Using ML Models to Detect Malicious Traffic: Testing ML Models

Paulo Pereira
Informatics
Dept. Uninove University
São Paulo, Brazil

Abstract

The purpose of this article (under construction) is to present an idea for a Postdoctoral research which can implement machine learning modeling for combined cyber-attacks. Combined cyber-attacks are those that use traditional attack techniques coupled with the use of IoTs to widen the attack surface to specific targets.

1. Introduction

The continuation of this research will occur by setting up a test lab with a hybrid network, including different operating systems (Windows, Linux) and IoTs devices (web cam, mobile phones) and microprocessors such as Raspberry / Arduino. Are at least two main concerns in using machine learning models to classify data from cyber-attacks: I) how to measure the accuracy of the model used and II) how to implement model-based detection rules. Once the lab has its connections established, machine learning models will be tested for classifying three attack modes: denial of service (and variations), data interception, and intruder detection.

2. Our Research Objectives

The latest information on cyber-attacks reveals that a combination of different types of attacks are being used against certain targets, combining the modalities for cybercrime [1]. We now have more questions than answers because we want to start the lab and then collect data. Combined attacks can target, for example, global hijacking of DNS services using IoT mechanisms and distributed computers or cause a service to discontinue.

3. Preparing the Scenario

The network that will serve as an experiment in the laboratory consists of the following elements:
a) A server that will be used as a sandbox and therefore will receive normal traffic and malicious traffic;
b) Data capture from this network will be made by a sniffer written in Python;
c) The time frame (traffic window) will be 10 days for the experiment to collect a consistent sample of data. This time may be longer according to the need of the experiment.

The server will be directed to capture data that will be output from a computer mounted with the Kali Linux system. Firstly, the Netwox tool will emit non-malicious request traffic for 5 uninterrupted days: they are requests that do not alter the response time of the server. Later, in the next 5 days, the tool will issue data packets with requests that force the server to recognize any request and thus have its response time tampered with, collapsing.

Our experiment will not use third party data, which implies generating our data until a consistent sample is obtained (from a statistical point of view). After obtaining this data, the intention is to train a deep learning model for the proper classification of this sample, creating two sets of data which will be called “the first 5 days” or simply “f5d” and, consequently, " the last 5 days” or just “l5d”.

The Netwox tool can send both normal and malicious traffic. The central aspect we want to understand is how accurate our model will be so that the separation between normal and malicious can reduce the margin of error to an acceptable level in a business environment that is attacked by these cybercrime parameters.

Our challenge starts with choosing a machine learning model that can classify attacks combined with a low false positive rate. Which existing models have the ability to handle data from a cyber-attack in our lab that can classify that attack as a denial of service, or an intrusion or data interception?

Network traffic can reveal many licit or illicit activities, such as spying on competitors, possible intrusion, malware persistence, among other elements. A package in a given traffic may contain hidden information, which at first glance is not captured by tools such as Wireshark. Malware can hide your connections and traffic analysis kicks in. Within Linux systems, an important tool that aids in finding hidden information is Bulk_Extractor. But it can also be cited the tshark. Considering that the focus of our experiment can established the relationship between DoS attack against HTTP connections, this focus is on discovering network

activities and use the Wireshark capabilities to extract HTTP objects. Note that Wireshark columns could have been changed for better analysis.

Cyber-attacks are notable for the evolution of techniques designed to deliver a malicious file to a victim. Most attacks rely on the victim clicking a link malicious or executing a malicious file. In 2017 there were more than 25 million ransomware attacks, which represents an extremely considerable volume of this type of attack. In a different direction, we have denial of service (DoS), which is a powerful and recurring attack with a high success rate, depending on the infrastructure used by the attackers. Companies can count on a strong line of defense against this type of attack, but what you see nowadays is certainly the absence of a machine learning solution applied to the problem. There are several machine learning experiments tested against a denial of service attack data set, but what our experiment proposes is a critical point of view regarding the error rates of the algorithms, if these rates are acceptable in the cyber security field. . A representative data set, that is, a statistically acceptable sample of denial of service attacks can be modeled in terms of a supervised scenario, which requires knowing what is being inserted into the model and expecting what we expect in response. But is this really acceptable? If acceptable, could the generalization from this classification detect an HTTP connection-level attack by considering a Trojan horse spreading malicious code or a fragmented file set, which after the infection would be recompiled by the attacker?

Simply put, the greater the number of machines sending request packets to the target, the more likely that target will be unavailable. When this attack is made by groups of attackers who come together to attack a target (for example, a government website, an environmentally damaging company website) it is referred to as distributed denial of service (DDoS). The attack that uses the psychological side that presses the victim to open a link or an email attachment (for example) is called phishing. This type of attack occurs constantly and is one of the main forms of ransomware delivery. An illustrative example of this type of attack is built using SET. A fake Google page is created and if the victim does not test the link before entering their data (in this case user “test” and password “password” a communication with the attacker's machine will be created (see figure below). The attacker uses Metasploit to create a malicious attack that also depends on the user's distraction. The more pressed the victim is, the more likely he is to click on the malicious artifact. The attacker's ability to make an artifact not detected by antivirus counts a lot in this mode of attack. Our experiment to create the Trojan horse to exploit HTTP-level network traffic will be created with the following command (typed in the Kali Linux terminal):

msfvenom -a x86 --platform windows -p windows / meterpreter / reverse_tcp lhost 192.168.56.4 lport = 135 -f exe -o svchost.exe

Metasploit will create a malicious artifact in low persistence state and detectable by any antivirus. The attacker needs to change these features to get an undetected malicious executable. What does this malicious file actually do? Let us explain the command syntax:

msfvenom -a x86 --platform windows -p windows / meterpreter / reverse_tcp lhost 192.168.56.4 lport = 135 -f exe svchost.exe

-a x86 creates an artifact for 32-bit Windows platforms.

--platform windows: Indicates the platform to attack.

-p windows / meterpreter / reverse_tcp: Indicates the payload connection type. In this case, it will be a reverse connection. This means that the attacking machine forces the connection to the victim's machine. But it still depends on the victim clicking the malicious file!

lhost: indicates the attacker's machine (local machine, localhost)

lport: indicates the port for reverse connection. It can be any open port in Windows, but it has to be a poorly monitored port.

-f: Prompts for malicious file format (in this case, exe, an executable).

-o: Prompts for the file name (-o is equivalent to --output). This part is key. In the example above svchost.exe was used because this is a widely used library in Windows. But it could be any name.

This type of attack depends on social engineering. This means that creating the artifact does not guarantee success in the attack. For this reason, the delivery of the malicious file to the victim depends greatly on the form (or means) of delivery. Note that the malicious file appears on Drive C of the Windows machine. This situation raises some fundamental questions:

- How did the attacker get this file in Drive C?
- How will the attacker convince the victim to click on this file?

While these difficulties exist, many attacks have this form. And many of these attacks are still successful, whether against ordinary users or against employees of a company. Since Cyber-attacks that are organized in various countries use techniques ranging from the use of malicious code (e.g., JavaScript) to distributed denial of service for the purpose of obtaining strategic information for commercial purposes [5].

Once the artifact is created, we intend that the host that will receive it will be responsible for unleashing the attack on the network to which the host is connected. In other words, distributed denial of service will be performed in a completely different model from what we are intended to witness: an attack from outside the network into the network will not be delivered, but instead the denial of service will occur within the network and will begin to be distributed by a machine on the network itself. Collected samples of a conventional denial of service attack (out-of-network standard) and trained deep learning models, the packets collected from the denial of service attack (in-network standard) will be tested by the machine learning for classification purposes. That is, if a machine learning learns what a denial of service attack is through the conventional aspect (the attack is expected to occur from the outside into a network), it would be possible that this machine learning will be able to classify our attack from within the network itself as a denial of service attack?

It is very common to see denial of service attacks gathering a large number of zombie machines causing the downfall of services and systems. Invariably, some elements are measured in this mode of attack, namely: the connection port, the bandwidth, the number of requests per millisecond, the associated protocol, among others. However, when a backdoor is programmed to deliver a series of fragmented files, which will be recompiled after the attacker's access to the compromised machine, we can set the malicious code to change the connection port after a certain time. So could machine learning classify our attack as denial of service?

In other words, our intention is to study the scenario in which denial of service (HTTP flooding) occurs from a compromised machine, with the possibility of the attacker being able to access other devices on this corporate network, using lateral movement, and, from success these actions, enable HTTP connections to be compromised. In parallel to this attack, the network data capture will be organized in the pattern we are setting, with 5 days of no attack capture and five days of attack capture. In the 5 days without attack, here called "f5d", the Netwox tool will issue the requests in the conventional pattern, that is, the requests will respect a formal packet send, receive (SYN) and response acknowledgment (ACK) standard. . However, when the experiment is in the last five days, here referred to as "l5d", the delivery of malware and its target contamination on the network will take action, followed by the outbreak of packets that will disrupt the sending pattern, receiving pattern and recognition response. Our first conclusions regarding the compromise of the target machine is that this mode of infection, overcoming the social engineering barrier step, is very effective and allows the attacker an initial control of the compromised machine shell. Assuming the attacker knows the minimum steps necessary to get root control of the target machine, in our case a 64-bit Windows 10 machine (which will lead the attacker to mitigate the UAC of the Windows system).

4. Preliminary Conclusions

The models to test are: Nearest Neighbor and Naive Bayes Classifier [2]. Depending on the tests results and accuracy, other models will be tested, but in a second step of the research. At this time, the key issues are those that require the detection of an intruder on a network and how long this is recognized by the model after submission of packets collected on the laboratory network [3].

With the sample collected and the two groups separated (normal and malicious), the tests performed so far point to the use of a hedonic regression tested with a dependent variable (response time) being tested against the independent variable (size in bytes), because we do not intend to model a curve of attributes passing through the network, but to find out how the server response time is tampered with and classify this change.

All stages of the privilege scale phase in the compromised system are very important for the measurement of commitment indicators. Our preliminary conclusion is that it is possible that our machine learning will be able to return these indicators to the end user with consistent alerts. However, this step is taken much later than the experiment. First, we want to test our algorithms for a reliable response, with a model error rate that is acceptable. Certainly, it plays an essential role in this experiment because we should eliminate as much as possible the risk of sample bias. Biasing a sample causes irreparable damage to a model. For this reason, when samples are being collected, monitoring of this collection will be done constantly by using Suricata on our server. Initially, we thought of a virtual network this experiment. We evaluated the possibility of using a physical network and this idea will be implemented with five computers: a Kali Linux machine, a Windows 10 64-bit machine and three Ubuntu machines (version 18 LTE).

The change in network is mainly due to the fact that virtual machines have network limitations and

lose essential properties when configured HTTP stream is not present. For this reason, the physical machines have been selected and will receive all of a corporate network: monitoring, network bandwidth provisioning, HTTP access scales, port management, among other elements.

A denial of service attack from a company's own network after compiling malicious code sent by a backdoor has the advantage of being discovered late by the company's technology team. In addition, there is a tendency for companies to hire information security companies. There may be a situation in which ignorance about the attacked network itself prevails. For this reason, the use of machine Learning may be essential to add an extra layer to the surface to be protected. Another factor to consider is the possibility of recognizing the indicators of compromise (which are not always clear enough to implement protective measures against this cyber-attack mode. For example, assuming that port 80 (and its complementary 8080) be used by the denial of service attack and malware can change the port at any given time; machine learning has to be able to identify a request and response pattern present in packets sent in. The strength of the machine learning implementation is classification, and this requires pattern recognition. Our aim is to find out what can be used to obtain these patterns and thereby avoid elements with a high degree of randomness (in this case, for example, The more consistent standards our network can provide, the lower the error rate of the model. even if preliminarily, given the error rate and statistical confidence margin of this experiment. One of the problems faced with a Hyper-V-mounted virtual network magnet is that the network latency presented an inconsistency for the same attack. In other words, by directing the first attack to the Windows 10 machine, the initial moment when the intensification of requests characterizing the beginning of the attack is not the same as that found in the second attack, maintaining the same controlled conditions of the experiment, but changing the types. of networks. In this case, the host-only network was the least discrepant, even simulating the HTTP connection through the Kali Linux machine server. Because of the discrepancies in detecting the start time of the attack, setting up a network of five physical machines became an important research strategy as all machines were wired and wirelessly connected. Initial testing will only be done with the wired Ethernet connection. Comparative measurement of the times will be done with Wireshark. Identifying this starting point of the attack is very important for a deep learning framework because the same attack, delivered against the same target and by the same attacking machine, cannot have two different start times and there are two distinct times for the start of the attack. attack, the area should be discarded. This cutting or sample selection criterion is very important for the absence of possible bias of these samples. Another sample selection criterion which is likewise relevant to counteracting the bias of collected samples concerns the number of malformed packages. If packets classified by Wireshark are malformed under the same conditions as network traffic, then possibly the criterion will be strong enough to be used. Finally, a third sample selection criterion, in addition to malformed packages, will be Wireshark's expert rating for the denial of service attack mode. When expert mode similarly identifies attacks at different times against the same target (same IP), then this sample is validated to be part of the model training.

5. References

[1] Hirani, M., Jones, S., Read, B., (2019). Global DNS Hijacking Campaign: DNS Record Manipulation at Scale, https://www.fireeye.com/blog/threat-research/2019/01/global-dns-hijacking-campaign-dns-record-manipulation-at-scale.html.

[2] Milosevic Nikola Milosevic, N., Dehghantanha, A., Choo, K.R., “Machine learning aided Android malware classification”, in Computers & Electrical Engineering, vol. 61, pp. Pages 1-384 (July 2017).

[3] Stallings, W., (2019). Effetive Cybersecurity. Addison-Wesley, pp. 501-502.

[4] DU, Wenliang. Computer & Internet Security. A hands-on Approach. Independently Published. Syracuse, 2019.

[5] McLure, M., Hacking Exposed: Network Security Secrets & Solutions. New York: McGraw-Hill, 2014, p. 323.

