Human-centric Computing and Information Sciences

RESEARCH Open Access

A survey of cloud-based network intrusion detection analysis

Nathan Keegan[1], Soo-Yeon Ji[2], Aastha Chaudhary[1], Claude Concolato[1], Byunggu Yu[1] and Dong Hyun Jeong[1*]

*Correspondence: djeong@udc.edu
[1] Department of Computer Science and Information Technology, University of the District of Columbia, 4200 Connecticut Avenue NW, Washington, DC 20008, USA
Full list of author information is available at the end of the article

Abstract
As network traffic grows and attacks become more prevalent and complex, we must find creative new ways to enhance intrusion detection systems (IDSes). Recently, researchers have begun to harness both machine learning and cloud computing technology to better identify threats and speed up computation times. This paper explores current research at the intersection of these two fields by examining cloud-based network intrusion detection approaches that utilize machine learning algorithms (MLAs). Specifically, we consider clustering and classification MLAs, their applicability to modern intrusion detection, and feature selection algorithms, in order to underline prominent implementations from recent research. We offer a current overview of this growing body of research, highlighting successes, challenges, and future directions for MLA-usage in cloud-based network intrusion detection approaches.

Keywords: Network intrusion detection analysis, Cloud computing, Mapreduce

Background

Since the dawn of computer networking, intrusion detection systems (IDSes) have played a critical role in ensuring safe networks for all users, but the shape of the role has changed throughout recent history. What began as system administrators manually monitoring user activities in the early '70s became teams sifting through audit logs in the '80s, transitioning to online analysis and dedicated programs in ensuing decades. None of these systems were able to effectively pinpoint attacks very quickly, however, and thus were generally used as forensic tools to examine security incidents ex post facto [1].

As traffic grew and attacks became more prevalent with the popularity of the Internet in the 1990s, it became apparent that swifter intrusion detection analysis was necessary to both diagnose and prevent attacks. To accomplish this, researchers worked to better understand network traffic patterns, which resulted in the greater development of signature-based and behavior-based detection techniques. Signature-based techniques compare traffic patterns to known attack signatures, whereas behavior-based techniques detect intrusions via deviations from normal or expected traffic behavior. While both techniques can be effective in real-time, significant limitations nonetheless exist—signature-based techniques cannot guard against unknown intrusions, and behavior-based techniques break down under heavy traffic or sudden traffic bursts [2].

To address these heavy traffic limitations, researchers harnessed the power of cloud computing technology to speed up computation. In recent years, the MapReduce computing platform, particularly through the Hadoop Distributed File System (HDFS), has been used to perform advanced intrusion detection analysis [3–7]. Hadoop, a popular open-source software framework for distributed storage and distributed processing of big data, uses MapReduce, a parallel processing paradigm that can perform rapid analysis to determine the presence of attacks or malicious activities in large quantities of network traffic. HDFS mainly provides scalable and reliable data storage for managing incoming network traffic data. To perform intrusion detection analysis using cloud computing, new computational models need to be designed by following the parallel processing paradigm. Alternatively, a modification of existing computational models should be applied to make them run on a cloud computing environment. Computing platforms such as Hadoop and MapReduce generally distribute computations among scores of computers or more, stymieing any algorithms with iterative or linear computations that require access to all input data to perform. Therefore, many popular computational algorithms cannot be directly utilized in cloud computing architecture [8]. To address this limitation, researchers have started designing scalable performant machine learning applications that run in the cloud computing environment [9–14]. In addition, many researchers emphasized the importance of machine learning algorithms (MLAs) to intrusion detection analysis using cloud computing technology [15–18]. Although the integration of MLAs to intrusion detection is important, it has not been studied broadly, primarily due to problems associated with leveraging optimal algorithms in a cloud computing environment. Thus, contemporary research in the field both proposes manners to adapt these MLAs for cloud server architecture [8, 19] and argues the importance of MLAs in network traffic analysis writ large [20–22]. In this paper, we attempt to synthesize the recent efforts of these researchers, addressing the importance of MLAs in network intrusion detection by identifying and classifying known solutions for intrusion detection analysis in the cloud.

This paper is organized as follows. In "Intrusion detection" section, we discuss the history, definition, and concepts of intrusion detection techniques. In "Using MLAs in intrusion detection" section, we introduce the concept of using MLAs in intrusion detection, highlighting implementation challenges and recent advancements in the field. Different types of machine learning algorithms, clustering and classifying algorithms, and evaluating their applicability to modern intrusion detection techniques are explained. "Utilization of MLAs in cloud computing" section collects recent researches that employ MLAs in cloud-based intrusion detection techniques. After providing problems and challenges of utilizing MLAs in cloud-based intrusion analysis in "Discussion" section, we will finish this paper by representing a conclusion and directions for future research in "Conclusion and futureworks" section.

Intrusion detection

Kemmerer and Vigna [1] outlined a handy history of intrusion detection, moving from the early days of manual detection and analysis by systems administrators in the 1970s to real-time solutions in the 1990s and early 2000s. Intrusion detection is defined as the process of monitoring events that occur on computers and networks [23]. As intrusions became more varied and sophisticated, the need for frameworks and classification has been emphasized. Debar et al. [24] performed a taxonomical approach of understanding IDSes by defining them as any mechanisms that process information coming from the system that is to be protected. They further described existing intrusion detection techniques using two important concepts: detection method and behavior on detection. We will discuss these concepts in "Using MLAs in intrusion detection" section.

Detection methods

Intrusion detection methods are generally divided into signature-based and behavior-based or anomaly-based techniques [25]. Signature-based techniques (also known as knowledge-based or misused-based techniques) reference databases of previous attack signatures and known system vulnerabilities. Signature-based techniques are quite accurate and effective for known attacks, but cannot guard against unknown attacks. To reduce this limitation, constant update of attack signatures needs to be performed. However, this might require considerable resources and overhead.

To guard against unknown attacks, behavior-based or anomaly-based techniques can be used. These detect intrusion attempts as deviation by comparing them to normal network activity (i.e. commands or traffic). While these two types of techniques maintain considerable overlap and are often regarded as identical in literature, we posit that they are slightly different. Hereafter, we will refer to anomaly-based techniques as those that involve first training a system to establish a normal profile, and then using that profile to detect deviations, and behavior-based techniques as those that do not necessarily compare against a baseline. For example, in behavior-based detection methods, an administrator might simply establish certain rules that would trigger alerts when broken. In practice, these two types of techniques are often one and the same, but it is nonetheless important to establish their subtle differences.

Behavior "around" detection

Once effective detection methods have been established, the question becomes: what kind of behavior does the system adopt after detection? While Debar et al. [24], for example, mentioned behavior on detection, we prefer behavior "around" detection, as it better describes a system's possible actions both before and after detection—IDSes are hardly ever simply reactionary systems that only take action after the fact.

Along these lines, Halme and Bauer [26] advanced one of the first taxonomies of anti-intrusion techniques, and divided them into six approaches: prevention, preemption, deterrence, deflection, detection, and countermeasures. The first three approaches (prevention, preemption, and deterrence) are passive measures to guard against attacks, and the latter three (deflection, detection, and countermeasures) are active measures to protect elements in a system. Many of these approaches are fluid, and can be used at many points throughout the process—before, after, or during attacks.

Regardless of the terminology or order of deployment, IDSes are critical systems that detect and act against attacks in a variety of ways. In late 1990s, researchers [24, 27] inventoried early real-time IDSes, many of which employed combinations of signature and behavior-based techniques by following Halme and Bauer's six approaches [26]. As time wore on, however, and network traffic continued its inexorable growth, these systems became untenable in real-time, so researchers have taken to the cloud to beef up analysis and computing of network intrusion detection.

Using MLAs in intrusion detection

Although the aforementioned approaches and techniques greatly improved intrusion detection techniques throughout past decades, many researchers have argued for the importance of MLAs in intrusion detection. On the surface, MLAs might seem like an easy candidate for improvement of these systems. After all, MLAs form the basis for anomaly detection in other seemingly related areas of computer science, such as spam-detection.

However, MLAs cannot be applied to intrusion detection directly. Sommer and Paxson [18] outlined a number of challenges for MLAs in intrusion detection, highlighting the fact that machine learning tools are most adept at finding activity similar to something previously seen, which goes against the general definition of the anomaly-based intrusion detection techniques that seek to identify novel attacks. In addition, the high cost of errors, lack of training data, and enormous variability of input data all impede the applicability of MLAs to intrusion detection.

Despite these considerable challenges, researchers have tackled problems and developed MLAs for intrusion detection. Below, we will discuss recent advancements in the field of MLAs in intrusion detection, highlighting the research that is successfully addressing the challenges that Sommer and Paxson [18] first set forth in 2010. This research generally employs two types of MLAs—clustering and classification algorithms—which are considered suitable for intrusion detection techniques.

Clustering algorithms

Clustering is a form of unsupervised machine learning that does not rely on training data or classification models. Instead, it splits input datasets into clusters on the basis of common features, in order to find similar patterns in input datasets. Similarity measures (e.g. Euclidean distance) are often utilized to uncover these patterns.

The *k*-means algorithm is a simple, clustering algorithm popular for general use [28]. The algorithm works by first selecting initial cluster centers (also known as centroids) and calculating the average distance between centroids and all other points in the system. This step can then be iterated continuously, establishing new centroids and relocating data points until average distance is decreased and no more relocation occurs.

Clustering algorithms can be used in a variety of fields, including market segmentation, geostatistics, computer vision, search, and medicine, among others. They are also widely used as preprocessing steps for other algorithms, where they can be useful in providing initial configurations. More importantly for our purposes, however, clustering algorithms have been used to tackle the important problem of network traffic classification. In the past, traffic classification was accomplished with entirely port-based and payload-based techniques. However, with the advent of applications that routinely use dynamic port numbers, masquerading techniques, and encryption, clustering algorithms have been implemented to exploit applications' distinctive characteristics in behavior-based approaches.

Clustering algorithms explored in recent network traffic researches include the *k*-means, DBSCAN, expectation maximization, and autoclass algorithms. Nguyen and Armitage [29] concluded that the autoClass algorithm produces the best accuracy when performing clustering for network traffic classification. McGregor et al. [30] found expectation maximization to be useful in finding network flow statistics. Bernaille et al. [31] employed *k*-means to classify traffic via TCP headers, possibly eliminating the need to collect whole packets to predict intrusion. From these researches, it is quite evident that clustering algorithms provide a number of salient applications to network traffic analysis, and to intrusion detection on the whole.

Classification algorithms

Unlike clustering, classification algorithms provide supervised machine learning for network intrusion detection. In this sense, good practice suggests that the presence of large learning datasets will lead to higher probability of success. Classification consists of two stages: training and testing. In the training step, a classifier model is selected to receive learning input, and once the classifier model has been sufficiently trained, testing is performed to determine accuracy through false positives, false negatives, true positives, and true negatives. A good classifier model, as we expect, returns a minimal quantity of false positives and negatives.

In computer networks, classification algorithms can be used to perform categorization of packets into "flows," in a process known as packet classification [32]. Packet classification has been used to support access control, quality of service, and intrusion detection [33]. Broadly used classification algorithms include Naïve Bayes (NB), Bayesian Network (BN), Logistic Regression (LR), Artificial Neural Networks (ANN), Support Vector Machines (SVM), Decision Tree (DT), Random Tree Classifier (rTree), Genetic Algorithms (GA), and Random Forest Classifier (rForest) [29, 34–38].

In the past, statistical analysis has been used broadly to detect intrusions by performing a statistical comparison of current network events to a pre-determined set of baseline criteria. Since the statistical analysis has a limitation of identifying different types of attacks, researchers proposed various alternative approaches. Wang [39] showed the effectiveness of using logistic regression modeling to detect multi-attack types. Cannady [40] emphasized the usefulness of using ANN for intrusion detection. However, ANN has not been widely applied since the accuracy of detecting intrusions is closely depending on the amount of used training datasets and methods. Also, it does not provide a detailed level of accuracy (including reasons). Therefore, it has been known as "black box" operation. NB is known as a simplified Bayesian probability model. NB classifier operates based on the likelihood that one attribute does not affect others. Amor et al. [41] utilized NB for detecting intrusions. Although NB is generally faster than DT for learning and classifying, they identified that there was no significant performance difference between the two for detecting intrusions. rForest is also one of the broadly used classification algorithms in intrusion detection. In recent, Albayati and Issac [42] found that rForest is more effective than NB for detecting intrusions. From the study of measuring the performance of detecting intrusions among NB, rTree, and rForest, they identified that rForest is superior to others with maintaining low false alarm rate. Interestingly, many researchers used SVM to conduct intrusion detection analysis [43–47] because it is good for classifying data by finding a hyperplane that maximizes the margin among all intrusion classes. It simply classifies the input data by a set of support vectors representing data patterns. However, SVM classification depends mainly on used kernel types and parameter settings [47]. It also requires longer training time than other classification algorithms. To address this limitation, Khan et al. [43] proposed an approach of integrating hierarchical clustering analysis. Different from other classification algorithms, GA approach has been in used for various purposes in intrusion detection as optimization, automatic model generation, and classification [48]. GA is a search algorithm that utilizes the mechanics of natural selection and genetics. It is often used to generate detection rules or to select appropriate features from the input data. However, the classification accuracy of using GA was slightly lower than tree algorithms such as J4.8 and CART [49]. Although numerous classification algorithms are used to detect intrusions, only a couple of algorithms have been used with an integration of cloud computing architecture. A detailed explanation how the MLAs are used with integration of cloud computing technology for detecting intrusions is included in "Utilization of MLAs in cloud computing" section.

Clustering versus classification

As we discussed above, in the past, researchers focused on identifying different algorithms' applicability to intrusion detection. However, in recent studies [36, 37], researchers has exclusively focused on the use of classification MLAs in network intrusion detection. From the study by Erman et al. [34], it has been found that clustering (i.e. autoClass) outperforms classification (i.e. Naïve Bayes) by up to 9 % in accuracy metrics like recall, precision, and overall accuracy.

Why then, do classification algorithms dominate the current literature on MLAs in intrusion detection analysis? Although clustering can produce better results than classification, it is important to note that clusters generally do not map 1:1 to applications [34]. In an ideal clustering environment, the number of clusters would equal the number of application classes (HTTP, SMTP, FTP, POP3, etc.) and each application class would dominate one cluster [29]. In reality, applications can spread out and dominate a number of clusters or dominate no clusters at all. In these situations, it can become quite sticky to map backwards from a cluster to the source application. Given this limitation, classification algorithms are much more commonly applied in the network intrusion detection sphere.

By and large, these MLA experiments and implementations relied on static, offline analysis of previously captured traffic. As network traffic data grows and more of these MLAs are adapted to the cloud, utilization of cloud computing for network intrusion detection is increasingly inevitable. In the following section, we discuss recent advances in the field.

Utilization of MLAs in cloud computing

The introduction of cloud computing provided a dramatic change in data management and processing across many different fields of computing—not only does it shift infrastructure and computation to the network, it also dramatically reduces costs associated with the management of hardware and software resources. Furthermore, it has resulted in the development of new programming models such as MapReduce (e.g. Hadoop), BigTable, and hybrid systems like Hive for analyzing large, complex, and disjointed data sets [50]. Using these models, numerous studies have been performed to conduct computation analysis in cloud computing [5, 19, 20].

As we have discussed above, many MLAs can be paired with cloud computing technologies to improve intrusion detection analysis. When dealing with massive amounts of data (on the order of terabytes or petabytes, for example), intrusion detection becomes extremely difficult, and single computers cannot handle the sheer size of data. Thus, we turn to the cloud. Recent cloud-based intrusion detection techniques have predominantly employed the MapReduce model, an abstract programming model that processes large datasets on clusters of computers. MapReduce is composed of two somewhat obvious steps—map and reduce. In the MapReduce model, a large dataset is split and each split sent to a node, also known as a mapper, where each split is independently processed. Mapper results are then shuffled, sorted, and passed to reducers that digest and prepare the final results [3]. A possible implementation of MapReduce for a cloud-based intrusion detection technique is inherently simple. The map step examines each split for traffic anomalies, and the reduce step combines them, packages them, and presents the overall report. Many researchers' intrusion detection techniques follow this general approach [3, 5, 51]. While particulars change, MapReduce remains a foundational tool throughout the most recent research on cloud-based intrusion detection analysis. Throughout the research, these implementations greatly reduce computing times for large traffic datasets.

Nonetheless, moving these intrusion detection techniques to the cloud brought with it its own unique set of challenges, most visible of which was the unsuitability of certain algorithms to cloud server architecture. Important and popular algorithm (such as the aforementioned *k*-means algorithm) cannot be directly implemented in the MapReduce framework due to iterative computations that reference all input data. As MapReduce splits input data to be processed among numerous computers, the algorithm cannot access inputs on different computers in the cloud architecture. However, researchers [8] adapted the *k*-means algorithm for use with MapReduce, by constructing and sharing a global array of centers that allow all distances to be calculated. As a popular cloud-based machine learning and data mining tool built on MapReduce, Mahout [52] and MLlib (aka SparkML) [53] support various MLAs including *k*-means functionality. However, writing new or customizing existing algorithms is too costly because all algorithms need to be implemented (or modified) by following fixed distributed runtime plans and underlying data-parallel framework. To address this limitation, researchers proposed several approaches of fast implementing approaches as SystemML [10], NIMBLE [11], MLbase [12], Distributed GraphLab [13], and Tupleware [14]. These approaches are classified as "Declarative ML" [54]. Declarative ML simplifies the development of MLAs by separating algorithm semantics from underlying framework and execution plans to make them run in a cloud computing environment more efficiently. Since Declarative ML is a rather new approach, it has not been broadly used to intrusion detection study yet.

Network flow and feature selection

While the MapReduce paradigm can be effective for larger datasets, some algorithms still have trouble swiftly dealing with large volumes of continuously generated traffic data. To combat this, some researchers have turned to network flow applications (such as NetFlow, sFlow, OpenFlow and IPFIX) to cut down on data overheads with filtering, sampling, and flow aggregation techniques [55].

Lee et al. [21], for example, used Cisco NetFlow to monitor internet traffic, filtering unnecessary data out of flow records using MapReduce to improve computation times by 72 %. Li et al. [35] combined sFlow, MapReduce, and different MLAs (SVM and Decision Tree) to successfully classify host roles-a critical component of intrusion detection.

Nonetheless, while these network flow applications offer random and deterministic packet sampling, as well as other filtering, feature selection, and aggregation tools, they are often not sufficient for the complex world of cloud-based intrusion detection. As such, in recent years, researchers have increasingly turned to more in-depth, algorithm-based feature selection techniques to improve the results of their cloud-based intrusion detection techniques.

In particular, while network flow applications do provide some feature selection techniques, the problem of which features to choose remains a critical one for intrusion detection techniques on the whole. Not every feature of the data will be relevant, and some features can actually introduce noise or redundancy. Thus, selecting the optimal subset of features has become of great interest to researchers in recent years, and MLAs have provided an effective way to zero in on the best feature choices.

Stein et al. [56] pioneered this work before the cloud, improving decision tree classification performance by introducing genetic algorithm-based feature selection that eliminates distracting or unnecessary features. In the cloud, Muthurajkumar et al. [51] employed a rough set-based feature selection algorithm that generates feature subsets designed to find the best balance between detection rates and false alarms in a cloud-based intrusion detection technique. Chen et al. [57] used a MapReduce-based implementation of the OneR classifying algorithm, alongside vertical compression to improve detection up to 184 times with only tolerable losses in performance in their SVM-based cloud intrusion detection technique.

Although these feature selection algorithms have improved their respective intrusion detection techniques, they are not without their drawbacks. In some cases, while feature selection may improve the speeds of the detection algorithms, the overall time run-time increases, and training data cannot be incrementally handled [56]. Thus, if feature selection algorithms have to periodically re-assess optimal feature subsets, they may provide even more overhead, or miss newer, more sophisticated attacks. Chen et al. [57] argued that these drawbacks can be improved by refining MapReduce performance and implementing newer algorithms with incremental clustering (or classifying) abilities.

Implementation examples

With the MapReduce paradigm, intrusion attempts per port number can be traced by analyzing a large login attempt dataset. First, a mapper grabs all port number data and places them into a key-value pair, and then a reducer condenses the data into more discrete, manageable sets (i.e. Port 22: 25 entries; Port 80: 50 entries, ...) In general, the reducer performs its job in three phases: shuffle, sort, and reduce. The reduced dataset can then be processed by MLAs to classify port activities, as was shown in [35]. Theoretically, when considering the analysis of the login attempts, the input data can be formatted to become a set of key-value pairs as (k_i, v_i) and the map function is applied to produce a list of intermediate key-value pairs as $map : (k_i, v_i) \rightarrow list(k_j, v_j)$. Since the intermediate list of key-value pairs indicates the outputs produced by mappers, they need to be merged by reducers as $reduce : (k_2, list(v_2)) \rightarrow list(k_3, v_3)$. This mechanism is also efficient for detecting anomalous network activities by analyzing IP flow records [4, 58].

When utilizing cloud computing architecture for intrusion detection, most cloud-based intrusion detection techniques are designed consisting of multiple components as data parser, data processing, data mapper and reducer. The data parser extracts essential information from the input data by eliminating unnecessary data. It mainly focuses on getting rid of useless (or redundant) features (i.e. variables) as well as identifying unknown but significant features from "Big Dimensionality" data [59]. The parsed information is then processed to determine important features, which are formatted as metadata file and distributed to HDFS nodes. Then, cloud job dispatcher launches the data mapper to assign jobs to each computing node. After completion of the mapping process, the data reducer is performed to reduce redundancy information by merging them. The MapReduce model can be adapted to run each component. For instance, the model is often used to extract features since it requires a longer processing time [60]. Most supervised MLAs requires separate training and testing datasets. With the training dataset, a learning model is generated. Then, the learning model is applied to validate and test with the testing dataset to show the effectiveness of the generated model. To run the supervised learning algorithms in the cloud, researchers proposed an idea of conducting the model generation as a sequential, but parallel processing on testing and training the data for detecting intrusions [61, 62]. Figure 1 shows a schematic diagram of classification procedure for intrusion detection with integrating the MapReduce model. It is important to note that this classification procedure is not a generic approach. For instance, the MapReduce model should be considered when dealing with numerous input features because it can speed up the computation of the data processing. Otherwise, the model is not necessary to be adapted to the data processing. In general, the MapReduce model can have multiple mappers and reducers. Although single mapper and reducer can be used depending on designed analysis models, it is important to note that if the number of reducers decreases, the computation time of merging the outcomes coming from the mappers increases. No reducer is also possible. For example, del Río et al. [63] used only multiple mappers (no reducer) for network intrusion detection with rForest. Since various decision trees are generated from multiple mappers, they considered using all outcomes (i.e. decision rules) as classification rules.

As we discussed above, MLAs are generally not the sole component of a functioning intrusion detection, often working in tandem with other algorithms. Below we outline some of the most salient examples of MLA implementation in cloud-based intrusion detection techniques in recent research, and how they fit in to their respective systems.

- Muthurajkumar et al. [51] introduced an intrusion detection model that used a combination of fuzzy SVM and feature selection algorithms to produce high detection rates and minimal false positives.

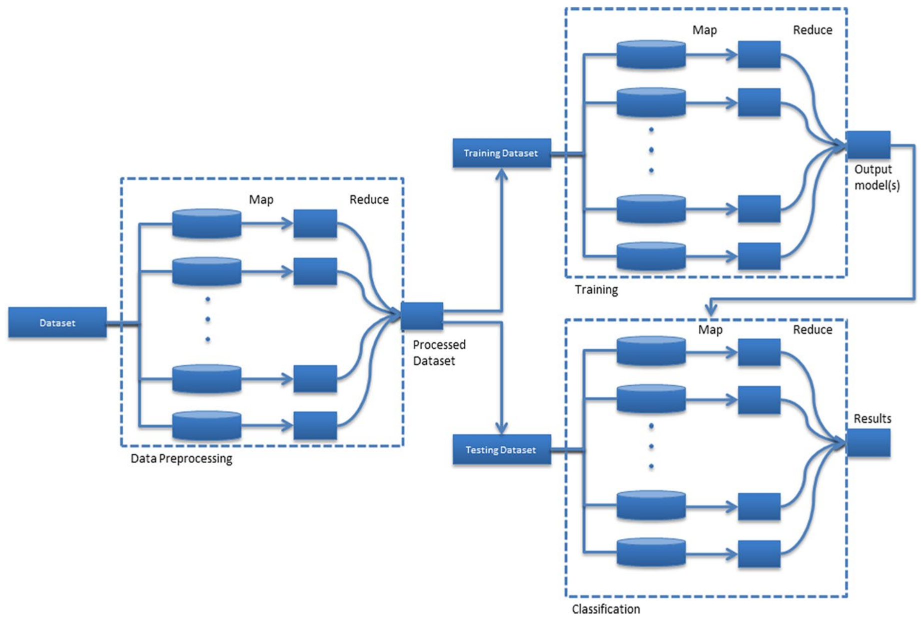

Fig. 1 A schematic diagram of MapReduce-based classification procedure for intrusion detection. It shows an example when MapReduce is applied multiple times to perform data processing, data training, and testing

- Vieira et al. [64] proposed a grid and cloud computing intrusion detection system (GCCIDS) that combats attacks by using both signature-based and anomaly-based techniques to detect intrusions. In order to train the system, the authors employed neural network classification algorithms, and the resulting system boasted low processing overhead and satisfactory performance for real-time implementation.
- Singh et al. [65] implemented rForest for peer-to-peer botnet detection that proved adept at classifying malicious traffic on a cluster, with low false positive rates and considerable precision and recall.

A further summary of network intrusion detection techniques that employ MLAs in cloud-computing environments is included in Table 1.

Discussion

Besides the obvious concerns about ever-growing volume of network traffic data and common trade-off considerations (e.g. overhead vs. speed), there are several major research challenges of utilizing MLA in cloud-based network intrusion analysis.

There are three salient challenges facing widespread MLA implementation in cloud-based network intrusion detection techniques [15]. First, MLAs trained on a particular datasets may not be suitable for other datasets, and that classification may not be robust over different datasets or domains. Although this remains a critical concern, some researchers have offered preliminary solutions to this problem. Second, MLAs in general are trained using a given number of class types, and hence large varieties of class types found in a dynamically growing dataset could lead to inaccurate classification results. Lastly, MLAs are developed based on a single learning task, and thus they are

Table 1 Network intrusion detection techniques that have been developed utilizing cloud-computing technology

Work	Goal	Dataset(s)	Major approaches	Cloud environment	ML algorithm(s)	Advantages	Challenges
Lee et al. [21]	Monitor internet traffic flow	Simulated NetFlow packets	(1) Packet sampling, (2) Flow aggregation, and (3) MapReduce programming model	Apache Hadoop	None	Flow computation time improved by 72 % over legacy tools	Batch-processing jobs and text input file formats difficult to handle; flow analysis tools are not adequately developed for the MapReduce interface
Singh et al. [65]	P2P botnet detection	Simulated and CAIDA sample datasets	(1) Information gain measurement and (2) Clustering (random forest) in mahout	Apache Hadoop	Random forest	Process high bandwidth in quasi-real-time, effectively classifies malicious traffic on a cluster	High packet drop rates, detection times still a little too high, cannot respond to newer, more sophisticated threats
Bhat et al. [67]	Anomaly intrusion detection	NSL-KDD 99	(1) Naïve Bayes (NB) tree and (2) A hybrid approach of NB tree and random forest	Amazon EC2	NB tree and random forest	Good performance, high accuracy, low false positive rate for NB tree/random forest hybrid implementation	High false positive rate for non-hybrid implementations
Chen et al. [20]	Phishing attack detection	Simulated dataset	Apache Hadoop	Eucalyptus, Apache Hadoop, and Amazon EC2	Collaborative algorithm based on distributed hash tables (DHT)	Practical scheme, can be generalized to other attacks	Not tested with various datesets
Chen et al. [57]	Intrusion detection	KDD 99, CMDC 2012	(1) Feature reduction, (2) Vertical compression, and (3) Intrusion detection	Apache Hadoop	OneR algorithm, affinity propagation, KNN, and SVM	Faster than traditional models	No incremental clustering ability—feature reduction and training steps can provide significant overhead
Marnerides et al. [22]	Malware detection	Simulated dataset	(1) Energy estimation, (2) Feature selection, and (3) Covariance analysis	Unknown	Choi-Williams distribution	Effective for identifying Kelihos injection	Not tested with various datesets

Table 1 continued

Work	Goal	Dataset(s)	Major approaches	Cloud environment	ML algorithm(s)	Advantages	Challenges
Muthurajkumar et al. [51]	Intrusion detection	Simulated dataset	(1) Feature selection and (2) Fuzzy SVM	Unknown	Rough set based feature selection algorithm (RSFSA), fuzzy SVM	Reduces number of decision attributes and the size of log data, faster than traditional models	Not tested with various datesets
Vieira et al. [64]	Intrusion detection technique	Simulated dataset	Utilization of grid and cloud computing	Unknown	Feed-forward neural network	Successfully explores communication events to mark intrusion	Large sample period of data is required and training cannot adapt new threats
Wang et al. [68]	Network traffic passive measurement	CAIDA dataset (anonymized traffic data collected from equinix-chicago and equinix-sanjose)	IP trace analysis system (IPTAS)	Unknown	None	Useful prototype of passive traffic analysis tool	Not provide a fine-grained traffic analysis

not suitable for multiple learning tasks and knowledge transfers required for effective intrusion detection and prevention.

Although Singh et al. [65] provided a concrete example of these latter concerns in our literature, detailing newer botnet architectures that allow for more efficient, less detectable communication inspired by ant-colony foraging behavior, it is important to note that current machine learning techniques cannot uncover or flag these types of instantaneous or stealthy behaviors. In addition, we should consider and assume that as newer threats develop, MLAs will have difficulty remaining one step ahead, without opportunity to train for newer class types or multiple learning tasks necessary to keep attackers at bay [18].

Even effective classification training cannot provide timely or accurate results for network intrusion. Vieira et al. [64] found that 10 days of usage simulation for artificial neural network training on their intrusion detection techniques fell considerably short, resulting in a high number of false negatives and high uncertainty. The longer a MLA takes to complete its learning phase, the slower it will be to adapt to new threats. Moreover, the lack of incremental clustering ability is considered as a possible research challenge [57]. Therefore, feature extraction or reduction algorithms often have to be performed, which could provide considerable overhead and slower response to newer, more sophisticated attacks [66].

Thus, the two-pronged challenge of swifter results and more efficient, nimble training looms large in the near future, as intrusion detection techniques work to become more nimble and responsive. In truth, we will likely never see a perfect cloud-based, MLA-integrated intrusion detection approach, but some of these deficiencies can be improved with general advancement of the field, dedicated refinement of algorithms, and some creative problem solving along the way.

Conclusion and futureworks

As more and more of our modern computing infrastructure migrates to the cloud, intrusion detection will become an ever more important piece of the research landscape. To protect infrastructures, organizations all over the world are spending considerable amounts on information security and privacy.

This paper examines the current state of network intrusion detection research, highlighting various MLAs that can be utilized not only to detect security incidents, but to proactively monitor networks, or train a system to improve their detection processes. The field of network traffic intrusion detection using machine learning has potential to grow—especially in the cloud but certain problems and challenges must be overcome.

In future work, we plan to conduct an extensive study to determine the effectiveness of machine-learning based solutions for network intrusion detection, focusing on performance of real-time cloud-based intrusion detection techniques, swifter and more effective training, and in-depth comparisons of existing network intrusion detection solutions. At the same time, we are going to design a comprehensive, but reliable intrusion detection approach by integrating machine learning techniques.

Authors' contributions

NK carried out the study of understanding previous works and drafted the manuscript. AC and CC participated in conducting literature review. SJ and BY worked on identifying issues and concerns of this study in order to enhance the quality of this paper. DJ conceived of the study and helped to draft the manuscript. All authors read and approved the final manuscript.

Author details

[1] Department of Computer Science and Information Technology, University of the District of Columbia, 4200 Connecticut Avenue NW, Washington, DC 20008, USA. [2] Department of Computer Science, Bowie State University, 14000 Jericho Park Road, Bowie, MD 20715, USA.

Acknowledgements

This work was supported by US Army Research Office (ARO) Grant W911NF-13-1-0143.

Competing interests

The authors declare that they have no competing interests.

Received: 14 April 2015 Accepted: 31 August 2016
Published online: 05 December 2016

References

1. Kemmerer RA, Vigna G (2002) Intrusion detection: a brief history and overview. Computer 35(4):27–30. doi:10.1109/mc.2002.1012428
2. Kind A, Stoecklin MP, Dimitropoulos X (2009) Histogram-based traffic anomaly detection. IEEE Trans Netw Serv Manag 6(2):110–121. doi:10.1109/TNSM.2009.090604
3. Fontugne R, Mazel J, Fukuda K (2014) Hashdoop: a mapreduce framework for network anomaly detection. In: 2014 IEEE conference on computer communications workshops (INFOCOM WKSHPS). pp 494–499. doi:10.1109/INFCOMW.2014.6849281
4. Francois J, Wang S, Bronzi W, State R, Engel T (2011) Botcloud: detecting botnets using mapreduce. In: 2011 IEEE international workshop on Information Forensics and Security (WIFS). pp 1–6. doi:10.1109/WIFS.2011.6123125
5. Kumar M, Hanumanthappa M (2013) Scalable intrusion detection systems log analysis using cloud computing infrastructure. In: 2013 IEEE international conference on computational intelligence and computing research (ICCIC). pp 1–4. doi:10.1109/ICCIC.2013.6724158
6. Lee Y, Lee Y (2011) Detecting ddos attacks with hadoop. In: Proceedings of The ACM CoNEXT Student Workshop, CoNEXT '11 Student. ACM, New York, pp 7–172. doi:10.1145/2079327.2079334
7. Tripathi S, Gupta B, Veluru S (2013) Hadoop based defense solution to handle distributed denial of service (ddos) attacks. J Inform Secur 4(3):150–164. doi:10.4236/jis.2013.43018
8. Zhao W, Ma H, He Q (2009) Parallel k-means clustering based on mapreduce. In: Jaatun M, Zhao G, Rong C (eds) Cloud Computing, Lecture Notes in Computer Science. Springer, Berlin, pp 674–679. doi:10.1007/978-3-642-10665-1_71
9. Apache mahout: scalable machine learning and data mining. https://mahout.apache.org/. Accessed 03 Sept 2014
10. Ghoting A, Krishnamurthy R, Pednault E, Reinwald B, Sindhwani V, Tatikonda S, Tian Y, Vaithyanathan S (2011) Systemml: declarative machine learning on mapreduce. In: Proceedings of the 2011 IEEE 27th international conference on data engineering, ICDE '11. IEEE Computer Society, Washington, DC, pp 231–242. doi:10.1109/ICDE.2011.5767930
11. Ghoting A, Kambadur P, Pednault E, Kannan R (2011) Nimble: a toolkit for the implementation of parallel data mining and machine learning algorithms on mapreduce. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '11. ACM, New York, pp 334–342. doi:10.1145/2020408.2020464 http://doi.acm.org/10.1145/2020408.2020464
12. Kraska T, Talwalkar A, Duchi JC, Griffith R, Franklin MJ, Jordan MI (2013) Mlbase: a distributed machine-learning system. In: 6th biennial conference on innovative data systems reserch (CIDR). http://cidrdb.org/cidr2013/program.html. http://cidrdb.org/cidr2013/Papers/CIDR13_Paper118.pdf
13. Low Y, Bickson D, Gonzalez J, Guestrin C, Kyrola A, Hellerstein JM (2012) Distributed graphlab: a framework for machine learning and data mining in the cloud. Proc VLDB Endow 5(8):716–727. doi:10.14778/2212351.2212354
14. Crotty Andrew AG, Kraska T (2014) Distributed machine learning on small clusters. IEEE Data Eng Bull 37(3):63–76
15. Suthaharan S (2014) Big data classification: problems and challenges in network intrusion prediction with machine learning. SIGMETRICS Perform Eval Rev 41(4):70–73. doi:10.1145/2627534.2627557
16. Hu B, Shen Y (2012) Machine learning based network traffic classification: a survey. J Inform Comput Sci 9(11):3161–3170
17. Yingqiu L, Wei L, Yunchun L (2007) Network traffic classification using k-means clustering. In: Second international multi-symposiums on computer and computational sciences, 2007. IMSCCS 2007, pp 360–365. doi:10.1109/IMSCCS.2007.52
18. Sommer R, Paxson V (2010) Outside the closed world: on using machine learning for network intrusion detection. In: 2010 IEEE symposium on security and privacy (SP), pp 305–316. IEEE, New York. doi:10.1109/sp.2010.25
19. Esteves RM, Pais R, Rong C (2011) K-means clustering in the cloud—a mahout test. In: Proceedings of the 2011 IEEE workshops of international conference on advanced information networking and applications, WAINA '11. IEEE Computer Society, Washington, DC, pp 514–519. doi:10.1109/WAINA.2011.136
20. Chen Z, Han F, Cao J, Jiang X, Chen S (2013) Cloud computing-based forensic analysis for collaborative network security management system. Tsinghua Sci Technol 18(1):40–50. doi:10.1109/TST.2013.6449406

21. Lee Y, Kang W, Son H (2010) An internet traffic analysis method with mapreduce. In: 2010 IEEE/IFIP network operations and management symposium workshops (NOMS Wksps). pp 357–361. doi:10.1109/NOMSW.2010.5486551
22. Marnerides A, Watson MR, Shirazi N, Mauthe A, Hutchison D (2013) Malware analysis in cloud computing: network and system characteristics. In: 2013 IEEE globecom workshops (GC Wkshps), pp 482–487. doi:10.1109/GLOCOMW.2013.6825034
23. Scarfone K, Scarfone K, Cybersecurity S, Mell P, Blank RM (2007) Secretary A: guide to intrusion detection and prevention systems (IDPS)
24. Debar H, Dacier M, Wespi A (1999) Towards a taxonomy of intrusion-detection systems. Comput Netw 31(8):805–822
25. Patcha A, Park JM (2007) An overview of anomaly detection techniques: existing solutions and latest technological trends. Comput Netw 51(12):3448–3470. doi:10.1016/j.comnet.2007.02.001
26. Halme LR, Bauer RK (1995) Aint misbehaving: a taxonomy of anti-intrusion techniques. In: Proceedings of the 18th national information systems security conference
27. Cannady96 J, Harrel J (1996) A comparative analysis of current intrusion detection technologies. In: Technology in information security conference (TISC), pp 212–218
28. Jain AK (2010) Data clustering: 50 years beyond k-means. Pattern Recognit Lett 31(8):651–666
29. Nguyen TTT, Armitage G (2008) A survey of techniques for internet traffic classification using machine learning. Commun Surveys Tutor 10(4):56–76. doi:10.1109/SURV.2008.080406
30. McGregor A, Hall M, Lorier P, Brunskill J (2004) Flow clustering using machine learning techniques. In: Passive and active network measurement. Springer, Berlin, pp 205–214
31. Bernaille L, Teixeira R, Akodkenou I, Soule A, Salamatian K (2006) Traffic classification on the fly. ACM SIGCOMM Comput Commun Rev 36(2):23–26
32. Gupta P, McKeown N (2001) Algorithms for packet classification. IEEE Netw 15(2):24–32
33. Qi Y, Xu L, Yang B, Xue Y, Li J (2009) Packet classification algorithms: from theory to practice. In: INFOCOM 2009. IEEE, pp 648–656. doi:10.1109/INFCOM.2009.5061972
34. Erman J, Mahanti A, Arlitt M (2006) Internet traffic identification using machine learning. In: Global telecommunications conference, 2006, GLOBECOM '06. IEEE, pp 1–6. doi:10.1109/GLOCOM.2006.443
35. Li K, Gibson C, Ho D, Zhou Q, Kim J, Buhisi O, Brown DE, Gerber M (2013) Assessment of machine learning algorithms in cloud computing frameworks. In: 2013 IEEE systems and information engineering design symposium (SIEDS), pp 98–103. doi:10.1109/SIEDS.2013.6549501
36. Singh K, Agrawal S (2011) Performance evaluation of five machine learning algorithms and three feature selection algorithms for ip traffic classification. IJCA Special Issue on Evolution in Networks and Computer Communications (1):25–32. http://www.ijcaonline.org/specialissues/encc/number1/3716-encc005
37. Stevanovic M, Pedersen JM (2014) An efficient flow-based botnet detection using supervised machine learning. In: 2014 international conference on computing, networking and communications (ICNC). pp 797–801. doi:10.1109/ICCNC.2014.6785439
38. Xia T, Qu G, Hariri S, Yousif M () An efficient network intrusion detection method based on information theory and genetic algorithm. In: 24th IEEE international performance, computing, and communications conference, 2005. IPCCC 2005, pp 11–17. doi:10.1109/PCCC.2005.1460505
39. Wang Y (2005) A multinomial logistic regression modeling approach for anomaly intrusion detection. Comput Secur 24(8):662–674. doi:10.1016/j.cose.2005.05.003
40. Cannady J (1998) Artificial neural networks for misuse detection. In: National information systems security conference, pp 443–456
41. Amor NB, Benferhat S, Elouedi Z (2004) Naive bayes vs decision trees in intrusion detection systems. In: Proceedings of the 2004 ACM symposium on applied computing, SAC '04. ACM, New York, pp 420–424. doi:10.1145/967900.967989
42. Albayati M, Issac B (2015) Analysis of intelligent classifiers and enhancing the detection accuracy for intrusion detection system. Int J Comput Intel Syst 8(5):841–853. doi:10.1080/18756891.2015.1084705
43. Khan L, Awad M, Thuraisingham B (2007) A new intrusion detection system using support vector machines and hierarchical clustering. VLDB J 16(4):507–521. doi:10.1007/s00778-006-0002-5
44. Mulay SA, Devale PR, Garje GV (2010) Intrusion detection system using support vector machine and decision tree. Int J Comput Appl 3(3):40–43
45. Yao J, Zhao S, Fan L (2006) An enhanced support vector machine model for intrusion detection. Proceedings of the first international conference on rough sets and knowledge technology., RSKT'06. Springer, Berlin, pp 538–543
46. Ji S-Y, Jeong B-K, Choi S, Jeong DH (2016) A multi-level intrusion detection method for abnormal network behaviors. J Netw Comput Appl 62:9–17. doi:10.1016/j.jnca.2015.12.004
47. Kausar N, Belhaouari Samir B, Abdullah A, Ahmad I, Hussain M (2011) A review of classification approaches using support vector machine in intrusion detection. In: Abd Manaf A, Sahibuddin S, Ahmad R, Mohd Daud S, El-Qawasmeh E (eds) Proceedings, part III, informatics engineering and information science: international conference, ICIEIS 2011, Kuala Lumpur, Malaysia, November 14-16, 2011. Springer, Berlin, pp 24–34. doi:10.1007/978-3-642-25462-8
48. Majeed PG, Kumar S (2014) Genetic algorithms in intrusion detection systems: a survey. Int J Innov Appl Stud 5(3):233–240
49. Pawar SN, Bichkar RS (2015) Genetic algorithm with variable length chromosomes for network intrusion detection. Int J Autom Comput 12(3):337–342. doi:10.1007/s11633-014-0870-x
50. Sakr S, Liu A, Batista DM, Alomari M (2011) A survey of large scale data management approaches in cloud environments. IEEE Commun Surv Tutor 13(3):311–336
51. Muthurajkumar S, Kulothungan K, Vijayalakshmi M, Jaisankar N, Kannan A (2013) A rough set based feature selection algorithm for effective intrusion detection in cloud model. In: Proceedings of the international conference on advances in communication, network, and computing, pp 8–13
52. Owen S, Anil R, Dunning T, Friedman E (2011) Mahout in action. Manning Publications Co., Greenwich

53. Meng X, Bradley JK, Yavuz B, Sparks ER, Venkataraman S, Liu D, Freeman J, Tsai DB, Amde M, Owen S, Xin D, Xin R, Franklin MJ, Zadeh R, Zaharia M, Talwalkar A (2015) Mllib: machine learning in apache spark. JMLR 17(34):1–7
54. Boehm M, Evfimievski AV, Pansare N, Reinwald B (2016) Declarative machine learning—a classification of basic properties and types. CoRR abs/1605.05826
55. Li B, Springer J, Bebis G, Hadi Gunes M (2013) A survey of network flow applications. J Netw Comput Appl 36(2):567–581. doi:10.1016/j.jnca.2012.12.020
56. Stein G, Chen B, Wu AS, Hua KA (2005) Decision tree classifier for network intrusion detection with GA-based feature selection. In: ACM-SE 43: Proceedings of the 43rd annual southeast regional conference. ACM, New York, pp 136–141. doi:10.1145/1167253.1167288
57. Chen T, Zhang X, Jin S, Kim O (2014) Efficient classification using parallel and scalable compressed model and its application on intrusion detection. Expert Syst Appl 41(13):5972–5983
58. Shu X, Smiy J, Yao DD, Lin H (2013) Massive distributed and parallel log analysis for organizational security. In: 2013 IEEE Globecom workshops (GC Wkshps), pp 194–199. doi:10.1109/GLOCOMW.2013.6824985
59. Zhai Y, Ong YS, Tsang IW (2014) The emerging "big dimensionality". IEEE Comput Intel Mag 9(3):14–26. doi:10.1109/MCI.2014.2326099
60. Hyunjoo Kim IK, Jonghyun K, Chung Tm (2015) Behavior-based anomaly detection on big data. In: Proceedings of 13th australian information security management conference, pp 73–80
61. Amy Xuyang Tan MK, Li Liu V, Thuraisingham B (2010) A comparison of approaches for large-scale data mining. Technical Report UTDSC-24-10, University of Texas at Dallas, Department of Computer Science
62. Aljarah I, Ludwig SA (2013) Mapreduce intrusion detection system based on a particle swarm optimization clustering algorithm. In: 2013 IEEE congress on evolutionary computation, pp 955–962. doi:10.1109/CEC.2013.6557670
63. del Río S, López V, Benítez JM, Herrera F (2014) On the use of mapreduce for imbalanced big data using random forest. Inform Sci 285:112–137. doi:10.1016/j.ins.2014.03.043
64. Vieira K, Schulter A, Westphall C, Westphall C (2009) Intrusion detection for grid and cloud computing. IT Prof Mag 4:38–43
65. Singh K, Guntuku SC, Thakur A, Hota C (2014) Big data analytics framework for peer-to-peer botnet detection using random forests. Inform Sci 278:488–497. doi:10.1016/j.ins.2014.03.066
66. Ji SY, Choi S, Jeong D (2014) Designing an internet traffic predictive model by applying a signal processing method. J Netw Syst Manag 26:1–18. doi:10.1007/s10922-014-9335-3
67. Bhat AH, Patra S, Jena D (2013) Machine learning approach for intrusion detection on cloud virtual machines. Int J Appl Innov Eng Manag (IJAIEM) 2(6):56–66
68. Wang H, Ding W, Xia Z (2012) A cloud-pattern based network traffic analysis platform for passive measurement. In: 2012 international conference on, cloud and service computing (CSC), pp 1–7. doi:10.1109/CSC.2012.8