Ostbayerische Technische Hochschule Amberg-Weiden

Fakultät Elektrotechnik, Medien und Informatik | Medieninformatik

Evaluation von Qubes OS für den Einsatz in klein- und mittelständischen Software-Entwicklungsunternehmen

Bachelorarbeit

von

Tobias Schotter

Ostbayerische Technische Hochschule Amberg-Weiden

Fakultät Elektrotechnik, Medien und Informatik | Medieninformatik

Evaluation von Qubes OS für den Einsatz in klein- und mittelständischen Software-Entwicklungsunternehmen

Bachelorarbeit

von

Tobias Schotter

<u>Bearbeitungszeitraum</u>: von 20.September 2021

bis 18.Februar 2022

1. Prüfer: Prof. Dr.-Ing. Christoph P. Neumann
2. Prüfer: Prof. Dr. rer. nat. Daniel Loebenberger

Bachelorarbeit von Tobias Schotter

Ostbayerische Technische Hochschule Amberg-Weiden

Fakultät Elektrotechnik, Medien und Informatik

Bestätigung gemäß § 12 APO

Name und Vorname
der Studentin/des Studenten: **Schotter, Tobias**

Studiengang: **Medieninformatik**

Ich bestätige, dass ich die Bachelorarbeit mit dem Titel:

Evaluation von Qubes OS für den Einsatz in klein- und mittelständischen Software-Entwicklungsunternehmen

selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Datum: 18.02.2022

Unterschrift:

Ostbayerische Technische Hochschule Amberg-Weiden

Fakultät Elektrotechnik, Medien und Informatik

Bachelorarbeit Zusammenfassung

Studentin/Student	Schotter, Tobias
Studiengang:	Medieninformatik
Aufgabensteller, Professor:	Prof. Dr.-Ing. Christoph P. Neumann
Durchgeführt in:	OTH Amberg-Weiden
Ausgabedatum: 20.09.2021	Abgabedatum: 18.02.2022

Titel:

Evaluation von Qubes OS für den Einsatz in klein- und mittelständischen Software-Entwicklungsunternehmen

Zusammenfassung

Aus einer zunehmenden Anzahl an Cyberangriffen bleibt kaum noch ein Unternehmen in Deutschland verschont. Durch Cybercrime-Delikte entstehen allein in Deutschland inzwischen Schäden von etwa 220 Milliarden Euro pro Jahr. Das Betriebssystem Qubes OS bietet einen Ansatz Computer durch Isolation von Prozessen, Anwendungen und Hardware, mittels Kompartimentierung, beispielweise in Form von virtuellen Maschinen, stärker zu schützen als herkömmliche IT-Sicherheitsmaßnamen. Qubes verfolgt den praktischen Ansatz Schaden zu begrenzen und einzudämmen, um so wertvolle Daten von risikoreichen Aktivitäten zu trennen.

Im Rahmen der vorliegenden Arbeit werden Konzepte evaluiert, auf welche Weise kleine- und mittelständischen Software-Entwicklungsunternehmen Qubes OS in ihren alltäglichen Gebrauch einbinden können. Hierzu werden typische Nutzungsszenarien von Software-Ingenieuren untersucht und mit einem nativen Windows 10 Betriebssystem gegenübergestellt. Zusätzlich sollen Einsatzszenarien von gängigen Büroanwendungen als auch Software-Entwicklungsprozessen mit typischen Entwicklungs- und Testwerkzeugen einer Tauglichkeitsprüfung unterzogen werden.

Auf Basis der Evaluierungen werden die nötigen Veränderungen im alltäglichen Lebenszyklus eines Softwareentwicklers dargelegt und mit den bekanntlich „Best Practices“ verglichen.

Abstract

Hardly any company in Germany is spared from an increasing number of cyber-attacks. Cybercrime offences now cause damage of around 220 billion euros per year in Germany alone. The Qubes OS operating system offers an approach to protect computers further than conventional IT security measures by isolating processes, applications, and hardware through compartmentalization, for example in the form of virtual machines. Qubes follows the practical approach of limiting and containing damage to separate valuable data from risky activities.

In the context of this thesis, concepts are evaluated on how small and medium-sized software development companies can integrate Qubes OS into their everyday use. For this purpose, typical usage scenarios of software engineers are examined and compared with a native Windows 10 operating system. In this context, usage scenarios of common office applications as well as software development processes with typical development and test tools are subjected to a suitability test.

Based on the evaluations, the necessary changes in the everyday cycle of a software developer will be presented and compared with the known "best practices".

Danksagung

An dieser Stelle möchte ich mich bei den Menschen bedanken, die mich bei dieser Bachelorarbeit und auf dem Weg dorthin unterstützt haben.

Zuerst gebührt mein Dank Herrn Prof. Christoph P. Neumann für das Engagement in der Betreuung meiner Bachelorarbeit und des Weckens meines Interesses an diesem Themenbereich. Er stand mir immer mit Rat zur Seite, die Gespräche mit ihm waren stets hilfreich und zielführend.

Ein großer Dank geht auch an Herrn Prof. Daniel Loebenberger, der die Zweitbetreuung meiner Bachelorarbeit übernahm und überdies hinaus Hardware für meine Untersuchungen zur Verfügung gestellt hat.

Außerdem möchte ich Ricciardo Baldassarre für das Korrekturlesen meiner Bachelorarbeit danken.

Einen abschließenden Dank gilt meiner Familie, die mir mein Studium durch ihre Unterstützung ermöglicht haben.

Tobias Schotter

Inhaltsverzeichnis

Abkürzungsverzeichnis

COTS	Components-off-the-shelf
MB	Megabyte
OS	Operating System
Pentest	Penetrationstest
SDK	Software Development Kit
SDL2	Simple Direct Media Layer
VM	Virtuelle Maschine
WSL 2	Windows-Subsystem für Linux

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Heutige IT-Systeme sind einer Vielzahl von Angriffen ausgesetzt, das gilt für Firmencomputer und Server-Systeme genauso wie für private Computer- und Desktopsysteme. Typische Wege diesen Risiken entgegenzuwirken sind Anti-Viren-Software und Firewalls. Dennoch verschaffen sich Angreifer regelmäßig Zugriff zu Firmencomputern und verursachen sehr große Schäden. Die geschätzten Schäden durch Cyberkriminalität liegen in Deutschland bei circa 220 Milliarden Euro pro Jahr. [1]

Um sich vor Cyberangriffen zu schützen, können Computer durch Isolation von Prozessen, Anwendungen und Hardware, mittels Kompartimentierung, geschützt werden. Ein Beispiel hierfür könnten Sandboxes[1] oder Container sein, welche in diesem Kontext auch Mikro-VMs oder AppVMs genannt werden. Die Kompartimentierung erfolgt mittlerweile eingebettet auf Ebene des Betriebssystems oder sie wird als Plattform-Schicht für die Anwendungen bereitgestellt.

Mit Qubes OS steht ein Open-Source-Betriebssystem zur Verfügung, welches Anwendungen des Benutzers in separaten und teilweise temporären virtuellen Maschinen kapselt und ausführt. Dadurch ist es stark erschwert, das ganze System durch Ausnutzung einer einzigen Schwachstelle unter die Kontrolle eines Angreifers zu bringen. Vor allem Phishing, Spear-Phishing und Malware-Angriffe werden stark erschwert, da Web-Links bzw. Internetseiten grundsätzlich in *DisposibleVMs* (Virtuelle Maschinen, welche sich beim Schließen selbst zerstören) geöffnet werden können. Falls es sich um einen bösartigen Link handelt, hat die Malware nur Zugriff auf die temporäre Virtuelle Maschine, welche keine Rechte besitzt auf persönliche Daten, Passwörter oder Hardwarekomponenten zuzugreifen. Beim Schließen des entsprechenden Webbrowsers wird die DisposibleVM inklusive jeglicher Schadensoftware, die mitinstalliert wurde, zerstört. Qubes OS bietet besonders nutzerfreundliche Möglichkeiten virtuelle Maschinen hochzufahren und Anwendungen auf separaten virtuellen Maschinen auszuführen.

Bei der Entwicklung von Qubes wurde von der Annahme ausgegangen, dass jede Software Bugs enthält. Darüber hinaus beeilen sich Softwareentwickler, Fristen einzuhalten und mit alarmierender Geschwindigkeit neuen Code zu generieren – viel schneller, als ein relativ kleines Team von Sicherheitsexperten jemals hoffen könnte, Schwachstellen zu analysieren, geschweige denn alles zu beheben. Anstatt zu verhindern, dass diese unvermeidlichen

[1] Eine isolierte Umgebung für eine geschützte Ausführung von Software.

Schwachstellen ausgenutzt werden, wurde Qubes unter der Annahme entwickelt, dass sie ausgenutzt werden.

In Anbetracht dieser ernüchternden Realität verfolgt Qubes einen äußerst praktischen Ansatz: Begrenzung, Kontrolle und Eindämmung des Schadens. Es ermöglicht, wertvolle Daten von risikoreichen Aktivitäten zu trennen und eine Kreuzkontamination zu verhindern. Das bedeutet, dass alles auf demselben physischen Computer erledigt werden kann, ohne sich Sorgen machen zu müssen, dass ein einziger erfolgreicher Cyberangriff das System zu Grunde bringt.

Qubes bietet praktische, brauchbare Sicherheit für gefährdete und aktiv angegriffene Personen wie Journalisten, Aktivisten, Whistleblowers und Forscher. Qubes wurde in dem Bewusstsein entwickelt, dass Menschen Fehler machen, und es ermöglicht Ihnen, sich vor Ihren eigenen Fehlern zu schützen. Es ist ein Ort, an dem Benutzer unbesorgt auf Links klicken, Anhänge öffnen, Geräte anschließen und Software installieren können. Es ist ein Ort, an dem Benutzer die Kontrolle über die Software haben, nicht umgekehrt.

Allerdings ist Qubes OS kein Mehrbenutzersystem, da Qubes auf Xen[2] basiert und eine sichere Isolierung zwischen unprivilegierten Benutzern fast unmöglich ist. Xen nicht dafür entwickelt wurde, eine sichere Kontrolle des „Xen Daemon Management Interface“ durch einen nicht privilegierten Benutzer zu ermöglichen. Qubes OS verfolgt den Ansatz den User vor verschiedenen Gefahren zu schützen, nicht das System vor dem User zu schützen.

1.1 Zielsetzung der Arbeit

Die vorliegende Arbeit hat zum Ziel, einen Erfahrungsbericht aus einer vergleichbaren Perspektive darzulegen. Typische Nutzungsszenarien in klein- und mittelständischen Software-Entwicklungsunternehmen sollen identifiziert werden und der Einsatz von Qubes als Desktop-OS für Software-Ingenieure wird untersucht und bewertet. In der Bewertung wird eine vergleichende Gegenüberstellung mit einem nativen Windows 10 mit WSL 2 und Docker Desktop für das Einsatzszenario erfolgen. Dafür ist es unvermeidlich erforderlich die grundlegenden Sicherheitsmechanismen von Qubes OS in eine Übersicht zu bringen.

Für Software-Entwicklungsunternehmen soll das Einsatzszenario sowohl gängige Büroanwendungen als auch Software-Entwicklungsprozesse mit gängigen Entwicklungs- und Testwerkzeugen umfassen. Dabei werden nicht nur die Entwicklungen von

[2] Xen, auch Xen Project, ist ein Hypervisor, also eine Software, die den Betrieb mehrerer virtueller Maschinen auf einem physischen Computer erlaubt.

kommandozeilenbasierten Server-Backends, sondern auch die Entwicklung von Anwendungen mit Web-, 2D- und 3D-fähigen Oberflächen berücksichtigt. Als Simulation werden git-Repositories gängiger SW-Projekte herangezogen, da eine Container-basierte Entwicklungsweise für Cloud-basierte Dienste den Stand der Technik bildet, und die gegenseitige Verträglichkeit verschiedener Virtualisierungstechniken nicht selbstverständlich ist, wird auch der Betrieb von Container-Werkzeugen wie Docker unter Qubes OS einer Tauglichkeitsprüfung unterzogen, um Stolperfallen identifizieren. Gemäß der vorgegebenen Aufgabenstellung soll untersucht werden, wie zentrale IT-Administrationen auf Rechnern mit Qubes OS automatisch Anwendungen verteilen und konfigurieren können. Des Weiteren soll evaluiert werden, wie Power-User selbständig Anwendungen unter Qubes OS installieren und privilegiert ausführen können. Es soll aufgezeigt werden, welche Grenzen Qubes OS einem Software-Ingenieur im Vergleich zu nativen Windows-Umgebungen setzt. Also welche veränderten Vorgehensweisen im Lebenszyklus der Softwareentwicklung werden durch Qubes OS erforderlich, oder kann auf bekanntlich „Best Practices“ zurückgegriffen werden.

2 Methodik

In diesem Kapitel wird die Vorgehensweise erläutert, nach der die in der vorliegenden Arbeit dargelegten Ergebnisse gewonnen werden. Die Beschreibungen der durchgeführten Entwurfsschritte werden den jeweiligen Kapiteln der Ausarbeitung zugeordnet, um dem Leser einen Überblick über den Aufbau dieser Arbeit zu geben.

In Kapitel 3 erfolgt eine Gegenüberstellung in der herkömmliche Sicherheitsmechanismen mit den allgemeinen Mechanismen von Qubes OS in eine Übersicht gebracht werden. Diese Ergebnisse sollen eine Grundlage bilden und die mögliche Relevanz von Qubes OS in kleinen- und mittelständischen Software-Entwicklungsunternehmen aufzeigen. Ebenso wird ein kurzer Einblick in Datenschutz mit Hilfe von Tor und Whonix gegeben.

In dem darauffolgende Kapitel 4 wird ein Erfahrungsbericht aus einer vergleichenden Perspektive erstellt. Typische Nutzungsszenarien in klein- und mittelständische Software-Entwicklungsunternehmen sollen identifiziert werden. Zu Beginn des Kapitels wird auf administrative Nutzungsszenarien eingegangen. Hierrunter fällt die Installation des Betriebssystems selbst, Möglichkeiten für die Installation von Software, inklusive der Untersuchung, ob Anwendungen automatischen Verteilung und konfiguriert werden können. Abschließend wird gezeigt, wie ein eigene Windows-Qube eingerichtet werden kann. Im Anschluss werden alltägliche Szenarien evaluiert. Hierunter zählt die Untersuchung von Büroanwendungen, sowie einer Erläuterung der Dateiverwaltung unter Qubes OS. Im nächsten Abschnitt werden die Nutzungsszenarien von Entwicklungsprozessen betrachtet. Darunter zählt die Entwicklung von kommandozeilenbasierten Server-Backends (Node.js/Java/.NET Core) und Anwendungen mit Web-, 2D- und 3D-fähigen Oberflächen. Ebenso wird untersucht, ob der Betrieb von Container-Werkzeugen wie *Docker* unter Qubes OS möglich ist.

In Kapitel 5 werden die Ergebnisse aus Kapitel 4 zusammengeführt und mit einer Windows 10 mit *WSL2* Umgebung verglichen und bewertet. Das Ziel ist es klare Unterschiede im Lebenszyklus eines Softwareingenieurs darzustellen. Zentrale Erwartungen an Qubes OS werden definiert und beantwortet. Anschließend wird gesondert auf Stolpersteine und Grenzen von Qubes OS eingegangen. Es werden Stolpersteine, welche über den Zeitraum dieser Arbeit aufgetreten sind, erläutert.

Abschließend werden in Kapitel 6 die Ergebnisse dieser Arbeit zusammengefasst. Ein Ausblick in die mögliche Richtung, in die sich Qubes OS begibt, wird gegeben.

3 Informationssicherheit

Als Einstieg in die Thematik wird mit einem Überblick über die Signifikanz von IT-Sicherheit begonnen, welche den Grundstein dieser Arbeit bildet. Am Anfang der Ausführung stehen die herkömmlichen Sicherheitstechniken welche unabhängig von Qubes eingesetzt werden können. Anschließend werden die verschiedenen Sicherheitsmechanismen von Qubes OS in eine Übersicht gebracht. Abschließend wird der Schutz von Daten in Bezug zu Qubes erläutert, mit einer kurzen Einführung in Whonix, Tor und Whonix-Qubes.

3.1 Relevanz von Informationssicherheit

Die meistverwendeten Betriebssysteme auf Desktop- und Laptop Computern sind Windows und MacOS. Diese Betriebssysteme sind beliebt, weil sie in der Regel einfach zu bedienen sind und auf vielen Computern vorinstalliert werden. Sie bringen jedoch Probleme mit sich, wenn es um Sicherheit geht. So kann es vorkommen, dass eine harmlos aussehende E-Mail oder Webseite unbemerkt Malware auf solchen Computern ausführt. Abhängig von der Art der Malware kann sie alles ausführen, von der Anzeige unerwünschter Werbung über das Protokollieren von Tastenanschlägen bis hin zur Übernahme des gesamten Computers. Dies könnte alle Informationen gefährden, die auf diesem Computer gespeichert sind oder auf die er Zugriff hat, z.B. private Mitteilungen oder vertrauliche Unternehmensdaten, welche nicht in die Hände eines Konkurrenten fallen sollen. Darüber hinaus kann Malware auch die Aktivitäten stören, die mit diesem Computer durchgeführt werden. Wenn beispielweise Finanztransaktionen über solch einen Computer durchgeführt werden, kann Malware betrügerische Transaktionen in falschen Namen verrichten.

3.2 Herkömmliche Sicherheitsmechanismen

In den folgenden vier Unterkapiteln werden bestehende Sicherheitsmechanismen erläutert, welche vollkommen betriebssystemunabhängig eingesetzt werden können. Dies beinhaltet ebenso den Einsatz in Kombination mit Qubes OS. Hierzu wird mit Antivirenprogrammen und Firewalls begonnen und aufgeklärt wieso diese allein keine Sicherheit garantieren. Anschließend wird auf Penetrationstests eingegangen und erläutert, wie diese IT-Sicherheit erhöhen können. Abschließend soll gezeigt werden, dass die Behebung von Sicherheitslücken eines der Relevantesten Themen in der IT-Sicherheit darstellt.

3.2.1 Antivirenprogramme und Firewalls

Leider reichen herkömmliche Sicherheitsmaßnahmen wie Antivirenprogramme und (Software- oder Hardware-) Firewalls nicht mehr aus, um Angreifer abzuwehren. Heutzutage ist es zum Beispiel üblich, dass Malware-Entwickler überprüfen, ob ihre Malware von signaturbasierten Antivirenprogrammen erkannt wird. Wenn dies der Fall ist, verschlüsseln sie ihren Code, bis er von den Antivirenprogrammen nicht mehr erkannt wird, und senden ihn dann aus. Die besten dieser Programme werden anschließend aktualisiert, sobald die Antivirenprogrammierer die neue Bedrohung entdecken, was jedoch in der Regel erst einige Tage nach dem Auftauchen der neuen Angriffe geschieht. Zu diesem Zeitpunkt ist es für diejenigen, die bereits gefährdet sind, zu spät. Fortschrittlichere Antivirensoftware mag in dieser Hinsicht besser abschneiden, ist aber immer noch auf einen erkenntnisbasierten Ansatz beschränkt. Es werden ständig neue *Zero-Day*-Schwachstellen (Eine Schwachstelle, die entweder denjenigen unbekannt ist, die an ihrer Behebung interessiert sein sollten, oder die bekannt ist und für die kein Patch zur Verfügung steht) in üblich genutzte Software, z. B. in unseren Webbrowsern, entdeckt. Kein Antivirenprogramm und keine Firewall können verhindern, dass alle diese Schwachstellen ausgenutzt werden.

Antivirenanwendungen stützen sich in erster Linie auf eine Datenbank mit bekannten bösartigen Sicherheitsartefakten, die mit den Informationen verglichen werden, die der Software zur Analyse vorgelegt werden. Informationen, die von Antivirenprogrammen mit einer Signatur abgeglichen werden, werden ebenfalls als böswillig angenommen. Diese Erkennungsmethode ist mit einer Reihe von Problemen behaftet. Durch das Zuordnen jeglicher Anwendungen, welche einer bestimme Signatur entspricht, als bösartig, kennzeichnen Antivirenanwendungen häufig Anwendungen unter falschen Vorbehalt. Diese Methode der Erkennung führt dazu, dass bösartige Anwendungen erst erkannt werden, nachdem ein Forscher eines Antivirenunternehmens, eine Signatur für die bösartige Anwendung anlegt. Bis diese Signatur für Endbenutzer freigegeben wird, ist das System weiterhin anfällig. Des Weiteren bei Erkenntnis des Angreifers, dass ein Antivirenprogramm nach einer bestimmten Signatur sucht, ist er in der Lage eine neue Malware zu erstellen, welche diese Signatur nicht aufweist.

Eine Studie wurde durchgeführt, um empirisch die Annahme zu testen, dass es trivial ist, eine Antivirenanwendung zu umgehen und die Wirksamkeit von solchen Programmen zu beurteilen, wenn sie mit einer Reihe von bekannten Umgehungstechniken konfrontiert werden. Eine bekannte bösartige Binärdatei wurde mit Umgehungstechniken kombiniert und gegen mehrere Antivirenprogramme eingesetzt, um deren Erkennungsfähigkeit zu testen. Die Ergebnisse

zeigten, dass es für einen Angreifer durchaus möglich ist mehrere Antivirenprogramme zu umgehen. Allgemein kann gesagt werden, dass eine Antivirenanwendung nützlich für den Schutz vor bekannten Bedrohungen ist, im Gegensatz weniger effektiv gegen ungekannte Angriffe. [2]

3.2.2 Penetrationstests

Neben Antivirus Programmen und Firewalls existieren noch weitere Techniken, um IT-Sicherheit zu erhöhen. Ein Penetrationstest, kurz *Pentest*, ist ein empirischer Sicherheitscheck unter definierten Rahmenbedingungen für einen umfassenden Sicherheitstest einzelner Rechner oder Netzwerke. Dabei werden reale bzw. gängige Mittel und Techniken eingesetzt, die auch von potenziellen Angreifern verwendet werden. Penetrationstests zeigen eine Momentaufnahme der getesteten Komponenten aus einer bestimmten Perspektive. Generell erhöht sich die Anzahl der gefundenen Schwachstellen mit dem eingesetzten Aufwand und den verfügbaren Informationen über das zu testende System. Zielsetzungen umfassen die Identifikation von Schwachstellen, das Aufdecken von potenziellen Fehlern, die sich durch fehlerhafte Bedienung ergeben, sowie die Erhöhung der Sicherheit auf technischer und organisatorischer Ebene. Aufgrund der ständigen Veränderungen von Bedrohungsmustern und sicherheitsrelevanten Faktoren in der Informationstechnologie sind Penetrationstests jedoch eher als Momentaufnahme zu verstehen. Im Extremfall kann das System unmittelbar nach Beseitigung der im Test gefundenen Schwachstellen durch neue Sicherheitslücken angegriffen werden. [3] Durch Tests wird jedoch häufig auch die Ursache des identifizierten Problems aufgedeckt. Jedoch liegt die Verantwortung bei dem Betreiber des Testsystems und der Applikation die Ursachen zu beheben. Die Abhilfen reichen von einer besseren Unterstützung von Systemen und Anwendungen bis hin zur Abschaltung oder zum Verkauf. Des Weiteren werden Rechner und Datenbank von Unternehmen nicht immer von außen über das Netzwerk angegriffen, sondern auch durch *Social-Engineering* (Eine zwischenmenschliche Beeinflussung mit dem Ziel bestimmte Verhaltensweisen hervorzurufen) über Mitarbeiter, um an Informationen und Zugänge zu gelangen. Aufgrund dessen gibt es spezielle Penetrationstests, die sich mit diesem Thema beschäftigen. Diese Tests sind darauf ausgelegt, Schwachstellen in Unternehmen zu finden und im Nachgang durch Schulungen und Aufklärung ernsthafte Angriffe zu erschweren.

3.2.3 Bugfixing

Ein Bug wird als Sicherheitsrelevant angesehen, wenn er eine Schwachstelle in der Software schafft, welche Angreifer ausnutzen können, um das System anzugreifen. Sicherheitslücken sind in veröffentlichter Software weit verbreitet und stellen eine ernsthafte Bedrohung für Unternehmen dar. Diese können zu erheblichen finanziellen Verlusten und Rufschädigungen führen. Daher hat die Behebung von Sicherheitslücken eine der wichtigsten Prioritäten bei der Softwarewartung. Um Entwickler bei der Behebung von Sicherheitslücken zu unterstützen, haben Forscher Techniken zu der Bewertung und Identifikation von Sicherheitslücken [4] [5], zur Vorhersage verwundbarer Komponenten [6] und zum Sicherheitstest [7] [8] entwickelt. Diese Studien konzentrieren sich hauptsächlich auf die Erkennung von Sicherheitslücken und deren Behebung. Die frühzeitige und schnelle Behebung von Sicherheitsfehlern ist essenziell. In der Praxis werden Sicherheitsprobleme jedoch häufig auf eine *ad-hoc* Art behoben und der Bugfixing-Prozess dauert oft lange. Es ist üblich, dass Sicherheitsprobleme in mehreren *Commits* behoben werden, nach einer vermeintlichen Lösung wieder geöffnet werden oder aufgrund von anderen Fehlern ungelöst bleiben [9]. Diese Ergebnisse deuten darauf hin, dass es einige Faktoren gibt, welche die Behebung von Sicherheitsproblemen erschweren. Um Entwicklern dabei zu helfen, Sicherheitslücken schneller und einfacher zu beheben, ist es notwendig zu erfahren, wie Sicherheitslücken in der Praxis behoben werden und wie dies verbessert werden kann. Mit dem Testen wird oft erst sehr spät im Lebenszyklus der Softwareentwicklung begonnen, kurz bevor sie eingesetzt wird. Es hat sich gezeigt, dass dies eine sehr ineffektive und unwissenschaftliche Praxis ist. Eine der besten Methoden, um das Auftreten von Sicherheitslücken in Produktionsanwendungen zu verhindern, ist den Lebenszyklus der Softwareentwicklung zu verbessern, indem man die Sicherheit in jede seiner Phasen einbezieht und ihn so zu einem sicheren Lebenszyklus der Softwareentwicklung erweitert. Ein Softwareentwicklungszyklus ist eine Reihe von Schritten oder Phasen, die ein Modell für die Entwicklung und das Lebenszyklusmanagement einer Anwendung oder eines Softwareprodukts darstellen.

Ein allgemeines Lebenszyklusmodell für die Softwareentwicklung, dass das Testen als orthogonale Dimension berücksichtigt, umfasst die Phasen Analyse, Entwurf, Entwicklung, Einsatz und Wartung. Jede Phase liefert spezifische Artefakte, d. h. die Analysephase führt zu Anforderungen, der Entwurf liefert Entwurfsmodelle, die Entwicklung liefert Code, die Bereitstellung führt zu einem laufenden System und schließlich werden alle Artefakte gewartet. Bei einem sicheren Softwareentwicklungszyklus werden Sicherheitsaspekte in jeder Phase der Softwareentwicklung berücksichtigt. Ein entscheidendes Konzept innerhalb des Lebenszyklus

der sicheren Softwareentwicklung ist das Risiko. Ein Risiko ist die Wahrscheinlichkeit eines unerwünschten Ereignisses und dessen Folgen für ein bestimmtes Gut. In Anbetracht des negativen Charakters vieler Sicherheitsanforderungen kann das Risikokonzept verwendet werden, um die Auswahl oder Anwendung von Sicherheitsmaßnahmen wie Tests zu steuern. In allen Phasen des Entwicklungsprozesses für sichere Software, insbesondere aber auf der Entwurfsebene bieten Risikoanalysen ein wirksames Mittel zur Steuerung von Sicherheitstests und damit zur Aufdeckung von Fehlern und Verwundbarkeiten. [7]

3.2.4 Security-Patching

Software für Computernetzwerke, Systeme und Anwendungen ist in der Regel mit Informationssicherheitslücken behaftet. Solche Sicherheitslücken können bei Ausnutzung zu erheblichen finanziellen Verlusten bei Unternehmen führen. Wenn Schwachstellen auftauchen, geben Softwarehersteller in regelmäßigen Abständen entsprechende *Patches* heraus. Für große Organisationen mit zehn- oder sogar hunderttausenden von Netzwerkgeräten ist die Bereitstellung von *Patches* eine kostspielige Angelegenheit, die sich erheblich auf die Systemverfügbarkeit auswirkt, was wiederrum Folgen für die Eigenschaften der Geschäftsprozesse, die Glaubwürdigkeit und die Einnahmen hat. Wenn ein Patch jedoch nicht bereitgestellt wird, besteht die Gefahr, dass die Host-Organisation der Ausnutzung von Schwachstellen ausgesetzt ist. Das für die Informationssicherheit zuständige Managementteam der Organisation muss unter Berücksichtigung der Unternehmensrichtlinien den richtigen Zeitpunkt für die Bereitstellung von Patches festlegen. Wie in anderen Bereichen der Informationssicherheit müssen auch bei der Entscheidung über den Einsatz von *Patches* Kompromisse zwischen dem Schutz der Vertraulichkeit des Systems und der Aufrechterhaltung seiner Verfügbarkeit eingegangen werden. [10]

3.3 Sicherheitsmechanismen von Qubes OS

Nach einer Auflistung von gängigen Sicherheitsmethoden, wird nun genauer auf die Sicherheit durch Qubes OS eingegangen. Qubes verfolgt einen Ansatz, der als Sicherheit durch Kompartimentierung bezeichnet wird und er es ermöglicht, die verschiedenen Bereiche von digitaler Nutzung in sichere isolierte Abteilungen, die „Qubes“ oder auch AppVMs genannt werden, aufzuteilen. Anzumerken ist, dass herkömmliche Sicherheitsmechanismen in

Kombination mit dem Sicherheitsansatz von Qubes OS eingesetzt werden. In den folgenden zwei Unterkapiteln wird auf die Sicherheit und den Datenschutz durch Qubes OS eingegangen.

3.3.1 Sicherheit durch Qubes OS

Der Ansatz der Kompartimentierung erlaubt es, verschiedene Tätigkeiten auf einem Computer sicher voneinander getrennt in isolierten AppVMs zu halten, so dass eine virtuelle Maschine, die kompromittiert wird, andere nicht beeinträchtigt. So kann beispielweise das Öffnen von nicht vertrauenswürdigen Webseiten und Online-Banking in separaten AppVMs eingerichtet werden. Auf diese Weise sind wichtige Handlungen nicht gefährdet, wenn eine Qube, welche für nicht vertrauenswürdige Webseiten eingerichtet wurde, durch Malware kompromittiert ist. Des Weiteren bestehen Möglichkeiten, dass E-Mail-Anhänge immer in eigenen Einweg-VMs/*DisposibleVMs* geöffnet werden. Auf diese Weise kann alles auf demselben physischen Computer erledigt werden, ohne dass ein einziger erfolgreicher Cyberangriff das gesamte System kompromittiert. In Abbildung 1 ist zu sehen, wie *DisposibleVMs* für zwei verschiedene Typen von E-Mails verwendet werden. Es wird demonstriert, dass eine DisposibleVM nach Beendigung immer vollkommen zerstört wird, inklusive möglicherweise mitinstallierter Malware.

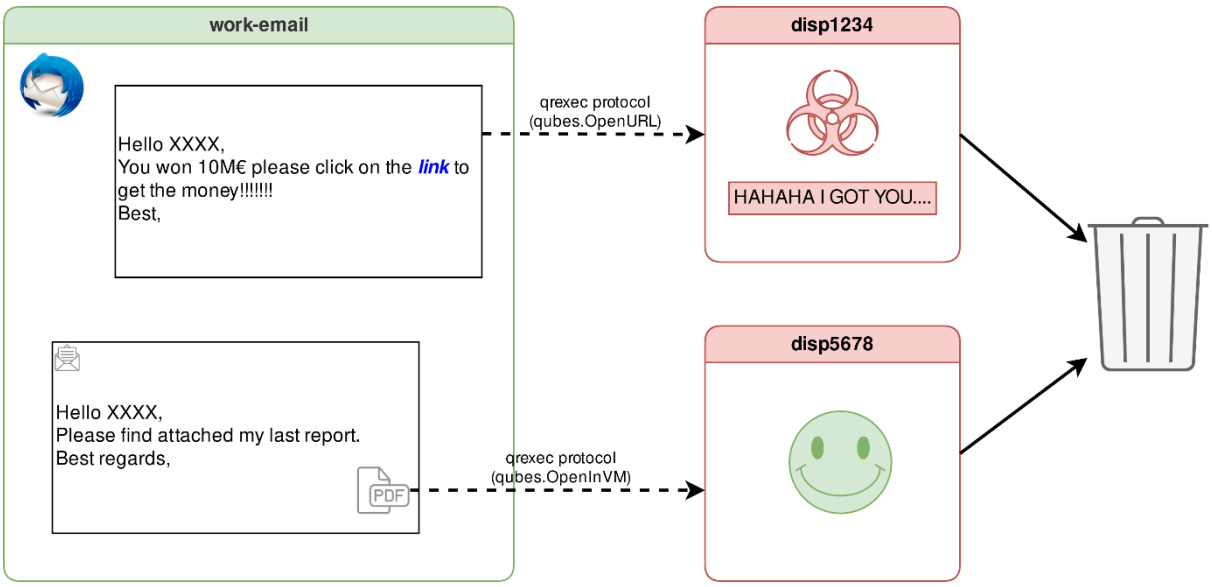

Abbildung 1: *Disposible* VMs für das öffnen nicht vertrauenswürdiger Links und Anhänge

Allerdings bei Kompromittierung einer *DisposibleVM*-Vorlage[3], kann jede *DisposibleVM*, welche auf dieser Einwegvorlage basiert, gefährdet sein. Insbesondere ist diese Vorlage wichtig, da sie von der Funktion „open in disposable“ verwendet wird. Das bedeutet, dass sie Zugriff auf alle Dateien hat, die mit dieser Funktion geöffnet werden. Aus diesem Grund sollte die *DisposibleVM*-Vorlage auf einer vertrauenswürdigen Vorlage[4] gestützt sein.

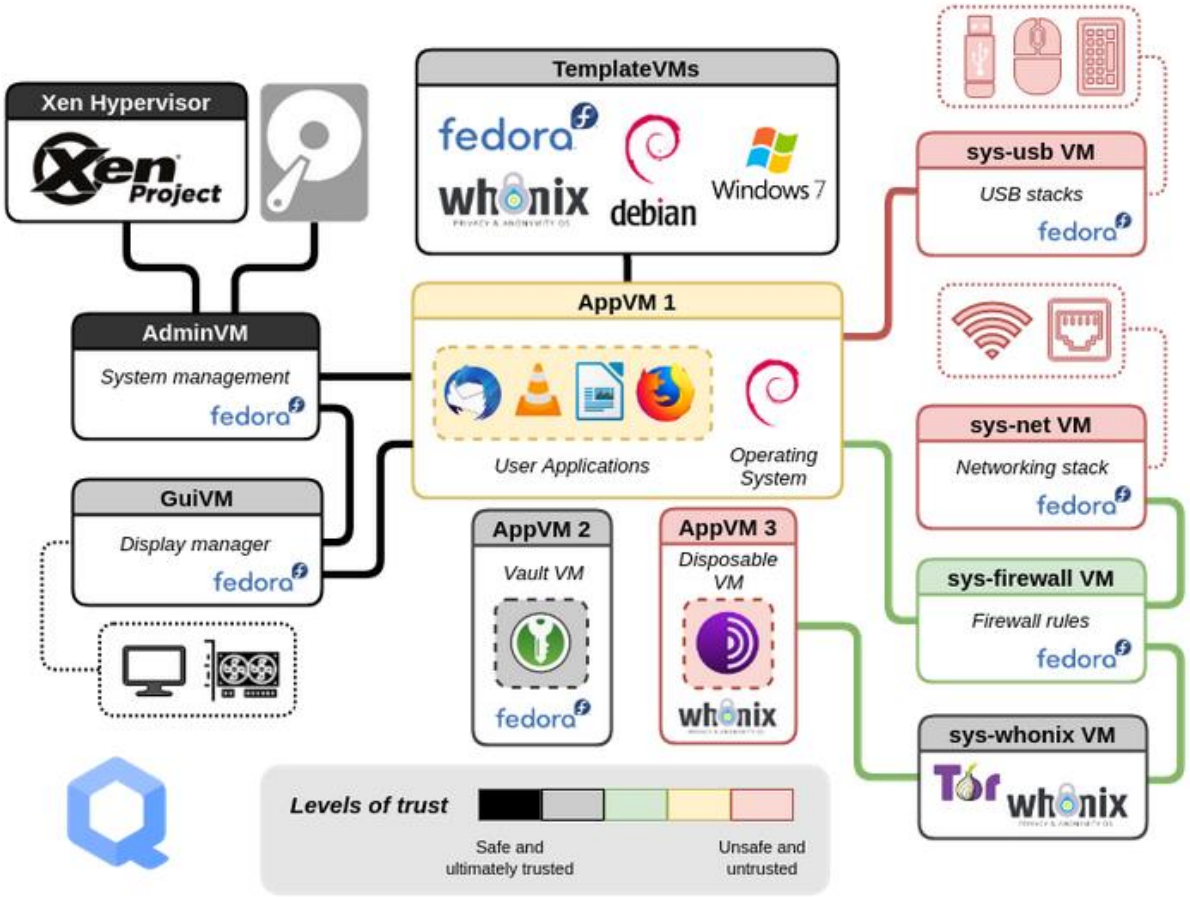

Abbildung 2: Das Architekturdiagramm von Qubes OS.

In der Abbildung 2 ist ein Architekturdiagramm von Qubes OS zu sehen. Hier lässt sich die Unterteilung in isolierte Bereiche, die sogenannten Qubes, erkennen. Durch die farbige Unterteilung lassen sich die einzelnen Sicherheitsstufen ermessen, wobei rot als unsicherste Stufe gekennzeichnet ist. Unter diese Stufe fallen auch die oben genannten DisposableVMs, welche sich nach Beendigung selbst zerstören. Ebenso sind die zuvor genannten AppVMs zu

[3] Eine Vorlage, welche für die Erstellung von DisposibleVMs herangezogen wird.
[4] DisposibleVM-Vorlagen sind auf Template VMs gestützt welche auch für die Erstellung von anderen VMs z.B. AppVMs zuständig sind.

sehen, welche grundlegend für Anwendungen ausgelegt sind. Allerdings wie in Abbildung 2 auch zu sehen, fallen DisposibleVMs auch in die Kategorie von AppVMs.

Darüber hinaus sind diese isolierten Qubes in ein einziges, benutzbares System integriert. Programme werden in ihre eigenen, separaten Bereiche isoliert, während alle Fenster in einer einzigen, einheitlichen Desktop-Umgebung mit nicht fälschbaren farbigen Fensterrändern angezeigt werden, so dass verschiedene Sicherheitsstufen leicht erkennbar sind. Übliche Angriffsvektoren wie Netzwerkkarte und USB-Controller werden in eigenen Hardware-Qubes isoliert, während ihre Funktionalität durch sichere Netzwerke, Firewalls und USB-Geräteverwaltung erhalten bleibt. Aber die bloße Erstellung einer VM für das Bereitstellen von USB-Controllern löst kaum ein Problem automatisch. Der eigentliche Sicherheitsvorteil ergibt sich aus der Reduzierung der Schnittstellen, die in dieser Software-Kompartimentierung zu Verfügung stehen. Derzeit gibt es nur begrenzte Unterstützung für den Umgang mit USB-Geräteklassen. Dies bedeutet, dass bei Nutzung einer USB-basierten Kamera in einer AppVM (z.B. für Online-Konferenzen), der USB-Controller (d.h. ein eigenes PCIe-Gerät) dieser VM zugewiesen werden muss. Dies bietet zwar gute Isolierung, allerdings ist der USB-Bus von Natur aus unsicher, so dass eine Isolierung auf Ebene des USB-Controllers besser ist. Wobei die Bequemlichkeit und Nützlichkeit dieses Ansatzes davon abhängt, wie viele physische USB-Controller in einem Laptop vorhanden sind und wie die Geräte mit ihm verbunden sind. Es ist wichtig, darauf hinzuweisen, dass es eine Klasse von USB-Geräten gibt, die so sicherheitsempfindlich ist, dass es wenig Sinn macht, ihre Handhabung an eine USB-VM zu delegieren. Hierbei handelt es sich um Mäuse und Tastaturen. Schafft es ein Angreifer diese zu übernehmen, besteht die Gefahr, dass das gesamte System kompromittiert wird. Das bedeutet, dass eine USB-Tastatur Zugriff auf eine VM haben muss und diese VM wiederum Zugriff auf das gesamte System. Denn Geräte der Art möchte ein Nutzer natürlich über das gesamte System hinweg benutzen. [11]

Ein weiterer Bereich, welcher bei Virtualisierungssystemen Anlass zur Sorge gibt, sind Daten-Leaks zwischen VMs. Hierbei wird zwischen zwei Arten von Leaks unterschieden, einerseits kooperative verdeckte Kanäle, in denen sich Malware in zwei bereits kompromittierten VMs zusammenschließen, um Daten über einen nicht autorisierten Kommunikationskanal auszutauschen. Andererseits existieren Seitenkanäle (nicht-kooperative verdeckte Kanäle), bei denen Malware in einer VM versucht, einige Fakten über Prozesse in anderen nicht kompromittierten VMs zu erfahren, um beispielweise private Schlüssel-Bits zu erraten. Es ist wichtig zu betonen, wie sehr sich diese beiden potenziellen Angriffe auf die Sicherheit

auswirken. Während der erste für die meisten Arbeitsabläufe weitgehend irrelevant ist, ist der zweite in praktisch jedem Szenario fatal. [12, pp. 12-13]

Die kooperativen verdeckten Kanäle sind durch eine Vielzahl überraschender Mittel möglich. Sie bieten oft nur eine sehr geringe Bandbreite und sind oft sehr unzuverlässig (vor allem in einem Desktop-System wie Qubes OS, auf dem Dutzende VMs gleichzeitig laufen können). Es wird allgemein davon ausgegangen [12, p. 12], dass die Eliminierung solcher kooperativen verdeckten Kanäle auf *COTS* x86-Hardware sehr schwierig oder unmöglich ist. Seitenkanäle hingegen sind fast ausschließlich als Krypto-Operationen mit dem Ziel gerichtet, zumindest einige Bits von privaten Schlüsseln zu extrahieren. Bestimme Krypto-Operationen, die immer wieder durchgeführt werden, erhöhen die Erfolgsaussichten eines Angreifers. Eine Desktop-Umgebung, wie Qubes OS, in der der Benutzer z.B. GPG-Verschlüsselungs-/Entschlüsselungsoperationen nur wenige Male pro Tag aufruft, erschwert den Vorgang deutlich. [12]

Darüber hinaus bietet Qubes integrierte kopier- und einfüge Vorgänge für Dateien und die Zwischenablage, welche die Arbeit über verschiedene Qubes hinweg erleichtern, ohne die Sicherheit zu gefährden. Das Template-System trennt die Software-Installation von der Software-Nutzung und ermöglicht es den AppVMs, ein gemeinsames Root-Filesystem zu nutzen, ohne die Sicherheit zu beeinträchtigen.[5] Ebenso ermöglicht Qubes OS PDFs und Bilder mit wenigen Klicks von möglichen Malware Anhängen zu reinigen, denn selbst diese Dateiformen können Malware mitführen.

3.3.2 Datenschutz durch Qubes OS

Es kann keinen Datenschutz ohne Sicherheit geben, da Sicherheitslücken die Umgehung von Datenschmutzmaßnahmen ermöglichen. Aufgrund dessen eignet sich Qubes besonders gut für die Implementierung effektiver Datenschutzwerkzeuge.

Qubes OS bietet in erster Linie Datenschutz durch die Integration von Whonix. Qubes gewährleistet keine eigenen Datenschutz- (im Gegensatz zu Sicherheits-) Eigenschaften in nicht-Whonix-Qubes. Zum Beispiel wird erwartet, dass eine Standard-Fedora-Qube im Grunde die gleichen Privatsphäre-Eigenschaften hat wie die vorgelagerte Fedora-Distribution, bis zu

[5] Software wird allgemein innerhalb von Template VMs und nicht innerhalb von AppVMs installiert. Dies hängt mit dem fehlenden Root-Filesystem von AppVMs ab. In Kapitel 4.1.2 und 4.2.2 wird genauer auf die Bedeutung dieser Aussage eingegangen.

einem gewissen Grad erweitert durch die Kontrolle, die Qubes über diese AppVM bietet. Für die meisten Benutzer mag dieses Maß an Datenschutz für viele gewöhnliche Aktivitäten ausreichen. Benutzer, die fortgeschrittenere Datenschutzfunktionen suchen, sollten jedoch Whonix-Qubes verwenden.

Der Schutz der Privatsphäre ist weitaus schwieriger, als häufig angenommen wird. Neben dem Webbrowser gibt es auch VM-Fingerabdrücke und fortgeschrittene Angriffe zur Deanonymisierung, an die die meisten Benutzer noch nie gedacht haben (und dies sind nur einige Beispiele). [13] Das Whonix-Projekt ist auf den Schutz vor diesen Risiken spezialisiert.

Um die gleichen Ergebnisse in Nicht-Whonix-Systemen (einschließlich DisposibleVMs) zu erzielen, müsste Whonix von Grund auf neu entwickeln werden. Eine solche Verdoppelung des Aufwands macht keinen Sinn, wenn Whonix bereits existiert und in Qubes OS integriert ist.

3.4 Whonix

Whonix ist ein open-source Desktop-Betriebssystem, das speziell für fortgeschrittene Sicherheit und Privatsphäre entwickelt wurde. Basierend auf Tor, Debian GNU/Linux und dem Prinzip der Sicherheit durch Isolation, geht Whonix realistisch auf gängige Angriffsvektoren ein. Online-Anonymität und die Umgehung von Zensur ist durch die ausfallsichere, automatische und desktopweite Nutzung des Tor-Netzwerks möglich, d.h. alle Verbindungen werden durch Tor erzwungen oder blockiert. Das Tor-Netzwerk schützt vor der Analyse des Datenverkehrs, indem es die Kommunikation über ein verteiltes Netzwerk von Relais weiterleitet, das von Freiwilligen weltweit betrieben wird. Ohne fortschrittliche, durchgängige Angriffe auf die Korrelation von Datenströmen kann ein Angreifer, der eine Internetverbindung überwacht, nicht ohne weiteres die besuchten Webseiten ermitteln und diese Webseiten können den physischen Standort des Nutzers nicht feststellen.

Whonix verwendet eine umfassend rekonfigurierte Debian-Basis (*Kicksecure Security Hardened*), die in mehreren virtuellen Maschinen auf dem Host-Betriebssystem ausgeführt wird. Diese Architektur bietet einen umfangreichen Schutz vor Malware und *IP-Leaks*. Whonix ist das einzige aktive Betriebssystem, das für den Betrieb in einer Virtuellen Maschine und in Verbindung mit Tor entwickelt wurde. Obwohl es sich technisch gesehen um ein Desktop-Betriebssystem handelt, ist Whonix mit seiner Sicherheits- und Anonymitätswerkzeugen ideal geeignet mit Tor-Diensten eingesetzt zu werden.

3.4.1 Tor-Netzwerk

Tor ist ein Netzwerk zur Anonymisierung von Verbindungsdaten. Die Software basiert auf einem Prinzip des *Onion-Routings* und wurde mit einigen Abwandlungen implementiert.

Hierbei werden die Webinhalte über ständig wechselnde Routen geleitet, welche in diesem Zusammenhang auch Knoten oder Relais genannt werden. Der Nutzer bekommt eine Liste von Relais über einen *Onion-Proxy*, welcher mit dem Tor-Netzwerk verbunden ist. Diese Liste ist mit einer digitalen Signatur versehen und wird von einem Verzeichnisserver aufbewahrt. Deren öffentliche Schlüssel werden mit dem Tor-Quellcode geliefert. Dieses Verfahren soll sicherstellen, dass der *Onion-Proxy* authentische Verzeichnisdaten enthält. Nachdem eine Liste empfangen wurde, wählt der *Onion-Proxy* eine zufällige Route über die Tor-Server. Der Client baut eine verschlüsselte Verbindung mit dem ersten Tor-Server auf. Wenn dieser verhandelt ist, wird sie um einen weiteren Server verlängert. Diese Prozedur wiederholt sich, so dass eine Verbindungskette von mindestens drei Tor-Servern entsteht. Die Zahl Drei wurde gewählt, um eine möglichst große Anonymität bei noch akzeptabler Verzögerungszeit zu erreichen. [14] Darüber hinaus werden die Pakete innerhalb des Tor-Netzwerkes immer verschlüsselt weitergegeben. Dadurch bleibt die wahre Identität desjenigen, der die Daten angefordert hat, für den Webserver auf der anderen Seite anonym. In Abbildung 3 ist das Kommunikationsmodel von Tor dargestellt. In Rot gekennzeichnet ist der Verzeichnisserver (Directory Server), welcher die digitalen Signaturen aufbewahrt. In blau sind die zuvor erwähnten Knoten markiert. Unter diesen blauen Knoten wird eine zufällige Route gewählt. Zwischen den Eintritts- und -Austrittsknoten wird separat unterschieden.

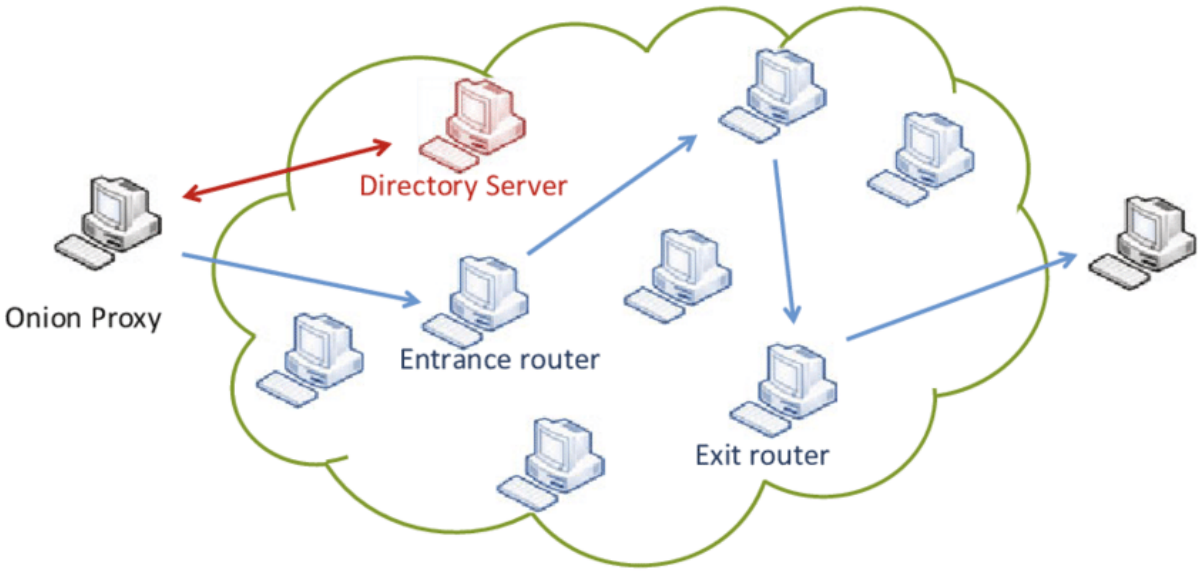

Abbildung 3: Die Tor Kommunikation auf Basis des Onion-Routings.

3.4.2 Unterschied zwischen Whonix und herkömmlichen VPNs

Virtual Private Networks (VPNs) kennen die Identität und jegliche Online-Aktivitäten ihrer Nutzer. Diese können unter verschiedenen Umständen rechtlich gezwungen werden, diese Informationen an Behörden weiterzugeben.

VPNs sind in der Regel schneller als Tor, jedoch sind sie keine Anonymitätsnetzwerke. [15] VPN-Administratoren können sowohl den Ort, von dem sich ein Benutzer verbindet, als auch die Ziel-Website protokollieren und damit die Anonymität aufheben. Versprechungen von VPN-Betreibern sind bedeutungslos, da sie nicht überprüft werden können. Tor bietet Anonymität durch Design, da es für einen einzelnen Punkt im Netzwerk unmöglich ist, sowohl den Ursprung als auch das Ziel einer Verbindung zu kennen. Anonymität durch Design bietet mehr Sicherheit, da Vertrauen aus der Gleichung entfernt wird.

3.4.3 Whonix-Qubes

Qubes-Whonix ist die Kombination von Qubes OS und Whonix für erweiterte Sicherheit und Anonymität. In dieser Konfiguration läuft Whonix in Kombination mit Qubes in virtuellen Maschinen. Das Design bietet eine zweigeteilte Sicherheitsarchitektur: Ein isoliertes *Whonix-Gateway* für die vollständige Weiterleitung des Datenverkehrs über Tor und *Whonix-Workstation* für alle Desktop-Anwendungen, welches als maßgeschneiderte Betriebssystemumgebung für Tor-Umgebung dient, um Tor-basierte Privatsphäre/Anonymität zu gewährleisten. Die Whonix TemplateVMs können für die Anpassung und Erstellung mehrerer Whonix-Gateway *ProxyVMs* und Whonix-Workstation *AppVMs* verwendet werden, was eine verbesserte Kompartimentierung der Benutzeraktivitäten für mehr Privatsphäre ermöglicht.

3.4.4 Zusammenfassung

In diesem Kapitel wurde ein Überblick über die Signifikanz von IT-Sicherheit gegeben, welcher den Grundstein dieser Arbeit bilden sollte. Am Anfang der Ausführung standen herkömmlichen Sicherheitstechniken welche unabhängig von Qubes eingesetzt werden können. Hierunter zählten Antivirenprogramme, Firewalls, Penetrationstests, Bugfixing und Security-Patching. Einzeln wurden diese Technologien erläutert. Aufbauend auf diesem Wissen wurden die Sicherheitsmechanismen von Qubes OS aufgelistet und erklärt. Der allgemeine Aufbau von Qubes OS ist durch ein Architekturdiagram aufgezeigt wurden. Abschließend wurde der Schutz

von Daten in Bezug zu Qubes OS gebracht. Eine Einführung in Whonix, Tor und Whonix-Qubes schloss das Kapitel ab, um eine gute Basis für die restliche Arbeit zu schaffen.

4 Analyse von Nutzungsszenarien unter Qubes OS

Dieses Kapitel analysiert eine Reihe von relevanten Nutzungsszenarien von Software-Entwicklungsunternehmen. Hierzu werden Abläufe einzelner Arbeitsbereiche und Funktionen untersucht, um die Tauglichkeit von Qubes OS hinsichtlich der Benutzung als Betriebssystem für Softwareingenieure zu prüfen. Hierbei wird mit den administrativen Nutzungsszenarien begonnen. Darunter fällt einerseits die Installation von Qubes OS. Es werden Methoden für die Installation des Betriebssystems aufgelistet und erläutert. Ebenso werden die Möglichkeiten für die Installation von Software analysiert. Hierunter fallen Programme für den generellen Nutzen innerhalb von kleinen- und mittelständischen Software-Entwicklungsunternehmen. Beispiele dafür sind Büroanwendungen, Konferenz-Tools sowie Programme für die Softwareentwicklung. Es wird darauf eingegangen, wie IT-Administrationen Software automatisch auf mehreren Rechnern verteilen können. Im Abschnitt 4.2 wird auf allgemeine Nutzungsszenarien eingegangen. Typische Büroanwendungen und Konferenzwerkzeuge werden untersucht und mit standartgemäß verwendeten Anwendungen unter Windows verglichen. Ebenso werden die Grundlagen des Dateimanagements unter Qubes OS geklärt. Abschließend werden die Nutzungsszenarien für Entwicklungsprozesse untersucht. Einerseits die Entwicklung von Server-Backends und andererseits die Entwicklung von Anwendungen mit Web-, 2D- und 3D-fähigen Oberflächen.

4.1 Administrative Nutzungsszenarien

In diesem Abschnitt werden administrative Nutzungsszenarien vorgestellt und aufgezeigt, wie die Installation von Qubes OS abläuft. In Kapitel 4.1.2 werden zwei Möglichkeiten für die Installation von Software aufgelistet. Ebenfalls wird erläutert, wie Anwendungen und Konfigurationen automatisch über viele Qubes OS Rechner verteilt werden können. Abschließend wird beschrieben, wie Windows innerhalb von Qubes OS installiert werden kann.

4.1.1 Installation von Qubes OS

Zu Beginn ist anzumerken, dass Qubes OS sehr spezifische Systemanforderungen hat. Um Kompatibilität zu gewährleisten, wird empfohlen Qubes-zertifizierte Hardware zu verwenden. Bei anderer Hardware muss möglicherweise eine Fehlersuche durchgeführt werden. Auf der Qubes OS Internetseite ist eine Übersicht von Systemanforderungen und eine Liste von

kompatibler Hardware zu finden[6]. Empfehlenswert ist es diese beiden Aspekte vor einer Installation abzugleichen. Des Weiteren muss sichergestellt werden, dass IOMMU-basierte Virtualisierung im BIOS oder UEFI aktiviert ist. Ohne diese Einstellung ist Qubes nicht in der Lage, die gewollte Isolierung durchzusetzen. Anzumerken ist, dass Qubes OS nicht dafür gedacht ist, innerhalb von virtuellen Maschinen installiert zu werden. Also verschachtelte Virtualisierung wird nicht unterstützt, damit eine strikte Kompartimentierung durchgesetzt werden kann, muss Qubes OS die Hardware direkt verwalten können.

4.1.1.1 Kopierung des Qubes ISO auf ein Installationsmedium

Anbeginn sollte das sicherste zu Verfügung stehende Betriebssystem für das Herunterladen des Qubes-ISO verwendet werden. Sobald die ISO-Datei als authentisch bestätigt wurde, kann sie auf ein beliebiges Installationsmedium installiert werden, beispielweise ein USB-Laufwerk. Zu beachten ist, dass jede aus dem Internet heruntergeladen Datei bösartig sein könnte, selbst wenn sie von einer vertrauenswürdigen Quelle stammt. Kein Betriebssystem, auch nicht Qubes OS, schützt wenn die Installation auf Hardware durchgeführt wird, die bereits kompromittiert ist.

Qubes-ISO auf USB-Medium via Linux

```
$ sudo dd if=Qubes-RX-x86_64.iso of=/dev/sdY status=progress
bs=1048576 && sync
```

„Qubes-RX-x86_64.iso“ muss für die zu Installierende Version angepasst werden. Ebenso muss „/dev/sdY“ in das richtige Zielgerät geändert werden.

Qubes-ISO auf USB-Medium via Windows

Unter Windows kann die Rufus-Software verwendet werden, um das ISO auf ein USB-Medium zu schreiben. Rufus ist eine freie Anwendung, um startfähige USB-Datenträger zu erstellen.

[6] https://www.qubes-os.org/doc/system-requirements/

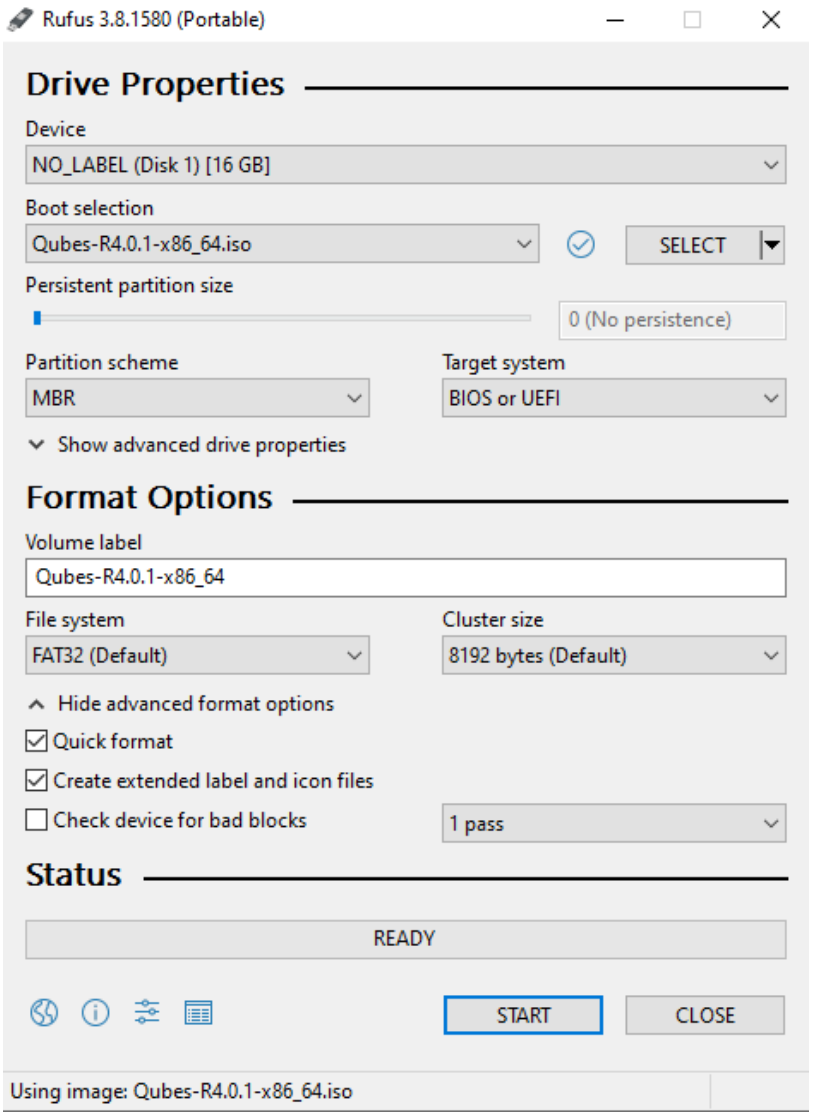

Abbildung 4: Rufus-Software für die Installation von Qubes OS auf einen USB-Medium.

Nach Starten des Installationsprozesses (s. Abbildung 4) muss anschließend „Write in DD Image mode“ im darauffolgenden Fenster ausgewählt werden. Nach Abschluss des Rufus-Prozesses kann mit der tatsächlichen Installation von Qubes OS begonnen werden.

4.1.1.2 Installation von Qubes OS über ein USB-Stick

Für die folgende Installationsbeschreibung wird angenommen, dass der Computer vollkommen überschrieben werden kann oder bereits auf Werkseinstellungen zurückgesetzt wurden ist und keine wichtigen Daten mehr enthält.

Zu Beginn der Installation muss auf das BIOS- oder UEFI-Menü des Computers zugegriffen werden, um den Computer von dem zuvor eingerichteten USB-Stick starten zu lassen. Um in das BIOS- oder UEFI-Menü zu gelangen, muss direkt nach dem Betätigen des Einschaltknopfes

eine bestimmte Taste gedrückt werden. Diese Taste variiert zwischen verschiedenen Herstellern. Für Lenovo ThinkPad Laptops ist z.B. die F1-Taste üblich. Vergleichsweise häufig verwendete Tasten sind sein: Entf, Esc, F1, F2, F10 oder F12. Nach dem Zugreifen auf das BIOS muss zu dem „Startup“ Menü navigiert werden, indem zugewiesen werden kann auf welche Art das System hochgefahren soll. Das Ziel ist es, dem Computer mitzuteilen, dass er von dem korrekten USB-Laufwerk starten soll, damit das Qubes-Installationsprogramm ausgeführt werden kann. In der Liste der Boot-Optionen ist es potenziell nicht direkt einsehbar, welches Gerät der korrekte USB-Anschluss ist. Bei mehreren Einträgen kann jedes USB-Laufwerk ohne Bedenken ausprobiert werden bis es funktioniert. Nach abgeschlossener Auswahl kann die F10 Taste betätigt werden, um zu speichern und das BIOS zu verlassen. Nach erneutem Starten und korrekter Wahl des USB-Laufwerkes, wird die Option für die Installation von Qubes-OS eingeblendet. Gegebenenfalls muss unter dem Punkt „Security“ im BIOS der Unterpunkt „Secure-Boot“ ausgeschalten werden, falls es zu Problemen kommt.

Nach korrektem Ablauf führt Qubes OS nun einige standartgemäße Installationsabfolgen aus, wie die Festlegung der Sprache, des Keyboard-Layouts, der Zeitzone und der zu verwendenden Festplatte. Ebenso verwendet Qubes automatisch Festplattenverschlüsselung via *AES* mit *LUKS*. Das hierzu festgelegte Passwort ist bei Verlust nicht mehr wiederherstellbar, deshalb sollte das Passwort mit Bedacht ausgewählt werden. Für eine genauere Beschreibung des Installationsablaufes mit Visualisierungen für jeden Schritt, verweise ich auf die offizielle Installationsanleitung. [7]

4.1.2 Installation von Programmen

Nach der Einrichtung des Betriebssystems selbst, kommen wir nun zu Programmen unter Qubes OS. Qubes OS unterstützt verschiedene Arten von virtuellen Maschinen. Für Installationen von Programmen muss hierbei unterschieden werden. Wir unterscheiden zwischen drei Arten von VMs:

1. App VM
2. Template VM
3. Standalone VM

Der große Unterschied einer AppVM ist, dass sie kein eigenes Root-Filesystem besitzt. Sie teilt das Root-Filesystem mit der Template VM aus welcher sie erstellt wurde. Eine AppVM besitzt

[7] https://www.qubes-os.org/doc/installation-guide/

dennoch ein Homeverzeichnis. Somit ist es trotzdem möglich Dateien und auch Applikationen direkt in einer AppVM herunterzuladen. Allerdings bei Installation eines Programmes, welches auf das Root-Filesystems zurückgreifen muss, wird das Programm bei Neustart der AppVM nicht mehr funktionieren. Um dies zu vermeiden, sollten Anwendungen immer in Template VM installiert werden. Das bedeutet, wenn ein Programm in einer AppVM installiert werden soll, welche auf einem Fedora-32 Template basiert, muss das Programm innerhalb dieses Fedora-Templates installiert werden. *Standalone* VMs sind aus Template VMs erstellt, teilen allerdings ihr Root-Filesystem nicht mit anderen VMs. Somit können in *Standalone* VMs Anwendungen simultan zu normalen Linux-Distributionen installiert werden. Sie haben allerdings nicht den Vorteil der neuen Erstellung aus der entsprechenden Template VM beim Starten der virtuellen Maschine, welche eine AppVM aufweist.

Installation über die Konsole innerhalb einer Template VM:

Fedora: `$ sudo dnf install <PACKAGE_NAME>`

Debian: `$ sudo apt install <PACKAGE_NAME>`

Anschießend muss das Template selbst und alle Qubes die auf diesem Template basieren neugestartet werden. Nun kann in dem Reiter „Applikationen“ innerhalb der Qube-Einstellungen die installierte Anwendung gefunden und ausgewählt werden. Die Abbildung 5 zeigt, wie Applikationen innerhalb der Einstellungen ausgewählt werden können. In Abbildung 6 sieht man nun, dass die Ausgewählten Programme als Shortcuts zur Verfügung stehen. Die Programme können anschließend über diese Shortcuts oder über ein Terminal gestartet werden.

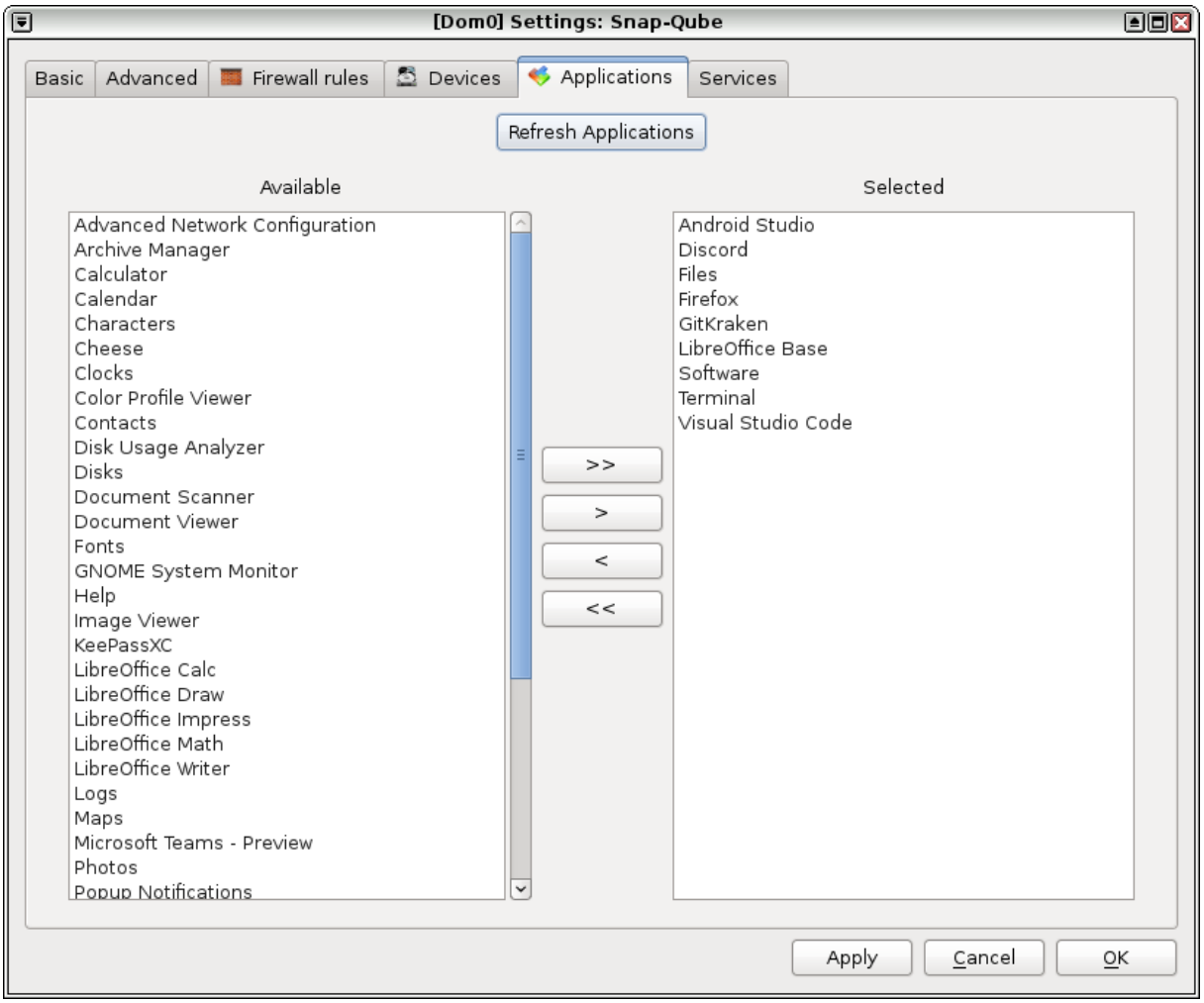

Abbildung 5: Auswahl von Applikationen innerhalb der Qube-Einstellungen.

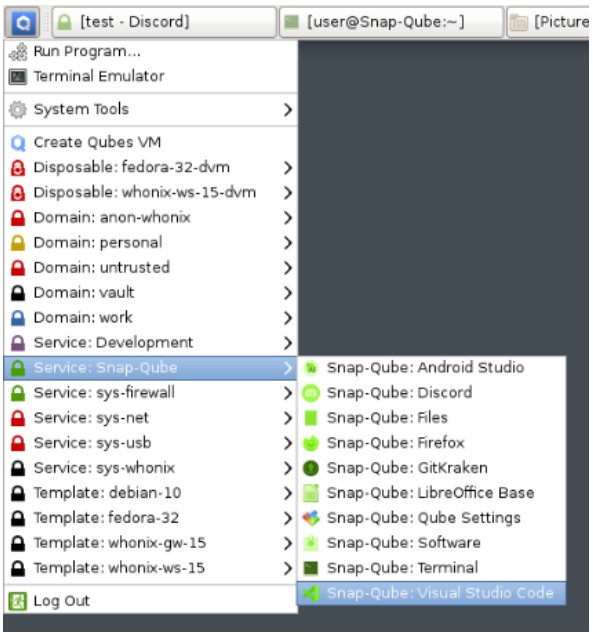

Abbildung 6: Shortcuts der ausgewählten Applikationen.

Paketmanager Snap

Eine andere Möglichkeit Software zu installieren ist der Paketmanager „Snap“. Snap ist kompatibel mit verschiedensten Linux-Distributionen und dementsprechend auch mit Qubes OS. Für die Installation von Snap auf einem Fedora-Distro:

Fedora: `$ sudo dnf install snapd qubes-snapd-helper`

Anschließend muss die Template-VM heruntergefahren werden:

Fedora: `$ sudo shutdown -h now`

Nun kann eine riesige Auswahl von unterstützen Programmen mit nur einem Befehl, ohne dass sich der Benutzer über das Management von Dateien kümmern muss, heruntergeladen werden. Innerhalb einer AppVM basierend auf dem Fedora-Template:

AppVM: `$ snap install <package>`

Abbildung 7 zeigt eine Übersicht der ersten Seite von Entwicklungsprogrammen, welche über Snap installierbar sind.

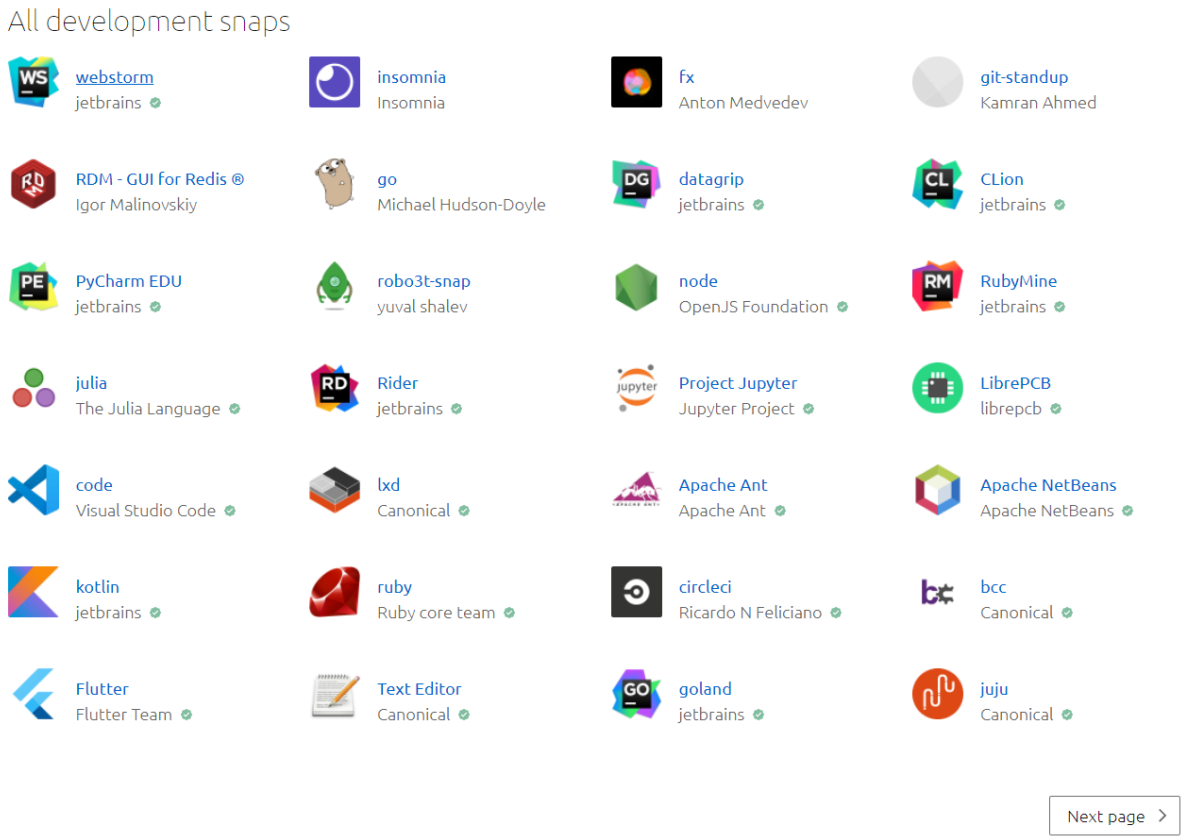

Abbildung 7: Eine Reihe von Applikationen in der Entwicklungs-Kategorie auf snapcraft.io.

4.1.3 Automatisch Anwendungen verteilen und konfigurieren

In Software-Entwicklungsunternehmen spielt das automatische Verteilen und Konfigurieren von Programmen eine große Rolle. Je mehr Arbeitsplätze ein Unternehmen besitzt, desto mehr wächst auch der Bestand nach Computer- und Netzwerktechnik. Typischerweise möchten Unternehmen eine Reihe von Computern auf die gleiche Art und Weise einrichten, sprich gleiche Programme und Einstellungen. Unpraktisch wird es, wenn die Einrichtung, wie in Kapiteln 4.1 und 4.2 beschrieben, dutzende Male wiederholt werden müsste.

Für Windowssysteme steht eine zahlreiche Auswahl an Hilfssoftware zu Verfügung, mit der Programme über mehrere Computer hinweg installiert und aktualisiert werden können. Eine weitere Option sind die von Windows bereitgestellten „Gruppenrichtlinien“. Ein Administrationswerkzeug, welches erlaubt Zugriffsrechte und Einstellungen zu beschränken, Wechseldatenträger zu deaktivieren, aber auch Programme über mehrere Clientcomputer zu verteilen.

Nun stellt sich die Frage, ob vergleichbare Optionen unter Qubes OS zu Verfügung stehen. Eine mögliche Option wäre das Erstellen eines Qubes OS Backups eines manuell eingerichteten Systems und der Verteilung dieses Backups über die restlichen Systeme. Ein damit verbundenes Problem ist, dass dom0 Inhalte in einen speziell hinterlegten Ordner speichert und Systemkonfigurationsdateien innerhalb von dom0 nicht gespeichert werden. Um dieses Problem zu umgehen, müssten diese Inhalte zuvor manuell in das dom0-Homeverzeichnis kopiert werden. Hierdurch kann eine hohe Komplexität entstehen durch verschiedene möglicherweise auftretende Probleme. Eine weitere Möglichkeit für identische Systeme (zumindest gleiche Festplatten), wäre das manuelle Einrichten eines Systems mit dem anschließenden Erstellen eines Speicherabbilds. Dieses Speicherabbild kann darauffolgend auf die anderen Computer geladen werden. Falls die ursprüngliche Installation verschlüsselt wurde, wären alle Zugangspasswörter über alle Systeme gleich. Die Vorteile dieser Installation überwiegen diesem Problem allerdings. Das anschließende Ändern der Passwörter ist ein geringeres Übel. Nun in dem Fall, dass es sich um verschiedene Computerhardware handelt, müssten nach der Installation von Qubes OS verschiedene „Salt Formula“ erstellt und einzeln ausgeführt werden. „Salt Formula“ sind im Voraus definierte Skripte, welche für die Installation und Konfiguration von Software verwendet werden können. Die Salt-Verwaltungssoftware ist mittlerweile standartgemäß mit einigen bereits konfigurierten Zuständen in Qubes integriert. Allerdings bringt Salt zu viele Eigenschaften mit sich, dass diese hier ins genauste beschrieben

werden könnten. Deshalb verweise ich auf die Dokumentation von Salt [16] und auf die eigene Dokumentation von Qubes OS über Salt. [17]

4.1.4 Installation einer Windows-VM

Um Windows native Anwendungen unter Qubes OS zu verwenden, bietet das Betriebssystem die Möglichkeit eine sogenannte *Windows-Qube* einzurichten. Nicht modifizierte Betriebssysteme, wie in diesem Fall Windows, können als HVM-Domäne (*Hardware-assisted Virtual Machine*) installiert werden. Diese ist eine vollständig virtualisierte oder hardwareunterstützte virtuelle Maschine, die Virtualisierungserweiterungen des Prozessors nutzt. HVMs sind aufgrund der erforderlichen Emulation in der Regel langsamer als paravirtualisierte virtuelle Maschine.

Bei der Installation muss allerdings unterschieden werden. Es bestehen vier Möglichkeiten der Windowsinstallation.[8]

- Einfache Windowsinstallation
- Import einer Windows VM
- Ein inoffizielles-Tool für die Installation
- Windows als Template VM

Die einfache Windowsinstallation unterstützt allerdings einige Features nicht. Es ist nicht möglich Text oder Dateien zwischen normalen Qubes und Windows VM zu kopieren und einzufügen. Ebenso ist es nicht möglich USB-Geräte zu zuweisen, wie in Abschnitt 4.2.1 beschrieben. Aus- und Eingabe von Ton und die Unterstützung von Treibern ist auch nicht gegeben. Es besteht allerdings die Möglichkeit native Windowsanwendungen zu Installierung und auszuführen. Dies ist der primäre Nutzen von Windows-Qubes und daher reicht die einfache Installation im Rahmen dieser Arbeit aus.

Installation

Zu Beginn muss eine ISO-Datei der gewünschten Windows Version auf einer separaten VM installiert werden. Anschließend kann wie gewohnt eine neue Qube erstellt werden. Hier muss eine Standalone-Qube nicht basierend auf einer Vorlage ausgewählt werden. Nach Erstellung muss der Systemspeicher und der Arbeitsspeicher erhöht werden.

- Systemspeicher: 30000+ MB

[8] https://github.com/Qubes-Community/Contents/blob/master/docs/os/windows/windows-vm.md

- Arbeitsspeicher: 4096+ MB

Anschließend kann innerhalb der Einstellung der Windows-Qube „*Boot from CDROM*“ ausgewählt werden. Hierbei öffnet sich ein weiteres Fenster, in dem die zuvor heruntergeladene ISO-Datei gewählt werden kann. Abschließend muss die herkömmliche Windowseinrichtung abschlossen werden. Hierfür wird ebenso eine Windowslizenz benötigt, um wirklich alle Funktionen von Windows nutzen zu können.

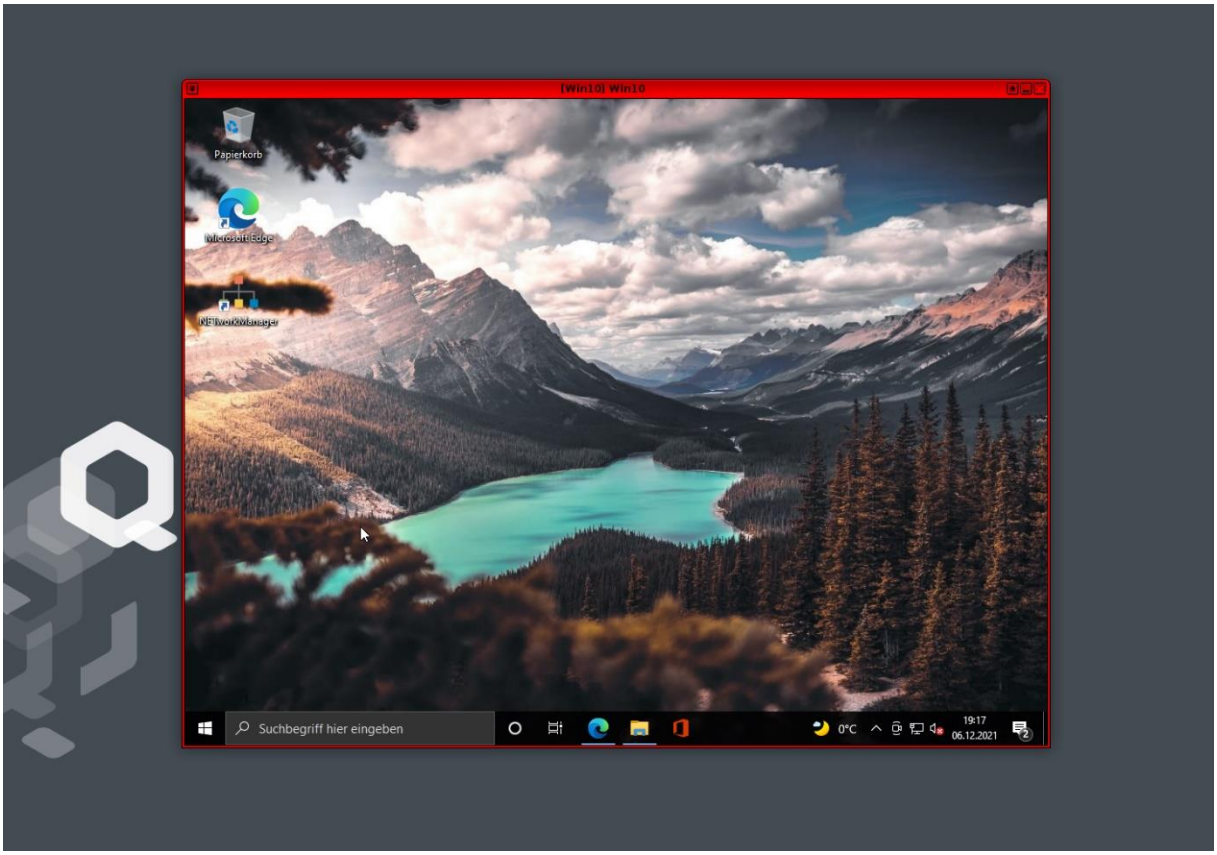

Abbildung 8: Eine Windows 10 VM innerhalb von Qubes OS.

In Abbildung 8 ist die erfolgreich installierte Windows VM zu sehen. Innerhalb dieser virtuellen Maschine können nur native Windows Anwendungen ohne Hardwarebeschleunigung ausgeführt werden.

4.2 Allgemeine Nutzungsszenarien

Nach erfolgreicher Einrichtung des Betriebssystems und einer kurzen Einführung wie Software automatisch verteilt und konfiguriert werden kann, wird nun genauer auf die allgemeinen Nutzungsszenarien eingegangen. Es werden Anwendungen für Bürotätigkeiten untersucht. Hierunter zählt Software, wie das Microsoft-Office-Paket. Darüber hinaus wird Kommunikationssoftware, für Konferenzen und E-Mailaustausch geprüft. In Kapitel 4.2.2 wird auf das Dateimanagement eingegangen und beschrieben, wie Daten zwischen verschiedenen VMs verschoben werden können.

4.2.1 Büroanwendungen

In diesem Abschnitt werden typische Büroanwendungen, welche relevant für Software-Entwicklungsunternehmen sind, untersucht. Ein möglich sehr populäres Beispiel hierfür könnte das Microsoft 365 Paket sein. Darunter fallen Programme wie Word, Excel, PowerPoint, Outlook und Teams. Diese Reihe an Programmen ist häufig sehr beliebt in Unternehmen. Unter Linux, dahingehend auch unter Qubes OS, gibt ebenso eine Zusammenstellung von Standartsoftware für Bürotätigkeiten, nämlich LibreOffice. Ein großer Vorteil vorweg ist, dass LibreOffice im Gegensatz zu Microsoft 365 vollkommen kostenlos ist. Zu LibreOffice gehören beispielsweise Programme für Textverarbeitung, Tabellenkalkulation und Präsentationen. Dementsprechend werden die allgemeinen Bedürfnisse eines Unternehmens für Büroanwendungen abgedeckt, denn *.docx* Dateien können ebenso frei mit LibreOffice geöffnet und bearbeitet werden. Also sollte es zu keinen Problemen kommen, falls mit Microsoft Office Benutzern kooperiert werden müsste. PowerPoint-Präsentationen funktionieren ebenso, denn auch sie können frei in LibreOffice erstellt oder geöffnet werden. Hier kann es allerdings zu Formatproblemen kommen. Falls eine Präsentation unter Microsoft PowerPoint gehalten werden muss, sollte eine in LibreOffice erstelle Präsentation im Vorfeld auf einem Windows-Computer kontrolliert werden.

In speziellen Fällen, falls dennoch der Gebrauch von Microsoft 365 Programmen nötig ist, gibt es Wege dies umzusetzen. Zum einen sind Office 365 Programme auch als Web-Version verfügbar. Also falls eine konstante Internetverbindung vorhanden ist, können alle Programme über den Webbrowser verwendet werden. Ebenso steht der Office365WebDesktop für Linux zu Verfügung. Dies ist eine Anwendung, welche webbasierte Inhalte innerhalb einer Desktopapplikation repräsentiert, ohne dass ein tatsächlicher Webbrowser nötig ist. Darüber

hinaus ist Microsoft Teams, über die in Kapitel 4.2 beschriebene Methode, als eigenständiges Programm unter Qubes OS installierbar. Des Weiteren bestehen zwei Möglichkeiten für die volle Installation von Microsoft 365 Programmen. Die erste und etwas simplere Methode ist die Installation von Windows 365 über das Hilfsprogramm CrossOver in einer Debian- oder Fedora Qube. Über CrossOver können verschiedene Windowsanwendungen unter Linux betrieben werden. CrossOver verspricht, dass Windowsanwendungen in nativer Geschwindigkeit laufen, im Gegensatz zu herkömmlichen Emulatoren.

Die zweite Methode ist das Installieren einer eigenständigen Windows-Qube. Das Problem hierbei liegt einerseits bei höheren Ressourcenkosten und andererseits bei hohem Aufwand eine vollkommende Windowsinstallation durchzuführen, bei der eventuell noch eine Windowslizenz erworben werden muss. Allerdings falls ein Benutzer weiteren Nutzen aus einer Windows-Qube beziehen kann als nur für Office 365 Produkte, kann eine solche Installation gerechtfertigt sein. Entsprechend erneut anzumerken ist, dass jegliche Software, welche lauffähig unter Linux ist, ebenso unter Qubes OS betrieben werden kann. Das gilt gleichermaßen für windowsexklusive Programme innerhalb von Windows basierenden Qubes.

Ein weiterer Bereich von Büroanwendungen sind Konferenzwerkzeuge. Hierunter fallen beispielweise *Microsoft Teams*, *Skype* oder *Big Blue Button*, wobei es noch duzende weitere gibt. Es wurde untersucht ob Mikrofon, integrierte Kamera und externe USB-Kamera innerhalb solcher Tools funktionieren. Allgemein kann gesagt werden, dass grundlegende Verbindung bzw. Teilnahme wie unter jeden anderen Betriebssystem abläuft. Kommt es nun zu USB-Geräten sieht das wiederum etwas anders aus. Im Normalfall kann ebenso angenommen werden, dass integrierte Mikrofone und Kameras in Laptops ebenso über USB angebunden sind. Der Anschluss eines USB-Gerätes an dom0 stellt ein Sicherheitsrisiko dar, da das Gerät beliebige USB-Treiber (die im Linux-Kernel enthalten sind) angreifen, Fehler beim Partitionstabellen-Parsing ausnutzen oder sich einfach als Tastatur ausgeben könnte. Um dieses Risiko zu vermeiden, werden USB-Qubes verwendet. [11]

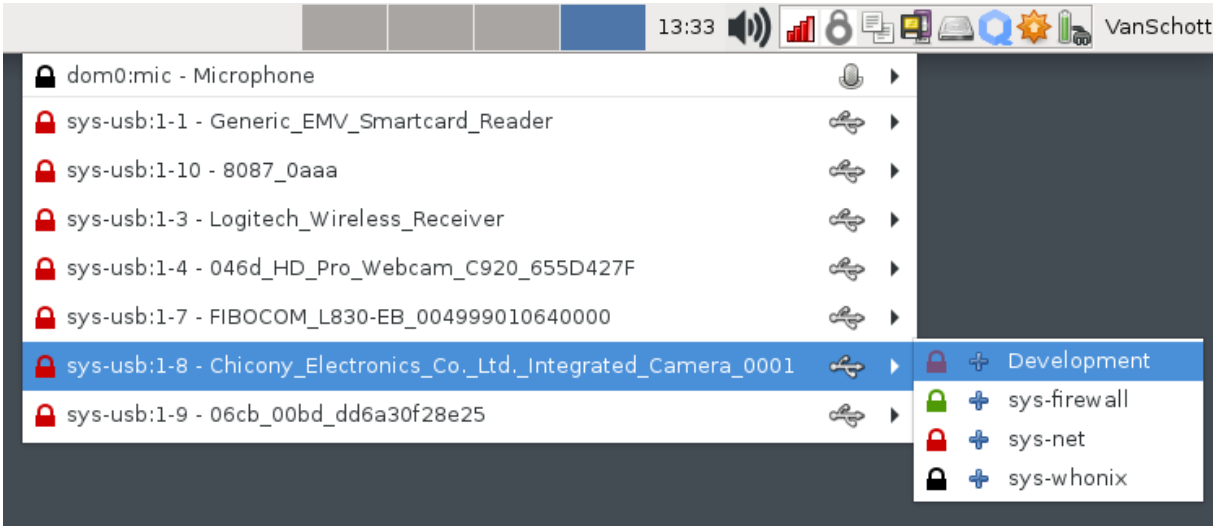

Abbildung 9: Qubes Device-Manger für die Verknüpfung von USB-Geräten mit VMs.

In der Abbildung 9 ist zu sehen, wie USB-Geräte über eine USB-Qube (*sys-usb* basierend auf einem Fedora-Template) mit anderen AppVMs oder StandaloneVMs verknüpft werden können. Jedes Gerät abgesehen von USB-Mäusen kann so simultan nur in einer VM verfügbar sein. Das Anschließen eines USB-Geräts an eine VM (USB-Passthrough) setzt die Ziel-Qube den meisten Sicherheitsproblemen im Zusammenhang mit dem USB-Stack aus. Aus diesem Grund sollten nicht vertrauenswürdige USB-Geräte zu Beginn an einer DisposibleVM getestet werden. Nach Verknüpfung des Mikrofons und der Kamera mit der entsprechenden Qube in der die Konferenzsoftware läuft, kann diese nun frei verwendet werden. Bei der Verwendung der Kamera des Lenovo ThinkPad p53s sind allerdings Probleme aufkommen, welche in Kapitel 5.2 unter Stolperfallen genauer beschrieben werden.

4.2.2 Dateimanagement

In diesem Unterkapitel wird auf das Dateimanagement unter Qubes OS eingegangen. Genauer gesagt werden die Grundlagen im allgemeinen, sowie fortgeschrittene Themen, wie das Verschieben von Dateien zwischen verschiedenen Virtuellen Maschinen, behandelt.

Zu Beginn muss klargestellt werden, dass dom0 eine Art Schnittstelle ist, die das gesamte System verwaltet. Teilweise im Hintergrund, teils durch den Benutzer selbst. Man könnte es auch als „Administrations-Qube“ bezeichnen. Wobei die tatsächliche Arbeit des Benutzers in den entsprechenden User-VMs (*AppVMs* und *StandaloneVMs*) getätigt wird.

Ein großer Teil der Systemadministration beinhaltet den Umgang mit Dateien und Verzeichnissen: Erstellen von Verzeichnissen, kopieren von Dateien, verschieben von Dateien oder Verzeichnissen und dem Löschen von Dateien. In Qubes OS haben AppVMs allerdings kein eigenes Root-Dateisystem. Dieses wird zwischen den AppVMs und dem Template aus welchem die AppVMs erzeugt wurden, geteilt. StandaloneVMs hingegen haben ein eigenes Root-Dateisystem. Das Root-Dateisystem unter Qubes OS basiert auf dem *Filesystem-Hierarchie-Standard* für Verzeichnisstruktur unter Unix-ähnlichen Betriebssystemen.

Verzeichnis	**Beschreibung**
/bin	Binärdateien grundlegender Befehle
/boot	Statische Dateien des Bootloaders
/dev	Gerätedateien
/etc	Host-spezifische Systemkonfiguration
/lib	Bibliotheken und Kernel-Module
/media	Einhängepunkt für Wechseldatenträger
/mnt	Für temporär eingehängtes Dateisystem
/opt	Zusätzliche Anwendungsprogramme
/run	Für laufende Prozesse
/sbin	Essenzielle Binärdateien des Systems
/srv	Dateien für Dienste
/tmp	Temporäre Dateien
/usr	Sekundäre Hierarchie
/var	Variable Dateien

Tabelle 1: 14 Hauptverzeichnisse unter der Filesystem-Hierarchie-Standard Richtlinie.

Verzeichnis	**Beschreibung**
/home	Benutzerverzeichnis für persönliche Dateien
/root	Verzeichnis des Root-Kontos
/lib…	Alternative Bibliotheken, beispielweise /lib32 und /lib64
/proc	Virtuelles Dateisystem, das Prozess- und Kernelinformationen bereitstellt
/sys	Enthält Informationen über Geräte-, Treiber- und Kernelfunktionen

Tabelle 2: Zusätzlich erforderliche Verzeichnisse durch Untersysteme von Qubes OS.

In Tabellen 1 sind die 14 Hauptverzeichnisse (mit einer entsprechenden Beschreibung in der rechten Spalte), welche immer automatisch angelegt werden unter Unix-ähnlichen Betriebssystemen. Alle zusätzlichen Verzeichnisse hängen einerseits von der entsprechenden Linux-Distribution ab und andererseits davon, ob Untersysteme des Root-Users existieren. In Tabelle 2 sieht man einige Beispiele von möglichen Zusatzverzeichnissen. In dem Fall von Qubes OS gibt es sogar verschiedene Arten von Untersystemen (Fedora oder Debian). Hier können sich die Verzeichnisse noch einmal leicht unterscheiden. Das */home* Verzeichnis ist besonders relevant für Qubes OS, da dies das Benutzerverzeichnis innerhalb von AppVMs ist. Hier findet der Benutzer automatisch angelegte Ordner wie: Desktop, Dokumente, Downloads, Musik, Bilder, Öffentlich, Templates und Videos. Hier kann der Benutzer wie gewohnt seine Dateien verwalten. Hierbei anzumerken ist, dass der automatisch allokierte Speicherplatz einer AppVM bei 2048MiB liegt. Da dieser schnell ausgeschöpft ist, kann er innerhalb der Qube-Settings über dom0 erhöht werden.

Das Verschieben und Kopieren von Dateien

Qubes OS unterstützt das sichere Verschieben und Kopieren von Dateien zwischen Qubes. Um Dateien zwischen VMs zu verschieben oder zu kopieren, muss der Dateimanager innerhalb der Qube in der sich die zu verschiebe Datei aufhält, geöffnet werden. Anschließend kann mit rechter Maustaste auf die Datei geklickt werden. Hier hat man zwei Möglichkeiten:

Copy to Other app qube...

Move to Other app qube...

Vergleiche Abbildung 10:

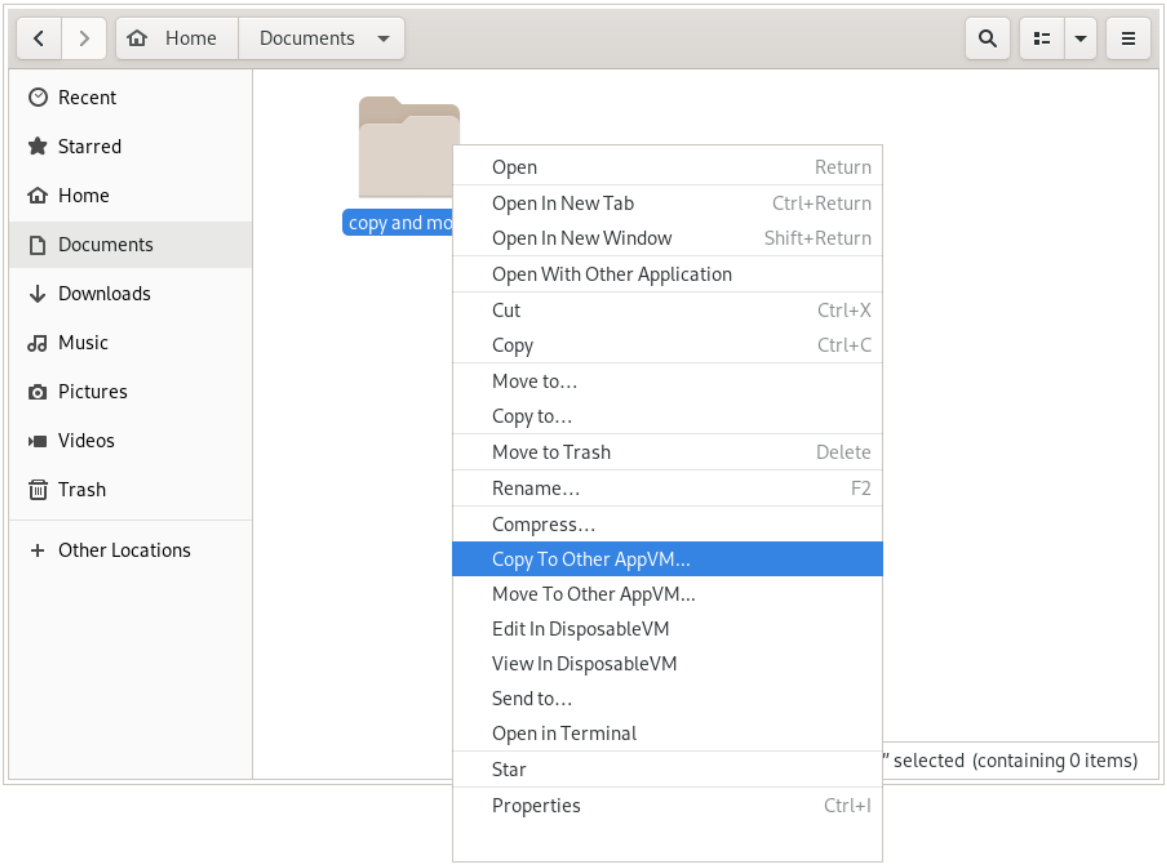

Abbildung 10: Verschieben und kopieren von Dateien.

In dom0 erscheint ein Dialogfeld, in dem der Name der Ziel-Qube abgefragt wird. Falls die Ziel-Qube noch nicht läuft, wird diese automatisch gestartet und die Datei wird dorthin kopiert. Sie erscheint in dem Verzeichnis:

/home/user/QubesIncoming/<source_qube>/<filename>

Die gleichen Vorgänge sind auch über diese Konsolenbefehle verfügbar:

```
$ qvm-copy [--without-progress] file [file]+
$ qvm-move [--without-progress] file [file]+
```

Das Kopieren von Dateien aus dom0

In dom0 (Admin-Qube) hat der Benutzer nicht die Möglichkeit Dateien über die in Abbildung 11 dargestellte Methode zu übertragen. Dateien in dom0 können allerdings über den folgenden Konsolenbefehl verschoben werden:

```
$ qvm-copy-to-vm <target_vm> <file>
```

Sicherheit

Das Dateikopiersystem zwischen Qubes ist sicher, da es anderen Qubes nicht erlaubt, die zu kopierenden Dateien zu stehlen. Ebenso wird der Quell-Qube nicht erlaubt beliebige Dateien der Ziel-Qube zu überschreiben. Außerdem verwendet dieses System keine Art von virtuellem Blockgeräten zum Kopieren von Dateien. Stattdessen wird der gemeinsam genutzte Xen-Speicher genutzt, wodurch ein Großteil der Verarbeitung von nicht vertrauenswürdigen Daten entfällt. Jedoch sollte bedacht werden, dass eine Datenübertragung von einer weniger vertrauenswürdigen Qube immer potenziell unsicher ist. Das liegt daran, dass die zu verschiebenden Daten einen Fehler in der Software, welche in der Ziel-Qube läuft, auszunutzen könnten. Daher sollten Daten immer nur von vertrauenswürdigen zu weniger vertrauenswürdigen Qubes kopiert werden. [18]

4.3 Nutzungsszenarien für Entwicklungsprozesse

In diesem Abschnitt wird Qubes OS explizit für Entwicklungszwecke getestet. Hier werden Entwicklungsprozesse mit gängigen Entwicklungs- und Testwerkzeugen umfasst. Es werden git-Repositories von gängigen Softwareprojekten herangezogen, da eine Container-basierte Entwicklungsweise für Cloud-basierte Dienste den Stand der Technik bildet.

4.3.1 Kommandozeilenbasiertes Server-Backend

Hier wird untersucht, ob kommandozeilenbasierte Server-Backends (Node.js/Java/.NET Core) problemlos unter Qubes OS kompilier- und ausführfähig sind. Im folgenden Teil wird ebenso erläutert wie Node.js, Java und .Net Core Projekte unter Qubes OS installiert und ausgeführt werden können.

4.3.1.1 Node.js mit Express

Zu Beginn wird ein Node.js mit Express REST-API getestet.[9] Hierzu wurde das Repository mit *git* lokal geklont. Git ist unter Qubes OS bereits vorinstalliert und kann über das Terminal

[9] GitHub Repository: https://github.com/rwieruch/node-express-server-rest-api

verwendet werden. Anschließend können in dem Verzeichnis, in dem das Projekt liegt, die folgenden Befehle ausgeführt werden:

```
$ npm install
```

```
$ npm start
```

Im Anschluss war das Projekt lauffähig und die REST-API konnte unter *http://localhost:8000* über die Konsole oder einem externen Programm wie *Postman* angesprochen werden.

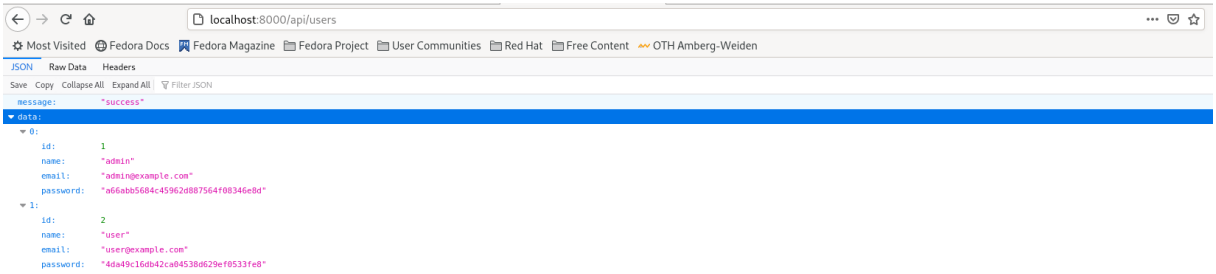

Abbildung 11: Node.js REST-API Ausgabe für die GET Route /api/users

In Abbildung 11 ist eine erfolgreiche GET Response der Node.js REST-API zu sehen. Da Node.js allgemeingesprochen sehr kompatibel mit Linux ist, gibt es unter Qubes OS ebenso keinerlei Probleme.

4.3.1.2 Java

Als zweites wird ein Java-Backend für die Generierung eines JWT-Tokens untersucht.[10] Voraussetzungen hierfür ist ein Java Development Kit (JDK) 8+ in Kombination mit der Community Version von Intellij. Die Intellij IDEA ist einfach über den Snap-Paketmanager aus Kapitel 4.1.2 installierbar. Ebenso wird Maven Version 3 oder höher benötigt. Nach der Ausführung von:

```
$ mvn -version
```

[10] GitHub Repository: https://github.com/VirgilSecurity/demo-backend-java

erscheint ein Vorschlag in der Konsole Maven über die Kommandozeile zu installieren. Folgend müssen die Umgebungvariablen im Verzeichnis */etc/environment* hinzugefügt werden. Nach erfolgreicher Installation werden die nötigen Informationen hierfür in der Konsole angezeigt. Das Hinzufügen geht über den Vim-Editor innerhalb der Kommandozeile. Das Repository selbst kann wieder über git geklont werden. Im Anschluss kann ein *Build* über den Maven Befehl:

```
$ mvn clean package -DskipTests
```

erzeugt werden. Der Server kann nun über den folgenden Befehl gestartet werden:

```
$ java -jar server.jar
```

Der Server ist nun lauffähig und kann über *http://localhost:3000* getestet werden. Die Serverlauffähigkeit kann über folgenden cURL-Befehl verifiziert werden:

```
$ curl -X POST -H "Content-Type: application/json" \-d
'{"identity":"my_identity"}'
\http://localhost:3000/authenticate
```

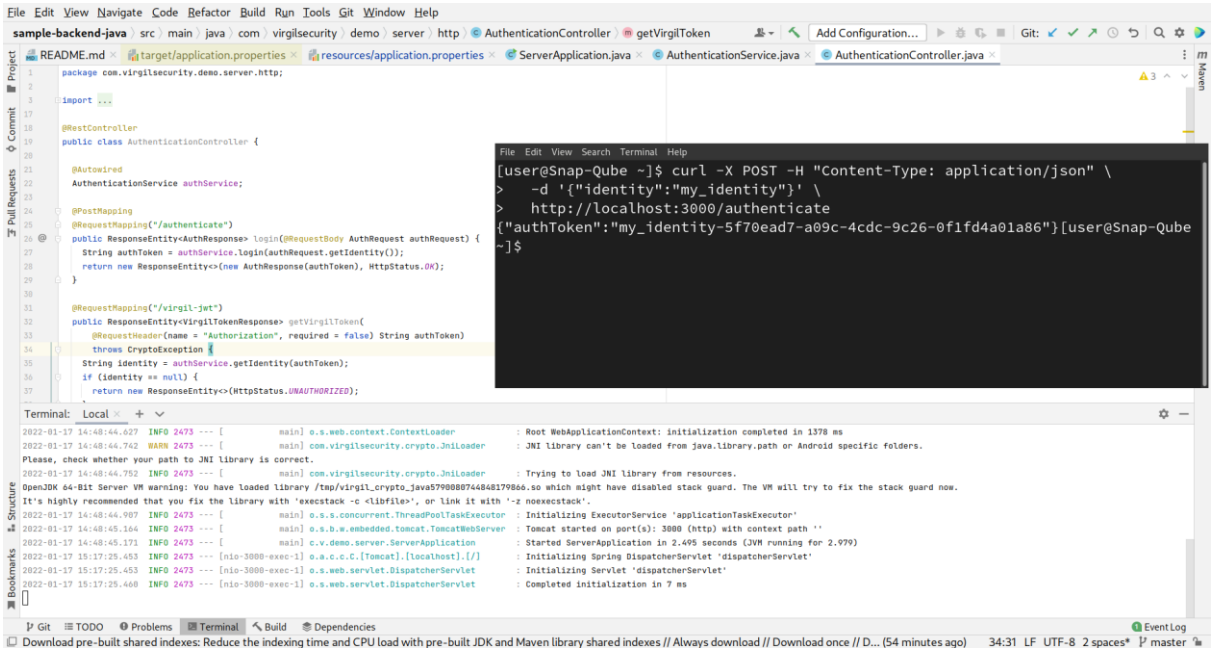

Abbildung 12: Java-Backend innerhalb der Intellij Community IDEA unter Qubes OS

In Abbildung 12 ist die Intellij IDEA mit lokal laufendem Server zu sehen. Oben rechts ist ein Terminalfenster mit zuvor genannten cURL-Befehl dargestellt. Dieser cURL-Befehl antwortet erfolgreich mit entsprechendem Authentifizierungs-Token.

4.3.1.3 ASP .NET Core

Als drittes wird ein ASP.NET Core Applikation eines E-Commerce Webshops getestet.[11] Dieses Projekt kann erneut mit git geklont. In diesem Fall hat das Projekt verschiedene Möglichkeiten ausgeführt zu werden. Um alleinig die Lauffähigkeit zu testen, wurde die lokale Methode verwendet. Allerdings muss vor der Ausführung die .NET-SDK installiert werden. Die benötigte Version war .NET 6 und diese ist zu dem Zeitpunkt der Erstellung dieser Arbeit nicht über die Standardpaketrepositorys verfügbar. Eine Methode, um die Version 6 zu installieren ist erneut der Paketmanager Snap. Die Befehle hierfür sind:

```
$ sudo snap install dotnet-sdk --classic --channel=6.0
```

Mit dem Parameter *–channel* kann die zu installierende Version angegeben werden. Bei Weglassen dieses Parameters wird die aktuell stabilste Version installiert. Anschließend musste der Befehl *dotnet* für das System registriert werden:

```
$ sudo snap alias dotnet-sdk.dotnet dotnet60
```

Der angegebene Alias *dotnet60* kann beliebig gewählt werden. In diesem Fall wurde der Befehl nach der spezifischen Version benannt, um von möglicherweise anderen installierten .NET Versionen unterscheiden zu können. Nach erfolgreicher Installation der SDK kann ein Terminal innerhalb des *PublicApi-Ordners* geöffnet werden und der folgende Befehl kann ausgeführt werden:

```
$ dotnet60 run
```

Im Anschluss muss aus dem *Web-Ordner* ein zweiter Befehl ausgeführt werden:

```
$ dotnet60 run --launch-profile Web
```

Nun ist das Projekt lokal unter *http://localhost:5001/* erreichbar.

[11] GitHub Repository: https://github.com/dotnet-architecture/eShopOnWeb

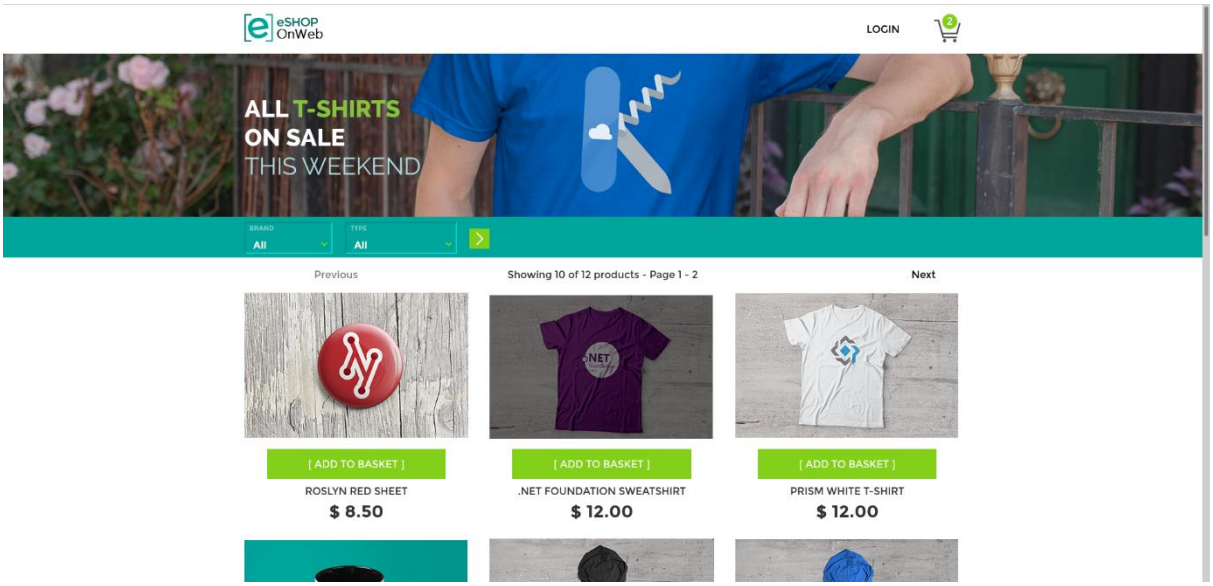

Abbildung 13: Ein ASP.NET Core E-Commerce Shop lokal kompiliert unter Qubes OS

In Abbildung 13 ist das erfolgreich kompilierte ASP .NET Core Projekt zu erkennen. Es ist zu sehen, dass dieses Projekt kein reines kommandozeilenbasiertes Server-Backend ist, sondern ebenso eine bereits integrierte Weboberfläche beinhaltet. Nichtsdestotrotz wurde erfolgreich getestet, dass ASP .NET Core Applikationen unter Qubes OS lauffähig sind.

Allgemeingesprochen konnte mit diesen Tests festgestellt werden, dass die Entwicklung von Server-Backends unter Qubes OS funktioniert. Aufgrund des soliden Linux-Fundamentes kann davon ausgegangen werden, dass kommandozeilenbasierte Funktionalitäten kompatibel sind. Ein Problem, das Qubes OS mit sich bringt, ist die nicht unterstützte Hardwarebeschleunigung. Dies hat hauptsächlich Sicherheitshintergründe, da eine Implementierung solcher Features eine große Komplexität im Bereich von GUI-Virtualisierung mit sich bringt. [19] App-Qubes verwenden eine software-basierte Implementation von OpenGL, welche für einfache Anwendungen ausreichen soll. Um dies genauer zu testen, wird in Abschnitt 4.3.2 die Entwicklung von Web-, 2D- und 3D-fähigen Oberflächen untersucht.

4.3.2 Anwendungen mit Web-, 2D- und 3D-fähige Oberflächen

Wie schon zu vor angemerkt ist die aktuell große Schwachstelle von Qubes OS, dass keine Hardwarebeschleunigung unterstützt wird. In diesem Kapitel soll untersucht werden, ob dies die Entwicklung von Web-, 2D- (*SDL2*, *DirectDraw*, *Direct2d*, *DirectWrite*) oder 3D (*Direct3D*, *OpenGL)* -fähigen Oberflächen beeinträchtigt. Zu gängigen Anwendungen, welche

Hardwarebeschleunigung nutzen, gehören Webbrowser wie Chrome und Firefox, Videobearbeitungsprogramme sowie Videospiele. Vorweg um Missverständnisse zu vermeiden, sind keine Probleme bei der Nutzung von Webbrowsern aufgekommen. Diese sind wie von anderen Betriebssystemen gewohnt auch unter Qubes OS nutzbar. Folgend wird untersucht, ob die Entwicklung von Web- und graphischen Oberflächen unterstützt wird.

4.3.2.1 Weboberflächen

In diesem Unterkapitel wird untersucht, ob Qubes OS die Entwicklung von frontendbasierten Weboberflächen unterstützt. Als erstes Repository wird ein Dashboard für Kryptowährungen, das auf dem Framework *Vue.js* basiert, getestet.[12] Nach dem Klonen des Projektes innerhalb der gewünschten Qube, kann das Projekt problemlos mit zwei Konsolenbefehle ausgeführt werden:

```
$ npm install
$ npm run serve
```

Anschließend kann das Dashboard unter *http://localhost:8080* getestet werden. Für Anpassungen des Codes wurde der Quelltext-Editor *Visual Studio Code* verwendet. In der Abbildung 14 ist das erfolgreich lokal ausgeführte Vue.js Projekt zu sehen. Die Anwendung war vollständig und ohne Einschränkungen nutzbar.

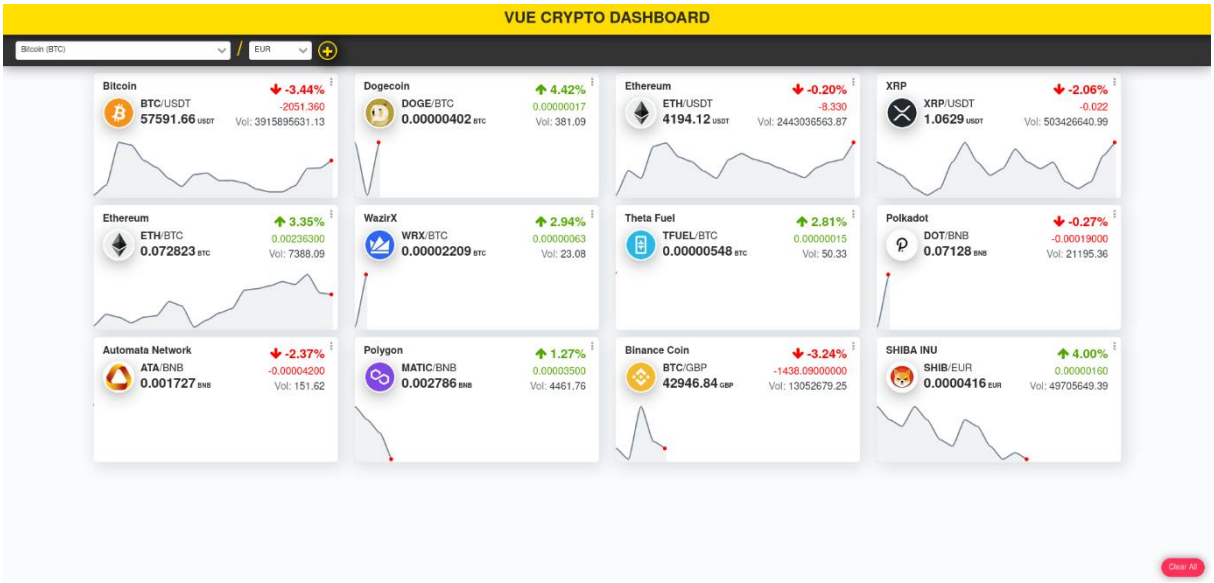

Abbildung 14: Ein Dashboard für Kryptowährungen entwickelt mit Vue.js.

[12] GitHub Repository: https://github.com/JayeshLab/vue-crypto-dashboard

Als zweites wird ein Prototypen-Werkzeug für die Generierung von verbundenen Klassenkomponenten in React.js untersucht[13], um die Verträglichkeit von verschiedenen Frontend-Frameworks unter Qubes OS zu vergleichen. Der Ablauf dieses Projektes ist ähnlich zu dem Vorherigen. Erneut muss das Projekt geklont werden. Anschließend müssen die Abhängigkeiten installiert werden:

```
$ npm install
```

Um das Web-Paket „build“ zu generieren:

```
$ npm run dev
```

In einem separaten Terminal:

```
$ npm start
```

Hier kam es allerdings zu Qubes OS unabhängigen Problemen kommen. In diesem Fall wurde das Node-Modul für *electron* inkorrekt installiert. Um dies zu beheben, muss das Paket manuell gelöscht und die Installation muss erneut ausgeführt werden. Ebenso wurde ein Fehler geworfen, da einige Module veraltet waren. Mit dem npm-Paketmanager können mit dem folgenden Befehl alle Module auf einmal aktualisiert werden:

```
$ npm update
```

Diese Fehler sind Qubes OS unabhängig und könnten in dieser Form auf jedem Betriebssystem entstehen. Nach der Ausführung der oben genannten Schritte startet das Programm. In Abbildung 15 sieht man die erfolgreich gestartete React-App innerhalb einer Qubes OS Umgebung. Die Anwendung war ohne Einschränkungen nutzbar.

[13] GitHub Repository: https://github.com/ReactPrimer/ReactPrimer

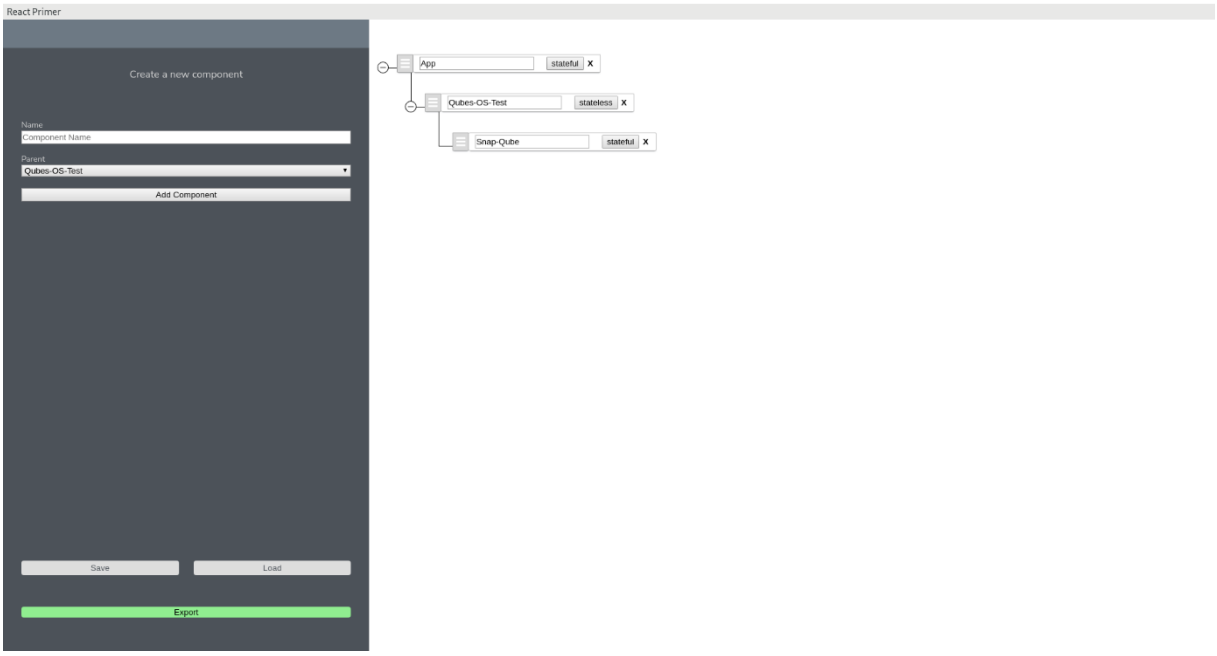

Abbildung 15: Die React-App *ReactPrimer* unter Qubes OS.

Durch diese Tests ist zu sehen, dass die Entwicklung von Webbasierten Anwendungen unter Qubes OS möglich ist. Vor allem Projekte auf Basis der Paketmager *npm* oder *yarn* laufen von dem Standpunkt des Betriebssystems uneingeschränkt.

4.3.2.2 2D-fähige Oberflächen

In diesem Abschnitt wird untersucht, ob die Entwicklung von 2D-fähigen Oberflächen durch die fehlende Hardwarebeschleunigung von Qubes OS begrenzt wird. Grafische Schnittstellen für 2D-fähigen Oberflächen könnten beispielweise *SDL2*, *DirectDraw*, *Direct2D* oder *DirectWrite* sein. Es muss allerdings angemerkt werden, dass *DirectDraw*, *Direct2D* und *DirectWrite* rein für Windows implementierte API-Schnittstellen sind. Aufgrund dessen können diese nur innerhalb einer Windows-Qube ausgeführt werden. Darüber hinaus verwenden diese Grafik-APIs Hardwarebeschleunigung. Hierunter fällt einerseits *Direct2D*, welches diese in allen Fällen verwendet. *DirectWrite* verwenden Hardwarebeschleunigung nur in gemeinsamen Einsatz mit *Direct2D* und *DirectDraw* greift auf Hardwarebeschleunigung zurück, falls diese auf dem entsprechenden Computer zur Verfügung steht. Mit Berücksichtigung dieser Fakten im Zusammenspiel mit der eingeschränkten Leistung der *hardwareunterstützten Windows VM* (englisch *hardware-assisted Virtual Machine*) kann ausgesagt werden, dass Qubes OS nur eingeschränkt geeignet für diese APIs ist. Innerhalb von

ausgewählten Umständen, also ohne Nutzen von Hardwarebeschleunigung und einer leistungsarmen Anwendung, kann ein *DirectWrite*/*DirectDraw* Projekt ausgeführt werden. Eine professionelle Entwicklung mit *DirectDraw*/*Direct2D*/*DirectWrite* unter Qubes OS ist hingegen nicht möglich.

Das Gegenstück für eine plattformübergreifende Entwicklung von Computergrafikanwendungen ist OpenGL und wird in Kapitel 4.3.2.3 untersucht. Zuvor wird die unter Linux (auch Windows) unterstützte *SDL2* API getestet. *SDL2* (englisch *Simple DirectMedia Layer*) ist eine Multimediabibliothek, welche auch für Linux zur Verfügung steht. Die Bibliothek stellt eine Schnittstelle für Grafik-, Sound- und Eingabegeräte bereit und dient hauptsächlich zur Entwicklung von Spielen und Multimediaanwendungen. Als Testprojekt wird ein rasterbasiertes 2D-Spiel auf Basis von *SDL2* verwendet.[14] Nach dem Klonen des Projektes müssen nun einige Abhängigkeiten installiert werden. Unter Linux ist dies mit einem Befehl möglich:

```
$ sudo apt install gut build-essential pkg-config cmake cmake-
data libsdl2-dev libsdl2-gfx-dev
```

Es ist zu empfehlen dies innerhalb einer *Standalone* VM auszuführen, da diese ein eigenständiges Root-Dateisystem besitzen. Dadurch stehen diese Abhängigkeiten auch nach einem Neustart zur Verfügung. Eine *App* VM wird beim Starten aus einer entsprechenden Vorlage erstellt und behält die installierten Abhängigkeiten nicht nach einem Neustart.

Um diese Anwendung zu testen kann die frei zur Verfügung stehende Entwicklungsumgebung *Code::Blocks* verwendet werden. Für die Installation kann erneut der Paketmanager *snap* verwendet werden:

```
$ sudo snap install code –classic
```

Anschließend kann das Projekt innerhalb der Entwicklungsumgebung oder über die Konsole mit folgenden Befehlen erstellt werden:

```
$ cmake ..

$ make
```

[14] https://github.com/aminosbh/falling-brick-game

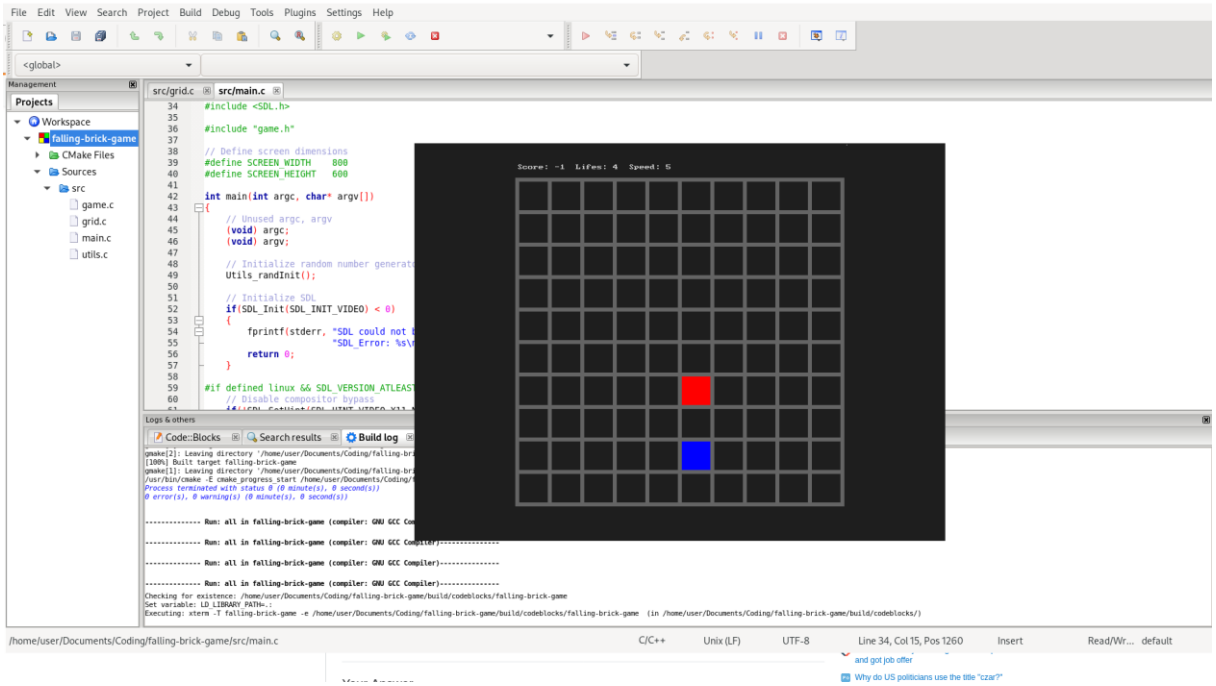

Abbildung 16: Ein rasterbasiertes 2D-Spiel auf Basis von SDL2.

In Abbildung 16 ist das erfolgreich über die Entwicklungsumgebung kompilierte Spiel zu sehen. Somit ist es möglich *SDL2* Projekte unter Qubes OS zu entwickeln und auszuführen. Keinerlei Einschränkungen sind im Zusammenhang mit Qubes OS und den in diesem Projekt verwendeten Technologien aufgefallen.

4.3.2.3 3D-fähige Oberflächen

Für die Entwicklung von 3D-fähigen Computergrafikanwendungen werden zwei verschiedene Programmierschnittstellen untersucht. Einerseits die von Microsoft entwickelte API *Direct3D*, welche nativ nur unter Windows einsetzbar ist. Andererseits die plattform- und programmiersprachenübergreifende Schnittstelle *OpenGL*.

Direct3D bringt ähnliche Probleme mit, wie die in Kapitel 4.3.2.3 angesprochenen Schnittstellen *DirectDraw*/*Direct2D*/*DirectWrite*. Es ist ebenso nur nativ innerhalb einer Windows-Umgebung einsetzbar. Somit kann Direct3D nur innerhalb einer Windows-Qube verwendet werden. Dies bringt die schon zuvor erwähnten Leistungseinschränkungen mit sich. Des Weiteren verwendet *Direct3D* Hardwarebeschleunigung, falls diese zur Verfügung steht. Somit ist es unter Qubes OS möglich Anwendungen auf Basis von *Direct3D*, welche nicht auf

Hardwarebeschleunigung angewiesen sind, zu entwickeln. Allerdings sind moderne Spiele mit aufwendigen Grafiken immer an Hardwarebeschleunigung gebunden.

Unter Berücksichtigung dieser Fakten kann ausgesagt werden, dass eine theoretische Entwicklung möglich ist. Jedoch ist Qubes OS nicht geeignet für eine professionelle Entwicklung von *Direct3D* Anwendungen.

Die Programmierschnittstelle OpenGL ist plattformunabhängig und kann unter Linux, somit auch unter Qubes OS verwendet werden. Nun soll untersucht werden, ob *OpenGL*-Anwendungen unter Qubes OS, trotz fehlender Hardwarebeschleunigung, ausführbar sind. Für diesen Test wurde ein Klon des populären Spiels *Minecraft* verwendet.[15] Zu Beginn kann das Projekt über *git* geklont werden. Bevor die Anwendung kompiliert werden kann, müssen einige Abhängigkeiten installiert werden. Die Installationsbeschreibung innerhalb des *Repositorys* ist ausgelegt für Ubuntu und benötigt nur zwei Konsolenbefehle. Für diesen Test wurde allerdings die standartmäßig unter Qubes OS installierte Fedora-Distribution verwendet. Die für *Fedora* benötigten Befehle können über den *yum*-Paketmanager installiert werden:

```
$ sudo yum -y install lubcurl-devel xorg-x11-apps libXinerama-
devel libXcursor-devel
```

Ebenso wird das Programmierwerkzeug *cmake* benötigt:

```
$ sudo dnf install cmake
```

Anschließend können in dem Verzeichnis des Projektes die folgenden Befehle ausgeführt werden:

```
$ cmake .
```

```
$ make
```

Wenn die Kompilierung erfolgreich war, kann das Programm über einen Befehl gestartet werden:

```
$ ./craft
```

In Abbildung 17 ist die erfolgreich ausgeführte OpenGL Anwendung zu sehen. Die Anwendung ist frei von Fehlern ausführbar. Allerdings läuft das Spiel durch die fehlende Hardwarebeschleunigung nur mit ungefähr sieben Bildern pro Sekunde und ist somit nicht tatsächlich spielbar.

[15] https://github.com/fogleman/Craft

Abbildung 17: Ein Minecraft-Klon basierend auf OpenGL.

Das Ziel dieses Tests war die Lauffähigkeit von OpenGL-Anwendungen zu untersuchen. Es kann ausgesagt werden, dass eine Entwicklung von OpenGL-Anwendungen unter Qubes OS möglich ist. Der signifikante Unterschied zu der Entwicklung innerhalb einer Windows Umgebung mit *WSL2* ist die fehlende Möglichkeit einer validen Leistungsprüfung der zu entwickelnden Anwendung. Darauffolgend müsste für eine professionelle Entwicklung von OpenGL-Anwendungen unter Qubes OS dennoch ein Windows-System für nachträgliche Leistungstests zur Verfügung stehen.

4.3.3 Docker unter Qubes OS

Da eine Container-gestützte Entwicklungsweise für Cloud-basierte Dienste den Stand der Technik bildet und die Laufähigkeit von Docker, trotz der Linux-Grundlage, nicht garantiert ist, soll nun untersucht werden, ob und wie Docker unter Qubes OS betrieben werden kann. Bevor die tatsächliche Lauffähigkeit getestet wird, muss untersucht werden, wie Docker unter Qubes OS installiert werden kann. Für den folgenden Installationsablauf wurde die Debian-Distribution gewählt. Eine Installation innerhalb einer Fedora-Distribution ist natürlich trotzdem möglich. Diese Beschreibung ist angelehnt an einer Installationsanleitung von Github[16]. Anzumerken ist, dass dies nicht der einzige Weg ist Docker unter Qubes OS zu

[16] https://gist.github.com/xahare/6b47526354a92f290aecd17e12108353

installieren. Mit der folgenden Beschreibung wird Docker innerhalb einer *Template* VM installiert und ist darauffolgend in AppVMs verwendbar. Es ist auch möglich eine standardgemäße Installation[17] innerhalb einer *Standalone* VM durchzuführen.

Zu Beginn muss eine neue *Template* VM erstellt werden. Hierfür kann die standartmäßig erstellte Debian *Template* VM dupliziert werden. Anschließend müssen *apt*-Pakete installiert werden:

```
$ sudo apt-get install apt-transport-https ca-certificates curl
gnupg2 software-properties-common
```

Vor dem nächsten Schritt muss innerhalb der Qubes-Einstellungen der Internetzugriff temporär aktiviert werden. Nun kann der digitale Fingerabdruck von Docker über einen *curl*-Befehl abgeglichen und hinzugefügt werden:

```
$ curl -fsSL https://download.docker.com/linux/$(. /etc/os-
release; echo "$ID")/gpg | sudo apt-key add -

$ sudo apt-key fingerprint <fingerprint>
```

Der Platzhalter für den Fingerabdruck muss mit dem offiziellen Fingerabdruck ersetzt werden. In der Konsole wir der Fingerabdruck angezeigt und dieser muss entsprechend abgeglichen werden. Falls die Fingerabdrücke übereinstimmen, kann Docker installiert werden. Mit dem folgenden Befehl wird das Repository angerichtet:

```
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/$(. /etc/os-release; echo
"$ID") $(lsb_release -cs) stable"
```

Anschließend kann das Paketverwaltungssystem von *Debian* aktualisiert und *Docker* installiert werden:

```
$ sudo apt-get update

$ sudo apt-get -y install docker-ce
```

Im nächsten Schritt wird *Docker* aktiviert, indem ein Benutzer gesetzt wird:

```
$ sudo groupadd docker

$ sudo usermod -aG docker user
```

[17] https://docs.docker.com/engine/install/ubuntu/

```
$ sudo systemctl enable docker
```

Abschließend muss innerhalb der Template VM eingerichtet werden, dass Änderungen auch innerhalb von *AppVMs* erhalten bleiben, durch das fehlende eigene Root-Filesystem:

```
$ sudo mkdir -p /rw/config/qubes-bind-dirs.d
$ sudo cat << EOF > /rw/config/qubes-bind-dirs.d/50_user.conf
$ binds+=( '/var/lib/docker' )
$ binds+=( '/etc/docker' )
$ EOF
```

Hier entsteht gegebenenfalls ein Rechteproblem bezüglich des *Docker-daemon-sockets*, dies kann über folgenden Befehl behoben werden:

```
$ sudo chmod 666 /var/run/docker.sock
```

Dieser Befehl erteilt volle Lese- und Schreiberechte für diese Datei. Nach Abschluss dieses Schrittes, kann die Template VM heruntergefahren werden. Nun muss eine *App* VM auf Basis der *Template* VM erstellt werden. Um die Funktionsfähigkeit von Docker zu testen, wird das offizielle „hello-world“ *Image* verwendet:

```
$ docker run hello-world
```

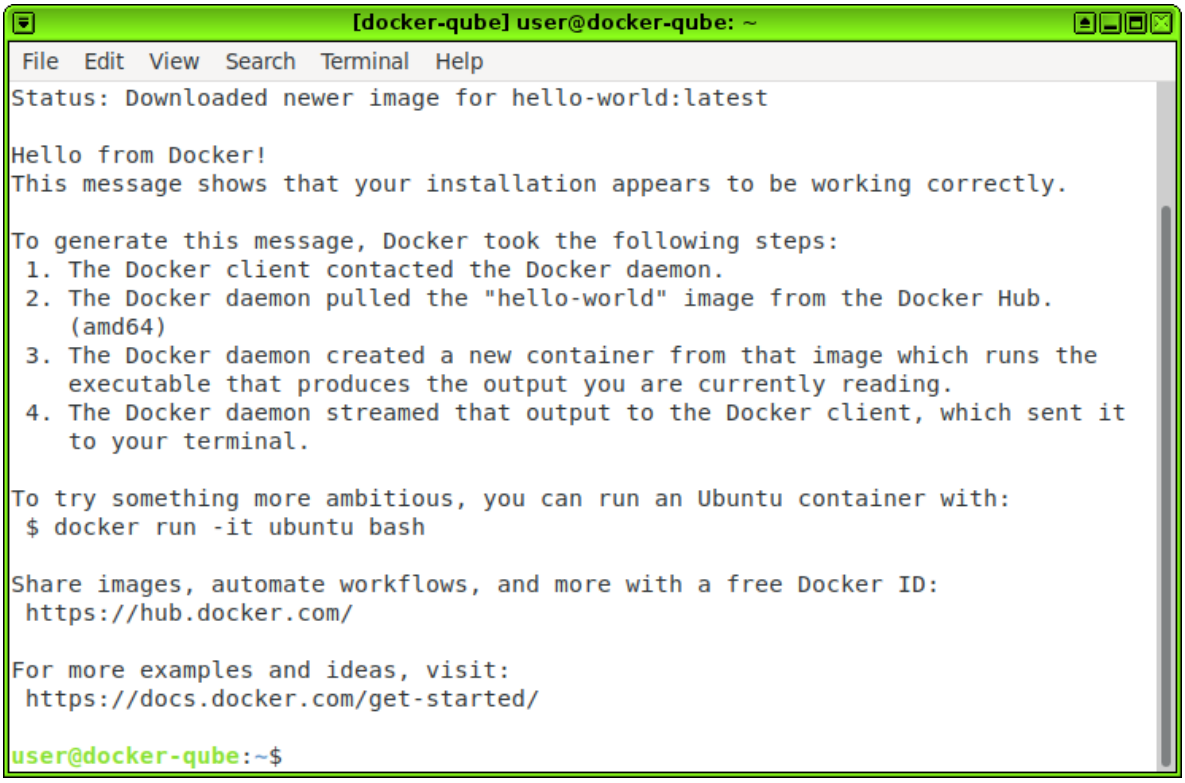

Abbildung 18: Die Konsolenausgabe des korrekt ausgeführten „hello-world“ Images.

In Abbildung 18 ist zu sehen, dass Docker erfolgreich ausgeführt werden konnte. Das einzige Problem, waren die fehlenden Rechte bezüglich des *Docker-daemon-sockets*. Nach der entsprechenden Rechtevergabe wurde kein Fehler mehr geworfen und Docker konnte ausgeführt werden.

Einschränkungen bezüglich der Nutzung sind keine aufgefallen. Nach Abschluss der Installation war Docker in vollem Umfang verwendbar. Anzumerken ist nur, dass die oben aufgeführte Installation eine gewisse Komplexität mitbringt. Eine vergleichbare Installation innerhalb einer *Standalone* VM oder unter Windows mit WLS2 ist einfacher.

4.3.4 Zusammenfassung

Dieses Kapitel hat ausgewählte Nutzungsszenarien von klein- und mittelständischen Software-Entwicklungsunternehmen dargestellt. Unter anderem wurde erläutert, wie IT-Administratoren Anwendungen automatisch verteilen und konfigurieren können. Hierbei wurde auch dargestellt, wie Benutzer selbstständig Anwendungen installieren können. Des Weiteren wurde erfolgreich getestet, dass Windows innerhalb einer Qubes OS Umgebung installiert und verwendet werden kann. Hierfür wurde eine Schritt-für-Schritt-Beschreibung erstellt.

Anschließend wurden Einsatzszenarien verschiedenster Technologien für Softwareingenieure untersucht. Darunter fielen allgemeine Nutzungsszenarien mit gängigen Büroanwendungen und einem Einblick in das Dateimanagement von Qubes OS. Genauer betrachtet wurden hier *Office365* Produkte und Konferenzwerkzeuge. Durch ein weites Spektrum an Optionen deckt Qubes OS alle Bedürfnisse eines klein- und mittelständischen Unternehmens im Bereich von Büroanwendungen ab. Einblicke in das Dateimanagement haben ebenso gezeigt, dass Qubes OS auf dem Betriebssystem Linux aufbaut. Starke Ähnlichkeiten im Aufbau des Dateisystems konnten festgestellt werden. Darüber hinaus wurde erläutert, wie Dateien zwischen verschiedenen virtuellen Maschinen verschoben werden können.

Abschließend wurde Qubes OS im Zusammenhang mit Technologien für Entwicklungen untersucht. Die Lauffähigkeiten wurden nicht nur bestätigt bzw. widerlegt, sondern es wurde explizit beschrieben, wie die Anwendungen kompiliert und ausgeführt werden können. Leider kam die fehlende Hardwarebeschleunigung im Bezug zu 2D- und 3D-fähigen Oberflächen zum Vorschein. Zum Ende dieses Kapitels wurde ein Augenmerk auf Docker geworfen. Es wurde erfolgreich getestet und beschrieben wie Docker innerhalb von *AppVMs* eingesetzt werden kann.

5 Kosten- und Nutzenabwägung zu Qubes OS

In dem vorherigen Kapitel wurden verschiedene für Entwickler relevante Szenarien und Technologien untersucht. Es wurde im Detail erläutert, wie entsprechende Untersuchungen durchgeführt wurden. In diesen sind unteranderem schon Probleme bzw. Stolpersteine aufgefallen. Das Ziel dieser Arbeit ist die Evaluation von Qubes OS im Bezug zu klein- und mittelständischen Software-Entwicklungsunternehmen. Aufgrund dessen sollen in den folgenden Abschnitten genauer auf die Grenzen und Stolpersteine von Qubes OS eingegangen werden. Die Ergebnisse werden mit einer Windows 10 mit *WSL2* Umgebung verglichen und bewertet. Das Ziel ist es klare Unterschiede im Lebenszyklus eines Softwareingenieurs darzustellen.

5.1 Ergebnisse der Evaluation

Dieser Abschnitt dient zur Zusammenführung der in Kapitel 4 durchgeführten Ergebnisse. Es wurden eine Reihe von relevanten Nutzungsszenarien für Software-Entwicklungsunternehmen analysiert. Für eine bessere Übersicht sind die Ergebnisse erneut in die drei Oberkategorien aus Kapitel 4 unterteilt. Zentrale Erwartungen an Qubes OS werden definiert und beantwortet.

Administrative Nutzungsszenarien

Installation von Qubes OS

> *Die Installation des Betriebssystems selbst ist eine der relevantesten Faktoren für Unternehmen. Es soll einfach möglich sein das Betriebssystem über mehrere Rechner zu installieren.*
>
> Auswertung: Qubes OS besitzt sehr spezifische Systemanforderungen und schränkt hiermit eine Nutzung direkt ein. Windows bzw. Linux Betriebssysteme haben im Gegensatz geringe Anforderungen und können somit auf fast jeder Hardware eingerichtet werden. Dennoch kann Qubes OS auf einer weiten Breite von Hardware installiert werden. Die Mindestvoraussetzung von Arbeitsspeicher sind 6 GB. Allerdings werden vor allem für den Entwicklungsnutzen 16 GB empfohlen. Diese Voraussetzungen sind unter Linux/Windows deutlich geringer. Der Installationsablauf innerhalb des Installationsprogrammes von Qubes OS ist sehr intuitive. Der Ablauf des Computerstartes über das BIOS ist gegebenenfalls etwas ungewohnt. Vor allem durch

das möglicherweise erforderliche Probieren, bis das korrekte Medium gefunden wurde. Das Einrichten des Installationsmediums (z.B. USB) ist über Linux, sowie Windows sehr einfach.

Installation von Programmen

Das Installieren von Programmen durch einen Power-User, sowie das automatische Verteilen von Anwendungen durch IT-Administrationen soll möglich sein.

Auswertung: Die Installation von Programmen unter Linux-Distributionen ist grundlegend sehr einfach. Falls ein Programm bereits in einem der Paketmanagement-Systeme zur Verfügung steht, beispielweise *dnf* unter Fedora (englisch *Dandified Yum*) oder über *snap*, kann dieses mit nur einem Befehl installiert werden. Dies ist verglichen mit dem standartgemäßen Weg Programme unter Windows zu installieren deutlich einfacher. Vor allem Anwendungen für Entwicklungsprozesse sind in einer weiten Breite in Paketmanagern enthalten. Dennoch um Probleme bezüglich des Root-Filesystems zu vermeiden, müssen Anwendungen zwingend innerhalb von Template VMs installiert werden.

Anwendungen automatisch über viele Rechner zu verteilen ist jedoch einfacher unter Windows-Systemen. Dort gibt es eine Menge Hilfsprogramme, die sich auf diesem Bereich spezialisieren. Unter Qubes OS bestehen drei Möglichkeiten. Die Erstellung eines Backups bringt allerdings einginge Hürden mit sich, da Systemkonfigurationsdateien nicht in *dom0* gespeichert werden. Eine weitere in Kapitel 4.1.3 angesprochene Möglichkeit ist die Erstellung eines Speicherabbildes. Diese ist eine solide Möglichkeit Anwendungen und Konfigurationen über mehrere Rechner mit gleicher Hardware zu verteilen. Bei unterschiedlicher Hardware muss allerdings auf die Management-Software Salt, welche standartgemäß in *dom0* integriert ist, zurückgegriffen werden.

Installation von Windows innerhalb von Qubes OS

Die Einrichtung einer Windows VM soll möglich sein, um die Ausführung von Windows nativen Anwendungen gewährleisten zu können.

Auswertung: Qubes OS unterstützt die Installation jeglichen Betriebssystemen in separaten *Standalone* VMs. Diese virtuellen Maschinen gehören in die Kategorie von Hardware unterstützten VMs und sind aufgrund dessen langsamer als herkömmliche

VMs unter Qubes OS. Darum ist eine Ausführung von leistungsintensiven Anwendungen nur eingeschränkt möglich. Der Nutzen von einfachen nativen Windowsanwendungen ist möglich.

Allgemeine Nutzungsszenarien

Büroanwendungen

Der Einsatz von gängigen Büroanwendungen (Windows, Linux) soll unter Qubes OS möglich sein.

Auswertung: Qubes OS unterstützt den nativen Nutzen von LibreOffice. Hierbei werden die allgemeinen Bedürfnisse bereits abgedeckt. Dennoch ist es möglich Windows native Büroanwendungen zu verwenden. Um dies zu erreichen sind in Kapitel 4.2.1 verschiedene Möglichkeiten aufgelistet wurden. In einer manuell installierten Windows VM sind alle nicht hardwarebeschleunigten Windows-Anwendungen einsatzfähig.

Ebenso sind Konferenztools in Qubes OS verwendbar. Um Mikrofon und Kamera innerhalb einer bestimmten VM benutzen zu können, muss das entsprechende USB-Gerät erst zugewiesen werden. Es ist nicht möglich ein USB-Gerät an mehrere VMs gleichzeitig zuzuweisen. Nach Zuweisung innerhalb von Qubes OS ist es möglich das entsprechende Gerät innerhalb der Konferenzsoftware auszuwählen.

Dateimanagement

Das Verwalten und Verschieben von Dateien innerhalb bzw. zwischen VMs soll effektiv und intuitiv möglich sein.

Auswertung: Da Qubes OS auf Linux basiert, wird der *Filesystem-Hierarchie-Standard* für Verzeichnisstrukturen unter Unix-ähnlichen Betriebssystemen verwendet. Dies kann auf Nutzer ohne bzw. mit wenig Linux Erfahrung diskursiv wirken. Darüber hinaus besitzen AppVMs kein eigenes Root-Dateisystem, wodurch Anwendungen in Templates und nicht in AppVMs selbst installiert werden müssen. Durch den primären Nutzen von Paketverwaltungs-Anwendungen ist jedoch wenig Verwaltung des Nutzers selbst nötig.

Qubes OS unterstützt das sichere Verschieben von Dateien zwischen virtuellen Maschinen. Hierbei muss zwischen dem verschieben aus einer *AppVM* bzw. *Standalone* VM und *dom0* unterschieden werden. Um Dateien aus dom0 zu einer anderen VM zu

verschieben, muss die Konsole verwendet werden. Für dom0 gibt es keine Möglichkeit der Verschiebung von Dateien über die Benutzeroberfläche wie in Abbildung 10 aus Kapitel 4.2.2. Allgemeingesprochen ist das Verschieben von Dateien eine essenzielle Funktion in Qubes OS und sie funktioniert schnell und problemlos.

Nutzungsszenarien für Entwicklungsprozesse

Kommandozeilenbasiertes Server-Backend

Die Ausführung und Kompilierung von kommandozeilenbasierenden Server-Backends soll unter Qubes OS möglich sein.

Auswertung: Durch das solide Linux-Fundament ermöglicht Qubes die Entwicklung von Server-Backends. In Kapitel 4.3.1 wurde ein breites Spektrum an Technologien evaluiert. Durch die dort ausgeführten Tests konnte festgestellt werden, dass Qubes im Bezug zu Kompilierung und Ausführung dieser Technologien keine Probleme aufweist. Eine Hürde kann die initiale Installation der entsprechenden Abhängigkeiten sein. Hier ist wichtig zu verstehen, dass bei Installation dieser Abhängigkeiten innerhalb einer AppVM die Ausführung nur bis zum ersten Neustart der AppVM möglich ist. Um dies zu vermeiden kann beispielweise eine separate *Standalone* VM nur für Entwicklungsprozesse eingerichtet werden. Dies ist die solidere Methode als jegliche Abhängigkeiten in Template VMs installieren zu müssen. Auch ist es simpel mehrere Konsolen in verschiedenen Verzeichnissen gleichzeitig zu öffnen. Darüber hinaus sind API-Testwerkzeuge wie *Postman* unter Qubes OS kompatibel.

Anwendungen mit Web-, 2D und 3D-fähigen Oberflächen

Die Ausführung und Kompilierung von Anwendungen mit Web-, 2D und 3D-fähigen Oberflächen soll unter Qubes OS möglich sein.

Web-Oberflächen

Das Installieren und Einrichten von Abhängigkeiten ist einfach über die Paketmanager *npm* oder *yarn* möglich. Auftretende Fehler bei der Installation der Abhängigkeiten sind leicht über die in der Konsole geworfenen Fehler rückverfolgbar. Das Starten der webbasierenden Anwendungen, sei es per *localhost* im Browser oder als Desktopanwendung, funktioniert konform zu der Ausführung unter anderen

Betreibsystemen. Ebenso waren keine Einbrüche in der Leistung der Anwendungen spürbar.

2D-Oberflächen

Bei Anwendungen mit 2D-fähigen Oberflächen muss unterschieden werden. Es wurden vier Technologien für 2D-Anwendungen (*SDL2*, *DirectDraw*, *Direct2D*, *DirectWrite)* untersucht. In Kapitel 4.3.2.2 wurde die Programmierschnittstelle SDL2 getestet. Hier konnten die nötigen Abhängigkeiten problemlos über einen Konsolenbefehl installiert werden. Auch konnte festgestellt werden, dass das Programmierwerkzeug *cmake* in Qubes OS einsatzfähig ist. Die graphische Oberfläche war im vollen Umfang verwendbar ohne merkbare Leistungseinbrüche.

Die Schnittstellen *DirectDraw, Direct2D und DirectWrite* bringen allerdings einige Probleme mit sich. Einerseits sind diese Schnittstellen rein für Windows implementiert. Das bedeutet, dass zu Beginn eine Windows VM eingerichtet werden muss. Wie in 4.3.2.2 erörtert bringt dies Einschränkungen im Bereich der Leistung mit. Darüber hinaus verwenden diese Schnittstellen Hardwarebeschleunigung. Diese ist zu dem Zeitpunkt dieser Arbeit nicht mit Qubes OS unterstützt. Darum ist eine Entwicklung von 2D-Oberflächen nur unter bestimmten Voraussetzungen möglich. Für professionelle Entwicklung von 2D-Anwendungen ist von Qubes OS abzuraten.

3D-Oberflächen

Für Anwendungen mit 3D-Oberflächen wurden die beiden Programmierschnittstellen *Direct3D* und *OpenGL* untersucht. *Direct3D* bringt ebenfalls die Probleme der 2D-Schnittstellen *DirectDraw, Direct2D und DirectWrite* mit. *Direct3D* ist erneut Windows nativ und benötigt ein Windowsbetriebssystem zur Verwendung. Darüber hinaus wird Hardwarebeschleunigung verwendet, wodurch eine professionelle Entwicklung mit *Direct3D* unter Qubes OS nicht möglich ist.

OpenGL im Gegensatz ist Plattformunabhängig. Ebenso wird keine Hardwarebeschleunigung benötigt. In Kapitel 4.3.2.3 wurde erfolgreich ein OpenGL Projekt eingerichtet und ausgeführt. Die Installation der nötigen Abhängigkeiten funktionierte problemfrei. Jedoch ist durch die fehlende Möglichkeit der Prozessorentlastung ein deutlicher Einbruch in der Leistung der Anwendungen zu bemerken. Um einen validen Leistungstest durchzuführen, muss gegebenenfalls nach der Entwicklung auf ein Windowscomputer zurückgegriffen werden. Dies Schränkt eine professionelle Entwicklung deutlich ein.

5.2 Grenzen und Stolpersteine

In diesem Abschnitt wird gesondert auf Solpersteine und Grenzen von Qubes OS eingegangen. Es werden Stolpersteine, welche über den Zeitraum dieser Arbeit aufgetreten sind, erläutert. In einigen Fällen wurden folgende Aspekte bereits in den vorherigen Kapiteln angesprochen.

Stolpersteine

Die Installation von Programmen kann auf verschiedene Arten durchgeführt werden. Paketmanagement-Programme erleichtern dieses signifikant. Jedoch kann es bei Installation von Anwendungen innerhalb von AppVMs Probleme geben. Nach Installation des Programmes funktioniert dieses in der AppVM, bis diese neugestartet wird. Dies trat vor allem im Bezug zu dem Paketmanager *snap* auf. Allerdings besteht dieses Problem bei jeglicher Software, welche Änderungen im Root-Filesystem durchführt. Um dies zu umgehen, bestehen drei Optionen. Einerseits kann eine *Standalone* VM für die Installation von Anwendungen, welche das Root-Filesystem benötigen verwendet werden. Beispielweise hierfür sind Programme für die Entwicklung. Für diese werden vielen Abhängigkeiten benötigt, welche im Root-Filesystem installiert werden. Andererseits können Anwendungen direkt innerhalb von Template VMs installiert werden. Somit wären Anwendungen auch in AppVMs, die aus dem entsprechenden Template erstellt wurden, verwendbar. Hier muss allerdings aufgepasst werden, da standardgemäß zwei Templates VMs bestehen. Einmal eine Fedora- und Debian-Distribution. Diese werden für alle AppVMs verwendet und ein Benutzer möchte im Normalfall nicht alle Anwendungen in jeder AppVM haben. Um dies zu vermeiden, müssten separate Template VMs erstellt werden. Beispielweise eine Template VM mit allen Anwendungen für Entwicklungsprozesse. Letztlich besteht die Möglichkeit Anwendungen beständig zu machen über die Funktion *bind-dirs*. Dieser Mechanismus bietet, dass Dateien, welche normalerweise aus einer Template VM stammen, über einem Neustart hinweg erhalten bleiben.[18]

Die WLAN-Verbindung des Computers wurde automatisch getrennt nach zu langem Sperrbildschirm. Es war nicht möglich die Verbindung über die Einstellungen wiederherzustellen. Die einzige Lösung war ein Neustart des Computers.

Probleme traten auf im Bezug zu dem Nutzen von Webcams innerhalb von Konferenzwerkzeugen. Die Integrierte Webcam des *Lenovo Thinkpads p53s*, sowie eine extern angeschlossene Kamera haben nicht funktioniert. In Absprache mit anderen Qubes OS Nutzern

[18] https://www.qubes-os.org/doc/bind-dirs/

stellte sich heraus, dass dies ein spezielles Verhalten in Abhängigkeit von der Hardware ist und nicht in dieser Form bei ihnen auftritt.

Qubes OS besitzt eine spezielle Zwischenablage, die das Kopieren und Einfügen von Text zwischen verschiedenen Qubes/VMs ermöglicht. Für das Kopieren des Textes muss die Kombination *Strg+Shift+C* gedrückt werden. Für das Einfügen in der Ziel-Qube *Strg+Shift+V*. Dies ist eine sehr nützliche Funktion, jedoch hat diese teilweise funktioniert und teilweise wurde nichts kopiert, obwohl eine Bestätigung für das Kopieren angezeigt wurde.

Abschließend anzumerken ist, dass die größte Grenze, die einem Software-Ingenieur durch Qubes OS gesetzt wird, die schon oft angesprochene Hardwarebeschleunigung ist. Diese ist vor allem bei der Entwicklung von 3D-Anwendungen bzw. Spielen essenziell. Eine mögliche Lösung hierfür soll die momentan noch im Test befindende „GUI-Domain“ sein. In Kapitel 6.2 wird ein kurzer Einblick in den aktuellen Stand der „GUI-Domain“ gegeben.

6 Fazit und Ausblick

In diesem Kapitel werden die Ergebnisse dieser Arbeit zusammengefasst. Anschließend wird ein Ausblick in die mögliche Richtung, in die sich Qubes OS begibt, gegeben.

6.1 Zusammenfassung

In dieser Bachelorarbeit wurde Qubes OS für den Einsatz in klein- und mittelständischen Software-Entwicklungsunternehmen evaluiert. Ein Erfahrungsbericht aus einer vergleichbaren Perspektive wurde erstellt. Hierbei wurden typische Nutzungsszenarien von Entwicklungsunternehmen identifiziert und in Kombination mit Qubes OS untersucht. Erforderlich hierfür war es die grundlegenden Sicherheitsmechanismen von Qubes OS in eine Übersicht zu bringen.

Als Ergebnis steht fest, dass Qubes OS als Desktop-OS für Software-Ingenieure geeignet sein kann. Das signifikante Defizit ist die fehlende Hardwarebeschleunigung, welche die Entwicklung von 2D- und 3D-Anwendungen deutlich einschränkt. Ebenso ist die Nutzung von einigen Windows native Anwendungen nur bedingt möglich. Hierunter fallen auch die in Kapitel 4.3.2 angesprochenen graphischen Schnittstellen. Softwareentwicklung in Bereichen ohne Hardwarebeschleunigung ist ohne veränderte Vorgehensweisen im Lebenszyklus von Ingenieuren möglich. Hierunter fallen auch die untersuchten Bereiche der Web-Entwicklung, also die Entwicklung von kommandozeilenbasierten Server-Backends, sowie Frontend-Entwicklung mit modernen Frameworks.

Durch die sehr spezifischen Systemanforderungen können erhöhte Kosten für die Anschaffung neuer Hardware entstehen. Hingegen ist Qubes OS selbst Open-Source und benötigt im Vergleich zu Windows keine Lizenz. Darüber hinaus können die Sicherheitsvorteile, durch Kapselung von Tätigkeiten in verschiedene virtuelle Maschinen, möglichen Nachteilen durchaus überwiegen. Nichtsdestotrotz ist eine gewisse Lernkurve von Nöten. Hierbei können zu vorige Linux-Kenntnis hilfreich sein.

Des Weiteren ist Qubes OS ist kein Mehrbenutzersystem, da eine sichere Isolation zwischen unprivilegierten Benutzern kaum erreichbar ist. Somit ist es für Unternehmen nicht möglich mehrere Mitarbeiter an einen Computer zu zuweisen, außer es werden *allgemeine* Benutzer angelegt. Also eine Individualisierung der einzelnen Nutzer an einem Computer ist ausgeschlossen.

Anhand dieser Vor- und Nachteile muss ein SW-Entwicklungsunternehmen für sich selbst abwägen, ob Qubes OS als Betriebssystem für ihren Nutzen herangezogen werden kann.

6.2 Ausblick

Diese Arbeit hat gezeigt, dass Qubes OS durchaus als Betriebssystem innerhalb von SW-Entwicklungsunternehmen eingesetzt werden kann. Im Laufe der Anfertigung dieser Arbeit ist eine überschaubare Anzahl an Problemen aufgefallen. Auf diese im Kapitel 5.2 nochmals eingegangen wurden. Diese sind unteranderem Hardwarebedingt bzw. kleine Stolpersteine, welche in einem gewissen Rahmen vernachlässigt werden können.

Jedoch besitzt Qubes OS aktuell zwei große Hürden, die nicht vernachlässigbar sind. Einerseits wird Qubes OS stark durch die fehlende Hardwarebeschleunigung eingeschränkt. Jedoch ist dies hauptsächlich relevant für die Entwicklung von anspruchsvollen 2D- und 3D-Anwendungen. Andererseits ist eins der größten Sicherheitsprobleme von Qubes OS, dass zu viel Autorität in *dom0* steckt. Sobald ein Angreifer Zugriff auf *dom0* hat, kann dieser alles tun, obwohl dom0 ziemlich effektiv von Anwendungen innerhalb von AppVMs getrennt wird. Dennoch ist dom0 eine große und komplexe Domäne, die viele verschiedene Funktionen ausführt. Sie verwaltet anderen Domänen, Anzeige- und grafische Schnittstellen, mehrere Geräte, Speicher- und Festplattenverwaltung.

Eines der größten nächsten Schritte von Qubes OS hat das Potenzial beide dieser Probleme zu lösen. Der Schritt ist die Entkopplung der grafischen Hardware, der Anzeige und Verwaltung über die sogenannte *GUI-Domäne*. Diese Domäne ist eine von *dom0* getrennte Qube, die alle displaybezogenen Aufgaben und ausgewählte Systemverwaltungsaufgaben übernimmt. [20] Die GUI-Domäne ist aktuell testbar in der Qubes Version 4.1. Diese ist zu dem Zeitpunkt der Bearbeitung dieser Arbeit noch in einer Testphase und eine Evaluierung dieser Domäne ist nicht Gegenstand dieser Arbeit.

Neben der genaueren Evaluierung von Qubes-Whonix oder einer Untersuchung von Zero-Downtime-Möglichkeiten, stellt eine Analyse der GUI-Domäne ein perfektes Thema für eine mögliche auf dieser Arbeit aufbauende Bachelorarbeit dar.

Literaturverzeichnis

[1] A. Weber, „bitkom-research,“ [Online]. Available: https://www.bitkom-research.de/de/pressemitteilung/angriffsziel-deutsche-wirtschaft-mehr-als-220-milliarden-euro-schaden-pro-jahr. [Zugriff am 1 Januar 2022].

[2] J. Haffejee und B. Irwin, „Testing antivirus engines to determine their effectiveness as a security layer,“ *IEEE,* 2014.

[3] B. Akin, S. Stender und G. McGraw, „Software penetration testing,“ *IEEE Security & Privacy,* 2005.

[4] Y. Cai, C. Jia, S. Wu, K. Zhai und W. K. Chan, „ASN: A Dynamic Barrier-based Approach to Confirmation of Deadlocks from Warnings for Large-Scale Mutlithreaded Programs,“ *IEEE Transactions on Parallel and Distributed Systems,* pp. 13-26, 2015.

[5] Y. Cai und W. K. Chan, „Magiclock: Scalable detection of potential deadlocks in large-scale multithreaded programs,“ *IEEE Transactions on Software Engineering,* pp. 266-281, 2014.

[6] L. K. Shar, H. B. K. Tan und L. C. Briand, „Mining Sql Injection and cross site scripting vulnerabilities using hybrid program analysis,“ *Proceedings of the 35th ACM/IEEE International Conference on Software Engineering (ICSE), San Francisco,* pp. 18-26, 2013.

[7] M. Felderer , M. Büchler, M. Johns, A. D. Brucker, R. Breu und A. Pretschner, Chapter one - Security Testing: A Survey, Elsevier, 2016.

[8] Y. Cai und Q. Lu, „Dynamic Testing for deadlocks via constraints.,“ *IEEE Trans Softw. Eng.,* pp. 825-842, 2016.

[9] S. Zaman, B. Adams und A. E. Hassan, „Security Versus Performance Bugs: A Case Study on Firefox,“ *Proceedings of the 8th Working Conference on Mining Software Repositories,* pp. 93-102, 2011.

[10] C. Ioannidis, D. Pym und J. Williams, „Information Security Trade-offs and Optimal Patching Policies,“ *University of Bath, Department of Economics,* 2011.

[11] qubes-os, „Device handling security,“ [Online]. Available: https://www.qubes-os.org/doc/device-handling-security/. [Zugriff am 11 November 2021].

[12] J. Rutkowska, „Software compartmentalization vs. physical seperation,“ *Invisible Things Lab,* 2014.

[13] qubes-os, „Privacy in non-Whonix qubes,“ [Online]. Available: https://www.qubes-os.org/faq/#:~:text=What%20about%20privacy%20in%20non%2DWhonix%20qubes. [Zugriff am 1 Februar 2022].

[14] R. Dingledine, N. Mathewson und P. Syverson, „Tor: The Second-Generation Onion Router,“ 2004.

[15] V. Perta, M. Barbera, G. Tyson, H. Haddadi und A. Mei, „A Glance through the VPN Looking Glass: IPv6 Leakage and DNS Hijacking in Commercial VPN clients,“ 2015.

[16] saltproject.io, „Salt documentation,“ [Online]. Available: https://docs.saltproject.io/en/latest/. [Zugriff am 11 November 2021].

[17] qubes-os, „Use salt in combination with qubes OS,“ [Online]. Available: https://www.qubes-os.org/doc/salt/. [Zugriff am 11 November 2021].

[18] qubes-os, „How to copy and move files,“ [Online]. Available: https://www.qubes-os.org/doc/how-to-copy-and-move-files/. [Zugriff am 1 Februar 2022].

[19] qubes-os, „Run application, which require hardware acceleration,“ [Online]. Available: https://www.qubes-os.org/faq/#can-i-run-applications-like-games-which-require-hardware-acceleration. [Zugriff am 11 November 2021].

[20] qubes-os, „The GUI Domain,“ [Online]. Available: https://www.qubes-os.org/news/2020/03/18/gui-domain/. [Zugriff am 1 Februar 2022].