Magdeburger Journal zur Sicherheitsforschung

Gegründet 2011 | ISSN: 2192-4260
Herausgegeben von Stefan Schumacher
Erschienen im Magdeburger Institut für Sicherheitsforschung
http://www.sicherheitsforschung-magdeburg.de/publikationen/journal.html

This article appears in the special edition »In Depth Security – Proceedings of the DeepSec Conferences«.
Edited by Stefan Schumacher and René Pfeiffer

Malware Analysis

Machine Learning Approaches

Chiheb Chebbi

Machine learning is obviously the hottest trend in the tech industry at the moment, thanks to the huge amount of data collected in many organizations. It is so powerful to make decisions and predictions, based on big data. Fraud detection, natural-language processing, self-driving cars and image recognition are a few examples of machine learning applications. Machine learning is a combination of statistics, computer science, linear algebra, and mathematical optimization methods.

Keywords: Malware, Machine Learning, Malware Analysis, Deep Learning

Citation: Chebbi, C. (2018). Malware analysis: Machine learning approaches. *Magdeburger Journal zur Sicherheitsforschung*, *16*, 893–899. Retrieved November 17, 2018, from http://www.sicherheitsforschung-magdeburg.de/uploads/journal/MJS_060_Chebbi_MachineLearning.pdf

Machine Learning

Machine learning by definition is the art of studying and the creation of algorithms that learn from experiences to make predictions later. Machine learning models are giving computers the knowledge needed to make decisions from sets of examples. Tom Mitchell, a Professor at the Carnegie Mellon University (CMU) defines machine learning as follows: »A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E«. 1 illustrates the difference between traditional algorithms and machine learning

Machine learning models can be divided into four major categories: Supervised learning, semi-supervised learning, reinforcement and unsupervised learning.

Supervised Learning:

Machine learning models are categorized based on the datasets. If we have both the input and output variables it's supervised learning. In this case usually we know the input data and we have knowledge about the output classes. Thus, in supervised learning we just need to map the function or the pattern between the two variables (Inputs and outputs). Supervised learning can also be categorized into two sub models: Classification (the predicted value is a categorical variable) and Regression (the predicted values are continuous.)

There are many supervised learning techniques used in the wild, some of them are the following:

- Bayesian Classifiers: This technique is based on the bias formula. We have a prior belief and we need to update it using data.
- Support vector machines: Are used when we need to identify a hyperplane between the represented data. Determining the right hyperplane is based on two parameters called *regularization* and *gamma.*
- Decision Trees: It's representing data as upside-down trees using the Iterative Dichotomiser 3 algorithm.

Semi-supervised Learning:

In case when we have a small amount of labeled data we talk about semi-supervised learning. In other words we are going to deal with both labeled and unlabeled data.

Reinforcement Learning:

Reinforcement learning occurs when the agent is interacting with the environment and optimizes its performance using a scoring system or a reward function, as presented in 2.

Unsupervised Learning:

If we don't have an idea about the output variables we are using unsupervised learning. The following are some unsupervised learning algorithms:

K-means Clustering:

The k-Nearest-Neighbors (kNN) is one of the most used clustering techniques. The aim of it is finding a similarity in data or what we call *feature similarity.* This algorithm requires high memory. The classification is done like a vote. First, to know the class of a selected data, the distance between the selected item and the other training item must be computed. But how can we calculate these distances? In general we have two major methods for calculating, we can use the *Euclidean distance* or the *Cosine similarity*.

Artificial Neural Networks:

Artificial networks are one of the hottest machine learning algorithms used nowadays. The goal of artificial neural networks is building models that can learn like a human mind. In other words, we try to mimic the human mind. That is why, in order to learn how to build neural network systems, we need to have a clearer understanding of how a human mind actually works. The human mind is an amazing entity. The mind is composed and wired by neurons. Neurons are responsible for transferring and processing information. 3 describes the logical analogy of a human neuron. It is called a Perceptron.

There are a lot of activation functions rather than a »Unit step function« such as: Sigmoid Function, Tanh Function and ReLu Function.

Deep Learning

Many fully connected perceptrons compose what we call a Multi-layer perceptrons (MLP) Network. A typical neural network contains:

- Input Layer
- Hidden Layers
- Output Layers

We talk about the term »Deep Learning« once we have more than three hidden layers. 4 illustrates MLP networks.

There are many types of Deep Learning networks used in the wild like CNNs and RNNs:

Convolutional Neural networks (CNN): Passing a huge amount of information through the input layer could cause a problem, for example, in image recognition. Passing every pixel of a big image is not an efficient solution. This is why we need a type of neural network called a convolutional neural network, which is composed of a convolutional layer and a pooling layer (sometimes called a sampling layer) and, of course, an input and an output layer. 4 illustrates a CNN network.

Recursive neural network (RNN): A RNN is a neural network that is used when the input is sequential information and the input and outputs are independent of each other. Generally, it is very popular for processing natural-language processing tasks. An RNN has a memory that captures information about what has been calculated so far.

Machine Learning Systems Workflow:

Every machine learning project should follow specific steps to achieve its goal. The first step is data

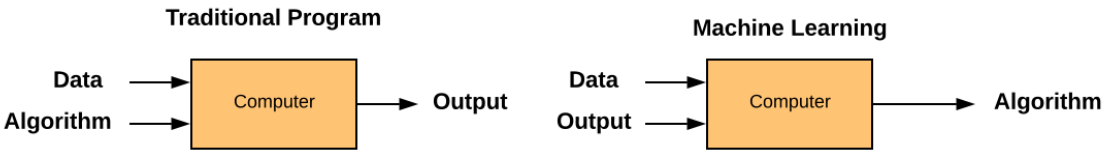


Figure 1: Traditional Program vs. Machine Learning

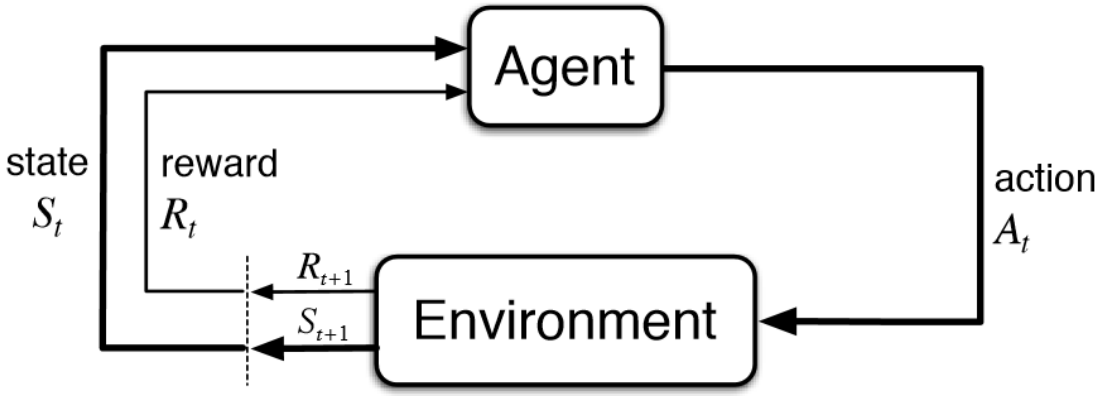


Figure 2: Source https://cdn-images-1.medium.com/max/1600/1*c3pEt4pFk0Mx684DDVsW-w.png

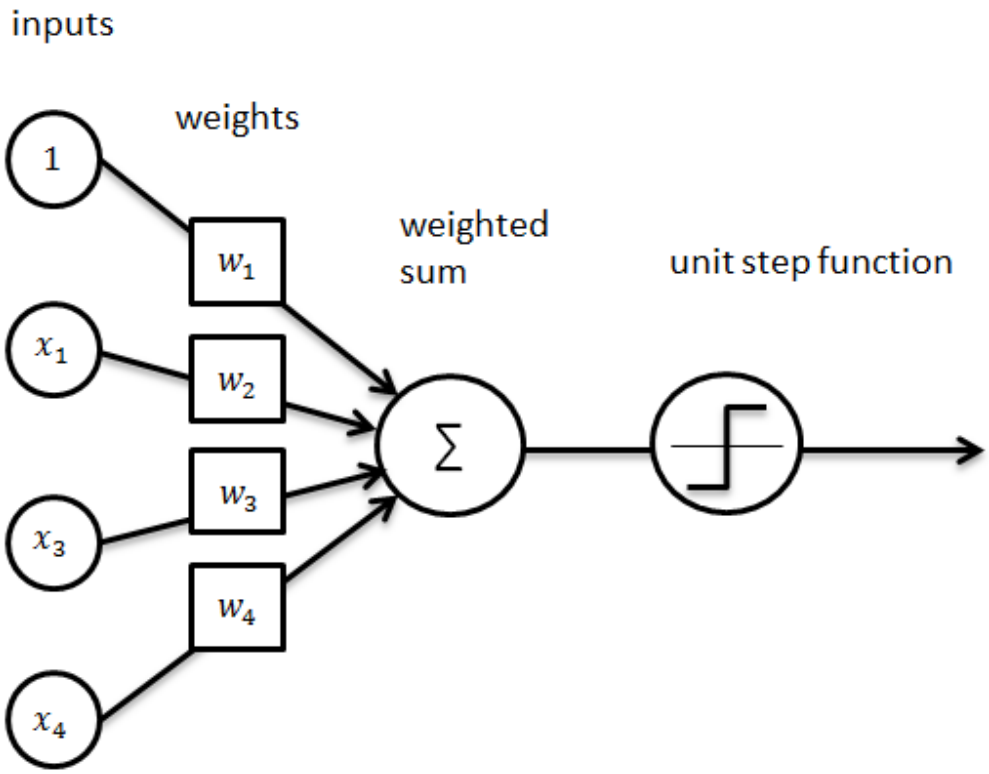


Figure 3: Source: http://ataspinar.com/wp-content/uploads/2016/11/perceptron_schematic_overview.png

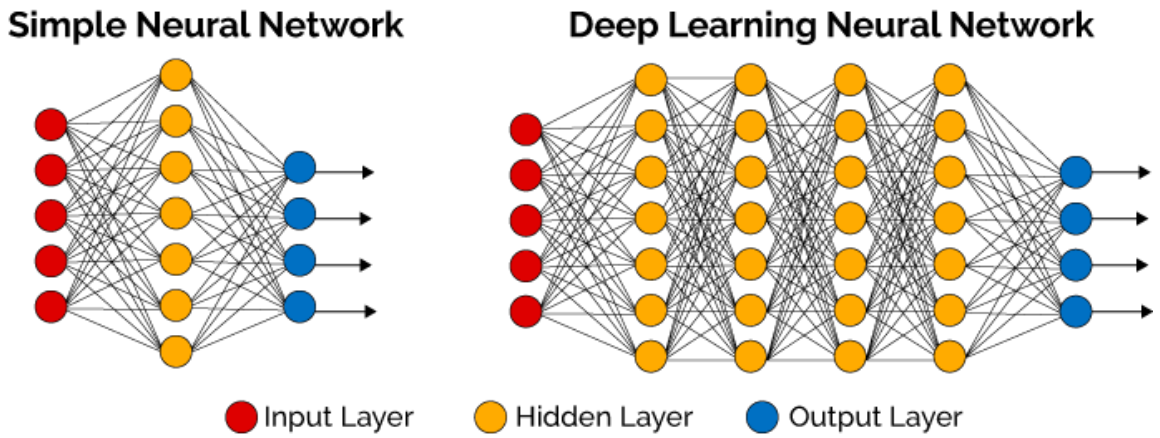


Figure 4: Source http://www.global-engage.com/wp-content/uploads/2018/01/Deep-Learning-blog.png

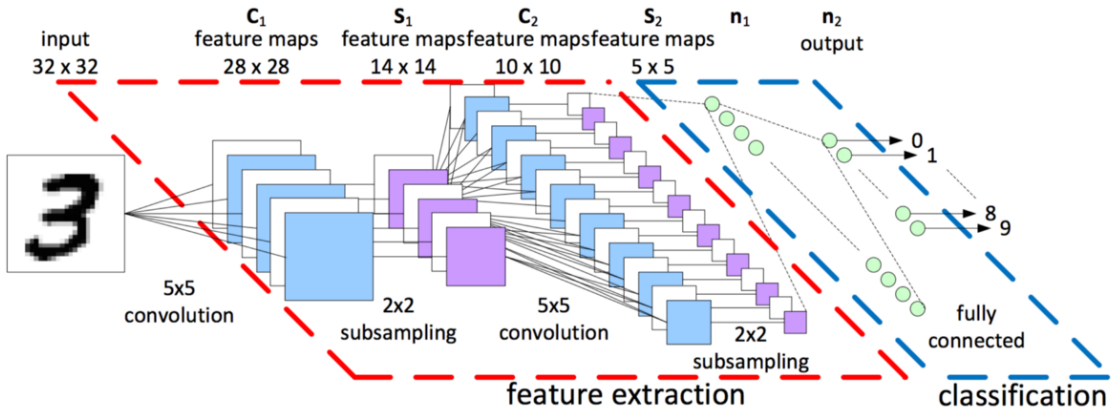


Figure 5: Source : https://cdn-images-1.medium.com/max/1200/1*EPpYI-llkbtwHgfprtTJzw.png

processing—during this step we need to extract the meaningful features from the raw data. This step is crucial because good feature engineering is needed to build a good machine learning model. After processing the data, we have to train and choose the best predictive model for our situation. Finally, after training the model, evaluation is an important process where we check the accuracy and the performance of the trained model to predict new data

Machine Learning Evaluation Metrics:

In order to evaluate the performance of machine learning models we can use many evaluation metrics such as: Precision, Recall, F-Score, accuracy based on 4 parameters: True Positive (TP), False Positive (FP), True Negative(TN) and False Negative (FN)

Precision:

Precision or Positive Predictive Value is the ratio of the positive samples that are correctly classified by the the total number of positive classified samples. Put simply, it is the number of the found samples that are correct hits.

Recall:

Recall or True Positive Rate is the ratio of true positive classifications by the total number of positive samples in the dataset.

F-Score:

F-Score of F-Measure is a measure that combines precision and recall in one formula

Accuracy:

Accuracy is the ratio of the total correctly classified samples by the total number of samples.

Confusion Matrix:

A confusion matrix is a table that is often used to describe the performance of a classification model based on the four discussed parameters.

Malware Analysis

Malware analysis is the art of determining the functionality, origin and potential impact of a given malware sample, such as a virus, worm, trojan horse, rootkit, or backdoor. As a malware analyst our main role is to collect all the information about the malicious software and have a good understanding of what has happened to the infected machines. Like in any process to perform a malware analysis we typically need to follow a certain methodology and a number of steps.

By definition a malware is a malicious piece of software with the aim of damaging computer systems by f.ex. data and identity stealing, espionage, legitimate users infection or gaining full or limited control over the system. To have a clear understanding of malware analysis, a malware categorization based on its behaviours is a must. Sometimes we cannot classify a malware because it uses many different functionalities, but in general malware can be divided in many categories some of them are described below:

- Trojan: is a malware that appear as a legitimate application
- Botnets: are networks of compromised machines which are generally controlled by a command and control (C2C) channel
- Ransomware: this malware encrypts all the data on a computer and usually asks the victim to pay ransom in the cryptocurrency Bitcoin to get the decryption key
- Spyware: as its name plainly states this is a malware that tracks all the user activities including search history and installed applications
- Rootkit: enables the attacker to gain unauthorized, usually administrative access, to a system. Basically it is unnoticeable and makes its removal as hard as possible

Like in any process to perform a malware analysis you typically need to follow a certain methodology and a number of steps. There are three fundamental approaches to malware analysis: Static analysis, memory analysis and dynamic analysis.

Static Malware Analysis

Static malware analysis refers to the examination of the malware sample without executing it. It consists of providing all the information about the malicious binary. The first steps in static analysis are to gain knowledge of the malware size and file type in order to have a clear vision about the targeted machines in addition to determining the hashing values, because cryptographic hashes like MD5 or SHA1 can serve as a unique identifier for the sample file. To dive deeper, finding strings, dissecting the binary and reverse engineering the code of the malware using a disassembler like IDA, could be a great step to explore how the malware works by studying the program instructions. Malware authors often are trying to make the work of malware analysts harder so they are always using packers and cryptors to evade detection. That is why, during static analysis, it is necessary to detect them using tools like PEiD.

Memory Malware Analysis

Memory malware analysis is widely used for digital investigation and malware analysis. It refers to the act of analysing a dumped memory image from a targeted machine after executing the malware to obtain a multiple number of artifacts including network information, running processes, API hooks, kernel loaded modules, Bash history, etc. ... This phase is very important because it is always a good idea to have a clearer understanding about the malwares capabilities. The first step of memory analysis is memory acquisition by dumping the memory of a machine using a various number of utilities. One of these tools is fmem, which is a kernel module to create a new device called /dev/fmem to allow direct access to the whole memory. After downloading it from their official repository and compiling it you can acquire the machine memory using this command:

```
# dd if=/dev/fmem of=...  bs=1MB count=...
```

Another tool is The Linux Memory Extractor. LIME is a Loadable Kernel Module (LKM) to allow volat-

ile memory acquisition from Linux and Linux-based devices such as Android. After having a memory dump, it is time to analyze the memory image. You can simply use volatility framework, which is an open source memory forensics tool written in Python. Volatility comes with various plugins and a number of profiles to ease obtaining basic forensic information about memory image files.

Dynamic Analysis

Performing static analysis is not enough to fully understand malware's true functionality. That is why running the malware in an isolated environment is the next step in a malware analysis process. During this phase the analyst observes all the behaviours of the malicious binary. Dynamic analysis techniques track all the malware activities including DNS summary, TCP connections, network activities, syscalls and much more.

Isolation is a security approach provided by many computer systems. It is based on splitting the system into smaller independent pieces to make sure that a compromised sub-system cannot affect the entire entity. Using a sandbox to analyse malware is a wise decision to run untrusted binaries. There are many sandboxes in the wild, such as Cuckoo Sandbox and LIMON, which is an open source sandbox developed by CISCO systems Information Security Investigator Monnappa K A as a research project. It is a Python script that automatically collects, analyzes, and reports on Linux malware. It allows one to inspect the Linux malware before execution, during execution, and after execution (post-mortem analysis) by performing static, dynamic and memory analysis using open source tools.

Malware Analysis Using Machine Learning:

Nowadays, information security is becoming a more pressing concern. Devices and networks play an important role in every modern organization. But if the organization does not properly test and secure their solutions and environment black hat hackers or adversaries can compromise these solutions, damage business functionality and steal data. Unfortunately, many organizations operate under the misapprehension that security scanners and antiviruses will reliably discover malware in their systems. In reality, effective cyber defense requires a realistic and thorough understanding of malware analysis techniques. Malware attacks are becoming more sophisticated and dangerous. With millions of malicious programs in the wild it becomes hard to detect zero-day attacks and polymorphic viruses. This is why the need for machine learning-based detection arises.

Many malware detectors based on machine learning have begun to surface. They can create great solutions for detecting advanced threats while malware analysts and security engineers can extract useful features from the collected data and build machine learning models. Information security professionals and data science enthusiasts are free to choose the most convenient machine learning algorithm and model for their purposes, which is why there are various explored machine learning anti-malware systems available. One of them is artificial neural networks and, in particular, deep learning.

As a demonstration, for example, if you want to build a malware classifier using deep learning, first you need to train your model with a big amount of data and malware samples to achieve great accuracy. In order to do that you can download the malware dataset used in the Kaggle malware classification challenge 2015. Kaggle is a public website that delivers machine learning and data science trainings and challenges in which companies and researchers post data, and statisticians and data miners compete to produce the best models for predicting and describing the data. In the Microsoft Malware Classification Challenge (BIG 2015) they are providing a dataset of 500 GB of malware samples as assembly format from the Microsoft Malware Protection Center.

Feature engineering is an essential task. To build the model you can convert the sample files into images and train the model like anydeep learning model using image classification. Based on a paper called »Malware Images: Visualization and Automatic Classification« (Lakshmanan Nataraj Vision Research Lab University of California, Santa Barbara) the University of Pittsburgh provided us with a great trick, which is not converting every hex variable but every byte of a file into a pixel color. 6 illustrates a Malware binary as a greyscale image

Conclusion

Threats are a growing problem for people and organizations across the globe. A good understanding of malware analysis and machine learning models is vital to ensure making wise decisions and building a secure environment by being capable of correctly identifying and mitigating such potential threats. Unfortunately, this is still a challenging area for information professionals, because threats and malware are becoming more sneaky and harder to detect everyday.

About the Author

Chiheb Chebbi is a Tunisian InfoSec enthusiast, author, and technical reviewer with experience of various aspects of information security, focusing on investigating advanced cyber attacks and researching cyber espionage. His core interests lie in penetration testing, machine learning, and threat hunting. He has been included in many Halls Of Fame. His talk proposals have been accepted by many world-class information security conferences.

References

1 Advanced infrastructure penetration testing: Author Chiheb chebbi - Packt publishing

2 e-Forensics Magazine: Forensics of Things - ISSUE 07/2017

3 Practical Malware Analysis: A Hands-On Guide to Dissecting Malicious Software 1st Edition - ISBN-13: 978-1593272906

4 Limon - Sandbox for Analyzing Linux Malwares

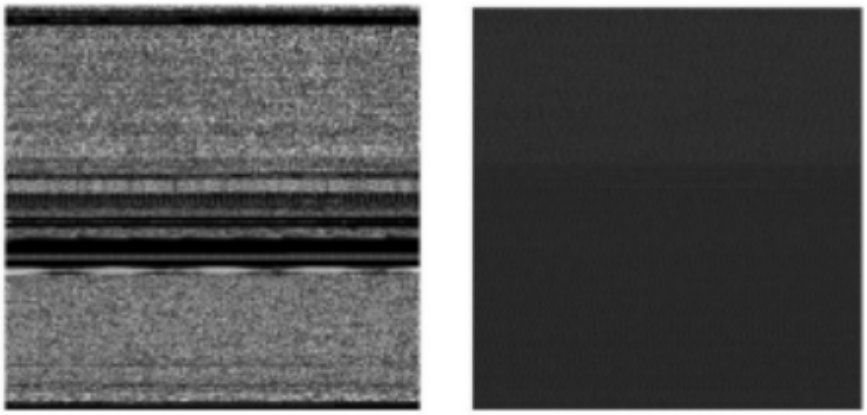


Figure 6: Malware Binary represented as a Greyscale Image

https://github.com/monnappa22/Limon

5 Volatility Framework www.volatilityfoundation.org.

6 The 10 Algorithms Machine Learning Engineers Need to Know http://www.kdnuggets.com/2016/08/10-algorithms-machine-learning-engineers.html

7 Microsoft Malware Classification Challenge (BIG 2015) https://www.kaggle.com/c/malware-classification

8 Confusion matrix:scikit-learn 0.19.0 documentation http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

9 coursera machine learning Course https://www.coursera.org/learn/Machine-learning

10 Deep Learning http://deeplearning.ai

11 »Malware Images: Visualization and Automatic Classification« (Lakshmanan Nataraj Vision Research Lab University of California, Santa Barbara)