A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection

Anna L. Buczak, *Member, IEEE*, and Erhan Guven, *Member, IEEE*

***Abstract*—This survey paper describes a focused literature survey of machine learning (ML) and data mining (DM) methods for cyber analytics in support of intrusion detection. Short tutorial descriptions of each ML/DM method are provided. Based on the number of citations or the relevance of an emerging method, papers representing each method were identified, read, and summarized. Because data are so important in ML/DM approaches, some well-known cyber data sets used in ML/DM are described. The complexity of ML/DM algorithms is addressed, discussion of challenges for using ML/DM for cyber security is presented, and some recommendations on when to use a given method are provided.**

***Index Terms*—Cyber analytics, data mining, machine learning.**

I. Introduction

THIS paper presents the results of a literature survey of machine learning (ML) and data mining (DM) methods for cyber security applications. The ML/DM methods are described, as well as several applications of each method to cyber intrusion detection problems. The complexity of different ML/DM algorithms is discussed, and the paper provides a set of comparison criteria for ML/DM methods and a set of recommendations on the best methods to use depending on the characteristics of the cyber problem to solve.

Cyber security is the set of technologies and processes designed to protect computers, networks, programs, and data from attack, unauthorized access, change, or destruction. Cyber security systems are composed of network security systems and computer (host) security systems. Each of these has, at a minimum, a firewall, antivirus software, and an intrusion detection system (IDS). IDSs help discover, determine, and identify unauthorized use, duplication, alteration, and destruction of information systems [1]. The security breaches include external intrusions (attacks from outside the organization) and internal intrusions (attacks from within the organization).

There are three main types of cyber analytics in support of IDSs: misuse-based (sometimes also called signature-based), anomaly-based, and hybrid. Misuse-based techniques are designed to detect known attacks by using signatures of those attacks. They are effective for detecting known type of attacks without generating an overwhelming number of false alarms. They require frequent manual updates of the database with rules and signatures. Misuse-based techniques cannot detect novel (zero-day) attacks.

Anomaly-based techniques model the normal network and system behavior, and identify anomalies as deviations from normal behavior. They are appealing because of their ability to detect zero-day attacks. Another advantage is that the profiles of normal activity are customized for every system, application, or network, thereby making it difficult for attackers to know which activities they can carry out undetected. Additionally, the data on which anomaly-based techniques alert (novel attacks) can be used to define the signatures for misuse detectors. The main disadvantage of anomaly-based techniques is the potential for high false alarm rates (FARs) because previously unseen (yet legitimate) system behaviors may be categorized as anomalies.

Hybrid techniques combine misuse and anomaly detection. They are employed to raise detection rates of known intrusions and decrease the false positive (FP) rate for unknown attacks. An in-depth review of the literature did not discover many pure anomaly detection methods; most of the methods were really hybrid. Therefore, in the descriptions of ML and DM methods, the anomaly detection and hybrid methods are described together.

Another division of IDSs is based on where they look for intrusive behavior: network-based or host-based. A network-based IDS identifies intrusions by monitoring traffic through network devices. A host-based IDS monitors process and file activities related to the software environment associated with a specific host.

This survey paper focuses on ML and DM techniques for cyber security, with an emphasis on the ML/DM methods and their descriptions. Many papers describing these methods have been published, including several reviews. In contrast to previous reviews, the focus of our paper is on publications that meet certain criteria. Google Scholar queries were performed using "machine learning" and cyber, and using "data mining" and cyber. Special emphasis was placed on highly cited papers because these described popular techniques. However, it was also recognized that this emphasis might overlook significant new and emerging techniques, so some of these papers were chosen also. Overall, papers were selected so that each of the ML/DM categories listed later had at least one and preferably a few representative papers.

This paper is intended for readers who wish to begin research in the field of ML/DM for cyber intrusion detection. As such, great emphasis is placed on a thorough description of the ML/DM methods, and references to seminal works for each ML

Manuscript received March 9, 2015; revised August 31, 2015; accepted October 16, 2015. Date of publication October 26, 2015; date of current version May 20, 2016. This work was supported by the Department of Homeland Security Network Security Deployment Division under contract HSSA01-13-C-2709 (Chris Yoon, Program Manager).

The authors are with The Johns Hopkins University Applied Physics Laboratory, Laurel, MD 20723 USA (e-mail: anna.buczak@jhuapl.edu).

Digital Object Identifier 10.1109/COMST.2015.2494502

and DM method are provided. Some examples are provided as to how the techniques were used in cyber security.

This paper does not describe all the different techniques of network anomaly detection, as do Bhuyan et al. [2]; instead it concentrates only on ML and DM techniques. However, in addition to the anomaly detection, signature-based and hybrid methods are depicted. The descriptions of the methods in the present survey are more in-depth than in [2].

Nguyen et al. [3] describe ML techniques for Internet traffic classification. The techniques described therein do not rely on well-known port numbers but on statistical traffic characteristics. Their survey only covers papers published in 2004 to 2007, where our survey includes more recent papers. Unlike Nguyen et al. [3], this paper presents methods that work on any type of cyber data, not only Internet Protocol (IP) flows.

Teodoro et al. [4] focus on anomaly-based network intrusion techniques. The authors present statistical, knowledge-based, and machine-learning approaches, but their study does not present a full set of state-of-the-art machine-learning methods. In contrast, this paper describes not only anomaly detection but also signature-based methods. Our paper also includes the methods for recognition of type of the attack (misuse) and for detection of an attack (intrusion). Lastly, our paper presents the full and latest list of ML/DM methods that are applied to cyber security.

Sperotto et al. [5] focus on Network Flow (NetFlow) data and point out that the packet processing may not be possible at the streaming speeds due to the amount of traffic. They describe a broad set of methods to detect anomalous traffic (possible attack) and misuse. However, unlike our paper, they do not include explanations of the technical details of the individual methods.

Wu et al. [6] focus on Computational Intelligence methods and their applications to intrusion detection. Methods such as Artificial Neural Networks (ANNs), Fuzzy Systems, Evolutionary Computation, Artificial Immune Systems, and Swarm Intelligence are described in great detail. Because only Computational Intelligence methods are described, major ML/DM methods such as clustering, decision trees, and rule mining (that this paper addresses) are not included.

This paper focuses primarily on cyber intrusion detection as it applies to wired networks. With a wired network, an adversary must pass through several layers of defense at firewalls and operating systems, or gain physical access to the network. However, a wireless network can be targeted at any node, so it is naturally more vulnerable to malicious attacks than a wired network. The ML and DM methods covered in this paper are fully applicable to the intrusion and misuse detection problems in both wired and wireless networks. The reader who desires a perspective focused only on wireless network protection is referred to papers such as Zhang et al. [7], which focuses more on dynamic changing network topology, routing algorithms, decentralized management, etc.

The remainder of this paper is organized as follows: Section II focuses on major steps in ML and DM. Section III discusses cyber security data sets used in ML and DM. Section IV describes the individual methods and related papers for ML and DM in cyber security. Section V discusses the computational complexity of different methods. Section VI describes observations and recommendations. Lastly, Section VII presents conclusions.

II. Major Steps in ML and DM

There is a lot of confusion about the terms ML, DM, and Knowledge Discovery in Databases (KDD). KDD is a full process that deals with extracting useful, previously unknown information (i.e., knowledge) from data [8]. DM is a particular step in this process—the application of specific algorithms for extracting patterns from data. The additional steps in the KDD process (data preparation, data selection, data cleaning, incorporation of appropriate prior knowledge, and proper interpretation of the results of DM) guarantee that useful knowledge is extracted from available data. However, there are many publications [e.g., Cross Industry Standard Process for Data Mining (CRISP-DM) [9]] and industry participants who call the whole KDD process DM. In this paper, following Fayyad et al. [8], DM is used to describe a particular step in KDD that deals with application of specific algorithms for extracting patterns from data.

There is a significant overlap between ML and DM. These two terms are commonly confused because they often employ the same methods and therefore overlap significantly. The pioneer of ML, Arthur Samuel, defined ML as a “field of study that gives computers the ability to learn without being explicitly programmed.” ML focuses on classification and prediction, based on known properties previously learned from the training data. ML algorithms need a goal (problem formulation) from the domain (e.g., dependent variable to predict). DM focuses on the discovery of previously unknown properties in the data. It does not need a specific goal from the domain, but instead focuses on finding new and interesting knowledge.

One can view ML as the older sibling of DM. The term data mining was introduced in late 1980s (the first KDD conference took place in 1989), whereas the term machine learning has been in use since the 1960s. Presently, the younger sibling (i.e., use of the term DM) is more popular than the older one, which might be the reason why some researchers actually label their work as DM rather than ML. This could be the reason that when queries “machine learning” AND cyber and “data mining” AND cyber were performed on Google Scholar, the first retrieved 21,300 results and the second retrieved 40,800 results. The methods used in the papers retrieved by the first query were not significantly different from those in the papers retrieved by the second query. Therefore, because this paper concentrates on methods, we will call these methods ML/DM methods.

An ML approach usually consists of two phases: training and testing. Often, the following steps are performed:

- Identify class attributes (features) and classes from training data.
- Identify a subset of the attributes necessary for classification (i.e., dimensionality reduction).
- Learn the model using training data.
- Use the trained model to classify the unknown data.

In the case of misuse detection, in the training phase each misuse class is learned by using appropriate exemplars from the training set. In the testing phase, new data are run through the model and the exemplar is classified as to whether it belongs to one of the misuse classes. If the exemplar does not belong to any of the misuse classes, it is classified as normal.

In the case of anomaly detection, the normal traffic pattern is defined in the training phase. In the testing phase, the learned model is applied to new data, and every exemplar in the testing set is classified as either normal or anomalous.

In reality, for most ML methods, there should be three phases, not two: training, validation, and testing. ML and DM methods often have parameters such as the number of layers and nodes for an ANN. After the training is complete, there are usually several models (e.g., ANNs) available. To decide which one to use and have a good estimation of the error it will achieve on a test set, there should be a third separate data set, the validation data set. The model that performs the best on the validation data should be the model used, and should not be fine-tuned depending on its accuracy on the test data set. Otherwise, the accuracy reported is optimistic and might not reflect the accuracy that would be obtained on another test set similar to but slightly different from the existing test set.

There are three main types of ML/DM approaches: unsupervised, semi-supervised, and supervised. In unsupervised learning problems, the main task is to find patterns, structures, or knowledge in unlabeled data. When a portion of the data is labeled during acquisition of the data or by human experts, the problem is called semi-supervised learning. The addition of the labeled data greatly helps to solve the problem. If the data are completely labeled, the problem is called supervised learning and generally the task is to find a function or model that explains the data. The approaches such as curve fitting or machine-learning methods are used to model the data to the underlying problem. The label is generally the business or problem variable that experts assume has relation to the collected data.

Once a classification model is developed by using training and validation data, the model can be stored so that it can be used later or on a different system. The Predictive Model Markup Language (PMML) is developed and proposed by Data Mining Group to help predictive model sharing [10]. It is based on XML and currently supports logistic regression and feed-forward neural network (NN) classifiers. The latest version (4.2) supports Naïve Bayes, k-Nearest Neighbor (k-NN), and Support Vector Machine (SVM) classifiers. The model supports several common DM metadata such as a data dictionary (e.g., discrete, Boolean, numerical), normalization, model name, model attributes, mining schema, outlier treatment, and output. Some popular data mining platforms such as Weka [11], R [12], and RapidMiner [13] support PMML models.

The CRISP-DM model [9] illustrates (see Fig. 1) commonly used phases and paradigms by DM experts to solve problems. The model is composed of the following six phases:

- *Business understanding*: Defining the DM problem shaped by the project requirements.
- *Data understanding*: Data collection and examination.
- *Data preparation*: All aspects of data preparation to reach the final dataset.

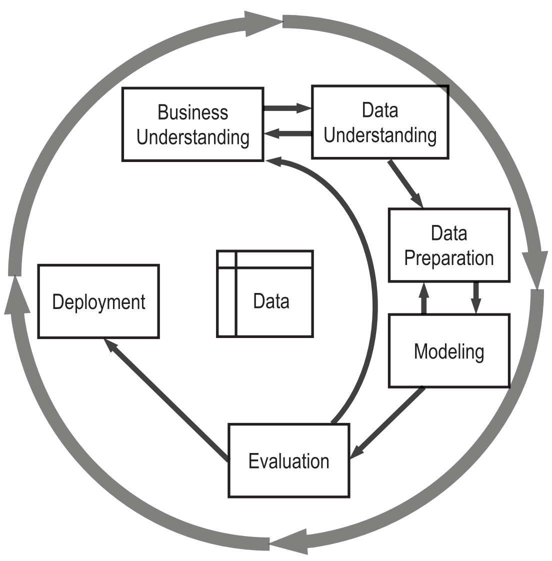


Fig. 1. CRISP-DM Process Diagram.

TABLE I
BINARY CONFUSION MATRIX

	Actual class: X	**Actual class: not X**
Predicted class: X	TP*	FP*
Predicted class: not X	FN*	TN*

*TP, TN, FP, and FN represent, respectively, True Positive, True Negative, False Positive, and False Negative

- *Modeling*: Applying DM and ML methods and optimizing parameters to fit best model.
- *Evaluation*: Evaluating the method with appropriate metrics to verify business goals are reached.
- *Deployment*: Varies from submitting a report to a full implementation of the data collection and modeling framework. Usually, the data analyst engages the phases until deployment, while the customer performs the deployment phase.

There are several classification metrics for ML/DM methods. Certain metrics are called by two or even three different names. In Section IV, the papers are described with the metric names used by the authors of the corresponding papers. To understand that section easier, the metrics with their different names are described next. For a binary classification problem, the metrics are computed from the confusion matrix (see Table I).

The metrics frequently used for binary classification (supervised learning) problems are:

- Accuracy or Proportion Correct: $(TP + TN)/(TP + TN + FP + FN)$. When classes are balanced, this is a good measure; however, when classes are unbalanced (e.g., 97% of items belong to class X and 3% to class Y, if all the items are classified as X, the accuracy would be 97% but all items from class Y would be misclassified), this metric is not very useful.
- Positive Predictive Value (PPV) or Precision: $TP/(TP + FP)$. Ratio of items correctly classified as X to all items classified as X.
- Sensitivity or Recall or True Positive Rate or Probability of Detection (P_D) or Detection Rate: $TP/(TP + FN)$.

Ratio of items correctly classified as X to all items that belong to class X.

- Negative Predictive Value (NPV): TN/(TN + FN). Ratio of items correctly classified as negatives (not X) to all items classified as not X.
- Specificity or TN Rate: TN/(TN + FP). Ratio of items correctly classified as negatives (not X) to all items that belong to class not X.
- FAR or FP Rate or Fall-out: FP/(TN + FP). FAR = 1-Specificity. Ratio of items incorrectly classified as positives (X) to all items that belong to a class not X.

In classification problems, there is a trade-off between Sensitivity and FAR (1-Specificity). This trade-off is illustrated by a Receiver Operating Characteristic (ROC) curve. ROC has FAR on the x-axis and Sensitivity on the y-axis. As the threshold for classification is changed, a different point on the ROC is chosen with different FAR and different Sensitivity. A higher FAR results in a higher Sensitivity and a lower FAR in a lower Sensitivity. The point on the ROC that provides a better classification is application dependent. Often, FAR cannot be higher than a certain number, and this is how the final classifier is chosen.

For a multi-class problem (classification into more than two classes), usually the following metrics are used:

- *Overall accuracy*: Exemplars classified correctly, all exemplars.
- *Class detection rate*: Exemplars from a given class classified correctly, all exemplars from a given class.
- *Class FAR or class FP rate*: Exemplars from a given class classified incorrectly, all exemplars not from a given class.

It is possible to compute PPV and NPV per class as well, but in the papers reviewed and described in Section IV, these metrics were not used.

There are two types of metrics for unsupervised methods: internal and external. Internal metrics are used on the data that were clustered, and class labels (because they are unknown by the clustering algorithm) are not used to compute those metrics. Metrics such as inter-cluster distance (distance between two different clusters, could be between their centroids), intra-cluster distance (distance between members of the same cluster, could be mean distance or distance between farthest members), and Dunn index (identifies dense and well-separated clusters) are frequently used.

External metrics operate on a data set for which the class labels are known. The metrics used resemble the supervised learning metrics. Several of the papers described in Section IV use unsupervised methods, but the final metrics provided are the Class Detection Rate and the Class FAR. This means that although the method was developed in an unsupervised fashion, there were labels available for the test data, so it was possible to compute classification metrics.

III. Cyber-Security Data Sets for ML and DM

For ML and DM approaches, the data is of great importance. Because these techniques learn from the available data, it is necessary to have an understanding of the data they use in order to understand how different authors applied different ML and DM algorithms. This section describes in detail the different types of data used by the ML and DM approaches—**p**acket **cap**ture (pcap), NetFlow, and other network data. Therefore, Section IV, which describes the methods in detail, cites only whether a method uses pcap, NetFlow, or other network data and does not describe the data in detail. The following subsections cover the low-level details of the data sets.

A. Packet-Level Data

There are 144 IPs listed by the Internet Engineering Task Force (IETF) including widely used protocols such as Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Internet Control Message Protocol (ICMP), Internet Gateway Management Protocol (IGMP), etc. Users' programs running these protocols generate the packet network traffic of the Internet. The network packets received and transmitted at the physical interface (e.g., Ethernet port) of the computer can be captured by a specific application programming interface (API) called pcap. Libpcap and WinPCap (the Unix and Windows versions, respectively) are the front-end packet capture software libraries for many network tools, including protocol analyzers, packet sniffers, network monitors, network IDSs, and traffic generators. A few popular programs that use pcap data are tcpdump [14], Wireshark [15], Snort [16], and Nmap [17].

At the network physical layer, an Ethernet frame is composed of the Ethernet header (i.e., Media Access Control [MAC] address), and up to 1500 bytes (Maximum Transmission Unit [MTU]) of payload. This payload contains the IP packet, which is composed of the IP (i.e., transport layer) header, and the IP payload. The IP payload might contain data or other encapsulated higher level protocols such as Network File System (NFS), Server Message Block (SMB), Hypertext Transfer Protocol (HTTP), BitTorrent, Post Office Protocol (POP) Version 3, Network Basic Input/Output System (NetBIOS), telnet, and Trivial File Transfer Protocol (TFTP).

Because the entire packet is captured by a pcap interface, the features of the data vary with respect to the protocol that packet carries. Table II lists the subsets of features captured for TCP, UDP, and ICMP. The IP addresses are in the IP header, which are handled at the Network Layer.

B. NetFlow Data

Originally, NetFlow was introduced as a router feature by Cisco. The router or switch has the ability to collect IP network traffic as it enters and exits the interface. Cisco's NetFlow version 5 defines a network flow as a unidirectional sequence of packets that share the exact same seven packet attributes: ingress interface, source IP address, destination IP address, IP protocol, source port, destination port, and IP type of service. The logical NetFlow architecture consists of three components: a NetFlow Exporter, a NetFlow Collector, and an Analysis Console. Currently, there are 10 versions of NetFlow. Versions 1 to 8 are similar, but starting with version 9, NetFlow differs significantly. For versions 1 to 8, the feature set in Table III

TABLE II
PACKET HEADERS OF CYBER-SECURITY DATASETS

IP Header (IPv4)	
Internet Header Length	The number of 32-bit words in the header
Total Length	The entire packet size, including header and data, in bytes
Time To Live	This field limits a datagram's lifetime, in hops (or time)
Protocol	The protocol used in the data portion of the IP datagram
Source address	This field is the IPv4 address of the sender of the datagram
Destination address	This field is the IPv4 address of the receiver of the datagram
TCP Packet	
Source port	Identifies the sending port
Destination port	Identifies the receiving port
Sequence number	Initial or accumulated sequence number
Acknowledgement number	The next sequence number that the receiver is expecting
Data offset	Specifies the size of the TCP header in 32-bit words
Flags (control bits)	NS, CWR, ECE, URG, ACK, PSH, RST, SYN, FIN
UDP Packet	
Source port	Identifies the sending port
Destination port	Identifies the receiving port
Length	The length in bytes of the UDP header and UDP data
ICMP Packet	
Type	Control (e.g., ping, destination unreachable, trace route)
Code	Details with the type
Rest of Header	More details

TABLE III
NETFLOW PACKET HEADER OF CYBER SECURITY DATASETS

NetFlow Data – Simple Network Management Protocol (SNMP)	
Ingress interface (SNMP ifIndex)	Router information
Source IP address	
Destination IP address	
IP protocol	IP protocol number
Source port	UDP or TCP ports; 0 for other protocols
Destination port	UDP or TCP ports; 0 for other protocols
IP Type of Service	Priority level of the flow
NetFlow Data – Flow Statistics	
IP protocol	IP protocol number
Destination IP address	
Source IP address	
Destination port	
Source port	
Bytes per packet	The flow analyzer captures this statistic
Packets per flow	Number of packets in the flow
TCP flags	NS, CWR, ECE, URG, ACK, PSH, RST, SYN, FIN

presents the minimum set of NetFlow data variables for a unidirectional sequence of packets (i.e., a flow).

NetFlow data include a compressed and preprocessed version of the actual network packets. The statistics are derived features and, based on certain parameters such as duration of window, number of packets, etc., set the NetFlow settings on the device.

C. Public Data Sets

The Defense Advanced Research Projects Agency (DARPA) 1998 and DARPA 1999 data sets [18], [19] are extensively used in experiments and frequently cited in publications. The DARPA 1998 set was created by the Cyber Systems and Technology Group of the Massachusetts Institute of Technology Lincoln Laboratory (MIT/LL). A simulation network was built and data were compiled based on TCP/IP network data, Solaris Basic Security Module log data, and Solaris file system dumps for user and root. Effectively, the assembled data set was composed of network and operating system (OS) data. The data were collected for 9 weeks, with the first 7 assigned as the training set and last 2 assigned as the testing set. Attack simulations were organized during the training and testing weeks.

Similarly, the DARPA 1999 data set was collected for a total of 5 weeks, with the first 3 assigned as the training set and the last 2 assigned as the testing set. This data set had substantially more attack types than the DARPA 1998 data set. In both collections, the data sets were processed and curated to be used in the experiments. The TCP dumps and logs were combined into one stream with many columns.

One of the most widely used data sets is the KDD 1999 data set [20], which was created for the KDD Cup challenge in 1999. The data set is based on DARPA 1998 TCP/IP data and has basic features captured by pcap. Additional features were derived by analyzing the data with time and sequence windows. The data set has three components—basic, content, and traffic features—making for a total of 41 attributes. The KDD 1999 data set is similar to NetFlow data, but has more derived and detailed features because the attacks were simulated. The complete list can be found in Table IV.

The KDD 1999 data set (with about 4 million records of normal and attack traffic) has been analyzed comprehensively by Tavallaee et al. [21] and found to have some serious limitations. A few inherent problems were noted, such as synthesizing the network and attack data (after sampling the actual traffic) because of privacy concerns, an unknown number of dropped packets caused by traffic overflow, and vague attack definitions. Tavallaee et al. also performed statistical evaluations and their own classification experiments. They reported a huge number of redundant records (78% in the training data and 75% in test data) causing bias. In addition, in the classification experiments the group conducted, they pointed out that by randomly selecting subsets of the training and testing data, often very-high, unrealistic accuracies can be achieved. They proposed a new data set, NSL-KDD, that consists of selected records of the complete KDD data set and does not experience the aforementioned shortcomings.

The DARPA 1998 set defines four types of attacks: Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L), and Probe or Scan. A DoS attack is an attempt to deny to the aimed users computing or network resources. A U2R attack grants root access to the attacker. An R2L attack grants local network access to the attacker. Probe or Scan attacks collect information about the network resources. DARPA 1999 added

TABLE IV
FEATURES OF TCP CONNECTION

Basic Features		
duration	integer	duration of the connection
protocol_type	nominal	protocol type of the connection: TCP, UDP, and ICMP
service	nominal	http, ftp, smtp, telnet... and other
flag	nominal	connection status
src_bytes	integer	bytes sent in one connection
dst_bytes	integer	bytes received in one connection
land	binary	if src/dst IP address and port numbers are same, then 1
wrong_fragment	integer	sum of bad checksum packets in a connection
urgent	integer	sum of urgent packets in a connection
Content Features		
hot	integer	sum of hot actions in a connection such as: entering a system directory, creating programs and executing programs
num_failed_logins	integer	number of incorrect logins in a connection
logged_in	binary	if the login is correct, then 1, else 0
num_compromised	integer	sum of times appearance "not found" error in a connection
root_shell	binary	if the root gets the shell, then 1, else 0
su_attempted	binary	if the su command has been used, then 1, else 0
num_root	integer	sum of operations performed as root in a connection
num_file_creations	integer	sum of file creations in a connection
num_shells	integer	number of logins of normal users
num_access_files	integer	sum of operations in control files in a connection
num_outbound_cmds	integer	sum of outbound commands in an ftp session
is_hot_login	binary	if the user is accessing as root or admin
is_guest_login	binary	if the user is accessing as guest, anonymous, or visitor
Traffic Features – Same Host – 2-second Window		
duration	integer	duration of the connection
protocol_type	nominal	protocol type of the connection: TCP, UDP, and ICMP
service	nominal	http, ftp, smtp, telnet... and other
flag	nominal	connection status
src_bytes	integer	bytes sent in one connection
dst_bytes	integer	bytes received in one connection
land	binary	if src/dst IP address and port numbers are same, then 1
wrong_fragment	integer	sum of bad checksum packets in a connection
urgent	integer	sum of urgent packets in a connection
Traffic Features – Same Service – 100 Connections		
duration	integer	duration of the connection
protocol_type	nominal	protocol type of the connection: TCP, UDP, and ICMP
service	nominal	http, ftp, smtp, telnet... and other
flag	nominal	connection status
src_bytes	integer	bytes sent in one connection
dst_bytes	integer	bytes received in one connection
land	binary	if src/dst IP address and port numbers are same, then 1
wrong_fragment	integer	sum of bad checksum packets in a connection
urgent	integer	sum of urgent packets in a connection
urgent	integer	sum of urgent packets in a connection

a new attack type—one where the attacker attempts to exfiltrate special files that have to remain on the victim computer.

IV. ML AND DM METHODS FOR CYBER

This section describes the different ML/DM methods for cyber security. Each technique is described with some detail, and references to seminal works are provided. Also, for each method, two to three papers with their applications to cyber domain are presented.

A. Artificial Neural Networks

ANNs are inspired by the brain and composed of interconnected artificial neurons capable of certain computations on their inputs [22]. The input data activate the neurons in the first layer of the network whose output is the input to the second layer of neurons in the network. Similarly, each layer passes its output to the next layer and the last layer outputs the result. Layers in between the input and output layers are referred to as hidden layers. When an ANN is used as a classifier, the output layer generates the final classification category.

ANN classifiers are based on the perceptron [23] and were very popular until the 1990s when SVMs were invented. Compared to the convex quadratic optimization applied in an SVM, ANNs often suffer from local minima and thus long runtimes during learning. Unlike an SVM, as the number of features in an ANN increase, its learning runtime increases. With one or more hidden layers, the ANN is able to generate nonlinear models. The back-propagation feature of the ANN makes it possible to model EX-OR logic.

With developments in this field such as recurrent, feed-forward, and convolutional NNs, ANNs are gaining in popularity again, and at the same time winning many prizes in recent pattern recognition contests (these contests are not yet related to cyber intrusion detection). Because the advanced versions of ANNs require even more processing power, they are implemented commonly on graphics processing units.

1) Misuse Detection: Cannady [24] used ANNs as the multi-category classifier to detect misuse. He used data generated by a RealSecure$^{\text{TM}}$ network monitor, which has the attack signatures built into the system. Ten thousand events were collected by the monitor, of which 3000 came from simulated attacks. The attacks were simulated by the Internet Scanner [25] and Satan [26] programs.

The data preprocessing stage resulted in the selection of nine features: protocol identifier (ID), source port, destination port, source address, destination address, ICMP type, ICMP code, raw data length, and raw data. Ten percent of the data was selected randomly for testing. The study then used the remaining normal and attack data to train an ANN, which learned the combined signatures. The paper expressed that the findings were preliminary and reported the error rates for training and testing, which were 0.058 and 0.070 root-mean-square (RMS) errors, respectively. Although the details were not disclosed, the output of the ANN was a number between 0 and 1 representing each of the two categories (attack and normal). Therefore, an RMS of 0.070 can be roughly considered as 93% accuracy for testing phase. Each packet or data instance was categorized as either a normal or an attack group.

2) Anomaly Detection and Hybrid Detection: Lippmann and Cunningham [27] proposed a system that uses keyword selection and artificial neural networks. The keyword selection was performed on transcripts of telnet sessions and statistics were computed of the number of times each keyword (from a predetermined list) occurred. The keyword statistics constitute the input to a neural network that provides an estimate of the posterior probability of an attack. A second neural network operates on the instances that were flagged as an attack, and attempts to classify them (i.e., provide an attack name). Both neural networks consisted of multilayer perceptrons with no hidden units. The system achieves 80% detection with roughly 1 false alarm per day. This false alarm rate represents a two orders of magnitude improvement from the baseline system with the same detection accuracy.

Bivens et al. [28] describe a complete IDS that employs a preprocessing stage, clustering the normal traffic, normalization, an ANN training stage, and an ANN decision stage. The first stage used a Self-Organizing Map (SOM), which is a type of unsupervised ANN, to learn the normal traffic patterns over time, such as commonly used TCP/IP port numbers. In this manner, the first stage quantized the input features into bins, which were then fed to the second stage, a Multilayer Perceptron (MLP) ANN. The MLP network parameters, such as the number of nodes and layers, were determined by the first-stage SOM. Once the MLP training was completed, it started predicting the intrusions. The system can be restarted for a new SOM to learn a new traffic pattern and a new MLP attack classifier to be trained. The study used TCP/IP data from the DARPA 1999 challenge [18], [19], where the data set consisted of network packet-level data. Unlike the previous study by Cannady [24], which classified each packet-level data separately, this system used time windows to perform the detection and classified a group of packets. Thus, the system was able to detect attack types of longer duration. Because the input was low-level network packet data (as opposed to NetFlow data), the granularity is high and the produced predictions still correspond to short durations. Bivens et al. [28] reported successfully predicting 100% of the normal behavior. Their overall approach is promising, even though some attacks were not fully predicted and the FAR for some attacks reached 76%.

B. Association Rules and Fuzzy Association Rules

The goal of Association Rule Mining is to discover previously unknown association rules from the data. An association rule describes a relationship among different attributes: IF (A AND B) THEN C. This rule describes the relationship that when A and B are present, C is present as well. Association rules have metrics that tell how often a given relationship occurs in the data. The support is the prior probability (of A, B, and C), and the confidence is the conditional probability of C given A and B. Association Rule Mining was introduced by Agrawal et al. [29] as a way to discover interesting co-occurrences in supermarket data. It finds frequent sets of items (i.e., combinations of items that are purchased together in at least N transactions in the database), and from the frequent items sets such as {X, Y}, generates association rules of the form: X $\rightarrow$ Y and/or Y $\rightarrow$ X.

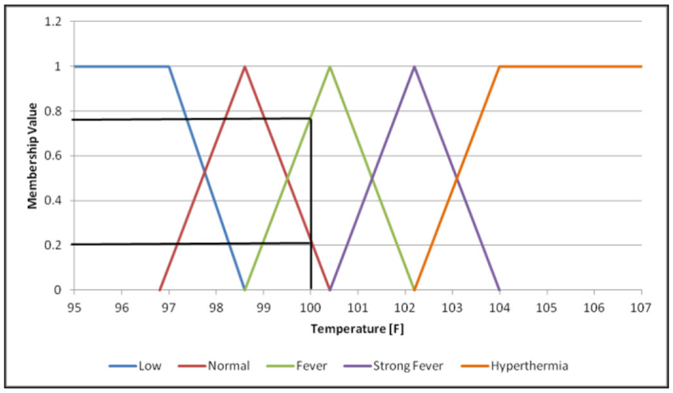


Fig. 2. Membership Functions for the Fuzzy Variable Human Body Temperature: Low, Normal, Fever, Strong Fever, and Hypothermia.

A simple example of an association rule pertaining to the items that people buy together is:

$$\text{IF (Bread AND Butter)} \rightarrow \text{Milk} \tag{1}$$

This rule states that if a person buys bread and butter, they also buy milk.

A limitation of traditional Association Rule Mining is that it only works on binary data [i.e., an item was either purchased in a transaction (1) or not (0)]. In many real-world applications, data are either categorical (e.g., IP name, type of public health intervention) or quantitative (e.g., duration, number of failed logins, temperature). For numerical and categorical attributes, Boolean rules are unsatisfactory. An extension that can process numerical and categorical variables is called Fuzzy Association Rule Mining [30].

Fuzzy association rules are of the form:

$$\text{IF (X is } \boldsymbol{A}) \rightarrow (\text{Y is } \boldsymbol{B}) \tag{2}$$

where X and Y are variables, and ***A*** and ***B*** are fuzzy sets that characterize X and Y, respectively. A simple example of fuzzy association rule for a medical application could be the following:

IF (Temperature is ***Strong Fever***) AND (Skin is ***Yellowish***) AND (Loss of appetite is ***Profound***) $\rightarrow$ (Hepatitis is ***Acute***)

The rule states that if a person has a ***Strong Fever***, ***Yellowish*** skin and ***Profound Loss*** of appetite, then the person has ***Acute*** Hepatitis. ***Strong Fever***, ***Yellowish***, ***Profound***, and ***Acute*** are membership functions of the variables Temperature, Skin, Loss of appetite, and Hepatitis, respectively. As an example of fuzzy membership functions, the membership functions for the variable Temperature are shown in Fig. 2.

According to the definition in the diagram, a person with a 100 °F temperature has a ***Normal*** temperature with a membership value of 0.2 and has a ***Fever*** with a membership value of 0.78. The use of fuzzy membership functions allows reasoning using linguistic terms. Those linguistic terms for human body temperature are ***Low***, ***Normal***, ***Fever***, ***Strong Fever***, and ***Hypothermia***. More information on fuzzy logic and fuzzy membership functions can be found in [31].

1) Misuse Detection: The study by Brahmi [32] is a good example of association rules applied to the DARPA 1998 data set to capture the relationships between TCP/IP parameters and attack types. In the rules, the antecedents are extracted from the DARPA 1998 data set and the consequents are the attack types. The work explains multidimensional Association Rule Mining, in which there is more than one antecedent in the rules, such as (IF (service AND src_port AND dst_port AND num_conn) THEN attack_type), which is an example of a four-dimensional rule. The work includes finding the rules with high support and high confidence. The best performance is achieved using six-dimensional rules with the detection rates 99%, 95%, 75%, 87% for the attack types DoS, Probe or Scan, U2R, and R2L, respectively. Experiments were not performed with higher dimensions because of computational cost. One of the main advantages of Association Rule Mining is that the discovered rules explain the relationships clearly. The approach is promising for building attack signatures.

Zhengbing et al. [33] proposed a novel algorithm based on the Signature a priori algorithm [34] for finding new signatures of attacks from existing signatures of attacks. They compared their algorithm's processing time with that of Signature a priori and found that their algorithm has a shorter processing time, and the difference in processing times increases with increasing size of the database. The contribution of that paper is its description of a novel method for finding new attack signatures from existing ones. Such an algorithm could be used for obtaining new signatures for inclusion into misuse detection systems, such as Snort [16].

2) Anomaly Detection and Hybrid Detection: The NETMINE framework [35] performs data stream processing, refinement analysis (by capturing association rules from traffic data), and rule classification. Data capture is performed simultaneously with online stream analysis. Traffic packets are captured by network capture tools developed at Politecnico di Torino [36] running on a backbone link of the campus network. The data captured are NetFlow data with attributes such as source address, destination address, destination port, source port, and flow size (in bytes).

NETMINE performs generalized association rule extraction to detect anomalies and identify recurrent patterns. Individual association rules (e.g., for one specific IP address) might be too detailed and have very low support. Generalized association rules (e.g., for a subnet traffic) allow raising the abstraction level at which correlations are represented. Generalized rules extraction is performed by the novel Genio algorithm [37], which is more effective than previous approaches for mining generalized rules. The generalized association rule extraction process is not designed to be performed in real time during data capture. However, some experiments show the feasibility of the approach for appropriate sliding window update frequencies.

Rule classification organizes the rules according to their semantic interpretation. Three basic classes of rules are defined:

- Traffic flow rules involve source and destination addresses.
- Provided services rules consist of destination port (i.e., the service) and destination address (i.e., the service provider).
- Service usage rules have destination port and source address (i.e., the service user).

The extracted rules are meant to assist the network analyst in quickly determining patterns that are worth further investigation. There is no automatic classification into normal and anomalous categories.

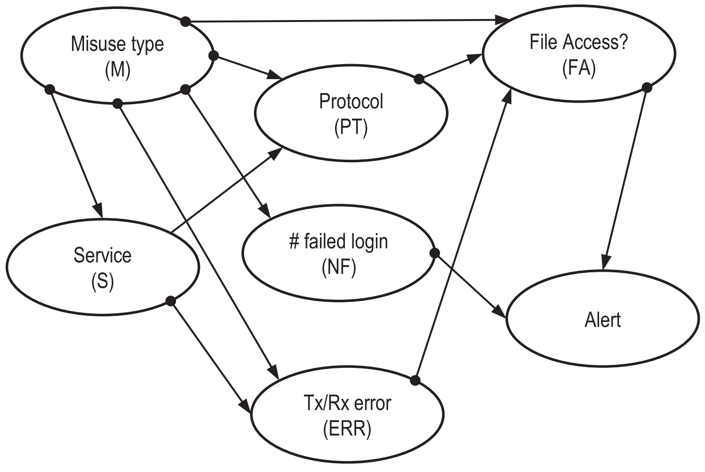
The work by Tajbakhsh et al. [38] used the KDD 1999 data set to perform Fuzzy Association Rule Mining to discover common relationship patterns. The study used the corrected version of the KDD set (see Section III) with approximately 300,000 instances. They used a clustering approach to define the fuzzy membership functions of the attributes, claiming it performs better than histogram-based approaches. To reduce the *items* (according to the paper, there are 189 different items: 31 numerical attributes with 3 values make 93 items and 10 nominal attributes with a total of 96), the study used an association hyper-edge method. For example, the item set {a, b} is considered a hyper edge if the average confidence of the rules ($a \rightarrow b$ and $b \rightarrow a$) is greater than a threshold. The work uses the thresholds of 98% for association hyper-edge reduction and 50% for confidence. The anomaly detection performance is reported as 100% accurate with a 13% FP rate. The performance drops quickly as the FP rate is reduced. The paper also states the benefits of the Association Rule Mining approach, such as human-comprehendible rules, easier handling of symbolic (nominal) attributes, and efficient classification on large data sets.

Integrating fuzzy logic with frequency episodes was attempted by Luo and Bridges [39] to find the fuzzy frequency episodes that represent the frequent fuzzy sequences in the data. The fuzzification (i.e., quantization of data with overlapping bins) helps to handle the numerical variables and to explain the variables to the cyber security analyst. The frequency episodes are determined by a user-supplied threshold. Effectively, the overall approach is similar to sequence mining. The experiment used data collected by tcpdump from a server on a university campus. Main features from the data contained TCP flags and port numbers, and they are quantized by fuzzy logic. Intrusion simulations were conducted by custom programs. The study reports the similarity measurements between the training and testing data sets. The highest reported similarity is 0.82. Although the work did not report performance measures, the overall approach is promising, thereby extending and improving the previous approaches in the literature.

C. Bayesian Network

A Bayesian network is a probabilistic graphical model that represents the variables and the relationships between them [40], [41]. The network is constructed with nodes as the discrete or continuous random variables and directed edges as the relationships between them, establishing a directed acyclic graph. The child nodes are dependent on their parents. Each node maintains the states of the random variable and the conditional probability form. Bayesian networks are built using expert knowledge or using efficient algorithms that perform inference.

Fig. 3 gives an example of a Bayesian network for attack signature detection. Each state (or network variable) can be an



File Access state input variables and values	P(FA = True)	P(FA = False)
M=R2H, PT=NSF, ERR=0	0.95	0.05
M=R2H, PT=FTP, ERR=0	0.99	0.01
M=Probe, PT=none, ERR=50%	0.80	0.20
M=Probe, PT=PING, ERR=0	0.50	0.50
M=DoS, PT=POP, ERR=100%	0.80	0.20
M= DoS, PT=HTTP, ERR=50%	0.90	0.10

Fig. 3. Example Bayesian Network for Signature Detection.

input to other states with certain set of state values. For example, the **protocol** state can pick values from available protocol numbers. Each of the state values that can go from a state to another state have an associated probability, and the sum of those probabilities will add up to 1 representing the entire set of state values. Depending on the application, the network can be used to explain the interplay between the variables or to calculate a probable outcome for a target state (e.g., **alert** or **file access**) using the input states.

The probability tables can be calculated from the available training data. Inferring unobserved variables, parameter learning, and structure learning are among the main tasks for training Bayesian networks.

1) Misuse Detection: In general, anomaly detection can be considered as reactive because the system responds to an input when the input is unexpected. Conversely, in a misuse detection problem, the system is proactive because the signatures extracted from the input are being checked continuously against a list of attack patterns. Taking the proactive approach as in misuse detection requires classifying the network streams. In their work, Livadas et al. [42] compared several methods applied to DoS problem. Their work tried to resolve the botnet traffic in filtered Internet Relay Chat (IRC) traffic, hence determining botnet existence and its origins. The study uses TCP-level data collected from 18 locations on Dartmouth University's wireless campus network over a period of 4 months. The TCP (IRC-based botnets do not use UDP or ICMP) data are used to generate the network streams or NetFlow data. A filter layer is used to extract IRC data from all network data.

Because the ground truth is hard to obtain, the study used certain aspects of IRC communication to tag data with IRC class labels. To label the IRC-botnet generated data is even harder, so the study uses simulated data for the experiments. The performance of the Bayesian network is reported as 93% precision with a FP rate of 1.39%. The other two classifiers, Naïve Bayes and C4.5 decision tree, both achieve 97% precision. However, their FP rates are higher—1.47% and 8.05% respectively.

In another study, Jemili et al. [43] suggested an IDS framework using Bayesian network classifiers. The work used nine features of the KDD 1999 data in the inference network. In the anomaly detection stage, the normal or attack decision is made by a junction tree inference module with a performance of 88% and 89% on the normal and attack categories, respectively. In the next phase, the attack types were recognized from the data labeled as attack data by the anomaly detection module. Performances of 89%, 99%, 21%, 7%, and 66% are reported for the DoS, Probe or Scan, R2L, U2R, and other classes, respectively. The study suggests the low performance of the R2L and U2R categories is because the number of training instances is much lower than for the other categories.

2) Anomaly Detection and Hybrid Detection: When the computing platform receives TCP/IP packets, the network stack of the underlying OS processes these packets. The network stack generates various logs and system kernel calls, and ultimately the packet data are processed at the application level invoked by the kernel. Kruegel et al. [44] used a Bayesian network to classify events during open and executive OS calls. The DARPA 1999 data set is used to excite the OS kernel by TCP/IP packets. Then a set of attributes based on these system calls, such as system call argument string length and distribution and character distribution, are compared using a Pearson test. Moreover, the structure of the string with respect to grammar (command syntax) and tokens is searched in the parameters of the system calls. These features are used in a Bayesian network to calculate the probability of a normal state or an anomaly state. Because the detection threshold is used to control the FAR, the system is flexible and can make self-adjustments against too many false alarms; 75% accuracy, 0.2% FAR and 100% accuracy, and 0.1% FAR are achieved by using different threshold values.

Determining complex attacks often requires observing the anomalies and the correlations between them to uncover scenarios or attack plans. To achieve this goal, alert correlation has been studied to reduce the volume of alerts generated by the system. Clustering, similarity measures, or expert knowledge is used to correlate anomalies and then reveal the complex patterns of attack scenarios.

A DoS intrusion detector that uses a Bayesian network is described by Benferhat et al. [45]. There is only one parent node representing the hidden variable (i.e., class – {normal, anomaly}), and the observed variables are child nodes. The child nodes are assumed to be statistically independent. The main goal of this approach is to perform alert correlation using historical data with minimal use of expert knowledge. The DARPA 2000 data set is used in the experiments. For each intrusion objective, a network is constructed. The study setup has two different scenarios extracted from the DARPA data set; the system detected one of the scenarios successfully but failed to detect the other. Unfortunately, the study does not report any numerical results.

D. Clustering

Clustering [46] is a set of techniques for finding patterns in high-dimensional unlabeled data. It is an unsupervised pattern discovery approach where the data are grouped together based on a similarity measure. The main advantage of clustering for intrusion detection is that it can learn from audit data without requiring the system administrator to provide explicit descriptions of various attack classes.

There are several approaches for clustering the input data. In connectivity models (e.g., hierarchical clustering), data points are grouped by the distances between them. In centroid models (e.g., k-means), each cluster is represented by its mean vector. In distribution models (e.g., Expectation Maximization algorithm), the groups are assumed to be acquiescent to a statistical distribution. Density models group the data points as dense and connected regions (e.g., Density-Based Spatial Clustering of Applications with Noise [DBSCAN]). Lastly, graph models (e.g., clique) define each cluster as a set of connected nodes (data points) where each node has an edge to at least one other node in the set.

An instance-based learning (also called lazy learning) algorithm, the k-NN, is another popular ML method where the classification of a point is determined by the k nearest neighbors of that data point. An example of this approach for cyber is described in [47]. Determining the category of the point exercises a majority voting, which can be a drawback if the class distribution is skewed. Because high-dimensional data affect the k-NN methods negatively (i.e., curse of dimensionality), a feature reduction is almost always required. The choice of the neighborhood order or the order in the data set is also considered to have high impact. Among the benefits of k-NN is its simplicity and absence of parametric assumptions (except the number k).

In graph theory, the clustering coefficient represents the affinity of the nodes that are close to each other [48]. Certain types of data such as social networks have high clustering coefficients. The global clustering coefficient is defined as the ratio of number of closed triplets to the number of connected triplets of vertices. The local clustering coefficient of a node is defined as the ratio of the number of sub-graphs with three edges and three vertices that the node is part of to the number of triples which the node is part of [49]. The coefficient is between 0 and 1, 0 being a single node and 1 being every neighbor connected to the node.

1) Misuse Detection: In the literature, there are fewer applications of the clustering methods for misuse detection than for anomaly detection. However, as demonstrated by one study [50], real-time signature generation by an anomaly detector can be an important asset. A density-based clustering scheme called Simple Logfile Clustering Tool (SLCT) is used to create clusters of normal and malicious network traffic. To differentiate between the normal and anomalous traffic, the study used a parameter M to set the percentage of fixed features that the cluster contains. A fixed feature for the cluster corresponds to a constant value for that feature. The study defines as a dense one-region the feature values that are at least in N percent of the instances. This is essentially the support value of a single antecedent, one of the metrics that Association Rule Mining uses. This way, when M is zero, all of the data are clustered and when M has a high value, ideally only malicious clusters remain. As an example, by setting the value of M to 97%, the study detects 98% of the attack data with a 15% FAR. The method can detect previously unseen (new or zero-day) attacks. After the clustering stage with the specified parameters, all of the clusters are considered as attacks that treat the cluster centroids as the signatures.

The system is composed of two clustering schemes: the first is used for the normal or attack detection (as previously described), and the second is used in a supervised manner to determine the normal traffic. The difference between them is the modification of parameter setting, practically having two clustering schemes to detect normal and anomalous traffic in parallel. The output of this stage goes to a rule-based module to extract signatures to be used by either the system or the cyber security professionals. One of the novelties in this study is that each anomalous cluster centroid is treated as a signature to be filtered out by the system (e.g., after hourly or daily signature updates by the system).

The study used the KDD data set in several experiments. The data sets were prepared with attack percentages of 0%, 1%, 5%, 10%, 25%, 50%, and 80%. Performance metrics such as cluster integrity were used in addition to accuracy. The study reported their performance as 70% to 80% detection rate for previously unknown attacks. The results are impressive, especially given the fact that the system had no prior knowledge of any of the attacks or attack patterns in the KDD data set.

2) Anomaly Detection and Hybrid Detection: In their study, Blowers and Williams [51] use a DBSCAN clustering method to group normal versus anomalous network packets. The KDD data set is preprocessed to select features using a correlation analysis. Although the actual FAR is not reported, the study uses the threshold of the clustering method for controlling the system's FAR. A 10% attack to no-attack ratio is set during preprocessing of the data. The reported performance is 98% for attack or no-attack detection. This is a very high value for an anomaly detector based on clustering, higher than most studies in the literature. Overall, the study presents a good example of summarizing the application of ML methods to cyber operations.

Sequeira and Zaki [52] captured 500 sessions with a long stream of commands from nine users at Purdue University. User command level (shell commands) data were used to detect whether the user is a regular user or an intruder. Each user's command stream in a session was parsed into tokens and subsequently represented as a sequence of tokens. Sequeira and Zaki tried several approaches involving sequences, such as sequence similarity measures and sequence matching algorithms. One of the promising approaches used was based on the longest common subsequence metric.

The command captures from nine users suggest that the data are small, possibly due to the processing requirements. It is not clear how many commands a typical session includes, but there is a good chance it is much longer than the maximum sequence length, which the study picked as 20. The study stated its performance of 80% accuracy with 15% FAR as a successful result. Sequeira and Zaki [52] refer to previous studies that used the same data set and achieved 74% with 28% FAR and point to an improvement. It could also be a good idea to use DARPA or KDD data set server logs for comparison.

E. Decision Trees

A decision tree is a tree-like structure that has leaves, which represent classifications and branches, which in turn represent the conjunctions of features that lead to those classifications. An exemplar is labeled (classified) by testing its feature (attribute) values against the nodes of the decision tree. The best known methods for automatically building decision trees are the ID3 [53] and C4.5 [54] algorithms. Both algorithms build decision trees from a set of training data using the concept of information entropy. When building the decision tree, at each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of examples into subsets. The splitting criterion is the normalized information gain (difference in entropy). The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then performs recursion on the smaller subsets until all the training examples have been classified.

The advantages of decision trees are intuitive knowledge expression, high classification accuracy, and simple implementation. The main disadvantage is that for data including categorical variables with a different number of levels, information gain values are biased in favor of features with more levels. The decision tree is built by maximizing the information gain at each variable split, resulting in a natural variable ranking or feature selection. Small trees (such as the one depicted in Fig. 4) have an intuitive knowledge expression for experts in a given domain because it is easy to extract rules from those trees just by examining them. For deeper and wider trees, it is much more difficult to extract the rules and thus the larger the tree, the less intuitive its knowledge expression. Smaller trees are obtained from bigger ones by pruning. Larger trees often have high classification accuracy but not very good generalization capabilities. By pruning larger trees, smaller trees are obtained that often have better generalization capabilities (they avoid over-fitting). Decision tree building algorithms (e.g., C4.5) are relatively simpler than more complex algorithms such as SVMs. As such, they have also a simpler implementation.

1) Misuse Detection: Most misuse detection systems perform detection by comparing each input to all rules (signatures). Snort [16], a well-known open-source tool, follows the signature-based approach. In Snort, each signature has a single line description. The matching process between input data and signatures is usually slow (especially when the number of signatures is large) and therefore inefficient to use in the front end.

Kruegel and Toth [55] replaced the misuse detection engine of Snort by decision trees. First they performed a clustering of rules used by Snort 2.0 and then derived a decision tree using a variant of the ID3 algorithm. Rule clustering minimizes the number of comparisons necessary to determine which rules are triggered by given input data. The decision tree picks the most discriminating features of the rule set, thus allowing parallel evaluation of every feature. This yields performance much superior to that of Snort.

The proposed technique was applied to tcpdump files from the 10 days of test data produced by MIT/LL for the 1999 DARPA intrusion detection evaluation. For that data set, the speed of operation of Snort and the decision-tree technique were compared. The actual performance gain varies considerably depending on the type of traffic; the maximum speed-up was 105%, the average 40.3%, and the minimum 5%. Experiments were also performed by increasing the number of rules from 150 to 1581 (full set used by Snort 2.0). With increasing number of rules, the speed-up of the decision tree method over Snort 2.0 is even more pronounced.

This study showed that clustering methods coupled with decision trees can substantially reduce processing time of a misuse detection system, possibly allowing them to be efficiently used in the front end.

2) Anomaly Detection and Hybrid Detection: EXPOSURE [56], [57] is a system that employs large-scale, passive Domain Name Service (DNS) analysis techniques to detect domains that are involved in malicious activity. The system consists of five main components: Data Collector, Feature Attribution Component, Malicious and Benign Domains Collector, Learning Module, and Classifier. The Classifier is built by the Weka J48 decision tree program, which is an implementation of the C4.5 algorithm capable of generating pruned or unpruned decision trees. Experimentation showed that the minimum error was achieved when all features were combined, and therefore all 15 features were used by the decision tree.

The data used consist of DNS data collected over a 2.5-month period (100 billion DNS queries that resulted in 4.8 million distinct domain names). The study examined several thousand malicious and benign domains and used them to construct the training set. Malicious domains were obtained from www.malwaredomains.com, the Zeus Block List, Anubis, etc. The initial malicious domain list consists of 3500 domains. Benign domains were from the Alexa top 1000 domains list.

By experimenting with different period lengths' values, the study determined that the optimal period of initial training for the system was 7 days. After this initial training, the classifier was retrained every day. The results vary considerably depending on the data set. Overall, using tenfold cross-validation, the detection accuracy of malicious domains was 98.5% and the FAR was 0.9%.

Additional experiments were performed to determine whether the method could detect malicious domains that were

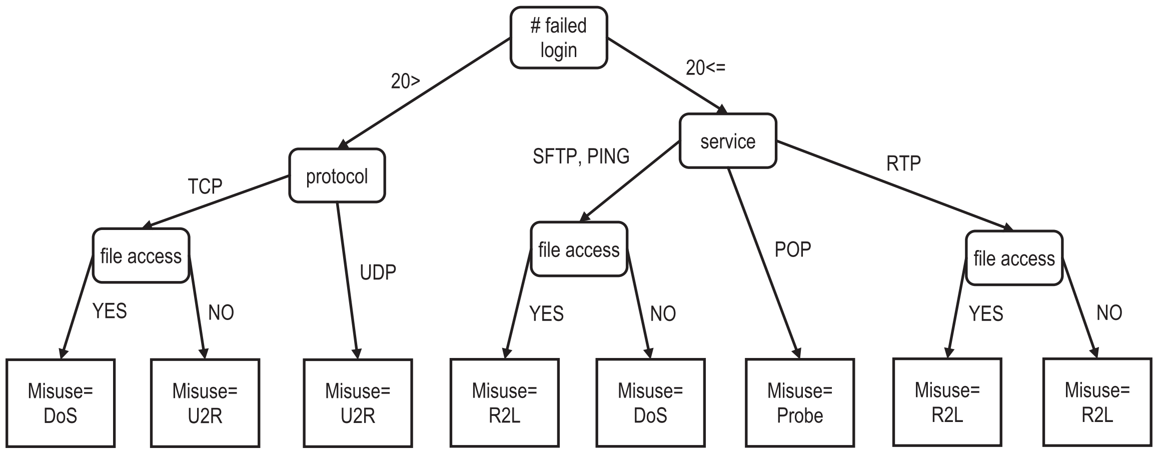


Fig. 4. An Example Decision Tree.

not present in the training set. In the first experiment, among the 50 randomly chosen domains from the list of 17,686 domains, the classifier detected three benign domains as being malicious (6% FP rate). In the second experiment, the study automatically crosschecked the malicious and suspicious domains identified by the classifier using online site rating tools such as McAfee Site Advisor, Google Safe Browsing, and Norton Safe Web. The FP rate was 7.9% for the malicious domains identified. In the third experiment, EXPOSURE classified 100 million DNS queries during a 2-week period, detected 3117 new malicious domains (previously not known to the system and not used in training), and did not generate any FPs during that time.

The experimental results show that the accuracy and FAR of EXPOSURE are satisfactory and that EXPOSURE is useful in automatically identifying a varied category of malicious domains (e.g., botnet command and control servers, phishing sites, scam hosts). The ability of EXPOSURE to detect a high number of previously unknown malicious domains from DNS traffic is an important achievement.

F. Ensemble Learning

In general, supervised learning algorithms search the hypothesis space to determine the right hypothesis that will make good predictions for a given problem. Although good hypotheses might exist, it may be hard to find one. Ensemble methods combine multiple hypotheses, hoping to form a better one than the best hypothesis alone. Often, ensemble methods use multiple weak learners to build a strong learner [58].

A weak learner is one that consistently generates better predictions than random. One of the ensemble algorithms uses boosting to train several weak learning algorithms and combine (i.e., summation) their weighted results. Adaptive Boosting (AdaBoost) [59] is one of the more popular algorithms used to reduce the over-fitting problem inherent to ML. Boosting can be seen as a linear regression where data features are the input to the weak learner h (e.g., a straight line dividing the input data points into two categories in space), and the output of the boosting is the weighted summation of these h functions. Long and Servedio [60] criticized boosting by suggesting that a non-zero fraction of mislabeled data can cause boosting to fail completely and result in an ROC of 0.5 (which is equivalent to random guess).

Bagging (bootstrap aggregating) is a method to improve the generality of the predictive model to reduce over-fitting. It is based on a model-averaging technique and known to improve the 1-nearest neighbor clustering performance.

The Random Forest classifier [61] is an ML method that combines the decision trees and ensemble learning. The forest is composed of many trees that use randomly picked data features (attributes) as their input. The forest generation process constructs a collection of trees with controlled variance. The resulting prediction can be decided by majority voting or weighted voting.

Random Forests have several advantages: a low number of control and model parameters; resistance to over-fitting; no requirement for feature selection because they can use a large number of potential attributes. One important advantage of Random Forest is that the variance of the model decreases as the number of trees in the forest increases, whereas the bias remains the same. Random Forests also have some disadvantages such as low model interpretability, performance loss due to correlated variables, and dependence on the random generator of the implementation.

1) Misuse Detection: Zhang et al. [62] approach the problem of misuse detection by employing an outlier detection module at the front end of their system. If the input is classified as abnormal network traffic, the data are further classified as belonging to one of the attack categories of KDD 1999 data set. The study provides a full system solution including an outlier detector, a signature-based attack predictor, and a pattern database. An anomaly database is also used to store the patterns that are labeled as anomaly either by a user (manually) or the system (automatically) using pre-labeled data. The parameters of the Random Forest are determined and optimized by trying different values on the balanced training set. The study created a balanced data set by replicating the least occurring attack instances, which might be considered an improper approach. There is no validation stage, which is not recommended because the system parameters are determined by the data from the training set, which reduces the model generalization. Nevertheless, the overall approach is sound because it uses the helpful properties of Random Forests successfully.

The study grouped the data set into attacks (DoS and Probe or Scan) and minority attacks (U2R and R2L). The reported performance on the misuse detection was 1.92% error on the original set and 0.05% on the balanced set with 0.01% and 0% FAR, respectively. The study also grouped the DoS and Probe or Scan attacks into four attack levels (1%, 2%, 5%, and 10%), which represent the percentage of attack data in the testing set. The accuracies achieved for these four attack levels are 95%, 93%, 90%, and 87%, respectively, with a constant 1% FAR. The minority attack (U2R and R2L) detection is reported as 65% with a 1% FAR. The implementation is fast enough to be employed as an online solution.

Another study [63] used a Random Forest for misuse detection on the same KDD 1999 data set in a fashion similar to Zhang et al. [62]. The main difference is that a normalization stage takes place in the preprocessing module. This study compares the performance of Random Forests to Gaussian maximum likelihood, Naïve Bayes, and decision tree classifiers. The accuracies reported for Random Forest are 97%, 76%, 5%, and 35% for DoS, Probe or Scan, R2L, and U2R, respectively.

An ensemble of three ANNs (see Subsection A, one SVM (see Subsection L), and one Multivariate Adaptive Regression Spline (MARS) [64] is used by Mukkamala et al. [65] for intrusion detection. The first ANN is a resilient back-propagation NN, the second is a scaled conjugate gradient ANN, and the third is a one-step secant algorithm ANN. The majority voting is used to combine the results of the five classifiers.

The data used are a subset of the DARPA 1998 data set. This subset is composed in 80% of attacks and in 20% of normal data. Performances of 99.71%, 99.85%, 99.97%, 76%, and 100% are reported for the Normal, Probe or Scan, DoS, U2R, and R2L categories, respectively. The ensemble method outperforms the accuracy of individual classifiers that are an input to the ensemble. The accuracy of four classes is in the 99% range; only the accuracy on the U2R class is much lower at 76%.

2) Anomaly Detection and Hybrid Detection: The DISCLOSURE system by Bilge et al. [66] performs Command and Control (C&C) botnet detection using Random Forests. It uses NetFlow data that is easily available but includes only aggregate metadata related to flow duration and number of packets transferred, not full packet payloads. Flow-based features are extracted from the data (e.g., flow size, client-access patterns, temporal features). There are no details in their paper of how many trees were used in the Random Forest classifier or how many attributes the trees used on average. DISCLOSURE was tested over two real-world networks and achieved a True Positive (detection) Rate of 65%, and False Positive Rate of 1%.

The application of Random Forests to anomaly detection is described by Zhang et al. [62], where an anomaly (outlier) detector was employed to feed a second threat classifier. In effect, the method is hybrid, using two Random Forest classifiers—one for anomaly detection and the other for misuse detection. The performance of the outlier detection was 95% accuracy with 1% FAR. The study illustrated how an anomaly detector can be implemented by using a proximity measure (i.e., the distance between two instances calculated from the forest) of the Random Forest. It used as the proximity measure the sum of squares of the proximities between trees and categories.

G. Evolutionary Computation

The term *evolutionary computation* encompasses Genetic Algorithms (GA) [67], Genetic Programming (GP) [68], Evolution Strategies [69], Particle Swarm Optimization [70], Ant Colony Optimization [71], and Artificial Immune Systems [72]. This subsection focuses on the two most widely used evolutionary computation methods—GA and GP. They are both based on the principles of survival of the fittest. They operate on a population of individuals (chromosomes) that are evolved using certain operators. The basic operators are selection, crossover, and mutation. They start usually with a randomly generated population. For each individual from the population, a fitness value is computed that reveals how good a given individual is at solving the problem at hand. The individuals with higher fitness have a higher probability of being chosen into the mating pool and thus being able to reproduce. Two individuals from the mating pool can perform crossover (i.e., exchange genetic material between them) and each can also undergo mutation, which is a random alteration of the genetic material of the individual. The highest fit individuals are copied into the next generation.

The main difference between GA and GP is how the individuals are represented. In GA, they are represented as bit strings and the operations of crossover and mutation are very simple. In GP, the individuals represent programs and therefore represent trees with operators such as *plus*, *minus*, *multiply*, *divide*, *or*, *and*, *not*, or even programming blocks such as *if then*, *loop*, etc. In GP the operators of crossover and mutation are much more complex than those used in GA.

1) Misuse Detection: Li [73] developed a method that uses GA for evolving rules for misuse detection. The DARPA intrusion detection data set was used. The GA chromosome is designed to contain the source IP address, destination IP address, source port number, destination port number, duration of the connection, protocol, number of bytes sent by originator and responder, and state of the connection. In the fitness function, the agreement on certain parts of the chromosome (e.g., destination IP address) is weighted higher than on others (e.g., protocol type). Traditional operators of crossover and mutation are applied to the individuals from the population. Niching techniques are used to find multiple local maxima (because many rules are needed, not just one). The best evolved rules become part of the rule base for intrusion detection. Although Li's paper describes an interesting technique and shows some rules evolved by the system, what is missing, despite the paper's 181 citations, is the accuracy of the rule set on test data.

Abraham et al. [74] use GP to evolve simple programs to classify the attacks. The three GP techniques used in the experiments are Linear Genetic Programming (LGP), Multi-Expression Programming (MEP), and Gene Expression Programming (GEP). The programs involved made use of $+$, $-$, $*$, $/$, sin, cos, sqrt, ln, lg, log2, min, max, and abs

TABLE V
SENSITIVITY AND SPECIFICITY OF GP WITH HOMOLOGOUS CROSSOVER

Type of Attack	Sensitivity	Specificity
Smurf	99.93	99.95
Satan	100.00	99.64
IP Sweep	88.89	100.00
Port Sweep	86.36	100.00
Back	100.00	100.00
Normal	100.00	100.00
Buffer Overflow	100.00	100.00
WarezClient	66.67	99.97
Neptune	100.00	99.56

as the function set. Different subsets of features are used by the different GP techniques. The DARPA 1998 intrusion detection data set was used. The data set contains 24 attack types that can be classified into four main categories: DoS, unauthorized access from a remote machine (R2L), unauthorized access to local super user (U2R), and surveillance and other probing (Probe or Scan). The FAR varies from 0% to 5.7% depending on the method used and the type of attack.

Hansen et al. [75] used GP with homologous crossover for evolving programs performing intrusion detection. Homologous crossover is a special crossover operator designed to reduce the tendency of evolved programs to grow ever larger with increasing number of generations. A subset of the KDD 1999 data was used, with 30,000 instances for training and 10,000 for testing. Training and testing data sets were chosen in such a way that they have the same proportions of attacks of each type as the full data set. Their accuracy ranged from 66.7% to 100% depending on the type of attack (lowest for U2R and highest for DoS). Specificity (Table V) ranged from 99.56% to 100%, meaning that the FAR is very low. GP with homologous crossover results are also compared with the results of the winner of the KDD 1999 Cup, and they are better. However, that comparison might be invalid because the results in this paper are only for a chosen test set of 10,000 signatures and the results for the winner of the KDD Cup could have been for a different test data set.

2) Anomaly Detection and Hybrid Detection: Khan [76] uses GA to evolve rules for detection of intrusions. Two subsets each of 10,000 instances from the KDD 1999 data set were used: one for training and one for testing. For the large number of attributes in that data, eight were chosen using Principal Component Analysis [77]. A population of 10 individuals was used, and it appears only one rule from the final population was used for classification of data into two classes: normal and attack. It is surprising that the results obtained on the test data set are actually better than the ones obtained on the training data set. The accuracies and FARs are 93.45% (10.8% FAR) and 94.19% (2.75% FAR) for the normal and attack classes, respectively.

Lu and Traore [78] presented a rule evolution approach based on GP for detecting known and novel attacks on the network. The initial population of rules was selected based on background knowledge from known attacks, and each rule can be represented as a parse tree. GP evolves these initial rules to generate new rules using four genetic operators: reproduction, crossover, mutation, and dropping condition operator. The dropping condition operator randomly selects one condition in the rule, and then this condition is no longer considered in the rule, thereby preventing the programs from growing increasingly complicated with the number of generations. The fitness function used was based on the support and confidence of a rule (see Subsection 1). The paper uses a subset of the DARPA intrusion detection data. The training data set consists of 1 day's connection records (i.e., 10,000 connection records) with eight attack types. The testing data set consists of another day's connection records with 10 attack types (two attack types are new). In practical evaluation, the rule base instead of single rule is used to test the performance of the IDS. Ten thousand runs of GP are executed and the average results are reported. The average value of FAR is 0.41% and the average value of P_D is 0.5714. The ROC shows P_D close to 100% when the FAR is in the range between 1.4% and 1.8%. However, when the FAR is close to 0%, the P_D is only about 40%. The P_D falls in a broad range from 40% to 100% because the number of rules in the rule base is different for each run. The results are reported on data that contain novel attacks as well as known attacks. Unlike many other papers that report only the best results, Lu and Traore include the average results. Their best results, with P_D close to 100% and FAR is in the range between 1.4% and 1.8% are good.

H. Hidden Markov Models

Markov chains and Hidden Markov Models (HMMs) belong to the category of Markov models. A Markov chain [79] is a set of states interconnected through transition probabilities that determine the topology of the model. An HMM [80] is a statistical model where the system being modeled is assumed to be a Markov process with unknown parameters. The main challenge is to determine the hidden parameters from the observable parameters. The states of an HMM represent unobservable conditions being modeled. By having different output probability distributions in each state and allowing the system to change states over time, the model is capable of representing non-stationary sequences.

An example of an HMM for host intrusion detection is shown in Figure 5 [81]. In this example, each host is modeled by four states: *Good*, *Probed*, *Attacked*, and *Compromised*. The edge from one node to another represents the fact that, when a host is in the state indicated by the source node, it can transition to the state indicated by the destination node. The state transition probability matrix P describes the probabilities of transitions between the states of the model. The observation probability matrix Q describes the probabilities of receiving different observations given that the host is in a certain state. π is the initial state distribution. An HMM is denoted by (P, Q, π).

1) Misuse Detection: In the study by Ariu et al. [82], attacks on web applications (such as XSS and SQL-Injection) are considered, and HMMs are used to extract attack signatures. According to the study, 50% of the discovered vulnerabilities in 2009 affected web applications. The study describes a system named HMMPayl, which examines the HTTP payloads using n-grams and builds multiple HMMs to be used in a classifier fusion scheme. The outlined multi-classifier can also be considered as an ensemble learning approach; however, unlike

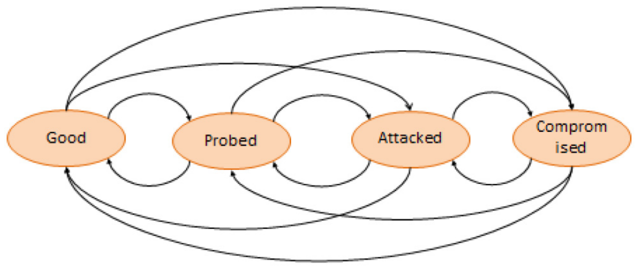


Fig. 5. An Example Hidden Markov Model.

an ensemble method, Ariu et al. [82] set up the classifiers in a competitive rather than complementary manner. In the experiment section, Ariu's study also used the DARPA 1999 data set as well as some other HTTP data sets. In most of the experiments, the mean Area Under the ROC Curve (AUC) of 0.915 to 0.976 is achieved, for an FP rate ranging from 10E-4 to 10E-1. For FP rates higher than 10E-3, HMMPayl attains detection rates higher than 0.85, a noteworthy result. For smaller FP rates, the percentage of detected attacks decreases but still remains higher than 70% for a FP rate of 10E-4.

2) Anomaly Detection and Hybrid Detection: HMMs were used for intrusion detection by Joshi and Phoha [83]. They use an HMM with five states and six observation symbols per state. The states in the model are interconnected in such a way that any state can be reached from any other state. The Baum-Welch method [84] is used to estimate the HMM parameters. The KDD 1999 data set was used, and 5 out of 41 features were chosen for modeling. P_D was 79%; the remaining 21% is accounted for as a FP rate (i.e., classifying anomaly as normal) and an FN rate (i.e., classifying normal as an attack). The authors claim that they could significantly improve the accuracy by using more than five features. Although this is possible, it is far from being proven that by using a larger subset of features (or the full set), the accuracy of an HMM will increase.

I. Inductive Learning

Two basic techniques of inferring information from data are deduction and induction. Deduction infers information that is a logical consequence of the information present in the data and proceeds from the top down. Inductive reasoning moves from the bottom up, that is from specific observations to broader generalizations and theories. In inductive learning, one starts with specific observations and measures, begins to detect patterns and regularities, formulates some tentative hypotheses to be explored, and lastly ends up developing some general conclusions or theories. Several ML algorithms are inductive (e.g., C4.5 for building decision trees), but when researchers refer to inductive learning, they usually mean Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [85] and the algorithm quasi-optimal (AQ) [86].

RIPPER builds rules for two-class problems. It induces rules directly from the training data by using a separate-and-conquer approach. It learns one rule at a time in such a way that the rule covers a maximal set of examples in the current training set. The rule is pruned to maximize a desired performance measure. All of the examples that are correctly labeled by the resulting rule are eliminated from the training set. The process is repeated until the training set becomes empty or a predetermined stopping criterion is reached.

An example RIPPER rule [87] could be:

Guess :- failed-logins $>= 5$

Which means that if number of failed-logins is greater than or equal to 5, then this connection is a "guess" (i.e., a guessing password attack)

1) Misuse Detection: Lee et al. [87] developed a framework in which several ML and DM techniques were used (e.g., Inductive Learning, Association Rules, Sequential Pattern Mining). First, Sequential Pattern Mining (also called Frequent Episodes) was used to construct temporal and statistical features. A lot of experimentation with the time window is needed when doing this. When Sequential Pattern Mining is performed on the intrusion patterns, it extracts features that will be used in misuse detection. In the next step, those features are used by RIPPER to generate rules. However, RIPPER does not use all the features identified in its rules because, as with most classification algorithms, it has a built-in feature selection mechanism.

RIPPER generates rules for classifying the telnet connections from the DARPA 1998 data set. Seven weeks of data were used for training, and two separate weeks of data were used for testing. The test data contain 38 attack types, with 14 types present in test data only. The accuracy achieved for a new type of attack ranged from 5.9 (R2L) to 96.7% (Probe or Scan), while that for an old type of attack ranged from 60 (R2L) to 97% (Probe or Scan). Their method was able to detect with high accuracy new attacks that are Probe or Scan (96.7%) and U2R (81.8%). The accuracy of detecting new types of attacks for DoS and R2L was less than 25%.

2) Anomaly Detection and Hybrid Detection: A true anomaly detection problem, by definition, does not offer the abnormal type of data in advance. Therefore, the major difficulty of the anomaly detection lies in discovering the boundaries between known and unknown categories [88]. In the study, an artificial anomaly generator was designed to improve the performance of the anomaly detector in terms of its generalization ability. The first approach was distribution-based anomaly generation and the second was filtered artificial anomalies. The DARPA 1998 data set was used in the experiment setup. An impressive 94% anomaly detection rate with a 2% FAR was achieved. The FAR is generally desired to be much less than 1%, especially with DARPA 1998 data set types where the rate of packets is at the TCP/IP level (thousands of packets per second). Nevertheless, the study successfully demonstrated how true anomaly detection should be conducted and how the data should be used by not employing a binary classifier for anomaly detection, like many other studies performed.

J. Naïve Bayes

Naïve Bayes classifiers [89] are simple probabilistic classifiers applying the Bayes theorem. The name originates from the fact that the input features are assumed to be independent, whereas in practice this is seldom true. The conditional probabilities, $p(C|f_1, f_2, \ldots, f_m)$, form the classifier model, and the classifier assigns a class label as follows:

$$y(f_1, f_2, \ldots, f_m) = argmax_{k\in\{1,\ldots K\}} p(C_k) \prod_{i=1}^{m} p(f_i|C_k) \tag{3}$$

where m is the number of features, K is the number of classes, f_i is the i-th feature, C_k is the k-th class, $p(C_k)$ is the prior probability of C_k, and $P(f_i|C_k)$ is the conditional probability of feature f_i given class C_k.

Naive Bayes classifiers can handle an arbitrary number of independent features whether continuous or categorical. They reduce a high-dimensional density estimation task to a one-dimensional kernel density estimation, using the assumption that the features are independent.

Although the Naïve Bayes classifier has several limitations, it is an optimal classifier if the features are conditionally independent given the true class. Commonly, it is one of the first classifiers that is compared to the more sophisticated algorithms. In addition, certain types of users expressed that they understand the classification model more intuitively compared to other complex classifiers (e.g., SVM). One of the biggest advantages of the Naïve Bayes classifier is that it is an online algorithm and its training can be completed in linear time.

1) Misuse Detection: Panda and Patra [90] employed the Naïve Bayes classifier from the Weka package [11] and used the KDD 1999 data set for training and testing. The data were grouped into four attack types (Probe or Scan, DoS, U2R, and R2L) and the classifier achieved 96%, 99%, 90%, and 90% testing accuracy, respectively, on these categories. The cumulative FAR was reported as 3%.

The paper compared the results to an NN classifier and stated that the Naïve Bayes classifier has a higher accuracy but a higher FAR, which is undesirable. The results are reported to be better than the highest score achieved in KDD competition.

2) Anomaly Detection and Hybrid Detection: The framework developed by Amor et al. [91] uses a simple form of a Bayesian network that can be considered a Naïve Bayes classifier. Constructing a general Bayesian network is an NP-complete problem, but the simple network of a root node and leafs for attributes form the Naïve Bayes classification structure.

This paper also used the KDD 1999 data set and grouped the categories in three arrangements to reflect different attack scenarios and performance measurements. A single attack and the normal data were contained in the first set. The second set contained all four attack types of the KDD 1999 data set, and the problem to be solved was a multi-class classification for misuse detection. The third set contained the normal data and all four attack types (combined into one category), and the problem to be solved was an anomaly detection problem.

The paper reported the results as 97%, 96%, 9%, 12%, and 88% accuracy achieved for Normal, DoS, R2L, U2R, R2L, and Probe or Scan categories, respectively. No false alarm is reported in the work, but because 97% normal is achieved, the FAR can be assumed to be less than 3%. The anomaly detection experiment is reported as 98% and 89% accuracies for the Normal and Abnormal categories, respectively.

K. Sequential Pattern Mining

Sequential pattern mining has emerged as one of the important DM methods [92] with the advent of transactional databases where each transaction has a temporal ID, a user ID, and an itemset. An itemset is a set of distinct items purchased in a transaction (a strictly binary representation in which an item was or was not purchased). A sequence is an ordered list of itemsets. The length of the sequence is defined as the number of itemsets in the sequence. The order is determined by the time ID. Sequence A (of length n) is contained in a second sequence, B (of length m), when all the itemsets of A, $\{a_i\}$, are subsets of itemsets B, $\{b_j\}$, with a 1-to-1 mapping between indices i to j such that $a_1 \subseteq b_{j1}, a_2 \subseteq b_{j2}, \ldots, a_n \subseteq b_{jk}$ and $j_1 \leq j_2 \leq \cdots \leq j_k$. In other words, each of the itemsets of A is a subset of an itemset in B. If an itemset a_i is a subset of an itemset b_j, then the next itemset a_{i+1} must be a subset of an itemset b_{j+m}, where $m > 0$. In sequence B, itemsets that are not a subset of an itemset in A are allowed (i.e., $n \leq m$).

In a set of sequences, the sequence A is maximal if it is not contained in any other sequence. Consider a database, D, which contains sequences grouped by a certain nominal variable, such as IP address, denoted by p. If sequence A is contained in $D(p)$ [i.e., one of the sequences of $D(p)$ contains A], then A supports $D(p)$. The support is the fraction of sequences in $D(.)$ that A supports. A large sequence is defined as the sequence supporting a minimum threshold, *Th*. The problem of sequence mining is then to find all maximal sequences that are contained in D with a user-given minimum support *Th*. The maximal sequences themselves are generated from the sequences in D by enumerating all possible sequences.

1) Misuse Detection: In a slightly different field, database intrusion detection [93] used sequential pattern mining to detect database intrusions by examining the sequence patterns of database logs. These logs include several fields such as operation type, transaction name, transaction ID, begin time, end time, etc. The general approach presented in this work can be applied to similar intrusion analysis for cyber security such as U2R attacks. The study uses the AprioriAll [92] algorithm and generates the maximal sequences with a user-set support threshold. Generated patterns are the signatures for intrusions. The reported maximum performance was 91% detection rate and the worst FAR was around 30%. Nevertheless, the work successfully applied sequential pattern mining to detect intrusions to databases by examining database logs.

2) Anomaly Detection and Hybrid Detection: Li et al. [94] give a good example of sequential pattern mining and its application for reducing the redundancy of alerts and minimizing FARs. They used the AprioriAll algorithm to discover multi-stage attack patterns. The sequences were the attack patterns either previously discovered or provided by the cyber administrator. A pattern visualization tool was also generated for the cyber user. A support threshold of 40% was used, and the generated maximal sequences were used for alert verification and correlation. The work used the DARPA 1999 and DARPA 2000 data sets and was able to detect 93% of the attacks in 20 seconds. In a second set of experiments, the study reported a simulated real-time scenario where 84% attack detection was achieved. The paper stated that the support threshold was the main parameter to control the detection rate and the FAR. The undetected patterns simply did not exceed the support threshold.

L. Support Vector Machine

The SVM is a classifier based on finding a separating hyperplane in the feature space between two classes in such a way that the distance between the hyperplane and the closest data points of each class is maximized. The approach is based on a minimized classification risk [95] rather than on optimal classification. SVMs are well known for their generalization ability and are particularly useful when the number of features, m, is high and the number of data points, n, is low ($m >> n$).

When the two classes are not separable, slack variables are added and a cost parameter is assigned for the overlapping data points. The maximum margin and the place of the hyperplane is determined by a quadratic optimization with a practical runtime of $O(n^2)$, placing the SVM among fast algorithms even when the number of attributes is high.

Various types of dividing classification surfaces can be realized by applying a kernel, such as linear, polynomial, Gaussian Radial Basis Function (RBF), or hyperbolic tangent. SVMs are binary classifiers and multi-class classification is realized by developing an SVM for each pair of classes.

1) Misuse Detection: In the work by Li et al. [96], an SVM classifier with an RBF kernel was used to classify the KDD 1999 data set into predefined categories (DoS, Probe or Scan, U2R, R2L, and Normal). From the 41 features, a subset of attributes was selected by following a feature removal policy and a sole-feature selection policy. Lastly, the study used 19 features determined by the feature selection method.

In the same work, a subset of the training set was determined by Ant Colony Optimization (see Subsection G). One possible advantage of this approach is to maximize classifier generalization and minimize the bias in the KDD set, as reported in Section II. The study reported its tenfold cross validation performance as overall 98% accuracy with unknown variance. The lowest performance of 53% was for the U2R category. The strategy of using a subset of a training set to overcome the limitations of the KDD data set is definitely worthwhile, as demonstrated in this paper.

Amiri et al. [97] used a least-squared SVM to have a faster system to train on large data sets. To reduce the number of features in the KDD data set from 41 to 19 or less, they used three different feature selection algorithms. The first was based on picking the feature that maximizes the classification performance, the second was based on mutual information, and the third was correlation based. The experimental results showed the mutual information-based approach is more promising (albeit slightly) than the other two.

The data set was sampled randomly to have around 7000 instances (out of a total of 5 million) for each of the five classes (DoS, Probe or Scan, U2R, R2L, and Normal). A bootstrapping technique was used to resample the U2R attacks. To predict the attack type, five classifiers were built for each category. In this manner, a cost is associated with each category and the final classification was determined. The classification performance was reported as 99% on the DoS, Probe or Scan, R2L, and Normal classes and as 93% on the U2R class with 99% confidence interval.

TABLE VI
Comparison of Accuracy of Enhanced SVM With Snort and Bro

Method	Test Set	Accuracy (%)	FP Rate (%)	FN Rate (%)
Enhanced SVM	Normal	94.19	5.81	0.00
	Attack #1	66.60	0.00	33.40
	Attack #2	65.76	0.00	34.24
	Real	99.90	0.09	0.00
Snort	Normal	94.77	5.23	–
	Attack #1	80.00	–	20.00
	Attack #2	88.88	–	11.12
	Real	93.62	6.38	–
Bro	Normal	96.56	3.44	–
	Attack #1	70.00	–	30.00
	Attack #2	77.78	–	22.22
	Real	97.29	2.71	–

2) Anomaly Detection and Hybrid Detection: Hu et al. [98] used the robust support vector machine (RSVM), a variation of the SVM where the discriminating hyperplane is averaged to be smoother and the regularization parameter is automatically determined, as the anomaly classifier in their study. The Basic Security Module [18] portions from the DARPA 1998 data set were used to preprocess training and testing data. The study showed a good classification performance in the presence of noise (such as some mislabeling of the training data set) and reported 75% accuracy with no false alarms and 100% accuracy with a 3% FAR.

Wagner et al. [99] used NetFlow data collected from real-world and simulated attack data using the Flame tool [100] and other Internet Service Provider sources who provided real-world attack data, such as NetBIOS scans, DoS attacks, POP spams, and Secure Shell (SSH) scans. The study employed a one-class SVM classifier, which is considered a natural approach for anomaly detection. A new window kernel was introduced to help find an anomaly based on time position of the NetFlow data. More than one NetFlow record entered into this kernel, suggesting the sequential information was also retained for classification. Moreover, the window was checked for IP address and traffic volume. The window kernel operates as a similarity measure between the similar sequential NetFlow records. The performance was reported as 89% to 94% accurate on different types of attacks with FP rates of 0% to 3%.

The approach described by Shon and Moon [101] is a framework for detection of novel attacks in network traffic. Their approach is a combination of the Self-Organizing Feature Map (SOFM), GA, and SVM. It is described here because the main novelty of the method is the Enhanced SVM.

The four steps in the proposed system are as follows:

- A type of unsupervised ANN performing SOFM clustering [102] is used for packet profiling.
- Packet filtering is performed using Passive TCP/IP Fingerprinting.
- GA selects features.
- Data are classified using the Enhanced SVM, which is derived from a one-class SVM, and the supervised

soft-margin SVM. The first provides the unlabeled classification capability of the one-class SVM, and the second provides the high detection performance of the supervised SVM.

The study used the 1999 DARPA intrusion detection data set. To make the data set more realistic, the subset chosen consisted of 1% to 1.5% attacks and 98.5% to 99% normal traffic. The results from the Enhanced SVMs had 87.74% accuracy, a 10.20% FP rate, and a 27.27% FN rate. Those results were substantially better than those from the one-class SVMs, but not as good as soft-margin SVMs. However, the Enhanced SVM can detect novel attack patterns, whereas a soft-margin SVM cannot.

Another experiment was performed on a data set collected at the institute where one of the authors works. The performance of the Enhanced SVM-based system was compared to the performance of Snort [16] and Bro [103]. Testing was performed on normal data, two sets of known attacks, and so-called real data (i.e., data collected by the author's institute). In case of normal data (see Table VI), Bro performed the best. Snort and the Enhanced SVM performed similarly, with Snort being slightly better. In the case of known attacks, the performance of Snort and Bro was better than that of the Enhanced SVM. In case of real data, the Enhanced SVM outperforms Snort and Bro.

V. Computational Complexity of ML and DM Methods

The literature contains a limited number of performance comparisons for ML and DM algorithms. The concept of calibrating the prediction involves a kind of smoothing on the output of the predictions to fit them more suitably to a distribution. Therefore, an appropriate performance comparison should include calibration and no calibration of the predictions with a suitable approach such as Platt Scaling and Isotonic Regression [104]. According to the empirical comparison in [104], bagged trees, Random Forests, and ANNs give the best results. After calibration, boosted trees and SVMs perform better. The study also reports that generalizations do not hold, there is significant variability across the problems and metrics, and model performances are not always consistent. Although some algorithms are accepted to be better performing than others, the performance of a particular ML algorithm is application and implementation dependent.

Table VII provides the computational complexity (i.e., time complexity) of the different ML and DM algorithms. The elements of Table VII were found through an extensive literature and Internet search. Naturally, some of the time complexities are debatable, and they are based on the experience of the user and the skills of the implementer. Most of these algorithms have well-maintained open-source implementations. The assumptions made in Table VII are that the data consist of n instances, each described by m attributes, and n is much greater than m.

As a rule of thumb, the O(n) and O(n log n) algorithms are considered to be linear time and are usable for online approaches. O(n^2) is considered as acceptable time complexity for most practices. O(n^3) and higher are considered to be much slower algorithms and used for offline approaches.

A higher percentage of the papers covered in this survey paper present their approaches as offline methods. The processed data are ready and input to the system as whole. When the system pipeline is designed to work as an online system, or when the system processes streaming data, several concerns have to be addressed, such as the data input/output streams and buffering, running the methods online, and displaying results with the appropriate timing information. A few studies [35], [42], [52], [62] described their systems as online-running and processing the input data in soft real time. Interestingly, some of these studies even use slower algorithms such as sequence mining [52] to perform intrusion detection. There are also system-level issues to be addressed such as partitioning the input data streams (e.g., MapReduce framework [105]), employing the learning methods, and collecting and aggregating the results in parallel.

In general, when training prediction models, or learning network traffic features, an online suitable method addresses, at a minimum, three factors: time complexity, incremental update capability, and generalization capacity.

- The time complexity of each algorithm is presented in Table VII. A method should be close to roughly O(n log n) to be considered a streaming algorithm. However, slow algorithms such as sequence mining methods or ANNs are also used within streaming systems by keeping the input data windowed and having a small n.
- For the incremental update capability, the clustering algorithms, statistical methods (e.g., HMM, Bayesian networks), and ensemble models can easily be updated incrementally [89], [106]. However, updates to ANNs, SVMs, or evolutionary models may cause complications [89], [106].
- A good generalization capacity is required so that the trained model does not drastically deviate from the starting model when new input data are seen. Most of the state-of-the-art ML and DM methods have very good generalization ability.

The testing phase for methods is generally fast, mostly on the order of linear time with respect to the input data size. Therefore, once trained, most methods can be used online.

VI. Observations and Recommendations

The extent of papers found on ML and DM for cyber intrusion detection shows that these methods are a prevalent and growing research area for cyber security. The question is: Which of these methods are the most effective for cyber applications? Unfortunately, this is not yet established.

A. *Observations Related to the Data Sets*

Table VIII lists the representative ML and DM method papers applied to cyber domain that were reviewed (and described in Section IV), including the number of times they have been cited, the cyber problem they are solving, and the data used. It is interesting that of the 39 papers listed in Table VIII, 28 used the DARPA 1998, DARPA 1999, DARPA 2000, or KDD 1999 data sets. Only two used NetFlow data, two used tcpdump data,

TABLE VII
COMPLEXITY OF ML AND DM ALGORITHMS DURING TRAINING

Algorithm	Typical Time Complexity	Streaming Capable	Comments
ANN	O(emnk)	low	Jain et al. [107] e: number of epochs k: number of neurons
Association Rules	>> O(n^3)	low	Agrawal et al. [108]
Bayesian Network	>> O(mn)	high	Jensen [41]
Clustering, k-means	O(kmni)	high	Jain and Dubes [46] i: number of iterations until threshold is reached k: number of clusters
Clustering, hierarchical	O(n^3)	low	Jain and Dubes [46]
Clustering, DBSCAN	O(n log n)	high	Ester et al. [109]
Decision Trees	O(mn^2)	medium	Quinlan [54]
GA	O(gkmn)	medium	Oliveto et al. [110] g: number of generations k: population size
Naïve Bayes	O(mn)	high	Witten and Frank [89]
Nearest Neighbor k-NN	O(n log k)	high	Witten and Frank [89] k: number of neighbors
HMM	O(nc^2)	medium	Forney [111] c: number of states (categories)
Random Forest	O(Mmn log n)	medium	Witten and Frank [89] M: number of trees
Sequence Mining	>> O(n^3)	low	Agrawal and Srikant [92]
SVMs	O(n^2)	medium	Burges [112]

one used DNS data, one used SSH commands, and four used some other type of data. When identifying the representative papers, this study primarily considered the ML or DM method the authors used and the fact that the papers represented a misuse, anomaly, or hybrid approach. Another important factor was that the papers were highly referenced, which was considered to be some indication of their quality. However, some promising emerging methods were also included even if they have not yet had a chance to be highly referenced. Although the study targeted papers written in 2000 or later, two earlier papers were well written and highly cited and therefore merited inclusion in this survey paper.

The fact that so many papers use the DARPA and KDD data sets is related to how difficult and time consuming it is to obtain a representative data set. Once such a data set is available, researchers tend to reuse it. Additionally, reusing the same data set should allow for easy comparison of the accuracy of the different methods. As discussed earlier, in the case of the DARPA and KDD data sets, this was actually not entirely true because those data sets are so huge that researchers chose to work on different subsets. The two papers that discussed the use of NetFlow also discussed anomaly detection performance. This is obvious because NetFlow does not have such a rich set of features as tcpdump or DARPA or KDD data nor the features necessary for detecting particular signatures in misuse detection. (Its features are limited to flow information generated by higher-end routers.)

B. Factors Related to IDS Performance

One of the most important factors related to the performance of IDSs is the type and level of the input data. As previously discussed, several studies used DARPA or KDD data sets because they are easy to obtain and contain network-level data (either tcpdump or NetFlow) as well as OS-level data (e.g., network logs, security logs, kernel system calls). Primarily, the attack data coming to the network stack and the effect of these packets on the OS level carried important information. As a result, it is advantageous that an IDS be able to reach network- and kernel-level data. If only NetFlow (much easier to obtain and process) data are available for the IDS, these data must be augmented by network-level data such as network sensors that generate additional features of packets or streams. If possible, the network data should be augmented by OS kernel-level data. As discovered, several studies approached the intrusion detection problem by examining OS-level commands (i.e., host-based IDS), not network packets.

The second factor related to the performance of the IDSs is the type of ML and DM algorithms employed and the overall system design. The literature survey revealed that many studies used DARPA and KDD data sets and applied different types of ML methods. These studies did not actually build an IDS, but examined the performances of ML and DM methods on some cyber security data. However, categorizing the studies with respect to the authors' affiliations reveals studies that built actual IDSs and employed real-world data captured from campus networks or Internet backbones. All of these studies appear to have used systems integrated with more than one ML method and several modules related to attack signature capture, signature database, etc.

C. Comparison Criteria

There are several criteria by which the ML/DM methods for cyber could be compared:

- Accuracy
- Time for training a model
- Time for classifying an unknown instance with a trained model
- Understandability of the final solution (classification)

TABLE VIII
ML AND DM METHODS AND DATA THEY USE

ML/DM Method	Paper	No. of Times Cited (as of 7/28/2015)	Cyber Approach	Data Used
ANN	Cannady [24]	463	misuse	Network packet-level data
ANN	Lippmann & Cunningham [27]	235	anomaly	DARPA 1998
ANN	Bivens et al. [28]	135	anomaly	DARPA 1999
Association rules	Brahmi et al. [32]	3	misuse	DARPA 1998
Association rules	Zhengbing et al. [33]	33	misuse	Attack signatures (10 to 0 MB)
Association rules	Apiletti et al. [35]	14	anomaly	NetFlow
Association rules – Fuzzy	Luo and Bridges [39]	192	anomaly	tcpdump
Association rules – Fuzzy	Tajbakhsh et al. [38]	124	hybrid	KDD 1999 (corrected)
Bayesian network	Livadas et al. [42]	208	misuse	tcpdump – botnet traffic
Bayesian network	Jemili et al. [43]	31	misuse	KDD 1999
Bayesian network	Kruegel et al. [44]	260	anomaly	DARPA 1999
Clustering – density based	Hendry and Yang [50]	6	misuse	KDD 1999
Clustering – density based	Blowers and Williams [51]	2	anomaly	KDD 1999
Clustering – sequence	Sequeira and Zaki [52]	214	anomaly	shell commands
Decision tree	Kruegel and Toth [55]	155	misuse	DARPA 1999
Decision tree	Bilge et al. [56]	187	anomaly	DNS data
Ensemble learning	Mukkamala et al. [65]	255	misuse	DARPA 1998
Ensemble – Random Forest	Gharibian and Ghorbani [63]	19	misuse	KDD 1999
Ensemble – Random Forest	Bilge et al.[66]	49	anomaly	NetFlow
Ensemble – Random Forest	Zhang et al. [62]	92	hybrid	KDD 1999
Evolutionary Comp - GA	Li [73]	235	misuse	DARPA 2000
Evolutionary Comp - GP	Abraham et al. [74]	83	misuse	DARPA 1998
Evolutionary Comp - GP	Hansen et al. [75]	52	misuse	KDD 1999 – subset
Evolutionary Comp - GA	Khan [76]	15	anomaly	KDD 1999
Evolutionary Comp - GP	Lu and Traore [78]	124	anomaly	DARPA 1999
HMM	Ariu et al. [82]	25	misuse	HTTP payload
HMM	Joshi and Phoha [83]	61	anomaly	KDD 1999
Inductive learning	Lee et al. [87]	1358	misuse	DARPA 1998
Inductive learning	Fan et al. [88]	195	anomaly	DARPA 1998
Naïve Bayes	Benferhat et al. [45]	17	anomaly	DARPA 2000
Naïve Bayes	Panda and Patra [90]	90	misuse	KDD 1999
Naïve Bayes	Amor et al. [91]	277	anomaly	KDD 1999
Sequence mining	Hu and Panda [93]	151	misuse	Database logs
Sequence mining	Li et al. [94]	18	anomaly	DARPA 2000
SVM	Li et al. [96]	56	misuse	KDD 1999
SVM	Amiri et al. [97]	84	misuse	KDD 1999
SVM – Robust	Hu et al. [98]	114	anomaly	DARPA 1998
SVM	Wagner et al. [99]	1	anomaly	NetFlow
One-class SVM and GA	Shon and Moon [101]	166	hybrid	DARPA 1999

If one were to compare the accuracy of several ML/DM methods, those methods should be trained on exactly the same training data and tested on exactly the same testing data. Unfortunately, even in the studies that used the same data set (e.g., KDD 1999), when they compared their results with the best methods from the KDD Cup (and usually claimed their results were better), they did so in an imperfect fashion—they used a subset of the KDD data set, but not necessarily the same subset that the other method used. Therefore, the accuracy of these results is not comparable.

The time for training a model is an important factor due to ever changing cyber-attack types and features. Even anomaly detectors need to be trained frequently, perhaps incrementally, with fresh malware signature updates.

Time for classifying a new instance is an important factor that reflects the reaction time and the packet processing power of the intrusion detection system.

Understandability or readability of the classification model is a means to help the administrators examine the model features easily in order to patch their systems more quickly. This information (such as packet type, port number, or some other high level network packet feature that reflects the cyber-attack footpath) will be available through the feature vectors that are tagged by the classifier as an intrusion category.

D. Peculiarities of ML and DM for Cyber

ML and DM have been extremely useful in many applications. The cyber domain has some peculiarities that make those methods harder to use. Those peculiarities are especially related to how often the model needs to be retrained and the availability of labeled data.

In most ML and DM applications, a model (e.g., classifier) is trained and then used for a long time, without any changes. In those applications, the processes are assumed to be quasi-stationary and therefore the retraining of the model does not happen often. The situation in cyber intrusion detection is different. Models are trained daily [56], whenever the analyst requires [43], or each time a new intrusion is identified and its pattern becomes known [75]. Especially when the models are supposed to be trained daily, their training time becomes important. (It must definitely take less than 1 full day to retrain the model.) Traditionally, ML and DM methods start the training from scratch. However, if a model needs to be retrained often (e.g., daily) because of just a few changes in the data, it makes more sense to start from the trained model and continue training it or else use self-adaptive models. A fertile area of research would be to investigate the methods of fast incremental learning that could be used for daily updates of models for misuse and anomaly detection.

There are many domains in which it is easy to obtain training data, and in those domains ML and DM methods usually thrive (e.g., recommendations that Amazon makes for its clients). In other areas where data are difficult to obtain (e.g., health monitoring data for machinery or aircraft), applications of ML and DM can be abundant. In the cyber domain, much data could be harvested by putting sensors on networks (e.g., to get NetFlow or TCP), which although not an easy task is definitely worthwhile.

However there is a problem with the sheer volume of that data—there is too much data to store (terabytes per day). Another problem is that some of the data need to be labeled to be useful, which might be a laborious task. Data for training definitely need to be labeled, and this is even true for pure anomaly detection methods. These methods have to use data that are normal; they cannot develop a model with the attack data intermingled. Additionally, because they have to be tested with novel attacks, some novel attack data are required as well. The biggest gap seen is the availability of the labeled data, and definitely a worthwhile investment would be to collect data and label some of it. Using this new data set, significant advances could be made to ML and DM methods in cyber security and breakthroughs could be possible. Otherwise, the best possible available data set right now is the KDD 1999 corrected data set. (However, being 15 years old, this data set does not have examples of all the new attacks that have occurred in the last 15 years.)

E. ML and DM Recommendations for Misuse and Anomaly Detection

IDSs are usually hybrid and have anomaly detection and misuse detection modules. The anomaly detection module classifies the abnormal network traffic. The misuse detection module classifies attack patterns with known signatures or extracts new signatures from the attack-labeled data coming from the anomaly module.

Often, an anomaly detector is based on a clustering method. Among clustering algorithms, density-based methods (e.g., DBSCAN) are the most versatile, easy to implement, less parameter or distribution dependent, and have high processing speeds. In anomaly detectors, one-class SVMs also perform well and much can be learned by extracting association rules or sequential patterns from available normal traffic data.

Among misuse detectors, because the signatures need to be captured, it is important that the classifier be able to generate readable signatures, such as branch features in a decision tree, genes in a genetic algorithm, rules in Association Rule Mining, or sequences in Sequence Mining. Therefore, black-box classifiers like ANNs and SVMs are not well suited for misuse detection

Several state-of-the-art ML and DM algorithms are suitable for misuse detection. Some of these methods are statistical such as Bayesian networks and HMMs; some are entropy-based such as decision trees; some are evolutionary such as genetic algorithms; some are ensemble methods like Random Forests; and some are based on association rules. The designers of the system should investigate whether the training data are of good enough quality and have statistical properties that can be exploited (e.g., Gaussian distribution). It is also important to know whether the required system will work online or offline. Answers to such questions will determine the most suitable ML approach. In the opinion of the authors of this paper, the network data cannot be properly modeled using a simple distribution (e.g., Gaussian) due to the fact that, in practice, a single network packet might contain a payload that can be associated to dozens of network protocols and user behaviors [113]. The variability in the payload is characterized by sums of multiple probability distributions or joint probability distributions, which are not directly separable. Therefore, methods like Bayesian networks or HMMs may not be the strongest approach because the data do not have the properties that are the most appropriate for them. Evolutionary computation methods may take a long time to run and therefore may not be suitable for systems that train online. If the training data are scarce, Random Forests might have an advantage. If the attack signature capture is important, decision trees, evolutionary computation, and association rules can be useful.

VII. Conclusions

The paper describes the literature review of ML and DM methods used for cyber. Special emphasis was placed on finding example papers that describe the use of different ML and DM techniques in the cyber domain, both for misuse and anomaly detection. Unfortunately, the methods that are the most effective for cyber applications have not been established; and given the richness and complexity of the methods, it is impossible to make one recommendation for each method, based on the type of attack the system is supposed to detect. When determining the effectiveness of the methods, there is not one criterion but several criteria that need to be taken into account. They include (as described in Section VI, Subsection C) accuracy, complexity, time for classifying an unknown instance with a trained model, and understandability of the final solution (classification) of each ML or DM method. Depending on the particular IDS, some might be more important than others.

Another crucial aspect of ML and DM for cyber intrusion detection is the importance of the data sets for training and testing the systems. ML and DM methods cannot work without representative data, and it is difficult and time consuming to obtain such data sets. To be able to perform anomaly detection and misuse detection, it is advantageous for an IDS to be able to reach network- and kernel-level data. If only NetFlow data are available, these data must be augmented by network-level data such as network sensors that generate additional features of packets or streams. If possible, the network data should be augmented by OS kernel-level data.

The biggest gap observed is the availability of labeled data, and definitely a worthwhile investment would be to collect data and label some of it. Using this new data set, several promising ML and DM methods could be used to develop models and compared, narrowing the list of ML and DM effective for cyber applications. Significant advances could be made to ML and DM methods in cyber security using this data set and breakthroughs could be possible.

There are some peculiarities of the cyber problem that make ML and DM methods more difficult to use (as described in Section VI, Subsection D). They are especially related to how often the model needs to be retrained. A fertile area of research would be to investigate the methods of fast incremental learning that could be used for daily updates of models for misuse and anomaly detection.

REFERENCES

[1] A. Mukkamala, A. Sung, and A. Abraham, "Cyber security challenges: Designing efficient intrusion detection systems and antivirus tools," in *Enhancing Computer Security with Smart Technology*, V. R. Vemuri, Ed. New York, NY, USA: Auerbach, 2005, pp. 125–163.

[2] M. Bhuyan, D. Bhattacharyya, and J. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surv. Tuts.*, vol. 16, no. 1, pp. 303–336, First Quart. 2014.

[3] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Commun. Surv. Tuts.*, vol. 10, no. 4, pp. 56–76, Fourth Quart. 2008.

[4] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Comput. Secur.*, vol. 28, no. 1, pp. 18–28, 2009.

[5] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of IP flow-based intrusion detection," *IEEE Commun. Surv. Tuts.*, vol. 12, no. 3, pp. 343–356, Third Quart. 2010.

[6] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Appl. Soft Comput.*, vol. 10, no. 1, pp. 1–35, 2010.

[7] Y. Zhang, L. Wenke, and Y.-A. Huang, "Intrusion detection techniques for mobile wireless networks," *Wireless Netw.*, vol. 9, no. 5, pp. 545–556, 2003.

[8] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The KDD process for extracting useful knowledge from volumes of data," *Commun. ACM*, vol. 39, no. 11, pp. 27–34, 1996.

[9] C. Shearer, "The CRISP-DM model: The new blueprint for data mining," *J. Data Warehouse.*, vol. 5, pp. 13–22, 2000.

[10] A. Guazzelli, M. Zeller, W. Chen, and G. Williams, "PMML an open standard for sharing models," *R J.*, vol. 1, no. 1, pp. 60–65, May 2009.

[11] M. Hall, E. Frank, J. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explor. Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.

[12] R Language Definition. (2000). *R Core Team* [Online]. Available: ftp://155.232.191.133/cran/doc/manuals/r-devel/R-lang.pdf, accessed on Nov. 2015.

[13] M. Graczyk, T. Lasota, and B. Trawinski, "Comparative analysis of premises valuation models using KEEL, RapidMiner, and WEKA," *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*. New York, NY, USA: Springer, 2009, pp. 800–812.

[14] V. Jacobson, C. Leres, and S. McCanne, *The Tcpdump Manual Page*. Berkeley, CA, USA: Lawrence Berkeley Laboratory, 1989.

[15] G. Combs. *Wireshark* [Online]. Available: http://www.wireshark.org, accessed on Jun. 2014.

[16] Snort 2.0. *Sourcefire* [Online]. Available: http://www.sourcefire.com/technology/whitepapers.htm, accessed on Jun. 2014.

[17] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. USA: Insecure, 2009.

[18] R. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das, "The 1999 DARPA offline intrusion detection evaluation," *Comput. Netw.*, vol. 34, pp. 579–595, 2000.

[19] R. Lippmann *et al.*, "Evaluating intrusion detection systems: The 1998 DARPA offline intrusion detection evaluation," in *Proc. IEEE DARPA Inf. Surviv. Conf. Expo.*, 2000, pp. 12–26.

[20] S. J. Stolfo, *KDD Cup 1999 Data Set*, University of California Irvine, KDD repository [Online]. Available: http://kdd.ics.uci.edu, accessed on Jun. 2014.

[21] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A detailed analysis of the KDD Cup 1999 data set," in *Proc. 2nd IEEE Symp. Comput. Intell. Secur. Defense Appl.*, 2009, pp. 1–6.

[22] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, pp. 359–366, 1989.

[23] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.

[24] J. Cannady, "Artificial neural networks for misuse detection," in *Proc. 1998 Nat. Inf. Syst. Secur. Conf.*, Arlington, VA, USA, 1998, pp. 443–456.

[25] Internet Security Scanner (ISS). *IBM* [Online]. Available: http://www.iss.net, accessed on Feb. 2015.

[26] B. Morel, "Artificial intelligence and the future of cybersecurity," in *Proc. 4th ACM Workshop Secur. Artif. Intell.*, 2011. pp. 93–98.

[27] R. P. Lippmann and R. K. Cunningham, "Improving intrusion detection performance using keyword selection and neural networks," *Comput. Netw.*, vol. 34, pp. 597–603, 2000.

[28] A. Bivens, C. Palagiri, R. Smith, B. Szymanski, and M. Embrechts, "Network-based intrusion detection using neural networks," *Intell. Eng. Syst. Artif. Neural Netw.*, vol. 12, no. 1, pp. 579–584, 2002.

[29] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in *Proc. Int. Conf. Manage. Data Assoc. Comput. Mach. (ACM)*, 1993, pp. 207–216.

[30] C. M. Kuok, A. Fu, and M. H. Wong, "Mining fuzzy association rules in databases," *ACM SIGMOD Rec.*, vol. 27, no. 1, pp. 41–46, 1998.

[31] L. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338—35, 1965.

[32] H. Brahmi, B. Imen, and B. Sadok, "OMC-IDS: At the cross-roads of OLAP mining and intrusion detection," in *Advances in Knowledge Discovery and Data Mining*. New York, NY, USA: Springer, 2012, pp. 13–24.

[33] H. Zhengbing, L. Zhitang, and W. Junqi, "A novel network intrusion detection system (NIDS) based on signatures search of data mining," in *Proc. 1st Int. Conf. Forensic Appl. Techn. Telecommun. Inf. Multimedia Workshop (e-Forensics '08)*, 2008, pp. 10–16.

[34] H. Han, X. Lu, and L. Ren, "Using data mining to discover signatures in network-based intrusion detection," in *Proc. IEEE Comput. Graph. Appl.*, 2002, pp. 212–217.

[35] D. Apiletti, E. Baralis, T. Cerquitelli, and V. D'Elia, "Characterizing network traffic by means of the NetMine framework," *Comput. Netw.*, vol. 53, no. 6, pp. 774–789, Apr. 2009.

[36] NetGroup, *Politecnico di Torino, Analyzer 3.0* [Online]. Available: http://analyzer.polito.it, accessed on Jun. 2014.

[37] E. Baralis, T. Cerquitelli, and V. D'Elia. (2008). *Generalized Itemset Discovery by Means of Opportunistic Aggregation*. Tech. Rep., Politecnico di Torino [Online] https://dbdmg.polito.it/twiki/bin/view/Public/NetworkTrafficAnalysis, accessed on Jun. 2014.

[38] A. Tajbakhsh, M. Rahmati, and A. Mirzaei, "Intrusion detection using fuzzy association rules," *Appl. Soft Comput.*, vol. 9, pp. 462–469, 2009.

[39] J. Luo and S. Bridges, "Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection," *Int. J. Intell. Syst.*, vol. 15, no. 8, pp. 687–703, 2000.

[40] D. Heckerman, *A Tutorial on Learning with Bayesian Networks*. New York, NY, USA: Springer, 1998.

[41] F. V. Jensen, *Bayesian Networks and Decision Graphs*. New York, NY, USA: Springer, 2001.

[42] C. Livadas, R. Walsh, D. Lapsley, and W. Strayer, "Using machine learning techniques to identify botnet traffic," in *Proc 31st IEEE Conf. Local Comput. Netw.*, 2006, pp. 967–974.

[43] F. Jemili, M. Zaghdoud, and A. Ben, "A framework for an adaptive intrusion detection system using Bayesian network," in *Proc. IEEE Intell. Secur. Informat.*, 2007, pp. 66–70.

[44] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, "Bayesian event classification for intrusion detection," in *Proc. IEEE 19th Annu. Comput. Secur. Appl. Conf.*, 2003, pp. 14–23.

[45] S. Benferhat, T. Kenaza, and A. Mokhtari, "A Naïve Bayes approach for detecting coordinated attacks," in *Proc. 32nd Annu. IEEE Int. Comput. Software Appl. Conf.*, 2008, pp. 704–709.
[46] K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1988.
[47] K. Leung and C. Leckie, "Unsupervised anomaly detection in network intrusion detection using clusters," in *Proc. 28th Australas. Conf. Comput. Sci.*, vol. 38, 2005, pp. 333–342.
[48] P. W. Holland and S. Leinhardt, "Transitivity in structural models of small groups," *Comp. Group Stud.*, vol. 2, pp. 107–124, 1971.
[49] J. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, Jun. 1998.
[50] R. Hendry and S. J. Yang, "Intrusion signature creation via clustering anomalies," in *Proc. SPIE Defense Secur. Symp. Int. Soc. Opt. Photonics*, 2008, pp. 69730C–69730C.
[51] M. Blowers and J. Williams, "Machine learning applied to cyber operations," in *Network Science and Cybersecurity*. New York, NY, USA: Springer, 2014, pp. 55–175.
[52] K. Sequeira and M. Zaki, "ADMIT: Anomaly-based data mining for intrusions," in *Proc 8th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2002, pp. 386–395.
[53] R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.
[54] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann, 1993.
[55] C. Kruegel and T. Toth, "Using decision trees to improve signature-based intrusion detection," in *Proc. 6th Int. Workshop Recent Adv. Intrusion Detect.*, West Lafayette, IN, USA, 2003, pp. 173–191.
[56] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "EXPOSURE: Finding malicious domains using passive DNS analysis," presented at the 18th Annu. Netw. Distrib. Syst. Secur. Conf., 2011.
[57] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "2014 Exposure: A passive DNS analysis service to detect and report malicious domains," *ACM Trans. Inf. Syst. Secur.*, vol. 16, no. 4, Apr. 2014.
[58] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits Syst. Mag.*, vol. 6, no. 3, pp. 21–45, Third Quart. 2006.
[59] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, vol. 96, pp. 148–156.
[60] P. Long and R. Servedio, "Boosting the area under the ROC curve," *Adv. Neural Inf. Process. Syst.*, pp. 945–952, 2007.
[61] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[62] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Trans. Syst. Man Cybern. C: Appl. Rev.*, vol. 38, no. 5, pp. 649–659, Sep. 2008.
[63] F. Gharibian and A. Ghorbani, "Comparative study of supervised machine learning techniques for intrusion detection," in *Proc. 5th Annu. Conf. Commun. Netw. Serv. Res.*, 2007, pp. 350–358.
[64] J. H. Friedman, "Multivariate adaptive regression splines," *Anal. Statist.*, vol. 19, pp. 1–141, 1991.
[65] S. Mukkamala, A. Sunga, and A. Abraham, "Intrusion detection using an ensemble of intelligent paradigms," *J. Netw. Comput. Appl.*, vol. 28, no. 2, pp. 167–182, 2004.
[66] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: Detecting botnet command and control servers through large-scale netflow analysis," in *Proc. 28th Annu. Comput. Secur. Appl. Conf. (ACSAC'12)*, Orlando, FL, USA, Dec. 3–7, 2012, pp. 129–138.
[67] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Mach. Learn.*, vol. 3, no. 2, pp. 95–99, 1988.
[68] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.
[69] H. G. Beyer and H. P. Schwefel, "Evolution strategies: A comprehensive introduction," *J. Nat. Comput.*, vol. 1, no. 1, pp. 3–52, 2002.
[70] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, vol. IV, pp. 1942–1948.
[71] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.
[72] J. Farmer, N. Packard, and A. Perelson, "The immune system, adaptation and machine learning," *Phys. D: Nonlinear Phenom.*, vol. 2, pp. 187–204, 1986.
[73] W. Li, "Using genetic algorithms for network intrusion detection," in *Proc. U.S. Dept. Energy Cyber Secur. Group 2004 Train. Conf.*, 2004, pp. 1–8.
[74] A. Abraham, C. Grosan, and C. Martin-Vide, "Evolutionary design of intrusion detection programs," *Int. J. Netw. Secur.*, vol. 4, no. 3, pp. 328–339, 2007.
[75] J. Hansen, P. Lowry, D. Meservy, and D. McDonald, "Genetic programming for prevention of cyberterrorism through dynamic and evolving intrusion detection," *Decis. Support Syst.*, vol. 43, no. 4, pp. 1362–1374, Aug. 2007.
[76] S. Khan, "Rule-based network intrusion detection using genetic algorithms," *Int. J. Comput. Appl.*, vol. 18, no. 8, pp. 26–29, Mar. 2011.
[77] T. Jolliffe, *Principal Component Analysis*, 2nd ed. New York, NY, USA: Springer, 2002.
[78] W. Lu and I. Traore, "Detecting new forms of network intrusion using genetic programming," *Comput. Intell.*, vol. 20, pp. 470–489, 2004.
[79] A. Markov, "Extension of the limit theorems of probability theory to a sum of variables connected in a chain," *Dynamic Probabilistic Systems*, vol. 1, R. Howard. Hoboken, NJ, USA: Wiley, 1971 (Reprinted in Appendix B).
[80] L. E. Baum and J. A. Eagon, "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology," *Bull. Amer. Math. Soc.*, vol. 73, no. 3, p. 360, 1967.
[81] A. Arnes, F. Valeur, G. Vigna, and R. A. Kemmerer, "Using Hidden markov models to evaluate the risks of intrusions: System architecture and model validation," *Lect. Notes Comput. Sci.*, pp. 145–164, 2006.
[82] D. Ariu, R. Tronci, and G. Giacinto, "HMMPayl: An intrusion detection system based on hidden Markov models," *Comput. Secur.*, vol. 30, no. 4, pp. 221–241, 2011.
[83] S. S. Joshi and V. V. Phoha, "Investigating hidden Markov models capabilities in anomaly detection," in *Proc. ACM 43rd Annu. Southeast Reg. Conf.*, 2005, vol. 1, pp. 98–103.
[84] P. Dempster, N. M. Laird, and D. B. Robin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc.*, Series B (methodological), pp. 1–38, 1977.
[85] W. W. Cohen, "Fast effective rule induction," in *Proc. 12th Int. Conf. Mach. Learn.*, Lake Tahoe, CA, USA, 1995, pp. 115–123.
[86] R. Michalski, "A theory and methodology of inductive learning," *Mach. Learn.*, vol. 1, pp. 83–134, 1983.
[87] W. Lee, S. Stolfo, and K. Mok, "A data mining framework for building intrusion detection models," in *Proc. IEEE Symp. Secur. Privacy*, 1999, pp. 120–132.
[88] W. Fan, M. Miller, S. Stolfo, W. Lee, and P. Chan, "Using artificial anomalies to detect unknown and known network intrusions," *Knowl. Inf. Syst.*, vol. 6, no. 5, pp. 507–527, 2004.
[89] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. San Mateo, CA, USA: Morgan Kaufmann, 2011.
[90] M. Panda and M. R. Patra, "Network intrusion detection using Naive Bayes," *Int. J. Comput. Sci. Netw. Secur.*, vol. 7, no. 12, pp. 258–263, 2007.
[91] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naïve Bayes vs. decision trees in intrusion detection systems," in *Proc ACM Symp. Appl. Comput.*, 2004, pp. 420–424.
[92] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proc. IEEE 11th Int. Conf. Data Eng.*, 1995, pp. 3–14.
[93] Y. Hu and B. Panda, "A data mining approach for database intrusion detection," in *Proc. ACM Symp. Appl. Comput.*, 2004, pp. 711–716.
[94] Z. Li, A. Zhang, J. Lei, and L. Wang, "Real-time correlation of network security alerts," in *Proc. IEEE Int. Conf. e-Business Eng.*, 2007, pp. 73–80.
[95] V. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer, 2010.
[96] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai, and K. Dai, "An efficient intrusion detection system based on support vector machines and gradually feature removal method," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 424–430, 2012.
[97] F. Amiri, M. Mahdi, R. Yousefi, C. Lucas, A. Shakery, and N. Yazdani, "Mutual information-based feature selection for IDSs," *J. Netw. Comput. Appl.*, vol. 34, no. 4, pp. 1184–1199, 2011.
[98] W. J. Hu, Y. H. Liao, and V. R. Vemuri, "Robust support vector machines for anomaly detection in computer security," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 282–289.
[99] C. Wagner, F. Jérôme, and E. Thomas, "Machine learning approach for IP-flow record anomaly detection," in *Networking 2011*. New York, NY, USA: Springer, 2011, pp. 28–39.
[100] D. Brauckhoff, A. Wagner, and M. May, "Flame: A low-level anomaly modeling engine," in *Proc. Conf. Cyber Secur. Exp. Test*, 2008.

[101] T. Shon and J. Moon, "A hybrid machine learning approach to network anomaly detection," *Inf. Sci.*, vol. 177, no. 18, pp. 3799–3821, Sep. 2007.

[102] T. Kohonen, *Self-Organizing Map*. New York, NY, USA: Springer, 1995.

[103] V. Paxson. (2004). *Bro 0.9* [Online]. Available: http://bro-ids.org, accessed on Jun. 2014.

[104] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proc. ACM 23rd Int. Conf. Mach. Learn.*, 2006, pp. 161–168.

[105] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[106] H. Guang-Bin, D. H. Wang, and Y. Lan, "Extreme learning machines: A survey," *Int. J. Mach. Learn. Cybern.*, vol. 2, no. 2, pp. 107–122, 2011.

[107] K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[108] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, "Fast discovery of association rules," *Adv. Knowl. Discov. Data Min.*, vol. 12, no. 1, pp. 307–328, 1996.

[109] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Knowl. Discov. Data Min.*, vol. 96, pp. 226–231, 1996.

[110] P. S. Oliveto, J. He, and X. Yao, "Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results," *Int. J. Autom. Comput.*, vol. 4, no. 3, pp. 281–293, 2007.

[111] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973.

[112] J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, 1998.

[113] K. Ahsan and D. Kundur, "Practical data hiding in TCP/IP," in *Proc. ACM Multimedia Secur. Workshop*, 2002, vol. 2, no. 7.

Anna L. Buczak photograph and biography not available at the time of Publication.

Erhan Guven photograph and biography not available at the time of Publication.