Qubes OS: A virtualization-based approach to protect against common operating systems security risks

Radu V. Popa, Victor A. Truică
Computer Science Department, University of Bucharest, Romania

Abstract

The desktop operating system (OS) is the most important component of a machine, since it's responsible for accessing the hardware resources (CPU, memory, etc.) and running all of the applications installed. It is also what the users interact with the most in their daily activities. The design philosophy behind recent operating systems prioritizes ease of access and convenience instead of focusing on security and bug fixing, resulting in many documented unpatched vulnerabilities. These issues can severely compromise the most popular OSs and can put user data at risk. In this paper we will examine if the virtualization-based operating system Qubes OS is a valid and practical alternative for providing security against common vulnerabilities and exploits enabled in most major operating systems today. We will present virtualization types, the architecture and the main components behind Qubes OS, the compartmentalization of applications and its benefits. Finally, we will present our findings with regards to common OS attacks and to what extent it is safe to use Qubes OS as an operating system performing common tasks.

1 Introduction

Currently, no operating system of modern complexity could be considered completely secure. Security in operating systems represents a growing concern nowadays because of how processes are handled in modern implementations, namely how the processes interact and share data, often times the output for a process could be used by many other processes. This results in an increased complexity of process interacting and data sharing. One of the biggest challenges in developing a secure operating system is designing security mechanisms for protecting the execution of processes and the output data in such a complex and interactive environment. Usually such a solution comes with a heavy performance loss and with very limited functionality. [1] Existing desktop operating systems (OS) are not made to provide the user with a secure environment but instead they provide ease of access and many features, however they can easily get compromised. [2]

Virtualization is useful in providing an additional security layer against most security issues in an OS, because it enables the creation of isolated containers, also known as Virtual Machines (VMs), which gives the user the ability to host multiple specialized guest OSs on a single machine, and contain potential breaches to a single virtual environment. [3]

Another use case of virtualization would be to leverage its isolation features to run each application in its own environment, all under one hypervisor acting as the base OS. An example would be Qubes OS, which is based on the Xen hypervisor, and intends to ensure the security of the user through isolation of installed applications, while also being compatible with most of the software available, such as applications or drivers designed for popular OSs (Windows and Linux distributions). [2]

2 Qubes OS Architecture

There are multiple ways to achieve virtualization of an entire operating system. In section, we present a short overview of the main current available types of virtualization methods. [4]

Operating System-level virtualization is a method used for virtualization where the host OS is a modified kernel that can share said kernel between

multiple isolated containers or virtualized environments that can be executed independently. *Paravirtualization* is a method based on introducing a new set of instructions also known as Hypercalls that replace the existing instruction set found in the physical hardware architecture. The hypervisor runs on top of the hardware level, which is the highest privilege level of code execution (ring 0). *Binary translation* is another virtualization method based on emulating a processor architecture on top of a different processor architecture. This permits a hypervisor to run a guest OS on top of an existing host OS. *Hardware assisted virtualization* is achieved using AMD and Intel specific virtualization extensions. This method provides another level of privilege above the ring 0 where the hypervisor can operate (ring -1 or root mode), so the guest OSs are able to run with ring 0 level of privilege. [4]

The Hypervisor is the software that is responsible for the virtualization of all available resources on a machine, and allocating said resources for the creation and execution of different VMs. There are two types of hypervisors. A type I hypervisor has a privilege level of just above the ring 0 and controls all of the available VMs. It is more complex and has memory, CPU and I/O resource management components. A type II hypervisor runs in the same privilege level ring as the host OS installed on the machine, and can control VMs through the use of the host OS processes. The Xen hypervisor falls in the type I category and can support both paravirtualization and hardware assisted virtualization. [4]

Qubes OS is based on the Xen hypervisor and is designed to ensure integration between different applications hosted in different VMs on a single machine. Besides the X Window Manager, every other user application and system application is hosted in a separate AppVM. [2]

AppVMs are individual virtual machines that the operating system creates to host Linux or Windows based applications. To help reduce the disk space used for the AppVMs, all of the applications that use the same system distribution (such as Linux) use the same file system. Since this could still be a big issue in compromising user data, the disk and file system storage code are isolated in a different VM, named the storage domain. [2]

The *network domain* is represented by an unprivileged VM which contains the drivers and the protocol stacks required to make networking possible. This domain only has access to the WiFi and Ethernet cards, so that in case of a potential exploit, the attacker would only obtain access over this domain. [2]

The *storage domain* contains the file system sharing mechanisms, the disk drivers and the stack. This domain is made of a dedicated unprivileged VM, that can only access the disk controller, as well as USB and CD/DVD controllers. [2]

The *graphical user interface (GUI) domain*, also known as Dom0 is the component that the user interacts with the most, it has direct access to the input and output devices. This domain is made of the X Windows System server and the Window Manager that allows the user to display the desktop and to control all of the installed applications. The implementation in Qubes OS is a very simple interface with no sensitive code running (networking, etc.), and is used to control other domains, to minimize the security implications of an attack inside an AppVM. [2]

The *Xen hypervisor* is responsible for providing the containment of all the different virtual machines, which makes it the most important component from a security point of view, since an exploit here could compromise the entire system. Xen can support both para-virtualized as well as fully virtualized machines. [2]

The *I/O emulation software* is very complex and therefore it is always safe to assume it can be exploited. The Xen hypervisor assumes this and is able to contain the I/O emulator the same way it does for a regular VM and it can protect the rest of the system in case it gets compromised. [2]

Drivers are also containerized, so that even if a bug is present in a bad implementation or is introduced in an update, the rest of the system is secure. This is done through the implementation of driver domains, which are limited privileges VM-like domains that can only access certain PCI devices through the use of INTEL VT-d. [2]

3 Security risks in Qubes OS

Various attacks and known vulnerabilities have been considered in a two-way format - the ease and the impact of exploiting the vulnerability. This is considered in two impact contexts, one consider-

ing the possibility of compromising an AppVM, the other considering the possibility of compromising the hypervisor, or the entire machine.

The threat models have been limited to the execution of untrusted software downloaded from the internet or implanted via external media. Only the following 4 "initial access" items from the *MITRE Attack framework* : Drive by compromise, Replication through removal media, Spearphishing attachment, Spearphishing link and Hardware additions. Threat models specifically scoped out are anything not found above, examples including hardware supply chain attacks and compromise of the OS source code. [5]

Attacks considered have been Evil maid, VM escape, ROP-based attacks, Spectre/Meltdown attacks and attacks that leverage all mentioned in a cumulated attack.

3.1 Evil maid attacks

Evil maid attacks refer to the situations in which a threat actor would have physical access to the machine for a limited time period and be able to compromise it without leaving any visible signs [6]. QubesOS relies on full disk encryption based on LUKS/DM-crypt.

Concrete attacks have been demonstrated against similar full disk encryption tools [7] .

QubesOS has the possibility of installing an anti-evil-maid module, however it introduces the trade-off between trusting a USB device and protecting against evil-maid attacks, given the fact that it requires attaching a USB device to configure it. The system is not, by default, secured against such attacks, but it can be made so. [7] [8]

3.2 VM escape

There have been concrete examples of Xen vulnerabilities (*CVE-2007-4993, CVE-2007-5497, CVE-2015-7835*) that allowed VM escaping, or being able to gain access to the hypervisor via the VM. Given that Qubes OS is based on the Xen hypervisor, it was vulnerable to such vulnerabilities before the patch was released and it will be prone to future variations of the attack. This vulnerability is being addressed by switching the isolation of I/O devices from para-virtualization to hardware assisted virtualization starting with Xen version 4.0. [9] VM escape attacks are the only way to compromise the entire system and not just the AppVM and for this to happen, the attack must be chained with an exploit on the AppVM. [9]

3.3 Return-Oriented Programming (ROP) exploits

In the case of QubesOS, ROP based exploits are as effective against either the AppVMs or the hypervisors as against any other modern Linux OS. This happens even though standard protections like ASLR are in place. The basic necessities for ROP attacks to be successful, such as knowledge of the code composition (location of instruction) and the operating system environment (API, system call table), will be in place. [10] However, the amount of potential machine instruction sequences that can be found and used reliably for a ROP-based attack (gadgets) is significantly reduced, thus reducing the likelihood of a successful attack, even though all the premises for it are in place. [11] The impact can be considered limited and the advantage comes in the compartmentalization and limited-timespan nature of the VMs.

3.4 Meltdown/Spectre attacks

Meltdown and Spectre exploit critical vulnerabilities in modern processors. In the case of Meltdown patches have been applied at the VM OS level as well as the hypervisor. We ran various Spectre exploits found online on a machine running QubesOS with results indicating that exploitation is clearly possible. [12]

The following output shows how a string variable ("The") that was kept in memory has been read using a Spectre exploit demonstrating the aplicability of the vulnerability on the OS.

```
[user@personal SpectrePoC-master]$
    ./spectre.out
Using a cache hit threshold of 80.
Build: RDTSCP_SUPPORTED MFENCE_SUPPORTED
    CLFLUSH_SUPPORTED
    INTEL_MITIGATION_DISABLED
    LINUX_KERNEL_MITIGATION_DISABLED
Reading 40 bytes:
Reading at malicious_x =
    0xffffffffffffdfc0... Unclear: 0x54=T
    score=854 (second best: 0x00=?  score=779)
```

```
Reading at malicious_x =
    0xffffffffffffdfc1... Unclear: 0x68=h
    score=999 (second best: 0x00=?  score=909)
Reading at malicious_x =
    0xffffffffffffdfc2... Unclear: 0x65=e
    score=999 (second best: 0x01=?  score=903)
```

Due to its microkernel design, having a minuscule memory footprint and having a limited interface with the guest, a successful Meltdown/Spectre attack can only extend itself to the memory bounds of the guest VM, thus being unable to reach within the hypervisor or another guest VM. [13]

4 Conclusion

Qubes OS is a security focused operating system based on virtualization. Its strength lies in the way the system has been architected and not in the singular strength of its components. The likelihood of a full system compromise is much smaller than in traditional OSs and the difficulty to perform it requires a very advanced threat actor that can successfully chain two or more complex exploits in a limited timespan. The impact of a VM compromise is negligible due to the idea their purpose is disposable and time limited, although this is very subjective depending on what the end user is processing inside the disposable VM.

No OS should be considered 100% safe, but Qubes OS is a reasonably secure solution given the premise of an advanced user operating it, knowing which decision to take when it comes to interfacing the device or the time allocated to maintaining a particular VM in operation.

References

[1] T. Jaeger, "Operating system security," *Synthesis Lectures on Information Security, Privacy and Trust*, vol. 1, no. 1, pp. 1–218, 2008.

[2] J. Rutkowska and R. Wojtczuk, "Qubes os architecture," *Invisible Things Lab Tech Rep*, vol. 54, p. 65, 2010.

[3] J. Rutkowska, "Software compartmentalization vs. physical separation," *Invisible Things Lab*, 2014.

[4] F. Rodríguez-Haro, F. Freitag, L. Navarro, E. Hernánchez-sánchez, N. Farías-Mendoza, J. A. Guerrero-Ibáñez, and A. González-Potes, "A summary of virtualization techniques," *Procedia Technology*, vol. 3, pp. 267–272, 2012.

[5] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "Mitre att&ck: Design and philosophy," *Technical report*, 2018.

[6] O. B. Makarevich, A. Elçi, M. A. Orgun, and S. A. Huss, "Sin'10-proceedings of the 3rd international conference of security of information and networks: Foreword," in *SIN'10-Proceedings of the 3rd International Conference of Security of Information and Networks*, pp. iii–v, 2010.

[7] J. Rutkowska and A. Tereshkin, "Evil maid goes after truecrypt," *The Invisible Things Lab*, 2009.

[8] J. Rutkowska, "Anti evil maid," *The Invisible Things Lab (September 2011)*, 2011.

[9] "Xen Exploitation 4." `https://blog.quarkslab.com/xen-exploitation-part-3-xsa-182-qubes-escape.html`. Accessed: 2021-06-20.

[10] V. Pappas, "Defending against return-oriented programming. 2015," 2018.

[11] E. Buchanan, R. Roemer, H. Shacham, and S. Savage, "When good instructions go bad: Generalizing return-oriented programming to risc," in *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 27–38, 2008.

[12] "SpectrePOC." `https://github.com/crozone/SpectrePoC`. Accessed: 2021-06-23.

[13] "Xen Project Software Overview." `https://wiki.xenproject.org/wiki/Xen_Project_Software_Overview`. Accessed: 2021-06-20.