Evolutionary Intrusion Detection System

By Jared Clark

Overview

- Objective
- Background
- Implementation
- Results
- Future Work

Objective

Research Goal

Determine if effective rules for a CAN-based Intrusion Detection System (IDS) can be created via evolution

Background

Intrusion Detection System

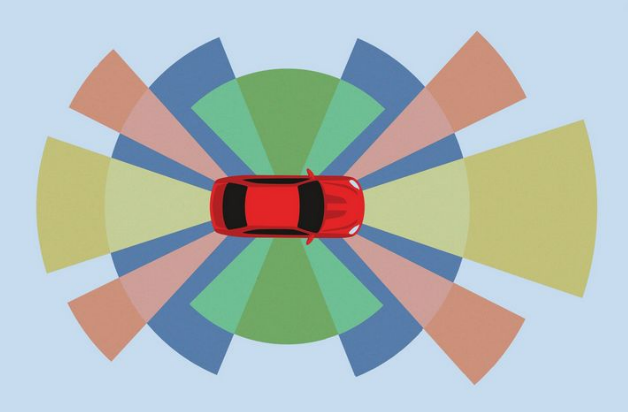
- Observes network traffic
- Determine if traffic matches rules to detect suspicious behavior
- TCP/IP solutions usually resolve problems by aborting/rejecting connections

Evolutionary Computation (EC)

1. Possible solutions are encoded as a genome (think DNA)
2. A population of individuals exist with their own DNA
3. Must be able to quantify performance of individuals (fitness)
4. Progress towards optimal solution comes from evolution and natural selection
    a. Evolution: new DNA created with mutations and reproduction
    b. Natural Selection: Individuals that perform better are more likely to survive

Genetic Algorithms

- Evolves a solution to a problem
- Example: sensor placement



Genetic Programming

- Evolves code
- Example: Evolving rules for an IDS

```
if (rule violated)
    perform action
```

Using Genetic Algorithms for Network Intrusion Detection [1]

- Evolving rules for an IDS
- Focus was TCP/IP network connection data
- Features
  - Source/Destination IP and port
  - Duration of connection
  - Protocol Used
  - Amount of data sent by sender and receiver
- No implementation, only idea

Implementation

Source code

- Python
- ~~DEAP~~ [2]
- Organization:
  - “Individual” class
  - Function to run evolutions and log (and compress) data
  - Scripts to create graphs
  - Scripts to compute train and test accuracy

Data

- Labeled CAN attack datasets [3]
- CAN data logged from real vehicle
- DoS and fuzzy attack datasets used
  - DoS dataset: 3.7 million CAN messages
  - Fuzzy dataset: 3.8 million CAN messages
- Fields: timestamp, arbitration field, message data (0-8 bytes), label
- Datasets processed to only keep unique messages (excluding timestamp)
  - DoS dataset: 24,273 unique messages
  - Fuzzy dataset: 82,773 unique messages

Genome

- Binary genome
  - One bit for every bit in a CAN message
  - Goal is to match with CAN attack data
  - Genome also evolves which bits of data to ignore

CAN Data	Bit 1		Bit 2		...
Genome	Important?	Value	Important?	Value	...

Training Loop

1. Evaluate individuals
2. Log individuals, their fitnesses, max, min and average overall fitness
3. Copy n best individuals without modification to new population
4. Select which of the remaining individuals will advance based on fitness
5. Perform random crossover between advancing individuals
6. Perform random mutations on advancing individuals
7. Create new individuals to bring population up to correct size

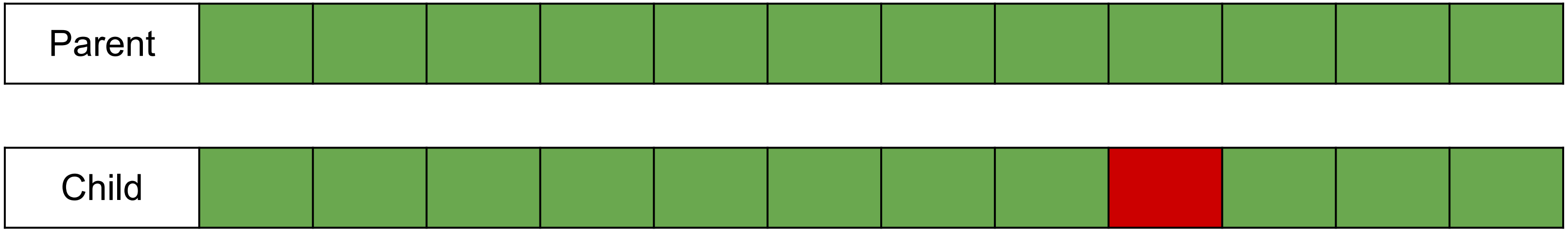
Fitness Function

```
for can_message in CAN_data
    match_level = 0
    for bit in can_message
        if (bit == important) and (bit == genome)
            match_level += 1
    if can_message is malicious
        total_fitness += match_level
    else
        total_fitness -= match_level
```

Mutation

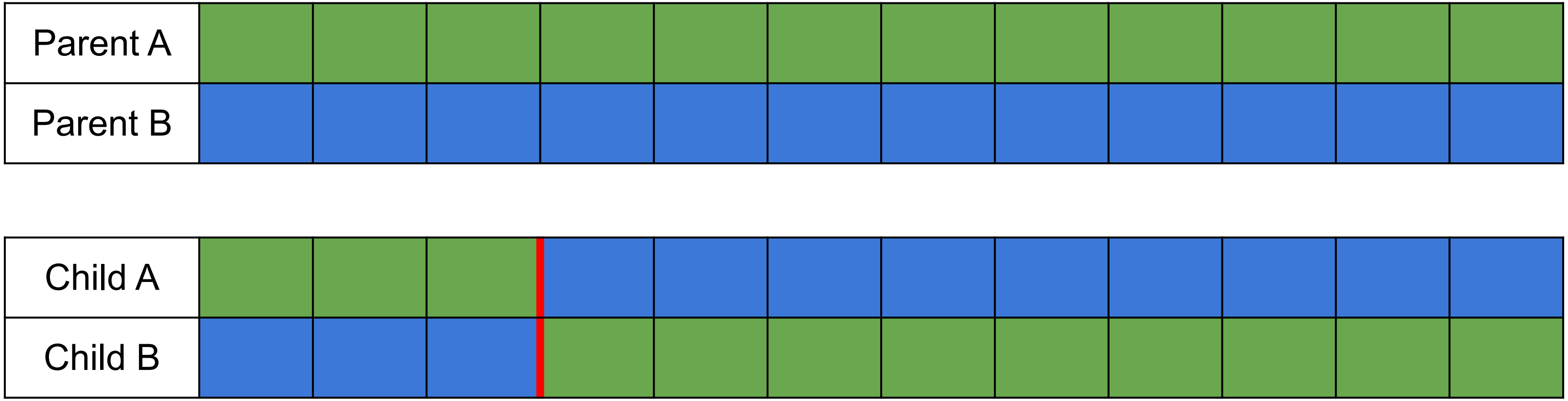
- Mutation rate was varied per evolutionary run

```
for bit in genome
    if random_value < mutation_rate
        bit = not bit
```



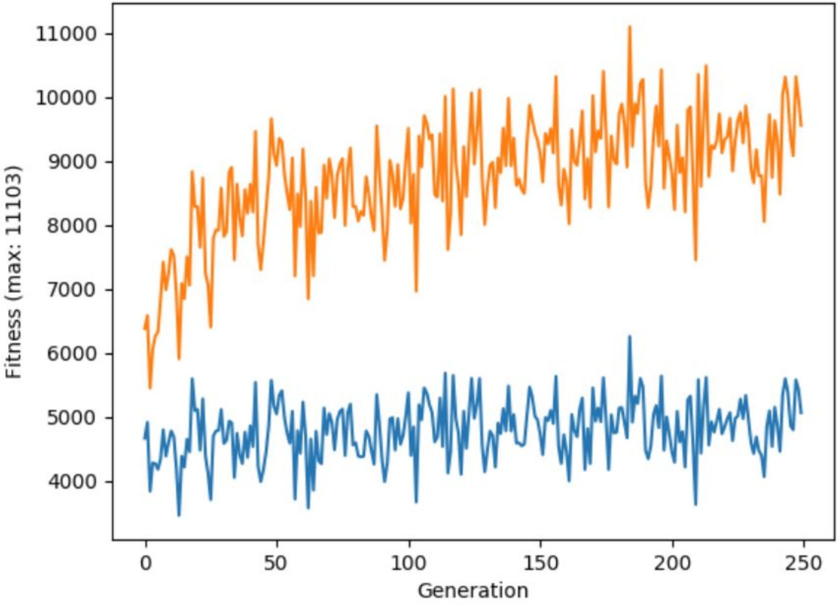
Crossover

- Crossover is when two individuals swap sections of their genome
- Random crossover was performed between neighboring individuals
- Crossover point was picked at random



Dealing with Data

- Using all data to evaluate every individual took way too long
- Picking a random subset of the data did not yield good results



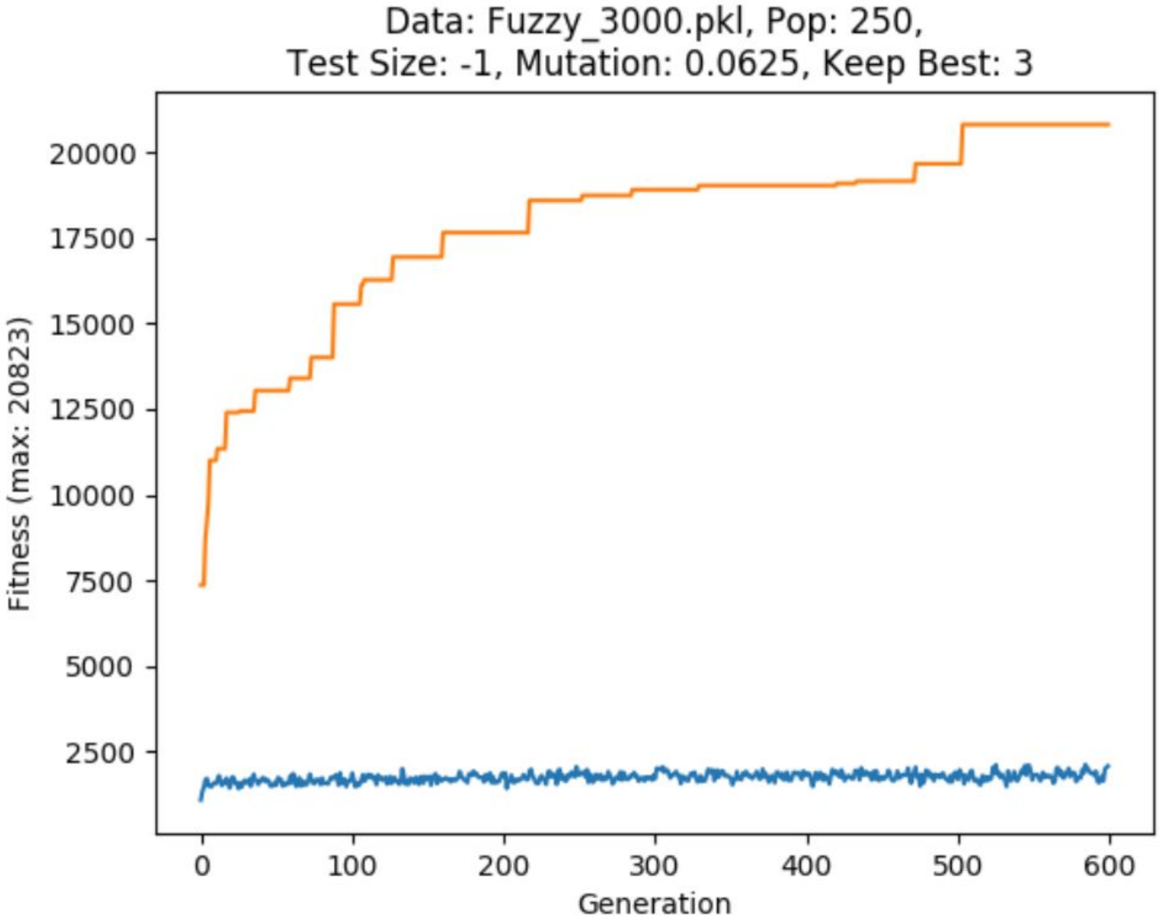
Decision: pick a subset of data to use for the entire evolutionary run

Results

Fuzzy Results 1

Accuracies

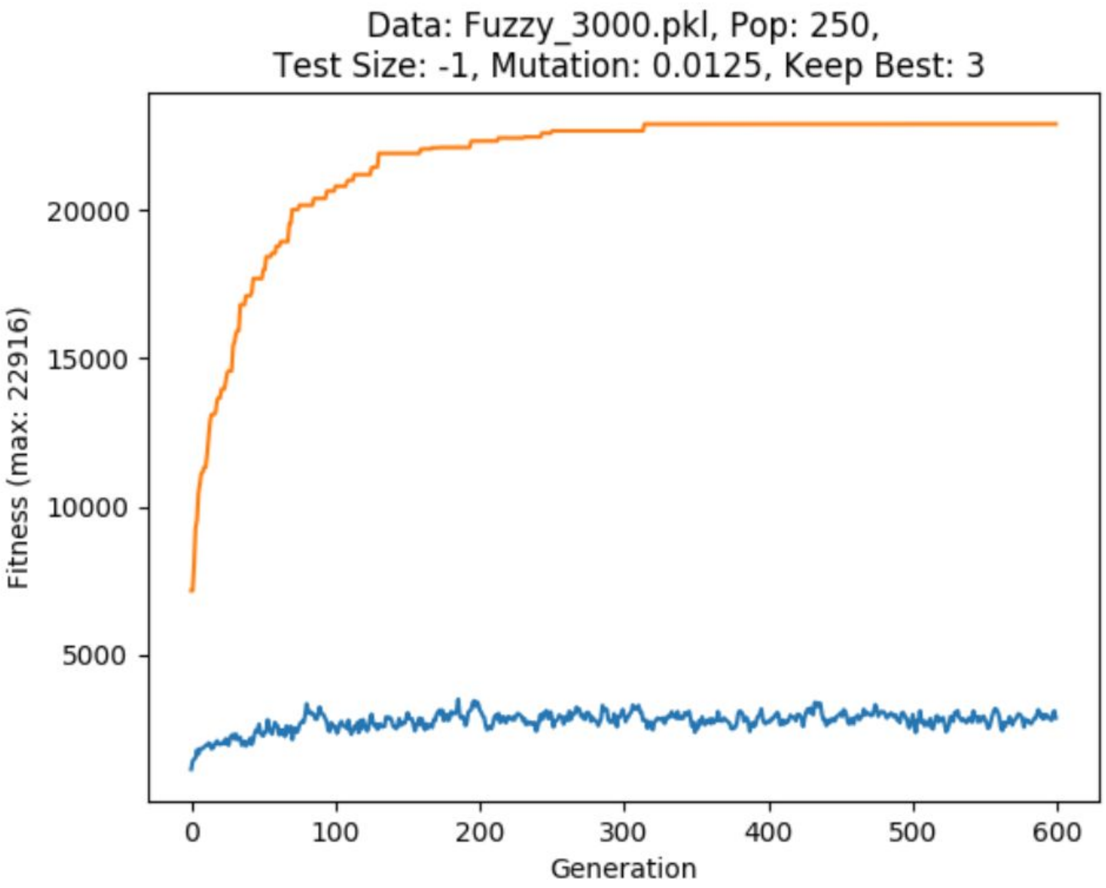
	Train	Test
1	.721	.729
2	.685	.688
3	.713	.705
4	.552	.561
5	.551	.576



Fuzzy Results 2

Accuracies

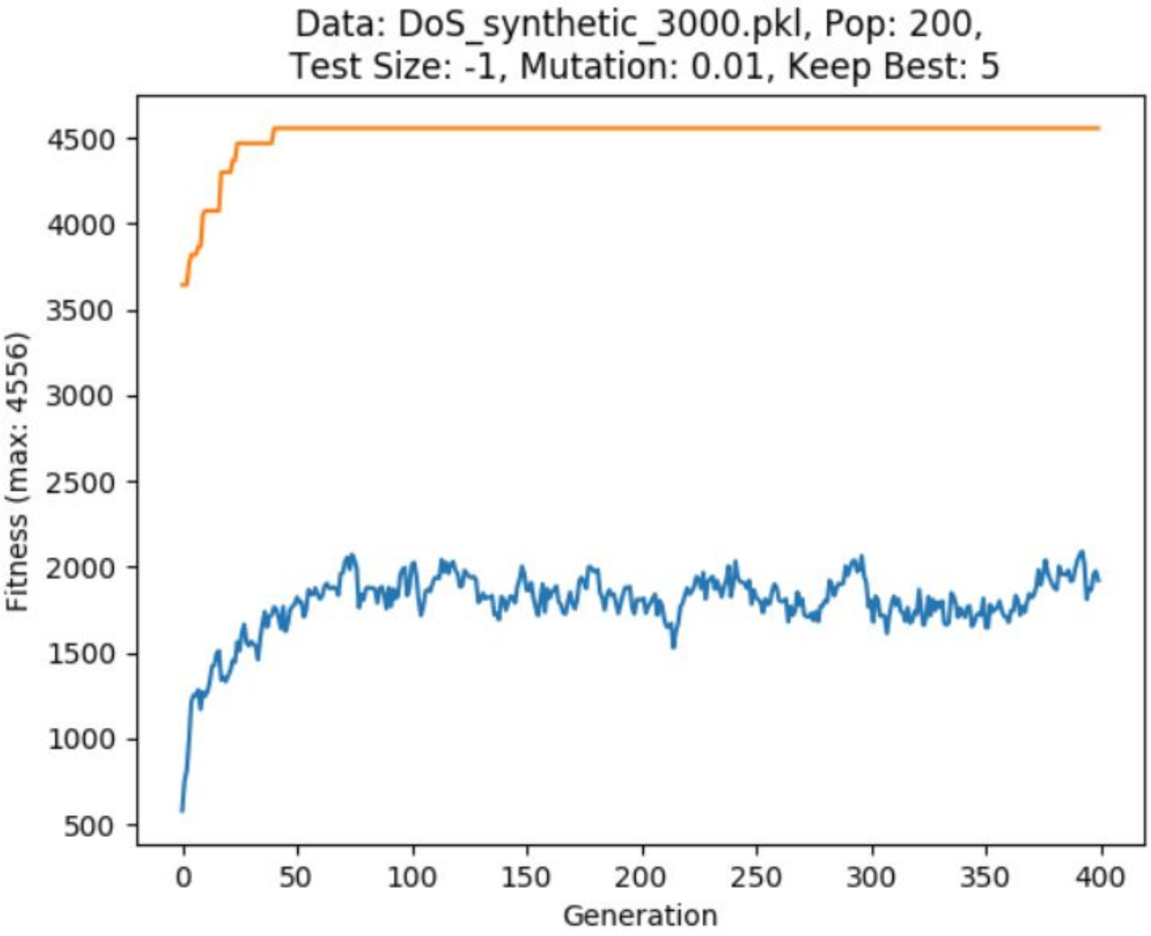
	Train	Test
1	.691	.681
2	.691	.681
3	.672	.663
4	.691	.681
5	.672	.681



DoS Results

Accuracies

	Train	Test
1	.500	.500
2	.500	.500
3	.500	.500
4	.500	.500
5	.500	.500



Future Work

Future Work Directions

- Rerun Denial of Service experiments with non-binary genome
- Adjust mutation rate during runs
- Implement parallel fitness evaluation

References

1. Li, W. (2004). Using genetic algorithm for network intrusion detection. Proceedings of the United States Department of Energy Cyber Security Group, 1, 1-8.
2. https://github.com/deap/deap
3. http://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset

Questions?