JOHANNES KEPLER
UNIVERSITÄT LINZ | JKU

TNF

Technisch-Naturwissenschaftliche
Fakultät

Sicherheitsaspekte bei Virtualisierungen

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplomingenieur

im Masterstudium

Netzwerke und Sicherheit

Eingereicht von:
Markus Jäger BSc

Angefertigt am:
Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM)

Beurteilung:
Univ.-Doz. Dipl.-Ing. Dr. Gerhard Eschelbeck

Linz, Juli 2014

Vorwort und Dankesworte

Sicherheitsaspekte bei Virtualisierungen – ich bin froh, dieses von Dr. Eschelbeck vorgeschlagene Thema ausgewählt und bearbeitet zu haben. Es war von Anfang an klar, dass dieser riesige Bereich der Informatik bzw. die besondere Behandlung von Sicherheitsaspekten ein eher theoretisches Arbeitsgebiet sein wird. Im Zuge der Recherchen haben sich nicht viele praktische Möglichkeiten ergeben, die für das Thema genutzt werden können. Trotzdem freut es mich, dass ich mich in diesem Bereich vertieft habe. Virtualisierungen sind wichtiger und verbreiteter als je zuvor, weshalb ihnen, meines Erachtens, viel Aufmerksamkeit zukommen soll. Sollte sich in Zukunft die Gelegenheit ergeben, möchte ich mich auf jeden Fall weiter mit Virtualisierungen beschäftigen.

Dieser Abschnitt ist dafür gedacht, jenen Menschen zu danken, die mich bei dieser Arbeit sowohl privat als auch beruflich unterstützt haben. Ganz besonderer Dank gilt meinem Betreuer, Dr. Gerhard Eschelbeck, für die vorbildliche und kollegiale Zusammenarbeit während der Erstellung dieser Arbeit. Ich wünsche jedem Studierenden einen solchen Betreuer, der einem viel Freiheit bei der Erarbeitung eines Themas lässt, aber dennoch die nötigen richtungsweisenden Schritte setzt, sollte man sich einmal auf dem Holzweg befinden.

Danke an Lara Aigmüller, Gerlinde Jäger und Veronika Kalcher für das zeitaufwändige und doch recht kurzfristige Korrekturlesen und für die Verbesserungsvorschläge bei einigen Formulierungen. Dank euch ist diese Arbeit in guter deutscher Sprache verfasst.

Herzlichen Dank an die Mitglieder des Prüfungssenates meiner abschließenden Masterprüfung, welche sich trotz der Kurzfristigkeit der Terminvereinbarung dazu bereit erklärt haben, mich zu prüfen: Dr. Michael Sonntag als Vorsitzender und Prüfer dieser Arbeit, Dr. Gerhard Eschelbeck als Kernfachprüfer in "Netzwerke und Sicherheit" sowie Dr. Josef Küng als Nebenfachprüfer in "Software Engineering".

Danke auch an Dr. Rudolf Hörmanseder für die Unterstützung während meiner Zeit am Institut und für die gute und vor allem zeitaufwändige Zusammenarbeit bei der Erstellung der Publikation "Cloud Security Problems caused by Virtualization Technology Vulnerabilities and their Prevention" für die Konferenz IDIMT (Interdisciplinary Information and Management Talks) im September 2014 in Poděbrady, CZ. Das Paper, auf dem ein Großteil der Inhalte von Kapitel 6 basieren, haben wir gemeinsam erarbeitet, die Grafik 6.1 stammt aus seiner Feder. Auch an Dr. Michael Sonntag noch einmal Danke für die wertvollen Optimierungshinweise, die dieser Publikation auch dazu verholfen haben, für die Konferenz akzeptiert zu werden.

Zusammenfassung

Virtualisierung ist heutzutage sehr stark verbreitet, umso wichtiger ist ihre Sicherheit, welche in dieser Arbeit thematisiert wird. Kapitel 1 gibt eine kurze Einführung in die Thematik, Definitionen und einen historischen Rückblick; in Kapitel 2 wird eine Klassifikation von Virtualisierungsarten vorgenommen und das Modell der Privilegierungen erläutert.

Kapitel 3 behandelt allgemeine Überlegungen zur Erhaltung der Sicherheit, besonders bei Virtualisierungen, gibt einen Überblick über die Arten und Funktionsweisen von Schadsoftware und erklärt, warum die Vorgehensweise des Sandboxings für Virtualisierungssicherheit wichtig ist. Weiters werden detaillierte Sicherheitsaspekte bei der System- bzw. Server-Virtualisierung erläutert. Hypervisoren bzw. Virtual Machine Monitors (VMM) haben einen besonderen Stellenwert bei groß angelegten Virtualisierungsumsetzungen. Die verbreitetsten VMMs werden in Kapitel 4 behandelt und in Hinblick auf mögliche Sicherheitsoptionen untersucht.

Einen besonderen Stellenwert nehmen Virtual Machine Based Rootkits (VMBR) bzw. hardwarevirtualisierende Rootkits ein (Kapitel 5). Sie sind in Bezug auf Virtualisierung deswegen interessant, da es einigen Ausführungen von VMBRs gelingt, nicht nur laufende Betriebssysteme sondern auch VMMs zu untergraben und die Kontrolle über die Hardware zu erlangen. Kapitel 6 behandelt Virtualisierungssicherheit in Zusammenhang mit Cloud Computing und Kapitel 7 gibt einen Ausblick, besonders in Bezug auf mobile Systeme.

Keywords: Cloud Computing, Hypervisor, Virtualisierung, Virtualisierungssicherheit, Virtual Machine Based Rootkits (VMBR), Virtual Machine Monitor (VMM)

Abstract

The use of Virtualization is widespread in these days, even more important is its security, which is discussed in this work. Chapter 1 introduces Virtualization and its history, and chapter 2 shows, which types of Virtualization exist, how they work and introduces the Privileging-Model.

In chapter 3 general security aspects are discussed as well as malware and how malware works is described, including, why Sandboxing is important for Virtualization Security. Especially VMMs/Hypervisors are important for todays Virtualization technologies, so the most important VMMs are described in chapter 4.

Virtual Machine Based Rootkits (VMBR) are of special interest, as some of them are able to undermine not only running operating systems, but also active VMMs (chapter 5). Closing this thesis, there are Virtualization Security issues concerning Cloud Computing discussed in chapter 6 and an outlook towards mobile systems and Virtualization and Virtualization Security in the future is given in chapter 7.

Keywords: Cloud Computing, Hypervisor, Virtualization, Virtualization Security, Virtual Machine Based Rootkits (VMBR), Virtual Machine Monitor (VMM)

Aufgabenstellung

Es soll untersucht und evaluiert werden, welche Risiken bei Virtualisierungsarten auftauchen und wie Angriffe abgewandt werden können. Dabei soll neben der Prozessvirtualisierung, zu der die Applikations- und Bibliotheks-Virtualisierung zählen, sowie der Systemvirtualisierung, welche die Emulation, Betriebssystem-, Voll- und Paravirtualisierung beinhaltet, auch auf das Verfahren des Sandboxings eingegangen werden.

Der erste Teil der Arbeit soll einen Überblick über die aktuelle Lage und die vorhandenen Technologien geben sowie die Funktionsweise der einzelnen Virtualisierungsarten erläutern. Dabei sollen auch die Hardwareabhängigkeiten der einzelnen Virtualisierungsarten untersucht werden.

Nachdem ein guter Überblick über das Thema gegeben wurde, sollen mögliche Szenarien und Risiken erörtert, sowie eine Darstellung ermittelt werden, welche Möglichkeiten man im Software- und Hardwarebereich hat, um diesen Risiken entgegen zu wirken. Vor allem bei den Hardware-Möglichkeiten soll darauf eingegangen werden, ob eine Erweiterung der Hardware mehr Sicherheit bringt oder anderweitige Verfahren zur Angriffsabwehr vorgezogen werden sollen. Auch die Virtual Machine Monitors (VMM) sollen dahingehend untersucht werden, welche Möglichkeiten man in Bezug auf Sicherheit hat. Diese Bereiche sollen den Großteil der Arbeit darstellen, weshalb ihnen auch der höchste Detailgrad zugesprochen wird.

Seit ca. 2006 gibt es im Bereich der Virtualisierungen eine neue Art von Bedrohung: die Virtual Machine Based Rootkits (kurz VMBR). Da diese Art von Schadsoftware für Betriebssysteme und in weiterer Folge auch für Virtualisierungen besonders gefährlich ist, wird ihnen ein eigener Abschnitt gewidmet.

Da die immer stärker werdende Anwendung von Clouds bzw. Cloud Computing hauptsächlich durch Virtualisierungstechnologien umgesetzt wird, sollen auch Sicherheitsaspekte und Gefährdungen von Clouds behandelt werden.

Als Abschluss der Arbeit soll eine Zusammenfassung und ein Resümee über die erlangten Informationen gemacht, sowie ein Ausblick über weitere Möglichkeiten gegeben werden.

Die Forschungsfrage, die mit dieser Arbeit beantwortet und behandelt werden soll ist, wie der derzeitige Stand von Technik und Sicherheit in Bezug auf Virtualisierungen in deren Gesamtheit ist.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

Virtualisierung – was ist das? Die eigentlich bessere Frage lautet: Was ist heutzutage *nicht* mehr Virtualisierung? Die Anwendung von Abstraktion im IT-Sektor ist ein technologischer Bereich, der kaum noch wegzudenken ist. Egal ob es um den virtuellen Speicher, die Einbindung von virtuellen Laufwerken, virtuelle lokale Netzwerke (VLANs) oder virtuelle Server geht.

1.1 Motivation

Es gibt vermutlich kein Unternehmen mehr, welches ohne Virtualisierung auskommt, auch wenn die meisten Personen wahrscheinlich gar nicht wissen, was bei ihnen alles virtualisiert wird. Es kann vorkommen, dass auf virtualisierten Maschinen gearbeitet wird und die einzigen, die davon wissen, sind die Systemadministratoren. Es ist also schon nach wenigen Sätzen klar, dass der Einsatz von virtuellen Maschinen (künftig als VM oder im Plural als VMs bezeichnet) sehr verbreitet ist. Welche Vorteile und Möglichkeiten Virtualisierung mit sich bringt, wird in den folgenden Kapiteln beschrieben.

Was jedoch für diese Arbeit noch interessanter ist, ist die Frage, welche Sicherheitsrisiken Virtualisierung mit sich bringt und welche neuen Möglichkeiten von Attacken von Innen und Außen sich damit als potenzielle Gefahren ergeben. Da Sicherheit im IT-Sektor immer schon eine große Rolle gespielt hat – und diese Rolle in Zukunft bestimmt nicht an Bedeutung verlieren wird – und auch die Anwendung von Virtualisierung immer weiter wächst, soll diesem speziellen Thema, die gezielte Betrachtung von Sicherheitsthemen bei Virtualisierungen, besondere Aufmerksamkeit zukommen.

Zusätzlich zu den klassischen Virtualisierungsarten bietet auch das Verfahren des Sandboxing viele interessante Aspekte, da bei diesem Verfahren speziell die Sicherheit im Vordergrund steht.

Neben diesen Gründen und Fragestellungen gibt auch das persönliche Interesse des Autors Anlass zur Behandlung der Thematik. Des weiteren gaben die Inhalte, die in der Lehrveranstaltung "System Security" an der Johannes Kepler Universität (JKU) Linz von Dr. Eschelbeck gelehrt werden, die Intention für diese Arbeit.

1.2 Definitionen

Im Folgenden werden Definitionen aus unterschiedlichen Quellen gegenübergestellt:

> *"Virtualization, in its broadest sense is the emulation of one or more workstations/servers, within a single physical computer. In other words, it is the emulation of hardware within a software platform. This type of virtualization is sometimes referred to as full virtualization and it allows one physical computer to share its resources across a multitude of environments. This means that a single computer can essentially take the role of multiple computers."* [26]

Diese Definition von Menken/Blokdijk bringt es sehr gut, wenn auch etwas verallgemeinert, auf den Punkt: *"[...] im weitesten Sinne die Emulation von einem oder mehreren Rechnern auf einem einzigen physischen Computer"*. Anwendung findet diese Beschreibung vor allem bei der Server- und Workstation-Emulierung.

> *"Virtualisierung bezeichnet in der Informatik die Erzeugung von virtuellen (d. h. nicht physikalischen) Dingen wie einer emulierten Hardware, eines Betriebssystems, Datenspeichers oder Netzwerkressource. Dies erlaubt es etwa, Ressourcen von Computern (insbesondere im Server-Bereich) transparent zusammenzufassen oder aufzuteilen, oder ein Betriebssystem innerhalb eines anderen auszuführen."* [37]

Zitate und Referenzen von und aus Wikipedia eignen sich hervorragend dafür, mittels Gegenüberstellung von Definitionen eine Einführung bzw. einen Überblick über die Thematik zu erhalten. Vor allem beim Vergleich von Definitionen des selben Themas in unterschiedlichen Sprachen kann oft ein gutes Grundverständnis des Fachgebietes vermittelt werden.

> *"Virtualization, in computing, is a term that refers to the various techniques, methods or approaches of creating a virtual (rather than actual) version of something, such as a virtual hardware platform, operating system (OS), storage device, or network resources."* [38]

Weitere zutreffende bzw. sehr gut beschreibende Erklärungen und Definitionen wurden in der Lehrveranstaltung "Systemadministration" an der JKU, im Vorlesungsteil "Virtualisierung" gegeben:

- *"Isomorphismus, der ein virtuelles Gast System auf ein reelles Host System abbildet.*
- *Führt Software in der gleichen Art und Weise aus, wie die Maschine, für die die Software geschrieben wurde.*
- *Kombination aus den Ressourcen einer echten physischen Maschine und spezieller Virtualisierungs Software.*
- *Die Ressourcen der virtuellen Maschine können sich in Zahl und Qualität von denen des Hosts unterscheiden."* [28]

Im Buch "Running Xen" wird im ersten Kapitel sehr gut beschrieben, wie Virtualisierung funktioniert, was Virtualisierung aber eigentlich ist, wird jedoch nicht direkt behandelt. Lediglich in der Einleitung wird die Bedeutung von Virtualisierung kurz angeschnitten:

> *"Xen is a virtual machine monitor (hypervisor) that allows you to use one physical computer to run many virtual computers..."* [24]

1.3 Geschichte

Das Virtualisierungskonzept existiert schon viel länger, als den meisten Leuten bewusst ist. Erst in der heutigen Zeit gewinnt es an Bekanntheit, weil es mehr und mehr genutzt wird und immer größere Hardwareressourcen zur Verfügung stehen. Das erste Auftauchen von Virtualisierung gab es in den 1960er Jahren, als die Firma IBM ihre großen Mainframe-Maschinen in kleinere VMs mittels logischer Partitionierung aufteilen wollte, um die Effizienz dieser zu optimieren. Das Problem war, dass jeder Mainframe zu einem Zeitpunkt immer nur an einem Prozess arbeiten konnte. Gelöst wurde das Problem, indem die Hardware des Mainframes in separate Entitäten aufgeteilt wurde.

Im Jahre 1977 erschien das erste kommerziell verfügbare Computersystem – das IBM System 370 – welches für Virtualisierung geeignet war, mit dem Betriebssystem CP/CMS (Control Program/Cambridge Monitor System), welches es ermöglichte, mehrere Betriebssysteme auf dem IBM System 370 laufen zu lassen. Das weiterführende VM/CMS (Virtual Machine/Cambridge Monitor System) war für die folgenden Maschinen der System-z Reihe das vorwiegende Betriebssystem (siehe Kapitel 4.5).

Danach geschah im kommerziellen Virtualisierungssektor lange Zeit nichts Innovatives. 1999 gab es VMware für x86-Architekturen, 2003 XEN Paravirtualisierung und Microsoft's VirtualPC. Seit 2005 arbeiten Prozessorhersteller an weiterer Hardware-Unterstützung für ihre Produkte. Intel hat seine Virtual Technology (VT) unter dem Namen "Vanderpool" entwickelt, während AMD an der AMD Virtualization (AMD-V) unter dem Namen "Pacifica" arbeitete. Eines der Ziele war unter anderem, explizit angegebene Funktionalitäten zu den bestehenden Virtualisierungsarten hinzuzufügen, um eine höhere Performanz bei Vollvirtualisierungs-Hypervisoren zu erreichen.

Erst seit dieser Zeit wird wieder intensiv an der Verbesserung von Virtualisierungsmöglichkeiten gearbeitet. Neue Technologien sind die Linux Kernel Based VM (KVM) (siehe Kapitel 4.3), die unmodifizierten Gastsysteme von Xen (siehe Kapitel 4.4), seit 2008 der "neue Stern am Virtualisierungs-Himmel", Microsofts Hyper-V (siehe Kapitel 4.2) und seit 2009 der integrierte Windows XP Mode im Betriebssystem Windows 7. [1]

[1] Danke an dieser Stelle an DI Dr. Christian Praher für die Aufbereitung einer übersichtlichen Zeitlinie.

Die Wichtigkeit von Virtualisierung zeigt sich auch in der intensivierten Anwendung im Serverbereich. Laut [5] bzw. Statistiken der International Data Corporation (IDC) wurden 2012 doppelt so viele virtualisierte Server wie physische Server ausgebracht und 2011 waren 55% aller installierten Server virtualisiert, alle neu installierten Server sind mit einem Anteil von 67% virtualisiert.

1.4 Vorteile und Ziele

Die folgenden Aufzählungen von Vorteilen, Zielen und Gründen für die Nutzung von Virtualisierung wurden aus [24], [30] und [26] entnommen:

Increase usage of hardware resources: Aktuelle Maschinen haben in der Regel nur eine sehr geringe Auslastung. Mit Virtualisierung kann dieser Umstand beseitigt werden, indem einzelne Maschinen für mehrere Aufgaben genutzt werden und somit die Hardware besser ausgelastet wird. Der jeweilige System-Administrator kann frei entscheiden, welche Dienste und Services in einer VM und welche/wieviele VMs auf einem physischen Server laufen, natürlich abhängig von den Ressourcen, die zur Verfügung stehen.

Reduce management and resource costs / Resource optimization: Wenn viele Rechner auf einer Maschine laufen, erleichtert und zentralisiert dies die Administration und Wartung – weit weniger zeitraubende Hardwarefehler müssen behoben werden und viele Maschinen können von einem Ort aus konfiguriert und überwacht werden. Auch das Sichern und Wiederherstellen bei virtualisierten Umgebungen verläuft wesentlich unkomplizierter als bei physikalischen Maschinen. Selbiges gilt für Services: mehrere Services auf einem Server laufen zu lassen ist nicht zu empfehlen. Laufen diese Services in VMs, können so viele Services bzw. VMs parallel auf einem Gerät ausgeführt werden, soweit es die vorhandenen Ressourcen erlauben. Weiters ist es auch einfacher, ein einzelnes Gerät mit Strom zu versorgen und ausreichend zu kühlen, als einen ganzen Serverraum. Somit wird auch Geld gespart, wenn für jede Aufgabe nicht ein eigenes Gerät angeschafft werden muss.

Rapid deployment: Da die virtuelle Festplatte einer VM oftmals in einer einzelnen Datei auf der physischen Festplatte abgelegt ist, ist eine Portierung sehr einfach. Auch das Duplizieren von VMs ist damit sehr unkompliziert. Eine VM kann auch als Vorlage dienen, beispielsweise für Arbeitsplatzrechner: einmal konfiguriert kann die Maschine für beliebig viele Arbeitsplätze dupliziert und verwendet werden.

Improve business flexibility: Auch hier spielt der Kostenfaktor eine große Rolle. Muss ein Unternehmen die Anzahl an Arbeitsplätzen und Servern vergrößern, ist es deutlich einfacher, weitere virtuelle Umgebungen zu initialisieren und zur Verfügung zu stellen, als neue Hardware anzuschaffen. Nach einer gewissen Anzahl von Erweiterungen müssen neue Server zur Infrastruktur hinzugefügt werden. Das ergibt sich aus der begrenzten Belastbarkeit einer jeden Maschine, ist aber immer noch billiger und flexibler als die Anschaffung von Arbeitsplatzrechnern.

Improve security and reduce downtime / Portability: Wenn eine Maschine nicht mehr funktioniert, ist es oft so, dass die gesamte Software und alle Daten, die darauf gespeichert waren, nicht mehr zugänglich sind. Es dauert meist eine gewisse Zeit, bis diese so genannte Downtime überwunden wurde und ein anderer Rechner zum Einsatz bereit steht. Funktioniert eine VM nicht mehr, so hat man zumindest bei der Datenrekonstruktion leichteres Spiel, da die Festplattendatei der Maschine noch vorhanden ist. VMs sind autonome Objekte untereinander, d. h. wenn eine Maschine beispielsweise von einem Virus infiziert ist oder nicht mehr funktioniert, so ist sie von den anderen isoliert und die Gefahr kann sich nicht ausbreiten, so zumindest der theoretische Ansatz.

Weiters sind VMs nicht hardwareabhängig, das bedeutet, wenn ein Server nicht mehr funktioniert, können die einzelnen Maschinen einfach auf einen anderen Server migriert/portiert werden. Dies ermöglicht geplante Stehzeiten, die auf ein Minimum reduziert werden können. Es sind auch Portierungen während der Laufzeit möglich. Eine maximale Verfügbarkeit ist heutzutage für Unternehmen extrem wichtig, da auch schon kurze Ausfälle für Geschäftsverluste oder unzufriedene Kunden und Partner ausreichen.

Problem free testing / Sandboxing / Application separation: VMs können als Testmaschinen verwendet werden, um beispielsweise die Stabilität von Programmen zu testen, ohne den laufenden Betrieb gefährden zu müssen. Sie können ebenfalls für Sicherheits-Szenarien genutzt werden: lässt man eine VM in isoliertem Modus laufen, können die Auswirkungen von Schadsoftware untersucht werden und somit ergibt sich eine sehr gute Möglichkeit, die Bedrohung zu verstehen und schützende Maßnahmen zu ergreifen.

Elimination of compatibility issues: Wegen hardwarespezifischer Anforderungen ist es nicht immer einfach, Betriebssysteme wie Windows, Mac OS X oder unterschiedliche Derivate von Linux auf der selben Maschine laufen zu lassen – mit Virtualisierung wirft diese Anforderung keinerlei Probleme mehr auf.

Debugging operating- and security systems: Mittels Virtualisierung haben Entwickler die Möglichkeit, neue Betriebssysteme in Gastumgebungen auf Funktionsweise und Stabilität zu testen. Diese Vorgehensweise ist effizienter, als dies über konventionelle Hardware zu tun. Selbiges gilt für die Weiterentwicklung von Sicherheitsaspekten, da einzelne VMs gut voneinander isoliert werden können. Ebenso die Rekonstruktion von infizierten Systemen kann bei der Verwendung von VMs viel einfacher erfolgen, da beispielsweise mittels Zugriff auf Backups oder Snapshots[2] ein nicht infizierter Zustand der Maschine wiederhergestellt werden kann.

Hypervisors are useful for developers: Entwickler, die auf mehreren Betriebssystemen arbeiten, oder Programme entwickeln, welche auf unterschiedlichen Betriebssystemen funktionieren sollen, haben dank des Hypervisors/Virtual Machine Monitors (VMM) die Möglichkeit, ohne lästige Neustarts beliebig zwischen unterschiedlichen VMs mit den benötigten Betriebssystemen zu wechseln.

[2]Snapshots sind Sicherungspunkte des jeweiligen Zustandes der Maschine.

2 Klassifikation und Arten von Virtualisierung

Virtualisierungen können nach verschiedenen Gesichtspunkten eingeteilt werden: zum einen nach dem, *Was* virtualisiert wird, zum anderen *Wie* virtualisiert wird. Weiters stellt sich die Frage des Ziels, welches mit Virtualisierung erreicht werden soll und wie es umgesetzt wird. Neben den genannten Themenbereichen wird in diesem Kapitel zuerst auf das Modell der Privilegierungen, besonders von Prozessoren, eingegangen.

2.1 Privilegierungen

Bevor die Arten der Virtualisierung behandelt werden, wird eine kurze Einführung über die 32-bit Sicherheitsringe bzw. Privilegierungsstufen gegeben. Hardware, besonders aber Prozessoren, verfügen über so genannte Privilegien-Ringe, welche Sicherheitsniveaus für Befehle und Zugriffe darstellen. Je niedriger die Ebene, desto mehr Rechte hat der jeweilige Prozess. Systemkritische Operationen, welche direkt auf der Hardware ausgeführt werden, können so nur von einigen wenigen privilegierten Systemprozessen ausgeführt werden.

Das Modell, welches in Abbildung 2.1 (links) dargestellt ist, sieht vor, dass in der untersten Ebene der Kernel des Betriebssystems bzw. das Betriebssystem selbst residiert und nur dieser direkten Zugriff auf die Hardware hat. Wollen Prozesse oder Anwendungen aus höheren Ebenen auf die Hardware zugreifen, so muss das vom Betriebssystem genehmigt werden.

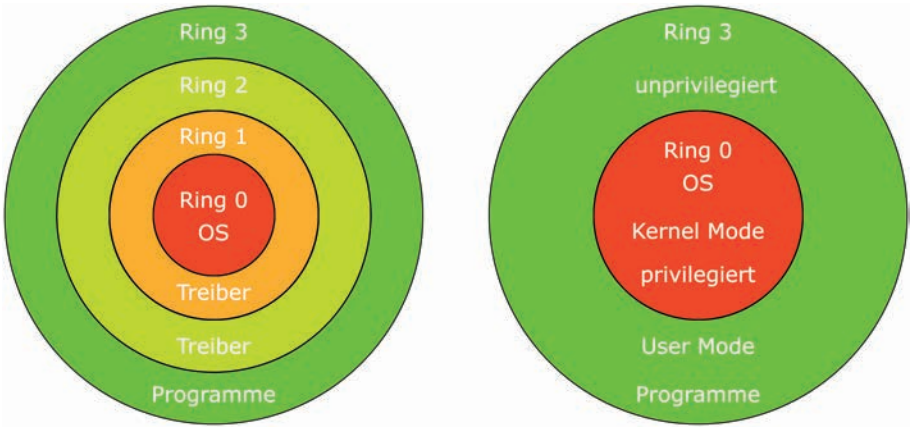

Abbildung 2.1: Modell Privilegierung und Anwendung

Da die Umsetzung des Privilegierungsmodells in der Praxis nicht zu den trivialsten Angelegenheiten gehört, verwenden die meisten Betriebssysteme ein eingeschränktes Modell, in dem nur die Ebenen 0 und 3 verwendet werden, wie in Abbildung 2.1 (rechts) zu sehen ist. In diesem Modell ist die Ebene 0 ebenfalls für den Kernel bzw. das Betriebssystem gedacht, die Ebene 3 für die Anwendungsprogramme.

2.2 Was und wie wird virtualisiert?

Die Einteilung danach, was virtualisiert wird, ist zwar etwas unüblich und mitunter auch schon veraltet, bietet aber einen guten Einstieg. Sie sieht Virtualisierung mehr aus dem Blickwinkel der Netzwerk-Technologien bzw. -Infrastrukturen und kann grob in die drei Kategorien Speicher-, Netzwerk- und Servervirtualisierung unterteilt werden: [30]

Speichervirtualisierung führt mehrere voneinander getrennte physikalische Speichermedien zusammen und lässt diese als ein einziges erscheinen, oder umgekehrt. Im Folgenden die am häufigsten verwendeten Speichervirtualisierungstechnologien [26]:

- **Network-attached Storage (NAS):** Hier handelt es sich um einen Server, der ausser Filesharing (Dateiserver)-Diensten keine weiteren Funktio-

nalitäten anbietet. Die Verwendung von NAS kann viele Vorteile bieten, beispielsweise die Anwendung von RAID (Redundant Array of Independet Disks) zur Geschwindigkeitserhöhung (RAID0: Striping) oder Erhöhung der Datensicherheit (RAID1: Mirroring), sowie Kombinationen daraus (RAID5: Striping mit Paritätsbit).

- **Storage Area Network (SAN):** Ein SAN ist ein (kleines) Unternetzwerk, das nur aus Speichergeräten besteht. Es sollte immer so aufgebaut sein, dass alle Mitglieder eines Netzwerkes darauf zugreifen können (sowohl LAN (Local Area Network) als auch WAN (Wide Area Network)). Neben der besseren Nutzung der Hardware bieten SANs auch den Vorteil einer besseren Skalierbarkeit (Speichergeräte die dem SAN hinzugefügt werden, sind sofort verfügbar) und Verfügbarkeit (Endbenutzer haben speziell durch das SAN einen besseren Zugriff auf ihre Daten).

- **Internet Small Computer System Interface (iSCSI):** Bei SCSI handelt es sich primär um einen Controller, um Hardware anschließen zu können; es benötigt also keine Software um zu funktionieren. Der Vorteil gegenüber IDE (Integrated Drive Electronics) liegt in der höheren Datenübertragungsgeschwindigkeit, was gerade bei Servern eine wichtige Rolle spielt. Ein weiterer Vorteil ist das Anschlussverhalten von SCSI: man kann mehrere Geräte (beispielsweise Festplatten) an einen SCSI-Controller anschließen, als mit dem herkömmlichen IDE Controller möglich ist (Master und Slave). iSCSI ist eine Erweiterung der Funktionalität von SCSI um die Einbindung des Internet Protocols (IP). iSCSI unterstützt bestehende Ethernet-Strukturen und erlaubt es, Netzwerkgeräte wie Router oder Switches miteinander zu verbinden (sofern diese über ein iSCSI-Interface verfügen). Das bedeutet, iSCSI Speichergeräte können sehr schnell über das Netzwerk angesprochen werden.

Netzwerkvirtualisierung verbindet Rechnerressourcen in einem Netzwerk, indem es die verfügbare Bandbreite in verschiedene Kanäle aufteilt, welche anschließend individuell in Echtzeit einzelnen Geräten zugeordnet werden kann.

Servervirtualisierung verbirgt sämtliche physische Ressourcen und Eigenschaften von Servern vor der Software, die auf ihnen ausgeführt wird. Wenn man von Virtualisierung spricht, ist zumeist die Servervirtualisierung gemeint. Diese Virtualisierungsart wird in Kapitel 2.4 näher beschrieben.

Wie wird virtualisiert? Weitaus häufiger erfolgt die Klassifizierung von Virtualisierungsarten danach, *wie* virtualisiert wird. Diese unterscheidet sich in Prozessvirtualisierung und System- bzw. Servervirtualisierung, welche in den folgenden Abschnitten beschrieben werden.

2.3 Prozess-Virtualisierung

Diese ermöglicht nur eine partielle Virtualisierung von Prozessen im Hostbetriebssystem bzw. virtualisiert nie ein gesamtes Betriebssystem. Die Prozessvirtualisierung kann weiter in Applikations-Virtualisierung und Bibliotheks-Virtualisierung unterteilt werden:

2.3.1 Applikations-Virtualisierung bzw. Managed Runtime

Sie ist am ehesten bekannt von Java- und den .Net-Plattformen – hier werden Programme in plattformunabhängige Zwischensprachen übersetzt. Es wird die Applikation mit ihren nötigen Ressourcen (wie beispielsweise DLL - Dynamic Link Librarys) zusammengepackt, das bedeutet, dass eine Applikation nicht auf die Ressourcen des Betriebssystems angewiesen ist. Vorteile sind die gute Portabilität, Sicherheit (Schadsoftware kann nicht auf das Betriebssystem gelangen), die Möglichkeit zur Parallelverwendung mehrerer Versionen des selben Programms, Unterstützung von Computern mit älterer Hardware und die Möglichkeit einer relativ sicheren und effizienten Entwicklung [28]. Die bekannteste Umsetzung von Applikations-Virtualisierung ist die VMware ThinApp (Thinstall Virtualization Suite)[3].

[3] `http://www.vmware.com/products/thinapp/overview.html`

2.3.2 Bibliotheks-Virtualisierung

Hier handelt es sich um die Emulation von Betriebssystem-Unterteilen, somit können fremde Programme direkt im Betriebssystem ausgeführt werden. Im Vergleich zu anderen Virtualisierungsarten macht die Bibliotheks-Virtualisierung nicht den Eindruck, dass es sich um ein einzelnes System mit einem vollständigen Betriebssystem handelt. Sie bietet die fehlende Programmierschnittstelle (API - application programming interface) für Anwendungsentwickler an. Nachteil ist die schlechte Performanz. Das bekannteste Beispiel für eine bibliotheksvirtualisierende Laufzeitumgebung ist Wine[4].

2.4 System-/Server-Virtualisierung

Hier handelt es sich um Komplettvirtualisierungen von Betriebssystemen. Sie werden unterteilt in Emulation, Betriebssystem-Virtualisierung, Voll-Virtualisierung und Para-Virtualisierung: [24]

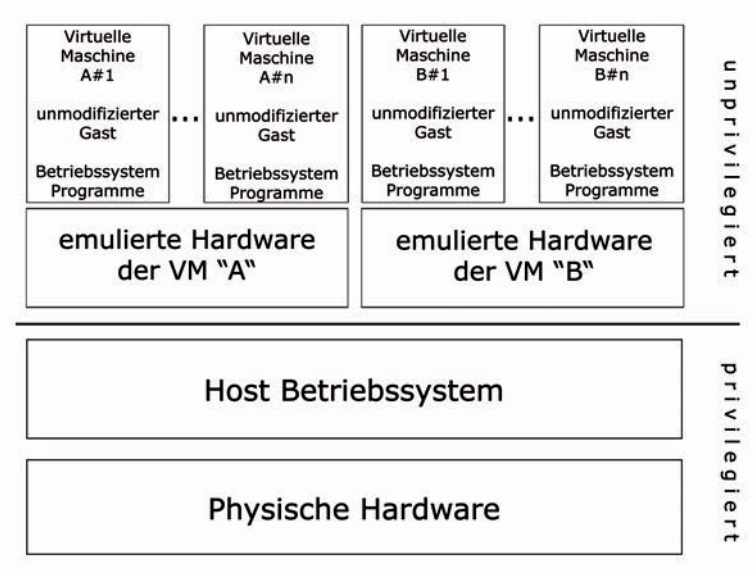

Abbildung 2.2: Aufbau Emulation

[4] http://www.winehq.org/

2.4.1 Emulation

Streng genommen handelt es sich bei Emulation um keine Virtualisierung sondern um Simulation. Es wird die benötigte Hardware für die Gast-Betriebssysteme mit Software emuliert, das heißt, die Hardware des Hosts muss nicht mit der des Gastes ident sein. Weiters erlaubt Emulation die Verwendung von unmodifizierten Gast-Systemen und ist ideal für die Entwicklung neuer Systeme, für die noch keine passende Hardware verfügbar ist. Der Vorteil von Emulation ist, dass Hardware simuliert wird, die physisch nicht vorhanden ist. Der Nachteil hingegen ist die schlechte Performanz. Der Aufbau von Emulation ist in Abbildung 2.2 dargestellt.

2.4.2 Betriebssystem-Virtualisierung

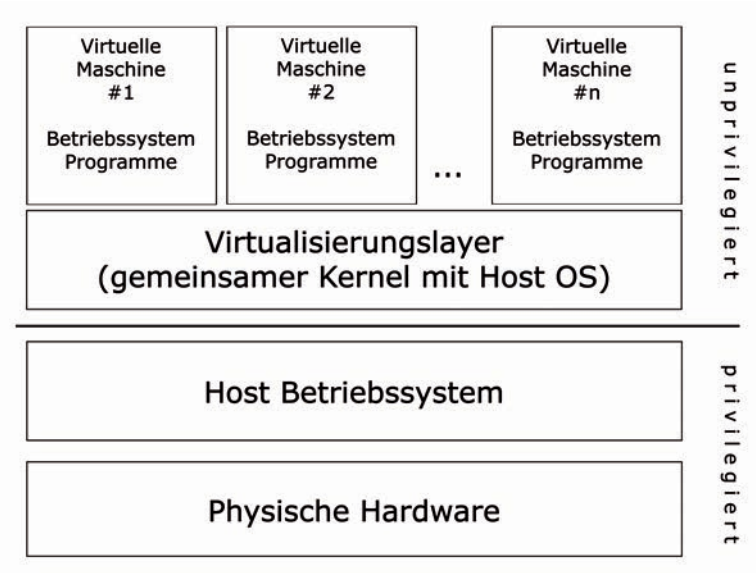

Abbildung 2.3: Aufbau Betriebssystem-Virtualisierung

Hier wird die Virtualisierung vom Betriebssystem übernommen, das heißt der Host und die Gast-Systeme laufen zwar in unterschiedlichen Umgebungen (so genannten Containern), verwenden aber den selben Kernel. Die Gast-Systeme sind somit keine vollständigen Betriebssystem-Virtualisierungen, sondern eine Gruppe von eng miteinander verbundenen Prozessen, die im Benutzermodus laufen. Hier wird das Prinzip

des Sandboxing (siehe Kapitel 3.3) angewandt. Diese Art der System-Virtualisierung hat die höchste Performanz bzw. höchste Effizienz – es muss keine Hardware emuliert werden – der Nachteil ist, dass die Gast-Systeme weniger stark voneinander isoliert sind. Der schematische Aufbau von Betriebssystem-Virtualisierung ist in Abbildung 2.3 dargestellt.

2.4.3 Voll-Virtualisierung

Wenn von Virtualisierung gesprochen wird, ist normalerweise von Voll-Virtualisierung die Rede. Sie ist der Inbegriff der Virtualisierung. Wird kein Hardware-Support unterstützt, so wird der Kernel-Code des Gast-Systems "on the fly" umgesetzt. Diese Virtualisierungsart existiert in den Varianten bare-metal (Abbildung 2.4) und hosted (Abbildung 2.5).

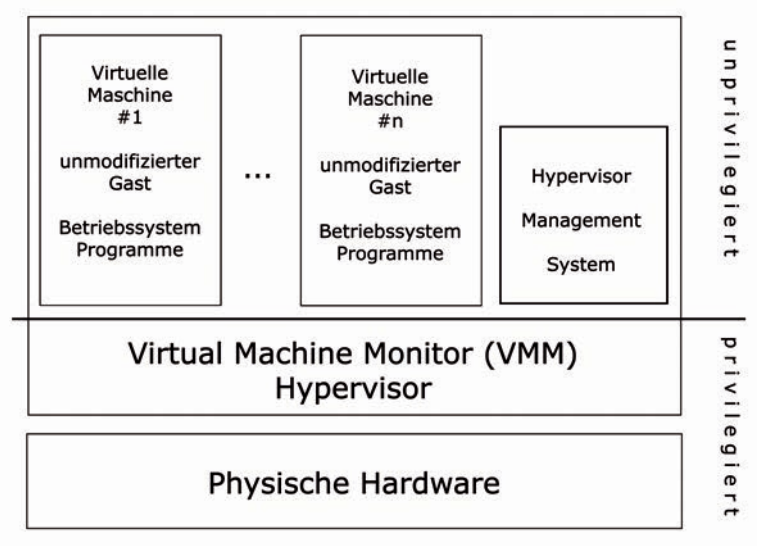

Abbildung 2.4: Aufbau bare-metal Voll-Virtualisierung

Vorteile sind die gute Performanz, abhängig von der Anzahl der laufenden Gastsysteme bzw. von der Lastverteilung und die hohe Flexibilität, da Betriebssysteme von unterschiedlichen Anbietern ausgeführt werden können. Der Hypervisor bietet bei dieser Virtualisierungsart eine komplette VM an, welche die selbe Architektur wie der

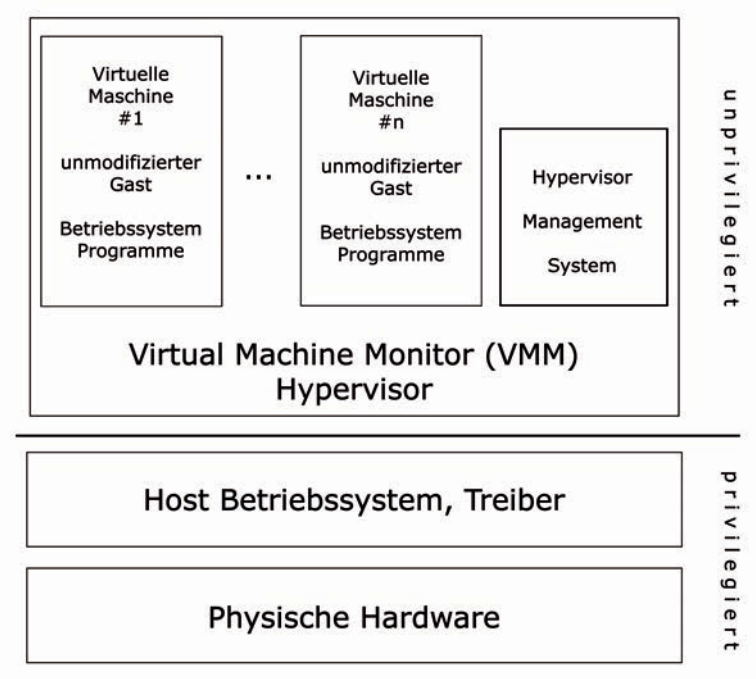

Abbildung 2.5: Aufbau hosted Voll-Virtualisierung

Host hat, die das Ausführen unmodifizierter Gäste erlaubt. Als Nachteil wird hier die Transparenz gesehen, da die Gäste nicht wissen, dass sie virtualisiert werden.

2.4.4 Para-Virtualisierung

Klassische Hardware, die nach der x86-Architektur arbeitet, ist nicht in der Lage, Virtualisierungsstrategien umzusetzen. Grund dafür ist, dass der Prozessor Anweisungen unterschiedlich ausführt, je nachdem ob er im privilegierten Modus arbeitet oder nicht. Ein virtualisierter Prozessor kann nicht im privilegierten Modus ausgeführt werden, was nebenbei bemerkt auch große Sicherheitsrisiken mit sich bringt, wenn man ihm diese Rechte geben würde. In virtualisierten Umgebungen können die Kernels der Gast-Systeme nicht direkt auf ihren physischen Prozessor zugreifen, somit müssen die Instruktionen, die das möchten, abgeändert werden. Weiters ist eine Anpassung der Privilegierungen notwendig, wie in Abbildung 2.6 ersichtlich ist. Das Betriebssystem wird eine Ebene nach oben verschoben und der Hypervisor residiert in Ebene 0 mit

Vollzugriff und vollen Privilegien.

Die Para-Virtualisierung ist somit der bare-metal Voll-Virtualisierung ähnlich. Der schematische Aufbau ist in Abbildung 2.7 ersichtlich. Statt der Binärübersetzung der Kernel-Mode Befehle rufen die Gast-Systeme den Hypervisor direkt mittels so genannter Hypercalls auf, das bedeutet, es ist eine Zusammenarbeit bzgl. Privilegierungen zwischen den Gast-Systemen und dem Hypervisor nötig. Daher ist Para-Virtualisierung

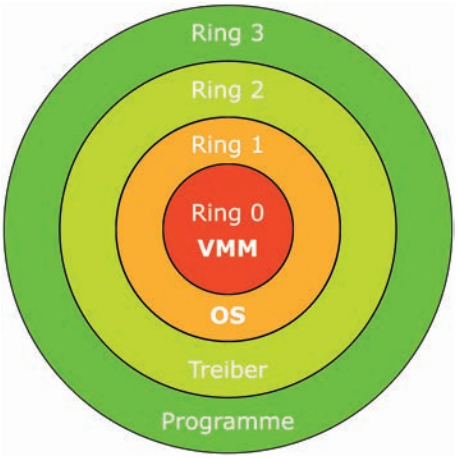

Abbildung 2.6: Privilegienring Para-Virtualisierung

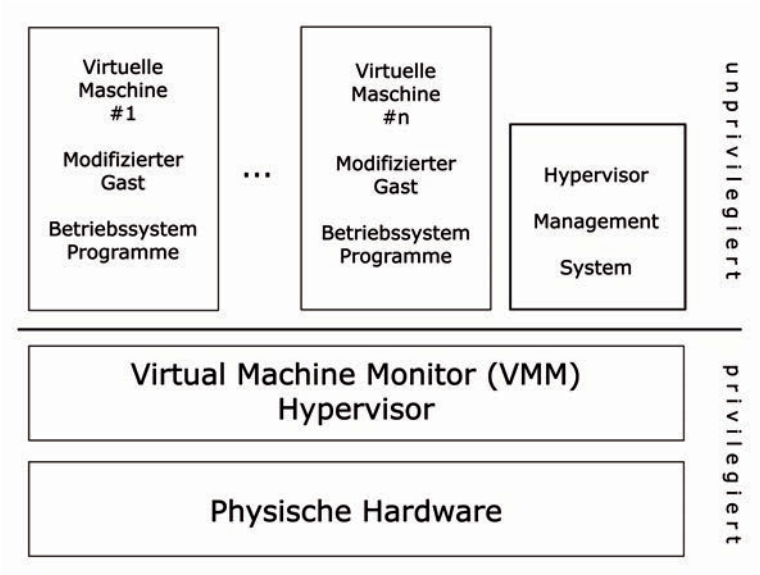

Abbildung 2.7: Aufbau Para-Virtualisierung

nur bei Betriebssystemen möglich, deren Source-Code quelloffen ist oder vom Hersteller so modifiziert wurde, dass Para-Virtualisierung unterstützt wird.

Unabdinglicher Vorteil ist die extrem gute Performanz, die laut [24] nur einen Verlust von 0,5% - 3,0% im Vergleich zur nativen Ausführung beträgt. Nachteil hingegen ist die Notwendigkeit der Modifikation der Gast-Betriebssysteme, um Hypercalls verwenden zu können, anstatt sensitiver hardwarespezifischer Instruktionen.

Obwohl sich die Architektur der x86-Prozessoren nicht optimal für Virtualisierung eignet, gibt es dennoch verschiedene Ansätze, um Virtualisierungen zu ermöglichen. Neben der Anpassung des Gast-Betriebssystems gibt es auch das Verfahren der Hardware-Prozessorerweiterung bzw. der "Binary Translation". Hier wird ein zusätzlicher Ring eingeführt, der eine Ebene "-1" bezeichnet. Vorteil davon ist, dass die üblichen Ringe 0 bis 3 unverändert verwendet werden können und der Hypervisor die Kontrolle über die Hardware weiter behält. Der modifizierte Privilegienring ist in Abbildung 2.8 ersichtlich.

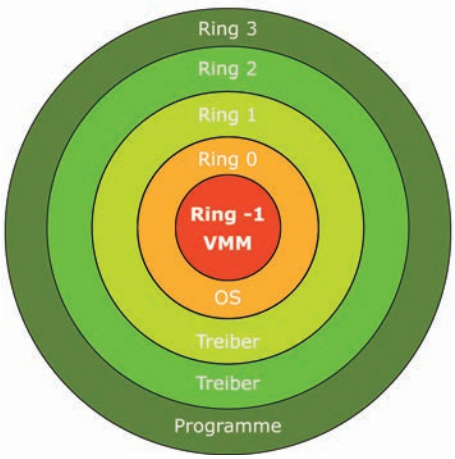

Abbildung 2.8: Privilegienring Hardwareerweiterung

Vorgehensweise bzw. Ziel: Bei der System-/Servervirtualisierung kann man nach der Vorgehensweise unterscheiden, welches Virtualisierungsziel erreicht werden möchte:

- **Partitionierung/Virtualisierung:** Auf einem physischen Rechner sollen viele logische Rechner laufen, also Virtualisierung, wie man sie herkömmlich versteht.

- **Server Aggregation/Grid Computing/Clustering:** Mehrere physische Ressourcen sollen zu einer logischen Ressource zusammengefasst werden, also das Bündeln von Ressourcen.

2.5 Hypervisor / Virtual Machine Monitor (VMM)

Es stellen sich die Fragen "Welche Ressourcen müssen geteilt bzw. virtualisiert werden?" und "Wer ist dafür zuständig?". Die Antworten finden sich einerseits in den Ressourcen, die benötigt werden (Prozessor, Speicher, Netzwerk und die Geräte), andererseits im Hypervisor, einer Art Verwaltungszentrale für die virtualisierten Umgebungen. Der Hypervisor verwaltet zum einen die Ressourcen, die aufgeteilt werden müssen, zum anderen erlaubt er es, mehrere Betriebssysteme auf einer Plattform laufen zu lassen. Garfinkel definiert einen VMM wie folgt:

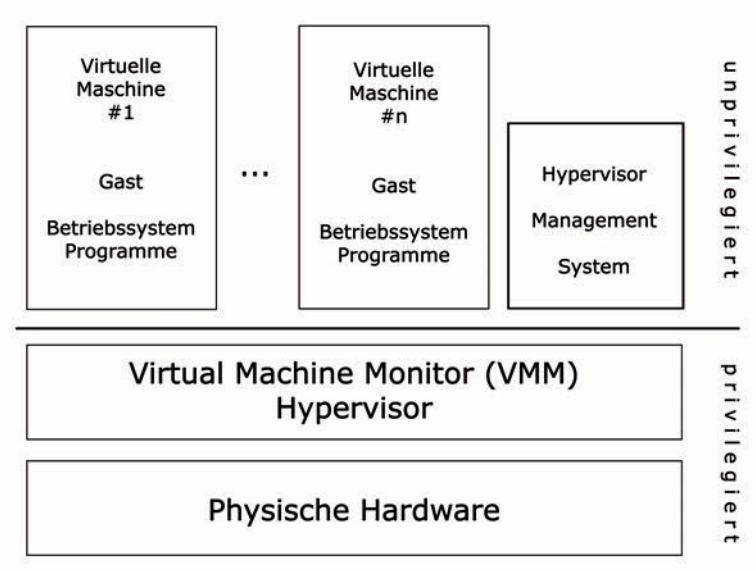

Abbildung 2.9: Aufbau bare-metal VMM

"A virtual machine monitor (VMM) provides a layer of software between the operating system(s) and hardware of a machine to create the illusion of one or more virtual machines (VMs) on a single physical platform. [...] A VMM's

central role is providing secure isolation. [...] In essence, a virtual machine monitor is nothing more than a microkernel with a hardware compatibility layer." [13]

Unterschieden werden kann der Hypervisor nach seiner Installationsart, bare-metal oder hosted:

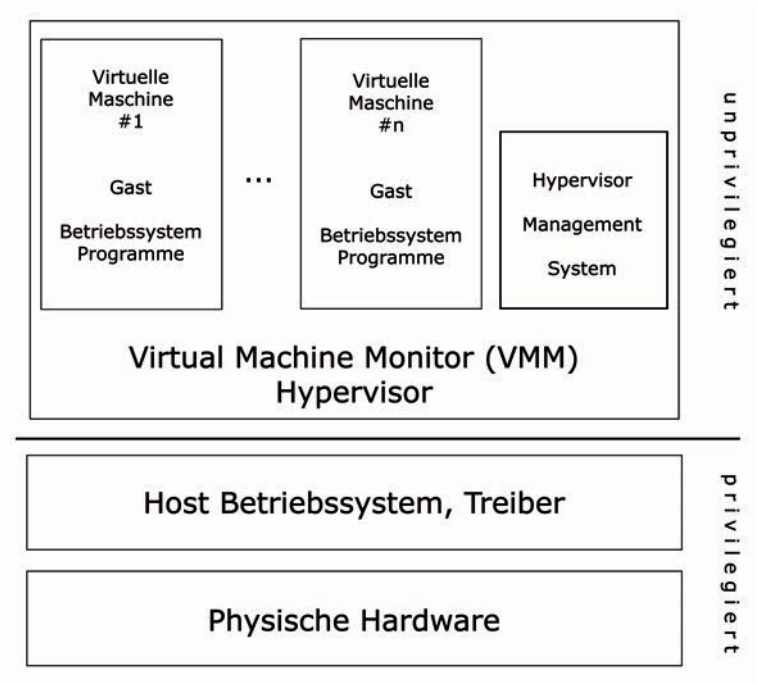

Abbildung 2.10: Aufbau hosted VMM

Bare-Metal (Typ 1): Der Hypervisor setzt direkt auf der Hardware auf und wird im privilegierten Modus ausgeführt. Alle virtualisierten Gäste laufen in höheren Leveln, die über dem Hypervisor angesiedelt sind. Ermöglicht wird das Aufsetzen direkt auf der Hardware mit einem speziellen Kernel, der Virtualisierungs-Funktionen implementiert. Die Verwaltung des Hypervisors bzw. dessen Management-Systems wird ebenfalls auf dem bare-metal Hypervisor ausgeführt, also eigentlich auf der selben Ebene wie die virtualisierten Gast-Betriebssysteme. Die Verwaltung der Hardware muss der Hypervisor übernehmen. Das Management-System hat aber direkten Zugriff auf die Hardware. Der Aufbau ist in Abbildung 2.9 dargestellt.

Hosted (Typ 2): Bei diesem Ausführungstyp läuft der Hypervisor als herkömmliches Programm in dem Betriebssystem, in dem es installiert wurde. Die Hardware-Ressourcen müssen vom Betriebssystem zur Verfügung gestellt und verwaltet werden. Die VMs erhalten nur über spezielle Treiber Zugriff auf die Hardware. Der Aufbau ist in Abbildung 2.10 dargestellt.

3 Sicherheitsaspekte

In diesem Kapitel werden allgemeine Überlegungen zur Sicherheit bei Virtualisierungen erläutert und Malware im allgemeinen sowie die Funktionsweise von Malware erklärt. In weiterer Folge wird auf das Prinzip des Sandboxings eingegangen und erläutert, warum es für Virtualisierungssicherheit von besonderer Bedeutung ist. Abschließend werden Sicherheitsüberlegungen zu den einzelnen Virtualisierungsarten besprochen.

3.1 Allgemeine Überlegungen zur Sicherheit

Wie bereits erwähnt, gibt es heutzutage mehr Gründe als jemals zuvor, warum Unternehmen die Vorteile von Virtualisierung nutzen. Ein großes Missverständnis beim Umstieg von physischen auf virtuelle Umgebungen besteht darin zu glauben, dass die aktuellen Sicherheitsmaßnahmen nun nicht mehr benötigt werden, weil Virtualisierung eine weitere Sicherheitsebene zum Schutz umsetzt. Anbei werden ein paar allgemeine Überlegungen zu Sicherheitsbedenken gebracht, die nicht weit her geholt sind und durchaus ihre Berechtigung haben, hier aufzuscheinen. Die Bereichsüberschriften in diesem Abschnitt sind jeweils aus [26] entnommen.

All it takes is the infiltration of just one machine: Wenn auf virtualisierte Umgebungen umgestellt wird, müssen sich die Verantwortlichen immer vor Augen halten, dass es genügt, wenn nur eine von mehreren auf einem Server laufenden VMs von außen infiltriert wird. Ein großer Nachteil bei Virtualisierungen ist somit die Sicherheit in diesem Bezug, da es sich beim VMM in gewisser Weise um einen SPOF (Single Point of Failure) handeln kann.

The differences between virtual and physical are becoming less and less: Ein Rechner ist ein Rechner, egal ob physisch oder virtuell – dementsprechend müssen auch VMs aktuell gehalten und mit einem Virenschutz versehen werden.

The failure of one machine can cripple a business like never before: Wenn früher ein Server ausfiel, gab es meist keine guten Backup-Strategien oder Möglichkeiten zur Ersetzung und die Problembehebung beanspruchte ein hohes Zeitkontingent. Diese Probleme haben sich mit Virtualisierung zwar vermindert, ein neu auftretendes Problem ist aber, dass mehrere Dienste bzw. VMs auf einem physischen Gerät laufen. Fällt ein physischer Rechner aus, auf dem wichtige Dienste laufen, tut sich ein neuer SPOF auf.

Keep all machines up-to-date: Alle Betriebssysteme sollten immer mit den neuesten Updates und Patches versehen sein. Abhilfe können hier so genannte Virtualisierungs-Management-Programme bieten, welche dafür sorgen, dass die VMs immer auf dem aktuellen Stand bleiben.

Keep applications to a minimum: Erfahrene Administratoren und auch Nutzer wissen, dass sich viele installierte Programme negativ auf die Performanz von Systemen auswirken können. Es gilt die Anzahl der installierten Programme, sowohl bei physischen als auch bei virtualisierten Rechnern, gering zu halten.

Use firewalls on virtual machines: Auch wenn die gesamte Unternehmensinfrastruktur durch eigene Firewalls geschützt wird, schadet es nicht, auf jedem einzelnen Rechner die betriebssystemeigene Firewall aktiviert zu lassen. Neben der Gefahr von aussen, die damit minimiert wird, unterbinden Firewalls auch das Risiko der Ausbreitung von Malware auf infizierten Maschinen.

Use Anti-Virus software on virtual machines: Wie der Name schon sagt, sollten neben aktivierten Firewalls auch aktuelle Virenschutzprogramme auf allen Maschinen installiert sein.

Don't browse the Internet from a physical host machine: Damit ist gemeint, niemals von einem Rechner aus im Internet zu surfen, welcher VMs ausführt. Entweder wird ein einzelner Arbeitsplatzrechner verwendet oder eine VM, die

speziell für das Internetsurfen ausgelegt wurde (beispielsweise mit verstärkten Sicherheitsrichtlinien). Da das Internet der potentiell gefährlichste Ort ist, um sich einen Virus oder Schadsoftware einzufangen, sollte dieser Gedankengang nicht vernachlässigt werden.

Harden your physical machines host operating system: Neben der permanenten Wartung der einzelnen virtuellen Umgebungen muss natürlich auch das Betriebssystem der physischen Maschine aktuell und sicher gehalten werden. Physische Maschinen haben trotz Virtualisierung immer noch die selben Schwachstellen wie zuvor.

Power down both virtual and physical unused machines: Jede Maschine bietet eine Angriffsfläche für Attacken. Darum sollen alle Maschinen, die gerade keine Aufgabe erfüllen oder nicht benötigt werden, abgeschaltet werden, um die Anzahl potentieller Angriffsziele zu minimieren.

Disable unused hardware: Dasselbe gilt auch für unbenutzte Hardware. Wenn bei einer VM die Soundkarte oder USB-Ports nicht benötigt werden, sollten diese auf jeden Fall deaktiviert werden.

Monitor both host machine and physical machine logs: Log-Dateien beinhalten oft wichtige Informationen über das Verhalten von Computern bzw. Programmen oder bieten Aufschluss darüber, ob Programme ungewöhnliches Verhalten zeigen oder sich irgendwo Sicherheitslücken auftun. In der Regel ist es üblich, Log-Dateien von physischen Rechnern zu überprüfen. Von diesem Vorgehen sollte auch bei VMs nicht Abstand genommen werden.

Start-up the virtual machine without connecting it to the network: Beim ersten Starten einer Maschine sollte die Netzwerkkarte deaktiviert werden, um allfällige Einstellungen zu tätigen und um die Stabilität der Maschine zu testen.

Testing and development: Ein großer Vorteil von VMs ist ihre Nutzung als Testumgebung, sowohl für normale Software als auch für Schadsoftware-Szenarien. Es können auf einfachstem Wege mehrere verschiedene Umgebungen geschaffen und Szenarien getestet werden. Tester haben die Möglichkeit, verschiedene System-

zustände (und auch Sicherheitszustände) auf unterschiedlichen VMs zu testen. Wird bei diesen Maschinen nicht auf Sicherheit geachtet, können sie schnell zu einem Pool von infizierten Systemen werden und die restliche Infrastruktur gefährden.

Difficulty of keeping track of what each virtual machines role is: Da es oft ein leichtes Vorgehen ist, beliebig viele (sofern es die Ressourcen erlauben) VMs zu erstellen, sollte dennoch auf Übersichtlichkeit geachtet werden. Zum einen um zu wissen, welche Funktionalitäten und Dienste die jeweilige Maschine leistet, zum anderen, damit die Fehlersuche nicht zu viel Zeit in Anspruch nimmt. Zum einen um zu wissen, was welche Maschine eigentlich macht, zum anderen um bei der Fehlersuche nicht zu viel Zeit zu verlieren.

Preventing virtual machines from being copied by unauthorized personnel: Ein Punkt, der wohl den meisten Betroffenen nicht bewusst ist. Da eine VM eigentlich nur eine Datei auf einer Festplatte ist, muss darauf geachtet werden, dass diese Festplattendatei nicht von unauthorisierten Personen vervielfältigt wird. Sollte es einem Eindringling gelingen, Zugriff auf die Festplattendatei zu erlangen, so kann er sich in Ruhe Zugang verschaffen und benötigt dann beim direkten Angriff auf die laufende Maschine im schlimmsten Fall nur so wenig Versuche um erfolgreich zu sein, dass dies nicht einmal auffällt.

Dies ist nur ein Ausschnitt einer großen Menge an Überlegungen, die bestimmt niemals vervollständigt werden kann. Weitere Überlegungen betreffen beispielsweise die Nachteile von nützlichen Zusatzfunktionen. In [13] wird das Problem von "Rollback"-Funktionen genannt: das Wiederherstellen von alten Systemzuständen ist insofern problematisch, da hier Zustände eingenommen werden können, welche das System angreifbar für bereits beseitigte Probleme macht. Unter Umständen wird auch ein Zustand wieder eingenommen, bei dem das System von Schadsoftware infiltriert ist.

3.2 Malware

In diesem Abschnitt soll ein kurzer Exkurs bzw. eine Einführung und ein Überblick zu den Arten von Schadsoftware, Wege zur Infektion, Verschleierungsmechanismen und Ansätzen zur Suche von Malware gegeben werden. Der Großteil dieses Kapitels sowie die Strukturierung der Inhalte beruht auf den Informationen und dem Aufbau, die in "Kapitel II. Malware" im Buch "Virtual Machine based Rootkits" [15] angeführt sind.

> *"Der Begriff* **Malware** *setzt sich aus den ersten drei Buchstaben des englischen Wortes* **mal***icious (boshaft, schlecht, schlimm) und den letzten vier Buchstaben von Soft***ware** *zusammen. Bei Malware handelt es sich um Software, welche mit dem Ziel entwickelt wurde, auf einem Computer unerwünschte und in der Regel schädliche Aktionen durchzuführen. Unter dem Begriff Malware werden unterschiedliche Arten von Schadsoftware wie z.B. Viren, Würmer, Trojanische Pferde und Rootkits zusammengefasst."* [15]

Mit Hilfe von Malware ergeben sich Möglichkeiten, virtualisierte Systeme zu infiltrieren und zu schädigen, wie in den folgenden Kapiteln näher beschrieben wird.

3.2.1 Arten von Malware

Viren: Dies sind Programme, die sich zum einen selbst vervielfältigen, zum anderen Schadroutinen ausführen. Viren müssen im Normalfall zumindest einmal bzw. das erste Mal vom Benutzer ausgeführt werden, bevor sie aktiv werden können. Da Viren die älteste Art von Malware darstellen, wird der Begriff Virus oft fälschlicherweise als Synonym für Malware verwendet.

Würmer: Würmer sind Viren sehr ähnlich. Sie versuchen aber nicht, sich nur am lokalen System zu replizieren, sondern auch über das Netzwerk in andere Rechner einzudringen. Hier ist keine Aktion von Benutzerseite erforderlich, da Würmer oft Fehlkonfigurationen oder Schwachstellen von Systemen ausnutzen (so genannte "Exploits").

Backdoor: Ist mittels einer Malware eine Backdoor (Hintertür) etabliert worden, so wird diese von Dritten genutzt, um Kontrolle über den infizierten Rechner zu gewinnen. Backdoors ermöglichen Zugriff zum System meist außerhalb bzw. unter Umgehung der vorhandenen Sicherheitsfunktionen.

(Distributed) Denial-of-Service-Attacke: Diese ist prinzipiell keine Malware, aber ein Vorgehen, welches durch die Infizierung mittels Malware erreicht wird. Funktionsweise einer DoS-Attacke ist, ein System von vielen externen Stellen mit so vielen Anfragen zu überhäufen, dass es die Arbeit einstellt, da die hohe Anzahl an Anfragen nicht mehr abgearbeitet werden kann.

Bot-Net: Hier handelt es sich um Netzwerke aus prinzipiell unabhängigen Rechnern, die von Dritten genutzt werden, um beispielsweise Denial-of-Service-Attacken durchführen zu können. Sie nutzen die Anbindung der verschiedenen Rechner um diese Funktionalität zu gewährleisten.

Trojaner: Diese Art von Malware täuscht den Benutzer, in dem sie vorgibt, nützliche Dienste anzubieten. Bei Trojanischen Pferden (meist nur als "Trojaner" bezeichnet) gibt es sowohl Programme, die tatsächlich einen Dienst anbieten und solche, die keinen Dienst anbieten. In jedem Fall aber führt ein Trojaner Aktionen durch, die der Benutzer nicht bemerkt und diesen schädigen. Unterschieden werden können noch jene Arten von Trojanern, die entweder eine schädliche Zusatzfunktionalität ausführen, wenn das vermeintlich nützliche Originalprogramm gestartet wird, oder jene, die ein fremdes Programm manipuliert haben und sich so in das System einschleusen. Ziel von Trojanern ist es meist, sensible Daten des Benutzers auszuspionieren oder eine Backdoor für weiteres Eindringen bereit zu stellen. Solche Backdoors werden häufig dafür genutzt, den infizierten Rechner als Bot für Denial-of-Service-Attacken in Botnets zu missbrauchen. Trojaner besitzen in der Regel keine Replizierungsfunktionalität.

Spyware/Adware: Diese Art von Malware dient zum Ausspionieren des Benutzers. Spyware konzentriert sich hierbei auf das Verhalten des Benutzers (beispielsweise die Eingabe von Passwörtern über die Tastatur), und leitet anschließend die Daten an Dritte weiter. Adware hingegen wird häufig benutzt, um Daten für

Werbezwecke oder Marktforschung zu erhalten (beispielsweise die Frage, welche Seiten dieser Nutzer im Internet besucht, was sind seine Interessen, ...). Diese Art der Malware führt prinzipiell keine Schadroutinen aus.

Scareware: Diese Methode wird angewandt, um den Benutzer zu verängstigen und ihn zur Installation von Schadsoftware, die sich als nützliche Software ausgibt, zu verleiten.

Ransomware: Ransomware ist darauf ausgelegt, den Zugriff auf das System zu blockieren und es nur mittels Kennwort wieder freizuschalten. Das Kennwort erhält man nur durch Zahlung eines gewissen Geld-Betrages. Unter Ransomware fallen auch jene Schadprogramme, die Dateien verschlüsseln und die Entschlüsselung nur gegen Zahlung eines Betrages ermöglichen.

Rootkits: Diese stellen eine neue Art der Bedrohung dar und werden in Kapitel 5 näher behandelt. Rootkits existieren als software-virtualisierende und als hardware-virtualisierende Variante. Ihr Ziel ist es, Schadroutinen außerhalb des Betriebssystems einzurichten, um so für herkömmliche Schutzmechanismen unsichtbar bzw. unauffindbar zu bleiben.

Neben diesen Hauptarten von Malware gibt es noch folgende weitere Arten:

Dialer: Diese wählen sich bei Mehrwertnummern ein. Problematisch war dies vor allem zu Zeiten der Einwahlmodems, heute ist dies bei den meisten Dauer-Internetanbindungen alleine aus technischer Sicht oft nicht mehr möglich.

Grayware: Hier handelt es sich um Malware, die nicht direkt Systemfunktionen beeinträchtigt, aber trotzdem als unerwünschter Gast im System anwesend ist.

Rogueware: Diese Art von Malware gibt ich als Schutzprogramm aus, installiert in Wahrheit aber Schadsoftware.

3.2.2 Wege zur Infektion eines Systems

Es existieren multiple Verfahren, derer sich Malware bedienen kann um einen Rechner zu infizieren. Wissen über diese Verfahren ist auch für die Erstellung von Abwehrmechanismen notwendig. Im Folgenden werden die häufigsten bzw. bekanntesten Infektionswege beschrieben:

Infektion des Bootsektors: Diese Methode hat das Ziel, den Bootsektor einer Festplatte, welcher meist im MBR (Master Boot Record) gespeichert ist, so zu verändern, dass zuerst die Malware ausgeführt wird und diese erst danach das Betriebssystem im alten MBR startet. Somit hat die Malware mehr oder minder die volle Kontrolle über das ausgeführte Betriebssystem.

Infektion über Dateien: Bei diesem Infektionsweg werden Dateien infiziert, indem diese verändert werden. Die Methoden hierbei reichen vom einfachen und sehr auffälligen Überschreiben von ganzen Dateien, über das Verändern von Anfangs-/Endsektoren von Dateien oder das Einbetten von Schadcode in stark fragmentierten Mustern, bis hin zu komplexen Verschlüsselungsarten.

Infektion über Win32: Bekannterweise gibt es kaum Malware für Linux- oder MAC OS-Betriebssysteme. Grund dafür ist die starke Verbreitung von Microsoft Windows Betriebssystemen. Es existieren multiple Möglichkeiten, über die von Microsoft bereitgestellten Portable-Executable-Dateiformate (PE) das Betriebssystem zu infizieren, auf die hier aufgrund des hohen Umfangs der genauen Erklärung der Funktionsweise nicht näher eingegangen wird.

3.2.3 Mechanismen zur Verschleierung

Damit Malware effizient arbeiten kann, muss sie sich vor Schutzmechanismen und Sicherheits-Programmen verstecken können. Die Wege zur Verschleierung von Malware werden hierbei immer komplexer. Diese Tatsache führt zu neuen Herausforderungen für Schutzprogramme. Im Folgenden werden die häufigsten Verschleierungs-Mechanismen gelistet:

Garbage Insertion: Es werden Operationen, die keine Auswirkung haben, in den ursprünglichen Virencode eingesetzt. Damit ändert sich zwar die Signatur (siehe Kapitel 3.2.4) der Malware, was es Virenscannern schwerer macht diese zu finden, die Funktionalität bleibt aber gleich.

Register Renaming: Bei dieser Methode werden nur die adressierten Register verändert, was die Signatur ändert und es Virenscannern erschwert, die Malware aufzuspüren.

Code Reordering: Hier werden, erneut um die Signatur des Programmes zu verändern, lediglich einzelne Befehle in eine andere Reihenfolge gebracht. Die korrekte Funktionalität wird mit gesonderten Sprungbefehlen beibehalten.

Entry Point Obfuscation: In der Regel untersuchen Anti-Malware-Programme hauptsächlich Anfang und Ende von Dateien auf Schadsoftware, da eine Analyse aller Programme und eine vollständige Untersuchung aller Dateien zu zeitaufwändig wäre. Dieses Verhalten nutzen Malware-Hersteller um eine neue Art der Verschleierung zu erreichen: der Startpunkt von Malware wird beliebig im Inneren einer Datei bzw. eines Programmes gesetzt. Von diesem Punkt aus wird dann die Malware gestartet. Dieses Vorgehen macht es Virenscannern sehr schwer, Schadsoftware zu finden. Andererseits ist hier aber nicht garantiert, dass die Malware zum Einsatz kommt, da die Möglichkeit besteht, dass der geänderte Einstiegspunkt gar nicht erst aufgerufen wird.

Komprimierung: Die Komprimierung von Malware hat zwei große Vorteile: zum einen kann die Dateigröße verringert werden, dies führt dazu, dass das Schadprogramm weniger auffällig ist; zum anderen ändert sich je nach Komprimierungsverfahren die Signatur der Malware.

Geteilte Malware: Bei diesem Verfahren wird die Nutzlast des Schadprogrammes auf mehrere Datenpakete verteilt und am Zielsystem wieder zusammengesetzt. Hier kann es lange dauern, bis das Schadprogramm vollständig auf dem Zielsystem angelangt ist. Durch diese Tatsache wird die Methode ineffizient. Die Auffindbarkeit für Antivirenprogramme ist aber dementsprechend schwer.

Verschlüsselung: Ein Weg um Malware noch schwerer auffindbar zu machen ist, sie zu verschlüsseln. Zum einen wird dadurch wieder die Signatur verändert, zum anderen erschwert dieser Ansatz die nähere Untersuchung der Funktionsweise, da verschlüsselter Code meist nicht rückinterpretiert werden kann (Verhindung von Reverse Engineering).

Polymorphe Malware: Diese Art verwendet neben unterschiedlichen Verschlüsselungsverfahren auch viele unterschiedliche Chiffrierschlüssel, was es enorm erschwert, eine einheitliche Signatur der Schadsoftware zu erstellen, da sich diese mit jeder Infizierung und Entschlüsselung der Software ändert. Einziger Anhaltspunkt hier ist, dass die eigentliche Schadroutine, der Kern der Malware, immer gleich bleibt.

Metamorphe Malware: Der Unterschied zur polymorphen Malware ist hier, dass die metamorphe Malware nicht nur die Schlüssel und Entschlüsselungsroutinen bei jeder Anwendung ändert, sondern auch den Kern der Schadroutine jedesmal selbst modifiziert. Diese Art der Verschleierung stellt eine der komplexesten und am schwierigsten lösbaren Methode für Antivirenprogramme dar.

3.2.4 Suche nach Malware und Analyseverfahren

Prinzipiell kann jede Schadsoftware entdeckt werden, wenn genug Zeit und Ressourcen zur Verfügung stehen. Da diese Voraussetzungen zumeist nicht gegeben sind, muss man sich anderer Methoden bedienen, um eine gute Chance zu haben, Malware verlässlich zu finden. Nachfolgend werden die bekanntesten Methoden zur Suche von Malware kurz erläutert:

Signaturen: Wie im vorherigen Unterkapitel beschrieben, werden Signaturen von Malware verwendet, um diese schneller aufspüren zu können. Erstellt werden Signaturen anhand der Byte-Folge der Schadroutinen. Hier kann es auch vorkommen, dass zufälligerweise Byte-Folgen von normalen Anwendungen ähnliche oder gleiche Signaturen ergeben wie Malware, was zu einer Falschklassifizierung (false-positives) führt.

Neuronale Netze: Diese Methode ist noch nicht ausgereift bzw. befindet sich noch im Forschungsstadium, verspricht aber gute Ansätze für die Detektion von Malware. Das Prinzip neuronaler Netze ist es, Zusammenhänge aus bereits bekannten Dateien und Arten von Schadsoftware zu erlernen, um damit noch unbekannte Malware – beispielweise aufgrund ihres Verhaltens – erkennen zu können. Der Ansatz bedient sich an Methoden aus der Biophysik und Machine Learning.

Sandboxing: Dieser Ansatz wird in Kapitel 3.3 näher behandelt.

Arbeitsweise von Betriebssystemen: Betriebssysteme sorgen in gewissem Maße selbst für Schutz – beispielsweise indem sie Maßnahmen zur Autorisierung ("Wer hat welche Rechte auf welche Ressourcen?") und Authentifizierung ("Ist der Benutzer der, der er vorgibt zu sein?") vornehmen. Dieses Vorgehen dient nur bedingt zum Schutz vor Malware, da mit Schadsoftware infizierte Programme und Dateien zumeist mit den Rechten des aktuellen Benutzers ausgeführt werden. Einzige Abhilfe bzw. Unterstützung bietet hier die Einschränkung von Rechten: werden Benutzern, die nur wenig wichtige Aufgaben erledigen (im Kontext von benötigten Rechten), alle signifikant Rechte auf angreifbare Ziele entzogen, kann das Infektionsrisiko maßgeblich gesenkt werden. Hierzu zählen beispielsweise Benutzeraccounts für reines Internetsurfen – ein Benutzer der nur diese Aktivität ausführt benötigt keine Administrator-Rechte.

Integritätsprüfung: Hier wird die Tatsache ausgenutzt, dass sich Viren meist selbst reproduzieren und Änderungen an den infizierten Originaldateien vornehmen. Sicherheitsprogramme, welche die Integrität von Dateien auf Veränderungen untersuchen, können auf diesem Wege Malware identifizieren. Der einfachste und auch verbreitetste Weg der Integritätsprüfung ist die Erstellung einer Checksumme der Datei mittels Cyclic-Redundancy-Check (CRC)[5], welche sich ändert, wenn auch nur ein Byte der Originaldatei verändert wurde.

Verhaltens-Blockierer: Die sogenannten Behavior-Blocker arbeiten nach einem ganz anderen Prinzip zur Malware-Erkennung. Sie überwachen das Verhalten von Dateien und Suchen nach Mustern und Auffälligkeiten. Besonders das Ausführen

[5] `http://www-stud.rbi.informatik.uni-frankfurt.de/~haase/crc.html`

und Ändern von Programmen, Änderung von Systemeinstellungen und der Aufbau von Netzwerkverbindungen werden überwacht.

Heuristische Verfahren: Bei diesem Verfahren wird versucht, verdächtige Programme mittels einer Art Schätzung (basierend auf spezifischen Merkmalen wie Eigenschaften oder Verhaltensmuster von Schadsoftware) zu finden. Nach ermittelter Erstinformation werden weitere Schritte unternommen, das Suchergebnis zu präzisieren. Heuristische Verfahren können sowohl statisch als auch dynamisch arbeiten, es wird hier aber nicht näher auf deren Funktionsweise eingegangen. Aktuell liegt die Erfolgsquote von heuristischen Scannern beim Auffinden von Malware zwischen 70% und 80%.

3.3 Sandboxing

Unter einer Sandbox (Sandkasten) versteht man einen abgegrenzten bzw. isolierten Bereich in einem System, in dessen Innerem keine Aktion eine Auswirkung auf die Außenwelt hat. Sandboxing wird zum Testen von Software verwendet, um sicher zu gehen, dass keine signifikant wichtigen Systemteile beeinträchtigt werden.

Viel häufiger werden Sandboxes dafür verwendet, Malware zu erkennen und zu analysieren. Ziel dabei ist es, der Malware in einer Sandbox (beispielsweise ein vollständiges virtuelles Betriebssystem oder Teile davon) alle Ressourcen uneingeschränkt zur Verfügung zu stellen, um so das Verhalten der Software beobachten und analysieren zu können. Dabei ist es wichtig darauf zu achten, dass die Schadsoftware nicht aus der isolierten Umgebung ausbrechen und sich auf das Hostsystem ausbreiten kann. In diesem Sinne sollte besonders auf die Einschränkung von virtualisierten Netzwerkfunktionalitäten geachtet werden. Trotz der Vorteile von Sandboxes beim Testen von Schadsoftware gibt es auch Nachteile dieses Verfahrens [15]:

- Schadsoftware, die verschlüsselt ist oder welche Netzwerkschnittstellen emuliert, kann sich möglicherweise in einer Sandbox nicht produktiv entfalten, da die Umgebung zu langsam ist, somit schlägt das Auffinden fehl.

- Es existieren Arten von Schadsoftware, die erkennen können, ob sie in einer virtuellen oder nativen Umgebung ausgeführt werden. Werden solche Programme zu Untersuchungszwecken in einer VM ausgeführt, ist es sehr wahrscheinlich, dass die Software ihre Funktionalität nicht zeigt.
- Schadsoftware, die erst nach einer gewissen Zeit aktiv wird, kann nur bedingt in virtualisierten Umgebungen erkannt werden, da sich die Software meistens zum Zeitpunkt der Prüfung in einem inaktiven Modus befindet.
- Programmierer von Schadsoftware wissen über das Prinzip des Sandboxings sehr wohl Bescheid und können somit ihre Programme dahingehend konfigurieren, dass diese beispielsweise bei eingeschränkter Netzwerkverfügbarkeit, wie es eben bei Sandboxes üblich ist, gar nicht erst aktiv werden.

Im Folgenden wird auf das Verfahren des Jailbreakings eingegangen sowie ein Betriebssystem vorgestellt, welches das Konzept des Sandboxings für alle Prozesse und Anwendungen umsetzt.

3.3.1 Jailbreaking (chroot)

Bei dieser Funktion handelt es sich um einen Befehl auf unix-basierten Betriebssystemen, mit der das Rootverzeichnis[6] geändert werden kann. Dabei sind immer die Ausführungsrechte des aktuellen Prozesses bzw. aller zugehörigen Kindprozesse betroffen. Die damit erzeugte bzw. geänderte Umgebung wird als *chroot jail* bezeichnet, weil der Prozess, der innerhalb des *jails* läuft, normalerweise nicht auf Ressourcen ausserhalb zugreifen kann, sich also in einem Gefängnis befindet. Somit ist eine Art des Sandboxing innerhalb des Betriebssystems erreicht, welches aber prinzipiell nicht als Sicherheitsoption gesehen werden kann. Sie wird primär für die Initialisierung virtueller Umgebungen genutzt. Neben dieser Verwendung dient *chrooting* auch für isolierte Testumgebungen, für die Rechtetrennung oder, da auch Netzwerkdienste virtualisiert werden können, als Lockvogel für potentielle Angreifer oder Schadsoftware, so genannte *Honeypots*.

[6] In einem hierarchischen (Datei-)System ist dies stets die höchste Ebene bzw. das Stammverzeichnis, oft auch als Wurzel bezeichnet.

Chroot kann im Normalfall nur vom Administrator (root) ausgeführt werden. Sinn dessen ist es, dass unprivilegierte Benutzer mit wenig Rechten nicht die Möglichkeit erhalten sollen, eine chroot-Umgebung zu nutzen um beispielsweise Rechte vergeben zu können.

Obwohl die chroot-Methodik primär für Kommandozeilen-Anweisungen gedacht ist, gibt es die Möglichkeit, Programme mit grafischer Oberfläche in einer solchen Umgebung auszuführen (xhost[7], xNest, xchroot[8], etc.).

3.3.2 Qubes OS

Bei diesem von Joanna Rutkowska[9] 2011 erstmals vorgestelltem linux-basierten Betriebssystem wird der Ansatz "Security by Isolation" (Sicherheit durch Isolation) angewandt. Die Virtualisierung wird von einem Xen Hypervisor (siehe Kapitel 4.4) übernommen; die Betriebssystem- bzw. Benutzerumgebung basiert auf dem Linux Betriebssystem Fedora[10]. Details zum Projekt selbst findet man auf der offiziellen Webseite [11].

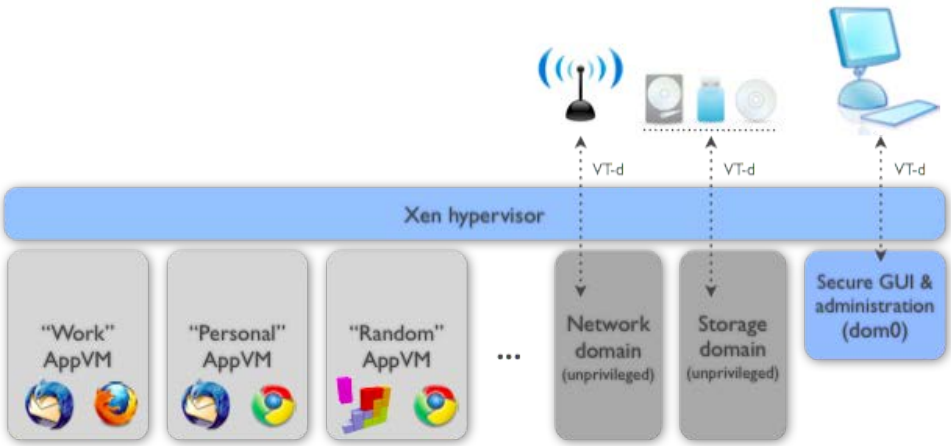

Abbildung 3.1: Darstellung Isolation bei Qubes OS

[7] http://theory.uwinnipeg.ca/XFree86/4.8.0/xhost.1.html
[8] http://www.elstel.org/xchroot/
[9] http://de.wikipedia.org/wiki/Joanna_Rutkowska
[10] www.fedoraproject.org
[11] www.qubes-os.org

Der Grundgedanke als Motivation zur Erhöhung der Sicherheit bzw. Anstoß zur Entwicklung dieses System ist, dass es kein Betriebssystem geben kann, das absolut fehlerfrei ist. Da jedes Betriebssystem aus unzähligen Quellcodezeilen besteht und noch viel mehr Hardware/Software Interaktionen stattfinden, bedarf es nur einer kleinen Unachtsamkeit im Code, um Malware eine Angriffsmöglichkeit bieten zu können. Darum bedient man sich bei Qubes der Idee, jede Aktion bzw. jeden Task oder Prozess in einer eigenen, isolierten VM auszuführen. Somit ist es nicht möglich, dass sich Schadsoftware, die beispielsweise aus einem Trojaner heraus versucht das System zu infizieren, auf andere Bereiche des Systems, beispielsweise einen Browser bei dem gerade Online-Banking Aktivitäten durchgeführt werden, ausbreitet. In Abbildung 3.1 ist die Isolation bei Qubes schematisch dargestellt

3.4 Sicherheit bei Prozess-Virtualisierung

Bei der Bibliotheks- bzw. Applikations-Virtualisierung wird kein ganzes Rechnersystem virtualisiert, sondern nur eine Ausführungsumgebung. Es wird also der Teil eines Betriebssystems bereitgestellt, der für die aktuellen Aktionen benötigt wird.

Die Vorteile, welche die Prozess-Virtualisierung bietet, schlagen sich auch in der Sicherheit nieder: Es ist keine Installation der Programme nötig, weshalb zum einen auch keine Inkompatibilitäten zwischen bereits installierten Anwendungen auftreten können (beispielsweise durch die Veränderung von Systemeinstellungen oder durch die Verwendung von gemeinsam genutzten Bibliotheken), zum anderen wird die Registrierung und das Dateisystem des lokalen Betriebssystems nicht verändert. Dies hat auch den Vorteil, dass nach einer Deinstallation der Anwendung keine Spuren mehr in der Registrierung zurückbleiben, wie es meist bei allen Anwendungen der Fall ist.

Weiters stellt dies einen Vorteil in Bezug auf die Sicherheit dar, da etwaige Schadsoftware sich durch einen Verbleib in der Registrierung oft Wege eröffnet, um später bzw. nach einer vermeintlichen Entfernung, wieder aktiv zu werden. Diese Möglichkeit hat Schadsoftware hier nicht, da bei der Prozess-Virtualisierung keine Backdoors auf diese Art hinterlassen oder eingeführt werden können.

Beispiele für Bibliotheks- bzw. Applikations-Virtualisierungen sind, wie in Abschnitt 2.3 bereits erwähnt, die VMware ThinApp und Wine, sowie Microsofts Application Virtualization (App-V)[12] oder Altiris SVS Symantec Endpoint Virtualization Suite[13].

3.5 Sicherheit bei System-/Servervirtualisierung

Nachfolgend wird auf Sicherheitsaspekte bei Desktop-Virtualisierungen, bei Virtualisierungssoftware und bei virtualisierungs-bezogenen Linux-Erweiterungen eingegangen und in weiterer Folge werden Sicherheitsprobleme aufgrund der Architektur der einzelnen Virtualisierungs-Technologien besprochen.

3.5.1 Desktop-Virtualisierungen

Einen weiteren Bereich an Virtualisierungs-Technologien, neben den bereits behandelten, stellt die Desktop-Virtualisierung dar. Ziel dieser ist es, Benutzern komplette Desktops zur Verfügung zu stellen, welche in zentralisierten Rechenzentren verwaltet werden. Mit dem vermehrten Aufkommen von mobilen Geräten (Smartphones und Tablets) erfährt dieser Bereich neue Anforderungen, die erfüllt werden müssen: nicht nur auf einem Standrechner oder einem Notebook soll einem Benutzer immer die selbe Arbeitsumgebung präsentiert werden, auch auf den vom Benutzer selbst mitgebrachten Geräten soll eine vertraute Arbeitsumgebung seitens der IT-Infrastruktur geschaffen werden, auf welcher in gewohnter Weise gearbeitet werden kann.

Die Anfänge der Desktop-Virtualisierungen liegen in der Anwendung von Terminals unter der Verwendung von Terminalservern. Diese wurden vermehrt in Unternehmen eingesetzt noch bevor der Personal Computer das primäre Arbeitsgerät wurde. Zu dieser Zeit war Hardware teuer und musste effizient genutzt werden. Ein Terminalserver bot Dienste und Programme an, welche an den einzelnen Terminals (Thin Clients) genutzt werden konnten. Hierbei war ein Terminal nicht viel mehr als eine Netzwerk-

[12] http://www.microsoft.com/en-us/windows/enterprise/products-and-technologies/mdop/app-v.aspx
[13] http://www.symantec.com/workspace-streaming

schnittstelle bzw. ein Gerät, an dem Maus, Tastatur, Netzwerkkabel und ein Bildschirm angeschlossen wurden.

Heute wird unter Desktop-Virtualisierung wieder die Bereitstellung von ganzen Desktops verstanden, die auf einem zentralen Virtualisierungsserver liegen. Ziel ist unter anderem der sichere Zugang von und zu jedem Gerät, mit dem die Desktop-Virtualisierung genutzt wird. Der große Vorteil in Bezug auf Sicherheit liegt in der zentralisierten Verwaltung. Es wird an einem Punkt die gesamte Software-Umgebung und die Betriebssystem-Aktualität überwacht.

Die größten Vorteile von Desktop-Virtualisierungen sind laut [35] in erster Linie die Einsparungspotentiale bei Infrastrukturkosten und die Möglichkeit zur zentralisierten Datenverwaltung und -sicherung.

Lösungen für Desktop-Virtualisierung sind beispielsweise Citrix XenDesktop, Microsoft Enterprise Desktop Virtualization und VMware View.

3.5.2 Sicherheit bei Virtualisierungssoftware

Stellvertretend für andere Derivate von Virtualisierungssoftware wurde Oracle VM VirtualBox[14] in Hinblick auf mögliche Sicherheitseinstellungen untersucht.

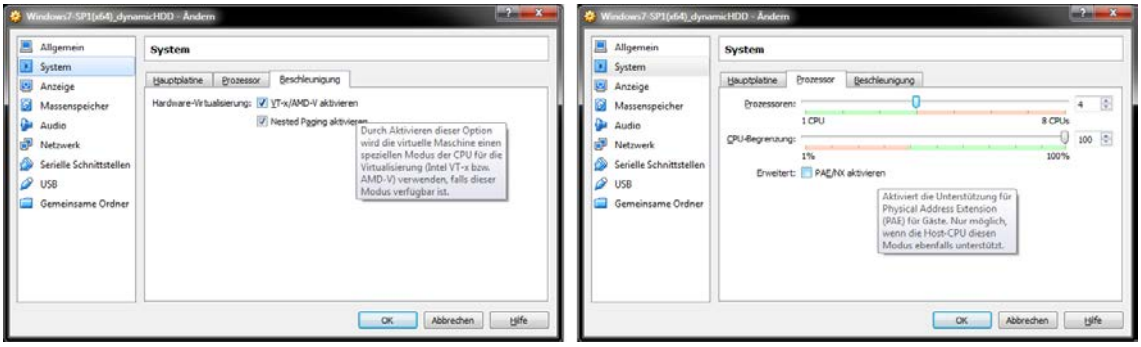

Abbildung 3.2: Einstellungen für Hardwarevirtualisierung und PAE

Wie zu erwarten war, gibt es keine spezifischen Konfigurationsmöglichkeiten für Si-

[14] https://www.virtualbox.org/

cherheit oder sicherheitsrelevante Aspekte, was dazu führt, dass der Benutzer selbst dafür zu sorgen hat, dass seine (sowohl physischen als auch die virtualisierten) Systeme sicher sind.

Lediglich Einstellungen zur Aktivierung von Hardware-Virtualisierungsunterstützung (Intel VT-x, AMD-V, Nested Paging) und zur Physical Adress Extension (PAE) stehen neben anderen Konfigurationsmöglichkeiten zur Verfügung (siehe Abbildung 3.2).

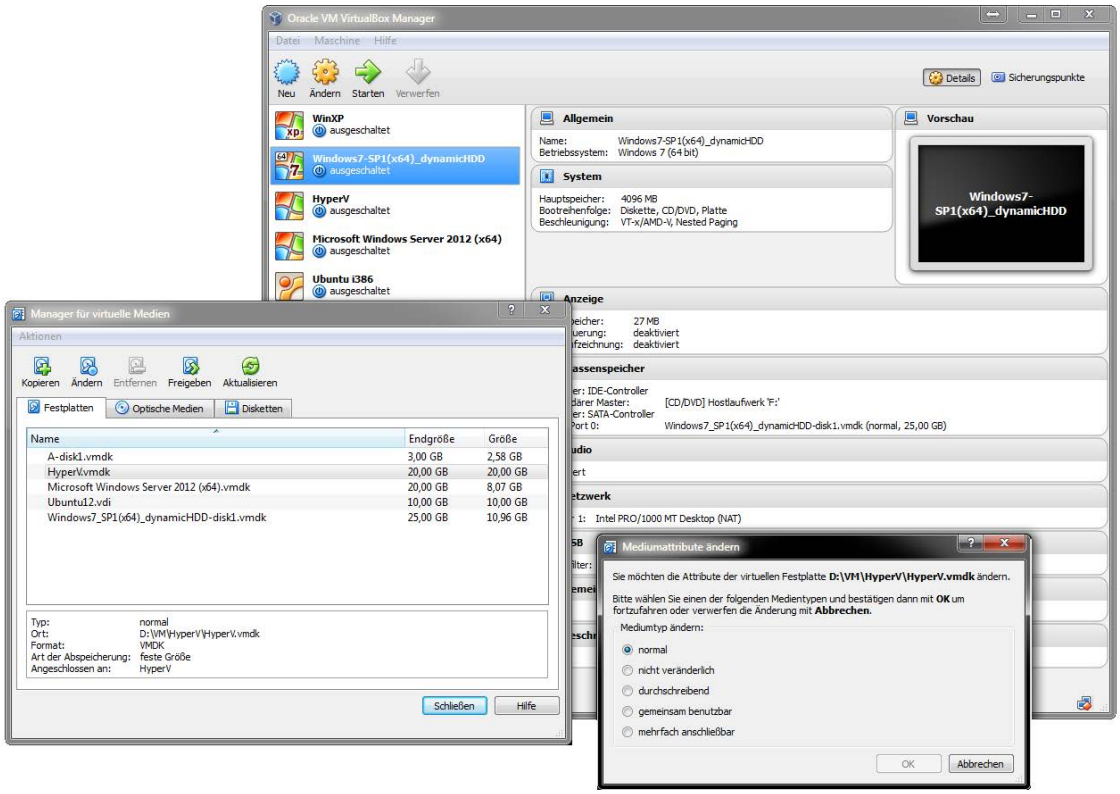

Abbildung 3.3: Oracle VM VirtualBox Manager für virtuelle Medien

Besonders benutzerfreundlich ist der von Oracle angebotene "Manager für virtuelle Medien", mit dessen Hilfe die virtuellen Festplatten einfach alteriert werden können (siehe Abbildung 3.3).

Es erscheint dem Autor etwas suspekt, dass unter `https://www.virtualbox.org/wiki/Security` von Oracle den Benutzern der Software geraten wird, für sicherheitsbezogene Angelegenheiten Bug-Reports (beispielsweise nach einem Programm-

Absturz) nicht zu öffnen oder gefundene Fehler zu veröffentlichen, sondern zuerst Oracle in diesen Belangen zu kontaktieren.

Weitere Produkte, welche die selben oder ähnliche Funktionalitäten wie Oracle VM VirtualBox anbieten, sind Parallels Workstation, QEMU, Cooperative Linux sowie einige Produkte von VMware (Workstation, Player, Server).

3.5.3 Sicherheit bei Linux und Linux-Erweiterungen

SELinux

Security Enhanced Linux ist eine Erweiterung der Funktionalität des Linux Kernels in Bezug auf Sicherheit. Alle laufenden Prozesse werden in einer Sandbox ausgeführt, damit sich die Schadroutinen kompromittierter Prozesse nicht auf andere Prozesse ausbreiten können. Da bei KVM (siehe Kapitel 4.3) die Instanzen der VMs als Prozesse ausgeführt werden, lässt sich dieses Konzept in Bezug auf Sicherheit auch auf die Virtualisierungslösung ausweiten. SELinux sandboxt nicht nur die einzelnen Instanzen der VMs, sodass diese voneinander isoliert sind, sie sind auch vom Hypervisor klar abgetrennt, damit die VMs auch vor eventuell auftretenden Attacken auf den Hypervisor geschützt sind (VM-Escapes, siehe Kapitel 4.6).

LynuxWorks

Das von der Lynx Software Technologies angebotene Real Time Operating System (RTOS) heißt LynxOS und ist ein linux-basiertes Betriebssystem, welches sich sehr stark auf Sicherheit bei Virtualisierungen konzentriert. LynxOS ist ein Typ 1 Hypervisor (bare-metal) und ermöglicht Para- und Voll-Virtualisierung in eingebetteten Systemen[15].

[15] `http://www.lynx.com`

3.5.4 Sicherheitsprobleme aufgrund der Architektur

Im Folgenden werden mögliche Sicherheitsprobleme aufgrund der zugrundeliegenden Architektur der verschiedenen Virtualisierungsarten besprochen.

Emulation

Da bei der Emulation lediglich Hardware für unmodifizierte Gäste emuliert und keine tatsächliche Umgebung virtualisiert wird, stellen sich bzgl. Virtualisierungssicherheit eher weniger Bedenken. Lediglich die Sicherheit des zugrundeliegenden Betriebssystems muss gewährleistet sein und VM-Escapes aus den VMs müssen unterbunden werden, wie auch bei den anderen Virtualisierungsarchitekturen. Emulation wird in Kapitel 2.4.1 sowie in Abbildung 2.2 beschrieben.

Betriebssystem-Virtualisierung

Das größte Problem bei der Betriebssystem-Virtualisierung ergibt sich daraus, dass die emulierten Umgebungen (bzw. die VMs, welche als isolierte Container unter dem Prinzip des Sandboxings als eng miteinander verbundene bzw. ähnliche Prozesse ausgeführt werden) den selben Kernel wie das zugrundeliegende Betriebssystem nutzen. Hier besteht einerseits die Gefahr, dass infiltrierte VMs Kontrolle über das Betriebssystem übernehmen und somit sowohl die Maschine selbst als auch alle anderen VMs gefährden, andererseits muss besonders auf die Sicherheit des Kernels geachtet werden, da bei einem Eindringen in das Betriebssystem selbst die Sicherheit aller VMs gefährdet ist. Betriebssystem-Virtualsierung wird in Kapitel 2.4.2 sowie in Abbildung 2.3 beschrieben.

Voll-Virtualisierung

Da die Voll-Virtualisierung den Inbegriff von Virtualisierung darstellt, muss ihr besondere Aufmerksamkeit bzgl. Sicherheit zugedacht werden. Es gelten die generellen Sicher-

heitsbedenken, die auch bei der Betriebssystem-Virtualisierung genannt wurden, sowie eine besonders durchdachte Gewährung von Sicherheit beim VMM, da die Verwaltung der VMs durch diesen erfolgt. Die Voll-Virtualisierung wird immer durch einen Hypervisor unterstützt bzw. ausgeführt, weshalb ihm die größten Bedenken bzgl. Sicherheit zukommen sollten.

Da die bare-metal Voll-Virtualisierung der Para-Virtualisierung sehr ähnlich ist, stellen sich bei dieser ähnliche Bedenken (siehe nächster Abschnitt). Voll-Virtualisierung wird in Kapitel 2.4.3 sowie in den Abbildungen 2.4 und 2.5 beschrieben.

Para-Virtualisierung

Das Konzept der Hypercalls steht bei der Para-Virtualisierung eine der größten Sicherheitsbedenken dar. Werden hardware- und privilegiensensitive Befehle direkt an den Hypervisor weitergegeben und diese nicht ausreichend auf mögliche Angriffsausführungen überprüft, haben infiltrierte VMs bzw. Angreifer theoretisch die Möglichkeit, VM-Escapes zu vollziehen und können somit teilweise Kontrolle über den VMM erlangen. Para-Virtualisierung wird in Kapitel 2.4.4 sowie in Abbildung 2.7 beschrieben.

4 Sicherheit beim VMM

Im Folgenden werden die bekanntesten bzw. verbreitetsten Hypervisor-Lösungen vorgestellt und in Bezug auf Sicherheit behandelt.

Generelle Sicherheitsangelegenheiten, die auf VMMs zutreffen, werden in [27] besprochen. Es wird dahingehend argumentiert, dass Schadsoftware, sobald diese auf einem Hypervisor aktiv wird, die Ressourcen des Hypervisors selbst nutzt. Das impliziert, dass der Angreifer ein Interface des VMM nutzt, welches von diesem selbst angeboten wird. Das Angriffsrisiko steigt in diesem Falle, je komplexere und je mehr Schnittstellen vom Hypervisor angeboten werden.

Prinzipiell können bare-metal Hypervisoren als sicherer angesehen werden, da sie ein schmaleres Interface an Funktionen anbieten und ihre Programmierung weniger Codezeilen benötigt als die von herkömmlichen Betriebssystemen. Aus Sicherheitsgründen sollte die Größe eines VMMs somit immer so klein wie möglich gehalten werden, um potentielle Angriffsstellen zu minimieren.

4.1 VMware ESX(i)

Dieser Hypervisor der Firma VMware wurde bisher in zwei Varianten angeboten: ESX kann als Vollversion inklusive Konsolenbetriebssystem angesehen werden, während ESXi eine abgespeckte, aber kompakter zu bedienende, Version des Hypervisors ist. Seit der Version 5.1 (September 2012) wird nur mehr die ESXi Variante fortgeführt. Die Besonderheit bei allen Virtualisierungsprodukten, welche von WMware angeboten werden, ist die Anwendung einer Technologie namens "Binary-Translation" – das bedeutet,

dass alle Anweisungen, welche von den VMs kommen, vom Hypervisor abgefangen und in echte Prozessoranweisungen übersetzt werden. Dies hat den Vorteil, dass keine Anweisung Parameter oder Vorgaben des virtualisierten Prozessors verletzen kann.

Eine weitere Besonderheit bei VMware ESXi ist das Vorhandensein eines Hypervisor-Separations-Kernels (vmkernel), dessen Hauptaufgabe es ist, die VMs voneinander zu isolieren. Neben diesem Kernstück des VMware ESXi existiert auch noch der "VMX helper", welcher den VMs eine Voll-Virtualisierung ermöglicht, in dem er die benötigte Hardware emuliert [27].

Sicherheitsbedenken hinsichtlich des VMware ESXi Hypervisors existieren dahingehend, dass der Quellcode nicht Open-Source ist und somit nur der Hersteller selbst für das Vorhandensein von Sicherheit garantieren kann. Somit gibt es nicht die Möglichkeit einer Entwicklungs-Community, welche sich zusätzlich um die Weiterentwicklung und Verbesserung des VMMs kümmert. Es existieren jedoch Möglichkeiten, den Herstellern gefundene Sicherheitsmängel zu melden; behobene Mängel werden veröffentlicht. Weiters gibt es Anleitungen zur Erhöhung der Sicherheit von VMware ESXi[16].

4.2 Microsoft Hyper-V

Der von Microsoft angebotene VMM Hyper-V (Entwicklungs-Codename "Viridian") ist der Nachfolger von Microsofts Windows Server Virtualisierung. Hyper-V ist sowohl als hosted VMM als Bestandteil der Server-Betriebssysteme Windows Server 2008, 2008 R2, 2012 und 2012 R2 und in der 64-bit Version von Windows 8 und 8.1 verfügbar, als auch als bare-metal VMM bzw. stand-alone Betriebssystem unter dem Namen Hyper-V Server 2008 bzw. 2012. In der stand-alone Version ist nur ein Kommandozeileninterface vorhanden. Die Konfiguration der VMs über die grafische Benutzeroberfläche ist übersichtlich und klar verständlich zu bedienen.

Die Architektur des Hyper-V unterscheidet sich von den anderen VMMs dahingehend, dass der Hypervisor selbst im Ring -1 und das zugehörige Verwaltungsprogramm in

[16] http://pubs.vmware.com/vsphere-51/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-511-security-guide.pdf

Ring 0 ausgeführt wird. Die Instanzen der VMs werden wie gewohnt unprivilegiert im Ring 3 ausgeführt (siehe Abbildung 2.1).

Ähnlich wie bei Virtualisierungssoftware (siehe Kapitel 3.5.2) gibt es auch bei Hyper-V keine dezidierten Möglichkeiten, Sicherheit zu konfigurieren. Das führt erneut dazu, dass die verantwortliche Person über alle Risiken Bescheid wissen und selbst für die Sicherheit des physischen Systems und der VMs sorgen muss.

4.2.1 Konfiguration des Hyper-V

Das Einrichten der Hyper-V Funktionalität innerhalb eines Microsoft Windows Server Betriebssystemes erfolgt nach einem einfachen Ablauf. Nach dem Hinzufügen von Hyper-V mittels "Neue Serverrollen und Features hinzufügen" werden die Hyper-V GUI und die Hyper-V PowerShell in der Serverfunktionalität ergänzt. Zusätzliche Einstellungen für virtuelle Switches, VM Live-Migration und Speicherorte der virtuellen Medien werden vorab getroffen. Nach zwei Neustarts des Servers steht der Hyper-V Manager zur Verfügung, über den sehr einfach und benutzerfreundlich neue VMs hinzugefügt und bestehende verwaltet werden können.

Obwohl Hyper-V als "der neue Stern am Virtualisierungshimmel" angepriesen wurde, stehen auch bei diesem VMM keine spezifischen Konfigurationsmöglichkeiten für Sicherheit zur Verfügung. Positiv bemerkenswert sind aber die generellen Einstellungsmöglichkeiten bei den VMs, welche deutlich mehr Detailkonfigurationen ermöglichen (siehe Abbildung 4.1) als beispielsweise Oracle VM VirtualBox.

Erwähnenswert ist noch die Angabe von "Generationen" beim Erstellen einer neuen VM. Hyper-V unterscheidet Generation 1 und 2. Wobei Generation 1 den VMs die selbe virtuelle Hardware anbietet, wie alle bisherigen Virtualisierungsprodukte von Microsoft, und Generation 2 den VMs spezielle Boot-Funktionalitäten wie Secure Boot, SCSI Boot und PXE Boot ermöglicht. Bei VMs der Generation 2 werden aber zumindest Microsoft Windows Server 2012 oder 64-bit Versionen von Windows 8 gefordert, das bedeutet dass nur neue Betriebssysteme von Microsoft unterstützt werden (siehe Abbildung 4.2).

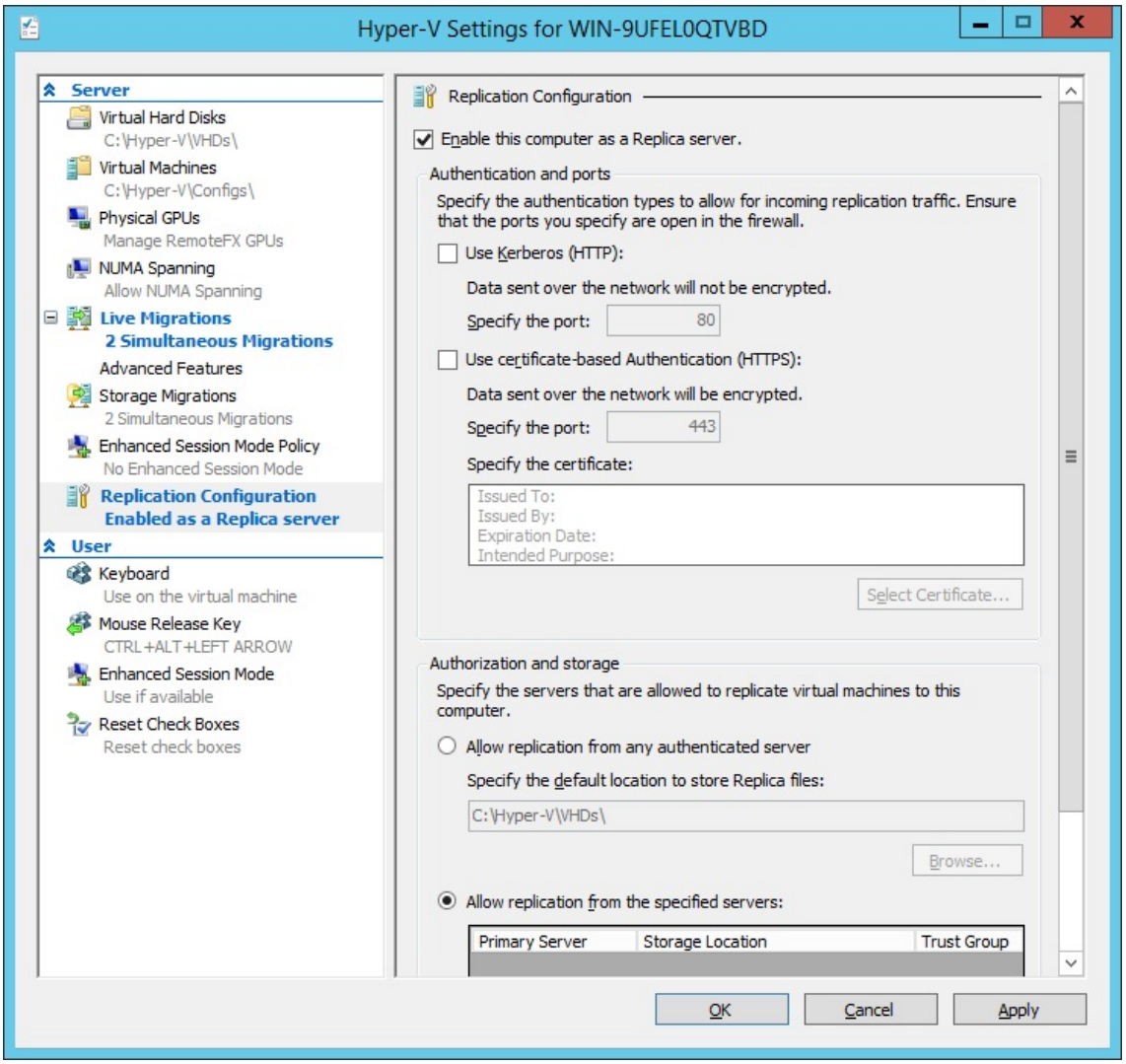

Abbildung 4.1: Allgemeine Einstellungsmöglichkeiten bei Hyper-V

4.2.2 Sicherheit bei Hyper-V

Im Microsoft TechNet[17] gibt es Anleitungen für die Erhöhung der Sicherheit bei der Anwendung von Hyper-V und VMs. Ein Großteil der dort zur Verfügung gestellten Informationen bezieht sich aber auf allgemeine Belange zur Erhaltung der Sicherheit, wie beispielsweise das ständige Aktualisieren von Antiviren-Software oder das Durchfüh-

[17] `http://technet.microsoft.com`

ren von Systemupdates, damit keine unnötigen Sicherheitslücken entstehen oder offen bleiben. Lediglich etwas spezifiziertere Anleitungen für die Konfiguration von rollenbasierten Zugriffssteuerungen und das Verwenden eines Autorisierungs-Managers werden angeboten.

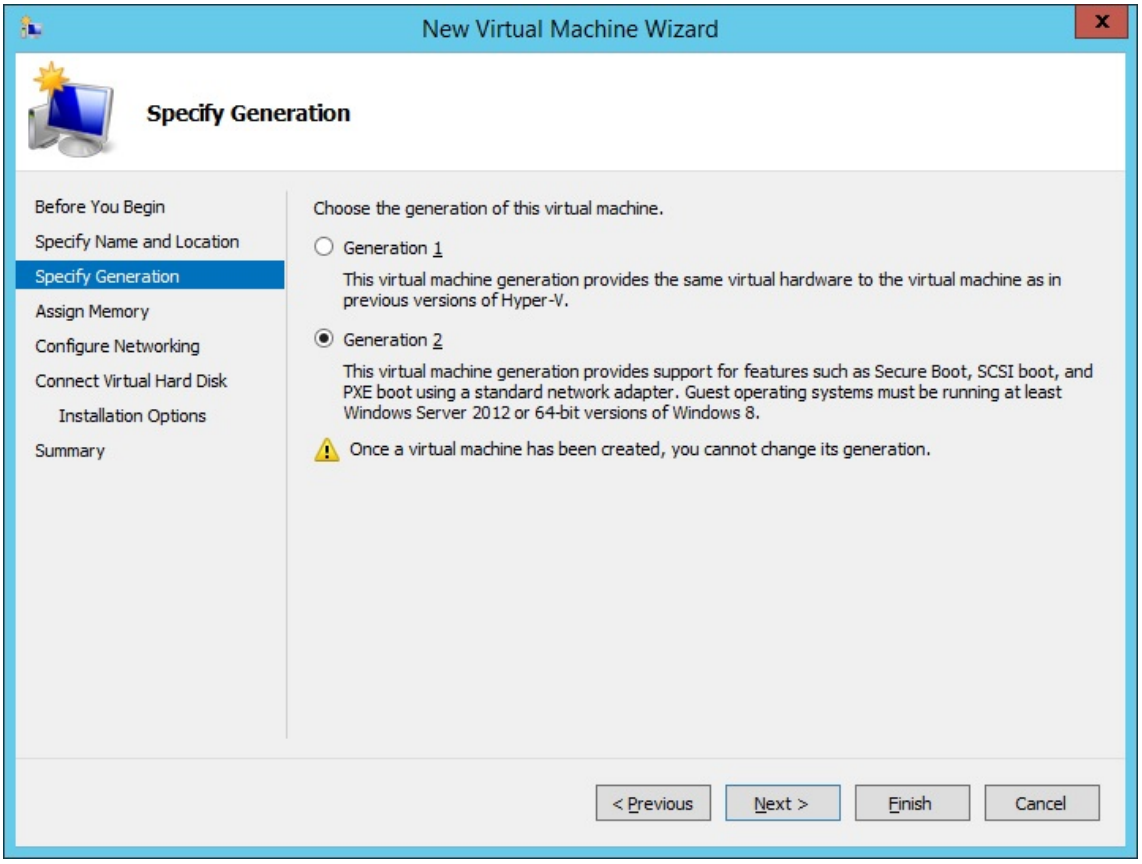

Abbildung 4.2: Konfiguration der VM Generation bei Hyper-V

Der Umstand, dass bei Hyper-V keine spezifizierten Sicherheitsoptionen bzw. Konfigurationen in Bezug auf Sicherheit getroffen werden können, ist etwas bedenklich. Es sollten zumindest Möglichkeiten zur stufenweisen Isolierung (Sandboxing) der Instanzen der VMs ermöglicht werden.

4.3 KVM

Kernelized (oder kernel-based) Virtual Machines wurden erstmals 1977 als KVM/370 (Umbau des VM/370 Betriebssystems von IBM, Vorgänger von z/VM, siehe Kapitel 4.5) namentlich erwähnt [14] und sind seit 2007 als eine Virtualisierungs-Infrastruktur in Linux Kernels enthalten.

KVM ist somit ein integriertes Linux Kernelmodul und ermöglicht dem Betriebssystem Virtualisierungsdienste als Typ 1 (bare-metal) Hypervisor anzubieten. KVM besteht aus zwei Hauptkomponenten: einerseits das KVM Kernel-Modul und andererseits QEMU für die Emulation von Hardware.

KVM bietet Voll-Virtualisierung, Para-Virtualisierung (als Teil von QEMU, nicht als Teil des Kernels) sowie eine starke Isolierung der Gäste (einer der großen Stärken von KVM), ähnelt anderen x86-basierten Hypervisors und wird, aufgrund der Open Source Charakteristik, ständig verbessert und weiterentwickelt. Dies wirkt sich vor allem dahingehend aus, dass Basisfunktionen nicht neu entwickelt werden müssen, sondern sich die Entwickler auf die Optimierung von vorhandenen, bereits bewährten, Teilen von Linux zur Virtualisierung konzentrieren können [5].

Viele Vorteile von KVM laut [39] sind beispielweise die geringen Kosten, die ständige Weiterentwicklung, die Ausnutzung von betriebssystemspezifischen Vorteilen von Linux und die hohe Effizienz.

Die stärkste Weiterentwicklung wird zurzeit von der Open Virtualization Alliance (OVA)[18] betrieben, deren Mitglieder unter anderem HP, IBM, Intel, Red Hat und SUSE sind. IBM nutzt KVM für die öffentliche SmartCloud Enterprise Cloud und für ihre private Research Compute Cloud.

Ein weiterer Sicherheitsvorteil von KVM ist, dass der System-Kernel von Linux und der KVM-Kernel zur Laufzeit nicht voneinander getrennt werden können, da beide Kernel im selben Adressraum mit den selben Privilegien der Hardware laufen. Somit müssen sich sowohl der Linux-Kernel als auch der KVM-Kernel darauf verlassen können, dass beim jeweils anderen alle Sicherheitsanforderungen an einen VMM erfüllt sind [27].

[18] `https://openvirtualizationalliance.org/`

Dieser Umstand stellt aber in gewissen Belangen auch einen Nachteil dar, da ähnlich wie bei Xen besonders auf die Sicherheit beider Kernels geachtet werden muss und die Kompromittierung eines sensiblen Systemteils gravierende Auswirkungen auf den anderen bzw. auf den Betrieb des Gesamtsystems haben kann.

Ausgeführt werden die VMs in KVM im Ring 3 des Privilegierungsmodells, wobei, wie in den meisten Anwendungsfällen, die Ringe 1 und 2 nicht verwendet werden. Somit läuft der Hypervisor in Ring 0 (Kernel-Mode, höchste Privilegierungsstufe) und die VMs in Ring 3 (siehe Abbildung 2.1).

4.4 Xen

Der Xen Hypervisor ist das Herzstück eines jeden Xen Systems. Er residiert typischerweise zwischen der Hardware und den VMs, verwaltet Ressourcen und kümmert sich um die Sicherheit und um die Isolation der VMs. Er stellt den VMs ein Interface zur Hardware zur Verfügung und definiert somit die virtuelle Hardware, welche die VMs anstatt der physischen Hardware sehen. Dies zieht den Vorteil nach sich, dass Gastbetriebssysteme und Anwendungsprogramme innerhalb der VMs möglichst wenig modifiziert werden müssen, wenn die virtualisierte Hardware der physischen sehr ähnlich ist. In Abbildung 4.3 ist die Beziehung zwischen dem Xen Hypervisor, der Hardware und den VMs dargestellt.

Eine Besonderheit – oder auch ein Nachteil – die bei Xen beachtet werden muss, sind die unterschiedlichen Kernels des Hypervisors. Da es unterschiedliche Kernels gibt, gilt es, diese Kernels in Bezug auf Sicherheit konsistent zu halten. Der Grund ist, dass sonst sowohl beim privilegierten als auch beim unprivilegierten Kernel Möglichkeiten für Eindringlinge eröffnet werden können.

Der Domain-0-Kernel (dom0) ist jener Kernel, der beim Start des Hypervisors ausgeführt bzw. gestartet wird. Dieser Kernel stellt das Kernsystem dar, welches den eigentlichen VMM verwaltet und andere Domänen (VMs) ausführen und verwalten kann. Die anderen Domänen, Domain-Unprivileged-Kernels (domU) bzw. domU-Kernels entsprechen den eigentlichen VMs, die vom Xen Hypervisor verwaltet werden. Der dom0

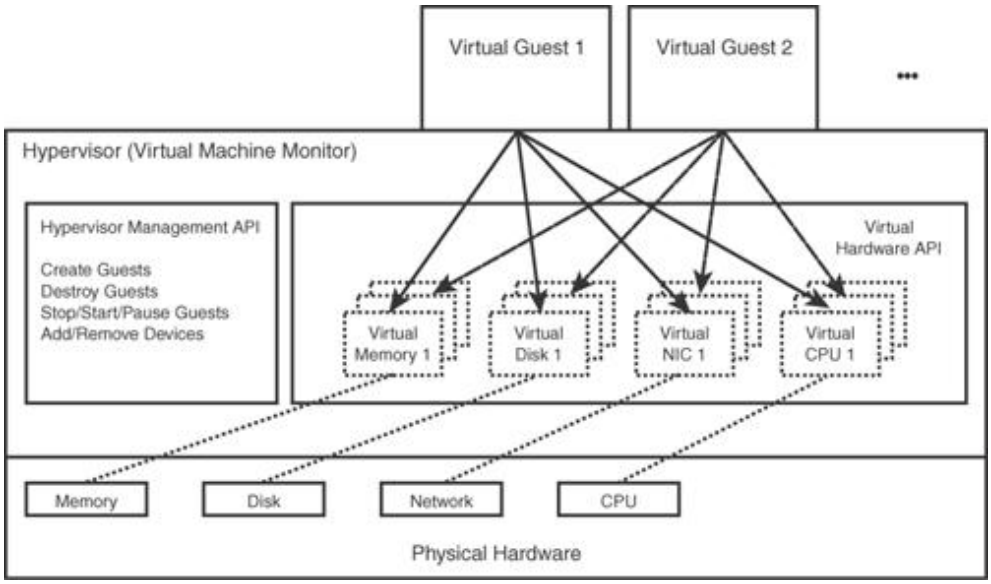

Abbildung 4.3: Der Xen Hypervisor (aus [24])

Kernel benötigt einen Hauptprozessor, der dessen Befehlssatz unterstützt (Intel VT-x oder AMD-V), also genau der Kernel, der auch wirklich auf der Hardware ausgeführt wird. Sämtliche VMs, welche als domU ausgeführt werden, müssen von der realen Hardware gar nichts wissen, denn der dom0 Kernel verwaltet alle Ressourcen und simuliert/emuliert die benötigte Hardware (dom0 = Para-Virtualisierung, domU = Voll-Virtualisierung). Da der dom0 Kernel das Kernsystem von Xen und dessen Funktionalität darstellt, kommt diesem Kernel ein besonders hohes Maß an vorausgesetzter Sicherheit zu.

Auf einem Xen System läuft der Hypervisor in Ring 0, die Instanzen der VMs in Ring 1 und Applikationen in den VMs in Ring 3. Die domU-Kernels bzw. die Instanzen der VMs sind immer unmodifizierte Gäste, die Voll-Virtualisierung der Hardware benötigen. Wie auch bei KVM wird bei den VMs die Voll-Virtualisierung durch QEMU gewährleistet, zusätzlich wird die paravirtualisierte Nutzung von Gerätetreibern ermöglicht, damit die von Xen verfügbar gemachte Hardware effizienter genutzt werden kann [27].

Xen stellt jeder VM ein virtualisiertes BIOS zur Verfügung. Somit wird sichergestellt, dass jede VM alle Ressourcen und Befehle dort findet und ausführen kann, wie sie es sich von einer nativen Umgebung erwarten würde. Mittels Hypercalls (Paravirtua-

lisierung, siehe Kapitel 2.4.4) können VMs Anforderungen an den Hypervisor stellen (beispielsweise Speicherverwaltung). Aus Sicherheitsgründen sind einige der (in Summe zurzeit 39 implementierten) möglichen Hypercalls bei Xen dom0 vorbehalten - diese privilegierten Hypercalls sind beispielsweise DOMCTL (domain configuration at runtime), SYSCTL (system parameter settings) und ACM_OP (access control settings).

Mittels Xen Security Modules (XSM), ein Sicherheitsframework für Xen[19], welches eine Ableitung der Linux Security Models (LSM) ist, kann ein Administrator weitgehende Sicherheitseinstellungen im dom0 sowie in den domU Kernels vornehmen[20]. Die XSM Module werden während des Startvorganges des Hypervisors geladen und verlinkt.

Direkt angwandt werden kann XSM über Flux Advanced Security Kernel (FLASK), welches ein Framework für XSM, entwickelt von der NSA (United States' National Security Agency) und anderen, ist. Neben FLASK gibt es an XSM Framworks auch noch ACM/sHype von IBM sowie Dummy (XSM Standard).

4.5 z/VM

Das z/VM (vormals VM/CMS, siehe Kapitel 1.3) ist weniger für gewöhnliche Anwender vorgesehen, da es sich um ein Betriebssystem für Großrechner (IBM Mainframe) handelt und somit exklusiv zur Anwendung kommt, wenn dementsprechende Ressourcenanforderungen vorliegen. Da die Funktionalität der eines Hypervisors gleicht, werden hier die Sicherheitsaspekte behandelt, welche bei z/VM großteils durch eine starke Isolierung der Gäste umgesetzt wird.

Die beiden Hauptteile von z/VM sind das CP (Control Program), welches als der eigentliche Hypervisor angesehen werden kann, und das CMS (Cambridge Monitor System), welches jeweils den Instanzen der VMs entspricht[21].

Die VMs, die auf einem z/VM System laufen, wissen nichts voneinander. Dies wird durch die IEF (Interpretive Execution Facility) erreicht. Die IEF startet mittels der SIE

[19] http://mail.xen.org/files/summit_3/coker-xsm-summit-090706.pdf
[20] http://wiki.xen.org/wiki/Xen_Security_Modules_:_XSM-FLASK
[21] http://www.vm.ibm.com/overview/

(Start Interpretive Execution) Instruktion eine Instanz einer VM als einzelne Operation. Die VM wird so ausgeführt, dass die Firmware der physischen Maschine nicht weiß, dass sie virtualisiert wird. Die VM wird so lange ausgeführt, bis eine Instruktion aufgerufen wird, welche nicht virtualisiert werden kann (beispielsweise Zugriff auf eine nicht virtualisierte Speicheradresse). An diesem Punkt übernimmt das CP wieder die Kontrolle und liefert die benötigten Operationen und Ergebnisse an die VM. Weitere Maßnahmen für Sicherheit bei z/VM sind eine starke Anwendung von Kryptographie, Intrusion Detection Systeme (speziell für Logins) und Zertifizierungen (Common Criteria der ISO). Besonders hilfreich ist die Möglichkeit des Debuggens der virtuellen Umgebungen im laufenden Betrieb, sowie eine Einteilung aller VMs und Benutzer in speziell privilegierte Userklassen.

Für weitere Details zu Sicherheitsaspekten von z/VM siehe Kapitel 1.2 in [11].

4.6 VM-Escapes

Unter der Bezeichnung VM-Escape wird der schlimmste Fall einer Anwendung von Schadsoftware im Virtualisierungsbereich verstanden, nämlich das Ausbrechen von Malware, welche in einer VM ausgeführt wird und Kontrolle über den VMM übernimmt. Hat Schadsoftware Kontrolle über das laufende System innerhalb einer VM und ist es ihr möglich, aus der eigentlich isolierten Umgebung der VM auszubrechen und so den zugrundeliegenden Hypervisor zu infiltrieren, gilt die gesamte physische Maschine als kompromittiert, sofern es sich um einen bare-metal Hypervisor handelt (wobei auch bei einem hosted Hypervisor ein VM-Escape nicht weniger Schaden anrichten kann).

Wie in Kapitel 6.2 beschrieben wird, gibt es mehrere Arten von Ausbrüchen bei virtualisierten Infrastrukturen, welche als VM-Escapes klassifiziert werden können. Besonders nennenswert dabei sind die folgenden Attacken (vergleiche die Nummerierungen in Abbildung 6.1):

d.) VM zu VM direkt: Diese Art von VM-Escape stellt eine technische Hürde dar, nämlich das Infiltrieren einer anderen VM, ohne dass der VMM etwas davon merkt bzw. ohne dass die Attacke über das Internet geht. Diese Hürde ist auch

der Grund dafür, dass diese Art von Attacke bisher noch nie dokumentiert wurde.

e.) VM zu externen Speichern: Hierzu zählen neben dem Speichern von VM Images auf externen Speichern auch die Funktionalität von so genannten gemeinsam verwendeten Ordnern. Gemeinsame Ordner werden oft dazu eingesetzt, Daten zwischen VMs untereinander und zwischen VMs und VMM auszutauschen. Hier hat Malware eine sehr einfache Möglichkeit, aus der eigentlich isolierten Umgebung auszubrechen.

Das Verwenden von gemeinsamen Ordnern muss zumeist vom Benutzer selbst aktiviert werden, dadurch tut sich hierbei eine Sicherheitslücke auf, welche besonders beachtet werden muss. Eine gezielte Überwachung des gemeinsam verwendeten Ordners beispielsweise mit Anti-Viren-Software ist ein erster Schritt als Maßnahme, einem VM-Escape gezielt entgegen zu wirken.

Ein konkreter Fall zu dieser Art von VM-Escape findet sich in der VMware betreffenden CVE-2008-0923[22].

f.) VM zu VM über VMM: Diese Attacke ist als Inbegriff von VM-Escapes anzusehen. Schadsoftware, die in einer vermeintlich isolierten VM ausgeführt wird, kann es durch diverse Sicherheitslücken gelingen, Kontrolle über den Hypervisor zu erlangen. Dieser Art von Ausbruch passiert in der Regel nicht "in the wild", kann aber durchaus durch gezielte Infiltrationsversuche bzw. Angriffe möglich sein. Beispiel für eine solche VM-Escape findet man unter dem Namen "VUPEN Method"[23].

Weitere exemplarische Beispiele für VM-Escapes finden sich in einem Hyper-V Blog von Aidan Finn[24] sowie in [33]

[22] http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2008-0923
[23] http://www.securityweek.com/vupen-method-breaks-out-virtual-machine-attack-hosts
[24] http://www.aidanfinn.com/?p=13363

5 Virtual Machine Based Rootkits

Dieser Kategorie von Sicherheitsbedrohungen im Virtualisierungssektor (abgekürzt VM-BR) wird ein eigener Abschnitt gewidmet, da sie zu den neuesten Entwicklungen in diesem Bereich gehören. Der Überbegriff für diese Art von Malware ist "virtualisierende Rootkits", der sich in die Bereiche der hardwarevirtualisierenden und softwarevirtualisierenden Rootkits gliedert.

5.1 Rootkits allgemein

Bei einem Rootkit handelt es sich um ein Paket von Programmen oder Diensten von Schadsoftware, das verwendet wird, um nach der Kompromittierung eines Systems die Spuren der Kompromittierung und der Schadsoftware zu verbergen. Zumeist wird es auch dafür benutzt, um sich die durch die Kompromittierung erlangten Rechte zu sichern, indem Teile von Systemfunktionalitäten verändert werden. Das Hauptziel von Rootkits ist heutzutage nicht mehr der Austausch von Anwendungen, sondern das Verbergen von Informationen durch Eingriffe ins Betriebssytem und das Öffnen weiterer Backdoors. Peter Szor definiert Rootkits in "The Art of Computer Virus Research and Defense" wie folgt:

> *"Rootkits are a special set of hacker tools that are used after the attacker has broken into a computer system and gained root-level access. Usually, hackers break into a system with exploits and install modified versions of common tools. Such rootkits are called user-mode rootkits because the Trojanized application runs in user mode. Some more sophisticated rootkits, [...] have kernel-mode module components. These rootkits are more*

> *dangerous because they change the behavior of the kernel. Thus, they can hide objects from even kernel-level defense software. For example, they can hide processes, files in the file system, registry keys, and values under Windows, and implement stealth capabilities for other malicious components. In contrast, user-mode rootkits cannot typically hide themselves effectively from kernel-level defense software. User-mode rootkits only manipulate with user-mode objects; therefore, defense systems relying on kernel objects have chance to reveal the truth."* [34]

Funktionalitäten von Rootkits bzw. Techniken sind, beispielsweise nach den Inhalten der LVA "System Security", die folgenden:

> *"Modify operating system behavior; Techniques for hiding running processes, files, data, logs, registry, programs; Fine line between malicious and benign; Kernel and User mode; Hiding place for other malware; Inline function hooking"* [10]

Prinzipiell versuchen Rootkits Kontrolle über das jeweilige System zu erlangen. King et al. beschreiben in ihrem Whitepaper "SubVirt: Implementing malware with virtual machines" mit welcher Vorgehensweise Kontrolle erlangt werden kann:

> *"A major goal of malware writers is control, by which we mean the ability of an attacker to monitor, intercept, and modify the state and actions of other software on the system. Controlling the system allows malware to remain invisible by lying to or disabling instrusion detection software."* [20]

Was Kontrolle über ein System ist, bringt folgende Beschreibung auf den Punkt:

> *"Control of a system is determined by which side occupies the lower layer in the system. Lower layers can control upper layers because lower layers implement the abstractions upon which upper layers depend."* [20]

Laufen sowohl angreifende Malware und abwehrende Schutzsoftware auf demselben Privilegienring, hat keiner einen maßgeblichen Vorteil gegenüber dem anderen – derjenige, der effizienter arbeitet und den anderen besser versteht, wird die Kontrolle erlangen.

> *"If one were asked to classify viruses and worms by a single defining characteristic, the first word to come to mind would probably be 'replication'. In contrast, the single defining characteristics of rootkit is 'stealth'. Viruses reproduce, but rootkits hide."* [9]

5.2 Klassifikation

Bei Rootkits ist eine unterschiedliche Klassifikation durch deren Einordnung in verschiedene Systemschichten, abhängig von ihrer Funktionsweise, möglich:

User-Mode: Diese Form von Rootkits ist am einfachsten zu implementieren und kann auch am einfachsten von Antivirensoftware aufgespürt werden. Diese wird vor allem unter Windows eingesetzt, da es einige Möglichkeiten anbietet, Rootkit-Funktionalitäten im laufenden Betrieb einzubinden. Da User-Mode Rootkits meist mit den selben Rechten exekutiert werden wie die Applikationen, die sie unterwandert haben, verfügen sie nur über beschränkte Rechte auf Systemebene. Sie müssen keine Funktionalitäten auf Kernel-Ebene beeinflussen, daher sind sie verhältnismäßig leicht zu implementieren, dies ließ sie zu einer beliebten Art von Malware werden. Eine Unterart sind *Application-Mode Rootkits*, die auch als *Binary-Mode Rootkits* bezeichnet werden. Sie ersetzen reguläre Anwendungen oder bestehen aus veränderten Systemprogrammen, werden heute aber nicht mehr verwendet, da sie sehr leicht aufzuspüren sind.

Kernel-Mode: Diese Rootkits werden in der Schicht ausgeführt, in der die meisten Privilegien vorhanden sind (Ring 0), indem sie entweder Routinen des Betriebssystems verändern oder neue hinzufügen. Die meisten Betriebssysteme erlauben das Einbinden von Geräte-Treibern, welche auf der selben Ebene ausgeführt werden. Aus diesem Grund werden Kernel-Mode Rootkits oft als solche Geräte-Treiber getarnt. Wie in Kapitel 5.1 bereits erklärt, ist Malware, die auf der selben Ebene wie die Schutzmechanismen ausgeführt wird, sehr schwer aufzuspüren, was Kernel-Mode Rootkits zu einer großen Bedrohung macht. Sie haben auch die Fähigkeit, mittels "Direct Kernel Object Modification (DKOM)" (siehe Kapitel 5.3.1) Daten-

strukturen im Kernel zu beeinflussen. So können beispielsweise Übergänge von Kernel- und User-Mode verändert werden, um sich so selbst zu tarnen (daher die Bezeichnung "Stealthware"). Eine Unterart von Kernel-Mode Rootkits sind so genannte *Bootkits*, welche speziell darauf ausgerichtet sind, den Bootvorgang eines Systems zu verändern bzw. zu infiltrieren, indem sie den Master Boot Record (MBR) eines Dateisystems so umschreiben, dass das Bootkit beim nächsten Neustart ausgeführt wird, bevor das eigentliche Betriebssystem gestartet wird. Dies hat zur Folge, dass das Bootkit volle Kontrolle über das System hat und das Betriebssystem – inklusive aller vorhandenen Schutzsysteme – mit weniger Privilegien als das Bootkit ausgeführt wird.

Hypervisor-Level: Diese Art von Rootkits wurde das erste Mal im Zuge des Papers SubVirt [20] im Forschungsbereich vorgestellt. Es wurden dabei Hypervisor-Technologien von Intel's Virtualization Technology (Vanderpool, Intel VT-x) und AMD's Pacifica (AMD-V) folgendermaßen genutzt: das Rootkit nistet sich im Privilegienring -1 ein und verschiebt dabei das Betriebssystem im laufenden Betrieb in eine VM, ohne dass der Nutzer davon etwas bemerkt. Diese Art von Rootkits sind der Inbegriff von Virtual Machine Based Rootkits (VMBR) wie SubVirt oder Bluepill, auf die in diesem Kapitel noch näher eingegangen wird.

Hardware-Level: Rootkits, die auf Hardware- bzw. Firmware-Level arbeiten, werden, wie der Name schon sagt, oft direkt in die Firmware eines Gerätes geschrieben, um eine persistente Kontrolle über das Gerät zu erhalten. Problematisch ist hier die beschränkte Größe der Firmware-Chips. Da diese meist nur sehr wenig Speicherplatz anbieten, muss bei der Integration des Rootkits darauf geachtet werden, dass der Schadcode sehr kompakt gehalten wird. Von der ursprünglichen Firmware dürfen keine signifikant wichtigen Codeteile überschrieben werden, da daraus resultieren kann, dass das System als Ganzes nicht mehr lauffähig ist.

5.3 Arbeitsweise

In diesem Abschnitt wird die Vorgehensweise der Implementierung von Rootkits, um die veränderten Informationen des laufenden Betriebssystems, welche auf die Existenz von Rootkits schließen lassen, vor Schutzprogrammen zu verbergen, beschrieben.

Hooking: Dieser Begriff bezeichnet die Möglichkeit, Betriebssysteme mittels von ihnen bereitgestellten Schnittstellen in der Funktionalität mit fremdem Programmcode zu erweitern. Dabei ist nicht relevant, ob Funktionen erweitert oder ersetzt werden. Rootkits nutzen dabei die Möglichkeit, Rücksprungadressen oder Rückgabewerte so zu manipulieren, dass die Malware aufgerufen wird oder Daten erhält, die eigentlich nicht für sie bestimmt sind. Solche Hooks können gezielt dafür verwendet werden, Anwendungen falsche Informationen zu liefern. Eine beliebte Anwendung ist "Import Address Table Hooks" (IAT), siehe Kapitel 5.3.3. Oft werden Boot-Viren mit dem Interrupt `INT 13h` und Datei-Viren mit dem Interrupt `INT 21h` angewandt [34].

Inline Hooking: Schutzprogramme können von Malware ausgenutzte Hooks deshalb erkennen, weil die Schnittstellen (speziell Anfang und Ende der Erweiterung) den Signaturen von Malware entsprechen. Beim Inline-Hooking werden bestehende, nicht schädliche Erweiterungen so manipuliert, dass der Rootkit-Aufruf erst später in der Erweiterung, aber nicht bei der Schnittstelle, erfolgt. Da meist nur Anfang und Ende überprüft werden, versagen die meisten Schutzprogramme.

Layered Filter Drivers: Für Zwecke der Wiederverwendbarkeit und Erweiterbarkeit von Funktionen unterstützt Windows das Prinzip der geschichteten Gerätetreiber. Damit können Teile von Treibern verändert oder verbessert werden, ohne den gesamten Gerätetreiber neu implementieren zu müssen. Man kann sich sozusagen "in eine Schicht einklinken" und Funktionen nutzen, die in den Schichten darüber oder darunter bereits vorhanden sind. Genau nach diesem Prinzip arbeiten auch Virenscanner: sie implementieren beispielsweise einen Filter, der sämtliche Dateien durchsucht, um so Malware oder infizierte Dateien aufspüren zu können.

Object Manipulation: Wenn ein Rootkit nicht den Ablauf eines Programmes oder eines Betriebssystems verändern will, hat es die Möglichkeit, die jeweiligen im Speicher abgelegten Objekte zu manipulieren. Hier muss der genaue Aufbau aller Dateistrukturen bekannt sein, was einen offenen bzw. bekannten Quelltext des Betriebssystems voraussetzt – ein Problem, das beispielsweise bei Microsoft Windows Betriebssystemen nicht auftritt.

Die häufigsten Anwendungen dieser Methoden sind *Import Address Table Hooks (IAT)*, *System Service Descriptor Table Hooks (SSDT)* und die *Direct Kernel Object Manipulation (DKOM)*, die im Folgenden beschrieben werden, vergleiche dazu [15], [16] [34].

5.3.1 Direct Kernel Object Manipulation (DKOM)

Informationen von Prozessen werden vom jeweiligen Betriebssystem in Kernelobjekten im Speicher abgelegt. Die Verwaltung und der Zugriff auf diese abgelegten Informationen erfolgen meist über einen eigenen Objektmanager. Wie bereits bei der "Object Manipulation" erklärt, werden diese Daten direkt manipuliert – es wird also der Objektmanager umgangen. Werden die Daten im Kernspeicher direkt verändert, ohne dass das Betriebssystem oder der Objektmanager etwas davon mitbekommt, hat man eine elegante Möglichkeit gefunden, Objektdaten zu verfälschen und so den normalen Betriebssystemablauf zu manipulieren. Neben der Voraussetzung des quelloffenen Betriebssystemcodes stellt sich auch noch das Problem, dass sich bei den meisten Betriebssystemen die internen Strukturen nach dem Ausführen von Updates maßgeblich ändern, dies bringt einen enormen Aufwand bei der Aktualisierung von Rootkits, welche DKOM verwenden, mit sich.

In [16] geben Greg Hoglung und James Butler in Kapitel 7 "Direct Kernel Object Manipulation" eine Schritt für Schritt Beschreibung, wie man mittels DKOM den Ablauf des Betriebssystems manipulieren kann bzw. wie Prozesse, Ports und Gerätetreiber versteckt oder die Ausführungs-Privilegienstufe von Prozessen und Threads herausgefunden und verändert werden können.

5.3.2 System Service Descriptor Table Hooks (SSDT)

Als zentrale Recheneinheit ist die CPU (Central Processing Unit) neben der Verwaltung der Privilegierungen auch für viele andere Dinge zuständig, beispielsweise für das Verhalten bei Fehlern, die Kommunikation zwischen Programmen oder das Wechseln der Ausführung bei Multi-Threading-Programmen.

Da die CPU nicht alle Informationen intern abspeichern kann, greift sie auf Tabellen zurück, in denen die Softwareroutinen und Adressen abgelegt sind. Die wichtigsten dieser CPU-Tabellen sind die "Global Descriptor Table (GDT)", die "Local Descriptor Table (LDT)" und das "Page Directory" die verwendet werden, um Adressen aufzuspüren, sowie die "Interrupt Descriptor Table (IDT)" um Fehlerbehandlungsroutinen (Interrupt Handlers) zu finden [16].

Neben der CPU verfügt das jeweilige Betriebssystem über Tabellen, die der Ausführungsunterstützung dienen. Die wichtigste dieser betriebssystembasierten Tabelle ist die "System Service Dispatch/Descriptor Table (SSDT)" von Windows, welche Systemaufrufe verwaltet und alle nativen Adressen von Betriebssystemunterteilen von Windows, wie Win32, POSIX und OS/2 beinhaltet (Malware Analyst's Cookbook [21]).

5.3.3 Import Address Table Hooks (IAT)

Das PE-Format (Portable Executeable) ist ein Format für ausführbare Programme, sowohl für Win32- als auch für Win64-Systeme. Typische Anwendung findet es bei *.exe und *.dll Dateien. Dieses Format nutzt Informationen aus einer Tabelle, in der nachgeschlagen wird, wenn eine Anwendung eine Funktion aufruft. In dieser Tabelle stehen die Verweise auf die Anfänge der Funktionen – es werden also Adressen importiert, zu denen gesprungen wird, daher der Name "Import Address Table".

Bei IAT Hooks handelt es sich um Rootkits, die auf Usermode-Level arbeiten und Einträge in der IAT verändern. Dabei werden Adressen in der IAT ausgetauscht, sodass Aufrufe nicht mehr auf die eigentlichen Programme und Anwendungen bzw. Funktionen zeigen, sondern auf Schadroutinen des Rootkits. Eingeführt wurde diese Arbeitsweise

von Schadsoftware vom Virus W32/Cabanas[25] und wird oft in anderen Win32 Viren eingesetzt bzw. kann in fast allen Win32 Plattformen angewandt werden.

Funktionsweise: Programme legen die Adressen der meisten importierten APIs in der .idata-Sektion ab (Zeile 8). Die Schadsoftware muss somit nur die dort hinterlegten Adressen durch jene ersetzen, die auf die eigenen APIs zeigen (Zeilen 3 & 4 sowie 9 & 10).

Anwendungsbeispiel IAT nach [34]:

```
1  .text (CODE)
2  0041008E E85A37000    CALL 004137ED
3  004137E7 FF2568004300 JMP [00430068]
4  004137ED FF256C004300 JMP [0043006C]
5  004137F3 FF2570004300 JMP [KERNEL32!ExitProcess]
6  004137F9 FF2574004300 JMP [KERNEL32!GetVersion]
7
8  .idata (00430000)
9  00430068 830DFA77 ;-> 77FA0D83 Entry of New GetProcAddress
10 0043006C A10DFA77 ;-> 77FA0DA1 Entry of New FindFirstFileA
11 00430070 6995F177 ;-> 77F19569 Entry of KERNEL32!ExitProcess
12 00430074 9C3CF177 ;-> 77F13C9C Entry of KERNEL32!GetVersion
13
14 NewJMPTable:
15 77FA0D83 B81E3CF177 MOV EAX,KERNEL32!GetProcAddress ; Original
16 77FA0D88 E961F6FFFF JMP 77FA03EE ;-> New handler
17 .
18 .
19 77FA0DA1 B8DBC3F077 MOV EAX,KERNEL32!FindFirstFileA ; Original
20 77FA0Da6 B9F3F6FFFF JMP 77FA049E ;-> New handler
```

[25] http://www.f-secure.com/v-descs/cabanas.shtml

5.4 Auffinden von Rootkits

Rootkits können oft nur schwer entdeckt werden, besonders wenn sie im Kernel residieren, da sie die Möglichkeit haben, Kernelfunktionen zu beeinflussen, welche von Sicherheitsprogrammen benötigt werden, um Rootkits aufzuspüren. Hoglund und Butler geben in [16] Kapitel 10: "Rootkit Detection" zwei gute Ansätze für das Aufspüren von Rootkits: Auffinden durch Präsenz oder Auffinden durch Verhalten von Rootkits.

5.4.1 Auffinden durch Präsenz

Überwachen der Eintrittspunkte: Da jede Software im Arbeitsspeicher laufen muss, muss sie auch irgendwann einmal dorthin gelangen. Es wird daher empfohlen, alles zu überwachen, was in den Arbeitsspeicher gelangt – Treiber, Prozesse, etc.

Überwachung im laufenden Betrieb: Hier ist das Überwachen des Speichers und der Festplatte gemeint. In periodischen Abständen alle vorhanden Datenbestände zu scannen und auf Unregelmäßigkeiten zu achten, soll die Chance erhöhen, Rootkits aufzufinden.

Finden von Hooks: Ein weiterer Ansatz in diesem Bereich ist das Auffinden von Hooks, welche bereits beschrieben wurden. Wenn potentielle Angriffsstellen wie IAT, SSDT, IDT, IRP, etc. überwacht werden, erhöht sich die Chance, ein Rootkit zu finden.

5.4.2 Auffinden durch Verhalten

Versteckte Dateien und Registrierungseinträge: Das Programm RootkitRevealer[26] ist in der Lage, versteckte Dateien und Registrierungsschlüssel als Rootkitzugehörig zu identifizieren, indem es die gesammelten Werte mit den originalen Referenzdateien und -Schlüssel von Windows vergleicht und so Änderungen

[26] `http://technet.microsoft.com/de-de/sysinternals/bb897445.aspx`

aufspürt. Dieses Vorgehen ist ein zuverlässiger Ansatz zum Auffinden, benötigt jedoch viel Zeit und Ressourcen.

Versteckte Prozesse: Versteckte Prozesse sind mitunter die gefährlichsten Bedrohungen durch Rootkits, weil sie im System aktiv sind, ohne bemerkt zu werden. In [16] werden einige Ansätze gegeben, was man gegen diese Bedrohung unternehmen kann.

Weitere Ansätze sind: "A Normality Based Method for Detecting Kernel Rootkits" [36], welcher unter Linux Loadable Kernel Module (LKM) Rootkits dadurch aufspüren zu können verspricht, dass Sonderfälle bei Performancemessungen die ausschlaggebenden Hinweise liefern; "Defeating Dynamic Data Kernel Rootkit Attacks via VMM-based Guest-Transparent Monitoring" [19], wo mittels VMM-Überwachung des Kernels Attacken von Dynamic Data Kernel Rootkits verhindert werden sollen; oder das Framework "Paladin: Automated Detection and Containment of Rootkit Attacks" [2], auf welche hier aufgrund des Umfanges nicht näher eingegangen wird. In der Literatur sind unzählige weitere Techniken zu finden.

5.5 Anwendungsbeispiele von VMBRs

Bevor vertiefend auf VMBRs eingegangen wird, soll hier noch kurz ein Exkurs zu einer früheren Variante gemacht werden: SMBR-Rootkits. Diese basieren auf Intels SMM (System Management Mode), welche viele Möglichkeiten bietet, Schadsoftware-Entwickler zur Erzeugung von Rootkits zu motivieren. SMM hat einen eigenen privaten Speicherbereich und eine eigene Ausführungsumgebung, welche unsichtbar für Code ist, der außerhalb ausgeführt wird. Weiters ist er nicht preemptiv, unterstützt keine Privilegierungskonzepte und setzt keine Speicherschutzkonzepte um. Diese Eigenschaften machen es zu einer attraktiven Umgebung für Rootkits [9]. Zur Gegenüberstellung von SMBRs und VMBRs:

Spezifizierung/Kontrolle: SMBRs arbeiten eher Chipsatz-spezifisch (IRQ – Interrupt Requests), während VMBRs Prozessor-spezifisch (Prozessorunterbrechungen, Debug-/Control-Register, Speicherzugriff, ...) implementiert werden.

Betriebssysteme und -Umgebungen: SMBRs laufen auf 16-bit Systemen ohne Paging, sind betriebssystem-unabhängig und können die Speicherzugriffe verbergen. VMBRs sind ebenfalls betriebssystem-unabhängig, können die getätigten Speicherzugriffe verbergen (Speichervirtualisierung vorausgesetzt) und laufen auf 32-bit Systemen mit Paging (je nach Implementierung im protected mode).

Einsatz: SMBRs sind hauptsächlich auf Systemen, welche vor 2005 gebaut wurden, lauffähig, VMBRs auf Systemen nach 2006.

Abwehr: Bei SMBRs müssen betroffene bzw. gefährdete Register in der frühen Bootphase des Betriebssystems gesichert werden. Die beste Möglichkeit gegen VMBRs vorzugehen, ist das Installieren einer sicheren VM (bzw. das Implementieren eines VMM), welche die Installation einer VM eines externen Dritten (VMBR) verhindert.

Anwendungsbeispiele von SMM-Attacken, beispielsweise auf Intels TXT (Trusted Execution Technology) findet man in [41] (Attacking SMM Memory via Intel® CPU Cache Poisoning) sowie [43] (Another Way to Cicumvent Intel® Trusted Execution Technology).

Im Folgenden werden VMBRs für Linux und Windows besprochen. Anwendungsbeispiele für Mac OS, wie beispielsweise Vitriol, finden sich in [44].

5.5.1 SubVirt

Ein Kooperationsprojekt zwischen der University of Michigan und Microsoft Reserach im Jahr 2006 war der "Startschuss" für VMBRs. Im Paper "SubVirt: Implementing malware with virtual machines" [20] werden erstmals Rootkits vorgestellt, welche sich nicht als klassische Rootkits in ein System einnisten, sondern die gesamte Kontrolle über ein System übernehmen, in dem sie sich unter das laufende Betriebssystem setzen.

King et al. beschreiben dabei schrittweise, wie man ein VMBR auf einem System installiert und wie man nach erfolgreicher Infiltrierung noch weitere Schadsoftware so einbringt, dass dem Nutzer möglichst viel Schaden zugefügt wird bzw. dieser ausspio-

niert wird. Neben den theoretischen Ansätzen wurden zwei Proof-Of-Concept VMBRs implementiert, jeweils für Linux unter VMware (Gentoo Linux) und für Windows unter VirtualPC (Microsoft WindowsXP Preinstallation-Environment (PE)).

Zur Vorgehensweise (schematische Darstellung in Abbildung 5.1: die Grafik zeigt, wie ein bestehendes Zielsystem in eine VM verschoben wird. Die grauen Bereiche sind dabei Teile des VMBR.): Damit sich ein VMBR etablieren kann, muss es zuerst Zugriff auf das laufende System mit Administratorrechten erlangen. Dies lässt sich beispielsweise mit Hilfe eines Trojaners oder anderen bekannten Schwachstellen sehr leicht bewerkstelligen (vergleiche Kapitel 3.2.2). Danach muss sich das Rootkit im persistenten Speicher des Systems niederlassen. Im Anschluss muss die Boot-Sequenz des Systems dahingehend verändert werden, dass sichergestellt wird, dass das VMBR beim nächsten Neustart vor dem eigentlichen Betriebssystem geladen wird. Mit Administratorrechten kann die Bootsequenz ohne Probleme manipuliert werden.

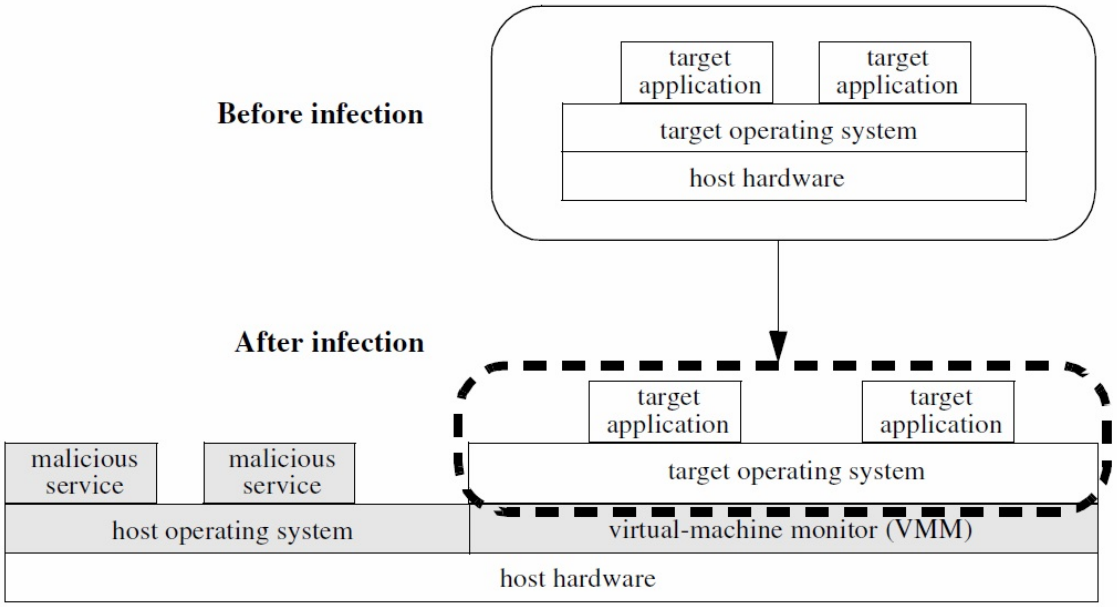

Abbildung 5.1: SubVirt (aus [20])

Viele Sicherheitsprogramme bemerken allerdings Veränderungen in der Boot-Sequenz (vor allem wenn diese nicht vom Betriebssystem selbst vorgenommen werden) – hier bedient sich SubVirt einer sehr praktischen Handhabung: die Boot-Sequenz wird nicht im

laufenden Betrieb verändert, sondern in den letzten Phasen des Herunterfahrens, nachdem die meisten Kernelsubsysteme, Prozesse und Sicherheitsprogramme schon beendet wurden. Bei WindowsXP wird hierfür das Kernel-Modul `LastChanceShutdownNotification` genutzt. Bei Linux-Systemen werden im User-Mode die Shutdown-Scripts geändert, sodass beim nächsten Start die geänderte Boot-Sequenz greift. Einmal infiziert, kann das System für alle Arten von Schadsoftware verwendet werden.

Da Schadsoftware auf längerfristige Aktivitäten ausgelegt ist, verfügen die meisten VMBRs über Mechanismen, welche deren Entfernung verhindern sollen. Die einzige Zeitspanne, in der ein VMBR keine Kontrolle über das System hat, ist jene zwischen dem Einschalten der Maschine und dem Laden des VMBRs. Jede Routine, welche in dieser Zeit ausgeführt wird, wird nicht vom VMBR beherrscht, was zu dessen Umgehung führt. Beispielsweise startet in dieser Zeitspanne das BIOS (Basic Input Output System), welches entscheidet, von welchem Medium gestartet werden soll. Wird ein Datenträger mit einem lauffähigen Betriebssystem eingelegt, von dem zuerst gebootet wird (beispielsweise Knoppix[27]), so unterliegt das VMBR diesem Betriebssystem und ist machtlos.

Da VMBRs in dieser Zeitspanne Kontrolle über das System verlieren, versuchen sie die Anzahl der Systemneustarts zu minimieren. Dazu geben sie dem Benutzer einfach ein Ausschalten des Systems vor, wenn dieser das System herunterfahren möchte, fahren das System aber nicht wirklich nieder. Der Bildschirm wird ausgeschaltet, die Lüfter schalten sich ab, die Festplatte arbeitet nicht mehr, etc., aber der Arbeitsspeicher bleibt mit dem aktiven VMBR intakt. Beim erneuten Einschalten wird dem Benutzer ein normaler Start vorgetäuscht, das VMBR hat aber niemals die Kontrolle des Systems verloren. Versucht der Benutzer in diesem Fall von einem anderen Medium zu booten, unterliegt das alternative Betriebssystem trotzdem dem Einfluss des VMBRs. Um hier auf Nummer sicher gehen zu können, muss ein tatsächlicher Kaltstart des Systems erzwungen werden, was meist nur durch eine Trennung von der Stromversorgung gewährleistet wird.

In der Evaluierung nach [20] gibt es teilweise gravierende Geschwindigkeitsunterschiede, je nachdem ob ein System sauber oder mit VMBR ausgeführt wird. Für Standard-

[27] `http://www.knoppix.org/`

Benutzer ist kein Unterschied im Systemverhalten bemerkbar, wohingegen erfahrene Benutzer Verdacht schöpfen könnten. Die VMBRs dieser Zeit müssten in Performanz noch deutlich verbessert werden, damit Benutzer nicht mehr auf die Änderungen aufmerksam werden. Beispielsweise sind Anwendungen mit hoher 3D-Grafikanforderung durchaus von der Auswirkung von VMBRs in ihrer Geschwindigkeit betroffen.

Auffinden von VMBRs VMBRs sind in der Regel viel schwerer aufzufinden als traditionelle Rootkits, weil sie die Umgebung des Betriebssystems virtualisieren und ein VMBR meist nichts innerhalb eines Zielsystems verändert. Es gibt trotzdem einige Anhaltspunkte, mit deren Hilfe VMBRs aufgespürt werden können:

Sicherheit unter dem VMBR: Die einfachste Möglichkeit, ein VMBR aufzufinden ist, Sicherheitssoftware eine Privilegierungsschicht darunter laufen zu lassen. Beispiele hierfür sind Programme, welche den phyischen Speicher auslesen, die reale Festplatte untersuchen können, Kontrolle über die Bootsequenz haben oder Betriebssysteme, welche von einem statischen Medium booten. Eine andere Möglichkeit ist, einen sicheren Hypervisor zu verwenden.

Sicherheit über dem VMBR: Ein VMBR in dessen kontrollierter Umgebung aufzuspüren zählt zu den schwierigsten Aufgaben, da es hierfür beinahe keine Optionen gibt. Die naheliegendste und effizienteste Möglichkeit diesbezüglich ist eine Analyse der Start- und Antwortzeiten des Systems, da durch das Vorhandensein eines VMBR diese durchaus länger ausfallen können, als bei sauberen Systemen.

5.5.2 Red Pill & Blue Pill

Die Computersicherheitsexpertin Joanna Rutkowska erregte im Bereich der VMBRs sowie der virtualisierenden Rootkits im Jahr 2006 bei diversen Black Hat Konferenzen[28] große Aufmerksamkeit mit der Vorstellung des so genannten *Blue Pill Rootkits*, welches Schwachstellen von Microsoft Windows Vista x64 (unter AMD SVM/Pacifica) ausnutzt und eine nicht detektierbare Malware in Form eines Thin-Hypervisors on-the-

[28] `http://www.blackhat.com/`

fly installiert und so die Kontrolle über das laufende Betriebssystem übernimmt. Dabei wird vom Rootkit so viel Arbeitsspeicher allokiert, dass sensible Kernel-Codeteile auf die Festplatte ausgelagert werden müssen. Dort können sie, im Gegensatz zum Arbeitsspeicher, problemlos verändert werden. Nachdem der Kernel-Code benötigt und erneut in den Hauptspeicher geladen wird, werden die Änderungen ausgeführt und das Rootkit erhält Kontrolle über das System. Die Vorstellung erfolgte über Rutkowskas Blog in mehreren Schritten: "Introducing Blue Pill" [29] im Juni, "The Blue Pill Hype" [30] im Juli, sowie "Blue Pill Detection!" [31] im August 2006.

Im Kinofilm Matrix (1999) wird der Hauptcharakter vor die Wahl zur Einnahme einer blauen oder roten Pille gestellt:

> *"This is your last chance. After this, there is no turning back. You take the blue pill — the story ends, you wake up in your bed and believe whatever you want to believe. You take the red pill — you stay in Wonderland and I show you how deep the rabbit-hole goes."* [8]
>
> (Das ist deine letzte Chance. Danach gibt es kein Zurück. Schluckst du die blaue Kapsel, ist alles aus – du wachst in deinem Bett auf und glaubst an das, was du glauben willst. Schluckst du die rote Kapsel, bleibst du im Wunderland – und ich führe dich in die tiefsten Tiefen des Kaninchenbaus.)

Diese Analogie verwendete auch Rutkowska für ihre beiden Projekte. Während *Red Pill* [31] herausfindet, ob ein Betriebssystem in einer virtualisierten Umgebung ausgeführt wird, installiert *Blue Pill* [32] einen Thin-Hypervisor der die Kontrolle übernimmt, ganz ohne irgendwelche Änderungen im BIOS oder MBR vornehmen zu müssen, auch wird kein Neustart (vergleiche SubVirt) des Systems benötigt. Es wird dabei unsignierter Kernel-Code ins System eingeschleust, in dem soviel Arbeitsspeicher vom Rootkit angefordert wird, dass der sichere, signierte Windows-Kernel-Code auf die Festplatte ausgelagert werden muss, wo eine potenzielle Möglichkeit einer Angriffsfläche entstehen kann.

[29] `http://theinvisiblethings.blogspot.co.at/2006/06/introducing-blue-pill.html`
[30] `http://theinvisiblethings.blogspot.co.at/2006/07/blue-pill-hype.html`
[31] `http://theinvisiblethings.blogspot.co.at/2006/08/blue-pill-detection.html`

Der Source-Code von Red Pill:

```
/* VMM detector, based on SIDT trick
 * written by joanna at invisiblethings.org
 *
 * should compile and run on any Intel based OS
 *
 * http://invisiblethings.org
 */

#include <stdio.h>
int main () {
  unsigned char m[2+4], rpill[] = "\x0f\x01\x0d\x00\x00\x00\x00\xc3";
  *((unsigned*)&rpill[3]) = (unsigned)m;
  ((void(*)())&rpill)();

  printf ("idt base: %#x\n", *((unsigned*)&m[2]));
  if (m[5]>0xd0) printf ("Inside Matrix!\n", m[5]);
  else printf ("Not in Matrix.\n");
  return 0;
}
```

Das Kernstück der Red Pill Funktionsweise basiert dabei auf den SIDT Anweisungen (Store Interrupt Descriptor Table), welche im Ring 3 ausgeführt werden, allerdings Informationen über das vom Betriebssystem verwendete Register zurückliefert. Da es nur ein IDTR Register gibt (Interrupt Descriptor Table Register), allerdings zwei Betriebssysteme darauf zugreifen (Host und Gast), muss der VMM das Register neu unterbringen (Adressübersetzung wird verwendet), was dazu führt, dass der unprivilegierte Gast nun Zugriff auf einen privilegierten Registerbereich bekommt.

5.5.3 Xen 0wning Trilogy

Bei der Hackerkonferenz BlackHat 2008 in Las Vegas[32] stellte das Team der *Invisiblethingslabs* (Rafal Wojtczuk, Joanna Rutkowska, Alexander Tereshkin)[33] in drei Teilen umfassende Möglichkeiten zur Unterwanderung des Xen Hypervisors vor. Das zugehörige wissenschaftliche Dokument wurde von Wojtczuk verfasst [40]. BluePill und Vitriol sind gute Beispiele dafür, einen bösartigen VMM auf einem System auszubringen, wo kein Hypervisor vorhanden ist, daher stellt sich die Frage, was ist mit Systemen, auf denen bereits ein Hypervisor läuft? Es herrscht die Meinung, dass künftig alle Systeme mit irgend einer Art Hypervisor arbeiten werden. In diesem Projekt wird beschrieben, welchen Möglichkeiten und Hindernissen künftige Angreifer gegenüberstehen, um Systeme mit bereits existierendem Hypervisor zu infiltrieren.

Part 1: Subverting the Xen Hypervisor: Wojtczuk beschreibt, wie einfach mittels der Manipulation von DMA (Direct Memory Access) eine Backdoor in Xen-Code eingebracht werden kann; unter anderem unter Zuhilfenahme der Netzwerkkarte. Weitere beschriebene Möglichkeiten, in das System zu gelangen sind mittels dem *Xen Loadable Module Framework* (dieses ermöglicht es, kompilierten C-Code in Xen zu laden, ähnlich wie bei den *Linux Loadable Kernel Module (LKM)*) und über die Modifizierung der *Debug Register* (DR3 und DR7). Als weitere Methode wird eine Vorgehensweise mittels "fremden Backdoors" vorgestellt, da Verändungen im System durch die DR-Backdoors durch Performance-Analysen festgestellt werden können. Xen bietet eine API für die Speicheranalyse von fremden Domains an.

Part 2: Preventing and Detecting Xen Hypervisor Subversions: In diesem Teil geben Rutkowska und Wojtczuk eine Anleitung, welche Möglichkeiten man mit der Intel VT-d Technologie hat um den Xen Hypervisor zu schützen und wie dieser Schutz umgangen werden kann. Es wird gezeigt wie über SMM Rootkits eingebracht werden können. Weiters wird ein Vergleich zwischen Xen, Microsofts Hyper-V und VMWares ESX/vSphere gezogen, welche der Virtualisierungstech-

[32] `https://www.blackhat.com/html/bh-usa-08/bh-usa-08-archive.html`
[33] `http://theinvisiblethings.blogspot.co.at/2008/08/our-xen-0wning-trilogy-highlights.html`

nologien potentiellen Schwachstellen aufweist [42]. Ausserdem wird ein BIOS für Hypervisoren mit dem Namen HyperGuard vorgestellt, welches zum Schutz vor VMBRs und ähnlichen Attacken dienlich sein soll.

Part 3: Bluepilling the Xen Hypervisor: Hier wird von Rutkowska und Tereshkin eine neue Variante von BluePill eingeführt – Blue Pill Boot – welche aus einer VM heraus die Management Domain von Xen dahingehend verändert, dass die Bootreihenfolge beim nächsten Start zuerst das BluePill Rootkit aufruft anstatt den Xen Hypervisor (dom0).

5.5.4 Weitere Attacken

In [29] "Following the White Rabbit: Software attacks against Intel VT-d technology" beschreiben Wojtczuk und Rutkowska weitere Möglichkeiten, den Xen Hypervisor auf einer Intel VT-d hardwaretechnologie-basierten Plattform mittels SIPI (Start-up Inter Processor Interrupt), manipulierten System-Calls und #AC-Exception Injektionen zu unterminieren.

Abschließend kann festgestellt werden, dass die Bedrohung durch VMBRs immer noch aktuell ist, diese Art von Malware aber ihren ursprünglichen Zweck verfehlt hat: VMBRs sind heute weniger als Bedrohung anzusehen sondern vielmehr als ein Anstoß für die Entwickler von Betriebssystemen und Virtualisierungslösungen, noch viel mehr Bedacht auf die Sicherheit ihrer Systeme zu nehmen.

6 Virtualisierung & Cloud Computing

Immer mehr Dienste werden über Clouds angeboten, welche über das Internet erreichbar und abrufbar sind. Warum auch diese Technologien von Sicherheitsbedenken bzgl. Virtualisierungen betroffen sind, soll in diesem Abschnitt erläutert werden.

6.1 Definition Cloud Computing

Während im IT-Bereich immer noch versucht wird, eine genaue Definition einer "Cloud" zu geben, gibt es schon einige sehr gute Ansätze diesbezüglich, wobei es die offizielle NIST-Definition (National Institute of Standards and Technology [34]) am besten trifft:

> *"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources [...] that can be rapidly provisioned and released with minimal management effort or service provider interaction."* [25]

Das Cloud Modell lässt sich weiter anhand fünf Charakteristika, drei Service Modellen und vier Ausbringungsarten beschreiben:

Charakteristika: *On-demand Self-service* (jeder Nutzer kann beliebig viele Ressourcen aus der Cloud anfordern und bekommt diese zugeordnet), *Resource Pooling* (alle Ressourcen des Cloud Anbieters sind in einem großen "Pool" verfügbar – so-

[34] `http://www.nist.gov/`

mit ist der Anbieter in der Lage, alle Anfragen der Nutzer befriedigen zu können – die physischen Ressourcen jedoch sind verteilt), *Rapid Elasticity* (Kapazitäten können jederzeit und einfach zugeteilt und wieder freigegeben werden – der Nutzer hat den Eindruck, unbegrenzte Ressourcen zur Verfügung zu haben), *Broad Network Access* (damit keine Geschwindigkeitseinbußen auftreten, ist meist eine enorme Bandbreite gegeben, um allen Anfragen gerecht zu werden) sowie *Measured Service* (die Vergabe und Kontrolle von Ressourcen wird meist automatisiert durchgeführt; weiters wird die Auslastung und Nutzung aller Komponenten überwacht und bietet somit Transparenz sowohl für den Anbieter als auch für die Nutzer).

Service Modelle: *Software as a Service – SaaS* (es werden Dienste angeboten, welche über diverse Geräte und Schnittstellen erreichbar sind, beispielsweise ein Webmail-Service; der Nutzer hat keinerlei Kontrolle über das Betriebssystem, Ressourcen oder sonstige Infrastruktur), *Platform as a Service – PaaS* (dem Nutzer wird ermöglicht, über eine vorgegebene (Programmier-)Schnittstelle Programme in die Cloud zu laden; in der Cloud selbst wird im Anschluss für die Ressourcenzuteilung gesorgt), und *Infrastructure as a Service s- IaaS* (der Nutzer hat ähnlich wie bei SaaS und PaaS keine Kontrolle über die darunterliegende Cloud-Infrastruktur, er hat aber viele Rechte bzgl. Verwaltung von Ressourcen, Netzwerkkapazitäten und Speicher). IaaS ist die für Virtualisierungssicherheit interessanteste Architektur, da es von den meisten bekannten Cloudanbietern genutzt wird und, falls es von Schadsoftware infiltriert wird, auch die darauf aufbauenden Architekturarten (Paas, SaaS) betroffen sind.

Ausbringungsarten: *Private Cloud* (die Infrastruktur wird nur einer exklusiven Anzahl von Nutzern zur Verfügung gestellt – verwaltet durch beispielweise eine einzige Organisation), *Community Cloud* (die Infrastruktur wird nur einer bestimmten Art von Nutzern zur Verfügung gestellt – verwaltet beispielsweise durch einen Zusammenschluss von Organisationen oder Unternehmen, welche das selbe Ziel verfolgen), *Public Cloud* (wird der Öffentlichkeit angeboten – meistens im Besitz von staatlichen Einrichtungen oder öffentlichen Bildungseinrichtungen) und *Hybrid Cloud* (die Infrastruktur ist eine Mischung aus den anderen drei Ausbringungsarten, beispielweise für Lastenausgleich zwischen Clouds).

6.2 Sicherheitsprobleme durch Virtualisierung

Virtualisierungssicherheit ist für Anbieter von Cloud-Diensten wichtig, da viele VMs auf einer physischen Maschine ausgeführt werden. Dies eröffnet Malware bzw. einem Angreifer viele Möglichkeiten, Schaden zu verursachen. Wenn man Dienste eines Cloud-Anbieters nutzt, hat man weder Kontrolle darüber, auf welchen physischen Maschinen man tatsächlich arbeitet, noch wie viele andere Instanzen von VMs auf der selben Maschine laufen, bzw. wer im Besitz dieser ist.

Auch wenn man ein Großkunde ist und viele Instanzen von VMs nutzt, weiß man nie, ob und wie viele andere Kunden auf der selben bzw. auf der einen physischen Maschine angesiedelt sind, oder auf wie vielen Maschinen die eigenen Daten verteilt sind. Die Tatsache, dass meist auch andere Nutzer auf einer physischen Maschine angesiedelt sind, verstärkt die Bedenken hinsichtlich Sicherheit, da eine infizierte Instanz einer VM immer eine Bedrohung für andere Instanzen darstellen kann, im schlimmsten Fall ist sogar eine Infiltrierung des Hypervisors möglich, die beträchtliche Auswirkungen auf die anderen VMs hat.

In Abbildung 6.1 sind insgesamt sechs unterschiedliche Arten von Angriffsmethoden bzw. Arten von Infiltrierungen dargestellt, auf die im Folgenden kurz eingegangen wird. Da die Architektur von kommerziellen Cloud-Anbietern nicht öffentlich zugänglich bzw. bekannt ist, wurde eine vereinfachte Darstellung gewählt.

Es wird angenommen, dass sich immer mehrere Instanzen von VMs auf einer physischen Maschine befinden - in der Abbildung sind der Übersichtlichkeit wegen immer nur jeweils vier Instanzen abgebildet.

a.) VM zu VM über das Internet: Wenn eine Maschine (oder VM) infiltriert wurde, können von Malware die Ressourcen gegen das Internet bzw. beliebige Ziele verwendet werden, aber auch gegen VMs auf der selben physischen Maschine.

b.) Aus dem Internet: Wenn ein Angreifer keine Kontrolle über eine VM hat, so besteht dennoch die Möglichkeit, von außen anzugreifen. In diesem Fall ist es egal, ob der Angriff tatsächlich von außen oder von einer VM auf der selben physischen Maschine kommt.

c.) Von der VM nach außen: Diese Angriffsart stellt eine typische Verwendung für (D)DoS-Attacken ((Distributed) Denial of Service) oder einen Teil eines Bot-Nets (Spam, etc.) dar.

Diese drei Arten stellen klassische Attacken dar, welche sich weniger auf Virtualisierungssicherheit beziehen und werden daher nicht weiter behandelt.

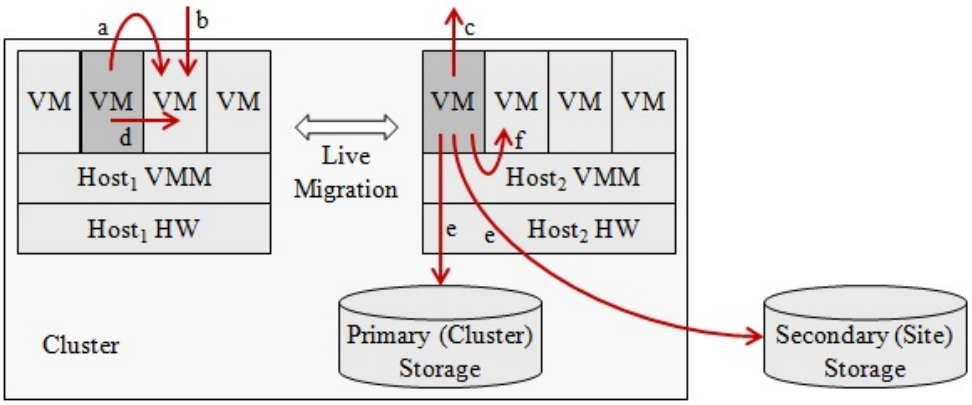

Abbildung 6.1: Vereinfachte Darstellung eines Cloud Clusters (aus [17])

d.) VM zu VM direkt: Obwohl diese Art von Ausbruch einer Malware möglich ist, gibt es bisher keine konkreten Fälle, in denen eine VM eine andere VM infiziert hat, ohne über das Internet oder über den VMM zu kommunizieren.

e.) VM zu externen Speichern: Bei den externen Speichern handelt es sich beim Primary Storage um einen gemeinsam geteilten Speicherbereich, beispielsweise ein geteilter Ordner, eine Funktionalität, welche in den meisten Virtualisierungslösungen angeboten wird (Shared Folder). Beim Secondary Storage hingegen handelt es sich um eine Art Auslagerungsspeicher, in dem Maschinen (zwischen-) gelagert werden, während diese inaktiv sind oder während ein Migrationsvorgang läuft.

   Das Problem beim Shared Folder ist, dass eine infizierte VM die Möglichkeit hat, andere, nicht infizierte Maschinen zu infiltrieren und der Malware einfache Möglichkeiten eröffnet werden, sich zu verteilen. Beim Secondary Storage besteht im Normalfall keine Gefahr, da inaktive VMs bzw. darauf residierende Malware

weder Schaden anrichten noch sich verbreiten kann, allerdings besteht die Gefahr, dass eine infizierte, abgeschaltete Maschine auf einer sauberen, physischen Maschine reaktiviert wird und dort Schaden anrichtet.

f.) VM zu VM über VMM: Dieses Szenario stellt den schlimmsten Fall von ausbrechender Malware dar. Wenn es einer Schadsoftware gelingt, via VM-Escape oder VMBRs tatsächlich Kontrolle über den Hypervisor zu erlangen, so ist jegliche Sicherheit der physischen Maschine weg. Dies hätte zur Folge, dass für die restliche Cloud keine Sicherheit mehr garantiert werden kann.

VM-Escapes werden in Kapitel 4.6 behandelt, weitere Sicherheitsbedenken bzgl. Clouds und Virtualisierungstechnologien werden in [18] behandelt.

6.3 Möglichkeiten zum Schutz und zur Prävention

Neben den bekannten Techniken von Intrusion Detection Systemen (IDS) bzw. Intrusion Prevention Systemen (IPS) und deren Pendants für virtualisierte Umgebungen (vIDS/vIPS) – diese werden in [6] als eine Möglichkeit zur Gewährleistung von Sicherheit in Cloud Computing Systemen verwendet – gibt es noch Ansätze für Mischformen bzw. Zusammenlegungen der Systeme zu Intrusion Detection/Prevention Systemen (IDPS). Diese bekannten Vorgehensweisen werden hier nicht näher beschrieben.

Weiter unten in diesem Abschnitt werden zwei ausgewählte Verfahren für Sicherheit in Clouds vorgestellt; keinesfalls als vollständige Auflistung von Möglichkeiten, sondern vielmehr als Beispielgebung. In [23] wird eine Einführung einer zusätzlichen Sicherheitsschicht zwischen VMM und den Instanzen der VMs vorgeschlagen, was jedoch nur eine rudimentäre Vorgehensweise ist, da Sicherheit immer ein Teil bzw. Aufgabe des Hypervisors ist und eine Auslagerung dieser keinen Sinn macht.

6.3.1 Virtual Machine Introspection (VMI)

Bei VMI werden die Instanzen der VMs von außen überwacht und es wird versucht, Infiltrierungen durch Malware festzustellen, indem ungewöhnliches Verhalten der VMs bzw. deren Kernels festgestellt wird. Mit dieser Vorgehensweise bzw. mit den so ermittelten Daten hat man die Möglichkeit, Angriffe in ihrer Anfangsphase aufzuspüren und so können vorzeitig Maßnahmen ergriffen werden. Da erwartet wird, dass Angriffe zumeist mittels bereits bekannter Schadsoftware ausgeführt werden, hat ein VMI-basiertes IDPS gute Chancen, den Angriff frühzeitig zu erkennen. VMI entspricht somit dem Modell, bei dem die Sicherungsmaßnahmen in einer Stufe ausgeführt werden, welche zumindest um eine Ebene mehr Privilegien hat als die zu überwachende VM.

6.3.2 Self Cleansing for Intrusion Tolerance (SCIT)

SCIT[35] ist ein Forschungsprojekt an der George Mason Universität in Virginia, welches in erster Linie auf Redundanzen abzielt. In der Annahme, dass jede Instanz einer VM nach einer gewissen Zeit online als infiltriert angesehen werden muss, muss diese anschließend vom Netz genommen werden, damit sie gereinigt werden kann. Erst nach der Bereinigung und Sicherstellung, dass keine Schadsoftware mehr auf der VM vorhanden ist, kann die Instanz wieder auf ein (vermeintlich sauberes) physisches System online gegeben werden.

Die Effizienz von SCIT basiert auf konstanten Rotationsalgorithmen, damit die Zeitfenster für Attacken minimiert werden. Das bedeutet, dass die Instanzen, abhängig davon wie vertrauenswürdig ein System, welches aus mehreren VMs besteht, zurzeit ist (ohne auf die konkrete Einstufung einzugehen, wann ein System wie vertrauenswürdig ist), unterschiedlich oft offline genommen und gereinigt werden.

SCIT ist kein Ersatz für konventionelle Systeme zur Abwehr von Schadsoftware, es soll lediglich helfen, die Gesamtsicherheit auf Systemen zu erhöhen.

[35] `http://cs.gmu.edu/~asood/scit/`

6.3.3 Advanced Cloud Protection System (ACPS)

Die Autoren von [22] stellen einen weiteren interessanten Ansatz zur Erhöhung der Sicherheit bei Clouds bzw. Virtualisierung vor: sie haben eine neue Architektur entwickelt, welche sich Advanced Cloud Protection System nennt und auf einem hosted Hypervisor für zusätzliche Sicherheit sorgt. Das System ist dabei hostbasiert und baut auf dem Prinzip von VMI auf, wobei es als Zusatzprogramm, welches im Hostbetriebssystem läuft, mittels Überwachung des Virtualisierungsdienstes für Sicherheit sorgen soll.

Diese Variante ist zwar ein guter Ansatz um für mehr Sicherheit zu sorgen, jedoch kann diese Vorgehensweise nicht bei bare-metal Hypervisoren angewandt werden und, wie bereits erwähnt, sollte Sicherheit immer Angelegenheit des VMM sein und nicht als zusätzliches Programm oder als weiterer Service Layer umgesetzt werden.

7 Ausblick

In diesem Kapitel soll ein Ausblick auf virtualisierende Systeme in Bezug auf Sicherheit und ein kurzer Einblick in mobile Systeme und deren Anwendung von Virtualisierungen gegeben werden.

7.1 Virtualisierung bei Mobilen System

Mobile Geräte sind heutzutage ständige Begleiter. Es gibt kaum Menschen, welche nicht zumindest ein Mobiltelefon bei sich tragen. Nach der Entwicklung im mobilen Bereich in den letzten Jahren sind Smartphones nicht nur mehr mobile Telefone, sondern eigentlich mobile Computer. Dieser Umstand lässt sich noch viel mehr bei Tablets beobachten. Da auch die Leistung auf diesen kleinen Geräten immer besser wird, lassen sich inzwischen viele Dienste mittels Virtualisierung realisieren.

Ein Aspekt, auf den hier nicht näher eingegangen wird ist, dass mobile Geräte auch anfällig für jegliche Art von Malware sein können. Da es sich um mobile Computer handelt, unterscheiden sich die Geräte in dieser Hinsicht kaum noch von Notebooks – für Schadsoftware ist jedes Gerät gleich anfällig.

7.1.1 Advanced RISC Machines (ARM)

Diese Architektur wird hauptsächlich auf 32bit Mikroprozessoren angewandt, welche vorwiegend in mobilen Geräten (Smartphones und Tablets) zum Einsatz kommen. Grund für die häufige Verwendung dieser Mikroprozessor-Architektur ist die hohe Leis-

tungsfähigkeit bei verhältnismäßig geringem Energieverbrauch. ARM implementiert in der Version 8 seit 2013 erstmals auch eine 64bit-Architektur[36].

KVM für ARM

In [7] wird eine Virtualisierungsmöglichkeit für ARM Prozessoren vorgestellt: KVM für ARM (KVM/ARM). Mit dieser Technologie wird ermöglicht, Gäste nahezu unmodifiziert mittels Lightweight-Para-Virtualisierung auf ARM Prozessoren auszuführen. Leightweight-Para-Virtualisierung ist eine skriptbasierte Methode, den Quellcode eines Betriebssystem automatisch dahingehend zu modifizieren, dass das Betriebssystem in einer virtualisierten Umgebung ausgeführt werden kann. Diese Art der Para-Virtualisierung ist architektur-bezogen, aber betriebssystem-unabhängig. Im Vergleich dazu ist Para-Virtualisierung sowohl architektur- als auch betriebssystem-abhängig, benötigt genaue Kenntnis über die Kernel-Arbeitsweise des Gastsystems und muss laufend gewartet werden, damit alle Betriebssystemaktualisierungen auch weiterhin paravirtualisiert werden können.

Die größte Herausforderung bei diesem Projekt war es, Virtualisierung auf ARM zu ermöglichen, obwohl ARM Virtualisierung eigentlich nicht unterstützt. Eine Architektur, welche Virtualisierung unterstützt, ermöglicht es den Gästen, Zugriff auf die Hardware zu haben, während der VMM volle Kontrolle über die CPU behält. Dies wird bei den meisten Virtualisierungsarten umgesetzt, indem der VMM im privilegierten Modus und die Gäste im nicht-privilegierten Modus ausgeführt werden. Diese Funktionalität bietet ARM in seiner nativen Ausführung nicht an. Die Autoren von [7] präsentieren die Umsetzung der KVM/ARM CPU Virtualisierungsunterstützung und Speichervirtualisierung.

Xen für ARM

Der Autor von [12] untersucht die technischen Fähigkeiten von StrongARM dahingehend, ob die paravirtualisierenden Eigenschaften von Xen auf dieser Mikroprozessoren-

[36] `http://de.wikipedia.org/wiki/ARM-Architektur`

architektur umsetzbar sind und so VMs unterstützt bzw. deren Anwendung ermöglicht werden kann. Das Ziel war es, unmodifizierte Gäste auf einem StrongARM basierten Hypervisor ausführen zu können.

7.1.2 Sicherheit bei mobilen Geräten durch Virtualisierung

Die Autoren von [3] argumentieren dahingehend, dass der Trend, mobile Anwendungen und Geräte immer quelloffener zu gestalten (wie auch bei Linux), dazu führt, dass den laufenden Betriebssystemen nicht vollständig hinsichtlich Sicherheit vertraut werden kann. Zum einen gibt es natürlich eine aktive Community, welche an der Weiterentwicklung arbeitet, es kann aber nie garantiert werden, dass alle möglichen Sicherheitslücken geschlossen werden. Sicherheit ist deswegen besonderes bei mobilen Geräten notwendig, weil Nutzer meist viele persönliche Daten auf diesen gespeichert haben, mehr als beispielsweise auf stationären Arbeitsgeräten.

Eine weitere Herausforderung ist, dass es nicht nur eine konkrete mobile, sondern viele unterschiedliche Virtualisierungsplattformen gibt, die genutzt werden können. Auch das Ausmaß der Virtualisierung ist unterschiedlich: manche Ressourcen werden vollständig virtualisiert, manche paravirtualisiert, manche gar nicht virtualisiert.

FlaskDroid

Die Verwendung von SELinux (FLASK) auf Android wird in [4] behandelt. Die Autoren besprechen das Fehlen einer systemweiten verpflichtenden Zugangskontrolle (Mandatory Access Control, MAC) bei Android Systemen. Mit dem Framework FlaskDroid (FLASK für Android) soll dem vorherrschenden Problem Abhilfe geschaffen werden.

VMware Horizon View

Diese Anwendung ermöglicht beispielsweise einem mobilen Gerät, welches mit Android arbeitet, die Herstellung einer Remote-Desktop-Verbindung mit einem Arbeitsplatz-

rechner innerhalb eines Unternehmens[37]. Bei Horizon View handelt sich eigentlich um eine Lösung für Desktop-Virtualisierung, die um die Kompatibilität auf mobilen Systemen erweitert wurde.

Cells

Grundgedanke von Cells ist es, dass Menschen oft mehr als ein Mobilgerät mit sich herumtragen (beispielsweise ein Firmentelefon und ein Privattelefon), vorwiegend um Kontaktdaten und Telefonabrechnungen voneinander getrennt zu haben. Mittels Cells wird es den Benutzern ermöglicht, auf nur einem Gerät die selben Anforderungen erfüllt zu bekommen. Es ist das erste vollständige Framework für Betriebssystem-Virtualisierung auf mobilen Geräten [1].

Cells ist eine Virtualisierungs-Architektur, welche es erlaubt, mehrere virtuelle Smartphones (VP) auf einem physischen Smartphone voneinander isoliert auszuführen. Dabei läuft immer ein VP im Vordergrund – aufgrund der kleinen Displaygröße von Smartphones wird auch nicht erwartet, dass jemals mehr als eine Applikation bzw. mehr als ein VP gleichzeitig angezeigt werden müssen. Es unterstützt dabei vollständige 3D Grafikbeschleunigung, volle Energiemanagementfunktionen und alle Anforderungen, die man zum Telefonieren braucht – inklusive unterschiedlich zuweisbaren Telefonnummern auf dem selben Gerät. Dafür wird ein Voice Over IP (VoIP) Service genutzt, der es ermöglicht, mit nur einer Subscriber Identity Module (SIM) Karte mehrere Nummern zu verwalten.

Die Virtualisierung erfolgt bei Cells dahingehend, dass nicht mehrere Betriebssystem-Instanzen benötigt, sondern mittels Lightweight-Betriebssystem-Virtualisierung mehrere Betriebssystem Namespaces verwendet werden können. Cells isoliert dabei die einzelnen VPs voneinander, damit sich Schadsoftware nicht ausbreiten kann (vergleiche Prinzip des Sandboxings in Kapitel 3.3).

[37] `http://www.vmware.com/products/horizon-view`

7.2 Virtualisierungssicherheit in Zukunft, Resümee

Dieser Punkt ist ein sehr spekulatives Themengebiet. Da man nicht sagen kann, was die technologische Entwicklung von Virtualisierungen in Zukunft bringt, kann auch nur wenig über die spezifische Sicherheit ausgesagt oder vermutet werden. Da auch nicht mit großen technologischen Neuerungen oder Durchbrüchen gerechnet wird, kann man davon ausgehen, dass auch in Bezug auf Sicherheit keine bahnbrechenden Innovationen stattfinden werden.

Der sicherste Weg ist nach wie vor, wenn man als Verantwortlicher der Betreuung von Virtualisierungsanwendungen darauf achtet, dass alle Komponenten eines Systems als sicher angesehen werden können. Die in Kapitel 3 besprochenen Anliegen zur Erhaltung der Sicherheit bzw. Punkte auf die geachtet werden müssen, decken einen Großteil der nötigen beachtenswerten Bereiche ab, welche in Bezug auf Sicherheit besonders betreut werden sollten.

Welche Folgen das Auffinden von langwierigen Sicherheitslücken, wie beispielsweise der Heartbleed-Bug[38] im Frühjahr 2014 mit sich bringt, bzw. welche Schäden dadurch bei virtualisierten Systemen entstanden sind, kann nicht beurteilt oder abgeschätzt werden, in der Hoffnung, dass Vorkommnisse wie diese seltenste Einzelfälle bleiben.

Solange alle aktiven Systeme auf dem aktuellen Stand (Updates, etc.) gehalten werden, für ausreichend konventionelle Sicherheit (Anti-Virenprogramme, Firewall, etc.) gesorgt ist und die Systeme sowohl einzeln im Detail, als auch in ihrer Gesamtheit regelmäßig auf mögliche Schwachstellen untersucht und gefundene Mängel schnellstmöglich behoben werden, kann man im Allgemeinen davon ausgehen, dass das System sicher ist.

Besonders gefährlich anzusehen sind VM-Escapes (wie in Kapitel 4.6 besprochen), welche Kontrolle über den Hypervisor erlangen können. Vor ihnen sollten sich Betreiber von Virtualisierungsumgebungen, aber viel mehr noch Betreiber von Cloud-Services, in Acht nehmen und ihre Systeme besonders davor schützen. Sollte Malware tatsächlich ein VM-Escape in einer virtualisierten Umgebung gelingen, kann das gesamte physische

[38] `http://www.us-cert.gov/security-publications/Heartbleed-OpenSSL-Vulnerability`

System nicht mehr als sicher angesehen werden. Neben der technischen Katastrophe und dem immensen Aufwand der Wiederherstellung ist auch noch der Schaden für die Konsumenten zu beachten, der in weiterer Folge schwere finanzielle Konsequenzen für den Betreiber der Cloud mit sich ziehen kann.

Obwohl VMBRs und VM-Escapes selten als geläufige Attacken "in the wild" auftauchen, besteht, wie proof-of-concept Versuche gezeigt haben, die technische Möglichkeit für solche Vorfälle. Die Entwickler von Schadsoftware werden in Zukunft vermutlich auch nicht davon absehen, weitere Methoden zu entwickeln um Systeme zu infiltrieren.

In Zukunft wird dieses Thema immer mehr an Bedeutung gewinnen, auch wenn voraussichtlich keine innovativen Entwicklungen in Bezug auf Sicherheit bei Virtualisierungen kommen werden. Immer mehr Dienste werden (beispielsweise innerhalb von Unternehmen) virtualisiert oder generell über Cloud-Dienste, welche Virtualisierung nutzen, angeboten, was deren Sicherheit unabdinglich voraussetzt.

Literaturverzeichnis

[1] Jeremy Andrus, Christoffer Dall, Alexander Van't Hof, Oren Laadan, and Jason Nieh. Cells: A virtual mobile smartphone architecture. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 173–187, New York, NY, USA, 2011. ACM.

[2] Arati Baliga, Xiaoxin Chen, and Liviu Iftode. Paladin: Automated detection and containment of rootkit attacks. Technical report, 2006.

[3] Jörg Brakensiek, Axel Dröge, Martin Botteck, Hermann Härtig, and Adam Lackorzynski. Virtualization as an enabler for security in mobile devices. In *Proceedings of the 1st workshop on Isolation and integration in embedded systems*, IIES '08, pages 17–22, New York, NY, USA, 2008. ACM.

[4] Sven Bugiel, Stephan Heuser, Ahmad reza Sadeghi, and Technische Universität Darmstadt. Towards a framework for android security modules: Extending se android type enforcement to android middleware. Technical report, 2012.

[5] Gary Chen. Kvm: Open virtualization becomes enterprise grade (white paper). Technical report, IDC, 2013.

[6] Mihai Christodorescu, Reiner Sailer, Douglas Lee Schales, Daniele Sgandurra, and Diego Zamboni. Cloud security is not (just) virtualization security: A short paper. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, CCSW '09, pages 97–102, New York, NY, USA, 2009. ACM.

[7] Christoffer Dall and Jason Nieh. Kvm for arm. Proceedings of the 12th Annual Linux Symposium, 2010.

[8] IMDb Internet Movie Database. Matrix (film). `http://www.imdb.com/title/tt0133093/`, 1999.

[9] Shawn Embleton, Sherri Sparks, and Cliff Zou. Smm rootkits: A new breed of os independent malware. In *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks*, SecureComm '08, pages 11:1–11:12, New York, NY, USA, 2008. ACM.

[10] Univ.-Doz. DI Dr. Gerhard Eschelbeck. System security - malware. Vorlesungsfolien, 2011.

[11] Paola Bari et al. *Security on z/VM*. IBM Redbooks, 2007.

[12] Daniel R. Ferstay. Fast secure virtualization for the arm platform. Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP 2011), 2006.

[13] Tal Garfinkel and Mendel Rosenblum. When virtual is harder than real: security challenges in virtual machine based computing environments. In *Proceedings of the 10th conference on Hot Topics in Operating Systems - Volume 10*, HOTOS'05, pages 20–20, Berkeley, CA, USA, 2005. USENIX Association.

[14] B. D. Gold, R. R. Linde, and P. F. Cudney. KVM/370 in retrospect. pages 13–23, 1984.

[15] E. Hermann. *Virtual Machine Based Rootkits: Aufbau - Funktionsweise - Erkennung - Abwehr*. Schriftenreihe Recht und Informationstechnologie / Schriftenreihe Recht und Informationstechnologie. Trauner, 2009.

[16] Greg Hoglund and Jamie Butler. *Rootkits: Subverting the Windows Kernel*. Addison-Wesley Professional, 2005.

[17] Rudolf Hörmanseder and Markus Jäger. Cloud security problems caused by virtualization technology vulnerabilities and their prevention. In *Proceedings of the 22nd Interdisciplinary Information Management Talks*, IDIMT, 2014.

[18] Amani S. Ibrahim, James Hamlyn Hamlyn-harris, and John Grundy. Emerging security challenges of cloud virtual infrastructure, 2010.

[19] Xuxian Jiang, Xinyuan Wang, and Dongyan Xu. Stealthy malware detection and monitoring through vmm-based "out-of-the-box" semantic view reconstruction. *ACM Trans. Inf. Syst. Secur.*, 13(2), 2010.

[20] Samuel T. King, Peter M. Chen, Yi-Min Wang, Chad Verbowski, Helen J. Wang, and Jacob R. Lorch. Subvirt: Implementing malware with virtual machines. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, SP '06, pages 314–327, Washington, DC, USA, 2006. IEEE Computer Society.

[21] Michael Ligh, Steven Adair, Blake Hartstein, and Matthew Richard. *Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code*. Wiley, 2010.

[22] Flavio Lombardi and Roberto Di Pietro. Secure virtualization for cloud computing. *Journal of Network and Computer Applications*, 34(4):1113 – 1122, 2011. Advanced Topics in Cloud Computing.

[23] Shengmei Luo, Zhaoji Lin, Xiaohua Chen, Zhuolin Yang, and Jianyong Chen. Virtualization security for cloud computing service. In *Cloud and Service Computing (CSC), 2011 International Conference on*, pages 174–179, Dec 2011.

[24] Jeanna N. Matthews, Eli M. Dow, Todd Deshane, Wenjin Hu, Jeremy Bongio, Patrick F. Wilbur, and Brendan Johnson. *Running Xen: A Hands-On Guide to the Art of Virtualization*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1 edition, 2008.

[25] Peter Mell and Timothy Grance. The nist definition of cloud computing. Technical Report 800-145, National Institute of Standards and Technology (NIST), Gaithersburg, MD, September 2011.

[26] I. Menken and G. Blokdijk. *Virtualization–the complete cornerstone guide to virtualization best practices*. Emereo Pty Limited, 2008.

[27] Stephan Mueller. Kvm: Security comparison. Technical report, atsec information security corporation, 2009.

[28] DI Christian Praher. Virtualisierung. Vorlesungsfolien, 2013.

[29] Joanna Rutkowska Rafal Wojtczuk. Following the white rabbit: Software attacks against intel vt-d technology.

[30] Edward Ray and Eugene Schultz. Virtualization security. In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, CSIIRW '09, pages 42:1–42:5, New York, NY, USA, 2009. ACM.

[31] Joanna Rutkowska. Red pill... or how to detect vmm using (almost) one cpu instruction. `http://web.archive.org/web/20070911024318/http://invisiblethings.org/papers/redpill.html`, 2004.

[32] Joanna Rutkowska. Blue pill project. `http://web.archive.org/web/20080418123748/http://www.bluepillproject.org/`, 2006.

[33] F. Sabahi. Virtualization-level security in cloud computing. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pages 250–254, May 2011.

[34] Peter Szor. *The Art of Computer Virus Research and Defense*. Addison-Wesley Professional, 2005.

[35] Brett Waldmann and Randy Perry. Steigerung des geschäftserfolgs durch desktop-virtualisierungen (white paper). Technical report, IDC, 2013.

[36] Doug Wampler and James H. Graham. A normality based method for detecting kernel rootkits. *SIGOPS Oper. Syst. Rev.*, 42(3):59–64, April 2008.

[37] Die Freie Enzyklopädie Wikipedia. Virtualisierung (informatik). `http://de.wikipedia.org/wiki/Virtualisierung_%28Informatik%29`, 2013.

[38] The Free Encyclopedia Wikipedia. Virtualization. `http://en.wikipedia.org/wiki/Virtualization`, 2013.

[39] George Wilson, Michael Day, and Beth Taylor. Kvm: Hypervisor security you can depend on. Technical report, IBM, 2011.

[40] Rafal Wojtczuk. Subverting the Xen hypervisor. In *Black Hat*, 2008.

[41] Rafal Wojtczuk and Joanna Rutkowska. Attacking smm memory via intel® cpu cache poisoning. http://invisiblethingslab.com/resources/misc09/smm_cache_fun.pdf, 2009.

[42] Rafal Wojtczuk, Joanna Rutkowska, and Alexander Tereshkin. Xen 0wning trilogy, slides. http://invisiblethingslab.com/resources/bh08/part1.pdf, http://invisiblethingslab.com/resources/bh08/part2.pdf, http://invisiblethingslab.com/resources/bh08/part3.pdf, 2008.

[43] Rafal Wojtczuk, Joanna Rutkowska, and Alexander Tereshkin. Another way to circumvent intel® trusted execution technology. http://invisiblethingslab.com/resources/misc09/Another%20TXT%20Attack.pdf, 2009.

[44] Dino A. Dai Zovi. Hardware virtualization rootkits, slides. http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Zovi.pdf, 2006.

Markus Jäger BSc

Curriculum Vitae

Persönliche Angaben

Geburt	1987, 4710 Grieskirchen
Staatsbürgerschaft	österreichisch
Wohnort	4040 Linz, Dornacherstraße 4722 Peuerbach, Steinbruck
Telefon & e-Mail	+43 (0)664 512 12 22, markus.jaeger.privat@gmail.com
Familienstand	ledig

Bildung

seit 2012/04	Masterstudium „Netzwerke und Sicherheit“ an der Technisch-Naturwissenschaftlichen Fakultät und Masterstudium „Recht und Wirtschaft für TechnikerInnen“ an der Rechtswissenschaftlichen Fakultät der Johannes Kepler Universität Linz
2008/03 – 2012/03	Bachelorstudium der Informatik an der Technisch-Naturwissenschaftlichen Fakultät der Johannes Kepler Universität Linz, Abschluss Bachelor of Science (BSc)
2001/09 – 2006/07	Höhere Technische Lehranstalt der Stadtgemeinde Grieskirchen, Fachrichtung EDV & Organisation, abschließende Reife- und Diplomprüfung, Schwerpunktarbeit in Betrieblicher Organisation: „Personalmanagement“
1997/09 – 2001/07	Gymnasium Dachsberg, Prambachkirchen
1993/09 – 1997/07	Volksschule Peuerbach

Berufliche Tätigkeiten

seit 2013/08	Studentischer Mitarbeiter im Forschungsbetrieb, Institut für Mikroprozessortechnik und Informationsverarbeitung, JKU Linz
2012/10 – 2013/01	Nebenberuflicher Lektor an der FH Oberösterreich, Campus Wels
2011/09	Praktikumsstelle als freier Dienstnehmer beim Verein Kulturforum Melodium (Peuerbach), Organisation
2011/07 – 2011/08	Ferialanstellung bei Fa. dynaTrace Software GmbH (Linz-Urfahr), Abteilung IT
2008, 2009, 2010	Ferialanstellung bei Fa. Schaumann (Taufkirchen an der Trattnach), Abteilung Logistik
2006/07 – 2007/11	16 Monate Präsenz- und „Militärperson auf Zeit“–Dienst bei der Militärmusik Niederösterreich, Bundesministerium für Landes-verteidigung und Sport (BMLVS), Hesserkaserne St. Pölten
2005/07 – 2005/08	Ferialanstellung bei Fa. Lindner & Hager OEG (Peuerbach), Gastronomie
2004/07	Pflichtpraktikum bei Fa. Leitz (Riedau), Abteilung Elektronik
2003/07	Pflichtpraktikum bei Fa. Hexcel (Neumarkt-Kallham), Abteilung EDV (Netzwerk- und Systemadministration)
2002/07	Ferialanstellung bei Fa. Högl (Taufkirchen an der Pram), Abteilung Elektronik/Schlosserei

Ehrenamtliche Funktionärstätigkeiten

Musikverein Peuerbach

seit 2008/05	Instrumentenarchivar, EDV-Betreuung, Kapellmeister Stellvertreter
2005 - 2008	Jugendreferent

Österreichische Hochschülerschaft, JKU Linz

seit 2013/04	Vorsitzender der Fakultätsvertretung TNF
seit 2011/05	Mandatar in der Kommission für Leistungs- und Förderstipendien der TNF, Beirat für Lehre TNF, Institutskonferenz für das Institut „Informationsverarbeitung und Mikroprozessortechnik (FIM)“ und
seit 2012	Diverse Berufungs- und Habilitationskommissionen
seit 2011/03	Mandatar in der Studienkommission Informatik
seit 2010/10	Mandatar der Fakultätsversammlung der TNF an der JKU Linz
2012/03 – 2014/03	Vorsitzender der Studienvertretung Informatik
2011/05 – 2014/02	Mandatar in der Studienkommission ISI-Hagenberg, Mitglied Institutskonferenz für das Institut „Formale Modelle und Verifikation (FMV)“
2011/07 – 2012/03	Stv. Vorsitzender der Studienvertretung Informatik
2010 – 2012	Erstsemestrigentutor, Mentoringprogramm

Verein der Absolventen und Förderer der HTL Grieskirchen

seit 2007/03	Beauftragter für Kommunikation
2007/03 – 2007/12	Kassier Stellvertreter

Musikalische Ausbildungen

2004 – 2006 & 2008 – 2010	Kapellmeister- und Orchesterleiterausbildung des oberösterreichischen Blasmusikverbandes (OÖBV) und des oberösterreichischen Landesmusikschulwerkes (OÖLMSW)
2005 – 2006	Ausbildung zum diplomierten Vereinsjugendreferenten des OÖBV
1999, 2002	Jungmusikerleistungsabzeichen (JMLA) 1999 in Bronze und 2002
2004, 2006	in Silber auf dem Tenorhorn, JMLA 2004 in Silber und 2006 in Gold auf der Bass-Posaune, jeweils mit „Sehr Gutem Erfolg“

Weitere Kenntnisse

- Erweiterte Fähigkeiten im Soft- und Hardwarebereich der EDV sowie ausgezeichnete Kenntnisse im Umgang mit der Microsoft Produktpalette
- Grundkenntnisse sowie mehrjährige Erfahrung mit Design-Tools wie Adobe Photoshop und Adobe InDesign
- Semiprofessionelle Fotografie
- Erfahrung in Führungspositionen durch mehrjährige Schul- und Klassensprecher-Tätigkeiten, leitende Vorstandsfunktionen, Vorsitz von Studienvertretung und Fakultätsvertretung sowie die Dirigenten- und Orchesterleiterarbeit
- Englisch fließend in Wort und Schrift (CEFR: C1)
- Spanisch Grundkenntnisse (CEFR: A2)
- CCNA1 (Cisco Certified Network Associate)
- Führerschein Klasse B

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe. Die vorliegende Masterarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Linz, Juli 2014