SAND2021-8147C

MLDL

Machine Learning and Deep Learning Conference 2021

Tysons Whirlwind of RL Projects

- Tyson Bailey / SNL

- DOE, LDRD, and 5680 POC Tourney.

First a joke (or 3)

Why did the chicken cross the road?
Well he was being thrown off the farm.

I have a machine learning joke,
but it is not performing as well on a new audience

I had a joke about reinforcement learning
but last time no one laughed, so I won't bother telling it again

Sources:
https://github.com/enzoampil/tito-joker
https://www.kdnuggets.com/2020/08/joke-about.html
My Brain

First a joke (or 3)

Why did the chicken cross the road?
Well he was being thrown off the farm.

$\leftarrow$ AI Generated

I have a machine learning joke,
but it is not performing as well on a new audience

I had a joke about reinforcement learning
but last time no one laughed, so I won't bother telling it again

Sources:
https://github.com/enzoampil/tito-joker
https://www.kdnuggets.com/2020/08/joke-about.html
My Brain

Abstract

I tend to work on a wide range of things and wanted to share the projects I'm working on. I would like to share some information/thoughts/results on 3 ML/DL/RL type projects and related tooling.

Frameworks used:

https://cee-gitlab.sandia.gov/gym/sicke

https://cee-gitlab.sandia.gov/vahvis/agents

RELACSS

Reinforcement Learning Approach for Cyber Security in Space (RELACSS) is an LDRD with the goal of detecting threats in a Groundstation and Satellite network. This is early work.

For the underlying system we have a low fidelity model and a high fidelity model.

For the reinforcement Learning agent on the attacker side we may need a multi layer agent.

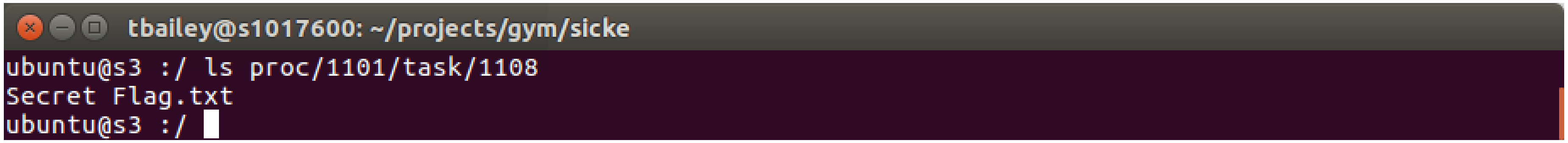
This enables us to train an agent to learn generally what kinds of attacks to perform in the first layer, the second layer will be choosing the specific kind of attack necessary to accomplish that goal.

RELACSS – Low Fidelity Model

For the Low Fidelity Model we are using Simulated Intelligent Capture the Key Environment (SICKE).

SICKE is intended to be a low overhead python based environment which will allow Reinforcement Learning agents the ability to interact with what appears to them to be computers in a network. SICKE mimic's behaviors of computers (and uses similar syntax) to allow for massive parallel training with complete state control. Also complete inspection of the system being used to train.

SICKE can load 70 "computers" in under 1 second with around 100MB of memory required.



```
ubuntu@s3 :/ ls proc/1101/task/1108
Secret Flag.txt
ubuntu@s3 :/
```

Example showing ls being used in SICKE

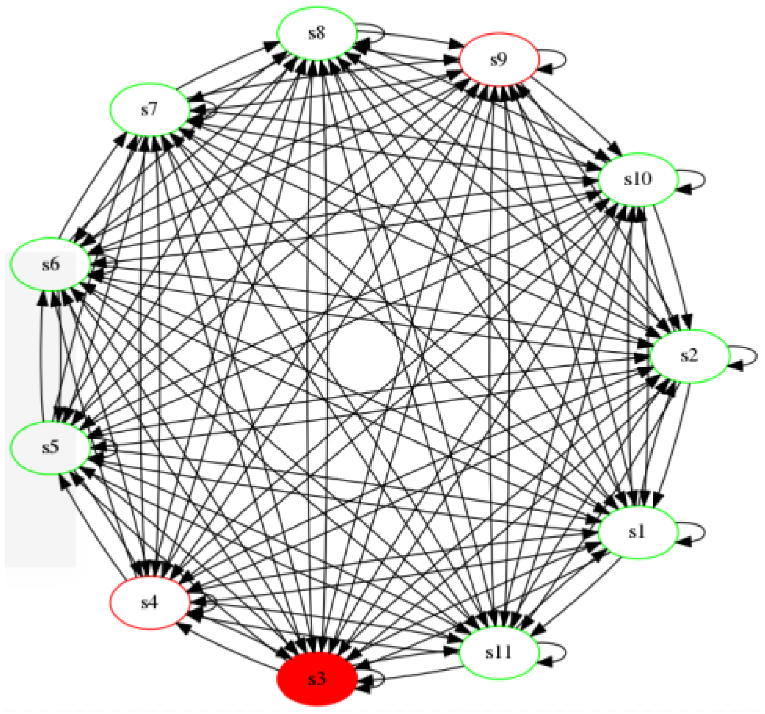
https://cee-gitlab.sandia.gov/gym/sicke

RELACSS – High Fidelity Model

This is a computer network built using VM’s and consists of various kinds of machines, and at least one Kali box.

Our high fidelity model we believe would mimic a real groundstation network. (And allow for emulating an attack).

Example Fully Connected Network



RELACSS – Layer 1 Actions

Layer 1 Attacker

Actions:
Initial Access
Persistence
Privilege Escalation
Credential Harvesting
Collection
Lateral Movement
Command and Control
Defense Evasion
Exfiltration
New Objectives

Potential Defense Actions

Actions:
Format Machine
Run Antivirus
Segregate Machine from rest of network

RELACSS – BASICS Connection

BASICS uses a decision support tool, which takes a Partially Observable Markov Decision Process (POMDP) and solves it, and then provides the user with recommendations based on the POMDP solver, to defend a network using actions such as blocking different segments of the network and cleaning the machines that are infected.

The overlap of the work is the idea that you have a network with actions being taken against it, and then a defender that attempts to stop the attacker.

We spent time implementing agents and a SICKE network for the BASICS work prior to the RELACSS work. The following slides will show the prior work that is becoming a part of the RELACSS work.

BASICS

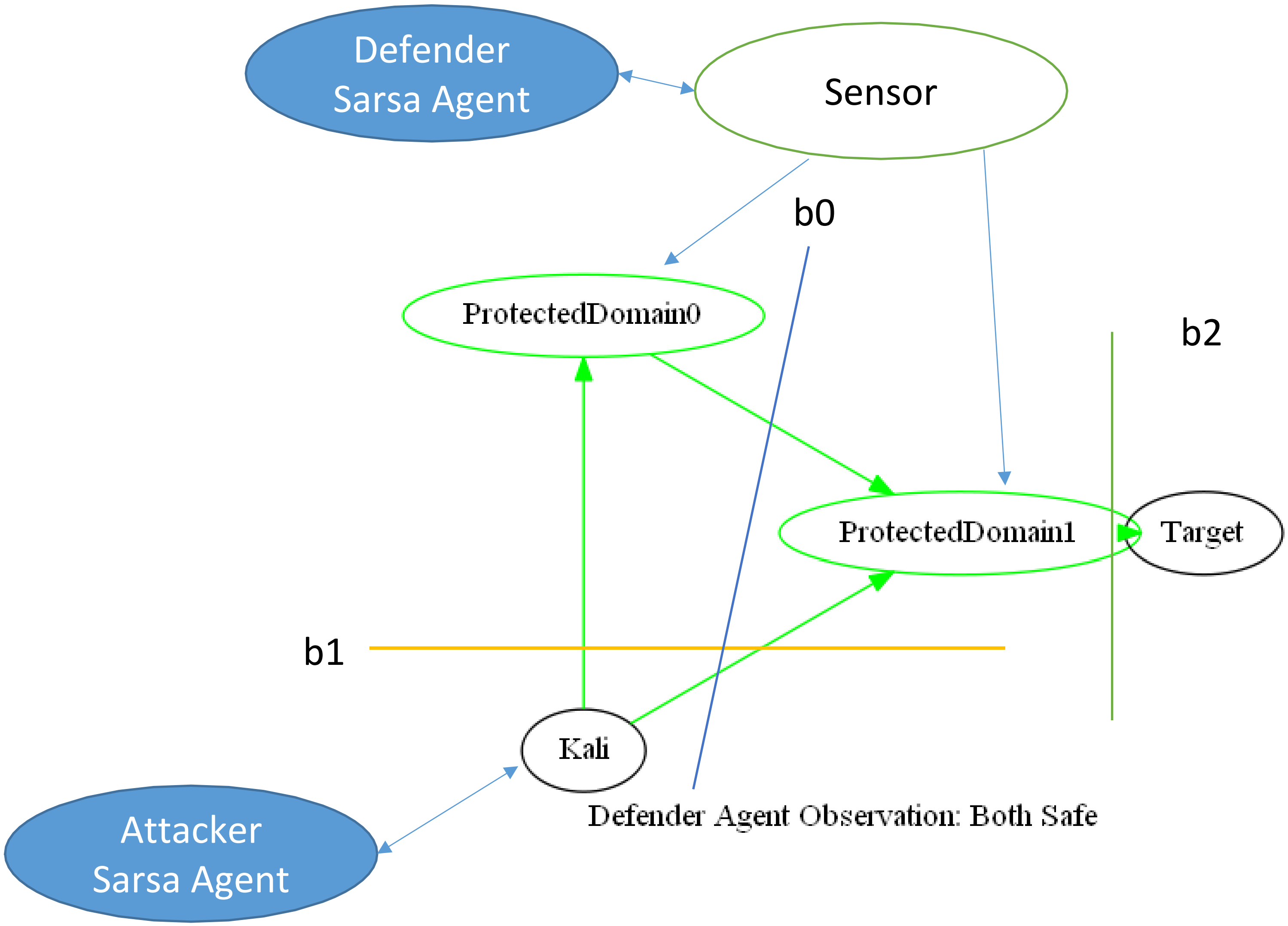
Structurally the network was broken into 2 protected domains, with a target behind one of the protected domains. There is also a “kali” machine that is connected to the protected domains by default. These machines are all connected through a firewall which defines the connections.

The firewall can block connections and then prevent the kali box from reaching a combination of the domains.

As part of this implementation we built a kali agent and a defender agent which make the decisions for how to react in the environment.

The defender has only partial insight into the state of the network. So it has to make decisions with limited understanding.

BASICS -- RL


Defender Sarsa Agent
Sensor
b0
ProtectedDomain0
b2
ProtectedDomain1
Target
b1
Kali
Defender Agent Observation: Both Safe
Attacker Sarsa Agent

BASICS - Results

- Turn Based
- Synchronous
- Defender has access to all of its network devices at all times
  - Observations are noisy but are provided as part of decision making
- Attacker can only access devices the firewall allows
  - Observations are precise but are considered an action
- 500 timesteps or actions available per agent

The agents have different reward structures so it's possible for both to grow and is not a zero sum reward structure.

Attacker:

PD0 and PD1 infected == 3

PD0 or PD1 infected == 2

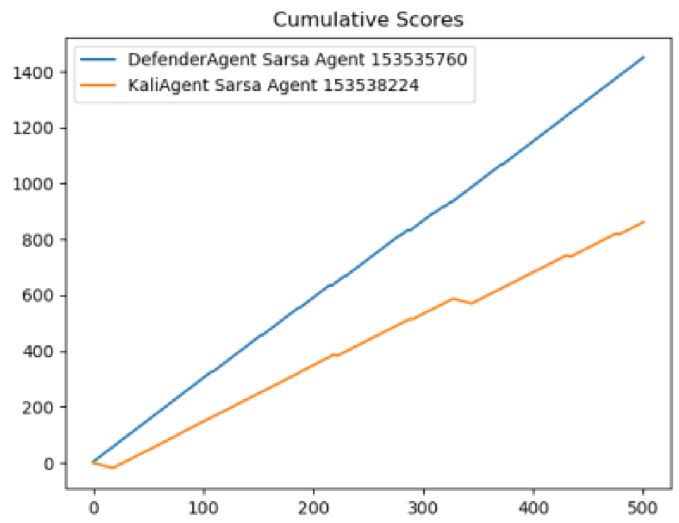
Neither Infected == -1

Defender

PD0 and PD1 infected = -3

PD0 or PD1 infected = -1

Neither infected = 3



RELACSS/BASICS -- Wrapup

Future experiments will involve using PPO, A2C and DQN type networks as they have(are) developed.

The Satellite Ground station project will use much of the same infrastructure as above, but with different actions, and a different network structure.

Major challenges for the attacker:

- What to do on a machine (and avoid detection)
- Where to move in the network (and avoid detection)

Major challenges for the defender:

- The size of the data/network and the amount of introspection available

Both of the agents will require designing a reward structure that balances simplicity and complexity.

Shifting Sand

Shifting Sand's goal is to take binary files compiled for both arm and x86 and train a transformer to detect what the underlying architecture is. This is something similar to sentiment detection but using 1's and 0's.

This work is still fairly early in it's experimentation, the proposed dataset has been created of 20 single file libraries.

We are using Huggingface's XLM Roberta Transformer and fine tuning it.

Each training binary is extracted as ascii 1's and 0's in 8 byte lumps.

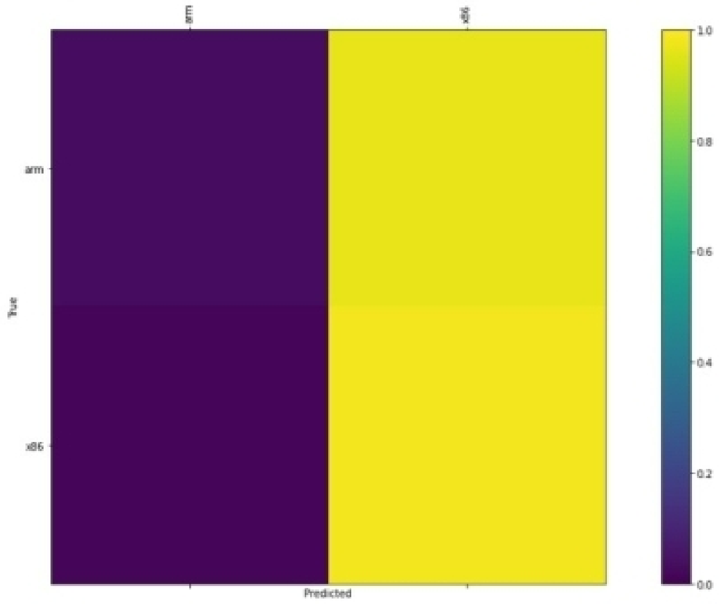
```
01111111 01000101 00010011 00100011 00000001 00000001 00000001 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000011 00000000 00011111 00000000
00000001 00000000 00000000 00000000 00000011 00000011 00000000 00000000 00000000 00000000
00000000 00000000 00000001 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000111 00010111 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000001 00000000 00000111
```

(Example X86 data)

Shifting Sand – Preliminary Results

Initial results are poor; basically random, this is using fine tuning

At this point we likely need either more data, or we may need to train from scratch.



```
Class Average Accuracy: 0.5079992006060869

(array([[  6244, 175684],
        [  3531, 189179]]),
 array([[0.03432127, 0.96567873],
        [0.01832287, 0.98167713]]))
```

Conclusions

There is a lot of interesting work going on at Sandia. I’m part of only a very small part of it.

Please see me at the Q&A for questions!