Using Portable Virtualization for Exclusively-Public Computer Users

Jon Volden
University of Kansas

Jacob Marshall
University of Kansas

Walter Goettlich
University of Kansas

Matt Comi
University of Kansas

Sarah Smith
University of Kansas

William Staples
University of Kansas

Perry Alexander
University of Kansas

Drew Davidson
University of Kansas

Abstract

Computing resources have become social and economic necessities, but the way in which those resources are accessed is influenced by larger issues of social and economic inequality. Those who do not have their own computer to access digital resources must rely on public computing systems such as those offered by public libraries. These users face a unique set of challenges: their computing lives are resource-limited, transient, and less private and secure than those who own their computers.

In this work, we investigate usable security and privacy solutions for those who rely exclusively on public computers. This work requires re-envisioned threat models and technical solutions that do not rely on frequently-made assumptions about computer ownership.

1 Introduction

Many social and economic expectations of modern society assume that individuals own a personal computer and have high-speed internet access. When this expectation is not met, individuals must rely on shared, public computers, such as those offered at public libraries. Numerous inequalities stem from this reliance, both in the suitability of the public computer for social and economic expectations, and in threats to the security and privacy of the exclusive- or primarily-public computer user. While access to library computers is important, their current operation does not provide the social function of a personal computer and thereby disadvantage users reliant on these systems. One of the key limitations is that the state of the machine is reset between sessions using *reboot to restore* software such as Faronics Deep Freeze [4]. Deep Freeze conceptually creates a base image and restores the base image periodically (such as when a user logs out of the public computer), rolling back any changes made during the session. Deep Freeze gives users a consistent software configuration, and helps administrators to keep a "clean" image. However, the rollback is not without negative consequences for security and privacy tools (and usability in general). Fundamentally, many of the assumptions of "typical" personal computer use rely on two key properties: *personalization* and *persistence*. We use personalization to refer broadly to a user's ability to customize his or her computing experience; the user should be able to install programs necessary to fulfill the task at hand and to configure these tools to the extent available to a personal computer user. We use persistence to refer broadly to the ability of a user to maintain data, both in the form of personalization preferences and data processed as part of the work that they are doing.

One reality of presenting solutions to exclusively-public computer users is that the population of such users tends to be correlated with low socioeconomic status and relatively low computer literacy skills. As such, it is unrealistic to expect these users to operate software programs that deviate significantly from what they would use otherwise. Our experience in our user study has shown that most users are familiar with the Microsoft Office suite, browsers such as Firefox, Chrome, and Microsoft Explorer/Edge. Even seemingly similar tools, such as the Google Drive suite instead of the Microsoft Office, causes difficulty for these users.

In order to address the major considerations outlined above while still meeting the unique challenges of this domain, we propose a prototype solution that we call the PUPS (for personalized user privacy and security). The

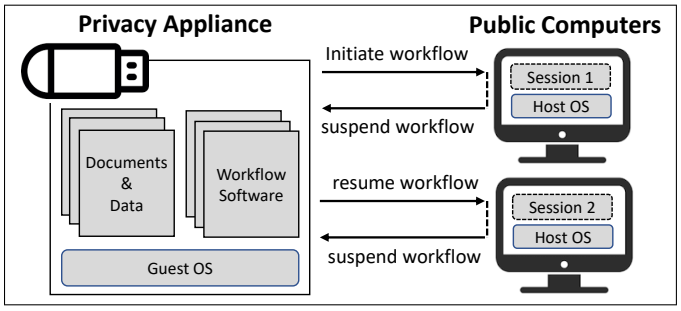

Figure 1: The isolated environment on the USB device allows the user to maintain private data and programs securely across multiple public-use base computers.

PUPS is manifested as a commodity hardware storage device (such as a USB key), loaded with customized OS-level virtualization software and an OS image. When the user of a public computer needs to begin a session, they insert the PUPS into the public computer, and boot into their virtual environment. Since the OS image and the corresponding software travel on the device, the user need not rely on third-party infrastructure such as a cloud service or software on the host system in order to work in a customized, persistent environment.

2 Overview

The PUPS is designed to offer tools and technologies that most users have already used. From a hardware perspective, the use of a commodity USB key presents a familiar analogy to users; it is the physical representation of their personal digital space. By building on virtual machine technology, users can leverage a familiar guest environment without worrying about the details of the host operating system. In our observations, library systems almost exclusively run Windows operating systems. The most direct consequence of this observation is that the guest machine should run Windows as well.

The PUPS Workflow: Figure 1 provides a high-level outline for how the PUPS can be used throughout several sessions: The user starts by inserting the PUPS device into the public host computer. The user then invokes a custom launcher from the mounted drive, which is visible as a startup application on the USB-mounted drive. The PUPS contains the virtual environment and modified virtual machine, so no software is required on the host; the launcher scans the local machine for optimal configuration, then invokes the virtual machine application from the USB device. Once the virtual environment is launched, it will (by default), enter a full-screen configuration to avoid confusing the user between host and guest applications. The user may freely use applications and data within the guest environment. When the session ends, the user shuts down the guest environment, retrieves and USB key, and is able to pick up on another machine.

Deploying a specialized virtual machine via a USB key helps to restore the key considerations of *personalization* and *persistence*: the user has the ability to install programs as desired within the guest machine without privileges on the host machine (since those changes do not impact the security of the host). Furthermore, since state persists on the USB key, the changes are maintained across multiple work sessions and are not subject to the restore-on-reboot behavior of the host.

The PUPS offers a unique vantage point to install a number of personalized privacy and security tools that would otherwise be unavailable to unprivileged users, relying on the key factors of personalization and persistence. These tools are pre-configured and installed on the PUPS image, but they can (and in some cases, must) continue to be updated throughout the lifetime of the PUPS. As such, there are several benefits to the PUPS in its current form for security and privacy: The isolated nature of the virtual machine prevents infection from the host, and the availability of a persistent environment re-enables existing state-of-the-art security tools.

3 Technical Details

In this section, we discuss how implementation details of our system, and discuss some of the protection mechanisms that we implemented. A number of recent deployment factors have enabled the opportunity to deploy a system like the PUPS in a practical setting. In addition to explaining our implementation effort, we point out some of these factors.

At its core, the PUPS is a modified version of the hosted virtual machine monitor QEMU [1]. We took several steps to adapt QEMU for our use case. The majority of our efforts involve configuring the virtual machine monitor to run seamlessly from the USB device, to leverage features that improve performance without burdening the user, and to disable mechanisms that are not supported and could confuse or frustrate users. We considered other open-source alternatives to QEMU, such as VirtualBox [2]. Consistent with prior work, we found

```
X:\qemu-system-x86_64w.exe
   -hda X:\windows10.img
   -boot c
   -usb -device usb-tablet
   -accel whpx
   -machine q35
   -cpu qemu64
   -m 3G
   -vga virtio -sdl
   -full-screen
```

Figure 2: Example command to run the QEMU virtual machine used within the PUPS, where X is the drive letter assigned to the USB device.

that VirtualBox implemented a number of features that made it faster than QEMU [3]. However, QEMU met a number of deployment constraints not offered in other systems: QEMU can be run without requiring runtime administrator permissions and needs little configuration on the host machine to run. However, failure to properly configure QEMU causes it to run so slowly as to render the system unusable. Due to the popularity of virtualization platforms, we believe that these features will continue to see widespread adoption.

We implemented a small number of software modifications in order to support a more seamless UI/UX experience when running QEMU from portable media. We required full-screen emulation, seamless mouse and keyboard input, and minimal user interaction required to begin and end sessions. Along with our timing data and profile requirements we enable features through command line parameters that suit our needs.

Many requirements are met with simple static command line switches. However, minimizing user interaction requires custom software to handle the details for the host/guest session. We use a separate program to handle details pertaining to the host machine. This software, simply called launcher, collects information about the host machine and generates optimal command parameters for that machine. Some of this information includes memory size, total CPU cores, USB speed of current port, internet connection, Windows Hypervisor availability and others. Users can run the launcher's executable from the USB device to begin a new virtual session.

One key enabling technology that underlies our implementation is the improvement in USB transfer speeds. For our prototype implementation of the PUPS, we used the SanDisk Extreme PRO USB 3.1 (Model: SDCZ880-128G). This product offers an advertised write speed of 380 MB/s and an advertised read speed of 420 MB/s. Previous work profiled this type of device at lower specifications (264 MB/s write, 297 MB/s read), which is consistent with our use case. Nevertheless, these speeds exceed the performance of many modern HDD hard drives; the Western Digital WD Black HDDs advertises read and write speeds of 256 MB/s.

Although we do not consider the selection of USB drives to be a contribution of our work, we note that the ready availability of such devices indicates why solutions such as the PUPS are timely; small, external media can meet the transfer speeds necessary to have a near-native performance, especially for non-memory intensive operations.

The primary security goal of the PUPS is to protect a population of users that typically have a low level of technical literacy. In addition to providing a platform that is intrinsically isolated from attacks on the public host machine, an advantage of the PUPS approach is that it allows the distributor of PUPS devices to preload the environment with privacy-preserving, security-enhancing tools. For our prototype deployment, we also pre-load the browser configurations with more privacy-conscious options, such as sending the "Do not track" request and enabling the Privacy Badger extension to help block privacy-invasive third parties. We note that a key feature of the PUPS is its ability to admit persistent customization. As such, users are entirely free to roll back these protections; the PUPS is non-prescriptive, but simply enforces a more private-by-default stance. We note that the software deployed within the virtual environment deserves significant study in its own right.

4 Evaluation

In this section we evaluate different characteristics of the PUPS. The characteristics discussed include speed, size, and complexity of user interaction. Our initial experiments show that although the PUPS does incur performance overhead, we believe that it is acceptable for the use cases for which it is intended to be deployed: the PUPS has an average startup and shutdown time under 1 minute, works on the types of machines that are in use as

Conf	Default	+RAM	CPU	Speed
A	01:01.78	00:53.46	00:51.56	00:19.19
B	00:50.65	00:46.76	N/A	00:15.22
C	00:51.98	01:14.67	00:56.30	00:15.69

Table 1: Results for the performance tests. Note that AMD Ryzen does not have a CPU profile to test.

public computers, and can withstand common misuse.

To test the applicability of our tool, we considered three configurations: an Intel i7-8700 @ 3.20GHz, 32GB RAM, 512GB NVMe, VT-x, USB 3.1 (Conf A), an AMD Ryzen 5 PRO 2400GE, 8GB RAM, Samsung 240GB SSD, AMD-V, USB 3.1 (Conf B), and an Intel i5-7400T @ 2.40GHz, 8GB RAM, Samsung 120GB SSD, VT-x, USB 3.0 (Conf C).

Conf A, a private development machine, served as our development testbed for creating and refining the PUPS. Conf B and Conf C represent two configurations that we repeatedly saw in real public computer facilities.

Since we expect that the main facilities for providing public host computers will be public libraries, an important design constraint for the PUPS is to ensure that it runs on the hardware commonly employed by public libraries.

At a minimum, the QEMU software requires a few virtualization options enabled. The Intel VT-x or AMD-V virtualization must be enabled in the BIOS. The host Windows 10 machine must have Windows Hypervisor Platform enabled in the Windows Features menu. These features are standard on all systems manufactured after 2015. All of the machines that we observed in actual public computer facilities, met these requirements.

The speed of the PUPS is an important concern when considering end-user usability. In order to find optimal QEMU settings for speed, basic timing tests, shown in Table 1, are performed to obtain a baseline duration for each work session.

We ran four tests on three machines to test key factors in emulation speed, memory size and CPU profile. The first three tests were simple startup and shutdown sequences using different QEMU settings. The **Default** test uses recommended settings by QEMU developers and community, described in Figure 2, that ensure compatibility for the widest range of 64 bit computers. The **+RAM** test uses the default settings and increases memory to 6 GB. **CPU** test uses the default settings and changes the CPU flag to match the host system, when available. (Note: AMD Ryzen does not currently have a QEMU CPU profile.) A **Speed** test performs a basic common function like opening and closing a web browser. For some activities performance is a limiting factor. Tests show usability is acceptable for the simple use cases we target but futher optimization is needed.

To test how robust the PUPS is to operational disruption, we attempt to purposely corrupt the PUPS guest image by unplugging the USB device during operation. While this scenario is catastrophic for the work session, we hypothesizing that such events will happen in practice due to an impatient user or on accident. To our surprise, we did not corrupt the guest image in any of our trials. While we do not believe that pulling a USB key during operation is ever advisable, we believe that the setup is at least somewhat robust to corruption.

5 Conclusion

In conclusion, we believe portable, whole-system virtualization is a promising direction for enhancing the security, privacy, and utility of exclusively-public computer users. Our current PUPS implementation constitutes an promising prototype.

References

[1] Fabrice Bellard. QEMU, a fast and portable dynamic translator. In *USENIX Annual Technical Conference, FREENIX Track*, volume 41, page 46, 2005.

[2] Pradyumna Dash. *Getting started with oracle vm virtualbox*. Packt Publishing Ltd, 2013.

[3] Peng Li. Selecting and using virtualization solutions: our experiences with vmware and virtualbox. *Journal of Computing Sciences in Colleges*, 25(3):11–17, 2010.

[4] Dale L Lunsford. Virtualization technologies in information systems education. *Journal of Information Systems Education*, 20(3):339, 2009.