Beacon classifier

Anonymous Authors[1]

Abstract

In large networks, malware programs hide in plain sight. Whereas signatures such as periodicity and bunching exist, they are buried in what is often billions of rows of data. We present two classifiers: one is inspired by a Bayesian belief network and determines the bias of an events' payload, the other compares the time elapsed between connections to what is expected from human or random behavior using the Kullback-Leibler (KL) divergence. We manually labelled the connections of a dataset as 'beacon' or 'no beacon' and used them to compute the receiver operating characteristic (ROC) curve to tune and test the classifiers.

1. Research problem and motivation

Malware is any software used to disrupt computer operations, illegally collect information, inappropriately gain access to private networks, or cause damage to computational resources. A common behavior of malware is known as "beaconing" which implies that infected hosts communicate to Command and Control (C2) servers at regular intervals (time lapse) with relatively small payload size variations. Malware with beaconing behavior range in severity from annoying, but relatively harmless, to potentially calamitous in the hands of criminal groups or state-sponsored actors. Additionally, there might be random gaps in the signal by accident or by design, and parameters such as connection frequency are completely at the discretion of the attacker.

2. Technical contribution

We developed two classifiers to arrange attributes of connections (number of bytes transferred, time elapsed between connections, etc.) into 'beacon' or 'not beacon' with an error rate set by the user. The first classifier integrates evidence in a Bayesian way to determine if the values in a collection are biased towards having the same value or sets of values. The second one uses a measure of divergence between two probability distributions to determine if the values in a collection could have been generated by a known or random process, or not. We define a set of repeating events as consists of connections that share the same client IP address, user agent (app), and internet host in a given time range (so they are likely originating from the same computer and repeatedly connecting or attempting to connect to the same host).

2.1. Bayesian bias classifier

The bias classifier works best with payloads and is inspired by a Bayesian belief network instance that has an analytic solution: that in which each observation is identical and independently distributed. An example is a series of coin tosses in which the coin might be biased. $p(\theta)$ is the Bayesian prior probability distribution that the coin is biased towards 'heads' ($x = 1$) or 'tails' ($x = 0$) and it can be shown that:

$$p(\theta|X) \propto p(\theta)\theta^{N_{x=1}}(1-\theta)^{N_{x=0}} \tag{1}$$

where $p(\theta|X)$ is the Bayesian posterior probability given a set of observations X. To illustrate this idea, we show in Fig. 1 how the probability distribution is updated for a coin that lands 'heads' 80 percent of the time. The blue probability distribution is the initial estimation of the bias and it is flat since there is no information. The red probability distribution is the estimation after 10 tosses, for example $X = [0, 1, 1, 1, 1, 0, 1, 1, 1, 1]$. Its maximum is already at 80 percent, but it is wide because there is significant uncertainty (it's only 10 coin tosses). The green probability distribution is obtained after 100 tosses; the maximum is still at 80 percent and the distribution is narrower. The results are more accurate with more tosses (or connections), which is appropriate for beaconing events that, by definition, repeat many times. The Bayesian credible interval is analogous to the confidence interval in frequentist statistics and can be defined in several ways. The equal-tailed interval is such that the probability of being below the interval is as likely as being above it; this interval will always include the median. A smaller range can be chosen where the probability density is highest, and this interval will include the mode.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

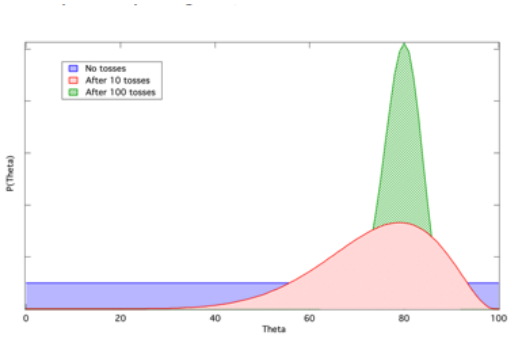

Figure 1. Bayesian probability that a coin is biased towards landing heads 80% of the time after 0, 10, and 100 tosses (blue, red, green lines respectively).

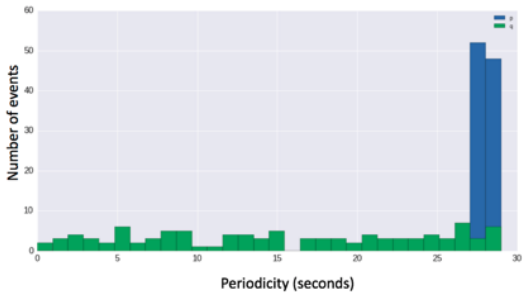

Figure 2. The distribution p of a potential beacon (blue) and a random distribution q that has the same number of events and same longest time lapse (green). The two distributions are very different so the KL divergence is large.

2.2. Kullback-Leibler divergence classifier

The KL classifier works best with time intervals between connections and uses the Kullback-Leibler (KL) divergence to compare a set of actual time intervals to what would be expected from a random source, a known source, a typical human, etc. connecting the same number of times. The KL divergence from q to p is defined as:

$$D_{KL}(p||q) = \sum_i p(i) \log \frac{p(i)}{q(i)}, \qquad (2)$$

the expectation value of the logarithmic difference between probability distributions p and q. The number of events $I = \sum_i$ is used to create a synthetic distribution q in which the time range is from 0 to the longest lapse in the distribution of interest p and otherwise has the statistical properties of a target distribution. For example, the behavior of a known IP address could be used to create a synthetic distribution with the same properties. For testing the methodology, we used random distributions, and this is illustrated in Fig. 2. If a user connects to a host 20 times in 2 minutes, it is likely that the time elapsed between connections will be random, whereas a beacon will be trying to connect, e.g., every 6 seconds or so. The KL divergence will be close to 0 if the distribution of interest is indistinguishable from random and larger than 0 otherwise. In order to get an accurate value for the KL divergence, the synthetic distribution q needs to be generated multiple times, and for each case, the divergence of every q is calculated.

2.3. Experiment

We manually labeled datasets to tune each classifier and to test their inherent quality. For the bias classifier, we used 500 sets of repeating events and for the KL classifier we only needed 200 sets. A human decided whether they represented beaconing behavior or not. In each case, 80% of the events were randomly selected and used to train the classifier, and the remaining 20% were used to test the performance. A receiver operating characteristic (ROC) curve is a common way to evaluate and tune classifiers. In Fig. 3, the gray diagonal line shows the performance of a completely random classifier. Points above the diagonal represent better than random results. The area under the curve is a measure of how good the classifier is; the parameters that result in the smallest distance to $x = 0, y = 1$, are the ones that optimize the performance of the classifier (maximize the recall and minimize the fallout). Also shown in Fig. 3 is the performance improvement (detailed by dotted lines) of the classifier when there is a longer collection time (20 minutes vs 40 minutes). When used with the rest of the algorithm, repeating event data will be collected until a determination can be made at the error tolerance specified by the user.

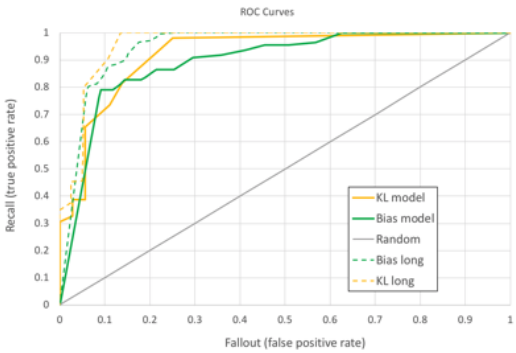

Figure 3. ROC curve for two models used in the current invention with a collection of time of 20 minutes and 40 minutes (long).