Applications of Machine Learning to Threat Intelligence, Intrusion Detection and Malware

Charity Barker

A Senior Thesis submitted in partial fulfillment
of the requirements for graduation
in the Honors Program
Liberty University
Spring 2020

Acceptance of Senior Honors Thesis

This Senior Honors Thesis is accepted in partial
fulfillment of the requirements for graduation from the
Honors Program of Liberty University.

_____
Robert J. Tucker, Ph.D.
Thesis Chair

_____
Melesa Poole, Ph.D.
Committee Member

_____
David Schweitzer, Ph.D.
Assistant Honors Director

_____
Date

Abstract

Artificial Intelligence (AI) and Machine Learning (ML) are emerging technologies with applications to many fields. This paper is a survey of use cases of ML for threat intelligence, intrusion detection, and malware analysis and detection. Threat intelligence, especially attack attribution, can benefit from the use of ML classification. False positives from rule-based intrusion detection systems can be reduced with the use of ML models. Malware analysis and classification can be made easier by developing ML frameworks to distill similarities between the malicious programs. Adversarial machine learning will also be discussed, because while ML can be used to solve problems or reduce analyst workload, it also introduces new attack surfaces.

Keywords: Machine learning, threat intelligence, intrusion detection, malware analysis, adversarial machine learning

Applications of Machine Learning to Threat Intelligence, Intrusion Detection, and Malware

Introduction

Artificial intelligence (AI) and machine learning (ML) have become marketing buzzwords over the last few years and are often used interchangeably. There is a subtle but important difference between them. AI is a more conceptual term, providing a thousand-foot view of machines with human-like capabilities completing tasks usually carried out by humans. ML is a collection of specific approaches to creating a machine capable of AI (Polyakov, 2018). It is useful to think of ML as the implementation behind an artificially intelligent machine. This paper focuses solely on ML instead of ML and AI because ML is a collection of specific approaches rather than a concept. Therefore, ML use cases are much more plentiful than their AI counterparts. A discussion of ML rather than AI also allows for further discussion of specific techniques and algorithms not visible from the broad perspective of artificial intelligence.

Machine learning is sometimes seen as a quick fix or add on that can provide an adequate solution for almost any problem. In reality, machine learning provides solutions for very specific problem types. These types of problems must have a clear use case and a large amount of relevant data (Google, 2019). Three specific problem areas within cyber security are suited for machine learning solutions: threat intelligence, intrusion detection, and malware analysis or classification.

Threat Intelligence

Machine learning has many potential applications to threat intelligence. A major unsolved issue in cyber threat intelligence is the problem of attack attribution. Due to the asymmetric and remote nature of cyberattacks, it is very difficult to conclusively determine the

origin of an attack. In this context, asymmetric refers to asymmetric warfare, which can be defined as "the use of surprise force by a weaker party against a stronger force's vulnerability" (Wang & Stamper, 2002, para. 1). A subcategory of asymmetric warfare is war by proxy, which occurs when warfare is covertly carried out by a group acting on behalf of a nation or state, but not by the government itself (Lele, 2014). Some cyberattacks, especially those against critical infrastructure, fall into this category further confusing the issue of attribution.

Attribution is an important piece of the threat intelligence puzzle. Simply put, threat intelligence is "the process of understanding the threats to an organization based on available data points" (Bromiley, 2016, p. 1). A more descriptive definition of threat intelligence is "evidence-based knowledge, including context, mechanisms, indicators, implications and actionable advice about an existing or emerging menace or hazard to assets that can be used to inform decisions regarding the subject's response to that menace or hazard" (McMillan, 2013, para. 1). Threat intelligence is more than just data – it is a combination of the data and context that can be used to protect an organization from threats or make decisions during the process of responding to a threat.

There are several types of threat indicators (or indicators of compromise), including file hash values, IP addresses, domain names, network artifacts, host artifacts, tools, and tactics, techniques and procedures, or TTPs (Bianco, 2014). Each indicator is tied to a specific group of threat actors with varying degrees of certainty. For example, it is relatively easy to change the hash value of a malicious executable, an IP address, or a domain name, so these indicators are loosely tied to groups of threat actors. Network and host artifacts, tools and TTPs are more difficult to change, and are more strongly tied to threat actor groups (Bianco, 2014). The goal of

threat intelligence is to detect incidents as soon as possible and in the best case prevent them (Bromiley, 2016).

Microsoft Defender Advanced Threat Protection

The development of threat intelligence is a problem that is well suited for machine learning. Threat intelligence is often gained through manual analysis of collected data. Different ML algorithms can be used to automate the process of data analysis and produce actionable intelligence. For example, the Microsoft Defender Advanced Threat Protection (ATP) Research Team has created a natural language processing system that extracts TTPs from publicly available documents, identifies categories, and labels relationships between the identified categories (Soman, 2019).

This system is "trained on documentation of known threats, [and] takes unstructured text as input and extracts threat actors, attack techniques, malware families, and relationships to create attacker graphs and timelines" (Soman, 2019, para. 4). This machine learning model was used to determine common techniques between identified threat actor groups and the Emotet malware family, enabling organizations to "place defensive choke points to detect or prevent these attacker techniques so that they can stop not only annoying commodity malware, but also the high-profile targeted attacks" (Soman, 2019, para. 14). This is an excellent example of machine learning being used to provide actionable threat intelligence that can be used to prevent attacks.

APTinder

APTinder is a machine learning model currently under development by FireEye to assist in the automation of the daunting manual intelligence analysis process and subsequent threat

actor grouping. FireEye has a large existing data set to work with, and the goals for the model are to: "create a single interpretable similarity metric between groups, evaluate past analytical decisions, and discover new potential matches" (Berninger, 2019, para. 7).

Each topic or potentially unique feature has its own model, which not only allows for each model to be fine-tuned but also allows for topic weights to be changed for the final grouping. Like the Microsoft Defender ATP Research Team's approach, the data for FireEye's project is collected from a vast body of reports. A technique called Inverse Document Frequency (IDF) is used to score uniqueness of terms from reports, which are in turn vectorized (Berninger, 2019). Cosine similarity is used to determine similarities between various groups, which are represented by the vectors from the previous step (Berninger, 2019). Cosine similarity is the value of the cosine of the angle between two vectors, which essentially measures how parallel the vectors are. This process is repeated for each category or topic. At the time of Matt Berninger's CAMLIS talk in 2018, the categories were weighted with a straight average, but Berninger hopes to build an objective weighting system for the overall model based on existing data (Berninger, 2018).

The goal of the APTinder project is to automate and enable analysis of large amounts of data in a scalable fashion, ultimately resulting in more robust attack attribution:

> We've seen groups pretend to be other groups sometimes; it's sort of an info operation where they put up a social media account or some public account and say they're somebody else doing this for some other reason. Making sure we have a diverse data set, […] where we're looking at infrastructure and methodology and things that are harder to fake, that gives us […] more robust and less brittle attribution because it's easy to just

> grab malware out of a forum; it's really hard to fundamentally change the way you do your operations. (Berninger, 2018, 20:29)

Applications of ML to threat intelligence, especially attribution, are currently being developed, tested, and tweaked. Attribution will remain a difficult problem due to its convoluted and often political nature. However, ML can help automate parts of the analysis process, increasing the scalability of threat intelligence and attribution efforts by reducing the threat intelligence analyst workload.

Intrusion Detection

Intrusion detection is the first step to responding to a successful attack and is also the starting point of the incident response lifecycle. The National Institute of Standards and Technology (NIST) provides reference materials and compliance standards for different technological areas, including cyber security. NIST defines the incident response life cycle (See Figure 1) not as a linear process but as a cycle between several activities, including preparation, detection and analysis, containment eradication and recovery, and post-incident activity (U.S. Department of Commerce, 2012). While preparation and planning for an incident are important and can define how smoothly the rest of the process proceeds, the rest of the life cycle cannot proceed without an initial detection. Therefore, it is essential that incident detection occurs accurately and in a timely manner.

Unfortunately, "for many organizations, the most challenging part of the incident response process is accurately detecting and assessing possible incidents" (U.S. Department of Commerce, 2012, p. 26). Many characteristics of incident detection make this problem difficult. These characteristics include that "incidents may be detected through many different means [,]

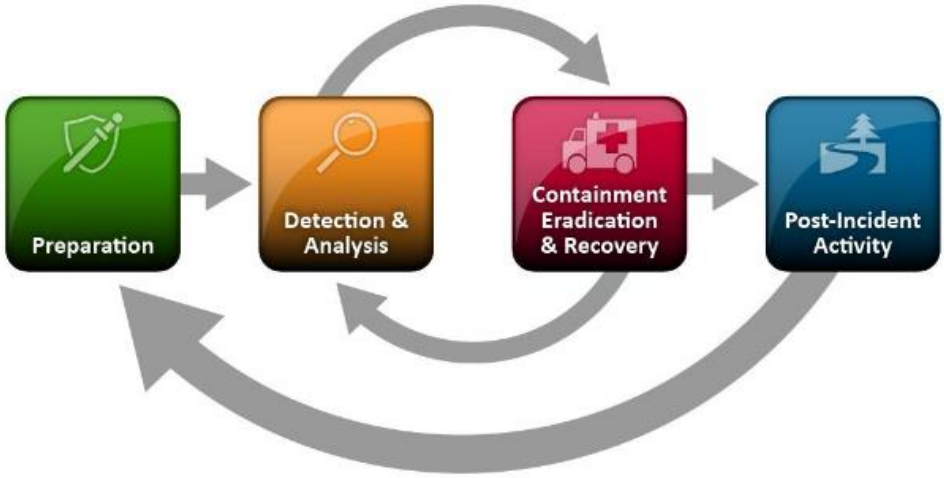

Figure 1. Incident Response Life Cycle. Adapted from *Computer Security Incident Handling Guide*, by NIST, 2012, retrieved from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf.

the volume of potential signs of incidents is typically high [, and] deep, specialized technical knowledge and extensive experience are necessary for proper and efficient analysis of incident-related data" (U.S. Department of Commerce, 2012, p. 26). Use-cases for machine learning can address each of these problems by collecting, processing and classifying data points from network-based and host-based sources at a high volume and potentially reducing workload for analysts.

Intrusion Detection Systems

There are two major categories of intrusion detection systems (IDSs), network-based and host-based. Network-based systems focus on the detection of malicious or anomalous network traffic using network artifacts, while host-based systems are focused on detecting malicious events and artifacts in event logs on individual computers or servers. Much of intrusion detection revolves around defining and detecting what is not normal for a system, group of systems, or network and then alerting an analyst when abnormal events occur. A significant difficulty in automating detection of abnormal events, or intrusion detection, is that

normal usage is different for different groups of users, so intrusion detection systems must be tailored to the systems and networks they monitor.

Rule-based intrusion detection systems trigger an alert when a specific set of conditions or rules is met. These systems can reliably detect a predetermined set of attacks but will miss all potential indicators of compromise that do not exactly match a rule that has been created in the system. In some situations, this is beneficial, because there will be few false positive alerts, which reduces alert fatigue. On the other hand, rule-based intrusion detection systems are almost guaranteed to miss attacks that do not yet have a signature. Cybersecurity is a field that is constantly changing, and sets of rules cannot keep up with new attacks. Machine learning can be used to fill the gap between rule-based intrusion detection systems and constantly evolving attacks. IDSs that make use of machine learning techniques are commonly categorized as anomaly-based detection systems:

> Anomalies are considered important because they indicate significant but rare events and can prompt critical actions to be taken in a wide range of application domains; for example, an unusual traffic pattern in a network could mean that a computer has been hacked and data is transmitted to unauthorized destinations. (Ahmed, Mahmood, & Hu, 2016, p. 20)

Before a machine learning system can be implemented, a large amount of relevant data must be collected. Netcap is an open source tool designed to automate the process of converting "a stream of network packets into platform neural typesafe structured audit records that represent specific protocols or custom abstractions" (NETCAP, 2019, para. 1). Then, data must be labeled

before it can be used. The collection, processing, and labeling of new datasets is often time consuming, but tools are available to automate portions of this process.

Network-based Applications of Machine Learning

Machine learning can be applied to network-based intrusion detection systems. However, it is important to note that such IDSs must be tailored specifically for the network they are to be used on. For example "an intrusion detection technique in a wired network may be of little use in a wireless network" (Ahmed et al., 2016, p. 20). There are several different types of anomalies, including point anomalies, contextual anomalies and collective anomalies. A point anomaly exists when one datapoint deviates from the rest of the dataset. An example of network traffic containing a point anomaly is when users remotely access a system on the network that they have never connected to previously. Contextual anomalies occur when anomalies are triggered within specific conditions. For example, most users in a company won't use PowerShell, but members of the IT team may use PowerShell as a part of their assigned duties. Finally, a collective anomaly occurs when "a collection of similar data instances behave anomalously with respect to the entire dataset (Ahmed et al., 2016, p. 22).

Different types of network attacks can be detected as different types of anomalies. For example, an attacker using a computer in an HR organizational unit using PowerShell to enumerate users on a domain is a contextual anomaly. Because there are so many complex variations of privilege escalation, it is typically considered a point anomaly, as there is no way to categorize such a diverse group into a contextual or collective anomaly (Ahmed et al., 2016).

Relevant algorithms and techniques. There are many types of ML algorithms, but two categories commonly used for network-based intrusion detection systems are classification

algorithms and statistical analysis algorithms. Relevant classification algorithms include support vector machine and Bayesian networks, which are a type of neural network with fuzzy logic. Relevant methods of statistical analysis include principal component analysis, or PCA, and signal processing. In ML, there are two types of classification: binary and multi-class. Binary classification "outputs one of two mutually exclusive classes", while multi-class classification algorithms are used to solve "solve problems that distinguish among more than two classes" ("Binary Classification"; "Multi-Class Classification"). Binary classification algorithms are useful for determining whether network traffic is malicious or not, and multi-class algorithms are useful for categorizing malicious traffic into types of network attacks.

The Support Vector Machine, or SVM algorithm is a binary classification algorithm which generalizes well, with performance that does not significantly degrade for high-dimensional data (Jha & Ragha, 2013). SVM is also capable of classifying events in near real time, and updating the model dynamically, which are both important for incident response or detection applications (Jha & Ragha, 2013). However, SVM has some disadvantages, including being computationally expensive, restricted to binary classifications, and unable to weight features (Jha & Ragha, 2013). However, many of the aforementioned issues can be resolved by preprocessing and postprocessing (Jha & Ragha, 2013).

Bayesian neural networks are directed acyclic graphs, and each node "reflects the state of the random variable and contains a conditional probability table [which provides] the conditional probability of a node being in a specific state" (Ahmed et al., 2016, p. 23). Additionally, Bayesian networks also represent parent-child dependencies among nodes, so that changes in parent nodes affect child nodes (Ahmed et al., 2016). Like SVM, Bayesian networks

have limitations, including a tendency to produce false positives and an inability to determine unusual or anomalous events from truly malicious events (Ahmed et al., 2016). Implementing the Bayesian network as a replicator neural network by adding an input for additional information can help distinguish between unusual and malicious events, therefore decreasing false positives (Ahmed et al., 2016).

Fuzzy logic is a classification method that relies upon "*degrees of truth* rather than the usual true or false Boolean logic [which] allows an object to belong to different classes at the same time" (Zamani & Movahedi, 2015, p. 7). This flexible classification matches the similarly unclear boundary between normal network traffic and malicious or anomalous traffic. Fuzzy logic allows rules to be weighted, so small changes in an unimportant criterion will not trigger a false positive (Zamani & Movahedi, 2015). Additionally, a high degree of certainty from a rule that is a strong indicator of compromise is more likely to trigger as a true positive (Zamani & Movahedi, 2015). Fuzzy logic has been combined with other tools and techniques such as genetic algorithms to classify attacks into different classes and types "with an overall true positive rate of 98.95% and a false positive rate of 7%" (Zamani & Movahedi, 2015, p. 8).

Statistical analysis techniques can be applied to machine learning models as either preprocessing or postprocessing steps, including principal component analysis (PCA) and signal processing. PCA is a statistical technique used to analyze high dimensional data sets, which are linear combinations of random variables (Ahmed et al., 2016). These linear combinations can be uncorrelated, sorted by variance, or can "have a total variance equal to the variance of the original data" (Ahmed et al., 2016, p. 25). PCA has advantages over traditional methods such as the chi-squared squared test because it does not assume any statistical distribution and it can

reduce the dimensionality of a dataset without loss of information (Ahmed et al., 2016). In an intrusion detection scenario, it is safe to assume that the amount of normal network traffic will be much greater than the amount of malicious network traffic. These assumptions can be used to determine the boundary between normal traffic and malicious anomalies.

Signal processing can be used to detect distributed network traffic anomalies, which are indicators of Distributed Denial of Service attacks and worm propagation, among other malicious network activities (Zonglin, Guangmin, Xingmiao, & Dan, 2009). This technique improves upon PCA-based methods, because it is able to include "the change of correlation between network-wide anomalous space," which provides context for potential anomalous network traffic flows (Zonglin et al., 2009). Figure 2 demonstrates the detection steps using this method. Results from a simulation using this method have demonstrated that detection mechanisms based on network-wide data effectively detect attacks that may not trigger anomaly detection mechanisms focusing on a single connection. (Huang, Chang, & Huang, 2009).

Host-based Applications of Machine Learning

Host-based intrusion detection systems rely on log data to detect abnormal events on a system. This information is typically stored in a Security Information and Event Management system (SIEM) (U.S. Department of Commerce, 2012). In the case of a rule-based IDS, events are generated by the SIEM when specific rules or conditions are met. As is the case with network-based intrusion detection systems, rule-based implementations are only capable of detecting abnormal or potentially malicious events that match pre-existing rules. SIEMs ingest a variety of log types, including operating system, service, application, and command line logs (U.S. Department of Commerce, 2012).

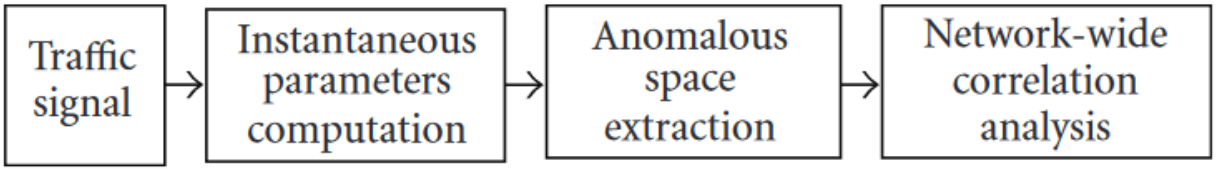

Figure 2. DDoS Detection Steps for the Signal Processing Method. Adapted from *Detecting Distributed Network Traffic Anomaly with Network-Wide Correlation Analysis*, by Zonglin, L., Guangmin, H., Xingmiao, Y., & Dan, Y., 2009, retrieved from doi:10.1155/2009/752818.

Obfuscated command detection. Command obfuscation with tools such as Invoke-Obfuscation, Invoke-DOSfuscation, and Bashfuscator is a technique used by attackers to make detection of malicious commands "even more difficult by adding a layer of indirection between the visible syntax and the final behavior of the command" (Hedge, 2018, para. 2). Obfuscated commands will not match rules created to detect attack techniques, so rule-based systems fall short. Traditionally, obfuscation has been "employed to hide the presence of malware", and serves two purposes: "make it harder to find patterns […] that can easily be detected by defensive software [and] make it harder for reverse engineers and analysts to decipher and fully understand what the malware is doing" (Hedge, 2018, para. 5). Normal users do not obfuscate commands, so obfuscation is an anomaly and a strong indicator of malicious behavior. While it is possible to use regular expressions to detect some obfuscated commands, "it is virtually impossible to develop regular expressions to cover every possible abuse of the commands line" (Hedge, 2018, para. 10).

Rule-based detection leaves significant gaps in detecting obfuscated commands. A convolutional neural network (CNN) and a Gradient Boosted Tree (GBT) were implemented and used in an experiment that used training data from actual host endpoints and obfuscated commands from Invoke-DOSfuscation (Hedge, 2018). The CNN and GBT models yielded similar results, however the CNN was not as accurate as the GBT. To further test the trained

models, complicated but not obfuscated commands as well as custom obfuscated commands (not generated using a tool) were created and classified (Hedge, 2018). These tests differentiated the results given by the two models, and the GBT method was more accurate (Hedge, 2018). However, the author notes that the higher number of misclassifications from the CNN model was "most likely the result of inadequate tuning of the CNN, and not a fundamental shortcoming of the featureless approach" (Hedge, 2018, para. 28).

Machine learning, specifically Gradient Boosted Tree and convolutional neural network algorithms, has been successfully used to detect obfuscated commands that would not be detected by traditional rule-based detection systems. In addition, ML is likely to detect new evolutions of obfuscated commands: "the more comprehensive ML approach is flexible enough to catch new variations in obfuscation, and when gaps are detected, it can usually be handled by adding some well-chosen evader samples to the training set and retraining the model" (Hedge, 2018, para. 29).

Reinforcement learning using sequences of system calls. Reinforcement learning (RL) consists of "a family of algorithms that learn an optimal policy, whose goal is to maximize return when interacting with an environment" ("Reinforcement Learning"). Host-based intrusion detection can make use of sequences of system calls, which "are recorded by monitoring the execution of different processes in the host computer" (Xu & Xie, 2005, p. 997). Each process leaves a trail of system calls, which can be used to distinguish normal behavior from abnormal behavior.

Markov chain models have been applied to the problem of detecting malicious behavior from sequences of system calls. Additionally, a reward function was added, "where

rewards are defined for state transitions [that] improve temporal pattern prediction of intrusion detection" (Xu & Xie, 2005, p. 998). This is an improvement over a purely statistical Markov chain model, as the reward function decreases the computational expense of the algorithm.

A model was trained on a set of 10 abnormal traces and 20 normal traces from "lpr trace data […] of 2766 normal print jobs [and a] single lprcp symbolic link intrusion that consists of 1001 print jobs" (Xu & Xie, 2005, p. 1002). During testing, the Markov chain model was compared to other algorithms, including SVM. The results of this test demonstrated that "temporal difference algorithms in reinforcement learning can be used to realize model-free value function prediction so that accurate intrusion detection can be realized in a computationally efficient way" (Xu & Xie, 2005, p. 1003).

DeepLog. An alternative method to inspecting system call patterns is log analysis. As previously noted, SIEMs are commonly used to aggregate and alert upon log data. DeepLog utilizes a specific type of machine learning:

> [DeepLog uses a] deep neural network model utilizing Long Short-Term Memory (LSTM) to model a system log as a natural language sequence. This allows DeepLog to automatically learn log patterns from normal execution, and detect anomalies when log patterns deviate from the model trained from log data under normal execution. (Du, Li, Zheng, & Sirkumar, 2017, p. 1)

The DeepLog project parses logs, and stores relevant information including "the time elapsed between *e* [the current entry] and its predecessor" (Du et al., 2017, p. 2). There are three primary components of DeepLog's architecture, including "the log key anomaly detection model, the

parameter value anomaly detection model, and the workflow model to diagnose detected anomalies" (Du et. al, 2017, p. 2).

The model is trained using log entries from normal system usage, during which "each log entry is parsed to a log key and a parameter value vector" (Du et. al, 2017, p. 2). Additionally, for each log key, DeepLog "maintains a model for detecting system performance anomalies" (Du et. al, 2017, p. 2). Anomaly detection is accomplished by comparing predictions for normal execution given previous sequences of log entries to current log data (Du et. al, 2017). If the prediction does not match the new data, an anomaly alert is generated.

Although DeepLog assumes integrity of log data, it shows promising results in detecting malicious anomalies, including denial of service attacks, port scanning, and buffer overflow attacks like Blind Return Oriented Programming (BROP) attacks (Du et. al, 2017). DeepLog is also capable of applying user feedback, by supporting "online update/training to its LSTM models, [enabling it] to incorporate and adapt to new execution patterns" (Du et. al, 2017, p. 13). Accurate and timely intrusion detection systems are key for effective incident responses. Machine learning can improve upon traditional rule-based intrusion detection systems for both host-based and network-based applications, specifically in terms of adaptability and lower false positive rates. Because intrusion detection launches the incident response cycle, it is paramount that intrusions be detected accurately and in a timely manner. The machine learning applications discussed can significantly improve the incident detection and response life cycle, by providing the security operations center with accurate, actionable, and timely alerts.

Malware

Machine learning can be applied to malware both defensively and offensively. Defensive applications include malware analysis and detection. Offensive attacks against machine learning models used for malware detection will be discussed later as adversarial machine learning.

Malware can cause significant damage to affected organizations, especially in the case of Ransomware, an increasingly popular subcategory of malware. Ransomware is "a type of malicious software that gains access to files or systems [which are] held hostage using encryption until the victim pays a ransom in exchange for an encryption key" (Groot, 2019, para. 4). If an organization does not have recent backups to restore data from, and a strong form of encryption is used by the malware, the organization has no choice other than pay up or lose critical data. Organizations with critical data and infrastructure are frequently targeted, including hospitals, city transportation agencies and airports, which increases the likelihood of payment over accepting loss of data (Groot, 2019). Malware analysis and detection can help prevent these types of attacks and the subsequent loss of critical data.

Malware Analysis

Malware is incredibly common, and many dedicated analysts spend hours reverse engineering and analyzing malware samples and developing detection mechanisms. Machine learning is used to lighten analyst workload, either through improving malware analysis tools or by partially automating the analysis process. Tools have been created to help reduce analyst workload during the malware analysis process, including StringSifter and tools which provide automated analysis and feature extraction to analysts.

StringSifter. Strings is a Linux program that displays the "printable character sequences that are at least 4 characters long and are followed by an unprintable character" (Linux Man Page). Strings is often the first tool used by analysts to determine readable contents of a malicious binary:

> A binary will often contain strings if it performs operations like printing an error message, connecting to a URL, creating a registry key, or copying a file to a specific location – each of which provide crucial hints that can help drive future analysis. (Tully, Haigh, Gibble, & Sikorski, 2019, para. 1)

Researchers from FireEye, a threat intelligence and security consulting company, created a tool called StringSifter that uses the Gradient Boosted Decision Tree (GBDT) algorithm to place interesting printable characters at the beginning of strings output (Tully et al., 2019).

Training data was generated by running Strings on "over 25 thousand binaries" to create a list of strings output (Tully et al., 2019, para. 8). Using natural language processing, these lists were translated into vectors "containing natural language processing features […] together with domain-specific signals like the presence of indicators of compromise (e.g. file paths, IP addresses, URLs, etc.)" (Tully et al., 2019, para. 8). The model was trained using a weak supervision approach to rank strings according to their relevance to malware analysts (Tully et al., 2019).

After training, the GBDT model was used to predict rankings for "the strings belonging to an input file that was not originally part of the training data" (Tully et al., 2019, para. 10). The sorted output from this demonstration reveals "the potential C2 server and malicious behavior on the host" at the top of the output, followed by "user-related information

[…] still worthy of analysis" and finally "common strings […] that tend to raise no red flags for the malware analyst" (Tully et al., 2019, para. 11). This tool may prove incredibly useful to analysts, as it "could significantly reduce the overall time required to investigate suspected malicious binaries at scale" (Tully et al., 2019, para. 17). The developers plan to improve the model based on analyst feedback. StringSifter is an example of how the power of machine learning can be harnessed to reduce analyst workload during the process of malware analysis.

Automated analysis. Parts of the malware analysis process can be completely automated using machine learning. The three main objectives of malware analysis are detection, similarity analysis, and categorization (Aniello, Baldoni, & Ucci, 2019). Of these three, the largest portion of machine learning applications to malware analysis surrounds the problem of detecting malware (Aniello et al., 2019). Like incident response, malware analysis relies upon detection. If a piece of malware is not detected, it will never be analyzed. Another piece of the malware analysis puzzle that can be addressed using machine learning is categorization. Categorizing malware is an important part of the analysis process, as categorizing a piece of malware allows analysts to gather threat intelligence and understand the nature and purpose of the piece of malware. For example, "if a new malware resembles very closely other binaries that have been analyzed before, then its examination is not a priority" (Aniello et al., 2019, p. 36). Machine learning has been used to categorize malware by type (spyware, ransomware, etc.) and also by malware families (Aniello et al., 2019).

An important part of automating the malware analysis process using machine learning is feature extraction. Features are traditionally extracted by analysts using static and dynamic analysis methods. The majority of machine learning applications rely upon "execution traces

using sandboxes or emulators" which are dynamic analysis methods (Aniello et al., 2019, p. 10). Additionally, some methods extract features using static analysis techniques or a combination of the two, or hybrid analysis (Aniello et al., 2019). Typical features extracted from Windows executables include byte sequences, strings, opcodes, API calls, network activity, file system modifications, and PE file characteristics (Aniello et al., 2019). The automated extraction of these features reduces analyst workload.

Malware Detection

Many antivirus vendors are beginning to incorporate machine learning into endpoint malware detection software. However, such implementations that make use of "a large set of useful features […] come with substantial development costs" (Coull & Gardner, 2018, para. 1). Therefore, alternative options are being explored, particularly convolutional neural networks. FireEye has created a CNN model that detects malware "simply by looking at the raw bytes of Windows Portable Executable (PE) files" with accuracy that can rival traditional (feature-based) machine learning models (Coull & Gardner, 2018, para. 2). Further analysis of this model has revealed "a number of important aspects of the classifier's operation, weaknesses, and strengths", including the importance of "ASCII-based import features", low-level instruction features which "capture specific behaviors […] associated with malware" and end-to-end features that "map to common manually-derived features" of feature-based machine learning applications (Coull & Gardner, 2018, para. 24). Because convolutional neural networks do not require feature engineering and other time-consuming efforts, "deep learning offers a promising path toward sustainable, cutting-edge malware classification, [however] significant

improvements will be necessary to create a viable real-world solution" (Coull & Gardner, 2018, para. 25).

Nazca: detection of large-scale malware infections. Nazca is a graph-based project that is designed to detect "infections in large scale networks" by collecting and analyzing network traffic from across the entire network as a whole instead of inspecting specific connections separately (Invernizzi et al., 2014, p. 2). To do this, Nazca "looks at the telltale signs of the malicious network infrastructures that orchestrate these malware installation[s] that become apparent when looking at the collective traffic produced by many users in a large network" (Invernizzi et al., 2014, p. 1). Because Nazca observes patterns in network traffic, specifically HTTP requests, it does not suffer from problems that plague rule-based systems: gaps between rules and obfuscation (Invernizzi et al., 2014).

Nazca makes use of a three step detection process, targeting the download and installation phase of malware installation, because "from the network point of view, such connections are hardly suspicious," but "when considering many malware downloads together – performed by different hosts, but related to a single campaign – a malware distribution infrastructure becomes visible" (Invernizzi et al., 2014, p. 2). The first step is network-wide monitoring of HTTP requests, including metadata extraction (Invernizzi et al., 2014). Secondly, connections are classified as either suspicious or not suspicious, and features of suspicious connection streams are extracted and stored in real time. For this project, suspicious traffic is any HTTP traffic that "initiates a download of a file whose MIME type is not in [the] whitelist" which consists of MIME types of "popular formats for images, videos, audio, and other innocuous files" (Invernizzi et al., 2014, p. 3). The final step is the aggregation of suspicious

connections using a graph, with two goals in mind: “find related, malicious activity so as to reduce potential false positives and focus the attention on the most significant infection events” (Invernizzi et al., 2014, p. 2).

A machine learning classifier is used during the third and final step to find content delivery networks (CDNs) and distinguish between malicious and legitimate CDNs (Invernizzi et al., 2014). This classifier uses several features, including domain co-location, number of unique top-level domain names, number of matching URI paths and matching file names, and served file types. Then, a “malicious neighborhood graph” is generated, which shows the relationship between infected hosts, command and control (C2) servers, and servers which delivered the initial infection (Invernizzi et al., 2014, p. 6).

Nazca was trained and tested using “nine days of traffic” from an ISP, with two days of traffic reserved for training data and the rest allocated as testing data (Invernizzi et al., 2014, p. 8). The results of the test demonstrated “59.81% precision and 90.14% recall on the malicious class, and 99.69% precision and 98.14% recall on the benign class”, and many misclassifications occurred due to insufficient data about potentially malicious infrastructure in the training set (Invernizzi et al., 2014, p. 12). Some notable findings include the discovery that ISP caching servers were contributing to the delivery of malware after C2 infrastructure was taken down, and the detection of previously unidentified malware samples (Invernizzi et al., 2014). These findings are significant because they demonstrate the ability of machine learning to detect and classify previously undetected malware attacks.

Adversarial Machine Learning

Adversarial machine learning (AML) is the "art of studying how to break and secure machine learning models" and is where offensive security and machine learning collide (Chebbi, 2018, Adverserial machine learning section). AML can be used to defeat both traditional rule-based detection systems and detection systems that make use of machine learning models. These evasion tactics can be used by malware and active attackers. Many attacks revolve around finding areas where the model is overfit or underfit to training data, and adjusting malicious code or attack vectors to fit through the cracks (Chebbi, 2018). Underfitting and overfitting are terms used to describe discrepancies between the function that the model learns and the function that most accurately defines the dataset (Chebbi, 2018). Other attacks involve poisoning the training data to shift the model or classifier to alter the classification of malicious activities (Chebbi, 2018).

Cylance Antivirus

Security researchers were able to reverse engineer Cylance's machine learning model at the core of their endpoint protection (antivirus) product, and ultimately found "a bias enabling a universal bypass" (Ashkenazy & Zini, 2019, para. 13). First, the researchers enabled verbose logging to gain insight into how the model scores and flags potentially malicious files (Ashkenazy & Zini, 2019). Using the information from the log file and reverse engineered source code, features used by the Cylance model were enumerated, and heavily weighted features were investigated. The researchers were able to identify a bias in the model and discovered a universal bypass. This was achieved by adding strings from whitelisted programs to the end of malicious executables (Ashkenazy & Zini, 2019).

In testing, “almost all of [the malicious] samples have changed from the most evil file on the planet, to your friendly neighborhood file” (Ashkenazy & Zini, 2019, para. 72). Cylance scores files from -1000 (malicious) to 1000 (safe), and the bypass technique was able to change the average score of the malicious files from -920 to 630 (Ashkenazy & Zini, 2019). If the “secret sauce” strings were added to the end of the malicious file multiple times, the average score rose to 750, resulting in “88.54% of our malicious files” being marked as benign (Ashkenazy & Zini, 2019, para. 74). This example demonstrates that machine learning models and security software must be designed with security in mind in order to avoid the success of adversarial machine learning techniques.

Conclusion

Machine learning has many current and potential applications to threat intelligence, intrusion detection, and malware analysis and detection. ML can automate some of the tedious and repetitive analysis involved in attack attribution, aid in the detection of new intrusions and network attacks, and power tools that help automate the malware analysis process. However, machine learning is not a silver bullet. As machine learning grows as a new technology in the cybersecurity space, adversarial machine learning grows with it. Security researchers are constantly discovering new ways to subvert security products and evade detection.

References

Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications, 60,* 19-31. doi:10.1016/j.jnca.2015.11.016

Aniello, L., Baldoni, R., & Ucci, D. (2019). Survey of machine learning techniques for malware analysis. *Computers & Security, 81,* 123-147. doi:10.1016/j.cose.2018.11.001

Ashkenazy, A., & Zini, S. (2019, September 7). Cylance, I kill you [Web log post]. Retrieved from https://skylightcyber.com/2019/07/18/cylance-i-kill-you/

Berninger, M. (2018, November 16). APTinder: An optimized approach for finding that perfect APT match [Video file]. Retrieved from https://www.youtube.com/watch?v=zMdHGY53VEw

Berninger, M. (2019, March 12). Going ATOMIC: clustering and associating attacker activity at scale [Web log post]. Retrieved from https://www.fireeye.com/blog/threat-research/2019/03/clustering-and-associating-attacker-activity-at-scale.html

Bianco, D. (2014, January 17). The pryamid of pain [Web log post]. Retrieved from http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html

"Binary Classification" (2020). In *Machine Learning Glossary*. Retrieved from https://developers.google.com/machine-learning/glossary#binary-classification

Bromiley, M. (2016). *Threat intelligence: what it is, and how to use it effectively*. Retrieved from https://www.sans.org/reading-room/whitepapers/analyst/threat-intelligence-is-effectively-37282

Chebbi, C. (2018). *Mastering machine learning for penetration testing* [O'Reilly Media version]. Retrieved from https://learning.oreilly.com/library/view/mastering-machine-learning/9781788997409/

Coull, S., & Gardner, C. (2018, December 13). What are deep neural networks learning about malware [Web log post]. Retrieved from https://www.fireeye.com/blog/threat-research/2018/12/what-are-deep-neural-networks-learning-about-malware.html

Du, M., Li, F., Zheng, G., & Sirkumar, V. (2017, October). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas,* 1285-1298. doi:10.1145/3133956.3134015

Google. (2019). *Identifying Good Problems for ML*. Retrieved from https://developers.google.com/machine-learning/problem-framing/good

Groot, J. D. (2019, October 24). A history of ransomware attacks: The biggest and worst ransomware attacks of all time [Web log post]. Retrieved from https://digitalguardian.com/blog/history-ransomware-attacks-biggest-and-worst-ransomware-attacks-all-time

Hedge, V. (2018, November 29). Obfuscated command line detection using machine learning [Web log post]. Retrieved from https://www.fireeye.com/blog/threat-research/2018/11/obfuscated-command-line-detection-using-machine-learning.html

Huang, C., Chang, R., & Huang, P. (2009). Signal processing applications in network intrusion detection systems. *EURASPI Journal on Advances in Signal Processing*. doi:10.1155/2009/527689

Invernizzi, L., Lee, S.-J., Miskovic, S., Mellia, M., Torres, R., Kruegel, C., . . . Vigna, G. (2014, February). Nazca: detecting malware distribution in large-scale networks. In L. Bauer (Chair), *Network and Distributed System Security symposium.* Symposium conducted at the meeting of the NDSS, San Diego, CA. doi:10.14722/ndss.2014.23269

Jha, J., & Ragha, L. (2013). Intrusion detection system using support vector machine. *International Journal of Applied Information Systems*, 25-30. Retrived from https://research.ijais.org/icwac/number3/icwac1342.pdf

Lele, A. (2014). Asymmetric warfare: A state vs non-state conflict. *OASIS*, *20*, 97-111. Retrieved from https://revistas.uexternado.edu.co/index.php/oasis/article/view/4011

Linux Man Page. Strings(1). Retrieved from https://linux.die.net/man/1/strings

McMillan, R. (2013). *Definition: Threat intelligence*. Retrieved from https://www.gartner.com/en/documents/2487216

"Multi-Class Classification" (2020). In *Machine Learning Glossary*. Retrieved from https://developers.google.com/machine-learning/glossary#multi-class-classification

NETCAP: A framework for secure and scalable network traffic analysis. (2019). Retrieved from https://github.com/dreadl0ck/netcap

Polyakov, A. (2018, October 4). Machine learning for cybersecurity 101 [Web log post]. Retrieved from https://towardsdatascience.com/machine-learning-for-cybersecurity-101-7822b802790b

"Reinforcement Learning" (2020). In *Machine Learning Glossary*. Retrieved from https://developers.google.com/machine-learning/glossary#reinforcement-learning-rl

Soman, B. (2019, August 8). From unstructured data to actionable intelligence: Using machine learning for threat intelligence [Web log post]. Retrieved from https://www.microsoft.com/security/blog/2019/08/08/from-unstructured-data-to-actionable-intelligence-using-machine-learning-for-threat-intelligence/

Tully, P., Haigh, M., Gibble, J., & Sikorski, M. (2019, May 29). Learning to rank strings output for speedier malware analysis [Web log post]. Retrieved from https://www.fireeye.com/blog/threat-research/2019/05/learning-to-rank-strings-output-for-speedier-malware-analysis.html

U.S. Department of Commerce, National Institute of Standards and Technology. (2012). *Computer security incident handling guide* (NIST Publication No. 800-61). doi:10.6028/NIST.SP.800-61r2

Wang, V. & Stamper, G. (2002). Asymmetric war? Implications for China's information warfare strategies. *American Asian Review*, *20*(4), 167-207. Retrieved from http://ezproxy.liberty.edu/login?url=https://search.proquest.com/docview/211469069?accountid=12085

Xu, X., & Xie, T. (2005). A reinforcement learning approach for host-based intrusion detection using sequences of system calls. *Proceedings of the International Conference on Intelligent Computing, China,* 1021-1029. Retrieved from https://link-springer-com.ezproxy.liberty.edu/content/pdf/10.1007%2F11538059.pdf

Zamani, M., & Movahedi, M. (2015). *Machine learning techniques for intrusion detection.* Retrieved from https://arxiv.org/pdf/1312.2177v2.pdf

Zonglin, L., Guangmin, H., Xingmiao, Y., & Dan, Y. (2009). Detecting distributed network traffic anomaly with network-wide correlation analysis. *EURASIP Journal on Advances in Signal Processing*. doi:10.1155/2009/752818