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.

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?

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

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

CCS ’23, November 26–30, 2023, Copenhagen, Denmark
© 2023 Copyright held by the owner/author(s).
https://doi.org/10.1145/3576915.3624404

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 1), 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 2) 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 3), 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 4) 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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WMD</strong></td>
<td>0.21 ± 0.02</td>
<td>0.37 ± 0.01</td>
<td>0.43 ± 0.02</td>
<td>0.50 ± 0.01</td>
<td>0.54 ± 0.01</td>
<td>0.56 ± 0.01</td>
<td>0.57 ± 0.01</td>
<td>0.58 ± 0.01</td>
</tr>
<tr>
<td><strong>±0.11</strong></td>
<td>0.32 ± 0.11</td>
<td>0.53 ± 0.11</td>
<td>0.62 ± 0.12</td>
<td>0.69 ± 0.12</td>
<td>0.74 ± 0.12</td>
<td>0.77 ± 0.12</td>
<td>0.80 ± 0.12</td>
<td>0.82 ± 0.12</td>
</tr>
<tr>
<td><strong>±0.12</strong></td>
<td>0.37 ± 0.11</td>
<td>0.60 ± 0.11</td>
<td>0.69 ± 0.12</td>
<td>0.78 ± 0.12</td>
<td>0.83 ± 0.12</td>
<td>0.86 ± 0.12</td>
<td>0.90 ± 0.12</td>
<td>0.93 ± 0.12</td>
</tr>
</tbody>
</table>

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 disjointed applications but also would not be meaningful when comparing the same application.
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 adoption 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


