1 - This past summer, I worked with Yinuo to begin the development of an autonomous cyber defensive agent, and we left off with a lot of exciting work to continue doing, and I wanted to share some ideas with the group for the next steps our project will take.

Today I’ll be presenting on two approaches I explored in literature for our project in cyber deception, in particular, approaches to train an effective defender with Model-Based Reinforcement Learning and Sim2Real Transfer.

2 - The agenda for the presentation consists of three main parts. 1) I will first discuss the progress and current state of our project, developing a network in the cloud with a defensive agent, aimed to trick attackers. 2) I will then talk about the main obstacle we face with effectively training a defender agent: sample efficiency which led me to learn more about a few papers that address the problem of sample efficiency which could prove useful for our problem’s context that I wanted to share with the group today. These two approaches are Sim2Real Transfer and Model-based RL

3 - Built-in deceptive measures in a network is a countermeasure used in cybersecurity that can make an attacker’s job harder, by evoking confusion, doubt, wasted resources and dead-ends. I read a paper that states deceptive actions are useful even after discovered as they create doubt in attacker actions. This also gives time for cybersecurity specialists to investigate attacks. If we can trap an attacker with our deceptive tactics, we can investigate who they are and what their intentions are.

4 - One downside to these built-in measures is that they can become ineffective as attackers learn their way around the system, which creates a need for adaptive deceptive defense measures.

5 - Previously in REUSE, I deployed a real-network in AWS with an integrated policy network that is able to observe the network through an observation vector, decide a defensive action, and perform the action on the network. Actions include deceptive acts such as introducing fake data, introducing fake permission to the attacker, and deploying honey services to isolate the attacker

6 - Our next step is to develop an adaptive defensive agent that is able to respond to attackers in real cyber time. Our previous policy network was randomly initialized and served as a proof-of-concept, but we are now looking to make it take more reasonable, informed actions.

7 - The environment step is selected by us to emulate a reasonable period of time where a network might experience variation. For example, seeing a network state before and after an attack, an environment step of 10 seconds would allocate sufficient spacing to capture these changes. It also takes into account the time taken for a defensive action taken by the policy.

8 - By training in a simulated environment then transferring the learned policy to the real-environment, we are able to speed up the learning by generating samples in simulation

9 - The motivation behind Sim2Real gap modeling is that training policies in simulation is efficient, but real-world deployment can fail due to *sim-to-real gap*, a discrepancies between the simulation and real world environment. The objective of Sim2Real Transfer is bridge this gap and transfer policies trained in simulation to work effectively in real-world environments.

10 -

11 -

12 - In MBRL, world-models are prone to errors due to limited data. Ensemble network can mitigate this by combining multiple models, reducing overfitting and uncertainty in long-horizon predictions.

This is the key method of MBPO, which trains NN models to predict next state & reward given a current state and action pair.

13 -

14 -

Initialize policy, predictive model, and replay buffers, one to store real environment transitions and one to store simulated model transitions

We first train the predictive model from sampling environment data from D\_env and train it to minimize error in predicting next state and reward.

Sample an action from the current policy and interact with real environment to obtain the next state. Then store the transition in the environment buffer D\_env.

Then for each model rollout M, sample a state from the environment buffer D\_env and perform a k-step rollout starting from the state, using the learned dynamics model and current policy.

Store the generated transitions in the model buffer D\_model

Then for each gradient update, sample data from the D\_model and perform a policy update using these simulated transitions to optimize the policy parameters.

15 - As seen in training curves, MBPO is able to learn a more effective policy much faster than its MFRL counterparts.

16 -

17 - Dreamer is a MBRL algorithm that relies on a learned latent world model to predict future trajectories and trains policies in that latent space. This allows Dreamer to solve tasks by imagination without interacting with the environment directly, which makes it very computationally efficient.

18 - In this stage, Dreamer learns the dynamics of the environment.

Dreamer begins by collecting **trajectories** of observations, actions, and rewards from the environment. These are stored in the **replay buffer**.

Next Dreamer samples sequences from the replay buffer and encodes the sequences into latent states with its representation model.

The transition model then predicts the next latent state given the current state and action.

Then the representation, transition, and reward model is updated

19 - Once Dreamer has learned a good world model, the next step is learning an effective policy that allows the agent to choose the best actions.

Dreamer computes the value estimates which is used to then update the action model

20 - Dreamer starts by calculating latent state from past observations and actions, and

predicts next action to take based on current latent state

Takes action in the environment

Collects next observation and reward from the environment

21 - Dreamer achieved high performance in significantly less interactions, and was able to generalize across multiple tasks, in both deterministic and stochastic environments.

22 - Dreamer latent imagination would allow many more learning updates than possible in real-world interaction. able to outperform model-free approaches in terms of final performance, data efficiency, an computation time in a variety of tasks

23 - This includes:

* State transitions: how changes in network traffic, attacks, of defenses, alter the network’s state
  + Reward model: Quantify the effectiveness of defense actions (minimize attacker progression)\

Assumptions for attacker deterministic or stochastic –

* Maybe domain randomization for network to improve
* More uncertainty about the reward, then prioritize that policy
* Be clearer about presentation formulation
* Multiple attackers, during testing