How to Secure Machine Learning in Security Systems

AI under Attack

kaspersky bring on the future

Introduction

During the mid-2010s, Artificial Intelligence (AI) and its key component Machine Learning (ML) were the hot topics in information security. These technologies were set to deliver on the over-hyped expectations coming out of 'Big Data'; we'd learned to collect a lot of numbers, and we needed ways to extract all the good stuff they contained. Vendors of so-called Next-Gen security solutions played this up in a big way — 'legacy antivirus' was now dead, killed by the mighty new 'AI antivirus' approach (though actual test results hardly bore this thesis out).

Learn more

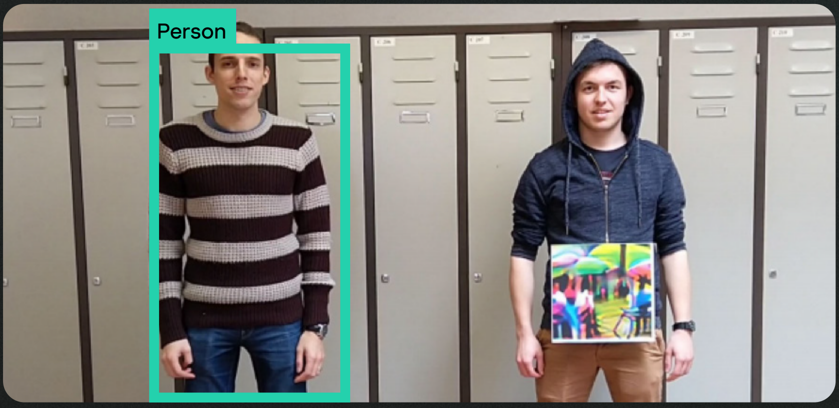
According to Cisco[1]

the percentage of information security officers enthusing about machine learning has cooled from 77% in 2018 to 67% in 2019, while interest in artificial intelligence has also dropped from 74% to 66%

Learn more

Since then, AI fervor has gone somewhat off the boil. According to Cisco[1], the percentage of information security officers enthusing about machine learning has cooled from 77% in 2018 to 67% in 2019, while interest in artificial intelligence has also dropped from 74% to 66%. Even the Gartner analysts who initially praised Next-Gens have become more selective in 2019, stating that 'artificial intelligence' does not automatically mean a 'better product' in today's security.

One of the main reasons for this cooling off (and we're certainly not talking an 'AI winter' here — just an adjustment of unrealistic expectations) is that ML algorithms, once released from the confines of the lab and introduced into the real world, have turned out to be pretty fallible in terms of detection performance. On top of which, they're vulnerable to potential attacks designed to force them into making deliberate errors.



A person detector fooled by an adversarial patch. The illustration from KU Leuven research[2]

All this is bad news for the security industry. Suddenly, at security conferences over the last couple of years, we're talking not just 'AI', but 'Adversary AI'. We now have to face up to ML hacks. Some of the most spectacular examples of how to baffle ML based models are the simplest — a fragment of paper glued to a road sign means it's identified as a completely different sign, facial recognition software is easily tricked by a pair of paper glasses, and just holding a picture can render you unidentifiable as a human being.

[1] The CISO Benchmark Report 2019
[2] Simen Thys, Wiebe Van Ranst, Toon Goedeme. Fooling automated surveillance cameras: adversarial patches to attack persondetection. — arXiv, 2019

Okay, so we're not purely reliant on automatic image recognition for identity checks right now. But mistakes in ML-based recognition have already led to innocent people being branded as criminals. In November 2018, the Shanghai police system accused a famous Chinese businesswoman of 'jaywalking', after a security camera captured her face on an advertisement on the side of a bus. At around the same time, in New York, a student filed a $1 billion lawsuit against Apple after the tech giant's facial-recognition software apparently wrongly identified him as someone stealing from their stores. In May this year, the San Francisco authorities banned the use of facial recognition software by police and other city agencies, to prevent similar mistakes and abuses.

Malware detection worldwide uses ML learning methods very similar to those used in facial recognition systems. And the impact of attacks on ML-based anti-malware systems could be devastating: a mis-identified Trojan means millions of devices infected and millions of dollars lost.

The good news is that, by meeting specific conditions and by protecting ML systems appropriately, these threats can be averted. In this paper, we present an overview of popular attacks on ML algorithms in information security, and discuss methods to protect ML solutions from these threats.

How ML works

Machine learning (ML), a subset of Artificial Intelligence (AI), is often described as a set of methods and technologies that give computers the ability "to learn without being explicitly programmed." In other words, an ML algorithm is a program that can itself build programs for solving different problems. To do this, an ML algorithm can either learn from a set of already solved cases (known as supervised learning), or it can find previously unknown similarities and correlations in a given dataset (unsupervised learning).

In supervised learning, ML works either in training mode or in battle mode. In training mode, the ML algorithm is given a training dataset of objects represented by their features, and their labels. If we're talking malware detection, the objects could be files, their features might be different file meta-data or behaviors (file statistics, list of used API functions, etc.) and their labels could simply be 'malware' or 'benign'.

Based on this training dataset of known malware and known benign objects, the ML algorithm must create a predictive model which should then correctly label (as 'malware' or 'benign') previously unseen objects (new files). After this training, the ML model moves into battle mode and is used for detection. In unsupervised learning, we're interested in revealing hidden patterns and clusters in the data being examined by the algorithm — groups of similar objects, highly correlated features or events. So the data given to ML algorithm for training is not labelled, and the ML algorithm works out the correlations by itself.

In the cybersecurity industry, unsupervised learning is of particular value for behavioral analysis and anomaly detection.

Forms of attacks and how to defend against them

Forms of attack	Defense
Label poisoning	Double-check and ensemble working
Training dataset poisoning	Protect your datasets and discern 'strangers'
White box / black box adversarial attack	• Use cloud ML models • ML-created detection records • Provably robust ML model
Attacks on pretrained and outsourced ML models	Don't trust third parties
Data leaks via trained models	Reduce access, anonymize data
Hardware based attacks	Methods independent of local architecture

Label poisoning

For this type of attack on ML algorithms, the hacker needs to be able to access the training dataset, so he could add incorrectly labelled objects. Trained on this incorrect data, the ML model will make detection errors when faced with similar objects.

But getting access to this supervised dataset must be pretty difficult? In fact, no. Many vendors exchange threat data through threat intelligence feeds, and there are known examples of where that data has been tampered with. For example, data from threat intelligence aggregators like VirusTotal has been poisoned by specially crafted clear files with inserted malware features. After one antivirus scanner erroneously labels one of these clear files as malicious, this incorrect data may be passed on to other security solutions, causing a chain reaction of 'false positive' identifications (detecting similar clear files as malicious) worldwide.

Defense:

Double-check and ensemble working

All labelled files received by Kaspersky from third-party feeds are double-checked with our own databases to ensure they're correctly classified. Mistakes in ML-based classification are also reduced by 'ensembling' — enabling different ML models to work in harness, and in combination with human expert analysis.

Training dataset poisoning

With access, an attacker can also poison a dataset by adding special objects that degrade the performance of the prediction model. In this type of attack, the labels are correct (or, in the case of unsupervised learning, the dataset is not labelled) but the added objects themselves are strange: for example, a file that's very different from those commonly used (a 'black swan'). This form of threat is particularly insidious, as many ML developers, including some Next-Gen vendors, use public datasets that can easily be poisoned by third parties.

Defense:

Protect your datasets and discern 'strangers'

If an attacker doesn't know what samples you used to train your model, it's harder to create out-of-distribution objects. Kaspersky's training datasets are in general collected by ourselves, and they are not public: only some specific types of analyzed malware are exported via external feeds. The logs for our behavioral ML models are based on a unique internal mechanism that can't be accessed by outsiders. We've also developed a method to estimate the 'level of trust' for a prediction made about a particular object. This gives our behavioral ML-based systems the ability to discern 'strange' files and reject their classification, so they won't break the model. And, last but not least, multi-layered protection is a good defense against this type of attack. Even if ML-based static analysis is fooled, the malware still would be detected by dynamic analysis (using an emulator or sandbox).

White box / black box adversarial attack

An attacker who doesn't have access to training datasets can still interact with an ML model. A determined hacker with full access to the model itself, perhaps in the local client product, can study it privately as long as it takes, and may then be able to reverse-engineer the code. Through this, the attacker can learn the model architecture, establish what file features the model uses, and then create malware that circumvents these features — this is known as a 'white box' or 'model stealing' attack.

One example of such an attack is the Cylance AI bypass uncovered in July 2019. Analyzing the code of the endpoint protection product, researchers found the name of a popular online game in the 'whitelisting' mechanism. They added strings from this game's main executable into a malicious file — and this modified malware then became immune from detection by the Cylance product.

If the source code is unavailable, an attacker can 'brute-force' the ML model by repeatedly making small changes in the malware created, and testing the results against the ML model until a weak point is discovered. This 'black box' form attack is labor-intensive, so may be highly automated, using specially developed 'adversarial AI' to generate the attacking samples.

Defense:
Use cloud ML models

Cloud based models mean an intruder can't play with the model locally. Take the example of our ML threat detection for Android. Here, an agent on the user's device collects the features of a new application, this meta-data is then sent to a powerful cloud ML model trained on millions of samples, and the decision is immediately sent back to the mobile device.

Of course, technically, an attacker could still try to brute-force the cloud ML model. But a black box attack like this takes a lot of time and would be easily detected.

Defense:
ML-created detection records

Both cloud and in-lab ML models can create discrete detection records to be added to the product's database. If the attacker manages to reverse-engineer a detection record, he could only fool that particular record. He can't break the ML model, which automatically generates a new record with which to detect his modified threat.

Here's an example. In Kaspersky's Similarity Hash Detection System, in-lab ML is used to find the features common to a whole group of similar malicious files. Based on these features, Similarity Hashes (SHs) are created and sent as detection records to local products via the Kaspersky Security Network cloud. An endpoint product calculates the specific SH for an examined file locally, then compares this with the SH databases.

This approach allows our products to detect whole families of quickly changing polymorphic malware — with no risk of the ML model being hacked.

Defense:
Provably robust ML model

Another way to protect your ML system from adversarial attacks is to build an ML model that won't be broken by changes in adversarial samples. For our behavioral ML system, we developed a concept of monotonic classification models that ensures predictions are consistent over execution time and are provably stable in the face of injections of any noise or `benign-looking' activity into the program's behavior.

The predictions of such models change monotonically through the execution log, in the sense that the addition of new lines into the log may only increase the probability of the file being found malicious, which make these ML models suitable for real-time classification on an endpoint.

Attacks on pretrained and outsourced ML models

Lack of resources means that some developers make use of third-party ML architectures, which have been created for standard data processing tasks. These off-the-shelf ML solutions may already be well-known to malefactors, making it easy to organize a white box attack.

At a recent security conference, a well-known vendor proudly announced "We publish all our ML models on GitHub". Asked about the risk of white box attacks, a speaker said that the average hacker couldn't do this because such attacks would require highly sophisticated skills. It was a very bold answer when, literally just next door, the results of ML hacking research were being presented.

Another threat vector resulting from reliance on third parties is that of ML models trained by external groups of data scientists or by public ML services — these models can contain backdoors.

Defense:

Don't trust third parties

We train our models ourselves, on our own hardware. Actually, there's an additional reason for this in-house approach: interpretability. ML model development is a complex process dealing with a very intricate architecture of thousands of nodes and weightings. If a developer wants to verify the correctness of the system's decisions, understand the reason for possible mistakes and be able to investigate the intrusion, he or she must know how to interpret the ML model's results. That's something we do regularly with our ML models — analyze their interpretability. With outsourced ML models, this would be much harder (and perhaps impossible) to do.

Data leaks via trained models

In some cases, an intruder could feed the ML model with specially selected samples in order to gain information about objects used in the training dataset. This could be a threat if the objects contain sensitive information (like personal medical records) or if the very fact that the object is being presented in the dataset is sensitive (for example, criminals may find out that their pictures are being used to train a police facial recognition system).

Defense:

Reduce access, anonymize data

Just as in white box attacks, an intruder usually needs full access to the ML model in order to play with it until sensitive information is obtained. One way to avoid this is to use multilayered cloud ML models instead of models in client products. Another good idea is to anonymize the data used in training. Some vendors offer ML models that work specifically with encrypted data.

Hardware based attacks

Some heavy ML methods require a lot of computation, and the results may vary on different processors. For example, a model is trained in-lab on a powerful computer, but then has to work on a client's phone. An attacker can create a special file that is misclassified on some phone models, which in turn could lead to the incorrect labelling of this file in other detection systems (similar to a data poisoning attack).

Defense:

Methods independent of local architecture

This threat is just another reason not to have ML models on client products, like phones. We ourselves use cloud ML or discrete detection records created by our in-lab ML systems (see above). It's also worth mentioning that some ML methods (e.g. decision trees) are less dependent on hardware differences than others (e.g. neural networks).

Conclusion

For all the over-hyping, AI and ML play a valuable role in information security. Here at Kaspersky, we began using ML-based algorithms long before the Next-Gen buzz, and these algorithms are utilized in many stages of our detection pipeline, including clustering methods to pre-process incoming file streams in-lab, deep learning models for cloud detection, and ML-created records in product databases. However, our studies reveal that ML algorithms could be vulnerable to many forms of attack. Some key considerations should be applied to ML use in security systems:

The security vendor should understand and carefully address

essential requirements for ML performance in the real, potentially hostile, world — requirements that include extremely low false positive rates, a robustness to potential adversaries, and the interpretability of ML models. ML/AI-specific security audits and 'red-teaming' should be a key component of ML/AI development

In assessing the security of an ML solution, questions

should be asked about how much the solution depends on third party data and architectures, as so many attacks are based on third party input (we're talking threat intelligence feeds, public datasets, pre-trained and outsourced ML models)

ML methods should not be viewed as 'the ultimate answer'.

They need be a part of multi-layered security approach, where complementary protection technologies and human expertise work together, watching one other's backs.

Further information

Kaspersky TechoWiki is a great source of information and thought-leadership on advanced security technologies including AI, ML and behavior-based protection.

Learn more

Securelist.com provides the most recent and detailed data on modern malware,targeted attacks and other cyber-criminal trends across the world.

Learn more

www.kaspersky.com

#kaspersky
#bringonthefuture