AISploit: Deep Learning for Cybersecurity Adversary Simulation in Dynamic Environments

Helena Grace, Research Fellow

The Invisible College, Knights of the Lambda Calculus

Abstract

AISploit is a framework for employing Deep Learning techniques for use in Red Team scenarios. Specifically, these models may be trained with prior attack data, in a reinforcement fashion, or ideally inside of a live environment. The ideal AISploit model will be able to navigate through the attack lifecycle steps, make decisions based on current state and prior knowledge, and adapt novel approaches to adapt to changes.

Introduction

The role of the Red Team in cybersecurity is to provide an accurate simulation of an advanced actor attempting to penetrate a live environment. Current tools to automate parts of this process are based on known exploits and common attack scenarios, but do not adapt or create new solutions. Red Team operators are constantly challenged to stay on par with the world's most advanced threat actors in all aspects of an engagement. Both automated and blue-team controlled security responses are typically predictable, and AISploit seeks to employ Deep Learning techniques to predict and adapt based on prior knowledge.

In this paper, we propose numerous Deep Learning approaches to solving the Red Teaming problem.

The contribution of this paper seeks to introduce an approach which:

1. Establishes a framework for further research into the field of Red Team Machine Learning.
2. Proposes multiple approaches of varying complexity for solving several problems relating to the topic.
3. Proves, in a reproducible manner, the validity of Machine Learning approaches to Adversary Simulation.

Preliminaries

We will be touching upon a number of techniques, most of which are commonly established in the field of Deep Learning, or otherwise.

I "The Environment"

A ideal testing environment would be one that is indistinguishable from the real world. This may be a container/virtualized environment, or more primitively some kind of randomized script to provide realistic responses/scenarios to the model. An early iteration may just have generated labeled datasets in order to prove prediction accuracy.

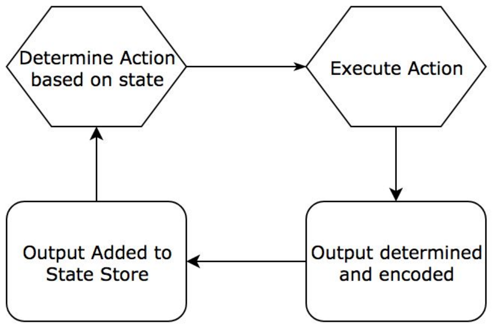
II "The Black Box" Encoder
An environment will be needed in order to train models. In order to seamlessly transition from various environments (Simulation to Real-Life), a Black Box environment must be created. Ideally, this will be an abstraction layer between the ML model and the actual environment that is indistinguishable to the model.
One such method may be to design a middleware layer that encodes environmental data and responses/rewards (real or simulated) into a dataset for the model to respond to, and similarly encodes model predictions/responses/actions back into the simulation or real environment.

III Structuring Data and State
With the afformented Black Box, an agreed upon data format should be adhered to. In simple terms, the model will take inputs (The Current State, Former State, etc) and output an action. This will be detailed more in a further section.

IV Supervised Learning
As a proof of concept, Supervised Learning techniques may be employed in order to bootstrap a model with data generated either by mining information about real-life engagements, or by generating data representing known successful scenarios.

V Reinforcement Learning
Ideally the model will need to act autonomously and learn based on many engagements. For this to be possible, a reinforcement or reward system should be in place, as well as a training environment. Utilizing reinforcement learning as opposed to Supervised Learning will allow for the model to adapt to current engagements at a faster rate without the need for retraining.

Data, State and Structure


Determine Action based on state
Execute Action
Output determined and encoded
Output Added to State Store

Flow of actions to state, etc

Abstractly, data sent to the model will adhere to the following structure:

Current Action
The Current Action is the last choice the model made. This may also include "attack stage" information, for instance: "Stage: Exploit, Action: Strutspwn.py, Target: xyz"

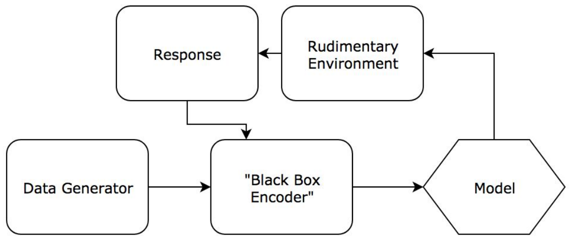
Current State
Current State data will contain the response from the computer to the current action. For example, an unsuccessful exploit attempt may return an error, or possible a timeout. The error, as well as any relevant error messages, should be returned.

Environmental Knowledge
Mainly this will consist of any reconnaissance data (for instance the results of an NMAP scan), as well as any incidental data gathered from previous actions (For instance, "Port 22 Username Ubuntu returns a response, requires key")

Methodology

Iteration I

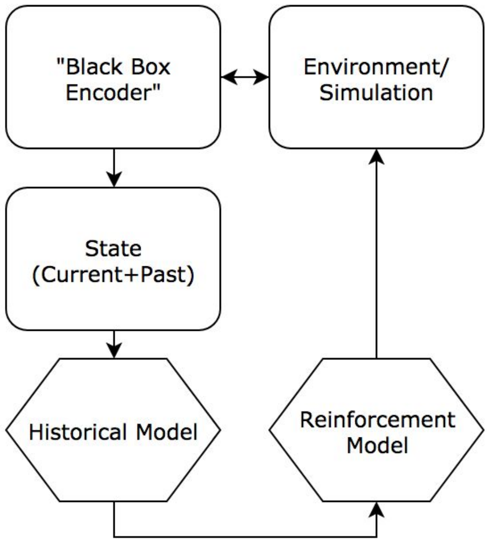
The initial iteration of AISploit will seek to establish the Black Box concept, to easily translate environmental data (or generated data) into a format in which the model will understand. The initial proof of concept will utilize labeled data generated from known scenarios, and will use categorical responses in order to predict a successful next course of action.



A model is trained off of generated labeled data. Once trained, the model is given actual scenario data, encoded into the same format.

Iteration II

Once the environment and encoder is fully matured, the method for training the model will be modified to use a reinforcement learning approach, alongside the initial model from Iteration I. This ensemble approach will allow for decisions to be made cooperatively by two models, one interpreting and learning from the current environment, and another that holds "memories" of any data it is trained upon.



An ensemble approach is used to augment decisions made by a reinforcement learning model. This allows for pre-loading a model with knowledge of many scenarios while also allowing it to adapt to current conditions.

Architecture

Environment

The initial environment will need to accept inputs from the "Black Box" encoder, and provide data about the state to the encoder. Additionally, the environment must have knowledge of the target's state in order to send a response when the target has been compromised. This can initially be approached by a simple script that simulates the state of a system (like an adventure game), but eventually to progress, an actual system will need to be used. Ideally this system will be able to be rapidly changed to many configurations, in order to provide many scenarios to the reinforcement learning model. Possibly something like Docker can be used.

Black Box Encoder
The encoder will likely just be a python script that translates console output data into a Pandas Dataframe. The encoder will need to be able to receive both “real” output as well as simulated output, and encode model responses into commands. (“SCAN” may translate to a specific nmap command, “EXEC STRUTSPWN” may map to a specific command w/ host information, etc)

ML Model
At the moment, Python and TensorFlow seem to be the most resilient option for both Supervised and Reinforcement Learning. Since the amount of data being processed is minimal, a more distributed solution might not be necessary in the future.