Machine Learning Security Survey

Turhan Kimbrough
Department of Computer Science
Towson University
Towson, Maryland
tkimbr1@students.towson.edu

***Abstract*—Machine learning has experienced a significant growth in usage over the past few decades. Due to its data-centric approach in modeling, machine learning has seen use in a variety of subfields in computer science. In particular, researchers have been interested in incorporating machine learning into the domain of cybersecurity, utilizing it from the perspective of an adversary or ally. Researchers have also been concerned with the security state of current machine learning models. This survey paper provides an overview of machine learning, vulnerabilities and mitigations to current machine learning systems, machine learning use cases for adversaries, and future research directions.**

***Keywords*—Neural networks, security, classification, machine learning model, training, validation**

I. Introduction

With the increasing widespread adoption of machine learning technology, its usage is being observed in a variety of different fields. Some notable examples include image recognition, voice assistant technologies, email spam filters, and search engines. Much of its recent popularity can be attributed to the availability of frameworks such as tensorflow, allowing people of almost any background to quickly draft a machine learning application.

However, one growing concern tied to the ubiquity of machine learning is its accessibility to adversaries. Based on the assessment of current and prior research, there are a number of vulnerabilities in current machine learning models which can be exploited with little knowledge of a system's domain. In addition, attackers have been able to leverage machine learning technology to assist the deployment of cyberattacks. Enterprise machine learning applications may often contain large datasets of important information, becoming a potential candidate of a targeted attack. This paper will survey the domain of cybersecurity with regard to machine learning. This topic will explore the fundamentals of machine learning, current vulnerabilities in machine learning systems, mitigations to machine learning vulnerabilities, machine learning technology from the perspective of adversaries, and future research directions.

II. Overview

This section will introduce the fundamentals of machine learning. Additionally, there will be a discussion about the key terms used by members of the community.

A. Machine Learning Basics

Traditionally, when software developers are tasked with solving a problem, they use a combination of rules and logic to find a solution. The basic routine consists of finding appropriate input values, creating the logic and rules to process the input, and producing the appropriate output. The traditional approach to software development allows for fine-tuned control of program behavior to achieve the solution. However, this approach does not scale with the complexity of additional rules and/or possible solutions. An example is image classification, where the logic needed to compare images is complex. This becomes a bigger concern when new classifications need to be derived with new image data. Machine learning flips the traditional programming paradigm on its head, by taking a series of solutions as input, and letting the machine develop the rules by detecting patterns in the solutions. The result is a self-propagating mathematical model, capable of making decisions on newly supplied data. The mathematical model is often represented as an *Artificial Neural Network* (ANN), whose name is inspired by the biological brain, mimicking the way neurons interact with one another. ANN's comprise of an input layer, one or more hidden layers for data processing, and an output layer for decision-making. This approach relies on large sets of well-defined data, and has the flexibility for being used in many different applications.

Briefly mentioned earlier, a common use case for machine learning is *classification*, where a dataset is

categorized into different groups based on one or more *features*. A *feature* is defined as some measurable property or characteristic being observed in a dataset. There are several types of classifications, including *binary classification*, *multi-class classification*, and *multi-label classification*. *Binary classification* categorizes data based on whether a feature is present or not, resulting in an outcome of true or false. *Multi-class classification* categorizes data into different groups, where each data instance is assigned according to its feature. *Multi-label classification* categorizes data into different groups, where each data instance is assigned according to its expression of on or more features.

Training is the process of teaching a machine learning model to detect patterns in datasets. There are two types of training mechanisms, *supervised training* and *unsupervised training*. *Supervised training* requires each data instance to have one or more labels, defining which category or feature it expresses. In contrast, *unsupervised training* omits the need for labels, and the machine learning model will categorize datasets on its own. Typically, after the training phase, a machine learning model will go through the process of *validation*. *Validation* typically consists of classifying a separate dataset to guarantee the accuracy of a model and preventing a phenomenon called *over-fitting*, where a model will only 'memorize' characteristics or patterns of training data.

III. Vulnerabilities in Machine Learning Models

Due to the unique mechanism for constructing machine learning models, there are several ways which they can be exploited by an attacker. As a result, the attack surface which exists on machine learning models will differ from traditional software systems. This section will discuss several vulnerabilities which are currently present in machine learning.

A. Data Poisoning

Data Poisoning is the act of manipulating, removing, or adding data during the training phase of machine learning. This type of attack is known as a *black box attack*, where an attacker does not need to know the implementation of a system to attack it.

One popular instance of a data poisoning attack is the adversarial example. This requires the attacker to have some knowledge of the training data, such as its dimensions and data type. The attacker would then manipulate data instances in a way where the machine learning model would be fooled, but appears normal to a human observer. The reason for manipulating data instances in this way is two-fold. First, machine learning models are often constrained to data fitting a specific dimension, shape, size, or length of characters. This is typically done to prevent incompatible data from entering the model during training. Second, there is often one or more people who are observing the model with testing or validation datasets. An attacker would want to minimize any evidence of the data being tampered with.

Often, the goal of this attack is to make a machine learning model incorrectly classify data. The consequences of this attack can be devastating. Examples include tricking an autonomous vehicle to misinterpret traffic signs, sneaking malicious data past an intrusion detection system, or bypassing email spam filters.

B. Membership Inference

Membership inference is a mechanism of data extraction, where an adversary intends to know whether certain samples were used as training data for a machine learning model. This type of attack is also classified as a *black box attack*.

This particular vulnerability requires significant effort from the attacker. The attacker would need access to a dataset of sufficient size mimicking the data in the target model. The data would then be used to create several *shadow models*, which are used only to recognize differences in the target model's behavior. This is done to expose *overfitting*, when a model's analysis corresponds too closely to its training data. Therefore, an attacker can interpret whether certain samples were used in the training dataset based on the target model's confidence level in classifications.

Often, this can exploit the confidentiality of information on a system. An attacker has the capability to correlate information between datasets to target individuals for other cyberattacks.

C. Transfer Learning

Transfer learning is a mechanism where an adversary has the ability to study a publicly available machine learning model, and use that insight to sneak past and/or corrupt similar target systems. This type of attack is classified as a *white box attack*, since the attacker would need to have full access to at least one machine learning model.

This particular vulnerability requires the attacker to have knowledge of the input data, learning mechanism, and output behavior of a machine learning model similar to the target. Once this information is obtained, the

attacker would test the system and learn its overall behavior to enumerate flaws in the model's logic. The assumption is that the flaws found in the available model's logic would appear in a target system.

Often, the goal of this attack is similar to the other two mentioned above. An attacker would use the information to trick, fool, manipulate, or sneak passed a target machine learning model.

IV. Mitigations to Vulnerabilities

Due to the large attack surface found in machine learning models, there has been a significant focus in its security. This section will discuss several mitigations to the vulnerabilities discussed in the previous section.

This section will also reference a generic process known as the *machine learning lifecycle*. This lifecycle consists of building the dataset, feeding the input to the model, training the model, verifying the model accuracy, deployment, and maintenance.

A. Selecting Trusted Datasets

To secure a machine learning system, it is important to start at the beginning of the machine learning lifecycle, securing the input data itself. Since the training phase typically requires a large amount of data, many projects utilize publicly available datasets. Publicly available datasets are similar to open source projects, in that contributions are made by and verified by a community of individuals. This means that the trust is placed on the community to provide legitimate information. To find trusted datasets, one can check the policy of verifying contributions, number of community members/maintainers, and entities backing the project. For example, datasets from the Government, Universities, and repositories such as kaggle are typically trustworthy.

If a dataset is chosen from a location with an unknown reputation, it is recommended to perform an audit to ensure the data consists of relevant samples, proper labels, and a balanced set of classes.

B. Sanitizing Input

The next step in securing a machine learning system is to use mechanisms which filter malicious samples from clean samples during the training phase. There are several different approaches to sanitizing depending on the type of data being fed to the model. A general procedure for sanitizing input data is still an open research topic.

One naive approach for filtering data is the use of *gradient masking*. This simple technique will manipulate each input sample to create a sharper decision boundary for the machine learning model to work on. A common implementation of this technique for images is *binary thresholding*, where each pixel in an image is converted to black or white depending on its color value. This technique will mitigate against perturbations to data.

Another approach called ANTIDOTE, proposed by Rubinstein *et al*, uses an *anomaly-based* detection scheme to characterize data samples. ANTIDOTE uses statistics to differentiate between normal and malicious samples. Once a malicious sample is detected, it is automatically discarded from the model.

C. Machine Learning Retraining

After a machine learning model is finished with its initial training and verification, it is typically ready to be deployed. After deployment, a good practice to maintain a machine learning model is to periodically retrain it. Retraining can either use the original dataset, or a new dataset containing similar information as the original. This is ideal for machine learning models which continuously learn post-deployment.

A specialized approach to retraining using samples of malicious data is a technique called *adversarial retraining*. This process requires a sufficient number of malicious samples with perturbations, along with the original training data. The technique involves labelling malicious data with perturbations as adversarial samples, and training the model to detect them appropriately. This ensures that adversarial samples will not affect the overall accuracy of the model, by learning to filter perturbations added to any new data. The model will learn to differentiate between legitimate and adversarial data when re-deployed.

D. Differential Privacy

During deployment, data will typically be flowing in and out of a machine learning model. Information coming in and out of a machine learning model may be sensitive, so systems will typically incorporate APIs to interface with it. Even if a system is robust, a data leak will completely compromise confidentiality. To circumvent this issue, *differential privacy* is a mechanism to effectively store information while hiding confidential details.

The implementation of differential privacy will differ depending on the data being processed. A standard mechanism for differential privacy is to present output data showing non-sensitive information normally, while sensitive information is encrypted. Another mechanism

is presenting the output information of a machine learning model as a reference to another dataset, which will then need to be queried with subsequent processing.

V. Assisting Cyberattacks with Machine Learning

After discussing the vulnerabilities/mitigations for machine learning systems from the perspective of the defender, this section will discuss how machine learning technology is used to assist with the deployment of cyberattacks from the perspective of the attacker.

A. Evasive Malware

Modern-day malware detection schemes use several different techniques to classify malicious and safe programs. One method is the use of detection engines, which are often available in two distinct types, *signature-based* detection engines and *anomaly-based* detection engines. A signature-based detection engine keeps a record of malicious behaviors and footprints, comparing software behaviors against the record to identify malware. On the other hand, an anomaly-based engine will compare software behaviors against a record of 'normal' system behavior to detect deviations, classifying them as malicious actions. Both approaches can be extended with machine learning capabilities to improve their detection rates. In particular, supervised learning has been shown to be an effective strategy.

However, even with the advancements in malware detection schemes, research has shown that there can be a significant flaw associated with detection engines using static features and definitive labels. In particular, a framework named *DQEAF* has demonstrated the ability to evade malware-detection by the use of reinforcement learning. DQEAF operates by using a machine learning agent to interact with different malware samples. During its interaction, it will slightly modify the behaviors of detected malware samples without impacting its overall structure and/or functionality. The agent will continuously modify detected malware samples until it is able to be completely undetected by a malware detection engine.

B. CAPTCHA Bypass

CAPTCHA is an acronym which stands for the "Completely Automated Public Turing test to tell Computers and Humans Apart". CAPTCHAs are a mechanism to combat against the growing number bots which are used on the internet, ideally providing a challenge which is relatively easy for humans to solve, but difficult for machines. CAPTCHAs come in a variety of types, including the deciphering of obfuscated text, interpreting audio messages, selecting images based on descriptions, and the tracking of end-user behavior.

Over the past few years, CAPTCHAs have grown to be more difficult due to the techniques used by adversaries. The techniques include the exploitation of weaknesses in the CAPTCHA generating mechanism and using machine learning for CAPTCHA solving. Machine learning CAPTCHA solving is an especially attractive option, given the automation capabilities. In fact, research has suggested that reinforcement learning has provided the capability to solve one of Google's reCAPTCHA mechanisms, which records mouse movements to distinguish human and bot behavior. This was achievable by representing the pixels on a screen as a matrix, and using a software agent to move the cursor across the screen. The *Markov Decision Process* is the algorithm which was used to generate random movements, and trains the software agent to behave more human-like over time. This process solved reCAPTCHAs with greater than 90% accuracy.

C. Brute Forcing Passwords

One of the most prevalent cybercrimes in today's age is account hijacking. From banking, social media, streaming, and more, digital accounts play a major role in the lives of everyday people. Adversaries are interested in gaining access to these accounts in an effort to unravel useful information. With the works of prior research and the open source community, there are a significant number of tools available for account hijacking. One of the simplest tools are *brute force password-crackers*, which typically employ a random combination of characters to guess the resulting password.

Due to the increasing awareness in cybersecurity best practices, many sites with accounts will employ mechanisms to enforce stronger passwords. As a result, simple password crackers are no longer able to make guesses within a reasonable amount of time. However, research suggests that a generative machine learning model is capable of creating more accurate guesses based on previously exposed password datasets. The approach uses the *Markov Model* with a neural network to derive characteristics from a password dataset, and generate new passwords with the same characteristics. Comparing the generated passwords against a separate leaked dataset, the researchers were able to match up to 42% of the passwords using the generative machine learning model.

VI. Future Work

Much of the research which has been conducted in the field of machine learning security has been relatively new. As a result, the research community still has many questions as to what components are necessary and/or feasible for securing future machine learning systems. This section will outline potential directions for future research based on what has been reviewed in prior sections.

A. Trusted Platforms

Currently, the usage of *Trusted Platform Modules* (TPMs) are seeing a rise in popularity for securing systems. TPM technology typically consists of a set of cryptographic algorithms along with a dedicated processor known as a TPM chip. The purpose of a TPM chip is to provide a dedicated hardware device to independently verify that a software system has not been tampered with.

An interesting research direction would be to incorporate TPMs into a machine learning system. In addition, there has been a recent surge in mobile devices which ship with dedicated machine learning processing units, such as Apple MacBooks powered by M1 processors and Samsung Galaxy S-series flagship devices. It would not be surprising if researchers would be interested in using a TPM and machine learning processing unit as building blocks for a trusted platform.

B. Generative Adversarial Networks

The concept of Generative Adversarial Networks is still a relatively new, being discovered only within the last decade. Generative Adversarial Networks consist of two separate neural networks, a *generator* and *discriminator*. Given an initial training data set, the goal is to produce new data which mimic the characteristics of the training set. This is accomplished by using the *generator* to produce realistic fake data from a random seed, and the *discriminator* which learns to differentiate the fake data and real data. The two networks are in an adversarial relationship, hence the name, and work toward minimizing the differences between the fake and real data. This concept has brought significant insight into the inner-workings of machine learning along with insights on the relationships between data samples.

An interesting research direction would be to use Generative Adversarial Networks to aid in model retraining. In order to do this, a variety of malicious data samples would need to first be aggregated. Then the *generator* and *discriminator* pair would work to create a model which can realistically create new malicious data samples. This new (malicious) model can then be used to generate labeled malicious data, which will be fed to machine learning model which needs to undergo retraining. The machine learning model which is being retrained will have a more reliable malicious dataset, which can strengthen its detection mechanism.

C. Reinforcement Learning Framework for Anti-malware Engines

VII. Conclusion

References

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, “Title of paper if known,” unpublished.

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.