Big Data Analytics for Intrusion Detection: An Overview

Luís Dias
CINAMIL, Academia Militar, Instituto Universitário Militar, Portugal

Miguel Correia
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

ABSTRACT

Intrusion detection has become a problem of big data, with a semantic gap between vast security data sources and real knowledge about threats. The use of machine learning (ML) algorithms on big data is already successfully applied in other domains. Hence, this approach is promising for dealing also with cyber security big data problem. Rather than relying on human analysts to create signatures or classify huge volumes of data, ML can be used. ML allows the implementation of advanced algorithms to extract information from data using behavioral analysis or to find hidden correlations. However, the adversarial setting and the dynamism of the cyber threat landscape stand as difficult challenges when applying ML. The next generation security information and event management (SIEM) systems should provide security monitoring with the means for automation, orchestration and real-time contextual threat awareness. However, recent research shows that further work is needed to fulfill these requirements. This chapter presents a survey on recent work on big data analytics for intrusion detection.

Keywords: Security Analytics, Intrusion Detection, SIEM, Cyber Security, Big Data, Machine Learning, Data Mining,

INTRODUCTION

Over the past two decades *network intrusion detection systems* (NIDSs) have been intensively investigated in academia and deployed by industry (Debar et al., 1999). More recently, intrusion detection has become a big data problem because of the growing volume and complexity of data necessary to unveil increasingly sophisticated cyberattacks. The *security information and event management* (SIEM) systems adopted during the last decade show limitations when processing with *big data*, even more in relation to extracting the information it can provide. Therefore, new techniques to handle high volumes of security-relevant data, along with *machine learning* (ML) approaches, are receiving much attention from researchers. This chapter presents an overview of the state-of-the-art regarding this subject.
The Cloud Security Alliance (CSA) suggested that *intrusion detection systems* (IDSs) have been going through three stages of evolution corresponding to three types of security tools (Cárdenas et al., 2013):

- IDS: able to detect well-known attacks efficiently using signatures (misuse detection) and to unveil unknown attacks at the cost of high false alarm rates (anomaly detection);

- SIEM: collect and manage security-relevant data from different devices in a network (e.g., firewalls, IDSs, and authentication servers), providing increased network security visibility by aggregating and filtering alarms, while providing actionable information to security analysts;

- 2nd generation SIEM: the next generation, that should be able to handle and take the best from big data, reducing the time for correlating, consolidating, and contextualizing even more diverse and unstructured security data (e.g., global threat intelligence, blogs, and forums); they should be able to provide long-term storage for correlating historical data as well as for forensic purposes.

The European Agency for Network and Information Security (ENISA) stated that the next generation SIEMs are the most promising domains of application for big data (ENISA, 2015). According to recent surveys from the SANS Institute, most organizations are just starting to evolve from traditional SIEMs to more advanced forms of security analytics and big data processing (Shackleford, 2015, 2016). In fact, the industry developments in the area led Gartner to start publishing market guides for user and entity behavior analytics (UEBA) (Litan, 2015). While recent UEBA technologies target specific security use cases (e.g., insider threats), typical SIEM technologies provide comprehensive rosters of all security events which are also important for compliance requirements. Recent guides from Gartner (Bussa et al., 2016; Kavanagh et al., 2018) state that "Vendors with more mature SIEM technologies are moving swiftly to incorporate big data technology and analytics to better support detection and response", revealing the tendency of moving towards 2nd generation SIEMs.

The focus of this survey is on state-of-the-art techniques that can contribute for such next generation SIEMs, and on the challenges of big data and ML applied to cybersecurity analytics. There are a few related surveys available in the literature, none with this focus. Buczak & Guven (2016) analyze papers that use different ML techniques in the cybersecurity domain; although interesting, most of the experiments of those studies were done with datasets that date back to 1999 and that do not represent the actual cybersecurity landscape. Bhuyan et al. (2014) provide a comprehensive overview of network anomaly detection methods. Zuech et al. (2015) review works considering the problem of big heterogeneous data associated with intrusion detection. While this last work touches topics similar to this chapter, it lacks a comprehensive study on recent techniques that tackle the problem of extracting useful information from such volumes of data.

When selecting the papers to investigate, we prioritized recent work – less than 5 years old – and those that report experiments with real-world data. The rationale is that older approaches and approaches evaluated with synthetic datasets (e.g., KDD99) may be inadequate to detect real/recent attacks. Furthermore, we restricted the focus to papers published in major conferences/journals, with few exceptions.

The studied papers were grouped into four categories: *cluster and classify* (Yen et al., 2013; Gonçalves et al., 2015; Veeramachaneni et al., 2016); *graph models* (Oprea et al., 2015; Nagaraja, 2014; Milajerdi et al., 2019); *real-time log analytics* (Cinque et al., 2017; Du et al., 2017; Mirsky et al., 2018); and *frameworks and context* (Stroeh et al., 2013; Giura & Wang, 2012). The first, consists in detectors based on clustering and classification. These are appealing approaches because of their ability to detect zero-day attacks, i.e., attacks that were previously unknown. The second is about using graph models to represent network flows and events, to extract features which can detect, e.g., botnets (Nagaraja, 2014) and advanced persistent threats (Giura & Wang, 2012). The third group contain proposals to process log streams in real-time, either by using of entropy measurements (Cinque et al., 2017) or by modeling system logs as a natural language sequence (Du et al., 2017). The fourth category explores studies that focus on providing high-level frameworks to address the needs of future security monitoring.

The remainder of this chapter is organized as follows: Section 2 explains basic concepts of big data and ML regarding intrusion detection; Section 3 presents the main challenges; Section 4 presents the state-of-the-art techniques; finally, in Section 5 the researchers draw their conclusions.

BIG DATA AND MACHINE LEARNING

Clearly intrusion detection is facing a big data challenge, whether there is a single source of events or a major architecture that aggregates events from heterogeneous sources (Zuech et al., 2015). To tackle this problem, one must deploy big data handlers, i.e., components that process such data. Furthermore, since handling such high volumes of data is beyond human capabilities, the use of ML to extract useful information from large and multi-dimensional data, seems to be the only way.
This section gives a brief overview of the application of some big data handler technologies and ML techniques in the cybersecurity domain.

Big Data Handlers

Hadoop is an Apache project (White, 2009) that includes open source implementations of the MapReduce distributed programming framework (Dean & Ghemawat, 2004) and of the Google file system (Ghemawat et al., 2003), among other components. In the MapReduce paradigm, the input of a computational job is split into parts that are processed (mapped) by a set of worker nodes; the results of this first processing are themselves processed (reduced) by workers. The Hadoop file-system provides the parallelism for the MapReduce worker nodes to be executed on a large cluster of commodity machines. The runtime system handles scheduling and failures of program executions across the machines.
In the cybersecurity domain, a Hadoop cluster seems to be a suitable option for long-term event storage due to processing, fault tolerance and scalability provided. Hence, it can be used to do analytics to infer trends, create knowledge about old attacks, and feed real-time systems with discovered behavioral signatures, providing a tool for forensic investigations and for contextual awareness. Traditional Hadoop MapReduce does batch processing, so it is unsuitable for data streams processing. It is also not suitable for iterative algorithms that depend on previous computed values (Cloud Security Alliance, 2014). However, some years ago Hadoop evolved to version 2.0 that provides more flexibility, decoupling MapReduce from the filesystem (HDFS), and introducing a resource management layer called YARN. This allowed the appearance of real-time analytics platforms on top of Hadoop 2.0. Apache Spark is the most popular at the moment, as it provides in-memory cluster computing in opposition to disk-based processing (Zaharia et al., 2016; Meng et al., 2016). Spark has shown impressive results when computing machine-learning iterative algorithms, running up to 100x faster than Hadoop MapReduce. More information on big data technologies can be found in (Cloud Security Alliance, 2014).
In what concerns the cybersecurity industry, it is gradually tackling the big data challenges leveraged by SIEMs. Some vendors implemented ElasticSearch (Kononenko et al., 2014) in their software to improve search times. Although it seems to help for faster queries, it does not solve the storage problem. Several vendors are offering tight integration of their SIEMs with big data analytics platforms (e.g., QRadar (Wuest, 2014) or Splunk (Splunk, n.d.)). They provide bi-directional communication, either to collect external big data feeds, or to export old logs. Some companies are exporting older logs from SIEMs (e.g., 1-month old logs) to a Hadoop cluster for longer-term storage, analysis and reporting. However, this option seems costly and has several drawbacks, such as: impact of transferring such big data on normal operations; context in the SIEM is lost; yet another system to manage. The big players in this market are enhancing their new platforms with scalability and big data in mind (Kavanagh et al., 2018). As an example, IBM launched QRadar Data Node which allows to add nodes to the actual SIEM deployment transparently for the end-user. Despite traditional SIEM systems are evolving to more advanced analytics and big data processing, new players with this setup as concept arise, such as Securonix (Securonix, n.d.) or Exabeam (Exabeam, n.d.). The soundest features refereed by these state-of-the-art solutions are: the use of a ML approach for *user and entity behaviour analytics* (UEBA) (Litan, 2015) and big data ready architectures for scalable data collection and processing.
In what concerns academic research, a few works that used Hadoop technologies for cybersecurity are: Gonçalves et al. (2015) implemented an outlier detector using a Hadoop cluster; Lee & Lee (2013) experimented and measured the performance of a DDoS detector using Hadoop technologies; Cheon &

Choe (2013) used a Hadoop cluster to do parallel processing with logs collected from several Snort IDS sensors. The results of these studies using Hadoop for IDS, are constrained by the fact that MapReduce is not suited to real-time processing, and often the evaluation is inconclusive. Clearly more research and experimentation are needed in this area. Some big data platforms that combine several Hadoop tools along with Spark, visualization and ML tools, are emerging and can help to easily deploy an environment for cybersecurity research. For instance, Apache Spot (Apache Spot, n.d.) and Apache Metron (Apache Metron, n.d.) are fully dedicated to cybersecurity, although more generic platforms such as Elastic Stack (ELK, n.d.) (previously ELK Stack) or MapR Hadoop distribution (MapR, n.d.) can also be valuable in this field.

ML Approaches

Samuel (1959), a pioneer of ML, defined it as a "field of study that gives computers the ability to learn without being explicitly programmed". ML is often concerned with classification and prediction based on previously learned training data without a specific analytic solution (ideal target function) to the problem; instead, it searches for an approximation to that ideal solution (optimization of a hypothesis function). Typically, the more data you give to a ML classification algorithm, the more accurate it becomes in providing correct outputs. Data mining is a related field of study that uses techniques similar related to those of ML, but often with different purposes. Data mining is usually carried out in a specific situation, on a particular dataset, with a goal in mind, and the result has to be comprehensible for humans. Both systems search through data to look for patterns. However, ML uses data to adjust program actions accordingly.
Next an introduction to the most popular ML categories (supervised and unsupervised learning) is given. Furthermore, some considerations on feature engineering and model evaluation are provided.

Supervised Learning

ML typically involves a dataset that is used for training, i.e., for preparing the ML scheme for later operation. Supervised learning is characterized by the use of a training dataset containing labeled data. The labelling is often done by humans, but the size of the labeled data tends to be small and limited, whereas unlabeled, real-world, data can be much larger or even unlimited.
Classification is a common form of supervised learning. Classification is the act of mapping input data to classes (or labels). Classification has been much used in the security domain for spam detection, and is particularly relevant in this chapter as it is at the core of most forms of malicious activity detection that are based on ML. The input of a classification algorithm is a set of discriminating features (e.g., count of words that tend to appear in spam e-mail), which are used to predict the class that the input item shall be assigned to (e.g., the e-mail is spam or ham, i.e., not spam). The system outputs if the class of the item (e.g., the e-mail is likely to be spam or ham). To accomplish this, the classifier has to be trained with large datasets of both spam and ham e-mails. Some example classification algorithms include: artificial neural networks (ANNs), decision trees, support vector machines, and nave Bayes.
Regression is a second form of supervised learning. The main difference between classification and regression is that classification is concerned with assigning items to classes, whereas regression outputs (predicts) numeric values. Exampled of regression algorithms are: linear regression, polynomial regression, radial basis functions, MARS, and multi-linear interpolation.

Unsupervised Learning

Unsupervised learning aim to find hidden structures on input data. This class of algorithms are characterized by not requiring labeled data. In this case, the only condition for the input data is the presence of representative features that can be further used for knowledge extraction.

Clustering is the most popular method of unsupervised learning. As a simple example, consider that a cyberattack has a behavioral signature (e.g., a combination of steps dispersed in various security logs). First, one selects the distinctive features that security experts would look upon to find indicators of compromise (e.g., volume of uploaded traffic, number of unpopular domains contacted, etc.). Second, the features are extracted from the logs according to a time-window and an entity (e.g., five minutes of logs belonging to the same user or IP), creating a vector of extracted features for each entity. Third, the computed vectors are given as input to a clustering algorithm. Finally, the clustering algorithm groups different entities – i.e., returns clusters – according to their behavioral signature. Often, larger clusters contain entities that have normal behavior, and the smaller clusters contain outliers.
Typical algorithms for unsupervised learning include K-means clustering, gaussian mixture modelling, spectral clustering, hierarchical clustering, principal component analysis, and independent component analysis.

Feature Selection

Feature engineering is an important aspect of ML. Feature engineering is the process of defining a set of features that allow improving the predictive power (performance) of a ML algorithm to solve a certain problem. The rationale is that well-chosen features can improve the results of a classifier, or any other supervised on unsupervised scheme. Intuitively, features like "presence of the word pill" or "presence of the word money" are better to predict if an e-mail is span that the feature "text font used".
According to Domingos (2012) "some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used". If features are independent from each other and each of the feature correlates well with the class, learning is easy. On the other hand, if the class is a complex function of the features, learning can be impossible. All features should be relevant for the decision process. Domingos also says that feature engineering is the most difficult part when doing a ML project. This happens because it is domain-specific, while learning models are general purpose and mastered enough.

Model Evaluation and Tuning

Tuning and evaluating a ML model is a research topic per se. The two processes require a dataset that is used for that purpose. A common solution consists in breaking that dataset in two parts: one that is used for training, the other for tuning and evaluation. However, if the training dataset is not representative of the other (or of real-world data), the predictive power may be low. A related problem overfitting: the model may be excellent in predicting with the training data, but be too specific, providing bad results with real-world data.
Cross-validation works better because it does not depend on that splitting of the dataset in two. k-fold cross-validation consists in breaking the dataset in k parts, and iteratively train the model with 1 part and test it with the other $k - 1$. A common value is $k = 10$. The result is the average of the k iterations. The cross-validation error provides a good estimation of the model error on new data.
Evaluation of ML models is usually based on a set of metrics. For classifiers, or specifically for intrusion detection, metrics are based on the notions of true positive (TP) and true negative (TN). A TP means that a hit (a detection) is indeed a hit, whereas a TN means a no hit is indeed a not hit. False positives (FP) and false negatives (FN) mean wrong positives and negatives, respectively. Common metrics are accuracy (overall performance of the model), precision (how good the positive predictions are), recall or sensitivity (coverage of the positive sample), false positive rate, and false negative rate. There are many others.

CYBERSECURITY CHALLENGES

The ability to process and correlate high volumes of data is a prerequisite for security analytics solutions. This capacity then allows doing analytical queries, long-term forensic analysis of the data, automated and rapid response, reports, flexible dashboard views and, ultimately, contextual threat awareness. Another requirement is diversity of sources of data, as this is a requirement for performing advanced correlations and support contextual awareness. ML plays a crucial role in helping find correlations that we do not know about and to discover trends and behavioral patterns. Rather than searching attack evidence after being attacked, the predictive capability can be leveraged with ML, allowing to stop threats based on current and past behavior. Furthermore, obtaining more data on which to perform root cause and forensics analysis after an attack happens, should feed and improve an active learning system (Gonçalves et al., 2015; Veeramachaneni et al., 2016).

This section discusses the most challenging problems and difficulties of deploying ML techniques in the cybersecurity domain. In other domains, such as product recommendation, natural language translation or fraud detection, ML plays a fundamental role. That is not yet the case in cybersecurity, except for spam detection. A ML spam detection method was initially proposed by Graham (2002) using Bayesian filtering trained with large datasets of both spam and ham.

Despite extensive research, both academic and industrial, anomaly intrusion detection based on ML is not widely deployed. Apparently, the reason is that most approaches do not fulfil operational requirements such as accuracy. Actually, the great majority of actual IDS deployments use almost exclusively traditional misuse-based detection. However, in the last few years, ML is being used as a buzzword by several security companies. Some examples of ML used by the cybersecurity industry are: Cylance (Cylance, n.d.) claims to be the next-generation anti-virus using ML techniques; WEBROOT (WEBROOT, n.d.) focus on cloud-based threat intelligence services; Lancope's StealthWatch (Lancope, n.d.) infers suspicious movements from Netflow data; DARKTRACE (DARKTRACE, n.d.) claims to leverage artificial intelligence to detect threats that others miss; IBM adapted their Watson project (IBM, n.d.) to be used in cybersecurity, aiming to learn cybersecurity language and process unstructured data from blogs, articles, videos, reports, and alerts, besides other data. ML is also a core capability in the recent product category that Gartner called UEBA (Bussa et al., 2016), which perform behavior analytics to systems, devices, applications, etc. UEBA solutions are still divided on the type of data (e.g., security devices logs, network packets) they ingest and analyze, thus, in general, these solutions are seen as a complement to SIEMs. In fact, most are being integrated in major SIEMs.

Next, we point out the main challenges of ML applied to cybersecurity.

Attack Evolution

The cybersecurity landscape is extremely dynamic with new threats appearing every day. The democratization of cyber-threats increases the global impact along with organized cyber-crime which leverages highly sophisticated techniques. Furthermore, with the growth of technologies and concepts like software as a service (SaaS), cloud infrastructures, or bring your own device (BYOD), enterprise borders are not well-defined anymore. There is an evident need to collect every single data-point that can be security relevant and apply ML to extract information from disparate sources.

Most approaches of ML focus on using outlier detection to find novel attacks. This is commonly accepted as the best way to tackle the problem of attack evolution. As Sommer & Paxson (2010) refers, this approach has to be carefully considered since it raises several problems: high rates of false alarms; attack-free data for the training process is hard to get; attacker can gradually teach the system to accept malicious activity (as Klein & Peter (2015) and W. Xu & Evans (2016) showed). According to (Sommer & Paxson, 2010), the "strength of machine learning tools is finding activity that is similar to something previously seen", although not being necessary to detail that activity up front (as in misuse detection). Furthermore, newer attacks can be significantly different from old ones (both technically and modus operandi). This is a crucial aspect since the feature selection process using an old dataset can generate a feature set that may be different from the one needed to detect future attacks.

In summary, the adversarial setting changes how researchers have to approach the network intrusion detection problem. This adversarial setting seems to be the root cause for the difficulty (at least in the next few years) in removing human experts from the loop of an intrusion detection system.

Heterogeneous Data Sources

At the early days of intrusion detection, security analysts observed logs in order to detect malicious activity or abnormal behaviors, which is something hard to do manually. Moreover, looking for events individually in each security device or software (e.g., firewall, IDS, anti-virus) gives only a narrow vision and does not provide global security awareness. Looking to individual data sources is also not enough to respond fast in case of a breach or to detect multi-step attacks, even less to predict future attacks.
SIEMs aggregate security related logs coming from a myriad of devices (e.g., that have different security functions, or the same function but different vendor). They structure that data in formats that support correlation, generation of alerts, and visualization by the analyst. SIEMs have performance issues due to big data volumes and the need to parse, structure and store that data in databases. This is especially true with large enterprise networks where current SIEM technologies sometimes cannot parse this overwhelming quantity of data into actionable information. Moreover, the parsing and normalization phase compose themselves a big challenge in what concerns timestamp normalization, crucial in global networks and fundamental before performing data correlation.
Nevertheless, as IT environments grow and are increasingly complex, it is commonly accepted that collecting data from every security-relevant source, both traditional (e.g., firewall, database access, web access) and non-traditional (e.g., social media, blog posts, external threat feeds, domain registration) is necessary for a security system that can predict and detect new attacks. The more data we have, the more information we should extract.

Big Data

According to ENISA (ENISA, 2015), “one of the advantages of NoSQL databases, is that they can store such data in a format that is scalable and at the same time allow to create queries and understand the data better. In reality big data offers as added value not only the correlation between unstructured data, but also SIEM scalability”. In short, most definitions of big data analytics present it as a set of tools and techniques to store and process large datasets efficiently for that purpose, i.e., for analytics. Big data analytics handles very large datasets (Volume), provides extremely fast insertion (Velocity), manipulates multiple data types (Variety), provides complex data analysis capabilities and requires distributed parallel processing.
In the cyberspace domain, the increasing volume of security log data that enterprises are keeping, be it for compliance or incident response reasons, do not allow SIEM platforms to properly handle such volumes of data (at least in the desired time window). According to Nassar et al. (2013), even one single security event source such as network traffic data, can become a big data challenge. As an example, consider performing deep packet inspection on 1Gbps traffic. Hence, according to Zuech et al. (2015), correlating security events across multiple heterogeneous data sources, where each can pose its own big data challenges is much more complex than performing intrusion detection on a single homogeneous big data source.
Another example, reported in (Cárdenas et al., 2013), comes from HP that at the time of report generated “1 trillion events per day or roughly 12 million events per second”. This is clearly a big data problem, that required a large effort just to store that data. The company concluded that they “must move to a scenario where more data leads to better analytics and more actionable information.”
At the 2012 RSA conference, Zions Bancorporation showed how Hadoop and analytics can power better security intelligence. They announced that a query to their traditional SIEMs (1 month of data) could take up to an hour. In their new Hadoop system, running queries with the Apache Hive data warehouse software Apache Hive (n.d.), they got the same results in about one minute. Although this happened some years ago, it is still a reality in several enterprises running traditional SIEMs.

In the cybersecurity domain, big data handlers are necessary. They have to exist either by means of new technologies (e.g., Hadoop), data reduction techniques (e.g., raw packet data transformed into Netflow) and dimensionality reduction (e.g., obtaining a set of uncorrelated principal variables). Li et al. (2012) recommended cloud-based enterprise security monitoring (ESM) as a natural solution to handle the big data problem. ESM has the SIEM as core component, enhanced with business context information to provide intelligence for actions and decision making. Their approach envisions a cross boundary information security management and the fusion of enterprise-wide security data, willing to achieve global correlation and intelligence. It is known that there are companies that decided to adopt a managed SIEM off premises, mainly to avoid scalability issues. However, this is a sensitive decision regarding privacy and business-wise.

Privacy and Integrity

As one collect sources of both structured and unstructured data that can be security relevant, privacy (both in the sense of confidentiality and protection of personal data) and integrity concerns arises. This leads to the problem of big data for security versus securing big data (Cárdenas et al., 2013). This is part of any big data problem in general. The trustworthiness of each source has to be assured to protect against tampering and adversarial ML techniques. In other hand, one has to restrict access to big data analytic solutions, so that it does not allow users to invade others privacy. CSA provided generic guidance on how to protect big data (Cloud Security Alliance, 2013).

Dataset Limitations

Intrusion detection can be considered to be a binomial classification problem with two classes: normal or suspicious. ML classification algorithms must be trained with examples of both classes, as this is essential for detection performance. However, considering that we are looking for novel attacks, one cannot train the system with data about these attacks. Unlike other areas of research, there is a significant lack of assessment tools for evaluating IDS. The most used datasets (KDD19, DARPA98), are over 15 years old, thus with very limited interest for current research. On recent surveys on ML applied to cybersecurity (e.g., (Azad & Jha, 2013), (Buczak & Guven, 2016)), it was verified that even recent studies still use the old datasets, which is disappointing. As it was described earlier, feature selection process using an old dataset generates features set that is inadequate to detect actual attacks (new techniques and modus operandi).
Real-world or simulated data are alternatives to the old datasets. However, most organizations are hesitant or legally constrained to disclose such data that can contain sensitive information (e.g., vulnerabilities, privacy concerns, etc.). Furthermore, those datasets can be unreliable or suffer from high anonymization. The problem with lab simulated traffic is the possible unrealistic properties which can result in inaccurate evaluations.
An alternative to generate close to real-world datasets can be done using Honeypots or Honeynets. As an example, Nagaraja (2014) built a Honeynet to get ground truth for specifically injected malware. Although building a honeynet can be costly, it seems a good approach for specific purpose testing (e.g., test accuracy on detecting known botnets).
Finally, very recently new datasets have been appearing, mostly created by the Canadian Institute for Cybersecurity from University of New Brunswick (Canadian Institute for Cybersecurity, n.d.). They have been producing several datasets, real and synthetic, to be used for intrusion detection evaluation. Turcotte et al. (2018) also provided a new dataset containing host and network events from The Los Alamos National Laboratory enterprise network.

Detection Performance

IDS detection performance is of extreme importance because of the costs that a misclassification can represent. In the cybersecurity domain, a false positive lead to an analyst spending time to examine an incident that does not represent malicious activity. If those false alarms happen frequently, the analyst will end up lacking confidence in the system. On the other hand, if we are in the presence of a false negative and the alert is not triggered, the incident can be very disruptive and damage the organization. For better understanding the problem, in the case of spam detector, the false positives can be costly and should be avoided at the cost of having some false negatives with a fairly low cost. Although new variations of spam can easily bypass the system, it seems a necessary trade-off. Again, in IDS, if we are seeking to find novel attacks, this trade-off does not seem appropriate.

Semantic Gap

To get actionable information from a ML-based intrusion detection solution is a considerable challenge. The fact that abnormal does not necessarily means malicious, implies that the system must provide distinctive information so that the analyst can easily understand what the cause and impact of an incident is (contextual information), or at least, what does it mean (e.g., connection burst, unpopular domain contacted, etc.). Typically, outlier detection approaches just report what diverges from normal, whether it is malicious or not. Furthermore, things get more complicated in a scenario where network traffic is extremely unpredictable, varying vastly at different times of day and with new technologies (such as new software or hardware).

Real-Time/Streaming Capabilities

To detect and prevent attacks at an early stage, one must be able to perform close to real-time intrusion detection. The idea is that even if the event count is large, events have to be processed in a short window of time. Specific types of attributes may need to be applied during processing (e.g., universal time stamping). Furthermore, security data analytics platforms should allow for fast contextualization (sorting data into consistent fields and formats for analysis) and fast metadata extraction during processing (Stroeh et al., 2013).

RECENT STUDIES

This section gives an overview of the state-of-the-art in cybersecurity analytics research, with a focus on big data handling and the use of ML. When selecting the papers to investigate, the recent ones (less than five years) and those that made experiments with real-world data were the priority. The rationale for choosing recent papers is that older solutions may be inadequate to detect actual attacks. Furthermore, we restricted our attention mostly to papers published in selective conferences/journals.
The studied papers were grouped in four categories: *cluster and classify* (Yen et al., 2013; Gonçalves et al., 2015; Veeramachaneni et al., 2016), *graph models* (Oprea et al., 2015; Nagaraja, 2014; Milajerdi et al., 2019), *real-time log analytics* (Cinque et al., 2017; Du et al., 2017; Mirsky et al., 2018) and *Frameworks and context* (Stroeh et al., 2013; Giura & Wang, 2012). The first, focuses in generic outlier detection methods; the second on the use of graph models for specific attacks (botnets, advanced persistent threats – APTs, and failure prediction); the third, contains proposals to process log streams in real-time; the last, explores methods for normalization and event aggregation to achieve contextual awareness. *Table 1* presents a summary of the papers that were analyzed.

Table 1 - Summary of *the analyzed papers*

Paper	Published	Year	Approach	Dataset	Achievements
Cluster and Classify					

Beehive (Yen et al., 2013)	ACSAC	2013	- Clustering: K-means after PCA;
			- Unique/rare pattern ⇒ outlier
				Webproxy;
				DHCP;
				AV;VPN.
					Generic outlier detector.

Big Data analytics(...) (Gonçalves et al., 2015)	Trustcom	2015	- Integrate threat intelligence features; - Clustering (Expectation-Maximization); - Classification (Naive Bayes); - Analyst must check if labels are correct, if needed he can correct, improving classifier accuracy.	DHCP; 802.1x; FW.	Classify after clustering to automate the process.
AI2 (Veeramachaneni et al., 2016)	BigData Sec.	2016	- Oriented to web applications. - Ensemble of unsupervised-learning methods: density-based, matrix-decomposition, neural networks; mix individual scores to get a rank; - Top ranked events are labeled by analysts to seed supervised model.	FW; WebServer.	Ensemble of unsupervised models. Rank alerts.
Graph Models					
(...)Mining Large-Scale Log Data (Oprea et al., 2015)	DSN	2015	- Method: use belief propagation to unveil suspicious hosts/domains (C&C like); - Classify suspects using linear regression.	DNS; Webproxy.	Detection of DGA domains. Unveil malicious and unknown domains.
Botyacc (Nagaraja, 2014)	ESORICS	2014	- Detection of P2P botnets; - Collect Netflow ->graph-model - edge partition; - Markov diffusion process - Random walks; - Infer comparing to Honeynet ground truth.	Netflow; SFlow.	Aprox.97% accuracy to unveil known botnets. FAR (0.1%)
HOLMES (Milajerdi et al., 2019)	S&P	2018	- Map audit logs to APT stage; - Leverage MITRE ATT&CK ;	Windows ETW; Linux auditdw.	Real-time APT detection; Highlevel graph with attack steps.
Real-Time Log Analytics					
Entropy-Based(...) (Cinque et al., 2017)	ICDSN	2017	- Anomaly detector; - Measure log entropy and find deviations from - Baseline.	logs of an Air Traffic Control System.	New technique for real-time processing; Can be adapted to process other data types.

DeepLog(Du et al., 2017)	CCS	2017	- Anomaly detector;
			- Model logs as natural language;
			- Deep Neural Network model using LSTM to learn log patterns and find deviations
				HDFS;
				OpenStack;
				Emulated: VAST 2011;
					General purpose framework for online log anomaly detection; Extensive evaluation with promising results.

Kitsune(Mirsky et al., 2018)	NDSS	2018	- Anomaly detector NIDS; neural networks; - Ensemble AutoEncoders; - Online feature extraction framework.	IP camera network; IoT network.	General purpose framework for online log anomaly detection.
Frameworks and Context					
Correlation of Security Events(...) (Stroeh et al., 2013)	JISA	2013	- Collect snort events ->Normalize (ext. IDMEF); - Group events from same attack and generate meta-events; - Classify (SVM, Bayesian networks, Decision tree) meta events ->Attack or false positive	DARPA99 Scan of the month.	Bridge Semantic gap by aggregating Snort events into real alerts.
(...) (Giura & Wang, 2012)	ASE	2012	- Framework proposal. - Use all recorded events (not only security alerts). - All logs have potential to be useful. - Define planes (physical, user, network, application). - Map events to targets (CEO, servers) - Contexts construction (correlate planes) ->Alert	—	Guidelines for a scalable intrusion detection framework

Cluster and Classify

This group of works includes outlier detectors based on clustering algorithms. Beehive (Yen et al., 2013) uses only the clustering model. The other two (Gonçalves et al., 2015; Veeramachaneni et al., 2016) perform both clustering and classification: event data is given as input to the clustering algorithm; the clusters are then labeled by humans and the classification model is trained with the labeled examples (e.g., normal, malicious).

Beehive: Large-Scale Log Analysis for Detecting Suspicious Activity in Enterprise Networks

Yen et al. (2013) proposed one of the first systems exploring the challenges of big data security analytics for real-world log data. It was tested using two weeks of EMC real logs, with an average of 1.4 billion messages per day (about one terabyte of security logs). Clearly a big data problem.

Beehive uses several different logs: web proxy logs (outbound connections), DHCP logs (for IP to machine correlation), Windows domain controller logs (authentication attempts), antivirus logs (scan results), and VPN server logs (remote connections). Those raw logs were stored at a commercial SIEM. Even so, logs were noisy because of different timestamps (even from different time zones), out of order arrival, different identifiers (IP, usernames, hostnames). The goal was to detect threats in two different scenarios: the first involves a host that is infected or at imminent risk (e.g., contacting C&C servers[1], exfiltrating data, malware); the second concerns policy violations (e.g., peer-to-peer sharing, adult content). To overcome these problems, the Beehive approach considers three layers:

- Parse, filter and normalize log data – First define how to normalize the timestamps of all log entries to UTC. Second, define how to bind IP addresses to hosts (using DHCP logs). Finally, normalize these attributes of every log entry.
- Feature extraction from the normalized data – Features were selected by security experts, observing known malware and policy violations behavior. Features where grouped in four types: Destination-based (e.g., new destinations, unpopular destinations), Host-based (new user-agent string), Policy-based (e.g., blocked domain, challenged domain) and traffic-based (e.g., connection spikes or bursts). Daily, for each host, a feature vector is created.
- Perform clustering to identify outliers

The authors of Beehive used real data, so there was no ground-truth, i.e., no information about which attacks were caught in the dataset. To tackle this problem of lack of ground-truth, they applied an unsupervised learning approach (K-means clustering). Hosts that behave similarly (e.g., all hosts in a department, all users with same role) are represented within large clusters, while outliers (with unique behavioral patterns) should appear as smaller clusters (few hosts). They also applied Principal Component Analysis to remove dependencies between features.

In what concerns evaluation and ability to detect security incidents, the Beehive project uncovered 784 security incidents (25% malware related, 39% policy violation, 35% automated unknown software) whereas the enterprise existing security tools only detected 8 of these incidents. It is also noteworthy to refer that by applying techniques such as custom white-listing (popular domains removal), they reduce the data to be processed by 74%.

Beehive represents a remarkable step in outlier detection methods in the context we consider. However, some limitations are remarked to better perceive future work direction: the need for manual analysis to classify the meaning of the unveiled outliers; the evaluation of the system not providing exact information about detection performance (e.g., false positive rate); the analytics is done in a daily basis using processed data from the previous day. The batch processing took more than five hours just for feature extraction of 160GB (after reduction) of logs corresponding to one day.

Big Data Analytics for Detecting Host Misbehavior in Large Logs

Gonçalves et al. (2015) present an approach that "combines data mining and both supervised and unsupervised ML in order to make the detection process as automatic as possible, without having to instruct

[1] botnet command and control servers

the system about how entities misbehave". In this case the real-world data was obtained at a major telecommunication company. They considered logs from DHCP (IP to host binding), authentication servers (how often user log in and sessions flows) and perimeter firewalls (security alerts, actions, ports, source and destination).

At first a normalization is performed to clean and combine logs (with similar techniques to Beehive). Next is the feature selection and extraction phase. Finally, the system performs clustering to group entities with similar behavior and then the security analyst must manually classify them as behaving normally or misbehaving. Afterwards, the normal period of execution begins. The overall process is conducted using logs within a time period (e.g., 1 day of logs).

The clustering algorithm (Expectation-Maximization) outputs clusters of similar behaving entities to be classified according to the previously trained classification model. The analyst should always review and confirm classifier output such that the classification model gets updated and more accurate.

Regarding feature selection, they remark that it was a long process to look for features that are potentially able to separate good from bad behavior (e.g., number of sessions, number of admin login tries, unpopular domains). In total they used 34 individual features. It is noteworthy that they also used external feeds, namely threat intelligence from AlienVault (bad reputation IPs), Internet Storm Center (malicious subnets), Sitevet list of worse Autonomous Systems, and CleanTalk (spam activity). To be able to process big data, the authors performed feature extraction using Hadoop MapReduce.

A distinctive improvement in relation to Beehive, is the use of a classification mechanism to minimize the effort of manually labeling clusters. They used Naive Bayes "as it converges quicker than other models". Their approach requires intensive human labor at a first phase to create the classification model (manually labeling a training set), afterwards the analyst has only to monitor and check if results are wrong or not and update the model accordingly (over time the detection performance should increase).

The most relevant limitations of this system are the same as Beehive's. However, there were improvements regarding semi-automatic cluster labeling and the implementation of feature extraction using MapReduce, which potentially can reduce human intervention and processing times. The Hadoop MapReduce framework is intended to execute the jobs in batch; this fact constrains using their scheme for blocking attacks.

AI2: Training a Big Data Machine to Defend

Veeramachaneni et al. (2016) developed a system that follows the same basic ideas of (Yen et al., 2013; Gonçalves et al., 2015). However, there are some distinctive aspects. The chosen feature set is focused on finding web application attacks (e.g., account takeover or new account fraud). During the unsupervised-learning phase, they fuse together three different methods (density-based, matrix decomposition, and ANNs) by projecting individual scores into probabilities. The system tags the events with a rank as a function of the computed probabilities, so that the analyst can prioritize actions. If an event is highly scored by all algorithms it will be considered, highly ranked and shown to the analyst as such.

Regarding the data sources, they used three months of real logs (3.6 billion log lines) from a webscale platform, providing web and firewall logs. To tackle the big data problem of feature extraction, they divided this process in two: activity tracking and activity aggregation. In short, as the system absorbs the generated log stream, it identifies the entities (e.g., IP, user) in each log line and updates the corresponding activity records (designed for efficient feature extraction) which are stored in a temporal window (one minute). To improve performance, they overlap time-scopes and maintain different time-windows (minute, hourly, daily, weekly). Finally, the feature extraction process can be done in any time interval (limited to 24 hours) just by computing on the aggregated activity records. They selected the features (24 in total), which seem to represent indicators an expert would use to investigate an attack to the web platform (e.g., logins on the last 24 hours, time from login to checkout, location, etc.).

They made a simulation of the system using 3 months of logs, claiming that their "results show that the system learns to defend against unseen attacks" getting more accurate over-time along with analysts'

feedback. The detection rate shows an increasing trend improving 3.4x and reducing false positives by more than 5x when compared to state-of-the-art unsupervised anomaly detector.
As in the previous systems, automation is hard to achieve. The need for qualified human analysts is fundamental for adequate learning process. However, this research brought new insights on outlier detection mechanism applied to web application. Furthermore, it introduced a ranking mechanism for prioritization of outlier events, with improved confidence and accuracy based on an outlier detection ensemble (to compensate individual model bias). Although performance evaluation is not detailed in their study, the authors believe this choice can have high impact in performance.

Graph Models

The studies in this section, use graphs to represent network flows and events to extract features which can unveil APTs (Oprea et al., 2015; Milajerdi et al., 2019) and botnets (Nagaraja, 2014).

Detection of Early-Stage Enterprise Infection by Mining Large-Scale Log Data

Oprea et al. (2015) proposed a framework to detect early-stage APTs based on belief propagation (Yedidia et al., 2003) from graph theory. An APT is a multi-stage attack that targets a specific organization, often involves sophisticated malware, and adapts to the target environment while maintaining a low profile. The authors leveraged the following common infection patterns to unveil APTs: visits to several domains in a short period (redirection techniques to hide the attacker); connections to C&C servers; use of HTTP(S) to circumvent firewalls; and use of domains related to an attack, sharing location/IP space, same hosts, same time access. The goal was to unveil suspicious communications by internal hosts (indicator of an APT).
The authors used DNS and web proxy logs. A communication graph, either based on DNS or Webproxy logs, is built by modeling domains and hosts as graph vertices, and connections as graph edges. Since they only look to outbound requests, the resulting representation is a bipartite graph having hosts (one side) contacting domains (other side). To keep the graph manageable, they applied a reduction technique that consists in restricting the graph to what they call rare domains and to the hosts contacting them. Rare domains are those never seen before and contacted by a small number of hosts. To further reduce data, they removed popular domains and used domain folding to speed up data processing.
In this framework, belief propagation is used to unveil communities of domains that are likely part of campaigns and have similar features. They start by seeding the algorithm with known malicious domains or hosts, either obtained by security operation center (SOC) hints from previous incidents or by previously computed suspicious domains (C&C-like behavior). The algorithm iteratively computes scores to obtain other rare domains contacted by compromised hosts. The score computation is performed based on the similarity with C&C like behavior or with other suspicious domains from previous iterations (sum of weighted features). Finally, they use a supervised model to classify the domains as suspicious or not suspicious given a threshold. The algorithm outputs suspicious domains and respective hosts connections (with suspiciousness level).
The features set they choose aims to characterize C&C like outbound connections. They extracted the timestamps (to infer periodic communications), hostname, destination domain, user-agent (UA) string (infer rare UA), and web-referrer (infer automated connections).
The approach was evaluated on two different datasets. The first contained two months (1.15TB) of DNS records from the Los Alamos National Lab containing 20 independent and simulated APT-like infections. Their technique achieved 98.33% true positive rate with low false positive rate. The second dataset contained two months of web proxy logs (38.41TB) collected from a large enterprise. They confirmed that 289 of 375 detected domains were malicious and that 98 domains were new discoveries (not recognized by state-of-the-art products). Another remarkable achievement is that they were able to unveil domains generated by malware (using a domain generation algorithm, DGA, to hide the attacker's infrastructure) even before registration (allowing preventive measures).

Some limitations of this approach are that attackers can escape detection by: randomizing C&C connection timings; using standard UA strings; communicating directly to IPs instead of domains. However, these limitations are also a problem to the attackers: randomizing connections would make campaign orchestration harder; UA strings are typically used by malware to send information; using IPs unveils their infrastructure.
Overall, this approach is a complement to existing tools and can be used as a feed to correlate with other sources.

Botyacc: Unified P2P Botnet Detection Using Behavioral and Graph Analysis

Nagaraja (2014) proposed a novel technique to detect peer-to-peer (P2P) botnets. Botmasters started using P2P communications to create resilience and anonymity by decentralizing command throughout the network. The paper's approach consists in leveraging the design characteristic of P2P communication by using Markovian diffusion processes over network flows represented in a graph model. The P2P botnet graph design is distinguishable from those of other communications in network. Finding patterns in connectivity (who talks to whom) and traffic flows (who talks like whom) is the key idea.
The system architecture is composed by four components: monitor, the component which collects flows either being vantage points (e.g., routers with Netflow) or hosts (sflows); aggregator, that receives and collects flows from monitors, and computes an algorithm to separate bot candidates from legitimate traffic; honeynet, to create ground-truth for specifically injected malware and to train the inference mechanism; inference mechanism, which uses as input the traffic traces from both the aggregator and the honeynet to unveil the botnet.
An overview of the approach is the following. The communication graph is given by $G = (V,E)$ where V is the set of hosts and E is the set of flows. Consider the subgraph $Gb = (H,M)$ where H is the set of bots (infected hosts) and M is the set of botnet flows. Notice that $Gb \subseteq G$, and $H \subseteq V$, and $M \subseteq E$. This is an edge-partitioning problem from a graph-theoretic view. Each edge $e \in E(G)$, models a traffic flow represented by a k-dimensional vector. The inference algorithm is a stochastic diffusion process over similar edges. To infer edges similarity, random walks on graphs are performed. The novelty of this walk is that the choice of the next step (outgoing edge) is based on its similarity with the previous (incoming edge), hence, the walk has a bias towards similar flows. To infer similarity, the algorithm computes the Euclidean distance where each flow is represented by a vector of scalar elements defining Cartesian coordinates.
The evaluation of the approach was done using real-world dataset from a university gateway. The captured traffic (corresponding to one month) was injected with popular P2P botnets (Zeus, Spyeye and Miner). The malware traffic flows were obtained using a testbed with 25 servers in a test network, which was connected to the internet. The detection rates ranged from 95% to 99% with very low false positive rate ($\approx$ 0.1%).
This approach represents a different method with interesting results. It seems to be suitable to find known P2P botnets, but accuracy on novel ones is hard to assess.

HOLMES: Real-Time APT Detection Through Correlation of Suspicious Information Flows

Milajerdi et al. (2019) proposed HOLMES, a system with the goal of detecting activities that are part of APT campaigns. Their system collects audit data from hosts (e.g., Linux auditd or Windows ETW data) and other alerts available in the enterprise (e.g., from IDSs) to produce a detection signal. At a high-level, the techniques they developed leverage the correlation between suspicious information flows (e.g., files, processes) that arise during an attack campaign. The system design enables the generation of alerts semantically close to activity steps of APT attackers: the kill chain. The cyber kill chain models APT lifecycle phases: from early reconnaissance to the goal of data exfiltration.
For example, the internal reconnaissance step depends on successful initial compromise and establishment of a foothold. Both lateral movement and exfiltration phases use data gathered during the reconnaissance

phase. Correlating APT steps from low-level events and information flow originates the emerging kill chain used by a particular APT actor. To bridge the semantic gap between low-level system-call events and high-level kill chain stages, the authors leveraged the use of MITREs ATT&CK framework[2] (MITRE, n.d.; Strom et al., 2018) to map audit logs to APT stages. MITREs ATT&CK framework describes close to 200 behavioral patterns – tactics, techniques, and procedures (TTPs) – observed in the wild, which can be captured on audit data.
To assist analyst interpretation, HOLMES enables the construction of a high-level scenario graph (HSG) which refer to TTPs. The edges of the graph represent the information flow among TTPs and their entities. The HSG summarizes the attacker's actions in real-time.
The authors evaluated the system using a dataset provided by the DARPA Transparent Computing program, corresponding to streams of audit data of 7 computers (Windows and Linux based) with injected attacks. They published the audit data streams into the Kafka stream processing server and performed real-time analysis and detection by using a previous developed system called SLEUTH (Hossain et al., 2017). Attacks constituted less than 0.001% of the audit data volume. HOLMES was able to distinguish between APT campaigns injected in the dataset and benign activities.
The main limitation of HOLMES is that it assumes the auditing system and the OS kernel are reliable, as well as that the initial attack must originate outside the enterprise, using means such as remote network access. Furthermore, the evaluation was done with a limited number of audit log sources.

Real-Time Log Analytics

This section refers three recent works that provide new methods to capture security-relevant information from logs in real-time. The first (Cinque et al., 2017) uses entropy measurements to infer deviations from the baseline; the second is inspired in natural language processing and interprets logs as elements of a sequence that follows grammar rules (Du et al., 2017); the third (Mirsky et al., 2018) uses an ensemble of ANNs (autoencoders) to differentiate between normal and abnormal traffic patterns.

Entropy-Based Security Analytics: Measurements from a Critical Information System

Cinque et al. (2017) proposed a method to find abnormal behavior within textual, heterogeneous, runtime log streams. They used an entropy-based approach, which has no assumptions on the structure of underlying log entries. The goal is to detect anomalies (i.e., unexpected activity) that should be followed up by analysts, by monitoring entropy for deviations above certain thresholds.
Their technique comprises two main tasks named log entropy sampling and collector. Sampling is a periodic task that computes log entropy of new entries within a predefined time window. The entropy is calculated based on the count of terms in logs, i.e., of sequences of characters separated by a whitespace. The collector acquires the most recent log entropy observations for all logs of the system. It stores the observations in vectors (one vector per log source) where each position of the vector corresponds to a log stream entropy observation.
They evaluated the system measuring logs from a real-world Air Traffic Control information system. To overcome the big data problem, they deployed Apache Storm (Apache Storm, n.d.) as data analytics framework. Experiments showed that entropy-based methods can be a valuable complement to security analytics solutions.

Deeplog: Anomaly Detection and Diagnosis from System Logs through Deep Learning

[2] The name of the framework is indeed ATT&CK but it comes from adversarial tactics, techniques, and common knowledge.

Du et al. (2017) proposed a deep neural network model that uses long short-term memory (LSTM) to model a log as a natural language sequence. The key insight is that log entries are the result of structured code. Hence, logs can be seen as a sequence of events of a structured language.
In short, as a log entry arrives it is parsed into a log key (e.g., type of event) and a parameter value vector (e.g., event content). An entry is labeled as an anomaly if either its log keys or its parameter values vector is predicted as being abnormal. In a first phase, DeepLog has to learn what good behavior is, being trained with log data from normal execution. After this initial training, it can detect anomalies (i.e., log patterns deviated from normal) over incoming log streams. In addition, the authors proposed a technique to incrementally update the model in runtime, adapting it to new log patterns over time, although this requires to incorporate user feedback. Moreover, DeepLog allow the analyst to perform root cause analysis, by constructing workflows from the underlying system logs.
Although DeepLog is an innovative method, at the training stage it assumes having good behavior logs, which is something hard to assure.

Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection

Mirsky et al. (2018) propose what they call a plug-and-play NIDS. This NIDS can learn to detect attacks on the local network, without supervision and online, in an efficient manner. The authors point out that Kitsune aims to overcome the shortcomings of typical approaches using ANNs as NIDS. Some of these shortcomings are: the need for a labeled dataset; the fact that most supervised learning approaches are limited to known attacks and need a strong CPU to train a model; run the newly trained models is only possible after transferring updates to organizations NIDS. In light of these challenges, Kitsune aims to be a lightweight NIDS based on an ANN, online, using unsupervised learning. It is designed to run on network routers in real-time.
In sum, the Kitsune framework is composed by a packet capturer and parser (e.g., meta information of the packet), a feature extractor (e.g., statistics of the channel), and a feature mapper to map features into a bigger set to be processed by the anomaly detector. Kitsunes core algorithm for anomaly detection, KitNET, uses an ensemble of small ANNs – called autoencoders – to differentiate between normal and abnormal traffic patterns. The feature extraction framework efficiently tracks the patterns of every network channel in an online fashion. To accomplish this, the framework uses damped incremental statistics maintained in an hash table.
To evaluate Kitsune, they made experiments on an operational IP camera surveillance network and IoT network deploying several attacks. They achieved good results running Kitsune on a simple Raspberry PI. As in the previous studies, if the network is already contaminated, Kitsune is arguably inefficient.

Frameworks and Context

This last group includes two studies that focus on providing high-level frameworks to address the future needs of security analytics.

Using Large Scale Distributed Computing to Unveil Advanced Persistent Threats

Giura & Wang (2012) propose a framework for detecting APTs and a generalized method that can be implemented in any organization. They introduced a conceptual attack model, which they named attack pyramid, starting from the notion of attack tree. The attack goal is at the top of the pyramid, while the lateral planes represent environments where the attack can occur. The environments (planes) are as follows: physical, network, user, application, other. From the base to the top of the pyramid, it is represented the attack time-line evolving the following stages: reconnaissance, delivery, exploitation, operation, data collection and exfiltration (goal). The unfolded attack pyramid version, gives the possible paths (and vulnerable elements), naturally traversing multiple planes, towards the goal.

To determine long-term events across planes and infer if they are part of a coherent attack is a hard task: an APT is not detected by observing a specific event; it has to be unveiled by event correlation. Hence, they tackle the problem by exploring new large-scale distributed systems as a processing platform, allowing to cover more attack paths and more security-relevant sources. A prototype of multiprocessing implementation with MapReduce is presented. The idea is to correlate in parallel all the relevant events across all the pyramid planes into contexts. They recommend collecting data in all planes such as "physical" location of hardware or users, "users" importance (e.g., admins, CFO, etc.), security and "network" devices logs, "application" logs (web-server, critical end-points).
In summary, the method consists in profiling potential APT targets (servers, important employees, data center location) and group collected events into contexts (set of events corresponding to a target). For each target there is an attack pyramid. Different pyramids can share one or more planes and each plane is defined by a set of events. The big data ready large-scale distributed computing should Map events to targets and construct contexts, then Reduce to target and alert (detection mechanisms).
Apparently, the proposed system was not implemented, and only preliminary results are presented in the paper.

An Approach to the Correlation of Security Events based on Machine Learning Techniques

Stroeh et al. (2013) proposed a ML method to tackle the excess of false positives of an IDS. They do not use network traffic or other sources; they solely use IDS logs (e.g., Snort).
Their approach has two major steps. The first step is to normalize the incoming events (from the IDS) according to an extended version of the Intrusion Detection Message Exchange Format (IDMEF). After normalization, events from the same attack are aggregated into meta-events. The second step consists in classifying meta-events into attacks (alerts) or false positives.
The alerts aggregate events that represent an action (e.g., failed login), a condition (e.g., port unreachable) or suspiciousness (deduction from previous two). Furthermore, the alerts belong to some class of attack (e.g., DoS, probing, remote access, etc.). Each alert can be aggregated into high-level objects named meta-alerts. The meta-alert as a bit-array field where each bit of that field, represents an alert-type, and the more alert-type count the more likely to have an attack. The internal product of two arrays is used to infer the similarity between two meta-alerts. Hence, this field is essential for the classification process using supervised models (SVM, Bayesian networks and decision tree). They choose different models to infer which one was more accurate and concluded that Bayesian networks had the best results.
To conclude, it is noteworthy that they evaluated their approach using the DARPA99 dataset and data from the Scan-of-the-month Honeynet. They achieved data to information ratios in the order of 2,1% and 50% respectively.

CONCLUSION

Both the cybersecurity industry and academic research have been engaging the big data challenge faced by security analytics and intrusion detection. Major SIEM vendors are evolving to distributed computing technologies and are integrating behavioral analytics (using ML) in their solutions. These new capabilities materialize the emergence of a new generation SIEMs, that the CSA designated 2nd generation SIEMs.
ML algorithms are being used to deal with big data in several domains, but in cybersecurity there is a sense that there is much more to be done. In the cybersecurity domain, the use of ML is being more successful on specific use cases (e.g., high value assets, track privileged users) in opposition to more generic goals (e.g., generic outlier detector), which may lead to high rates of false alarms. When applying ML to cybersecurity, the greatest challenge is the fact that we are dealing with human adversaries that are to some extent unpredictable. One of the consequences of this adversarial setting is the evolution of attacks, which

does not allow to obtain the necessary ground-truth to a fully automated system that can predict new attacks accurately.
Security experts are still necessary in the intrusion detection cycle. A stand-alone and fully automated solution seems hard to achieve. Even so, handling such high volume of data produced in the security domain, is surely beyond human capabilities, and the use of ML to extract useful information from large and multi-dimensional data, seems to be the best complement to achieve a close to real-time contextual threat awareness.

ACKNOWLEDGMENT

This research was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2019 (INESC-ID), by the Portuguese Army (CINAMIL), and by the European Commission under grant agreement number 830892 (SPARTA).

REFERENCES

Apache Hive (n.d.). Retrieved from https://hive.apache.org/.

Apache Metron (n.d.). Retrieved from http://metron.apache.org/.

Apache Spot (n.d.). Retrieved from http://spot.incubator.apache.org/.

Apache Storm (n.d.). Retrieved from http://storm.apache.org/.

Azad, C., & Jha, V. K. (2013). Data mining in intrusion detection: a comparative study of methods, types and data sets. International Journal of Information Technology and Computer Science, 5, 75.

Bhuyan, M., Bhattacharyya, D., & Kalita, J. (2014). Network anomaly detection: Methods, systems and tools. IEEE Communications Surveys Tutorials, 16, 303–336.

Buczak, A., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys Tutorials, 18, 1153–1176.

Bussa, T., Litan, A., & Phillips, T. (2016). Gartner Inc. Market guide for user and entity behavior analytics.

Canadian Institute for Cybersecurity (n.d.). Retrieved from http://www.unb.ca/research/iscx/dataset/index.html.

Cárdenas, A., Manadhata, P., & Rajan, S. (2013). Big data analytics for security intelligence. Cloud Security Alliance.

Cheon, J., & Choe, T.-Y. (2013). Distributed processing of snort alert log using Hadoop. International Journal of Engineering and Technology, 5, 2685–2690.

Cinque, M., Corte, R. D., & Pecchia, A. (2017). Entropy-based security analytics: Measurements from a critical information system. In 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (pp. 379–390). doi:10.1109/DSN.2017.39.

Cloud Security Alliance (2013). Expanded top ten big data security and privacy challenges.

Cloud Security Alliance (2014). Big data taxonomy.

Cylance (n.d.). Retrieved from https://www.cylance.com/.

DARKTRACE (n.d.). Retrieved from https://www.darktrace.com.

Dean, J., & Ghemawat, S. (2004). MapReduce: simplified data processing on large clusters. In Proceedings of the 6th USENIX Symposium on Operating Systems Design and Implementation (pp. 137–150).

Debar, H., Dacier, M., & Wespi, A. (1999). Towards a taxonomy of intrusion detection systems. Computer Networks, 31, 805–822.

Domingos, P. (2012). A few useful things to know about machine learning. Communications of the ACM, 55, 78–87.

Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). Deeplog: Anomaly detection and diagnosis from system logs through deep learning, .

ELK (n.d.). Retrieved from https://www.elastic.co/.

ENISA (2015). Big data security - good practices and recommendations on the security of big data systems.

Exabeam (n.d.). Retrieved from https://www.exabeam.com/.

Ghemawat, S., Gobioff, H., & Leung, S.-T. (2003). The Google file system. ACM SIGOPS Operating Systems Review, 37, 29.

Giura, P., & Wang, W. (2012). Using large scale distributed computing to unveil advanced persistent threats. Science Journal, 1, 93–105.

Gonçalves, D., Bota, J., & Correia, M. (2015). Big data analytics for detecting host misbehavior in large logs. In Proceedings of the 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications.

Graham, P. (2002). A plan for spam. Retrieved from http://www.paulgraham.com/spam.html.

Hossain, M. N., Milajerdi, S. M., Wang, J., Eshete, B., Gjomemo, R., Sekar, R., Stoller, S. D., & Venkatakrishnan, V. (2017). Sleuth: Real-time attack scenario reconstruction from cots audit data. In Proc. USENIX Secur. (pp. 487–504).

IBM (n.d.). Cognitive security with watson. Retrieved Jan 25, 2019, from https://www.ibm.com/security/artificial-intelligence.

Kavanagh, K. M., Sadowski, G., & Bussa, T. (2018). Magic quadrant for security information and event management.

Klein, B., & Peter, R. (2015). Defeating machine learning. Blackhat Briefings USA.

Kononenko, O., Baysal, O., Holmes, R., & Godfrey, M. W. (2014). Mining modern repositories with ElasticSearch. In Proceedings of the 11th IEEE Working Conference on Mining Software Repositories.

Lancope (n.d.). Retrieved from https://www.lancope.com.

Lee, Y., & Lee, Y. (2013). Toward scalable internet traffic measurement and analysis with Hadoop. ACM SIGCOMM Computer Communication Review, 43, 5–13.

Li, Y., Liu, Y., & Zhang, H. (2012). Cross-boundary enterprise security monitoring. In Proceedings of the IEEE International Conference on Computational Problem-Solving (pp. 127–136).

Litan, A. (2015). Gartner Inc. Market guide for user and entity behavior analytics.

MapR (n.d.). Retrieved from https://www.mapr.com.

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S. et al. (2016). Mllib: Machine learning in apache spark. The Journal of Machine Learning Research, 17, 1235–1241.

Milajerdi, S. M., Gjomemo, R., Eshete, B., Sekar, R., & Venkatakrishnan, V. (2019). Holmes: Realtime apt detection through correlation of suspicious information flows. In Proceedings of the 40th IEEE Symposium on Security and Privacy (S&P).

Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: An ensemble of autoencoders for online network intrusion detection. In Proceedings of the Network and Distributed System Security Symposium (NDSS) 2018. doi:10.14722/ndss.2018.23204.

MITRE (n.d.). Adversarial tactics, techniques and common knowledge. Retrieved from https://attack.mitre.org/.

Nagaraja, S. (2014). Botyacc: Unified p2p botnet detection using behavioural analysis and graph analysis. In European Symposium on Research in Computer Security (pp. 439–456).

Nassar, M., al Bouna, B., & Malluhi, Q. (2013). Secure outsourcing of network flow data analysis. In Proceedings of the 2013 IEEE International Congress on Big Data (pp. 431–432).

Oprea, A., Li, Z., Yen, T.-F., Chin, S. H., & Alrwais, S. (2015). Detection of early-stage enterprise infection by mining large-scale log data. In Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (pp. 45–56).

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. IBM Journal of research and development, 3, 210–229.

Securonix (n.d.). Retrieved from https://www.securonix.com/.

Shackleford, D. (2015). Analytics and Intelligence Survey. paper SANS Institute – InfoSec Reading Room.

Shackleford, D. (2016). Security Analytics Survey. paper SANS Institute – InfoSec Reading Room.

Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In Proceedings of the 30th IEEE Symposium on Security and Privacy (pp. 305–316).

Splunk (n.d.). Retrieved from https://www.splunk.com.

Stroeh, K., Madeira, E. R. M., & Goldenstein, S. K. (2013). An approach to the correlation of security events based on machine learning techniques. Journal of Internet Services and Applications, 4,7.

Strom, B. E., Applebaum, A., Miller, D. P., Nickels, K. C., Pennington, A. G., & Thomas, C. B. (2018). MITRE ATT&CK: Design and philosophy.

Turcotte, M. J. M., Kent, A. D., & Hash, C. (2018). Unified host and network data set. In Data Science for Cyber-Security chapter Chapter 1. (pp. 1–22).

Veeramachaneni, K., Arnaldo, I., Cuesta-Infante, A., Korrapati, V., Bassias, C., & Li, K. (2016). AI2: Training a big data machine to defend. In Proceedings of the 2nd IEEE International Conference on Big Data Security on Cloud.

W. Xu, Y. Q., & Evans, D. (2016). Automatically evading classifiers. In Proceedings of the 2016 Network and Distributed Systems Symposium.

WEBROOT (n.d.). Retrieved from http://www.webroot.com/.

White, T. (2009). Hadoop: The Definitive Guide. O'Reilly.

Wuest, B. (2014). Integrating QRadar with Hadoop. White Paper from IBM.

Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2003). Understanding belief propagation and its generalizations. Exploring artificial intelligence in the new millennium, 8, 236–239.

Yen, T.-F., Oprea, A., Onarlioglu, K., Leetham, T., Robertson, W., Juels, A., & Kirda, E. (2013). Beehive: large-scale log analysis for detecting suspicious activity in enterprise networks. In Proceedings of the 29th ACM Annual Computer Security Applications Conference.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J. et al. (2016). Apache spark: a unified engine for big data processing. Communications of the ACM, 59, 56–65.

Zuech, R., Khoshgofthaar, T., & Wald, R. (2015). Intrusion detection and big heterogeneous data: a survey. Journal of Big Data, (pp. 90–107).

KEY TERMS AND DEFINITIONS

Anomaly Detection: Finds deviations from normal. Rare events or observations raise suspicions when differ significantly from most of the data.

Big Data: Traditional technologies of data processing, have difficulties to handle in tolerable time a large dataset.

Clustering: Aims to group data automatically according to their degree of similarity.

Log Analysis: It seeks to extract knowledge about threats or malfunctions, from records generated by a device.

Security Analytics: Is an approach to cybersecurity focused on the analysis of data.

SIEM: A software solution that allows to collect, manage and correlate security events generated by several network devices.

Stream Processing: Given a data set (a stream), an operation is applied to all elements of the stream.