A Survey of Intrusion Detection Systems Leveraging Host Data

Glass-Vanderlan, Tarrah R.
glassvandetr@ornl.gov

Iannacone, Michael D.
iannaconemd@ornl.gov

Vincent, Maria S.
vincentms@ornl.gov

Chen, Qian (Guenevere)
geunevereqian@utsa.edu

Bridges, Robert A.
bridgesra@ornl.gov

May 18, 2018

Abstract

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. Similarly, a section surveying algorithmic developments that are applicable to HIDS but tested on network data sets is included, as this is a large and growing area of applicable literature. To accommodate current researchers, a supplementary 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.

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 [7]. 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 [8] focused on automating IDS by isolating abnormal behavior in system's audit data, and work of Denning and Neumann [34] 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. [17, 85]

Increasingly over the last 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 cybercrime economy and nation-state ambitions [3, 151]. 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 [145]. 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. [20, 15] 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 3-8 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. On the other hand, 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 [42, 61]. 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

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

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 [150].

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 [166]. 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. Supervised learners can often classify anomalies as attacks.

Feature selection is influential on both accuracy and performance of IDSs 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 and 8), 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.

1.2 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 [119], location is a critical architectural decision. This can be viewed most coarsly 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.

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. [93].

Finally, we note that side-channel detection—detectors leveraging physical characteristics such as power consumption, electromagnetic radiation, vibrations, timings—has gained traction in cyber-physical IDS research [51, 24, 25, 75] but is a growing area of research for traditional computers using host-level (albeit physical) data [29, 75]. 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 & 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. Algorithmic research that is developed for NIDS, but portable to host data sources, is isolated and included. 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, 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. A large body of research focuses more on algorithmic development for detection than on specific data source

applications; in particular, many of these works test the proposed methods on KDD- and DARPA-related network datasets, but the algorithms are applicable to host-based data sets. Many of these works are captured in Section 8. Our final section gives conclusions. To assist current research, the publicly available datasets and databases referenced in the literature for IDS validation are collected in Supplemental Materials.

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, and (2) for a given data source of interest, elicit progressions in the literature and identify gaps, trends, or novel directions for future contributions.

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.

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 [10]	HIDS, NIDS	NA	NA	Discussed	Discussed
Patcha & Park '07 [130]	HIDS, NIDS	NA	Discussed	Discussed	Discussed
Kabiri & Ghorbani '05 [78]	NIDS	NA	NA	NA	Discussed
Lazarevic et al. '05 [100]	HIDS, NIDS, DIDS	NA	NA	Discussed	NA
Sabahi et al. '08 [142]	HIDS, NIDS	Discussed	Discussed	Discussed	NA
Mehmood et al. '13 [115]	HIDS, NIDS, DIDS, VMM	Discussed	NA	Discussed	Discussed
Liao et al. '13 [105]	HIDS, NIDS, DIDS, VMM	NA	NA	Discussed	NA
Modi et al. '13 [119]	HIDS, NIDS, DIDS, VMM	Discussed	NA	NA	NA
Kumar & Gohil '15 [98]	HIDS, NIDS, DIDS	Discussed	NA	NA	NA
Kahn et al. '16 [86]	HIDS, NIDS	Discussed	NA	NA	NA
Chiba et al. '16 [23]	HIDS, NIDS, DIDS, VMM	Discussed	NA	NA	NA
Buczak & Guven '16 [18]	NIDS	NA	Discussed	Discussed	NA
Mishra et al. '17 [118]	VMM	Discussed	NA	NA	NA

Axelsson [10] surveys anomaly and misuse papers pre-2000 and organizes the research by sorting on the proposed problem's level of difficultly.

Patcha & Park [130] perform an in-depth review of anomaly and hybrid based intrusion detection papers spanning from 2000-2006. The papers are organized based on the classification 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 & Ghorbani [78] 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. [100] 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, raw pages/introspection are excluded.

Sabahi et al. [142] 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. [115] 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. [105] 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. [119] offer recommendations for IDS/IPS placement within cloud environments to reach common security goals in next-generation networks.

Kumar & Gohil [98] discuss traditional attack types and analysis techniques used for HIDSs, NIDSs, and DIDSs.

Khan et al. [86] 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. [23] 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 [18] 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. [118] 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.

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, etc. 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. We note that system calls can be considered a subset of audit data, but because there is a rich progression of research which 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.

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.

Table 2: HIDS with System Logs

IDS Reference	Technique	Dataset	Classifier	Learning
Reuning '04 [140]	Anomaly	DARPA99	TF-IDF	Supervised
Guan et al. '05 [53]	Anomaly	NA	NN	Supervised
Zhaojun & Chao '10 [183]	Anomaly	NA	NN	Supervised
Tchakoucht et al. '15 [160]	Anomaly	Simulation	Clustering	Unsupervised
Wang & Zhu '17 [171]	Anomaly	KDD99	C5.0 DT	Supervised

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 following references, [36, 139, 168, 167].

3.1.1 System Log IDS Research

Reuning [140] 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. [160] improve upon the IDS of Yacine et al. [14]. 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 [14] 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.

3.1.2 System Log IDS Architecture Research

Guan et al. [53] 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 the SSL library in Java and a Certificate Authority for client to server authentication. The NN module is used to analyze received log files for abnormal behavior. In this work, no experiments are presented.

Zhaojun & Chao [183] describe a new type of HIDS architecture based on an analysis of 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 & Zhu [171] 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, 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.

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.

Ilgun [72] 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. [136]. 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

[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
Ilgun '93 [72]	Misuse	NA	Rules	NA
Ye et al. '01, '02 [179, 178]	Misuse, Anomaly	DARPA98 + simulation	DT, T^2 test, χ^2 test, Markov model	Both
Botha & Von Solms '03 [13]	Misuse	Self made	Rules + Fuzzy logic	NA
Li & Manikopoulos '04 [104]	Anomaly	Self made	OCSVM	Unsupervised
Shavlik & Shavlik '04 [148]	Anomaly	Self made	Winnow, NB	Supervised
Lin et al. '10 [109]	Hybrid	NA	OSSEC, NN	Supervised
Mehnaz & Bertino '17 [116]	Anomaly	Self made	FSA rule-mining	Unsupervised

process and make patterns more flexible to adopt to different instances of equivalent attacks. This proposed IDS is able to detect attacks which 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. [179] 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-on work by Ye et al. [178] present 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 & Von Solms [13] 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%.

Li & Manikopoulos [104] model user profiles with one-class support vector machines (OCSVMs),

an unsupervised support vector machine (SVM) technique requiring only the user's own legitimate sessions to build the user's profile, using a year of Windows audit data with a focus on masquerade detection. This approach allows for easier user management, such as adding and removing users legacy 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 & Shavlik [148] that performs statistical profiling of users and system behavior on Windows 2000. Their algorithm uses measurements taken from two-hundred Windows 2000 attributes at one second intervals to 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. User behavior is accurately identified using features with assigned weights. Moreover, unique signatures are created by assigning individual user feature weights.

Winnow [110], a simple linear binary classification fitting algorithm, generally used with a large number of features, is tested against Naïve Bayes (NB) classifier. Succeeding training, all features "vote" every second as "for" or "against" the likelihood of an intrusion occurrence. The weighted votes are then compared against a threshold to determine if there is an intrusion. During training, Winnow changes the weight of features that fired on incorrectly labeled instances, similar to perceptron training. 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.

Lin et al. [109] propose an HIDS architecture combining open-source misuse detection with supervised learning. Misuse detection is implemented using OSSEC [161] (Open Source HIDS SECurity), an open-source IDS framework capable of conducting analysis of log files; anomaly detection is implemented using a BPNN. The misuse detection process consists of collecting log data, pre-processing and analyzing data with OSSEC, and finally reporting results. The anomaly detection process consists of training the BPNN. This can be trained with login or session activities, resource utilization, file operation activities, among other data types. BP training may take days or weeks and was cited as an area of active research. In this work, no testing was discussed.

Mehnaz & Bertino [116] 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. 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%.

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 [122, 101, 177, 181].

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 [28]. Often the unit of data used for detection is a system call

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

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 [9].

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 of sequences of calls are configured to produce normal/attack classifications. See Forrest et al. [43] 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, etc.
- 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 which 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.

Table 4: HIDS Leveraging System Calls

IDS Reference	Technique	Dataset	Classifier	Learning
Forest et al. '96 [44]	Anomaly	Simulated	Rules	Unsupervised
Kosoresow et al. '97 [93]	Anomaly	Self made	FSA	Unsupervised
Hofmeyr et al. '98 [66]	Anomaly	Simulated + Self made	FSA	Unsupervised
Ghosh et al. '99 [48, 49]	Hybrid	DARPA 99	NN	Both
Warrender et al. '99 [172]	Hybrid	UNM, DARPA98	Rules, HMM	Unsupervised
Sekar et al. '01 [147]	Anomaly	Simulated + Self made	FSA	Unsupervised
Wagner & Dean '01 [169]	NA	Self made	Static Analysis	NA
Liao & Vemuri '02 [106]	Hybrid	DARPA98	k-NN	Both
Abad et al. '03 [1]	Hybrid	Self made	RIPPER	Both
Feng et al. '03 [41]	Anomaly	Simulated + Self made	FSA	Unsupervised
Hoang et al. '03, '09 [64]	Hybrid	UNM	HMM + Rules	Unsupervised
Kruegel et al. '03 [96]	Anomaly	DARPA 99	BN	Unsupervised
Kruegel et al. '03 [97]	Anomaly	DARPA 99	Probability models	Unsupervised
Jha et al. '04 [74]	Anomaly	UNM	Filtering, Markov Models	Unsupervised
Tandon & Chan '05, '06 [157, 158]	Anomaly	UNM, DARPA98	Rules	Unsupervised
Han & Cho '05 [60]	Anomaly	DARPA99	ENN	Unsupervised
Zhang et al. '05 [182]	Hybrid	DARPA98	k-NN, Robust SVM, SVM, OCSVM	Both
Gao et al. '06 [46]	Anomaly	Self made	HMM-based distance	Unsupervised
Hu et al. '09 [68]	Anomaly	UNM, DARPA98	HMM	Unsupervised
Ahmed et al. '09 [2]	Hybrid	UNM	RBFNN	Supervised
Tong et al. '09 [162]	Hybrid	DARPA	RBFNN + ENN	Supervised
Ye et al. '10 [180]	Anomaly	NA	Rules	Unsupervised
Jewell & Beaver '11 [73]	Anomaly	Self made	Rules	Unsupervised
Elgraini et al. '12 [39]	Anomaly	UNM	NB with a MM	Unsupervised
Xie et al. '13, '14 [174, 176, 175]	Anomaly	ADFA-LD	k-NN, OCSVM, k-Means	Unsupervised
Creech & Hu '14 [28]	Anomaly	UNM, ADFA-LD, KDD98	ELM NN	Supervised
Anandapriya & Lakshmanan '15 [6]	Anomaly	ADFA-LD	SVM, ELM NN	Supervised
Gupta & Kumar '15 [54]	Misuse	UNM	Rules	Unsupervised
Haider et al. '15 [58]	Anomaly	ADFA-LD	k-NN	Unsupervised

4.1 Sequential Features (n-Grams)

Early work of Forrest and Longstaff [44] 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.

Similar anomaly detectors based on modeling n-grams of system calls were explored by others, but without statistically modeling their frequencies [66, 93]. Helman & Bhangoo [63] rank system call traces by the likelihood of n-grams in normal versus attack scenarios. Ye et al. [180] 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.

For each process, Jewell & Beaver [73] 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 system call sequences observed plateau for normal user activity faster than other definitions, and 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. [39] estimate the probability of a sequence of calls conditioned on the class (normal/attack) using a first order Markov model—$P(s_1, s_2, ...|C) = P(s_1|C)P(s_2|s_1, C)....$ 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 & Hu [28] 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. [69]. 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 & Lakshmanan [6] also test anomaly detection results using semantic features with the ELM on the ADFA-LD dataset.

Gupta & Kumar [54] 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.

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 & Vemuri [106] 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, 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

[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`.

rate and 0.59% false positive rate with threshold of 0.8. This method is computationally efficient, with complexity $\mathcal{O}(N)$, with N as the number of processes.

Continuing the system call/trace interpretation as words or documents, respectively, Zhang et al. [182] 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 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 & Hu, and Xie et al. [174, 176, 175] 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. [58] 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. [174, 176, 175], for similar false positive metrics, but are far less accurate than the computationally expensive work of Creech et al. [28]. 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. [172] 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 [26], 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. [46] 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. [65] 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 [64], 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. [68] 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. [74] introduce a novel statistic-based anomaly detection algorithm for system call sequences. They observe that after Markov models (not HMMs) 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 co-operating. 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. [48, 49] 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 & Cho [60] introduce an IDS utilizing evolutionary neural networks (ENNs) to simultaneously calculate the NNs 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. [162] 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. [48]. 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 [48] and [77], ultimately producing higher accuracy and lower false positive rates.

Ahmed & 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 & Dean [169] 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 [70, 120].

Kruegel et al. [96] 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 $\mathcal{O}(N)$ 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 & Stolfo [102]. Results show an increased detection rate and a decreased false positive rate.

For each system call (e.g., read, write, ..) for each process (e.g., sendmail), Kruegel et al. [97] build models of normal arguments' string lengths, characters, and structure. Similar features found in Kruegel et al. [96] 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 5Kb of memory is required, and 18% (of a 2003 era) processor was used. Follow-on research of Mutz et al. [123] uses the same Bayesian network of Kruegel et al. [96] to combine these system call argument feature anomaly detectors into an ensemble.

Tandon & Chan [157, 158] 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. [12] and Sufatrio & Yap [153].

Sekar et al. [147] 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. [41] incorporate dynamic extraction of return addresses in addition to the FSA approach, yielding additional accuracy without increased cost.

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 [50, 79, 80, 81, 95, 169] or malicious call sequences are sufficiently small to evade detection [156, 155].

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 [67]. Consequently, Registry monitoring has been leveraged by many researcher efforts for forensic analysis [19, 35, 114]. Below we survey the few works that build HIDS from Registry data.

Initial work by Apap et al. [9] 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 [45], 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. [62] and a follow-on publication of Stolfo et al. [152] 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.

Table 5: Registry Anomaly Detection Systems

IDS Reference	Technique	Performance Cost	Memory Cost
Apap et al. '02 [9]	PAD	$\mathcal{O}(v^2d^2)$	$\mathcal{O}(vd^2)$
Heller et al. '03 [62]	OCSVMs	$\mathcal{O}(dL^3)$	$\mathcal{O}(d(L+T))$
Stolfo et al. '05 [152]	OCSVMs	$\mathcal{O}(dL^3)$	$\mathcal{O}(d(L+T))$
Topallar et al. '04 [163]	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.

PAD takes time $\mathcal{O}(v^2d^2)$ and space $\mathcal{O}(vd^2)$ where v denotes the number of unique records, and d denotes number of record components (dimension). The OCSVM takes time $\mathcal{O}(dL^3)$ and space $\mathcal{O}(d(L+T))$ 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 3MB, and 3%-5% of CPU usage.

Topallar et al. [163] 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.

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.

First available in 1992, Tripwire[10], from Kim & Spafford [90], is perhaps the most notable file-integrity tool. Tripwire is an open-source and now commercially available IDS for detection and

[10] See `www.tripwire.org`.

Table 6: File System Detection Systems

IDS Reference	Name	Rule Base
Kim & Spafford '94 [90]	Tripwire	Checksum
Griffin et al.'03 [52]	Disk-Based IDS	Policy
Pennington et al. '03 [133]	Storage-Based IDS	Checksum & policy
Patil et al. '04 [131]	I^3FS	Checksum & policy

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. [52] 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 which 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. [133] 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 [90]) 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 of the system is investigated, and results show under 2MB 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. [131] describe I^3FS, 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 [90] and can overcome its limitations—intruder tampering, large performance overhead, and inability of real-time detection. I^3FS 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 which 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^3FS 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 & Takefuji [137] 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. [92] 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. [55, 56] describe a lightweight and platform independent HIDS based on monitoring file system integrity while running as privileged VM. Jin et al. [76] 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. [33]. 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 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.

Table 7: Program Analysis for Detection Works

IDS Reference	Data Source	Classifier	Learning
Schultz et al. '01 [146]	Binary, DLL, calls	RIPPER, NB, MNB	Supervised
Newsome & Song '05 [125]	Binary	Rules	Unsupervised
Moscovitch et al. 07 [121]	Program's resource utilization	DT, NB, BN, ANN	Supervised
Khan et al. '16, '17 [88, 87]	Process' network utilization	AdaBoost	Supervised
Vaas & Happa '17 [165]			

Schultz et al. [146] 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.

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.

Table 8: Schultz et. al. [146] 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.

TaintCheck, a system that can "perform dynamic taint analysis by performing binary rewriting at run time" was developed by Newsome & Song [125]. 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 permits semantic analysis using signatures.

Moscovitch et al. [121] 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, etc. 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. [88] introduce an anomaly detection HIDS based on a modified AdaBoost ensemble classifier. They add an "information fractal (cognitive) dimension approach", which assigns higher weights to weak classifiers, which puts emphasis on misclassified samples to improve their estimation. This idea stems from fractal dimension theory [91]. 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 1 hour and used malware detected by 3 out of 54 antivirus companies, according to VirusTotal[11]. To build a classifier, the dataset was partitioned into 70% and 30% as 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. [149] uses k-NN with a fractal weighting approach on network data for detection, and further work in Khan et al. [87] attempting to model polymorphic malware with fractal analysis of the process tree.

Vaas & Happa [165] 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, in order to make application models available more quickly, parallel machine learning techniques are utilized.

8 Host-Relevant Algorithms Tested on Network Traffic

This subsection surveys select works that focus on algorithmic development. Most focus on advancing detection metrics or performance as tested on the DARPA- and KDD-related datasets; hence, their presentation falls under NIDS applications, but the detection algorithms can be applied to HIDS. Note a progression in the literature from testing standard machine learning algorithms against each other,

[11] See `www.virustotal.com`.

Table 9: Selected algorithmic developments for IDS, not tested on host data but applicable

IDS Reference	Technique	Dataset	Classifier	Learning
Barbará et al. '01 [11]	Anomaly	DARPA99	Rule mining, DT	Supervised
Ambwani '03 [4]	Hybrid	DARPA98	SVM	Supervised
Amor et al. '04 [5]	Hybrid	KDD99	NB	Supervised
Liu et al. '04 [111]	Anomaly	KDD99	Genetic Clustering	Unsupervised
Chen et al. '07 [21]	Misuse	KDD99, DARPA98	NN	Supervised
Kayacik et al. '07 [84]	Anomaly	KDD99	Hierarchical SOM	Unsupervised
Özyer et al. '07 [128]	Hybrid	KDD99	Rules+boosting	Supervised
Peddabachigari et al. '07 [132]	Anomaly	KDD99	DT, SVM, DT-SVM, Ensemble	Supervised
Tajbakhsh et al. '09 [154]	Hybrid	KDD99	ABC	Supervised
Wang et al. '10 [170]	Hybrid	KDD99	FC-ANN	Both
Nadiammai et al. '11 [124]	Misuse	KDD99	ZR, DT, RF	Supervised
Lin et al. '12 [107]	Misuse	KDD99	DT and SVM	Supervised
Kim et al. '14 [89]	Hybrid	NSL-KDD	DT and SVM	Supervised
De la Hoz et al. '14 [30]	Hybrid	DARPA98, NSL-KDD	NSGA-II+GH-SOM	Supervised
Eesa et al. '15 [38]	Misuse	KDD99	CFA-DT	Supervised
Lin et al. '15 [108]	Hybrid	KDD99	Clustering	Supervised
Ravale et al. '15 [138]	Hybrid	KDD99	k-means + RB-SVM	Supervised
Harshaw et al. '16 [61]	Anomaly	Self-made	Graphprints	Unsupervised
Ikram & Cherukuri '16 [71]	Hybrid	NSL-KDD, GURE-KDD	PCA+SVM	Supervised
Mishra et al. '16 [117]	Hybrid	KDD99	NB, NN, C4.5 DT	Supervised
Zhu et al. '17 [184]	Hybrid	NSL-KDD, GURE-KDD, KDD99	I-NSGA-III	Supervised

to testing ensembles to gain accuracy, to parallelization and GPU acceleration efforts, and finally to pre-composing the classifiers with data-driven feature selection algorithms. For the interested reader, we note that this section complements the survey of Buczac & Guven [18], which also surveys only a minority of the many works focusing on machine learning advancements for detection, most of which are also tested on NIDS datasets, though Buczac & Guven focus on more detailed analyses of the algorithms involved.

Barbará et al. [11] introduce a new IDS, Audit Data Analysis and Mining (ADAM), for anomaly detection. ADAM is designed to be used on-line and is implemented using a data mining technique to build rules of normal behavior and a classifier to identify attacks in TCP dump recorded traffic. First, using data mining, ADAM builds a repository of attack-free items. Then using an online, sliding-window algorithm, it identifies frequent items to compare with the repository. Abnormal items are classified via a DT as a known attack, unknown attack, or false alarm. The system was tested using DARPA99 dataset and took third.

Ambwani et al. [4] propose a multi-class SVM classifier using a one-versus-one method[12] to identify various misuse and attacks by type. The authors cite previous research to claim one-versus-one outperforms one-versus-all methods. An experiment is conducted using labeled 10% training and testing sets from the KDD99 dataset. Accuracy (91%) and performance are reported.

Amor et al. [5] compare NB and DT classifiers for intrusion detection, concluding DT is slightly

[12] To create an n-class classifier, the one-versus-one method trains binary classifiers to distinguish each of the $n(n-1)/2$ pairs of classes. Usually a voting scheme collects the $n(n-1)/2$ votes to make a final classification.

more accurate but learning and classifying is seven times longer. There were three experiments performed on the KDD99 traffic data, and both algorithms perform in the 91-94% range.

Liu et al. [111] describe an unsupervised, clustering-based approach, using a genetic algorithm to classify network traffic. This approach achieves good results overall when evalutated with the KDD99 datasets, and includes an adjustable sensitivity factor, i.e., a parameter to trade false positives for false negatives.

Chen et al. [21] present an evolutionary NN—meaning the NN structure is learned as well as the parameters during training—trained by a particle swarm optimization (PSO) algorithm with flexible bipolar sigmoid activation functions. This work focuses on improving the intrusion detection performance by selecting an effective small subset of the input features. The algorithm's effectiveness is demonstrated on network data from the KDD99 dataset, and then its application to host-based datasets is discussed.

Kayacik et al. [84] demonstrate an unsupervised approach, using a hierarchy of SOMs, which results in similar detection rates as supervised approaches in other publications. When tested on the KDD99 dataset, the false positive rate of this unsupervised approach is about three times higher than supervised methods, and the authors discuss ways to reduce this further.

Özyer et al. [128] describe an intelligent intrusion detection system (IIDS) that iteratively learns association and classification rules. Fuzzy association rule mining and filtering is used for each class, then a boosting algorithm is used for classification. Results appeared to be close to the KDD99 winning entry and proved the usefulness of this boosted fuzzy classifier. The authors believe that results can be farther improved with an increased training dataset.

Peddabachigari et al. [132] evaluate four classifiers for intrusion detection: DT, SVM, hybrid DT-SVM (first data is passed through a DT, and then the same data plus the result of the DT is passed through an SVM), and an ensemble, with the main goal to find the most accurate algorithm. Experiments are conducted on the KDD99 dataset, and it was determined that the ensemble approach performed the best. It classified non-attack, Probe, DoS, U2R, and R2U attacks with 99.70%, 100%, 99.92%, 68%, and 97.16% of accuracy, respectively. To utilize the achieved results, a hybrid IDS is proposed, employing ensemble classifiers.

Tajbakhsh et al. [154] describe an IDS using fuzzy association rules and Association Based Classification (ABC) for misuse detection and an ABC extension for anomaly detection. The objective is a fast data mining technique that can be used to implement an IDS. The association rule induction algorithm grows exponentially according to the dataset's feature quantity. To accelerate this technique they introduce a concept called association hyper-edge, which reduces the above problem to a graph problem. The main challenge of the proposed classification is to define appropriate matching measures; two solutions are described and tested. Evaluations of their misuse detection approach using the KDD99 dataset realized a 91% detection rate and a 3.34% false positive rate; the anomaly detection approach had a detection rate of 80.6% with 2.95% false positive rate for all records with total execution time of about 500s. Detection of novel attacks was much less effective, as the misuse detection approach resulted in 11% detection rate for new attacks, and the anomaly detection approach resulted in 20.5% detection rate for these new attacks.

Wang et al. [170] describe an IDS using fuzzy clustering (FC) to partition training data into subsets and train an NN on each subset; the results are aggregated with a fuzzy aggregation module for detection. An experiment was conducted using the KDD99 dataset, and results concluded that FC-ANN works similar to BPNN, NB, and DT in detecting high frequency attacks, and outperforms in detecting low frequency attacks. The average accuracy of this method is 96.71%. Training time is higher than the other methods, but can be improved with parallelization.

Nadiammai et al. [124] determine the severity of attacks in a dataset with data mining methods. Several classifiers are used to determine the classification accuracy. The Zero R (ZR) classifier, DT classifier, and Random Forest (RF) classifier are compared using the KDD99 dataset. Results show that RF outperforms the other classification algorithms.

Lin et al. [107] develop an IDS based on SVMs and DTs, and use Simulated Annealing (SA)—a

probabilistic optimization technique for avoiding local optima—for feature selection and parameter selection. This is evaluated with the KDD99 dataset, and the results show very high accuracy. SVM and SA can find the optimal subset of features for anomaly intrusion detection accuracy evaluation while DT and SA can obtain decision rules for new attacks and can improve accuracy of classification.

Mishra et al. [117] propose two IDS frameworks for a cloud network to detect a broad range of attacks and focus on speed and efficiency by employing parallelization techniques. Two types of parallelization were introduced, splitting traffic into multiple sensor nodes and using GPU acceleration programmed in CUDA. Experiments were conducted using the KDD99 dataset. The first setup used a single node running an ensemble of NB, NN, and DT machine learning algorithms and resulted in 80-99.9% of accuracy of each attack category. The authors note that NN requiress significant training time, so frequent retraining would likely be problematically time consuming. The second setup was done with the use of a multi-node environment for parallelization with each node running a single classifier: NB, NN, or DT. The second setup allows for a training time reduction for classifiers. The anomaly detection accuracy of DT and NN is about the same, ranging from 69.10-99.88% for each attack category and NB produced the poorest result ranging from 45.20-98.30%.

Harshaw et al. [61] introduce a novel graph-mining and robust statistics technique to detect multi-scale (host and network level) anomalies. Self-harvested network flow data is used and a simple graph is constructed for each 31s time window with 1 second overlap between consecutive windows. The graphs' vertices represent an IP, and directed edges represent network flows, with two edge colors to encode port information. For network-level anomaly detection, small vertex-induced subgraphs, called graphlets, are counted—this records how many small subgraphs make up the whole network flow graph for each time window—and streaming anomaly detection is done on this sequence of vectors of graphlet counts. Anomalies in this sequence indicate the network has some local topological changes. For host-level detection, the host's role in an incident graphlet is counted (automorphism orbits), and anomalies in this sequence are flagged. The upshot of this method is that network-level anomalies can be pinpointed to the hosts that changed communication patterns. To perform the anomaly detetion on the sequence of vectors, a gaussian is fit using the Minimum Covariance Determinant method of Rousseeuw & Driessen [141]—this robust fitting algorithm automatically omits $h\%$ of outliers and fits the gaussian distribution to the remaining inlier—with the goal of preventing attacks and other anomalies in the training (historical) data from affecting the models. Accurate detection of anomalous bit torrent traffic and port scans (although non malicous) were exhibited.

Bridges et al. [15] focus on mathematical foundations for setting the threshold of probabilistic anomaly detectors. Their method can be considered in two ways—either one can set the threshold to bound the number of alerts (useful in high-throughput data situations to prevent flooding downstream systems) or if the alert rate bound is broken that indicates the probability model is a poor fit to the data. To exhibit the second alternative, Bridges et al. show that the robustly fit gaussians of Harshaw et al. are overfit to the inliers, i.e., $h = 15\%$ is too high for this application.

Many of the literature reviews found in this survey discuss detailed techniques for improved feature selection via reduction and construction. Additional feature selection algorithms for IDSs can be found in the survey by Chen et al. [22].

De la Hoz et al. [30] use the NSGA-II algorithm of Deb et al. [32], a multi-objective (maximize classification metrics while minimizing the number of features) feature selection algorithm with "an unsupervised clustering procedure based on Growing Hierarchical Self-Organizing Maps" (GH-SOMs) for both attack and anomaly detection. DARPA98 and NSL-KDD datasets used for evaluation. The selected feature sets are computed and reach 99.12% accuracy while yielding normal and anomalous traffic detection rates as high as 99.8% and 99.6%, respectively. Additionally, a reduction in the GH-SOM size improves the computational efficiency, making it a viable option for performing additional tasks (e.g., IP blocking) in real time.

As non-attack data quantities far outnumber attack quantities, class imbalance is a perennial problem for machine learning intrusion detection algorithms. Zhu et al. [185] propose I-NSGA-III. Their algorithm is an improvement of NSGA-III, another multi-objective feature selection algorithm

of Deb et al. [31], to both alleviate redundant features and improve the accuracy in the face of imbalanced classes. Building on De la Hoz et al. [30], they also use GH-SOM classifier for IDS and a similar method for feature selection. The algorithm can identify the attacks and distinguish between the different attack classes. The enhanced NSGA-III algorithm (I-NSGA-III) is compared to NSGA-II and NSGA-III algorithms among other algorithms. All are evaluated with NSL-KDD, KDD99, and GURE-KDD datasets.

Results indicate that I-NSGA-III algorithm requires the least training and testing time with the best or near best detection rate.

Kim et al. [89] introduce a new hybrid IDS for detection of anomalies and misuses. Their method is a combination of C4.5 DT and OCSVM methods. First, a DT is used for misuse detection and partitioning data into smaller subsets. Then, multiple OCSVMs are applied to each subset for anomaly detection. Using the NSL-KDD dataset for evaluation, results indicate that the proposed hybrid IDS has higher detection rate by about 10%, while false positive rate is below 10%, and shows faster training compared to other conventional methods. Training and testing time was 56.58s and 11.20s, respectively.

Ravale et al. [138] create a hybrid learning approach by first applying k-means clustering, to reduce the large heterogeneous dataset into multiple, smaller homogeneous subsets, and to select features. Following this, a Radial SVM classifier is used with the selected features. Evaluation with the KDD99 dataset resulted in increased performance with greater rates of accuracy and detection, while achieving a fewer false alarms than either SVM or k-means independently.

Eesa et al. [38] describe an IDS approach that minimizes the quantity of features utilized while maximizing the detection rate. For feature-selection, a new cuttlefish optimization algorithm (CFA) [37] is used as a search method to determine the ideal feature subset, and then a DT uses the CFA-selected subset of features to perform classification. An experiment was conducted using the KDD99 dataset; this was processed eight times through the DT classifier with different subsets of of the CFA-selected features, ranging from 5 to 41 features. The results reveal that the feature subsets containing 20 features or less, ascertained with CFA, provide greater rates of both detection and accuracy while presenting fewer false alarms than results utilizing every feature.

Lin et al. [108] describe a feature representation method that combines Cluster Centers and Nearest Neighbors (CANN). The CANN method converts original features into a single "distance-based feature" that is fed to a k-NN classifier. This gives an initial computation cost for the dimension reduction, but yields better performance in detection. Results on KDD99 data show that CANN yields better accuracy, and true positive and false positive rate than either k-NN or SVM classifiers with unmodified six features. CANN is, however, ineffective at both U2R and R2L attack detection.

Ikram & Cherukuri [71] describe a hybrid IDS with improved accuracy and lower resource consumption, using principal component analysis (PCA) for initial dimension reduction followed by an optimized SVM. The novelty of this proposed approach is optimization of an SVM punishment factor and RBF kernel's gamma parameter using automatic parameter selection. To test this proposed approach, NSL-KDD and GURE-KDD network traffic datasets were used. Detection accuracy for NSL-KDD data is reported before and after applying PCA of each class—Probe, DoS, U2R, R2L and Normal classes—from 85.65% to 90.84%, from 46.41% to 80.43%, from 33.33% to 78.35%, from 87.19% to 78.35% and from 97.38% to 63.62%, respectively.

We conclude this subsection with an introspective work on the flurry of algorithmic research spawned by the KDD and DARPA datasets. Motivated by many machine learning algorithms achieving greater than 90% detection rates with less than 1% false positive rates using the KDD99 dataset [135, 103, 40, 82], Kayacik et al. [83] conduct feature-relevance analysis to gain insight. The most relevant features with respect to dataset's labels are determined by employing information gain in conjunction with DTs; therefore, the highest information gain determines the most discriminative features, which makes classification easier. Due to the need for distinct features for information gain calculation, continuous features are converted to discrete features by equally partitioning with the equal frequency intervals method [173]. Three of the provided classes, "normal", "smurf", and "nep-

tune", are highly related to certain features and are comprised of 98% of the training data thus making it easily classifiable for machine learning algorithms. While the quantity of data exchanged is the most discriminating feature for 14 of the 23 classes, some features have no effect on intrusion detection, indicating that they are not needed.

9 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. In order 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.

9.1 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. On the other hand, 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 & Paxson work [150]. 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[13]. Research is needed to leverage the many diverse but related data sources available to an HIDS, (not only 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.

[13] See `https://github.com/stucco`.

We note that the new Unified Host and Network Data set of Turcotte et al. [164] contributes to these first two directions by providing real network and host data for researchers. Further, efforts such as Ilgun [72] 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 [101] and for threshold tuning [15].

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.

Acknowledgement

The authors would like to 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. 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 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.

References

[1] C. Abad, J. Taylor, C. Sengul, W. Yurcik, Y. Zhou, and K. Rowe. Log correlation for intrusion detection: A proof of concept. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 255–264, Las Vegas, NV, USA, 2003. IEEE.

[2] U. Ahmed and A. Masood. Host based intrusion detection using rbf neural networks. In *Emerging Technologies, 2009. ICET 2009. International Conference on*, pages 48–51, Islamabad, Pakistan, 2009. IEEE.

[3] M. Alazab, S. Venkatraman, P. Watters, M. Alazab, and A. Alazab. Cybercrime: the case of obfuscated malware. In *Global Security, Safety and Sustainability & e-Democracy*, pages 204–211. Springer, Berlin, Heidelberg, 2012.

[4] T. Ambwani. Multi class support vector machine implementation to intrusion detection. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 2300–2305, Portland, OR, USA, 2003. IEEE.

[5] N. B. Amor, S. Benferhat, and Z. Elouedi. Naive bayes vs decision trees in intrusion detection systems. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 420–424, New York, NY, 2004. ACM.

[6] M. Anandapriya and B. Lakshmanan. Anomaly based host intrusion detection system using semantic based system call patterns. In *Intelligent systems and control (ISCO), 2015 9th international conference*, pages 1–4, Coimbatore, India, 2015. IEEE.

[7] J. P. Anderson. Computer security technology planning study. volume 2. Technical report, Anderson (James P) and Co Fort Washington PA, 1972.

[8] J. P. Anderson et al. Computer security threat monitoring and surveillance. Technical report, Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, 1980.

[9] F. Apap, A. Honig, S. Hershkop, E. Eskin, and S. Stolfo. Detecting malicious software by monitoring anomalous windows registry accesses. In *Recent Advances in Intrusion Detection*, pages 36–53, Berlin, Heidelberg, 2002. Springer, Springer.

[10] S. Axelsson. Intrusion detection systems: A survey and taxonomy. Technical report, Technical report, 2000.

[11] D. Barbara, J. Couto, S. Jajodia, L. Popyack, and N. Wu. Adam: Detecting intrusions by data mining. In *In Proceedings of the IEEE Workshop on Information Assurance and Security*, New York, NY, 2001. Citeseer, IEEE.

[12] S. Bhatkar, A. Chaturvedi, and R. Sekar. Dataflow anomaly detection. In *Security and Privacy, 2006 Symposium*, pages 15–pp, Berkeley/Oakland, CA, USA, 2006. IEEE.

[13] M. Botha and R. Von Solms. Utilising fuzzy logic and trend analysis for effective intrusion detection. *Computers & Security*, 22(5):423–434, 2003.

[14] Y. Bouzida and S. Gombault. Eigenprofiles for intrusion detection, profils propres pour la detection d'intrusion. In *Département RSM GET/ENST*, Bretagne, France, 2003. Actes du Symposium SSTIC.

[15] R. A. Bridges, J. D. Jamieson, and J. W. Reed. Setting the threshold for high throughput detectors: A mathematical approach for ensembles of dynamic, heterogeneous, probabilistic anomaly detectors. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1071–1078, Boston, MA, USA, Dec 2017. IEEE.

[16] S. T. Brugger and J. Chow. An assessment of the darpa ids evaluation dataset using snort. *UCDAVIS department of Computer Science*, 1(2007):22, 2007.

[17] G. Bruneau. The history and evolution of intrusion detection. *SANS Institute*, 1(2f), 2001.

[18] A. L. Buczak and E. Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176, 2016.

[19] H. Carvey. The windows registry as a forensic resource. *Digital Investigation*, 2(3):201–205, 2005.

[20] Q. Chen and R. A. Bridges. Automated behavioral analysis of malware: A case study of wannacry ransomware. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 454–460, Cancun, Mexico, Dec 2017. IEEE.

[21] Y. Chen, A. Abraham, and B. Yang. Hybrid flexible neural-tree-based intrusion detection systems. *International journal of intelligent systems*, 22(4):337–352, 2007.

[22] Y. Chen, Y. Li, X.-Q. Cheng, and L. Guo. Survey and taxonomy of feature selection algorithms in intrusion detection system. In *Information security and cryptology*, pages 153–167, Berlin, Heidelberg, 2006. Springer, Springer.

[23] Z. Chiba, N. Abghour, K. Moussaid, A. El Omri, and M. Rida. A survey of intrusion detection systems for cloud computing environment. In *Engineering & MIS (ICEMIS), International Conference on*, pages 1–13, Agadir, Morocco, 2016. IEEE.

[24] W. Choi, H. J. Jo, S. Woo, J. Y. Chun, J. Park, and D. H. Lee. Identifying ecus through inimitable characteristics of signals in controller area networks. *IEEE Transactions on Vehicular Technology*, PP(99):1–1, 2018.

[25] S. S. Clark, B. Ransford, A. Rahmati, S. Guineau, J. Sorber, W. Xu, K. Fu, A. Rahmati, M. Salajegheh, D. Holcomb, et al. Wattsupdoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices. In *Proceedings of USENIX Workshop on Health Information Technologies*, page 11p., Washington D.C., 2013. USENIX.

[26] W. W. Cohen. Fast effective rule induction. In *Proceedings of the twelfth international conference on machine learning*, pages 115–123, Tahoe City, CA, 1995. Elsevier.

[27] G. Creech and J. Hu. Generation of a new ids test dataset: Time to retire the kdd collection. In *Wireless Communications and Networking Conference (WCNC), 2013*, pages 4487–4492, Shanghai, China, 2013. IEEE.

[28] G. Creech and J. Hu. A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns. *IEEE Transactions on Computers*, 63(4):807–819, 2014.

[29] J. A. Dawson, J. T. McDonald, J. Shropshire, T. R. Andel, P. Luckett, and L. Hively. Rootkit detection through phase-space analysis of power voltage measurements. In *12th IEEE International Conference on Malicious and Unwanted Software (MALCON 2017)*, San Juan, Puerto Rico, 2018. IEEE.

[30] E. de la Hoz, E. de la Hoz, A. Ortiz, J. Ortega, and A. Martínez-Álvarez. Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical self-organising maps. *Knowledge-Based Systems*, 71:322–338, 2014.

[31] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Trans. Evolutionary Computation*, 18(4):577–601, 2014.

[32] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[33] R. F. DeMara and A. J. Rocke. Mitigation of network tampering using dynamic dispatch of mobile agents. *Computers & Security*, 23(1):31–42, 2004.

[34] D. Denning and P. G. Neumann. *Requirements and model for IDES-a real-time intrusion-detection expert system*. SRI International, California, 1985.

[35] B. Dolan-Gavitt. Forensic analysis of the windows registry in memory. *digital investigation*, 5:S26–S32, 2008.

[36] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen. Revirt: Enabling intrusion analysis through virtual-machine logging and replay. *ACM SIGOPS Operating Systems Review*, 36(SI):211–224, 2002.

[37] A. Eesa, A. Abdulazeez, and Z. Orman. Cuttlefish algorithm - a novel bio-inspired optimization algorithm. *International Journal of Scientific & Engineering Research*, 4(9):1978–1986, 2013.

[38] A. S. Eesa, Z. Orman, and A. M. A. Brifcani. A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems. *Expert Systems with Applications*, 42(5):2670–2679, 2015.

[39] M. T. Elgraini, N. Assem, and T. Rachidi. Host intrusion detection for long stealthy system call sequences. In *Information Science and Technology (CIST), 2012 Colloquium in*, pages 96–100, Fez, Morocco, 2012. IEEE.

[40] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. *Applications of data mining in computer security*, 6:77–102, 2002.

[41] H. H. Feng, O. M. Kolesnikov, P. Fogla, W. Lee, and W. Gong. Anomaly detection using call stack information. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 62–75, Berkeley, CA, USA, 2003. IEEE.

[42] E. M. Ferragut, J. Laska, and R. A. Bridges. A new, principled approach to anomaly detection. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 210–215, Boca Raton, FL, USA, 2012. IEEE.

[43] S. Forrest, S. Hofmeyr, and A. Somayaji. The evolution of system-call monitoring. In *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, pages 418–430, Anaheim, CA, USA, 2008. IEEE.

[44] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for unix processes. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 120–128, Oakland, CA, USA, 1996. IEEE.

[45] N. Friedman and Y. Singer. Efficient bayesian parameter estimation in large discrete domains. In *Advances in neural information processing systems*, pages 417–423, Cambridge, MA, USA, 1999. The MIT Press.

[46] D. Gao, M. K. Reiter, and D. Song. Behavioral distance measurement using hidden markov models. In *RAID*, pages 19–40, Berlin, Heidelberg, 2006. Springer.

[47] S. Garcia, M. Grill, J. Stiborek, and A. Zunino. An empirical comparison of botnet detection methods. *computers & security*, 45:100–123, 2014.

[48] A. K. Ghosh and A. Schwartzbard. A study in using neural networks for anomaly and misuse detection. In *USENIX security symposium*, volume 99, page 12, Berkeley, CA, USA, 1999. USENIX.

[49] A. K. Ghosh, A. Schwartzbard, and M. Schatz. Learning program behavior profiles for intrusion detection. In *Workshop on Intrusion Detection and Network Monitoring*, volume 51462, pages 1–13, Berkeley, CA, USA, 1999. USENIX.

[50] J. T. Giffin, S. Jha, and B. P. Miller. Automated discovery of mimicry attacks. In *RAID*, volume 4219, pages 41–60, Amsterdam, The Netherlands, 2006. Springer.

[51] C. R. A. González and J. H. Reed. Power fingerprinting in SDR integrity assessment for security and regulatory compliance. *Analog Integrated Circuits and Signal Processing*, 69(2-3):307–327, 2011.

[52] J. L. Griffin, A. Pennington, J. S. Bucy, D. Choundappan, N. Muralidharan, and G. R. Ganger. On the feasibility of intrusion detection inside workstation disks. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 2003.

[53] D. Guan, K. Wang, X. Ye, and W. Feng. A collaborative intrusion detection system using log server and neural networks. In *Mechatronics and Automation, 2005 IEEE International Conference*, volume 2, pages 874–877, Niagara Falls, Ont., Canada, 2005. IEEE.

[54] S. Gupta and P. Kumar. An immediate system call sequence based approach for detecting malicious program executions in cloud environment. *Wireless Personal Communications*, 81(1):405–425, 2015.

[55] S. Gupta, P. Kumar, A. Sardana, and A. Abraham. A secure and lightweight approach for critical data security in cloud. In *Computational Aspects of Social Networks (CASoN), 2012 Fourth International Conference on*, pages 315–320, Sao Carlos, Brazil, 2012. IEEE.

[56] S. Gupta, A. Sardana, and P. Kumar. A light weight centralized file monitoring approach for securing files in cloud environment. In *Internet Technology And Secured Transactions, 2012 International Conference for*, pages 382–387, London, UK, 2012. IEEE.

[57] W. Haider, G. Creech, Y. Xie, and J. Hu. 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):29, 2016.

[58] W. Haider, J. Hu, and M. Xie. Towards reliable data feature retrieval and decision engine in host-based anomaly detection systems. In *Industrial Electronics and Applications (ICIEA), 2015 IEEE 10th Conference on*, pages 513–517, Auckland, New Zealand, 2015. IEEE.

[59] W. Haider, J. Hu, X. Yu, and Y. Xie. Integer data zero-watermark assisted system calls abstraction and normalization for host based anomaly detection systems. In *Cyber Security and Cloud Computing (CSCloud), 2015 IEEE 2nd International Conference on*, pages 349–355, New York, NY, USA, 2015. IEEE.

[60] S.-J. Han and S.-B. Cho. Evolutionary neural networks for anomaly detection based on the behavior of a program. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(3):559–570, 2005.

[61] C. R. Harshaw, R. A. Bridges, M. D. Iannacone, J. W. Reed, and J. R. Goodall. Graphprints: towards a graph analytic method for network anomaly detection. In *Proceedings of the 11th Annual Cyber and Information Security Research Conference*, pages 1–15, New York, NY, 2016. ACM.

[62] K. A. Heller, K. M. Svore, A. D. Keromytis, and S. J. Stolfo. One class support vector machines for detecting anomalous windows registry accesses. *In Proc. of the workshop on Data Mining for Computer Security*, 9, 2003.

[63] P. Helman and J. Bhangoo. A statistically based system for prioritizing information exploration under uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 27(4):449–466, 1997.

[64] X. D. Hoang, J. Hu, and P. Bertok. A multi-layer model for anomaly intrusion detection using program sequences of system calls. In *Proc. 11th IEEE Intl. Conf*, Sydney, NSW, Australia, 2003. Citeseer, IEEE.

[65] X. D. Hoang, J. Hu, and P. Bertok. A program-based anomaly intrusion detection scheme using multiple detection engines and fuzzy inference. *Journal of Network and Computer Applications*, 32(6):1219–1228, 2009.

[66] S. A. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. *Journal of computer security*, 6(3):151–180, 1998.

[67] G. Hoglund and J. Butler. *Rootkits: subverting the Windows kernel*. Addison-Wesley Professional, Indianapolis, Indiana, 2006.

[68] J. Hu, X. Yu, D. Qiu, and H.-H. Chen. A simple and efficient hidden markov model scheme for host-based anomaly intrusion detection. *IEEE network*, 23(1):42–47, 2009.

[69] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 985–990, Budapest, Hungary, 2004. IEEE.

[70] N. Idika and A. P. Mathur. A survey of malware detection techniques. *Purdue University*, 48, 2007.

[71] S. T. Ikram and A. K. Cherukuri. Improving accuracy of intrusion detection model using pca and optimized svm. *Journal of computing and information technology*, 24(2):133–148, 2016.

[72] K. Ilgun. Ustat: A real-time intrusion detection system for unix. In *Research in Security and Privacy, 1993. Proceedings., 1993 IEEE Computer Society Symposium on*, pages 16–28, Oakland, CA, USA, 1993. IEEE.

[73] B. Jewell and J. Beaver. Host-based data exfiltration detection via system call sequences. In *ICIW2011-Proceedings of the 6th International Conference on Information Warfare and Secuirty: ICIW*, page 134, England, 2011. Academic Conferences Limited, Academic Conferences Limited.

[74] S. Jha, L. Kruger, T. Kurtx, Y. Lee, and A. Smith. A filtering approach to anomaly and masquerade detection. In *University of Wisconsin, Tech. Rep*, Madison, 2004. Dept. of Computer Science, Univ. of Wisconsin.

[75] J. M. H. Jiménez, J. A. Nichols, K. Goseva-Popstojanova, S. Prowell, and R. A. Bridges. Malware detection on general-purpose computers using power consumption monitoring: A proof of concept and case study, 2017.

[76] H. Jin, G. Xiang, D. Zou, S. Wu, F. Zhao, M. Li, and W. Zheng. A vmm-based intrusion prevention system in cloud computing environment. *The Journal of Supercomputing*, 66(3):1133–1151, 2013.

[77] C. Jirapummin, N. Wattanapongsakorn, and P. Kanthamanon. Hybrid neural networks for intrusion detection system. *Proc. of ITC–CSCC*, 7:928–931, 2002.

[78] P. Kabiri and A. A. Ghorbani. Research on intrusion detection and response: A survey. *IJ Network Security*, 1(2):84–102, 2005.

[79] H. G. Kayacik, M. Heywood, and N. Zincir-Heywood. On evolving buffer overflow attacks using genetic programming. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1667–1674, New York, NY, USA, 2006. ACM.

[80] H. G. Kayacik and A. N. Zincir-Heywood. On the contribution of preamble to information hiding in mimicry attacks. In *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, volume 1, pages 632–638, Niagara Falls, Ont., Canada, 2007. IEEE.

[81] H. G. Kayacik and A. N. Zincir-Heywood. Mimicry attacks demystified: What can attackers do to evade detection? In *Privacy, Security and Trust, 2008. PST'08. Sixth Annual Conference on*, pages 213–223, Fredericton, NB, Canada, 2008. IEEE.

[82] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood. On the capability of an som based intrusion detection system. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 1808–1813, Portland, OR, USA, 2003. IEEE.

[83] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood. Selecting features for intrusion detection: A feature relevance analysis on kdd 99 intrusion detection datasets. In *Proceedings of the third annual conference on privacy, security and trust*, St. Andrews, New Brunswick, Canada, 2005. Dalhousie University.

[84] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood. A hierarchical som-based intrusion detection system. *Engineering applications of artificial intelligence*, 20(4):439–451, 2007.

[85] R. A. Kemmerer and G. Vigna. Intrusion detection: a brief history and overview. *Computer*, 35(4):supl27–supl30, 2002.

[86] M. A. Khan. A survey of security issues for cloud computing. *Journal of Network and Computer Applications*, 71:11–29, 2016.

[87] M. S. Khan, S. Siddiqui, and K. Ferens. Cognitive modeling of polymorphic malware using fractal based semantic characterization. In *Technologies for Homeland Security (HST), 2017 IEEE International Symposium on*, pages 1–7, Waltham, MA, USA, 2017. IEEE.

[88] M. S. Khan, S. Siddiqui, R. D. McLeod, K. Ferens, and W. Kinsner. Fractal based adaptive boosting algorithm for cognitive detection of computer malware. In *Cognitive Informatics & Cognitive Computing (ICCI* CC), 2016 IEEE 15th International Conference on*, pages 50–59, Palo Alto, CA, USA, 2016. IEEE.

[89] G. Kim, S. Lee, and S. Kim. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, 41(4):1690–1700, 2014.

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

[91] W. Kinsner. A unified approach to fractal dimensions. In *Fourth IEEE Conference on Cognitive Informatics, 2005.(ICCI 2005).*, pages 58–72, Irvine, CA, USA, 2005. IEEE.

[92] R. K. Ko, P. Jagadpramana, and B. S. Lee. Flogger: A file-centric logger for monitoring file access and transfers within cloud computing environments. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 765–771, Changsha, China, 2011. IEEE.

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

[94] A. Kountouras, P. Kintis, C. Lever, Y. Chen, Y. Nadji, D. Dagon, M. Antonakakis, and R. Joffe. *Enabling Network Security Through Active DNS Datasets*, pages 188–208. Springer International Publishing, Cham, 2016.

[95] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, and G. Vigna. Automating mimicry attacks using static binary analysis. In *Proceedings of the 14th conference on USENIX Security Symposium-Volume 14*, pages 11–11, Berkeley, CA, USA, 2005. USENIX.

[96] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Bayesian event classification for intrusion detection. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 14–23, Las Vegas, NV, USA, 2003. IEEE.

[97] C. Kruegel, D. Mutz, F. Valeur, and G. Vigna. On the detection of anomalous system call arguments. In *European Symposium on Research in Computer Security*, pages 326–343, Berlin, Heidelberg, 2003. Springer.

[98] U. Kumar and B. N. Gohil. A survey on intrusion detection systems for cloud computing environment. *International Journal of Computer Applications*, 109(1):6–15, 2015.

[99] M. L. Labs. Darpa intrusion detection evaluation, 2017.

[100] A. Lazarevic, V. Kumar, and J. Srivastava. Intrusion detection: A survey. In *Managing Cyber Threats*, pages 19–78. Springer, Amsterdam, The Netherlands, 2005.

[101] J.-H. Lee, M.-W. Park, J.-H. Eom, and T.-M. Chung. Multi-level intrusion detection system and log management in cloud computing. In *Advanced Communication Technology (ICACT), 2011 13th International Conference on*, pages 552–555, Seoul, South Korea, 2011. IEEE.

[102] W. Lee, S. J. Stolfo, and K. W. Mok. A data mining framework for building intrusion detection models. In *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, pages 120–132, Oakland, CA, USA, 1999. IEEE.

[103] I. Levin. Kdd-99 classifier learning contest: Llsoft's results overview. *SIGKDD explorations*, 1(2):67–75, 2000.

[104] L. Li and C. N. Manikopoulos. Windows nt one-class masquerade detection. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, pages 82–87, West Point, NY, USA, 2004. IEEE.

[105] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013.

[106] Y. Liao and V. Vemuri. Use of k-nearest neighbor classifier for intrusion detection11an earlier version of this paper is to appear in the proceedings of the 11th usenix security symposium, san francisco, ca, august 2002. *Computers & Security*, 21(5):439 – 448, 2002.

[107] S.-W. Lin, K.-C. Ying, C.-Y. Lee, and Z.-J. Lee. An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection. *Applied Soft Computing*, 12(10):3285–3290, 2012.

[108] W.-C. Lin, S.-W. Ke, and C.-F. Tsai. Cann: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-based systems*, 78:13–21, 2015.

[109] Y. Lin, Y. Zhang, and Y.-j. Ou. The design and implementation of host-based intrusion detection system. In *Intelligent Information Technology and Security Informatics (IITSI), 2010 Third International Symposium on*, pages 595–598, Jinggangshan, China, 2010. IEEE.

[110] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

[111] Y. Liu, K. Chen, X. Liao, and W. Zhang. A genetic clustering method for intrusion detection. *Pattern Recognition*, 37(5):927–942, 2004.

[112] M. Mahoney and P. Chan. An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. In *Recent advances in intrusion detection*, pages 220–237, Amsterdam, The Netherlands, 2003. Springer.

[113] J. McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):262–294, 2000.

[114] V. Mee, T. Tryfonas, and I. Sutherland. The windows registry as a forensic artefact: Illustrating evidence collection for internet usage. *Digital Investigation*, 3(3):166–173, 2006.

[115] Y. Mehmood, U. Habiba, M. A. Shibli, and R. Masood. Intrusion detection system in cloud computing: Challenges and opportunities. In *Information Assurance (NCIA), 2013 2nd National Conference on*, pages 59–66, Rawalpindi, Pakistan, 2013. IEEE.

[116] S. Mehnaz and E. Bertino. Ghostbuster: A fine-grained approach for anomaly detection in file system accesses. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, CODASPY '17, pages 3–14, New York, NY, USA, 2017. ACM.

[117] P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula. Efficient approaches for intrusion detection in cloud environment. In *Computing, Communication and Automation (ICCCA), 2016 International Conference on*, pages 1211–1216, Noida, India, 2016. IEEE.

[118] P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula. Intrusion detection techniques in cloud environment: A survey. *Journal of Network and Computer Applications*, 77:18–47, 2017.

[119] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan. A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*, 36(1):42–57, 2013.

[120] A. Moser, C. Kruegel, and E. Kirda. Limits of static analysis for malware detection. In *Computer security applications conference, 2007. ACSAC 2007. Twenty-third annual*, pages 421–430, Miami Beach, FL, USA, 2007. IEEE.

[121] R. Moskovitch, S. Pluderman, I. Gus, D. Stopel, C. Feher, Y. Parmet, Y. Shahar, and Y. Elovici. Host based intrusion detection using machine learning. In *Intelligence and Security Informatics, 2007 IEEE*, pages 107–114, New Brunswick, NJ, USA, 2007. IEEE.

[122] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection using neural networks and support vector machines. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 2, pages 1702–1707, Honolulu, HI, USA, 2002. IEEE.

[123] D. Mutz, F. Valeur, G. Vigna, and C. Kruegel. Anomalous system call detection. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):61–93, 2006.

[124] G. Nadiammai, S. Krishnaveni, and M. Hemalatha. A comprehensive analysis and study in intrusion detection system using data mining techniques. *International Journal of Computer Applications*, 35(8):51–56, 2011.

[125] J. Newsome and D. Song. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In *In In Proceedings of the 12th Network and Distributed Systems Security Symposium*, page 43, San Diego, California, USA, 2005. Internet Society.

[126] U. of California's San Diego Supercomputer Center. Center for applied internet data analysis, 2018.

[127] T. R. of the University of New Mexico. Sequence-based intrusion detection, 2006.

[128] T. Özyer, R. Alhajj, and K. Barker. Intrusion detection by integrating boosting genetic fuzzy classifier and data mining criteria for rule pre-screening. *Journal of Network and Computer Applications*, 30(1):99–113, 2007.

[129] V. K. Pachghare, V. K. Khatavkar, and P. Kulkarni. Pattern based ids using supervised, semi-supervised and unsupervised approaches. In *International Conference on Computer Science and Information Technology*, pages 542–551, Berlin, Heidelberg, 2012. Springer.

[130] A. Patcha and J.-M. Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470, 2007.

[131] S. Patil, A. Kashyap, G. Sivathanu, and E. Zadok. I3fs: An in-kernel integrity checker and intrusion detection file system. In *LISA*, volume 4, pages 67–78, Berkeley, CA, USA, 2004. USENIX.

[132] S. Peddabachigari, A. Abraham, C. Grosan, and J. Thomas. Modeling intrusion detection system using hybrid intelligent systems. *Journal of Network and Computer Applications*, 30(1):114–132, Jan. 2007.

[133] A. G. Pennington, J. D. Strunk, J. L. Griffin, C. A. Soules, G. R. Goodson, and G. R. Ganger. Storage-based intrusion detection: Watching storage activity for suspicious behavior . In *USENIX Security Symposium*, Berkeley, CA, USA, 2003. USENIX.

[134] Perona., Gurrutxaga, Arbelaitz, Martn, Muguerza, and Prez. gurekddcup database, 2008. `http://aldapa.eus/res/gureKddcup/`.

[135] B. Pfahringer. Winning the kdd99 classification cup: bagged boosting. *ACM SIGKDD Explorations Newsletter*, 1(2):65–66, 2000.

[136] P. A. Porras and R. A. Kemmerer. Penetration state transition analysis: A rule-based intrusion detection approach. In *ACSAC*, pages 220–229, San Antonio, TX, USA, 1992. IEEE.

[137] N. A. Quynh and Y. Takefuji. 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*, pages 194–202, New York, NY, USA, 2007. ACM.

[138] U. Ravale, N. Marathe, and P. Padiya. Feature selection based hybrid anomaly intrusion detection system using k means and rbf kernel function. *Procedia Computer Science*, 45:428–435, 2015.

[139] W. Ren and H. Jin. Distributed agent-based real time network intrusion forensics system architecture design. In *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, volume 1, pages 177–182, Taipei, Taiwan, 2005. IEEE.

[140] J. R. Reuning. Applying term weight techniques to event log analysis for intrusion detection. In *Masters thesis, University of North Carolina at Chapel Hill*, pages 1–60, Chapel Hill, NC, 2004. School of Information and Library Science.

[141] P. J. Rousseeuw and K. V. Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.

[142] F. Sabahi and A. Movaghar. Intrusion detection: A survey. In *Systems and Networks Communications, 2008. ICSNC'08. 3rd International Conference on*, pages 23–26, Sliema, Malta, 2008. IEEE.

[143] M. Sabhnani and G. Serpen. Why machine learning algorithms fail in misuse detection on kdd intrusion detection data set. *Intelligent data analysis*, 8(4):403–415, 2004.

[144] S. K. Sahu, S. Sarangi, and S. K. Jena. A detail analysis on intrusion detection datasets. In *Advance Computing Conference (IACC), 2014 IEEE International*, pages 1348–1353, Gurgaon, India, 2014. IEEE.

[145] K. Scarfone and P. Mell. Guide to intrusion detection and prevention systems (idps). *NIST special publication*, 800(2007):94, 2007.

[146] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo. Data mining methods for detection of new malicious executables. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 38–49, Oakland, CA, USA, 2001. IEEE.

[147] R. Sekar, M. Bendre, D. Dhurjati, and P. Bollineni. A fast automaton-based method for detecting anomalous program behaviors. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 144–155, Oakland, CA, USA, 2001. IEEE.

[148] J. Shavlik and M. Shavlik. Selection, combination, and evaluation of effective software sensors for detecting abnormal computer usage. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 276–285, New York, NY, 2004. ACM.

[149] S. Siddiqui, M. S. Khan, K. Ferens, and W. Kinsner. Detecting advanced persistent threats using fractal dimension based machine learning classification. In *Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics*, IWSPA '16, pages 64–69, New York, NY, USA, 2016. ACM.

[150] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 305–316, Berkeley/Oakland, CA, USA, 2010. IEEE.

[151] A. K. Sood, R. Bansal, and R. J. Enbody. Cybercrime: Dissecting the state of underground enterprise. *Ieee internet computing*, 17(1):60–68, 2013.

[152] S. J. Stolfo, F. Apap, E. Eskin, K. Heller, S. Hershkop, A. Honig, and K. Svore. A comparative evaluation of two algorithms for windows registry anomaly detection. *Journal of Computer Security*, 13(4):659–693, 2005.

[153] Sufatrio and R. H. Yap. Improving host-based ids with argument abstraction to prevent mimicry attacks. In *International Workshop on Recent Advances in Intrusion Detection*, pages 146–164, Berlin, Heidelberg, 2005. Springer.

[154] A. Tajbakhsh, M. Rahmati, and A. Mirzaei. Intrusion detection using fuzzy association rules. *Applied Soft Computing*, 9(2):462–469, 2009.

[155] K. Tan, K. Killourhy, and R. Maxion. Undermining an anomaly-based intrusion detection system using common exploits. In *Recent Advances in Intrusion Detection*, pages 54–73, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[156] K. M. Tan and R. A. Maxion. "why 6?" defining the operational limits of stide, an anomaly-based intrusion detector. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 188–201, Berkeley, CA, USA, 2002. IEEE.

[157] G. Tandon and P. K. Chan. Learning useful system call attributes for anomaly detection. In *FLAIRS Conference*, pages 405–411, Palo Alto, California, 2005. AAAI.

[158] G. Tandon and P. K. Chan. On the learning of system call attributes for host-based anomaly detection. *International Journal on Artificial Intelligence Tools*, 15(06):875–892, 2006.

[159] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pages 1–6, Ottawa, ON, Canada, 2009. IEEE.

[160] T. A. Tchakoucht, M. Ezziyyani, M. Jbilou, and M. Salaun. Behavioral appraoch for intrusion detection. In *Computer Systems and Applications (AICCSA), 2015 IEEE/ACS 12th International Conference of*, pages 1–5, Marrakech, Morocco, 2015. IEEE.

[161] O. P. Team. Open source security, 2017.

[162] X. Tong, Z. Wang, and H. Yu. A research using hybrid rbf/elman neural networks for intrusion detection system secure model. *Computer physics communications*, 180(10):1795–1801, 2009.

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

[164] M. J. M. Turcotte, A. D. Kent, and C. Hash. Unified Host and Network Data Set. *ArXiv e-prints*, abs/1708.07518, Aug. 2017.

[165] C. Vaas and J. Happa. Detecting disguised processes using application-behavior profiling. In *Technologies for Homeland Security (HST), 2017 IEEE International Symposium on*, pages 1–6, Waltham, MA, USA, 2017. IEEE.

[166] K. Veeramachaneni, I. Arnaldo, V. Korrapati, C. Bassias, and K. Li. Ai^2: training a big data machine to defend. In *Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2016 IEEE 2nd International Conference on*, pages 49–54, New York, NY, USA, 2016. IEEE, IEEE.

[167] L. Vokorokos and A. Baláž. Host-based intrusion detection system. In *Intelligent Engineering Systems (INES), 2010 14th International Conference on*, pages 43–47, Las Palmas, Spain, 2010. IEEE.

[168] L. Vokorokos, A. Balaz, and M. Chovanec. Intrusion detection system using self organizing map. *Acta Electrotechnica et Informatica*, 6(1):1–6, 2006.

[169] D. Wagner and R. Dean. Intrusion detection via static analysis. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 156–168, Oakland, CA, USA, 2001. IEEE.

[170] G. Wang, J. Hao, J. Ma, and L. Huang. A new approach to intrusion detection using artificial neural networks and fuzzy clustering. *Expert systems with applications*, 37(9):6225–6232, 2010.

[171] Z. Wang and Y. Zhu. A centralized hids framework for private cloud. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2017 18th IEEE/ACIS International Conference on*, pages 115–120, Kanazawa, Japan, 2017. IEEE.

[172] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, pages 133–145, Oakland, CA, USA, 1999. IEEE.

[173] A. K. Wong and D. K. Chiu. Synthesizing statistical knowledge from incomplete mixed-mode data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(6):796–805, 1987.

[174] M. Xie and J. Hu. Evaluating host-based anomaly detection systems: A preliminary analysis of adfa-ld. In *Image and Signal Processing (CISP), 2013 6th International Congress on*, volume 3, pages 1711–1716, Hangzhou, China, 2013. IEEE.

[175] M. Xie, J. Hu, and J. Slay. Evaluating host-based anomaly detection systems: Application of the one-class svm algorithm to adfa-ld. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2014 11th International Conference on*, pages 978–982, Xiamen, China, 2014. IEEE.

[176] M. Xie, J. Hu, X. Yu, and E. Chang. Evaluating host-based anomaly detection systems: Application of the frequency-based algorithms to adfa-ld. In *International Conference on Network and System Security*, pages 542–549, Berlin, Heidelberg, 2014. Springer.

[177] X. Xu, G. Liu, and J. Zhu. Cloud data security and integrity protection model based on distributed virtual machine agents. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2016 International Conference on*, pages 6–13, Chengdu, China, 2016. IEEE.

[178] N. Ye, S. M. Emran, Q. Chen, and S. Vilbert. Multivariate statistical analysis of audit trails for host-based intrusion detection. *IEEE Transactions on computers*, 51(7):810–820, 2002.

[179] N. Ye, X. Li, Q. Chen, S. M. Emran, and M. Xu. Probabilistic techniques for intrusion detection based on computer audit data. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 31(4):266–274, 2001.

[180] Q. Ye, X. Wu, and B. Yan. An intrusion detection approach based on system call sequences and rules extraction. In *e-Business and Information System Security (EBISS), 2010 2nd International Conference on*, pages 1–4, Wuhan, China, 2010. IEEE.

[181] T. Zhang and R. B. Lee. Cloudmonatt: An architecture for security health monitoring and attestation of virtual machines in cloud computing. In *Computer Architecture (ISCA), 2015 ACM/IEEE 42nd Annual International Symposium on*, pages 362–374, Portland, OR, USA, 2015. IEEE.

[182] Z. Zhang and H. Shen. Application of online-training svms for real-time intrusion detection with different considerations. *Computer Communications*, 28(12):1428–1442, 2005.

[183] G. Zhaojun and W. Chao. Statistic and analysis for host-based syslog. In *Education Technology and Computer Science (ETCS), 2010 Second International Workshop on*, volume 2, pages 277–280, Wuhan, China, 2010. IEEE.

[184] M. Zhu and Z. Huang. Intrusion detection system based on data mining for host log. In *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 1742–1746, Chongqing, China, 2017. IEEE.

[185] Y. Zhu, J. Liang, J. Chen, and Z. Ming. An improved nsga-iii algorithm for feature selection used in intrusion detection. *Knowledge-Based Systems*, 116:74–85, 2017.

Supplementary Section: 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. Here 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 10 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 & Trust (IMPACT) The Department of Homeland Security maintains the IMPACT database[14]. 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, etc., is available freely in this database. Additionally, Digital Corpora provides a research corpus of real data acquired from around the world; however, usage is limited.

DARPA Intrusion Detection 1998, 1999, & 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 [99].

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 comprised of `praudit`[15] 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. [59]

Numerous issues have been documented with this dataset [113, 112, 16]. Brugger and Chow [16] 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. [112]

[14] See `https://www.impactcybertrust.org`.

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

Table 10: 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
DARPA98, 99, 00	Network traffic, System calls	DOS, U2R, R2L, PROBE	http://www.ll.mit.edu/mission/communications/cyb
KDD99	Network traffic	DOS, U2R, R2L, PROBE	https://kdd.ics.uci.edu/databases/kddcup99/kddcu
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://www.unsw.adfa.edu.au/australian-centre-
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	http://www.malware-traffic-analysis.net
NETRESEC	PCAP DBs list	Various	http://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-
CAIDA	PCAP headers, other internet data	Unlabeled	http://www.caida.org/data
Unified Host and Network Dataset	Network & host audit data	Unlabeled	https://csr.lanl.gov/data/2017.html

KDD Cup 1999 (KDD99) This dataset was created by processing the tcpdump[16] 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 [129]. However, evaluating machine learning algorithms, such as DTs [135, 103], NNs [82], and SVMs [40], with KDD99 substantiates that it is not possible to accurately detect

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

U2R and R2L attacks. Sabhnani & Serpen [143] 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. KDD99s record redundancy hinders an algorithms 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. . [159]

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 [144]. The original dataset size is 9.3GB, and the 6% dataset size is 4.2GB. [134]

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" [39]. 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). [127]. 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 [27].

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, 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 LaTeXdocument 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. [27]

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. [57]

Active DNS Project Over a terabyte of "unprocessed DNS packet captures" (PCAPs) along with a plethora of daily de-duplicated DNS records. [94].

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. [17]

NETRESEC Data This data is a list of public packet capture repositories, which are freely available on the Internet. Most of the sites listed below share Full Packet Capture (FPC) files, but some do unfortunately only have truncated frames. This includes SCADA/ICS Network Captures.

CTU 13 The data contains 13 datasets [18], each containing a malware binary, a network flow `.csv` file from the ARGUS flow sensor [19], 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. [47]

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

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 periodic 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[20] 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 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. [126]

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's (LANL) over approximated 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. [164]

9.2 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 11. The table and definitions provided below can be used as a quick reference.

Table 11: DARPA Attack Classes

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

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 tries to gain access. This is commonly achieved through the Internet using password guessing attacks or exploits allowing remote code execution.

[20] See `https://www.bro.org/sphinx/broids/index.html`.