Poster: VULCAN – Repurposing Accessibility Features for Behavior-based Intrusion Detection Dataset Generation

Christian van Sloun
sloun@comsys.rwth-aachen.de
RWTH Aachen University
Aachen, Germany

Klaus Wehrle
wehrle@comsys.rwth-aachen.de
RWTH Aachen University
Aachen, Germany

ABSTRACT

The generation of datasets is one of the most promising approaches to collecting the necessary behavior data to train machine learning models for host-based intrusion detection. While various dataset generation methods have been proposed, they are often limited and either only generate network traffic or are restricted to a narrow subset of applications. We present VULCAN, a preliminary framework that uses accessibility features to generate datasets by simulating user interactions for an extendable set of applications. It uses behavior profiles that define realistic user behavior and facilitate dataset updates upon changes in software versions, thus reducing the effort required to keep a dataset relevant. Preliminary results show that using accessibility features presents a promising approach to improving the quality of datasets in the HIDS domain.

CCS CONCEPTS

• **Security and privacy → Intrusion/anomaly detection and malware mitigation**.

KEYWORDS

Intrusion Detection, Dataset Generation, Accessibility Features.

ACM Reference Format:
Christian van Sloun and Klaus Wehrle. 2023. Poster: VULCAN – Repurposing Accessibility Features for Behavior-based Intrusion Detection Dataset Generation. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3576915.3624404

1 INTRODUCTION

With human security monitoring capabilities quickly reaching their limits due to the increasing number of cyber attacks, organizations rely on Intrusion Detection Systems (IDSs) based on Machine-Learning (ML) to deal with the number of potential alerts and focus the attention of personnel to only deal with security-relevant events. However, training such systems relies on high-quality datasets to distinguish between benign and malicious behavior. This raises two critical questions: (i) where does the training data come from, and (ii) how to judge whether the training data is useful?

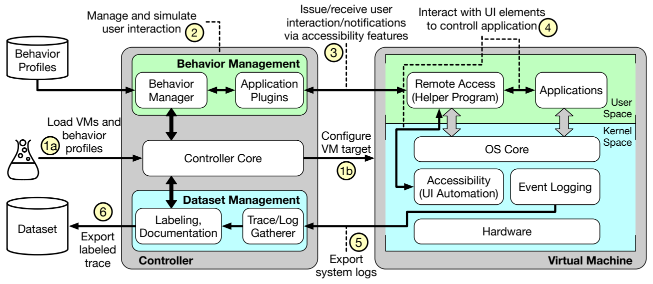


Figure 1: VULCAN's architecture design.

To improve the performance of IDSs, researchers rely on public datasets as these allow objective comparison of different approaches, and several datasets for both Network IDSs (NIDSs) and Host IDSs (HIDSs) have been proposed over the years [3, 5, 14]. However, especially for HIDSs, relevant datasets are equally often criticized as being unrealistic [5, 10], outdated [3, 5, 6], containing errors [1, 4], or simply missing features that are required for a new IDS approach [5]. Consequently, significant effort has gone into developing realistic datasets [5, 8, 14] and dataset generation methods [2, 9, 13]. We observe a significant bias toward generating realistic network traffic for NIDS use [11, 13, 14]. At the same time, the generation methods of existing datasets, or mechanisms designed to create new datasets for HIDS use, presently rely on rudimentary tools, *e.g.*, network services [3, 5, 8] or remote access and dedicated Application Programming Interfaces (APIs) [2, 9], that only support a narrow subset of applications and behavior.

We focus on extending dataset generation methods to a wide range of applications, making the generation of host behavior more representative of real-world environments and more maintainable. We propose to repurpose accessibility features already present in most Operating Systems (OSs) to simulate the system interaction by real users that provide realistic system behavior and use this to derive system behavior in a reproducible and repeatable way. We build upon the intuition that most system behavior ultimately originates from human input, either directly, *e.g.*, via a Graphical User Interface (GUI), the network, *e.g.*, by users interacting through a web interface, or other applications requiring network connectivity. The same holds for most malicious behavior, *e.g.*, internal attacks or attacks carried out via the network.

Therefore, we design VULCAN, a novel approach to generating host activity based on predefined user behavior by repurposing accessibility features as a universal control method to simulate human interaction with applications and systems.

2 VULCAN

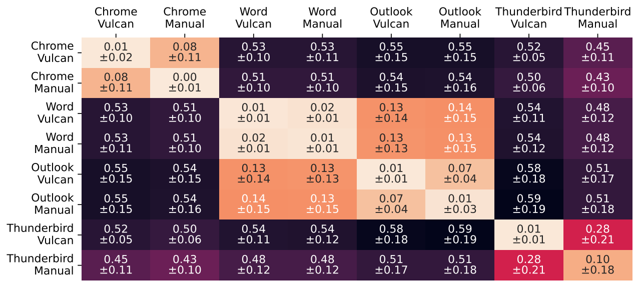
VULCAN generates IDS datasets based on user-behavior models to overcome the limitations of previous approaches w.r.t. behavior generation for arbitrary applications. The focus of Vulcan is facilitating the generation of IDS datasets by generating realistic system traces, *e.g.*, system calls or logs, or network data, by repurposing accessibility features to simulate the interaction of human users with applications. By enabling direct interaction with GUI elements, accessibility features render the need for dedicated remote control APIs unnecessary and promise to support a wide range of OSs and applications. Since accessibility features mimic human interaction, they are ideal for realizing programmatic control over various (GUI-based) applications. They can simulate human-like interactions and enable Vulcan to generate realistic and comprehensive datasets that closely model real-world environments.

Fig. 1 visualizes the general architecture of Vulcan. It uses a database of *behavioral profiles* describing a user's interaction with an application as the basis for its simulation. Each profile defines sequences of interactions with applications or how the simulated user reacts to external events, *e.g.*, receiving a notification. As accessibility features use unique identifiers or the position of UI elements in an abstract hierarchy of UI elements, we add an abstraction layer by introducing the concept of application-specific plugins that allow behavior profiles to work on a more abstract description of user interaction. For example, instead of defining the specific sequence of interactions with UI elements required to save a file, it allows specifying such tasks by their intention, *i.e.*, saving the file.

The main components of Vulcan are a *controller* and Virtual Machines (VMs) serving as the end hosts generating host data. To generate new datasets, the controller loads a description of the available VMs and behavior profiles (Step 1a), which act as blueprints for the system traces, *i.e.*, the recorded host activity, that is generated when simulating the user's behavior on the target machine. The controller configures the VMs required for the planned execution (Step 1b) by installing necessary applications and configuring logging. Subsequently, Vulcan establishes a control channel to allow the controller access to the internally available accessibility features of the OS running in the VM. A subcomponent of the controller, the *behavior management* (Step ②), manages the current step in the behavior profiles for each virtual user and decides which actions are executed next.

Furthermore, the *behavior management* interprets events, *e.g.*, notifications, that users would react to and adapts the planned interactions accordingly. The plugins then translate the interaction into application-specific UI actions (Step ③) and react to potential prompts, *e.g.*, error messages, by interpreting the result and handing control back to the *Behavior Manager*. An OS-specific helper program, started inside each VM, accesses the accessibility interface of the OS (Step ④) and returns relevant information about running processes, available windows, and UI elements to the controller. Further, it receives the instructions generated by the applications-specific plugins and executes the requested interactions using available accessibility features. During execution, each VM streams relevant system information as traces to the controller (Step ⑤), where the data is collected.

Additionally, the controller uses the information from the Behavior Manager to label the traces by associating the received log data to the behavior profile and its current state of execution in the corresponding VM. Finally, the traces are exported (Step ⑥), potentially compressed, and stored as part of the new dataset.



	Chrome Vulcan	Chrome Manual	Word Vulcan	Word Manual	Outlook Vulcan	Outlook Manual	Thunderbird Vulcan	Thunderbird Manual
Chrome Vulcan	0.01 ±0.02	0.08 ±0.11	0.53 ±0.10	0.53 ±0.11	0.55 ±0.15	0.55 ±0.15	0.52 ±0.05	0.45 ±0.11
Chrome Manual	0.08 ±0.11	0.00 ±0.01	0.51 ±0.10	0.51 ±0.10	0.54 ±0.15	0.54 ±0.16	0.50 ±0.06	0.43 ±0.10
Word Vulcan	0.53 ±0.10	0.51 ±0.10	0.01 ±0.01	0.02 ±0.01	0.13 ±0.14	0.14 ±0.15	0.54 ±0.11	0.48 ±0.12
Word Manual	0.53 ±0.11	0.51 ±0.10	0.02 ±0.01	0.01 ±0.01	0.13 ±0.13	0.13 ±0.15	0.54 ±0.12	0.48 ±0.12
Outlook Vulcan	0.55 ±0.15	0.54 ±0.15	0.13 ±0.14	0.13 ±0.13	0.01 ±0.01	0.07 ±0.04	0.58 ±0.18	0.51 ±0.17
Outlook Manual	0.55 ±0.15	0.54 ±0.16	0.14 ±0.15	0.13 ±0.15	0.07 ±0.04	0.01 ±0.03	0.59 ±0.19	0.51 ±0.18
Thunderbird Vulcan	0.52 ±0.05	0.50 ±0.06	0.54 ±0.11	0.54 ±0.12	0.58 ±0.18	0.59 ±0.19	0.01 ±0.01	0.28 ±0.21
Thunderbird Manual	0.45 ±0.11	0.43 ±0.10	0.48 ±0.12	0.48 ±0.12	0.51 ±0.17	0.51 ±0.18	0.28 ±0.21	0.10 ±0.18

Figure 2: Comparison of the symmetric Word Mover's Distance and standard deviation between Vulcan and manually generated traces.

3 EVALUATION

The design of Vulcan hinges on the assumption that accessibility features, part of common OSs, allow for the automated generation of realistic datasets for HIDS. To evaluate the quality of datasets generated via accessibility features, we implemented a prototype for Windows 11 that supports programmatic control of Google Chrome, Microsoft Word, Microsoft Outlook, and Mozilla Thunderbird. We used Sysmon and adapted the `sysmonconfig-trace` configuration by F. Roth [12] to log security-relevant events.

Behavior Profiles. We evaluated our framework through three scenarios that are conceivable workflows performed by employees interacting with their workstations: (i) Using a web browser to request a webpage, navigating to several subsequent pages via hyperlinks before closing the application. (ii) Using the OS's file explorer to open a Microsoft Word document, edit the document, and save changes back to disk. (iii) Receiving and reading the newest email and forwarding it to a new recipient. The last scenario is repeated for both Microsoft Outlook and Thunderbird.

Evaluation Metrics. As system traces originate from an application executing its program code, they inherit a structure defined by the program flow. Thus, they can be represented as a *sentence* containing system events. To compare the dissimilarity of the resulting traces, we use the Word Mover's Distance (WMD) [7] since it measures the dissimilarity between two sentences. Therefore, to objectively quantify the difference between Vulcan and manually performed executions of the same workflow, we use a `word2vec` model trained on 100 Vulcan traces from each scenario and compare 30 traces produced by Vulcan to 30 traces that are performed by hand. Additionally, we split the preprocessed corpus of the Vulcan-generated traces into training, test, and validation sets.

We aim to study whether dataset traces generated by Vulcan resemble *realistic* behavior and if the proposed control method via accessibility features introduces anomalies in the recorded system behavior. As each trace consists of sentences representing the execution of each process, we measure the dissimilarity between two traces by computing the WMD for matching processes contained in both traces. When comparing traces for which one contains a process that has no equivalent in the other trace, *e.g.*, a process checking for updates, no direct comparison can be performed. In such cases, we compute the minimum WMD of the process in question and all processes in the other trace. Using the maximum would produce a high dissimilarity between disjunct applications but also would not be meaningful when comparing the same application.

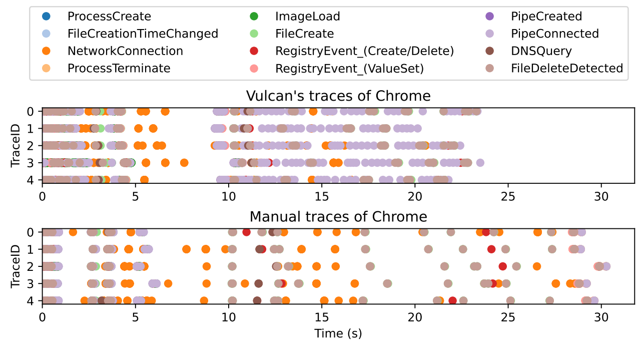


Figure 3: The temporal structure of different events across five randomly chosen Google Chrome traces.

Preliminary Results. We compare Vulcan and manual traces for all four applications to evaluate whether Vulcan introduces behavioral anomalies and compute the average dissimilarity and standard deviation across compared traces. Fig. 2 visualizes the dissimilarity of Vulcan and manual traces across all four applications. The comparison shows a low dissimilarity (0 denotes that traces are identical) between traces generated using Vulcan and also between manually generated traces, implying that the applications behave consistently. The only exception, the manually generated traces of Thunderbird, show an increase in dissimilarity. When comparing manual traces and those generated by Vulcan, a slight increase in the average dissimilarity and standard deviation can be observed for Google Chrome, Microsoft Word, and Outlook. However, Thunderbird shows a significant increase that is caused by the already higher dissimilarity between the manual traces, affecting the comparison with Vulcan traces. Comparing applications reveals that they generate markedly distinct traces. Notably, Microsoft Word and Outlook show a lower degree of dissimilarity, which the shared code basis of these applications can explain.

The comparison, especially for Microsoft Word, shows that Vulcan produces realistic traces indistinguishable from traces generated by a human user. Nevertheless, there are slight variations for other applications. A closer look at Google Chrome and Microsoft Outlook shows very low dissimilarity between manual and Vulcan traces but an increased dissimilarity when comparing them. Upon inspection (*cf.* Fig. 3) of different Sysmon events across a subset of five Vulcan/manual Chrome traces, the Vulcan traces contain "PipeConnected" events that cannot be found in manual traces. This results from how MS UI Automation provides access to accessibility features for applications as it connects named pipes to each accessed process. Due to how Google Chrome's GUI framework operates (multiple processes are started to perform rendering tasks), many such events are generated and logged by our Sysmon configuration. However, using named pipes, the associated events can easily be identified. Sysmon can be configured to ignore these events, improving the dissimilarity between manual and Vulcan Chrome traces to 0.059 ± 0.068. Thunderbird's dissimilarity is caused by telemetry transmissions to Mozilla, which appear in 27 of 30 manual executions. Interestingly, all Vulcan and three manual traces do not show this behavior, which indicates that outside influences may cause this. Excluding these processes improves the dissimilarity between Vulcan and manual Thunderbird traces to 0.095 ± 0.040.

4 CONCLUSION AND FUTURE WORK

Our results show that Vulcan can accurately simulate human interaction with minimal effect on the operation of the controlled application. Furthermore, effects introduced by Vulcan can be identified and filtered out, demonstrating its ability to simulate realistic system activity. However, our work is still preliminary, and we are extending it to investigate several specifics:

Universality. Firstly, while application-specific plugins promise to ease the adaption of Vulcan to additional OSs and applications, the effort required for broader applicability warrants further analysis.

Scalability. Second, using VMs that are coordinated via the network promises scalability. However, quantifying the overhead introduced by accessibility features when controlling complex user behaviors requires a dedicated performance evaluation.

Realistic user/attacker behavior. Third, while it is possible to define behavior manually, to make a synthetic dataset resemble the real world, methods deriving behavior from real-world or red-team activity would significantly reduce the effort to generate large-scale datasets. We believe that accessibility features could also be a promising approach to *record* how a user interacts with a system and allow for easy generation of behavior profiles.

Quality of generated datasets. Finally, to assess the quality of datasets generated by Vulcan, it is necessary to evaluate the generated dataset against an IDS. We plan to extend our evaluation by generating a large-scale dataset containing user behavior.

Despite the items listed above, we believe that with Vulcan, we present a promising approach to improving the quality of datasets in the HIDS domain due to its ability to simulate users interacting with systems realistically. We believe realistic datasets to become increasingly relevant to improve the ability of IDS to detect internal attackers or advanced persistent threats.

REFERENCES

[1] M. M Anjum et al. 2021. Analyzing the Usefulness of the DARPA OpTC Dataset in Cyber Threat Detection Research. In *ACM SACMAT*.
[2] H.-K Bui et al. 2021. CREME: A Toolchain of Automatic Dataset Collection for Machine Learning in Intrusion Detection. *J. Netw. Comput. Appl.* (2021).
[3] G Creech et al. 2013. Generation of a New IDS Test Dataset: Time to Retire the KDD Collection. In *IEEE WCNC*.
[4] G Engelen et al. 2021. Troubleshooting an Intrusion Detection Dataset: The CICIDS2017 Case Study. In *IEEE SPW*.
[5] M Grimmer et al. 2019. A Modern and Sophisticated Host Based Intrusion Detection Data Set. In *IT-Sicherheit als Voraussetzung für eine erfolgreiche Digitalisierung*.
[6] W Haider et al. 2017. Generating Realistic Intrusion Detection System Dataset Based on Fuzzy Qualitative Modeling. *J. Netw. Comput. Appl.* (2017).
[7] M Kusner et al. 2015. From Word Embeddings to Document Distances. In *Proceedings of the 32nd International Conference on Machine Learning*.
[8] M Landauer et al. 2022. Maintainable Log Datasets for Evaluation of Intrusion Detection Systems. *IEEE Trans. Dependable Secure Comput.* (2022).
[9] M Landauer et al. 2021. Have It Your Way: Generating Customized Log Datasets With a Model-Driven Simulation Testbed. *IEEE Trans. Reliab.* (2021).
[10] J McHugh. 2000. Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. *ACM Trans. Inf. Syst. Secur.* (2000).
[11] M Ring et al. 2017. Flow-Based Benchmark Data Sets for Intrusion Detection. In *ECCWS*.
[12] F Roth. 2023. Sysmon-Config | A Sysmon Configuration File. https://github.com/Neo23x0/sysmon-config Accessed: 2023-08-17.
[13] I Sharafaldin et al. 2018. Towards a Reliable Intrusion Detection Benchmark Dataset. *Software Networking* (2018).
[14] I Sharafaldin et al. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization:. In *ICISSP*.