Assignment 1

Drew Springall, Daniel Tauritz, Deacon Seals

February 8, 2022

1 Introduction

Though many attack vectors exist, network-based attacks continue to be a major source of initial infections due to their ubiquity and stealth. Additionally, almost all types of malware "phone home" over the network for the purposes of data exfiltration, command and control, fetching additional functionality, and the like. Fortunately, canonical policy-based defenses such as firewalls and IDS/IPS systems exist which allow defenders to specify rules to enforce on network traffic with varying degrees of granularity. The fundamental issue, however, is that policy-based decisions:

- rely on rules that can only be created *after* the details of a new attack are known by defenders, and/or
- are based on predictive heuristics of what an attack *might* do, which may or may not hold-true given an arbitrary attack strategy or piece of malware. In either case, minor variations in malware and attack strategies can be devised to circumvent known defenses due to the dynamic behavior of attackers versus the relatively static behavior of defenders.

Because of these and other complications, machine-learning (ML) has often been touted as "the Great Savior" of network-based defenses. With its ability to classify based on otherwise undetectable features and to dynamically learn new features, this sentiment is not entirely dishonest. There are, however, many unsolved problems in the context of ML which show how misleading these types of general-solutions can be when faced with an intelligent and reactive adversary.

The most obvious of these is the fact that the opaque nature of standard ML classification makes it nearly impossible for a human to understand the rationale behind *why* a classification was made by an ML model, much less predict how the model would classify a never before seen input. In this project, you will explore both sides of ML-based network protections in an attempt to build (and later defeat) such an implementation. Each group will submit a solution which:

- partitions "good" network traffic from "bad" network traffic

- acts as a real-time decision-engine for network traffic based on an incomplete view of network traffic
- attempts to circumvent other groups' implementations of both the previous requirements

2 Context

For useful context, you can envision yourself operating a Small-/Mid-Size Business (SMB) network consisting of Linux-based clients and servers. On this network, there are "benign clients" performing standard actions such as fetching internal web-pages, uploading/downloading files, and various other normal user-activity. Additionally, there are zero or more "malicious clients" which are launching well-known attacks against other devices on the network.

3 Phase 1: Building a Full-Knowledge Classifier

In this phase, you will:

- build a processing program to extract your chosen ML features and output them in a self-descriptive JSON representation,
- build an ML model which, with the help of some boiler-plate code, will read-in your processed JSON data and make a determination on which clients are malicious, and
- learn to ingest and operate on the JSON representation of network traffic as opposed to the more-common PCAP file format for raw traffic (link to data); it's created from a standard PCAP file via the command:

```
tshark -r XXXX.pcap -T json | gzip > XXXX.json.gz
```

Your guiding-tasks for this phase of the project are:

- From a provided set of JSON network traffic:
  - Identify benign and malicious clients' traffic
  - Identify useful features that can be used to identify the malicious clients and benign clients
- Build a "pre-processor" program which reads-in an JSON network file and outputs a set of structured JSON records with your chosen features
- Build a "model builder" program which reads-in your self-produced JSON records plus a list of known-malicious clients and uses that information to train and output an ML model

- Build a "classifier" program which reads-in your self-produced JSON records plus your pre-trained ML model and output a list of asserted-malicious clients

Processing Program Requirements:

- Through the '–input-data' flag, you will take a filesystem path to a JSON network file which will be read
- Through the '–output-records' flag, you will take a filesystem path to a file to which you will write your newline separated JSON records

Training Program Requirements:

- There are no requirements as it will not be run for grading.

Classifying Program Requirements:

- Through the '–input-records' flag, you will take a filesystem path to a newline separated JSON file (generated by your Processing Program) to be read
- Through the '–output-benign' flag, you will take a filesystem path to a file which you will write newline separated IP for hosts classified as benign
- Through the '–output-malicious' flag, you will take a filesystem path to a file which you will write newline separated IP for hosts classified as malicious

3.1 GRADING

Your pre-processing program will be graded on:

- Ability to handle seen and unseen network traffic
- Correctness of your described processing actions
- Understandability of your output JSON records
- Determinism (the same input file should always output the same JSON records)
- Readability of your processing architecture
- Readability of your code

Your classifier will be graded on:

- Ability to handle seen and unseen network traffic
- Correctness of identifying benign clients
- Correctness of identifying malicious clients

- Determinism (the same JSON records should always output the same classifications)
- Readability of your code