# Toward a (Secure) Path of Least Resistance: An Examination of Usability Challenges in Secure Sandbox Systems

1st Adam Beauchaine
*Department of Computer Science*
*Worcester Polytechnic Institute*
Worcester, United States
ajbeauchaine@wpi.edu

2nd Craig A. Shue
*Department of Computer Science*
*Worcester Polytechnic Institute*
Worcester, United States
cshue@wpi.edu

***Abstract*—The computing hardware and software tools for virtualizing and isolating execution environments have matured, enabling a paradigm shift towards cloud computing and containerization. Unfortunately, the use of these technologies in traditional computing is largely limited to servers and systems administered by professional IT teams. At another extreme, mobile device operating systems extensively leverage isolation to prevent malicious activity from mobile applications. In stark contrast, despite possessing the resources to support virtual machines, traditional endpoint systems tend to have a single execution environment for all of a user's activity.**

**In this work, we explore the usability challenges that may prevent widespread use of isolation mechanisms, such as virtual machines, on traditional computing endpoints. We explore systems using a common workflow, where a user wants to experiment with a new software tool, to compare existing virtualization systems and motivate a new design. We use keystroke-level modeling to quantitatively compare systems and identify optimization opportunities.**

***Index Terms*—Usable Security, Isolation, System Sandboxing, Usability Modeling**

## I. Introduction

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. This has enabled the cloud computing paradigm [4] and was key to protecting mobile applications in smartphones [37].

While isolation mechanisms play a significant role in the cloud [33] and mobile devices [10], they 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 each other. End-users need a way to divide their computation 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 user-oriented cyber attacks. The recent prevalence of ransomware attacks is an example, which has resulted in a roughly $20 billion dollar loss in 2021 [35]. It has also led to widespread damage to corporations, public services, and higher education institutions [13], [18], [36]

End-users need to be able to use isolation techniques for their work. A barrier to doing so could be the technical difficulties and usability obstacles inherent in today's tools, such as the QubesOS [41], VMWare [1], and VirtualBox [34] VM management tools. Prior work has found users often struggle to understand and leverage security tools, especially when interacting with multiple security domains [29]. 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.

In this work, we make the following research contributions:

- *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 (Section III).
- *Usability Analysis of Current Isolation Tools*: We employ the Keystroke Level Model (KLM) [11] 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 four existing tools: VMWare, VirtualBox, Docker [21], and QubesOS [41] (Section IV).

- *Prototype Interfaces for the Sandbox Workflow*: We create a prototype isolation tool front-end using Adobe XD [2]. 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, highlighting how similarly designed tools may approach the sandboxing workflow differently (Section V).

## II. Background and Related Work

We review prior work in security isolation tools, usable security, and usability metrics. We examine prior work to improve existing tool workflows.

The security advantages of sandboxing tools are well established across a wide range of systems [5], [6], [9], [14], [17], [32]. From a usability perspective, endpoint sandboxing can be transparent in cases, such as the Microsoft Windows Sandboxing feature analyzed by Đuranec et al. [42], or the OS X Chromium-based sandbox Seatbelt [40]. However, the overall adoption of endpoint sandboxing remains low [19], [41]. The Qubes OS is an example operating system designed for endpoints to isolate applications into different zones of trust. Qubes reported just under 40,000 users at the beginning of 2023 [41], which is significantly lower than Linux distributions without isolation. Since these technologies are proven to increase end user security, adoption hurdles such as usability may be an issue. We consider Gligor's [15] 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 importance of usability as a security consideration has been noted across a range of analyses [7], [24], [38]. The usability hurdles in sandboxing workflows are not often discussed; however, they have been noted in several works. In SAFE-OS [29], the authors design and implement a sandboxing system that adds several feautures 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. [39] 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. [20] 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.

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) [27], as well as other evaluations and proposed standards [22], [23]. Keystroke level modeling (KLM), a successor of the GOMS model, has been widespread in system usability evaluations since Card et al. [11] 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 [3]. 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 [3], [25], as well as usage across platform interfaces [12], [30]. We draw upon these use cases in designing our own procedures for sandbox evaluation.

Building on these insights, we explore usability in the adoption of endpoint sandboxing. Taking modeling techniques and applying them to sandboxing tools, we identify usability limitations and build a prototype interface aimed at preserving functionality and expanding usability.

## III. Creating Sandbox Environments

The goal of testing a new tool or exploring a data asset in a safe environment may be common in computing [31], but existing tools do not seem to 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. We explore the components of a standardized workflow and how they apply to existing tools that implement this idea.

Given the lack of 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 1, we show the workflow for creating a sandbox environment, with five distinct sub-workflows composed of steps and actions. These sub-workflows and actions seem common across the tools we explored.

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.

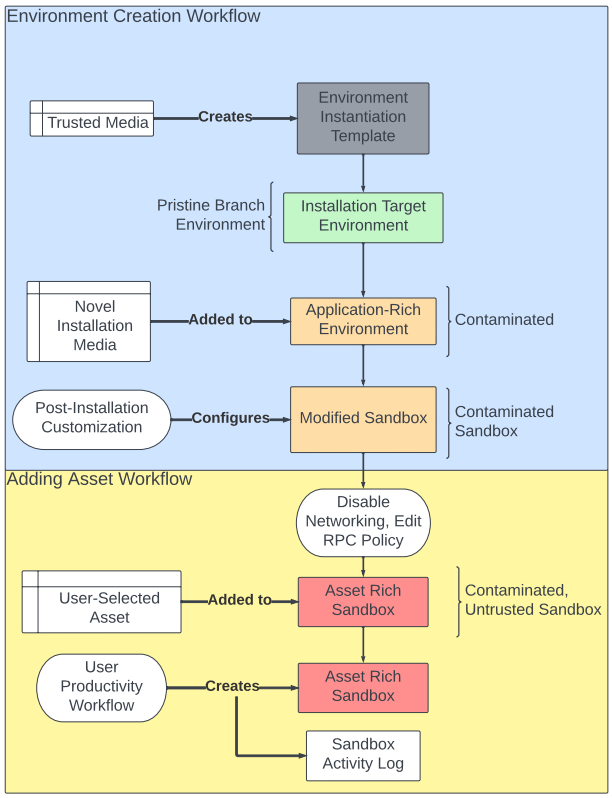


Fig. 1. The sandbox creation workflow can be divided into an environment creation phase and an asset integration phase.

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.

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. Since we scope this work to focus on the sandbox creation workflow, we do not further discuss the reclassification workflow. Within the sandbox creation workflow, the transition from pristine to contaminated is unidirectional.

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 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.

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 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. 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 [8], 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).

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 [16], have a host op-

erating 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.

## IV. Evaluating Current Isolation 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 II, 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 [12]: 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.

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.

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. [11], we calculated an execution time for each workflow. We use standard estimation times for each KLM character, with K=0.28s B=0.1s, P=1.1s, H=0.4s, and M=1.35s [28].

TABLE I
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 |

In Table I, 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.

### A. 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. 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. [11], after the recording is finished. We then sum these values to create completed execution times for 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.

Our results show that some isolation tools struggle in specific areas while others are more successful in those areas.

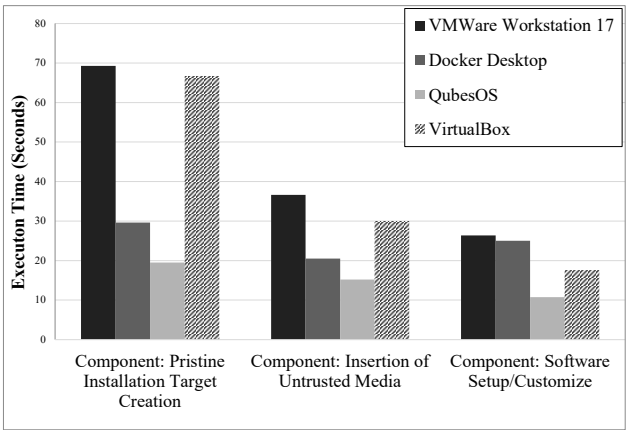


Fig. 2. Environment Creation Workflow timing results

In Figures 2 and 3, 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 keyboard stroke (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.

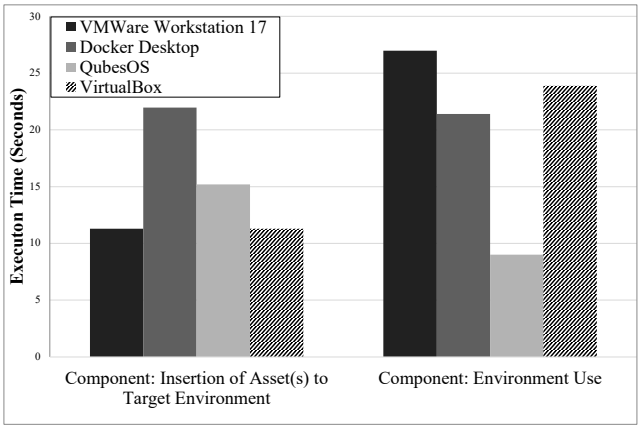


Fig. 3. Asset Movement Workflow timing results

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.

## V. Creating a Novel Sandboxing UI

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 [26] 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 4, we show the interface of our sandbox configuration tool, including all interactive elements of the tool.

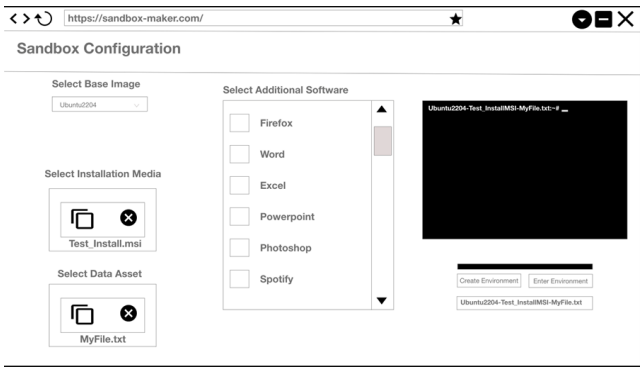


Fig. 4. Our prototype interface with all components of the tool displayed as in the window.

Our tool simplifies base templates using a drop-down menu, eliminating the need for file system navigation 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 [8]. 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 Adobe XD tool enables actuation of UI elements, allowing us to perform usability testing in the same manner as with existing tools.

### A. Novel KLM Usability Results

Our prototype leads to faster end-user interactions for each workflow component. 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 the "Environment Use" step to automate any needed customization.

Our decreased UI complexity leads to an average reduction of 66.24% in the final execution time in KLM measurements across all workflow components when compared with Qubes, the best prior performer. While many underlying processes are abstracted from user interaction within our proposed tool, the core functionality of configuring and customizing sandbox environments, and the capacity to name and enter them, matches other tested tools.

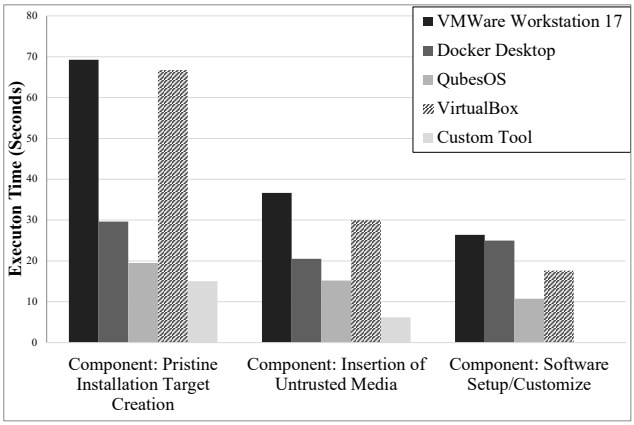


Fig. 5. Environment Creation Workflow with our Prototype

Our empirical KLM results show the specific areas in which existing sandboxing and isolation tools have usability limitation in an approach that identifies specific tasks and sub workflows. We leverage this information in the construction of our own proposed tool, aimed at reducing keyboard usage and menu complexity. Upon further 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.

## VI. Discussion

Existing isolation tools have usability challenges in several workflow components associated with creating a sandbox environment to test 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. We design a prototype UI to address usability challenges by reducing workflow steps and interface complexity. Our custom tool has lower KLM times than existing tools.

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 in this space may include 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 usability for end-users.

## References


[1] 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.

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

[3] 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.

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

[5] Ömer Aslan and Refik Samet. Investigation of possibilities to detect malware using existing tools. In *IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, pages 1277–1284, 2017.

[6] Stanley Bak, Karthik Manamcheri, Sayan Mitra, and Marco Caccamo. Sandboxing controllers for cyber-physical systems. In *IEEE/ACM International Conference on Cyber-Physical Systems*, pages 3–12, 2011.

[7] 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.

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

[9] Ashish Bijlani and Umakishore Ramachandran. A lightweight and fine-grained file system sandboxing framework. In *Asia-Pacific Workshop on Systems*, New York, NY, USA, 2018. Association for Computing Machinery.

[10] 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.

[11] 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.

[12] 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.

[13] Lorenzo Franceschi-Bicchierai. Hackers claim vast access to western digital systems. electronic, techcrunch.com, Apr 2023.

[14] Tal Garfinkel, Ben Pfaff, Jim Chow, Mendel Rosenblum, and Dan Boneh. Terra: A virtual machine-based platform for trusted computing. *SIGOPS Oper. Syst. Rev.*, 37(5):193–206, oct 2003.

[15] 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.

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

[17] Chris Greamo and Anup Ghosh. Sandboxing and virtualization: Modern tools for combating malware. *IEEE Security and Privacy*, 9(2):79–82, 2011.

[18] Joel Griffin. Lincoln college closure a testament to the threat posed by ransomware. electronic, securityinfowatch.com, Jun 2022.

[19] Justin Henderson and John Hubbard. 2019 SANS survey on next-generation endpoint risks and protections. electronic, SANS.org, Jan 2023.

[20] 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.

[21] Docker Inc. Docker docs: How to build, share, and run applications. Electronic, https://docs.docker.com/, 2023.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

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

[29] François Lesueur, Ala Rezmerita, Thomas Herault, Sylvain Peyronnet, and Sébastien Tixeuil. SAFE-OS: A secure and usable desktop operating system. In *International Conference on Risks and Security of Internet and Systems*, pages 1–7, 2010.

[30] 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.

[31] 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.

[32] Stuart E. Madnick and John J. Donovan. Application and analysis of the virtual machine approach to information system security and isolation. In *Proceedings of the Workshop on Virtual Computer Systems*, page 210–224, New York, NY, USA, 1973. Association for Computing Machinery.

[33] 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.

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

[35] 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.

[36] Carly Page. Ransomware attack forces dallas to shut down courts, disrupts some 911 services. electronic, techcrunch.com, May 2023.

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

[38] 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.

[39] 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.

[40] Robert NM Watson, Jonathan Anderson, Ben Laurie, and Kris Kennaway. Capsicum: Practical capabilities for unix. In *USENIX Security Symposium*, volume 46, page 2, 2010.

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

[42] A. Đuranec, S. Gruičić, and M. Žagar. Forensic analysis of windows 10 sandbox. In *International Convention on Information, Communication and Electronic Technology (MIPRO)*, pages 1224–1229, 2020.