Dataset Report: LID-DS 2021

Martin Grimmer(✉), Tim Kaelble, Felix Nirsberger, Emmely Schulze, Toni Rucks, Jörn Hoffmann, and Erhard Rahm

Leipzig University, Augustuspl. 10, 04109 Leipzig, Germany
{grimmer,kaelble,nirsberger,schulze,rucks,hoffmann,
rahm}@informatik.uni-leipzig.de

Abstract. To advance research on system call based HIDS, we present LID-DS 2021, a recording framework, a dataset for comparative analysis, and a library for evaluating HIDS algorithms.

1 Introduction

Unfortunately, attacks on IT systems are commonplace these days. To avoid major damage, it is important to become aware of an ongoing attack as soon as possible. Intrusion Detection Systems (IDSs) provide timely detection of attacks on IT systems.

According to Debar et al. [4], IDSs are distinguished by the monitored platform and the attack detection method, among others. Network Intrusion Detection Systems (NIDS) use network interactions to monitor the behavior of possibly multiple systems, while Host Based Intrusion Detection Systems (HIDSs) focus on collecting data directly on the host. Though signature-based IDS can make the detection for either NIDS or HIDS of known attacks transparent and comprehensible, they are incapable of detecting novel, previously unknown attacks. Anomaly-based IDS open up the possibility of detecting these. Therefore new methods for anomaly-based attack detection are being researched to keep up with ever-changing developments by attackers. As more and more network communications take place in encrypted form, it is worth taking a closer look at system call-based HIDS, as they can, among other things, directly monitor the impact of those communications on the affected systems.

Modern datasets and comparable results are key factors for the progress of HIDS research. Prior work has largely been based on inadequate datasets, which have also been used in different (incomparable) ways. Some studies did not stick to the partition of training, validation and test data given by the datasets. To make it worse, some of them were not evaluated comparably to each other. For example, by using different definitions of detection and false alarm rates. All this leads to non-comparable results and hinders trustworthy scientific progress in the field of anomaly-based HIDS.

1.1 Our Contribution

For these reasons, we present a new version of the **L**eipzig **I**ntrusion **D**etection **- DataSet**, the **LID-DS 2021** consisting of: an open source framework for

B. Hämmerli et al. (Eds.): CRITIS 2022, LNCS 13723, pp. 63–73, 2023.
https://doi.org/10.1007/978-3-031-35190-7_6

generating HIDS datasets, a modern and comprehensive system call dataset, including the implementation of all 15 of its scenarios and an open source library for evaluating and comparing HIDS algorithms on the given and other datasets. In doing so, we aim to contribute to the anomaly-based HIDS research not only a dataset but also guide other researchers to benefit from existing work and create comparable results in the future.

1.2 Paper Outline

The remainder of this paper is structured as follows: In Sect. 2, we address two questions: What makes a good HIDS dataset and what HIDS datasets are currently available? Then, we introduce the new LID-DS-2021 in Sect. 3. We will discuss bugs that we have fixed, how we generate better data in the new version and provide assistance in processing the data and in evaluating HIDS algorithms with the LID-DS-2021. In Sect. 4, we will compare the LID-DS-2021 with the other datasets presented in Sect. 2 and perform a simple baseline evaluation. In the final section, we summarize this paper and provide directions on which further research with the LID-DS can go.

2 Host Based Intrusion Detection Datasets

To answer the question "What makes a good HIDS dataset?" in this section, we will discuss what we believe to be important technical and then functional features. We will then present the HIDS datasets available so far and briefly refer to their content.

2.1 Preferred System Call HIDS Dataset Features

Technical Features
In our opinion, a HIDS dataset should meet the following technical characteristics to support the widest possible range of HIDS algorithms.

System Call Names: The basic information for system call based HIDS.

Arguments and Return Values: Different system calls have different parameters like quantity specifications such as the number of bytes to be written, identifiers such as file descriptors, strings for path specifications, flags, pointers to memory areas, read bytes, written bytes or error codes. As shown by Wagner and Paolo [15], so-called mimicry attacks can simulate the benignity of system call sequences but not of their arguments and return values. This is why these are not recognized by HIDSs which only analyze the sequences. Therefore this information should be taken into account by HIDS.

Thread Information and Process Ids: This information is important to avoid mixing data from different processes and threads running in parallel, as shown in the work of Pendleton et al. [13] and Grimmer et al. [6].

Timestamps: One should be able to reconstruct the exact sequence of system calls since most algorithms are based on this. In addition, timestamps of the system calls and their distances from each other can be useful features for anomaly detection. Timestamps can also be useful to model periodically occurring behavior. They should be as precise as possible.

User Ids: Invoking certain system calls is part of the normal behavior of a process and a corresponding user. The same system calls executed by the same process but another user may indicate an anomaly or an attack.

Data Buffers: This data may also contain important information. With them, it is even conceivable to learn models based on the unencrypted data transmitted e.g. via HTTPS. To what extent this is recommended and what significance it has for data security and privacy shall remain unanswered here. We are not aware of any previous use of the data buffers in a HIDS.

Network Data: Network data is an equally large and comprehensive data source from which much valuable information can be extracted. A HIDS dataset that also contains the host's network packets allows the development of hybrid IDS by combining system call and network data. In addition, a dataset containing both system calls and network packets allows the comparison of the performance of HIDS and NIDS approaches concerning a single host.

Resource Consumption: The resource consumption of a host can also be used to create a profile of the normal. A too-large or small consumption can indicate anomalies. Therefore, a dataset should also contain information about CPU, memory, network and storage usage.

Functional Features
In addition to the purely technical requirements, we believe the following more functional requirements for a HIDS dataset are just as important.

Real World Data: Ideally, a HIDS dataset contains real-world data. Any kind of simulated data is just simulated in the end. As the work of Arp et al. [1] states: This means that no matter how well prepared a laboratory-simulated dataset may be, it rarely can guarantee that the collected data sufficiently represents the true data distribution of the underlying security problem. But, real-world data also brings difficulties: What about the privacy of the data? Does it contain personal data? These are important points that must not go unnoticed.

Topicality: Software and operating systems change over time. That is why the dataset must represent up-to-date software and operating systems.

Complexity: The scenarios included in the dataset should vary in complexity to represent as many application areas as possible.

Labeled Attacks: The data for a HIDS dataset should be labeled as accurately as possible. Only then is it possible to automatically evaluate and confirm correct classifications, clustering, etc. Ideally, the labels are manually investigated to prevent label inaccuracy as described in [1]. Typically, HIDS datasets contain many recordings, each over a more or less short period of time. Accurate labels here also mean that within such a recording, a distinction is made between which part is normal and which is anomalous, i.e. contains an attack.

Availability: A dataset for development, evaluation, and research for HIDS is only useful if it is also available to as many researchers as possible. Therefore, there should be no barriers (e.g. a payment or login) to accessing the dataset.

Comprehensibility: To be able to explain possible observations in need of clarification in detail, it should be comprehensible how the data of a HIDS dataset was generated. Therefore, a HIDS dataset should publish its source code (as open source) with which it was created.

Expandability: Suppose a HIDS method requires more training, validation, or test data than provided in the dataset for a specific investigation. Ideally, it should be possible to extend a HIDS dataset with more examples of the existing scenarios or even add entirely new ones. In this way, potential gaps in the data can be filled in later.

Support in Processing the Data: To avoid errors or inaccuracies that would lead to incomparable results, a HIDS dataset should provide guidance on how to parse and load the data. Ideally, there are even ready-implemented data loaders that allow researchers to process the dataset without writing custom code to load the data.

2.2 HIDS Datasets

In the previous section, we discussed various technical and functional criteria for assessing HIDS datasets. Based on these criteria, we now want to assess existing system call based HIDS-datasets.

DARPA 1998 and DARPA 1999 Datasets: The DARPA Intrusion Detection Datasets from 1998 [10] and 1999 [11] consist among others of the Basic Security Module (BSM) part. It contains Solaris audit logs in the form of system call sequences, and provides arguments and return values, but no thread IDs. Also, no additional statistics about the system are included. With publication dates from 1998 and 1999, the datasets are also significantly too old to provide sufficient comparability with modern systems.

UNM Dataset: Another IDS dataset was released in 1999 by the University of New Mexico, called the UNM dataset. Just like the Darpa datasets, it is outdated. It has 9 scenarios, differentiated by normal and attack behavior. However, the records only consist of a sequence of process IDs and integer-coded system calls. Thus, neither arguments nor metadata are supplied. The UNM does deliver extensive descriptions of the scenarios but leaves the actual implementation of the recordings in closure and is therefore not expandable.

ADFA-LD: Even more, reduced in its data scope is the 2013 released ADFA-LD from the University of New South Wales (UNSW) in cooperation with the Australian Defense Force Academy (ADFA), which only consists of sequences of integer-coded system calls. Thus, system call parameters, processes and threads are fully ignored. Having only the system call identifiers it also lacks information about system statistics and the recorded software.

ADFA-WD: In 2014 Creech published the ADFA-WD [3], a Windows system call based HIDS dataset. It contains sequences of system calls and additional information such as Process names, PIDs and return values.

NGIDS-DS: Another UNSW cooperation with the ADFA had taken place in the year 2017. The NGIDS-DS contains thread information but lacks parameters and fine granular timestamps. Also, no information about the system resources is existent. Even though Haider et al. provided comprehensive specifications about the recording process of the dataset, the de facto implementation of the recording software is closed source.

AWSCTD: The Attack-Caused Windows OS System Calls Traces Dataset, is a Windows-based HIDS system call dataset published in 2018. Among other things, the paper of Čeponis and Goranin [2] mentioned the following goals of the dataset: Attacks should be publicly available to assure renewal and independent verification, should contain system call names, passed arguments, return values, changed files by attacks and the network traffic generated by attacks. However, the publicly available version[1] of the AWSCTD does not contain any of these data, but only simple system call integer sequences without any further parameters, return values or threads and so on.

LID-DS-2019: In 2019, the first version of the LID-DS [7,14] was released. Since we present a new dataset version in this paper, we name the old dataset LID-DS-2019 to distinguish it from LID-DS-2021, the new one. The publication of the LID-DS-2019 represented an attempt to overcome the problems of the previously described datasets. System calls were recorded with all their parameters, thread IDs, user IDs, payloads and fine granular timestamps. The dataset consists of several up-to-date attack scenarios that were comprehensively recorded and contrasted with benign data to deliver a reliable base for the evaluation of modern anomaly detection systems. Unfortunately, the implementation of the included scenarios had not been published.

3 The New LID-DS

Since we released the original LID-DS, we have been able to do a lot of interesting research with the data it contains. We have received feedback from other users of the dataset and have been able to learn from papers that use it as their data. However, in the process, we noticed a few inaccuracies, errors, and hurdles in the LID-DS-2019. In this section, we will list these and address how we fixed them in the new version of the framework, dataset and library.

[1] https://github.com/DjPasco/AWSCTD - date accessed: October 17, 2022.

3.1 Bugs

Attack Timing: Due to the method used to determine the attack timing, some of them were incorrect. In detail, this was because in LID-DS-2019 the time at which an attack was started on the attacker's system was used as the attack time of the label. Since some attacks in the dataset require a specific startup time, the corresponding effects on the victim can only be found later. In addition, in some rare cases, there are traces where an attack is expected according to the data record, but none occurs. The reason for this is that the attack, due to the large startup time, apparently did not start until after the end of the recording. To fix this problem, in an automatic post-processing step, we searched for the attacker's network packets in the pcap data, matched them with the victim's receiving system calls, and thus found the true start time of the attack on the victim's system.

Databuffer Encoding: The data buffers in LID-DS-2019 contained the first 80 bytes of all data buffers passed to system calls. This made it possible to analyze e.g. parts of SQL statements or other data. However, these data buffers could also contain non-printable byte values. All these values were encoded as "." (dot) in the records in LID-DS-2019. That's why in the new version the data buffers are encoded as base64 strings. In this way, we prevent potentially important information from being lost. Nevertheless, we maintain a human-readable encoding for the rest of the data.

3.2 Better Data

Distribution of User Actions: In the LID-DS-2019, almost all user behaviors had the same temporal distribution. This was modeled after the paper by Deng. [5] We changed this in the new version. Now user behavior is sampled from real-world web-server log files. Thus the scenarios are still not real-world data, but at least their access timestamps are generated from real-world data.

System Resources: Since the LID-DS-2019 did not contain information about consumed system resources, we have fixed this in the new version. We use Docker's built-in method to query (victim) container resource usage statistics to get them once per second. So for all its records, the LID-DS-2021 contains the following values once per second: CPU usage, memory usage, network received, network send, storage read and storage written.

Network Data: A HIDS that not only considers the behavior of the system to be monitored but also the communication of the system, can achieve better results. Therefore, in addition to the system calls and system resources, we also recorded the network communication to and from the victim in pcap format using tcpdump. This even allows HIDS and NIDS to be compared on the dataset in terms of their detection and error rates.

IP Range: In LID-DS-2019, the attacker and all normal users had the same IP because they were running as scripts on the recording host. As a result, all network communication took place only between the host that performed the recording and the victim. While this is not necessarily unrealistic, it is also not a standard use case. With LID-DS-2021, however, all users and the attacker are each running from their own Docker containers with their own IP addresses. As a result, there are no longer only connections to one other host in the data.

Multi Step Attacks: All scenarios of LID-DS-2019 have a perfect attacker, i.e. one who directly attacks the vulnerability without analyzing the system beforehand. In reality, this is usually not the case. Therefore, for the new version, we have also designed scenarios that include not only the attack itself, but also the steps of preparation and several exploited vulnerabilities.[2] For example, we describe scenario CVE-2017-12635_6[3] which implements vulnerabilities CVE-2017-12635 and CVE-2017-12636 in CouchDB up to version 1.7.0. In this scenario, the attacker performs several steps in succession that a real attacker could also perform. First, he performs a port scan using Nmap. Then, in an optional second step, he performs a brute-force attack using hydra in combination with the rockyou password list to log in to the database. In the next step, using CVE-2017-12635, a privilege escalation technique, the attacker changes the database user's privilege to give them administrator rights. Finally, a remote code execution occurs using CVE-2017-12636 to open a reverse shell. It is worth mentioning that each of these steps is labeled in the LID-DS-2021.

Source Code: With the original LID-DS, we released the recording framework as open source. This allowed new scenarios to be added to the dataset. However, it was not possible to create more recordings of existing scenarios with this, as the source code of the scenarios was not included. Therefore, in addition to the source code of the new framework, we also released the source codes of all 15 scenarios of the new dataset as open source. This makes it possible to create new scenarios, extend existing ones, and also perform in-depth analyses where the calling code from the attacker, normal user or victim is necessary.

3.3 LID-DS Library

Establishing the comparability of different research results is not an easy task. On the one hand, there are papers like [12] that do not adhere to the division of the data into training, validation and testing as specified in the datasets.

[2] A list of the LID-DS-2021 scenarios including their description, classification by simple/multi-step, and their source code can be found at https://github.com/LID-DS/LID-DS/wiki/Scenarios.

[3] Common Vulnerabilities and Exposures (CVE): a reference-method for publicly known information-security vulnerabilities and exposures. See: https://cve.mitre.org/.

On the other hand, not all studies use the same methodology to determine the performance metrics such as detection rate, false positives and so on. It happens that some works evaluate their algorithms for entire records as the work of Wunderlich et al. [16], although both normal and abnormal ranges can exist within a record as shown in the paper from Grimmer et al. [6]. As a consequence, results are not comparable even if they would use the same dataset. We have addressed this issue with the LID-DS-2021.

First, along with the dataset, we released a Python library that contains a data loader that can be used by researchers to load data from different HIDS datasets and encourages them to stick to the dataset division of training, validation and testing. At the moment, the following datasets are supported: ADFA-LD, LID-DS-2019, and LID-DS-2021 and it provides an easy-to-understand interface with functions like `training_data()`, `validation_data()` or `test_data()`. Second, in addition to the data loader, using a system of so-called building blocks, existing features and algorithms can be freely combined to build complex HIDS algorithms, as indicated in Fig. 1. A building block represents one step in a machine learning pipeline like extracting a feature from a system call, calculating an embedding, n-gram or an anomaly score using an autoencoder. On top, new features and algorithms can be added simply by implementing the existing interfaces. Thirdly, the full process starting with loading, feature extraction, anomaly detection to evaluation can be performed automatically using our library.[4] The method used in the paper by Grimmer et al. [6] for system call accurate evaluation is applied here.

4 Evaluation

We conduct an initial, simple evaluation of the LID-DS-2021 by comparing the dataset in tabular form with the other datasets presented in this paper. Then we will give information about its size and performance regarding a baseline algorithm.

As can be easily seen from Table 1, the LID-DS-2021 beats all other HIDS datasets in terms of the requirements we worked out before. The number of system calls and the average baseline performance over all scenarios using the STIDE algorithm [9] implemented with our library can be seen in Fig. 2. As the lower F-score, lower detection rate, and higher number of false alarms show, the 2021 version is slightly more difficult to solve with the base algorithm than the 2019 version. A code snippet for this baseline evaluation can be seen in Listing 1.1. The large amount of system call data including the mentioned system call attributes, network data and resource consumptions should allow many different types of HIDS to be developed and evaluated. From simple sequence-based

[4] Example code: https://github.com/LID-DS/LID-DS/wiki/ids_example.

Table 1. Datasets evaluated with respect to the presented criteria. Ratings: 0 = feature not included, 0.5 = feature partially included, 1 = feature included.

	DARPA	UNM	ADFA-LD	ADFA-WD	NGIDS-DS	AWSCTD	LID-DS-2019	LID-DS-2021
technical features								
system calls	1	0.5	0.5	1	1	0.5	1	1
arguments	1	0	0	0.5	0	0	1	1
thread info	0.5	0.5	0	0.5	0.5	0	1	1
timestamps	0.5	0	0	1	0.5	0	1	1
user ids	1	0	0	0	0	0	1	1
data buffers	0	0	0	0	0	0	0.5	1
network data	1	0	0	0	1	0	0	1
resource consumption	0	0	0	0	0	0	0	1
functional features								
real-world data	0	0	0	0	0	0	0	0
topicality	0	0	0	0	1	1	1	1
complexity	0	0	1	1	1	1	1	1
labeled attacks	0.5	0.5	0.5	0.5	1	0.5	1	1
availability	0	1	1	1	1	1	1	1
comprehensibility	0	0	0	0	0	0	1	1
expandability	0	0	0	0	0	0	0.5	1
support in processing	0	0	0	0	0	0	0	1
sum	5.5	2.5	3	5.5	7	4	11	15

algorithms over sophisticated ML methods to provenance-based graph solutions like Unicorn [8]. The LID-DS-2021 enables this and perhaps completely novel methods that take into account all the information contained in the dataset.

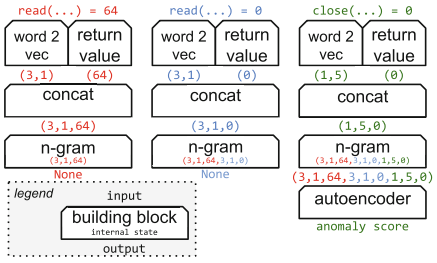

Fig. 1. Example usage of Building Blocks (BBs): Input (from left to right) is a stream of system calls (read, read, close) including their return values. Here, the BBs are combined to create n-grams with $n = 3$ of embedded system calls and their return values, which are then input to an autoencoder for anomaly detection.

LID-DS-2019		LID-DS-2021	
Scenario	**# Syscalls in million**	**Scenario**	**# Syscalls in million**
CVE-2012-2122	5.7	**CVE-2012-2122**	20.7
CVE-2014-0160	4.0	**CVE-2014-0160**	1.9
CVE-2017-7529	1.8	**CVE-2017-7529**	1.3
		CVE-2017-12635_6	1311.7
CVE-2018-3760	19.2	**CVE-2018-3760**	115.1
CVE-2019-5418	18.0	**CVE-2019-5418**	400.9
		CVE-2020-9484	223.6
		CVE-2020-13942	849.1
		CVE-2020-23839	33.9
Bruteforce	5.7	**Bruteforce**	9.5
EPS_CWE-434	126.2	**EPS_CWE-434**	296.3
		Juice-Shop	484.9
PHP_CWE-434	22.2	**PHP_CWE-434**	97.5
SQL_Injection	23.6	**SQL_Injection**	96.2
ZipSlip	252.1	**ZipSlip**	111.1
F-score	0.63	**F-score**	0.57
detection rate	0.65	**detection rate**	0.59
avg. false alarms	19.80	**avg. false alarms**	24.47

Fig. 2. Number of system calls in scenarios of the LID-DS and average results of the baseline algorithm STIDE over all scenarios, with $n = 7$ and $w = 100$.

```
# map each system call to an integer
int_embedding = IntEmbedding()
# build ngrams from these integers
ngram = Ngram([int_embedding], False, ngram_length=7)
# calculate the STIDE algorithm using these ngrams
stide = Stide(ngram)
# build a sum over a stream window of length 100
stream_sum = StreamSum(stide, window_length=100)
# use our IDS class for automatic evaluation
ids = IDS(... , resulting_building_block=stream_sum, ...)

```

Listing 1.1. Baseline HIDS Algorithm (STIDE) using LID-DS library

5 Summary

In this paper, we presented the LID-DS-2021. A recording framework, a dataset and a library for system-call-based HIDS. We have shown which alternatives are currently available and that these are not sufficient or have errors. We described how we fixed the bugs of the LID-DS in the new version and briefly touched on the new possibilities of the associated library. As a result, the LID-DS-2021 again takes research for system call-based HIDS a step further. The data, source code, examples and documentation of the LID-DS are available via GitHub[5].

Acknowledgement. This work was supported by the German Federal Ministry of Education and Research(BMBF, 01IS18026B) by funding the competence center for Big Data and AI "ScaDS.AI" Dresden/Leipzig.

References

1. Arp, D., et al.: Dos and don'ts of machine learning in computer security. In: Proceedings of the USENIX Security Symposium (2022)
2. Čeponis, D., Goranin, N.: Towards a robust method of dataset generation of malicious activity for anomaly-based HIDS training and presentation of AWSCTD dataset. Baltic J. Modern Comput. **6**(3), 217–234 (2018)
3. Creech, G.: Developing a high-accuracy cross platform host-based intrusion detection system capable of reliably detecting zero-day attacks, Ph. D. thesis, UNSW Sydney (2014)
4. Debar, H., Dacier, M., Wespi, A.: Towards a taxonomy of intrusion-detection systems. Comput. Netw. **31**(8), 805–822 (1999)
5. Deng, S.: Empirical model of www document arrivals at access link. In: Proceedings of ICC/SUPERCOMM1996-International Conference on Communications, vol. 3, pp. 1797–1802. IEEE (1996)
6. Grimmer, M., Kaelble, T., Rahm, E.: Improving host-based intrusion detection using thread information. In: Meng, W., Katsikas, S.K. (eds.) EISA 2021. CCIS, vol. 1403, pp. 159–177. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-93956-4_10

[5] https://github.com/LID-DS/LID-DS.

7. Grimmer, M., Röhling, M.M., Kreusel, D., Ganz, S.: A modern and sophisticated host based intrusion detection data set. IT-Sicherheit als Voraussetzung für eine erfolgreiche Digitalisierung, pp. 135–145 (2019)
8. Han, X., Pasquier, T., Bates, A., Mickens, J., Seltzer, M.: Unicorn: runtime provenance-based detector for advanced persistent threats. arXiv preprint arXiv:2001.01525 (2020)
9. Hofmeyr, S.A., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. J. Comput. Secur. **6**(3), 151–180 (1998)
10. MIT Lincoln Laboratory: 1998 darpa intrusion detection evaluation data set. https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset (1998). Accessed 10 Mar 2022
11. MIT Lincoln Laboratory: 1999 darpa intrusion detection evaluation data set. https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset (1998). Accessed 10 Mar 2022
12. Park, D., Kim, S., Kwon, H., Shin, D., Shin, D.: Host-based intrusion detection model using Siamese network. IEEE Access **9**, 76614–76623 (2021)
13. Pendleton, M., Xu, S.: A dataset generator for next generation system call host intrusion detection systems. In: MILCOM 2017–2017 IEEE Military Communications Conference (MILCOM), pp. 231–236. IEEE (2017)
14. Röhling, M.M., Grimmer, M., Kreubel, D., Hoffmann, J., Franczyk, B.: Standardized container virtualization approach for collecting host intrusion detection data. In: 2019 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 459–463. IEEE (2019)
15. Wagner, D., Soto, P.: Mimicry attacks on host-based intrusion detection systems. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 255–264 (2002)
16. Wunderlich, S., Ring, M., Landes, D., Hotho, A.: Comparison of system call representations for intrusion detection. In: Martínez Álvarez, F., Troncoso Lora, A., Sáez Muñoz, J.A., Quintián, H., Corchado, E. (eds.) CISIS/ICEUTE -2019. AISC, vol. 951, pp. 14–24. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-20005-3_2