Extended Abstract: LID-DS 2021

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

all authors: Leipzig University, Augustuspl. 10, 04109 Leipzig, Germany
{lastname}@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.

Modern datasets and comparable results are key factors for the progress of HIDS research. Previous work was based mainly on insufficient datasets that were not used consistently [1, 2, 4]. To make matters worse, some of them were not evaluated comparably to each other. All this hinders trustworthy scientific progress in the field of anomaly-based HIDS. For this reason, we present a new version of the **L**eipzig **I**ntrusion **D**etection **- D**ata**S**et, the **LID-DS 2021** consisting of (1) an open source framework for generating HIDS datasets, (2) a modern and comprehensive system call dataset, the including the implementation of all 15 of its scenarios and (3) 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.

The **framework** provides interfaces for defining scenarios. A scenario describes an environment consisting of three roles: *Attacker*, *Victim* and *User* (with benign behavior and timings sampled from real-world webserver logs). Each of them is defined as a Docker container and executed by the framework. Records can then be automatically created containing the system calls (incl. arguments, return values and other metadata), network data and other statistics of the victim. A schematic representation can be seen in Figure 1.

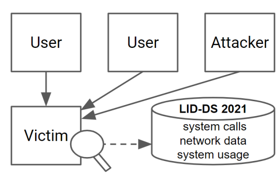

Fig. 1: Schematic representation of a recording.

The **dataset** consists of the source code and recordings of 15 scenarios corresponding to real security vulnerabilities. We presented the size of each scenario from the 2019 and 2021 versions of LID-DS in Figure 2. The new one contains about 8.5 times as many system calls as the old one.

The **library** enables the uniform loading of HIDS datasets.[1] In addition, the complete process from loading, feature extraction, anomaly detection to evaluation can be performed automatically. Using a system of so-called building

[1] currently supported: LID-DS 2019 and LID-DS 2021

blocks, existing features and algorithms can be freely combined to build complex HIDS algorithms, as indicated in Figure 3. On top, new features and algorithms can be added simply by implementing the existing interfaces.

For an **initial evaluation**, we run a variant (as described in [1]) of the STIDE [3] algorithm as a baseline on both the LID-DS 2019 and the LID-DS 2021 and present the results in Figure 2. As the lower F-score, lower detection rate, and higher number of false alarms show, the 2021 version is more difficult to solve with the base algorithm than the 2019 version.

The data, source code and documentation of the LID-DS are available via GitHub[2]. Example algorithms can also be found there.

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 $w = 100$.

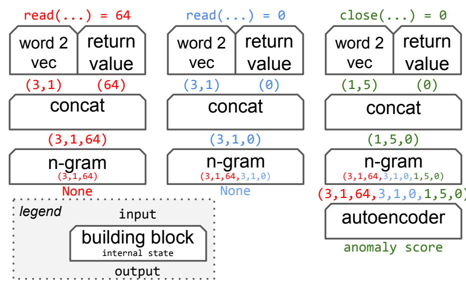

Fig. 3: Example usage of Building Blocks (BBs): Input (from left to right) is a stream of system calls (read, read, close) including their return values. Here, the BBs are combined to create n-grams with $n = 3$ of embedded system calls and their return values, which are then input to an autoencoder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

References

1. Grimmer, M., Kaelble, T., Rahm, E.: Improving host-based intrusion detection using thread information. In: International Symposium on Emerging Information Security and Applications. pp. 159–177. Springer (2021)
2. 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)
3. Hofmeyr, S.A., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. Journal of computer security **6**(3), 151–180 (1998)
4. 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)

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