Design of the Next-Generation SecureDrop Workstation

Freedom of the Press Foundation

I. Introduction

Whistleblowers expose wrongdoing, illegality, abuse, misconduct, waste, and/or threats to public health or safety. Whistleblowing has been critical for some of the most important stories in the history of investigative journalism, e.g. the Pentagon Papers, the Panama Papers, and the Snowden disclosures. From the Government Accountability Project's Whistleblower Guide (1):

> The power of whistleblowers to hold institutions and leaders accountable very often depends on the critical work of journalists, who verify whistleblowers' disclosures and then bring them to the public. The partnership between whistleblowers and journalists is essential to a functioning democracy.

In the United States, shield laws and reporter's privilege exists to protect the right of a journalist to not reveal the identity of a source. However, under both the Obama and Trump administrations, governments have attempted to identify journalistic sources via court orders to third parties holding journalist's records. Under the Obama administration, the Associated Press had its telephone records acquired in order to identify a source (2). Under the Trump Administration, New York Times journalist Ali Watkins had her phone and email records acquired by court order (3). If source—journalist communications are mediated by third parties that can be subject to subpoena, source identities can be revealed without a journalist being aware due to a gag that is often associated with such court orders.

Sources can face a range of reprisals. These could be personal reprisals such as reputational or relationship damage, or for employees that reveal wrongdoing, loss of employment and career opportunities. In severe cases, criminal charges or even violence may result from a source's disclosure. For example, in one case in 2015, former CIA employee Jeffrey Sterling was sentenced to 42 months in federal prison after he was convicted of espionage for providing documents to journalist James Risen (4). He was identified and convicted based on circumstantial evidence: phone and email metadata (5; 6). In another example, potential sources participating in undercover investigations of factory farm facilities would also be subject to severe reprisals, as such investigations are criminalized in many states via so-called "Ag-Gag" laws (7). Source protections are also needed for non-state adversaries. For example, whistleblowers Erika Cheung and Tyler Schultz were surveilled and threatened when sharing information about fraud from biotechnology firm Theranos (8).

Modern journalists must also face the growing threat of electronic surveillance and cyberattacks: compromise of the news organization website or social media accounts (9; 10; 11), phishing and malware targeted at individual journalists (12; 13), or the monitoring of source—journalist communications. The threat of phishing and malware is particularly challenging for journalists as the nature of their work involves opening files from potentially unknown sources.

The SecureDrop system (described more fully in §III) was developed in 2012 by Aaron Swartz, James Dolan, and Kevin Poulsen to enable news organizations to safely contact and interact with journalistic sources. In the years since, Freedom of the Press Foundation took over stewardship of

the project. Dozens of news organizations now use SecureDrop, including The New York Times, The Intercept, and The Washington Post. SecureDrop provides a standard secure submission interface for news organizations, which is infeasible for small or medium news organizations to develop and maintain on their own. The journalist workflow of the original architecture includes multiple physical devices that bridge an airgap. Inside the airgap, a viewing station for journalists contains a private key used to decrypt submissions. In five years of production use, we have identified ways to reduce SecureDrop's operational complexity for journalists and administrators while increasing its compartmentalization.

In this paper, we describe the motivation and design for a next-generation SecureDrop workstation. Instead of using multiple physical machines to compartmentalize operations, the new workstation separates them using software-based compartmentalization on commodity hardware. This is done through the use of QubesOS: a single-user desktop distribution of the Xen hypervisor, where each application runs in its own Virtual Machine (VM).

In §II we describe the high-level goals of the overall SecureDrop system. In §II-D we describe in more detail the threats we consider. We provide an overview of the entire SecureDrop system in §III, with a description of the challenges in the journalist workflow in §III-B. We next describe the technical goals of the Next-Generation SecureDrop Workstation for journalists in §IV and discuss some related work in §V. An overview of QubesOS is described in §VI and a detailed description of our architecture in §VII. We enumerate the most important countermeasures applied to the SecureDrop Workstation in §VIII. We conclude with a discussion of potential future improvements in §IX and a summary in §X.

II. Goals

The security goals of the overall SecureDrop system are in two broad categories: to protect journalists and their sources.

A. Protecting sources

Goal II.1. *Prevent identification of journalistic sources.* Source identities should become public if and when a sources decides to come forward. SecureDrop aims to be one solution to the "first contact" problem: how a source and journalist can meet without their first meeting producing an incriminating metadata trail.

Goal II.2. *Preserve confidentiality of source materials.* We also want to preserve the confidentiality of source materials until the time of publication if source documents are to be published. If source materials are available prematurely, sources may be at risk if they have not removed themselves from potentially dangerous situations.

B. Protecting journalists

Goal II.3. *Prevent journalists from being compromised via malicious submissions.* Given the threats to journalists described in §I, journalist devices and materials must be protected from compromise in order to ensure that they can continue to do their reporting work without putting their other sources at risk. We also want to protect the corporate IT network, by preventing attackers from getting a foothold in a news organization's network through malware submitted through their SecureDrop.

C. Other design considerations

1) Usability: Other important design considerations include usability of the system, which is of particular importance for users in high-risk and high-stress situations such as working with journalistic sources (14). Usability of security and privacy tools have a history of lacking sufficient UX input, leading to catastrophic errors made by users in user studies (15; 16). Poor usability can also lead to users associating a given tool with reduced trustworthiness (15). Usability for journalists is an important goal since we want to prevent journalists from losing a lead or story due to finding the tools too challenging to use. Finally, usability for administrators in the hosting organization is also an important consideration: if lack of proper maintenance can leave a system in an insecure and dangerous state, the maintenance effort required should be considered a usability goal for administrators.

2) Guarantees of Security and Anonymity: No individual journalist or digital security tool can provide a complete guarantee of anonymity and security to a journalistic source. The goal of SecureDrop is to provide the safest possible method for source—journalist communications, not to render it completely impossible for sources to be identified. One important consideration for source users is that Tor does not conceal that a user is accessing the Tor network, so if Tor usage is very low in a given region, it is possible that the use of Tor would raise suspicion. Broader awareness and usage of Tor would help mitigate this.

3) Legal and Governmental Threats: As a disclaimer, these threats are jurisdiction and country-specific. While the goal of the SecureDrop Workstation is to provide a strong *technical* solution for whistleblowers and journalists to communicate, it cannot provide complete protection against the increasing surge of legal and governmental threats against newsrooms across the world (17; 18). If raid or seizure occurs, provided the servers and workstations are powered off, all source materials are encrypted on-disk. However, if a news organization can be compelled to tamper with/subvert their SecureDrop, hand over encryption keys, or otherwise provide information on sources to authorities, SecureDrop (nor any technical solution) cannot defend against this scenario.

D. Threat Model

We consider a range of adversaries: from state or corporate actors to trolls or curious hackers. Our threat model accounts for an adversary who can monitor local network traffic, or who can tamper with or drop traffic. We assume they have XKEYSCORE-like capabilities, wherein internet metadata is captured for later analysis (19). We assume an adversary can directly monitor or acquire through court order the data of any third party, including search engine data, direct messages (Facebook, Twitter, etc.), e-mail messages, cellphone call and text metadata, cell location data, as well as physical mail metadata (20; 21).

An adversary can pose as an anonymous source and submit directly to the SecureDrop system. They can submit malware, spam or other harassing material to journalists through SecureDrop. We also assume that an adversary may attempt to compromise journalist devices (this motivates the compartmentalization of the SecureDrop system from the rest of the news organization's IT infrastructure (22)) as well as attempt to compromise the SecureDrop servers or web applications directly.

III. SecureDrop Overview

In Figure 1, we show an overview of the original SecureDrop architecture. Each SecureDrop deployment consists of two servers: an application server and a monitoring server. The application

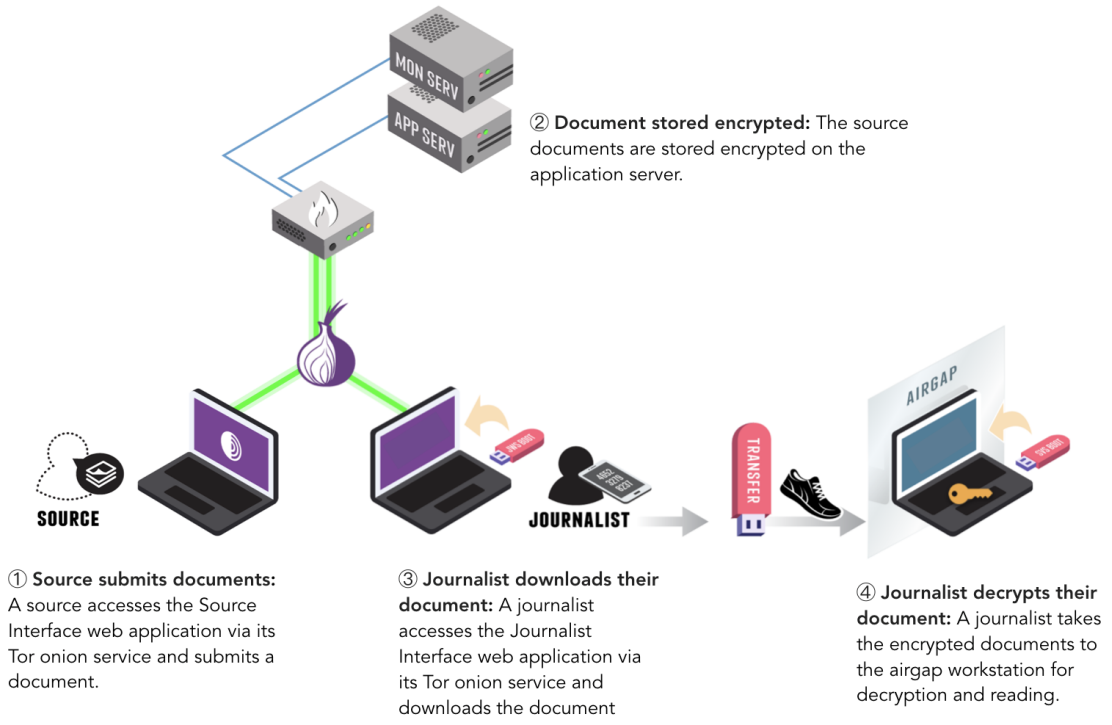

Fig. 1. Workflow of the existing SecureDrop workstation architecture. In steps 1-2 we show the source workflow as the source logs in using Tor Browser to submit documents which will be stored encrypted on the SecureDrop application server. In steps 3-4 we show the journalist workflow where a journalist downloads and decrypts documents for viewing.

server runs two Python web applications:

- Journalist Interface: A web application available over an authenticated Tor Onion Service for journalists to use to download documents.
- Source Interface: A web application available over a Tor Onion Service for sources to use to submit documents.

Source anonymity is ensured by the use of Tor onion services: all source traffic is routed through the Tor anonymity network (23). Sources are further encouraged to use the Tails Operating System (24) if possible for its amnesiac properties. The Journalist Interface is exposed only as an Authenticated Tor Onion Service primarily for defense in depth: if a web application vulnerability is found in the Journalist Interface web application, an attacker would first need to provide a secret in order to access the onion service in order to use the vulnerability.

By default administrators who SSH into these two servers for administration will use SSH interfaces only available over Tor onion services. While source and journalist HTTP traffic and administrator SSH traffic is routed through Tor through the use of onion services, most other traffic is not routed through Tor. Apt traffic to and from upstream Ubuntu package repositories and the Freedom of the Press Foundation package repository[1], SMTP traffic (for email alerts), and NTP traffic goes directly between the SecureDrop server and the external service.

The monitoring server runs a host-based Intrusion Detection System (IDS) (OSSEC): it sends encrypted email alerts to administrators within the news organization. Both these servers are

[1] https://apt.freedom.press

segmented from the rest of the news organization through the use of a network firewall running pfSense.

A. *Journalist workflow: Airgap*

In order to successfully decrypt a submission using this original architecture, a journalist must boot into two distinct Tails environments (24) on two physically separate computers: an online computer (used for downloading encrypted submissions), and an offline air-gapped computer (used for decryption and viewing of submissions). To do this, at least three USB drives are required: one for booting the online workstation, one for booting the offline workstation, and one for transferring data between the two workstations.

The workflow is as follows: they first boot their dedicated *online* workstation to Tails (on a USB drive). This drive is configured with the credentials required to access the journalist interface. They download encrypted documents and copy them to a data transfer device. The data transfer process is most commonly done using reused USB drives as described above, but in other cases is done using CDs. This data transfer device is brought into the airgap to the dedicated workstation again running Tails. The airgap workstation is the only place in the SecureDrop architecture where the private submission key is stored. Journalists decrypt and read documents there.

B. *Challenges with the Airgap*

The airgap workflow has been in production since 2013 (25), and in that time several challenges have been observed due to operational complexity and insufficient compartmentalization in the air-gap environment. More specifically:

1) **Cumbersome process**: The time-consuming process of using an air-gapped system, especially when a large volume of material is being submitted through SecureDrop, can lead to security failures where users work around the airgap for convenience. Additionally, since the process is entirely manual, it is subject to user error. This means that mistakes can have catastrophic implications, for example decrypting documents directly on the USB transfer drive which will be later connected to an online workstation, effectively breaking the airgap.
2) **Breaking the airgap**: In many deployments, the airgap is not a true airgap due to the use of USB drives to traverse the airgap. These USB drives may be reused. This is done in most cases for convenience due to the overhead of burning CDs to bring documents into the airgap which are then destroyed. An attacker that is able to get code execution in the air-gap environment can put sensitive data on the USB drive for exfiltration. This is due to the lack of isolation in the journalist workstation environment: the submission private key and source documents are stored together. The attacker knows that a journalist will take this USB drive back to a network connected device as part of normal operations.
3) **Lack of updates**: Unless a news organization is diligent in applying updates, the air-gap environment will lag behind in security updates. This is a significant issue as the air-gap environment should be the most hardened and the most protected part of the SecureDrop architecture since it contains the private key for the instance and plaintext source documents, and because it is the location where potentially malicious files are opened.

4) **Targeted malware to exfiltrate data from the airgap**: There is a rich literature on exfiltrating data from airgaps (e.g. see studies in (26; 27)). The SecureDrop team has observed malware specifically targeting the SecureDrop airgap (28).

IV. SecureDrop Qubes Workstation: Technical Goals

To accomplish the security goals described in §II, we have the following technical goals for the Qubes workstation, an alternative to the journalist workflow described in §III-A:

1) Ensure known vulnerabilities are patched.
2) Isolate the submission private key from potentially malicious submissions.
3) Isolate each source's documents. If source A is malicious, they should not be able to access source B's material.
4) Recover from an attacker getting code execution in the VM used to open submissions.
5) Provide defense in depth to protect against unknown vulnerabilities.
6) Automate high risk operations to minimize the possibility of user error in manual workflows.

In addition, we have the following ancillary goals:

1) Be maintainable by non-specialist IT staff at a news organization.
2) Be usable by non-technical staff.

V. Related Work

Fig. 2. Screenshot from Air Force SecureView presentation (29)

.

Unclassified documents indicate that the U.S. government has developed systems similar to the SecureDrop—Qubes workstation. The Air Force's SecureView project is a secure workstation where documents from different levels of classification are isolated in different VMs (29). The workstation runs on an open-source baremetal hypervisor (XenClient / OpenXT) using

commodity hardware. They developed a color-based system called GlowView, where the color of the backlit keys and an LCD screen on a keyboard indicate which security domain the user is currently working on.

Tails is a Debian-derivative Linux distribution intended for users particularly focused on protecting their anonymity. It is already in use in the SecureDrop architecture (24). It is intended for applications including working on sensitive documents, but does not have the same strong isolation provided by Qubes VMs when opening documents. However, it remains the best choice for a potential journalistic source to use when submitting to SecureDrop.

Subgraph OS is another hardened OS with similar security goals to the SecureDrop workstation (30). Its alpha release included a Tor integration and kernel hardening using the grsecurity patch set. Many applications used for opening common file types are launched in oz sandboxes, their own sandbox technology (31). Given its alpha state and the stronger protections provided by the use of VMs in Qubes, we opted to build the SecureDrop workstation on QubesOS.

VI. Introduction to QubesOS

QubesOS is a desktop-based Xen distribution (32). Xen is a type 1 (baremetal) hypervisor commonly used in cloud environments (33). It is aimed at the end user who wishes to enforce security by compartmentalization. See Figure 3 for an overview of QubesOS' key concepts. The administrative domain `dom0` is a privileged domain that runs the control stack for the rest of the VMs. In QubesOS the window manager runs in `dom0`, and along with logic used to provision the rest of the system's VMs. The desktop user defines security domains called AppVMs and runs applications or sets of applications within them. None of these unprivileged domains has direct access to the hardware. Attachments such as microphones, webcams, or USB devices must be passed through to the VM guest.

A. Template VMs

Each AppVM is based on a template, called a TemplateVM. For SecureDrop this is primarily Debian 10 (Buster). For QubesOS this is generally Fedora: `dom0` runs an older version of Fedora (Fedora 25 as of Qubes 4.0) and more recent versions of Fedora for system-provided domains. The relationship between AppVMs and their templates is as follows: when a given AppVM boots it begins with the state of the parent TemplateVM at the time of boot. That AppVM will reset to the state of the template VM after it is shutdown, except for a few directories including `/home`, `/usr/local`, and `/rw/config`, which will persist in the AppVM (34).

B. Disposable VMs

Another useful concept in Qubes is the DispVM, or disposable VM. This is a VM that is used once. Upon shutdown, it is destroyed. As with AppVMs, this DispVM is based on the state of the parent TemplateVM at the time of boot. The next time a disposable VM of a given type is booted, it begins with the state of the parent Template VM.

C. Non-networked VMs

Since each unprivileged domain does not have direct access to the underlying hardware, we can have a VM which does not have a network interface from its perspective. This is a location where we can store secrets including private keys or other sensitive data such as source documents.

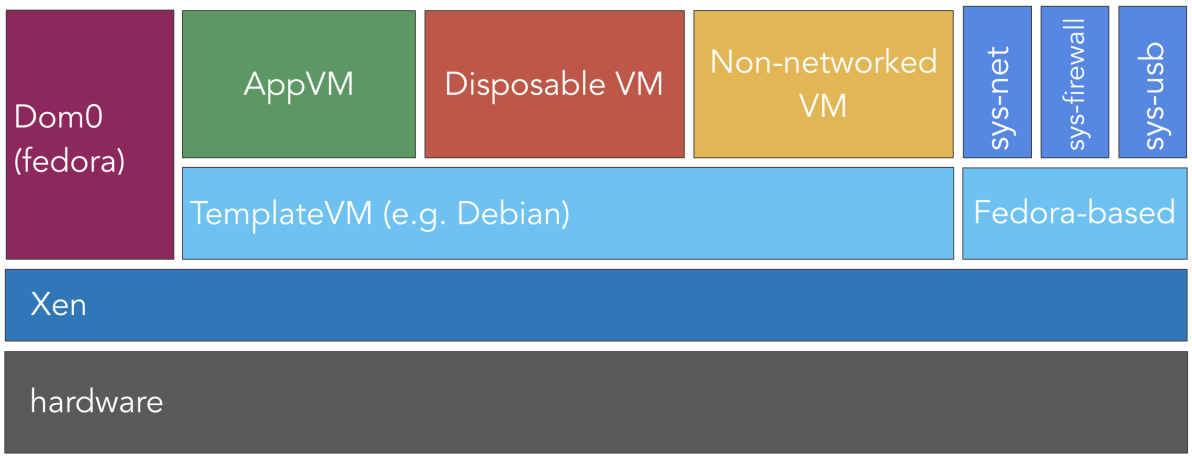

Fig. 3. Overview of QubesOS use of Xen.

D. QubesOS-provided System VMs

Qubes provides three system VMs: `sys-net`, `sys-firewall`, and `sys-usb`. The networking stack runs in `sys-net`. These utilities are isolated such that if there is a vulnerability in `dhclient` or another part of the networking stack, only the `sys-net` VM will be compromised. Firewall rules are applied in (via `iptables`) in `sys-firewall`. USB devices are assigned to `sys-usb` and can be assigned to other VMs upon demand by the user.

E. Inter-VM Communication

Inter-VM communication in Qubes is done using Qubes' `qrexec` (35), based on Xen's libvchan (see §VIII-H for more detail). This is the mechanism we use to move files and data between VMs.

VII. SecureDrop Qubes Workstation: Architecture

This section describes the design of the Qubes-based SecureDrop workstation which can be run on a single physical computer supporting Intel VT-x (for hardware assisted virtualization) and Intel VT-d (Intel's I/O MMU virtualization technology required for effective isolation of network and USB VMs) or their AMD equivalents. Instead of using physical isolation between the online and air-gap workstations, we use the isolation between security domains in Qubes to do so.

In Figure 4, we show an overview of the VM architecture of the SecureDrop Qubes workstation. We configure the workstation using Qubes' salt management stack (36). We assume that these workstations are kept in a physically secure environment to prevent physical attacks (including tampering), however full disk encryption partially mitigates the threat of search or seizure of the workstation.

Beginning with the VM directly accessing the network, we have `sys-net` and `sys-firewall`, our QubesOS-provided NetVMs. `sys-firewall` is in turn connected to a VM used for running the `tor` process, required in order to connect to the journalist interface running on the SecureDrop application server.

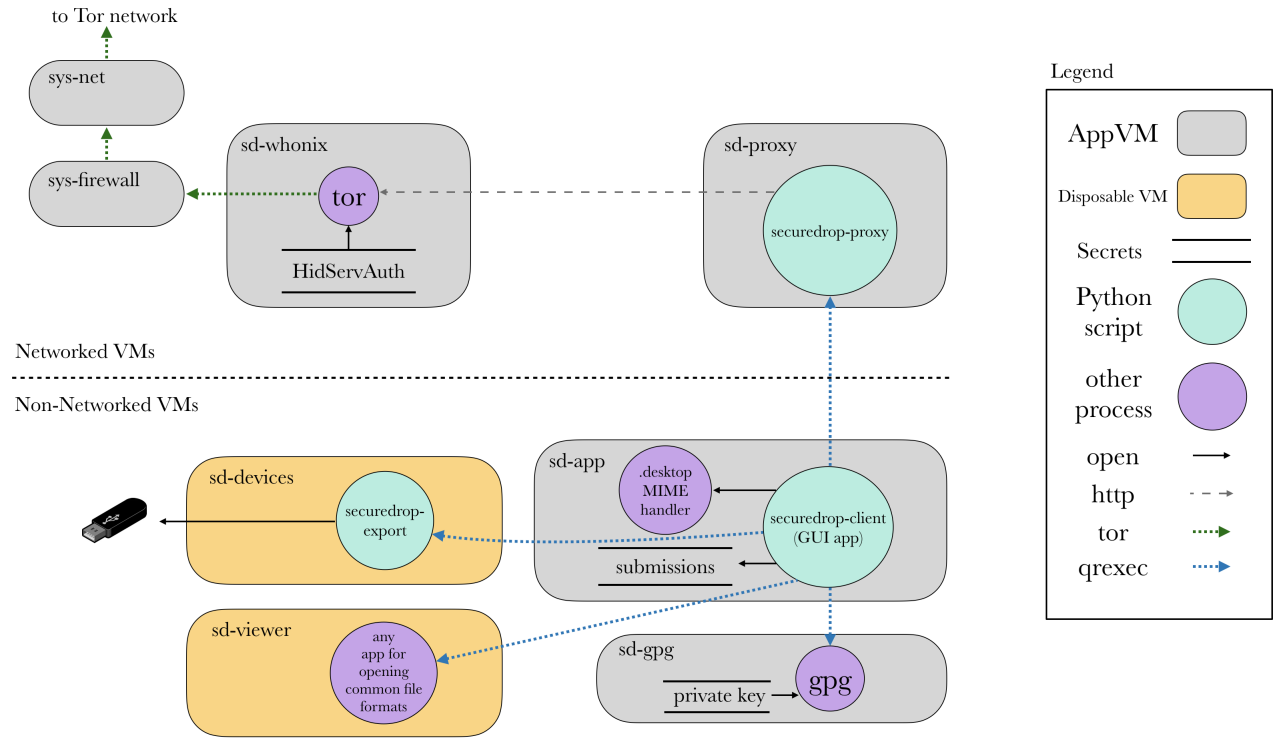

Fig. 4. Data Flow Diagram of the SecureDrop Qubes Workstation. Three Python subcomponents are shown in the figure in the teal circles: a chat-like GUI (https://github.com/freedomofpress/securedrop-client) running in the `sd-app` VM, a stateless application for passing requests/responses between the SecureDrop server and the user (https://github.com/freedomofpress/securedrop-proxy) running in `sd-proxy`, and scripts for exporting and printing documents (https://github.com/freedomofpress/securedrop-export) running in `sd-devices`. In the figure we also show the secrets stored in the workstation: the submissions in `sd-app`, the private key in `sd-gpg`, and the `HidServAuth` secret used to connect to the Journalist Interface onion service in `sd-whonix`.

The VM running `tor` is connected to the final VM that has network access, `sd-proxy`. This VM takes requests (passed via `qrexec`) from the user, constructs an HTTP request and passes that to the VM that is running `tor`. Similarly when a response is returned that HTTP response is passed back to the user via `qrexec`. The server address is hardcoded in the forwarder VM. This split-VM architecture between the `tor` process and the program that uses it was operationalized by the Whonix project, which we use in the SecureDrop workstation (37).

The journalist user interacts with a GUI application, which runs in a non-networked VM `sd-app`. When the user triggers some action that requires contacting the server, e.g. downloading a document, the application makes a `qrexec` call to `sd-proxy` VM which has network access. Since `sd-proxy` has the server address hardcoded, an attacker that gets code execution in the GUI VM can only communicate with the SecureDrop server, not arbitrary network locations, unless they are able to successfully perform a Xen escape or pivot into a network connected VM.

The private key material is stored in a separate VM (discussed further in §VIII-C). The GUI VM needs to call out to it to perform cryptographic operations. When a user wants to open a file, it gets opened in a disposable VM that is also non-networked, one per document. Since these file-opening VMs are where malicious files will be opened, they represent one of the highest-risk areas in the workstation so further hardening is applied, described in §VIII.

VIII. Countermeasures

In this section we enumerate the countermeasures that are applied to minimize risk to the system.

A. Dedicated VMs

Attack surface reduction is accomplished by isolating complex functionality, e.g. the network stack, into their own VMs. Dedicated VMs are used for all sensitive operations: cryptographic operations and the opening of malicious files.

B. DispVMs

Plaintext messages are decrypted and displayed in the GUI chat interface running in `sd-app`. All other files are decrypted, and then opened in non-networked dedicated disposable VMs, one per document. Additional hardening is performed in these disposable file opening VMs: kernel hardening described in §VIII-F, the use of AppArmor described in §VIII-G.

C. qubes-split-gpg

Private key material is isolated in the `sd-gpg` AppVM via the use of `qubes-split-gpg` (38). The user is prompted for initial access to the gpg key and notifications appear for each key access thereafter. The `qrexec` RPC policies only allow cryptographic operations from the `sd-app` AppVM. This reduces exposure of the private key.

D. GUI is in a non-networked AppVM

Since the GUI chat interface renders user-supplied input, it runs in the non-networked VM `sd-app` AppVM. If for example an HTML injection vulnerability exists in the UI, if a user clicks on a link provided by the attacker, an attacker will not be able to exfiltrate data or access resources from the internet.

E. Updates

All VMs receive security patches by updating the underlying templates. In order to start the GUI chat interface, a user must first apply all security patches, and if updates in their NetVMs or `dom0` (such as a Xen update) are required, they must apply the patches and reboot before running the GUI application or interacting with source documents.

F. Kernel Hardening

To defend against both known and unknown vulnerabilities, SecureDrop-maintained templates use the grsecurity-hardened kernel (also used on SecureDrop servers). This complicates the exploitation of memory corruption based vulnerabilities (39). Most importantly, this kernel is used in the VMs where potentially malicious files are opened. This mitigation makes it harder for an attacker to get code execution on the workstation.

G. Mandatory Access Control

In addition to kernel hardening, AppArmor is used in several VMs to restrict the capabilities of targeted programs that handle untrusted user data. AppArmor is a Linux kernel security module that provides Mandatory Access Control: it enables the definition of a profile which defines the capabilities the program is allowed. We apply upstream-maintained AppArmor profiles for some common applications used to open potentially malicious files in the file opener VM, including Evince and Libreoffice. The GUI chat application also has an AppArmor profile applied as it displays untrusted user input. The `tor` process (running in `sd-whonix`) also has an AppArmor profile defined and enforced.

H. Inter-VM Policies

QubesOS provides an RPC framework for inter-VM communication (35). QubesOS builds upon this RPC mechanism to implement primitives and tools to expose various methods of inter-VM communication to end users. We also can define our own RPC services. For example:

- `qubes.Filecopy`: Allow transfer of files from a source VM to a target VM. This is used by the `qvm-copy-to-vm` tool.
- `qubes.Gpg, qubes.GpgImportKey`: Allow basic GPG-based operations. This is used by the `qubes-gpg-client-wrapper` tool used by Qubes split-gpg functionality (38).
- `qubes.ClipboardPaste`: Allows the use of a shared clipboard across the entire system, allowing to copy-paste from one VM to another (40).
- `securedrop.Log`: Allows securedrop VMs to send log messages to a centralized logging VM. Described further in §VIII-K.
- `securedrop.Proxy`: Passes requests/responses between a non-networked AppVM and the SecureDrop server.

While these tools provides significantly improved functionality when working with multiple VMs, this increases attack surface for VMs, especially when the system is processing user-supplied and potentially malicious files. By default, all SecureDrop Workstation provisioned VMs adopt a deny-by-default policy (from an ask-by-default default Qubes policy), for both incoming and outgoing RPC calls. Only explicitly-defined sources and targets are permitted for policies that are strictly required for maintaining workstation functionality (for example, `sd-app` to `sd-proxy` allowing the client to send and retrieve data from the journalist interface using `securedrop.Proxy`, or `sd-app` to `sd-gpg` to allow the client to decrypt GPG-encrypted files or messages using `qubes.Gpg`).

I. Xen Vulnerabilities

Once an attacker has code execution in a file-opening AppVM, they can escalate privileges by (1) pivoting into a network connected VM by exploiting a misconfiguration or implementation vulnerability in the Qubes RPC management layer, or (2) exploiting a Xen breakout vulnerability.

Attackers able to perform a successful Xen breakout would have complete control over the workstation and its data therein.

Any Xen breakout relying on paravirtualization will not work in the SecureDrop Qubes Workstation due to the use of fully-virtualized VMs (see also §VIII-J). Exploitation of a Xen breakout vulnerability that applies to fully-virtualized VMs is complicated by generic exploit mitigations provided by the hardened kernel. In addition, known Xen vulnerabilities will be patched thanks to updates in §VIII-E.

J. CPU Vulnerabilities

The Meltdown and Spectre vulnerabilities in 2018 demonstrated severe issues in the speculative execution capabilities of modern processors: enabling an attacker to read privileged memory locations from userspace (41; 42). Attacks of this kind are partially mitigated due to the use of fully-virtualized VMs (also known as a Hardware-Assisted VM or HVM where the hardware is simulated using `qemu`) for SecureDrop workstation VMs: Meltdown was not exploitable in fully-virtualized VMs, instead only in paravirtualized environments (43; 41).

In 2019, further vulnerabilities in the speculative execution capabilities of modern processors were discovered (44; 45; 46). At this time, upstream QubesOS disabled hyperthreading to ensure that data cannot be leaked between VMs from a concurrent hyperthread in a Zombieload-type attack to an unprivileged attacker. In addition, the latest microcode updates are applied as part of automatic updates in order to ensure the latest mitigations have been applied to vulnerabilities of this class[2].

K. Logging

Logs from SecureDrop Workstation VMs are aggregated into a dedicated VM, `sd-log`, through a `securedrop.Log` RPC policy as defined in §VIII-H. These logs will help both debug possible failures or programming errors, as well as help in incident response.

L. Full Disk Encryption

Full disk encryption using `LUKS`/`dm-crypt` is enabled. This is enforced by default for all QubesOS systems.

M. Guards against user error

In the `sd-proxy` AppVM where files are initially downloaded before being moved to the `sd-app` AppVM, we configure the MIME type handler to a GUI window instructing the user not to open files here. In the `sd-app` AppVM, the MIME type handlers are configured to open the file in the `sd-viewer` disposable VM. These mitigations guard against a user using the file manager in individual VMs to circumvent the workstation protections, whereby potentially malicious files should only ever be opened in the `sd-viewer` VMs.

N. Secure Software Development Lifecycle

All changes are reviewed by at least one other developer. All dependency updates are reviewed and pushed to a dedicated pip mirror for use at package build time. Static code analysis (via the Bandit project (47)) is performed at review time to catch common issues. All SecureDrop workstation code is published as free and open source software. All production release tags, build artifacts, and dependency checksums are gpg-signed for integrity and authenticity. We also expanded the scope of our existing bug bounty program[3] to include the SecureDrop/Qubes workstation.

[2]via a Qubes-maintained package: https://github.com/QubesOS/qubes-intel-microcode
[3]https://bugcrowd.com/freedomofpress

O. Summary

Recapping the technical goals from §IV along with how the specific mitigations above address them:

1) Ensure known vulnerabilities are patched. This is performed via applying updates prior to running the GUI in all VMs via updating the base templates (§VIII-E).
2) Isolate the submission private key from potentially malicious submissions: the submission private key is isolated in its own VM (§VIII-C, §VIII-A).
3) Isolate each source's documents: each file is isolated in its own VM (§VIII-A).
4) Recover from an attacker getting code execution in the VM used to open submissions: each file viewing VM is destroyed after shutdown (§VIII-B).
5) Provide defense in depth to protect against unknown vulnerabilities: kernel hardening (§VIII-F) and AppArmor complicate exploitation of both known and unknown vulnerabilities (§VIII-G).
6) Automate high risk operations to minimize the possibility of user error in manual workflows: automation of the previously manual workflows is done via split-gpg (§VIII-C) and the opening of documents automatically in safe environments (§VIII-B, §VIII-M).

IX. Future Work

In this section we describe the current status and planned pilot, as well as discuss some additional mitigations that could be applied to further secure the workstation.

A. Status

A third-party audit of the alpha stage version of this project was performed in late 2018, and no-medium, high, or critical vulnerabilities were found (48). Results will be forthcoming from a four-month closed beta test with targeted news organizations.

B. Future Countermeasures

In this section we describe potential improvements for future security improvements to the workstation.

There are further countermeasures that could be applied to further reduce the threat of side-channel or covert-channel attacks on the workstation. For example, an attacker who is able to get code execution in multiple VMs can attempt to set up a covert channel to transfer data between VMs. In this scenario, one VM would be networked and one non-networked: the covert channel would serve to move the data to the networked VM for exfiltration. In order to mitigate this and other side-channel attacks, we could prevent the concurrent execution of trusted and untrusted VMs, though there is a significant user impact. Another possibility is to pin sets of VMs to certain CPU cores, such that untrusted VMs run together on the same core, and e.g. cryptographic operations occur on a dedicated core.

In the current version of the SecureDrop workstation, the laptop firmware, including the BIOS is considered trusted. The SecureDrop workstations are expected to be kept physically secure to prevent tampering. Additional work could focus on further improvements in this area.

Currently while we aggregate logs as described in §VIII-K, we are not yet performing analysis or alerting on these logs. Analysis could be performed by organization administrators, or the SecureDrop team on an opt-in basis.

C. Future Functionality

One of the major advantages of QubesOS is its ability to convert potentially malicious files to safe formats in disposable VMs. Qubes' includes a utility for converting untrusted PDFs to "trusted" PDFs (49). In their approach, PDFs are rendered in a disposable VM, RGB bitmaps of the PDF pages are returned to the calling VM, which are reconstructed into a new "trusted" PDF. This is effectively like taking a screenshot of each page of the PDF and stitching them back together. If a vulnerability in PDF parsing is exploited, it will be running in a disposable VM which will be destroyed after the RGB bitmap is returned to the caller VM. In future work this approach could be extended to more file types, such as office documents.

X. Conclusion

In this paper we presented the Next-Generation SecureDrop workstation, a QubesOS/Xen based desktop workstation for decrypting and reading malicious documents. It runs on any commodity hardware with sufficient memory that supports Intel virtualization technologies. By ensuring vulnerabilities are patched before workstation use, ensuring the isolation of key material and source documents, and providing defense in depth against the exploitation of unknown vulnerabilities, the SecureDrop workstation provides a secure environment to work with potentially malicious documents.

XI. Revision History

February 26, 2020: Initial beta announcement

XII. Acknowledgements

We thank the Mozilla Foundation through the Open Source Support program for its support of this project. Thanks to Tor, Qubes, and Tails for creating products critical to source and journalist SecureDrop users. We also thank the Open Technology Fund for funding an audit of the alpha version of this project.

References

[1] G. A. Project, *Working with Whistleblowers: A Guide for Journalists*, 2017. [Online]. Available: http://www.whistleblower.org/wp-content/uploads/2017/11/whistleblowerguidejournalism.pdf

[2] C. Savage and L. Kaufman, "Phone records of journalists seized by u.s." *The New York Times*, 2013. [Online]. Available: https://www.nytimes.com/2013/05/14/us/phone-records-of-journalists-of-the-associated-press-seized-by-us.html

[3] U. P. F. Tracker, "Doj secretly seizes phone and email records belonging to new york times reporter ali watkins," *U.S. Press Freedom Tracker*, 2018. [Online]. Available: https://pressfreedomtracker.us/all-incidents/doj-secretly-seizes-phone-and-email-records-new-york-times-reporter-ali-watkins/

[4] P. Maass, "Jeffrey sterling, convicted of leaking about botched cia program, has been released from prison," *The Intercept*, 2018. [Online]. Available: https://theintercept.com/2018/01/19/jeffrey-sterling-cia-leaking-prison/

[5] B. Trending, "Jeffrey sterling's trial by metadata: Free speech stories," *BBC*, 2015. [Online]. Available: https://www.bbc.com/news/blogs-trending-31141256

[6] R. W. Borders, "Jeffrey sterling latest victim of the us' war on whistleblowers," *RSF*, 2015. [Online]. Available: https://rsf.org/en/news/jeffrey-sterling-latest-victim-us-war-whistleblowers

[7] C. Fassett, "Unconstitutional "ag-gag" laws criminalize journalism and insulate factory farms from accountability," *Freedom of the Press Foundation*, 2018. [Online]. Available: https://freedom.press/news/unconstitutional-ag-gag-laws-criminalize-journalism-and-insulate-factory-farms-accountability/

[8] J. Carreyrou, *Bad Blood: Secrets and Lies in a Silicon Valley Startup*. RANDOM HOUSE Incorporated, 2018. [Online]. Available: https://books.google.com/books?id=43q3tgEACAAJ

[9] S. Choney, "New york times hacked, syrian electronic army suspected," *NBC News*, 2013. [Online]. Available: https://www.nbcnews.com/tech/internet/new-york-times-hacked-syrian-electronic-army-suspected-f8C11016739

[10] D. Uberti, "The looming threat of newsroom cyber attacks," *Columbia Journalism Review*, 2015. [Online]. Available: https://archives.cjr.org/behind_the_news/newsroom_cyber_attacks.php

[11] N. Perlroth, "Washington post joins list of news media hacked by the chinese," *New York Times*, 2013. [Online]. Available: https://www.nytimes.com/2013/02/02/technology/washington-posts-joins-list-of-media-hacked-by-the-chinese.html

[12] D. O'Brien, "State-sponsored attacks: open season on online journalists," *Committee To Protect Journalists*, 2012. [Online]. Available: https://cpj.org/blog/2012/06/state-sponsored-attacks-open-season-on-online-jour.php

[13] C. T. P. Journalists, "Cpj safety advisory: Journalist targets of pegasus spyware," *Committee To Protect Journalists*, 2019. [Online]. Available: https://cpj.org/2019/11/cpj-safety-advisory-journalist-targets-of-pegasus-.php

[14] T. Matthews, K. O'Leary, A. Turner, M. Sleeper, J. P. Woelfer, M. Shelton, C. Manthorne, E. F. Churchill, and S. Consolvo, "Stories from survivors: Privacy & security practices when coping with intimate partner abuse," New York, NY, USA, 2017, pp. 2189–2201. [Online]. Available: http://dl.acm.org/citation.cfm?id=3025875

[15] K. Gallagher, S. Patil, B. Dolan-Gavitt, D. McCoy, and N. Memon, "Peeling the onion's user experience layer: Examining naturalistic use of the tor browser," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1290–1305. [Online]. Available: https://doi.org/10.1145/3243734.3243803

[16] G. Norcie, K. Caine, and L. J. Camp, "Eliminating stop-points in the installation and use of anonymity systems: a usability evaluation of the tor browser bundle," in *5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETS)*. Citeseer, 2012.

[17] B. News, "Abc raid: Australian public broadcaster loses legal challenge," *BBC News*, 2020. [Online]. Available: https://www.bbc.com/news/world-australia-51526607

[18] R. sans frontières, "Classement mondial de la liberté de la presse 2019," accessed: 2020-02-19. [Online]. Available: https://rsf.org/fr/ranking/2019

[19] G. Greenwald, "Xkeyscore: Nsa tool collects 'nearly everything a user does on the internet'," *The Guardian*, 2013. [Online]. Available: https://www.theguardian.com/world/2013/jul/31/nsa-top-secret-program-online-data

[20] R. Nixon, "Us postal service logging all mail for law enforcement," *The New York Times*, 2013. [Online]. Available: https://www.nytimes.com/2013/07/04/us/

monitoring-of-snail-mail.html
[21] J. Valentino-DeVries, "Tracking phones, google is a dragnet for the police," *The New York Times*, 2019. [Online]. Available: https://www.nytimes.com/interactive/2019/04/13/us/google-location-tracking-police.html
[22] S. Biddle and M. Cole, "Team of american hackers and emirati spies discussed attacking the intercept," *The Intercept*, 2019. [Online]. Available: https://theintercept.com/2019/06/12/darkmatter-uae-hack-intercept/
[23] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Naval Research Lab Washington DC, Tech. Rep., 2004.
[24] Tails, "Tails operating system," https://tails.boum.org/.
[25] A. D. Sorkin, "Introducing strongbox," *The New Yorker*, 2013. [Online]. Available: https://www.newyorker.com/news/amy-davidson/introducing-strongbox
[26] M. Guri, Y. Solewicz, A. Daidakulov, and Y. Elovici, "Fansmitter: Acoustic data exfiltration from (speakerless) air-gapped computers," *arXiv preprint arXiv:1606.05915*, 2016.
[27] M. Guri, Y. Solewicz, and Y. Elovici, "Mosquito: Covert ultrasonic transmissions between two air-gapped computers using speaker-to-speaker communication," in *2018 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE, 2018, pp. 1–8.
[28] SecureDrop, "Security advisory: Do not scan qr codes submitted through securedrop with connected devices," https://securedrop.org/news/security-advisory-do-not-scan-qr-codes-submitted-through-securedrop-connected-devices/.
[29] A. F. R. Laboratory, "Secureview overview." [Online]. Available: https://www.fbcinc.com/e/cdse/presentations/day2/10-SecureView_Overview_-_Capt_Alex_Gwin_-_FINAL_-_Day_2.pdf
[30] Subgraph, "Subgraph os: Adversary resistant computing platform," https://subgraph.com/.
[31] ——, "Oz technical details," https://github.com/subgraph/oz/wiki/Oz-Technical-Details.
[32] J. Rutkowska and R. Wojtczuk, *QubesOS Architecture*, 2010. [Online]. Available: https://www.qubes-os.org/attachment/wiki/QubesArchitecture/arch-spec-0.3.pdf
[33] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, p. 164–177, Oct. 2003. [Online]. Available: https://doi.org/10.1145/1165389.945462
[34] QubesOS, *QubesOS Template Implementation*, https://www.qubes-os.org/doc/template-implementation/.
[35] ——, *Qrexec: secure communication across domains*, https://www.qubes-os.org/doc/qrexec/.
[36] ——, "Management infrastructure," https://www.qubes-os.org/doc/salt/.
[37] Whonix, *Multiple Whonix-Workstation*, 2019. [Online]. Available: https://www.whonix.org/wiki/Multiple_Whonix-Workstation
[38] QubesOS, *Qubes Split GPG*, https://www.qubes-os.org/doc/split-gpg/.
[39] Grsecurity, *Grsecurity*, 2020. [Online]. Available: https://grsecurity.net
[40] QubesOS, *Qubes copy/paste*, https://www.qubes-os.org/doc/copy-paste/.
[41] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin *et al.*, "Meltdown: Reading kernel memory from user space," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 973–990.
[42] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher *et al.*, "Spectre attacks: Exploiting speculative execution," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 1–19.

[43] QubesOS, "Qubes security bulletin 37: Information leaks due to processor speculative execution bugs," 2018. [Online]. Available: https://www.qubes-os.org/news/2018/01/11/qsb-37/
[44] M. Schwarz, M. Lipp, D. Moghimi, J. Van Bulck, J. Stecklina, T. Prescher, and D. Gruss, "ZombieLoad: Cross-privilege-boundary data sampling," in *CCS*, 2019.
[45] S. van Schaik, A. Milburn, S. Österlund, P. Frigo, G. Maisuradze, K. Razavi, H. Bos, and C. Giuffrida, "RIDL: Rogue in-flight data load," in *S&P*, May 2019.
[46] C. Canella, D. Genkin, L. Giner, D. Gruss, M. Lipp, M. Minkin, D. Moghimi, F. Piessens, M. Schwarz, B. Sunar, J. Van Bulck, and Y. Yarom, "Fallout: Leaking data on meltdown-resistant cpus," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019.
[47] Bandit, "Bandit," https://github.com/PyCQA/bandit.
[48] SecureDrop, "Third party audit of integrated securedrop workstation completed," https://securedrop.org/news/third-party-audit-integrated-securedrop-workstation-completed/.
[49] QubesOS, "Github: qubes-app-linux-pdf-converter," https://github.com/QubesOS/qubes-app-linux-pdf-converter.