Integrating Captive Portal Technology into Computer Science Education: A Modular, Hands-On Approach to Infrastructure

Lianting Wang
University of Toronto Scarborough
lianting.wang@alumni.utoronto.ca

Marcelo Ponce
University of Toronto Scarborough
m.ponce@utoronto.ca

ABSTRACT

In this paper, we present an educational project aimed to introduce students to the technology behind *Captive Portals* infrastructures. For doing this, we developed a series of modules to emphasize each of the different aspects and features of this technology. The project is based on an open source implementation which is widely used in many computer network courses, making it well-suited and very appealing for instructors and practitioners in this field.

KEYWORDS

Computer Networks, Captive Portal, Computer Science Education

1 INTRODUCTION

A *Captive Portal* is a form of network connectivity where access to the Internet is restricted until certain requirements are satisfied. It presents the access-point provider's requirements through an interface, typically a web-page type, such as reading advertisements, accepting acceptable usage policies, or providing some form of credentials [6].

Nowadays, Captive Portals are ubiquitous in our lives. Whether we are in airports, hotels, or restaurants, Captive Portals have always provided an easy way to authenticate our identity and access the Internet. Although the time spent interacting with Captive Portal interfaces is short and in some cases considered a formality, the technology encompasses a rich set of network protocols and authentication processes. HPC and distributed compute systems can use this approach to validate users and access to resources, e.g. gateways to specialized portals, disclose systems' policies, etc. Automated pop-ups from a mobile phone or computer device further illustrate the versatility and complexity of this technology. These features will stimulate people's curiosity and encourage them to further explore the field of computer network security and infrastructure [1].

The pervasive nature of Captive Portals in everyday scenarios presents a unique opportunity for computer science educational exploration, especially in the field of computer networks. Students are already familiar with the user-oriented aspects of the technology and are standing on the threshold of deeper exploratory learning. By demystifying the core principles of Captive Portals, we can transform students' casual encounters with this technology into profound learning experiences. This educational journey promises not only to help students gain a deeper understanding of network communications and its related protocols, but also to foster critical thinking about the security measures inherent in seemingly mundane digital interactions.

The teaching case model proposed in this paper is an innovative educational tool designed to explore Captive Portal technology in depth through hands-on practice, thereby promoting students' comprehensive understanding of basic network technologies. By simulating the operation of a typical Captive Portal, students are able to intuitively learn and understand the core technical principles in networking, including the reasons for the use of media access control (MAC) addresses, the role of address resolution protocol (ARP) requests, the working mechanism of switches, the processing of domain name system (DNS) requests, the methods of applying web servers, and the key differences between HTTP and HTTPS. This type of learning is not limited to academic exploration; it also helps students relate their theoretical knowledge to the construction and operation of networks in their daily lives, leading to a more intuitive understanding of how networks work.

This model emphasizes the close integration of technology and life, encouraging students to become more engaged in their studies by increasing their interest, while also developing their practical application skills [7]. Through this model, students are able to not only grasp the technical fundamentals of the Captive Portal system, but also critically assess its security and further understand the importance of network security and infrastructure. This pedagogical approach seeks to bridge theoretical knowledge with its practical application, enabling students to apply what they have learned to solve real-life network problems they may encounter, thereby developing their comprehensive understanding and application skills.

2 DESIGN PRINCIPLES

2.1 Introduction to Design Principles

The Captive Portal instructional project is underpinned by a set of core design principles aimed at maximizing the educational value

for students. These principles are crafted to ensure a blend of practical learning experiences with a strong theoretical foundation. By engaging students in hands-on activities involving Software-Defined Networking (SDN) and Captive Portal technologies, the project bridges the gap between conceptual knowledge and real-world applications. The overarching goal is to foster a deep understanding of network communication principles, system architecture, and the challenges of implementing network-based applications.

2.2 Extensive Coding Instructions

Central to our approach is the provision of detailed coding instructions and guidelines. Before embarking on code development, students are required to engage with comprehensive documentation that outlines the system's architecture, design considerations, and specific module requirements. This preparatory step is designed to shift the focus from minutiae to the broader system perspective, encouraging students to appreciate the interconnectedness of system components. Supplementary resources, including helper functions and baseline code, are provided to streamline the implementation process, enabling students to concentrate on solving more complex problems.

2.3 Modular Design

Acknowledging the diverse focus areas and difficulty levels required across different courses, the Captive Portal project is inherently modular. This flexibility allows instructors to tailor the project to fit educational objectives, selecting modules that align with specific aspects of networking, such as SDN or DNS. Moreover, the modularity supports scalability in complexity, making it suitable for a wide range of student expertise. Such a design not only enhances the learning experience but also empowers educators to customize the curriculum to meet students' needs more effectively.

2.4 Critical Thinking and Problem-Solving

The tiered difficulty levels within the modular design introduce a unique opportunity for critical evaluation. Students are challenged to assess the adequacy of the current module design, promoting critical thinking. This evaluative process deepens their understanding of the system, encouraging a thoughtful consideration of design efficiency, scalability, and performance. Through this, students develop a keen eye for optimizing solutions, a skill critical in the fields of HPC and technology development [3].

2.5 Stable Development Environment

Recognizing the complexity of setting up a development environment for such a diverse project, we provide a pre-configured virtual environment. This environment includes all necessary dependencies and tools, ensuring uniformity across all modules. By eliminating the setup barriers, students can immediately focus on the learning objectives and practical experience, rather than grappling with configuration challenges.

2.6 Test-Driven Development

Embracing best practices in software development, the project emphasizes test-driven development (TDD). For each module, comprehensive test suites are provided, guiding students through the
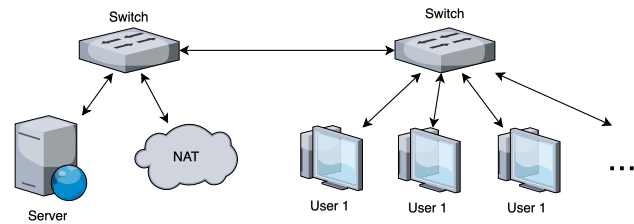

Figure 1: Basic structure for a typical Captive Portal configuration.

development process. This approach not only facilitates immediate feedback on the correctness of implementations but also instills a disciplined coding methodology [4]. By integrating TDD, students learn to prioritize functionality and reliability from the outset, a practice that significantly benefits their future professional endeavors.

3 IMPLEMENTATION

For our project we decided to use an open-source tool named *mininet* –https://mininet.org/– [5, 9], which has been widely used and adapted in many computer networks courses due to its versatility, flexibility and open approach. Employing *mininet* as a *software defined network* (SDN) solution, we developed our main infrastructure for the teaching modules and educational projects.

Fig. 1 depicts a typical infrastructure for a Captive Portal deployment, where users and servers are connected through two Switches.

Using SDN technology, we can manage connections at the link layer, directing users with unique MAC addresses either to the network address translation (NAT) or to the Captive Portal server. However, since the network layer focuses solely on delivering data to the correct IP address without regard to the connection's status, we employ specific techniques to reroute user requests to the appropriate server.

One possible method is the so-called *DNS Spoofing* – see Fig. 2. In this case when a user tries to access an external website, their device sends out a DNS request to find the website's IP address. Normally, this would lead them to the intended website, but we intervene by using a special DNS server. This server directs all domain name requests to the Captive Portal server, ensuring the user encounters the authentication page regardless of their intended destination.

Another approach we use is *IP Forgery*. With this technique, when a user's device sends a DNS request, we provide the correct IP address. However, when the user attempts to access the website, we intercept the web request and respond with an HTTP redirect message that redirects the user to the Captive Portal server's domain name. The user's second connection will connect to the Captive Portal server.

Regardless of the specific methodology used, the framework of the entire project is fixed, as shown in Figs. 3 and 4.
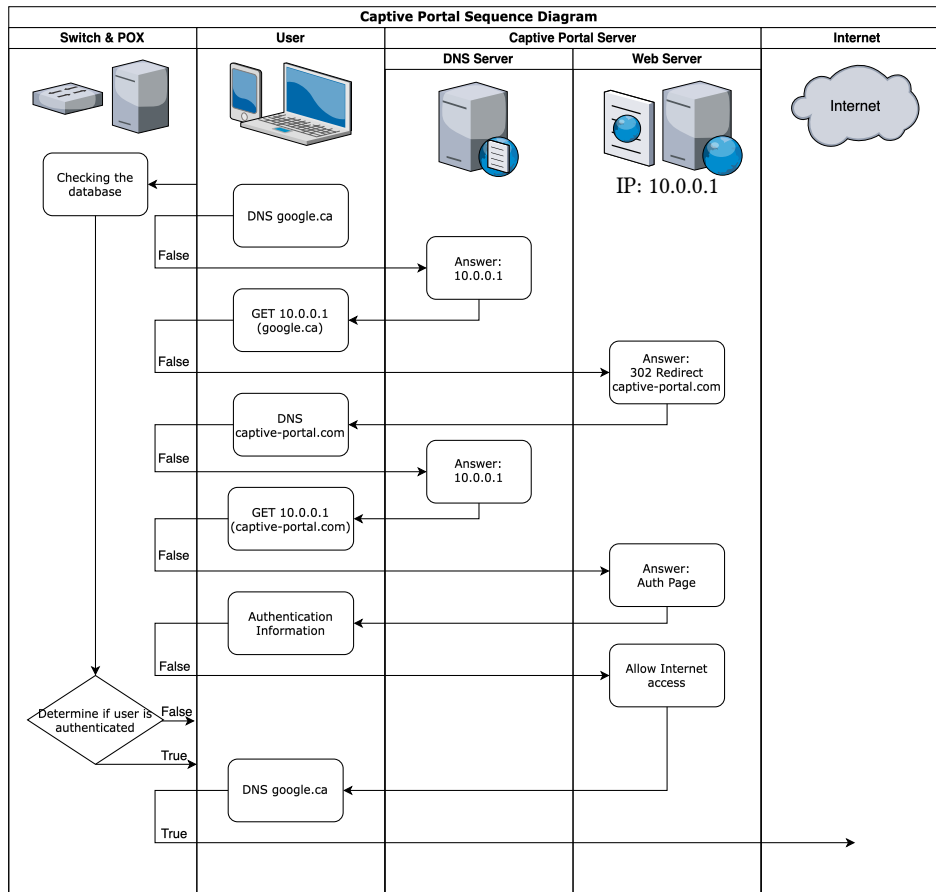
Figure 2: Sequence Diagram of Captive Portal with DNS Spoofing Technology

4 MODULES

In this section we describe the main modules/projects we have originally designed with the goal of guiding interested readers through the different instances of the Captive Portal technology.

The details of the implementations, along with specific examples, demos and any software and configuration details are available in our public repository, https://github.com/Lianting-Wang/Captive-Portal-Education.

Further details about these are also available in Appendix A.

- **Module 1 - TCP Sever and Client**: This module lays the foundational understanding of TCP connections, vital for implementing captive portal projects. It aims to solidify students' network programming skills and introduce them to Python coding for network applications.
- **Module 2 - SDN Switch**: Focuses on understanding Software-Defined Networking (SDN) controllers and the manipulation of network traffic. Through hands-on practice with the POX SDN Controller, students will learn how to program switches and process network packets.
- **Module 3 - DNS Server**: Introduces the functionality and implementation of DNS servers within the context of captive portal technology. This module covers different approaches to handling DNS requests, emphasizing hands-on experience with DNS manipulation.
- **Module 4 - Web Server**: Explores the deployment and configuration of a local web server to interact directly with network requests, crucial for the operation of a captive portal.
- **Module 5 - Mininet Script**: Delves into network simulation using Mininet, teaching students to construct and modify network topologies to route user requests through specific servers, mimicking a captive portal environment.

5 CODE PROVISIONING TOOLS

To facilitate a streamlined educational experience in Captive Portal development, we have implemented a comprehensive suite of code provisioning tools, designed to integrate seamlessly into academic settings. This suite includes a meticulously crafted flowchart –see Fig. 5– and an accompanying website, which together serve as a central hub for instructional resources and starter code packages tailored to various teaching objectives. These tools, in addition to other elements, like demos and tutorials can be found in our publicly available repository:

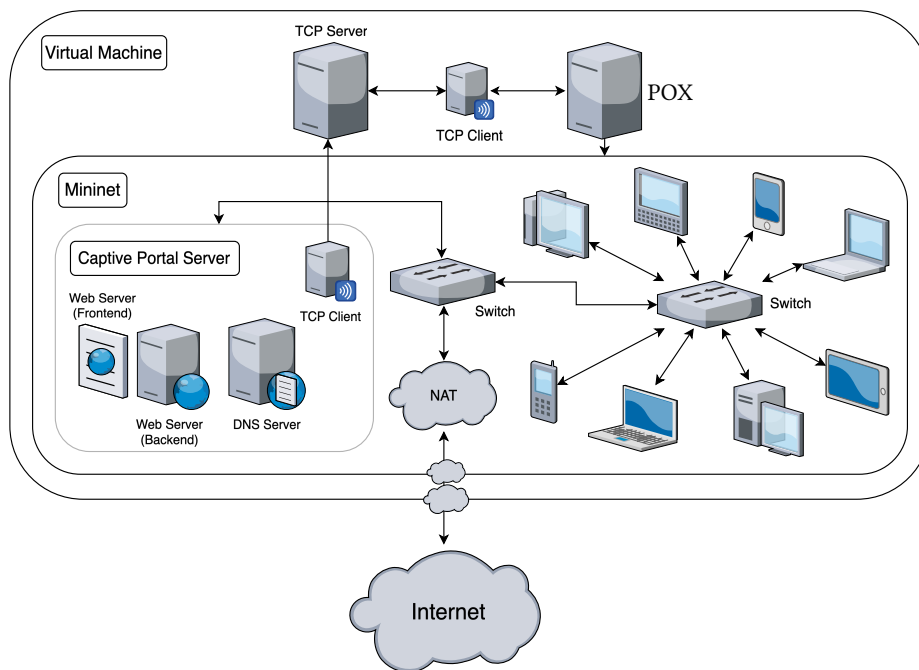https://github.com/Lianting-Wang/Captive-Portal-Education.

Figure 3: Framework of Captive Portal project.

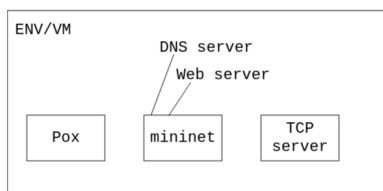Figure 4: Simplified view of the main components of the Captive Portal setup as described in Fig. 3: POX controler, mininet SDN emulator, and TCP server.

In addition to that, we have developed an automated script to assist users with the setup process, as well as, an automated test script, which also verifies whether the environment is configured correctly.

5.1 Flowchart Design

The flowchart, Fig. 5, is engineered to guide educators through a structured selection process, enabling them to identify and choose the most appropriate instructional materials and starting code bases that align with their specific teaching goals and the learning outcomes desired for their students. This visual tool simplifies the complexity of course design by categorizing materials based on topics, difficulty levels, and the progression of concepts essential to Captive Portal development.

5.2 Website Interface

Complementing the flowchart, the website offers an intuitive, user-friendly platform where educators can effortlessly download pre-packaged instructional materials and starting code. This eliminates the traditionally time-consuming process of manual compilation and customization of teaching resources. Educators can make selections based on the guidance provided by the flowchart, with the website offering detailed descriptions, download links, and additional resources for each package.

5.3 Key Features

- Customization: Educators can customize packages to suit the unique needs of their curriculum, allowing for flexibility in teaching advanced topics or focusing on foundational skills.
- Accessibility: The website ensures easy access to up-to-date materials, fostering an adaptive learning environment that can quickly respond to the evolving landscape of Captive Portal technologies.

6 DISCUSSION

The transition to environments supportive of Apple's M-series chips has unveiled challenges, notably in virtualization capabilities crucial for educational settings. Our ongoing efforts to integrate Docker as a foundational component for our instructional environment have encountered obstacles, particularly with the integration of the X Window System. Despite these hurdles, we remain optimistic about resolving these issues, ensuring a seamless experience across diverse hardware platforms.

An alternative to be considered for Apple silicon, would be alternative emulation environments like the one offered by EMU
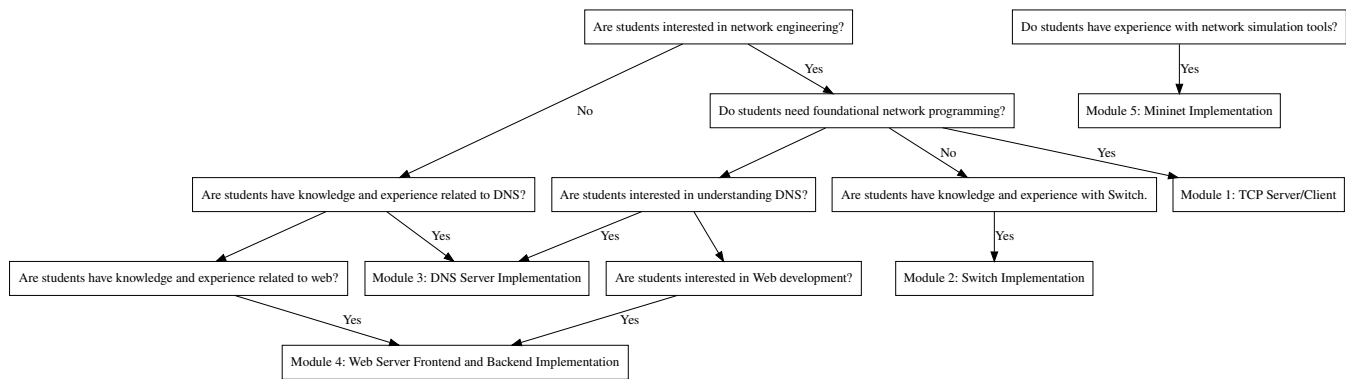
Figure 5: Captive Portal education modules *recomendation system* – flowchart used to determine what modules are recommended to use based on the desired educational targets and goals. Also available as an interactive tool in the web-interface from the Captive Portal educational repository.

(https://getutm.app/) or UTM (https://www.qemu.org/). In these cases, it would require porting the whole infrastructure associated to the SDN layer and additional components of the Captive Portal implementation.

In the past, we have also students attempting a fresh installation and compilation of mininet on native Mac OS with relative success although this it not complexly clear is a solution that implies a substantial efforts commitment as well as is not clear whether it will be sustained in time.

We believe this is important information to share from educators perspective, as it has proven to be an additional annoyance for some students to deal with. The palliative solution we offer some of our students is to use Linux based lab. machines, which can be accessed in-situ as well as remotely – for in which cases, we will teaching technologies such as remote connectivity via ssh, X-forwarding, VNC and even VPN due to universities policies and cyber-security best practices [8].

In addition to platform compatibility concerns, our evaluation of Mininet –sometimes also considered as a lightweight network simulation tool– has revealed limitations in its capacity to simulate hardware intricacies accurately. This shortfall has prompted a quest for more sophisticated alternatives capable of delivering a purer simulation experience. Our aim is to enrich the educational experience by offering students hands-on exposure to configuring a variety of network protocols from the ground up. This endeavor seeks to bridge theoretical knowledge with practical application, enhancing the comprehensiveness of our instructional approach.

7 CONCLUSIONS

In this work we present a module-based project aiming to familiarize students with Captive Portal technologies. We do so, by using a series of small problems employing the mininet SDN tool.

Our project is open source and hosted in publicly available repository, emphasizing relevant aspects of its open source approach, such as reproducibility, modularity and sustainability. Furthermore, because of its openness, it is possible to continue to grow this project, by adding more modules, accepting contributions from the community, etc.

In conclusion, the development and implementation of a Captive Portal model for educational purposes embodies a concrete contribution to enrich computing education. By integrating comprehensive technical content related to computer networks, this model not only illuminates the operational essence of Captive Portals but also paves the way for additional educational practices in the field of computing and information technology.

REFERENCES
[1] Suzan Ali, Tousif Osman, Mohammad Mannan, and Amr Youssef. 2019. On Privacy Risks of Public WiFi Captive Portals. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2019 International Workshops, DPM 2019 and CBT 2019, Luxembourg, September 26–27, 2019, Proceedings* (Luxembourg, Luxembourg). Springer-Verlag, Berlin, Heidelberg, 80–98. https://doi.org/10.1007/978-3-030-31500-9_6

[2] Ramon dos Reis Fontes and Christian Esteve Rothenberg. 2016. Mininet-WiFi: A Platform for Hybrid Physical-Virtual Software-Defined Wireless Networking Research. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 607–608. https://doi.org/10.1145/2934872.2959070

[3] Anita Heijltjes, Tamara Van Gog, Jimmie Leppink, and Fred Paas. 2015. Unraveling the effects of critical thinking instructions, practice, and self-explanation on students' reasoning performance. *Instructional Science* 43 (2015), 487–506.

[4] Christopher G. Jones. 2004. Test-driven development goes to school. *J. Comput. Sci. Coll.* 20, 1 (oct 2004), 220–231.

[5] Faris Keti and Shavan Askar. 2015. Emulation of Software Defined Networks Using Mininet in Different Simulation Environments. In *2015 6th International Conference on Intelligent Systems, Modelling and Simulation*. 205–210. https://doi.org/10.1109/ISMS.2015.46

[6] K Larose, D Dolson, and H Liu. 2020. RFC 8952: Captive Portal Architecture.

[7] Beatriz Pérez and Ángel L. Rubio. 2020. A Project-Based Learning Approach for Enhancing Learning Skills and Motivation in Software Engineering. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 309–315. https://doi.org/10.1145/3328778.3366891

[8] Marcelo Ponce and Ramses van Zon. 2023. Cybersecurity Training for Users of Remote Computing. *The Journal of Computational Science Education* 14 (Nov. 2023), 10–17. Issue 2. https://doi.org/10.22369/issn.2153-4136/14/2/3

[9] Zuo Xiang and Patrick Seeling. 2020. Chapter 11 - Mininet: an instant virtual network on your computer. In *Computing in Communication Networks*, Frank H.P. Fitzek, Fabrizio Granelli, and Patrick Seeling (Eds.). Academic Press, 219–230. https://doi.org/10.1016/B978-0-12-820488-7.00025-6

A MODULES DETAILS

The following provides details about the initial set of modules created for the Captive Portal education project.

Module 1: TCP Server and Client

This module lays the foundational understanding of TCP connections, vital for implementing captive portal projects. It aims to solidify students' network programming skills and introduce them to Python coding for network applications.

Objectives

- Comprehend the architecture and function of TCP servers and clients.
- Understand the reason for using TCP to exchange information in Captive Portal.
- Have a general understanding of the entire Captive Portal framework.

Pre-requisites	Implementations	Learning Outcomes
Basic understanding of TCP packets and python scripting.	Set up a TCP client in Mininet and enable it to communicate with a TCP server hosted in the virtual machine.	Students will gain hands-on experience with network programming, enhancing their understanding of how data is reliably transmitted across a network.

Module 2: SDN Switch

Focuses on understanding Software-Defined Networking (SDN) controllers and the manipulation of network traffic. Through hands-on practice with the POX SDN Controller, students will learn how to program switches and process network packets.

Objectives

- Achieve proficiency in utilizing the POX controller to program a network switch.
- Explore the feasibility of modifying a standard switch to act as a captive portal.

Pre-requisites	Implementations	Learning Outcomes
Basic knowledge of networking concepts, including switches, MAC addresses, and packet flow.	Students are provided with a basic POX code framework and tasked with implementing a learning switch using POX, thus applying their theoretical knowledge in a practical setting.	Enhanced understanding of SDN controllers and the ability to manipulate network traffic for specific applications such as captive portals.

Module 3: DNS Server

Introduces the functionality and implementation of DNS servers within the context of captive portal technology. This module covers different approaches to handling DNS requests, emphasizing hands-on experience with DNS manipulation.

Objectives

- Learn to uniformly respond to all DNS queries with a specific IP address.
- Develop skills to parse DNS requests, make requests on behalf of the user, and return genuine responses.
- Modify DNS request destination IPs using iptables for direct server responses.

Pre-requisites	Implementations	Learning Outcomes
Basic knowledge of UDP packets and DNS protocols.	Students can explore one of three different DNS configuration methods, from simple IP redirection to complex request handling and response modification.	Students will understand how DNS servers can be manipulated to redirect or handle requests in a captive portal setup.

Module 4: Web Server

Explores the deployment and configuration of a local web server to interact directly with network requests, crucial for the operation of a captive portal.

Objectives

- Set up and configure a local web server.
- Use iptables to redirect web requests to the local server.

Pre-requisites	Implementations	Learning Outcomes
Basic understanding of web server operation and network request handling.	Students will configure iptables to redirect web requests to a local server, ensuring direct response capability.	Practical skills in web server setup and configuration, with an emphasis on interaction with network requests in a captive portal context.

Module 5: Mininet Script

Delves into network simulation using Mininet, teaching students to construct and modify network topologies to route user requests through specific servers, mimicking a captive portal environment.

Objectives

- Understand network structure and the necessary modifications for captive portal functionality.
- Learn to use Mininet for creating and testing network topologies.

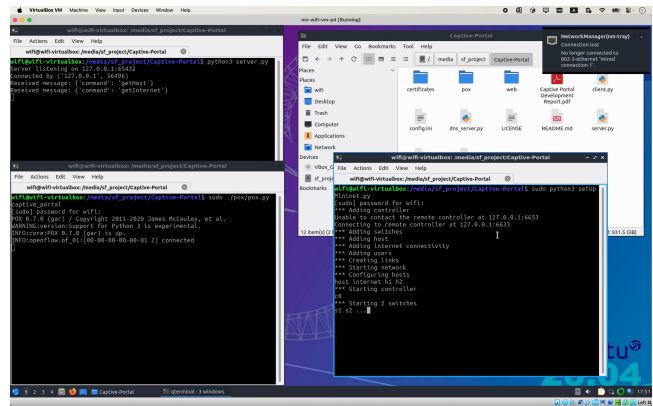| Pre-requisites	Implementations	Learning Outcomes
Basic knowledge of network topology and mininet.	Students have the flexibility to design the network according to the requirements of the captive portal setup they envision.	Ability to simulate and test network configurations, crucial for the development and deployment of captive portal technologies.

B MANUAL CONFIGURATION & GUI ATTACHMENT

For advanced courses and users of Mininet who wish to gain deeper insights into the technology and procedural setup, the following steps provide a brief introduction to run Mininet with a graphical user interface and additional networking tools:

(1) Initiate the virtual environment that hosts the Mininet installation.

(2) Start the POX controller and server setup as shown in Fig. 6a.

(3) Open additional terminals or sessions to run the corresponding web and DNS servers as in Fig. 6b.

(4) Launch an X-terminal to run a web-browser and optionally a network packet sniffer, such as `wireshark`, as illustratged in Fig. 6c.
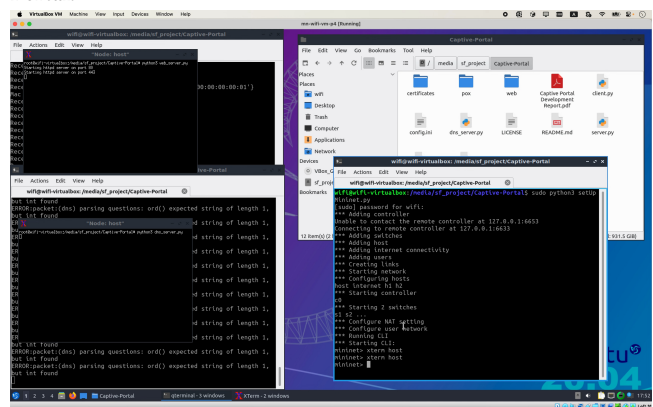
This process allows users to manually configure and customize their Mininet setup, including the option to run graphical user interfaces (GUIs), trigger browser pop-ups typical of Captive Portals, or capture network traffic using `Wireshark`[1]. It also enables users to explore other Mininet base configurations, such as wireless mininet setup [2].

However, it is important to note that these steps are meant to demonstrate what the final product will look like once students have completed all or part of their project code, regardless of how many modules the students complete, the final product will be a fully functional Captive Portal, it is not an exact reflection of how students will test their code.
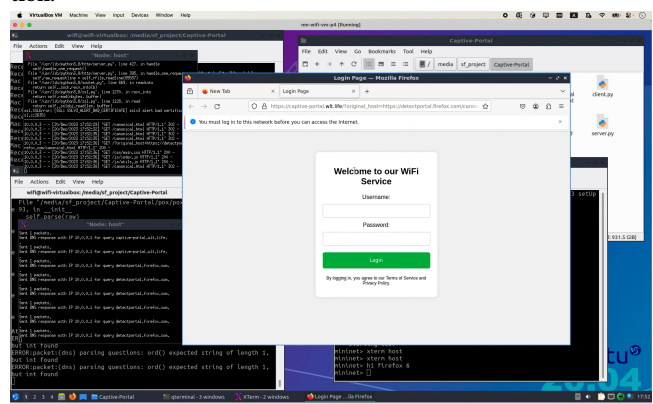
Notice that to properly demonstrate these capabilities, some additional configuration steps are required beyond what is shown in the images. These additional steps, as well as more specific configuration details, will be provided in the project's GitHub repository.

(a) **Configuration setup in a VM for running mininet as a Captive Portal.**

(b) **Setup of DNS server with the mininet Captive Portal configuration.**

(c) **Demonstration of the DNS server interception of network traffic via a web-browser request.**

Figure 6: Demonstration of the sequence of steps required to setup and execute the DNS spoofing technology to forge the Captive Portal allowance.

---

[1]https://www.wireshark.org/