Step 0: Basics

* AWS
* [Gymnasium](https://gymnasium.farama.org/index.html) and [RLlib](https://docs.ray.io/en/master/_modules/ray/rllib/env/multi_agent_env.html)
* gRPC and Protocol Buffers
  + [What is RPC?](https://www.geeksforgeeks.org/remote-procedure-call-rpc-in-operating-system/)
  + [Intro to gRPC and Protocol Buffers](https://grpc.io/docs/what-is-grpc/introduction/)
  + [More on Protocol Buffers](https://protobuf.dev/overview/)
  + [gRPC Basics in Python](https://grpc.io/docs/languages/python/basics/)

-----------------------------MUST DO-------------------------------

Step 1: Baseline

* Interact with AWS in command line as a defender
  + <https://github.com/aduyinuo/2024REUSE/blob/main/daedalus_network/c2/defender_cli.py>
* Interact with AWS in command line as an attacker
  + <https://drive.google.com/drive/folders/1uKhD0WP4P7ds5dahJfI225Ye0vEtb-aO>
* Implement an attacker agent
  + in Python code or with Caldera
  + Robustness test
* Manually defend against the autonomous attacker
  + Empirical measurements

Step 2: Real2Sim Translation

* Implement the API for defender
  + <https://github.com/aduyinuo/2024REUSE/blob/main/daedalus_network/c2/network.py>
* Integrate defender’s API into the Gymnasium environment
  + Example: <https://github.com/aduyinuo/2024REUSE/blob/main/daedalus_network/yinuo_code/cybersec.py>
* Interact with AWS through the gym wrapper

Step 3a: Train an RL defender on AWS

> Unsure about the feasibility

* Stress testing of the attacker agent and defender API
* Try to train a defender agent with an off-the-shelf algorithm

Step 3b: Implement a simulated environment

> If training in the real network turned out to be infeasible

* Implement a simulated attacker
* Implement the API for *simulated* defender
  + Simulate the effect of the defense actions instead of actually execute the actions in the real network on AWS
* Train a defender agent with an off-the-shelf algorithm

-------------------------STRETCH GOAL------------------------------