DEEPARMOR®
by SPARKCOGNITION

How Patented Machine Learning Techniques Predict and Prevent Future Cyber Attacks

With the overabundance of cybersecurity solutions promising "cutting-edge AI protection," organizations can struggle to find out if they live up to their promises. Solutions that advertise AI must be evaluated on how they work, how they use AI, and what value AI is truly adding.

In roughly half an hour, any company could select a few features, create a basic machine learning model for detecting cyber threats, and claim AI capabilities. But will the model in question protect systems with greater accuracy? Just because a vendor has added machine learning capabilities to their product doesn't mean they've added any value.

This paper will first go over the most important components to a machine learning cybersecurity solution. Then it will discuss SparkCognition's cybersecurity solution, DeepArmor®. SparkCognition™ is an artificial intelligence company first, with a seasoned research team dedicated to advancing the science of AI in cybersecurity. DeepArmor takes advantage of this expertise by employing AI at every stage of operation.

Basic Approaches to Malware Detection

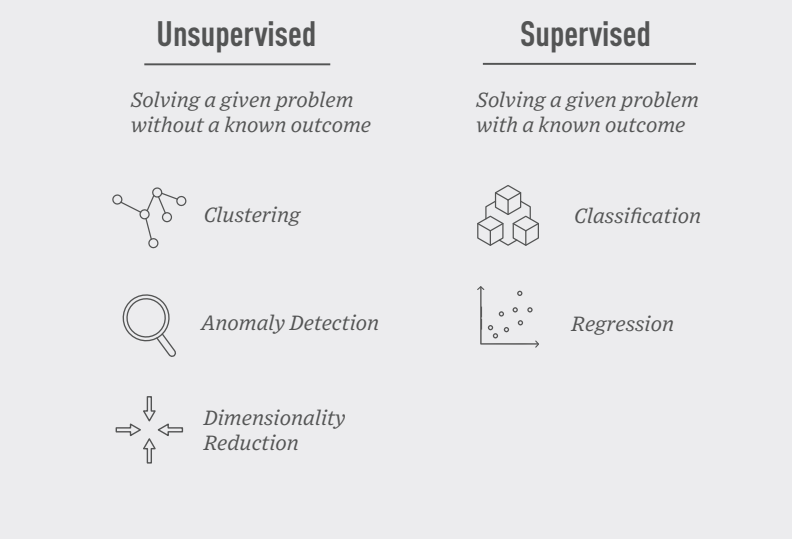
An efficient, robust, and scalable malware recognition engine is the key component of every effective cybersecurity platform. There are multiple approaches to malware detection, including heuristics, file reputation, sandboxing, signatures, and machine learning, but only machine learning can keep up with the current cybersecurity landscape.

Early in the cyber era, the number of malware threats was relatively low, and simple handcrafted pre-execution rules were often enough to detect threats. But in the last decade, the tremendous growth of the malware stream meant anti-malware solutions could no longer rely solely on the time-consuming, expensive manual creation of detection rules. It was natural for anti-malware companies to start augmenting their malware detection and classification with machine learning, a computer science area that has shown success in image recognition, searching, and decision-making. Today, machine learning augments malware detection using various kinds of data on host, network, and cloud-based anti-malware components.

Machine Learning: Concepts and Definitions

According to the classic definition given by AI pioneer Arthur Samuel, machine learning is a set of methods that allows computers to learn beyond their explicit programming.

In other words, a machine learning algorithm discovers and formalizes the principles that underlie the data it sees. With this knowledge, the algorithm can reason through the properties of previously unseen samples. In malware detection, a previously unseen sample could be a new file, and its hidden property could be whether it is malware or benign. A mathematically formalized set of principles underlying data properties is called the model. There are two primary approaches used by machine learning, depending on the specifics of the task at hand: unsupervised learning and supervised learning.



Unsupervised Learning

One machine learning approach is unsupervised learning. In this setting, there is only a data set without the right answers for the task. The goal is to discover the structure of the data or the law of data generation.

One important example is clustering, a task that includes splitting a data set into groups of similar objects. Another task is representation learning, which involves building an informative feature set for objects based on their low-level description.

Large unlabeled datasets are available to cybersecurity vendors but the cost of their manual labeling by experts is high. This makes unsupervised learning valuable for threat detection. Clustering can help with optimizing efforts for the manual labeling of new samples. Specifically, it allows us to decrease the number of labeled objects needed for the usage of the next machine learning approach in the arsenal: supervised learning.

Supervised Learning

Supervised learning is a setting used when both the data and the right answers for each object are available. The goal is to fit the model that will produce the right answers for new objects. Supervised learning consists of two stages:

- *Training a model and fitting it to available training data*
- *Using the trained model to generate predictions by applying it to new samples*

the task:

- *For a given a set of objects*
- *Each object is represented with feature set X*
- *Each object is mapped to right answer or labeled as Y*

This information is utilized during the training phase, when searching for the best model that will produce the correct label Y' for previously unseen objects given the feature set X'.

In the case of malware detection, X could be some features of file content or behavior, such as file statistics and a list of used API functions. Labels Y could be "malware" or "benign," or even a more fine-grained classification, such as a virus, Trojan-Downloader, or adware. In the "training" phase, some family of models, like neural networks or decision trees, must be selected.

Usually each model in a family is determined by its parameters. Training means that searching for the model from the selected family with a particular set of parameters that gives the most accurate answers for train objects according to some metric. In other words, the model "learns" the optimal parameters that define valid mapping from X to Y.

After training a model and verifying its quality, it's time for the next phase—applying the model to new objects. In this phase, the type of the model and its parameters do not change. The model only produces predictions. In the case of malware detection, this is the protection phase. Vendors often deliver a trained model to users where the product makes decisions based on model predictions autonomously. Mistakes can cause devastating consequences for a user—for example, removing an OS driver. It is crucial for the vendor to select a model family properly. The vendor must use an efficient training procedure to find a model with a high detection rate and a low false positive rate.

Machine Learning Application Specifics

User products that implement machine learning make decisions autonomously. The quality of the machine learning model impacts the user system performance and its state. Because of this, machine learning-based malware detection has specific requirements.

Large representative datasets are required - It is important to emphasize the data-driven nature of this approach. A created model depends heavily on the data it has seen during the training phase to determine which features are statistically relevant for predicting the correct label. Imagine we collect a training set, but overlook the fact that in our data set all files larger than 10 MB are malware, which is certainly not true for real-world files. While training, the model will exploit this property of the dataset, and will learn that any files larger than 10 MB are malware. It will use this property for detection.

When this model is applied to real world data, it will produce many false positives. To prevent this outcome, benign files with larger sizes need to be added to the training set. Generalizing this, models must be trained on a data set that correctly represents the conditions where the model will be working in the real world. This makes the task of collecting a representative data set crucial for machine learning to be successful.

The trained model has to be interpretable - Most of the model families used nowadays, like deep neural networks, are called black box models. Black box models are given the input X, and they will produce Y through a complex sequence of operations that can't really be interpreted by a human. This could pose a problem in real-life applications. For example, when a false alarm occurs, the question is: Was it some problem with a training set or the model itself? The interpretability of a model determines how easy it will be for us to manage it, assess its quality, and correct its operation.

False positive rates must be extremely low - False positives happen when an algorithm mistakes a malicious label for a benign file. SparkCognition's aim is to make the false positive rate zero. This

is atypical for a machine learning application. But even one false positive in a thousand benign files can have serious consequences for users. To address this problem, strict requirements must be imposed for both machine learning models and metrics that will be optimized during training, with a clear focus on low false positive rate (FPR) models.

This is still not enough, because new benign files that go unseen earlier may occasionally be falsely detected. SparkCognition takes this into account and implements a flexible model design that allows false positives to be fixed on the fly, without completely retraining the model. Examples of this are implemented in the pre and post-execution models, which are described in following sections.

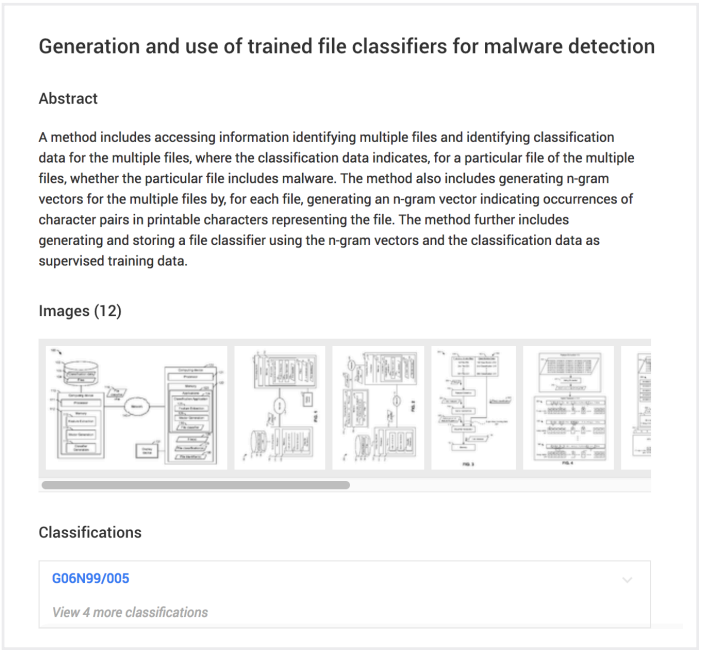
Algorithms must allow us to quickly adapt them to malware writers' counteractions - Outside the malware detection domain, machine learning algorithms regularly work under the assumption of fixed data distribution. But after applying machine learning to malware detection, data distribution is still imperfect, as active adversaries constantly release new varieties of malware and software companies release new types of benign executables.

To overcome this issue, the cybersecurity solution architecture needs to be flexible and allow model updates between retrainings. Vendors must also have effective processes for collecting and labeling new samples, enriching training data sets, and regularly retraining models.

Our Approach to ML Malware Detection

Attack Vector Identification and Sampling

DeepArmor's models are crafted with advanced data science methods, as SparkCognition's dedication to research allows it to continually pioneer new techniques, many of which will be described below. SparkCognition holds a number of patents for these new techniques, which are what allow DeepArmor to detect and block cyber attacks with an efficacy and accuracy far beyond what more traditional solutions can manage. This includes patents on training a file classifier on n-gram feature vectors; identifying classification data for multiple files; and generating a sequence of entropy indicators for each of multiple files. The efficacy of these patented techniques can be seen in comparing DeepArmor's overall effectiveness against zero-day attacks versus other leading next-generation anti-virus solutions.


Generation and use of trained file classifiers for malware detection

Abstract

A method includes accessing information identifying multiple files and identifying classification data for the multiple files, where the classification data indicates, for a particular file of the multiple files, whether the particular file includes malware. The method also includes generating n-gram vectors for the multiple files by, for each file, generating an n-gram vector indicating occurrences of character pairs in printable characters representing the file. The method further includes generating and storing a file classifier using the n-gram vectors and the classification data as supervised training data.

Images (12)

Classifications

G06N99/005

View 4 more classifications

DeepArmor's process begins by researching all possible attack vectors for a given file type, be it an Office file or an executable. Once the attack vectors have been identified, the DeepArmor team gathers a comprehensive, representative set of clean and malicious samples.

Feature Extraction and File Features

The next step is the creation of a feature extractor for the given file type. This is an area where DeepArmor's approach diverges widely from what is standard. Using subject matter experts to hand-select features introduces human bias into the model, relying on preconceptions about what malware should look like in a time when malware is diversifying at an unprecedented rate.

The DeepArmor team extracts as much data as possible from each file, be it embedded strings, entropy levels, API calls, system calls, or leveraged DLLs, and puts everything through a machine learning model. The model examines the data set and learns which features are the most statistically predictive of a cyber attack.

Where most solutions may be constructed using around 500 to 1,000 features, DeepArmor's process results in 23,000 features. This lends DeepArmor a predictive power beyond what most solutions can accomplish.

- *RosAsm Base3.exe PE File Structure*
- *Dos MZ Header*
- *DOS Stub*
- *PE File Header*
- *PE Signature*
- *Image_Option_Header*
- *Section Table*
  Array of Image_Section_Headers
- *Data Directories*
- *PE Imports*
- ***Sections:** .idata, .rsrc, .data, .text, .src*

CodeSize: 104448; SubsystemVersion: 5.0; InitializedDataSize: 568320; ImageVersion: 0.0 FileSubtype: 0; FileVersionNumber: 0.6.1.8 LanguageCode: Neutral; FileFlagsMask: 0x003f

KERNEL32.dll: CreateToolhelp32Snapshot GetLastError

A file can be inspected to explicitly look for the strings it contains, such as code fragments, author signatures, names, and system resource information. These types of strings have been shown to provide valuable information for the malware analysis process. Once strings are extracted, it is possible to gather information on factors such as number and presence of specific strings, which can unveil key cues to gain additional knowledge on a file. String analysis techniques like n-grams can be used to develop features from string data without domain knowledge or analyst bias. In addition, SparkCognition has developed patented approaches to analyzing string data that are leveraged in DeepArmor, which can be seen in U.S. Patent 9,864,956 B1 on training a file classifier on n-gram feature vectors.

Strings - Most solutions search for known malicious or clean keywords in a file—say, "abcd," a command to activate a payload. DeepArmor instead uses a patented approach to locate these strings

using only two characters at a time—say, "ab," then "a_c." This produces similar accuracy to the traditional approach, but uses far less CPU and takes less time.

Entropy - DeepArmor also takes a groundbreaking approach to analyzing entropy levels in a file. Entropy refers to the number of repeat characters in a string of text. In normal English, entropy is fairly low, but the randomized strings in encrypted files have far higher entropy. Rather than rating a file as having high or low entropy overall, DeepArmor looks at the sections of code within the file. This prevents files that are large but contain only a short section of encrypted code from slipping through DeepArmor's detection engine.

API/system calls - API calls are the function calls used by a program to execute specific functionality. System API calls available through standard system DLLs have to be distinguished from user API calls provided by user-installed software. These are designed to perform a pre-defined task during invocation. DeepArmor uses sophisticated techniques to extract potential red flags such as suspicious API calls, anti-VM, and anti-debugger hooks and calls.

File headers - File headers represents a collection of metadata related to a portable executable file. Basic features that can be extracted from a PE32 header, for example, are size of header, size uninitialized data, and size of stack reserve, which may indicate if a binary file is malicious or benign. DeepArmor analyzes structural information for describing malicious and benign files, including raw header characteristics and section characteristics to detect unknown malware in a semi-supervised approach.

Functions - Functions embedded in a file can contain a number of identifying features, including function calls and function length, or the number of bytes contained within a function. By combining this information with other static and dynamic features, DeepArmor is able to use this feature as another indicator of a potentially malicious file.

Training

Once features have been extracted and converted to numerical values, models are iteratively trained over generations on millions of samples of malicious and benign files. Training data is carefully constructed to remain representative at each generation, without simply throwing any false positives back into the mix for subsequent generation (an approach that skews the data).

Model Development

To analyze files, DeepArmor employs a number of cutting-edge techniques, including gradient boosting and deep learning neural networks. Gradient boosting is a technique best suited for regression and classification problems. It ensembles together a set of weaker decision tree models to create a single, highly accurate model with minimized bias and variance. This technique is a powerful tool for handling large training sets with correspondingly vast sets of features. The models it creates are both effective and small, and generate predictions quickly.

Gradient boosting combines multiple decision trees to create a single, highly accurate model. Current gradient boosting techniques are well-suited for malware detection because, among other reasons, they deal well with missing values. Folder sizes, heap allocations, and similar values account for many features in DeepArmor's feature set, and having different values for an empty folder and a nonexistent folder greatly help the efficacy of DeepArmor's algorithm. Also, gradient boosting works well with wide feature sets, because the algorithm can essentially discard columns that are unnecessary. DeepArmor's model training sets are curated to tackle individual file formats. Files that are not malicious enough, unknown, corrupt, or otherwise unable to have their maliciousness validated are discarded. In addition, DeepArmor's training sets include known false positives and false negatives, although these are added in a proportion that doesn't overwhelm the model.DeepArmor's data science team uses two different tests for efficacy: holdout and out-of-band. Holdout tests involve holding out 20% of files to test on; these determine backward efficacy of models, or how well models do on previously seen data. Out-of-band tests are performed on either recent malware or files from a diverse source; these tests attempt to measure recall on more unexpected data. Although testing can always be improved, these two provide a snapshot of a model's effectiveness.

Additionally, the results DeepArmor produces are not a binary rating of a file as "malicious" or "benign." Instead, DeepArmor evaluates each feature of a file to assign it a rating describing how likely the file is to be malicious, then alerts the user past a certain threshold.

Model Maintenance and Enhancement

While initial models are created with all available files, threat vectors are always changing. Thus, models must be continually updated based on newly available threats.There is a balance to strike between updating a model's gaps (for example, on new malware that a model doesn't detect) and keeping its overall efficacy. Overall, internal tests have shown that adding only false positives or false negatives to a model can have deleterious effects on its efficacy. In general, the best training set iterations of new models combine a previous model's training set, a subset of new files the previous model was weak at, and a random grouping of new files. This mix helps retain the integrity of a previous model.

Conclusion

As a data science company, SparkCognition's goal is to determine the best algorithm for each problem and each model, discovering new ways of keeping organizations safe, rather than trying to make use of whatever algorithm is in vogue.

DeepArmor isn't just a security product. It's the continually evolving brainchild of a dedicated research team that is always looking to solve the fundamental problem of cybersecurity with increasing efficiency, accuracy, and finesse.