Captive Portal Development Report

Course: CSCD94 Computer Science Project
Supervisor: Marcelo Ponce
Name: Lianting Wang

Table of Contents

1. Understanding Captive Portal Workflow:

Background and definitions

Captive Portal is a network technology used to control users to complete specific actions before accessing the Internet through a network access point. It is commonly used in public Wi-Fi networks to require users to agree to terms of service, enter credentials, or view advertisements. The architecture of the technology is defined in detail in IETF RFC 8952.

Main references

In delving deeper into Captive Portal technology, I have referred to the following information provided by Prof. Marcelo to deepen my understanding of Captive Portal technology:

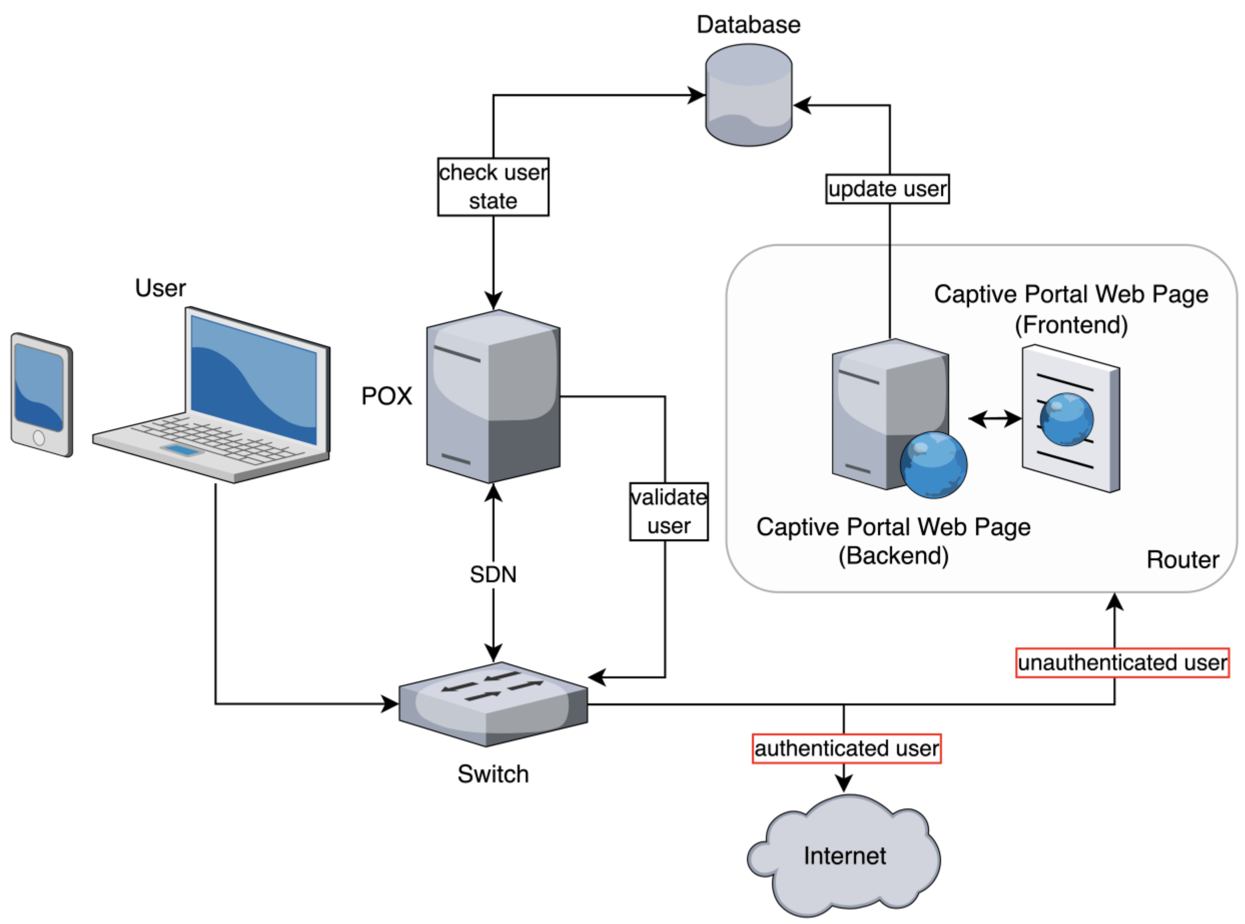
- The Captive Portal Architecture as defined by the IETF.
  https://www.rfc-editor.org/rfc/rfc8952.html
- Some general considerations.
  https://www.talentica.com/blogs/sdn-captive-portal/
- Some implementations using mininet.
  https://github.com/Talentica/of_odl/tree/master
  https://github.com/ederlf/CapFlow/tree/master

Based on the materials, the workflow can be summarized in the following steps:

1. User Device Connection: A user's device connects to the network and attempts to open a web page.
2. SDN (Software Defined Networking) Check: The SDN infrastructure checks the user's state to determine if authentication is required.
3. Captive Portal Redirection: If unauthenticated, the user is redirected to the Captive Portal's web page.
4. Authentication Process: The user submits credentials via the Captive Portal's frontend, which are then validated against the backend system.
5. Access Grant: Upon successful validation, the user's state is updated, and access to the internet is granted.

Implementation Method Chosen

The selected implementation method integrates a Software Defined Networking (SDN) approach to manage the Captive Portal's workflow efficiently. The detailed steps are as follows:



1. Device Detection and User State Verification: Upon connection to the network, the user's device is detected by the SDN controller, which checks the user's state to determine whether the device has been authenticated.

2. Captive Portal Trigger: If the user is unauthenticated, the SDN controller redirects the user to a Captive Portal Web Page. This consists of a frontend interface where users interact and a backend system that processes the authentication.

3. User Validation: The backend system is responsible for validating the user's credentials. Upon attempting to access a web page, the user is presented with a frontend authentication page requiring login credentials.

4. Database Interaction: The system interacts with a database to check the credentials provided by the user. This database could be an internal user management system or an external authentication service.

5. Internet Access Granting: After successful validation, the SDN controller updates the user's state to 'authenticated,' which allows the user to access the internet without further Captive Portal interruptions.

6. Post-Authentication Flow: For authenticated users, the internet access requests bypass the Captive Portal and are routed directly to their destinations, ensuring a seamless browsing experience.

This implementation method was selected for its scalability, flexibility, and the ease with which it can be integrated into existing network infrastructures. Additionally, the use of SDN allows for dynamic management of network resources, real-time monitoring, and the ability to implement complex policies for network access control.

2. Selection of development tools for implementation:

Mininet: flexible network simulation

The choice of Mininet as the core tool for our project is based on several considerations. First, the usage of Mininet is something I learned in Prof. Marcelo's course, which means that I already have the basic knowledge and experience to use this tool. This familiarity will help us carry out the project more efficiently.

Second, Mininet provides a completely software-based platform for simulating network environments, which allows us to conduct our research without being constrained to specific hardware or network architectures. This flexibility is important because it allows us to quickly test and validate the Captive Portal solution in different network environments. In this way, we can explore a wider range of possible scenarios, thus increasing the generality and robustness of our solution.

POX: A Flexible and Familiar Controller Framework

In choosing POX as another key tool for our project, I was mainly inspired by the implementation of a previous assignment in Prof. Marcelo's course. In that assignment, I used POX as a controller to implement network functionality, an experience that provided me with valuable hands-on experience and inspiration.

POX, as an open source SDN controller, provides flexibility and scalability, which is crucial for our Captive Portal project. Using POX, we have precise control over network behavior and can implement customized network management policies. This was critical to developing a Captive Portal solution that could respond to a variety of network environments and user needs.

The combination of Mininet and POX allowed us to develop and test the Captive Portal solution in a flexible, controlled simulation environment. This combination not only utilized the knowledge and experience I gained from Professor Marcelo's course, but also provided our project with a powerful combination of tools that ensured we were able to achieve our goals efficiently and effectively.

3. Deepening Understanding and Application of POX in Captive Portal Development:

In this phase, the focus shifted towards an in-depth exploration and practical application of POX, a vital component in our captive portal setup. In the forwarding folder of POX, I found a file called l2_learning.py that contained a pre-implemented switch code. I chose this as my starting point and began making the necessary changes. The objective was to refine the control over device connectivity based on MAC addresses.

Challenges Encountered

- Dynamic MAC Address Issue: The first challenge arose from the dynamic nature of MAC addresses in the Mininet hosts. This required a tedious process of continually reading the MAC address from Mininet, modifying the POX file accordingly, and re-running POX.
- Inconsistent Test Results: Further tests led to inconsistent outcomes, alternating between success and failure without any changes to the code. This inconsistency prompted a deeper investigation into the underlying problem.

Root Cause Analysis

Upon a detailed analysis, a critical realization was made: the sequence of running POX and Mininet was crucial. It was observed that if POX was interrupted after Mininet had started, Mininet would often not attempt to reconnect to POX.

Specifically, the need to frequently interrupt the POX process to update the MAC addresses in the code led to an unintended disruption in the connection between POX and Mininet. Thus, each time POX was closed for updates and then restarted, there was no guarantee that Mininet would re-establish a proper connection with the restarted POX instance, leading to erratic test results.

Solution Implementation

To address these issues, a novel approach was adopted:

- Integration of a TCP Client in POX: A TCP client was enabled within POX.
- Creation of a TCP Server on Mininet Host: A separate TCP server was set up on the Mininet host's network. This server played a pivotal role in recording the MAC addresses of various components, including the captive portal server, internet, and user devices.
- Verification Process: The TCP client in POX was tasked with requesting verification from the server to ascertain whether a given MAC address was authenticated.

Core Code Implementation

The culmination of this phase was the development of a core code segment in POX. This code effectively managed network traffic based on MAC address authentication. The logic can be summarized as follows:

- Traffic from the Captive Portal MAC:
    - If the destination is valid, the packet is dropped; otherwise, it's directed to the appropriate port.
- Traffic from the Internet MAC:
    - The packet is dropped if the destination is invalid; otherwise, it's directed to the designated port.
- For other sources:
    - If the source is valid, the packet is directed to the Internet MAC port. If not, it's redirected to the Captive Portal MAC port.

This core code signifies a significant step in our captive portal development, demonstrating advanced control over network traffic based on MAC address authentication.

4. Configuring the Mininet Network Topology:

This section outlines the steps taken to configure the Mininet network topology, focusing on controlling IP, DNS, and router addresses for each user. Challenges encountered and solutions implemented are detailed below.

Initial Setup with isc-dhcp-server

Installation and Configuration: The isc-dhcp-server was installed and initiated on a Mininet node. The DHCP configuration included specific options for subnet-mask, routers, domain-name-servers, and host settings.
Configuration Details:

- Subnet Mask: `255.255.255.0`
- Routers: `10.0.0.2 10.0.0.1`
- Domain Name Servers: `8.8.8.8`, `10.0.0.1`
- Host Settings for Captive Portal Server and Internet with specific hardware ethernet and fixed addresses.
- Host Settings for users with dynamic addresses.

Challenges and Modifications

- Issue Encountered: Upon running `dhclient` on the user side, it was observed that only the user's IP could be controlled through the DHCP server, with no effect on DNS and routing address.
- Alternative Approach: To address this, DNS server addresses and router addresses were added to the user during initialization. This deviation from Mininet's automated control was necessitated due to limitations in understanding Mininet's control process.

Integration with POX and Route Configuration

- Routing Addresses: Two routing addresses were added for the user:
  - ip route add 0.0.0.0/0 via 10.0.0.2 metric 100
  - ip route add 0.0.0.0/0 via 10.0.0.1 metric 200
- Routing Logic: `10.0.0.2` is configured as a router with access to the external network, and `10.0.0.1` is the captive portal server address. This setup ensures users first attempt to connect via `10.0.0.2`, redirecting to `10.0.0.1` if blocked by POX.

Captive Portal Server's DNS Handling

- DNS and HTTP Request Handling: The captive portal server is designed to respond to both DNS and HTTP requests. The DNS return address is uniformly set to the IP of the captive portal server to handle network connection issues.
- DNS Responder Code: A Python script using Scapy was implemented to capture and respond to DNS requests. The script checks for DNS question records and sends a DNS response with the captive portal server's IP address.

External Network Connectivity and Final Configuration

- Connectivity Issue: An encountered issue was the external network connectivity with Mininet. Allowing the captive portal server access to the external network prevented it from capturing DNS packets effectively.
- Resolution: A workaround solution was implemented, though it was considered less desirable. It involved limiting the captive portal server's external network access to ensure DNS packet capture.

5. Writing the HTTP Server:

The primary objective of this step is to develop an HTTP server that plays a crucial role in the captive portal system. The server is responsible for handling web traffic, maintaining the state of original URL requests, and facilitating automatic redirection post-authentication.

Challenges Encountered:

A significant challenge faced in this step, like the one in Part 4, was the inability to capture HTTP packets after connecting to the network where the host is located. This issue hindered the process of passing MAC information of authenticated users through the network.

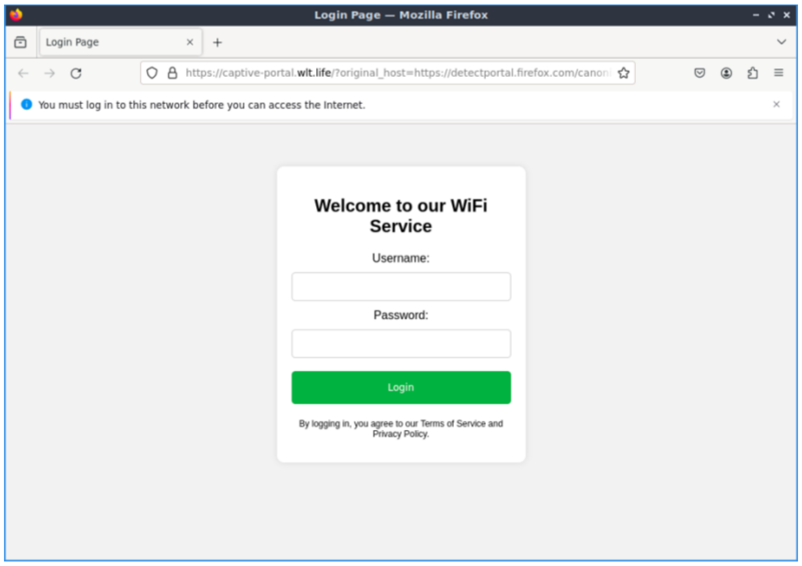
Innovative Solution:

To circumvent this challenge, a unique approach was adopted. The solution involved modifying the contents of a file using Python to append the MAC authentication information. This method ensured that the essential data could be transferred without direct packet capture. Furthermore, an external server, operating outside the Mininet environment, was set up to monitor and react to changes in this file. This allowed the system to transmit the MAC details of authenticated users to the external TCP server, effectively bypassing the limitations encountered within the Mininet environment.

Functionality of the HTTP Server:

The designed HTTP server is equipped with the following functionalities:

- State Preservation: It retains the state of the original URL requests. This is crucial for providing a seamless user experience.
- Automatic Redirection: Post successful authentication, the server is capable of automatically redirecting users back to their originally requested web pages. This feature enhances user convenience and improves the overall efficiency of the network access process.



Login Page — Mozilla Firefox

https://captive-portal.wlt.life/?original_host=https://detectportal.firefox.com/canon

You must log in to this network before you can access the Internet.

Welcome to our WiFi Service

Username:

Password:

By logging in, you agree to our Terms of Service and Privacy Policy.

6. Problems and Outlook

Internet Connection and Device Monitoring in Mininet

A significant challenge we faced was the simultaneous management of internet connections and device monitoring within the Mininet environment. This limitation impacted our ability to efficiently track and manage network traffic and user interactions in real time. The inability to concurrently observe device behavior and maintain internet connectivity posed a substantial hurdle in the development process.

DHCP Control Limitations

Another notable issue was the automated control of DHCP (Dynamic Host Configuration Protocol) by Mininet. This automated process, while efficient in standard scenarios, presented challenges in customizing network configurations specific to our captive portal requirements. The rigidity in DHCP settings hindered our ability to fine-tune the network environment to suit the specific needs of our project.

Despite the challenges encountered, the project has made significant strides in developing a functional captive portal system. The issues identified have provided valuable insights and learning opportunities, paving the way for future improvements and enhancements. As I continue to refine and evolve the system, I will remain committed to overcoming these hurdles with a focus on creating a more efficient, reliable, and user-friendly captive portal experience.