

Attacking Machine Learning models as part of a cyber kill chain

Tam N. Nguyen
North Carolina State University
<https://www.linkedin.com/in/tamcs/>

Abstract—Considering the gaining popularity of "defense in depth" strategy, plus increasing amounts of money invested in information security layers, and considering adversaries' perspective while carrying out a long-term advanced-persistent attack campaign; avoiding (short term) detections may not be as beneficial as having a deeper knowledge about targeted "defense in depth" system. Probing and stealing information security machine learning models for organized cyber attack campaigns should not focus only on obvious results (a yes/no classification of attacks) but also on other factors.

I. INTRODUCTION

There is a significant gap between the amounts of connected devices and the number of cyber security professionals. Per U.S. Bureau of Labor Statistics [1], a projected growth in cyber security jobs from 2014 to 2024 is 18% while Cisco [2] predicted a 50% increase in connected smart devices, from 4 billions in 2016 to 8 billions units in 2021. Global data traffics will also increase 5 times by 2021. At the presence, large oil companies are using autonomous drones to check their oil pipelines. Hospitals network-enable their equipments to help with detecting abnormalities in daily clinical operations. Car companies can push updates to their cars remotely via the Internet. The gap is huge, and Machine Learning (ML) models came to the rescue. This paper will walk quickly from the concept of cyber intrusion kill chain, to a 4-step process of a daily S.O.C routine, highlighting why and how ML models can be of great helps. A list of commonly used ML models in Cyber Security will then be briefly evaluated, focusing on each model's capability in misuse detection, anomaly detection rather than theoretical maths or algorithm efficiency (Big O). The paper then goes deeper by open up a discussion on how ML models can be attacked. This is where the maths get heavier but not too heavy. The reason is that sometimes, in order to steal or to poison a ML model, attackers do not need to use any fancy math at all. Once readers have a good, general image of how S.O.C operates, the pros and cons of commonly used ML models, and attack methods; the paper will discuss several possibilities that can be employed per phases of a cyber kill chain. Finally, a fun example will be provided to put everything into context, illustrating how a ML model can be cloned illegally as well as how attackers can predict the impacts of future attacks within their lab environment.

tnn

April 28, 2017

II. SECURITY OPERATION CENTER (S.O.C) PROCESSES

Security Operation Center (S.O.C) is part of a "Defense in depth" strategy. Metaphorically, "defense in depth" is more like an artichoke than an onion, even though both of them share the same layered structure. When 10 of its layers got peeled away, an artichoke will still maintain almost the same shape (posture) while an onion will quickly shrink and reveal its core. Contribution factors such as organization structures, different levels of budget allocations and distributions, geographic separations, etc. make it also impossible to have big protection layers, covering the whole structure and laying on top of one another. In short, a "defense in depth" system consists of interlaced, overlapping-but-independent protection layers backing each other up. In response, adversaries employs "advanced persistent" attack strategies in which persistent organized efforts can be categorized into phases also known as "intrusion kill chain" which was originally introduced by Lockheed Martin Corporation [3].

A. The Cyber Intrusion Kill Chain

There are 7 phases of a Cyber Intrusion Kill Chain (Kill Chain).

- 1) Reconnaissance (recon) : adversaries research about the targeted system's structure, capabilities, vulnerabilities, etc
- 2) Weaponization : adversaries code up deliverable payload
- 3) Delivery : adversaries use different ways to deliver weaponized payload to destination
- 4) Exploitation : payload was executed and vulnerabilities of target system(s) were exploited
- 5) Installation : malware(s) were installed and the "persistent" factor can now be established
- 6) Command and Control (CnC) : adversaries established a hidden control channel with compromised entities within the victim system to further organize and expand the attack campaign
- 7) Actions : adversaries now perform actions on their true objectives

S.O.C daily operations align very closely with cyber kill chains for several good reasons. At first, it reminds S.O.C analysts that proper attentions and responses are expected at all times rather than just at or after the point of compromise. It also encourages the analysts to always seek a broader picture,

locating an attack as part of a past-present-future orchestrated moves. The real work has just begun once an incident had been identified. Often, S.O.C analysts will need to answer two important questions: how did the attack get to this point? and what is the attackers' true intention? For example, if an installation of a malware was identified (phase 5 of kill chain), it is crucial to traverse to earlier phases of the kill chain and find out: what vulnerability(ies) in what subsystem was exploited? In what way the initial attack payload was delivered? How the payload was built? and how did the attackers know of such vulnerability(ies)? In the mean time, analysts also need to look forward and try to predict how attackers will use the piece of malware to further exploit the system. Good news is there are usually patterns within each phase of the kill chain, and between kill chains of different attack campaigns. An advanced persistent attack campaign is usually a combination of those patterns [3]. For example, if there are 5 patterns within each phase, a kill chain being carried by Alice can be 1-1, 2-4, 3-1, 4-4, 5-3, 6-5 and 7-1, and that of Bob can be 1-1, 2-2, 3-1, 4-5, 5-1, 6-1 and 7-2. Therefore, in-depth knowledge of past kill chains will help with detecting, preventing and predicting future kill chains. Since all of those tasks cannot be carried out by humans alone considering constraints on time, budgets, physical limitations, etc; Machine Learning (ML) models are used. While there are three different types of S.O.Cs (threat-centric, compliance-based, and operational-based), most of their ML related workflows can be categorized into 4 common steps : Data pre-processing, clustering/threshold adjustment, event attribution, and monitoring.

B. Data pre-processing

Raw captured data is huge and came from various sources such as full packet capture (PCAP), NetFlow (generated by Cisco network devices), protocol metadata, application logs, machine logs, telemetry streams. Just within a NetFlow record itself, there are 104 possible data fields with 22 additional fields reserved by Cisco[4]. During this phase, unsupervised ML models are used to help with grouping data into clusters using measures of association. For example, the Tukey Quick Test can provide general clarification between classes based on the differences in statistical tests of feature sets. Spearman's rank correlation [5] is another possibility. In such case, system can identify and group all packages relating to a TCP conversation into one cluster including handshake traffics, DNS traffics, https traffics, duplicated packages caused by different switches along the path, NAT'ed traffics, etc. Common issues within this data pre-processing phase include but are not limited to streaming fact issue, noisy data, and the trustworthiness of data. Streaming fact issue refers to wrong data clustering decision made on incomplete facts (key information has not yet arrived). Some systems try to overcome this problem by using ML models to make inferences as data are coming in. Noisy data and untrustworthy data can be caused by hackers or by aging devices, network congestion, or simply, configuration mistakes made by field admins.

C. Clustering/threshold adjustment

It is estimated that the amount of false positive alarms raised by signal based IDS is 5 to 20 times the amount of true positive ones. Having too many false positive alarms will cost time, money and corrode the credibility of the intrusion detection system. Therefore, an Adaptive Cluster Tuning process (ACT) is used to adjust the initial categorization/clustering done by machines. First, ACT admins will look at original thresholds suggested by ML models and make necessary adjustments. ACT admins will then load validation data, and ML system will come up with detections results. If ACT admins deem the results acceptable, they will put the models into deployment. Otherwise, ACT admins will re-evaluate attack attribution per cluster basis, and adjust the threshold again.

D. Event attribution

At this stage, analyst can view and select which features from a provided list of features to be associated with a cluster per identified patterns.

E. Monitoring

Finally, models are deployed. Traffics are monitored and packages got sent to first stage for pre-processing - the cycle repeats. It is important to do so in order to identify concept-drift due to evolution of attacking methods or software usage patterns, and concept-evolution such as the emergence of a completely new method of attack.

As we can see, at operational levels, S.O.C has to deal with many issues already. In addition, S.O.C has to balance the goal of early attack detection and the cost of false positive alarms; to catch up with the speed of evolving attacks while tuning the system. The need for ACT is obvious, and while the human factor can help the ML models in many ways, it is also important to note that humans are another attack surface. Human makes mistakes.

III. COMMON MACHINE LEARNING MODELS USED IN S.O.C

In their daily operations, S.O.C trains and re-trains many models simultaneously and there are two main reasons. On one hand, it is about the spirit of "defense-in-depth" where there must be independent modules (models) backing up each other rather than just one big identity. On the other hand, there are inherent limitations in each ML model type due to its math foundations and its designs. Based on an earlier survey done by Buczak and Guven [6], some most common ML models together with their pros and cons will be briefly discussed. While some statistics will be presented for the convenience of assessing the general effectiveness of each model, those numbers are by no mean be completely correct nor can be directly compared between models. Different experiments for different ML models may be performed by different groups with differences in datasets used, tuning methods, etc.. The survey paper by Buczak and Guven [6] gave references to further resources regarding the details of each experiment. The

best way to really evaluate and compare ML models, however, is to build and test them in one's own lab environment and have same analysts with same tuning experience to work with the models.

A. Artificial Neural Networks

By design, ANNs are fit for non-linear problems but tend to suffer from local minima leading to long learning time, and as the number of features increases, the longer it takes to learn. With Misuse detection, ANNs demonstrated more than 90% accuracy. With anomaly and hybrid detection, multi layer ANNs were used to analyze data based on time windows. For that reason, the models can reach deeper into lower network layer data and was able to detect some low/slow type attacks. However, the performance of ANNs was not consistent. While they can identify 100% of the normal behavior, the amount of false alarms may sometimes reach 76% depending on what kind of attacks were being executed.

B. Bayesian network

Bayesian network is a probabilistic directed acyclic graph type with nodes as variables and the edges as their relationships. Based on the relationships, a node can "walk" to another. Each node has a probabilistic value and at the end of the walk, a final probabilistic score is formed. Relationship links that have high true positive score will be verified and formed into rules. Therefore, a Bayesian network is proactive even in misuse mode. In a test of using model to label IRC-botnet generated data, the precision rate is 93% with a false positive rate of 1.39% (detecting fewer cases than some other models but generating less false alerts). In other tests, the reported precision rates are 89%, 99%, 21%, 7%, and 66% for DoS, probe/scan, remote-to-local, user-to-root, and "other" classes of attacks respectively. In the case of anomaly detection, ACT process was used to tune the system. Accuracy of 100% with 0.1% false alarm rate were reported in lab experiments analyzing TCP/IP packets.

C. Clustering

Some popular clustering models are k-means, k-nearest neighbor, density-based spatial clustering of applications with noise (DBSCAN), etc. Because the models were designed in order to find patterns in unlabeled multi-dimensional data, explicit descriptions of classes are not required. A weakness of this model is known as the "curse of dimensionality". Too many features may confuse the model and any imbalance in the feature set will negatively affect its decisions. Therefore, analysts with background knowledge of the systems will tune the models by selecting only sets of features needed, deciding a proper neighborhood (k) value in order to maximize the impacts. In misuse mode, two processes will be executed in parallel. On one hand, the model will classify data into "normal" and "attack" classes. On the other hand, analysts work with the model to precisely identify normal data. The combined outputs are signatures that will be used in a rule-based monitoring module. In some experiments, this method

can detect up to 80% of unknown attacks. In anomaly mode, studies suggest that clustering models can be really accurate (98%) in analyzing captured PCAP packages but performance goes down when dealing with streaming data (false alarm rate may go up to 28%)

D. Decision trees

Decision tree is a flow-chart like structure built on concepts of information gain/entropy where each node choose the best fit attribute to split current set of examples into subsets. Normally, decision trees provide the benefits of high accuracy with simple implementation. However, it is not usually the case with larger trees. Also with large, complex trees, the model tends to favor attributes with more levels. To overcome issues with large trees, analysts will have to do some pruning to get smaller trees. In misuse mode, decision trees can significantly reduce the time needed to compare inputs with signatures (up to 105% faster than traditional methods). In anomaly detection mode, experimental results of using decision tree model to detect bad domain names from passive DNS queries show that the model is accurate with acceptable false alarm rates (reduced by constant re-training).

E. Evolutionary computation

While Genetic Algorithm (GA) and Genetic Programming (GP) are most used Evolutionary Computation (EC) methods; Particle swarm optimization, Ant Colony Optimization, Evolution Strategies are also parts of the group. The main concept is based on the idea of "the strongest will prevail" and basic operators are selection, crossover, and mutation. In misuse mode, an initial set of features and population will be picked and in the end, the best rules will surface to be used in a rule-based module. Experiments with various attack types show that the average false alarm rate is very low. However, the sensitivity in detecting new attacks varies greatly (from 66% to 100%) depending on attack types.

F. Naive Bayes

Naive Bayes model calculates the final conditional probability of "attack" (or "normal") with a strong (naive) assumption that the used features are independent from each other. That assumption is the biggest limitation of this model. However, if the features are indeed independent from each other, naive bayes can be very powerful thanks to its simple algorithm that allows the model to be highly scalable and be used as an online classifier. In misuse mode, experimental results suggest a decent accuracy (above 90%) but with a quite unacceptable false alarm rate (around 3%). In anomaly detection mode, experimental results show a tremendous difference in accuracy. For example, accuracy of identifying data as "normal" is 97% for DoS type attacks and only 9% for remote-to-local type attacks.

G. Supported vector machine

Supported Vector Machine (SVM) is a binary classification model by design. With a kernel such as linear, polynomial,

Gaussian Radial Basis Function, or hyperbolic tangent; the model will try to draw a hyperplane that divides the feature space into two classes. Sometimes, when overlapping is unavoidable, slack variables will be added and each overlapping data point will be assigned a cost value. In misuse detection experiments, a large set of features is reduced by using feature selection policies or feature selection algorithms. The model is quite accurate but also shows limitations at identifying certain types of attacks such as user-to-root attack. In anomaly detection mode, usually SVMs will use more sophisticated kernel to help with the drawing of the hyperplane. Experimental results show great variations in accuracy (from 65% to 99.9%) and sometimes, false negative rate can get really high (over 30%)

IV. ATTACKS ON MACHINE LEARNING MODELS

As we can see, no single ML model is perfect and the ML work process at S.O.C has some bumps. Knowing the types of the models being used will help with planning attacks, but it is certainly not enough. A defense-in-depth system is like a maze where each one of the ML models is a trap. Traps are open and close at different time windows. Having clones of the traps allows simulations to be done, and it helps intruders to plan their moves. Thus there is a strong motivation to clone a security ML model. In 2016, Tramer et. al. [7] proposed several methods to perform model extraction on several ML types. While those methods will be discussed in the following section, some clarifications are first needed.

- When training a ML model, provided inputs go through the model on one end, and results are given at the other end. When performing model extraction, inputs are given to a trained model, end results (outputs) got harvested, and clone model learns from the inputs-outputs data pairs. While it appears that the two processes are pretty much the same, model cloning does not have to deal with training data noises; which are duplicated, broken or faulty data entries that contribute nothing or can mislead the learning process. In addition, model cloning does not have to deal with optimization issues such as local minima/maxima traps.
- The attacks methods to be mentioned are not weaponized. While the methods work and can be effective in extracting some online models hosted by Amazon AWS or BigML, those methods are far from being ready to be deployed against ML models used in a defense-in-depth system. In some cases, it required Tramer and his team [7] to perform thousands of probes in order to extract a model, and that number is too high to be effective in a secured environment.
- Math background for this paper
 An ML model $f : X \rightarrow Y$
 An input vector $X = X_1 \times X_2 \times X_3 \times \dots \times X_d$
 Pre-processed space of inputs used to perform feature extraction: M
 Feature extraction model : $ex : M \rightarrow X$
 Class labels : Z_C or $[0, 1]^C$

For a given $x \in X$ and $i \in Z_C$, $f_i(x)$ is the i^{th} component of $f(x) \in Y$

Probability of class label i : $f_i(x)$

Predicted class : $argmax_i f_i(x)$

Distance measure : d_Y

For $Y = Z_C$: $d(y, y') = 0$ if $y = y'$; 1 if otherwise

For $Y = [0, 1]^C$: $d(y, y') = \frac{1}{2} \sum |y[i] - y'[i]|$

Training model : T

Inputs of T : $\{(x_i, y_i)\}_i$, where $(x_i, y_i) \in X \times Y$

Output of T is ML model f_T that is close to f

Adversary: A

Test set: D

Errors between f_T and f : R_{test} and R_{unif}

Total-variation distance : Accuracy of f_T when compared to f (also known as the closeness of class output)

$R_{test}^{TV}(f, f_T)$ and $R_{unif}^{TV}(f, f_T)$

Test error:

$R_{test}(f, f_T) = \sum_{(x,y) \in D} d(f(x), f_T(x)) / |D|$

Uniform error:

$R_{unif}(f, f_T) = \sum_{x \in U} d(f(x), f_T(x)) / |U|$

A. Equation-solving attack

This form of attack is fit for logistic regression types such as binary logistic regression (BLR), multi-class logistic regression (MLR), and multi-layer perceptron (MLP). Because the models can be represented as equations with variables, attackers just need to feed the known variable values in, use mathematics to solve the equations and get the rest of the unknown values. For example, with BLR, we have :

$w \in R^d, \beta \in R$ with $f_i(x) = \sigma(w \times x + \beta)$

where

$\sigma(t) = 1 / (1 + e^{-t})$

Attacker will feed x_i to the trained model and the model will give $y_i = f(x_i) = \sigma(w \times x_i + \beta)$. If we have enough x_i, y_i , we should be able to solve the equations to get w and β . The math will be more complicated when dealing with MLRs and MLPs. With softmax model in MLR for instance, we have:

$c > 2, w \in R^{cd}, \beta \in R^c$
 $f_i(x_i) = e^{w_i \times x + \beta_i} / (\sum_{j=0}^{c-1} e^{w_j \times x + \beta_j})$

The function can be solved by minimizing its cost/loss functions. The methods proved to be effective in Tramer's experiments [7]. With BLR, they were able to achieve $R_{test} = R_{unif} = 0$ with an average probe of 41. The number of probe is much greater with MLR and MLP. For instance, it required them to perform 54,100 queries on average in order to achieve 100% accuracy on a 20 hidden node MLP. Unlike BLR, it is sometimes very hard to estimate a correct amount of probes needed for MLP and MLR cloning. Especially with MLP, it is hard for attackers to guess how many hidden neuron layers are there, and how many neurons per layer. Attackers will also not be able to tell how many classes an original MLP can identify. However, everything can be different in actual cyber attack scenario. Instead of 100% accuracy, attackers may only need to clone a model with 90% accuracy for their purposes, and the

amount of probes needed may be significantly lower. Another reason to not aiming for 100% accuracy is that original ML models get tunned on a fast pace, daily. A 100% accurate cloned model of today maybe different from the actual model next week.

B. Model inversion attack

Given feature dimension d with feature vector x_1, x_2, \dots, x_d , some knowledge about some of the features, and access to f - the model, Fredrikson et al. [8] proposed that a black box model inversion attack which involves finding an optimal x that maximizes the probability of some known values.

$$x_{opt} = \operatorname{argmax}_{x \in X} f_i(x)$$

For instance, if an image of Bob was used to train model M to recognize category "man". If that exact image is fed into M , the result will be "man" with 100% confidence while images do not belong to the training set will never get such absolute score from the model. An attacker can start with one pixel and find the pixel value that gives the maximum score possible of category "man". The process continues to other pixels and the end result is an image very close to Bob's original image in the training set. Tramer et al. [7] upgraded this approach by performing inversion attack on a cloned model M' of M . The reported improvement is a 6-hour faster recovering time for 40 faces. This kind of attack opens a theoretical possibility of which attackers can gain some insightful knowledge about a security model's trained data set if they could clone the model with 100% accuracy and somehow was able to tunnel it out.

C. Path-finding attack

Tramer et al. [7] also extended prior works on tree attacks and proposed their "path-finding" attack which can be used to map binary trees, multi-nary trees, and regression trees. We have a tree T with v nodes and at each node, there is an identifier id_v . With $x \in X$, an oracle query will give $O(x) = id_v$. If $x \in X_1 \cup \perp \times \dots \times X_d \cup \perp$, $O(x)$ will return the identifier at the node where T stops. To begin the attack, we pick $x \in X_1 \cup \perp \times X_2 \cup \perp \times \dots \times X_{i1} = [a, b] \times \dots \times X_d \cup \perp$. $O(x)$ gives id_{Lv} at the leaves of the tree. We then can separate $[a, b]$ into n sub ranges where n is the number of the corresponding known leaves (at this point). For each X_{i1} sub range, we repeat the process and find another nodes/leaves. This was referred to as the top-down approach which is of higher performance than the bottom-up approach. Reported performance evaluations of this approach show that in order to achieve 100% on $1 - R_{test}$ and $1 - R_{unif}$, it will take 29,609 queries to clone a tree with 318 leaves, 8 layers of depth; 1,788 queries to clone a tree with 49 leaves, 11 layers of depth; and 7,390 queries to clone a tree with 155 leaves and 9 layers of depth.

D. Other attacks

There are several more ways to attack ML models. The Lowd-Meek attack [9] targets linear classifiers that give only class labels as models' outputs. The general idea of this approach is using adaptive queries to throw sample points at the suspected positions of the hyperplane. Another way to

attack was described by Bruckner [10] as a single Stackelberg Prediction Game (SPG). In this game, the Leader (L) is the one with the original ML model M . the Follower (F) is the attacker. F will attack L by generating and feeding model M data that at least will prevent M from learning new knowledge or at most, teach M new faulty knowledge. Theoretically, this can be achieved by providing learning data that maximize the cost function of model M . In real life situations, there are more than one attacker with different attacking goals and L does not know how many F are there and what exactly each F is trying to do. This escalates to the Bayesian Stackelberg Game. Zhou and Kantarcioglu described it as "Nested Stackelberg Game" [11] suggesting a solution of using and switching a set of models to confuse the attacker. Kantarcioglu later on also proposed the concept of "planning many steps ahead" in this game. Details of these methods will be further studied and evaluated within a defense-in-depth environment and will be discussed in future works.

V. THE MACHINE LEARNING MODEL KILL CHAIN

ML models are gaining popularity among S.O.C's and it is hard to imagine a future with lots of inter-connected devices and without ML models. However, there are many issues coming from within the S.O.C's operational structures, from models' inherent limitations, and from outside active attacks. Below is a high level kill chain targeting ML models based on all the ML vulnerabilities and limitations that we have discussed so far.

1) Recon phase

The goal of this early phase is to carefully gather as much basic information about a targeted ML system as possible. The two most important questions should be: "What are the ML model(s) used to protect the system against the kind of attacks I am planning to perform?" and "Is the targeted system based on an opensource project?" From our theoretical evaluations of ML models, we know that defenders will have certain favorite ML models in defending against certain types of attacks. We also know that open sourcing is the trend and most commercial cyber security systems have their own open source siblings. In such case, studying the anatomy of the open source system(s) can provide tremendous knowledge about the actual defense system being used. Proving efforts in this phase should be fragmented into small, seemingly uncorrelated attempts to avoid early detections. Social engineering or insider leak can be really helpful. Non technical information such as what model types are being used?, how many analysts within a certain shift? how many generated alerts created per day on an average? etc can be safely communicated out without bursting the cover of the insider.

2) Weaponization

Based on collected basic knowledge about the targeted model, attackers at this stage will work on an optimized set of probes and ideally build an adaptive engine for

the set. For example, if the model uses SVM, the values of probes should be close to where the polynomial hyperplane goes. Because a model has to deal with many attackers at the same time, the actual polynomial hyperplane at the point of attack may have a shape that is slightly different from before. In this situation, the adaptive engine will kick in, adjust the probes on the fly, and try to minimize the number of probes to a minimum.

3) Delivery

Once attackers are confident that they have prepared the best set of data probes and a good adaptive engine to handle real-time data changes, they will launch the first wave of attacks. The attack will target the decision border where the difference between true-positives and true-negative is low. After a while, analysts may "tune" the model and relax the border and/or the model's cost functions resulting in some true-negative to be accepted as false-positives.

4) Exploitation

Attackers now want to gather deeper data about the model and may as well expand probes to other models. It is reasonable to believe that at this point, attackers can deliver a full set of probes to clone the model and tunnel the data out thanks to the relaxed boundaries. Model inversion attacks can then be performed off-line to extract details about trained data. Finally, a prediction model can be built. Attackers can simulate certain attack scenarios and the prediction model can reply with its predictions of how the targeted system will react; including how the model will be tuned, how fast new rules will be formed, or even the rankings of threats. The more information attackers can harvest, the more precise this model can predict.

5) Installation

At this phase, attackers will use coordinated probes with the helps of the adaptive engine together with the prediction engine to poison the ML model. the goal now is setting up new rules and/or features that allow future attacks to happen.

6) Command and Control

Now as the attackers have established a safe path to get through the ML Model, they will move on with setting up a hidden command and control channel with the helps of compromised entities within the victim's system to further organize and expand the attack campaign. This can be combined with the regular attack kill chain and malwares.

7) Action

Finally, attackers act on their main objectives.

A. A demonstration

To demonstrate the concepts presented in the above-mentioned ML kill chain, let us take a look at a attack on a IBM cognitive classifier between domestic cats and pumas. Choosing this type of dataset gives us several benefits. First, it is non-linear and complicated enough yet classification results

are easy to be observed and understood. Second, there are many variations in between two main classes such as Golden Cat, Jungle Cat, cheetah, bobcat, lynx, etc which can be very helpful when used as attacking data. In order to simplify the scope of this demonstration, images will be focused on the animals' faces and the color patterns will be tan, brown or dark brown.

1) Setting up : A set (A1) with 10 images of cats and 10 images of pumas was fed into IBM Watson Visual Recognition demo [12] to train a model M. The Visual Recognition demo is a web interface maintained by IBM explicitly for demonstration purposes and was under IBM's control.

2) Probe 1

Attacker use set A2 with different images of cats (10 images) and pumas (10 images) to probe M. With each probe, M returns a classification (A2-C) and a confidence score (A2-S). Attacker will set up his own model M', also based on IBM Visual Recognition MLaaS. While the underlying technology is the same, M' is independent and different from M. In M', images in set A2 were uploaded and for each image in the database, there are associated A2-C and A2-S harvested from M.

3) Probe 2

Attacker will use another set A3 with images of cats and pumas but this time, for each image, attacker runs it through M' first and M second. M' will compare the image with its image library and return the closest match which includes : the image, and the associated A2-C and A2-S. M' will give a classification C and a confidence score S. If A2-C = C and A2-S = S, the process will continue other wise, the current image will be added to M' database with associated C and S scores.

4) Attack planning

Attacker now will collect a set A3 containing images of jungle cat, leopard cat, bob cat, cheetah, female lion etc and run the samples through M' and obtain associated C score and S score for each image. Attacker will then pick the ones that have S score of approximately 0.5. Those samples form a new set - A4

5) Data poisoning and assessment

For simulation purpose, we will go back to the IBM Visual Recognition demo page and perform a supervise learning like this: + We run each image to the classifier. If the classifier gives a class "Domestic Cat" with a score of 0.47 (or something close to 0.5) just as M' predicted, we will put that image into M trained database as "Pumas" and vice versa. After we are done with all items in set A4, we run items from set A2 through the newly poisoned model M again and observe the changes in confidence scores (degradation)

6) Tuning assessment

We now form a new set - A5 - with 3 times the number of items in that of set A4. We properly retrain model M with A5. We then run A2 through M and observe the new

confidence scores (degradation now with corrections)
Further details including source codes, images, outputs, etc can be found at Project WolfEye [13] on Github.

B. Future works

Since most of the mentioned ML attack methods were not evaluated as parts of a cyber attack simulation, future works will involve deploying those methods within a defense-in-depth environment where there are sensors and some basic security measures in place. Benchmarks will be done and scores such as: number of queries that can be done without being detected, total number of queries needed to achieve goals with the cost of being detected, etc will be recorded and compared. It is also within plans to target a well-known ML IDS open source project such as Apache Spot [14] and try to find ways to attack it using methods inspired by previous works [11], [9], [7]. The proposed ML kill chain will be further developed, integrating more methods and techniques that may or may not be directly related to Machine Learning but are relevant in helping the main ML attacking methods to be more efficient and less detectable. Deeper research into modern S.O.C Machine Learning work flows, policies and processes will also be done.

VI. CONCLUSION

The field of Machine Learning in Cyber Security is a very excited field. It is packed with tremendous opportunities for both legal users and hackers. Even with a well funded Security Operation Center, the challenges are still there coming from within the system and from outside. Long gone the day of detection-and- avoiding-detections games. Attackers are also be equipped with Machine Learning powers, building systems that can predict how the defending systems will react in certain attack scenarios - a multi step game in which each opponent tries to predict the other's several moves ahead. Investment in human talents are now more important than ever because among all the "digital-minds" are the human minds orchestrating them all. On the other hand, when running an advanced persistent attack campaign, it maybe helpful to also try to find patterns from the administrators such as the ways they usually tune their systems.

REFERENCES

- [1] US Bureau of Labor Statistics, "Occupational Outlook Handbook: U.S. Bureau of Labor Statistics," 2016. [Online]. Available: <https://goo.gl/foVZk0>
- [2] C. V. N. I. Forecast, "Cisco visual networking index: Global mobile data traffic forecast update 2016-2021," *Cisco Public Information, February*, 2017. [Online]. Available: <https://goo.gl/tIlk4D>
- [3] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains." [Online]. Available: <https://goo.gl/HOhsqh>
- [4] Cisco, "NetFlow Version 9 Flow-Record Format [IP Application Services]," 6 2011. [Online]. Available: <http://tiny.cc/lcvoky>
- [5] M. Blowers and J. Williams, "Machine Learning Applied to Cyber Operations." Springer New York, 2014, pp. 155–175.
- [6] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 22 2016. [Online]. Available: <http://ieeexplore.ieee.org/document/7307098/>
- [7] F. Tramèr, F. Zhang, F. E. Epfl, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing Machine Learning Models via Prediction APIs," in *Proceedings of the 25th USENIX Security Symposium*, 2016.
- [8] M. Fredrikson, S. Jha, and T. Ristenpart, "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures." [Online]. Available: <http://dl.acm.org/citation.cfm?id=2813677>
- [9] D. Lowd and C. Meek, "Adversarial Learning." [Online]. Available: <http://dl.acm.org/citation.cfm?id=1081950>
- [10] M. Brückner and T. Scheffer, "Stackelberg Games for Adversarial Prediction Problems."
- [11] Y. Zhou and M. Kantarcioglu, "Modeling Adversarial Learning as Nested Stackelberg Games." [Online]. Available: <https://goo.gl/UiJsUK>
- [12] IBM, "IBM - Visual Recognition Demo." [Online]. Available: <https://visual-recognition-demo.mybluemix.net/>
- [13] Tam Nguyen, "Project Wolf Eye," 2017. [Online]. Available: <https://github.com/genterist/wolfeye>
- [14] Apache Organization, "Apache Spot." [Online]. Available: <http://spot.incubator.apache.org/>