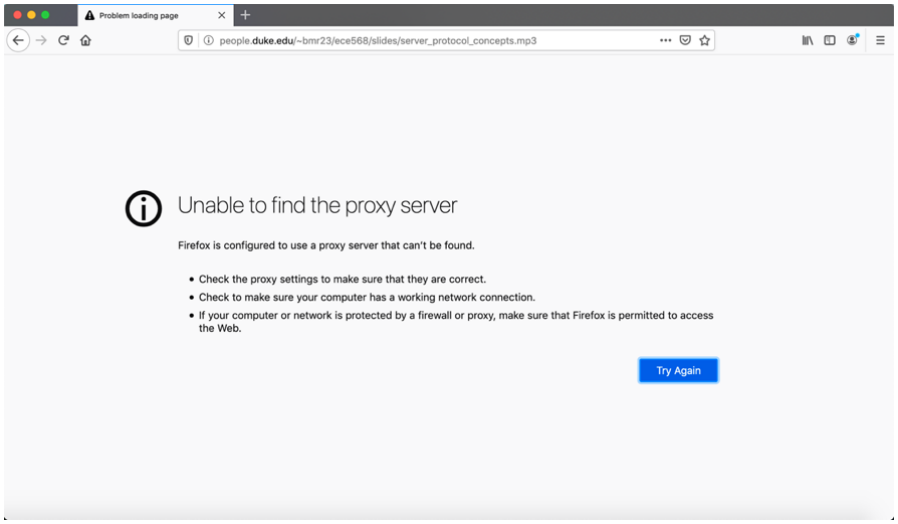
B. Vulnerable Parts

1. Cannot Open Audio Lecture in ECE 568 class page

The link is: http://people.duke.edu/~bmr23/ece568/slides/server_protocol_concepts.mp3
We provide some screenshots to demonstrate the error. In the terminal, it shows that the segmentation fault occurred:

```
wz125@vcm-12475:~/ece568/hw3/erss-hwk2-drl21-at334$ sudo docker-compose up
Starting ersshwk2drl21at334_proxy_1 ...
Starting ersshwk2drl21at334_proxy_1 ... done
Attaching to ersshwk2drl21at334_proxy_1
proxy_1  | server: waiting for connections...
proxy_1  | 3
proxy_1  | Using multithread
proxy_1  | Listening for a connection
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | hostname: incoming.telemetry.mozilla.org port: 443
proxy_1  | Listening for a connection
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | hostname: incoming.telemetry.mozilla.org port: 443
proxy_1  | hostname: incoming.telemetry.mozilla.org port: 443
proxy_1  | hostname: people.duke.edu port: 80
proxy_1  | Caught error: client recv: Bad file descriptor: 25
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | Listening for a connection
proxy_1  | hostname: ocsp.digicert.com port: 80
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | hostname: ocsp.digicert.com port: 80
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | hostname: ocsp.digicert.com port: 80
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | hostname: people.duke.edu port: 80
proxy_1  | Segmentation fault (core dumped)
ersshwk2drl21at334_proxy_1 exited with code 139
```

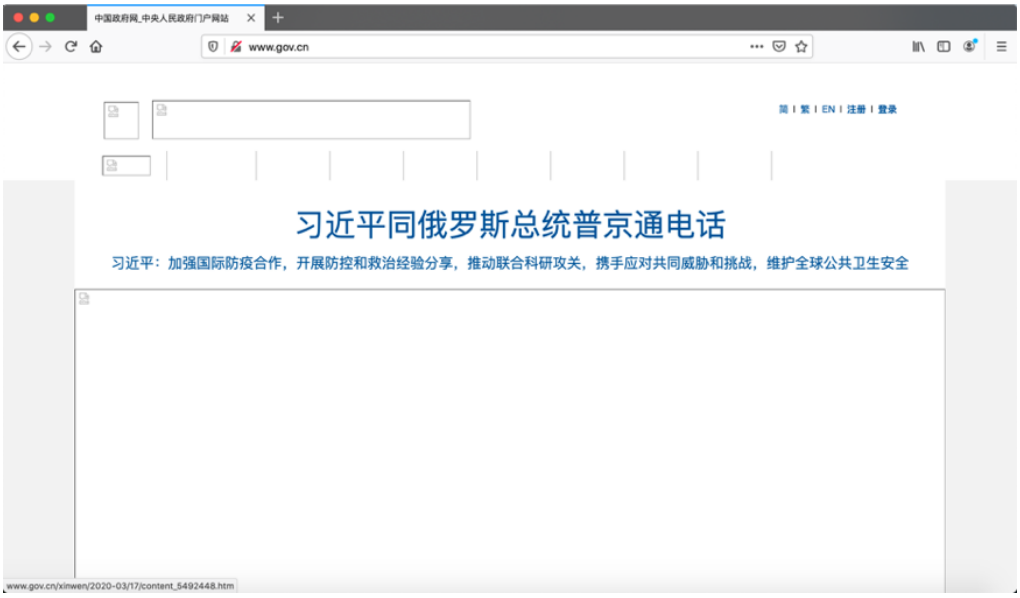
Then the proxy server program interrupts and the web browser shows:



2. Cannot load websites correctly

- a. The link is http://www.gov.cn/

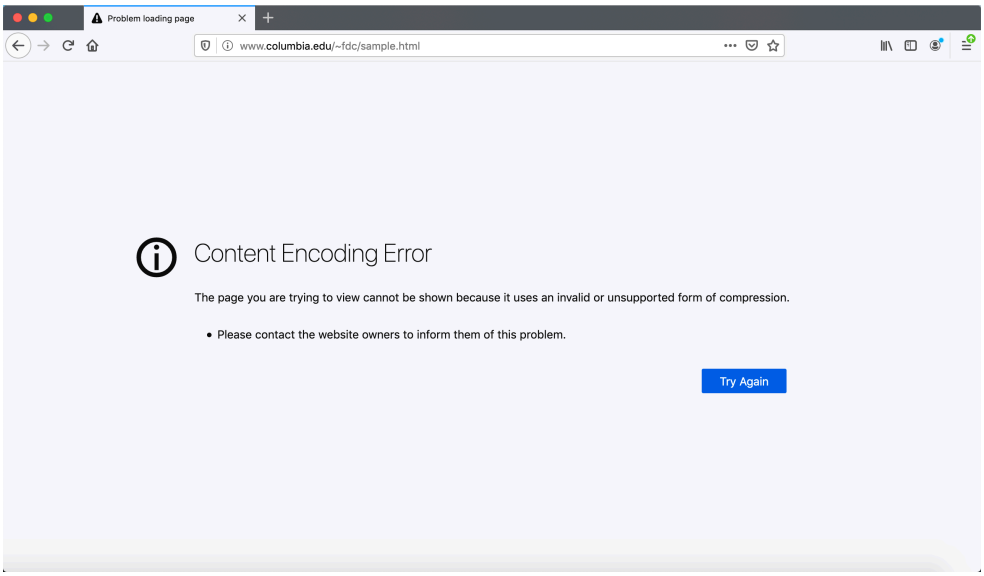
This website cannot be opened normally, like some pictures on the website cannot be loaded. Here is the screenshot of opening that website.



After a while, the terminal shows that the proxy server program interrupted because of segmentation fault.

```
wz125@vcm-12475:~/ece568/hw3/erss-hwk2-drl21-at334$ sudo docker-compose up
Starting ersshwk2drl21at334_proxy_1 ...
Starting ersshwk2drl21at334_proxy_1 ... done
Attaching to ersshwk2drl21at334_proxy_1
proxy_1  | server: waiting for connections...
proxy_1  | 3
proxy_1  | Using multithread
proxy_1  | Listening for a connection
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | hostname: www.gov.cn port: 80
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | hostname: ysp.www.gov.cn port: 80
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | hostname: user.www.gov.cn port: 443
proxy_1  | Listening for a connection
proxy_1  | server: got connection from 107.15.252.4
proxy_1  | hostname: ocsp.globalsign.com port: 80
proxy_1  | Segmentation fault (core dumped)
ersshwk2drl21at334_proxy_1 exited with code 139
```

- b. The link is: http://www.columbia.edu/~fdc/sample.html
This website cannot be open correctly as well.



3. Cannot find the proxy.log after running the code
Since they did not create the log file correctly, we could not figure out what parts are wrong. For example, in the CONNECT method we cannot find out whether the tunnels the proxy established for the communication between server and client are closed or not. Also, they did not invoke the cache part code, so there will be only some basic GET request without revalidation procedures.

C. Immune Parts

- They handle the chunked message very well. The picture could be loaded entirely rather than showing up bit by bit. So, if the client close the web page in the process of loading a chunked response, the process would not be shut down.
- The proxy could reply with a 400-error code when it receives a malformed request.
- When we enter an invalid URL, the proxy returns a 404-error code and the connection does not break.

Homework 1&2 Lessons

Kuan Wang

Hw1:

- We used try and catch in the project to handle exceptions, but we could make use of this throughout the whole program to ensure each exception is properly handled.
- It is a good practice to put helper functions in a separate file. This abstraction can help with debugging and future deployment.

- We did not set strict rules on user's password and also did not use password confirmation. We could add password strength validation by using *UserCreationForm* in Django to not allow user to enter an easy password. Also, we could add password confirmation in case a user made a typo.
- Null field was but should not be allowed in ride request page. We should set NOT NULL for all field in the model of *Request*.
- If user enters an incorrect format (not the format of "YYYY-MM-DD") of start time in request time page, that will lead to *ValidationError*, which is unacceptable. We could add the requirement of correct format of time to avoid this error.
- If a user logs out and then enters a URL to her home page or other information page, it will lead to *AttributeError*. To solve this error, we could reset the current login status of the user when she logs out, instead of simply redirecting to the login page.
- We only checked the user's email address when a user directly enters a URL. However, an attacker can add this session and go to other user's page if she somehow knows the user's email address. We could use Django's inbuild validation system to avoid this security problem.

Hw2:

- We defined the cache as a global object. It exposed the variable to the other portion of the program that does not interact with cache. This is not a goo programming design and may cause security problems. We move the cache the part of main function where it is used.
- One malloc() in our program was not freed. It causes memory leak that an attacker can take advantage of. Also, our program would abort after opening around 20 websites. This is possibly related to the memory leak mentioned above. We should free each allocated memory space to avoid this kind of memory leak.
- In our program, information was transferred between the client side and the server side without any encryption. This is dangerous because malicious attackers can easily steal the confidential information. We could incorporate the knowledge we learned about cryptography to strengthen the security.
- We used a fixed-size array of 65535 to receive message. Although the size is quite large, it is still possible that the data size exceeds 65535 and causes buffer overflow. We should use *vector* instead to avoid loss of information.
- We did not check the validation of response before updating the cache. Hence, malicious response would also be stored in the cache, which might cause serious security problem. We should check the response code before storing a response into the cache to exclude the malicious responses.

- Our proxy server could not handle an invalid URL and in this case the program would crash and exit. In terms of an invalid URL, we should reject the thread request and return an 404 error to keep the program running.

Weihan Zhang

Hw1:

- We forgot to update the new value into the database when there are some changes in the previous orders. The undo and delete functions are not realized very nice due to the time limitation. In order to correct all of these we need to update the orders in a more reasonable way. We would be better to search for all the users who related to this orders and inspect their operations for this order in the database.
- For the problem where the ride owner can only review the numbers of passengers in their own party even if there are sharers who joined into this order, we need to add a field in the form to show up the current total passenger numbers.
- The problem with only one sharer could joined the order is important. We thought about it for a long time during our development. Since at that time we did not learn about the database systematically, we cannot figure out how to do that. Now, we think we could make use of Many To Many to solve that problem. And if we fixed the only-one-sharer problem, there would be no longer replace one sharer without notification situation.
- As for the abstraction and malicious attacking, since we used the Django platform and python to develop the website, it would be a easier way to write clean code and also avoid some malicious attack problem.

Hw2:

- We only considered the situation where there is only one response per connection in GET HTTP method. However, in some websites since they are extremely large and they will send multiple different responses to the client to open that website. So, we need to forward multiple times until the server side would not send any new responses to the current related requests any more.
- In our project we forgot to handle some malicious corner cases, like if the client closed the webpage in the process of receiving chunked messages from server. After it was closed, the send function produced a SIGPIPE signal and the process would shut down.
- It is not appropriate to use hardcode and we need to avoid the primitive obsession. For example, like the size of the char array we used to receive messages is fixed into 65536 each time. Instead, it would be better to use vector<char>.
- We forgot to handle some specific exceptions, like cannot open the log file or if there is no content-length field in the POST requests. Also, we remembered that we added the mutex

to lock the cache. But in our submitted version when we operate the cache, it was not locked. So, we need to double check the project code version before we submitted it.

- Since we only stored the first line of a request into our cache, it would dangerous when someone execute a cache poisoning attack. So, it would be better if we stored the entire requests from client into the cache as the key.