Modeling and Security Analysis of Attacks on Machine Learning Systems

Anoop Singhal

NIST

Email: anoop.singhal@nist.gov

***Abstract*— Machine learning (ML) systems are more and more widely deployed. At the same time, ML systems also introduce new security threats that are not seen in traditional software and network systems. For example, data poisoning and adversarial attacks generate incorrect output; membership and denial of service attacks exploit the new attack surface which waste computation resources. However, existing security analysis frameworks fail to sufficiently handle the new causality relationships, notions and elements introduced by ML systems. In this paper, we present a new way to model causality relationships in ML systems for security analysis. We present our model with a running example and discuss how our model can be used for security analysis and risk mitigation.**

Keywords — Machine learning, System security, Causality modeling, Attack Graphs

I. INTRODUCTION

Last several years have witnessed rapidly increasing use of machine learning (ML) systems in multiple industry sectors. Auto driving cars are using object detection systems to process the images/videos from the cameras to understand the traffic signals and real time traffic around them. ML has been used to translate text from one language to another in several systems. Deep Learning has been used in products such as Google and Mozilla to understand speech. The security of such ML systems thus becomes a critical issue that can have high impact in a lot of associated applications.

In existing traditional network security analysis, causality analysis has been a very important tool. It usually performs logic reasoning towards the existing preconditions (such as network connections, services provided, and vulnerabilities in programs, etc.) to infer what security risks can be posed. For example, if an attacker can remotely access a vulnerable server program, this can cause the vulnerability being exploited. Causality relations exist between the preconditions and the exploit. Attack graph is one of the fundamental tools for network security causality analysis that mainly focus on relationships between security vulnerabilities (such as CVEs - Common Vulnerabilities and Exposures) and exploits (which mainly focus on newly gained permissions/accesses).

Inspired by the causality analysis for traditional network security, it is thus very natural to explore the possibility of performing security analysis towards ML systems by analyzing the causality relations existing in the ML systems. However, it is widely recognized that the existing security causality analysis frameworks and techniques, which were developed to analyze enterprise (software) systems and networks, are not very suitable for analyzing ML systems. ML systems have new kinds of causality relationships which are unable to be sufficiently handled by current approaches for security analysis. Moreover, we found that the notions (and elements) used in modeling the causality relationships captured by existing frameworks also fall short of modeling the causality relationships involved in ML system-specific attacks. For example, when we model the causality relationships involved in data poisoning attacks, which refer to modifying the training data, the notion of gained permissions/accesses is no longer adequate in modeling the exploits intended by data poisoning attacks. Instead, in order to model such exploits, we found that new kinds of notions, such as backdoors inside a DNN, are actually required. Note that such backdoors should not be viewed as a special kind of access, since they don't enable the attacker to gain any read/modify access to the DNN itself. Rather, such backdoors correspond to a primary causal effect of data poisoning attacks, that is, the trained ML model is maliciously altered.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

Based on the above observations, there is an urgent need for the security causality analysis research community to answer the two following questions:

– Q1: What are the new kinds of concepts that would play an essential role in modeling the causality relationships involved in ML system-specific attacks?

– Q2: What kind of security causality analysis models would be resulting from these new kind of concepts?

In this paper, we seek to take the steps towards answering these two questions. In particular, our main contributions are as follows. First, we explain why traditional causality analysis models fall short when analyzing ML systems. Second, we present a new way to model ML system's causality relationships for ML security analysis. We identify the key concepts for modeling causalities in ML system-specific attacks and define dependencies in the context of ML systems. Finally, we discuss how our model can be used for security analysis to solve security problems.

The rest of the paper is structured as follows. Section 2 discusses why existing security causality frameworks are not suitable for analyzing ML systems. Section 3 presents our proposed model. Section 4 shows how our model can be used for security analysis. Section 5 discusses related works, and Section 6 is the conclusions.

II. Why Existing Network Security Analysis Models Fall Short of Analyzing ML Systems?

A. Background

In traditional enterprise network security analysis, capturing and leveraging the causalities related to exploits is an important approach to reveal potential security risks. It's not surprising that various kinds of software vulnerabilities may exist in an enterprise network. Exploiting these vulnerabilities can help attackers gain new accesses or permissions on the victim machine. A successful exploit usually requires certain prerequisites to be satisfied. For example, if attackers want to exploit a vulnerability on a server, all the following conditions need to be satisfied: a) the attackers have (remote) access to the machine, b) the machine is running a specific type of service; and c) the corresponding server program has a vulnerability. Once all three conditions are satisfied, the server program can be compromised, and attackers may run arbitrary malicious code on this server. In this scenario, a causality relationship exists among the three existing conditions (namely pre-conditions) and the consequence (namely post-condition) of "attacker running arbitrary code on the server".

Although the existing security analysis frameworks are mature enough to handle the traditional attacks towards enterprise networks, they fall short of analyzing the ML systems due to the reasons below.

B. The First Reason

ML systems have new kinds of causality relationships which are unable to be sufficiently handled by existing security causality analysis frameworks. For example, as the fundamental tool for enterprise security causality analysis, attack graphs mainly focus on relationships between security vulnerabilities and exploits. However, in ML systems, attacks are not necessarily towards a certain vulnerability. Take the data poisoning attack as an example, the entire attack process only involves crafting malicious data samples and feeding them into the model training phase. The ML system is vulnerable because it relies on the data for training the model, but the system itself may not have an explicit vulnerability that can be exploited. Therefore, traditional security causality analysis tools are no longer sufficient. To capture the new types of causality relationships in ML systems, we need to examine the new dependency relations brought in by ML systems but may not be present in traditional networks.

- Data dependencies show how data is used to train a model. An important threat against ML systems is the potential adversarial control of the training data or the training process, with the goal of modifying the model Such dependencies can capture the dependency relationships among data and models, and can thus be used to analyze attacks such as data poisoning attacks.
- Model dependencies can start with pre-trained models (if the ML developers do not want to start from scratch) or with model training (if the ML developers don't use existing models but start from scratch). Taking model evaluation as an example, it takes all trained models as one of the inputs, and outputs the best-performing model.
- Library dependencies show dependencies due to software libraries that are used for development of the ML system. For example, ML framework (scikit-learn, TensorFlow and so on.) will support any process (model training, model testing, post-deployment inference, etc.) that involves the model.

C. The Second Reason

The notions (and elements) used in modeling the causality relationships captured by existing frameworks also fall short of modeling the causality relationships involved in ML system-specific attacks. Currently in dependency attack graphs, a vertex represents a condition, and the edges between vertices represent the causality relationships. The condition is a statement describing a specific system or network condition, such as "the program has a vulnerability", "the web server has access to the database server", or "can execute arbitrary code on a machine". The edges represent the causalities among the preconditions and postconditions. These causalities are mainly about the vulnerabilities and exploits. With the new dependencies introduced by the ML systems, only specifying the system or network conditions is not sufficient. More pre- and post-condition types are needed to describe details about the data, models, and libraries. Since the software libraries are mostly traditional program, the notions in traditional attack graphs can still be reused to capture exploits to the vulnerabilities in these software libraries. Such exploits may further affect other components in the ML systems. Software library dependencies are a great bridge to connect traditional enterprise network security analysis with the new ML security analysis. In addition, the causality relationships need to be significantly enriched due to introduction of new dependencies in data, models, and libraries.

Identify applicable funding agency here. If none, delete this text box.

III. A New Security Analysis Model for ML Systems

A. A Running Example

To demonstrate our model, we will use the word-to-word (w2w) translation machine learning (ML) system as the running example. As the name suggests, the word-to-word translation machine learning system takes a word from one language and outputs a word of the same meaning in another language. For simplicity, we will use English to Spanish as the example, referred to as the Eng-Spa w2w ML system, as shown in Fig. 1.

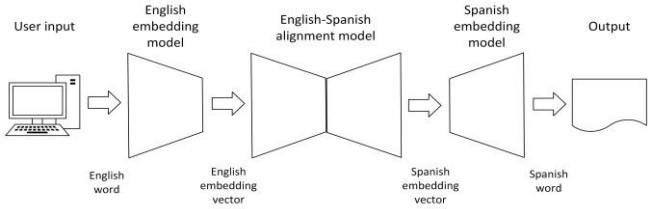


Fig.1: Running example: Eng-Spa w2w ML system.

The w2w ML system has two parts, embedding model(s) and alignment model(s). The input word first goes into the embedding model, where the word is converted to an embedding vector, and then the vector is input to the alignment model for mapping to another language's embedding vector. Finally, the mapped embedding vector is transformed into another language using the corresponding language's embedding model. In the case of Eng-Spa system, the English word first goes into the English embedding model and gets transformed into English word embedding; then, the English word embedding goes into the Eng-Spa alignment model and gets mapped to the corresponding Spanish word embedding; and, finally, the Spanish word embedding gets transformed back to the Spanish word using the Spanish embedding model.

In the w2w translation ML system, the embedding model is trained using unsupervised learning with articles in the corresponding language. For example, web pages from Wikipedia can be crawled and processed into the training data for this unsupervised learning process. The alignment model, on the other hand, is trained using supervised learning approach. A set of labeled source and target language word pairs needs to be prepared for this process. Using the embedding models of the two languages, word pairs can be transformed to embedding pairs, and the alignment model can be trained with the embedding pairs using supervised learning.

As shown above, the w2w translation ML system contains causality relationships which cannot be sufficiently handled by existing security causality analysis frameworks. For example, the input data needs to go through three models to produce the final outputs. Also, the alignment model can only be successfully trained with acceptable embedding models. These introduce complex data dependencies. Also, considering that there have already been many existing language embedding models and libraries, developers can utilize them instead of developing everything from scratch, which will also introduce complex model and library dependencies.

B. Threat Modeling for ML Systems

As indicated by the running example, when the correctness of an ML model is compromised, a most noticeable "sign" is misclassification. As mentioned in Section 2, various (classes of) ML system-specific attacks would result in an incorrect model for Word to Word translation.

Notions for Modeling the Causality in ML System-Specific Attacks. What are the new kinds of notions that would play an essential role in modeling the causality relationships involved in ML system-specific attacks? To answer this question properly, we take a two-step approach. In Step 1, we identify the preconditions and postconditions relevant to each class of attacks. In Step 2, we synthesize all the identified preconditions and postconditions with a focus on the notions that play an essential role in modeling two or more classes of attacks. In particular, our synthesis notions are as follows:

- Backdoors are a set of tiny patches maliciously put on a normal input of the target DNN model. They can be put in in any stage before output is produced, for example, by tainting the raw input, when raw data is being processed, or even by flaws during the training process.
- Dataset-Model Dependencies: When a dataset is used to train a model, we say that the model's parameters depend upon the dataset.
- Model-Model Dependencies: When a pre-trained model serves as the starting point for training a new model, we say that the new model's parameters depend upon the pre-trained model.
- Library-Model Dependencies: When the training of a model involves the execution of code in a particular library, we say that the model's parameters depend upon the library. When the inference computation of a model involves the execution of code in a particular library, we say that the model's prediction outputs depend upon the library.

All existing security causality analysis models allow analysts to specify the system's attack surface (e.g., the attack surface specified in [15]) at the abstraction level of how vulnerabilities interact with hosts and each other. From the viewpoint of security administration, this abstraction level is appropriate: each host (i.e., endpoints) has a specific security configuration and is a basic unit of security management. However, the new notions don't provide a clear understanding of how ML system-specific vulnerabilities interact with hosts and each other.

Without bridging the gap, an ML system-specific security causality analysis model needed by analysts is unlikely to be developed. On the other hand, one main objective of such a model is to provide analysts with both a clear understanding and a clear view of how ML system-specific vulnerabilities interact with hosts and each other. Let's take a specific DNN model backdoor as an example.

First, since the backdoor is a specific flaw of a DNN model running on a specific target host Hy, the backdoor and the host can jointly define a "backdoor model vulnerability on Hy" node in our proposed causality graph.

Second, in case the adversary does not have direct access to host Hy, Hy is actually not part of the attack surface, and the

backdoor must be implanted through a host Hx which is part of the attack surface. Accordingly, Hx should be a node in our causality graph. In the word-2-word translation example, Hx is a website holding maliciously modified training data. Hence, this part of the attack surface can be defined as a "data contamination on Hx" node in our causality graph.

Definition 1 (Dataset Dependency between Hosts): Host H2 depends upon H1 if a dataset used by ML programs running on H2 includes one or more data samples collected from H1.

Definition 2 (Embedding Dependency between Hosts): Host H2 depends upon H1 if an embedding (i.e., a set of vectors) used by ML programs running on H2 includes one or more vectors generated on H1.

Definition 3 (Model Dependency between Hosts): Host H2 depends upon H1 if a pre-trained model used by ML programs running on H2 is pre-trained on H1.

Definition 4 (Library Dependency): Host H2 depends upon software supply chain SSC-1 if a library used by ML programs running on H2 is created by SSC-1.

Using the four definitions, our causality graph is able to find all the missing pieces we intend to find. First, using the aforementioned "data contamination on Hx" node as a seed, we can find all the hosts that depend upon Hx in terms of Dataset Dependency. Note that directed edges will be used to connect the seed node to each newly identified node. Second, we can find the transitive closure of the set of nodes we have identified so far. Note that the transitive closure is defined not only based on Dataset Dependency, but also based on Embedding Dependency and Model Dependency. In addition, it should be noticed that whenever the "backdoor on target Hy" node is (newly) identified as a member of the transitive closure, we will treat it in a special way, that is, we will not use it to further expand the transitive closure since the "backdoor on target Hy" node is the ultimate defect pursued by the adversary in launching data poisoning attacks. Third, we remove all the paths in the graph (we have obtained so far) which don't contain the "model vulnerability on Hy" node. Fourth, if there exist library dependencies between any nodes in the graph and any software supply chains, we will add the corresponding SSC nodes (and edges) to the graph. Finally, we will add the "attack consequence: misclassification" node to the graph and use a directed edge to connect the "backdoor on target Hy" node to this node.

Definition 5 (Mapping between parts of the attack surface and the ultimate defect node): This is a mapping table associated with the entire causality graph. Each row in the table corresponds to one attack scenario and has 4 fields: the first field specifies a particular part of the attack surface; the second field specifies the path(s) involved in the attack scenario; the third field specifies the conditions under which the defect specified in the ultimate defect node would be caused; the fourth field specifies the intermediate defects. For example, Table. 1 shows the mapping table for the two example attack scenarios, which are depicted in Fig. 2b and Fig. 2c.

Table 1: Example mapping table

	Attack scenario #1	Attack scenario #2
Attack surface	Data contamination on host Hx	Word embedding vectors on Hq
Path	Wikipedia contamination on Hx –> Wikipedia corpus on Hp –> Word embedding vectors on Hq –> Backdoor on target Hy	Word embeddings on Hq –> Backdoor on target Hy
Conditions	Contamination is significant enough & Web crawler on Hp does not check word validity & Corpus on Hp includes all bogus entries	Attacker has write access to word embeddings on Hq
Intermediate defects	Poisoned Wikipedia corpus on Hp & Poisoned word embedding vectors on Hq	

Note that the mapping table intends to provide a clear understanding on not only "what would be sufficient in launching a successful attack" but also "what is a complete attack scenario". This notion of completeness could be very useful when conducting diagnosis: when both complete and incomplete attack scenarios are identified, the security analyst may ignore the incomplete ones.

Backdoor on target Hy by data poisoning. In the first attack scenario, the attacker plants a backdoor on Hy by poisoning its training data. The first field is about attack surface, which is "data contamination on H_wiki" in this scenario. Here, H_wiki is the Wikipedia website where the adversary makes up a new English word and create N

Wikipedia entries using the new word. The second field talks about the impact propagation path, which is the following path in the causality graph: the "data contamination on Hx" node →the "Wikipedia corpus on Hp" node (this host runs a web crawler) →the "word embedding vectors on Hq" node (this host runs a self-supervised machine learning program which uses the Wikipedia corpus to train the Word2Vec neural network) →the "backdoor on target Hy" node (this host runs the deep learning program which outputs the word translation neural network). The third field holds the following conditions: (the value of N is greater than a threshold) AND (the web crawler running on Hp does not check word validity) AND (the corpus on Hp includes all the bogus Wikipedia entries). N is the number of Wikipedia entries created using the new word. The fourth field holds the following intermediate defects: (a) the Wikipedia corpus on Hp is poisoned; (b) the word embedding vectors on Hq are poisoned.

It can be inferred from Figure 2 and Table 1 that, because the original data has been contaminated, a large part of the ML system has been tainted, including corpus on Hp, embedding

vectors on Hq, and Hy where the model is located. The attacker has penetrated Hp, Hq, and Hy.

Backdoor on target Hy by directly altering word embedding. In the second attack scenario, the attacker plants a backdoor on Hy by directly modifying the embedding vectors. The first field is the attack surface, which is "word embedding vectors on Hq". Here, the adversary has write access to this host. With the write access, the adversary directly adds a few bogus embedding vectors into the word vector lookup table on this host. The second field holds the following path in the causality graph: the "word embedding vectors on Hq" node →the "backdoor on target Hy" node (this host runs the deep learning program which outputs the word translation neural network). The third field holds the condition under which the attack can success: the attacker has write access on host Hq. The fourth field holds the following intermediate defect: the word embedding vectors on Hq are poisoned.

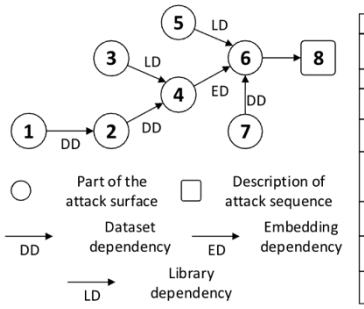
It can be inferred from Figure 2 and Table 1 that, because the attacker has write access on host Hq, where the embedding vectors are located, components afterward have been tainted, including embedding vectors on Hq, and Hy where the model is located. The attacker has penetrated Hq and Hy.

Modeling how ML system-specific vulnerabilities interact with each other. So far, we have only discussed how to model individual vulnerabilities. In this section, we will discuss how to model the interactions between vulnerabilities.

This aspect of modeling has three notable benefits: First, when a ML system has multiple kinds of vulnerabilities, the same type of prediction incorrectness (e.g., model misclassification) could be due to the exploitation of different kinds of vulnerabilities. Therefore, from the perspective of diagnosis, our causality graph should help security analysts solve the diagnosis challenge. Second, some nodes in our causality graph may be shared by two or more attack scenarios. This indicates that the possible attack scenarios in a complex ML system are not completely independent. In other words, adversary A could "reuse" the (intermediate) effects of an attack previously launched by adversary B to make his or her attack less difficult and less costly. From the perspective of risk mitigation, the shared nodes may deserve more attention (and investment) from security teams. Third, from the viewpoint of security management, the administrator responsible for managing the security of host Hz can use our causality graph to gain comprehensive awareness of all the vulnerabilities (and attack scenarios) related to Hz.

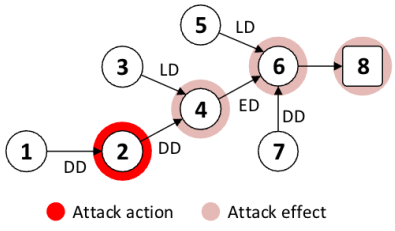
Regarding how to model the interactions between vulnerabilities, our insights are as follows: Let's assume that we have already constructed one causality graph for vulnerability A and another causality graph for vulnerability B. The two causality graphs can be merged through the following procedure: First, when one node in Graph A and another node in Graph B are the same node, we will merge the two nodes into a single node while keeping all the involved edges in the merged graph. Second, in the merged graph, we remove redundant edges as follows: assume two nodes are connected by a set of edges, if any subset of edges in the set are identical, we will use a single edge to replace the subset.

Regarding the two mapping tables associated with Graph A and Graph B, respectively, since each attack scenario is complete, the two mapping tables can co-exist in the merged causality graph without being merged. Nevertheless, since two different kinds of ML system-specific attacks might be interfering with each other, we will add a special "conflicting conditions" table to provide security analysts with adequate awareness about how different kinds of attacks may interfere with each other. In particular, each row in the special table contains the following fields: The first and second field holds the identifier of one attack scenario associated with Graph A and B respectively and the identifier of one or more condition for this attack scenario.The third field specifies the conflict. For example, one such conflict could be as follows: "If the condition mentioned in the first field is satisfied, then the condition mentioned in the second field won't be satisfied."

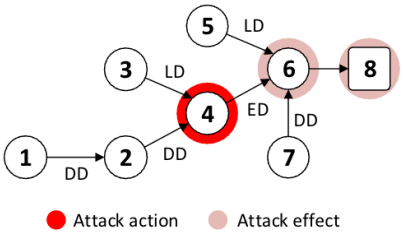


Node ID	Description
1	A few English words submitted to host H_wiki.
2	The corpus on host H_p is modified.
3	Changes are made to the supply chain of the libraries used in Word2Vec.
4	The database of embedding vectors on host H_q is modified.
5	Changes are made to the supply chain of the libraries used in neural network training.
6	A backdoor is implanted in the alignment model trained on host H_y.
7	English-Spanish word pairs on host H_pairs
8	The attack consequence of downgraded output correctness.

(a) General causality graph.



(b) Causality graph of data poisoning attack.



(c) Causality graph of directly modifying embeddings.

Fig.2: Causality graphs. responsible for managing the security of host Hz can use our causality graph to gain comprehensive awareness of all the vulnerabilities (and attack scenarios) related to Hz.

IV. Application of the Proposed Model for Security Analysis

In this section, we will illustrate using our model for security analysis using the running example in Subsection 3.1. We will present some of the challenges for designing mitigation strategies against attacks on ML Systems. Here is a list of some of the questions that our causality graphs can be used to answer: 1) Which part of the ML system gets tainted due to the attack? 2) How far into the ML system does the attack penetrate? 3) Will the defenses proposed in the literature be effective in preventing the attack?

We have directly answered the first and second questions in Section 3 with the causality graph and mapping tables presented. For example, in the scenario of backdoor by poisoning raw data, we can infer that a large part of the ML system has been tainted, including corpus on Hp, embedding vectors on Hq, and Hy where the model is located, and the attacker has penetrated Hp, Hq, and Hy.

To answer question 3, the effectiveness of defense mechanism, we can look at how the proposed defense affect the attack scenario. Our model can help deploy defense mechanism by 1) firstly constructing the propagation path; and then 2) identifying key components in the propagation path that can be isolated or enhanced. For starters, if the attack path is blocked (or slowed down), the attack impact can be reduced. In many cases, the direct effect of an AI attack is not the ultimate attack goal. Usually there exist one or more impact propagation paths that lead the AI attack to the ultimate attack goal. If such paths are blocked, the ultimate attack goal cannot be achieved. Taking the data poisoning attack as an example, the direct effect of the attack is contaminated corpus. Strong access control can be enforced to prevent the contamination, and, through security-aware active learning, a good portion of the corrupted data samples may get excluded from the labeled training set on Hq. In this way, the attack impact on the trained model will be significantly reduced.

Instead of blocking (or slowing down) the impact propagation, attacks can be prevented by ensuring that no propagation path is formed in the first place. Taking the data poisoning attack as an example again, in some close-loop deployment environments (e.g. a factory) of an ML system, it is actually feasible to certify all the data providers. Therefore, the attack impact can be greatly reduced at host Hp.

Despite the passive strategies above, the attack impact can be substantially reduced by taking a (proactive) quarantine and isolation strategy. For example, as shown in Subfigure 2b and 2c, different hosts are used for different tasks. Such isolation can reduce the attack impact should one of the tasks is tainted. Also, before the collected the data is used for model training, the developers can check the validity of the data through semantics or outlier detection on Hp and/or Hq.

V. Related Work

A. Causality Modeling for Enterprise Network Security Analysis

From the perspective of modeling, the following works are most relevant to this paper: attack surface [15], attack graph [3, 12, 22] and system object dependencies [23].

Attack surface focuses on how the attackers may gain initial access to the victim system. It focuses on modeling how the system interacts with the outside environment, like file I/O, function call, and network communications. Attack graphs, on the other hand, can generate possible attack paths by analyzing the causal relationships between security vulnerabilities (such as CVEs existing in the network) and exploits, illustrating how the attackers can hop from the entry point to other hosts in the system, and finally their ultimate goal. However, attack graph analysis is not adequate in analyzing ML systems due to the following reasons. First, attack graph analysis assumes that an attack path usually has several stages, and each stage requires the adversary to take a particular attack action. Attack graph analysis tools focus on analyzing vulnerabilities. However, ML security analysis tools should focus on analyzing dependencies. Second, MulVAL [17], a widely-used attack graph analysis tool, assumes that each security reasoning rule usually matches multiple kinds of vulnerabilities in real enterprise networks. This is because of the existence of widely-accepted grouping criteria for software vulnerabilities, like Common Vulnerability Scoring System (CVSS) [1]. Whenever new vulnerabilities show up, it is very likely that they will fall into one or more known categories of previously found vulnerabilities. Based on this assumption, new CVEs usually will not require MulVAL to add new rules. However, in the context of ML system security analysis, we find that each security reasoning rule usually only matches one particular type of adversarial attack due to the lack of such widely-accepted grouping criteria.

B. ML System-specific Attacks

The NIST Artificial Intelligence Report NIST.AI.100-2e2023 [24] provides a comprehensive taxonomy of attacks against ML systems. Based on the taxonomy, the attacks against Predictive AI systems are classified into Evasion Attacks, Poisoning Attacks, and Privacy Attacks. In addition, the attacks against Generative AI systems are classified into AI Supply Chain Attacks, Direct Prompt Injection Attacks, and Indirect Prompt Injection Attacks.

In an evasion attack, the adversary's goal is to generate adversarial examples which trigger the ML model to misclassify or generate other kinds of incorrect output. In particular, attackers craft input samples that cause the model to produce incorrect or misleading outputs during inference, despite appearing innocuous to human observers. Evasion attacks [5, 19, 26, 28] exploit the inherent vulnerabilities of ML models and can compromise the reliability and trustworthiness of the system.

Poisoning attacks are usually launched to achieve one of four attack objectives: availability poisoning, targeted poisoning [4, 11, 21], backdoor poisoning, or model poisoning [8, 9, 20, 25]. Except for availability poisoning, all poisoning attacks aim to compromise the correctness of the output(s) of an ML model.

Poisoning attacks can have severe consequences, particularly in safety-critical applications or decision making systems, as they can lead to erroneous decisions, financial losses, or even physical harm.

Privacy attacks are usually launched to achieve one of four major attack objectives: data reconstruction, membership inference, model extraction, or property inference. Privacy attacks may reconstruct sensitive information about the training data from the model parameters or outputs [7, 27]. Privacy attacks can be carried out by analyzing the model's behavior, outputs, or gradients, and can be particularly effective in scenarios where the model memorizes or overfits to the training data. Membership inference attacks attempt to determine whether a specific data point was present in the training data used to build the model, potentially revealing sensitive information about individuals [6, 10].

C. Analysis of ML System-Specific Attacks

Besides the report [24] documenting concepts and terminologies in the field of adversarial machine learning, the AI risk management framework [2] is also published to provide guidance for managing risks. However, most works in this area focus on individual attacks or qualitative systematic analysis.

Individual adversarial attacks: Such works focus on a particular ML attack and/or a particular component of an ML system. We have already enumerated many such attacks earlier, so we will not discuss such work in detail here. This kind of work shares the same limitation. That is, isolated security analysis provides limited understanding about how the adversary could exploit the dependencies between different components of an ML system. In many cases, simply adding up (e.g. weighted sum) all the individual security (and robustness) scores does not automatically result in very meaningful ML-system-level security analysis. Therefore, another approach at the ML system level is necessary. There are also works that evaluates an ML model's security against a whole class of attacks, like ML-Doctor [14] which focuses on inference attacks and DEEPSEC [13] which focuses on evasion attacks. However, security analysis at the whole-system level is still lacking discussions.

Qualitative ML system security analysis: Such works [16, 18] try to answer the question of how to synthesize the aforementioned individual security analysis results in a meaningful way. They first model the whole ML system, enumerate all possible attack surfaces and impacts, and then try to provide prevention/mitigation suggestions. However, being "qualitative" is not enough. We believe ML system security should be quantified in a meaningful way.

VI. Conclusion

Since the existing security causality analysis frameworks and techniques were mainly developed to analyze traditional security issues in enterprise (software) systems and networks, they are no longer very suitable for analyzing ML systems. ML systems have new causality relationships which are not sufficiently modeled by those approaches. Also, existing notions and elements in the existing modeling frameworks fall short in the case of ML systems. In light of this, we take the initial step by presenting a new modeling approach to analyze security issues about output correctness. We present a new way to model causality relationships in ML systems for security analysis of output correctness with a running example, and then analyze a real-world ML system with the proposed model. We then discuss how our model can be used for security analysis. Some future directions include: 1) Extending the model for analyzing other types of ML attacks, like privacy-centered attacks including model stealing and membership inference attacks; 2) Analyzing the cross-effect of different types of ML attacks; 3) Developing quantitative security measurements, etc.

Disclaimer

Commercial products are identified in order to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the identified products are necessarily the best available for the purpose.

References

[1] Common Vulnerability Scoring System SIG (Nov 2023), https://www.first.org/cvss, [Online; accessed 31. May 2024]

[2] AI, N.: Artificial intelligence risk management framework (ai rmf 1.0) (2023)

[3] Cao, C., Yuan, L.P., Singhal, A., Liu, P., Sun, X., Zhu, S.: Assessing attack impact on business processes by interconnecting attack graphs and entity dependency graphs. In: Data and Applications Security and Privacy XXXII: 32nd Annual IFIP WG 11.3 Conference, DBSec 2018, Bergamo, Italy, July 16–18, 2018, Proceedings 32. pp. 330–348. Springer (2018)

[4] Chen, X., Liu, C., Li, B., Lu, K., Song, D.: Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526 (2017)

[5] Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)

[6] Hayes, J., Melis, L., Danezis, G., De Cristofaro, E.: Logan: Membership inference attacks against generative models. arXiv preprint arXiv:1705.07663 (2017)

[7] He, Z., Zhang, T., Lee, R.B.: Model inversion attacks against collaborative inference. In: Proceedings of the 35th Annual Computer Security Applications Conference. pp. 148–162 (2019)

[8] He, Z., Rakin, A.S., Li, J., Chakrabarti, C., Fan, D.: Defending and harnessing the bit-flip based adversarial weight attack. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14095–14103 (2020)

[9] Hector, K., Moëllic, P.A., Dumont, M., Dutertre, J.M.: A closer look at evaluating the bit-flip attack against deep neural networks. In: 2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS). pp. 1–5. IEEE (2022)

[10] Hu, H., Salcic, Z., Sun, L., Dobbie, G., Yu, P.S., Zhang, X.: Membership inference attacks on machine learning: A survey. ACM Computing Surveys (CSUR) 54(11s), 1–37 (2022)

[11] Jagielski, M., Severi, G., Pousette Harger, N., Oprea, A.: Subpopulation data poisoning attacks. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 3104–3122 (2021)

[12] Jha, S., Sheyner, O., Wing, J.: Two formal analyses of attack graphs. In: Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15. pp. 49–63. IEEE (2002)

[13] Ling, X., Ji, S., Zou, J., Wang, J., Wu, C., Li, B., Wang, T.: Deepsec: A uniform platform for security analysis of deep learning model. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 673–690. IEEE (2019)

[14] Liu, Y., Wen, R., He, X., Salem, A., Zhang, Z., Backes, M., De Cristofaro, E., Fritz, M., Zhang, Y.: {ML-Doctor}: Holistic risk assessment of inference attacks against machine learning models. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 4525–4542 (2022)

[15] Manadhata, P.K., Wing, J.M.: An attack surface metric. IEEE Transactions on Software Engineering 37(3), 371–386 (2010)

[16] McGraw, G., Figueroa, H., Shepardson, V., Bonett, R.: An architectural risk analysis of machine learning systems: Toward more secure machine learning. Berryville Institute of Machine Learning, Clarke County, VA. Accessed on: Mar 23, 2020 (2020)

[17] Ou, X., Govindavajhala, S., Appel, A.W., et al.: Mulval: A logic-based network security analyzer. In: USENIX security symposium. vol. 8, pp. 113–128. Baltimore, MD (2005)

[18] Papernot, N.: A marauder's map of security and privacy in machine learning. arXiv preprint arXiv:1811.01134 (2018)

[19] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia conference on computer and communications security. pp. 506– 519 (2017)

[20] Rakin, A.S., He, Z., Fan, D.: Bit-flip attack: Crushing neural network with progressive bit search. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1211–1220 (2019)

[21] Schwarzschild, A., Goldblum, M., Gupta, A., Dickerson, J.P., Goldstein, T.: Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. In: International Conference on Machine Learning. pp. 9389–9398. PMLR (2021)

[22] Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: Proceedings 2002 IEEE Symposium on Security and Privacy. pp. 273–284. IEEE (2002)

[23] Sun, X., Dai, J., Liu, P., Singhal, A., Yen, J.: Using bayesian networks for probabilistic identification of zero-day attack paths. IEEE Transactions on Information Forensics and Security 13(10), 2506–2521 (2018)

[24] Vassilev, A., Oprea, A., Fordyce, A., Anderson, H.: Adversarial machine learning: A taxonomy and terminology of attacks and mitigations (2024), https://doi. org/10.6028/NIST.AI.100-2e2023, (National Institute of Standards and Technology, Gaithersburg, MD) NIST Artificial Intelligence (AI) Report, NIST Trustworthy and Responsible AI NIST AI 100-2e2023

[25] Wang, J., Zhang, Z., Wang, M., Qiu, H., Zhang, T., Li, Q., Li, Z., Wei, T., Zhang, C.: Aegis: Mitigating targeted bit-flip attacks against deep neural networks. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 2329–2346 (2023)

[26] Zhang, L., Liu, P., Choi, Y.H., Chen, P.: Semantics-preserving reinforcement learning attack against graph neural networks for malware detection. IEEE Transactions on Dependable and Secure Computing 20(2), 1390–1402 (2022)

[27] Zhang, Y., Jia, R., Pei, H., Wang, W., Li, B., Song, D.: The secret revealer: Generative model-inversion attacks against deep neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 253–261 (2020)

[28] Zou, Q., Zhang, L., Singhal, A., Sun, X., Liu, P.: Analysis of neural network detectors for network attacks. Journal of Computer Security (Preprint), 1–28 (2023)