Application of form honeypot based on CSS style in ticket booking system

Shaoquan Li
College of Biomass and Engineering, Sichuan University, Chengdu, 610000, China

sqli@stu.scu.edu.cn

Abstract. Since various means of transportation and entertainment activities are restricted by objective conditions, the number of tickets is always limited, so ticket scalpers continue to be a companion product to this day. In the current Internet environment, the web application of the ticketing system faces challenges from ticket-grabbing robots controlled by ticket dealers. The current security procedures have significantly increased the workload and time consumption of effective users. The question now is whether ticket-grabbing bots can be stopped without increasing the workload of legitimate users and being as unnoticed by users as possible. In response to this problem, this article proposes a ticket booking system model integrated with a form-based honeypot, which can achieve the function of deceiving robots without being noticed by ordinary users. This model provides a feasible method for implementing network applications that are both secure and efficient, ensuring that users have a more seamless and protected experience.

Keywords: Ticketing system, Honeypot, Scalping robot

1. Introduction

Ticket scalping is a gray industry that has existed for a long time and has huge profit potential. In an era where offline ticket sales are the mainstay, ticket sellers often wait in front of the ticket window all night so that they can buy all the tickets within a short time after the ticket sales begin, and then sell them at a high price to consumers who need them. Such a "black market for tickets" appeared in American theaters as early as the 1800s. For example, in the 1860s, tickets for Charles Dickens' second American tour worth $5 were sold by ticket sellers for ten times the price, or $50 [1]. The spring festival travel rush (SFTR) phenomenon is a travel peak that occurs around 40 days during the Chinese Lunar New Year period. It sends billions of passengers on average every year. Ticket dealers make huge profits while causing a shortage of tickets, thus also triggering some related social problems [2-3].

With the popularity of electronic ticketing, the cost for ticket dealers to snatch and scalp tickets has become lower. Ticket sellers only need to use crawler robots and other means to easily grab tickets. The robots can guarantee to grab tickets faster and more efficiently than ordinary people. Faced with the problem of ticket robots snatching tickets, one solution is to increase the number of tickets issued, so that consumers who did not grab tickets the first time can still buy tickets. For example, China's railway department has certain measures to increase railway transport capacity. To a certain extent, it alleviated the ticket shortage problem during the SFTR period [4]. Another option is to promote the combination of real-name authentication and administrative law enforcement [5], but this option requires a large investment of social resources and is costly. Some articles propose using publicity means to guide consumers to resist scalpers [6], but the effect is difficult to quantify.

In addition to administrative methods, the application of robot detection technology is also an important means to prevent ticket-grabbing robots. Completely automated public turing test to tell computers and humans apart (CAPTCHA) is a classic detection mechanism. Its main content is to add a verification puzzle, you can proceed to the payment step after completing the verification puzzle. After years of development, CAPTCHA technology currently has various forms such as text-based, picture-based, audio-based, video-based, and puzzle-based [7]. However, there are some problems with CAPTCHA technology. More complex verification mechanisms often require larger files as support. For example, Video-based CAPTCHA needs to load a large video file of several megabytes [8]. The loading process consumes a lot of consumer time. Some technologies can also block robots, such as Proof of work (PoW). Some studies have proposed combining Internet protocol geographical information with PoW technology and deploying it in the ticketing system [9], but PoW technology can also cause "extra time" questions [9].

This article proposes a lightweight form-based honeypot application model in the ticketing system given the ticketing system's need to block ticket-grabbing robots and the problems of heavy user workload and long time-consuming solutions such as CAPTCHA and PoW. This model uses a honeypot based on Cascading Style Sheet (CSS) forms, aiming to solve the key problem of lightweight implementation of defense against ticket-grabbing bots, and is of great value in building a smoother user ticket purchasing experience.

2. Requirements Analysis and Design Elements

In scenarios where ticket supply is limited, addressing the challenge posed by ticket-grabbing bots is crucial. Particularly during peak travel periods and on popular routes, it becomes essential to not only counteract the influence of automated bots but also to cater to the urgent needs of customers aiming to swiftly secure tickets. Presently, widely adopted CAPTCHA systems leveraging audio, video, or puzzles effectively deter most robots. However, their application in ticketing systems introduces concerns related to increased network download traffic and heightened user workload. While competition among legitimate users for ticket purchases is inherent in scenarios of insufficient supply, the incorporation of resource-intensive CAPTCHA systems, such as large downloads and intricate puzzles, can significantly deteriorate the user experience, leading users to attribute frustrations to system design inadequacies.

2.1. Requirements

2.1.1. Functional Requirement. Beyond facilitating the standard ticket booking process, the system must adeptly thwart robotic activities without impeding the regular access of legitimate users. Striking a balance between robust anti-bot measures and ensuring a seamless user experience is paramount.

2.1.2. Non-functional Requirements. Users aspire to expedite the ticket booking process, minimizing the time spent on CAPTCHA challenges and related steps. To achieve user satisfaction, the system should adhere to the following principles.

2.2. Elements

2.2.1. Imperceptibility. Honeypot design should discreetly impede robotic interactions without disrupting the normal usage patterns of legitimate users. Users should navigate through the system seamlessly without noticeable hindrances.

2.2.2. Lightweight Design. Component sizes should be optimized to prevent undue strain on user networks and mitigate extended waiting times caused by large file sizes. Striking a balance between functionality and efficiency is crucial to maintaining a responsive and user-friendly system.

3. Solution

This method improves the method of "randomly assigning Identify Documents (IDs) to forms and modifying CSS visible attributes" proposed by Nassar et al. [10]. to automatically generate honeypots and valid forms and distinguish real users from robots through the visibility of form elements.

All IDs in this method are pre-stored in the library, and JavaScript is only used to select the IDs assign them to the form, and perform a callback operation on the server to determine which form is valid. Since the IDs are pre-generated, the number of modifications to the CSS style sheet is reduced. At the same time, since the effective IDs are not predetermined, even if the attacker can create a library for those legal IDs, the attacker cannot know in advance which form is valid.

3.1. Work process of Honeypot

Initially, the CSS styles corresponding to all IDs in the library are "hidden". When the user clicks the "Book" button, the JavaScript function generates a random number of forms, assigns these form IDs from the library, and then sends a request to the server to determine the legal forms. The server designates one of the used IDs as the temporary legal ID for this query, and modifies the CSS attribute corresponding to the ID to "visible". When the "Submit" button is triggered, the server collects scheduled information and initializes the CSS style sheet while checking the honeypot. Here in Figure 1 is a simplified model.

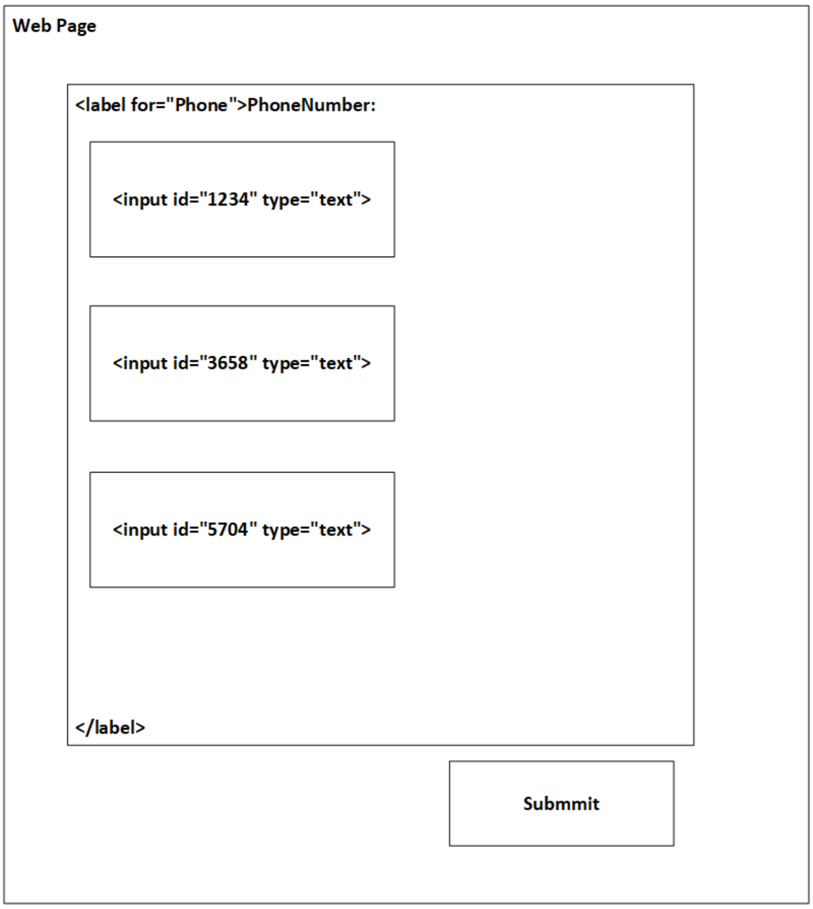

Web Page
<label for="Phone">PhoneNumber:
<input id="1234" type="text">
<input id="3658" type="text">
<input id="5704" type="text">
</label>
Submmit

Figure 1. Example of CSS-based Honeypot

3.2. Architecture of solution

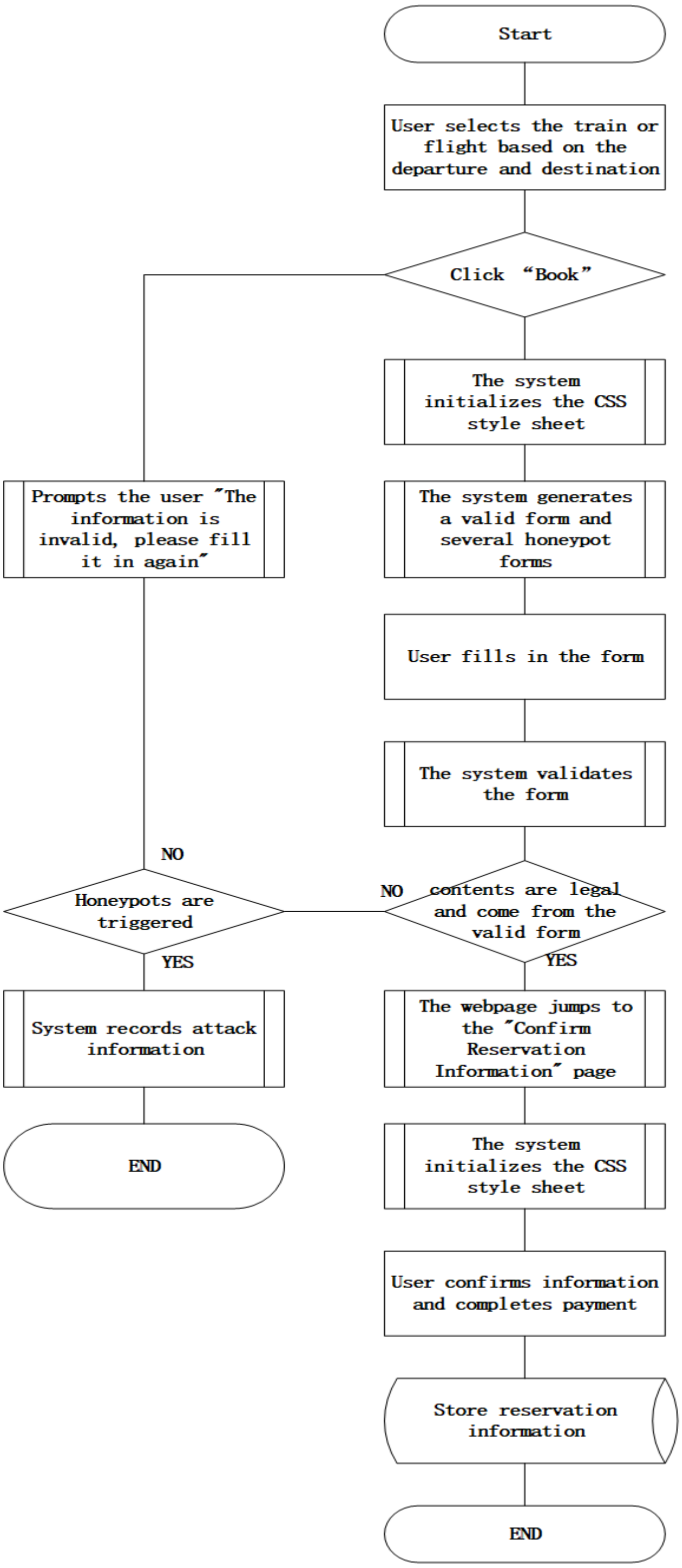

Start
User selects the train or flight based on the departure and destination
Click "Book"
The system initializes the CSS style sheet
Prompts the user "The information is invalid, please fill it in again"
The system generates a valid form and several honeypot forms
User fills in the form
The system validates the form
NO
Honeypots are triggered
NO
contents are legal and come from the valid form
YES
YES
System records attack information
The webpage jumps to the "Confirm Reservation Information" page
END
The system initializes the CSS style sheet
User confirms information and completes payment
Store reservation information
END

Figure 2. Model Architecture

As shown in Figure 2, the form generator is embedded in the reservation system to generate forms and honeypots. Every time a user clicks "Book" and tries to fill in the information, the system initializes the CSS style sheet and then generates a form group consisting of a valid form and multiple honeypots.

When the user fills in the reservation information, the system verifies the legality of the information and whether it comes from a valid form. If the honeypot is not triggered, it means that the reservation information is illegal, and the system sends an error message to remind the user to refill it. If the honeypot is triggered, the system collects the attack information from the honeypot and terminates.

After passing the verification, the website jumps to the reservation information verification interface. Once the user completes the payment, the reservation information is saved in the database and the reservation is completed.

3.3. Implementation measures

To ensure that the honeypot is deceptive enough, the honeypot and the valid form should be designed to have a higher degree of similarity, but at the same time, to prevent legitimate users from accidentally touching the honeypot, there must be a clear difference between the two.

This method chooses to modify the size of the input box to distinguish the honeypot from the effective form. The honeypot input box is set to be very small so that it cannot be recognized by human users with the naked eye and clicked with the mouse, while the input box size of the effective form is set. To be recognized by the naked eye, other parts are the same. This design ensures sufficient distinction between honeypots and valid forms while maintaining extremely high similarity at the code level.

4. Results and Discussions

4.1. Function implementation

In the actual design, based on providing a complete ticket purchase process, the system added a honeypot system to hinder robot users while minimizing the increase in user operation time.

4.1.1. Ticket booking function. Users can check the relevant train and flight status through the departure place and destination, and book tickets by providing passport information, name, and contact number.

Anti-bot measures: The system uses form-based honeypots to intercept ticket-grabbing robots.

4.1.2. Lightweight components. The system uses lightweight ancillary components to maintain low response time, reduce user network burden, and ensure smooth operation.

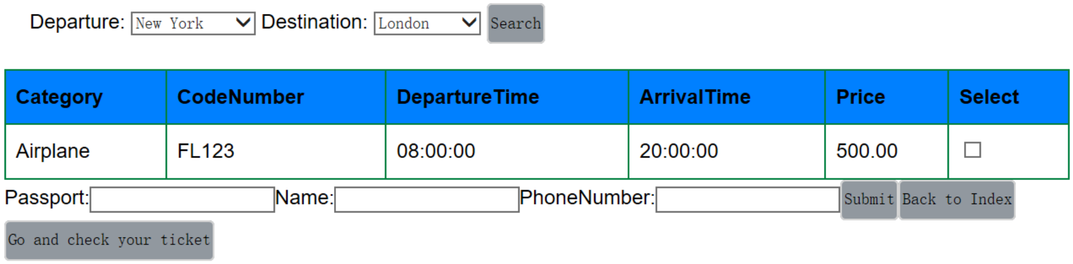
4.2. Assessment and Analysis

This design can normally realize the basic ticket booking process of query-select-book-submit and basic functions such as query orders.

4.2.1. Security. The randomly generated honeypot form can confuse the robot and collect attack information. The attacker has no way of knowing which form is valid, thus providing higher security than ordinary honeypots. The honeypot technology places higher technical requirements on ticket-grabbing robots, thereby increasing their intrusion costs and ensuring security while keeping the component size small.

4.2.2. Response speed. The honeypot form designed based on CSS has very little impact on the response speed of the web page. The main contribution to the delay in loading responses comes from the interaction between the front end and the database because reading a large ticket database requires a greater amount of work than a form. If the user's operation time is considered, the response time in milliseconds is negligible compared to Video-based CAPTCHA or Audio-based CAPTCHA which lasts for several seconds or even ten seconds. Therefore, the honeypot component can be considered not to affect the user's operational fluency.

4.3. Model results display



Departure: New York Destination: London

Category	CodeNumber	DepartureTime	ArrivalTime	Price	Select
Airplane	FL123	08:00:00	20:00:00	500.00	☐

Passport: Name: PhoneNumber:

Figure 3. Web page model

As shown in Figure 3, the honeypot is deployed in a static ticketing system based on a local database; that is, the ticketing information is not updated in real time. The input boxes of the three elements "Passport," "Name," and "Phone" are set up with multiple honeypot input boxes, but they are not visible in the web interface.

4.4. Overall evaluation

Judging from the model design results, the concealment of the honeypot is well demonstrated, and the existence of the honeypot cannot be visually discovered. At the same time, the response speed of the ticket booking system can be considered to mainly come from the operation of the huge ticket database. However, the safety measurement of this model requires follow-up research to determine.

5. Limitation and Future Outlook

This model currently has certain limitations, mainly in the following two aspects: security in practical applications and difficulty in maintaining the CSS form.

5.1. Limitations

In view of the lack of complete robot attack testing of this model, its applicability in contemporary network environments and reliability data against the latest attacks are currently unknown. The current security tests accepted by this model are not strong enough to represent real attacks in the current network environment. Even inconspicuous parts of non-commercial websites may be attacked tens of thousands of times per month on average [11]. This part of the data still requires subsequent testing.

To simplify the operation of the CSS form, this method uses a style that is pre-assigned to each ID, and the original CSS style is modified when the form is generated, but this brings other problems. A large number of unused IDs are idle, which will waste a lot of space and increase the file size. At the same time, due to the original performance of the selector, too many idle styles will also have a certain impact on web page rendering. Additionally, assigning a style to each ID is not conducive to CSS style sheet maintenance.

5.2. Outlook

This model has the advantages of lightweight and concealment. In the future, we plan to use a variety of robot attack methods to comprehensively test the security performance of this model. In addition, the multi-dimensional joint deployment of this model with web honeypots and CAPTCHA technology is also a feasible development direction.

6. Conclusion

In this paper, a ticket booking system model is proposed that applies a form honeypot to block ticket-grabbing bots. This method generates multiple identical forms before the user fills in the ticket booking information, assigns an ID to the form with a random number, and randomly determines whether each form is a honeypot form through the backend, thereby realizing a randomized honeypot. The honeypot generated by this method is temporary, which ensures that the ticket-grabbing robot cannot crack the honeypot by conducting multiple attacks and generating a honeypot ID database. This method has the advantages of low cost, lightweight, and high concealment, and can reduce the occupation of network resources and user time consumption. The research on this model provides new ideas for building multi-dimensional honeynets on web design and provides solutions for other robot prevention systems that have lightweight requirements. This model currently lacks security testing in actual network environments. It also has problems such as a high vacancy rate of CSS style sheets and high maintenance costs. In the future, this model is expected to be combined with other security technologies to build composite protection systems.

References

[1] Devine J A 2014 Ticket Scalping in the Late 1800s and the early 2000s – Much has Changed Much is the Same (New Jersey: Seton Hall University)

[2] Ying Zhou 2012 Shortage and Injustice: Sociological Analysis about The phenomenon of "Spring Festival" Transportation (Shanghai: East China University of Science and Technology Press)

[3] Guan Y Wu B and Jia J 2020 Transport Res. F-Traf. 73 143

[4] Zhang C 2014 The thinking and the way of train tickets in railway transport during spring festival (Chengdu: Southwest Jiaotong University Press)

[5] Dong F 2023 "Electronic scalpers" are infested, how to cure the stubborn problem of ticket scalp ing and scalping Legal Daily 2023-12-21. http://www.legaldaily.com.cn/sylm/content/2023-12/21/content_8941664.html

[6] Jin X 2023 Business Observation 29 10

[7] Alharbou Y S 2019 International Conference on Computer Applications & Information Security (ICCAIS) (Riyadh, Saudi Arabia), IEEE p 1

[8] Abdalla K and Kaya M 2017 Int. J. Sci. Res. Inf. Syst. Eng 2 3

[9] Kaiser E and Feng W 2010 INFOCOM IEEE Conference on Computer Communications Workshop (San Diego, CA, USA), IEEE p 1

[10] Nassar N and Miller G 2012 International Conference on Computational Aspects of Social Networks (CASoN) (Sao Carlos, Brazil), IEEE p 250

[11] John J P Yu F Xie Y Krishnamurthy A and Abadi M 2011 Proceedings 20th international conference on World wide web (New York: Association for Computing Machinery) p 207