A Foundation for Orchestrating Multiple Security Environments in Endpoint Systems

By:

Adam Beauchaine

Thesis Proposal

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

By

April 2024

APPROVED:

Professor Craig A. Shue, Thesis Advisor

Professor Jun Dai, Thesis Reader

Professor Craig A. Shue, Department Head

Abstract

The technologies behind the virtualization and isolation of execution environments have become widespread, leading to their usage in cloud computing environments and software containerization scenarios. In both instances, these technologies combat a number of modern security threats. Despite these use cases, isolation-centric systems have seen limited deployment on modern endpoint devices, even in instances where prior research has noted such systems would benefit greatly through their implementation. There are several reasons for this lack of adoption. Isolation-centric, or "multi-environment" systems require burdensome mandatory access controls to define and delineate asset security levels and their corresponding security environments. This is often done through laborious manual generation of security metadata. Additionally, existing tools that allow the user to engage with isolation on endpoint devices are not widely employed in the context of enterprise workflows, and their usage remains largely restricted to technical specialists. In this thesis, we aim to address these important challenges to create a strong technical and procedural foundation for the usage of isolation-centric security in modern endpoint devices.

We explore the challenges of access control complexity in isolation-centric systems. Our work proposes the usage of unsupervised learning as a labeling mechanism for assigning data assets to security groups. Our approach leverages a combination of UI context data gathered from an endpoint device, as well as on-screen natural language data associated with a given asset. We compare our approach with offline strategies for security labeling. Moreover, we address the usability challenges associated with isolation tools on endpoint devices. We conduct classical usability modeling of hypervisors and containerization software using our own tool independent workflow. Our results serve as an analytical usability framework for these tools. We use these results in the construction of a novel tool design that improves upon prior best performers in all measured categories. We introduce a fusion of these components through the notion of system sandboxing as a means of addressing uncertainties in unsupervised learning output. Our results show promise in addressing the most impactful challenges for the usage of multi-environment systems in endpoint devices.

Contents

List of Figures

List of Tables

1 Introduction

Ransomware attacks have resulted in a $20 billion loss in 2021 across enterprise computing environments [64]. One of the factors that makes ransomware and similar attacks successful is the prevalence of single-environment endpoint systems. In a single environment, when a user interacts with a malicious item, their entire system is placed at risk, including data assets uninvolved with the associated workflow. The configuration complexity of drafting access control schemes for these environments is high [63], leading to security policy goals being challenging to conceptualize and maintain among system administrators. A 2022 survey revealed that 84% of organizations want to simplify their access control methodology, and 56% of organizations have attempted and failed to implement a new access control methodology within the past year [68].

Users often fill multiple roles in an organization, a 2020 study found that employees average 2.3 roles beyond their primary job [1]. In a single environment model, even when associated with different user roles, data assets can become intermingled. This intermingling often forces organizations to adopt a "one size fits none" policy of access control, in which attempts to implement fine-grained access control policy are marred with configurational complexity. Users can often request allowance rules for workflows they wish to engage in, which can balloon policy drastically and lead to overly complex, ad hoc policy rules. The simple maintenance of such rules has additionally been noted as infeasible for smaller organizations. This is due to the time requirements of determining what software or data assets certain users should be allowed to have on their devices [33]. Another challenge facing single-environment endpoint systems is that of addressing end user actions that fall outside expressed security policy. Many of these environments simply default-deny any unspecified action [13], even when such an action might not necessarily pose a security risk. This can lead to a significant decrease in productivity for end users in scenarios where completing their work may constitute a violation of security policy.

In contrast, leveraging multiple security environments in endpoints has been shown to reduce policy complexity while increasing resistance to cyber-attacks [56]. Such an approach enables system designers to group their data assets into multiple security domains within an endpoint system. This grouping of data allows for simpler policy management, as policies may be written for asset groups as opposed to individual data assets in a single-environment model. Multi-environment, or isolation-based systems, are commonly implemented with the usage of virtualization or containerization to create separate trust domains within a single device. Multi-environment endpoint systems may additionally provide techniques to allow for secure demonstrations of policy overreach. Certain "risky" user actions may be worth allowing, but limiting to a

separate environment in which a cyber-attack would be unable to impact the rest of the system. Such an approach allows for future policy alterations based on what experimentation end users engage in, all without disrupting an end user's workflow.

Despite these advantages, multi-environment endpoint systems can be challenging to implement and maintain. QubesOS, an isolation-based operating system built with the goal of providing multiple data environments, suffers from configurational complexity and usability challenges [15]. Multi-environment system configuration typically involves the labeling of data assets to ensure proper isolation is performed, a process researchers note as tedious and prone to errors [32]. Given our focus on reducing configurational requirements on system designers, multi-environment systems appear to simply shift a workload of drafting access control policy to drafting security isolation labels. Our work aims to address these labeling requirements. In the first component of our work, we leverage natural language processing of application text input, as well as user interface data, as a vector for an unsupervised learning model to dynamically assign data assets to security group environments during real-time use. Multi-environment systems additionally allow us to address the issue of "default-denying" policies that prevent workflows not specifically allowed by a system designer through the usage of environment sandboxing. For example, if a user engages in a workflow with a data asset not expressly allowed by policy, such as opening a document with a new text editor. Instead of denying this workflow, the user could be permitted to continue their work in a sandbox, isolating the risks of the workflow from the rest of the system. While current data sandboxing tools allow for dedicated environments for the testing of potentially dangerous workflows, their adoption in enterprise systems remains incomplete. The second component of our work will observe usability challenges with these sandboxing tools, and devise our own improved model tool to allow for such workflows to take place without hindrance to the end user. Upon full implementation, our work enables for the dynamic security grouping of data assets within an endpoint environment to create security policy, as well a strategy for enabling user experimentation of workflows outside the range of this policy. A view of this implementation is shown below in Figure 1.

Prior efforts have identified security vulnerabilities and convoluted access control policies stemming from single-environment endpoints, leading to significant work in the area to address these issues. We draw on existing efforts to further orchestrate the classification of organizational data assets, as well as usability of the process for user experimentation via sandboxing. Such efforts includes endpoint access control strategies based on virtual machines [56], copy on write (COW) file systems [16], and attribute-based-access control [55]. Unlike existing approaches that construct policy directly from user input, our work leverages a novel multi-environment orchestration method via content-clustering of data assets. We combine this approach

with usable workflow experimentation. In so doing, we combine the advantages of existing approaches with respect to increased endpoint security, as well as simplified policy engineering.

Prior work in the field of content-based unsupervised classification algorithms has focused on the dynamic clustering of data assets in the context of social media [69], data forensics [58], and cyber-threat identification [51]. Our work leverages the outputs of a clustering algorithm for security groups, similar to the contributions of Promyslov et al. [65]. We additionally leverage natural language text vectors as input, in a similar manner to the work of Katz et al. [41] on data loss prevention systems, though expressive content modeling through words is not a component of our solution. Rather, we leverage the clustering label decisions as security labels for isolation, without directly quantifying security requirements for each individual cluster. This simplification on prior work is done to implement the real-time labeling component of our solution, in which we leverage a data streaming clustering algorithm in a similar vein to Liang et al. [48].

We also investigate current issues and improvements to dynamic experimentation via sandboxing in multi-environment endpoint systems. The usability of current sandboxing isolation tools has been noted as an issue [45]. We draw on human computer interaction (HCI) usability quantification techniques such as Keystroke Level Modeling [19] to provide a standardized analysis of current tools, as well as propose our own improvements. We design standardize workflow between current tools to serve as an analytical framework to assist with our experimentation. To formalize and guide our efforts within these spaces, we ask the following research questions:

- To what extent can we link UI workflow data with security group content within the context of a security group membership decision procedure?
- To what extent can we link and leverage UI data and natural language processing to dynamically cluster data assets into distinct security groups?
- What are the usability challenges with existing sandbox and isolation tools?
- What opportunities are there to improve the workflows related to sandboxing, in terms of productivity and user comprehension?

To explore these research questions, we construct and tune a streaming, unsupervised learning algorithm to analyze and sort data assets into security groups. Our clustering is done with input vectors consisting of NLP and UI interface data. We implement a UI data gathering system to dynamically gather data from user workflows on an endpoint system, then construct the resulting data vectors. These vectors are generated on a per-asset basis, and clustered at regular intervals in using a landmark window model. We additionally

identify and quantify existing usability challenges in current endpoint isolation sandbox systems. We measure how well the existing technologies address usability challenges, identify improvement opportunities for the existing technologies, and implement a front-end prototype that incorporates such improvements. As a result of this exploration, we make the following research contributions:

- *Fusion of User Interface Data and Natural Language Processing*: We explore a novel fusion of user interface trace data and natural language processing of environment program text input as an vector for a clustering mechanism. We leverage the results of this mechanism for security labeling of data assets within an endpoint system. Through our system outputs, we may dynamically assign novel assets to existing cluster groups based on similarities, or update cluster group membership based on incoming data.

- *Real Time User Interface and Natural Language Data Gathering*: We construct a software system to leverage the Windows UIAutomation library to gather relevant on-screen data to current user workflows. The data we gather includes on screen natural language text, position and name of data asset on screen, and time of access. We parse and format this data to include relevant information to our clustering approach, and limit information to singular data assets in an environment. This facilitates a real-time data stream of asset information that is used in our clustering approach.

- *Data asset security group clustering analysis and optimization*: We identify and tune areas of an unsupervised security labeling algorithm for performance in an enterprise environment. Through a process of input sanitization of NLP and UI data, as well as model selection and hyperparameter tuning, we provide an outline of effective techniques and recommendations for the usage of streaming clustering algorithms as security group providers. We analyze and compare results of classical clustering labeling with our streaming clustering approach on identical datasets. Our implementation costs include user training and deploying such an algorithm to multiple endpoints across an enterprise environment.

- *Sandboxing Workflow Analysis and Standardization*: We explore the workflow associated with creating a sandbox environment and the steps to implement the workflow across different isolation tools. We identify and label common actions and milestones across tools to enable a common point of reference across a diverse range of systems leveraged for sandboxing.

- *Usability Analysis of Current Isolation Tools*: We employ the Keystroke Level Model (KLM) [19] to analyze and evaluate the usability of current isolation systems. Using our sandboxing workflow, we explore where specific sandboxing tools have usability limitations and how the tools compare with different workflow components. We analyze and compare the results of several different tools.

- *Prototype Interfaces for the Sandbox Workflow*: We create the front-end of a prototype isolation tool. In doing so, we aim to address each of the limitations in current isolation tools. We evaluate our prototype and compare its usability with existing tools. Implementation costs include the construction of a functional back-end within an endpoint system to ensure trusted sandbox creation.

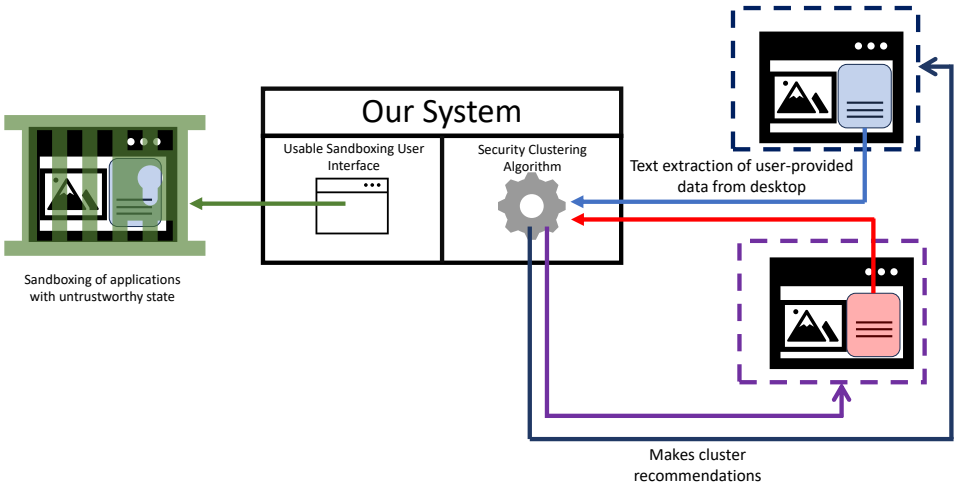


Figure 1: Our implementation goals as viewed in the context of a single endpoint system.

2 Related Work

We explore existing approaches for access control in enterprise systems. We note scalability challenges and configurational complexity across multiple current techniques. We additionally review existing work within the data asset clustering and labeling fields, as well as applications of machine learning as a security labeling technique. Finally, we investigate existing sandbox and isolation-based security instances with an eye toward usability. We also investigate current techniques used within the human computer interaction (HCI) field with regard to evaluating the usability of system sandboxing workflows.

2.1 Access Controls and Isolation in Endpoint Systems

Access control exists as a long-running paradigm of computer system security. Access control may be discretionary (user-defined) or mandatory (lattice-based) in nature. The configurational complexity of both of these approaches is explored by Osborn et al. [63] who note that both of these techniques require can lead to a significantly complex administration of access rights across a system. Additionally, overly simplified

access control implementations in distributed systems have been shown to cause conflicting operational and security semantics[28]. Regardless of the nature of access control, coarse grained approaches may lead to poor system security, whereas fine grained approaches significantly increase configuration requirements.

Isolation within computing systems has been leveraged as a tool in improving security outcomes while reducing the required expressiveness of access controls. Isolation of untrusted processes may function as a less invasive "default-deny" rule in an access control system without directly revoking read or write privileges for an end user. Isolation may be at the user's discretion in some cases, such as the work of Backes et al.[11], which leverages Android process isolation to effectively sandbox a user-specified application. Other work leverages isolation as an added security layer in existing mandatory access control solutions. Helal et al. [34] implement an isolation-enabled relational database in which unvetted users may access isolated copies of resources in emergency scenarios. Many implementations of isolation produce compelling advantages, but are limited in scope. Our work aims to facilitate isolation-enabled access control for endpoint devices in a pragmatic, desirable manner for end-users and system administrators.

In a similar vein to our own work, McCune et al. propose Shamon, [56] a mandatory access control enforcement system for distributed computing environments. This work allows for the segmentation of endpoints into virtual machines and remote trust attestation of endpoint virtual machine management systems to simplify access control policy. We incorporate these ideas in our approach. Related work such that of Bratus et al. [16] notes the weaknesses of approaches such as SELinux [55], and develop a copy-on-write solution for user workflows not expressly allowed by policy, as opposed to strictly denying them. Our work seeks to build on these observations, and create a foundation for a multi-environment endpoint system of comprised of numerous security groups, through the usage of clustering and sandboxing techniques.

2.2 Unsupervised Learning and Security

Prior work has noted the efficacy of unsupervised clustering algorithms in regard to various security problems. Gurumdimma et al. [30] parse and cluster system failure logs in an effort to improve log readability and identify structured events within failure log data. This work highlights how clustering can improve understand-ability for system administrators, through ability to simplify and streamline system log presentation. As part of a broader intrusion detection effort, Aussei et al. [10] leverage NLP-based clustering to perform anomaly detection in industrial systems. Additionally, Abb et al. [2] cluster user interface logging data for the purpose of usability engineering, allowing for a far more streamlined view of common and anomalous UI events. Our approach draws on these enhancements to understand-ability in the context of a

complex, unlabeled computing system environment, to enable the delineation of diverse data asset groups.

Unsupervised clustering techniques have frequently been leveraged in traditional information retrieval problems, and calibrated accordingly. Alsmadi et al. [8] evaluate a wide array of data vectorization and clustering techniques for the purposes of clustering email contents, noting N-Gram vectorization as providing the strongest internal validation metrics among the tested approaches. Additionally, Li et al. [46] leverage a classical expectation maximization algorithm to derive clusters in a limited set of document data. The authors then develop a support vector machine algorithm using these clusters to classify new document data. Such information retrieval techniques naturally allow for the clustering of system data assets. Sethi et al. [69] provide a content-based clustering algorithm for social media messages. Nassif et al. propose a clustering technique for the content of files for digital forensic analysis [58]. These works serve to highlight the effectiveness of content-based clustering, as well as its usage in defining security-relevant groups of data assets.

As further proof of this effectiveness in the field of security, Promyslov et al. [65] propose a method of asset security grouping within an industrial setting via defining labeled data assets to serve as centroids of security clusters, each defining a unique security group. Katz et al. [41] employ unsupervised clustering on an existing collection of unlabeled security documents, and observe key terms of clusters that indicate confidentiality requirements, and define a formalism for labeling according to these cluster terms. In a similar vein, Alzhrani et al. [9] record external validation measurements on security labeling decisions made from supervised learning and, through semi-supervised methods, achieve statistically superior results through an engineered procedure based on clustering. In contrast to these efforts, we combine existing data gathering approaches, through the utilization of NLP and UI data vectorization, to produce dynamic asset security groupings on a per-endpoint basis. Additionally, we employ a streaming variant of classical clustering approaches to allow for dynamic security labeling over time, as opposed to existing strategies that employ clustering over a pre-existing document corpus. Such real-time groupings may serve as the foundation for separate security environments based on the usage context of assets, which we attain through the fusion of natural language data and UI interactivity data.

Streaming clustering algorithms allow for clustering to be performed on a real time stream of input data vectors. As the work of O'Callaghan et al. [61] and others have demonstrated, streaming clustering may be a performant and effective strategy for high dimensional data such as documents. Algorithms such as BIRCH [75] allow for significantly reduced storage requirements and increased performance on large, high dimensional data sets, such as text documents. Liang et al.[48] leverage a Bayesian clustering algorithm for topic modeling of short web documents. In a similar vein, Looks et al. [52] develop a streaming hierarchical

clustering model for text document streams, with a goal of adaptive reorganization of organizational concept hierarchy. Our own work diverges from existing document streaming clustering approaches through the concatenation of user interface engagement data onto our data vectors, with the desired goal of allowing for a more effective conceptual clustering framework in regard to accuracy and outlier detection for security purposes.

2.3 Usability Analysis in Endpoint Sandboxing Systems

A system sandbox refers to a secure, trusted computing environment for the execution of untrusted, potentially malicious items. The benefits of sandboxing have been noted across multiple application areas such as smartphones [66] and servers [7]. Sandboxing is less common in traditional enterprise desktop systems, with poor usability being noted as a potential reason [45]. We explore the usability of sandboxing systems as a potential blocker to widespread enterprise adoption in greater detail than previous efforts. To conduct such an exploration, we draw on HCI techniques which seek to quantify program usability.

The importance of usability as a security consideration has been noted across a range of analyses [67, 39, 12]. The usability hurdles in sandboxing workflows are not often discussed; however, they have been noted in several works. In SAFE-OS [45], the authors design and implement a sandboxing system that adds several features identified as lacking in Qubes, such as VM specialization. They also design a user interface built to emulate a traditional desktop, noting usability concerns in Qubes. Sun et al. [71] observe usability challenges in both configurational complexity and end user comprehensibility in existing VM provisioning systems. They provide a web-based provisioning tool to automate environment creation via an XML file. In a similar vein, Huang et al. [36] note configurational challenges with virtualized network infrastructure and propose a Package Virtual Network hypervisor to deploy virtual networks dynamically. Our approach focuses on using specific usability modeling techniques to identify and resolve sandbox usability issues. Since these technologies are proven to increase end user security, adoption hurdles such as usability may be an issue. We consider Gligor's [26] insights into functionality often coming at the cost of usability; we explore the issue of sandbox usability with a goal of retaining functionality while resolving usability challenges.

The research community has explored ways to evaluate the usability performance of user interfaces. This has resulted in models such as the goals, operators, methods, and selection rules model (GOMS) [43], as well as other evaluations and proposed standards[37, 38]. Keystroke level modeling (KLM), a successor of the GOMS model, has been widespread in system usability evaluations since Card et al. [19] pioneered the approach. It enables system evaluators to decompose and label user tasks as well as quantify usability

performance. KLM generates strings that offer algorithmically computable estimations for the times required for task completion [5]. These strings may then be used as baseline estimates for usability of systems. KLM has seen positive evaluations of its accuracy and value to system designers [40, 5], as well as usage across platform interfaces [47]. We draw upon these use cases in designing our own procedures for sandbox evaluation.

This work blends usable sandbox environments with a security group labeling tool. The core of our work is the implementation of a novel endpoint security labeling system, based on a streaming clustering algorithm of vectors representing data assets, featuring natural language and user interface data associated with a user's workflow. The clustering output serves as the foundation for isolated security groups within a user's machine, allowing for isolation of previously unlabeled data assets. In the case of outliers or other clustering uncertainties, we address the usability challenges of endpoint sandboxing to allow for a policy approach of "default sandbox" for assets that should be isolated further. These contributions should significantly advance the practicality of multiple, isolated environments in enterprise endpoint systems.

3 Approach Overview

In this work, we leverage unsupervised environment clustering and usable sandboxing to serve as foundational infrastructure for a secure, multi-environment endpoint system. This foundation allows for the dynamic classification of data assets into separate security groups based on context data of asset usage. It additionally allows for the dynamic expansion and reclassification of data assets based on novel context data through the usage of streaming clustering. Separately, we provide a usability-focused interface for enabling end users to securely experiment with new assets in a manner not allowed for by policy via sandboxing. We make these contributions in part as a response to the configurational complexity challenges identified by Bratus et al. [16] and other in regard to modern isolation-based security in endpoint systems. In combination, these contributions address several long-standing issues in the effort to allow for isolation-based secure endpoint devices. They allow for real-time security labeling of data assets on a user's machine, as well as a usability focused design for a sandboxing system aimed at addressing classification errors and uncertainties.

We first implement a UI monitoring system in an endpoint system to harvest UI workflow data as input to an unsupervised, content-based clustering algorithm. We collect UI data relevant to security group status in the graphical interface of an operating system on a per data asset basis. Our system implementation is a direct modification of the work of Chuluundorj et al. [20] who leverage the Windows UIAutomation library to record context data in relation to network activities. We expand their "Harbinger" system to capture all

relevant on screen data in regard to a data asset such as: all natural language text, window size, and time of access. These values are condensed and output as a trace describing the current on-screen data asset. Our contributions result in a novel system for contextual data gathering dubbed "GuardCluster."

For our data assets themselves, we utilize the Enron email dataset [21] due to its variance of natural language topics and range of potential security categories. To simulate real user interaction in this system, we leverage Sikuli [35], a UI automation tool that interacts with graphical elements on an endpoint device, to simulate real user workflows. We specifically target an email client, Mozilla Thunderbird, to automate the sending and receiving of emails in an effort to simulate the user activity involved in the creation of our source email dataset.

Our GuardCluster output is sent over a networked connection to a streaming clustering server for classification. Such a data vector is well suited to many clustering approaches including the K-Means [53] and Hierarchical [29] techniques. We modify and tune the linguistic preprocessing of data, as well as standard clustering hyperparameters, to ensure a performant model. Our approach differs from existing strategies toward content clustering through its usage of UI data as a part of the data asset feature vector. This model, along with our usable sandboxing investigation, serves as the basis of a multi-environment endpoint system based on data asset security groupings.

We evaluate the success of our model based on a number of interal and external validation techniques for unsupervised learning. We implement silhouette coefficient [70] and Davies-Bouldin index [23] scoring within our clustering algorithm to display internal evaluation of clustering success. Due to the nature of the Enron email dataset, limited labels may be provided for the data through manual review and labeling of potential data security groups. This allows us to leverage external validation techniques, such as the F-Score, as well as manual inspection of clustering quality. These clusters will serve as security groups that may be applied as asset labels for future policy implementations. The usage of these techniques allows us to critically evaluate the success of our clustering approach on a per iteration basis in our streaming clustering algorithm. Due to our usage of streaming clustering, we additionally inspect how our clustering mechanism responds to the phenomenon of conceptual drift, or changing cluster centroids over time, which may present challenges for labeling accuracy over a period of multiple user workflows.

The second component of our approach allows for dynamic experimentation of workflows not expressly allowed by policy via sandboxing. We study usability challenges with existing systems that offer similar functionality such as VMWare [3], and QubesOS [72] via KLM system usability measurements [19]. We conduct this experimentation after developing a standard sandboxing system workflow that may be applied to all tools and serve as a standardized "point of reference" to where certain tools succeed and fail in usability.

This point of reference subdivides sandboxing workflows into separate system components, such as environment creation and asset insertion. Using these insights, we devise a "Wizard of Oz"[42] implementation of a usable system for endpoint devices. We construct this tool through use of the front end development suite Adobe XD [4].We then evaluate our tool based on the same KLM evaluative process used for previous existing systems to directly observe improvements upon existing techniques. Significantly, this approach allows for the construction of usable endpoint sandboxing systems. This may be leveraged in combination with our dynamic environment clustering algorithm to provide a secure, extensible foundation for the support of multiple environment endpoint systems.

The fusion of dynamic asset labeling and usable sandboxing comprises a foundational understanding of how multi-environment systems may function in enterprise environments. Unsupervised clustering allows for the removal of security labeling workflows associated with existing access control systems. However, clustering is an imperfect solution and capable of making mistakes. Sandboxing can provide for workflows in which potential mistakes in access control are not threatening to the entire data environment of a system. Together, these technologies allow for a significant departure from traditional, single environment access control systems, but presents significant advantages for data security. We detail our approaches to these components in Sections 4 and 5 respectively.

4 Clustering of Data into Appropriate Operating Environments

Security administrators can often struggle to understand and quantify security risks associated with a wide variety of data assets. This struggle often presents a major hurdle for the deployment of multi-environment endpoints, which require a strong understanding of data asset security labels in order to perform effectively. These labels may be generated manually, but the process of doing so is often noted as difficult and tedious, even for security experts [32]. This presents real challenges for environment security, given the disastrous implications a single misapplied label could present for an enterprise system.

Alternatively, defining policies in a single environment without expressly labeling assets presents its own collection of challenges. Adequately securing all assets in a single environment may result in complex, error-prone access control policy for environmental data assets, given the need to expressively account for all potential security policy violations within a single environment. The maintenance of such policies is challenging, and failure to perform sufficient upkeep can drastically degrade policy effectiveness [17].

Both of these scenarios present current realities for system designers. While current systems nearly universally leverage single environments with specific access controls, the costs associated with both single

and multi-environment endpoint systems make each undesirable to a large share of system designers. Many organizations that overcome these problems do so by investing high amounts of labor into policy generation. This approach is untenable for many enterprise environments, which lack the resources to be verbose in security policy definition.

Unsupervised learning is a data science discipline that excels in offering useful insights into collections of unstructured data. Despite widespread research engagement with the broader subject, security-based applications of unsupervised learning with regard to organizational data assets remain largely incomplete. This is cause for investigation, since unsupervised techniques such as data clustering may offer a streamlined procedure for data asset security labeling when compared to human inspection. Additionally, existing approaches often only consider limited, external sources of data as clustering inputs, such as asset security properties. Our work focuses on the collection of multiple data sources including natural language and user interface data within endpoint devices. We leverage a clustering technique to discern security labels from collected data within an access-control policy generation system, with the goal of simplified endpoint policy management and multi-environment endpoint support.

Security administrators need a way to enact effective system access control policies without expressly defining security labels for organizational data assets and groups. They need a way to maintain and revise these policies when new software or assets are introduced to an environment, all without incurring the costs of manual labeling of assets or drafting complex access control schema. Existing asset labeling techniques such as Microsoft Purview [54] and Fortra [50] present high maintenance costs. Alternatively, existing automated labeling techniques are solely offline in their application, such as leveraging named entity recognition [31], or unsupervised clustering [41]. Our work aims to bridge the gap between these existing data asset management paradigms by enabling unsupervised labeling of data assets in a dynamic, real time manner. To this end, we ask the following research questions: *To what extent can we link UI workflow data with security group content within the context of a security group membership decision procedure? To what extent can we link and leverage UI data and natural language processing to dynamically cluster data assets into distinct security groups?*

4.1 Clustering System Architecture

We detail our approach to a data asset clustering system for security group labeling. We discuss specific functionality regarding endpoint logging techniques, data vectorization strategies, and overall clustering algorithm implementation. We also provide a high level overview of system architecture and goals within

Section 4.1.3. This architecture may be visualized in figure 15. We detail the results of these efforts in Section 4.2.

4.1.1 Existing Access Control Challenges and System Overview

Within modern information systems, end user action accounts for a large share of transformation procedures performed on organizational data. As such, each individual user is likely to interact with a wide variety of data assets on a given day. One user's role might require them to access organizational finance data, as well as internal HR documents. This presents an immediate issue if both of these assets exist within the same environment, as cybersecurity risks associated with a user workflow could impact an unrelated data asset, if a threat manages to affect an entire endpoint system. Because security metadata is uncommon in current enterprise environments, the creation of endpoint access control policies for specific data assets can be a tedious, manual process. We aim to expedite the process of policy grouping through a streaming clustering algorithm for data assets, in which cluster membership denotes security grouping within an endpoint system. In contrast to previous efforts, our usage of streaming clustering allows for faster security decisions regarding data actively in use on a system. We intend this to serve as the basis of a multi-environment system for endpoint devices, in which specific clusters of data assets will be confined to corresponding environments.

At the core of our approach is the fusion of textual and user input data within the framework of a security asset clustering algorithm. We believe both user input text, as well as UI data, should offer valuable insights into the security group an asset should be placed within. Textual data in this case refers to all input data a user types into a program window associated with a data asset, for instance an email or word processing document. Textual data additionally includes on screen data not directly entered by a user, such as text on a webpage or application window. Such data may be formalized as unique documents for each asset, each of which may be factored into our clustering approach. This approach enables us to construct security asset labeling as a data retrieval problem among a wide set of documents, with results indicating security group labels. The second component of this strategy involves the usage of user interface data as a component of our cluster labeling strategy. The user interfaces of modern operating systems may produce an enormous amount of data. As such, we only refer to UI information we find to be of high relevance to asset security group membership. This includes membership of other programs on screen at the time of opening a new data asset, position of asset window on screen, and specific UI elements interacted with during a data asset transformation workflow.

The usage of an unsupervised clustering algorithm for security group labeling is likely to spark some confusion, given such a system's capacity to produce occasionally uncertain, or at worst incorrect labeling

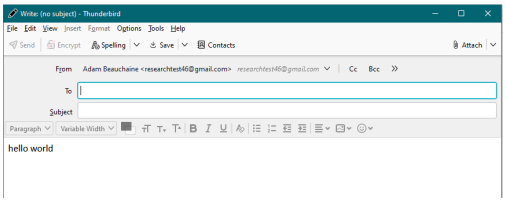
decisions. Our rationale behind the usage of such a system involves several factors. While a traditional classification approach involving supervised learning could produce better results, such a system is impractical to implement within a modern enterprise environment due to the challenges of acquiring labeled data relevant to security group membership. Even if this could be addressed, models generated run the risk of needing to be retrained on a fresh dataset if environment security variables change significantly, which may be a frequent occurrence in many modern information systems. These issues remain, although to a lesser extent, in alternative solutions that might leverage deep learning or statistical inference as security classification tools. Streaming clustering allows for fast, easily updatable group membership labels. The usage of streaming clustering allows for the drifting of cluster centroids over time, accounting for data changes over the lifespan of a data environment, in a phenomenon known as conceptual drift. Additionally, we allow for the accounting of errors in clustering labeling via the general system design. Through sandboxing and separate environments, we may isolate errors to specific clusters or sandbox environments depending on organizational needs. A lack of certainty may result in a "default sandboxing" strategy for access control, similar to a "default-deny" that currently exists when policy fails to produce an answer for a specific query.

4.1.2 Endpoint Logging Techniques

To obtain this data for use in our labeling algorithm, we leverage a modified version of the previous "Harbinger" system [20], dubbed "GuardCluster," which is capable of collecting and formatting specific data of relevance to asset group labeling from a user workflow. GuardCluster leverages a fusion of kernel level IO monitoring and UI element accession data from the UIAutomation library for Windows. While Harbiner's previous use case involved gaining brief, temporal amounts of context data for network access control platform, we significantly modify GuardCluser's behavior and outputs. We instrument dynamic collection of natural language data of programs on a user's screen, including email, web pages, or word processing programs. We additionally collect other useful access context information such as time of access or whether a given window is in full screen mode. This data serves as the foundation for our clustering and identification of different assets within a data environment.

The kernel-level component of the GuardCluser application is primarily focused on collecting contextual user events such as focus changes and keyboard interactions. This is done through the construction of a global event handler for focus changes, as well as a hook function for mouse clicks and keystrokes. Both of these components are run asynchronously from the main UI observation component. For our approach, we ignore these components of the tool to instead focus on the data provided by the UIAutomation library, though the focus change event handler was leveraged extensively during testing of this application.

The main thrust of our efforts leverages the UIAutomation library to record on screen natural language from Windows applications. This data is generated from Microsoft's .NET framework, a component of which is the Windows Presentation Foundation graphical subsystem, [24] which contains accessibility support for on-screen graphical elements. The UIAutomation library serves as an API into .NET applications, allowing us to query information such as window title, size, and any rendered elements displayed on the application. While acknowledging the .NET platform is not universally used, its large market share when compared to similar tools [25] makes it the most desirable option for instrumenting UI data collection on as many applications as possible in an enterprise Windows environment. For example, our testing involved the successful data collection from Microsoft Office products such as Word and Excel, web browsers such as Mozilla Firefox, and an email client in Mozilla Thunderbird. An example of this may be visualized below in figure 2



(a) On-screen Window.

```
Time=210020,
ASSET: [Write:<no subject> - Thunderbird],
FS = false,
Size = 734088
N=[{[Message Body]}],
NL=[{[hello world]}]
```

(b) Resulting GuardCluser Trace

Figure 2: Example of Trace Formalism and Application

As applications are rendered on screen, UIAutomation represents each graphical element as part of a tree structure, with the user's Windows desktop serving as the root node of the tree. Applications are branch nodes, and different graphical elements within such applications are the children of these nodes. Our approach involves walking and parsing individual tree nodes to locate nodes of specific relevance to on-screen text data. GuardCluster records this data as a "trace," which targets the most recent element a user has interacted with on screen. These traces may be configured to obtain additional information based on the given UIAutomation element, and store this collected data for future use. For example, a user could click a paragraph in Microsoft Word while GuardCluser is running, and GuardCluser could record the text, document name, and title bar details associated with that action.

This approach is helpful in calibrating our data collection to elements of specific relevance to a user's current workflow. For natural language, we gather nodes of type "name", and "textEdit." These node types comprise the bulk of on screen natural language data, in both application text and user generated text. Our approach allows for the acquisition of these elements through a simple tree traversal, as all information for a given program may be done by walking up the tree until reaching that programs root node, we then walk

through all children of that node and append the data we want to include in our trace. We output this data on set intervals, five seconds in our testing, in which the most recent GuardCluser trace is acquired. We find this approach sufficiently captures workflow details of a user's accessed elements in a given program, assuming the given interval is short enough.

4.1.3 Data Gathering and Vectorization Strategies

Upon generation of these traces, we implement a separate program for the parsing and clustering of this data into security groups. As we detail previously, these traces are short, uniform, and capable of rapid transmission across a network connection. We leverage these attributes in a simple socket program addition to our work, which sends traces as they are generated to a python server. This server may be hosted locally or remotely. In our testing, we chose to implement a remote server for our work. This approach could be expanded to include multiple UI gathering clients in the future, though our current work involves a single client machine.

Our Python server consists of two core functions: a data parsing service as well as a streaming clustering algorithm. Data parsing refers to the cleaning and vectorization of trace values as they arrive from the UIAutomation client program. At the core of the data parsing workflow is the service of asset delineation, or discovering which traces are associated with which assets when performing clustering of data vectors. To resolve this issue, our program analyzes the "Asset Name" field, which is derived from the window name currently open on the client machine. Any change in this field is associated with a focus change in Windows, and thus associated with a new data asset being displayed on screen. As an example, a user could change focus from one browser tab to another, and this action would be classified as creating a new data asset due to the resulting change in window title, named after the title of the new browser tab. There are edge cases in which data assets could be missed, such as a sufficiently complex web application in which a user engages with multiple types of data assets without changing their OS window. Despite the existence of such cases, our work is inclusive of a high amount of operating system workflows, given the ubiquity of window name alterations in the Microsoft Windows operating system.

Within our python server data parsing function, trace data is received and stored in a buffer until a change in asset is detected, delineated by a change in the "Asset Name" field. When we detect such a change, the storage buffer, which is a Python list of all traces received, is sent to the server, where parsing and condensation into a single data vector occurs. Each data vector is associated with the asset name. Each trace string is parsed as a comma delineated list, and pruned for punctuation and other non-relevant data. If a trace contains multiple instances of natural language data, or human readable element name data, it will

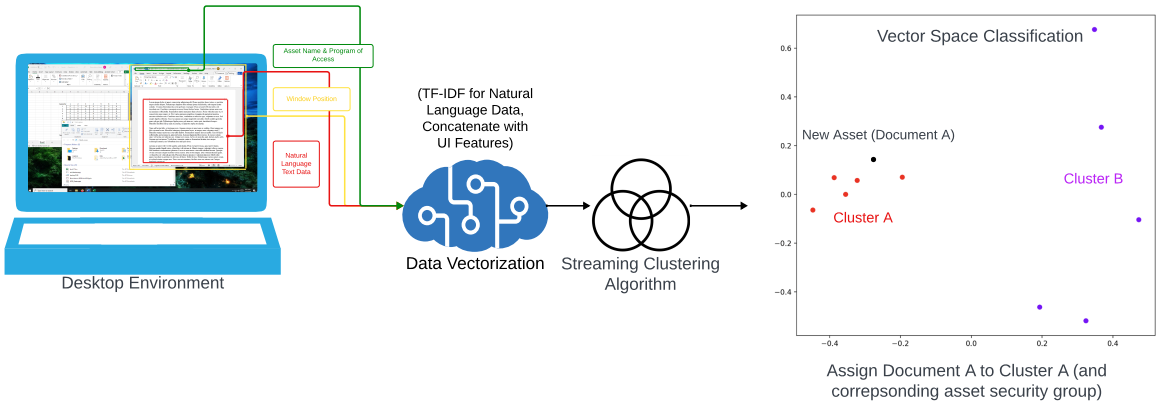


Figure 3: Our Full Implementation Scheme

be concatenated and placed into a single list element. Due to our usage of "bag of words" clustering, this simple approach causes no issues with the clustering algorithm. Finally, because many trace strings may be associated with a single data asset, we merge all trace lists into a single, five value data vector associated with a current asset.

- **Time of Access**: Time of access stored as an integer and averaged between all traces associated with an asset.
- **Asset Name**: Name of asset as derived from system window. This value is consistent between traces
- **Fullscreen**: Boolean value symbolizing whether an asset is fullscreen on a users desktop
- **Window Size**: Area size of the window stored as an integer
- **On Screen Natural Language**: A concatenated string of all on screen natural language associated with an asset, either typed or viewed by a user.

This data may be stored in a dictionary with assets as keys, and other attributes as dictionary values. We may then iterate over dictionary keys and retrieve specific values for an entire corpus, such as all text data the program has seen so far. This is used in our streaming clustering approach, as text document vectorization requires knowledge of all text associated in a given clustering window. Given the timing and data window requirements for streaming clustering, the full clustering procedure is only performed every `n` assets, in which n is an adjustable value for clustering window size. This alters the size of initial training data, and the time period between re-clustering runs. Our implementation of the live clustering approach is detailed in Section 4.1.3.2.

To allow for faster testing of a large email dataset, we use Sikuli [35], a separate UI automation engine, to iterate over and engage with emails in the same manner as an end user to be recorded by GuardCluster. Our Sikuli script iterates over a directory of emails taken from the Enron dataset, marking and saving the time of sending and whether the email was sent or received. The script then automatically opens a Mozilla Thunderbird email client on the client machine, and types the email in the same way a user interact with the software. The timing of these emails is based loosely on the time of sending in the source email directory, but has been reduced to allow for testing in a reasonable timescale. When an email is finished and sent, the script idles until the next email is to be sent. In the case that an email is received, the script simply displays the email on the screen instead.

Figure 3 shows our complete scheme of implementation for our clustering algorithm. It details the gathering and concatenation of UI data into per asset data vectors. A simple example of vector space classification may also be viewed in this figure, where a new asset, "Document A" has arrived as a vector to the clustering service. The resulting label gained through vector space classification may be re-applied to the source asset, though this application is not a component of this thesis. These vectors are amenable to a wide variety of clustering methods beyond our current streaming approach, this lead to a high degree of experimentation with such approaches when tuning our clustering strategy.

4.1.3.1 Clustering Algorithm Design and Tuning

A critical component of our clustering algorithm is the usage of the Enron dataset as a data source. The dataset consists of approximately 500,000 emails over six years, with wide ranging topics involving corporate and personal natural language messages. We limit our clustering dataset to a single user's email, to better simulate the system's intended usage scenario. In dataset selection, we identified three randomly selected target users whose for our testing. Our experimental scenario involves a single endpoint device, so three tests and evaluations were subsequently performed, a single test per user.

Prior to the construction of our system, we evaluated a number of document clustering and vectorization strategies for use in our approach. We test clustering strategies on Enron dataset samples, and evaluate using internal and external verification of correctness and usability. Clustering is performed in an offline manner during these experiments, the results of which are leveraged in designing our online streaming clustering algorithm. For our testing, we observed similar silhouette scores for a partitioning strategy (K-Means), and hierarchical strategy (agglomerative). In an evaluation of a subset of 10,000 random emails taken from the Enron dataset, the K-Means approach achieved a silhouette coefficient of **0.70** and the hierarchical approach achieved a similar score of **0.73**. We additionally experimented with several different linkage strategies for

cluster generation. While cosine similarity based linkage is generally preferred in high dimensional settings, we additionally experimented with euclidean linkage in tandem with Principle Component Analysis (PCA) reduced dimensionality vectors.

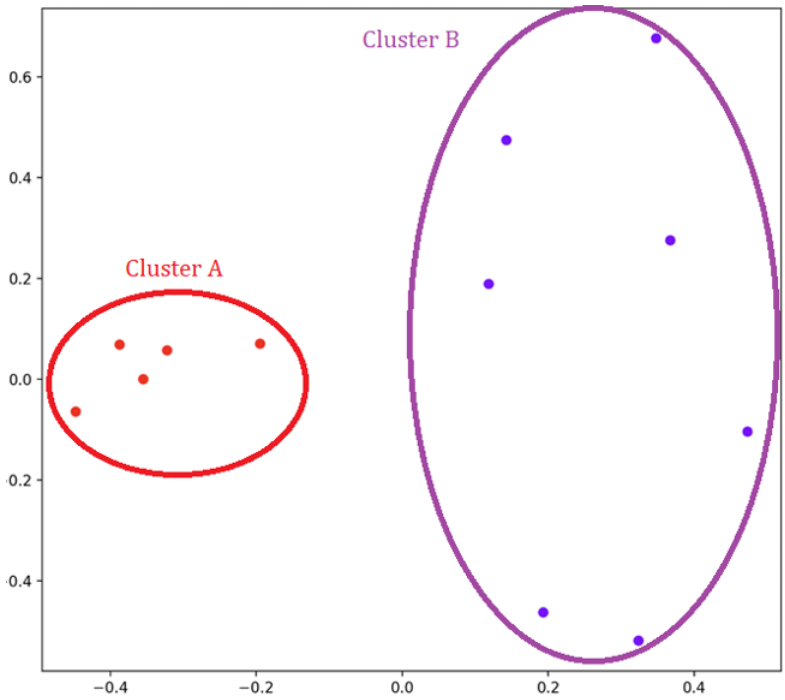
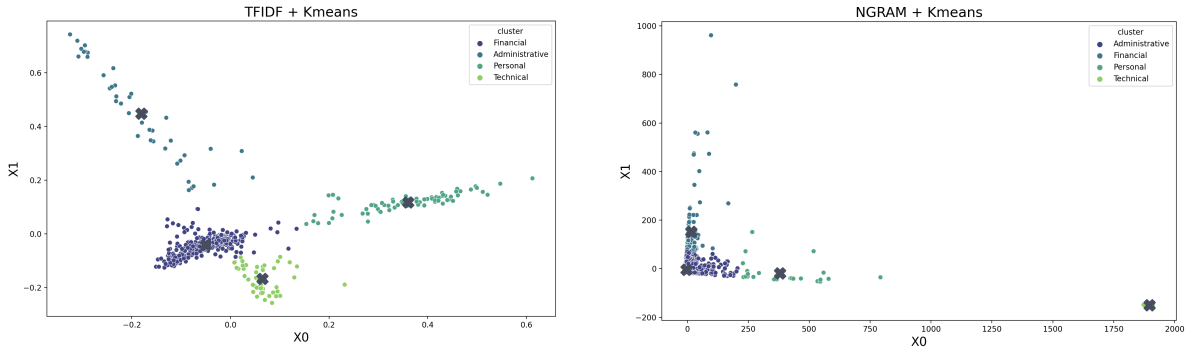


Figure 4: Cluster A Keywords: (Property, Tenants), Cluster B Keywords: (Demand, Energy)

Our external validation of these clusters is based on manual content inspection of cluster members, and applicability of results to security groups. A simple version of this approach is identifying the top n keywords per cluster by feature scores, and displaying those words after clustering completion. A low data demonstration of this idea appears in figure 4, in which 10 emails are clustered into two separate groups, with group keywords enabling an observer to evaluate email content groupings. This exernal validation was performed on all listed clustering strategies, in addition to multiple vectorization strategies. We experimented with Term Frequency-Inverse Document Frequency (TF-IDF), word n-gram model (N-GRAM), and word frequency based vectorization strategies. Due to the scale and word variability of our dataset, TF-IDF was selected as the best candidate after observing clustering results with all three. Word frequency-based vectorization produced low quality clustering results, whereas NGRAM produced poorer internal validation metrics to TF-IDF on the same dataset. Additionally, NGRAM-based clusters were determined to be significantly less usable for security labeling than TF-IDF, likely due to the frequency of repeated terminology in our test email corpus. These results may be visualized below in figure 5.



(a) Example TF-IDF Clusters

(b) Example NGRAM Clusters

Figure 5: TFIDF and NGRAM based clustering approaches

Based on our initial results, as well as the construction of our problem, we elected to utilize an hierarchical clustering approach to minimize problem-specific hyper parameters. The construction of clusters is likely to vary based on user, and we observed minimal variation in effectiveness between hierarchical and partitioning strategies in offline testing. While dynamic discovery of cluster centers is performed in prior work, the effectiveness of our hierarchical strategy led us to not consider this approach. We further detail our implementation of hierarchical streaming clustering in the following section 4.1.3.2.

4.1.3.2 Streaming Clustering Algorithm

Our streaming clustering approach is designed to cluster and classify novel data assets during a user workflow. In this Section, we detail our usage of the Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) hierarchical clustering algorithm as an online clustering tool. We describe our landmark data window model and clustering interval strategy. Finally, we demonstrate output of our streaming clustering service throughout a user workflow, and highlight responses to potential instances of concept drift within a constantly updating corpus.

Our selection of the BIRCH hierarchical clustering algorithm is based on several key factors. BIRCH is amenable to large, high dimensional datasets, given the data output of enterprise organizations we seek to capture in our experiments, such scalability is a main concern. Additionally, given our client-server model of clustering service, the scalability of BIRCH provides a simplified path for implementing multi-client functionality in the future. BIRCH additionally simplifies hyperparameter tuning requirements for large scale data sets, due to its usage of euclidean radius of cluster members (threshold value), as the sole factor to consider when implementing clustering, assuming base BIRCH is utilized. Finally, BIRCH provides

for ease of implementation for online learning (vector space classification) goals. Through the usage of the SciKit-Learn Python library, we may leverage simple function calls to partially fit the model and predict data points without fully re-running the algorithm.

Our goal is to process the entirety of received data during a given clustering interval. This is because older data remains just as relevant to OS security context as novel data. This marks a contrast of our approach to existing streaming clustering algorithms, which tend to weight newer data higher than older data. It is for this reason we avoid the usage of dampened or sliding window models [76]. Our model bears the most similarity to the landmark window mode, but we expand subsequent clustering intervals to include all previously received data, weighted equally. We further detail this window model below in figure 6. Our model rebuilds clusters every 20 instances of data, while performing online clustering for data received in between intervals. Upon the subsequent clustering interval, this data may be reclassified into novel or existing clusters once the model has been refitted with all new data.

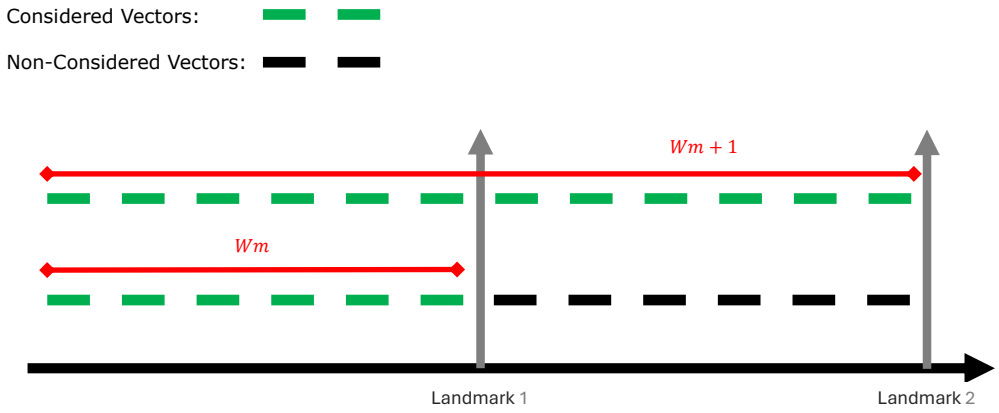
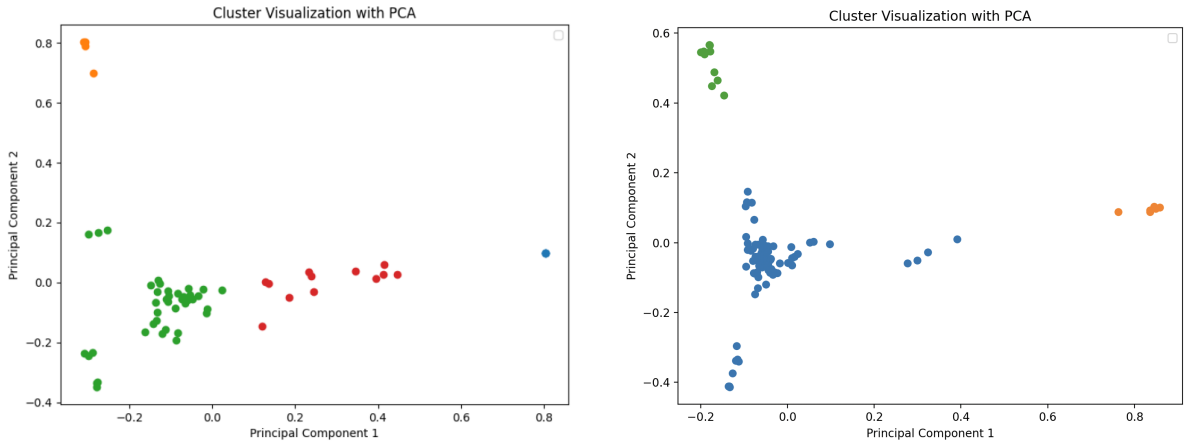


Figure 6: Our Window Model Includes all Previous Instances of Received Data

The evolution of clustering membership is observable in our results, with clusters being actively formed throughout our experimental runtime. Figure 7 provides an example of this evolution, in which two previously distinct clusters are merged as more data becomes available. This presents a visualized indication of concept drift throughout a clustering procedure, in which two groups of assets are deemed similar enough with the presence of more data to be merged into a single group, based on clustering threshold values.



(a) Example Clusters after 50 samples

(b) Example Clusters after 100 samples

Figure 7: Streaming Cluster Membership Development

4.2 Evaluation

In this Section we detail our experimental methodology for evaluating clustering performance across three distinct areas in regard to security grouping effectiveness. We document and detail the results of these experiments, and highlight areas of significant interest. Due to the novel nature of our clustering system and outputs, we include classical forms of clustering validation as well as our own interpretations of system effectiveness.

4.2.1 Internal Validation

Our initial internal validation results suggested significantly stronger performance with the sole usage of TF/IDF vectors for streaming clustering. This is likely due to the limitations of our UI workflow automation, resulting in the concatenated vector set not impacting results in a notable manner. Coupled with the lack of any smoothing or minimization functions made deriving meaningful results from the concatenated vector set difficult for the scope of this work. For these reasons, we shift our focus to the natural language vectors when gathering our current results.

When conducting internal validation, we leverage the silhouette score [59] and Normalized Mutual Information [74] (NMI) as evaluations of clustering quality. The silhouette score refers to an individual points vector similarity with other points within the same cluster. As existing work in the space has noted, scores over 0.5 are seen as "reasonably performant" for most clustering applications [59]. We take silhouette mea-

surements at each individual clustering iteration in our streaming model. These results are viewable below in Table 8. Our clustering model achieved strong internal validation scores across these iterations, with all but a single instance score falling in the commonly accepted range of performant clustering. These results may be observed in visual examples such as Figure 7, in which cluster membership groups are visually well defined in a two dimensional space.

Tested User	**Silhouette Coefficient (20 Samples)**	**Silhouette Coefficient (60 Samples)**	**Silhouette Coefficient (100 Samples)**	**NMI**
User 1	0.87	0.65	0.56	0.38
User 2	0.74	0.86	0.88	0.18
User 3	0.42	0.68	0.55	0.21

Table 1: Silhouette Coefficient and Normalized Mutual Information Scores

We additionally compare NMI with offline clustering to check for membership familiarity. We derive this approach largely from the work of Katz et al. [41], who show the effectiveness of offline clustering as a model for security group membership in a data loss prevention system. Due to the similarities of our methods, we take offline clustering data as ground truth for the purposes of this evaluation, and investigate clustering overlaps between both approaches. Prior work has noted the ability of NMI to indicate successful community modeling at scores of 0.1-0.3 [74]. Our own work scores within these ranges for all evaluated, with the added observation that differences in clustering performance are to be expected, given the additional vectorized data of the streaming dataset. Given all of these factors, our NMI evaluation makes a strong case for successful community modeling, as it preserves general community structure in all instances, as well as introduces an expected degree of novel group assignments given additional data.

4.2.2 External Validation

In this Section, we shift our evaluative focus from internal to external metrics of clustering performance evaluation. We detail a single experiment on a set of externally labeled data, and compare results with our streaming clustering algorithm. We compare performance with similarly high dimensional data clustering research.

Group Name	**Documents**
Personal	1, 2, 5, 9, 12, 15, 20
Business	4, 7, 8, 10, 11, 13, 17, 18, 19
Travel	3, 6, 16, 14

Table 2: Human Classifier on Example Dataset

Group Name	Documents
Group 1	15, 2
Group 2	1, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 17, 18, 19, 20
Group 3	16, 6, 14

Table 3: Clustering Classifier on Example Dataset

External validation involves the comparison of clustering approaches with a labeled classifier for an identical dataset. In our case, we have a human classifier assign labels to 20 documents, and apply our streaming clustering algorithm to the same 20 samples of data. The results of this labeling are three identified security groups corresponding to organizational role: "Personal," "Business," and "Travel." The direct classification and clustering results may be compared above in Figure 4. We leverage these insights to compute a more formalized metric for clustering effectiveness in purity score.

We record classification results and display purity measurements of our approach. Purity is defined formally as $\frac{1}{N}\sum_{i=1}^{k} max_j(c_i \cap t_j)$, in which k is the number of clusters. $(c_i \cap t_j)$ refer to an individual cluster, and the external classification label with which it shares the most data. A view of the confusion matrix for our experimental model can be seen below Table 4.

	Personal Data	Business Data	Travel Data
Cluster 1	2	0	0
Cluster 2	5	9	1
Cluster 3	0	0	3

Table 4: Streaming Clustering External Confusion Matrix for N=20

Based on these results, our purity score is equivalent to (2 + 9 + 3) / N = **0.7**. We additionally note that our clustering performs fairly well at topic approximation in regard to the identified groups, it correctly identifies the "Travel" group, as well as the distinction between the remaining two categories. Additionally, our purity metrics on this example are directly comperable to existing high-dimensional dataset external validation efforts, including the WebACE and TREC-5 dataset evaluations, which both average a **0.71** and **0.75** purity score across multiple clustering models [6]. This relative success indicates our models capacity to detect conceptual changes within text data.

4.2.3 Online Classification Performance

Finally, we evaluate the results of online clustering performance. The capacity for our algorithm to offer immediate group membership decisions is key to our performance goals. Given this, we evaluate online internal performance (silhouette score) and external performance in relation to the subsequent offline clustering iteration. Our results indicate strong performance of online clustering in relation to our offline model, suggesting

such an approach would prove effective for real time data security labeling in an endpoint environment.

For these experiments, we observe online performance for our test datasets of 100 samples. Due to the initial window clustering requirements, evaluation is performed on 80 examples. We present the final confusion matrix of classification decisions, as well as the resulting purity scores between the online and offline approaches. We additionally include online silhouette scores before the final clustering iteration. By including measures of both clustering quality and correctness, we seek to demonstrate that online, immediate labeling decisions carry similar predictive power to offline approaches. All of these metrics are displayed below in Table 5.

Tested User	**Online Correct Instances**	**Online Incorrect Instances**	**Purity Score**	**Final Online Silhouette Coefficient**
User 1	45	35	0.56	0.51
User 2	50	30	0.62	0.54
User 3	41	39	0.51	0.62

Table 5: Online Clustering Performance

Our online results are comparable to our offline evaluation in several instances. Silhouette coefficient scores remain in the same strong range as initial offline evaluations, though scores understandably decrease as our system nears the next clustering interval. The provided scores are taken at the end of a clustering iteration, or the theoretic worst point for scoring. Our external validation is compared with the offline clustering approach, and the resulting purity scores indicate a similar community model with our online data.

4.3 Section Summary

We identify several evaluation approaches and metrics for our online clustering approach. We find our approach produces strong internal validation scores, with silhouette coefficient being at or above standard performant levels. Our scores across multiple recorded iterations average out to **0.69**. Our external clustering purity evaluations additionally show promise in security group modeling when compared with a human classifier, achieving a purity score in our experiment of **0.7**. Finally, we include similar evaluative metrics for the online component of our work, showing weaker but acceptable scores for internal and external evaluation. This is to be expected given the nature of online clustering. Given the nature of our system and unsupervised learning, we expect clustering mistakes to be an unavoidable component of our work. However, due to our goal state of multi-environment endpoint systems, such mistakes do not necessarily imply security compromises

to a system. Such uncertainties may be countered through the usage of environment sandboxing, as we detail in prior Sections, and furthermore in the following component of our work.

5 Analyzing and Improving Usability of Sandboxing Environments

We aim to leverage the insights gained from our clustering approach in Section 4 in the context of supporting multiple, isolated environments in a single endpoint machine through containerization or virtualization. Currently, even in cases where isolation seems feasible for organizations, it is seldom used in the context of endpoint devices. This Section is dedicated to the investigation and analysis of isolation tools in endpoint environments, and the usability factors that prevent such tools from seeing widespread usage.

Existing isolation tools can help contain risk to prevent malicious activity from propagating across processes and systems. Virtual machines and containerization systems are designed to carefully manage and contain operating environments to enable independence from other computing activities on the same infrastructure. Given correct labeling, orchestrating isolation in multi-environment systems is a straightforward task, as shown in the Qubes operating system architecture [72]. This component of our work is concerned with the implementation of workflows to support isolation, even in cases without such specific group labels. The process of implementing isolation without a specific group membership/access controls is known as "sandboxing," which serves the primary focus of our efforts in this Section.

We consider sandboxing workflows as important to the foundation of multi-environment endpoint systems for two reasons. First, instances where a clustering approach produces uncertain results regarding group membership, or identifies an outlier in the clustering vector space, could make certain assets challenging to classify in our approach. In such scenarios, rather than "default deny" a user action, we instead propose a "default sandbox" approach, in which a workflow is permitted to continue in an environment with strict permissions and highly limited access to external resources. Second, end users often struggle with understanding isolation-centric security tools [45], given the nature of our system, the usability of isolation mechanisms in enterprise environments is necessary to consider. Even if we automate the challenge of environment labeling, users may still struggle to understand and conceptualize the idea of multiple security environments on the same machine. In our exploration of these areas, we aim to overcome not just the automated labeling of data assets, but also the streamlining of user interaction with those assets.

Within the domain of isolation, sandboxing and other mechanisms play a significant role in the cloud [60] and mobile devices [18], but are less commonly used in traditional endpoint systems, such as desktop or laptop computers. Instead, users of these devices typically have a single execution environment in which

assets and applications can be intermingled and affect one another. We detail the challenges that arise from this approach in Section 1. End-users need a way to divide their computational environments. Many users must interact with tools and assets of unknown origin that have not been vetted by their organization. They need the ability to do so without risking every other asset on their system, given the frequency of cyber attacks directly targeting end-users.

End-users additionally need to be able to use isolation techniques for their work. A barrier to doing so is likely the technical difficulties and usability obstacles inherent in today's tools, such as the QubesOS [72], VMWare [3], and VirtualBox [62] VM management tools. Prior work has found users often struggle to understand and leverage security tools, especially when interacting with multiple security domains [45]. Accordingly, we ask the following research questions: *What are the usability challenges with existing sandbox and isolation tools? What opportunities are there to improve the workflows related to isolation, in terms of productivity and understand-ability?*

To explore our research questions, we identify a concrete workflow that leverages isolation: the creation of a sandbox to experiment with a new tool or asset. We deconstruct this sandbox environment creation workflow into key components, such as actions and milestones, that are common across isolation tools. We define these components and clarify their relationships. We then implement this workflow in existing virtualization technologies. We measure how well the existing technologies work, identify improvement opportunities for the existing technologies, and implement a front-end prototype that incorporates such improvements. In doing so, we aim for a system that will be significantly easier for end-users with less complexity that will ask fewer questions, and will be faster to initiate. The system will require a new design for this specific workflow.

5.1 Identifying Workflows for Sandboxing Services

The goal of testing a new tool or exploring a data asset in a safe environment may be common in computing [49], but existing tools do have a standardized workflow for doing so. These tools have specific actions, such as "create a virtual machine" or "mount a shared folder," that can be used as steps in such a workflow, but they do not have a guided process to unify the steps in this process. In addition, many of these tools employ technically similar isolation techniques despite differences in naming convention. Other workflows, such as creating a virtual machine and creating a container, are technically distinct, but may be viewed as identical in an evaluative scenario of sandboxing procedures. We propose and explore the components of a standardized workflow and how they apply to existing tools that implement this idea.

Given the lack of existing standardization in sandboxing workflows, we initially examine the steps needed for such a workflow, and design a tool-independent standard workflow that is composed of actions, milestones, and intermediate states. We label these components for future reference and analysis across the tools we explore. By generating and applying such labels to sandboxing workflows, we aim to identify usability challenges with greater specificity to see the exact processes end users may find difficult.

In Figure 8, we show the workflow for creating a sandbox environment, with five distinct sub-workflows composed of steps and actions. These sub-workflows and actions were ubiquitously observable across a wide range of isolation-based tools, which lent further motivation for developing a standardized frame of analysis for these tools. We include an expanded version of this figure as a component of our overall approach toward multi-environment systems in Section 5.4.

When considering the workflow from a security perspective, it is helpful to consider varying degrees of trustworthiness of an environment. We declare that an environment is *pristine* when it has been created using trusted media by trusted experts following best security practices. For example, a "golden image" is often used as the base image for the persistent storage that will be used for computers or virtual machines. Such images are created using trusted installation media (e.g., software directly from a trusted software vendor) for all operating systems and software.

An environment can be considered *contaminated*, the opposite of pristine, once the environment receives untrusted input. Untrusted input can come from untrusted installation media, assets, storage, or network connections. An environment is also considered contaminated if it interacts with any person who is not a trusted expert following best security practices. Environment contamination does not necessarily imply a security vulnerability has been introduced, merely an degree of untrustworthiness associated with a specific environment.

Organizations may perform vetting on a contaminated environment to reclassify it to a pristine state. For example, after thoroughly analyzing all untrusted media or assets that resulted in a contaminated label, an organization may accept that environment as pristine and usable as a base image. Within the sandbox creation workflow, the transition from pristine to contaminated is unidirectional due to the scope of this component of our work. We further detail in this thesis how asset administrators may leverage this categorization in the approval of new access control policies based upon experimentation in Section 5.4.

The sandbox creation workflow begins with trusted media to construct a base image. With virtual machines, operating system installation media is used to format and populate an otherwise empty disk image. Once the installation is complete, the environment can be halted. At that point, the populated disk

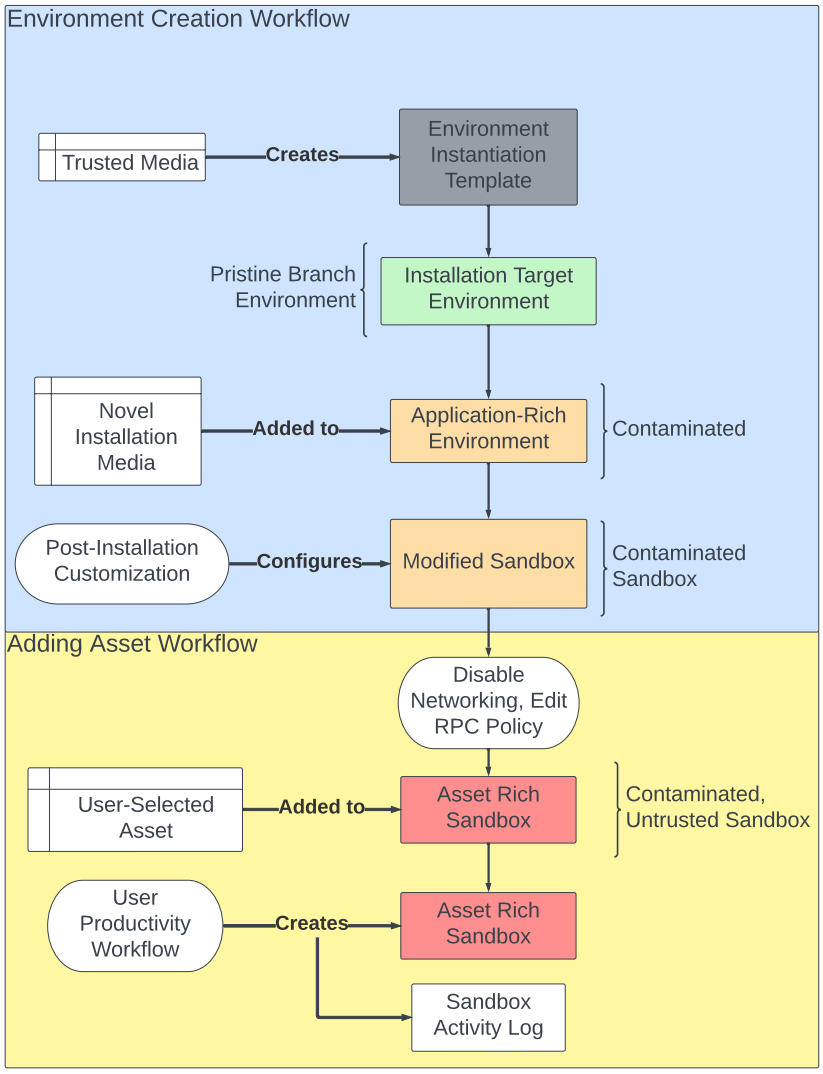


Figure 8: The sandbox creation workflow can be divided into an environment creation phase and an asset integration phase.

image can be used as a base image, a template that is copied each time a new environment is created. We refer to each copy of the base image as an "installation target environment." This target environment is what will be consumed by the remainder of the sandbox creation workflow. The target environment is pristine since it was created using trusted media by a trusted expert.

Once an end user begins interacting with the target environment in production, it becomes contaminated. When experimenting with new software, an end user will want to transform the "installation target environment" by providing untrusted installation media. This may require actions to install the new software, including configuration after the initial installation. The outcome is a sandbox environment. Up through this point, the sandbox environment is created without access to assets. It can interact with outside systems,

such as update servers, without risks to data confidentiality goals (i.e., the only confidentiality risks are exposure of the installed tools themselves). However, since untrusted media could contain malicious software, organizations may wish to limit the system to public network access and prevent the system from accessing other organizational systems to prevent threat propagation within the organization.

Once the sandbox is ready, we transition to the segment of the workflow in which data assets may be introduced. Once an asset is added, organizations must be careful about their security goals, particularly as it relates to confidentiality (e.g., preventing data leaks). The safest measure is to disable network access for the sandbox and to prevent write access to any storage devices (e.g., shared folders or mount points) that are not fully contained within the environment itself. The removal of write access prevents both confidentiality and integrity security goal violations related to assets. While the untrusted sandbox may maliciously alter or destroy assets copied into it, these protections prevent it from harming the original asset or environments outside the sandbox.

The next step is to optionally add assets to the sandbox environment. This step may not occur in cases where a user only wishes to experiment with new software. In the event they wish to use existing assets, or to work with an untrusted asset, the asset must be added to the environment. This can be done via read-only shared folders or network mount points. Many isolation tools provide such mechanisms for such data sharing, but they may not default to a read-only mode. This could lead users to unintentionally creating novel attack vectors that would undermine their security goals.

Once the asset is integrated, the end user will go through a series of steps to access the environment they have created. They can then perform arbitrary actions inside the sandbox to achieve their productivity goals. This workflow would be identical to traditional workflows performed in non-sandboxed environments. All outputs produced by the sandbox would be restricted to the sandbox environment. Other workflows could then be used to extract and analyze any outputs before allowing their use outside the sandbox.

Not all isolation tools follow this specific workflow for sandboxes. Some programs, such as Microsoft Application Guard [14], implement similar workflows in the background, potentially without the user knowing that sandboxing is in use. This does limit the sandboxing to specific scenarios and trust models. For the Application Guard tool, the applications themselves are included in the trusted computing base (TCB). The transparent nature of these tools may remove the ability to intentionally experiment from the end-user, or increase the difficulty of integrating the results of sandboxing into a broader isolation-based system.

Type 1 hypervisors are isolation tools that have fewer dependencies in their TCBs, namely that of the hardware and the hypervisor itself. Type 2 hypervisors, such as those in VMWare, VirtualBox, or

Parallels [27], have a host operating system that must be included in the TCB as well. Containerization isolation tools, such as Docker, have a similar TCB as Type 2 hypervisors with the additional requirement to allow the contained environment to directly interact with the host operating system APIs rather than interacting with a virtualized OS.

As a result of this analysis, we have identified the following workflow components:

- **Pristine Installation Target Creation:** This component is the creation of an isolated workspace that will serve as the environment for a user to execute untrusted software or work with untrusted assets.
- **Insertion of Untrusted Media:** In this component, a user may optionally insert untrusted software tools into a pristine environment. Connecting untrusted media removes the environment's "pristine" label.
- **Software Setup/Customization:** This component performs any customization or setup needed to make the software tool ready to run and function properly within a sandbox environment.
- **Insertion of Asset(s) to Target Environment:** This component allows a user to copy a data asset, or a set of data assets, into the sandbox environment in a read-only fashion. When assets will be inserted, the environment must be configured to meet confidentiality security goals (e.g., read-only interfaces, network isolation).
- **Environment Use:** This component is where the end user accesses and operates the sandbox environment.

In the remainder of this work, we analyze isolation tools along these workflow components.

5.2 Usability Analysis of Existing Tools

With our sandboxing workflow, and the labeled components of that workflow, we examine the usability aspects of current isolation tools. Usability can be analyzed in a variety of ways, including usage scenarios and user studies. As noted in Section 2.3, the usability community has developed techniques to quantify the complexity of user interactions to enable comparisons across workflows.

We focus on using the Keystroke Level Model (KLM) approach to quantify the interactions with isolation tool interfaces. With KLM, we can calculate a time estimate for the activity required to perform a workflow in a UI. KLM divides activities into five groups [22]: user keystrokes (K), user pointing the cursor (P), user

hand movements transitions to/from keyboard and mouse (H), user clicking the mouse (B), user mental preparation (M), and the system response time (R). Since we are focused on the usability and complexity of operations in the UI, we omit the system response time (R) from our analysis. These operations can be concatenated together into KLM strings. For example, the string "KHPB" would denote a keystroke, repositioning a hand to the mouse, pointing the cursor, and clicking the mouse.

KLM Action	Interaction Time (seconds)
Keystroke T(K)	0.28
Button Press T(B)	0.10
Pointing with a mouse T(B)	1.1
Switching hands between devices T(H)	0.4
Mental Preparation Time	1.35
System Response Time	N.A.

Table 6: KLM characters and their respective interaction times

We test four isolation tools available for end-users: VMWare Workstation Player, VirtualBox, Docker Desktop, and the Qubes OS. We examine the KLM strings associated with each of the components of the workflow and with the string representing all the steps to complete the workflow. We developed a keylogging program in Python to record our execution of the each isolation tool workflow. As we perform actions, our program records the corresponding KLM actions (e.g., button presses, mouse clicks) and formats each action into a KLM character. We use this program while executing the workflows in each tool.

Our tool functions by having a skilled user approach a tool with the goal of creating a sandboxing environment in which to test a piece of software. Experimentation is performed through the usage of our Python keylogger, which features on-screen buttons for beginning and ending a test. During testing, the keylogger uses the Pynput library to record all keystrokes and mouse button interactions a user performs. Upon finishing the workflow, a user immediately presses the button to end the test. To account for this action, the final move and button press characters are excluded from the KLM string. The program then outputs the results of the testing, in the form of a simplified KLM string consisting of a string of "K","P", "B", and "H". As system response ("R") and mental thought ("M") times are not infer-able directly though a keylogger, our system additionally outputs a simple estimation of these times though a subtraction function of the KLM estimated time from the total time between beginning and ending of a workflow.

We then convert these KLM string representations into a common unit, time (in seconds), to enable comparisons. Using the same formulas outlined by Card et al. [19], we calculated an execution time for each workflow. We use standard estimation times for each KLM character, viewable in Table 6, these times are derived from prior work in usability observations, as detailed in the work of Kieras et al. [44] and others. While KLM experimentation involves a skilled user that approaches tasks in an optimized manner, prior

work has been successful in generalizing results of the metric to wider population sizes of varying technical expertise.

In Table 7, we show the outcome of the KLM calculations for the Qubes OS as an example of the analysis. By analyzing the KLM measurements by workflow components, we can see which components require different tyes of user engagement. From this, we can see that the creation of an installation target takes the most time. We further see that the installation of untrusted media requires a significant amount of mouse activity. This data serves as the framework for our further experimentation, and we detail these efforts in the following Section 5.2.1.

Table 7: KLM Analysis of the Qubes OS to Create an Environment

	Aggregate Action Time (in seconds)				
Workflow Component	***Keystroke T(K)***	***Mouse Button T(B)***	***Mouse Point T(P)***	***Hand Moves T(H)***	***Mental Thought T(M)***
Installation Target Creation	2.5	0.9	9.9	0.8	5.4
Insertion of Untrusted Media	0.8	0.7	8.8	0.8	4
Software Setup/Customization	0	1.2	5.5	0.8	4
Total Time (Seconds)	3.3	2.8	24.2	0.24	13.4

5.2.1 Aggregate KLM Usability Results

Using these tools and methods, we ran an experiment to identify usability challenges in existing isolation tools. We conducted these experiments with the tools and techniques outlined previously. We organized our cross-tool analysis and task comparison by workflow component. Through the usage of our keylogger, this enables direct comparisons of tasks such as creating environments or adding newly downloaded software to them.

For each tool, we perform the entire workflow while running our key-logging software to take note of all mouse clicks (T(B)), mouse points (T(P)), keyboard strokes(T(K)), and moving of hands between keyboard and mouse (T(H)). We then add mental preparation time (T(M)) to each string, based the set of heuristics outlined by Card et al. [19], after the recording is finished. Due to the nature of our sub-workflow graph, we manually mark points of the KLM string to denote the end of each sub-workflow during testing, though the usage of a specific keystroke command known to the tester. This allows for the entire workflow to be completed in a single testing session, while additionally providing delineated KLM string metrics for each of the other sub-workflows.

Our experiment begins in all cases from a desktop (or home screen) in an operating system. The same user performed tests on all tools to minimize any variation in usage patterns. In the case of software such as VMWare, shorcuts were placed on the desktop to allow for efficient tool access, as well as consistent workflow start times between experiments. We then sum the resulting KLM values to create completed execution times for each individual workflow component. We omit system response time (T(R)) and we limit all text entry actions to the minimum allowed by each system. This produces an optimal scenario for each tool for comparison. We then process these strings to derive execution time and other significant values.

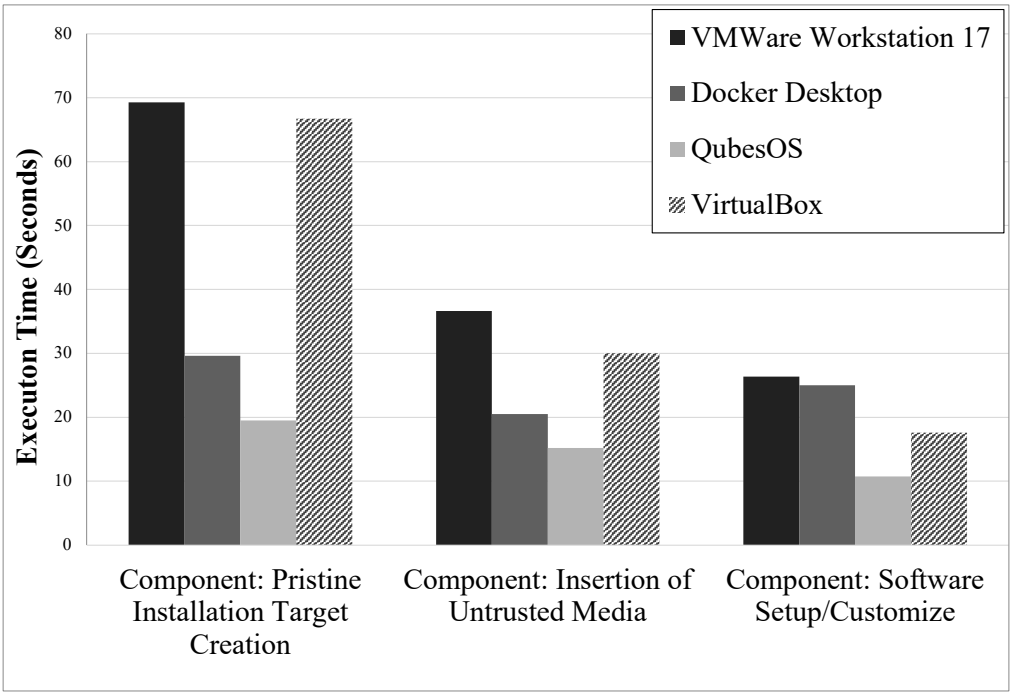


Figure 9: Environment Creation Workflow timing results

Our results show that some isolation tools struggle in specific areas while others are more successful in those areas. In Figures 9 and 10, we show the KLM execution times by workflow component and tool. Both Type 2 hypervisors tested, VMWare and VirtualBox, required significant user time to create an environment, specifically the "Installation Target Creation" component. In contrast, Docker, a container-based tool, was faster for creating an environment, but its command-line interface used a relatively high amount of time to enable data asset movement. VMWare, VirtualBox and Docker have high keystroke (T(K)) measurements in environment creation and configuration due to the need to manually name environments and enter login information to access environments after creation. Inserting untrusted media and data assets into a target environment required higher volume of mouse interaction (T(B) and T(P) scores) for both VMWare and VirtualBox in contrast to the high keyboard activity in Docker. This is due to the amount of pointing and

clicking required to manually select an asset from the host file system.

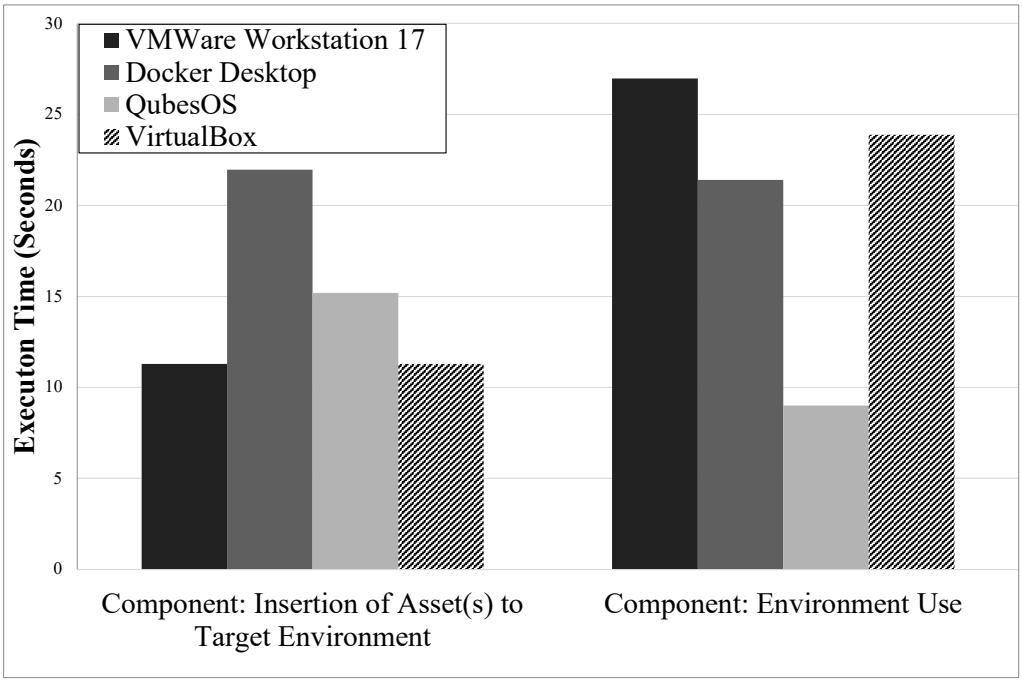


Figure 10: Asset Movement Workflow timing results

The Qubes OS is designed for an isolation-centric endpoint. It required the least amount of time for all the Environment Creation Workflow components and on the Environment Use component while scoring third in Insertion of Assets to Target Environment component. Qubes resolves the high keyboard interaction scores of previous tools with its UI design, but its menu navigation can be complex. Users need to access a wide range of menus to create the isolation environment. There are opportunities to optimize this by merging tasks, several of which we detail in our own work to optimize the usability of these workflows.

5.2.2 Designing a Novel Sandboxing Workflow

Based on the UI challenges with the isolation tools, we explored a prototype implementation of an sandboxing workflow. We use the standard "Wizard of Oz" technique [42] in which an interface is developed without a corresponding backend implementation for usability evaluations. We create an interface to simplify the sandboxing workflow. We use Adobe XD, a user interface design and evaluation tool, to create interactive mock-ups. In doing so, we keep in mind the underlying actions that would be needed in a system to implement the interface. In Figure 11, we show the interface of our sandbox configuration tool, including all interactive elements of the tool.

Our tool simplifies base templates using a drop-down menu, eliminating the need for file system navigation

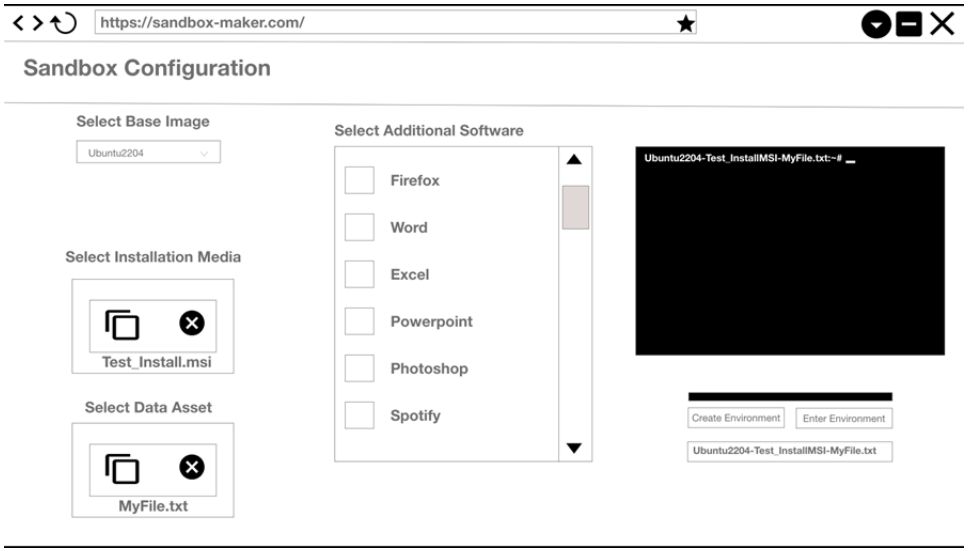


Figure 11: Our prototype interface with all components of the tool displayed as in the window.

for template selection for a sandbox. We use templates, rather than disk images or VM snapshots, to reflect a goal of reusing designated trusted image sources with simpler terminology. Similarly, we also designate environment identifiers based on the assets and tools selected in the sandbox construction window. We envision an environment organizational hierarchy using a tree of images from trusted sources. For the end user, we provide a software list, similar in appearance to the Qubes OS AppVM creation tool, that allows users to further customize their environment. We also offer immediate selection boxes for installation media and data assets that users could want to move into a sandboxing environment. By asking the user upfront, we can better understand the user's goal and generate labels for the sandbox. These changes can simplify the entire environment creation workflow as compared to existing tools.

For interacting with the environment, we include an in-browser remote desktop window, similar to services such as Microsoft Cloud PC [14]. This allows users to immediately access a sandbox environment that is automatically named and viewable upon creation, which simplifies the environment entry workflow. The actual running system may be on remote infrastructure or hosted locally. The assumption of a web service allows for us to retain flexibility in future implementations of our system.

The Adobe XD tool enables actuation of UI elements, allowing us to perform usability testing in the same manner as with existing tools. System response time is not simulated as it would vary based on hardware. The program workflow is designed such that workflows may be started, interrupted, and restarted in the same way as a real piece of software.

5.3 Novel KLM Usability Results

After the development of our custom tool, we perform identical experimentation with our keylogger. Testing was performed with the adobe XD file closed and accessible from the user's desktop, in the same manner as previously evaluated software. We consider the test finished when the user has engaged in the entire program workflow by clicking the "Enter Environment" button, although no environment will be produced. After experimental analysis, we determined that our prototype leads to faster end-user interactions for each workflow component. These results are observed in both our identified workflow categories of "Environment Creation" and "Adding Assets." We outline specific results for both categories inline.

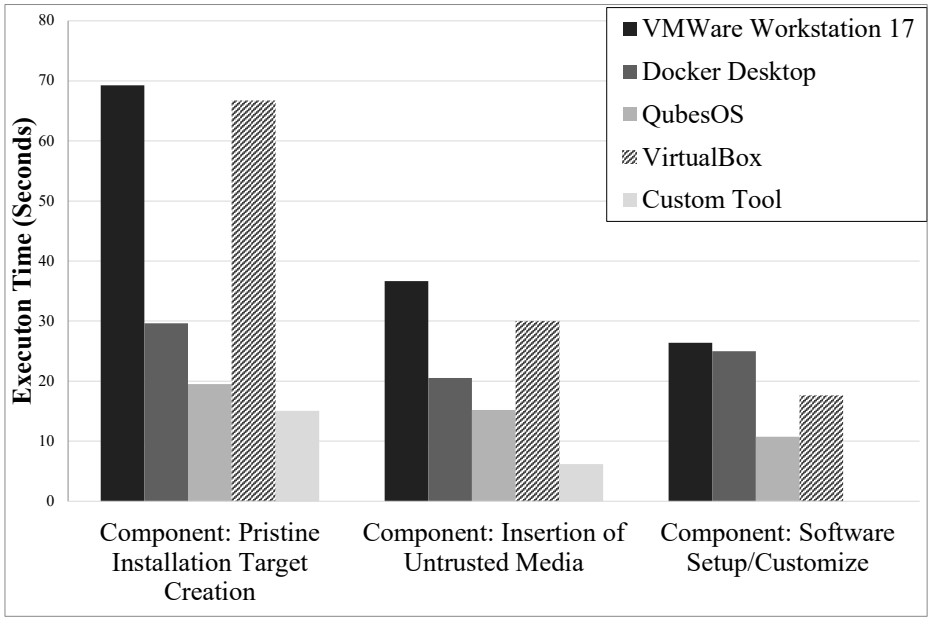


Figure 12: Environment Creation Workflow with our Prototype

While many underlying processes are abstracted from user interaction within our proposed tool, the core functionality of configuring and customizing sandbox environments as described in our "environment creation" workflow framework component matches other tested tools. Within the environment creation category of sub-workflows, we draw significant improvement over all tools in the "Insertion of Untrusted Media" component, as process-specific automation removes the need for complex movement of data across trust domains. Our decreased UI complexity leads to an average reduction of **67.48%** in the final execution time in KLM measurements across all sub-workflow components in the environment creation category when compared with Qubes, the best prior performer. These improvements may be inferred in Figure 12. We present complete results in Table 8 One component,"Software Setup/Customization," is eliminated altogether since the "Insertion of Untrusted Media" and "Insertion of Assets to Target Environment" steps happen

before the sandbox is created, allowing any further installation effort to be automated prior to (or during the) "Environment Use" step.

We observe similar performance improvements in the "Adding Asset" sub-workflow category, our tool preserves and improves the capacity of streamlined asset insertion as seen in virtual machine infrastructure through the inter-machine file sharing or a mounted file system. Our tool not only abstracts the associated configuration procedures required for an end user, but streamlines the process further with a simple selection tool of the asset a user wishes to add to an environment. The most noticeable improvement is the entering and usage of a novel environment, in which our single button press tool allows for a significantly faster workflow than the the GUI navigational requirements of type 2 hypervisors or Qubes-OS. Our tool attains an average **68.98%** reduction in KLM measurements over existing tools for this sub-workflow. Asset insertion is additionally streamlined, but still requires a single instance of file system navigation in our example tool, leading to a **58.50%** reduction in KLM measurements. Further abstraction for this sub-workflow is likely to be challenging, due to the user engagement requirements for asset selection.

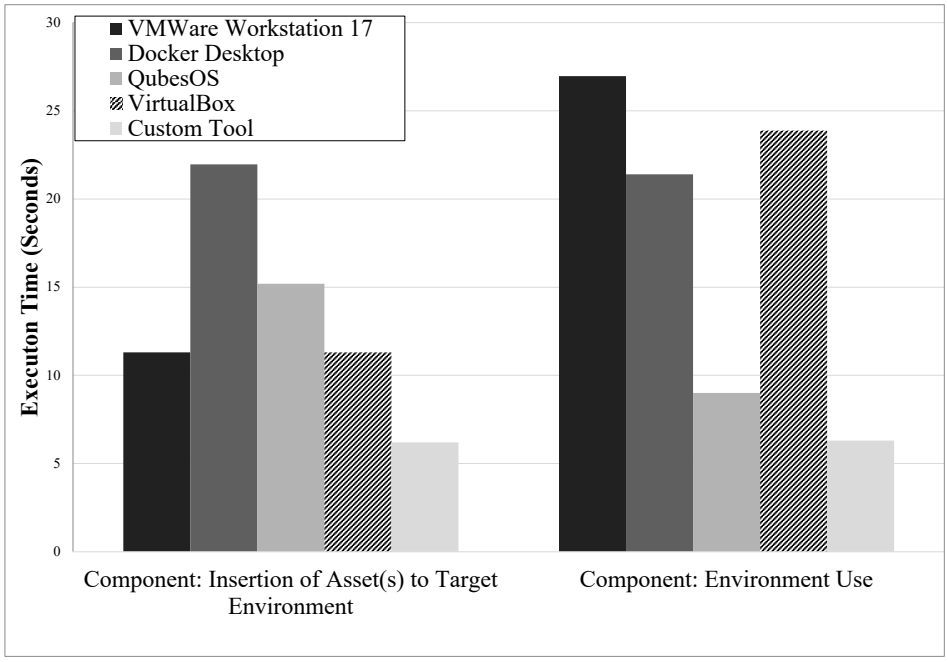


Figure 13: Adding of Asset Workflow with our Prototype

Our empirical KLM results indicate the specific areas in which existing sandboxing and isolation tools have usability limitations in an approach that identifies specific tasks and sub workflows. Particularly notable cases of improvement were found in the "Software Setup/Customize" sub-workflow, as well as the "Environment Use" sub-workflow. Upon experimentation involving our own tool, we note substantial reductions in KLM completed execution times when compared with all existing tools. This is done while retaining functionality

from a user perspective regarding sandboxing workflows. As with many instances of software abstraction, the end user loses a degree of interactive flexibility when leveraging our tool as opposed to a traditional hypervisor or containerization program. The goal behind this approach is not to replace with these tools altogether, but rather to strip away the more complex components to allow for a greater degree of usability in enterprise endpoint systems.

sub-workflow Component	**Average Execution Time of Existing Tools (Seconds)**	**Custom Tool Execution Time (Seconds)**	**Seconds Percentage Decrease**
Pristine Installation Target Creation	46.08	15.05	67.48%
Insertion of Untrusted Media	25.59	6.2	75.77%
Software Setup/Customize	19.94	0	100%
Insertion of Asset(s) to Target Environment	14.94	6.2	58.50%
Environment Usage	20.31	6.3	68.98%

Table 8: Average Component Execution Times and Percentage Improvement

5.4 Section Summary

We noted existing isolation tools as having usability challenges in several components of their workflows. We observe user workflows associated with these tools specifically in the context of orchestrating sandbox environments to test novel tools or data assets. When evaluated using Keystroke Level Modeling, we see that the different tools have varying strengths and weaknesses in their workflow simplicity, but present opportunities for usability optimization. We design a prototype user interface to address usability challenges by reducing workflow steps and overall interface complexity. Our custom tool achieves significantly lower KLM evaluation scores than existing workflows though a combination of abstracting sandboxing components and streamlined interface design.

While KLM can provide important comparison points for tool interfaces, it cannot fully capture the usability challenges in these systems. The best-scoring current tool, Qubes, has terminology and concepts that are specific to that technology. While our focus was on workflow optimization, we also simplified the vocabulary and used existing workflow metaphors to aid end-user comprehension. Future work could potentially involve the analysis and simplification of technical terminology in tools such as QubesOS. Other

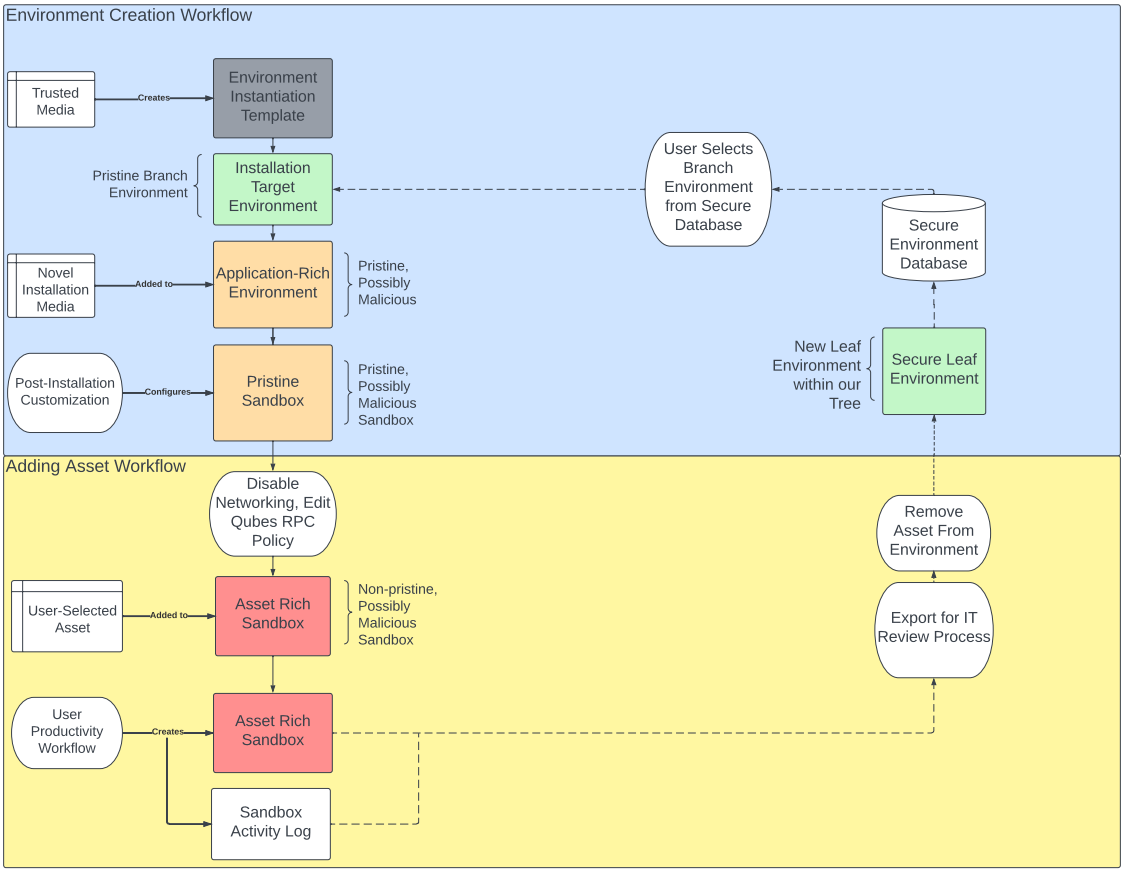


Figure 14: The sandbox creation workflow can be divided into an environment creation phase and an asset integration phase.

strains of future work could involve quantifying the understandability of isolation tools for end-users. Given the common usage of transparent sandboxing in the past, cases where users must directly engage with multiple environments or isolation in a workflow present important case studies into the applicability of usable isolation in the future.

Additional future work includes a working backend infrastructure to support the prototype UI along with full user studies. Future work could also examine implementations across endpoint devices, such as templates managed by an organization and end-users that wish to share sandboxes for collaboration. Finally, work could examine cloud-hosted, locally-hosted, and hybrid-hosted isolation environments, and their impact on usability and performance for end-users. This work could additionally be extended to the impact on organizational systems of data management, with a focus shift from asset management to environment management.

As a direct point of integration with the complete work of this thesis, we propose the inclusion of

sandboxing into a multi-environment endpoint system with clustering-based security labels. Sandboxing can correct for uncertainty in clustering output decisions, as well as allow for the experimentation of entirely novel workflows that may serve as the basis for future policy expansion. A view of this integration may be seen in Figure 14, in which our sandboxing workflow is expanded to encompass data asset administrators, who may observe the results of sandboxing before deciding to allow similar actions in the future. We provide further detail of this workflow in Section 6, which is dedicated to the fusion of our work components in the creation of multiple security environments in endpoint systems.

6 Summary of Foundation of Multi-Environments

In this Section, we detail the fusion of our work in the previous two Sections. We discuss how the technical contributions of this thesis allow for an improved understanding of secure, scalable multi-environment endpoint systems for enterprise environments. We examine the impact of dynamic security labels and the caveats of unsupervised learning as a label generation mechanism. We explore how such caveats necessitate the usage of additional isolation mechanisms such as sandboxing to ensure security. Finally, we document and visualize our complete proposed approach for isolation-centric endpoint security.

Our streaming clustering approach for asset security labeling provides an automated solution for labeling large quantities of previously unlabeled data assets. A complete visualization of our approach, including currently unimplemented components may be viewed below in Figure 15. Despite the labor-preservation and rapid labeling advantages such a model provides, its reliance on unfiltered unsupervised learning outputs is understandable cause for concern. The challenges and general uncertainties evoked through unsupervised learning and other AI methods have been well documented [73]. Such challenges can be addressed in many instances through the usage of layered security elements, removing a single point of failure associated with an AI-based system. Our system is unique in that clustering outputs are not designed to be subject to human review before being utilized as security labels for data assets, as such, we propose the usage of additional isolation tools, including those discussed in the second component of our work 4.

We previously discuss in Section 5 our conceptualization of "default sandboxing" and how isolation may be used as a solution for uncertainties in clustering output. The usage of sandboxing allows for user workflows to be preserved in the event of an uncertain labeling output. Depending on a user's intended workflow, this could happen frequently. Additionally, even in the event of a fully accurate labeling system, certain policies regarding software will more than likely remain part of an organization's access control approach. It is for these reasons we developed a usability-focused approach to sandbox environment provisioning from

the perspective of the end user. By promoting the usage of sandboxing and experimentation in a user's daily workflow, we aim to minimize the productivity-loss associated with poor labeling outputs, through empowering the end-user to continue to perform their work in a secure manner.

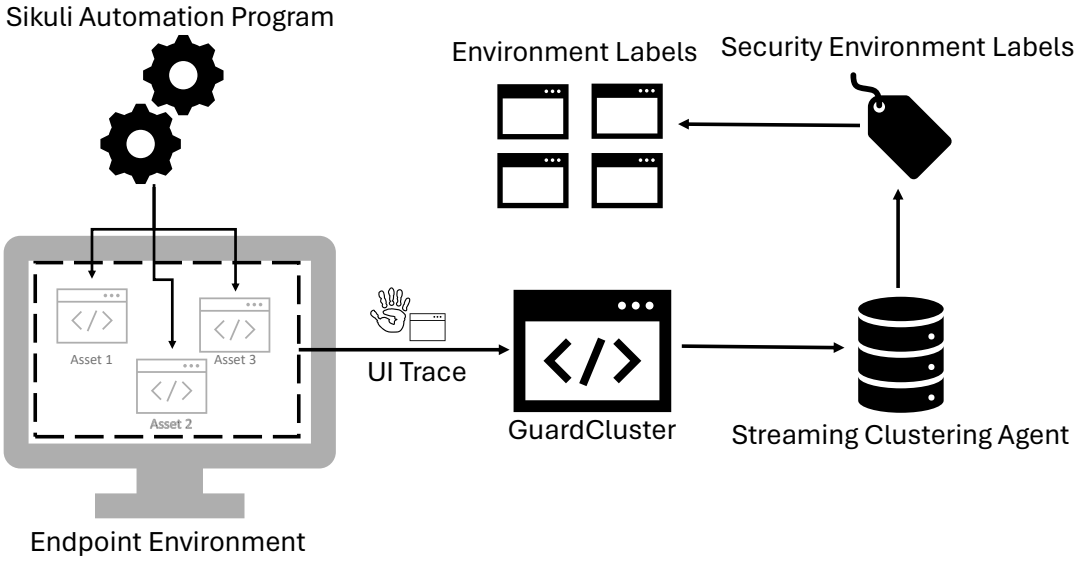


Figure 15: Our Full Implementation Scheme

In the current system implementation of our work, clustering is performed on a remote host after UI data is sent over a socket connection after collection, on a per endpoint basis. No network encryption provisions are currently involved in our system design. If a system were to continue the usage of clusters per endpoint, we propose the usage of encryption in a remote connection, such as a block cipher utilizing Galois/Counter Mode [57] to guarantee trace confidentiality and integrity. This approach allows for high data throughput while retaining the remote clustering approach we currently leverage. Additionally, this approach allows for the implementation of clustering systems involving data multiple endpoints, providing output to a collection of machines instead of a single one.

Alternatively, clustering could be performed locally to avoid the challenges of secure data transmission. Many type 1 hypervisor systems contain a single administrative domain or "dom0," in which driver and toolstack management is performed. Access to this domain is heavily restricted, due to such elevated permissions. In a similar manner, we propose the usage of a dedicated "clustering domain" for executing our unsupervised learning algorithm. This approach may be configured to preserve inter-domain confidentiality requirements, while providing localized clustering results.

7 Conclusion and Future Work

In this thesis, we explore and develop isolation technologies to promote stronger, more usable access control schemes in enterprise systems. We determine existing challenges with the "singular environment" model of endpoint devices in relation to security access controls. These challenges include an expanded attack surface for cyberattacks, complexity in policy management, and a "default-deny" approach to novel end user workflows. We identify "multi-environment" endpoints and isolation-centric security as a potential solution to these issues, although these environments also come with their own challenges.

To better accommodate for multi-environment endpoints in modern enterprise systems, we first introduce and develop an unsupervised clustering model for dynamic asset security labeling. These labels serve as the basis for security environment membership on an endpoint device. We design and implement GuardCluster, a data collection tool that formats UI trace data to include relevant program window information, as well as on-screen natural language data. We leverage BIRCH streaming clustering as a mechanism for assigning group membership labels through hierarchical clustering of UI data into conceptual groups. We accomplish this through the conversion of UI trace values into data vectors associated with a given asset, and concatenating these vectors with the TF/IDF vectorization of natural language data. Our results show strong performance of clustering using internal validation scoring approaches. These results are supported by strong external validation as well, in which our model manages to correctly identify community models when compared with a human classifier. Finally, online performance is not as strong as these approaches, but still allows for performant group classification decisions based on prior clustering outputs.

We additionally explore the usability challenges with existing isolation-centric security approaches, such as virtual infrastructure and containerization. We identify that the usability challenges in these tools are a barrier to adoption in modern endpoint systems, as well as an issue to address in the construction of multi-environment endpoint systems. We identify a standard, tool independent workflow for the purposes of using isolation as an experimental technique, or "sandboxing." We extend this model to analyze a wide variety of isolation tools, including type 1 hypervisors, type 2 hypvervisors, and containerization software. We find that traditional tools suffer from quantifiable usability challenges when analyzed through Keystroke Level Modeling in the context of our workflow model. We leverage these insights to design and implement a Wizard of Oz version of a new environment creation tool. This tool achieves significantly improved results on all measured sub-workflows.

Future work in this space is likely to expand our clustering approach to include additional streaming

clustering methods, or introduce additional layered security controls to account for potential unsupervised learning labeling mistakes. Additional UI data as well as other forms of context, such as network event logging, could also be included in trace data generated by GuardCluster. The results of this data could additionally be compared against offline clustering and labeling approaches. Additionally, federated learning could serve as a privacy preserving approach for supporting multiple systems connected to a single clustering algorithm. For the second component of this thesis, an expanded usability evaluation, one that potentially accounts for language understandability could be a potential area of future work. Additionally, a working backend infrastructure for our prototype Wizard of Oz program, that leverages usage of remote resources would be a natural topic for future efforts. Finally, usability evaluations of different technical approaches, including locally hosted environments, remote hosted environments, and hybrid models including the usage of "hot migration" techniques for environment hosting.

The usage of multiple isolated environments in endpoint systems represents a major improvement for security outcomes across a wide range of cyberattacks. This work proposes several strategies key to the adoption of these technologies in enterprise environments, and how they allow for the management of configurational complexities that such multi-environment systems currently require. This work additionally evaluates and improves upon the usability of existing toolsets used to implement endpoint isolation. Our results suggest our clustering and usability strategies are not only feasible, but lead to demonstrable performance improvements in both areas.

References

[1] How many hats do you wear? Electronic, https://www.zenbusiness.com/blog/how-many-hats-do-you-wear/, Jul 2020.

[2] Luka Abb, Carsten Bormann, Han van der Aa, and Jana R. Rehse. Trace clustering for user behavior mining. ECIS 2022, 2022.

[3] Keith Adams and Ole Agesen. A comparison of software and hardware techniques for x86 virtualization. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XII, page 2–13, New York, NY, USA, 2006. Association for Computing Machinery.

[4] Adobe. Adobe XD platform. Electronic, https://adobexdplatform.com/, 2023.

[5] Shiroq Al-Megren, Joharah Khabti, and Hend Al-Khalifa. A systematic review of modifications and validation methods for the extension of the keystroke-level model. *Advances in Human-Computer Interaction*, 2018:1–26, 12 2018.

[6] Ramiz M. Aliguliyev. Performance evaluation of density-based clustering methods. *Information Sciences*, 179(20):3583–3602, 2009.

[7] Ibrahim Alobaidan, Michael Mackay, and Posco Tso. Build trust in the cloud computing - isolation in container based virtualisation. In *2016 9th International Conference on Developments in eSystems Engineering (DeSE)*, pages 143–148, 2016.

[8] Izzat Alsmadi and Ikdam Alhami. Clustering and classification of email contents. *Journal of King Saud University - Computer and Information Sciences*, 27(1):46–57, 2015.

[9] Khudran Alzhrani, Ethan M. Rudd, C. Edward Chow, and Terrance E. Boult. Automated u.s diplomatic cables security classification: Topic model pruning vs. classification based on clusters. In *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–6, 2017.

[10] Nicolas Aussel, Yohan Petetin, and Sophie Chabridon. Improving performances of log mining for anomaly prediction through nlp-based log parsing. In *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 237–243, 2018.

[11] Michael Backes, Sven Bugiel, Christian Hammer, Oliver Schranz, and Philipp Styp-Rekowsky. Full-fledged app sandboxing for stock android. 08 2015.

[12] D. Balfanz, G. Durfee, D.K. Smetters, and R.E. Grinter. In search of usable security: five lessons from the field. *IEEE Security & Privacy*, 2(5):19–24, 2004.

[13] Hitesh Ballani, Yatin Chawathe, Sylvia Ratnasamy, Timothy Roscoe, and Scott Shenker. Off by default! ACM, November 2005.

[14] Andrea Barr, Dan Wesley, Radia Soulmani, David Coulter, and Colleen Williams. Microsoft edge and microsoft defender application guard. Electronic, learn.microsoft.com, Aug 2022.

[15] A. Beauchaine and C. A. Shue. Toward a (secure) path of least resistance: An examination of usability challenges in secure sandbox systems. In *2023 5th IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 240–246, Los Alamitos, CA, USA, nov 2023. IEEE Computer Society.

[16] Sergey Bratus, Alex Ferguson, Doug McIlroy, and Sean Smith. Pastures: Towards usable security policy engineering. pages 1052–1059, 04 2007.

[17] Bartosz Brodecki, Jerzy Brzeziński, Piotr Sasak, and Michał Szychowiak. Consistency maintenance of modern security policies. In P. Santhi Thilagam, Alwyn Roshan Pais, K. Chandrasekaran, and N. Balakrishnan, editors, *Advanced Computing, Networking and Security*, pages 472–477, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[18] Sven Bugiel, Lucas Davi, Alexandra Dmitrienko, Stephan Heuser, Ahmad-Reza Sadeghi, and Bhargava Shastry. Practical and lightweight domain isolation on android. In *ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, page 51–62, New York, NY, USA, 2011. Association for Computing Machinery.

[19] Stuart K. Card, Thomas P. Moran, and Allen Newell. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, 23(7):396–410, Jul 1980.

[20] Zorigtbaatar Chuluundorj, Curtis R. Taylor, Robert J. Walls, and Craig A. Shue. Can the user help? leveraging user actions for network profiling. In *2021 Eighth International Conference on Software Defined Systems (SDS)*, pages 1–8, 2021.

[21] Enron Corp and William W Cohen. Enron email dataset. Software, E-Resource. https://www.loc.gov/item/2018487913/.

[22] Daniel Cunha, Rui P. Duarte, and Carlos A. Cunha. KLM-GOMS detection of interaction patterns through the execution of unplanned tasks. In *Computational Science and Its Applications*, pages 203–219. Springer International Publishing, 2021.

[23] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1:224–227, 1979.

[24] Andy De George. Windows presentation foundation desktop guide, Jun 2024.

[25] D.Ramel. Stack overflow: Old. net framework usage still beats 'most loved'. net core/.net 5, 2021.

[26] Virgil Gligor. Security limitations of virtualization and how to overcome them. In Bruce Christianson and James Malcolm, editors, *Security Protocols*, pages 233–251, Berlin, Heidelberg, 2014. Springer.

[27] Parallels International GmbH. Parallels technical documentation. Electronic, https://www.parallels.com, 2023.

[28] Li Gong and Xiaolei Qian. Enriching the expressive power of security labels. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):839–841, 1995.

[29] Michael J Guess and Scott B Wilson. Introduction to hierarchical clustering. *Journal of clinical neurophysiology*, 19(2):144–151, 2002.

[30] Nentawe Gurumdimma, Arshad Jhumka, Maria Liakata, Edward Chuah, and James Browne. Towards detecting patterns in failure logs of large-scale distributed systems. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, pages 1052–1061, 2015.

[31] José María Gómez-Hidalgo, José Miguel Martín-Abreu, Javier Nieves, Igor Santos, Felix Brezo, and Pablo G. Bringas. Data leak prevention through named entity recognition. In *2010 IEEE Second International Conference on Social Computing*, pages 1129–1134, 2010.

[32] Michael Hart, Pratyusa Manadhata, and Rob Johnson. Text classification for data loss prevention. In Simone Fischer-Hübner and Nicholas Hopper, editors, *Privacy Enhancing Technologies*, pages 18–37, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[33] Michaela Ngaropaki Teresa Hart, Pratyusa K. Manadhata, and Rob Johnson. Text classification for data loss prevention. In *International Symposium on Privacy Enhancing Technologies*, 2011.

[34] Mohammad Rahat Helal, Weiqing Sun, and Ahmad Javaid. Efficient isolation enabled role-based access control for database systems. 07 2017.

[35] Raimund Hocke. Raiman's sikulix, 2017.

[36] Shufeng Huang, James Griffioen, and Ken Calvert. Pvns: Making virtualized network infrastructure usable. In *ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 147–148, 2012.

[37] Melody Y. Ivory and Marti A Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys*, 33(4):470–516, Dec 2001.

[38] Yong Gu Ji, Jun Ho Park, Cheol Lee, and Myung Hwan Yun. A usability checklist for the usability evaluation of mobile phone user interface. *International Journal of Human–Computer Interaction*, 20(3):207–231, 2006.

[39] Ronald Kainda, Ivan Fléchais, and A.W. Roscoe. Security and usability: Analysis and evaluation. In *International Conference on Availability, Reliability and Security*, pages 275–282, 2010.

[40] Christos Katsanos, Nikos Karousos, Nikolaos Tselios, Michalis Xenos, and Nikolaos Avouris. KLM form analyzer: Automated evaluation of web form filling tasks using human performance models. In *Human-Computer Interaction*, pages 530–537, Berlin, Heidelberg, 2013. Springer.

[41] Gilad Katz, Yuval Elovici, and Bracha Shapira. Coban: A context based model for data leakage prevention. *Information Sciences*, 262:137–158, 2014.

[42] John F. ("Jeff") Kelley. Wizard of oz (WoZ): A yellow brick journey. *J. Usability Studies*, 13(3):119–124, May 2018.

[43] David Kieras. A guide to GOMS model usability evaluation using NGOMSL. In Marting G. Helander, Thomas K. Landauer, and Prasad V. Prabhu, editors, *Handbook of Human-Computer Interaction*, pages 733–766. North-Holland, Amsterdam, 1997.

[44] David E. Kieras. Using the keystroke-level model to estimate execution times. *University of Michigan Published Papers*, 2003.

[45] François Lesueur, Ala Rezmerita, Thomas Herault, Sylvain Peyronnet, and Sébastien Tixeuil. Safe-os: A secure and usable desktop operating system. In *2010 Fifth International Conference on Risks and Security of Internet and Systems (CRiSIS)*, pages 1–7, 2010.

[46] Hua Li, Dou Shen, Benyu Zhang, Zheng Chen, and Qiang Yang. Adding semantics to email clustering. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 938–942, 2006.

[47] Hui Li, Ying Liu, Jun Liu, Xia Wang, Yujiang Li, and Pei-Luen Rau. Extended KLM for mobile phone interaction. In *CHI EA '10: CHI '10 Extended Abstracts on Human Factors in Computing Systems*, pages 3517–3522, 04 2010.

[48] Shangsong Liang, Emine Yilmaz, and Evangelos Kanoulas. Dynamic clustering of streaming short documents. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 995–1004, New York, NY, USA, 2016. Association for Computing Machinery.

[49] Martina Lindorfer, Clemens Kolbitsch, and Paolo Milani Comparetti. Detecting environment-sensitive malware. In *Recent Advances in Intrusion Detection*, pages 338–357, Berlin, Heidelberg, 2011. Springer.

[50] Fortra LLC. Fortra. Electronig, https://dataclassification.fortra.com/, 2023.

[51] Zi Long, Lianzhi Tan, Shengping Zhou, Chaoyang He, and Xin Liu. Collecting indicators of compromise from unstructured text of cybersecurity articles using neural-based sequence labelling. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019.

[52] Moshe Looks, Andrew Levine, G. Adam Covington, Ronald P. Loui, John W. Lockwood, and Young H. Cho. Streaming hierarchical clustering for concept mining. In *2007 IEEE Aerospace Conference*, pages 1–12, 2007.

[53] J. MacQueen. Some methods for classification and analysis of multivariate observations. 1967.

[54] Robert Mazzoli. Learn about microsoft purview, Jan 2024.

[55] Bill McCarty. *SELinux: NSA's Open Source Security Enhanced Linux*. O'Reilly Media, Inc., 2004.

[56] Jonathan M. McCune, Trent Jaeger, Stefan Berger, Ramon Caceres, and Reiner Sailer. Shamon: A system for distributed mandatory access control. In *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, pages 23–32, 2006.

[57] David A. McGrew and John Viega. The security and performance of the galois/counter mode (gcm) of operation. In Anne Canteaut and Kapaleeswaran Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004*, pages 343–355, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[58] Luís Filipe da Cruz Nassif and Eduardo Raul Hruschka. Document clustering for forensic analysis: An approach for improving computer inspection. *IEEE Transactions on Information Forensics and Security*, 8(1):46–54, 2013.

[59] Hieu Ngo, Hua Fang, Joshua Rumbut, and Honggang Wang. Federated fuzzy clustering for decentralized incomplete longitudinal behavioral data. *IEEE Internet of Things Journal*, 11(8):14657–14670, 2024.

[60] Lin Ni, Huanqing Cui, Mingqian Wang, Depeng Zhi, Kun Han, and Wangli Kou. Construction of data center security system based on micro isolation under zero trust architecture. In *Asia-Pacific Conference on Communications Technology and Computer Science*, pages 113–116, 2022.

[61] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-data algorithms for high-quality clustering. In *Proceedings 18th International Conference on Data Engineering*, pages 685–694, 2002.

[62] Oracle. Virtualbox technical documentation. Electronic, https://www.virtualbox.org/wiki/Technical_documentation, 2023.

[63] Sylvia Osborn, Ravi Sandhu, and Qamar Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Trans. Inf. Syst. Secur.*, 3(2):85–106, may 2000.

[64] Harun Oz, Ahmet Aris, Albert Levi, and A. Selcuk Uluagac. A survey on ransomware: Evolution, taxonomy, and defense solutions. *ACM Computing Surveys*, 54(11s), Sep 2022.

[65] Vitaly G. Promyslov, Kirill V. Semenkov, and Alexander S. Shumov. A clustering method of asset cybersecurity classification. *IFAC-PapersOnLine*, 52(13):928–933, 2019. 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019.

[66] Giovanni Russello, Mauro Conti, Bruno Crispo, and Earlence Fernandes. Moses: Supporting operation modes on smartphones. In *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*, SACMAT '12, page 3–12, New York, NY, USA, 2012. Association for Computing Machinery.

[67] M Angela Sasse and Ivan Flechais. Usable security: Why do we need it? how do we get it? In *Security and Usability: Designing secure systems that people can use.*, pages 13–30. O'Reilly, 2005.

[68] Keeper Security. Privileged access management survey: User insights on cost complexity. Technical report, Keeper Security, 2022.

[69] Manoj Sethi, Kunal Batra, Manali Biswas, and Mehar Lamba. An automated system for identification of tweets requiring customer service concern. In *2022 3rd International Conference for Emerging Technology (INCET)*, pages 1–6, 2022.

[70] Ketan Rajshekhar Shahapure and Charles Nicholas. Cluster quality analysis using silhouette score. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 747–748, 2020.

[71] Xian-He Sun, Cong Du, Hongbo Zou, Yong Chen, and Prerak Shukla. V-mcs: A configuration system for virtual machines. In *IEEE International Conference on Cluster Computing and Workshops*, pages 1–7, 2009.

[72] Michael Carbone Wojtek Porczyk, Marek Marczykowski-Górecki. Qubes-os statistics. Electronic, qubes-os.org/statistics, Apr 2023.

[73] Roman V. Yampolskiy and M. S. Spellchecker. Artificial intelligence safety and cybersecurity: a timeline of ai failures, 2016.

[74] Pan Zhang. Evaluating accuracy of community detection using the relative normalized mutual information. *Journal of Statistical Mechanics: Theory and Experiment*, 2015(11):P11006, November 2015.

[75] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96, page 103–114, New York, NY, USA, 1996. Association for Computing Machinery.

[76] Alaettin Zubaroglu and Volkan Atalay. Data stream clustering: A review. *CoRR*, abs/2007.10781, 2020.