The [redacted] Project - [redacted] proposal

Name: Emanuele Micheletti
Slack Username: [redacted]
Email Address: [redacted]
Primary Spoken Language: Italian

1. Please describe your preferred coding languages and experience.

I have experience working with several programming languages including Rust, C, and Python. My current focus is on **Rust**, where I am currently developing a microkernel that can run wasm at ring 0. Although the project is still in its *early* stages, I am making good progress and actively working on it. I have created a mitm proxy with ssl/tls capabilities that is currently in active development, this project has allowed me to gain experience in networking. Additionally, I maintain a blog where I document my projects and share my experiences with others.

Apart from systems programming, I also have experience in web development with **JavaScript** and related frameworks such as *React*, *Vue* and *Svelte*. I have created several applications for customers and for personal use. Furthermore, I created an e-commerce platform entirely in PHP to master the language. This project involved developing and integrating several features, such as a shopping cart, payment gateway, and a customer management system. Overall, my experience with various programming languages has allowed me to develop a versatile skill set that I believe will be an asset to any team.

3. Please describe any previous open source development experience.

- **Man In The Middle Proxy**: A MITM Proxy Written in Rust 🦀! Toolkit for HTTP/1, HTTP/2, and WebSockets with SSL/TLS Capabilities
- TakaraOS: 💎 A microkernel OS written in Rust that runs Wasm at level 0, allowing developers to write self-contained modules in multiple programming languages
- Nuxt-swipe: The Nuxt-Swipe plugin allows you to easily add swipe gesture support to your Nuxt3 projects.
- CHIP-8 Simulator: Added some features to base CHIP-8 Simulators from Rust In Action book

Through these contributions I gained experience working with Git and collaborating with other developers on open source projects.

4. What school do you attend and what is your specialty/major at the school?

I have a Bachelor of Science degree in Software Engineering at Politecnico di Torino, where I gained a deep understanding of software development methodologies and various software technologies. Additionally, I completed a Master of Science degree in Financial Math Engineering, where I focused on the development of financial software applications and gained experience working with various financial models and optimization algorithms.

5. How many years have you attended there?

3 years for Bsc and 2 years for Msc

6. What city/country will you be spending this summer in?

I am currently planning to spend this summer in Turin, Italy. However, I may travel to other locations within a one-hour time difference from the Italian timezone (UTC+1 to UTC-1) during this time (*but still to work, not on holiday*).

7. How much time do you expect to have for this project?

I have completed **all of my exams for this year** and I am now fully committed to delivering high-quality work for this project in a timely manner. I am willing to invest the necessary time and effort to ensure the success of the project and I am prepared to work *full-time* to meet the project's goals and deadlines.

8. Please list all jobs, summer classes, vacations, exams, and/or other commitments that you'll need to work around.

I have completed all of my exams, I do not have any other jobs or summer classes planned for this summer. However, I will be absent from **August 2nd** to **August 3rd** due to a prior commitment and I think I will take that whole week of 2-3 august as break addition. I will be dedicating a *minor part* of my time to contribute to my other open source projects as a code reviewer and developer. Despite these commitments, I am fully committed to this project and will make sure to prioritize my work on it to ensure its success within the specified timeframe.

9. Have you participated in any previous Summer of Code project? If so please describe your project and your experience.

This is my **first time** participating in the Summer of Code project.

10. Have you applied for (or intend to apply for) any other GSoC 2022 projects? If so, which ones?

I have contributed to the CERN project and I am considering applying for their Summer of Code program as well. But my first preference remains on this project.

11. If you have a URL for your resume/CV, please list it here.

- [redacted]
- /in/emanuelemicheletti/

12. If you wish to list any personal/blog URLs, do so here.

- Linkedin: /in/emanuelemicheletti
- Github: /emanuele-em
- [redacted]

13. Project Plan

As part of my proposal for gsoc with [redacted] I plan to implement a new API for starting and stopping the os proxy to intercept and modify own network traffic, initially focusing on macOS with the possibility of expanding to Linux. The implementation will most likely be done using *Rust*.

The proposed API will consist on adapting `start_os_proxy()` and `stop_os_proxy()`, that currently works only on Windows (trough `packet_sources/windows.rs`), to MacOS and Linux, I will create a new crate `osx.rs` that will use properly osx-specific APIs.

In addition, I will also adapt the existing `--mode [redacted][:pid or executable name]` command-line option to allow users to specify a *PID* or *executable name* for the process to be intercepted by[redacted]

To better familiarize with the basecode my intention is to solve a related issue: #5910 is a good candidate to achieve this goal, it is related to[redacted]mode but the environment is very similar to the main path.

Possible approaches to adopt

There are two possible approaches that can be taken at the moment

1. The first is to use **packet filter (pf)** for macOS, initially trying to swallow all traffic coming from the same machine that is launching **packet filter** itself. The second step in this approach will be to use some gimmick to figure out which packets are actually coming from [redacted] and filter them out to avoid loops, a possible solution for this could be to use **nettop**, launching it as a command directly from *rust*, the solution is not definitive and possible alternatives should be explored to understand the privilege issue as well.
2. A second approach is to use **NEAppProxyProvider**: an Apple proprietary framework that extends the functionality of the iOS and MacOS network stack. With **NEAppProxyProvider** we can implement custom rules for filtering, routing, and modifying network traffic based on a variety of criteria, such as the app, the network protocol, the destination address, or the user identity. As it is a proprietary framework, we need to develop in *Swift* or *objective-C* and then transmit the information to the main [redacted]

How other[redacted] actually work

- [redacted] wireguard has a apple version written in Swift, the adopted approach is the second one, it uses Apple Network Interfaces [source]
- [redacted] the same problem is currenlty present on BSD-based system (as MacOS) due to `pf` firewall `iptable`, as explained above
- [redacted] it uses POSIX standards and nothing OS Specifc, they do not even provide the possibility of transparent proxy over HTTPS protocol

My intention is to create a shared document with my mentor so that I can enter all my progress and any concerns day by day

Timeline

The following timeline outlines the tasks and milestones that I plan to accomplish within the 12-week period of gsoc:

- **Pre-GSOC Work**
  - Familiarize myself with [redacted] codebase and API
  - Research existing solutions for implementing `start_os_proxy/stop_os_proxy` on macOS
  - Solve [redacted] issue

- **Week 1:**
  - Study the feasability of two approaches in depth, creating some demo proxies.
  - Choice of possible approach after due tests
  - Set up a development environment with necessary tools for macOS development such as Apple developer account and similar.

- **Week 2-3:**
  - Continue the development of the API.
  - Write unit tests for the basic API.

- **Week 4:**
  - Refactor the proof-of-concept into production-quality code
  - Write documentation and user guide for the new macOS variant of `start_os_proxy` and `stop_os_proxy`
  - Submit the code for code review and incorporate feedback

- **Week 5:**
  - Implement the `--mode osproxy[:pid or executable name]` flag on macOS following one of the two approaches
  - Write test cases for the new flag and debug any issues

- **Week 6:**
  - Refactor and optimize the code for the `--mode osproxy[:pid or executable name]` flag
  - Write documentation and user guide for the new flag
  - Submit the code for code review and incorporate feedback

- **Week 7:**
  - Buffer time for potential delays caused by macOS implementation.
  - Start working on porting `start_os_proxy` and `stop_os_proxy` to Linux, the approach should be similar to the Windows one ,
  - Research and experiment with Linux related rust crates.

- **Week 8:**
  - Buffer time for potential delays caused by macOS implementation.
  - Develop a proof-of-concept for `start_os_proxy` and `stop_os_proxy` on Linux
  - Write test cases for the proof-of-concept and debug any issues

- **Week 9:**
  - Refactor the proof-of-concept into production-quality code

    - Write documentation and user guide for `start_os_proxy` and `stop_os_proxy` for Linux version
    - Submit the code for code review and incorporate feedback

- **Week 10:**
    - Implement the `--mode [redacted]:pid or executable name]` flag on Linux
    - Write test cases for the new flag and debug any issues

- **Week 11:**
    - Refactor and optimize the code for the `--mode [redacted]:pid or executable name]` flag on Linux
    - Write documentation and user guide for the new flag on Linux
    - Submit the code for code review and incorporate feedback

- **Week 12**:
    - Break

- **Week 13:**
    - Final testing and debugging of the implemented features on both macOS and Linux
    - Write a final report on the project and document the implementation details, challenges, and future work
    - Submit the final code and report to the mentor for review
    - Buffer time for problems or delays throughout the period.