Final Report

NetBSD Wifi Browser Project 2022

Dylan Roy, Stephen Loudiana, Kevin McGrane

Summary of Project

The NetBSD Wifi Browser Project was executed in order to bring a way to simplify the wifi connection process in NetBSD-base. To support this goal, a library was developed called `Surf` to handle the interfacing with `wpa_supplicant` and handle the configuration of networks to the extent necessary for a complete connection. The `Surf` library is intended to be used in the development of user interfaces that handle interactions with wifi connections, including connecting, disconnecting, forgetting, and manually configuring how `wpa_supplicant` is to connect to a network. As an example of how `Surf` could be used in a user interface, as well as to provide interfaces for NetBSD-base to connect to wifi access points, a CLI (command line interface), a TUI (terminal user interface), and a GUI (graphical user interface) were in development in addition to the `Surf` library. A fully interactive command line interface is included in the Wifi Browser Project. The CLI can interact with all methods of the Surf library. These methods help users interact with the configuration file. The CLI automatically connects users to `wpa_supplicant` at startup and allows users to add, edit, or delete networks from the configuration file. In addition to listing available and configured networks, connecting and disconnecting from already existing connections, auto configuration of networks and eap networks, clearing the screen and exiting the interface. The CLI is fully interactive with all methods in the surf library and allows users to edit the fields in the configuration file and connect to `wpa_supplicant`.

Details of Completed Parts

The only part of the project that reached total completion was the `Surf` library. `Surf` interfaced with `wpa_supplicant`'s own API, `wpa_ctrl` in order to edit the configuration file for `wpa_supplicant` without the need for root privileges. Through `wpa_ctrl`, `Surf` was able to support automatic configuration for basic networks that rely on PSK's (pre-shared keys), or WPA-PSK networks, and networks that rely on usernames and passwords, or WPA-EAP networks. So in essence, the minimum amount of information required to connect to a specific network will typically be only the SSID of the network and its PSK, or the SSID of a the network and a username and accompanying password. In addition to automatic configuration, `Surf` supports editing network configurations for advanced users, forgetting networks from known networks, disabling and enabling networks without forgetting, and listing visible networks in range of the connected wireless device and the known networks to `wpa_supplicant`. These features will allow a user interface implementer to worry only about presenting information and getting information from the user instead of also dealing with how to interface effectively with `wpa_supplicant`.

CLI has reached full completion on essential implementation of required parts. Users can fully interact with all methods in the surf API. Common commands users can enter include:

1. Connect to existing connections in the configuration file
2. Disconnect from an already existing connection
3. Add a new entry to the configuration file

4. Allow for both auto configuration and auto configuration of eap networks in the configuration file
5. Remove an already existing network in the configuration file
6. Edit an already existing network in the configuration file
7. List all configured networks in the configuration file
8. List all available networks
9. Connect to `wpa_supplicant`
10. Check the current connection
11. Clear the terminal screen and exit program

Commands for the CLI involve common words such as 'connect', 'disconnect', 'forget', 'edit', 'current', 'list', 'configure', 'add', 'clear', and 'exit'. Help methods with the CLI are detailed and easy for users to understand and include the correct syntax for each command. CLI has required implementation for users to edit and interact with existing networks in the configuration file, connect and disconnect from already existing networks, list networks, in addition to descriptive help methods given with each command.

Details of Incomplete Parts

CLI

CLI does not grant users the ability to enter commands using hyphens for shortened commands. Additionally, multiple commands cannot be processed simultaneously. However, implementation for standard `getopt_long` would require minimal implementation. Standard string comparsions are used for some commands where hyphens are detected for user input. Additional implementation would require `getopt_long` to detect all commands with each function call in the command table. Additional implementation for listing the fields associated with an existing network in the configuration file, in addition to detecting whether or not a network connection is an eap network would be required for full completion of the CLI. Additional methods might be preferred by the developer to be considered complete.

TUI

GUI

Since `Surf` was completed many weeks before the end of the senior project cycle, the scope of the NetBSD Wifi Browser Project was expanded to include the development of GUI implementation for `Surf`. In order to stay in compliance with NetBSD-base, the GUI was developed using the bare bones `XLib` library, which is the basic graphics library included with `XOrg` (the default display environment on NetBSD). This meant that development of the GUI would also include developing a library of graphics elements to simplify the development of a GUI interface. As such, combined with only 5 weeks of time left in the project, the GUI was only able to reach a state of listing the known and available networks. Progress was being made towards actually connecting to one of the available networks, but was ultimately not achieved due to a lack of time and a complexity of the task being performed.