A Survey of Intrusion Detection Systems Leveraging Host Data

ROBERT A. BRIDGES, TARRAH R. GLASS-VANDERLAN, MICHAEL D. IANNACONE, and MARIA S. VINCENT, Cyber & Applied Data Analytics Division, Oak Ridge National Laboratory
QIAN (GUENEVERE) CHEN, Electrical & Computer Engineering, University of Texas, San Antonio

This survey focuses on intrusion detection systems (IDS) that leverage host-based data sources for detecting attacks on enterprise network. The host-based IDS (HIDS) literature is organized by the input data source, presenting targeted sub-surveys of HIDS research leveraging system logs, audit data, Windows Registry, file systems, and program analysis. While system calls are generally included in audit data, several publicly available system call datasets have spawned a flurry of IDS research on this topic, which merits a separate section. To accommodate current researchers, a section giving descriptions of publicly available datasets is included, outlining their characteristics and shortcomings when used for IDS evaluation. Related surveys are organized and described. All sections are accompanied by tables concisely organizing the literature and datasets discussed. Finally, challenges, trends, and broader observations are throughout the survey and in the conclusion along with future directions of IDS research. Overall, this survey was designed to allow easy access to the diverse types of data available on a host for sensing intrusion, the progressions of research using each, and the accessible datasets for prototyping in the area.

CCS Concepts: • **General and reference → Surveys and overviews**; • **Security and privacy → Intrusion detection systems**;

Additional Key Words and Phrases: Intrusion detection, host, anomaly detection

ACM Reference format:
Robert A. Bridges, Tarrah R. Glass-Vanderlan, Michael D. Iannacone, Maria S. Vincent, and Qian (Guenevere) Chen. 2019. A Survey of Intrusion Detection Systems Leveraging Host Data. *ACM Comput. Surv.* 52, 6, Article 128 (November 2019), 35 pages.
https://doi.org/10.1145/3344382

The research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the Department of Energy (DOE) under Contract No. D2017-170222007. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

Authors' addresses: R. A. Bridges, 1 Bethel Valley Road, PO Box 2008, MS 6418, Oak Ridge, TN 37831; email: bridgesra@ornl.gov; T. Glass-Vanderlan; email: unbreakinglass@gmail.com; M. Iannacone; email: iannaconemd@ornl.gov; M. Vincent; email: vincentms@ornl.gov; Q. Chen; email: geunevereqian@utsa.edu.

0360-0300/2019/11-ART128 $15.00
https://doi.org/10.1145/3344382

1 INTRODUCTION

Intrusion detection research began in 1972, when James Anderson published a United States Air Force report discussing the need to detect security breaches of computing systems [5]. Manual investigations of logs and audit data were widely adopted by computer security operators (or system administrators) in the early age of IT technology, yet IDSs that fully depended on experienced security experts could not meet the new requirements of the developing computing technology. In response, automated IDS research emerged—Anderson's 1980 work [6] focused on automating IDS by isolating abnormal behavior in system's audit data, and work of Denning and Neumann [27] developed the first real-time detection system based on expert-written rules in 1985. This early research laid the groundwork for modern *intrusion detection*, comprised of manual techniques, algorithms, and commercial products all geared towards one thing, continual monitoring of computing assets for signs of compromise [15, 75].

Increasingly over the past 30 years, networked computing systems have emerged as ubiquitous assets on which state, personal, and industrial infrastructure critically depend. Moreover, the threat of cyber security breaches has risen, with adversaries now fueled by a profitable underground cyber-crime economy and nation-state ambitions [3, 136]. Consequently, breaches ranging from personal computers to large enterprises and governmental networks are now commonly reported, and governmental assistance, in terms of IDS tutorials, guidelines, and case studies have resulted [130]. Through the authors' ongoing collaborations with multiple security operations, we note that many operations now have widespread collection and query capabilities for logs and alerts. Yet, detection in practice has focused on signature and rule-based detection, often at the network or file levels, complemented by manual analysis of logs [12, 13, 18]. These rule-based IDSs are accurate for detecting known system cyber attacks but cannot identify unknown, novel, or polymorphic cyber threats. In addition, their computational overheads (i.e., time, CPU, and memory costs) are usually high. This has motivated parallel developments in the research literature for a wide variety of automated, fast, and efficient IDSs. From expert-crafted rules to sophisticated statistical learning algorithms, publications explore and push detection accuracy metrics and performance on a variety of data sources and locations within the network (see Sections 3–7 below).

1.1 IDS Components, Types, and Challenges

In general, all intrusion detection systems (IDSs) have three main components.

- *Data collection:* They ingest one or many data types, e.g., system calls or network flows.
- *Conversion to select features:* Some predefined unit of data, e.g., system calls in a process or flows in a time window, is represented as a list of attributes, called a feature vector.
- *Decision engine:* An algorithm or heuristic to decide if the given data, as represented as a feature vector, is believed to be an attack or not.

Common research for IDS development involves testing supervised classifiers and unsupervised anomaly or one-class detectors as the decision engine algorithm. The decision engine can then be configured to inform a user or some automated response system.

The decision engine can be categorized as *misuse*, *anomaly*, or a *hybrid* detector. Misuse intrusion detection uses predefined attack patterns, e.g., signatures of known malware or expert-crafted rules, to flag matching events. Consequently, zero-day attacks, i.e., novel attacks or attacks exploiting previously unknown vulnerabilities, generally bypass misused detection algorithms. Misuse detection systems dominate IDS use in practice, as they have been the main

focus of commercially-driven detection products. Host-based anti-virus such as McAfee[1] and Kasperski[2] and network-level rule-based systems such as Snort[3] are examples of very popular misuse detection systems. Generally, the first line of defense, misuse detection relies on an database of attack signatures, which is generally large, constantly growing, can be cumbersome to use efficiently and necessitates regular updates. However, misuse detection generally realizes a relatively low number of false positives, but a high number of false negatives.

In anomaly detection, a description of normal or expected behavior is learned from observations and a sufficient deviation from this normal profile is flagged as a potential attack; thus, the detection of never-before-seen attack patterns is possible. Anomaly detection systems that update in near real time can evolve models with the slowly changing system [33, 53]. The primary downside of anomaly detection is detection accuracy, most notably, that these techniques suffer from higher quantities of false alarms. Moreover, attacks can hide in the noise floor of ambient data if training data (from which normal behavior is learned) exhibits large variance. Similarly, if attacks are present in training data, detectors will potentially be trained to regard such behavior as normal [135].

Hybrid systems are also often studied; these take into account previous knowledge but seek to generalize to unseen data. For example, systems are proposed that seek to complement misuse detection with anomaly detection, using them in tandem [149]. When datasets with labeled intrusions are available, research will often experiment with combinations of feature selection and supervised learning algorithms. Supervised learning classifiers are generally less rigid than traditional misuse detection systems, as they are trained to generalize previously seen attack and non-attack examples.

Feature selection is influential on both accuracy and performance of IDS classifiers. In many applications, the number of features can grow to enormous quantities, but as feature vectors gain length so does the computational complexity, quantity of training data, and time needed for both training and inference. Additionally, poor features both decrease performance and add noise, reducing accuracy of the classifier while contributing to expense. To combat these factors, methods of dimension reduction seek to identify redundancy and find correlations in features, thereby reducing the number of features without losing information. This careful choice of features is the focus of many detection efforts. Where progressions of research built on the same datasets exist (e.g., see Sections 4), research generally trends from using raw data as features, to considering cost-to-accuracy benefits of various hand-crafted features, to data-driven techniques for dimension reduction and feature selection/creation.

Biased classes, in our case where non-attack data is in far more abundance than attack data, are a perennial problem for classifiers and a looming issue in the intersection of machine learning and intrusion detection. Much research seeks to use hybrid methods, ensembles, and advanced feature selection algorithms to circumvent the problem.

While incomplete training data (in particular, not having representative attack data available) is an issue for misuse supervised detectors, noisy training data is a common problem especially for anomaly detectors, which often characterize normal data from a history of the data. Most notably, if unknown attacks exist in training data, the detector may regard similar future attacks as normal. Robust statistical methods have been used to discard outliers when fitting anomaly detection models, which can help address these problems. As our survey is organized by data source, unique approaches to these challenges are pointed out in the sections in which they occur.

[1]See www.mcafee.com.
[2]See www.kaspersky.com.
[3]See www.snort.com.

1.2 IDS Location

An IDS is most often categorized based on the information source utilized by the IDS and its position within the network architecture. Since an IDS's capabilities depend largely on what data it has available [105], location is a critical architectural decision. This can be viewed most coarsely as network-based IDS (NIDS) versus host-based IDS (HIDS). Host-based IDSs are generally a software component located on the system being monitored and typically monitor a single system. This gives HIDS excellent visibility into the system state but poor isolation from the system, meaning that an attacker with access to the system can either mislead or disable the HIDS. Additionally, host-based data is often context-rich, allowing deeper understanding of processes and activities, but comes with added costs of requiring access to the host, configuration of distributed clients, and often requires collecting and managing potentially large and sensitive datasets from these hosts. Network IDSs are generally physically separate devices, located on the network "upstream" of the system being monitored, and they generally monitor many separate systems on a common network. The NIDS is often completely transparent to the systems being monitored, which provides good isolation and makes NIDSs much less susceptible to any interference from an attacker. However, these systems have little or no information available about the internal state of the systems they are monitoring, which can make detection more difficult. Further, network traffic is increasingly encrypted, potentially posing problems for NIDS [12, 122], although we note that "break-and-inspect" capabilities are becoming prevalent in practice to allow encryption of all traffic while giving visibility to network appliances, e.g., https://www.symantec.com/products/ssl-visibility-appliance.

Hybrid and distributed IDSs (DIDS) combine information from multiple sources into one system. Hybrid IDSs combine both host-based and network-based data, generally with the goal of achieving more complete visibility of a host. Distributed IDSs combine data from multiple sensor locations into a combined decision-making or combined alerting process; this may use information from host-based sensors, network-based sensors, or both.

The use of virtualized resources, e.g., cloud computing environments, provide opportunities for monitoring virtualized hosts from different locations, with trade offs in visibility and capabilities. For example, traditional IDSs can reside inside the virtualized host, or one can gain isolation from infections or compromises of the host, at the cost of poorer visibility into the host, by monitoring host data at the hypervisor-level to perform detection of one or many guest Operating Systems (OSs). Virtual machine monitor (VMM) IDSs involve monitoring a virtual machine's (VM) OS (or in some cases, its applications or services) from a logically external location on the same physical machine. Several of the above IDS techniques have been utilized within cloud environments. These cloud IDSs can include network-based data, host-based data, or both, and, while cloud infrastructure relies heavily on VMs, these systems do not necessarily include the same techniques as VMM IDSs.

Among HIDS, most systems can be categorized as either a program-level or OS-level IDS. Program-level IDSs focus on monitoring a single application, using information such as source code, byte code, system calls invoked, static or dynamic control flow, and other information on the application's state. Much of this relies significantly on research in related topics, such as vulnerability detection and malware analysis/detection, but here we focus on works that take these techniques and apply them to detecting intrusions or anomalies in applications at run-time.

OS-level IDSs monitor the overall system state, and may monitor the combined behavior of all processes, to distinguish between normal and abnormal behavior at the OS level. This can involve collecting data from system logs, Windows Registry data, system calls invoked, file system monitoring, or other sources. System calls have been well utilized for the detection of normal and

anomalous behavior. While sequences of system calls for a single application can be used in a program-level IDS, these are often combined, by monitoring all system calls of all processes, for use in an OS-level IDS. System call traces, the sequence of system calls of a given process, are used to find repeated patterns of system calls, enabling anomaly detection and misuse detection during execution [84].

Finally, we note that side-channel detection—detectors leveraging physical characteristics such as power consumption, electromagnetic radiation, vibrations, timing statistics, and so on—has gained traction in cyber-physical IDS research [20, 21, 42, 62, 67, 129] but is a growing area of research for traditional computers using host-level (albeit physical) data [25, 67]. For most of these works, the primary advantage is the detectors' physical separation from the host, which prevents software intrusions from tampering with the detector, and the fact that malicious changes to host necessarily induces physical changes from normal behavior. These researches are considered out of scope for the current survey.

1.3 Scope and Organization

This survey focuses on HIDSs, and attempts to capture and organize the variety of data sources used, methods tested, and general trends in the HIDS research. Because of the large volume of work in this area, we cannot comprehensively cover all relevant works. We prioritize a broad coverage each host-based data source and its use for intrusion detection, and we discuss research trends over time for each of these data sources. While VMM-IDSs and DIDSs leverage host-level data, their contributions generally focus on new architectures, instead of the analysis itself, and for this reason they are not included.

Section 2 describes several other IDS surveys. The following sections are sub-surveys of HIDS research, each for a different input data type. Section 3 focuses on system logs and audit data. While system calls are considered audit data, a flurry of targeted detection research brought on by labeled data sets merited their own section, Section 4. Section 5 gives, to the best of our knowledge, a comprehensive description of the few IDS works leveraging Windows Registry data. Section 6 reviews works leveraging file system monitoring for identifying malicious files, while Section 7 discusses a few works that leverage information about processes or stored binaries on a host for detection. To assist current research, the publicly available datasets and databases referenced in the literature for IDS validation are collected in Section 7. Our final section gives conclusions.

Overall, this survey organizes the diverse data sources available on a host that have been used for detecting intrusions, the literature developing IDS on each data source, and those open-source datasets needed for testing research in the area. All sections are accompanied by one or many tables, itemizing the discussed references and presenting their key characteristics for comparison. By organizing the literature by data source, we hope that current researchers (1) quickly see the panorama of data sources available for HIDS research, (2) for a given data source of interest, elicit progressions in the literature and identify gaps, trends, or novel directions for future contributions, and (3) can more quickly identify datasets needed for evaluating novel ideas.

2 RELATED SURVEYS

Several surveys provide discussions on existing IDS research, and we review the recent, related ones in this section. Table 1 presents a quick comparison, as it details many discussion topics and attributes of the surveys.

Axelsson [8] surveys anomaly and misuse papers pre-2000 and organizes the research by sorting on the proposed problem's level of difficultly.

Patcha and Park [115] perform an in-depth review of anomaly and hybrid-based intrusion detection papers spanning from 2000 to 2006. The papers are organized based on the classification

Table 1. Existing IDS Survey Comparison

Survey	IDS Types	IDS Location	Datasets	Data Source	Feature Selection
This Survey	HIDS, NIDS	Discussed	Discussed	Discussed	Discussed
Axelsson'00 [8]	HIDS, NIDS	NA	NA	Discussed	Discussed
Patcha and Park'07 [115]	HIDS, NIDS	NA	Discussed	Discussed	Discussed
Kabiri and Ghorbani'05 [70]	NIDS	NA	NA	NA	Discussed
Lazarevic et al.'05 [91]	HIDS, NIDS, DIDS	NA	NA	Discussed	NA
Sabahi et al.'08 [126]	HIDS, NIDS	Discussed	Discussed	Discussed	NA
Mehmood et al.'13 [102]	HIDS, NIDS, DIDS, VMM	Discussed	NA	Discussed	Discussed
Liao et al.'13 [96]	HIDS, NIDS, DIDS, VMM	NA	NA	Discussed	NA
Modi et al.'13 [105]	HIDS, NIDS, DIDS, VMM	Discussed	NA	NA	NA
Kumar and Gohil'15 [89]	HIDS, NIDS, DIDS	Discussed	NA	NA	NA
Kahn et al.'16 [78]	HIDS, NIDS	Discussed	NA	NA	NA
Chiba et al.'16 [19]	HIDS, NIDS, DIDS, VMM	Discussed	NA	NA	NA
Buczak and Guven'16 [16]	NIDS	NA	Discussed	Discussed	NA
Mishra et al.'17 [104]	VMM	Discussed	NA	NA	NA

algorithm used and discussed in terms of existing challenges, such as high false alarm rate. Additionally, numerous open challenges, such as failure to scale to gigabit speeds, are discussed.

Kabiri and Ghorbani [70] primarily focus on NIDS, but discuss the importance of feature selection with respect to dimension reduction, importance of the features, and their relation to one another in the feature space, which is neglected as its own topic in other surveys.

Lazarevic et al. [91] extensively cover attack types and categorizes them into classes. A generic architecture is defined for an IDS. The survey provides an overview on IDS taxomony and discusses information sources, including system commands, accounting, and logs as well as security audit processing. However, user-level logs, process profiling, file system, registry, and raw pages/introspection are excluded.

Sabahi et al. [126] provide a very brief survey of different IDS systems including HIDS, NIDS, and DIDS, covering data sources used to conduct detection and detection methods, such as misuse detection, protocol analysis and anomaly detection. They mention that detection can be conducted online or offline and provide examples of both centralized and distributed architectures.

Mehmood et al. [102] provide an overview of different intrusions for cloud-based systems and analyze several existing cloud-based IDSs with respect to their type, positioning, detection time, detection technique, data source, and attacks detection capabilities. The analysis also provides limitations of each technique to evaluate whether each fulfills the security requirements of the cloud computing environment.

Liao et al. [96] present a comprehensive survey of IDSs concentrating on signature-based, behavior-based, and specification-based methods. These detection methods are further divided into "statistics-based, pattern-based, rule-based, state-based, and heuristic-based" approaches.

Modi et al. [105] offer recommendations for IDS/IPS placement within cloud environments to reach common security goals in next-generation networks.

Kumar and Gohil [89] discuss traditional attack types and analysis techniques used for HIDSs, NIDSs, and DIDSs.

Khan et al. [78] briefly discuss HIDSs and NIDSs, and discuss their architecture and applicability, as well as highlighting shortcomings, such as the high communication and computational overhead of some approaches. A parametric comparison of the threats being faced by cloud platforms

is performed, which incorporates a discussion of how various intrusion detection and prevention frameworks can apply to various common security issues.

Chiba et al. [19] discuss cloud-based IDSs and analyze the systems based on their various types, positions, detection type, and data source. Strengths and weaknesses are discussed to determine the IDSs validity in a cloud computing environment.

Buczak and Guven [16] provide a survey of machine learning approaches for IDSs. Their work provides brief descriptions of important algorithms, including a table with algorithmic time complexity. This non-comprehensive survey primarily includes examples of NIDS research, with selection criteria for influential works to include examples of each classification algorithm used for an IDS. Similarly, a few data sources, e.g. packets, are discussed in detail, along with several open source datasets.

Mishra et al. [104] heavily focus on virtual machine introspection (VMI) and hypervisor introspection (HVI) as IDS techniques, and compares cloud security with network security. Cloud-specific threats and vulnerabilities are discussed via an attack taxonomy. Challenges are briefly discussed including availability of data sets, IDS position, performance, and IDS limitations. Parallel programming and the usage of GPUs are mentioned for performance improvement.

This survey provides an in-depth discussion of IDS work that leverages host-based data sources for attack detection. We organize works based on their data source with the goal of giving the interested reader a panoramic view of the different avenues for detection. Furthermore, this work provides an in-depth introduction to available host-level data sources, and discusses and their uses and limitations. Works that focus on algorithmic development, but are tested on network-level data, are included where they apply to HIDS. Additionally, a supplementary includes a list of publicly available datasets used in the research literature for evaluating IDSs, outlining their characteristics and shortfalls. This survey of HIDS research literature complements our concurrent work reporting results of interviews with security operators on how and what host data and tools they employ [12].

3 SYSTEM LOG AND SYSTEM AUDIT DATA IDSs

Log files are a collection of system-generated records that detail the sequence of events of a server, an OS, or an application. The log files are processed or stored for various analyses or forensics. Most programs and applications generate separate individual log files, associated with activities conducted by those programs' processes. As an example, a system log file is usually associated with records produced by the OS, including but not limited to warnings, errors, and system failures. Individual applications may produce log files associated with user sessions containing login time, authentication result, user-program interactions, and so on. While an OS-produced log file is considered a system log file, files produced by individual applications or users are considered audit data. Examples include successful and failed authentication logs, system calls, or user command logs.

Since such data sources document the sequence of events of the system or programs, they are a promising resource for detecting intrusions, as an HIDS can leverage the data to profile behavior of an individual user or system. Conversely, the downside to high fidelity audit data is the collection cost. Below, we survey literature leveraging system logs and audit data for intrusion detection. Table 2 itemizes the System Log IDS works surveyed. We note that system calls can be considered a subset of audit data, but because there is a rich progression of research that considers them independently, system-call-based IDSs merit their own section, Section 4.

3.1 System Logs IDSs

With any IDS, the goal is to perform in a cost-effective, adaptable, intelligent, and real-time manner.

Table 2. HIDS with System Logs

IDS Reference	Technique	Dataset	Classifier	Learning	Classes
Ryan et al.'98 [125]	Anomaly	NA	NN	Unsupervised	Binary
Reuning'04 [124]	Anomaly	DARPA99	TF-IDF	Unsupervised	Binary
Zhaojun and Chao'10 [165]	Anomaly	NA	NN	Supervised	Binary
Tchakoucht et al.'15 [144]	Anomaly	Simulation	Clustering	Unsupervised	Binary
Wang and Zhu'17 [154]	Anomaly	KDD99	C5.0 DT	Supervised	Binary
Verma and Bridges'18 [150]	Misuse	NP	NN	Supervised	Binary

This is especially challenging when analyzing system logs, which can be CPU intensive and typically requires human expertise. System log analysis requires that all performed actions by the OS be stored, and then feature extraction and classification can then be performed.

The following papers focus on analyzing system logs and various ways to achieve this goal. We break them into two subcategories—those focusing on detection accuracy, and those focusing on IDS architecture.

Due to the extensiveness of the topic, not all papers could be included in this survey. Other notable works include, but are not limited to, the References [29, 123, 151, 152].

3.1.1 System Log IDS Research. Reuning [124] describes an anomaly detection system based on Bayesian probability theory and the term frequency inverse document frequency (TF-IDF) information retrieval technique. TF-IDF is applied to event log messages, treating each entry as an individual document. First, system training is required, in which data over a chosen time interval is collected and indexed into hash table, where each term is mapped to its TF-IDF weight. Messages with high scores, defined as the sum of the TF-IDF scores of the messages' terms, are detected. Results of the experiment on the DARPA99 dataset suggest that using log data solely produces a high false-positive rate and many undetected attacks, but it can become a valuable component of a larger and more complex IDS.

Tchakoucht et al. [144] improve upon the IDS of Yacine et al. [11]. The goal is to help decrease User-to-Root (U2R) and Remote-to-Local (R2L) attacks that exploit operating system or software vulnerabilities. User activity is audited based on LoginFlow, LoginFails, SessionDuration, SessionCPU, FormatCounter, AccessFails, DataVolume, and QuotaOverloadFails, which provide a feature vector representation for each user's behavior over a given time period. To characterize user behavior, k-means clustering identifies groups of similar user behavior. Euclidean distance is calculated to compare new user behavior to the reference profile in the detection phase. An experiment was conducted with a health information system consisting of three users, a patient, doctor, and an administrator, including their behavior over 30 days. The experiment resulted in significant improvements during learning and testing over Yacine et al.'s previous work [11] with a sizable increase in successfully identifying users and a large decline in false positives. Achieving good results, Tchakoucht et al. identify two constraints that can affect accuracy, change in user behavior and the system's inability to handle large datasets.

Verma and Bridges [150] consider host logs, and note that while such logs are collected by security operation centers, they are high volume and only semi-structured (individual entries are sparse but there are a large amount of possibly, and many field types are strings) making information extraction from log streams tedious. To automate use of logs, in particular for intrusion detection, Verma and Bridges build a general metric space structure for the logs allowing distance to be computed between single log entries as well as sequences of logs (e.g., streams of logs from a host). This distance is claimed to preserve semantic meaning; that is, logs close to each other

in the metric report similar events. To test efficacy of this metric space, Windows Even Logs are analyzed with three use cases. (1) After building a behavior-based IDS from a first instance of ransomware on a host, the method shows high accuracy detection of three polymorphic copies of this ransomware and a handful of log streams with no infection. The detection simply compares a sequence of logs with the known bad sequence under the new distance metric (1 NN detection). Authors note that detection occurs faster than a standard AV tool. (2) Clustering of users based on log streams provides a data driven process to identify user types. (3) Informed visualizations of a user's daily activities permits anomaly detection. Authors note that this metric space is formulated in generality to apply to other databases (not just host logs), and that the three experiments presented used real network data, but limited in quantity—more extensive testing is required.

3.1.2 System Log IDS Architecture Research. Early work using system logs includes a neural network (NN) of Ryan et al. [125] trained to predict the distribution of commands employed by a user over a fixed time window. Hence, the output of the supervised NN framework is compared against the users current command use to detect anomalies, an unsupervised detector. Initial experiments using random vectors as ground truth anomalies produced anomaly detection rate of 96%, false alarm rate of 7%. Building on the NN framework of Ryan et al., Guan et al. [44] introduce KIT-I, an architecture for an IDS using system log data. Log data is stored in a secure server, so even if the computer is compromised, an intruder would not have the ability to modify log files to cover attack traces. The system consists of two modules, a transferring module and a neural network (NN) module. The transferring module is used to transfer log data at defined intervals from a client to a remote logging server via a secure channel, which is implemented using SSL of Java and Certificate Authority for client to server authentication. The NN module is used to analyze received log files for abnormal behavior. No experiments are presented by Guan et al.

Zhaojun and Chao [165] propose an HIDS based on system logs. The architecture contains five modules—log collection and pre-processing, saving and updating, search and analysis, statistics and analysis, and alarming. During execution of the first four modules, system logs are collected and turned into records containing fields extracted from three parts of the system logs, namely, a priority with "Facility" and "Severity" fields, a header with "Timestamp" and "Hostname" fields, and a message with "Tag" and "Content" fields. A constructed record is stored in a MySQL database and filtered using regular expressions to extract important records. For the decision engine, records from the database are transformed into numerical values and then passed through a back-propagation NN (BPNN) model for analysis. Once analysis is complete, the alarm module determines how to inform the user, if necessary.

Wang and Zhu [154] propose a centralized HIDS architecture for private cloud computing, with the main goal to reduce usage of system resources. Their model is built on OpenStack,[4] an open-source infrastructure platform for cloud computing, and consists of three nodes, compute, controller, and network nodes, and four modules, data collection, data pre-processing, detection, and alarm modules. The collection module uses Logstash[5] to gather system logs from all VMs and stores it into Elasticsearch[6] for farther analysis by the detection center, which uses a C5.0 decision tree (DT). If an anomalous event is detected, then the detection center alert to victim VM. This model was tested using the KDD99 dataset and compared to a traditional HIDS. Comparing the new centralized HIDS with a traditional HIDS shows that a centralized HIDS CPU utilization is approximately 14% lower, memory consumption is about 2% less, and the detection rate of 94% is about the same with a slightly longer detection time.

[4]See www.openstack.org.
[5]See https://www.elastic.co/products/logstash.
[6]See https://www.elastic.co/products/elasticsearch.

Table 3. HIDS with Audit Data

IDS Reference	Technique	Dataset	Classifier	Learning	Classes
Ilgun'93 [64]	Misuse	NA	Rule inference	NA	Multi
Ye et al.'01,'02 [160, 161]	Misuse, Anomaly	DARPA98 + simulation	DT, T^2 test, χ^2 test, Markov model	Both	Binary
Botha and Von Solms'03 [10]	Misuse	Self made	Rules + Fuzzy logic	NA	Multi
Li and Manikopoulos'04 [95]	Anomaly	Self made	OCSVM	Unsupervised	Binary
Shavlik and Shavlik'04 [133]	Anomaly	Self made	Winnow, NB	Unsupervised	Binary
Mehnaz and Bertino'17 [103]	Anomaly	Self made	FSA rule-mining	Unsupervised	Multi

3.2 Audit Data IDSs

In this survey, audit logs will refer to more granular information than system logs, collected with the goal of providing a chronological, detailed record of user activities. For example, audit logs allow visibility into network connections (e.g., source/destination bytes, protocols, etc.), command line actions (e.g., number of shells opened), privilege escalations, and changes to files. System calls are included in the audit logs, but the wealth of IDS research using them is discussed separately in the next section. Audit logs are high volume and costly to collect and manage, but they give higher fidelity for forensics and detection. Table 3 itemizes audit log IDS works surveyed.

Ilgun [64] illustrates a real-time IDS for UNIX operating system called USTAT (State Transition Analysis Tool for UNIX), which is the UNIX version of STAT described by Porras et al. [120]. It is a rule-based IDS and works by matching known patterns to the sequences of audit data gathered by the audit collection mechanisms of the OS. Some of the aims of USTAT are to automate a matching process and make patterns more flexible to adopt to different instances of equivalent attacks. This proposed IDS is able to detect attacks that involve cooperation of multiple user sections or accounts. USTAT analysis is based on state changes, where state is "the collection of all volatile, permanent, and semi-permanent data stores of the system at a specific time" and changes are called actions; therefore, an attack pattern is defined as a sequence of attacker actions. There are four main components, the data pre-processor, the knowledge-base component (containing fact-based data of objects of the system and rule-based data of state transitions), the inference engine (used to infer all states of the system and detect attacks), and a decision engine (used to chose an action and inform the user about results from inference engine). The conducted experiment was not focused on detection accuracy but instead on resources utilization running USTAT with other processes. This resulted in a limitation of disk throughput when running both USTAT and an audit daemon that collects audit trails.

Ye et al. [161] study data attributes for intrusion detection. Attributes include: (1) individual event occurrences (e.g., "audit events, system calls, user commands"), frequencies (e.g., "number of consecutive password failures"), and durations (e.g, "CPU time of a command, duration of a connection"), (2) event combinations, (3) multiple events frequency and distribution, and (4) event sequence/transition. They compare the intrusion detection performance of four methods—a supervised DT and three unsupervised anomaly detection algorithms utilizing both Hotelling's T^2 test (T^2 test) and the χ-squared distance (χ^2 test)—both multivariate statistical analysis methods, and a first order Markov model—for intrusion detection in their experiments with the DARPA98 dataset and simulated attacks. The Markov chain based on an ordering property showed superior performance. This verifies that the ordering and frequency of audit events provides useful information to detect intrusions. Follow-up work by Ye et al. [160] presents more results comparing T^2 test and the χ^2 test, on audit trails to detect anomalous behavior. The proposed techniques are better in session-wise analysis (an entire session is considered an intrusion if it contains a single intrusive event), and overall performance of χ^2 is better than T^2.

Botha and Von Solms [10] implement a hybrid IDS based on comparing user actions with intrusion actions using fuzzy logic. These actions are interpreted as phases of an intrusion, which they describe using their own schema: "probing," "initial access," "super-user access," "hacking," "covering," and "backdoor." These are represented as a graph for comparison using fuzzy logic. The authors developed a working prototype, and testing was done with the help of 12 users, where ten users were conducting both "legal" and "illegal" (presumably normal and unauthorized) activities and two users were conducting only legal activities. The system correctly identified both users conducting "legal" activities by assigning intrusion probabilities of 0%, while the remaining users had probabilities of intrusive activities between 12% and 48%. Output of the HIDS upon an alert gives probabilities for a variety of attacks, e.g., data loss, denial of service, and so on, likelihood.

Li and Manikopoulos [95] model user profiles with one-class support vector machines (OCSVMs), an unsupervised support vector machine (SVM) technique for detecting anomalies. The OCSVM approach requires the user's legitimate sessions to build the user's profile, specifically using a year of Windows audit data. Authors focus on masquerade detection. This approach allows for easier user management, such as adding and removing users, rather than multi-class classification methods. Results show the two-class training achieves a detection rate of 63% with a 3.7% false alarm rate and one-class training shows a 66.7% detection rate with a 22% false alarm rate. Even though the one-class training approach results in increased false alarms, this is offset by easier management and a reduction in training time.

Subsystems can monitor an abundance of system actions in the Windows OS. An anomaly detection system is presented by Shavlik and Shavlik [133] that performs statistical profiling of users and system behavior on Windows 2000. Measurements taken from 200 Windows 2000 attributes at one second intervals generate approximately 1,500 features. Examples of features are encodings of CPU utilization, data input-output quantities, process information, and differences and averages of current versus historical values, among others. The features are fed into Winnow [98], a supervised algorithm in which each non-zero feature has a weighted vote for the positive class. The weighted votes are then compared against a threshold to determine if there is an intrusion. To train the weights, other users' activity is labeled as anomalous and mixed with the current users' data to create a binary labeled training set; hence, by using others' activity as ground-truth anomalies, the overall detector is unsupervised, as no labeled "attack" data is needed. During training, Winnow changes the weight of features that fired on incorrectly labeled instances, similar to perceptron training. Winnow, is tested against a Naïve Bayes (NB) classifier. Self-collected data from multiple hosts is used for gathering a baseline for normal users, and the same from a held-out set of hosts is labeled "intrusions," with the motivation of identifying insider threats. Winnow yields a 95% detection rate with a low false-alarm rate (under one per day per computer), while NB has a 59.2% detection rate and has 2 false alarms per day.

Mehnaz and Bertino [103] present Ghostbuster, a HIDS that profiles users based on their file-system access patterns and detects anomalies. The Linux utility `blktrace`[7] is used to extract sequences of file access events. During the profile creation phase for each user, a feature vector is created by encoding file access by sizes (with blocks as units), frequencies, and patterns of files accessed. Statistical outliers of file access size and frequencies are a cause for alerts, and a finite state automata (FSA) for access patterns defines rule-based anomaly detection with six classes of anomalies. Performance evaluation is given for actual file accesses of 77 users for 560 target files over eight weeks, four for training and four for testing. Results are given for many simulated attacks, and overall high detection rates and low false-positive rates are reported; overhead is reported at 2%.

[7]See https://linux.die.net/man/8/blktrace.

We note that research for utilizing audit log data for intrusion detection in a cloud environment is a budding area of research, but is outside of the scope of this work [92, 109, 159, 163].

4 SYSTEM CALL IDSs

System call data is a popular choice for HIDS research, because they are a primary artifact of the OS kernel; that is, there is no filtering, interpretation, or processing (such as, in the production of log files) that can obfuscate events [24]. Often the unit of data used for detection is a system call trace, a sequence of all calls invoked by a single process in a given time window. Hence, these IDS developments sit at an OS-level, but the object of modeling is program level. System calls can be collected for example with the "strace" utility, although there are many other ways to collect this same information. Some common calls include "open," "close," "read," "write," "wait," "exit," "mmap," among many others. Modern OSes often have hundreds of syscalls, for example, the "syscalls" Linux manual page lists over 300. Drawbacks of system-call-based approaches include the large computational overhead needed for harvesting and analysis and the large possible variations that potentially lead to false positives [7].

Because each process produces a sequence of system calls, language modeling techniques are prevalent for system call-based HIDSs. In particular, many variations on *n*-gram features and Markov models (MMs) of sequences of calls are configured to produce normal/attack classifications. See Forrest et al. [34] for a more detailed survey of pre-2008 works leveraging system calls. Critical insights from this section are as follows:

- While research has shown that normal processes can be profiled using system calls, wide variations occur across processes, or for fixed processes across different user environments, installation configurations, and so on.
- Detection results are quite sensitive to the length of sequence-based features with six- to eight-grams being strong choices.
- Short sequences provide less computation during training but are easier to bypass than longer sequences.
- Augmenting calls with other information, such as arguments of the calls, program counters, and addresses, can yield higher accuracy with less overhead.

Overall, general trends indicate that features that model sequences are more costly than simple frequency counts of individual features, but yield better detection. Finally, meta-trends show that as labeled datasets become popular, a flurry of research ensues allowing IDS comparisons across papers and testing of many standard machine learning algorithms to flourish. See Table 4, which itemizes the system log works surveyed.

4.1 Sequential Features (*n*-Grams)

Early work of Forrest and Longstaff [35] provide preliminary HIDS results by characterizing normal (frequent) and then identifying abnormal (infrequent) short sequences of System calls. One way to conceptualize the main idea is that System calls are "words," sequences of calls form "phrases." The general trend incurs relatively large computational expense for feature extraction and/or model training, but reap strong detection metrics.

Other anomaly detectors based, similarly, on modeling *n*-grams of system calls were explored by others, but without statistically modeling their frequencies [58, 84]. Helman and Bhangoo [55] rank system call traces by the likelihood of *n*-grams in normal versus attack scenarios. Ye et al. [162] use set theory to design an algorithm that learns rules defining normal system call sequences, then detect anomalies based on votes from the rules, although no testing is presented.

Table 4. HIDS Leveraging System Calls

IDS Reference	Technique	Dataset	Classifier	Learning	Classes
Forest et al.'96 [35]	Anomaly	Simulated	Rules	Unsupervised	Binary
Kosoresow et al.'97 [84]	Anomaly	Self made	FSA	Unsupervised	Binary
Hofmeyr et al.'98 [58]	Anomaly	Simulated + Self made	FSA	Unsupervised	Binary
Ghosh et al.'99 [39, 40]	Hybrid	DARPA 99	NNs	Both	Binary
Warrender et al.'99 [155]	Hybrid	UNM, DARPA98	Rules, HMM	Unsupervised	Binary
Sekar et al.'01 [132]	Anomaly	Simulated + Self made	FSA	Unsupervised	Binary
Wagner and Dean'01 [153]	NA	Self made	Static Analysis	NA	Binary
Liao and Vemuri'02 [97]	Hybrid	DARPA98	k-NN + Signatures	Both	Binary
Abad et al.'03 [1]	Hybrid	Self made	RIPPER	Both	Binary
Feng et al.'03 [32]	Anomaly	Simulated + Self made	FSA	Unsupervised	Multi-class
Hoang et al.'03,'09 [56]	Hybrid	UNM	HMM + Rules	Unsupervised	Binary
Kruegel et al.'03 [87]	Anomaly	DARPA 99	BN	Unsupervised	Binary
Kruegel et al.'03 [88]	Anomaly	DARPA 99	Probability models	Unsupervised	Binary
Jha et al.'04 [66]	Anomaly	UNM	Filtering, MM	Unsupervised	Binary
Tandon and Chan'05,'06 [141, 142]	Anomaly	UNM, DARPA98	Rules	Unsupervised	Binary
Han and Cho'05 [52]	Anomaly	DARPA99	ENN	Unsupervised	Binary
Zhang et al.'05 [164]	Hybrid	DARPA98	k-NN, Robust SVM, SVM, OCSVM	Both	Binary
Gao et al.'06 [37]	Anomaly	Self made	HMM-based distance	Unsupervised	Binary
Hu et al.'09 [60]	Anomaly	UNM, DARPA98	HMM	Unsupervised	Binary
Ahmed et al.'09 [2]	Hybrid	UNM	RBFNN	Supervised	Binary
Tong et al.'09 [145]	Hybrid	DARPA	RBFNN + ENN	Supervised	Binary
Ye et al.'10 [162]	Anomaly	NA	Rules	Unsupervised	Binary
Jewell and Beaver'11 [65]	Anomaly	Self made	Rules	Unsupervised	Binary
Elgraini et al.'12 [30]	Anomaly	UNM	NB with a MM	Unsupervised	Binary
Xie et al.'13,'14 [156, 157, 158]	Anomaly	ADFA-LD	k-NN, OCSVM, k-Means	Unsupervised	Binary
Creech and Hu'14 [24]	Anomaly	UNM, ADFA-LD, KDD98	ELM NN	Supervised	Binary
Anandapriya and Lakshmanan'15 [4]	Anomaly	ADFA-LD	SVM, ELM NN	Supervised	Binary
Gupta and Kumar'15 [45]	Misuse	UNM	Rules	Unsupervised	Binary
Haider et al.'15 [50]	Anomaly	ADFA-LD	k-NN	Unsupervised	Binary
Rachidi et al.'16 [122]	Anomaly	DARPA99	Silhouette, NB, MM	Supervised	Binary
Mouttaqi et al.'17 [108]	Anomaly	UNM, ADFA-WD	NB, MM, Adaboost	Supervised	Binary

For each process, Jewell and Beaver [65] consider variable length sequence of system calls, defined as an observed sequence of system calls for which no call occurs twice. Comparing this with other sequential features, e.g., n-grams, they observe that the counts of the observed system call sequences plateau for normal user activity faster than other definitions and that the counts spike upon novel activity. With the goal of identifying malicious data exfiltration activities in real-time, an experiment in which researchers were challenged to exfiltrate three file collections on a given set of machines over two days is used to collect malicious and normal system call data, which is used to validate the approach.

Elgraini et al. [30] estimate the probability of a sequence of calls conditioned on the class (normal/attack) using a first order Markov model (MM) —$P(s_1, s_2, \ldots|C) = P(s_1|C)P(s_2|s_1, C)\ldots$. Finally,

a NB approach is used to find the most likely class. Results are compared to many other previous classifiers on data from the University of New Mexico (UNM), finding that this method performs similarly.

Creech and Hu [24] make two innovations for a HIDS based on kernel level system call traces, (1) creating semantic features of system call sequences (phrases) by defining a context free grammar and (2) using an extreme learning machine (ELM)—a neural network (NN) classifier of Huang et al. [61]. This approach takes per-host training that is computationally costly, taking days or weeks, although once trained, labeling (or decoding) is fast and accuracy results are very strong, reported via the Receiver Operating Characteristic (ROC) curve using the Darpa98[8] and ADFA-LD datasets. Anandapriya and Lakshmanan [4] also test anomaly detection results using semantic features with the ELM on the ADFA-LD dataset.

Gupta and Kumar [45] define a signature for a program as the admissible bigrams of calls, specifically those seen in training. This allows lightweight detection of programs with a variety of new two-sequences of calls that gives highly accurate results as tested on the UNM dataset. Their work discusses implementation for cloud infrastructure using multiple VMs.

Mouttaqi et al. [108] implement a Naïve Bayes classifier on the older (2004) UNM dataset showing very strong detection results, yet similar testing on the newer ADFS-WD dataset reveals severely less impressive detection capabilities with all accuracy metrics (F1, detection rate, AUC, ...) 20–40% worse. Markov models (MMs) of order 1–3, and Adaboost ensemble of MMs are also tested, showing themselves drops in efficacy between UNM and ADFS-WD datasets, but exhibiting much more impressive results. Authors' conclude that the third order MMs are the best of the tested models as it has the lowest error (2.4%), false alert rate (8.5%), and best detection rate (100%) on the ADFS-WD data set.

4.2 Frequency-based Features: A Cheaper Alternative

In response to the costly but effective sequence-based features, research to develop and test more computationally inexpensive, frequency-based features from system call traces finds, at least for the AFDS-LD dataset, that such features still produce strong accuracy results.

Liao and Vemuri [97] regard traces as documents represented with the vector of TF-IDF scores for each word (system call). The k−nearest neighbor (k-NN) with cosine similarity distance is used for anomaly detection. If a process is classified as intrusive, then the whole session it belongs to is also considered an attack session. Liao et al. performed the experiment using the names of System calls recorded in Basic Security Module[9] (BSM) audit data from DARPA98 dataset; they exhibited over 90% TPR with under 2% FPR. The second experiment preempted this TF-IDF anomaly detector with signature verification. First, each process is compared to a set of abnormal processes using cosine similarity, and, if they match, the process is marked as intrusive. Otherwise, the k-NN anomaly detection process is used to classify the process. This two-stage workflow produced 91.7% detection rate and 0.59% false-positive rate with threshold of 0.8. This method is computationally efficient, with complexity $O(N)$, with N as the number of processes.

Continuing the system call/trace interpretation as words or documents, respectively, Zhang et al. [164] propose two novel techniques to lower false positives. First, a modified TF-IDF score is crafted from system call traces; second, the authors build a detector using supervised training with Robust SVMs to battle noisy training data, OCSVMs for unsupervised training, and k-NN. Online-training of the SVMs is used to decrease training time while preserving accuracy of intrusion detection. Clean and noisy datasets are generated from system calls of privileged processes in the

[8]Darpa98 is sometimes referred to in other literature as KDD98.
[9]See https://docs.oracle.com/cd/E19457-01/801-6636/801-6636.pdf.

DARPA98 dataset, these are used to compare each classifier with and without their modifications. Results showed that the detection accuracy of the modified classifiers is the same or higher than the baseline, when tested with both the clean and noisy datasets, while the training time ratio for the modified SVM over the original is between 51.61% and 66.67% (i.e., retraining is significantly faster).

Xie and Hu and Xie et al. [156–158] consider simple features such as a trace's length, and the relative frequency of each call in that trace, and achieve "acceptable" detection results (i.e., ROC curves) in their testing with the ADFA-LD dataset, using simple one-class classifiers; namely, k-NN, OCSVM, and k-means algorithms.

Haider et al. [50] propose using different, but still inexpensive, statistical features on system call traces of the ADFA-LD dataset, with the same goal of fast performance of transforming data to features without sacrificing accuracy of detection. Four features, namely, the least/most repeated and the minimum/maximum values in a trace, are used to represent a trace to detect attacks, and three supervised learning algorithms, SVM with linear and radial basis kernels and k-NN, are used. Results show k-NN receives a 78% TPR, average(FPR, FNR) = 21%. These results increased the TPR over works of Xie & Hu, and Xie et al. [156–158], for similar false-positive metrics, but are far less accurate than the computationally expensive work of Creech et al. [24]. Although this set of work used the same dataset, it is not clear from the authors' treatment if the experiments provide a fair comparison across papers.

4.3 Hidden Markov Models (HMMs) for System Call Modeling

HMMs are a natural data model for sequential data and many other works employ HMMs for system-call-based IDSs. Warrender et al. [155] compare four methods of detection based on n-grams of system call traces: list-and-lookup of observed sequences, relative frequencies, RIPPER rule induction algorithm of Cohen [22], and HMMs. Their conclusions indicate that sufficiently accurate detection results are achievable by more computationally efficient algorithms than HMMs, and that accuracy results are more dependent on test datasets than the algorithm chosen.

Gao et al. [37] create a novel HMM-based metric that reports better IDS results than their previous "evolutionary distance (ED)" metric, while also obtaining 6% faster performance.

Hoang et al. [57] develop a hybrid detection scheme that uses both a HMM to model system call sequences and a "normal" database, which includes the frequency of each observed database short sequence. Fuzzy rules are defined to classify a newly observed sequence and take into account the sequence's probability (computed via the HMM) and frequency in the normal database.

HMM training is performed using an incremental method in conjunction with an initial parameter optimization method to reduce the high cost incurred during computation. Validation on the AFDA-LD dataset exhibit a lowering of false-positive rate of 48% while indicating greater anomaly detection than a "normal-sequence database scheme and a two-layer scheme." In addition, the HMM training time realized a 75% reduction while simultaneously decreasing the memory usage. This hybrid approach follows their earlier work [56], where first the "normal" database is used to determine frequency, and second HMM-likelihood is computed detect anomalies of only those sequences of system calls that are rare or unseen in training. Experiments on the UNM's dataset (only using sendmail program traces) prove that this approach is better in detecting anomalous behavior of programs in terms of accuracy and response time than a conventional single layer approach; however, the HMM model training is expensive. The integrated system is able to produce higher levels of anomaly signals as soon as an intrusion occurs. Known problems include storage requirements, reducing the training cost of the HMM, and determining the parameters of the model automatically.

Hu et al. [60] propose a pre-processing and training approach for HMMs that halves training time of traditional HMMs with "reasonable" accuracy, i.e., with some adverse effect to the false detection rate as tested on UNM and DARPA98 datasets. In general, the work breaks training sequences into many small sequences, and train many "small" HMMs, and finally take take a weighted average.

4.4 Other System Call IDS Works

Jha et al. [66] introduce a novel statistic-based anomaly detection algorithm for system call sequences. They observe that after Markov models (MMs) are learned from observed sequences of System calls, an observed sequence is assumed to be a mixture of the learned models and a chaotic model. Bayesian techniques are used to optimized the mixing parameter, and if it is greater than a specified threshold, an alert is raised. By using mixtures of Markov chains, their filtering approach can model mixtures of system call traces from multiple users, potentially in cases involving multiple users cooperating. Additionally, the filtering-based approach can address the masquerade-detection problem, allowing for the identification of the user that generated a given execution trace based on usage patterns. Results for many configuration parameters are given on the UNM data set. Comparing this technique to HMMs, one finds that Markov chain training is $O(m)$, with m the length of the trace, while HMM training has complexity $O(n * m^2)$, where n denotes the number of HMM states.

Ghosh et al. [39, 40] test artificial neural networks (ANNs) for misuse (supervised) and anomaly detection (unsupervised) using the DARPA99 dataset. For anomaly detection, a NN is trained using normal data and randomly generated data (for simulated attacks). ROC curves are given showing strong results, notably, a TPR of 77.3% and a FPR of 2.2%.

Han and Cho [52] introduce an IDS utilizing evolutionary neural networks (ENNs) to simultaneously calculate the NN's structure and weights. For labeled training data, ambient system call sequences are labeled normal (non-attack) and randomly generated sequences are labeled anomalous (attack) at a rate of 2-to-1. Experiments with an ENN produced a 0.0011% false-alarm rate while obtaining a 100% detection rate using the DARPA99 data set. Performance shows that training the ENNs takes about an hour; in comparison, this is about order of magnitude longer than training any single, comparably structured NN, but about an order of magnitude less than a grid search over many traditional NNs.

Tong et al. [145] propose a new hybrid IDS using Radial Basis Function (RBF) NN with Elman NNs (ElNN) for both anomaly and misuse detection. RBFNN classifies events in real time, passing output as an input into the ElNN. Positively (respectively, negatively) detected events by the RBFNN increase (respectively, decrease) a context weight in the ElNN, which improves accuracy and decreases the false-positive rate. This technique is advantageous due to its memory of prior seen sequences—it is robust to sparse occurrences of misuse or anomalies but will detect high temporal density of anomalies and misuse—and it exhibits faster training time, as compared to the Multilayer Perception (MLP) NN IDS of Ghosh et al. [39]. Evaluations with the DARPA dataset resulted in an anomaly detection accuracy of 93%, false-positive rate of 2.6%, and a misuse detection accuracy of 95.3% with a 1.4% false-positive rate. Results were compared to [39] and [69], ultimately producing higher accuracy and lower false-positive rates.

Ahmed and Masood [2] test radial basis function NNs on the UNM dataset, exhibiting accurate detection. Explicitly, they optimize $y(x) = \sum_1^N w_i \phi_{\sigma_i}(x - \mu_i)$, for a spherical radial basis function ϕ centered at μ_i with variance σ_i^2 (i.e., $\phi_{\sigma_i}(x) = \exp(-\sigma_i^{-2}\|x\|^2)$) to learn w_i, σ_i, μ_i, and they augment the training algorithm to also learn N, the number of basis functions.

Wagner and Dean [153] use static analysis to automatically derive three models of application's system call behavior. Immediate detection of a program's wrongful behavior allows for the

detection of intrusions. More generally static analysis is a large area of research that is outside the scope of this HIDS survey. See other static analysis surveys [63, 106].

Kruegel et al. [87] create anomaly detectors modeling four features of system calls. These four detectors outputs are dependent nodes in novel Bayesian network (BN), along with dependent nodes for the four detectors confidence, and a single independent node for the classification. Results show perfect detection rates with a 0.2% FPR. Training time is costly (NP-hard), but labeling is $O(N)$ with N as the number of nodes in the network.

4.5 Using System Call Arguments and Additional Data

In addition to modeling the system calls, incorporating the arguments of the calls or memory pointers has garnered IDS results.

Abad et al. [1] describe an IDS based on correlating network traffic to system calls and aiming to increase the detection rate and decrease the false-positive rate for both misuse and anomaly detection. Two approaches were taken, top-down, where attacks' behavior is analyzed to identify which logs can contain evidence of attack, and bottom-up, where multiple logs are analyzed to detect a specific attack. The bottom-up approach finds attacks through log correlation, and since logs may have millions of entries, the RIPPER data-mining tool is used for record filtration. To conduct the experiment, the authors used RIPPER, a rule mining algorithm that attempts to "predict the next system call," combined with log correlation using both System calls and network traffic. These ideas follow from work of Lee and Stolfo [93]. Results show an increased detection rate and a decreased false-positive rate.

For each system call (e.g., read, write, etc.) for each process (e.g., sendmail), Kruegel et al. [88] build models of normal arguments' string lengths, characters, and structure. Similar features found in Kruegel et al. [87] are used on system calls, not arguments. For anomaly detection, arguments with sufficiently different features are flagged, and the detector exhibits strong detection accuracy. Overhead is investigated, showing about 5 Kb of memory is required, and 18% (of a 2003 era) processor was used. Follow-up research of Mutz et al. [110] uses the same Bayesian network of Kruegel et al. [87] to combine these system call argument feature anomaly detectors into an ensemble.

Tandon and Chan [141, 142] develop an anomaly detection system based on rule learning techniques that leverage both system calls and their arguments. Results show gains over using just System calls, but at significant computational expense (an order of magnitude higher). Similarly, other works leveraging the arguments of System calls to enhance system-call-based detectors became prevalent at this time; e.g., see Bhatkar et al. [9] and Sufatrio and Yap [138].

Sekar et al. [132] use finite state automata (FSA) to model the programs' code path by combining System calls (transitions between states) with program counter information (to learn states). This is a computationally cheaper approach than the HMM and n-gram techniques, and also improves accuracy over these techniques. To create a model of the virtual path between calls, Feng et al. [32] incorporate dynamic extraction of return addresses in addition to the FSA approach, yielding additional accuracy without increased cost.

Rachidi et al. [122] consider a novel pre-processing approach of clustering the attributes (arguments, permission artifacts, domain knowledge from rules) of system calls in each process to obtain canonical system call + attribute representations. More specifically, for a given process and system call, all collections of attributes that occur with an instance of the given system call in an instance of the given process are clustered. Cluster centers then give a canonical system call + attribute representation, and the sequence of system calls constituting a process are replaced by their representative. Much of the work for this representation is in defining metrics on the

Table 5. Registry Anomaly Detection Systems

IDS Reference	Technique	Performance Cost	Memory Cost	Classes
Apap et al. '02 [7]	PAD	$O(v^2d^2)$	$O(vd^2)$	Binary
Heller et al. '03 [54]	OCSVMs	$O(dL^3)$	$O(d(L+T))$	Binary
Stolfo et al. '05 [137]	OCSVMs	$O(dL^3)$	$O(d(L+T))$	Binary
Topallar et al. '04 [146]	SOM			

Here v denotes number of unique records, d the number of features, L the number of training records, and T the number of testing records.

attributes. Finally, Naïve Bayes and Markov models are used as a supervised classifier on the DARPA99 dataset. Accuracy results are reported per classifier, per program.

4.6 System Call Mimicry Attacks

Finally, we note that system-call IDS developments are met with research designing attacks to evade such measures, with key ideas including "mimicry" attacks, where null-effect calls pad the malicious sequence of effective calls [41, 71–73, 86, 153] or malicious call sequences are sufficiently small to evade detection [139, 140].

5 WINDOWS REGISTRY IDSs

Windows Registry is the OS's key-value database containing configuration settings for all programs and hardware on that host. This database is heavily used during computer operation. All processes use the Registry, including malware that also often modify the Registry to achieve their aim [59]. Consequently, Registry monitoring has been leveraged by many researcher efforts for forensic analysis [17, 28, 101]. Below, we survey the few works that build HIDS from Registry data. See Table 5 for the anomaly detection systems using Registry data.

Initial work by Apap et al. [7] proposed the Registry Anomaly Detection (RAD) system, consisting of three components, an audit sensor to log Registry activities, a model of normal behavior, and a real-time anomaly detector. RAD extracts five raw features from Registry accesses, namely: (1) the process accessing the registry, (2) the query type requested, (3) the key used, (4) its value, and (5) the outcome (e.g., success, error) called the response. Importantly, any anomaly detection algorithm that can accommodate these sparse feature vectors is applicable. In this initial work, the probability of each feature (5 distributions), and conditional probability of pairs of features (20 distributions) are estimated following Friedman and Singer [36], and the detection system alerts if any of the 25 estimates are below a threshold. An advantage of this estimation is that models are continually updated without any user interaction.

Heller et al. [54] and a follow-up publication of Stolfo et al. [137] both test OCSVMs for the anomaly detection component of RAD with three different kernels and conclude that the probabilistic anomaly detector (PAD) of Apap et al. is much more accurate. Computational analysis is also given. PAD takes time $O(v^2d^2)$ and space $O(vd^2)$ where v denotes the number of unique records, and d denotes number of record components (dimension). The OCSVM takes time $O(dL^3)$ and space $O(d(L+T))$ where L, T denote the number of training records, and testing records, respectively. The comparison of algorithms was conducted on Pentium Celeron with 512MB RAM with memory usage of under 3 MB, and 3%–5% of CPU usage.

Topallar et al. [146] refer to the RAD system, but propose the use of Self-Organizing Maps (SOM), a NN model, as an algorithm for anomaly detection. The abstract claims their results demonstrate a low false-positive rate in comparison to other IDSs (paper is in Turkish preventing a detailed summary).

Table 6. File System Detection Systems

IDS Reference	Name	Rule Base
Kim and Spafford'94 [81]	Tripwire	Checksum
Griffin et al.'03 [43]	Disk-Based IDS	Policy
Pennington et al.'03 [117]	Storage-based IDS	Checksum and policy
Patil et al.'04 [116]	I^3FS	Checksum and policy

6 FILE SYSTEM IDSs

This section surveys work that propose or test IDSs that monitor host file-systems for detection. File systems have visibility to stored data, executables, and metadata used to service file requests. Malicious actions often involve modifying or adding new files or metadata (e.g., to allow unauthorized future access or remove evidence of previous access), leveraging file systems to monitor files, access to files, or determine legitimacy of any requests to the file system is a promising avenue for intrusion detection and prevention. Since file-system IDSs are logically separate from the OS, they are harder to disable and allow monitoring after compromise. The primary drawback of storage-based IDSs is their limited visibility. Table 6 gives the file system detection works surveyed.

First available in 1992, Tripwire,[10] from Kim and Spafford [81], is perhaps the most notable file-integrity tool. Tripwire is an open-source and now commercially available IDS for detection and remediation of malicious file and configuration changes originally designed for the UNIX system. A checklist of information about important files is created periodically and compared against previous versions to detect unexpected or unauthorized file changes. Details of the original system implementation and use are reported in the publication cited above. Notably, the system was deployed and in use before the publication.

Griffin et al. [43] implement "IDS functionality in the firmware of workstations' locally attached disks," where the majority of system files lie. The Intrusion Detection for Disks (IDD) system monitors the file system for suspicious file manipulations, such as unauthorized reads, writes, file meta-data modifications, suspicious access patterns, compromises of file integrity, or other events that may indicate an intrusion. Since this IDS is required to run on separate hardware, it is protected even if the system it is monitoring has been compromised, so long as the storage device and administrative computer are uncompromised. The system has four main design requirements: specifying access policies, securely administrating the IDD, monitoring, and responding to policy violations. The system's architecture consists of three main components: (1) the bridge process on the host computer, to connect the administrator and IDD, (2) the request de-multiplexer, to differentiate administrative requests from other requests, and (3) a policy manager on the IDD, to monitor the system for violations and generate alerts. An evaluation using a prototype disk-based IDS into a SCSI (Small Computer System Interface) disk emulator and using PostMark trans and SSH-build filesystem benchmarks indicates that it is feasible to include IDS functionality in low-cost desktop disk drives, in terms of CPU and memory costs.

Pennington et al. [117] propose an Intrusion Detection on Disk, a rule-based IDS embedded in the storage interface and monitoring the file system. The system prototype uses a set of rules to monitor important files and binary changes (following Tripwire [81]) and rules to detect patterns of changes to the file system. Testing on 16 rootkits and two worms shows that 15 are identified by the IDS, and three of the detected 15 modify the kernel to hide from other file-system integrity checkers (e.g., Tripwire). Examples of alerting activities include "modifying system binaries, adding files to system directories, scrubbing the audit log, or using suspicious file names." The overhead

[10] See www.tripwire.org.

Table 7. Program Analysis for Detection Works

IDS Reference	Data Source	Classifier	Learning	Classes
Schultz et al.'01 [131]	Binaries, DLL, calls	RIPPER, NB, MNB	Supervised	Binary
Newsome and Song'05 [111]	Binaries	Signatures	Unsupervised	Binary
Moscovitch et al.'07 [107]	Program resource utilization	DT, NB, BN, ANN	Supervised	Binary
Khan et al.'16,'17 [79, 80]	Process network utilization	AdaBoost	Supervised	Binary
Vaas and Happa'17 [148]		Binary		

of the system is investigated, and results show under 2 MB of memory is needed. The primary advantages of this storage-based IDS are its independence from the host (if the host system is compromised, extra steps are necessary to disable this IDS), and that modifications to the storage device are necessary if any malware is to persist across reboots.

Patil et al. [116] describe I^{3}FS, an In-kernel Integrity checker and Intrusion detection File System; this is an IDS based on real-time, in-kernel, on-access integrity file checking. The proposed IDS is modular and can be mounted on any file system. The main goal is to restrict access and notify administrators if an intrusion is detected. The system is compared against Tripwire [81] and can overcome its limitations—intruder tampering, large performance overhead, and inability of real-time detection. I^{3}FS uses security policies and cryptographic checksums of files computed using MD5, and stores both in four in-kernel Berkeley databases: policy, checksums, checksum metadata, and access counter databases. IDS security is implemented by adding an authentication mechanism that allows for file calls interception and by using policies and previously computed checksums to determine file integrity to allow or deny access to those files and possibly alerting system administrators. I^{3}FS is primarily designed to prevent replacement of legitimate files with files containing malicious content, unauthorized modification of data, and data corruption. The system was tested using CPU, I/O, and custom read benchmarks. Results indicate that performance overhead under normal user workload is 4%, and can be modified by setting system parameters and changing system policies.

File system monitoring is frequently used in HIDSs that leverage virtual environments. Quynh and Takefuji [121] propose monitoring a system by implementing sniffing and forwarding file system call logs (e.g., map, open, write) to a privileged VM. Ko et al. [83] design a "file-centric logger" that watches file accesses and transfers and can be implemented in cloud VMs and physical environments. A tool is provided for the end user to verify personal file tampering. Gupta et al. [46, 47] describe a lightweight and platform independent HIDS based on monitoring file system integrity while running as privileged VM. Jin et al. [68] implement VMFence, which includes file integrity monitoring, among other (network-oriented) features.

Distributed and more comprehensive IDS architectures leveraging file-integrity for detection exist as well, see Demara et al. [26]. Their work also provides a short survey of existing frameworks for file-system IDSs.

7 PROGRAM ANALYSIS AND MONITORING TECHNIQUES

This section focuses on a few works that leverage information about processes, process trees, or specific binaries on a host for detection. We note that this has significant overlap with other security sub-fields, such as dynamic malware analysis and application vulnerability analysis. A detailed survey of these related topics is out of scope for this survey. See Table 7, which itemizes the works using program analysis for detection.

Schultz et al. [131] describe a framework for automatic detection of malicious executables before they run. Different data-mining algorithms are explored to determine the best algorithm for new binaries.

Table 8. Schultz et al. [131] Results

Algorithm	Feature Type	TPR	FPR
Signatures	Bytes	33.75%	0%
RIPPER	DLLs	57.89%	9.22%
RIPPER	DLLfunction calls	71.05%	7.77%
RIPPER	DLLs with counted function calls	52.63%	5.34%
NB	Strings	97.43%	3.80%
MNB	Bytes	97.76%	6.01%

Here TPR is true-positive rate and FPR is false-positive rate.

Experiments used three data-mining algorithms; RIPPER, NB, and Multi-Naïve Bayes (MNB), and five types of features—Dynamically Loaded Libraries (DLL) used by the binary, DLL function calls made by the binary, number of unique function calls within each DLL, strings extracted from binary files, and byte sequences. To conduct the experiment, a dataset of malicious and benign executables were created from McAfee's virus scanner. Results were compared to conventional signature-based detectors and are summarized in Table 8.

Developed by Newsome and Song, TaintCheck [111] is a system that marks data as "tainted" if it comes from an untrusted source, tracks movement of data dynamically during execution of programs, and uses signatures to alert when tainted data is used inappropriately. Data originating from or influenced by any input, e.g., memory addresses and format strings that are not supplied by the code itself, i.e., are supplied by external inputs or mathematical computation, are considered tainted, and when used unsafely indicates likely vulnerable code. TaintCheck identifies tainted code, then monitors instructions that manipulate it (e.g, MOVE, LOAD, PUSH instructions), and finally identifies if the data is used in a manner that violates set policies (e.g., as input to a system call). It reliably detects most types of exploits while producing no false positives, and it permits semantic analysis using signatures.

Moscovitch et al. [107] test four machine learning techniques: DT, NB, BN, and ANN. Each has different feature subsets to detect unknown malware based on characteristics of known malware, in particular worms, using computer measurements, such as memory usage, disk usage, CPU usage, and so on. To examine worm behavior, the authors used five known worms, which all perform port scanning and other actions. A variety of configurations were created, using machines with different hardware and using different levels of activity from background tasks and user tasks. Four hypotheses were tested: the method can reach detection accuracy of known malware above 90% and detection accuracy of unknown worms above 80%; the computer configuration and background activities have no significant influence on detection; furthermore, at most 30 features are needed to attain the same accuracy as full set. All goals were achieved, with BNs consistently producing accurate results.

Khan et al. [80] introduce using fractal dimension theory [82] to analyze parent/child process counts as a feature for detecting malware. Follow-up work by Khan et al. [79] models polymorphic malware with fractal analysis of the process tree and build an anomaly detection HIDS based on a modified AdaBoost ensemble classifier, which assigns higher weights to weak classifiers and puts emphasis on misclassified samples to improve their estimation. A host sensor is utilized to collect the network profile of processes (process ID, time started, and the process's network connection information) and modules on Windows 7 OS. To conduct the experiment, authors collected 333,692 data samples for one hour and used malware detected by three out of 54 antivirus companies, according to VirusTotal.[11] To build a classifier, the dataset was partitioned into 70% and 30% as

[11]See www.virustotal.com.

training and testing sets, respectively. Results indicate that the proposed AdaBoost algorithm reduces the error 60% more than the traditional algorithm with 30 less iterations. A comparison shows an improvement in detecting true positives from 93.93% to 95.27% and a reduction of false negatives from 6.06% to 4.73%; however, detection of true negatives decreased from 100% to 97.14% and false positives increased from 0.0% to 13.7%. A similar fractal approach by many of the same authors, Siddiqui et al. [134] uses k-NN with a fractal weighting approach on network data for detection.

Vaas and Happa [148] design a client-server architecture that observes process' memory consumption. Snapshots of each process are collected over a time window containing its resource utilization and timestamp, with the goal of identifying anomalous behavior of a machine's processes on a per-application basis. The method consists of three phases: acquisition, learning, and production. A memory fingerprint is gathered during the acquisition phase. During the learning phase, a model of each application is computed from the fingerprint and the model is used to create an anomaly detector. During the production phase, the quality of the model is assessed. The model is then tested with user process data, and results indicate an ability to distinguish processes by their virtual memory fingerprints. To increase efficiency during the learning phase, to make application models available more quickly, parallel machine learning techniques are utilized.

DATASETS

Intrusion detection evaluation datasets are important resources for validation, comparison, and experimentation. Popularity of a labeled dataset among researchers allows comparison of detection metrics or performance across publications, and in many cases has stimulated a flurry of IDS research on a particular data source. Common pitfalls of such datasets are artificial artifacts correlated with targets, unrealistic attacks, and redundant or missing data, among others.

To assist researchers, we have compiled the datasets commonly used in the HIDS research literature with a brief description of their contents, and noteworthy advantages or drawbacks. Table 9 gives itemized information at a glance, and the website for each data source is at the conclusion of its description in the text.

Information Marketplace for Policy and Analysis of Cyber-Risk and Trust (IMPACT) The Department of Homeland Security maintains the IMPACT database.[12] Formerly known as PREDICT, the "Protected Repository for the Defense of Infrastructure Against Cyber Threats," IMPACT contains recent network operations data contributions from developers around the world aiming to improve cyber-risk research and development. The cyber-related dataset repository is publicly available.

Digital Corpora Computer forensics education research data including disk images, memory dumps, network packet captures, and so on, is publicly usable in this database. Additionally, Digital Corpora provides a research corpus of worldwide, real data; however, usage is limited.

DARPA Intrusion Detection 1998, 1999, and 2000 The "Cyber Systems and Technology Group" of MIT Lincoln Laboratory, working with DARPA (Defense Advanced Research Projects Agency) and AFRL (Air Force Research Laboratory), created the first public, standard corpora intended for evaluation of computer network intrusion detection systems [90].

The 1998 dataset is a widely-used collection of known attacks, and consists of system call-based audit data and network data, including full packet capture. The data is

[12] See https://www.impactcybertrust.org.

Table 9. Datasets and Public Datasets and Dataset Collections

Name/Abbreviation	Data Source	Attack Class	Website
IMPACT	NA	Various	http://www.impactcybertrust.org
Digital Corpora Database	NA	Various	http://digitalcorpora.org
DARPA'98,'99,'00	Network traffic, System calls	DOS, U2R, R2L, PROBE	http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data
KDD99	Network traffic	DOS, U2R, R2L, PROBE	https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
NSL-KDD	Network traffic	DOS, U2R, R2L, PROBE	http://www.unb.ca/cic/research/datasets/nsl.html
GURE-KDD	PCAPs	DOS, U2R, R2L, PROBE	http://aldapa.eus/res/gureKddcup
UNM	System calls	Buffer overflows, symbolic link, trojans	http://www.cs.unm.edu/~immsec/systemcalls.htm
ADFA-LD, ADFA-WD, ADFA-WD:SAA	System calls	Exfiltration, DDoS, other	https://bit.ly/2GDWrMJ
Active DNS Project	DNS PCAPs	Malware, spam, phishing, other	https://www.activednsproject.org
SecRepo	malware, NIDS, host logs, PCAPs	Various	http://www.secrepo.com
Malware Traffic Analysis	Malware, PCAPs	Malware	www.malware-traffic-analysis.net
NETRESEC	PCAP DBs list	Various	www.netresec.com/?page=PCAPFiles
CTU 13	Network flow, PCAPs	Botnet	https://goo.gl/i9WQq3
Malware Capture Facility Project	PCAPs	Botnet, Various	https://www.stratosphereips.org/datasets-malware
The Honeynet Project	Malware, PCAPs, logs	Various	http://honeynet.org/challenges
VAST Challenge 2013	Network flows, logs	DOS, FTP exfil., other	http://vacommunity.org/VAST+Challenge+2013
VAST Challenge 2012	Network logs	Botnet, scanning, exfil.	http://vacommunity.org/VAST+Challenge+2012
UNSW-NB15	PCAPs	Fuzzers, backdoors, DoS, exploits, recon, other	https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets/
CAIDA	PCAP headers, other internet data	Unlabeled	http://www.caida.org/data
Unified Host and Network Dataset	Network and host audit data	Unlabeled	https://csr.lanl.gov/data/2017.html
Comprehensive Mulit-Source Cyber-Security Events	Authentication, DNS, Processes, Network flows	unspecified red-team events	https://csr.lanl.gov/data/cyber1/
User-Computer Authentication Associations in Time	Authentication logs	None	https://csr.lanl.gov/data/auth/

comprised of `praudit`[13] and `list` files, as well as packet captures from `tcpdump`; The attacks conducted to generate this dataset were not automated, and they are considered high footprint attacks by subsequent researchers [51].

[13]See https://docs.oracle.com/cd/E19253-01/816-4557/auditref-76/index.html for description of the `praudit` command.

Numerous issues have been documented with this dataset [14, 99, 100]. Brugger and Chow [14] noted several issues with the dataset including inability to accommodate the latest attack trends, and the majority of malicious connections consisting of denial of service attacks and probing activity.

The 1999 dataset consists of a series of network packet dumps and BSM system call records. The data has been widely used in the intrusion detection and networking community, even though it is known to have a number of artifacts of its creation, including the lack of damaged or unusual background packets and uniform host distribution [99].

KDD Cup 1999 (KDD99) This dataset was created by processing the `tcpdump`[14] portions of the DARPA98 dataset. It provides labeled data for intrusion detection and contains four attack types; DoS (denial of service), U2R, R2L, and PROBE [114]. However, evaluating machine learning algorithms, such as DTs [94, 119], NNs [74], and SVMs [31], with KDD99 substantiates that it is not possible to accurately detect U2R and R2L attacks. Sabhnani and Serpen [127] investigated the KDD dataset deficiencies and concluded that for the U2R and the R2L attack categories no trainable pattern classification or machine learning algorithm can achieve an acceptable level of misuse detection performance on the KDD testing data subset if classifier models are built using the KDD training data subset. This is due to the omission of attacks and their records from the training data subset.

NSL-KDD The NSL-KDD dataset was created to improve upon the shortcomings of the KDD99 dataset. KDD99's record redundancy hinders an algorithm's ability to learn by causing a bias against infrequent records and, in turn, overlooking harmful attacks. This issue was resolve with the removal of duplicate records in both the training and testing sets. Consequently, the reduction makes it feasible to run the experiments on the full set without requiring random subset selection [143].

GureKddcup and GureKddcup6percent GureKddcup consists of the KDD99 connections with added network packet payload that allows for direct extraction by learning algorithms. The GureKDDcup dataset is generated by following the same steps as the KDD99 dataset and consists of numerous redundant entries. Bro-IDS is used for processing the `tcpdump` files to acquire connections along with their attributes. All connections are labeled with MIT's "connections-class" files [128]. The original dataset size is 9.3 GB, and the 6% dataset size is 4.2 GB [118].

University of New Mexico dataset (UNM) In 2004, the UNM dataset was released consisting of four datasets of systems calls executed by active processes; "Synthetic Sendmail UNM, Synthetic Sendmail CERT, live lpr UNM, and live lpr MIT" [30]. Several programs are included "(e.g., programs that run as daemons and those that do not), programs that vary widely in their size and complexity, and different kinds of intrusions (buffer overflows, symbolic link attacks, and Trojan programs)" [113]. The dataset consists of both "synthetic" and "live" traces, and a trace consists of a list of a unique process' system calls. The UNM dataset is as antiquated as the KDD data and focuses on individual processes rather than the entire OS [23].

ADFA IDS datasets (ADFA-LD, ADFA-WD, and ADFA-WD:SAA) Since performance on the Darpa98 and KDD99 datasets does not represent true performance against contemporary attacks, ADFA was developed as a modern benchmark for HID. The ADFA IDS labeled dataset is the successor of the KDD collection using the latest publicly available exploits and methods. There are three groups of data with raw system call traces: training,

[14] See https://danielmiessler.com/study/tcpdump/ for a tutorial on the `tcpdump` command/tool.

testing normal, and testing attack. The dataset is designed for use with an anomaly-based IDS so there are no attack traces used during training.

All training and validation data traces were gathered under normal host operations, during activities varying from browsing the web to LaTeX document generation. The ADFA dataset contains more similarities between attack data and normal data than either the Darpa98 or the KDD99 datasets. This allows for a more accurate portrayal of cyber attacks and better assessment of IDS performance [23].

Two Windows OS specific datasets were generated to protect from zero-day attacks, stealth attacks, data exfiltration, and DDoS attacks. ADFA-WD is comprised of known "Windows-based vulnerability oriented zero-day attacks" and ADFA-WD:SAA is an expansion used for resistance validation of prospective HIDS [49].

Active DNS Project Over a terabyte of "unprocessed DNS packet captures" (PCAPs) along with a plethora of daily de-duplicated DNS records [85].

Security Repo (SecRepo) The SecRepo is a compilation of security data including malware, NIDS, Modbus, and system logs. Additionally, it consists of several of the following datasets.

Malware Traffic Analysis Samples of malware binaries and PCAPs are provided along with an active campaign listing.[15]

NETRESEC Data This data provides a list of publicly accessible packet capture repositories via the Internet.

CTU 13 The data contains 13 datasets,[16] each containing a malware binary, a network flow `.csv` file from the ARGUS flow sensor,[17] and PCAP file(s) with botnet traffic. Included in every dataset is a `readme` file providing information for which IPs are infected or attacked and how [38].

Malware Capture Facility Project This dataset is an extension of the CTU 13 dataset, and consists of the similar information from around 350 attacks pertaining to malicious PCAPs.

The Honeynet Project Consists of a variety of data from all of the challenges, including PCAP, malware, and logs.

VAST Challenge 2013 Mini-Challenge 3 This is a cybersecurity challenge that includes data related to network flow, network status, and intrusion prevention systems. However, there are sizable data gaps.

VAST Challenge 2012 This challenge consists of two smaller tasks. The first involving situational awareness (e.g., metadata and intermittent status reporting) and the second involving forensics (e.g., Firewall and IDS logs).

UNSW-NB15 A comprehensive dataset for NIDS containing nine attacks types: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. The ARGUS flow sensor and Bro-IDS[18] tools are used along with the development of twelve algorithms for the generation of 49 features with the class label.

Center for Applied Internet Data Analysis (CAIDA) The CAIDA Anonymized Internet Traces 2016 annual dataset consists of anonymized traffic traces with a single trace generated quarterly. The internet traffic contains "application breakdown, security events, topological distribution, and flow volume and duration." Software capable of reading

[15] See http://www.malware-traffic-analysis.net/2018/index.html.
[16] See https://mcfp.weebly.com/the-ctu-13-dataset-a-labeled-dataset-with-botnet-normal-and-background-traffic.html.
[17] See http://qosient.com/argus.
[18] See https://www.bro.org/sphinx/broids/index.html.

Table 10. DARPA Attack Classes

Attack Class	Class Name
0	Normal
1	Probe
2	DoS
Compromises	
3	U2R
4	R2L

packet captures (PCAPs) in `tcpdump` format can read the traces. All traces are made anonymous with the same key using "CryptoPan prefix-preserving anonymization" and there is a complete packet payload removal. There is a negligible quantity of packet lost for some data [112].

Unified Host and Network Dataset The "Unified Host and Network Dataset" consists of both network and host event data gathered from Los Alamos National Laboratory (LANL) over approximately ninety days. The host event logs come from Microsoft Windows OS machines and the network event data comes from "router network flow records." Although there is overlap in the Windows OS machines use for both the network and host datasets, the network dataset also utilizes additional machines running other OSs [147].

Comprehensive, Multi-Source Cyber-Security Events This dataset contains anonymized authentication logs, flows, processes, and DNS records that are from LANL's network. A separated set of the authentication logs from a red-team event are included, although further details, such as attack methods used or exactly what red-team records are indeed a malicious attempt are not provided [76, 77].

User-Computer Authentication Associations in Time This is a real, anonymized dataset of over 7M authentication logs from LANL's network [48].

7.1 Common Attack Types in Publicly Available Datasets

The records in the DARPA- and KDD-related datasets include attack types and can be classified into one of five classes: Probe, DoS, U2R, R2L, and Normal.

Many papers included in this survey refer to the traditional attack classes by the numbering convention provided in Table 10. The table and definitions provided below can be used as a quick reference.

The last two classes are considered compromises and occur when an attacker gains privileged access to host access after hacking into the system through insecure points. Compromises are separated into two classes depending based on the source of the attack.

(1) Probing (surveillance, scanning): Attacker tries to gain information about the target host, e.g., port scanning. These attacks collect lists of potential vulnerabilities through network scans that can be utilized later in an attack against the machine or service.
(2) Denial of Service (DoS): Attacker tries to prevent legitimate users from using a service, e.g., using SYN flood. These attacks an occur on both the operation system; targeting bugs, or in the network; exploiting protocols and infrastructure limitations.
(3) User to Root (U2R): The attack is derived from within the system. An attacker who has local access to the victim machine tries to gain root access by exploiting a vulnerability, e.g., local buffer overflow attacks.

(4) Remote to Local (R2L): The attack is derived from outside the system, over the network. The attacker does not have access to any legitimate account on the victim machine; therefore, the attacker tries to gain access. This is commonly achieved through the Internet using password-guessing attacks or exploits allowing remote code execution.

8 CONCLUSION

This survey provides an overview of IDS types including various locations and types of IDSs. Related surveys are identified, and focus is given to HIDSs. To organize works and itemize the available data sources, the HIDS literature is presented per input data source. In particular, system logs, audit data, Windows Registry, file systems, and program analysis detection works are sub-categories investigated. Specific sections are allocated for system call IDS and another for algorithmic research tested on network-level data but applicable to host data. A large number of works fall into these two categories because of the publicly available labeled datasets with these types of data. We conclude with a subsection outlining limitations and budding directions for HIDS research. Additionally, this survey compiles a supplementary list of many publicly available datasets, with descriptions of their characteristics and shortcomings.

8.1 Suggested Future Directions

Although there is a wealth of IDS literature, successfully transitioned-to-practice HIDS techniques are rare, with OCSEC and Tripwire as outstanding counter examples. This is due to a number of factors that point to directions for future progress in HIDS research.

First, IDS research is constrained by limited available datasets and "in vitro" development (where test environments fail to capture the complexities of real networks). For many data sources, there are either no datasets publicly available, or those that are available are outdated, low-quality, lacking in attack diversity, or contain other serious flaws. This leads to researchers often simulating or otherwise building datasets that often lacks fidelity, complexity, or realistic benign activity. Alternatively, when real data is recorded and used, it is generally not sharable due to privacy or security concerns, and may still contain many of the limitations above (e.g. lack of attack diversity.) As evidenced by the explosion of IDS research spawned by the few well-adopted datasets (DARPA, KDD, UNM, ADFA, most notably), these facilitate quantifiable comparison of techniques across publications and provide accessible data to the hands of eager researchers, in spite of their many flaws. Up-to-date efforts to curate and publicize realistic, attack-labeled, and ideally multi-source host and network data sets will likely be met with a similarly large response from the IDS research community. Moreover, for supervised learning techniques, addressing the question of training for actual operational use is a necessity, e.g., providing a validated method for generating training data that combines a real host's/network's data with labeled attacks.

Second, HIDS research is preoccupied with (admittedly important) detection metrics at the expense of understandability of alerts. Indeed, for adoption of an HIDS, gaining the user's trust in terms adequate testing to establish an acceptable true-to-false-positive balance is a necessity. However, the myriad of publications that flex their statistical prowess and claim success upon incriminating detection rates often fail to provide actionable results to the operator. This "semantic gap" problem is perhaps first established by the famous Sommer and Paxson work [135]. Security information and event management (SIEM) tools, which correlate alerts and logs from diverse systems in real-time to enhance operators understanding, are emerging in the commercially available tools, and research providing open-source options also are developing, e.g., see Stucco.[19] Research is needed to leverage the many diverse but related data sources available to an HIDS, (not only

[19] See https://github.com/stucco.

to increase detection accuracy, but) to provide a contextual, situational awareness along with an accurate alert is needed to operationalize much of the work surveyed here.

We note that the new Unified Host and Network Data set of Turcotte et al. [147] contributes to these first two directions by providing real network and host data for researchers. Further, efforts such as Ilgun [64] provide an automated component to present the alerts to the user in a smart way.

Finally, while many researchers provide adequate investigations of the computational burden of their IDS, this is a known inhibitor of HIDS deployment. Research to dynamically change the IDS for dual optimization of increased security and decreased overhead is needed. Examples may include dynamic algorithms to adjust detector alert thresholds, change computational requirements, adjust data sources collected, or change the position or security posture of the host, based on current conditions to provide a more effective tradeoff between resources and security. Some works have begun these investigations, in particular, for cloud applications [92] and for threshold tuning [13].

Overall, we hope our treatment of the HIDS literature provides an organized panorama for researchers to gain insights, identify opportunities, and more quickly progress in advancing HIDSs.

ACKNOWLEDGMENTS

The authors thank the many reviewers who have helped polish this document, in particular, Jared M. Smith. The authors also thank Kerry Long for fruitful conversations contributing to this work's scope and direction.

REFERENCES

[1] Cristina Abad, Jed Taylor, Cigdem Sengul, William Yurcik, Yuanyuan Zhou, and Ken Rowe. 2003. Log correlation for intrusion detection: A proof of concept. In *Proceedings of the 19th Annual Computer Security Applications Conference*. IEEE, 255–264.

[2] Usman Ahmed and Asif Masood. 2009. Host-based intrusion detection using RBF neural networks. In *Proceedings of the International Conference on Emerging Technologies (ICET'09)*. IEEE, 48–51.

[3] Mamoun Alazab, Sitalakshmi Venkatraman, Paul Watters, Moutaz Alazab, and Ammar Alazab. 2012. Cybercrime: The case of obfuscated malware. In *Global Security, Safety and Sustainability, & e-Democracy*. Springer, Berlin, 204–211.

[4] M. Anandapriya and B. Lakshmanan. 2015. Anomaly-based host intrusion detection system using semantic-based system call patterns. In *Proceedings of the 9th International Conference on Intelligent Systems and Control (ISCO'15)*. IEEE, 1–4.

[5] James P. Anderson. 1972. *Computer Security Technology Planning Study. Volume 2*. Technical Report. James P. Anderson & Co., Fort Washington, PA.

[6] James P. Anderson et al. 1980. *Computer Security Threat Monitoring and Surveillance*. Technical Report. James P. Anderson & Co., Fort Washington, PA.

[7] Frank Apap, Andrew Honig, Shlomo Hershkop, Eleazar Eskin, and Sal Stolfo. 2002. Detecting malicious software by monitoring anomalous windows registry accesses. In *Recent Advances in Intrusion Detection*. Springer, Berlin, 36–53.

[8] Stefan Axelsson. 2000. *Intrusion Detection Systems: A Survey and Taxonomy*. Technical Report.

[9] Sandeep Bhatkar, Abhishek Chaturvedi, and R. Sekar. 2006. Dataflow anomaly detection. In *Proceedings of the Security and Privacy Symposium*. IEEE.

[10] Martin Botha and Rossouw Von Solms. 2003. Utilising fuzzy logic and trend analysis for effective intrusion detection. *Comput. Secur.* 22, 5 (2003), 423–434.

[11] Yacine Bouzida and Sylvain Gombault. 2003. EigenProfiles for intrusion detection, profils propres pour la detection d'intrusion. In *Département RSM GET/ENST*. Actes du Symposium SSTIC, Bretagne, France.

[12] Robert A. Bridges, Michael D. Iannacone, John R. Goodall, and Justin M. Beaver. 2018. How do information security workers use host data? A summary of interviews with security analysts. Retrieved from http://arxiv.org/abs/1812.02867.

[13] R. A. Bridges, J. D. Jamieson, and J. W. Reed. 2017. Setting the threshold for high throughput detectors: A mathematical approach for ensembles of dynamic, heterogeneous, probabilistic anomaly detectors. In *Proceedings of the*

IEEE International Conference on Big Data (Big Data'17). IEEE, 1071–1078. DOI: https://doi.org/10.1109/BigData.2017.8258031

[14] S. Terry Brugger and Jedidiah Chow. 2007. An assessment of the DARPA IDS evaluation dataset using snort. *UCDavis Dept. Comput. Sci.* 1, 2007 (2007), 22.

[15] Guy Bruneau. 2001. The history and evolution of intrusion detection. *SANS Inst.* 1, 2f (2001).

[16] Anna L. Buczak and Erhan Guven. 2016. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surveys Tutor.* 18, 2 (2016), 1153–1176.

[17] Harlan Carvey. 2005. The windows registry as a forensic resource. *Dig. Investig.* 2, 3 (2005), 201–205.

[18] Q. Chen and R. A. Bridges. 2017. Automated behavioral analysis of malware: A case study of wannacry ransomware. In *Proceedings of the 16th IEEE International Conference on Machine Learning and Applications (ICMLA'17)*. IEEE, 454–460. DOI: https://doi.org/10.1109/ICMLA.2017.0-119

[19] Zouhair Chiba, Noureddine Abghour, Khalid Moussaid, Amina El Omri, and Mohamed Rida. 2016. A survey of intrusion detection systems for cloud computing environment. In *Proceedings of the International Conference on Engineering & MIS (ICEMIS'16)*. IEEE, 1–13.

[20] W. Choi, H. J. Jo, S. Woo, J. Y. Chun, J. Park, and D. H. Lee. 2018. Identifying ECUs through inimitable characteristics of signals in controller area networks. *IEEE Trans. Vehic. Technol.* 99 (2018), 1–1. DOI: https://doi.org/10.1109/TVT.2018.2810232

[21] Shane S. Clark, Benjamin Ransford, Amir Rahmati, Shane Guineau, Jacob Sorber, Wenyuan Xu, Kevin Fu, A. Rahmati, M. Salajegheh, D. Holcomb et al. 2013. WattsUpDoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices. In *Proceedings of USENIX Workshop on Health Information Technologies*. USENIX.

[22] William W. Cohen. 1995. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*. Elsevier, 115–123.

[23] Gideon Creech and Jiankun Hu. 2013. Generation of a new IDS test dataset: Time to retire the KDD collection. In *Proceedings of the Wireless Communications and Networking Conference (WCNC'13)*. IEEE, 4487–4492.

[24] Gideon Creech and Jiankun Hu. 2014. A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns. *IEEE Trans. Comput.* 63, 4 (2014), 807–819.

[25] Joel A. Dawson, J. Todd McDonald, Jordan Shropshire, Todd R. Andel, Patrick Luckett, and Lee Hively. 2018. Rootkit detection through phase-space analysis of power voltage measurements. In *Proceedings of the 12th IEEE International Conference on Malicious and Unwanted Software (MALCON'17)*. IEEE.

[26] Ronald F. DeMara and Adam J. Rocke. 2004. Mitigation of network tampering using dynamic dispatch of mobile agents. *Comput. Secur.* 23, 1 (2004), 31–42.

[27] Dorothy Denning and Peter G. Neumann. 1985. *Requirements and Model for IDES-a Real-time Intrusion-detection Expert System*. SRI International, Menlo Park, CA.

[28] Brendan Dolan-Gavitt. 2008. Forensic analysis of the windows registry in memory. *Dig. Investig.* 5 (2008), S26–S32.

[29] George W. Dunlap, Samuel T. King, Sukru Cinar, Murtaza A. Basrai, and Peter M. Chen. 2002. ReVirt: Enabling intrusion analysis through virtual-machine logging and replay. *ACM SIGOPS Operat. Syst. Rev.* 36 (2002), 211–224.

[30] Mohammed Taha Elgraini, Nasser Assem, and Tajjeeddine Rachidi. 2012. Host intrusion detection for long stealthy system call sequences. In *Proceedings of the Colloquium on Information Science and Technology (CIST'12)*. IEEE, 96–100.

[31] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. 2002. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. *Appl. Data Min. Comput. Secur.* 6 (2002), 77–102.

[32] Henry Hanping Feng, Oleg M. Kolesnikov, Prahlad Fogla, Wenke Lee, and Weibo Gong. 2003. Anomaly detection using call stack information. In *Proceedings of the Symposium on Security and Privacy*. IEEE, 62–75.

[33] Erik M. Ferragut, Jason Laska, and Robert A. Bridges. 2012. A new, principled approach to anomaly detection. In *Proceedings of the 11th International Conference on Machine Learning and Applications (ICMLA'12)*, Vol. 2. IEEE, 210–215.

[34] Stephanie Forrest, Steven Hofmeyr, and Anil Somayaji. 2008. The evolution of system-call monitoring. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC'08)*. IEEE, 418–430.

[35] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. 1996. A sense of self for unix processes. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 120–128.

[36] Nir Friedman and Yoram Singer. 1999. Efficient Bayesian parameter estimation in large discrete domains. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 417–423.

[37] Debin Gao, Michael K. Reiter, and Dawn Song. 2006. Behavioral distance measurement using hidden Markov models. In *Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID'06)*. Springer, Berlin, 19–40.

[38] Sebastian Garcia, Martin Grill, Jan Stiborek, and Alejandro Zunino. 2014. An empirical comparison of botnet detection methods. *Comput. Secur.* 45 (2014), 100–123.

[39] Anup K. Ghosh and Aaron Schwartzbard. 1999. A study in using neural networks for anomaly and misuse detection. In *Proceedings of the USENIX Security Symposium*, Vol. 99. USENIX, 12.

[40] Anup K. Ghosh, Aaron Schwartzbard, and Michael Schatz. 1999. Learning program behavior profiles for intrusion detection. In *Proceedings of the Workshop on Intrusion Detection and Network Monitoring*, Vol. 51462. USENIX, 1–13.

[41] Jonathon T. Giffin, Somesh Jha, and Barton P. Miller. 2006. Automated discovery of mimicry attacks. In *Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID'06)*, Vol. 4219. Springer, Amsterdam, The Netherlands, 41–60.

[42] Carlos R. Aguayo González and Jeffrey H. Reed. 2011. Power fingerprinting in SDR integrity assessment for security and regulatory compliance. *Analog Integr. Circ. Signal Process.* 69, 2–3 (2011), 307–327.

[43] John L. Griffin, Adam Pennington, John S. Bucy, Deepa Choundappan, Nithya Muralidharan, and Gregory R. Ganger. 2003. *On the Feasibility of Intrusion Detection Inside Workstation Disks.* Technical Report. School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.

[44] Donghai Guan, Kejun Wang, Xiufen Ye, and Weixing Feng. 2005. A collaborative intrusion detection system using log server and neural networks. In *Proceedings of the IEEE International Conference on Mechatronics and Automation*, Vol. 2. IEEE, 874–877.

[45] Sanchika Gupta and Padam Kumar. 2015. An immediate system call sequence-based approach for detecting malicious program executions in cloud environment. *Wireless Personal Commun.* 81, 1 (2015), 405–425.

[46] Sanchika Gupta, Padam Kumar, Anjali Sardana, and Ajith Abraham. 2012. A secure and lightweight approach for critical data security in cloud. In *Proceedings of the 4th International Conference on Computational Aspects of Social Networks (CASoN'12)*. IEEE, 315–320.

[47] Sanchika Gupta, Anjali Sardana, and Padam Kumar. 2012. A light weight centralized file monitoring approach for securing files in cloud environment. In *Proceedings of the International Conference on Internet Technology and Secured Transactions*. IEEE, 382–387.

[48] Aric Hagberg, Alex Kent, Nathan Lemons, and Joshua Neil. 2014. Credential hopping in authentication graphs. In *Proceedings of the International Conference on Signal-Image Technology Internet-Based Systems (SITIS'14)*. IEEE Computer Society.

[49] Waqas Haider, Gideon Creech, Yi Xie, and Jiankun Hu. 2016. Windows-based data sets for evaluation of robustness of host-based intrusion detection systems (IDS) to zero-day and stealth attacks. *Future Internet* 8, 3 (2016), 29.

[50] Waqas Haider, Jiankun Hu, and Miao Xie. 2015. Towards reliable data feature retrieval and decision engine in host-based anomaly detection systems. In *Proceedings of the IEEE 10th Conference on Industrial Electronics and Applications (ICIEA'15)*. IEEE, 513–517.

[51] Waqas Haider, Jiankun Hu, Xinghuo Yu, and Yi Xie. 2015. Integer data zero-watermark assisted system calls abstraction and normalization for host-based anomaly detection systems. In *Proceedings of the IEEE 2nd International Conference on Cyber Security and Cloud Computing (CSCloud'15)*. IEEE, 349–355.

[52] Sang-Jun Han and Sung-Bae Cho. 2005. Evolutionary neural networks for anomaly detection based on the behavior of a program. *IEEE Trans. Syst. Man Cybernet. Part B (Cybernet.)* 36, 3 (2005), 559–570.

[53] Christopher R. Harshaw, Robert A. Bridges, Michael D. Iannacone, Joel W. Reed, and John R. Goodall. 2016. Graphprints: Towards a graph analytic method for network anomaly detection. In *Proceedings of the 11th Annual Cyber and Information Security Research Conference*. ACM, New York, NY, 1–15.

[54] Katherine A. Heller, Krysta M. Svore, Angelos D. Keromytis, and Salvatore J. Stolfo. 2003. One class support vector machines for detecting anomalous windows registry accesses. In *Proceedings of the Workshop on Data Mining for Computer Security*. 9.

[55] Paul Helman and Jessie Bhangoo. 1997. A statistically based system for prioritizing information exploration under uncertainty. *IEEE Trans. Syst. Man. Cybernet.-Part A: Syst. Hum.* 27, 4 (1997), 449–466.

[56] Xuan Dau Hoang, Jiankun Hu, and Peter Bertok. 2003. A multi-layer model for anomaly intrusion detection using program sequences of system calls. In *Proceedings of the 11th IEEE International Conference*. Citeseer, IEEE.

[57] Xuan Dau Hoang, Jiankun Hu, and Peter Bertok. 2009. A program-based anomaly intrusion detection scheme using multiple detection engines and fuzzy inference. *J. Netw. Comput. Appl.* 32, 6 (2009), 1219–1228.

[58] Steven A. Hofmeyr, Stephanie Forrest, and Anil Somayaji. 1998. Intrusion detection using sequences of system calls. *J. Comput. Secur.* 6, 3 (1998), 151–180.

[59] Greg Hoglund and James Butler. 2006. *Rootkits: Subverting the Windows Kernel*. Addison-Wesley Professional, Indianapolis, IN.

[60] Jiankun Hu, Xinghuo Yu, Dong Qiu, and Hsiao-Hwa Chen. 2009. A simple and efficient hidden Markov model scheme for host-based anomaly intrusion detection. *IEEE Netw.* 23, 1 (2009), 42–47.

[61] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. 2004. Extreme learning machine: A new learning scheme of feedforward neural networks. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, Vol. 2. IEEE, 985–990.

[62] Raid Khalid Hussein, Ahmed Alenezi, Gary B. Wills, and Robert J. Walters. 2016. A framework to secure the virtual machine image in cloud computing. In *Proceedings of the International Conference on Smart Cloud (SmartCloud'16)*. IEEE, 35–40.

[63] Nwokedi Idika and Aditya P. Mathur. 2007. A survey of malware detection techniques. *Purdue Univ.* 48 (2007).

[64] Koral Ilgun. 1993. USTAT: A real-time intrusion detection system for UNIX. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*. IEEE, 16–28.

[65] Brian Jewell and Justin Beaver. 2011. Host-based data exfiltration detection via system call sequences. In *Proceedings of the 6th International Conference on Information Warfare and Secuirty (ICIW'11)*. Academic Conferences Limited, Academic Conferences Limited, England, 134.

[66] S. Jha, L. Kruger, T. Kurtx, Y. Lee, and A. Smith. 2004. A filtering approach to anomaly and masquerade detection. Technical report, Department of Computer Science, University of Wisconsin, Madison.

[67] Jarilyn M. Hernández Jiménez, Jeffrey A. Nichols, Katerina Goseva-Popstojanova, Stacy Prowell, and Robert A. Bridges. 2017. Malware Detection on General-Purpose Computers Using Power Consumption Monitoring: A Proof of Concept and Case Study. *arXiv preprint arXiv:1705.01977*.

[68] Hai Jin, Guofu Xiang, Deqing Zou, Song Wu, Feng Zhao, Min Li, and Weide Zheng. 2013. A VMM-based intrusion prevention system in cloud computing environment. *J. Supercomput.* 66, 3 (2013), 1133–1151.

[69] Chaivat Jirapummin, Naruemon Wattanapongsakorn, and Prasert Kanthamanon. 2002. Hybrid neural networks for intrusion detection system. *Proc. ITC-CSCC* 7 (2002), 928–931.

[70] Peyman Kabiri and Ali A. Ghorbani. 2005. Research on intrusion detection and response: A survey. *IJ Netw. Secur.* 1, 2 (2005), 84–102.

[71] Hilmi Güneş Kayacik, Malcolm Heywood, and Nur Zincir-Heywood. 2006. On evolving buffer overflow attacks using genetic programming. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. ACM, New York, NY, 1667–1674.

[72] H. Gunes Kayacik and A. Nur Zincir-Heywood. 2007. On the contribution of preamble to information hiding in mimicry attacks. In *Proceedings of the 21st International Advanced Information Networking and Applications Workshops (AINAW'07)*, Vol. 1. IEEE, 632–638.

[73] Hilmi Günes Kayacik and A. Nur Zincir-Heywood. 2008. Mimicry attacks demystified: What can attackers do to evade detection? In *Proceedings of the 6th Annual Conference on Privacy, Security and Trust (PST'08)*. IEEE, 213–223.

[74] H. Gunes Kayacik, A. Nur Zincir-Heywood, and Malcolm I. Heywood. 2003. On the capability of an SOM-based intrusion detection system. In *Proceedings of the International Joint Conference on Neural Networks*, Vol. 3. IEEE, 1808–1813.

[75] Richard A. Kemmerer and Giovanni Vigna. 2002. Intrusion detection: A brief history and overview. *Computer* 35, 4 (2002), 27–30.

[76] Alexander D. Kent. 2015. *Comprehensive, Multi-Source Cyber-Security Events*. Los Alamos National Laboratory. DOI: https://doi.org/10.17021/1179829

[77] Alexander D. Kent. 2015. Cybersecurity data sources for dynamic network research. In *Dynamic Networks in Cybersecurity*. Imperial College Press.

[78] Minhaj Ahmad Khan. 2016. A survey of security issues for cloud computing. *J. Netw. Comput. Appl.* 71 (2016), 11–29.

[79] Muhammad Salman Khan, Sana Siddiqui, and Ken Ferens. 2017. Cognitive modeling of polymorphic malware using fractal-based semantic characterization. In *Proceedings of the IEEE International Symposium on Technologies for Homeland Security (HST'17)*. IEEE, 1–7.

[80] Muhammad Salman Khan, Sana Siddiqui, Robert D. McLeod, Ken Ferens, and Witold Kinsner. 2016. Fractal-based adaptive boosting algorithm for cognitive detection of computer malware. In *Proceedings of the IEEE 15th International Conference on Cognitive Informatics & Cognitive Computing (ICCI'16)*. IEEE, 50–59.

[81] Gene H. Kim and Eugene H. Spafford. 1994. The design and implementation of tripwire: A file system integrity checker. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*. ACM, New York, NY, 18–29.

[82] Witold Kinsner. 2005. A unified approach to fractal dimensions. In *Proceedings of the 4th IEEE Conference on Cognitive Informatics (ICCI'05)*. IEEE, 58–72.

[83] Ryan K. L. Ko, Peter Jagadpramana, and Bu Sung Lee. 2011. Flogger: A file-centric logger for monitoring file access and transfers within cloud computing environments. In *Proceedings of the IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom'11)*. IEEE, 765–771.

[84] Andrew P. Kosoresow and S. A. Hofmeyer. 1997. Intrusion detection via system call traces. *IEEE Softw.* 14, 5 (1997), 35–42.

[85] Athanasios Kountouras, Panagiotis Kintis, Chaz Lever, Yizheng Chen, Yacin Nadji, David Dagon, Manos Antonakakis, and Rodney Joffe. 2016. *Enabling Network Security Through Active DNS Datasets*. Springer International Publishing, Cham, 188–208. DOI: https://doi.org/10.1007/978-3-319-45719-2_9

[86] Christopher Kruegel, Engin Kirda, Darren Mutz, William Robertson, and Giovanni Vigna. 2005. Automating mimicry attacks using static binary analysis. In *Proceedings of the 14th Conference on USENIX Security Symposium*. USENIX, 11–11.

[87] Christopher Kruegel, Darren Mutz, William Robertson, and Fredrik Valeur. 2003. Bayesian event classification for intrusion detection. In *Proceedings of the 19th Annual Computer Security Applications Conference*. IEEE, 14–23.

[88] Christopher Kruegel, Darren Mutz, Fredrik Valeur, and Giovanni Vigna. 2003. On the detection of anomalous system call arguments. In *Proceedings of the European Symposium on Research in Computer Security*. Springer, Berlin, 326–343.

[89] Uttam Kumar and Bhavesh N. Gohil. 2015. A survey on intrusion detection systems for cloud computing environment. *Int. J. Comput. Appl.* 109, 1 (2015), 6–15.

[90] MIT Lincoln Labs. 2017. DARPA Intrusion Detection Evaluation. Retrieved from http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/.

[91] Aleksandar Lazarevic, Vipin Kumar, and Jaideep Srivastava. 2005. Intrusion detection: A survey. In *Managing Cyber Threats*. Springer, Amsterdam, The Netherlands, 19–78.

[92] Jun-Ho Lee, Min-Woo Park, Jung-Ho Eom, and Tai-Myoung Chung. 2011. Multi-level intrusion detection system and log management in cloud computing. In *Proceedings of the 13th International Conference on Advanced Communication Technology (ICACT'11)*. IEEE, 552–555.

[93] Wenke Lee, Salvatore J. Stolfo, and Kui W. Mok. 1999. A data-mining framework for building intrusion detection models. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 120–132.

[94] Itzhak Levin. 2000. KDD-99 classifier learning contest: LLSoft's results overview. *SIGKDD Explor.* 1, 2 (2000), 67–75.

[95] Ling Li and Constantine N. Manikopoulos. 2004. Windows NT one-class masquerade detection. In *Proceedings of the 5th Annual IEEE SMC Information Assurance Workshop*. IEEE, 82–87.

[96] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. 2013. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* 36, 1 (2013), 16–24.

[97] Yihua Liao and V Rao Vemuri. 2002. Use of k-nearest neighbor classifier for intrusion detection. *Comput. Secur.* 21, 5 (2002), 439–448.

[98] Nick Littlestone. 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Mach. Learn.* 2, 4 (1988), 285–318.

[99] Matthew Mahoney and Philip Chan. 2003. An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. In *Recent Advances in Intrusion Detection*. Springer, Amsterdam, The Netherlands, 220–237.

[100] John McHugh. 2000. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Trans. Info. Syst. Secur.* 3, 4 (2000), 262–294.

[101] Vivienne Mee, Theodore Tryfonas, and Iain Sutherland. 2006. The windows registry as a forensic artefact: Illustrating evidence collection for Internet usage. *Dig. Investig.* 3, 3 (2006), 166–173.

[102] Yasir Mehmood, Umme Habiba, Muhammad Awais Shibli, and Rahat Masood. 2013. Intrusion detection system in cloud computing: Challenges and opportunities. In *Proceedings of the 2nd National Conference on Information Assurance (NCIA'13)*. IEEE, 59–66.

[103] Shagufta Mehnaz and Elisa Bertino. 2017. Ghostbuster: A fine-grained approach for anomaly detection in file system accesses. In *Proceedings of the 7th ACM on Conference on Data and Application Security and Privacy (CODASPY'17)*. ACM, New York, NY, 3–14. DOI: https://doi.org/10.1145/3029806.3029809

[104] Preeti Mishra, Emmanuel S. Pilli, Vijay Varadharajan, and Udaya Tupakula. 2017. Intrusion detection techniques in cloud environment: A survey. *J. Netw. Comput. Appl.* 77 (2017), 18–47.

[105] Chirag Modi, Dhiren Patel, Bhavesh Borisaniya, Hiren Patel, Avi Patel, and Muttukrishnan Rajarajan. 2013. A survey of intrusion detection techniques in cloud. *J. Netw. Comput. Appl.* 36, 1 (2013), 42–57.

[106] Andreas Moser, Christopher Kruegel, and Engin Kirda. 2007. Limits of static analysis for malware detection. In *Proceedings of the 23rd Annual Computer Security Applications Conference (ACSAC'07)*. IEEE, 421–430.

[107] Robert Moskovitch, Shay Pluderman, Ido Gus, Dima Stopel, Clint Feher, Yisrael Parmet, Yuval Shahar, and Yuval Elovici. 2007. Host-based intrusion detection using machine learning. In *Proceedings of the Conference on Intelligence and Security Informatics*. IEEE, 107–114.

[108] Tarik Mouttaqi, Tajjeeddine Rachidi, and Nasser Assem. 2017. Re-evaluation of combined Markov-Bayes models for host intrusion detection on the ADFA dataset. In *Proceedings of the Intelligent Systems Conference (IntelliSys'17)*. IEEE, 1044–1052.

[109] Srinivas Mukkamala, Guadalupe Janoski, and Andrew Sung. 2002. Intrusion detection using neural networks and support vector machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'02)*, Vol. 2. IEEE, 1702–1707.

[110] Darren Mutz, Fredrik Valeur, Giovanni Vigna, and Christopher Kruegel. 2006. Anomalous system call detection. *ACM Trans. Info. Syst. Secur.* 9, 1 (2006), 61–93.

[111] James Newsome and Dawn Song. 2005. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In *Proceedings of the 12th Network and Distributed Systems Security Symposium*. Internet Society, 43.

[112] University of California's San Diego Supercomputer Center. 2018. Center for Applied Internet Data Analysis. Retrieved from http://www.caida.org/.

[113] The Regents of the University of New Mexico. 2006. Sequence-based intrusion detection. Retrieved from http://www.cs.unm.edu/ immsec/systemcalls.htm.

[114] Vinod K. Pachghare, Vaibhav K. Khatavkar, and Parag Kulkarni. 2012. Pattern-based IDS using supervised, semi-supervised and unsupervised approaches. In *Proceedings of the International Conference on Computer Science and Information Technology*. Springer, Berlin, 542–551.

[115] Animesh Patcha and Jung-Min Park. 2007. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Netw.* 51, 12 (2007), 3448–3470.

[116] Swapnil Patil, Anand Kashyap, Gopalan Sivathanu, and Erez Zadok. 2004. I3FS: An In-kernel integrity checker and intrusion detection file system. In *Proceedings of the Large Installation System Administration Conference (LISA'04)*, Vol. 4. USENIX, 67–78.

[117] Adam G. Pennington, John D. Strunk, John Linwood Griffin, Craig A. N. Soules, Garth R. Goodson, and Gregory R. Ganger. 2003. Storage-based intrusion detection: Watching storage activity for suspicious behavior. In *Proceedings of the USENIX Security Symposium*. USENIX.

[118] I. Perona, I. Gurrutxaga, O. Arbelaitz, J. I. Martín, J. Muguerza, and J. M. Pérez. 2008. gureKddcup database. Retrieved from http://aldapa.eus/res/gureKddcup/.

[119] Bernhard Pfahringer. 2000. Winning the KDD99 classification cup: Bagged boosting. *ACM SIGKDD Explor. Newslett.* 1, 2 (2000), 65–66.

[120] Phillip A. Porras and Richard A. Kemmerer. 1992. Penetration state transition analysis: A rule-based intrusion detection approach. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC'92)*. IEEE, 220–229.

[121] Nguyen Anh Quynh and Yoshiyasu Takefuji. 2007. A novel approach for a file-system integrity monitor tool of Xen virtual machine. In *Proceedings of the 2nd ACM Symposium on Information, Computer, and Communications Security*. ACM, New York, NY, 194–202.

[122] Tajjeeddine Rachidi, Oualid Koucham, and Nasser Assem. 2016. Combined data and execution flow host intrusion detection using machine learning. In *Intelligent Systems and Applications*. Springer, 427–450.

[123] Wei Ren and Hai Jin. 2005. Distributed agent-based real time network intrusion forensics system architecture design. In *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05)*, Vol. 1. IEEE, 177–182.

[124] John R. Reuning. 2004. *Applying Term Weight Techniques to Event Log Analysis for Intrusion Detection*. Master's Thesis, School of Information and Library Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, 1–60.

[125] Jake Ryan, Meng-Jang Lin, and Risto Miikkulainen. 1998. Intrusion detection with neural networks. In *Advances in Neural Information Processing Systems*. MIT Press, 943–949.

[126] Farzad Sabahi and Ali Movaghar. 2008. Intrusion detection: A survey. In *Proceedings of the 3rd International Conference on Systems and Networks Communications (ICSNC'08)*. IEEE, 23–26.

[127] Maheshkumar Sabhnani and Gursel Serpen. 2004. Why machine learning algorithms fail in misuse detection on KDD intrusion detection data set. *Intell. Data Anal.* 8, 4 (2004), 403–415.

[128] Santosh Kumar Sahu, Sauravranjan Sarangi, and Sanjaya Kumar Jena. 2014. A detail analysis on intrusion detection datasets. In *Proceedings of the IEEE International Advance Computing Conference (IACC'14)*. IEEE, 1348–1353.

[129] H. Sayadi, N. Patel, S. M. P D, A. Sasan, S. Rafatirad, and H. Homayoun. 2018. Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification. In *Proceedings of the Design Automation Conference (DAC'18)*. ACM/ESDA/IEEE, 1–6.

[130] Karen Scarfone and Peter Mell. 2007. Guide to intrusion detection and prevention systems (idps). *NIST Spec. Publ.* 800, 2007 (2007), 94.

[131] Matthew G. Schultz, Eleazar Eskin, F. Zadok, and Salvatore J. Stolfo. 2001. Data-mining methods for detection of new malicious executables. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'01)*. IEEE, 38–49.

[132] R. Sekar, Mugdha Bendre, Dinakar Dhurjati, and Pradeep Bollineni. 2001. A fast automaton-based method for detecting anomalous program behaviors. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'01)*. IEEE, 144–155.

[133] Jude Shavlik and Mark Shavlik. 2004. Selection, combination, and evaluation of effective software sensors for detecting abnormal computer usage. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 276–285.

[134] Sana Siddiqui, Muhammad Salman Khan, Ken Ferens, and Witold Kinsner. 2016. Detecting advanced persistent threats using fractal dimension-based machine learning classification. In *Proceedings of the ACM on International Workshop on Security And Privacy Analytics (IWSPA'16)*. ACM, New York, NY, 64–69. DOI: https://doi.org/10.1145/2875475.2875484

[135] Robin Sommer and Vern Paxson. 2010. Outside the closed world: On using machine learning for network intrusion detection. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'10)*. IEEE, 305–316.

[136] Aditya K. Sood, Rohit Bansal, and Richard J. Enbody. 2013. Cybercrime: Dissecting the state of underground enterprise. *IEEE Internet Comput.* 17, 1 (2013), 60–68.

[137] Salvatore J. Stolfo, Frank Apap, Eleazar Eskin, Katherine Heller, Shlomo Hershkop, Andrew Honig, and Krysta Svore. 2005. A comparative evaluation of two algorithms for windows registry anomaly detection. *J. Comput. Secur.* 13, 4 (2005), 659–693.

[138] Sufatrio and Roland H. C. Yap. 2005. Improving host-based IDS with argument abstraction to prevent mimicry attacks. In *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*. Springer, Berlin, 146–164.

[139] Kymie Tan, Kevin Killourhy, and Roy Maxion. 2002. Undermining an anomaly-based intrusion detection system using common exploits. In *Recent Advances in Intrusion Detection*. Springer, Berlin, 54–73.

[140] Kymie M. C. Tan and Roy A. Maxion. 2002. "Why 6?" defining the operational limits of stide, an anomaly-based intrusion detector. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 188–201.

[141] Gaurav Tandon and Philip K. Chan. 2005. Learning useful system call attributes for anomaly detection. In *Proceedings of the Florida Artificial Intelligence Research Society Conference (FLAIRS'05)*. AAAI, 405–411.

[142] Gaurav Tandon and Philip K. Chan. 2006. On the learning of system call attributes for host-based anomaly detection. *Int. J. Artific. Intell. Tools* 15, 06 (2006), 875–892.

[143] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA'09)*. IEEE, 1–6.

[144] Taha Ait Tchakoucht, Mostafa Ezziyyani, Mohammed Jbilou, and Mikael Salaun. 2015. Behavioral approach for intrusion detection. In *Proceedings of the IEEE/ACS 12th International Conference on Computer Systems and Applications (AICCSA'15)*. IEEE, 1–5.

[145] Xiaojun Tong, Zhu Wang, and Haining Yu. 2009. A research using hybrid RBF/Elman neural networks for intrusion detection system secure model. *Comput. Phys. Commun.* 180, 10 (2009), 1795–1801.

[146] M. Topallar, M. O. Depren, E. Anarim, and K. Ciliz. 2004. Host-based intrusion detection by monitoring Windows registry accesses. In *Proceedings of the IEEE 12th Signal Processing and Communications Applications Conference*. IEEE, 728–731.

[147] M. J. M. Turcotte, A. D. Kent, and C. Hash. 2017. Unified host and network data set. *ArXiv e-prints* abs/1708.07518.

[148] Christian Vaas and Jassim Happa. 2017. Detecting disguised processes using application-behavior profiling. In *Proceedings of the IEEE International Symposium on Technologies for Homeland Security (HST'17)*. IEEE, 1–6.

[149] Kalyan Veeramachaneni, Ignacio Arnaldo, Vamsi Korrapati, Constantinos Bassias, and Ke Li. 2016. AI^2: Training a big data machine to defend. In *Proceedings of the IEEE International Conference on High Performance and Smart Computing (HPSC'16), and IEEE International Conference on Intelligent Data and Security (IDS'16), IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity'16)*. IEEE, New York, NY, 49–54.

[150] M. E. Verma and R. A. Bridges. 2018. Defining a metric space of host logs and operational use cases. In *Proceedings of the IEEE International Conference on Big Data (Big Data'18)*. 5068–5077. DOI: https://doi.org/10.1109/BigData.2018.8622083

[151] L. Vokorokos and A. Baláž. 2010. Host-based intrusion detection system. In *Proceedings of the 14th International Conference on Intelligent Engineering Systems (INES'10)*. IEEE, 43–47.

[152] Liberios Vokorokos, Anton Balaz, and Martin Chovanec. 2006. Intrusion detection system using self organizing map. *Acta Electrotech. Informat.* 6, 1 (2006), 1–6.

[153] David Wagner and R. Dean. 2001. Intrusion detection via static analysis. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'01)*. IEEE, 156–168.

[154] Zhijian Wang and Yanqin Zhu. 2017. A centralized HIDS framework for private cloud. In *Proceedings of the 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD'17)*. IEEE, 115–120.

[155] Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. 1999. Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*. IEEE, 133–145.

[156] Miao Xie and Jiankun Hu. 2013. Evaluating host-based anomaly detection systems: A preliminary analysis of ADFA-LD. In *Proceedings of the 6th International Congress on Image and Signal Processing (CISP'13)*, Vol. 3. IEEE, 1711–1716.

[157] Miao Xie, Jiankun Hu, and Jill Slay. 2014. Evaluating host-based anomaly detection systems: Application of the one-class svm algorithm to ADFA-LD. In *Proceedings of the 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'14)*. IEEE, 978–982.

[158] Miao Xie, Jiankun Hu, Xinghuo Yu, and Elizabeth Chang. 2014. Evaluating host-based anomaly detection systems: Application of the frequency-based algorithms to ADFA-LD. In *Proceedings of the International Conference on Network and System Security*. Springer, Berlin, 542–549.

[159] Xiaolong Xu, Guangpei Liu, and Jie Zhu. 2016. Cloud data security and integrity protection model based on distributed virtual machine agents. In *Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC'16)*. IEEE, Chengdu, China, 6–13.

[160] Nong Ye, Syed Masum Emran, Qiang Chen, and Sean Vilbert. 2002. Multivariate statistical analysis of audit trails for host-based intrusion detection. *IEEE Trans. Comput.* 51, 7 (2002), 810–820.

[161] Nong Ye, Xiangyang Li, Qiang Chen, Syed Masum Emran, and Mingming Xu. 2001. Probabilistic techniques for intrusion detection based on computer audit data. *IEEE Trans. Syst. Man Cybernet. Part A: Syst. Hum.* 31, 4 (2001), 266–274.

[162] Qing Ye, Xiaoping Wu, and Bo Yan. 2010. An intrusion detection approach based on system call sequences and rules extraction. In *Proceedings of the 2nd International Conference on e-Business and Information System Security (EBISS'10)*. IEEE, Wuhan, China, 1–4.

[163] Tianwei Zhang and Ruby B. Lee. 2015. Cloudmonatt: An architecture for security health monitoring and attestation of virtual machines in cloud computing. In *Proceedings of the ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA'15)*. IEEE, 362–374.

[164] Zonghua Zhang and Hong Shen. 2005. Application of online-training SVMs for real-time intrusion detection with different considerations. *Comput. Commun.* 28, 12 (2005), 1428–1442.

[165] Gu Zhaojun and Wang Chao. 2010. Statistic and analysis for Host-based syslog. In *Proceedings of the 2nd International Workshop on Education Technology and Computer Science (ETCS'10)*, Vol. 2. IEEE, 277–280.

Received May 2018; revised May 2019; accepted July 2019