Sicherer Betrieb existierender Applikationen im Unternehmensumfeld mit Open Source-Werkzeugen

Masterthesis zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

im Studiengang
Informatik – Komplexe Softwaresysteme

an der Hochschule Bremen
Fakultät Elektrotechnik und Informatik
University of Applied Sciences

vorgelegt von
Andreas Peichert

Erster Gutachter: Prof. Dr. Richard Sethmann
Zweiter Gutachter: Dr. Arvid Requate

28. September 2015
Bremen, Deutschland

Zusammenfassung

Der virtuelle Betrieb von Servern liefert die Basis des Cloud Computing zur Bereitstellung von Rechenkapazitäten als Dienstleistung. Bei dieser Server-Virtualisierung verwaltet ein Hypervisor die Ressourcen der Gast-Betriebssysteme. Im Vergleich dazu verzichtet die Betriebssystem-Virtualisierung auf diese Zwischenschichten und stellt laufende Prozesse voneinander abgetrennt auf dem gleichen System bereit. Mit Hilfe bestehender Funktionalität des Linux-Kernels werden Prozesse isoliert und in ihren Zugriffsberechtigungen beschränkt. Die kürzere Verarbeitungskette liefert Geschwindigkeitsvorteile, birgt aber auch Risiken, da der Kernel während des Betriebs nicht zwischen verschiedenen root-Benutzern von Gast und Wirt unterscheiden kann. Bei der Ausführung fremder Applikationen ist die Gewährleistung der Sicherheit wichtig, da der gleiche Kernel gemeinsam genutzt wird. Die verwendete Methode der Bedrohungsanalyse ist an die vom Bundesamt für Sicherheit in der Informationstechnik (BSI) veröffentlichten IT-Grundschutz Vorgehensweise angelehnt. Zwei Schadensszenarien, durch übermäßige Ressourcenbeanspruchung und dem Ausbruch aus einer isolierten Umgebung stehen im Mittelpunkt der Untersuchung. Anhand dieser sind in einer Risikoanalyse die möglichen Auswirkungen auf die Sicherheit hin bewertet und Gegenmaßnahmen zur Absicherung ausgewählt worden. Für die prototypische Realisierung wurde Security-Enhanced Linux (SELinux) als Sicherheitserweiterung des Linux-Kernels eingesetzt. Dazu ist das Regelwerk der Docker-Policy von Fedora nach Debian portiert und angepasst worden. Mit Hilfe des so bereitgestellten Mandatory Access Control kann der Linux-Kernel unerwünschte privilegierte Aufrufe des root-Benutzers erkennen und zwischen Gast- und Wirt-System unterscheiden. Zur Protokollierung und Erkennung von Sicherheitsverletzungen wurde ein Security Information und Event Management (SIEM)-System eingesetzt, zur Erzeugung von Alarm-Benachrichtigungen aus Protokoll-Dateien. Als Ergebnis ist es somit einem Unternehmen möglich existierende Applikationen in virtualisierten Containern abgetrennt und sicher mit Hilfe von Open Source-Werkzeugen betreiben.

Summary

The operation of servers in a virtual environment provides the basis of cloud computing to deliver computing capacities as a service. During server virtualization a hypervisor manages the resources of the guest operating systems. By comparison, the operating-system-level virtualization provides processes separated from each other on the same system. By using existing functionality of the Linux kernel, processes are isolated and limited in their access permissions. The shorter processing chain provides performance advantages, but also involves risks because the kernel can not distinguish during operation between various root-users by the guests and host. When executing foreign applications ensuring safety is important, because the same kernel is shared. The employed method of threat analysis is based on the “IT Baseline Protection Catalogs” published by the *Federal Office for Information Security* (German: *Bundesamt für Sicherheit in der Informationstechnik*, abbreviated as BSI). Two damage scenarios are the focus of the investigation, caused by excessive resource usage and the breakout from an isolated environment. Based on this, in a risk assessment the potential impact on safety has been evaluated and countermeasures have been selected for protection. For the prototype implementation Security-Enhanced Linux (SELinux) was used as a security extension of the Linux kernel. For this purpose the rules of the Docker policy have been ported and adapted from Fedora to Debian. By using the provided Mandatory Access Control, the Linux kernel can detect unwanted privileged calls of the root user and can distinguish between guest and host system. For logging and intrusion detection a security information and event management (SIEM) system was used to generate alarm notifications from violation reports. As a result, it is possible for a company to separate existing applications in virtualized containers and let them safely operate by using open source tools.

Inhaltsverzeichnis

Zusammenfassung....II

Summary....II

Inhaltsverzeichnis....III

Abkürzungen....V

1 Einführung....1
1.1 Motivation....1
1.2 Unternehmen und Produkt....1
1.3 Problemstellung....1
1.4 Aufgabenstellung....2
1.5 Erwartete Endergebnisse....2
1.6 Aufbau der Arbeit....2

2 Relevante Grundlagen....3
2.1 Betriebssystem und Kernel-Modus....3
2.2 Varianten der Virtualisierung....4
2.3 Marktüberblick von Virtualisierungs-Implementierungen....7
2.4 Sicherheitsaspekte im Linux-Kernel....10
2.5 Grundlagen der Informationssicherheit....16
2.5.1 Bedrohungen der Werte durch existierende Applikationen....16
2.5.2 Eintrittswahrscheinlichkeit und sicherer Betrieb....17
2.5.3 Auswirkung auf Schutzziele und Geschäftsumfeld....17

3 Konzeption und Architektur....18
3.1 Virtualisierungsarchitektur....18
3.1.1 KVM-Hypervisor Architektur....18
3.1.2 Docker-Container Architektur....19
3.1.3 Vergleich zu Laufzeit....21
3.2 Systemkonzept....21
3.2.1 Ressourcenteilung....22
3.2.2 Kommunikationspfade....23
3.2.3 Berechtigungen....23
3.2.4 Eingesetzte Technologien....23

4 Bedrohungsanalyse....24
4.1 Methoden zur Einordnung von Bedrohungen und Risiken....24
4.1.1 Bewertung sicherer Applikationen....24
4.1.2 Vorgehen und Nomen zur Analyse....25
4.2 Erhöhter Schutzbedarf....28
4.3 Interessengruppen und Zielobjekte....29
4.4 Anwendung der Bedrohungsanalyse....31
4.5 Analyse der eingesetzten Technologien....34

5 Risikoabschätzung und Maßnahmen....37
5.1 Auswertung der Bedrohungsanalyse....37
5.2 Auswahl der Gegenmaßnahmen....40
5.2.1 Eingesetzte Technologien....40
5.2.2 Einschränkung der Zugriffsrechte....41
5.2.2.1 Mandatory Access Control....42
5.2.2.2 Security Enhanced Linux....43
5.2.3 Protokollierung und Auswertung....46
5.2.3.1 Security Information und Event Management....46
5.2.3.2 Open Source Security Information Management....48
5.2.4 Behandlung der Risiken....50
5.3 Anpassung des Systemkonzepts....52

6 Implementierung und Demonstration....54
6.1 Gewährleistung der Sicherheit....55
6.2 Prototypischer Aufbau....57
6.3 Darstellung der Schadensszenarien im Monitoring-Frontend....60

7 Diskussion der Ergebnisse....61
7.1 Wissenschaftliche Sichtweise....61
7.2 Praxisorientierte Sichtweise....62
7.3 Handlungsempfehlung....64

8 Fazit....65
8.1 Rückblick....65
8.2 Schlussfolgerung....65
8.3 Ausblick....65

9 Verzeichnisse....66
9.1 Abbildungen....66
9.2 Tabellen....66
9.3 Quelltexte....67
9.4 Literatur....67

Eidesstattliche Erklärung....70

Anhänge....71
Docker-Policy und Debian-Anpassungen....71
OSSIM Plugin-Erweiterung....72

Abkürzungen

AGPL	GNU Affero General Public License
BSI	Bundesamt für Sicherheit in der Informationstechnik
CPU	Central Processing Unit – Prozessorkern
DAC	Discretionary Access Control – Benutzerbestimmbare Zugangskontrolle
DoS	Denial of Service – Überlastung durch Ressourcenauslastung
DTE	Domain Type Enforcement
EPS	Events pro Sekunde
GNU	rekursives Akronym von „GNU's Not Unix!“
GPL	GNU General Public License
GSK	IT-Grundschutz-Katalog des BSI
IPC	Interprozesskommunikation
KVM	Kernel-based Virtual Machine
MAC	Mandatory Access Control – verpflichtende Zugangskontrolle
QEMU	Quirk-Emulator
RDP	Remote Desktop Protocol
RFB	Remote Framebuffer Protocol
SELinux	Security Enhanced Linux
SEM	Security Event Management – Echtzeitüberwachung und Korrelation von Events
SIEM	Security Information und Event Management
SIM	Security Information Management – Langzeiterfassung von Protokoll-Dateien
UCS	Univention Corporate Server
VM	virtuelle Maschine
VNC	Virtual Network Computing

1 Einführung

1.1 Motivation

An der Hochschule Fontys in Venlo begann mein Studium der Informatik im Fachbereich Software-Engineering und wurde mit dem Bachelor of ICT abgeschlossen. Das Thema der Abschlussarbeit war die „Integration von Elastic Compute Cloud in UCS Virtual Machine Manager“, also eine Möglichkeit zur gemeinsamen Verwaltung einer Hybrid-Cloud Computing IT-Infrastruktur. Das Ergebnis war die Anbindung von EC2 kompatiblen Schnittstellen wie die von Amazon AWS oder OpenStack in das Virtualisierungsprodukt UVMM von Univention.

Die vorliegende Masterthesis schließt das Studium der Informatik an der Hochschule Bremen im Bereich Komplexe Softwaresysteme mit dem Master of Science ab. Mein Interesse an Linux und Virtualisierung, erweitert um den Aspekt der Sicherheit, sind die relevanten Elemente zur Erstellung der Abschlussarbeit im praxisnahen und betrieblichen Umfeld.

Die Herausforderung ist dabei die Untersuchung und Gegenüberstellung von bewährter, gut dokumentierter und erprobter Virtualisierung-Technik im Vergleich zu der neuen, sich noch in der Entwicklung befindlichen, Container-Virtualisierung unter dem Gesichtspunkt der Sicherstellung der Sicherheit der beteiligten Systeme.

1.2 Unternehmen und Produkt

Die Univention GmbH wurde 2002 in Bremen gegründet und hat sich auf die Bereitstellung von Open Source-Lösungen spezialisiert. Ziel ist, alternative Linux-Komplettlösungen anzubieten, welche bei Unternehmen sowie Bundes- und Landesbehörden erfolgreich eingesetzt werden können. Wichtig ist Univention dabei nach eigenem Bekunden die Offenheit, Unabhängigkeit und Wirtschaftlichkeit der angebotenen Produktpalette [Univention GmbH 2015].

Univention entwickelt und vertreibt seit 2004 das Software-Produkt Univention Corporate Server (UCS), ein auf Debian GNU/Linux basierendes Serverbetriebssystem unter AGPL-Lizenz. Dies stellt ein zentrales Managementsystem zur plattformübergreifenden Verwaltung von Servern, installierten Diensten, Desktop-Rechnern sowie Benutzern zur Verfügung.

Im integrierten UCS „App Center“ können Partner ihr eigenes Software-Produkt einbinden lassen, so dass den Benutzern verschiedene Anwendungen von Drittanbietern wie Groupware, Datensicherung sowie ERP und CRM Systeme bereitgestellt werden. Der Funktionsumfang von UCS wird so einfach erweitert und auf die Bedürfnisse des Kunden angepasst.

1.3 Problemstellung

Über das UCS „App Center“ sind derzeit nur Anwendungen verfügbar, welche entweder von Univention selbst, oder von bekannten Unternehmen oder Partnern betreut werden. Beim Installationsprozess und bei der späteren Ausführung sind Systemweite Administrator-Rechte notwendig, weswegen eine Fehlerfreiheit der Anwendungen wichtig ist. Eine Öffnung der Plattform, um auch weniger bekannte Software anzubieten, würde die Anzahl der bereitgestellten Anwendungen deutlich erhöhen. Allerdings ist es nicht praktikabel jede neue Applikation und jede Versionsaktualisierung händisch zu überprüfen, da dies sehr zeitintensiv wäre.

Im Unternehmensumfeld ist die Verlässlichkeit der verwendeten Anwendungen wichtig. Zusätzlich geht der Trend weg vom Betrieb eigener Hardware, hin zum Cloud Computing mit der Möglichkeit in Abhängigkeit der Auslastung zu skalieren. Seit mehr als 10 Jahren sind per Vollvirtualisierung komplette Systeme in virtuelle Maschinen auslagerbar. In jüngster Zeit besteht auch die Möglichkeit lediglich die benötige Applikation per Teilvirtualisierung in einer Sandbox zu betreiben. Einzelne Anwendungen sollen voneinander getrennt werden, so dass Zugriffe und Berechtigungen nur nach einem zuvor erlaubtem Muster möglich sind. Sensible Firmeninformationen dürfen weder gelöscht noch nach extern kopiert werden.

1.4 Aufgabenstellung

Die Masterthesis ist in Kooperation mit der Entwicklungs-Abteilung von Univention entstanden. In einem parallelen Forschungsprojekt „SaferApps“ von Univention und Mitarbeitern des Forschungsbereichs Cyber-Physical Systems des Deutschen Forschungszentrums für Künstliche Intelligenz (DFKI) sollen bis März 2017 Mechanismen entwickelt werden, um erforderliche Berechtigungen von Applikationen verständlich zu präsentieren. IT-Verantwortliche sollen so entscheiden können, ob der Betrieb einer Fremdapplikation mit unternehmensweiten Sicherheitsrichtlinien vereinbar ist.

Der Schwerpunkt der vorliegenden Masterthesis ist die sichere Ausführung von bestehenden Applikationen im Unternehmensumfeld. Die sich daraus ergebende Forschungsfrage ist: Welche existierenden Möglichkeiten sind geeignet zur Virtualisierung von Open Source-Applikationen und wie kann dabei die Sicherheit im Unternehmensumfeld gewährleistet werden? Der aktuelle Stand der Virtualisierung, verbunden mit der Sicherheit unter Linux, sind Ausgangspunkt der Masterthesis.

1.5 Erwartete Endergebnisse

Das Ergebnis soll mindestens die gewünschten Funktionalitäten beinhalten:

- Analyse und Vergleich aktueller Virtualisierungstechnologien und Erstellung einer Entscheidungsmatrix mit relevanten Kriterien.
- Abschätzung von Bedrohungen und Abgabe einer Empfehlung als Basis für Sicherheitsentscheidungen im Unternehmensumfeld.
- Erstellung eines Prototyps für die Verarbeitungskette zur Absicherung von Applikationen mit den Mitteln Virtualisierung, Kapselung der Unternehmensapplikation, Sicherheit und Überwachung.

1.6 Aufbau der Arbeit

Die Masterthesis ist in Kapiteln geordnet, wobei generell zunächst die Theorie und die verwendeten Methoden beschrieben sind, um aus diesen auszuwählen, für die wissenschaftliche und praktische Erarbeitung des Themas.

Mit der Beschreibung der *relevanten Grundlagen* findet, mit der Virtualisierung und dem Linux-Kernel, der Einstieg in das Thema statt. Anschließend werden die Grundlagen der Informationssicherheit definiert.

Im darauf folgenden Abschnitt *Konzeption und Architektur* wird die eingesetzte Virtualisierungsarchitektur verglichen und das dazugehörige Systemkonzept vorgestellt.

Für die *Bedrohungsanalyse* werden zunächst verschiedene Methoden und Normen vorgestellt, aus welchen eine Variante gewählt wird. Anhand dieser werden mögliche Angriffsszenarien und Schwachstellen der Virtualisierungsarchitektur und der einsetzten Technologien analysiert.

Es folgt eine *Risikoabschätzung* zur Bewertung, um so bestehenden Risiken mit geeigneten Gegenmaßnahmen begegnen zu können, was zu einer Aktualisierung des Systemkonzeptes führt.

Im praktischen Teil werden Programmierarbeiten zur *Implementation und Demonstration* für den Aufbau eines Prototyps zur Gewährleistung der Sicherheit im Unternehmensumfeld beschrieben.

Das erreichte Ziel der Masterthesis wird in der *Diskussion der Ergebnisse* aus wissenschaftlicher und praxisorientierter Sicht besprochen, um daraus Handlungsempfehlungen abzuleiten.

Im abschließenden Kapitel wird das *Fazit* mit einem Rückblick, einer kritischen Schlussfolgerung, sowie dem Ausblick zu einer möglichen Fortführung des Themenbereichs vorgestellt.

2 Relevante Grundlagen

Das folgende Kapitel dient als Grundlage für die weitere Betrachtung zum sicheren Betrieb existierender Applikationen. Zunächst werden Basisfunktionalitäten im Bereich Sicherheit und Virtualisierung vorgestellt. Eigenschaften des Betriebssystems zur sicheren Trennung der Arbeitsbereiche mit verschiedenen Betriebs-Modi werden vorgestellt und erläutert, welche unterschiedlichen Arten der Virtualisierung derzeit existieren. Darauf aufbauend soll gezeigt werden, wie diese Basisfunktionalitäten umgesetzt sind. Im Linux-Kernel vorhandene Sicherheitsaspekte stellen die geschützte Ausführung von Applikationen sicher, während verschiedene Virtualisierungs-Implementierungen die Basis zur Trennung von Applikationen und Abhängigkeiten bilden. Im abschließenden Abschnitt der erfolgt ein Einblick in die Informationssicherheit und die Bedeutung potentieller Risiken, welche der Betrieb von Applikationen im Geschäftsalltag mit sich bringt.

2.1 Betriebssystem und Kernel-Modus

Betriebssysteme wie Windows oder Linux richten sich an Endanwender, so dass diese einen Computer verwenden können, ohne die zugrunde liegende Technik im Detail kennen zu müssen. Zur Abstraktion der Komplexität stellen Betriebssysteme intern die Verbindung bereit, um mit Komponenten wie Festplatten und Netzwerkkarten unterschiedlichster Hersteller zusammenzuarbeiten. Entwickler von Programmen nutzen die vom Betriebssystem bereitgestellte einheitliche Basis, um gewünschte Funktionalitäten zu implementieren. Das Betriebssystem spricht erst nachgelagert über den Betriebssystemkern, auch Kernel genannt, die dazugehörige Hardware an. Der Kernel, hat die Aufgabe die Zugriffe auf die Hardware zu steuern und zu synchronisieren. Als Schnittstelle ist das Betriebssystem zwischen der Hardware und den Benutzer-Programmen für die Kommunikation zuständig, wobei ausgeführte und laufende Programme als Prozesse bezeichnet werden.

Zur Trennung der Befehlsaufrufe bei Zugriff auf Dateninhalte und bei solchen Aufrufen, welche direkten Zugriff auf die Hardware erfordern, ist in der Betriebssystemarchitektur ein Mechanismus von Schutzringen (Abbildung 1) integriert.

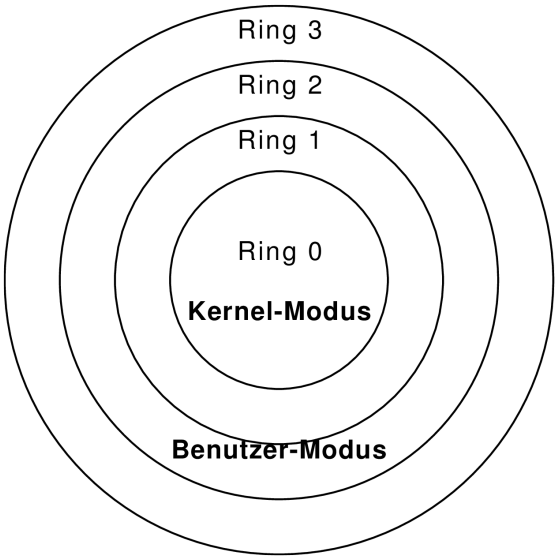

Abbildung 1: Schematische Darstellung der Schutzringe zur Trennung der Betriebsmodi

Der Prozessorkern (CPU) hat privilegierte Befehlssätze zum direkten Zugriff auf die Hardware und die Speicheradressen von Arbeitsspeicher und weiteren Geräten. Diese Unterscheidung ist wichtig, damit ein Prozess nicht auf die Speicheradressen eines anderen Prozesses zugreifen darf.

Diese Art der Vollzugriffe sind der CPU und den ausgeführten Prozessen nur im Kernel-Modus im Ring 0 erlaubt. Als Schnittstelle nach außen in Richtung Betriebssystem haben Prozesse im unprivilegierten Benutzer-Modus nur Zugriffsrechte auf ihre eigenen Speicheradressen. Die Ringe 1 und 2 sind für Gerätetreiber der verschiedenen Hardwarehersteller vorgesehen, in der Praxis werden diese Ringe aber aktuell nicht mehr verwendet. In Ring 3 sind in diesem Modell die vom Benutzer installierten Programme angesiedelt, also die Prozesse des Anwenders zur Laufzeit des Betriebssystems [Michel Rode u. a. 2012; S. 12-13].

Der Kernel verwaltet die Übergänge zwischen den Ringen, wobei der Übergang von einem Ring zum Nächsten von der CPU Rechenzeit erfordert. Durch einen solchen Kontextwechsel ist sichergestellt, dass nur berechtigte Prozesse den Zugriff auf privilegierte Speicherinhalte erhalten. Zwei verschiedene Arten sind dabei zu unterschieden. Die erste Form sieht regelmäßige Unterbrechungen vor, zur Versorgung anderer Prozesse mit Rechenleistung, da die CPU nur immer einen Prozess zeitgleich abarbeiten kann. Diese Funktionsweise ist als Multitasking bekannt, wobei der Kernel die Aufgabe hat die Speicherinhalte von Stack und den CPU-Registern zu sichern, um diese später für den vorherigen Prozess wiederherzustellen. Die zweite Form des Kontextwechsels ist der Zugriff auf systemnahe Hardware. Ein Prozess löst einen Interrupt aus, um von einer bestimmten Hardware Daten zu erhalten. Der Kernel kann so exklusiv auf die Hardware-Komponente zugreifen und anschließend den Prozess mit den gewünschten Informationen benachrichtigen.

Auf x86-Systemen ist ein Kontextwechsel seit dem 80286-Prozessor möglich, der 1982 von Intel veröffentlicht wurde. Der Prozessor bietet einen Real-Modus mit uneingeschränktem Zugriff auf den gesamten Arbeitsspeicher, und den geschützten Modus mit den beschriebenen vier Ringen an. Mit dem 80386-Prozessor kam ab 1985 der Betriebsmodus Virtual 8086 Mode hinzu, welcher sich aus Sicht der Prozesse wie der Real-Modus verhält, aber virtuell zwischen den Speicherbereichen im geschützten Modus umschaltet. Dies waren die ersten Schritte zur virtuellen Aufteilung der physikalisch verfügbaren Ressourcen, was heute als Virtualisierung bekannt ist.

2.2 Varianten der Virtualisierung

Virtualisierung ist mit dem Konzept der Emulation vergleichbar. Bei der Emulation gibt ein System vor, ein anderes zu sein. Mit Virtualisierung kann ein System vorgeben, dass zwei oder mehr existieren [Chisnall 2008; S. 3].

Hardware Emulation findet beispielsweise Verwendung, um ein Programm welches für ARM-Hardware erzeugt wurde, auf x86-Hardware auszuführen. Dazu fängt eine Software die Instruktionen ab und sendet diese übersetzt an die Hardware weiter (binary translation). Dieser Vorgang verursacht einen großen Overhead und eine geringere Performance [Michel Rode u. a. 2012; S. 15].

Im Folgenden wird der Einsatz der Virtualisierung im x86-Bereich genauer betrachtet. Die Aufgabe der Virtualisierung ist es, die zur Verfügung stehende Rechenleistung effektiv zu nutzen. Während die ersten Computer die Aufgaben der Reihe nach abarbeiteten, gibt es heute mit Multitasking die Möglichkeit der Unterbrechung zur parallelen Abarbeitung. Für den Computer ergeben sich so Lastspitzen und Ruhephasen, welche vom bestimmten Einsatzzweck abhängen. Durch Virtualisierung kann die vorhandene Hardware-Rechenleistung aufgeteilt und so besser genutzt und verteilt werden. Rechenintensive Aufgaben lassen sich in last-ärmeren Zeiten mit anderen Maschinen synchronisieren und statt auf mehreren physikalischen Computern, virtualisiert auf derselben physikalischen Hardware betreiben. Dies spart Anschaffungs- und Betriebskosten ein.

In einer Studie wurden über 1.500 ICT-Entscheider von Unternehmen mit mehr als 1.000 Mitarbeitern befragt [Kretschmer 2010]. Demnach werden bis 2015 im Bereich des Cloud Computing Kostensenkungen von 17 % erwartet. Allerdings sind der mögliche Einsatz von Virtualisierung und die damit verbunden Einsparpotentiale stark abhängig vom IT-Aufbau des einzelnen Unternehmens. Für eine realistische Berechnung muss jedes Unternehmen für sich selber die wirtschaftlichen Aspekte ermitteln und errechnen, was konkret an laufenden Betriebskosten statt Investitionen in eigene IT-Hardware anfällt. Hierbei hilft die Betrachtung von fixen und variable Kosten, wobei auch die Schulung der Mitarbeiter, veränderte Lizenzkosten und Kosten für die Integration in die eigene IT-Landschaft zu betrachten sind [Thorsten Reitz/Juan Villar 2011; S. 62].

Die Virtualisierung lässt sich in die Bereiche Paravirtualisierung, Hardwarevirtualisierung und Betriebssystem-Virtualisierung aufteilen. Alle haben gemeinsam, dass zum Betrieb ein Betriebssystem, zur Steuerung der Kernel und je nach Virtualisierungsart zusätzliche Software notwendig ist.

Im Betrieb wechselt der Prozessor selbstständig fortwährend den Modus, um privilegiert auf alle Adressbereiche des Speichers zugreifen zu können, und um laufend Prozesse zu pausieren und zu wechseln. Bei der Virtualisierung der CPU-Befehle ist die physikalische CPU allerdings von außen zu unterbrechen, damit die Aufgabe einer virtuellen CPU bearbeitet werden kann. Darüber hinaus gibt es weitere Ein- und Ausgabegeräte (I/O) wie Arbeitsspeicher oder Festplatten, welche durch Aufteilung und Partitionierung einer virtuellen Maschine zugeordnet sind, um mit dem Inhalt exklusiv zu arbeiten. Andere Geräte, wie Netzwerk- oder Grafikkarten erfordern für den Betrieb eine exakte Software-Nachbildung [Chisnall 2008; S. 3-5].

Zusätzliche Software übernimmt in Form des Hypervisors die Steuerung und Unterbrechung der CPU und der I/O Geräte von außen. Es handelt sich dabei um eine Abstraktionsebene zwischen dem Betriebssystem-Kernel und der zugrunde liegenden Hardware. Diese Schicht hebt per Software die direkte Bindung von Betriebssystem und Kernel zu der eingesetzten Hardware auf. Der Hypervisor stellt die für das Betriebssystem benötigte Hardware vollständig virtuell bereit. Dadurch ist es möglich, von einem Wirt-System aus mit Hilfe eines installierten Hypervisors weitere virtuelle Maschinen (VM) zu verwalten. Zum Betrieb existieren verschiedene Varianten an Hypervisoren, sog. unterschiedliche Virtualisierungsarten, die in Abbildung 2 gegenübergestellt sind. Der Typ 1 Hypervisor setzt direkt auf purer Hardware auf, ohne dass ein Betriebssystem installiert ist (native oder Bare-metal) und Typ 2 verwendet ein installiertes Wirt-Betriebssystem als Basis (hosted).

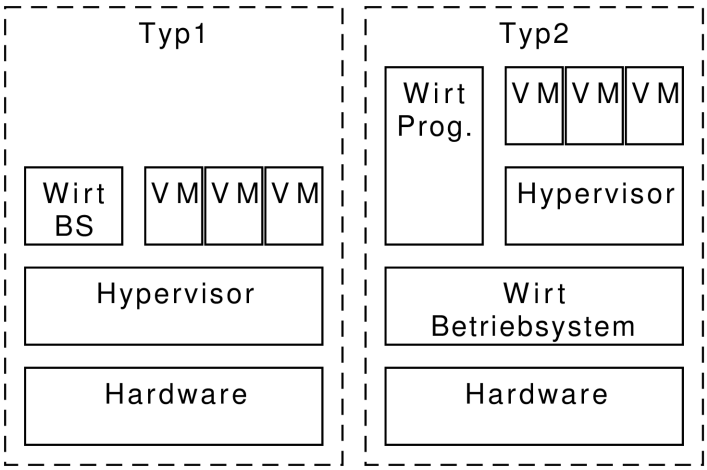

Abbildung 2: Architektur Hypervisor Typ 1 und Typ 2

Beide Hypervisor-Varianten unterscheiden sich somit in der Abstraktionsebene und finden je nach Implementierungsart in der Para- und der Hardwarevirtualisierung unterschiedliche Verwendung.

Der Hypervisor verarbeitet bei der *Paravirtualisierung* die Befehle auf dem Ring 0 und das Gast-Betriebssystem arbeitet nur noch auf Ring 1. Als Konsequenz der zusätzlichen Zwischenschicht verschiebt sich die Zuordnung zu den Schutzringen der CPU. Dies führt dazu, dass die hardwarenahen Befehle des Gast-Systems erkannt und abzufangen sind. Dieser Mechanismus setzt ein angepasstes Betriebssystem zum Betrieb als virtuelle Maschine voraus. Eine Implementierung vom Typ 1 ist der Xen Hypervisor, welcher 2003 veröffentlicht wurde. Befehle vom Gast-System führt der Hypervisor direkt aus, ohne die Hardware-Anfragen abfangen und übersetzen zu müssen. Somit ist eine höhere Performance möglich, wenngleich sich Gast-Systeme der virtuellen, statt realer Hardware bewusst sind. Es findet keine Hardware Emulation statt, sondern das Gast-System erhält nur bestimmte Komponenten, zu denen auch die entsprechenden paravirtualisierten Treiber bereitstellbar sind. Allerdings ist es nicht möglich, für jede verfügbare Hardware-Komponente die vollständige Funktionalität per Treiber anzubieten. Ein weiteres Problem ist, dass Hersteller proprietärer Betriebssysteme nicht bereit sind, diese für den virtuellen Betrieb anzupassen.

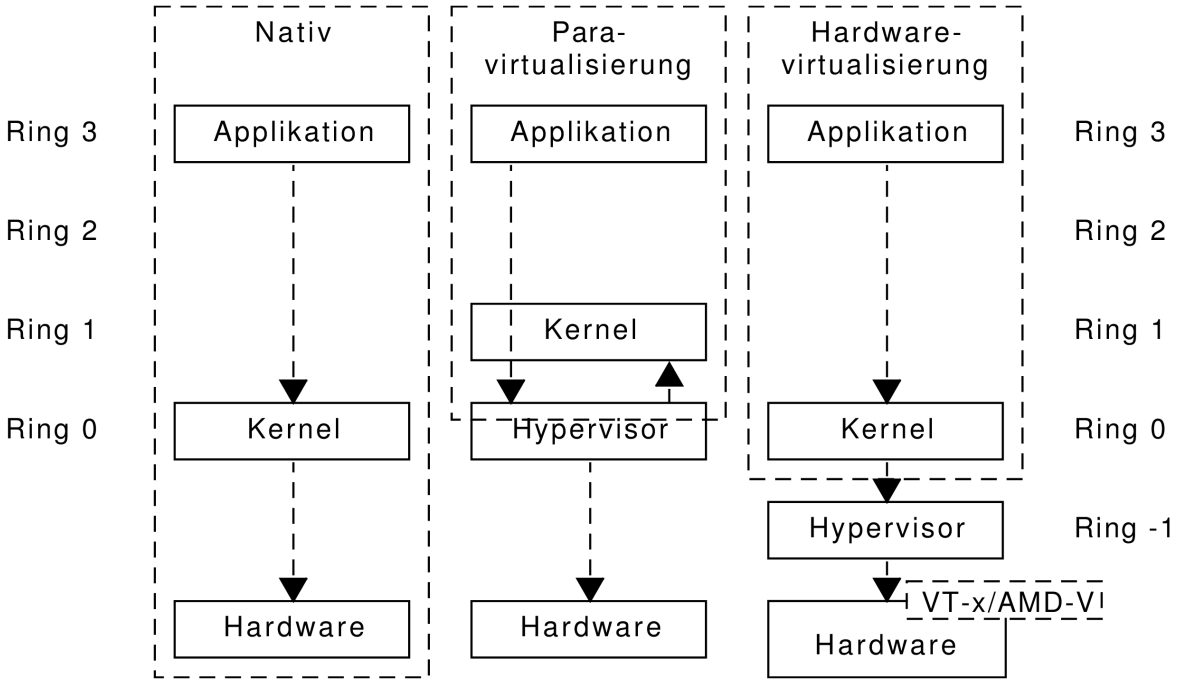

Abbildung 3: Native Betriebssystem-Kernel-Aufrufe, Para- und Hardwarevirtualisierung

Die *Hardwarevirtualisierung* erfordert keine komplette Emulation der Hardware. Notwendig ist dazu die direkte Unterstützung der CPU, um die Umwandlung der Aufrufe durch Software zu verringern. Prozessor-Aufrufe verarbeitet die physikalische CPU, während I/O Geräte wie CD-Rom-Laufwerke weiterhin in Software zu emulieren sind. In Abbildung 3 sind die unterschiedlichen Wege der Prozessor-Aufrufe ohne Virtualisierung der Para- und Hardwarevirtualisierung gegenübergestellt. Zur hardwareunterstützten Virtualisierung wurden x86-Prozessoren entwickelt, welche mit *Intel Virtualization Technology for x86* (VT-x) 2005 und mit *AMD Virtualization* (AMD-V) 2006 bereitgestellt wurden, aber zueinander nicht kompatibel sind. Um das bisherige Sicherheitskonzept nicht zu verändern, arbeitet der Kernel weiterhin auf Ring 0. Eine Erweiterung findet für den Hypervisor-Modus auf Ring -1 statt, um weitere Befehlssätze zur Virtualisierung anzubieten. Somit ist für ein Betriebssystem und dessen Kernel keine Kenntnis über einen virtuellen Betriebsmodus notwendig. Die Prozessorerweiterungen zur Virtualisierung können Hypervisor sowohl von Typ 1 als auch Typ 2 nutzen. Die Kernel-based Virtual Machine (KVM) ist ein Hypervisor vom Typ 2, der 2006 zur Hardwarevirtualisierung veröffentlicht wurde und inzwischen ebenfalls Paravirtualisierung anbietet.

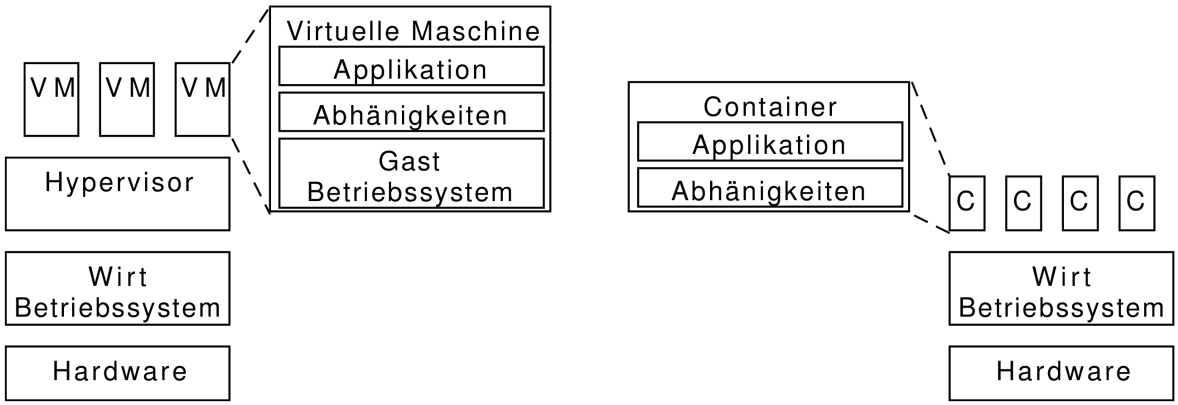

Abbildung 4: Vergleich Hypervisor Typ 2 und Betriebssystem-Virtualisierung

Bei der *Betriebssystem-Virtualisierung* findet gar keine Hardware-Emulation statt. Gast-Systeme teilen sich mit dem Wirt den gleichen Kernel, so dass kein Hypervisor notwendig ist zur Umlenkung der Befehle vom Gast zum Wirt-System. Als virtuelles System lassen sich nur die gleichen Betriebssystem-Arten verwenden, beispielsweise kann ein Linux-Wirt ein weiteres Linux bereitstellen, wobei aber die verwendete Distributionen auswählbar ist. Die virtuellen Gast-Systeme heißen hierbei Container, um den Unterschied zu einer vollwertigen virtuellen Maschine deutlich zu machen. Abbildung 4 zeigt die Komponenten von VM und Container im Vergleich. Abhängigkeiten sind dabei benötigte Softwarekomponenten und Bibliotheken, welche zum Betrieb der Applikationen notwendig sind. Der geringe Overhead macht die Betriebssystem-Virtualisierung hochperformant, so dass Container und darin enthaltene Applikationen innerhalb von Sekunden gestartet sind. Weitere generelle Unterschiede sind in Tabelle 1 aufgelistet.

Tabelle 1: Hypervisor-Virtualisierung im Vergleich zum Container-System

	Hypervisor	**Container**
Virtualisierungs-Level	Hardware-Ebene	Betriebssystem
Abstraktions-Level	Betriebssystem von Hardware	Applikation von Betriebssystem
Festspeicherplatzbelegung	Für jede VM-Instanz separat	Ausgangsdatei und Speicherung der Änderungen
Startzeit	Abhängig vom Betriebssystem typischerweise 20 Sekunden	Applikation steht nach wenigen Sekunden bereit

Eine in der Literatur verwendete Einteilung der Virtualisierungs-Technologien ist die Unterscheidung nach dem Einsatzzweck in Server und Desktop. Server-Virtualisierung dient zur Bereitstellung der Infrastruktur und Ausführung von Server-Diensten in einem Rechenzentrum. Desktop-Virtualisierung bezieht sich auf das Endgerät der Anwender, wobei die Virtualisierung einerseits auf lokaler Hardware stattfindet, beispielsweise um eine berufliche und private Umgebung getrennt zuhalten. Andererseits dient der Bildschirm lediglich zur Anzeige eines entfernt bereitgestelltem Desktops (Virtual Desktop Infrastructure) und die lokale Hardware findet lediglich als Interface (Thin Client) Verwendung [Dennis Zimmer u. a. 2012; S. 10-16].

Eine Einteilung ist auch nach dem Geschäftsmodell zur Bereitstellung der Virtualisierung als Service aus der Cloud heraus möglich. Infrastructure as a Service (IaaS) stellt Hardware und Hypervisor bereit, auf welcher VMs mit freier Auswahl des Betriebssystems ausführbar sind. Platform as a Service (PaaS) stellt bspw. Entwicklern vorinstallierte VMs oder Container bereit, um Software zu testen. Software as a Service (SaaS) richtet sich an die Endkunden, welche das fertige Produkt im täglichen Einsatz haben. Diese Einteilung findet aber in der Masterthesis keine Verwendung.

2.3 Marktüberblick von Virtualisierungs-Implementierungen

Im Folgenden werden einige Implementierungen der verschiedenen Virtualisierungsarten gegenübergestellt. Im Bereich der Server-Virtualisierung sind verschiedene Varianten der Paravirtualisierung und Hardwarevirtualisierung aufgelistet. Zur Gegenüberstellung sind anschließend verschieden Varianten der Betriebssystem-Virtualisierung dargestellt. Dies bildet die Grundlage, um aus jedem Bereich ein Kandidat auszuwählen zur weiteren Bearbeitung in den späteren Kapiteln.

Beide Virtualisierungsarten im Vergleich sind an denselben Kriterien messbar. Funktionale Anforderungen, die angeben was das Produkt tun soll, sind in Rücksprache mit dem Kunden, dass mehrere Gast-Systeme parallel, virtuell und sicher auf gleicher Hardware laufen sollen. Weitere nicht funktionale Kriterien, welche die Eigenschaften des Produktes betreffen, sind geringe Komplexität, wenige Abhängigkeiten und dadurch eine hohe Flexibilität. Die Zusammenarbeit mit UCS, und der verwendeten Open Source Lizenz AGPL soll möglich sein. Neben den vom Kunden vorgegeben Ausschlusskriterien, sind weitere eigene nicht funktionale Anforderungen zusätzlich festgelegt worden. Die Bereitstellung als Debian Paket wäre hilfreich zur Integration in UCS, aber auch die Aktualität der Virtualisierungs-Implementierung ist wichtig.

Für die Server-Virtualisierung steht vSphere als kostenloser Hypervisor bereit, um direkt auf purer Hardware installiert zu werden. Die Produktversion ESXi 6 basiert dabei auf Open Source, steht aber unter einer herstellereigenen proprietären Lizenz. Es handelt sich um einen Typ 1 Hypervisor, wobei die Administration über eine bereitgestellte Webseite oder per separatem Windows Client-Programm möglich ist.

Xen[1] ist ein ebenfalls ein kostenloser Typ 1 Hypervisor. Xen wurde 2003 unter GPL veröffentlicht und stellt von Anfang an Paravirtualisierung bereit. Ab Version 3 bietet Xen auch Hardwarevirtualisierung und Mischformen zur Kombination der Virtualisierungsarten untereinander an. Zum Betrieb ist allerdings ein spezieller Kernel notwendig, was die Komplexität erhöht. Mit dem Erscheinen von UCS 4.0 wurde Xen aus UCS herausgenommen, da der Betreuungsaufwand zu groß wurde.

KVM als Hypervisor vom Typ 2 wurde seit Veröffentlichung 2006 unter GPL v2 auf die Hardwarevirtualisierung zugeschnitten. Mit Kernel 2.6.20 (5. Feb. 2007) wurde die Funktionalität direkt über ein Kernel-Modul integriert. Somit ist zum Betrieb von KVM kein spezieller Kernel notwendig. KVM verwendet als Virtualisierungs-Programm im Benutzer-Modus des Kernels eine angepasste Version von QEMU (Quirk-Emulator) und kann hiermit angepasste Gast-Systeme nahezu nativ ausführen. Für unangepasste Gast-Systeme findet von QEMU eine Emulation der Hardware statt, was zuverlässig aber langsam ist. Inzwischen bietet KVM ebenfalls Paravirtualisierung an.

Tabelle 2: Server-Virtualisierung (Stand: 17.04.2015)

	vSpere Hypervisor	**Xen**	**KVM**
Hersteller	VMware, Inc.	Xen Project	Qumranet, Inc.
Aktualität (Version) / Ersterscheinung	13. März 2015 (ESXi 6) / 16. Juni 2006	14. Jan. 2015 (4.5) / 9. Sept. 2003	5. Sept. 2012 (1.2.0) / 5. Feb. 2007
Preis / Lizenz	Kostenlos / Proprietär	Frei / GPL	Frei / GPL v2
Virtualisierungsart	Hypervisor Typ 1	Hypervisor Typ 1	Hypervisor Typ 2
Wirt-System	Vorgegebenes Minimalsystem als Basis	Installiertes System wird auf Ring 1 verschoben und virtualisiert betrieben (dom0 Hypervisor)	Installiertes System wird durch Kernel-Modulerweiterung zum Hypervisor
Gast-System	Hardwarevirtualisierung	Paravirtualisierung (PV), Mischformen (PVHVM, PVH), Hardwarevirtualisierung (HVM)	Hardwarevirtualisierung, Paravirtualisierung
Bewertung	√ kompakte Größe - proprietäre Lizenz	√ Debian-Paket verfügbar - Komplex durch Spezial-Kernel	√ Debian-Paket verfügbar √ Flexibel durch Kernel-Modul

Für die Server-Virtualisierung scheidet der vSpere Hypervisor aus, da dieser unter einer proprietären Lizenz steht. Xen und KVM stehen unter Open Source Lizenzen und sind so zusammen mit UCS und der verwendeten AGPL Lizenz kompatibel. Eine aktuelle Version ist von Xen verfügbar, allerdings ist KVM bevorzugt worden, da der Betrieb per Kernel-Modul vorgesehen ist. Dies gestaltet den Einsatz flexibler, ohne einen speziell für Xen angepassten Kernel vorauszusetzen.

Neben der Server-Virtualisierung per Hypervisor mit virtuellen Maschinen, werden im Folgenden Implementierungen für die Virtualisierung auf Betriebssystemebene genauer betrachtet.

Mit den 2008 veröffentlichten Linux Containers (LXC) ist es möglich ein Linux-Betriebssystem in einer schlanken virtuellen Umgebung auszuführen. LXC wurde mit Version 2.6.29 (23. März 2009) direkt im Kernel integriert und richtet sich an Entwickler, die um einen Container zu erzeugen, ein neues Linux selbst vorbereiten (beispielsweise Debian per debootstrap) oder auf eine der mitgelieferten Betriebssystem-Vorlagen zurückgreifen.

1 XEN ist abgeleitet aus dem griechischen Wortstamm xeno, was Fremder oder Gast bedeutet.

Docker war mit der Veröffentlichung im März 2013 zunächst eine Weiterentwicklung der LXC-Schnittstellen, verwendet aber inzwischen eigene Schnittstellen und unterstützt LXC weiterhin als alternativen Container-Treiber. Docker richtet sich an eine breitere Zielgruppe, wobei bereitgestellte Container für Software-Entwickler zur Auswahl stehen, um Applikationen in einer virtuellen Umgebung auszuführen. Ein Docker-Container basiert auf einem Docker-Image und fasst Betriebssystem sowie Applikationen zusammen. Über die Docker-Plattform sind Images für andere Personen, beispielsweise Tester, bereitgestellt. Auch Server-Administratoren können die in Images verfügbaren Applikation mit allen benötigten Abhängigkeiten herunterladen und starten.

Unter den vorläufigen Projektnamen rkt („Rocket“) und appc („App Container“) arbeitet CoreOS derzeit an einem eigenen Container-System. Im Unterschied zu Docker soll dabei satt einer zentralen Applikationsentwicklerplattform, eine unabhängige neue Container-Technik zum Industriestandard mit dem Fokus auf Sicherheit im Produktivbetrieb entstehen. Rocket ist seit Dezember 2014 verfügbar und ist aktuell noch im Prototyp-Status.

Tabelle 3: Betriebssystem-Virtualisierung (Stand: 10.07.2015)

	LXC	**Docker**	**rkt**
Hersteller	D. Lezcano, S. Hallyn, S. Graber, und weitere	Docker, Inc.	CoreOS, Inc
Aktualität (Version) / Ersterscheinung	10. April 2015 (1.1.2) / 6. Aug. 2008	18. Juni 2015 (1.7.0) / 13. März 2013	17. Juni 2015 (0.6.1) / 27. Nov. 2014
Preis / Lizenz	GNU LGPL v.2.1	Apache License 2.0	Apache License 2.0
Virtualisierungsart	Betriebssystemebene	Betriebssystemebene	Betriebssystemebene
Wirt-System	Installiertes Linux-System	Installiertes Linux-System	Installiertes Linux-System
Gast-System	Linux-Betriebssystem	Linux-Betriebssystem	Linux-Betriebssystem
Bewertung	√ Debian-Paket verfügbar	√ Debianpaket verfügbar √ Plattform für Container-Lebenszyklus (erstellen, ausführen, verteilen)	- Prototyp-Status

Bei der Betriebssystem-Virtualisierung stehen alle vorgestellten Implementierungen unter Open Source Lizenzen und werden aktiv weiterentwickelt. Das Projekt rkt ist viel versprechend mit dem Fokus auf einen einheitlichen Standard, aber ist noch nicht für den produktiven Einsatz geeignet, da es sich aktuell noch im Prototyp-Status befindet. Der Einsatzzweck von LXC ist eher die virtuelle Bereitstellung eines kompletten Linux-Betriebssystems für Entwickler, während Docker ursprünglich einzelne Applikationen in einem Container bereitstellte, inzwischen aber auch ein komplettes Betriebssystem beinhalten kann. Aufgrund der vorbereiteten Infrastruktur zur öffentlichen Bereitstellung von Containern, über welche fertig konfigurierte Applikationen in Images verfügbar sind, ist Docker für den Produktivbetrieb im Endkundenbereich gut geeignet.

Abschließend ist festzuhalten, dass die Virtualisierungs-Implementierungen nur innerhalb ihrer jeweiligen Kategorie in Hinblick auf die Funktionalität vergleichbar sind.

Zusammenfassend zeichnet sich Virtualisierung durch den Verzicht auf exklusive Nutzung der zugrunde liegenden Hardware aus, die durch Mehrfachnutzung und Parallelisierung effizienter nutzbar ist. Bei der Virtualisierung per Para- und Hardwarevirtualisierung ist die Isolierung in eine virtuelle Umgebung mit den jeweiligen Zwischenschichten von Gast-Kernel über den Hypervisor zum Wirt-Kernel auch ein effektiver Schutz vor Fehlern in Gast-Applikationen. Ohne eigenen Gast-Kernel oder Hypervisor als Zwischenschicht ist bei der Betriebssystem-Virtualisierung deutlich, dass der Wirt mit dem Container sehr eng verbunden ist. Ob ein Hypervisor eine Zwischenschicht bildet oder ein gemeinsam genutzter Kernel die Aufgaben steuert; in jedem Fall ist der Wirt-Kernel als letzte Instanz der Verarbeitungskette zur Steuerung der Hardware-Befehle und somit für die Gewährleistung der Sicherheit verantwortlich.

2.4 Sicherheitsaspekte im Linux-Kernel

Der Linux-Kernel setzt das Konzept der CPU-Isolierung um, indem in Ring 0 nur Prozesse im Kernel-Modus und in allen anderen Ringen nur die Prozesse im Benutzer-Modus ausgeführt werden können. Der Benutzer `root` hat auf Betriebssystem-Ebene eine besondere Bedeutung, da Vollzugriff auf alle Ressourcen möglich ist und über den Kernel der Zugriff auch auf hardwarenahe Funktionalität unbeschränkt ist. Betriebssysteme entwickeln sich laufend weiter und sind nicht frei von Fehlern. Zusätzlich stellt der root-Benutzer ein lohnendes Angriffsziel da, um auf privilegierte Ports oder fremde Dateien zugreifen zu können. Als Gegenmaßnahmen hat Linux eine Reihe von Sicherheitsfunktionen in den Kernel integriert.

Aus Sicht des Linux Betriebssystems führen Benutzer Kommandos aus, wobei diese den normalen Benutzer und den root-Benutzer unterscheiden. Normale Benutzer haben unter Linux nur Zugriff auf eigene Dateien, Gruppen-Dateien und solche Dateien, die explizit für alle Benutzer gekennzeichnet sind. Die *Dateiberechtigungen* sind in den drei Bereichen „User“, „Group“ und „All“ auf Lesen, Schreiben und ausführenden Zugriff konfigurierbar. Auf weitere bereitgestellte Ressourcen, wie beispielsweise die Soundkarte, erhalten normale Benutzer nur Zugriff wenn das Gerät und der Nutzer einer gemeinsamen Gruppe zugeordnet sind. Der root-Benutzer hat übergreifende Berechtigungen auf alle Ressourcen des Systems, und kann ohne Einschränkung auf Dateien und Prozesse zugreifen. Durch Ausbrechen aus vorgegeben Beschränkungen ist dies auch für den normalen Benutzer denkbar. In einem nicht weiter abgesichertem System, erhalten somit normale und root-Benutzer unter dem Aspekt der Sicherheit eine besondere Beachtung.

Neben Programmen, die als Prozesse per Multitasking parallel ausgeführt werden, gibt es noch kleinere Einheiten an Aufgaben. Ein Muster in der Informatik ist das Prinzip von „Divide and conquer“, wobei der Algorithmus große Aufgaben in kleinere Pakete aufteilt, um diese getrennt, aber parallel zu verarbeiten.

Eine Möglichkeit ist es dabei eine Kopie eines laufenden Prozesses zu erstellen, einen sog. *Fork*. Im Vergleich zum ursprünglich gestarteten Prozess, dem Eltern-Prozess, hat der Kind-Prozess einen eigenen virtuellen Speicherbereich und eine eigene Prozess-ID. Der Kind-Prozess erbt dabei aber alle Berechtigungen, und führt den gleichen Code aus, bearbeitet diesen jedoch unabhängig. Ausgelöst durch einen Kontextwechsel, bekommen Änderungen an Speicherinhalten durch das „Copy on Write“-Verfahren anschließend einen eigenen Speicherbereich für den Kind-Prozess zugewiesen. Prozesse sind allerdings für kleinere Aufgaben unhandlich, da die Interprozesskommunikation (IPC) untereinander teuer ist, was auch für die Erstellung eines Fork gilt.

Eine leichtgewichtigere Variante demgegenüber ist der *Thread*, welcher keine Kopie des Prozesses ist, sondern den gleichen Speicherbereich des Prozesses nutzt, wodurch auch der Kontextwechsel schneller abläuft. Aufgaben sind so in kleinere Teile verpackt und führen zu einer parallelen Verarbeitung, während der Eltern-Prozess lediglich die Steuerung und Zusammenführung der Ergebnisse übernimmt.

Eine Problematik sind rekursiv in einer Endlosschleife erzeugte Prozesse, sogenannte Fork-Bomben. Der Linux-Kernel kontrolliert und limitiert standardmäßig per *ulimit* die erlaubte maximale Anzahl der Prozesse pro Benutzer, belegter CPU-Zeit und gleichzeitig geöffnete Dateien/Verbindungen pro Prozess [General Commands Manual 2014]. Die aktuellen Ressourcengrenzen sind mit dem Befehl `ulimit -a` in Listing 1 abrufbar.

Listing 1: Ressourcengrenzen mit ulimit

```
# ulimit -a

core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 7922
max locked memory       (kbytes, -l) 64
```

```
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 7922
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

Zur Abschottung von Verzeichnissen findet das Programm *chroot* Verwendung, es steht für „change root“ und dient dazu das Rootverzeichnis zu ändern, so dass ausgeführte Prozesse und Kind-Prozess in einer Sandbox laufen. Entwickelt wurde chroot seit 1979 und hat nicht die Absicherung von Applikationen im Fokus, sondern das Bereitstellen einer virtuellen Umgebung im Benutzer-Modus des Kernels. Allerdings ist es möglich dieser beschränkten Umgebung zu entkommen, vor allem wenn der Zugriff auf den `chroot` Befehl erlaubt ist, was besonders kritisch ist, da chroot über root-Rechte ausgeführt ist [Friedl 2002].

Die Berechtigungs-Bits *setuid* und *setgid* sind auf ausführbare Programme vom root-Benutzer anwendbar. Zur Prozess-Laufzeit ist es normalen Benutzern möglich, so markierte Programme mit den Berechtigungen des Programm-Eigentümers, also unter der jeweiligen Benutzer oder Gruppenkennung, auszuführen. Ein typischer Anwendungsfall ist die Vergabe von Berechtigungs-Bits, um die Ausführung von Programmen zu ermöglichen, welche für privilegierte Aktionen root-Rechte erfordern, aber ohne dass ein normaler Benutzer das Kennwort des root-Benutzers dazu zu benötigt. Zur Bereitstellung privilegierter Funktionalität für normale Benutzer, findet dieser Mechanismus beispielsweise Verwendung für das Programm zur Verwaltung der Benutzerkennwörter (passwd) und für das ICMP Netzwerk-Tool (ping). Die Passwortdatei des Systems muss vor fremdem Zugriff absolut geschützt sein, jedoch soll jeder Benutzer das eigene Kennwort ändern dürfen. Und das Programm `ping` benötigt einen geöffneten Socket, um die Antworten der ICMP-Anfragen zu empfangen. In beiden Fällen wäre die Nutzung der Programme einem normalen Benutzer nicht möglich, da privilegierte Berechtigungen notwendig sind. Im Dateisystem ist das setuid-Bit anstelle des ausführenden Benutzer-Zugriffs (mit „s“ statt „x“ in Listing 2) gesetzt.

Listing 2: Programm ping mit setuid-Bit

```
# ls -l /bin/ping
-rwsr-xr-x 1 root root 33616 Apr  8  2014 /bin/ping
```

Der Vorteil der Berechtigungs-Bits ist eine gezielte Rechtevergabe für bestimmte Programme, welche zu Laufzeit höhere Rechte benötigen, als für einen normalen Benutzer zugänglich sind. Der Nachteil ergibt sich aus dem daraus entstehenden möglichen Sicherheitsrisiko der erhöhten Rechtevergabe. Ein Fehler in dem jeweiligen Programm, ausgelöst durch nicht validierte Benutzereingaben, kann das ganze System kompromittieren [Wheeler 1999; S. 46].

Durch die Verwendung von *Capabilities* sind die Zugriffe auf den Linux-Kernel und die dafür benötigten Berechtigungen feiner abbildbar. Statt einem Programm generelle root-Rechte zu erlauben, lassen sich nur die wirklich benötigten Ressourcen für den vom Benutzer gestarteten Prozess freischalten. Die Unterstützung für Capabilities bringt der Kernel ab 2.2.11 (10. Aug. 1999) mit. Eine Auswahl für die in diesem Kapitel genannten Berechtigungen sind in der folgenden Tabelle aufgelistet. Aktuell sind derzeit 38 verschiedene Capabilities nutzbar, wobei die sogenannten Capability Attribute pro Thread aktiv sind [Linux Programmer's Manual 2015].

Tabelle 4: Berechtigungen mit Capabilities

Capability Attribut	Beschreibung
cap_chroot	Berechtigung für das Programm chroot.
cap_dac_override	Berechtigung zur Umgehung von Lese-, Schreib- und Ausführungs-Zugriff per Dateiberechtigung.
cap_dac_read_search	Berechtigung zur Umgehung von Leseberechtigungen auf Dateien und Verzeichnisse und Ausführen von Berechtigungsprüfungen. Zugriff auf `open_by_handle_at()` per Kernel-Aufruf zum direkten Zugriff auf Dateien per Inode-Nummer.
cap_fowner	Erlaubt die Umgehung der Prüfung, dass die Dateisystem-UID eines Prozesses mit der UID einer Datei übereinstimmen muss.
cap_setuid	Berechtigung um die Prozess-UID zu manipulieren (z. B. für das Programm setuid).
cap_setfcap	Berechtigung um Capabilities für Dateien zu setzen (z. B. für das Programm setcap).
cap_net_raw	Berechtigung um RAW-Sockets zu öffnen.
cap_net_admin	Privilegierte Operation im Zusammenhang mit Netzwerk zur Modifikation von IP-Adressen, Netzwerkkarten, Routen und Firewall-Regeln (iptables).
cap_sys_admin	Erlaubt eine Vielzahl an privilegierten Operationen, die nicht feiner aufzugliedern sind, wie die Modifikation des Rechnernamens, Domainnamens, Swap-Speichers und das Einbinden von Dateisystemen (mount).

Bestimmte Capabilities sind für genau einen Einsatzzweck zur Kommunikation mit dem Kernel vorgesehen, wie beispielsweise `cap_setuid` und `cap_chroot`. Dem root-Benutzer, oder einem normalen Benutzer mit entsprechender Berechtigung (`cap_setfcap`), ist es möglich Capabilities für Programme auf Dateiebene zu vergeben. Im nachfolgenden Beispiel ist, zur Erhöhung der Systemsicherheit, dem Programm `ping` das setuid-Bit entzogen worden. Aber um weiterhin Antworten auf ICMP-Nachrichten empfangen zu können, ist die entsprechende Capability notwendig, um als normaler Benutzer einen Socket öffnen zu dürfen (`cap_net_raw`). Listing 3 stellt dies exemplarisch dar.

Listing 3: Programm ping mit Capability anstatt setuid-Bit

```
# chmod -s /bin/ping
# setcap cap_net_raw+ep /bin/ping
```

Statt übergreifender Zugriffsrechte wie mit Berechtigungs-Bits ist der Vorteil von Capabilities, dass Berechtigungen nur für bestimme privilegierte Funktionalität gezielt aktiviert ist. Da die Berechtigungen von außen gesetzt sind, ist es einem normalen Benutzer nicht möglich diese zur Laufzeit zu beeinflussen. Der erhöhte Administrationsaufwand durch das Herausfinden der wirklich benötigten privilegierten Funktionalität ist allerdings ein Nachteil, wenn dies vom Entwickler des Programms nicht dokumentiert ist.

Eine weitere Funktion unter Linux ist seit Kernel 2.6.24 (24. Jan. 2008) mit *Cgroups* (Abkürzung von control groups) verfügbar. Die Kontrollgruppen bieten eine Schnittstelle zur Ressourcenbegrenzung der Hardware, Priorisierung, Kalkulation und Kontrolle einzelner Prozesse bis hin zu einem virtualisierten Betriebssystem. Linux ist mit Cgroups in Subsysteme unterteilt, welche zur Ressourcensteuerung notwendig sind. Die Steuerung erfolgt für jedes Subsystem separat über im Verzeichnissystem abgelegte Dateien, auf welche nur der root-Benutzer Zugriff hat. Somit ist es möglich Prozesse bezüglich unterschiedlicher Kategorien in Gruppen zu partitionieren. Prozesse innerhalb einer Gruppe teilen sich gemeinsam ein zugewiesenes Kontingent an Ressourcen. Zur Limitierung von Prozessen, sind die in der folgenden Tabelle aufgelisteten 11 Subsysteme verfügbar.

Tabelle 5: Ressourcenbegrenzung mit Cgroups

Cgroup Subsystem	Beschreibung
blkio	Block IO Controller: Limitiert Lese- und Schreibzugriffe auf Gerätedateien.
cpu	Limitierung der CPU-Zeit von Tasks einer Gruppe.
cpuacct	CPU Accounting Controller: Erfassung der CPU-Ressourcen von Tasks einer Gruppe.
cpuset	Mechanismus zur Zuordnung von CPU- und Arbeitsspeicher-Einheiten (Nodes) zu Tasks einer Gruppe.
devices	Device Whitelist Controller: Limitierung von Zugriffen auf Devices für Gruppen.
freezer	Suspend und Resume von Tasks.
hugetlb	HugeTLB Controller: Limitierung der Virtual Memory Tables.
memory	Memory Resource Controller: Limitierung des Arbeitsspeichers für Tasks einer Gruppe.
net_cls	Network classifier: Markieren von Netzwerkpaketen von Tasks einer Gruppe (zur späteren Verarbeitung per Netz-Filter z. B. `iptables` oder `tc`).
net_prio	Network priority: Prioritäten für Netzwerk-Traffic von Tasks einer Gruppe.
perf_event	Überwachung der Threads von Tasks einer Gruppe.

Ein unter Linux gewöhnliches Verhalten zur Erstellung von im Hintergrund laufender Daemon-Prozesse ist die Verwendung des Doppel-Forks. Das ausführende Programm startet einen Kind-Prozess, welcher einen weiteren Fork erzeugt, anschließend beenden sich Kind- und Eltern-Prozess. Der verbleibende Kind-Kind-Prozess, der nun als Daemon-Prozesse arbeitet, ist infolgedessen automatisch dem init-Prozess mit root-Berechtigungen zugeordnet. Somit entgeht der Prozess ursprünglich konfigurierten Ressourcen-Beschränkungen. Cgroups bieten Schutz im Zusammenhang mit per Fork erzeugten Prozessen, indem mit Cgroups festgesetzte Einschränkungen automatisch für alle Kind-Prozesse derselben Prozess-Familie gelten. So ist die übermäßige Ressourcenbelegung effektiv vermeidbar [Kernel Documentation 2015a].

Im Beispiel für das Programm `ping` ist mit dem Subsystem `net_cls` der Traffic markierbar, um den maximalen Datendurchsatz des Netzwerks in einem definiertem Verhältnis zu begrenzen. Der Vorteil liegt somit in der Partitionierung der Ressourcen des Systems, was ein normaler Benutzer nicht umgehen kann. Der Nachteil ist, dass die Ressourcen-Beschränkung zur Laufzeit, durch Zuordnen der erhaltenen Prozess-ID zu der jeweiligen Gruppe, aktiv von außen zu konfigurieren ist. Durch Verwendung von „CGroup Rules Engine Daemon“ lässt sich dieser Vorgang automatisieren, oder Programme zur Nutzung von Cgroups wie KVM und Linux-Container kümmern sich selber um die Konfiguration und Beschränkung ihrer Kind-Prozesse.

Mit der *Namespace*-Isolierung ist eine weitere Funktionalität seit Kernel 2.6.19 (29. Nov. 2006) verfügbar. Normalerweise startet ein Kind-Prozess in der Umgebung (Namensraum) des Eltern-Prozesses. Dies bedeutet, dass zwar jeder Prozess eigene Speicherbereiche bekommt, aber Zugriffe auf gemeinsame Ressourcen erlaubt sind. Mit der Namespace-Isolierung lassen sich Ressourcen so voneinander trennen, dass Prozesse anderer Benutzer und die dabei verwendete Ressourcen nicht mehr sichtbar und somit nicht mehr direkt beeinflussbar sind. Technisch gehört die Namespace-Isolierung nicht zu den Cgroups, meist findet allerdings beides in Kombination Verwendung. Der Linux-Kernel bietet aktuell seit Version 2.6.26 (13. Juli 2008) sechs verschiedene Flags von Namespaces an, die in der folgenden Tabelle aufgelistet sind.

Tabelle 6: Namensraumisolierung mit Namespaces

Namespace Flag	Beschreibung
CLONE_NEWIPC	IPC: Isolierung der Interprozesskommunikation
CLONE_NEWNS	Mount: Isolierung von Dateisystem (VFS)
CLONE_NEWNET	Network: Isolierung von Netzwerk
CLONE_NEWPID	PID: Isolierung der Prozesskennung
CLONE_NEWUSER	User: Isolierung der Benutzer ID
CLONE_NEWUTS	UTS: Erlaubt die Änderung des Rechnernamens

Wenn von einer Linux-Shell aus eine weitere Shell gestartet wird, so erbt diese im Normalfall alle Berechtigungen und teilt sich alle Ressourcen mit der übergeordneten Shell. So ist es beispielsweise möglich den aktuellen Netzwerk-Namespace abzutrennen, um so den Zugriff auf das Netzwerk zu verhindern. Die Funktionsweise der getrennten Namensräume macht das Programm `unshare` in Listing 4 sichtbar, wobei eine neue, aber beschränkte Linux-Shell gestartet wird.

Listing 4: Programm unshare zur Abtrennung des Netzwerk-Namensraums

```
# unshare --net /bin/bash
# ifconfig
# ping example.com
connect: Network is unreachable
```

Für den Kind-Prozess existiert in diesem Beispiel die Netzwerkkarte des Systems nicht mehr, so dass keine Programme mehr funktionieren, die zur Arbeit das Netzwerk benötigen, wie die Programme `ifconfig` oder `ping`. Der Vorteil ist die klare Isolierung des gestarteten Prozesses und die Abtrennung vom Basis-System. Ein Nachteil ist aber, dass dieser Mechanismus Prozesse in verschiedene Namensräumen mit gleicher Kennung (PID) und Benutzer mit gleicher Kennung (UID) nicht vollständig voneinander isolierten kann. Der Hintergrund ist, wenn beispielsweise mehrere Container parallel laufen und der Kernel zwischen verschiedenen root-Benutzern mit der gleichen UID 0 in unterschiedlichen Namensräumen unterscheiden soll. Dies macht sich bemerkbar auf Dateisystemebene und der Ebene der Prozesskommunikation, ebenso wenn Benutzer mit gleicher UID in verschieden in verschiedene Namensräumen aktiv sind [Kernel Documentation 2015b].

Eine weitere Sicherheitsfunktion des Linux-Kernel ist *Seccomp*, was für „SECure COMPuting with filters“ steht. Unterliegt ein Prozess dieser Beschränkung, dann ist die weitere Ausführung einem Sandbox-Mechanismus unterworfen, indem sämtliche Kernel-Funktionen (System-Calls) bis auf Minimalzugriffe (exit(), sigreturn(), read() und write()) abgelehnt werden. Seccomp ist seit Kernel 2.6.12 (17. Juni 2005) verfügbar, allerdings beschränkt sich die Nutzung auf bisher wenige Programme wie Firefox, Chrome, LXC und KVM-QEMU.

Zusammenfassend eigenen sich die vorgestellten Möglichkeiten um Prozesse in einem abgeschotteten Bereich auszuführen (chroot), Benutzern von außen per Berechtigungs-Bits benötigte root-Rechte für einen Prozess zu gewähren (setuid und setgid) und diese Rechte zu entfernen, um feiner nur notwendige Aktionen zu erlauben (Capabilities). Zusätzlich sind Kontrollen und das Priorisieren der Ressourcenverwendung von Dateien, CPU, Arbeitsspeicher und Netzwerk möglich (Cgroups). Die Sichtbarkeit auf bestehende Ressourcen kann für neue Prozesse eingeschränkt werden, um diese zu Isolieren (Namespaces) und um Aufrufe so einschränken zu lassen, dass Prozessen nur minimale Kernel-Funktionalität bereit steht (Seccomp).

Alle Varianten haben gemeinsam, dass Rechte, Ressourcen und Sichtbarkeit von außen, vom Eltern-Prozess entfernt und beschränkt werden. Ein Prozess kann selbstverständlich diese Berechtigungen auch freiwillig abgeben. Es handelt sich bei den vorgestellten Möglichkeiten um Standard-Funktionalitäten des Linux-Kernels, was bedeutet, dass keine zusätzlichen Module erforderlich sind und kein eigenes Anpassen und Kompilieren des Kernel-Quelltextes notwendig ist.

Aus der nachfolgenden Tabelle ist abzulesen, wie die vorgestellten Implementierungen der Server-Virtualisierung und Betriebssystem-Virtualisierung aus Kapitel 2.3 die von Linux bereitgestellten Sicherheitsaspekte nutzen. Während die Trennung des Wirt-root-Benutzers vom Gast-System bei Hypervisor basierter Virtualisierung zur Basisfunktionalität gehört, ist bei der Betriebssystem-Virtualisierung eine Aufsplittung nicht immer gegeben. In der Auflistung ist KVM, mit dem im Kernel-Modus laufenden Hypervisor, getrennt von dem im Benutzer-Modus ausgeführtem KVM-QEMU Programm betrachtet (Siehe Abschnitt 3.1.1).

Tabelle 7: Virtualisierungs-Implementierungen und Sicherheitsaspekte (Stand: 10.07.15)

	Chroot	**setuid**	**Capabilities**	**Control Groups**	**Namespaces**	**Seccomp**	**Wirt root-Benutzer getrennt von Gast**
Vspere Hypervisor (ESXi 6.0 GA)[2]	-	-	√	-	-	√	√
Xen Hypervisor (4.1.4)[3]	√	√	-	-	-	-	√
KVM Hypervisor (1.2.0)[4]	-	-	-	-	-	-	√
KVM-QEMU (2.3.90)[5]	√	√	√	√	√	√	√
LXC (1.1.2)[6]	√	√	√	√	√	√	√
Docker (1.7.0)[7]	√	√	√	√	√	- (bereits in Planung)	-
rkt (0.6.1)[8]	√	-	√	√	√	-	-

Die einzelnen Merkmale der Virtualisierungseigenschaften sind auf der Basis der Quelltexte, der Handbücher und Anleitungen der jeweiligen Produkte ermittelt worden (siehe Fußnoten).

2 ESXi Lizenzbedingungen: open_source_license_ESXI6.0_GA.txt (https://my.vmware.com/de/web/vmware/details?downloadGroup=ESXI600_OSS&productId=490).
3 XEN Schnittstelle in Linux auf GitHub (https://github.com/torvalds/linux/tree/master/include/xen).
4 KVM Schnittstelle in Linux auf GitHub (https://github.com/torvalds/linux/tree/master/virt/kvm).
5 QEMU Quelltext im Git (git://git.qemu-project.org/qemu.git – http://www.linux-kvm.org/page/Code).
6 LXC auf GitHub (https://github.com/lxc/lxc).
7 Docker auf GitHub (https://github.com/docker/docker).
8 rkt auf GitHub (https://github.com/coreos/rkt).

2.5 Grundlagen der Informationssicherheit

Im Zusammenhang mit der Informationssicherheit sind die drei Bereiche Organisation, Informationswerte und Sicherheitsziele deklariert [Heinrich Kersten u. a. 2013; S. 24].

Die Definition der Begriffe wird anhand eines Unternehmens oder einer Abteilung deutlich, was hierbei die Organisation darstellt. Vor Ort stehen Daten und Informationen bereit, wie beispielsweise Betriebsgeheimnisse, was den Informationswerten entspricht. In Sicherheitszielen ist die ordnungsgemäße Datenverarbeitung im Unternehmen definiert.

Die Informationssicherheit umfasst somit alle Bereiche, in welchen Daten erstellt, verarbeitet und zu schützen sind. Zur Einordnung der Begrifflichkeiten „Sicherheit“ und „Schutz“ wird der Begriff des Risikos verwendet.

„Als Risiko wird nach ISO 27005 eine Kombination aus der

- Eintrittswahrscheinlichkeit eines (unerwünschten, unerwarteten, schädlichen) Ereignisses und
- dessen Konsequenzen definiert.“ [Heinrich Kersten u. a. 2013; S. 27]

Der Umgang mit einem Risiko erfordert ein Risikobewusstsein, das Wissen von Schwachstellen und die Risikoeinstufung, um darüber hinaus Bedrohungen zu erfassen, zu bewerten und messbar zu machen. Dann erst sind konkrete Gegenmaßnamen möglich, um das Eintreten eines Risikos und damit verbundene mögliche Kosten wie Umsatzausfall oder Regressanspruch zu verhindern.

In einer mathematischen Formel ausgedrückt bedeutet dies:

Risiko = Eintrittswahrscheinlichkeit der Bedrohung × Schadensausmaß [Müller 2014; S. 605]

Die Eintrittswahrscheinlichkeit gibt eine Vorhersage an, mit welchem Grad der Gewissheit die Bedrohung eintreffen kann. Die Bedrohung ist die Gefahr, die aus einer möglichen Schwachstelle entsteht, so dass die Informationswerte bedroht sind. Das Schadensausmaß beschreibt die Auswirkung und das Schadenspotential der möglichen Konsequenzen auf den Geschäftsalltag der Organisation über die Schwächung vorhandener Schutzziele.

Im Folgenden sind die einzelnen Bereiche, aus welchen sich potentielle Risiken ergeben können, genauer betrachtet.

2.5.1 Bedrohungen der Werte durch existierende Applikationen

Für die Entwicklung neuer, sicherer IT-Systeme mit Open Source-Werkzeugen stehen für die Konzeption und Implementierung verschiedene Vorgehensmodelle bereit. Für Projekt-Management und in der Software-Entwicklung hat sich die Verwendung von anerkannten Softwaremustern (Pattern) zur Einhaltung von Software-Qualitäts-Standards etabliert.

Bestehende existierende Applikationen sind lediglich von außen auf die Einhaltung von Qualitätsstands bewertbar. Der Umfang des Quelltextes ist ein Anhaltspunkt, wie viele Fehler statistisch enthalten sein können. Durch statische Analysen lassen sich mögliche Programmierfehler aufdecken. In einer Studie wurde ermittelt, dass Open-Source-Projekte eine durchschnittliche Fehlerdichte von 0,69 haben, der Linux-Kernel 0,73 und KVM 1,54 [Coverity 2013]. Dabei gilt eine Anzahl von unter 1,0 als Industriestandard für hohe Qualität, was einem Fehler je 1.000 Quelltextzeilen entspricht.

Unentdeckte Softwarelücken, ob Programmierfehler oder sog. Hintertüren in Applikationen, gefährden nicht nur die Applikationen selbst, sondern können die ganze Systemsicherheit gefährden und letztendlich unternehmensweit die Werte bedrohen. Die dadurch bedrohten Informationswerte der Organisation sind Daten, Dateien und Datenträger wie beispielsweise:

- „Information
  - Informationen aus der Geschäftstätigkeit der Organisation, Daten, Dateien, Verzeichnisse, Datenbanken.

  - Eigene Dokumente wie Verträge, Verfahrens- und Arbeitsanweisungen, Schulungsmaterial, Notfall-Handbücher, Wiederanlaufpläne.
  - Externe Dokumente wie System-Beschreibungen und Nutzerhandbücher.
  - Alle Arten von Aufzeichnungen bzw. Protokollen.
- Software
  - Anwendungssoftware
  - System-Software
  - Entwicklungswerkzeuge." [Heinrich Kersten u. a. 2013; S. 24]

Aber auch physikalische Objekte wie Rechner oder Datenmedien, die Infrastruktur mit Rechenzentrum, genutzte und erbrachte Dienstleistungen, die Fähigkeiten der Mitarbeiter und das Ansehen des Unternehmens in der Öffentlichkeit sind weitere mögliche bedrohte Werte [Heinrich Kersten u. a. 2013; S. 25].

2.5.2 Eintrittswahrscheinlichkeit und sicherer Betrieb

Die Eintrittswahrscheinlichkeit gibt die Möglichkeit von Schäden der Applikation an. Anstatt einer zeitaufwändigen und fehleranfälligen Bewertung empfiehlt das BSI „sowohl für die Eintrittswahrscheinlichkeit als auch für die potentielle Schadenshöhe mit Kategorien zu arbeiten. Hierbei sollten nicht mehr als 3 bis 5 Kategorien verwendet werden, z. B.

- Eintrittswahrscheinlichkeit: selten, häufig, sehr häufig
- Potentielle Schadenshöhe: mittel, hoch, sehr hoch“ [Bundesamt für Sicherheit in der Informationstechnik 100-1 2009; S. 28]

Die Überwachung der Vorgänge und Protokollierung der Zugriffe im laufenden Betrieb der Applikation ist notwendig. Zur Auswertung können diese Informationen zur Erkennung von Sicherheitsverletzungen hilfreich sein.

2.5.3 Auswirkung auf Schutzziele und Geschäftsumfeld

„Unter die Sicherheitsziele für Informationswerte fallen unter anderem die bekannten Ziele der Vertraulichkeit, Integrität und Verfügbarkeit.“ [Heinrich Kersten u. a. 2013; S. 26]

Für die Informationssicherheit im Geschäftsumfeld ist die Einhaltung der allgemeinen Ziele wichtig:

- Vertraulichkeit: Das Unternehmen muss sich darauf verlassen können, dass Informationswerte in einer virtualisierten Umgebung nicht durch Dritte eingesehen werden können, was durch Zugriffsberechtigungen oder Verschlüsselung steuerbar ist.
- Integrität: Die Daten dürfen nicht in ihrem Inhalt verändert werden, was durch Signieren oder erstellen von Prüfsummen sicherstellbar ist.
- Verfügbarkeit: Die Daten sollen auf Abruf ohne Verzögerung bereitgestellt werden können und vor Datenverlust geschützt werden.

Neben den allgemeinen Schutzzielen aus der Informationssicherheit für Informationswerte, kommen „auch andere Ziele wie

- Authentizität
- Zurechenbarkeit oder
- Nicht-Abstreitbarkeit [in Frage].“ [Heinrich Kersten u. a. 2013; S. 56]

Wichtige, im Geschäftsumfeld zu schützende Werte wie Firmengeheimnisse oder Patente, erhalten für die Einhaltung der Schutzziele eine besondere Aufmerksamkeit.

3 Konzeption und Architektur

In diesem Kapitel werden durch die Bereiche Architektur und Systemkonzept, die notwendigen Voraussetzungen für die anschließende Risikoeinschätzung definiert.

Ausgangspunkt ist ein detaillierter Blick auf die zu untersuchende Virtualisierungsarchitektur, da hier eine genaue Kenntnis der Architektur in einer virtuellen Umgebung wichtig ist, um anhand dieser bestehende und mögliche Gefährdungen einzuordnen.

Anschließend wird das Systemkonzept zur Bereitstellung virtueller Komponenten und der geteilten Ressourcen und Kommunikationspfade beleuchtet. Dies ist zusammen mit den benötigten Berechtigungen und eingesetzten Technologien die Grundlage der anschließenden Bedrohungsanalyse.

3.1 Virtualisierungsarchitektur

KVM als Server-Virtualisierung und Docker als Betriebssystem-Virtualisierung unterscheiden sich durch die unterschiedlichen Konzepte zur Bereitstellung einer virtualisierten Umgebung. Eine spätere Untersuchung und Bewertung der Bedrohungen und Risiken erfordert zunächst eine vergleichende Darstellung der Virtualisierungs-Implementierungen der beiden eingesetzten Techniken.

3.1.1 KVM-Hypervisor Architektur

Die Architektur von KVM ergibt sich aus mehreren Einzelkomponenten, welche zusammen die für die Virtualisierung benötigte Funktionalität bereitstellen. Dies lässt sich in drei Gruppen aufgliedern; in den Hypervisor, QEMU und die Verwaltungstools. KVM stellt als Kernel-Modul den Kern der Virtualisierungs-Implementierung dar und wandelt den Linux-Kernel in einen Hypervisor vom Typ 2 um. QEMU ist für den Betrieb der VM und die Emulation von Hardware zuständig. Verschiedene Bibliotheken und Tools dienen zur Verwaltung der VM.

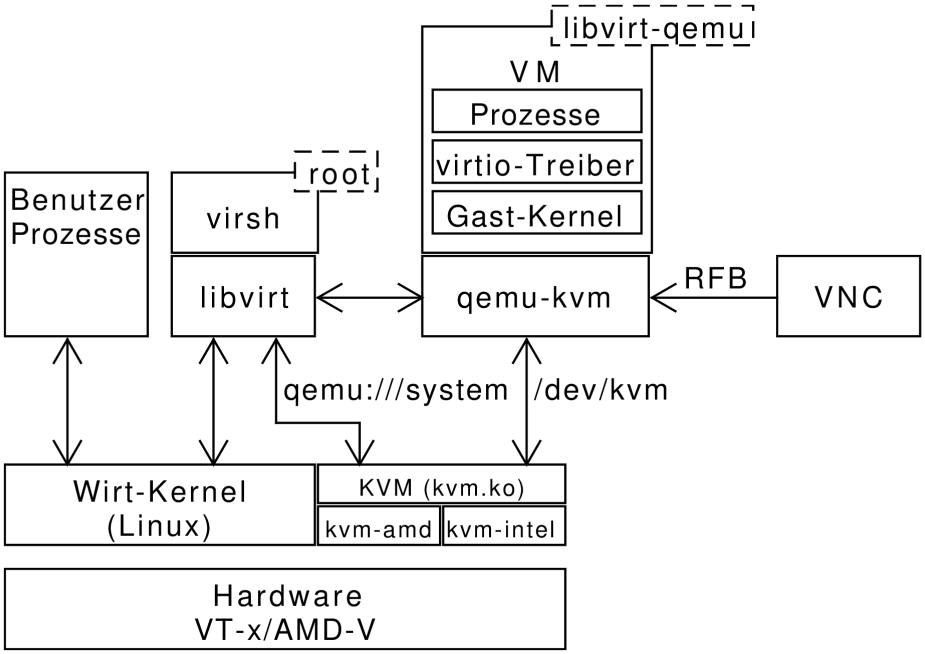

Abbildung 5: Virtualisierungsarchitektur von KVM 1.1.2 unter UCS 4.0-3 (Debian 7.8)

Die in Abbildung 5 dargestellte Architektur unterteilt sich von oben nach unten gesehen in drei Schichten: Benutzer-Prozesse, Kernel und Hardware mit Virtualisierungsunterstützung der CPU.

Die Administration einer KVM-basierten VM auf Seite des Wirt-Systems verläuft wie folgt:

- Der root-Benutzer verwendet das Programm `virsh` von der Linux-Shell aus, um eine VM anzulegen und zu verwalten. Intern kommt dabei die Bibliothek libvirt zum Einsatz, welche in der Programmiersprache C entwickelt ist und hardwarenah ohne Overhead mit dem Hypervisor per URI kommuniziert. Somit ist auch ein zentrales Management verteilter Visualisierungssysteme über das Netzwerk möglich.
- Das generische Kernel-Modul KVM (kvm.ko) stellt die benötigte Funktionalität des Hypervisors bereit. Zusätzliche Prozessor-Module (kvm-amd.ko und kvm-intel.ko) enthalten die für die jeweilige Hardwareplattform angepassten CPU-Befehle.
- Für eine neue VM sammelt `libvirt` die benötigten Informationen zusammen und speichert diese in einer XML-Datei. Das benötigte virtuelle BIOS liegt im Binär-Format vor (bios.bin, vgabios.bin und vgavios-cirros.bin) und ein zu installierende Betriebssystem ist über ein virtuelles CD-Rom-Laufwerk als Datei im iso-Format einzubinden.
- Die Festplatte liegt auf dem Wirt als langsam anwachsende Datei („thin-provisioned“), welche beispielsweise im qcow2-Format das Copy-On-Write-Verfahren, Verschlüsselung, Komprimierung und Snapshots unterstützt. Alternativ sind auch block-orientierte Netzwerk-Speicher, beispielsweise über iSCSI verwendbar.

Die Bereitstellung der VM durchläuft die folgenden Schritte:

- Jede VM startet über einen neuen separaten `qemu-kvm`-Prozess, also kein Kind-Prozess, mit den Benutzerrechten `libvirt-qemu`. Dieser Benutzer ist in der Benutzergruppe `kvm`, so dass die Kommunikation direkt mit dem Hypervisor über die Gerätedatei `/dev/kvm` möglich ist, ohne das dazu für den Prozess root-Rechte notwendig sind.
- Der `qemu-kvm`-Prozess belegt den ihm zugewiesenen Arbeitsspeicher auf dem Wirt-System komplett, wodurch für die jeweilige VM diese Menge garantiert ist.
- Innerhalb der VM emuliert QEMU die Gast-Hardware, was sehr Rechenintensiv ist, weswegen spezielle Treiber innerhalb der VM zum Einsatz kommen können. Dabei wird `virtio` als Linux-Standard für paravirtualisierte Treiber wie Blockgeräte und Netzwerkschnittstellen verwendet. Dies ist vergleichbar ist mit den Guest-Tools unter VMware und VirtualBox. Somit ist keine Emulation, sondern eine direkte Kommunikation mit dem Wirt-Kernel möglich, um annähernd native Performance bereitzustellen.
- Nach außen hin steht über den Port 5900 (und jeweils um eins inkrementiert für jede weitere VM) die grafische Ausgabe der VM über Remote Framebuffer Protocol (RFB) bereit. Dies ermöglicht eine entfernte Steuerung der VM per virtuellem Monitor, über Programme mit Hilfe von Virtual Network Computing (VNC) oder Remote Desktop Protocol (RDP). Diese entfernten Systeme sind somit auch als Desktop-Virtualisierung nutzbar (vgl. 2.2 Varianten der Virtualisierung).

3.1.2 Docker-Container Architektur

Im direkten Vergleich der Architektur von KVM (Abbildung 5) zu Docker (Abbildung 6) ist kein spezieller Prozessor mit Unterstützung von Hardwarevirtualisierung notwendig, was den Einsatz von Docker auch unter Kostenaspekten im Geschäftsalltag interessant macht.

Docker setzt sich im Vergleich zu KVM aus weniger einzelnen Elementen zusammen, wobei die Architektur von Docker aus den Komponenten Hub, Client und Daemon besteht. Technisch gesehen sind Client und Daemon das gleiche Programm und unterscheiden sich lediglich in der Ausführung ihrer Betriebsart.

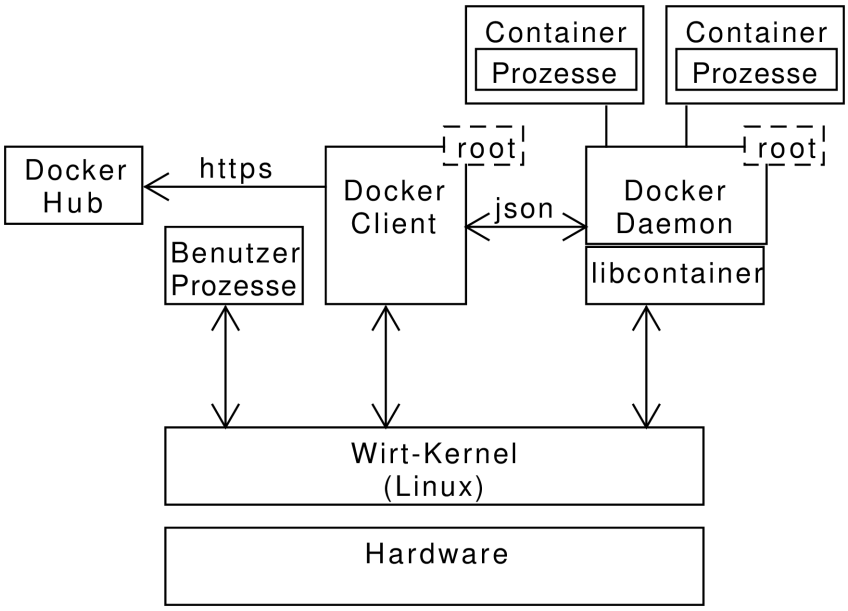

Abbildung 6: Virtualisierungsarchitektur von Docker 1.7.0 unter UCS 4.0-3 (Debian 7.8)

Die Administration von Docker findet per Client statt:

- Die Ausführung des Clients ist standardmäßig mit dem root-Benutzer möglich, oder alternativ mit normalen Benutzern in der Gruppe `docker`. Der Client kann Container starten, stoppen, löschen und auch Befehle innerhalb laufender Container ausführen.
- Die Kommunikation mit dem Daemon findet lokal per WebSocket (`/var/run/docker.sock`) über eine RESTful API im json-Format statt, oder wahlweise per TCP/IP zur verteilten Bereitstellung entfernter Docker Wirt-Rechner.
- Virtuelle Systeme basieren auf Images, welche über den Docker Hub heruntergeladen werden können. Alternativ ist es auch möglich lokal per `chroot` und `debootstrap` eigene Images zu erstellen und zu importieren. Instanzen dieser Images heißen zur Laufzeit Container.
- Diese Container-Dateien sind als einzelne Daten auf der Wirt-Festplatte abgelegt. Die Container-Festplatte ist über ein Overlay-Dateisystem (AUFS union file system) zusammengesetzt. Basis ist ein entpacktes Image in schreibgeschützter Form, und in einer darüber gelegten Schicht erhalten Applikationen transparent Lese- und Schreibberechtigungen. Zur Laufzeit verschmelzen die Schichten, um in der chroot-Umgebung aus Sicht des Containers als ein einzelnes Dateisystem zu arbeiten. Zusätzliche Schichten entstehen, wenn aus einem Container ein neues Image entsteht, um dieses auf dem Docker Hub bereitzustellen.

Die Bereitstellung aller Container findet über den Daemon statt:

- Der Daemon läuft mit root-Berechtigungen und verwaltet alle laufenden Container als Kind-Prozesse gemeinsam. Vom Daemon aus finden zur Bereitstellung von Ressourcen und Funktionalität direkte Zugriffe auf den Wirt-Kernel statt.
- Der Daemon-Prozess belegt dynamisch benötigten Wirt-Arbeitsspeicher nach Bedarf.
- Es findet keine Emulation oder Virtualisierung der Hardware statt, sondern der Wirt-Kernel ist direkt für die Container-Prozesse zuständig.
- Docker weist jedem Container per Netzwerk-Brücke eine wirt-interne IP-Adresse zu. Zur Kommunikation nach außen, können Dienste per Port-Weiterleitung an die Wirt-Adresse bereitgestellt werden.

3.1.3 Vergleich zu Laufzeit

In der Praxis lassen sich im direkten Vergleich von Server-Virtualisierung und Betriebssystem-Virtualisierung einige Unterschiede messen. In einer Untersuchung wurde das Verhalten von KVM und Docker zur Laufzeit in Bezug auf Fest-Speicherplatz, Startzeit und Berechnungen unter gleichen Bedingungen gemessen [Seo u. a. 2014].

Zum besseren Vergleich der Ergebnisse, sind die gemessenen Werte zusätzlich für diese Masterthesis erweitert und mit einem errechneten Prozentwert in Tabelle 8 versehen, um den Unterschied zueinander deutlich zu machen. Mit KVM sind nur so viele VMs möglich, wie der Wirt-Rechner auch an physikalischem Fest-Speicherplatz bereitstellen kann, um ein sog. Überbuchen auszuschließen. Die Startzeit eines Docker-Containers ist über sechsmal so schnell, was in den unterschiedlichen Konzepten durch den Verzicht auf virtualisierte Gast-Hardware (Hypervisor) und Gast-Abhängigkeiten (Gast-Kernel und Gast-Betriebssystem) begründet ist. Bei der manuellen Berechnung der Fakultät per Skript ist Docker im Vergleich zu KVM nur minimal schneller.

Tabelle 8: Performance-Vergleich KVM und Docker [Seo u. a. 2014] mit Auswertung

	KVM	**Docker**	**Auswertung**
Festspeicher (500 GB)	45 VMs (je 10 GB)	100 Container	+ 122,22 % Docker-Container belegen auf 500 GB weniger Speicher im Vergleich zu KVM, so dass knapp doppelt so viele VMs möglich sind.
Boot-Zeit (20 Instanzen)	11,48 s	1,53 s	– 86,67 % Docker benötigt bei 20 gestarteten Containern zum Start deutlich weniger Zeit im Durchschnitt.
Berechnungen (100.000!)	4,793 s	4,546 s	– 5,15 % KVM benötigt zur manuellen Berechnung der Fakultät per Skript minimal weniger Zeit.

In einer weiteren Untersuchung wurden die Komponenten CPU, Arbeitsspeicher, Festplatte und Netzwerk ohne Virtualisierung, also im nativen Betrieb, im Vergleich zu Docker und KVM gemessen [Felter u. a. 2014].

Dabei hat sich gezeigt, das bei der CPU-Leistung (mit dem Messprogramm LINPACK) und bei zufälligen Zugriffen auf die Festplatte (IOPS) die Performance im nativen Betrieb mit der Ausführung unter Docker vergleichbar ist. KVM erreichte hingegen jeweils knapp die Hälfte der Performance, was in den zusätzlichen Zwischenschichten von KVM durch Hypervisor und übereinandergelegte Dateisysteme durch Gast- und Wirt-Festplatte begründet ist. Docker überträgt die Daten per AUFS-Overlay direkt auf den Wirt und verwendet dabei das gleiche Dateisystem, wie beispielsweise ext4. Mit KVM dahingehen sind Festplattenänderungen des Gast-Dateisystems zur qcow2-Datei in das Wirt-Dateisystem hin zu übertragen. Bei zufälligen Zugriffen auf den Arbeitsspeicher und TCP-Durchsatz im Netzwerk waren alle Systeme im Versuchsaufbau gleichauf. Hierbei ist die Performance nicht durch Software, sondern durch die Hardware limitiert. Dies macht den großen Vorteil paravirtualisierter Treiber für die virtualisierten Geräte deutlich.

3.2 Systemkonzept

Das Systemkonzept setzt sich aus den für den Betrieb virtueller Umgebungen eingesetzten und gemeinsam geteilten Komponenten zusammen und ist die Grundlage zur Analyse von Bedrohungen und Risiken. Bei der Virtualisierung geht es in erster Linie um die Bereitstellung von gemeinsam nutzbaren Ressourcen, der Kommunikation untereinander und der dazu benötigten Berechtigungen. Die Architektur ist durch die benötigten Komponenten zum Betrieb einer simulierten Rechnerumgebung vorgegeben.

Die Ressourcen der realen Hardware umfassen dabei die virtuellen Geräte zur Kommunikation und Dateisysteme zur Verarbeitung und Persistenz der Daten.

Die interne Kommunikation findet auf dem Wirt-Betriebssystem zwischen einzelnen Prozessen auf der Kernel-Ebene statt (IPC). Für den Datenaustausch werden Informationen aus virtualisierten Geräten mit Hilfe beteiligter Prozesse mit unterschiedlichen Berechtigungen gelesen und geschrieben. Nach außen ist die Kommunikation per Netzwerk zur Bereitstellung von Diensten möglich.

Die Zusammenhänge der vom Wirt bereitgestellter Komponenten aus Hardware-Ressourcen, Kommunikationswegen und Berechtigungen sind in Abbildung 7 zusammengefasst.

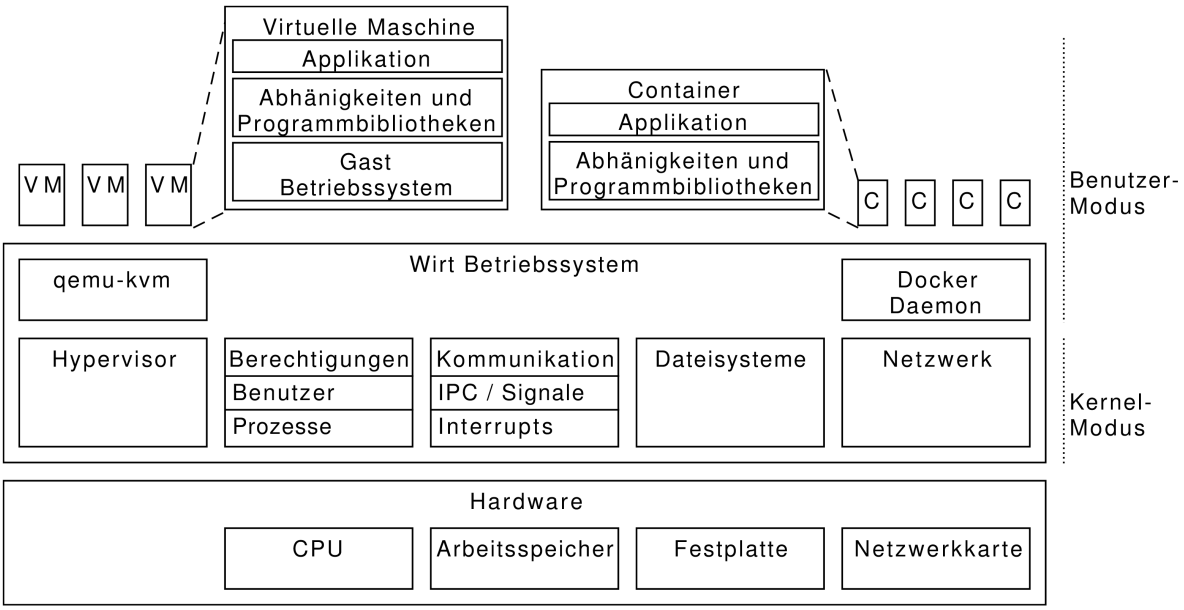

Abbildung 7: Systemkonzept der virtualisierten Komponenten

Im Folgenden wird der Zusammenhang genauer beschrieben.

3.2.1 Ressourcenteilung

„Der Zugriff des virtuellen IT-Systems auf die Ressourcen (Prozessor, Arbeitsspeicher, Massenspeicher, Netz) des Virtualisierungsservers wird durch die Virtualisierungssoftware gesteuert. Dazu erhält jedes virtuelle IT-System virtuelle Geräte, die den Zugriff auf diese Ressourcen erlauben.“ [Bundesamt für Sicherheit in der Informationstechnik B 3.304 2008; S. 91]

Die Virtualisierungssoftware arbeitet als Hypervisor im Kernel-Modus, sowie als qemu-kvm-Prozess im Benutzer-Modus für laufende VMs. Der Docker-Daemon für Container läuft ebenfalls im Benutzer-Modus. Es findet jeweils eine Verarbeitung und geteilte Nutzung der Ressourcen des Wirt-Betriebssystems statt, wobei der Wirt-Kernel in beiden Fällen die Kommunikation mit der realen Hardware übernimmt. Die physikalischen Ressourcen sind so über den Wirt-Rechner mehrfach für unterschiedliche Einsatzzwecke verwendbar. Auf diese Weise findet eine Ressourcenteilung von CPU, Arbeitsspeicher, Festplatte und Netzwerkkarte statt.

Um beispielsweise von einer VM aus auf die virtuelle Festplatte zuzugreifen, verläuft die Kommunikation über den Gast-Kernel, über den Benutzer-Prozess `qemu-kvm` und den Wirt-Kernel, auf das Dateisystem des Wirts, um zur physikalischen Festplatte zu gelangen.

„Die Kapselung der virtuellen IT-Systeme ist […] bei der Betriebssystemvirtualisierung im Vergleich mit der Servervirtualisierung geringer ausgeprägt. In der Folge kann auch die Isolation der virtuellen IT-Systeme untereinander und im Verhältnis zum Virtualisierungsserver weniger stark sein.“ [Bundesamt für Sicherheit in der Informationstechnik B 3.304 2008; S. 93]

Zur Beschränkung der Zugriffe und zur Isolation der geteilten Ressourcen sind unter Linux die `Namespaces` einsetzbar, und die Begrenzung geteilten Ressourcen ist mit `cgroups` möglich.

3.2.2 Kommunikationspfade

Zur Untersuchung der Gefährdungen sind die folgenden Kommunikationspfade relevant:

- „Übergriffe von einem Gastsystem auf den [Wirt]
- Übergriffe vom [Wirt] auf ein Gastsystem
- Angriffe von Außen auf den [Wirt]“ [Bitkom e.V. 2014; S. 7; Teil 3]

Per Netzwerk sind Zugriffe von außen auf den Wirt möglich, sowie direkt auf ein Gastsystem, wenn eine VM oder ein Container nach außen einen Dienst bereitstellt. Dabei kann der Zugriff sowohl aus dem öffentlichen WAN also auch über das interne LAN des Unternehmens erfolgen.

Ein Datenfluss findet zwischen Wirt- und Gast-System, sowie in virtuellen Umgebungen zwischen laufenden Prozessen untereinander per IPC, Signale und Interrupts über die CPU statt.

Die Trennung virtueller Netzwerke und IPC ist mit Linux per `Namespaces` möglich. Die Isolierung von Kernelfunktionalität durch Prozesse ist durch den Linux-Kernel selbst, sowie über `seccomp` abgeschirmt.

3.2.3 Berechtigungen

„Die Gastwerkzeuge werden wegen ihrer systemnahen Funktion häufig mit sehr hohen Berechtigungen ausgeführt. Häufig laufen sie im Kontext und damit mit den Rechten des Betriebssystemkerns der virtuellen Maschine.“ [Bundesamt für Sicherheit in der Informationstechnik B 3.304 2008; S. 52]

Auf der Ebene des Wirt-Betriebssystems unterteilen sich die Berechtigungen in die zum Betrieb der VM notwendigen Prozess-Rechte und Benutzer-Rechte der Anwender.

„Eine Aufteilung der Administratorrolle ermöglicht die gegenseitige Kontrolle der unterschiedlichen Administratorengruppen in einem arbeitsteiligen Rechenzentrumsbetrieb.“ [Bundesamt für Sicherheit in der Informationstechnik B 3.304 2008; S. 79]

Somit unterteilen sich die Berechtigungen der Anwender zusätzlich in die notwendigen Zugriffsrechte der Personen zur Administration der Infrastruktur und Wartung der Server. Durch Gruppen sind Benutzer ihrem Aufgabenbereich zugeordnet, statt generelle root-Rechte zu erlauben.

Die Trennung von Benutzern und Gruppen ist Kernfunktion von Linux. Weitere Sicherheitsaspekte bietet der Linux-Kernel mit den `setuid`- und `setgid`-Bits, sowie mit `capabilities` zur gezielten Rechtevergabe und `chroot` zur Bereitstellung einer Sandbox.

3.2.4 Eingesetzte Technologien

Basis der Virtualisierung zur Bereitstellung der Funktionalität ist ein Wirt-Betriebssystem. Dies wird von einem lokalen Systembetreuer unter dem Gesichtspunkt der Sicherheit eingerichtet, betrieben, und gewartet. Der Ursprung und die Administration fremder Gast-Betriebssysteme sind in der anschließenden Bedrohungsanalyse genauer zu beleuchten.

Zur Kommunikation der Gast-Systeme untereinander setzen die Server-Virtualisierung und die Betriebssystem-Virtualisierung standardmäßig ein IPv4-Netzwerk ein. Zur Einbindung in das Firmennetzwerk können mit KVM gestartete Systeme von einem externen DHCP-Server automatisch benötigte Konfigurationen wie IP-Adresse, DNS-Server und Gateway erhalten. Docker vergibt diese Informationen vom Docker-Daemon über virtuelle Schnittstellen auf dem Wirt. Virtuelle Systeme sind anschließend gleichberechtigt in das jeweilige Netzwerk eingebunden.

Abhängig von der Architektur existieren, zur internen Kommunikation zwischen der Virtualisierungssoftware und dem virtuellem System, verschiedene Protokolle und Schnittstellen. Die Kommunikation intern zwischen den Diensten zur Bereitstellung der Virtualisierung, wie bei Docker über einen Linux-Socket oder bei KVM über die Linux-Gerätedatei, ist von extern erreichbaren Schnittstellen wie RFB per VNC zu unterscheiden.

4 Bedrohungsanalyse

Im Folgenden werden die bisherigen Erkenntnisse aus den Grundlagen von Linux-Kernel und der Virtualisierung mit dem Systemkonzept verknüpft. Das Ziel ist es die einzelnen Komponenten zu analysieren und die dabei bestehende Bedrohung auf die Systemsicherheit zu ermitteln. Dies dient als Basis zur Bestimmung und Abwehr des Risikos in den weiteren Kapiteln.

Eine im Unternehmen bestehende Regelung zur Datensicherheit kann durch übergreifende Berechtigungen auf Virtualisierung und Linux Kernel-Funktionalität ausgehebelt werden. Mit der Untersuchung der verwendeten Virtualisierungsarchitektur und der eingesetzten Komponenten und Technologien sind Schwachstellen ermittelbar. Daraus ergeben sich aktuell bestehende Bedrohungen, deren Auswirkung einzuordnen und zu bewerten ist. Zusätzlich sind aktiv bisher unbekannte Bedrohungen erkennbar, um frühzeitig einem potentiellen Schaden zu verhindern. Das Vorgehen zur Analyse des Systemkonzepts in Bezug auf die Applikations-Absicherung sieht einen Vergleich und eine Auswahl aus verschieden Methoden vor. Anhand der möglichen Bedrohungsarten, den Absichten und Zielen verschiedener Interessengruppen, entsteht die Bedrohungsanalyse. Das Ergebnis ist die Basis zur Risikoabschätzung der individuellen Bedrohungen.

4.1 Methoden zur Einordnung von Bedrohungen und Risiken

Die Aufrechterhaltung der IT-Sicherheit erfordert viel technisches Wissen und gute Kenntnis der eingesetzten Systeme, um Schwachstellen ermitteln zu können. In einer Bedrohungsanalyse sind systematisch mögliche Ursachen und potentielle Bedrohungen zu ermitteln. Ausgehend von den Applikationen, über verschiedene Normen zur Durchführung einer Analyse, bis hin zu einem zertifizierten Unternehmen zum Betrieb einer virtualisierten Umgebung gibt es verschiedene Vorgehensweisen. Im Folgenden werden diese Methoden vorgestellt.

4.1.1 Bewertung sicherer Applikationen

Die Frage, was eine sichere Applikation ist, lässt sich mit Hilfe der vom *OWASP* (Open Web Application Security Project) bereitgestellten Richtlinien und Anleitungen beantworten. Diese Non-Profit-Organisation hat es sich zur Aufgabe gemacht Anleitungen zu Sicherheitsrisiken von Webanwendungen bereitzustellen. So lassen sich beim Entstehen einer neuen Applikation von Beginn an Risiken aufdecken und bei vorhandenen Anwendungen ist die Sicherheit verbesserbar. Als Methoden zur Prüfung und Verbesserung der Sicherheit sind allgemeine Vorgehensweisen, Code Reviews und Bedrohungsmodellierungen empfohlen. Penetration Tests und Filterung der Ein- und Ausgabe sind zur Verhinderung klassischer Angriffe wie XSS (Cross-site scripting) und CSFR (Cross-site request forgery) hilfreich [Gabriel Avramescu u. a. 2013]. Dabei liegt der Fokus auf Webanwendungen und den dabei verwendeten Programmiersprachen. Allerdings ist bei der Vielzahl der möglichen Ansätze und dem Mix aus möglichen Tools keine klare Vorgehensweise und strukturierte Prüfliste erkennbar. Dies erschwert meines Erachtens den Einsatz im Unternehmensumfeld.

Mit den *Common Criteria* sind allgemeine Kriterien für die Bewertung der Sicherheit von Informationstechnologie verfügbar. Der dabei zugrunde liegende Standard DIN ISO/IEC 15408 hat die Aufgabe, ein IT-Produkt auf seine Sicherheit hin zu prüfen und zu bewerten. Die Zertifizierung nach den Common Criteria bedeutet eine Bewertung der Sicherheit in Hinblick auf die Funktionalität und Vertrauenswürdigkeit des IT-Produkts. Die Prüftiefe der Vertrauenswürdigkeit ist numerisch von eins bis sieben angegeben (EAL1-7, Evaluation Assurance Level). Im Jahr 2012 wurde das Betriebssystem Red Hat in der Version 6.2 mit KVM per Common Criteria EAL4+ vom Bundesamt für Sicherheit in der Informationstechnik (BSI) zertifiziert [Bundesamt für Sicherheit in der Informationstechnik BSI-DSZ-CC-0754-2012 2012]. Allerdings stehen die Common Criteria in der Kritik, die Sicherheit nur auf dem Papier zu bewerten, ohne den praktischen Einsatz zu berücksichtigen. Auch ist die Bewertung eines fertigen Produktes im Nachhinein ineffizienter, als wenn schon während der Entwicklung der Software, beispielsweise mit ISO/IEC 15026 (Systems and software engineering – Systems and software assurance), gängige Sicherheitskriterien in die Architektur einfließen würden [Shuichiro Yamamoto/Tomoko Kaneko/Hidehiko Tanaka 2013; S. 331].

4.1.2 Vorgehen und Nomen zur Analyse

Mit einer *Bedrohungsmatrix* oder *Bedrohungslandkarte* werden durch Abstraktion Bedrohungen in Gefährdungsbereiche eingeteilt. In diesem Zusammenhang sind die Zeilen der Matrix dabei in die folgenden generischen Bedrohungen einzuteilen:

- Ausfall oder Einschränkung von Ressourcen, z. B. durch Softwarefehler oder technischen Defekt.
- Bedrohung der Verfügbarkeit durch Blockade, z. B. Überlastung der Infrastruktur durch Denial of Service (DoS-Angriff).
- Bedrohung der Informationsvertraulichkeit von Innen durch Beschädigung oder Verlust von Ressourcen, z. B. durch direkte Angriffe auf ein Speichermedium.
- Bedrohung der Integrität durch Verfälschung oder Manipulation, z. B. von Daten, Information oder Systemen.
- Täuschung durch Nutzung einer falschen Identität, z. B. durch Missbrauch erteilter Berechtigungen.
- Bedrohungen von Extern durch unzulässige Informationsgewinnung und unzulässige Nutzung der Unternehmensinfrastruktur, z. B. durch Entwendung von vertraulichen oder geheimen Daten.

Den Spalten sind potentiellen Auslösern zugeordnet, dies sind Personen wie Personal, Administratoren und externe Dienstleister, Objekte der Gebäude- und IT-Infrastruktur. In den Schnittpunkten der generischen Bedrohungen und Auslösern, sind konkrete mögliche Szenarien eines Angriffs angesiedelt, wie das Knacken eines Passworts oder direkter Speicherzugriff. Anhand der Angriffsszenarien lassen sich alle möglichen Gefährdungsbereiche und deren Auswirkungen auflisten. Dazu ist jeweils die Eintrittswahrscheinlichkeit ermittelbar, um das Bedrohungspotenzial zu erkennen.

Allerdings kann diese Art der Bedrohungsanalyse sehr umfangreich werden, ohne aber die Aspekte der Verkettung von Szenarien zur Erreichung eines Zwischenziels bei einem Angriff zu berücksichtigen. Der Versuch, alle möglichen Bedrohungen abzubilden, macht die Analyse mit der Bedrohungsmatrix komplex und unübersichtlich.

In dieser oder einer zusätzlichen *Risikomatrix* nach EN 60601 (Sicherheitsanforderungen und ergonomische Forderungen an medizinische elektrische Geräte und in medizinischen Systemen) lässt sich die Eintrittswahrscheinlichkeit dem Schadensausmaß tabellarisch gegenüber stellen, so dass anhand der Schnittpunkte die Risikostärke ablesbar ist. Mit Hilfe von Ampelfarben ist visualisiert, um nicht akzeptierte von tolerierten und akzeptierten Risiken zu unterscheiden.

Es wird empfohlen die Eintrittswahrscheinlichkeiten aus Gründen der Effizienz und der Komplexitätsreduktion in Klassen einzuteilen. So lassen sich Bedrohungsgruppen zusammenzufassen für welche die gleichen Bedrohungen gelten [Müller 2014; S. 217].

Der *Bedrohungsbaum* ist eine weitere Methode zur Bedrohungsanalyse. Die Technik ist dabei angelehnt an die Funktionsweise der Fehlerbäume zur Untersuchung der Zuverlässigkeit von Systemen nach der Norm DIN 25424 (Fehlerbaumanalyse). Statt generische Bedrohungen potentiellen Auslösern gegenüberzustellen, hat der Bedrohungsbaum die Analyse konkreter möglicher Szenarien im Fokus. Die Wurzel des Baums stellt ein mögliches Angriffsziel dar. Die damit verbundene Bedrohung auf das System ergibt sich aus den zu überwindenden Hürden.

Ein Angreifer muss zur Erreichung des Angriffsziels verschiedene Stationen durchlaufen, die jeweils selbst ein Angriffsziel darstellen. Die einzelnen Angriffsziele sind per UND- bzw. ODER-Konten miteinander verbunden. Bei einem UND-Knoten müssen alle Angriffsziele gemeinsam erreicht werden, um die Sicherheit eines Elternknoten zu bedrohen, während bei einem ODER-Knoten der Angreifer aus den möglichen Alternativen wählen kann. Die Blätter sind in der Baumdarstellung die einzelnen Angriffsschritte, um über die Äste und die jeweiligen Knoten die möglichen Wege von den einzelnen Blättern bis zur Wurzel des Baumes zu beschreiben.

Mit dem Bedrohungsbaum sind, im Unterschied zur Bedrohungsmatrix, konkrete Szenarien mit unterschiedlichen Angriffspfaden in einer Baumdarstellung grafisch als Angriffskette abbildbar und berücksichtigen dabei unterschiedliche Varianten zu Erreichung des gleichen Wurzel-Angriffsziels.

Generische mögliche Angriffsziele sind dabei:

- Bedrohung durch Vortäuschen von Identitäten zur Überwindung der Autorisierung.
- Bedrohung der Verfügbarkeit durch Verändern von Software und damit verbundene Umgehung von Schutzmechanismen zur Datenveränderung oder Ressourcenauslastung (DoS).
- Bedrohung der Datenintegrität durch lesenden oder schreibenden Zugriff auf gespeicherte Daten und Informationen.
- Bedrohung der Informationsvertraulichkeit durch Zugriff auf sensible Informationen.
- Bedrohung durch langfristige Beobachtung der Kommunikationspfade.

Die grafische Darstellung wird schnell umfangreich und komplex. In einer textuellen Auflistung sind alternativ die Teilbäume der Angriffspfade aufgelistet, wobei den jeweiligen Knoten UND bzw. ODER vorangestellt ist (vgl. Eckert [2012; S. 206]).

Aus der EU-Datenschutzrichtlinie 95/46/EG und dem Bundesdatenschutzgesetz (BDSG) vom 18. Mai 2001 leitet sich ab, dass ein Unternehmen ein Sicherheitskonzept haben muss. Dies gilt besonders für das Anbieten digitaler Dienstleistungen, die dem Teledienstgesetz (TDG) und dem Teledienstedatenschutzgesetz (TDDSG) unterliegen [Eckert 2012; S. 192]. Eine Möglichkeit dabei ist die Umsetzung der Norm ISO 27001 (IT-Sicherheitsverfahren) als Informationssicherheits-Managementsystem. Zertifizierte Unternehmen können so nachweisen, dass standardisierte Vorgehensweisen zur Erstellung, Betrieb, Überwachung und Wartung der IT-Infrastruktur existieren. Dies schließt eine Bedrohungsanalyse und Umgang mit IT-Risiken ein.

Das *Vorgehen nach BSI* sieht die Zertifizierung nach *ISO 27001 auf der Basis von IT-Grundschutz* vor. Der IT-Grundschutz unterstützt die Identifizierung und Umsetzung von Sicherheitsmaßnahmen im Unternehmen. Die Besonderheit ist dabei der Verzicht auf eine detaillierte Risikoanalyse als Ausgangspunkt, sondern die Etablierung einer Standard-Sicherheit zur Gewährleistung von ausreichendem Schutzes der IT-Systeme. Der deutsche BSI-Standard wird laufend an die international gültige Norm ISO/IEC 27001 angeglichen. Die ISO-Norm gibt dabei Ziele vor *was* sicherzustellen ist, während der BSI-Standard per Soll-Ist-Vergleich auflistet *wie* die Sicherheit in der Praxis realisierbar ist [Bundesamt für Sicherheit in der Informationstechnik 100-1 2009]. Mögliche Risiken des Unternehmens sind beispielsweise der Verlust von Firmengeheimnissen oder Patenten. Die Auswirkung von Sicherheitsvorfällen können Verstöße gegen Gesetze, Beeinträchtigung der Unternehmensdarstellung oder finanzieller Schaden bedeuten. Statt einer exakten Schadenberechnung unterteilt das BSI den Schutzbedarf in Bereiche:

- „normaler Schutzbedarf: Die Schadensauswirkungen sind begrenzt und überschaubar.
- hoher Schutzbedarf: Die Schadensauswirkungen können beträchtlich sein.
- sehr hoher Schutzbedarf: Die Schadensauswirkungen können ein existentiell bedrohliches, katastrophales Ausmaß erreichen.“ [Bundesamt für Sicherheit in der Informationstechnik 100-1 2009; S. 35]

Für den Bereich des normalen Schutzbedarfs empfiehlt das BSI ein allgemeines Vorgehen nach dem „Managementsysteme für Informationssicherheit“ (BSI-Standard 100-1) und der „IT-Grundschutzvorgehensweise“ (BSI-Standard 100-2). Die Vorgehensweise ist in unterschiedliche Bausteine, Gefährdungskataloge und konkrete Maßnahmenkataloge unterteilt. Diese bewirken bei Umsetzung der ausformulierten sog. „Best Practice“ die Verringerung des Risikos. So ist beispielsweise in der ISO 27001 vorgeschrieben, dass es im Unternehmen Sicherheitspraktiken zum Umgang mit Passwörtern geben muss (A.11.3.1 Password use). Erreichbar ist dies über eine BSI-Maßnahme zur Sicherstellung der Mindestlänge, Komplexität und Gültigkeitsdauer mit Ablaufdatum zum regelmäßigen Wechsel der Passwörter (M 2.11 Regelung des Passwortgebrauchs).

Für normalen bis mittleren Schutzbedarf liefert die IT-Grundschutzvorgehensweise bereits ausreichende Schutzmaßnahmen zur Sicherstellung der Standard-Sicherheit im Unternehmen.

Für Systeme die einen sehr hohen Schutzbedarf benötigen, da ein Schaden für das Unternehmen existenzbedrohend wäre, empfiehlt das BSI wegen Komplexität die Durchführung einer Risikoanalyse inklusive einer Bedrohungsanalyse. Zur Vermeidung der Gefahr des Missbrauchs personenbezogener Daten sind keine standardisierten Maßnahmen durchführbar, da die Ursachen und Lösungen individuell für jedes Unternehmen anzupassen sind. In einem solchen Fall ist eine detaillierte „Risikoanalyse auf Basis IT-Grundschutz“ (BSI-Standard 100-3) erforderlich.

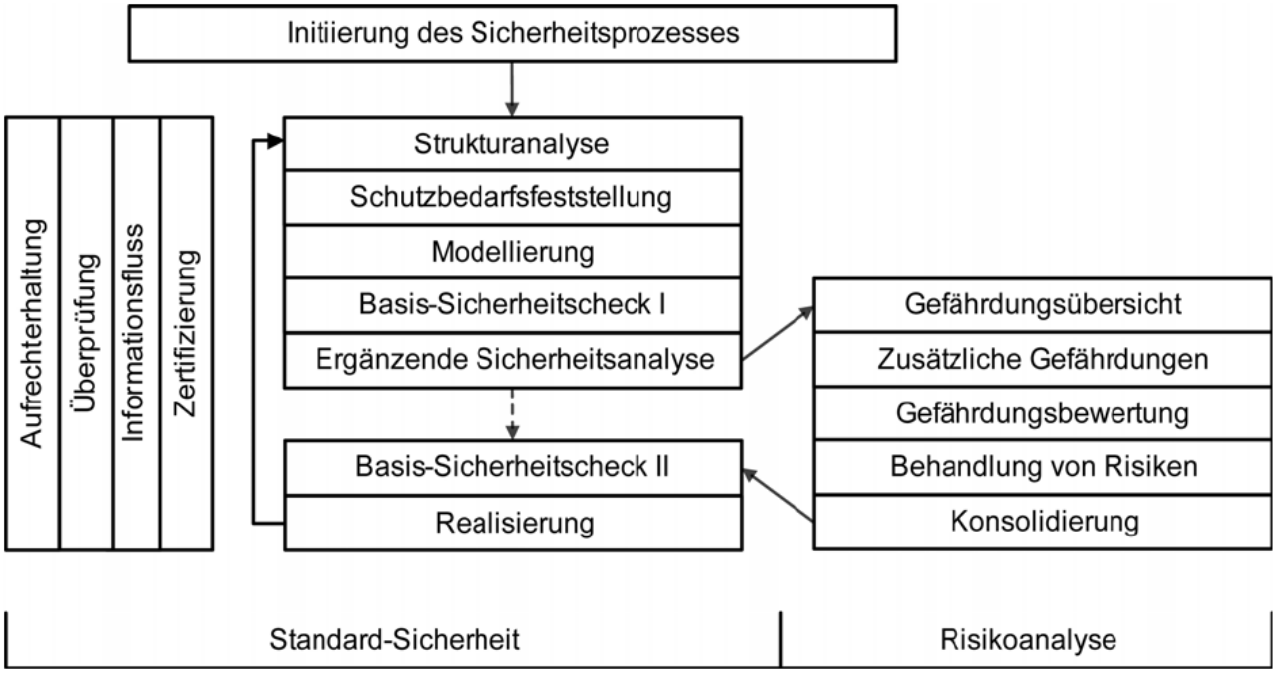

Abbildung 8: Aufbau Risikoanalyse auf Basis IT-Grundschutz (Quelle: Bundesamt für Sicherheit in der Informationstechnik 100-3 [2008; S. 5])

Zur Sicherstellung der Standard-Sicherheit nach BSI (Abbildung 8: linke Seite) ist in einer Strukturanalyse das Risiko zu bestimmen. Der Schutzbedarf ergibt sich aus den zu schützende Werten (beteiligte Systeme und Kommunikationspartner) und den Basis-Schutzzielen (Vertraulichkeit, Integrität und Verfügbarkeit). An dieser Stelle verschafft sich das Unternehmen den Überblick über IT-Landschaft und Software-Architekturen. Das Risiko ist anhand von Eintrittswahrscheinlichkeit und Schadenshöhe ermittelbar, wobei etablierte Modelle wie Bedrohungsmatrix und Bedrohungsbaum bereitstehen. So ist erkennbar, für welche Elemente eine ergänzende Sicherheitsanalyse notwendig ist, so dass für diese eine zusätzliche Risikoanalyse durchzuführen ist.

Ausgangspunkt der detaillierten Risikoanalyse (Abbildung 8: rechte Seite) ist die *Gefährdungsübersicht*, in welcher beteiligte Elemente, also Zielobjekte und Komponenten, mit hohem Schutzbedarf zu bestimmen sind. Die Zielobjekte lassen sich bestehenden Bausteinen des IT-Grundschutz-Kataloges (GSK) zuordnen in die Schichten: übergreifende Aspekte (B 1), Infrastruktur (B 2), IT-Systeme (B 3), Netze (B 4) Anwendungen (B 5). Jedem Baustein sind im GSK konkrete Gefährdungen zugeordnet.

Zusätzliche Gefährdungen, die über das Grundschutz-Modell hinausgehen, sind zur Bestimmung des höheren Schutzbedarfs durch die Beantwortung der folgen Fragen ermittelbar:

- „Von welchen möglichen Ereignissen aus dem Bereich höhere Gewalt droht besondere Gefahr für den Informationsverbund?
- Welche organisatorischen Mängel müssen auf jeden Fall vermieden werden, um die Informationssicherheit zu gewährleisten?

- Welche menschlichen Fehlhandlungen können die Sicherheit der Informationen und Anwendungen besonders beeinträchtigen?
- Welche speziellen Sicherheitsprobleme können beim jeweils betrachteten Zielobjekt durch technisches Versagen entstehen?
- Welche besondere Gefahr droht durch vorsätzliche Angriffe von Außentätern? […]
- Auf welche Weise können Innentäter durch vorsätzliche Handlungen den ordnungsgemäßen und sicheren Betrieb des jeweiligen Zielobjekts beeinträchtigen? […]
- Drohen besondere Gefahren durch Objekte, die nicht dem betrachteten Informationsverbund zuzurechnen sind?“ [Bundesamt für Sicherheit in der Informationstechnik 100-3 2008; S. 12-13; Hervorhebungen getilgt]

Im nächsten Schritt, der *Gefährdungsbewertung*, wird für jedes Zielobjekt geprüft, ob die vorgesehenen Sicherheitsmaßnamen im Systemkonzept einen ausreichenden Schutz bieten. Dazu sind die Kriterien Vollständigkeit, Mechanismenstärke und Zuverlässigkeit zu prüfen und zu bewerten. An dieser Stelle verzichtet das BSI auf eine Berechnung der Eintrittswahrscheinlichkeit von Schadensereignissen, da „die Abschätzung dieser Wahrscheinlichkeiten in der Praxis oft schwierig [und deren] Interpretation […] in vielen Fällen fraglich [ist]“ [Bundesamt für Sicherheit in der Informationstechnik 100-3 2008; S. 5]. Abzulesen ist, ob bereits ein ausreichender Schutz für eine ermittelte Gefährdung besteht, oder ob ein bestehendes Risiko noch behandelt werden muss.

Die *Behandlung von Risiken* umfasst Maßnahmen zur Abwendung der ermittelten Gefährdungen, die nicht durch den GSK abgedeckt sind. Es ist zu entscheiden wie mit dem Risiko umzugehen ist, ob entweder ein erhebliches Risiko oder die entstehenden Kosten zur Behebung der Gefährdungen akzeptabel sind. Zur Beseitigung der ermittelten Risiken sind weitere Sicherheitsmaßnamen zur Risiko-Reduktion, Umstrukturierung zur Risiko-Vermeidung, sowie die Übernahme oder Transfer des Risikos zur Auslagerung möglich. Zusätzlich ist auch die aktive Beobachtung eines Risikos denkbar, wenn es derzeit akzeptabel ist, aber zukünftig noch Handlungsbedarf bestehen könnte.

Im letzten Schritt der *Konsolidierung* sind neue hinzugekommene Sicherheitsmaßnamen hinsichtlich Auswirkungen und Integration in das bestehende Sicherheitskonzept zu prüfen auf Eignung, Zusammenwirken, Benutzerfreundlichkeit und Angemessenheit der Sicherheitsmaßnamen. Anschließend wird das Ergebnis der Risikoanalyse in die Standard-Sicherheit überführt, um dort mit dem erneuten Basis-Sicherheitscheck fortzufahren.

Zusammenfassend bedeutet die Zertifizierung nach *ISO 27001 auf der Basis von IT-Grundschutz* für Unternehmen die Identifizierung von Gefährdungen und Umsetzung beschriebener „Best Practices“ als Maßnahmen, um Standard-Sicherheit nach BSI im Unternehmen zu etablieren. Unternehmen, die digitale Dienstleistungen anbieten, haben bereits in einem Systemkonzept die IT-Landschaft und Software-Architektur analysiert, um ein verpflichtendes Sicherheitskonzept umzusetzen. Die Vorgehensweise nach BSI zur Identifizierung weiterer zusätzlicher Gefährdungen eignet sich gut, wenn bereits ein bestehendes Sicherheitskonzept vorhanden ist. Auch wenn ein Unternehmen die Zertifizierung nach ISO 27001 nicht anstrebt, so sind die „Best Practices“ und die bereitgestellten Punkte in Form der „Goldenen Regeln“ des IT-Grundschutz Bausteins „B 3.304 Virtualisierung“ ein guter Ausgangspunkt für Sicherheit in virtuellen Umgebungen.

4.2 Erhöhter Schutzbedarf

Üblicherweise folgt die Umsetzungsreihenfolge der Maßnahmen des BSI dem Lebenszyklus im Unternehmen mit den Phasen Planung und Konzeption, Beschaffung, Umsetzung und Betrieb.

Es ist davon auszugehen, dass im Unternehmen formale Regelungen zur IT-Sicherheit bereits konzipiert und etabliert sind. Ein bestehendes Sicherheitskonzept nach IT-Grundschutz, umgesetzte Maßnahmen aus „Best Practices“, oder die „Goldenen Regeln“ des Bausteins der Virtualisierung, werden hier vorausgesetzt. Das Konzept ist bereits geplant und wird befolgt, beispielsweise durch die Etablierung von getrennten Berechtigungsgruppen für unterschiedliche Administrationsaufgaben (ISO 27001: A.10.1.3 Segregation of duties).

Weitere Vorbedingung sind konzeptionelle Rahmenbedingungen, wie die Aufteilung der Gruppen-Rechte von Benutzern, die Absicherung und ein planmäßiges Vorgehen bei einem Ausfall der IT-Infrastruktur. Bei der Beschaffung hat die Auswahl der Software unter dem Gesichtspunkt der Bewertung der Sicherheit, eine erhöhte Bedeutung. Die Umsetzung umfasst den Infrastruktur-Aufbau und der Betrieb beinhaltet die Ausführung von Applikationen in einer virtuellen Umgebung.

Ausgangssituation ist die Sensibilisierung des Unternehmens, dass zum Betrieb von Applikationen im virtualisierten Umfeld ein sehr hoher Schutzbedarf nach BSI besteht. Die Verarbeitung wichtiger Unternehmensdaten kann bei Sicherheitsverletzungen einen existenzbedrohenden finanziellen Schaden für das Unternehmen bedeuten. Nach der Einteilung der Bedrohungen und der Festlegung der zu untersuchenden Gefährdungskataloge legt das Unternehmen die zu erreichenden Schutzziele fest. Auf Basis der zuvor getroffenen Annahme existieren bereits etablierte Maßnahmen im Unternehmen für die Planung und Konzeption mit Hilfe des IT-Grundschutzes. Aus diesem Grund findet keine Betrachtung der folgenden Bedrohungen aus dem Gefährdungskatalog statt:

- Höhere Gewalt z. B. Feuer, Wasser oder Personalausfall, beispielsweise durch Streik.
- Organisatorische Mängel z. B. ungeregelte Vorgehensweisen bei Entwicklung, Dokumentation, Test und Freigabe.
- Menschliche Fehlhandlungen z. B. durch die Missachtung von Sicherheitsvorschriften, unachtsame Zerstörung von Daten und Informationen.

Die verbleibenden zu analysierenden Bedrohungen im Betrieb der virtuellen Umgebung sind:

- Technisches Versagen z. B. durch Softwarefehler oder Ausfälle.
- Vorsätzliche Handlungen z. B. Angriffe wie Datenzerstörung oder Manipulation.

Für diese Bedrohungen ist der erhöhte Schutzbedarf zu analysieren. Anstelle der vom BSI vorgesehen Schadenseinstufung der Basis-Schutzziele (Vertraulichkeit, Integrität und Verfügbarkeit), wird als Vorgehensweise die STRIDE-Methode [Microsoft Corporation 2005] angewandt. Tabelle 9 zeigt die Bedrohungen und für diese Masterthesis erweitert, das jeweils verletzte Schutzziel. Durch die Verkettung verschiedener Aspekte als Zwischenziel sind so Varianten an Szenarien abbildbar.

Tabelle 9: Bedrohungen nach STRIDE-Modell mit verletzten Schutzzielen

Bedrohung	Verletztes Schutzziel
Vortäuschen einer Identität (Spoofing)	Authentifizierung des Benutzers (Authentication)
Manipulation von Daten (Tampering)	Bewahrung der Integrität (Integrity)
Leugnung einer Handlung (Repudiation)	Nichtabstreitbarkeit von Vorgängen (Non-repudiation)
Offenlegen von Informationen (Information Disclosure)	Sicherstellung der Vertraulichkeit (Confidentiality)
DoS-Angriffe (Denial of Service)	Bereithalten der Verfügbarkeit (Availability)
Ausweitung von Berechtigungen (Elevation of privilege)	Autorisierung und Genehmigung von Rechten (Authorization)

4.3 Interessengruppen und Zielobjekte

Bei Betrieb virtueller Maschinen ist zu beachten, dass der Schutzbedarf des Virtualisierungs-Wirts beeinflusst wird. Probleme eines Gast-Systems können sich zusätzlich möglicherweise auch auf andere virtuelle Systeme, die auf dem gleichen Wirt betrieben werden, auswirken (vgl. Bundesamt für Sicherheit in der Informationstechnik B 3.304 [2008; S. 1]).

Im Folgenden sind mögliche Szenarien dargestellt, um daraus abzuleiten, welche zu schützenden Werte im Unternehmen im Interesse des Angreifers stehen. Diese Schadensszenarien sind die Basis der Bedrohungsanalyse und zeigen mögliche Absichten der angreifenden Interessengruppen.

Tabelle 10: Schadensszenario 1: Auslasten von Wirt durch Gast

Ausgangssituation	• Im Gast-System sind wahlfrei Programme ausführbar. • Dies erlaubt Fuzz-Testing[9] innerhalb des Gast-Systems.
Reproduzierbarkeit	• Durch vorsätzliche Handlungen werden viele zufällige Kernel-Aufrufe mit zufälligen Werten und Rücksprungadressen ausgeführt. • Gezielt innerhalb eines kurzen Zeitrahmens kann das vom Linux-Kernel kontrollierte Fork-Limit aus ulimit erreicht werden (vgl. Abschnitt 2.4), was neue Prozesse an der Ausführung hindert und auch bestehende Kind-Prozesse vom Docker Daemon beendet.
Auswirkungen und mögliche Folgen	• Beenden von Prozessen innerhalb eigener oder fremder Gast-Systeme, durch gleiche Prozess-Abstammung aus Sicht von ulimit. • Absturz des Gast-Systems oder fremder Gast-Systeme. • Einfrieren des Wirt-Systems durch Absturz von KVM-QEMU.
Verletzte Schutzziele	• Durch nichtdeterministisches Verhalten des Fuzz-Test, durch die zufälligen Aufrufe, wird die Ursache verschleiert (Non-repudiation). • Mehr Ressourcen als zugeordnet erhalten (Availability).

Tabelle 11: Schadensszenario 2: Ausbruch aus Gast-Container

Ausgangssituation	• Gast teilt sich mit dem Wirt einen gemeinsamen Ordner. • Dieses Verfahren findet Einsatz, um beispielsweise mit einer gemeinsame Datenbank zu arbeiten, oder um Rechenaufgaben auf mehrere Gast-Systeme zu verteilen und die erhaltenen Ergebnisse zentral abzulegen.
Reproduzierbarkeit	• Durch technisches Versagen unter Ausnutzung eines Softwarefehlers, oder • Durch Wirt-Administrator, welcher den gemeinsamen Ordner verschiebt wodurch sich die Datenstruktur (inode) ändert und neu (einem anderen parent) zugeordnet wird. • Der Gast kann sich entlang des verschobenen gemeinsamen Ordners hin bis zur Dateiebene des Wirt-Systems vorarbeiten.
Auswirkungen und mögliche Folgen	• Ausbrechen aus Gast-Beschränkung. • Vollzugriff auf fremde Gast-Systeme erhalten. • Vollzugriff auf Wirt erhalten.
Verletzte Schutzziele	• Umgehung von Schutzmechanismen (Authentication). • Manipulation von Daten (Integrity). • Verschleierung des Angriffs durch Veränderung der Konfiguration (Non-repudiation). • Vollzugriff auf fremde Daten (Confidentiality). • Ausweiten von Berechtigungen (Authorization).

Die Motivation der Interessengruppen ist der Zugriff oder die Beeinflussung der zu schützenden Werte des Unternehmens.

Zu schützende Werte:

- Persistente Daten im Gast (Festspeicher)
- Flüchtige Daten im Gast (Arbeitsspeicher)
- Persistente Daten im Wirt, wie Konfigurationsdateien und Passwortdateien

9 Black-Box-Test zur automatisieren Erzeugung zufälliger Eingaben, um Sicherheitslücken aufzuspüren.

Die Zielobjekte und Komponenten sind aus diesem Grund vor dem Zugriff der Interessengruppen zu schützen. Die Komponenten ergeben sich der Architektur des Systemkonzepts (Kapitel 3.2).

Zielobjekte:

- Gast-VM (KVM)
- Gast-Container (Docker)
- Wirt
- Kommunikationspfade (vgl. Abschnitt 3.2.2)
  - Gast – Wirt
  - Wirt – Gast
  - Von außen – Wirt

Die in den Schadensszenarien aufgelisteten Aktionen schwächen die Schutzziele, indem zu schützenden Werte und Zielobjekte über die Kommunikationspfade attackiert werden. Ein erhöhter Schutzbedarf ist notwendig zur Vermeidung der möglichen Folgen durch Sicherheitsverletzungen im virtualisierten Umfeld. Dies ist der Ausgangspunkt der Bedrohungsanalyse nach BSI.

4.4 Anwendung der Bedrohungsanalyse

Um von einem Angriff sprechen zu können, welcher auf zu schützenden Werte abzielt, muss der Angreifer ein Schutzziel verletzen, um Zugriff auf die Zielobjekte zu erhalten. Aus den verletzten Schutzzielen ergeben sich so konkrete Bedrohungen. Im anschließenden Kapitel 5 wird evaluiert, welche individuellen Maßnahmen für höher schutzbedürftige virtuelle Systeme notwendig sind.

Die Bedrohungsanalyse nach dem Vorgehen nach BSI „Best Practice“ mit vorhandenen Bausteinen, setzt ein bestehendes Sicherheitskonzept voraus mit dem Bewusstsein für den erhöhten Schutzbedarf. Die Bausteine des BSI-Standards sind in fünf Schichten unterteilt: 1. Übergreifende Aspekte, 2. Infrastruktur, 3. IT-Systeme, 4. Netze und 5. Anwendungen.

Der Baustein B 3.304 Virtualisierung ist der dritten Schicht zugeordnet, in welcher Administratoren und Benutzer für Server- und Client-Systeme zuständig sind. Es stehen konkrete vorgefertigte Gefährdungen und Maßnahmen bereit für die vorbereitenden Zyklen der Planung und Konzeption, Beschaffung und Umsetzung zum Aufbau der virtuellen Infrastruktur. Im Fokus der Analyse liegt der Bereich zum Betrieb virtueller Maschinen mit den möglichen Bedrohungen durch *technisches Versagen* und *vorsätzliche Handlungen*.

Mögliche Gefährdungen durch technisches Versagen virtueller Umgebungen benennt das BSI in den Kategorien Ausfall und Störungen, sowie durch Ressourcen-Engpässe. Vorsätzliche Handlungen sind denkbar durch unberechtigtes Kopieren, unautorisierte Zugriffe, missbräuchliche Nutzung und Kompromittierung der virtualisierten Infrastruktur [Bundesamt für Sicherheit in der Informationstechnik B 3.304 2008; S. 2].

Inhaltlich finden im *Baustein Virtualisierung* beide Virtualisierungs-Varianten Beachtung. Zum einen die Server-Virtualisierung, von welcher das Wirt-System über den Wirt-Kernel und den Hypervisor als Zwischenschicht VMs bereitstellt. Innerhalb einer VM ist ein Gast-System mit eigenem Gast-Kernel aktiv. Zum anderen die Betriebssystem-Virtualisierung mit der gemeinsamen Nutzung des Wirt-Kernels für Gast-Systeme.

Für die Lebenszyklusphase *Betrieb* sind dabei ausschließlich generische Maßnahmen beschrieben, ohne auf technische Details einzugehen. Für die Infrastruktur empfiehlt das BSI die Überwachung der Ressourcen (M 2.448) und rät zur minimalen Nutzung der Konsole (M 2.449). Für die Netz-Schicht erfolgt außerdem ein Hinweis auf die Wichtigkeit der Zeitsynchronisation (M 4.348), um stets eine korrekte Zeitangabe für die Protokollierung virtueller Systeme zu erhalten. Der Abschnitt zum sicheren Betrieb (M 4.349) verweist auf die allgemeine Administration von Hardware-Servern [Bundesamt für Sicherheit in der Informationstechnik B 3.304 2008; S. 82-84 u. 112-126].

Für den sicheren Betrieb von Anwendungen in virtuellen Umgebungen gibt es im Baustein B 3.304 keine Maßnahmen des BSI. Das Ziel der Bedrohungsanalyse ist somit die Abgabe einer Empfehlung als Basis für Sicherheitsentscheidungen im Unternehmensumfeld zum sicheren Betrieb existierender Applikationen in virtualisierten Umgebungen.

Für die zu betrachtenden Virtualisierungs-Implementierungen ist für KVM durch die Zertifizierung nach Common Criteria davon auszugehen, dass diese Server-Virtualisierung bereits als hinreichend sicher eingestuft ist. Die kürzere Verarbeitungskette von Docker macht aber eine genaue Analyse der Betriebssystem-Virtualisierung notwendig. Zu den beiden zuvor genannten Schadensszenarien, also dem Ausbruch aus einem Gast-Container und der Überlastung des Wirt-Systems (Kapitel 4.3), liegt somit der Fokus in der Analyse von Docker. Aber auch KVM wird in Teilbereichen genauer betrachtet, um gegenseitige Beeinflussungen zu verhindern.

In den folgenden Tabellen wird für jedes Zielobjekt, sowie für die Kommunikationspfade untereinander jeweils untersucht, welche Gefährdungen hinsichtlich der Schutzziele bestehen. Mit Hilfe der STRIDE-Methode wird analysiert, was genau potentielle Bedrohungen sind und welche Voraussetzungen bestehen müssten, um die zu schützende Werte im virtualisierten Betrieb zu gefährden.

Tabelle 12: Bedrohungsübersicht Gast-VM

Gast-VM	
B1.1 Denial of Service	Belegung der Festplatte Die anwachsende „thin-provisioned“ qcow2-Datei gibt beim Löschen von Gast-Daten den Wirt-Speicher nicht wieder frei. Bei einer Konfiguration, die ein „Überbuchen“ erlaubt, ist zugesicherter vermeintlich freier Speicherplatz nicht nutzbar. Dies sorgt für ein Pausieren aller VMs, die schreiben müssen.
B1.2 Information disclosure	Auslesens des Arbeitsspeichers einer pausierten VM Wenn die VM pausiert und alle Speicherinhalte persistiert sind, können sensible Daten aus dem virtuellen Arbeitsspeicher in Ruhe ausgelesen werden (z. B. zur Laufzeit eingegebene Klartextpasswörter).
B1.3 Repudiation	Änderung der IP-Adresse durch den Benutzer Durch manuelles Ändern der Netzwerkeinstellungen durch den Benutzer der VM, können IP-Adressen doppelt vergeben werden. Bei festen Port- und IP-Weiterleitungen des Wirt-Systems erreicht die Kommunikation möglicherweise die falsche Gast-VM.

Tabelle 13: Bedrohungsübersicht Gast-Container

Gast-Container	
B2.1 Elevation of privilege	Verlassen der Gast-Beschränkung Angreifer kann durch eine mögliche Softwarelücke (Schadensszenario 2) weitreichenden Zugriff erhalten und Änderungen an eigenen oder anderen Zielobjekten vornehmen.
B2.2 Information disclosure	Zugriff auf Daten eines anderen Gast-Systems Angreifer erhält Berechtigung, um an Informationen aus einem anderen Gast gelangen. (setzt B2.1 Elevation of privilege voraus)

B2.3 Information disclosure	Zugriff auf Daten des Wirt-Systems Angreifer erhält Berechtigung, um auf Daten des Wirts zuzugreifen, wie beispielsweise zum Auslesen der Wirt-Passwortdatei. (setzt B2.1 Elevation of privilege voraus)
B2.4 Repudiation	Änderung der IP-Adresse durch Benutzer/Docker Durch Docker erhält der Container bei jedem Neustart eine neue IP-Adresse zugewiesen. Bei festen Port- und IP-Weiterleitungen des Wirt-Systems erreicht die Kommunikation möglicherweise den falschen Gast-Container.
B2.5 Spoofing	Unbefugter Zugriff auf Applikationsschnittstellen Durch unzureichende Authentifizierung, bei von extern vorbereiteten Images und den darin enthaltenden Applikationen, über vordefinierte Standardkennwörter.
B2.6 Tampering	Ausführen veränderter Container Durch unzureichende Überprüfung des Original-Zustandes des heruntergeladenen Images vom Docker Hub, oder aus externer Quelle durch lokal selbst erzeugte Images.
B2.7 Denial of Service	Belegung von Ressourcen Durch übermäßige Ressourcenauslastung einer Applikation des Gast-Systems, so dass andere Applikationen innerhalb des gleichen Containers am Betrieb gehindert werden.

Tabelle 14: Bedrohungsübersicht Wirt

Wirt	
B3.1 Spoofing	Unbefugter Zugriff auf Virtualisierungsschnittellen Durch unzureichende Authentifizierung erhält der Angreifer Vollzugriff auf den Wirt und kann so die Konfiguration manipulieren.
B3.2 Denial of Service	Vollständige Belegung der Festplatte Durch Gast-VM oder Gast-Container, so dass kein freier Speicher mehr verbleibt, was die Virtualisierungssoftware am Betrieb hindert, wenn das Schreiben von Protokoll-Dateien unmöglich ist.

Tabelle 15: Bedrohungsübersicht der Kommunikation Gast – Wirt

Kommunikation Gast – Wirt	
B4.1 Denial of Service	Gast überlastet Wirt Angreifer bewirkt eine hohe Auslastung der Systemressourcen des Gast-Containers, was sich auf dem Wirt auswirkt.
B4.2 Tampering	Veränderung der Kernel-Aufrufe Durch unzureichende Abschottung gegen Fuzz-Testing (Schadensszenario 1) greift der gemeinsame Wirt-Kernel auf zufällige Rücksprung-Adressen zu.

B4.3 Elevation of privilege	Zugriff auf Administrationsschnittellen Durch unzureichende Trennung des Gast-Netzwerks zum Wirt, kann der Angreifer Einsicht in Daten erhalten, beispielsweise durch Brute-Force-Angriff oder Denial of Service.

Tabelle 16: Bedrohungsübersicht der Kommunikation Wirt – Gast

Kommunikation Wirt – Gast	
B5.1 Information disclosure	Anzeige der Gast-Container-Prozesse Durch unzureichende Abschottung werden sämtliche Gast-Prozesse als Kind-Prozesse von Docker-Daemon in der Wirt-Prozessliste angezeigt.
B5.2 Tampering	Änderung der Gast-Daten von Wirt aus Durch Administrator sind Inhalte von Gästen von extern veränderbar.

Tabelle 17: Bedrohungsübersicht der Kommunikation von außen – Wirt

Kommunikation von außen – Wirt	
B6.1 Denial of Service	Überlastung des Wirt-Systems Durch Senden vieler Anfragen auf Schnittstellen, die der Wirt in der Menge nicht verarbeiten kann.
B6.2 Information disclosure	Anzeige von Gast Diensten / Schnittstellen Durch Weiterleiten von Ports können Angreifer auf Gast-Inhalte zugreifen.
B6.3 Elevation of privilege	Angriff auf Administrationsschnittellen Durch Brute-Force-Angriff zur Erlangung von Zugriffsrechten.

4.5 Analyse der eingesetzten Technologien

Die zuvor analysierten Gefährdungen beziehen sich auf zu schützende Werte, die Zielobjekte und deren Kommunikationspfade. Denkbar sind allerdings auch zusätzliche Gefährdungen durch den Einsatz der zugrunde liegenden Technologie. Im Folgenden werden die eingesetzten Technologien aus Kapitel 3.2.4 auf mögliche Schwachstellen hin analysiert. Dies erweitert die vorherige Bedrohungsanalyse um zusätzliche Aspekte zur Erkennung weiterer Bedrohungen, um ein mögliches Schadensausmaß zu minimieren.

Durch den Einsatz der Zwischenschichten aus Hypervisor und Gast-Betriebssystem, ist der Wirt vor Angriffen durch Viren, Trojaner und Root-Kits geschützt [V. Anitha/N. Keerthana 2013]. Zu Docker gibt es bisher nur wenige Untersuchungen. Eine Analyse kommt zu dem Fazit, dass die bereitgestellten Kernel-Funktionalitäten wie Capabilities und Namespaces grundsätzlich zur Sicherheit beitragen [Bui 2014]. Docker setzt die in Kapitel 2.4 beschrieben Capabilities ein, um laufenden Containern gezielt weniger Berechtigungen zur Kommunikation mit dem Linux-Kernel einzuräumen, als außerhalb des Container-Systems verfügbar sind.

Vor Docker Version 1.0.0 wurde unter dem Namen „Shocker“ eine Software-Lücke bekannt, die inzwischen behoben und detailliert analysiert ist [Andre 2014]. So war es möglich vom Container aus die Passwortdatei (`/etc/shaddow`) des Wirts auszulesen. Ursache für diese Sicherheitslücke war eine gesetzte Berechtigung, welche es dem Container per Capabilitiy erlaubte die Umgehung von Leseberechtigungen zu bewirken (`cap_dac_read_search`). Somit war per Kernel-Aufruf über die Inode-Nummer der direkte Zugriff auf Dateien außerhalb der beschränkten Umgebung möglich.

Mit dem Befehl `getpcaps $$` sind die Capabilitiy-Attribute der aktuellen Prozess-ID in der Linux-Shell abrufbar. In einer aktuellen Gegenüberstellung von Docker (Version 1.7.0) sind in Listing 5 die Unterschiede im Vergleich sichtbar. Berechtigungen von außerhalb (im oberen Bereich des Listings), sind innerhalb eines gestarteten Containers (im unteren Bereich) deutlich reduziert. Die hervorgehobenen Berechtigungen für weitreichende Administrationsaufgaben (`cap_net_admin` und `cap_sys_admin`), sowie die Möglichkeit der zuvor genannten Shocker-Sicherheitslücke (`cap_dac_read_search`), stehen innerhalb des Containers nicht bereit.

Listing 5: Capabilities außerhalb und innerhalb eines Containers

```
# getpcaps $$
Capabilities for `8744': =
cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_se
tgid,cap_setuid,cap_setpcap,cap_linux_immutable,cap_net_bind_service,cap_net_broadca
st,cap_net_admin,cap_net_raw,cap_ipc_lock,cap_ipc_owner,cap_sys_module,cap_sys_rawio
,cap_sys_chroot,cap_sys_ptrace,cap_sys_pacct,cap_sys_admin,cap_sys_boot,cap_sys_nice
,cap_sys_resource,cap_sys_time,cap_sys_tty_config,cap_mknod,cap_lease,cap_audit_writ
e,cap_audit_control,cap_setfcap,cap_mac_override,cap_mac_admin,cap_syslog,cap_wake_a
larm,cap_block_suspend,37+ep
# docker run -it ubuntu
root@a0dcacc1c3ab:/# getpcaps $$
Capabilities for `1': =
cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_
setpcap,cap_net_bind_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_write,ca
p_setfcap+eip
```

Allerdings schützt die Verwendung von Capabilities als grundsätzliche Sicherheitslösung nicht vor zukünftigen Sicherheitslücken. Neue und unbekannte Gefahren sind im Rahmen der Bedrohungsanalyse schwer zu bewerten. Meines Erachtens sind die verbleibenden Capabilities innerhalb des Containers zur Kommunikation mit dem Linux-Kernel weniger kritisch. Ob von der Erlaubnis zur Umgehung von Zugriffsberechtigungen (`cap_dac_override`) eine Bedrohung ausgeht soll im Folgenden untersucht und abgeschätzt werden (vgl. Tabelle 4: Berechtigungen mit Capabilities).

Tabelle 18: Ergänzung der Bedrohungsübersicht Gast-Container

Gast-Container	
B2.8 Elevation of privilege	Ausnutzung privilegierter Kernel-Aufrufe Durch die Möglichkeit zur Umgehung von Zugriffsberechtigungen (`cap_dac_override`) besteht die Gefahr von übergreifende Berechtigungen außerhalb der beschränkten Container-Umgebung.
B2.9 Repudiation	Verwendung von fremden Docker-Images Durch Einsatz von ungeprüften Images aus öffentlichem Docker Hub ist die Systemsicherheit gefährdet.

Das Betriebssystem einer VM ist in KVM per virtualisiertem CD-Rom-Laufwerk als Datei im iso-Format eingebunden. Hierbei wird direkt auf den vom Hersteller des Betriebssystems bereitgestellten Funktionsumfang zurückgegriffen. Durch die Herkunft ist die Authentizität und Vertraulichkeit des Betriebssystems durch den Hersteller weitestgehend sichergestellt.

Anders verhält es sich mit Docker-Images. Auf der Plattform Docker Hub darf sich jeder interessierte Benutzer anmelden und eigene Images bereitstellen. Images direkt vom Hersteller sind als „Official Repo“ gekennzeichnet. Für andere Images findet keine Prüfung vor der Bereitstellung statt, was die Sicherheit der IT-Infrastruktur des Unternehmens gefährden kann, aufgrund des unklaren Inhalts. Dies verstärkt die Bedrohung durch die Gefahr bei Nutzung fremder Docker-Images.

Die öffentlich bereitgestellten images im Docker Hub sind beim Herunterladen per HTTPS-Verbindung vor Veränderungen gesichert. Anschließend prüft Docker die Integrität der Datei per sha256-Prüfsumme. Die Gefahr der Modifizierung durch Übertragungsfehler ist so somit ausgeschlossen.

KVM und Docker verwenden standardmäßig ein gemeinsames Netzwerk mit IPv4, jedoch unterscheidet sich die konkrete Implementierung im Detail. KVM verwendet eine virtuelle Netzwerkbrücke zum lokalen bestehenden Netzwerk. An dieser Stelle kann ein externer DHCP-Server im Firmennetzwerk die Einstellungen für IP-Adresse, DNS-Server und Gateway vergeben. Allerdings ist ein fremder DHCP-Server, beispielsweise als Gast aus einer andern VM heraus, in der Lange die Vergabe der Einstellungen zu beeinflussen. Eine sich daraus ergebende Bedrohung ist ein Man-in-the-middle Angriff, wobei der Angreifer die IP-Kommunikation erfolgreich umleitet, um so den Inhalt der jeweiligen Pakete einzusehen und ggf. zu verändern. Docker vergibt auf einer internen virtuellen Netzwerkschnittstelle vom Wirt aus eine private IP-Adresse. Diese IP ist für den Container unveränderbar, sodass dort die Gefahr eines Man-in-the-middle Angriffs nicht besteht. Eine gezielte Verbindung mit dem Firmennetz findet per Port-Weiterleitung statt. Somit ist für KVM eine zusätzliche Bedrohung durch den Einsatz von DHCP vorhanden.

Tabelle 19: Ergänzung der Bedrohungsübersicht der Kommunikation Wirt – Gast

Kommunikation Wirt – Gast	
B5.3 Spoofing	Man-in-the-middle Angriff auf Gast-VM Durch Ausliefern manipulierter DHCP-Pakete.

KVM und Docker binden die Gast-Systeme jeweils in einem gemeinsamen Netzwerk mit anderen Gästen zusammen ein. Durch einen gezielten Netzwerk-Scan sind fremde Gast-Systeme ermittelbar und die jeweils angebotenen Dienste sind auf IP-Basis erreichbar. Durch das Senden von sehr vielen Daten besteht durch die Überlastung des Netzwerkes eine zusätzliche Bedrohung.

Tabelle 20: Ergänzung der Bedrohungsübersicht der Kommunikation Gast – Wirt

Kommunikation Gast – Wirt	
B4.4 Denial of Service	Auslastung von Wirt- und fremden Gast-Systemen Durch falsche Daten oder große Datenmengen in gemeinsam genutztem Netzwerk.

Der Docker Client kommuniziert über den lokalen Linux-Socket (`/var/run/docker.sock`) unverschlüsselt, und auch KVM spricht über mit der Linux-Gerätedatei (`/dev/kvm`) im Klartext. In beiden Fällen ist der Zugriff nur lokal vom Wirt-Rechner aus mit dem root-Benutzer möglich, sowie weiteren Benutzern, die explizit der jeweiligen Gruppe zugeordnet wurden. Hierdurch ergibt sich somit keine weitere direkte Gefährdung.

Der Docker Daemon lässt sich so konfigurieren, dass eine Administration per TCP/IP zur verteilten Bereitstellung von entfernten Docker-Wirt-Rechnern möglich ist. Dabei ist standardmäßig eine unverschlüsselte Kommunikation vorgesehen, jedoch erhält der Administrator einen deutlich sichtbaren Hinweis angezeigt, stattdessen die Verschlüsselung mit Zertifikaten per TLS zu aktivieren. Virtuelle Maschinen von KVM sind extern über RFB Version 3.8 per VNC erreichbar. Hierbei wird TLS oder alternativ ein SSH-Login zur verschlüsselten Kommunikation empfohlen.

Keine der untersuchten Komponenten verwendet SSL Version 3.0 welches im Juni 2015 offiziell als veraltet und unsicher eingestuft wurde mit RFC 7568 [Langley u. a. 2015]. Nach aktueller Recherche ist TLS bei entsprechender Konfiguration der angebotenen Chiffren als sicher anzusehen, sodass sich hieraus keine weitere Bedrohung ergibt.

5 Risikoabschätzung und Maßnahmen

Ermittelte Bedrohungen aus der Bedrohungsanalyse werden im Folgenden ausgewertet, um feststellen inwieweit eine Gefährdung besteht. Für bestehende Risiken werden geeignete Gegenmaßnahmen vorgestellt und ausgewählt. Anschließend werden die festgelegten Maßnahmen in das Systemkonzept integriert und auf ihre Wirksamkeit hin validiert.

5.1 Auswertung der Bedrohungsanalyse

Im dritten Schritt der Risikoanalyse nach BSI findet die Gefährdungsbewertung statt. Die Berechnung der Eintrittswahrscheinlichkeit ist bei dem Vorgehen nach BSI nicht notwendig, stattdessen findet eine Prüfung mit OK-Bewertung statt. So ist abzulesen für welche Bedrohungen, bei Umsetzung der Maßnahmen aus dem Baustein B 3.304 Virtualisierung, dem GSK und entsprechender Vorsichtsmaßnahmen, bereits ein ausreichender Schutz besteht (OK = J). Für darüber hinübergehende Bedrohungen sind weitere Maßnahmen zur stärkeren Absicherung erforderlich (OK = N). Grundlage ist der Fokus auf technisches Versagen und vorsätzliche Handlungen im Betrieb der virtuellen Umgebung. Einige Bedrohungen haben ihren Ursprung in den Schritten der Planung oder Umsetzung, so dass die jeweiligen Phasen zusätzlich zum Betrieb Beachtung finden.

Tabelle 21: Bedrohungsauswertung Gast-VM

Gast-VM		**Schutz**
B1.1 Denial of Service	Belegung der Festplatte Die Bedrohung durch „thin-provisioned“ Speicherplatzvergabe ist während der Umsetzung der virtuellen Infrastruktur nach *M 4.346 Sichere Konfiguration virtueller IT-Systeme* als „Überbuchung“ bereits benannt. In der Betriebs-Phase ist dies zu prüfen.	OK = N
B1.2 Information disclosure	Auslesens des Arbeitsspeichers einer pausierten VM Die Gefährdung ist benannt in *G 5.148 Missbrauch von Virtualisierungsfunktionen* und sieht für den Betrieb als Maßnahme in *M 2.448* die Überwachung vor. Durch zusätzliche Verschlüsselung sensibler Firmendaten innerhalb der Gast-VM, sind die enthaltenen Informationen vor fremdem Zugriff geschützt.	OK = J
B1.3 Repudiation	Änderung der IP-Adresse durch den Benutzer Für die Planung und Konzeption wird mit *M 5.153 Planung des Netzes für virtuelle Infrastrukturen* die Verwendung von VLAN zur Trennung von Netzsegmenten empfohlen. Während dem Betrieb sind weitere Maßnahmen erforderlich.	OK = N

Tabelle 22: Bedrohungsauswertung Gast-Container

Gast-Container		**Schutz**
B2.1 Elevation of privilege	Verlassen der Gast-Beschränkung Softwarelücken sind nach Bekanntmachung durch das zeitnahe Einspielen veröffentlichter Aktualisierungen behebbar.	OK = J
B2.2 Information disclosure	Zugriff auf Daten eines anderen Gast-Systems Durch den gemeinsamen Kernel kann nicht zwischen Gast und Wirt-root-Benutzer unterschieden werden, um ungewollte Zugriffe zu verhindern.	OK = N

B2.3 Information disclosure	Zugriff auf Daten des Wirt-Systems Beim Verlassen von Gast-Beschränkungen ist der Zugriff auf die Wirt-Passwortdatei denkbar. Diese kann kopiert werden, um einen Passwort-Hash in Ruhe zu knacken. Während dem Betrieb sind weitere Maßnahmen erforderlich.	OK = N
B2.4 Repudiation	Änderung der IP-Adresse durch Benutzer/Docker Eine manuelle Änderung der IP-Adresse des Containers ist durch den Benutzer nicht möglich, da die entsprechende Berechtigung (`cap_net_admin`) nicht vorhanden ist.	OK = J
B2.5 Spoofing	Unbefugter Zugriff auf Applikationsschnittstellen Bei Verwendung von Docker-Images sollte der Administrator sicherstellen, dass die verwendeten Erstellungs-Scripte vom Docker Hub keine Standardkennwörter enthalten, oder eventuell vorhandene Kennwörter anschließend ändern.	OK = J
B2.6 Tampering	Ausführen veränderter Container Für heruntergeladene Container besteht keine Gefahr durch Veränderungen während dem Herunterladen, da die sha256-Prüfsumme geprüft wird. Auf das manuelle Herunterladen sollte verzichtet werden, wenn Container über externe Quellen außerhalb vom Docker Hub bereitgestellt werden.	OK = J
B2.7 Denial of Service	Belegung von Ressourcen Zur Vorbeugung übermäßiger Ressourcenbeanspruchung von flüchtigem Speicher ist beim Start eines Docker-Containers über optionale Parameter eine Ressourcenbegrenzung für CPU, Arbeitsspeicher und IPC konfigurierbar. Während dem Betrieb sind weitere Maßnahmen erforderlich.	OK = N
B2.8 Elevation of privilege	Ausnutzung privilegierter Kernel-Aufrufe Nach aktueller Recherche bietet `cap_dac_override` keine Zugriffsrechte zur Ausführung von höher privilegierten Kernel-Aufrufen an [Linux Programmer's Manual 2015].	OK = J
B2.9 Repudiation	Verwendung von fremden Docker-Images Durch Vermeidung von sog. Community-Images aus dem öffentlichen Docker Hub. Ausschließliche Verwendung von Images des ursprünglichen Software-Herstellers.	OK = J

Tabelle 23: Bedrohungsauswertung Wirt

Wirt		**Schutz**
B3.1 Spoofing	Unbefugter Zugriff auf Virtualisierungsschnittellen Im GSK beschreibt das BSI in *M 4.237 Sichere Grundkonfiguration eines IT-Systems*, dass für Server mit hohem Schutzbedarf ein Paketfilter verwendet werden sollte. Eine Konfiguration, die Zugriff und Authentifizierung nur von zuvor definierten Kommunikationspartnern und IP-Adressen erlaubt, ist per Firewall realisierbar.	OK = J

B3.2 Denial of Service	Vollständige Belegung der Festplatte Die Gefahr wird beschrieben in *G 2.149 Nicht ausreichende Speicherkapazität für virtuelle IT-Systeme* und als Maßnahme kommt *M 2.448* zur Überwachung der Ressourcen in Frage. Darüber hinaus weist das BSI für die Umsetzungs-Phase im allgemeinen GSK in *M 4.105 Erste Maßnahmen nach einer Unix-Standardinstallation* darauf hin, dass die Partition `/var` ausreichend dimensioniert sein soll, „damit ein vorsätzliches Produzieren von Protokolldaten das Unix-System nicht zum Stillstand bringt“ [Bundesamt für Sicherheit in der Informationstechnik IT-Grundschutz-Katalog 2015; S. 3213].	OK = J

Tabelle 24: Bedrohungsauswertung der Kommunikation Gast – Wirt

Kommunikation Gast – Wirt		**Schutz**
B4.1 Denial of Service	Gast überlastet Wirt Die Gefahr der Überlastung wird mit Maßnahmen aus *M 2.448* durch die Überwachung der Ressourcen sichergestellt.	OK = J
B4.2 Tampering	Veränderung der Kernel-Aufrufe Durch den gemeinsamen Kernel ist vom Gast-System aus das Wirt-System beeinflussbar, ohne dass ungewollte Zugriffe vom Fuzz-Testing auf Ressourcen des Kernels zu verhindern sind.	OK = N
B4.3 Elevation of privilege	Zugriff auf Administrationsschnittellen Zur Schadensverhinderung durch Brute-Force-Angriffe wird in *M 4.396 Schutz vor unerlaubter automatisierter Nutzung von Webanwendungen* in der Umsetzung beschrieben, dass unautorisierte Nutzung durch Erraten von Passwörtern nicht begünstigt werden soll. Übermäßige Ressourcenbeanspruchung sollen in der Umsetzungs-Phase mit *M 4.405 Verhinderung der Blockade von Ressourcen (DoS) bei Webanwendungen und Web-Services* durch Grenzwerte und Zeitspannen geregelt werden.	OK = J
B4.4 Denial of Service	Auslastung von Wirt- und fremden Gast-Systemen In der internen Kommunikation zwischen Gast und Wirt ist keine Abwehr von DoS und Brute-Force-Angriffen vorgesehen.	OK = N

Tabelle 25: Bedrohungsauswertung der Kommunikation Wirt – Gast

Kommunikation Wirt – Gast		**Schutz**
B5.1 Information disclosure	Anzeige der Gast-Container-Prozesse Die Anzeige von Kind-Prozessen in der Prozessliste des Wirt-Systems ist eine Basis-Funktionalität von Linux. Nur der Wirt-Administrator kann diese sensiblen Informationen einsehen.	OK = J
B5.2 Tampering	Änderung Gast-Daten von Wirt aus Das Arbeiten auf Dateisystemebene des Wirt-Systems ist eine Basis-Funktionalität von Linux. Nur der Administrator des Wirt-Systems kann diese sensiblen Informationen einsehen.	OK = J

B5.3 Spoofing	Man-in-the-middle Angriff auf Gast-VM Im GSK wird für die Umsetzungs-Phase in *M 4.237 Sichere Grundkonfiguration eines IT-Systems* von der Verwendung von DHCP abgeraten. Während der Betriebs-Phase sind weitere Maßnahmen erforderlich.	OK = N

Tabelle 26: Bedrohungsauswertung der Kommunikation von außen – Wirt

Kommunikation von außen – Wirt		**Schutz**
B6.1 Denial of Service	Überlastung des Wirt-Systems Der GSK beschreibt die Gefahr in *G 0.40 Verhinderung von Diensten (Denial of Service)*. Allerdings sind Gegenmaßnahmen stark abhängig der verwendeten Applikationen und Protokolle, so dass als detaillierte Maßnahme zur Analyse der tatsächlichen Bedrohungen *M 5.150 Durchführung von Penetrationstests* in Frage kommt.	OK = J
B6.2 Information disclosure	Anzeige von Gast Diensten / Schnittstellen Durch Zugriffsschutz per Passwortschutz und Verbindungsverschlüsselung der Daten.	OK = J
B6.3 Elevation of privilege	Angriff auf Administrationsschnittellen Der GSK nennt in *G 5.18 Systematisches Ausprobieren von Passwörtern* die Gefahren von automatisiert ermittelbarer Kennwörter. Der SSH-Login erhält dabei besondere Beachtung, da hierüber eine direkte Administration per root-Benutzer möglich ist. In der Maßnahme *M 5.64 Secure Shell* wird anstatt der Verwendung von Kennwörtern der Einsatz von Zertifikaten mit Public-Key-Verfahren empfohlen zur Authentifizierung. Durch die reguläre Serverüberwachung und mit *M 5.71 Intrusion Detection und Intrusion Response Systeme* sind Brute-Force-Angriffe erkennbar. Die Maßnahme ist die Korrelation der Ereignisse.	OK = N

Die in der Auswertung der Bedrohungsanalyse ermittelten verbleibenden Risiken (OK = N), müssen weiter betrachtet werden, um diese durch die Auswahl von geeigneten Gegenmaßnahmen noch stärker abzusichern.

5.2 Auswahl der Gegenmaßnahmen

Zur weiteren Minimierung des Risikos der festgestellten verbleibenden Bedrohungen aus der Bedrohungsanalyse, sind im Folgenden entsprechende Gegenmaßnahmen beschrieben. Weitere Absicherungsmöglichkeiten sind im Folgenden vorgestellt für die Bereiche der eingesetzten Technologien und zur Einschränkung der Zugriffsrechte und Protokollierung von Sicherheitsverletzungen. Anschließend wird das Vorgehen nach BSI fortgeführt, mit konkreten Maßnahmen zur Abwendung der ermittelten Gefährdungen.

5.2.1 Eingesetzte Technologien

Die analysierten Virtualisierungs-Implementierungen setzten auf ihre Kernfunktionalität, also die Bereitstellung virtueller Ressourcen zum Betrieb von Betriebssystemen. Grundlegende Aspekte der Linux-Kernel-Sicherheit finden dabei in einem gegenüberstellenden Vergleich (Tabelle 7) Beachtung. Allerdings ist eine stärkere Absicherung oder Aufweichung der Standardeinstellungen, beispielsweise unter Docker, per optionalem Parameter auf der Linux-Shell möglich.

Dies erfordert eine umfangreiche Expertise, um eine von den Standardeinstellungen abweichende Konfiguration der capabilities, cgroups und namespaces vorzunehmen. Die Bordmittel des Linux-Kernels zur Ressourcenbegrenzung sind somit für den jeweiligen Anwendungsfall, also der innerhalb des Gast-Systems ausgeführten Applikation zu bewerten, in Bezug auf Vor- und Nachteile auf die Systemsicherheit des Wirt-Systems.

Zum Schutz vor neuen und zukünftigen Gefahren, die im Rahmen der Bestandsaufnahme durch die Bedrohungsanalyse nur schwer zu bewerten sind, sind weitere Aktivitäten zur Beobachtung notwendig. Meines Erachtens ist es eine konkrete Maßnahme, dass für die analysierten Open Source-Werkzeuge, die jeweiligen Git-Repositories zur Speicherung des Quelltextes im Blick zu behalten, um rechtzeitig auf grundlegende Änderungen vorbereitet zu sein. Zusätzlich liefern Sicherheits- und Entwickler-Mailinglisten, Fehldatenbanken wie Common Vulnerabilities and Exposures (CVE) und die Bug Tracker von KVM und Docker zeitnahe Informationen zu Sicherheitslücken. Gängige Schlagworte sind beispielsweise die Gefahr eines „Buffer Overflow“, was infolge „Privilege escalation“ ermöglicht und zu „Directory traversal“ führen kann. Im Unternehmen sind Prozesse zu etablieren, um neue Softwareversionen zeitnah zu testen und in den Produktivbetrieb zu nehmen.

5.2.2 Einschränkung der Zugriffsrechte

Zugriffsrechte sind in einer Matrix darstellbar, um dem zugreifenden Subjekt auf ein Objekt zwei mögliche Reaktionen zu geben: genehmigen oder verweigern des Zugriffs [Lampson 1971].

In einer mathematischen Formel ausgedrückt bedeutet dies die Zugriffsmatrix M:

$$M: \text{Menge der Subjekte} \times \text{Menge der Objekte} \rightarrow 2^{\text{Zugriffsrechte}} \text{ [Eckert 2012; S. 264]}$$

Zugriffsrechte sind unter Linux in Form der Dateiberechtigungen in die Bereiche „User“, „Group“ und „All“ auf Lesen, Schreiben und ausführenden Zugriff aufgeteilt. Ob die Zugriffsberechtigung erlaubt auf eine Ressource zuzugreifen, ist allein auf Basis der Benutzerkennung festgelegt.

Dieses Prinzip trägt den Namen Discretionary Access Control (DAC). Bei dieser benutzerbestimmbaren Zugangskontrolle steuert der berechtigte Benutzer selbst die Zugriffsberechtigungen, indem primär der Besitzer der Ressource selber festlegt, wer Zugriff erhält und wer nicht.

Die Definition von DAC besagt, dass berechtigte *Subjekte* selber die Entscheidung der Zugriffsberechtigung kontrollieren, um den Zugriff auf ein *Objekt* zu erlauben oder abzulehnen [Caplan 2007; S. 5-8].

Die Prüfung findet statt, wenn ein Benutzer (Subjekt) Zugriff wünscht und wertet aus, ob für eine bestimmte Datei (Objekt) die gewünschte Lese- und Schreibberechtigung besteht (`rw-`). Problematisch ist, dass der Dateieigentümer diese Berechtigungen bewusst verändern kann oder ungewollt durch Schadcode setzt. Beispielsweise ist es denkbar, dass ein sog. trojanisches Pferd die vollen Zugriffsberechtigungen für vertrauliche Dateien setzt, sodass die Informationen von anderen Benutzern einsehbar und veränderbar sind.

Dieses Prinzip des benutzerbestimmbaren Zugriffs gilt nicht nur für Dateien, sondern auch für von Benutzern gestartete Prozesse. Ein auf Linux-Ebene im Hintergrund laufender Daemon-Prozess ist meist dem root-Benutzer zugeordnet. Das Problem ist, dass die höheren root-Benutzerrechte immer vollen Zugriff erhalten und die Benutzerprüfung umgehen. Somit kommen die Sicherheitsaspekte im Linux-Kernel (Kapitel 2.4) zum Tragen, um die Kontrolle von Zugriffsberechtigungen zu steuern. Funktionalitäten sind vorhanden um Prozesse in einem abgeschotteten Bereich auszuführen (chroot), Berechtigungs-Bits zu gewähren (setuid und setgid) und feiner nur notwendige Aktionen zu erlauben (Capabilities). Weitere Eigenschaften sind die Steuerung von Ressourcen, um diese zu priorisieren (Cgroups) oder voneinander zu isolieren (Namespaces) und nur minimale Kernel-Funktionalität bereit zu stellen (Seccomp).

Die Schwächen von DAC sind, dass der Benutzer selber Berechtigungen verändern darf. Applikationen können unentdeckte Sicherheitslücken enthalten und Prozesse, die unter root-Rechten laufen, können zur Kompromittierung des gesamten IT-Systems führen.

5.2.2.1 Mandatory Access Control

Die verpflichtende Zugangskontrolle mit der Bezeichnung Mandatory Access Control (MAC) basiert auf dem Grundgedanken, dass Dateien dem Unternehmen selbst, statt einer einzelnen Person gehören. Im Gegensatz zu DAC, wo der Benutzer der Eigentümer einer Datei ist und selbst die Zugriffsberechtigungen darauf steuert, ist mit MAC eine zentrale Stelle für die Steuerung der Zugriffsberechtigung zuständig.

Die Basisfunktion von MAC ist die Steuerung des Zugriffs eines *Subjekts* per *Zugangskontrolle eines Referenzmonitors* auf ein *Objekt* [Caplan 2007; S. 5-8].

Subjekte lassen eine Unterscheidung zwischen Personen, Gruppen und Prozessen zu. Bei Objekten sind neben Dateien auch andere Prozesse zu unterschieden. Der Referenzmonitor ist eine zentrale Autorisierungs-Stelle außerhalb des Einflussbereichs normaler Benutzer. Mit MAC ist die unabhängige Steuerung von Benutzerberechtigungen und Prozessberechtigungen realisierbar, da MAC unabhängig der Rechte des root-Benutzers und laufender Daemon-Prozesse arbeitet. DAC und MAC ersetzen sich nicht gegenseitig und ermöglichen so den parallelen Einsatz. Die Erlaubnis für ein Subjekt zum berechtigten Zugriff auf ein Objekt ist in diesem Fall von beiden Konzepten zu bewilligen [Lindqvist 2006; S. 25]. Durch den Einsatz von MAC sind Daten vor unautorisierten Zugriffen geschützt. Dies garantiert die Einhaltung der Schutzziele (Kapitel 2.5.3) in Bezug auf die Vertraulichkeit und Integrität der Informationen im Unternehmen. Für die Anwendung des MAC-Konzepts stehen verschiedene Sicherheitsmodelle zur Auswahl.

Multi Level Security (MLS) definiert Sicherheitsstufen, um Informationen anhand dieser einzuordnen. In verschiedene Stufen unterteilt ist der notwendige Schutz hierarchisch festgelegt, beispielsweise in „öffentlich“, „vertraulich“, „geheim“ und „streng geheim“, mit welcher jedes Objekt und Subjekt markiert ist (Label). Der Informationsfluss innerhalb der gleichen Stufe in Bezug auf Lesen und Schreiben ist ohne Einschränkung möglich. Um jedoch Daten von einer in eine andere Stufe zu bewegen, sind verschiedene Konzepte vorhanden.

- Nach *Bell-LaPadula* (Abbildung 9: linke Seite) sind schreibende Übergänge nur in die gleiche oder eine höhere Stufe möglich und lesende Zugriffe nur in der gleichen oder in eine niedrigere Stufe. Somit ist die Vertraulichkeit der Daten sichergestellt, da Lesen höher eingestufter Informationen („no read up“) und wie die Bereitstellung in eine niedrigere Sicherheitsstufe („no write down“) nicht möglich ist [Caplan 2007; S. 9].
- Das *Biba*-Modell ist die Umkehr dieses Konzepts und gewährgeleistet so die Integrität der Daten, indem das Schreiben in höher berechtigte Stufen unmöglich ist und kein Lesen aus niedrigeren Stufen erlaubt ist.

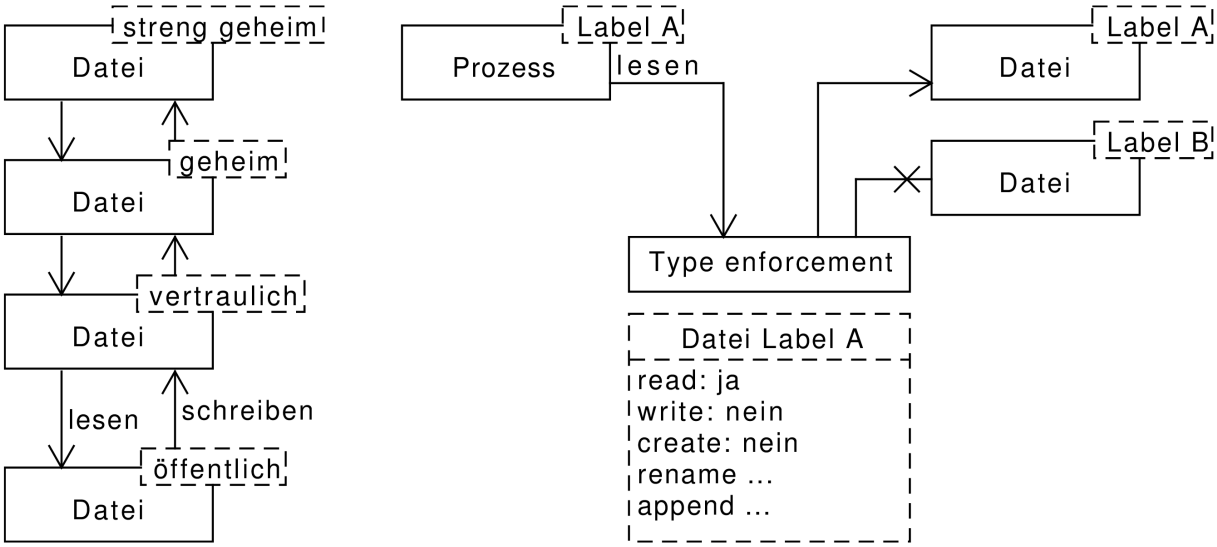

Abbildung 9: Bell-LaPadula Multi Level Security gegenüber Domain Type Enforcement

MLS hat seinen Ursprung im militärischen Umfeld, um den Inhalt von Informationen zu schützen (Vertraulichkeit) oder in einer Befehlshierarchie die Veränderung der Informationen zu verhindern (Integrität). Bei der Umsetzung dieser Modelle besteht der Nachteil, dass wiederholt neu eingestufte Informationen nach *Bell-LaPadula* abschließend „streng geheim“ und nach *Biba* „öffentlich“ sind, da nur in einer Richtung ein Durchlaufen der Sicherheitsstufen möglich ist.

Mit *Domain Type Enforcement* (DTE) steht ein weiteres Sicherheitsmodell zur Verfügung. Im Vergleich zu MLS findet ebenfalls eine zusätzliche Markierung (Label) statt, jedoch ohne die abgestufte Hierarchie zu verwenden. Für die Zugangskontrolle ist eine Entscheidung allein aufgrund von beteiligtem Subjekt und Objekt möglich. In Abbildung 9 ist auf der rechten Seite dargestellt, wie einem Subjekt als aktiver Prozess ein Label zugeordnet ist, welcher im Sicherheitskontext die *Domain* ist. Wohingegen *Type* das einem Objekt zugeordnete Label ist. Deutlich ist, dass die Entscheidung über den Zugriff nicht von einer Benutzergruppe oder einer Sicherheitsstufe abhängt, sondern allein auf Basis der beteiligten Label („A“) und dem Regelwerk entsteht. Durch den Verzicht auf starre Hierarchien, ist dieses Sicherheitsmodell zusätzlich dynamisch erweiterbar, was somit einen flexiblen Einsatz im Unternehmen ermöglicht [Caplan 2007; S. 10].

Weitere Sicherheitsmodelle existieren im Bereich der MAC wie *Chinese Wal* und *Clark-Wilson*, sind aber für den weiteren Verlauf dieser Masterthesis nicht relevant.

Die Umsetzung der verschiedenen Sicherheitsmodelle von MAC ist unter Linux unterschiedlich implementiert. In einer Reihe von Kernel-Patches sind mit „grsecurity“ neben MAC auch weitere Sicherheitshärtungen vorgenommen worden. Allerdings muss der Anwender den Kernel selber patchen, kompilieren und einbinden. Dieser Prozess ist in der Praxis mit einigen Hürden verbunden, beispielsweise mit der Frage der Unterstützung bei möglichen Problemen. Dies erschwert das Vorgehen, wenn es darum geht diese Variante Unternehmensweit durchzusetzen.

Seit Juni 2009 gibt es mit Kernel 2.6 die einheitliche Schnittstelle Linux Security Modules (LSM), welche an Stellen, die eine Zugangskontrolle erfordern, einen sog. Hook hinzufügt. Inzwischen nutzen die LSM-Schnittstelle die Kernel-Sicherheitslösungen: AppArmor, SELinux, Smack, TOMOYO und Yama. Der Vorteil ist, dass kein manuelles Kompilieren des Kernels notwendig ist und die LSM-Schnittstelle so bei Bedarf direkt zur Verwendung bereit steht.

Zusammengenommen sind die Vorteile von MAC die zentrale Steuerung von Zugriffsrechten. Die Schwächen von DAC, durch vom Benutzer anpassbare Berechtigungen, sind ausgeglichen, ohne dabei ein Konzept durch das andere ersetzen zu müssen. Die Implementierung und Administration von MAC ist im folgenden Abschnitt am Beispiel von SELinux beschrieben.

5.2.2.2 Security Enhanced Linux

Security Enhanced Linux (SELinux) implementiert MAC über die einheitliche Sicherheits-Schnittstelle LSM im Linux-Kernel. Dabei sind als Sicherheitsmodelle die folgenden Modelle integriert:

- MLS zur Vergabe hierarchischer Berechtigungen auf Level-Ebene in einer von Bell-LaPadula abgewandelten Form.
- Aufbauend auf MLS setzt SELinux zusätzlich Multi Category Security (MCS) ein, zu Trennung von Zugriffsrechten innerhalb des gleichen Levels.
- DTE als dynamisch erweiterbares Regelwerk zur Steuerung des Zugriffs von Domain (Prozess-Subjekt) auf einen Typ (Datei-Objekt).
- Role Based Access Control (RBAC) als ein weiteres Sicherheitsmodell, um individuelle Berechtigungen auch auf Rollen-Ebene zu steuern. Der Vorteil von Rollen ist die Vergabe von Berechtigungen anhand des Aufgabenbereichs beteiligter Subjekte und Objekte. RBAC steht im weiteren Verlauf dieser Masterthesis nicht im Mittelpunkt.

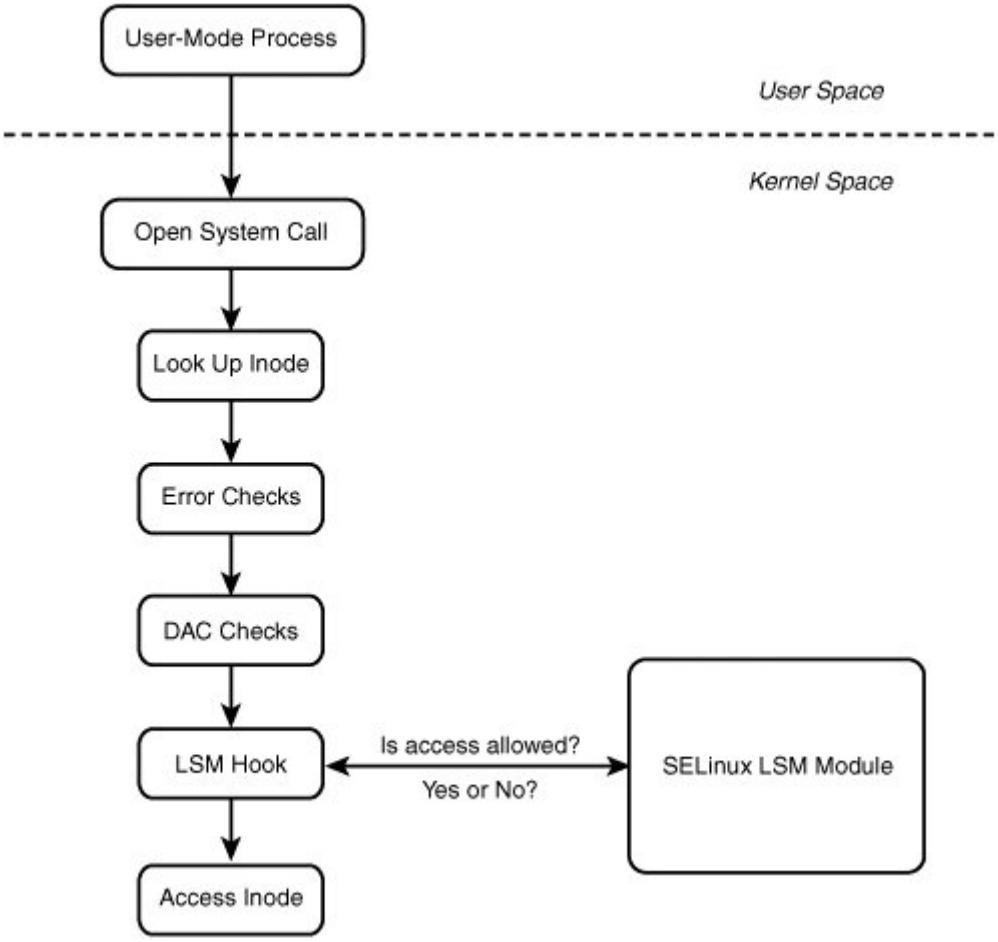

Abbildung 10: LSM Hook Architektur (Quelle: Caplan [2007; S. 40])

Die Architektur sieht vor, dass die Prüfung des Zugriffs auf Dateien nach wie vor per DAC auf Benutzer und Gruppen-Ebene für Lesen, Schreiben oder Ausführen stattfindet. Erst nach gelagert, wenn der Zugriff per DAC erlaubt ist, entscheidet SELinux per LSM-Hook anhand der integrierten Sicherheitsmodelle, über den Zugriff per Regelwerk. Somit können durch DAC zuvor abgelehnte Zugriffe nicht nachträglich durch SELinux wieder erlaubt werden.

Die Integration von SELinux macht eine Umsetzung des Regelwerks für das ganze IT-System erforderlich, also für alle Dateien, Prozesse und Benutzer. Dies ist ein entscheidender Unterschied im Vergleich zu AppArmor, wo Sicherheitsbeschränkungen lediglich für bestimmte Applikation greifen. Der Betriebsmodus von SELinux lässt sich allerdings konfigurieren, um das Regelwerk zu erzwingen (enforcing), Zugriffsverletzungen zuzulassen und zu protokollieren (permissive), oder komplett zu deaktivieren (disabled). Red Hat Enterprise Linux und die Open Source Variante Fedora haben SELinux bereits vorinstalliert und im enforcing-Modus aktiviert. Unter Debian und dem darauf basierendem UCS, ist zum Betrieb von SELinux die manuelle Installation und Aktivierung erforderlich. Bei einem erforderlichen Neustart erhalten sämtlichen Dateien im Dateisystem die entsprechenden Labels aus dem Regelwerk zugeordnet und im Dateisystem sind diese als erweiterte Dateiattribute gespeichert. Anschließend ist SELinux im permissive-Modus einsatzbereit.

SELinux liegt die Idee zugrunde, dass ein ausführbares Programm nur Zugriff auf die Ressourcen bekommt die zum Start erforderlich sind. Anschließend führt SELinux einen Übergang (type transition) aus, so dass der laufende Prozess ebenso nur die Berechtigungen erhält, die zum einwandfreien Betrieb notwendig sind. Die Funktionsweise von SELinux sieht prinzipbedingt vor, dass zunächst sämtliche Zugriffe über die LSM-Hooks abgelehnt werden und per Regelwerk eine explizite Erlaubnis erforderlich ist. Dies betrifft im Hintergrund laufende Daemon-Prozesse, denn der Administrator oder der Benutzer soll in seinen Routineaufgaben nicht beschränkt werden.

Mit SELinux zeigen bestehende Befehle auf Kommandozeile über einen zusätzlichen Parameter (Z) den Sicherheitskontext von Dateien, der Prozessliste oder beispielsweise des eingeloggten Administrators an, der unbeschränkten Zugriff per Label erhält (Listing 6).

Listing 6: Programm id mit Sicherheitskontext

```
# id -Z
unconfined_u:unconfined_r:unconfined_t:SystemLow-SystemHigh
```

SELinux listet den Sicherheitskontext auf in `Benutzer_u:Rolle_r:Typ_t:Level`, wobei anhand der jeweiligen Label durch die Erweiterung erkennbar ist, zu welchem Bereich ein Label gehört. Diese Konvention vereinfacht die Administration, denn für SELinux hat die Erweiterung keine Bedeutung.

Ist die Arbeitsweise eines Daemon-Programms bekannt, so sind bereits fertig konfigurierte Policy-Module über das „SELinux Reference Policy Project“ verfügbar. Das Regelwerk enthält die für die Zugriffsberechtigungen notwendigen Anweisungen. Einheitliche Regeln stehen bereit, auf dessen Basis vom Betriebssystem-Hersteller weitere Anpassungen erfolgen. Unbekannte Programme wie Docker ohne eigenes Regelwerk erhalten ein Standard-Label, um das Umgehen der MAC zu verhindern. Für bekannte Software wie Mail- (exim), Web- (apache2), oder SSH-Server sind so bereits die jeweiligen Label aktiv, um dazugehörige Ressourcen wie beispielsweise Ports, Sockets oder Berechtigungen auf zusätzliche Ressourcen wie die Protokoll-Datei zu erlauben.

Listing 7: Ausschnitt der Prozessliste mit Sicherheitskontext

```
system_u:system_r:init_t:s0             root     […] /usr/bin/docker -d -H fd://
system_u:system_r:exim_t:s0             Debian-+ […] /usr/sbin/exim4 -bd -q30m
system_u:system_r:httpd_t:s0            root     […] /usr/sbin/apache2 -k start
system_u:system_r:httpd_t:s0            www-data […]  \_ /usr/sbin/apache2 -k start
system_u:system_r:sshd_t:s0-s0:c0.c1023 root     […] /usr/sbin/sshd -D
system_u:system_r:sshd_t:s0-s0:c0.c1023 root     […]  \_ sshd: root@pts/0
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 root […]  \_ -bash
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 root […]      \_ ps auxfZ
```

Im oberen Listing hat so beispielsweise der SSH-Daemon ein eigenen Label-Typ (`sshd_t`), welcher für den Prozess als zusätzliches Attribut aktiv ist. Das Level kann sich in MLS-Level und MCS-Kategorie unterteilen. Mit `s0-s0` ist der Bereich der MLS-Level angegeben, welcher in diesem Fall der gleiche wie `s0` ist und `SystemLow` entspricht. Zur Vereinfachung sind alle Prozesse dem gleichen Sicherheitslevel zugeordnet, so dass MLS zwar aktiv ist, aber ungenutzt bleibt. Eine Unterscheidung findet per MCS statt, wobei `c0.c1023` alle Kategorien umfasst (`c0` bis `c1023`). Bei einer Trennung über Komma statt Punkt wären für einen Prozess nur einzelne MCS-Kategorien aktiv. Weiterhin sind die Benutzerrechte von jedem Prozess sichtbar. Zur besseren Darstellung sind in Listing 7 einige Details gekürzt (CPU, Arbeitsspeicher, Prozess-ID, Priorität und Startzeit) und es folgt der Prozess-Pfad mit den übergebenen Parametern. Das Beispiel macht deutlich, dass ein Daemon-Prozess zwar mit root-Rechten läuft, aber durch Label verpflichtend an MAC gebunden ist. Der root-Benutzer erhält dabei immer vollen Zugriff auf das System zur ungehinderten Administration per ssh. Die Hierarchie der Kind-Prozesse ist auch in der Prozessliste abgebildet.

Das Regelwerk entscheidet über Zugriffsberechtigung eines Prozesses auf Dateien eines anderen Typs anhand fein aufgegliederter Berechtigungen. Der Administrator kann selbst Regeln erstellen und zur Laufzeit in den Kernel laden. Der Zugriff auf eine Datei ist über `create`, `ioctl`, `read`, `getattr`, `lock`, `write`, `setattr`, `append`, `link`, `unlink` und `rename` detaillierter steuerbar als mit DAC. Die Menge an Konfigurationsmöglichkeiten macht nicht nur die Administration komplex, sondern auch das Erzwingen der Regeln im laufenden Betrieb. Die per Hook-Mechanismus geprüften Zugriffsregeln benötigen zusätzliche Rechenzeit. In einem Benchmark zeigte sich, dass mit SELinux das Öffnen und Schließen einer Datei 12 % und der Start eines Prozesses 5 % mehr Zeit benötigt, sowie das Erzeugen einer Datei fast doppelt so lange braucht [Ralf Spenneberg 2008; S. 51-53].

Abschließend ist festzuhalten, dass mit SELinux per MAC die zuvor genannten Probleme von DAC behoben sind, nämlich der Rechtevergabe durch den Ressourcen-Eigentümer selbst, sowie die Gleichbehandlung der Berechtigungen auf Benutzer- und Prozess-Ebene. Berechtigungen sind von gesetzten Labels abhängig und lassen sich von zentraler Stelle direkt aus dem Kernel heraus per Regelwerk steuern. Somit ist das System vor zukünftigen Sicherheitslücken geschützt, die eine Rechteausweitung verlangen und Zugriff auf normalerweise nicht zum Betriebsablauf gehörende Dateien wünschen, wie beispielsweise die Passwortdatei.

5.2.3 Protokollierung und Auswertung

Die Administration einzelner IT-Systeme verläuft über die Einrichtung von Zugangsberechtigungen und die Konfiguration der Applikationen. Bei einer Überlastung ist eine manuelle Analyse zur Behebung der Ursachen notwendig. Für die Auswertung eignen sich die Protokoll-Dateien der Applikationen und zur Untersuchung der Systemsicherheit stehen je nach Einsatzzweck verschiedene Hilfsprogramme bereit. Die Programme `chkrootkit` oder `rkhunter` helfen, um bei einer regelmäßigen Integritätsprüfung zu erkennen, ob sich beispielsweise die Passwortdatei verändert hat. Allerdings können vom Angreifer diese Programme oder die lokal gespeicherten Prüfsummen zur Erkennung von Änderungen ebenfalls kompromittiert worden sein.

Diese manuelle Vorgehensweise ist für Unternehmen nicht praktikabel, da zur Gewährleistung der Informationssicherheit eine größere Anzahl an Systemen, Benutzern und Zugriffen, sowie vielfältige zu schützende Werte zu überwachen sind. Rechner-basierte Intrusion Detection Systeme (IDS), wie `Open Source Tripwire` oder `OSSEC`, prüfen Systeme und speichern Prüfsummen extern, um so die Integrität der Daten zu sichern. Zur Sicherstellung der Verfügbarkeit und frühzeitigen Erkennung von Ausfällen, eignen sich Programme wie `Nagios` oder `Icinga`. Eine Kombination aus mehreren Diensten und Werkzeugen mit integrierter CVE-Datenbank, wird mit `OpenVAS` (Open Vulnerability Assessment System) angeboten. Das BSI empfiehlt OpenVAS als Schutzmaßnahme für Sicherheitsuntersuchungen zum Aufspüren von Schwachstellen und für das Schwachstellen-Management im Unternehmen [Bundesamt für Sicherheit in der Informationstechnik OpenVAS 2015]. Ein Unternehmen kann so je nach Bedarf und gewünschtem Einsatzfall, ein oder mehrere Programme zur Sicherstellung der Informationssicherheit einsetzen. Das Problem wurde so allerdings lediglich verlagert, da die Programme jeweils eine Vielzahl an Protokoll-Dateien erzeugen. Zur Festlegung von Schwellwerten für Alarmmeldungen, sind zusätzlich die Programme separat zu konfigurieren.

Das Ziel ist es, die eigene IT-Infrastruktur des Unternehmens im Blick zu behalten, statt für jeden Einsatzfall eine separate Konfiguration vorzunehmen. Bewährte Programme, zusammen genommen und aufeinander abgestimmt, sind die Grundlage einer pro-aktiven Sicherheits-Strategie. Systemausfälle, durch Angriffe oder Überlastung, sollen idealerweise erkennbar sein bevor dies der Anwender bemerkt. Lösbar ist dies mit Security Information und Event Management (SIEM), einer Kombination aus Softwarekomponenten zur Datenerfassung, Verarbeitung, Auswertung von Ereignissen und dem einheitlichen Erzeugen von Alarmen.

5.2.3.1 Security Information und Event Management

Die Ausgangssituation ist, dass ein Unternehmen nach Feststellung des Schutzbedarfs seine zu schützenden Werte absichern möchte. Nach der Analyse bestehender Bedrohungen sind die Szenarien bekannt, für welche ein erhöhter Schutzbedarf besteht, da eine Verletzung der Schutzziele zu befürchten ist. Für das Risikomanagement im Unternehmen ist es wichtig, dass die Kosten der Schutzmaßnahmen in Relation zum Ergebnis stehen, um so eine gezielte Reduktion der Eintrittswahrscheinlichkeit zu bewirken. Anstatt bereitgestellte Dienste zu minimieren oder die Verfügbarkeit des Netzwerks zur Schadensvermeidung zu reduzieren, ist das Ziel die IT-Sicherheit zu erhöhen und messbar zu machen.

Ein „Security Information und Event Management“ (SIEM) System setzt an dieser Stelle an. Die Basis ist die Vereinigung der beiden Überwachungssysteme „Security Information Management“ (SIM) und „Security Event Management“ (SEM). Dabei umfasst das SIM-Sicherheitsmanagement die Langzeiterfassung, sowie die Analyse und Berichterstattung von Protokoll-Dateien.

Das SEM-Sicherheitsmanagement beinhaltet die Echtzeitüberwachung, die Ergebniskorrelation sowie eine Alarm-Benachrichtigung. Zusammen bildet die Vereinigung von SIM und SEM ein effizientes Werkzeug, um die IT-Sicherheit in Echtzeit und auf Basis vergangener Ereignisse zu überwachen.

Ein SIEM-System besteht somit aus der Zusammenstellung von mehreren Komponenten, welche für die Verarbeitung untereinander reibungslos zusammen arbeiten müssen, denn anderenfalls kann das gesamte System nicht korrekt funktionieren [Miller u. a. 2011; S. 78].

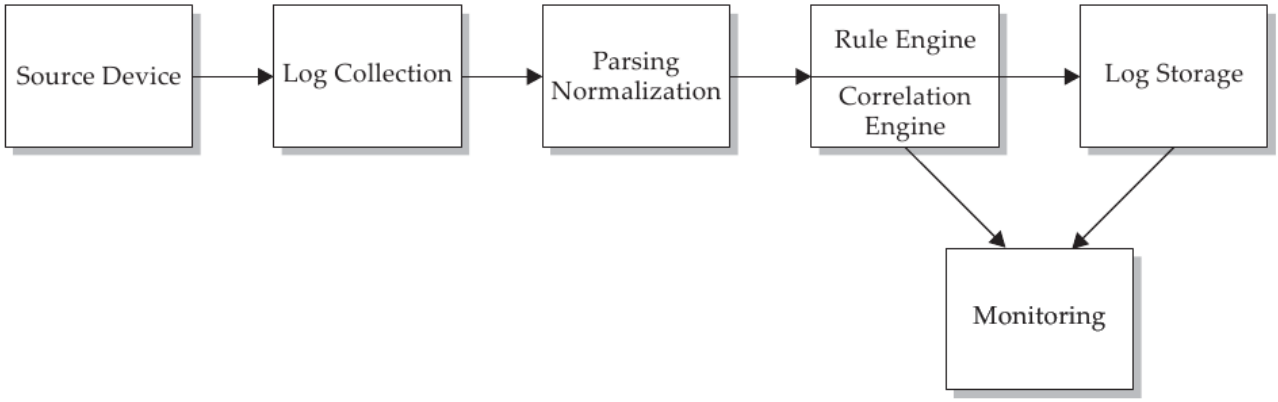

Abbildung 11: Verarbeitungskette von SIEM (Quelle: Miller u. a. [2011; S. 78])

Die in Abbildung 11 dargestellte Verarbeitungskette von SIEM besteht aus sechs Elementen, die jeweils eine bestimmte Aufgabe haben und das Ergebnis an die nachgelagerte Stelle übergeben.

Der Ausgangspunkt sind *Quellgeräte* (Source Device), welche über Router und Switches in das Netzwerk des Unternehmens eingebunden sind. Aktive Komponenten des Netzwerks liefern somit auch Protokoll-Dateien für SIEM. Für jedes Quellgerät ist nach der Art der Protokoll-Dateien zu unterscheiden. Das Betriebssystem liefert beispielsweise Informationen über fehlgeschlagene Anmeldeversuche, während ausgeführte Applikationen selbst Zugriffe protokollieren können. Standards, wie das Syslog-Protokoll aus RFC 5424, vereinheitlichen die Datenerfassung [Gerhards 2009]. Der Rechner speichert neben der tatsächlichen Meldung automatisch auch die Priorität, den Zeitstempel und den eigenen Rechnernamen. Für die spätere Verarbeitung (Normalisierung) sind diese Informationen zur Zuordnung der Ereignisse (Korrelation) von besonderer Bedeutung.

Im Schritt der *Protokollerfassung* (Log Collection) geht es darum, wie die erfassten Protokoll-Dateien an das SIEM-System zu übertragen sind. Dabei sind zwei Methoden zu unterscheiden. Bei der Push-Methode sendet das Quellgerät von sich aus die Daten an das SIEM-System. Ein Beispiel ist dabei Syslog, wobei Meldungen in Echtzeit per UDP übertragen werden. Allerdings ist zu beachten, dass UDP ein verbindungsloses Protokoll ist, was dazu führen kann, dass hohe Auslastung im Netzwerk und verursacht durch einen Angriff, die Zustellung der Protokoll-Daten an das SIEM-System verhindern könnte [Miller u. a. 2011; S. 82]. Eine andere Variante ist die Pull-Methode, bei welcher das SIEM-System aktiv die Daten vom Quellgerät abholt. Die Information erreichen das SIEM-System hierbei zeitverzögert, in vom Administrator definierten Intervallen.

In der *Syntaxanalyse und Normalisierung* (Parsing, Normalization) werden die Daten erstmalig verarbeitet. Die übertragenen Protokoll-Dateien sind zentral in ihrem ursprünglichen Format abgelegt und nun in ein für das eingesetzte SIEM-System geeignetes Format umzuwandeln. Für allgemein verwendete Informationen, wie das Syslog-Format, ist dies problemlos möglich. Für nicht standardkonforme Informationen, müssen zur Umwandlung vom Administrator eigene Filter erstellt werden. Die für SIEM wichtigen Informationen sind Uhrzeit und Datum aus dem Zeitstempel, die IP-Adresse, abgeleitet aus dem Rechnernamen und der Event, der sich aus der Protokoll-Meldung ergibt.

Im nächste Arbeitsschritt *Regelwerk und Korrelation* (Rule, Correlation Engine), verarbeitet das SIEM-System die normalisierten Events der Quellgeräte. Das Ziel ist es anhand der Meldungen auf Basis von Regeln einen Alarm auszulösen. Für eine vom verwendeten Betriebssystem unabhängige Regel zur Alarm-Benachrichtigung von lokal angemeldeten Administratoren, ist die Mel-

dung erfolgreicher Logins auf die Benutzernamen „administrator“ oder „root“ zu prüfen. Per Korrelation sind mehrere Meldungen in Kombination für einen Alarm ausschlaggebend. Wurden beispielsweise bereits drei fehlerhafte Anmeldeversuche verzeichnet, gefolgt von einem erfolgreichen Login, dann kann dies auf einen möglichen Brute-Force Angriff hindeuten [Miller u. a. 2011; S. 89].

Die verarbeiteten Daten werden im Schritt der *Protokoll-Speicherung* gesichert. Die Menge an Information, die das SIEM-System durchlaufen sollen, müssen zu Aufbewahrungszwecken und für spätere Abfragen vergangener Ereignisse aufbewahrt werden. Abhängig vom SIEM-System kann dies in Form einer Datenbank, einer Textdatei oder in einem proprietären Binärformat erfolgen.

Der finale Schritt in der Verarbeitungskette von SIEM ist das *Monitoring*. Hier findet eine Interaktion zwischen den gespeicherten Protokoll-Daten und dem Regelwerk statt. Per Interface, beispielsweise per Webanwendung oder auf Basis einer Desktop-Applikation, können Mitarbeiter des Unternehmens den aktuellen Sicherheitszustand der IT überwachen. Rohdaten sind einsehbar, um daraus neue Filterregeln entwickeln zu können. Events lassen sich auflisten und Alarm-Benachrichtigungen stehen zur Bearbeitung bereit.

Der erfolgreiche Betrieb eines SIEM-Systems im Unternehmen ist stark von der eingesetzten Hardware abhängig, um viele Events pro Sekunde (EPS) zu verarbeiteten. Derzeit gibt es eine Reihe verschiedener SIEM-Systemen von unterschiedlichsten Anbietern. In einer Untersuchung sind dabei 74 Produkte gegenübergestellt worden [Mosaic Security Research 2015]. Daraus geht hervor, dass fünf Produkte als Freeware verfügbar sind, aber meist einigen Beschränkungen unterliegen, wie beispielsweise einer maximalen EPS-Anzahl. Mit OSSIM von AlienVault und iView von Cyberoam stehen zwei Open Source SIEM-Systeme zur Verfügung. Die Implementierung und Administration eines SIEM-Systems ist im folgenden Abschnitt am Beispiel von OSSIM beschrieben.

5.2.3.2 Open Source Security Information Management

Das Open Source Security Information Management (OSSIM) ist ein SIEM-Sytem von AlienVault und basiert in der aktuellen Version 5.1 (Stand 17.08.2015) auf Debian 6. OSSIM steht unter der GPL v3 Lizenz[10] und ist somit bei Bedarf frei anpassbar, ohne dabei gegen Lizenzbedingungen zu verstoßen. Die Hardwareanforderungen verlangen mindestens 4 GB Arbeitsspeicher und einen 64-Bit Prozessor. OSSIM ist darauf ausgelegt in einer virtuellen Umgebung betrieben zu werden, wodurch bei steigendem Bedarf an Hardwareressourcen einfach die Konfiguration angepasst werden kann. Die Installations-CD als Datei im iso-Format steht auf der Hersteller-Webseite zum Herunterladen bereit, um mit dieser eine Gast-VM einzurichten. Nach der Installation kann beim ersten Zugriff über die Webanwendung, auf Wunsch das Subnetz nach weiteren Quellgeräten zur späteren Überwachung durchsucht werden. Zur Anzeige aufbereiteter Informationen im Frontend, sind zunächst die Daten der Quellgeräte zu konfigurieren und abzurufen.

OSSIM sammelt Daten auf unterschiedliche Weise über Sensoren zusammen (Abb. 12 unten Mitte), abhängig von zu überwachenden Quellgeräten. Informationen über geöffnete Ports, sind direkt vom SIEM-System aus per Netzwerk-Scan ersichtlich. Für die Protokollerfassung ist wahlweise auf dem Quellgerät ein sog. Agent (Abb. 12 unten rechts) beispielsweise als IDS-Software installiert, oder die Rechner liefern selbst per Syslog-UDP die Nachrichten-Meldungen.

Der Management-Server von OSSIM (Abb. 12 Mitte) übernimmt die Verarbeitung und Auswertung der Daten. Dabei sind für die Syntaxanalyse und Normalisierung der Daten bereits für viele Hersteller und deren Produkte entsprechende Konvertierungen vorbereitet. Das Regelwerk enthält für diese Hersteller bereits vorbereitete Prüfmethoden, und per Kommandozeile kann der Administrator selber eigene Regelwerke über reguläre Ausdrücke zur Verarbeitung der Daten anlegen. OSSIM enthält nur wenige Korrelationsmuster (82 Stück) zur Erkennung von Angriffen, wie beispielsweise einen Brute-Force-Portscan. Der Hersteller verweist an dieser Stelle auf seine kommerzielle SIEM-Version Unified Security Management (USM) mit umfangreicher Mustererkennung. Die Protokoll-Speicherung findet in einer lokalen MySQL-Datenbank statt, was die Gast-VM beständig anwachsen lässt. In der USM-Variante lässt sich die Datenbank skalieren und ist für die Langzeitspeicherung von Rohdaten und eine spätere forensische Analyse dieser Daten vorbereitet.

10 AlienVault Lizenzbedingungen (https://www.alienvault.com/eula).

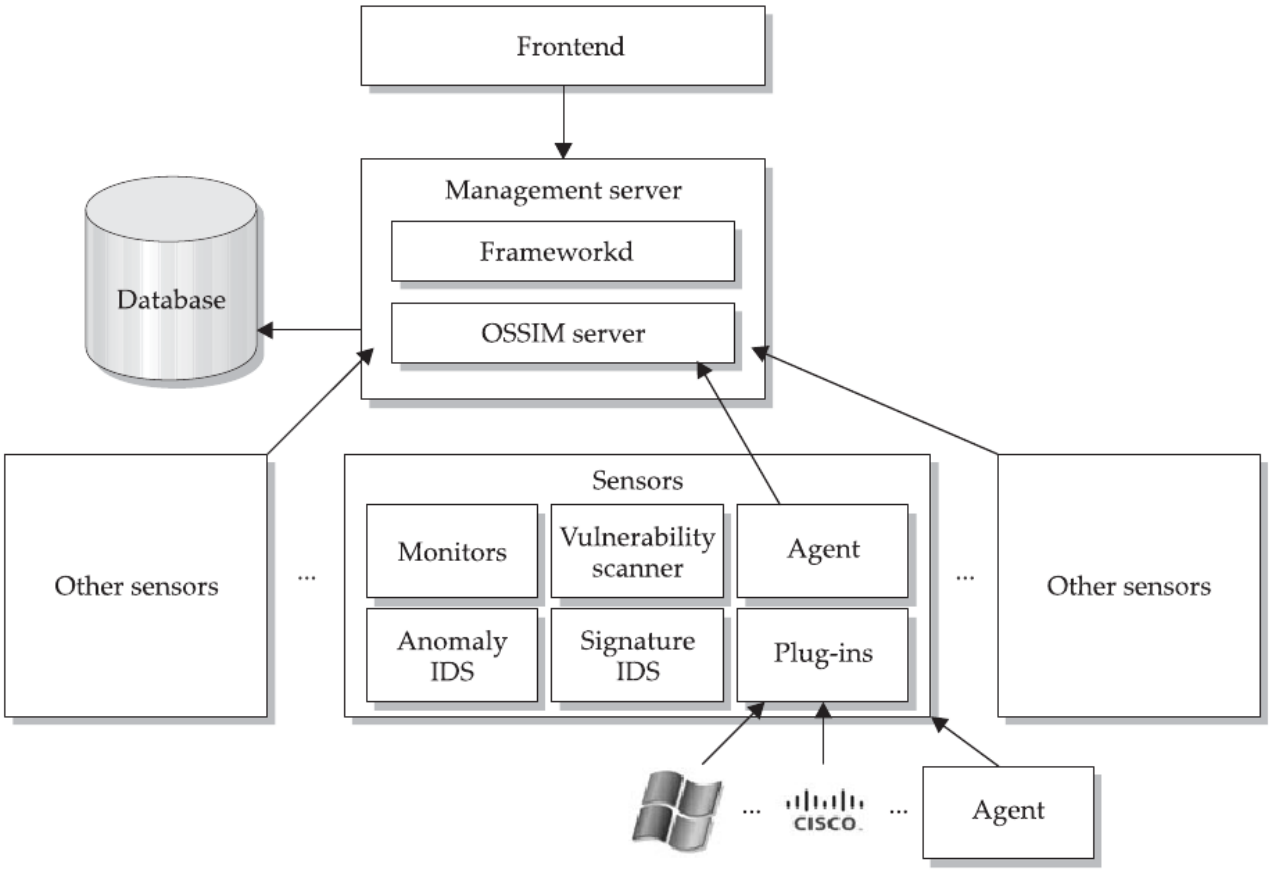

Abbildung 12: OSSIM Architektur (Quelle: Miller u. a. [2011; S. 148])

Über das Monitoring-Frontend (Abb. 12 oben) sind die Ergebnisse der verschiedenen Analysen aufgelistet. Neben einem Dasboard zur Übersicht und der Inventar-Verwaltung, werden die verwendeten Werkzeuge nahtlos unter einer gemeinsamen Webseite zusammengefasst. Die dahinter liegenden Werkzeuge für Intrusion detection (OSSEC), Schachstellenanalyse (OpenVAS) und die Anzeige der Verfügbarkeit (Nagios) sind nahtlos eingebunden und nur beim genauen Hinsehen an den URL-Aufrufen zu erkennen. Darüber hinaus sind in einer Live-Event-Ansicht aktuelle SIEM-Ereignisse zu sehen, sowie in einer Detailansicht die normalisierten Daten, mit den dazugehörigen Prioritäten und Alarm-Meldungen. Eine integrierte Ticket-Verwaltung zur gemeinsamen Bearbeitung der gemeldeten Sicherheitsverletzungen, durch das Sicherheits-Team im Unternehmen, ist vorhanden.

Eine weitere Funktion von OSSIM ist die integrierte Verknüpfung von Datenquellen und Korrelationsmuster mit den Forderungen von PCI DSS 2.0 und 3.0 (Payment Card Industry Data Security Standard), sowie dem Standard ISO 27001. So können eigene Korrelationsmuster erstellt werden, um beispielsweise den Nachweis regelmäßiger und getesteter Datensicherungen (ISO 27001: A.10.5.1 Information back-up) zu erbringen.

Abschließend ist festzuhalten, dass OSSIM zwar einigen Beschränkungen unterliegt, im Vergleich zu kommerziellen SIEM-Systemen, aber durch seine Open Source-Abstammung frei erweitert werden darf. Es ist denkbar, dass OSSIM, um zusätzliche Korrelations-Regeln und um weitere Datenquellen erweitert, bei Übereinstimmung eines Events automatisch ein Ticket generiert. So kann mit einem SIEM-System die Sicherheit verbessert und messbar gemacht werden, indem die für die Sicherheit verantworten Mitarbeiter sich vollständig auf die Bearbeitung der Tickets konzentrieren, ohne manuell im Syslog nach wiederkehrenden Anomalien zu suchen.

5.2.4 Behandlung der Risiken

Im vierten Schritte der Risikoanalyse nach BSI erfolgt die Behandlung von Risiken. Zu den ermittelten Bedrohungen aus der Auswertung der Bedrohungsanalyse (Kapitel 5.1), für die kein ausreichender Schutz besteht (OK = N), sind weitere Maßnahmen erforderlich.

Zur Beseitigung der Risiken findet eine Einteilung zur Kategorisierung statt:

A) Ergänzende Sicherheitsmaßnahmen, z. B. durch Anwendung von Empfehlungen aus der Dokumentation des Herstellers, Informationen aus Standards oder Veröffentlichungen zum Thema Informationssicherheit oder eigene Erfahrungen aus dem Unternehmen.

B) Risiko-Vermeidung durch Umstrukturierung zur Beseitigung des Risikos, z. B. durch Änderungen der Geschäftsprozesse oder der Sicherheitsarchitektur.

C) Risiko-Übernahme zur Akzeptanz des Risikos, z. B. wenn sich das Risiko im Geschäftsprozess nicht vermeiden lässt, oder nur unter ganz speziellen Voraussetzungen eintrifft.

D) Risiko-Transfer, z. B. durch Auslagerung von Geschäftsprozessen oder Abschluss einer entsprechenden Versicherung gegen das Risiko.

Ist ein Risiko aktuell akzeptabel, aber besteht zukünftig noch Handlungsbedarf, ist eine zusätzliche aktive Beobachtung sinnvoll. Den bestehenden Bedrohungen mit der jeweiligen Bedrohungsnummer (B) sind im folgenden jeweils eine oder mehrere konkrete Abwehrmaßnahmen (A) zugeordnet.

Tabelle 27: Verbleibende Bedrohung Gast-VM

Gast-VM	
B1.1 Denial of Service „Kategorie C“ + Beobachtung A1.1 Überwachung A1.1 Benachrichtigung des Administrators	Belegung der Festplatte Risiko-Übernahme: Durch Überwachung des freien Wirt-Speicherplatzes und Benachrichtigung des Administrators bei erreichen von einem zuvor definierten Limit. Während des Betriebs können Gast-VMs weiterhin zwangsweise pausiert werden, jedoch ist der Administrator bereits informiert, wodurch das Risiko durch „Überbuchen“ akzeptiert und beobachtet wird.
B1.3 Repudiation „Kategorie A“ A1.3 Überwachung	Änderung der IP-Adresse durch den Benutzer Ergänzende Sicherheitsmaßnahme: Durch zusätzliche Überwachung der einwandfreien Netzwerkfunktion in Hinblick auf IP-Adresskonflikte, können fehlkonfigurierte Gast-VMs keinen Schaden anrichten.

Tabelle 28: Verbleibende Bedrohung Gast-Container

Gast-Container	
B2.2 Information disclosure „Kategorie A“ A2.2 Mandatory Access Control	Zugriff auf Daten eines anderen Gast-Systems Ergänzende Sicherheitsmaßnahme: Das BSI nennt im GSK Mandatory Access Control als prinzipielle Zugriffsteuerung [Bundesamt für Sicherheit in der Informationstechnik IT-Grundschutz-Katalog 2015; S. 2865], setzt diese aber in keiner Maßnahme für Server ein. Für den sicheren Betrieb existierender Applikationen ist SELinux zu verwenden. Der Kernel kann so anhand gesetzter Label unterscheiden ob ein Zugriff auf Ressourcen anderer Gäste mit dem aktiven Regelwerk übereinstimmen.

B2.3 Information disclosure „Kategorie A“ A2.3 Mandatory Access Control A2.3 Überwachung A2.3 Benachrichtigung des Administrators	Zugriff auf Daten des Wirt-Systems Ergänzende Sicherheitsmaßnahme: Durch Verwendung von MAC zum sicheren Betrieb existierender Applikationen und zur Trennung zwischen Gast- und Wirt-System. Eine zusätzliche Protokollierung und Auswertung von Sicherheitsverletzungen, beispielsweise durch ein SIEM-System, führt zu einer Benachrichtigung des Administrators.
B2.7 Denial of Service „Kategorie A“ A2.7 Überwachung	Belegung von Ressourcen Ergänzende Sicherheitsmaßnahme: Anzeige der genutzten Ressourcen und Auswertung zur schnellen Erkennung bei erhöhter Ressourcenbeanspruchung. Generell sollte eine Zeitsynchronisation (M 4.348) stattfinden und TCP als zuverlässiges und verbindungsorientiertes Protokoll für Syslog zur Übermittlung der Meldungen verwendet werden.

Tabelle 29: Verbleibende Bedrohung der Kommunikation Gast – Wirt

Kommunikation Gast – Wirt	
B4.2 Tampering „Kategorie B“ A4.2 Anpassung der Architektur	Veränderung der Kernel-Aufrufe Risiko-Vermeidung durch Umstrukturierung: Durch Ausführung von Docker-Containern innerhalb einer Gast-VM auf Basis von KVM. Innerhalb der Gast-VM sind für den Gast-Container nur die gemeinsamen Ressourcen verfügbar, welche von außen für KVM-QEMU gesetzt sind (vgl. Abschnitt 3.1.1). Der Hardware-Wirt ist durch die Zwischenschichten aus Hypervior und einen separaten Kernel von den Gast-Systemen abgeschottet. Diese ermöglicht eine Aufteilung, die abhängig vom Schutzbedarf des Unternehmens oder des Kundens umgesetzt werden kann. Durch die längere Verarbeitungskette, können Applikationen in Gast-Containern ausgeführt werden ohne direkt die Systemsicherheit des Wirts zu beeinflussen. Zusätzlich können Applikationen besser skaliert oder leichter auf ein anderes Wirt-System migriert werden, da die ganze Funktionalität in der qcow2-Datei gespeichert ist und am Stück kopiert werden kann. Dies bedeutet zusätzliche Sicherheit und Flexibilität für den Betrieb von bestehenden Applikationen im Unternehmensumfeld.
B4.4 Denial of Service „Kategorie C“ + Beobachtung A4.4 Überwachung	Auslastung von Wirt- und fremden Gast-Systemen Risiko-Übernahme: Zum Schutz vor Brute-Force-Angriffen und DoS-Attacken bietet sich ein Intrusion Detection System an. Dies kann als Sensor auffällige Muster in der internen Netzwerk-Kommunikation zwischen Wirt- und Gast-Systemen erkennen.

Tabelle 30: Verbleibende Bedrohung der Kommunikation Wirt – Gast

Kommunikation Wirt – Gast	
B5.3 Spoofing „Kategorie A“ A5.3 Firewall	Man-in-the-middle Angriff auf Gast-VM Ergänzende Sicherheitsmaßnahme: Durch Blocken des UDP-Quellports 68 und des UDP-Zielports 67 per Firewall sind DHCP-Pakete nicht mehr im Gast-Netzwerk vorhanden.

Tabelle 31: Verbleibende Bedrohung der Kommunikation von außen – Wirt

Kommunikation von außen – Wirt	
B6.3 Elevation of privilege „Kategorie C“ + Beobachtung A6.3 Überwachung A6.3 Benachrichtigung des Administrators	Angriff auf Administrationsschnittellen Risiko-Übernahme: Neben der Überwachung per Intrusion Detection System ist die Korrelation und Auswertung von verschiedenen Angriffs-Mustern notwendig, wie beispielsweise die Erkennung eines SSH Brute-Force-Angriffs. Ein SIEM-System kann so automatisiert Muster erkennen und den Administrator mit einer Alarm-Meldung benachrichtigen.

5.3 Anpassung des Systemkonzepts

Im abschließenden fünften Schritt der Risikoanalyse nach BSI findet eine Konsolidierung zur Prüfung der Maßnahmen zu den analysierten Bedrohungen statt.

Zusammenfassung der Abwehrmaßnahmen:

- A1.1, A1.3, A2.3, A2.7, A4.4, A6.3 Überwachung
- A1.1, A2.3, A6.3 Benachrichtigung des Administrators
- A2.2, A2.3 Mandatory Access Control
- A4.2 Anpassung der Architektur
- A5.3 Firewall

Die ermittelten Maßnahmen sind geeignet zur Vermeidung der in den Schadensszenarien genannten Bedrohungen und deren Verletzung von Schutzzielen. Die Überwachung der unterschiedlichsten Komponenten erfordert spezielle Sensoren zur Erkennung von übermäßiger Ressourcennutzung, Sicherheitsverletzungen oder Erzeugen einer Alarm-Benachrichtigung für den Administrator.

Im Zusammenwirken einer regelmäßigen Zeitsynchronisation und dem Verzicht auf die verbindungslose Übertragung per UDP an Syslog, gewährleistet TCP einen zuverlässigen und sicheren Empfang der Protokoll-Meldungen. Für eine automatisierte Verarbeitung von Events bietet sich die Verwendung eines SIEM-Systems an.

Der Einsatz von Mandatory Access Control als zusätzliche Sicherheits-Strategie schränkt die Benutzerfreundlichkeit nicht ein. Per LSM-Hook findet eine zusätzliche Zugriffsteuerung der Berechtigungen statt. Im permissive-Modus protokolliert SELinux potentielle Sicherheitsverletzungen, damit diese anschließend ausgewertet und bei Bedarf über ein eigenes Regelwerk zu erlauben sind.

Die erarbeiteten Sicherheitsmaßnamen erweitern ein bestehendes Sicherheitskonzept im Unternehmen. Die dazu notwendige Anpassung der Architektur stellt sicher, dass die eingesetzten Open Source-Technologien den sicheren Betrieb existierender Applikationen im Unternehmensumfeld ermöglichen.

Die Änderungen der Architektur macht auch eine Anpassungen des Systemkonzepts erforderlich. In Abbildung 13 ist das aktualisierte Systemkonzept abgebildet.

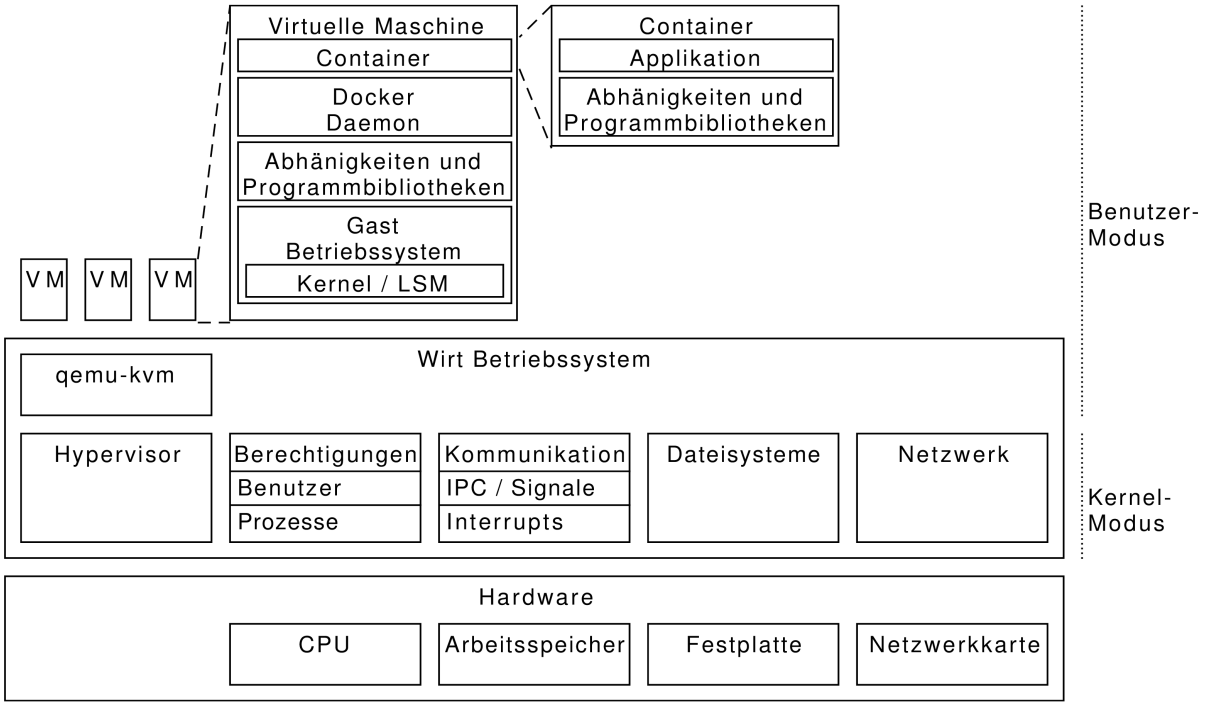

Abbildung 13: Angepasstes Systemkonzept

Im Vergleich zu der Ausgangssituation (vgl. Abbildung 7 auf Seite 22) läuft der Docker-Container innerhalb einer Gast-VM, welche zusätzlich den LSM-Hook nutzt. Dadurch sind zwei Ziele für den sicheren Betrieb erreichbar. Einerseits ist durch die zusätzliche Zwischenschicht die Trennung eines Gast-Containers zur Hardware und den Ressourcen möglich. Anderseits können bisher verborgene Sicherheitsverletzungen erkannt und protokolliert werden.

Aus technischer Sichtweise ergeben sich einige Vorteile in Hinblick auf den Betrieb existierender Applikationen. Während KVM eine stabile und skalierbare Basis durch feste Grenzwerte für Ressourcen liefert, kann die komplette VM über die langsam anwachsende qcow2-Datei bei Überbelastung auf andere Handware migriert werden. Wohingegen Docker eine agile Plattform bietet, die eine Bereitstellung vorkonfigurierter Applikationen erlaubt, für ein einfaches Ausrollen der Software und die Ausführung der Docker-Container in abgetrennter Umgebung.

Im Unternehmensumfeld bietet sich die Trennung nach den Aufgabengebieten an, um Entwicklungs-, Test- und Produktiv-Systeme zu separieren. So ist auch eine Unterteilung nach Zuständigkeiten (ISO 27001: A.10.1.3 Segregation of duties) möglich. In der Praxis bedeutet dies auch eine Zeitersparnis für Mitarbeiter und Administratoren. So ist eine zeitnahe und verlässliche Überprüfung von Software möglich, so dass beim Löschen von Containern keine Reste zurückbleiben, wie die im Vergleich zur Deinstallation eines Programms geschieht. Auch vereinfacht dies den Prozess der Auswahl von Software oder das automatische Testen für Mitarbeiter der Entwicklungs-Abteilung oder für die Qualitätssicherung. Zusätzlich ist dieses Vorgehen mit weniger Risiko verbunden, so dass bei einem Software-Update der komplette Container zu tauschen ist, ohne dass eine Störung oder Ausfall durch eine Migration von Software verursacht wird.

In Hinblick auf die Sicherheit, wenn ein Prozess mit root-Berechtigungen ausgeführt wird, kann der Linux-Kernel per Zugangskontrolle auf Basis von DAC kein unerwünschtes Laufzeitverhalten erkennen. Mit MAC zur Zugriffsteuerung per LSM-Hook, ist mit SELinux allerdings die Erkennung von Sicherheitsverletzung möglich. Die zusätzliche Auswertung von Protokoll-Daten macht den Aufbau robuster gegen die vorgestellten Schadensszenarien. Dies soll im folgenden Kapitel anhand eines prototypischen Aufbaus erläutert werden.

6 Implementierung und Demonstration

Im praktischen Teil dieser Masterthesis „Sicherer Betrieb existierender Applikationen im Unternehmensumfeld mit Open Source-Werkzeugen“ sind Entwicklung und Aufbau des Prototyps beschrieben. Während der Recherche und Analyse beteiligter Systeme für den schriftlichen Teil, sind die erwarteten Endergebnisse (Kapitel 1.5 auf Seite 2) konkretisiert worden. Diese Funktionalitäten sind für den praktischen Teil mit der MoSCoW-Methode[11] priorisiert worden.

Tabelle 32: Arbeitspakete der priorisierten Anforderungen des praktischen Teils

Beschreibung	Priorität
Arbeitspaket 1: Gewährleistung der Sicherheit in einer Gast-VM	
• Umsetzung der Schadensszenarien mit Docker	MUST
• Build des SELinux-Regelwerks für UCS als Debian-Paket	MUST
• Migration der Docker-Policy für UCS	MUST
• Protokollierung der Sicherheitsverletzungen	MUST
Arbeitspaket 2: Prototypischer Aufbau	
• Erarbeitung einer Architektur für die Verarbeitungskette	MUST
• Erfassung von Protokoll-Dateien zur zentralen Auswertung	SHOULD
• Zentrales Speichern von Protokoll-Dateien und Rechner-Zuständen	SHOULD
• Überwachung und Auswertung der Protokoll-Dateien	SHOULD
Arbeitspaket 3: Testlauf der Schadensszenarien	
• Erkennung der Überlastung durch einen Container	COULD
• Erkennung eines Ausbruchs aus dem Container	COULD

Die anfänglich linear festgelegten Funktionalitäten aus dem Grobkonzept einer möglichen Implementierung, sind für die Umsetzungs-Phase in Arbeitspakete zusammengefasst (Tabelle 32). Zusammengehörende Funktion können detailliert entworfen und im Rahmen der Zeitplanung eines Arbeitspakets agil implementiert werden. Diese Vorgehensweise entspricht dem Feature Driven Development (FDD), womit bei der Umsetzung flexibel die Planung an aktuelle Erkenntnisse der Recherche- und Analyse-Arbeit angepasst werden kann.

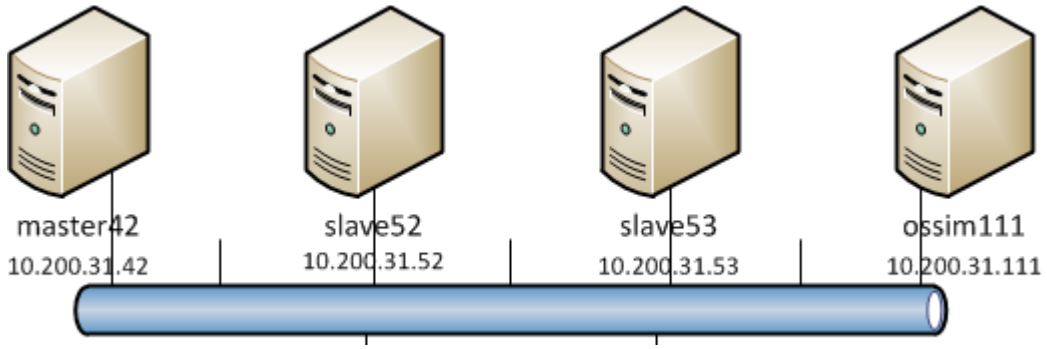

Abbildung 14: Prototypisches Unternehmensnetzwerk

Die in Abbildung 14 dargestellte Master/Slave-Architektur und externer Überwachung, symbolisiert ein Unternehmensnetzwerk für den prototypischen Aufbau im privaten IP-Adressbereich.

11 Die MoSCoW-Methode verwendet die Begriffe MUST, SHOULD, COULD und WON'T zur Priorisierung der Funktionalitäten nach RFC 2119 [Bradner 1997].

6.1 Gewährleistung der Sicherheit

Im Folgenden sind die vorbereitenden Maßnahmen zur Realisierung des Prototyps mit entstandenen Hindernissen und deren Lösung beschrieben.

Zur Entkopplung von der Unternehmensinfrastruktur, findet die *Umsetzung der Schadensszenarien mit Docker* in Gast-VMs auf Basis von KVM auf eigener Wirt-Hardware statt. Für die Absicherung mit SELinux ist mindestens Docker-Version 1.7 notwendig, da sich im Projektverlauf gezeigt hat, dass niedrigere Versionen (1.3 und 1.6) nicht mit SELinux funktionieren, obwohl laut Änderungsprotokoll Docker seit Version 0.11.0 auf SELinux vorbereitet ist[12]. Nach der Installation von Docker können verschiedene Container gestartet werden.

Starten des Fuzz-Testing (Schadensszenario 1) in Listing 8: Der Docker-Container stellt den Fuzz-Test „trinity“ bereit, welcher aus dem originalen Quelltext erzeugt wird[13] und somit keine vordefinierten Standardkennwörter enthält.

Listing 8: Docker Schadensszenario 1

```
# docker run -u nobody ewindisch/trinity
```

Ausbruch aus Docker-Container (Schadensszenario 2) in Listing 9: Ein Arbeitsverzeichnis wird von außen in den Gast-Container übergeben (a), in welches der Container-Benutzer wechselt (b). Der Administrator verschiebt den Ordner von außerhalb (c). Der Container-Benutzer kann beim Verlassen der Verzeichnisse den Inhalt der Passwortdatei des Wirts anzeigen (d).

Listing 9: Docker Schadensszenario 2

```
a) slave52:~# mkdir -p work/dir
   slave52:~# docker run -it -v /root/work/:/tmp/work ubuntu

b) 8e0686dc07d3:/# cd /tmp/work/dir/

c) slave52:~# mv work/dir/ /tmp/

d) 8e0686dc07d3:/tmp/work/dir/# cd ..
   8e0686dc07d3:/tmp/work/tmp/# cd ..
   8e0686dc07d3:/tmp/work/# cat etc/shadow
```

Zur Trennung der produktiv von Univention-Kunden genutzten Debian-Pakete, erfolgt der *Build des SELinux-Regelwerks für UCS* in einem separaten Bereich (Scope). Änderungen am Debian-Paket von SELinux[14] finden nicht an diesem selbst statt, sondern sind per Quelltext-Patch zu integrieren. Der Hintergrund ist, dass Änderungen durch den ursprünglichen Entwickler der Software oder durch die Debian-Paketbetreuer, nicht mit Anpassungen durch Univention kollidieren. So sind lediglich die Unterschiede (diff) der Anpassungen in der internen Versionsverwaltung (SVN) enthalten. Das Build-System erzeugt automatisch aus dem originalen Debian-Paket und vorhandener Patches ein neues Paket für UCS.

Die an Debian angepasste „SELinux Reference Policy“ ist für Docker zu erweitern. Die Basis der *Migration der Docker-Policy für UCS* ist Fedora 22[15]. Der komplette Patch beträgt 1.677 Zeilen und setzt sich aus den folgenden Komponenten und Funktionen zusammen:

- Übernommenes Regelwerk (Docker-Policy) aus Fedora 22 als Basis (762 Zeilen).
- Anpassung des Build-Prozesses zur reibungslosen Integration der Docker-Policy.

12 Docker-Änderungsprotokoll auf GitHub (https://github.com/docker/docker/blob/master/CHANGELOG.md)
13 Quelltext des Fuzz-Test-Containers auf GitHub (https://github.com/ewindisch/trinity-docker/blob/master/Dockerfile).
14 Selinux-policy-default 2:2.20140421-9 von Debian (https://packages.debian.org/sid/selinux-policy-default).
15 SELinux-Regeln von Fedora 22 auf GitHub (https://github.com/fedora-selinux/selinux-policy/blob/f22-contrib/docker.te).

- Hinzufügen von Funktionalität durch fehlende Abhängigkeiten anderer Policy-Module (kernel, devices, files, terminal, openvswitch, systemd, userdomain, init, virt, dbus, filesystem).
- Fehlender automatischer Übergang (type transition) für Docker-Container: Über die Datei `lxc_context` ordnet SELinux den Kind-Prozessen das richtige Label zu.
- In Fedora sind sämtliche LXC-Regeln in „sandbox“ umbenannt. Um das Regelwerk für UCS und Debian abwärtskompatibel zu halten, ist für die Masterthesis stattdessen ein SELinux Typ-Alias eingesetzt worden, um beide Varianten zuzulassen. Außerdem sind weitere spezifischer Fedora-Regeln an Debian angepasst worden.

Das gebaute Regelwerk mit Docker-Policy steht über das interne Repository im SELinux-Scope bereit und ist in der Gast-VM als zusätzliche Paketquelle einzubinden (Listing 10). Im Anhang ist der erstellte Patch mit Download-Möglichkeit und manuellem Build-Vorgang beschrieben.

Listing 10: Gast-VM: Dateiausschnitt /etc/apt/sources.list

```
deb http://192.168.0.10/build2/ ucs_4.0-0-selinux/all/
deb http://192.168.0.10/build2/ ucs_4.0-0-selinux/$(ARCH)/
```

Anschließend sind per UCR-Umgebungsvariable (Univention Configuration Registry) die zu verwendenden Paketquellen konfigurierbar (vorerst noch unbetreut und unsigniert), womit SELinux aus dem internen Repository installierbar ist. Nach Aktivierung und einem Neustart setzt das System automatisch die benötigten Label für Dateien, Programme und Verzeichnisse (Listing 11).

Listing 11: Gast-VM: Installation von SELinux

```
# ucr set repository/online/unmaintained='yes'
# ucr set update/secure_apt='no'

# univention-install selinux-basics selinux-policy-default
# selinux-activate
# reboot
```

Der Docker-Daemon ist im Folgenden für den Einsatz mit SELinux zu konfigurieren (Listing 12).

Listing 12: Gast-VM: Dateiausschnitt /etc/default/docker

```
DOCKER_OPTS="--selinux-enabled"
```

Das Ergebnis ist in Listing 13 an korrekt gesetzter Label für den Docker-Daemon und einen Docker-Container aus der Prozessliste abzulesen (Im Vergleich zu Listing 7 auf Seite 45).

Listing 13: Ausschnitt der Prozessliste mit aktiver Docker-Policy

```
system_u:system_r:docker_t:s0 root […] /usr/bin/docker -d -H fd:// --selinux-enabled
system_u:system_r:svirt_lxc_net_t:s0:c179,c439 root […] \_ /bin/bash
```

Für die *Protokollierung der Sicherheitsverletzungen* von Docker, die SELinux im permissive-Modus erkennt, ist `auditd` als ein zusätzlicher Protokoll-Daemon zu installieren (Listing 14).

Listing 14: Gast-VM: Installation von audit und fortlaufende Anzeige der Sicherheitsverletzungen

```
# apt-get install auditd
# tail -f /var/log/audit/audit.log
```

Für genaue Zeitangaben ist auf allen beteiligten Systemen ein Cronjob einzurichten, der regelmäßig über das Network Time Protocol (NTP) für eine fehlerfreie Zeitsynchronisation sorgt.

6.2 Prototypischer Aufbau

Zu den ermittelten Bedrohungen und den festgelegten Abwehrmaßnahmen (siehe Abschnitt 5.2.4) ist im Folgenden die Architektur und der Aufbau des Prototyps beschrieben.

Die *Erarbeitung einer Architektur für die Verarbeitungskette* basiert auf dem angepassten Sicherheitskonzept (Abbildung 13 auf Seite 53) und besteht aus den nachfolgenden Komponenten für den Aufbau des Prototyps:

- Wirt-Hardware zur Virtualisierung mit KVM-Hypervisor
- Gast-VM zur Kapselung von Applikationen mit Gast-Containern in Docker
  - LSM-Hook-Mechanismus: Regelwerk von SELinux mit Docker-Policy
  - Erfassen von Sicherheitsverletzungen und Nutzung eines IDS, sowie Übertragung der Protokoll-Dateien
- SIEM-VM zur Überwachung mit OSSIM
  - Normalisieren und Auswerten der erfassten Event-Daten
  - Erstellung von Alarmen und Tickets auf Basis von Sicherheitsverletzungen

Für die *Erfassung von Protokoll-Dateien zur zentralen Auswertung* bringt `auditd` als Abhängigkeit `audisp` mit. Erfasste Sicherheitsverletzungen sind nach Aktivierung dieser Funktionalität an ein entfernten Syslog-Server übertragbar (Listing 15).

Listing 15: Gast-VM: Dateiinhalt /etc/audisp/plugins.d/syslog.conf

```
active = yes
direction = out
path = builtin_syslog
type = builtin
args = LOG_INFO
format = string
```

Um protokollierte Sicherheitsverletzungen (`audit.log`) nicht zusätzlich lokal (`/var/log/syslog`) zuzustellen, ist eine zusätzliche Konfiguration notwendig. In Listing 16 ist für `rsyslog` für die Programme `root` und `audispd` eine Weiterleitung per TCP zur externen SIEM-VM definiert. Die zweite Zeile verhindert die zusätzliche lokale Speicherung der Syslog-Meldung.

Listing 16: Gast-VM: Dateiinhalt /etc/rsyslog.d/local-forward.conf

```
if $programname == 'root' or $programname == 'audispd' then @@10.200.31.111:514
if $programname == 'root' or $programname == 'audispd' then ~
```

Anschließend sind die Dienste `auditd` und `rsyslog` in der Gast-VM neu zu starten.

Für das *zentrale Speichern von Protokoll-Dateien und Rechner-Zuständen* erfolgt nach der Installation von OSSIM die Erstkonfiguration im Browser. Per Einrichtungsassistent kann das Subnetz nach weiteren Quellgeräten durchsucht werden. Für Syslog ist zunächst die entsprechende Datenquelle zu aktivieren, indem auf Kommandozeile über `ossim-setup` die Einstellung zu aktivieren ist (Configure Sensor → Configure Data Source Plugins → syslog). Der entsprechende TCP-Port ist an der Firewall zu erlauben (Listing 17), welche durch den Aufruf von `ossim-reconfig` aktiv wird.

Listing 17: SIEM-VM: Dateiausschnitt /etc/ossim/firewall_include

```
-A INPUT -p tcp -m state --state NEW -m tcp --dport 514 -j ACCEPT
```

Für den sicheren Empfang der Protokoll-Dateien ist die TCP-Funktionalität zu aktivieren (/etc/rsyslog.conf). Nach dem Neustart von `rsyslog` ordnet OSSIM die Daten dem jeweiligen Quellgerät zu.

Zusätzlich kann die Gast-VM per IDS an OSSIM angebunden werden. Die notwendigen Schritte sind die Nutzung des OSSEC-Repository zur Installation von `ossec-hids-agent`[16] und die Einbindung in OSSIM[17]. Durch den zusätzlichen Dienst wird die Systemintegrität und die Ressourcenbelegung der Gast-VM fortwährend geprüft und Anomalien erfährt das SIEM-System umgehend.

Im Folgenden ist *die Überwachung und Auswertung der Protokoll-Dateien* anhand der Schadensszenarien dargestellt. Jede empfangene Nachricht, sowohl von Syslog oder von IDS, entspricht für OSSIM einem Event. Für jedes eingehende Event errechnet OSSIM das Risiko:

Quellgerät-Wert (0-5) × Priorität (0-5) × Zuverlässigkeit (0-10) / 25 = Event-Risiko (0-10)

Standardmäßig ist der Wert des Quellgeräts „2“, Priorität und Zuverlässigkeit sind jeweils „1“ und beziehen sich auf die Meldung selbst. Alle Standardwerte sind anpassbar, so dass ein rechnerisches Risiko von mindestens „1“ für einen automatischen Alarm in OSSIM erforderlich ist.

Im ersten Schadensszenario zur Erkennung von übermäßiger Ressourcennutzung, liefert der IDS-Agent fortwährend Statusmeldungen an OSSIM. Für ein Alarm bei niedrigem Arbeitsspeicher, sind die Event-Einstellungen anzupassen über „Configuration → Threat Intelligence → Datasource“ für die Gruppe „AlienVault HIDS-service_availability“ (Abb. 15 ergibt rechnerisches Risiko von „1,2“).

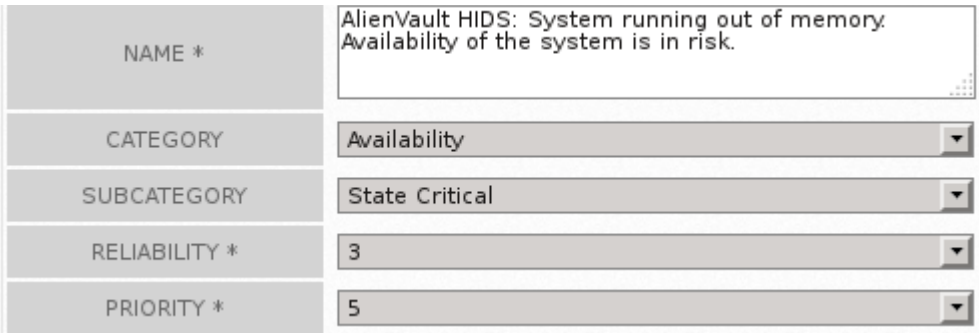

Abbildung 15: OSSIM: Event-Einstellung bei niedrigerem Arbeitsspeicher

Zur Erkennung des zweiten Schadensszenarios ist die Auswertung der protokollierten Sicherheitsverletzungen notwendig. Bei einem Ausbruch aus einem Container kann anschließend ein lesender Zugriff auf die Wirt-Passwortdatei erfolgen, um den Hash in Ruhe zu entschlüsseln. Der unerwünschte lesende Zugriff des Programms `cat` wird anhand der Label des Regelwerkes von einem Gast-Container (`svirt_lxc_net_t`) auf die Passwortdatei des Wirts (`shadow_t`) erkannt (Listing 18).

Listing 18: SIEM-VM: Übertragene Syslog-Nachricht der Audit-Protokoll-Datei

```
Sep 10 12:22:29 slave52 audispd: node=slave52 type=AVC
msg=audit(1441880549.133:5125): avc:  denied  { read } for  pid=2278 comm="cat"
name="shadow" dev="vda1" ino=131322
scontext=system_u:system_r:svirt_lxc_net_t:s0:c355,c977
tcontext=system_u:object_r:shadow_t:s0 tclass=file permissive=1
```

Per Syslog übertragene Roh-Daten liegen dem SIEM-System zur weiteren Verarbeitung vor. Die Normalisierung eigener Protokoll-Daten ist in OSSIM per Plugin erweiterbar. Ein Plugin besteht in OSSIM aus der Normalisierungs-Datei und einer Datenbank-Datei.

Per regulärem Ausdruck (Listing 19) lässt sich die Syslog-Nachricht in ihre Bestandteile in Datum, Gerät-Sensor, Quell-Programm und ausgelöstem Event zerlegen. Hierbei ist die Sicherheitsverletzung durch den Gast-Container-Prozess (`svirt_lxc_net_t`) auf die Passwortdatei des Wirts (`shadow_t`) von Bedeutung. Die normalisierten Daten werden über Python-Platzhalter internen Variablen zugewiesen, um diese weiter zu verarbeiten. Alle Dateien innerhalb des Gast-Containers haben ein einheitliches Label (`svirt_sandbox_file_t`), wodurch Fehlalarme auszuschließen sind.

16 Download-Seite von OSSEC (http://www.ossec.net/?page_id=19).
17 Anleitung zur Integration von OSSEC in OSSIM (https://www.alienvault.com/doc-repo/usm/threat-detection/AlienVault-USM-4.x-5.x-Deploying-HIDS-Agents-to-Linux-Hosts.pdf).

Listing 19: SIEM-VM: Dateiausschnitt /etc/ossim/agent/plugins/syslog.cfg

```
[syslog - audispd: container read shadowed password file]
event_type=event
regexp="^(?P<logline>(?P<date>\SYSLOG_DATE)\s+(?P<sensor>\S+)\s+(?P<source>\S+)\s+(?
P<logged_event>.*audit.*read.*svirt_lxc_net_t.*shadow_t.*)$)"
device={resolv($sensor)}
date={normalize_date($date)}
plugin_sid=2
src_ip={resolv($sensor)}
userdata1={md5sum($logline)}
userdata3={$logged_event}
```

Eine weitere Bedrohung bei einem Ausbruch aus einem Gast-Container, ist der Zugriff auf Dateien eins andern Containers, welche in der gleichen Gast-VM unter `/var/lib/docker/` gespeichert sind. Da Docker die von SELinux bereitgestellte Funktionalität MCS zur Vergabe von Kategorie-Label innerhalb des gleichen Sicherheits-Levels nutzt, ist diese Art der Komprimierung erkennbar.

Listing 20: SIEM-VM: Ausschnitt der Syslog-Nachricht bei Zugriff auf fremde Daten

```
scontext=system_u:system_r:svirt_lxc_net_t:s0:c355,c977
tcontext=system_u:object_r:docker_var_lib_t:s0:c106,c852 tclass=file permissive=1
```

In Listing 20 ist die Sicherheitsverletzung durch unterschiedliche MCS-Kategorien erkennbar. Eine Alarm-Benachrichtigung ist auszulösen, wenn die Quell-Kategorie nicht mit der Ziel-Kategorie übereinstimmt. Damit der dazu passende reguläre Ausdruck, in Listing 21, die richtige Rückgabe liefert, darf der Platzhalter der MCS-Quelle nicht dem MCS-Ziel entsprechen.

Listing 21: SIEM-VM: Dateiausschnitt /etc/ossim/agent/plugins/syslog.cfg

```
regexp="^(?P<logline>(?P<date>\SYSLOG_DATE)\s+(?P<sensor>\S+)\s+(?P<source>\S+)\s+(?
P<logged_event>.*audit.*read.*svirt_lxc_net_t:s0:(?
P<mcs>c\d{1,4},c\d{1,4})\s.*docker_var_lib_t:s0:(?!(?P=mcs)).*)$)"
```

Im Anhang ist die Plugin-Erweiterung vollständig dokumentiert, zusammen mit der Anpassung der Datenbank-Datei, um die Filterregeln im Frontend von OSSIM bereitzustellen.

OSSIM erstellt für Alarme nicht automatisch auch ein Ticket. Während ein Alarm ein temporäres Ereignis widerspiegelt, wie im ersten Schadensszenario einer kurzzeitigen Überlastung, ist im zweiten Szenario von einer Kompromittierung der überwachten Gast-VM auszugehen. Meines Erachtens ist aus diesem Grund die Erstellung eines Tickets sinnvoll, um den schwerwiegenden Vorfall zu untersuchen und das Ticket ggf. mit Abschaltung der entsprechenden VM zu bearbeiteten.

Die automatische Erzeugung eines Ticket für die erkannte Sicherheitsverletzung, ist in OSSIM unter „Configuration → Threat Intelligence“ über eine neue „Policy“ anzulegen, um als „Action“ ein Ticket zu erstellen (Abbildung 16). Der Event-Typ „Audit-read-shadow“ greift dabei direkt auf die zuvor erstellte Plugin-Erweiterung zurück.

CONDITIONS					
SOURCE	DEST	SRC PORTS	DEST PORTS	EVENT TYPES	ACTIONS
HOST: slave53 HOST: slave52	ANY	ANY	ANY	**DS Groups**: Auditd-read-shadow	New Ticket: Syslog

Abbildung 16: OSSIM: Event-Policy bei Zugriff auf die Passwortdatei

6.3 Darstellung der Schadensszenarien im Monitoring-Frontend

Die im Folgenden dargestellten Abbildungen zeigen eintreffende Events der Schadensszenarien (Abb. 17), die Alarm-Benachrichtigungen (Abb. 18) sowie offene Ticktets (Abb. 19) in OSSIM.

SIGNATURE	DATE GMT+2:00	SENSOR	OTX	SOURCE	DESTINATION	ASSET S → D	RISK
Syslog: syslog entry	2015-09-10 12:22:35	alienvault	N/A	slave52	0.0.0.0	2->2	0
Syslog: syslog entry	2015-09-10 12:22:35	alienvault	N/A	slave52	0.0.0.0	2->2	0
Syslog: syslog entry	2015-09-10 12:22:35	alienvault	N/A	slave52	0.0.0.0	2->2	0
audispd: container read file of other container	2015-09-10 12:22:35	alienvault	N/A	slave52	0.0.0.0	2->2	1
Syslog: syslog entry	2015-09-10 12:22:29	alienvault	N/A	slave52	0.0.0.0	2->2	0
Syslog: syslog entry	2015-09-10 12:22:29	alienvault	N/A	slave52	0.0.0.0	2->2	0
Syslog: syslog entry	2015-09-10 12:22:29	alienvault	N/A	slave52	0.0.0.0	2->2	0
Syslog: syslog entry	2015-09-10 12:22:29	alienvault	N/A	slave52	0.0.0.0	2->2	0
audispd: container read shadowed password file	2015-09-10 12:22:29	alienvault	N/A	slave52	0.0.0.0	2->2	1

Abbildung 17: OSSIM: eintreffende normalisierte Event-Nachrichten

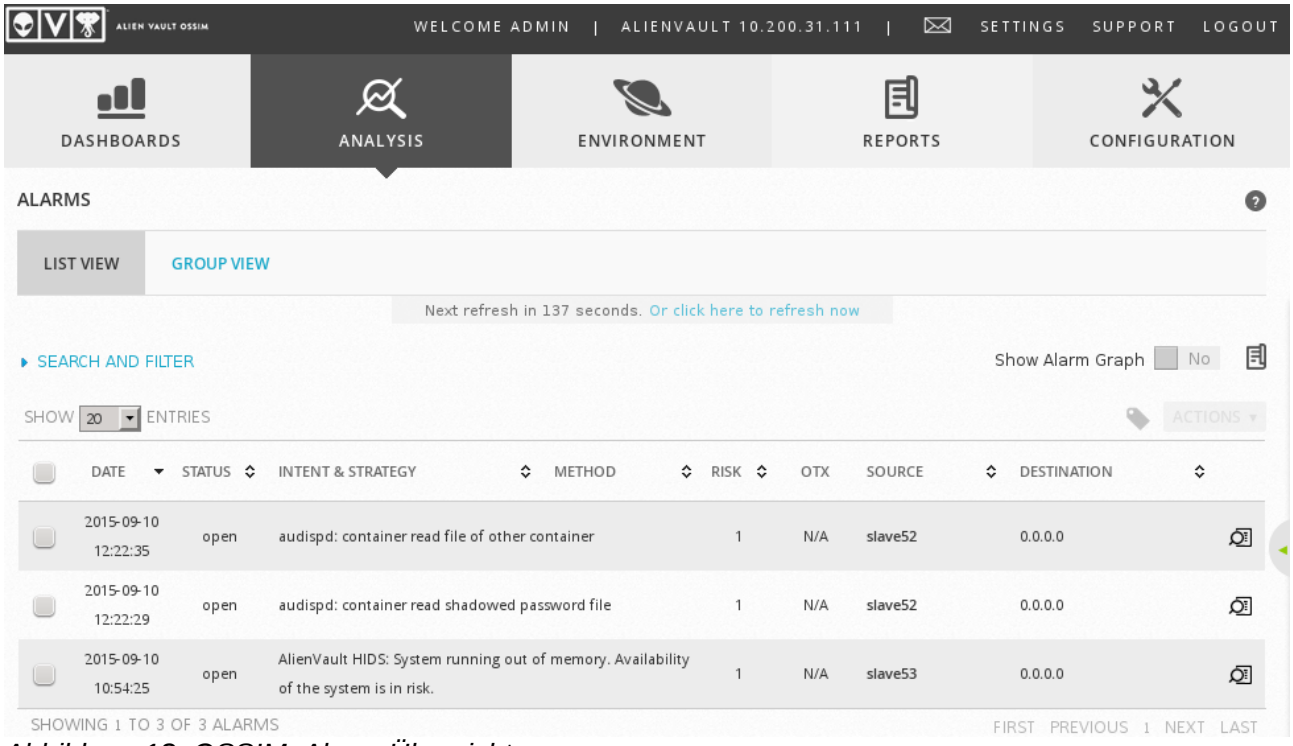

Abbildung 18: OSSIM: Alarm-Übersicht

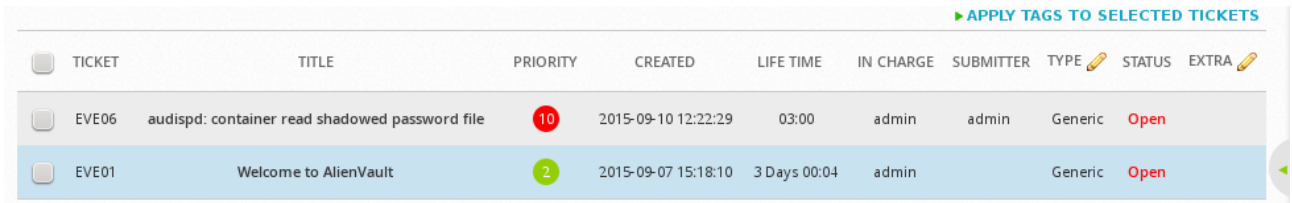

Abbildung 19: OSSIM: Ticket-Übersicht

7 Diskussion der Ergebnisse

In der vorliegenden Masterthesis wurde untersucht, welche existierenden Möglichkeiten zur Virtualisierung von Open Source-Applikationen geeignet sind und wie dabei die Sicherheit im Unternehmensumfeld gewährleistet werden kann. In einer systematischen Analyse der Bedrohungen wurden Gegenmaßnahmen zur Absicherung, in Form von der Härtung und der Überwachung der beteiligten Systeme, festgelegt. Für die praktische Umsetzung ist dies durch die Portierung der Docker-Policy von SELinux und die Auswertung über OSSIM als SIEM-System realisiert worden.

Die erreichten Endergebnisse dieser Masterthesis sind im Einzelnen:

- Untersuchung der Unterschiede von Server- und Betriebssystem-Virtualisierung (Kapitel 2.2), zur Analyse und Vergleich aktueller Virtualisierungstechnologien (Kapitel 2.3).
- Untersuchung der Sicherheitsaspekte des Linux-Kernels zur Trennung von Ressourcen und Prozessen (Kapitel 2.4) und Erstellung einer Entscheidungsmatrix mit relevanten Kriterien (Tabelle 7 auf Seite 15).
- Analyse der Architektur von KVM (Kapitel 3.1.1) und Docker (Kapitel 3.1.2) zur Untersuchung spezifischer Probleme der jeweiligen Virtualisierungs-Implementierung.
- Darstellung, dass ein erhöhter Schutzbedarf im virtualisierten Umfeld besteht (4.2), welcher anhand der STRIDE-Methode analysierte wurde.
- Erstellung zweier Schadensszenarien (Kapitel 4.3), um potentielle Angriffsziele zu untersuchen, wie durch übermäßige Ressourcennutzung oder dem Ausbruch aus einer beschränkten Docker-Umgebung zur Kompromittierung des Wirts oder anderer Gast-Container.
- Einsatz der vom BSI veröffentlichten IT-Grundschutz Vorgenweise als Methode zur Bedrohungsanalyse (Kapitel 4.4), zur Untersuchung der beiden Schadensszenarien.
- Abschätzung und Bewertung von Bedrohungen (Kapitel 5.1), um herauszufinden für welche Bereiche der virtuellen Umgebung zusätzliche Sicherheitsmaßnahmen erforderlich sind.
- Darstellung, dass eine Gefährdung ihre Ursache im Einsatz von DAC hat (Kapitel 5.2.2) und als Gegenmaßnahme zusätzlich MAC (5.2.2.1) eingesetzt werden sollte.
- Abgabe einer Empfehlung als Basis für Sicherheitsentscheidungen im Unternehmensumfeld (Kapitel 5.2.4) und Befürwortung der Nutzung der LSM-Schnittstelle auf Kernel-Ebene, da Sicherheitslücken nicht vermeidbar sind, sowie der Einsatz eines SIEM-Systems als essentielle Komponente, um aus Protokoll-Dateien heraus entstandene Sicherheitsverletzungen zu erkennen und zu melden.
- Implementierung des Prototyps mit der Portierung des SELinux-Regelwerks der Docker-Policy nach Debian, Protokollierung von Sicherheitsverletzung per Syslog und Auswertung und Korrelation per OSSIM (Kapitel 6.1).
- Erstellung eines Prototyps für die Verarbeitungskette zur Absicherung von Applikationen mit den Mitteln Virtualisierung, Kapselung der Unternehmensapplikation, Sicherheit und Überwachung (Kapitel 6.2).

7.1 Wissenschaftliche Sichtweise

In der Literatur werden die Bereiche der Virtualisierung und der Sicherheit unabhängig voneinander betrachtet. Fachliteratur zu den Virtualisierungs-Implementierungen Xen (vgl. Chisnall [2008]) und KVM (vgl. Michel Rode u. a. [2012]) haben den Fokus in der Konzeptvorstellung der Hypervisor-Varianten und Administration der virtualisierten Komponenten. Zur Erstellung sicherer Programme hat der Entwickler von Applikationen unter Linux alle Benutzereingaben zu prüfen, um eine Kompromittierung der Infrastruktur zu verhindern (vgl. Wheeler [1999; S. 46]). Einerseits ist die Server-Virtualisierung mit KVM auf Basis des Betriebssystems Red Hat auf die Sicherheit hin

bewertet worden nach Common Criteria (vgl. Bundesamt für Sicherheit in der Informationstechnik BSI-DSZ-CC-0754-2012 [2012]). Andererseits steht für die Betriebssystem-Virtualisierung auf Basis von Docker eine Untersuchung der Sicherheit aus. Ein methodisches Vorgehen nach Common Criteria zur Bewertung der Sicherheitseigenschaften von IT-Produkten, wäre mit dem Abarbeiten der Norm ISO 15408 in Form von Checklisten verbunden. Dabei bliebe allerdings der praktische Einsatz unberücksichtigt [Shuichiro Yamamoto/Tomoko Kaneko/Hidehiko Tanaka 2013; S. 331].

Die Masterthesis stellt die Verbindung zwischen der Sicherheit und Virtualisierung im Unternehmensumfeld her. Bedrohungen im praktischen Umfeld entstehen bei Verletzung der Sicherheit und den damit verbundenen Schutzzielen aus Vertraulichkeit, Integrität und Verfügbarkeit. Vorgehensweisen wie die Bedrohungsmatrix, die Bedrohungslandkarte oder die Risikomatrix helfen bei der Ermittlung generischer Bedrohungen, aber weniger bei realen Szenarien. Mit dem Bedrohungsbaum sind konkrete Abläufe und alternative Angriffspfade abbildbar. Das BSI liefert mit dem GSK neben Normen und Regeln, auch klare Maßnahmen zur Verbesserung der IT-Sicherheit im Unternehmen [Bundesamt für Sicherheit in der Informationstechnik IT-Grundschutz-Katalog 2015].

Für die Masterthesis wurde das Vorgehen nach BSI zur Bewertung von Risiken [Bundesamt für Sicherheit in der Informationstechnik 100-3 2008] in abgewandelter Form gewählt. Einerseits liefern zwei konkrete Schadensszenarien die Basis der Untersuchung der Virtualisierungs-Implementierungen KVM und Docker. Andererseits wurde anstatt der klassischen Schutzziele, mit der STRIDE-Methode (vgl. Microsoft Corporation [2005]) ein größeres Maß an möglichen Bedrohungen durch Sicherheitsverletzungen untersucht. Die Vorgehensweise nach BSI sieht vor, dass konkrete Bedrohungen und Gefährdungen aus der Praxis analysierbar sind, um in einer Auswertung entsprechende Gegenmaßnahmen festzulegen. Ein Nachteil der Vorgehensweise nach BSI ist aber, dass darüber hinausgehende Bedrohungen und Gefährdungen nicht ermittelt worden sind.

Ein zentrales Ergebnis aus wissenschaftlicher Sicht ist, dass die genutzte Sicherheitsfunktionen des Linux-Kernels (chroot, setuid und setgid, Capabilities, Cgroups, Namespaces, Seccomp) geeignet sind, um einen laufenden Prozess einschränken. Das Problem aber ist, dass nicht zwischen den root-Benutzern von Wirt- und Gast-System unterschieden werden kann. Die Ursache ist die gemeinsame Nutzung des gleichen Kernels, selbst wenn dieser per Namensraumisolierung getrennt wird (vgl. Kernel Documentation [2015b]). Linux nutzt als Sicherheitsmodell DAC zur Untergliederung der Lese-, Schreib- und Ausführungs-Zugriffe. Die Schwäche dieses Modells liegt einerseits darin, dass der Benutzer selbst die Zugriffsberechtigungen steuert und andererseits wird nicht zwischen Subjekt und Objekt beim Zugriff unterschieden. Mit MAC werden beide Probleme gelöst, indem die Zugangskontrolle zentral vom Linux-Kernel aus und mit Unterscheidung der Art des Zugriffs möglich ist. Bei Aktivierung nur erforderlicher Zugriffsberechtigungen, kann ein so geschütztes System auch vor zukünftigen Sicherheitslücken gesichert werden.

Ein weiteres Forschungsgebiet in diesem Zusammenhang wäre aus wissenschaftlicher Sicht die Ausweitung der Schadensszenarien, zur Bewertung aktueller Untersuchungen wie beispielsweise „The Memory Sinkhole“ was ein Ausbruch über eine dem „Ring -1“ vorgelagerte CPU-Einheit, dem System Management Mode (SMM), ermöglichen soll (vgl. Domas [2015]).

7.2 Praxisorientierte Sichtweise

Für Unternehmen ist in der Praxis die IT-Sicherheit (vgl. Eckert [2012]) und das damit verbundene IT-Sicherheitsmanagement nach ISO 27001 (vgl. Heinrich Kersten u. a. [2013]) eine Möglichkeit Sicherheitsmechanismen zu etablieren. Die ISO-Norm spezifiziert Anforderungen zur Sicherstellung der IT-Sicherheit, für den Lebenszyklus im Unternehmen mit den Phasen Planung und Konzeption, Beschaffung, Umsetzung und Betrieb. Ein Unternehmen kann sich von unabhängiger Stelle zertifizieren lassen, um die Einhaltung der Norm-Standards nachzuweisen.

Beim Vorgehen nach BSI ist die Zertifizierung nach ISO 27001 auf der Basis von IT-Grundschutz vorgesehen. Der IT-Grundschutz unterstützt die Identifizierung und Umsetzung von Sicherheitsmaßnahmen im Unternehmen [Bundesamt für Sicherheit in der Informationstechnik 100-2 2008]. Während die ISO-Norm dabei Ziele vorgibt was sicherzustellen ist, listet der BSI-Standard per Soll-Ist-Vergleich auf, wie die Sicherheit in der Praxis realisierbar ist. Dazu werden bekannte Bedrohun-

gen mit konkreten Maßnahmen im GSK verknüpft, um die Gefährdungen zu verhindern [Bundesamt für Sicherheit in der Informationstechnik IT-Grundschutz-Katalog 2015]. Für Bereiche, in denen Sicherheitsverletzungen existenziellen Auswirkungen auf das Unternehmen haben, liegt für die IT-Systeme ein besonders hoher Schutzbedarf vor.

In einer virtuellen Umgebung besteht kein physikalischer Schutz zwischen laufenden Maschinen, so dass per Software die Sicherheit gewährleistet werden muss. Durch die Verwendung der Betriebssystem-Virtualisierung Docker werden, durch die gemeinsame Nutzung des Linux-Kernels, die Software-Schichten weiter verringert. Im Gegensatz zur Server-Virtualisierung wird hierbei kein Hypervisor und kein separater Kernel für das Gast-System verwendet. Zusätzliche Bedrohungen können somit entstehen, bei der Verarbeitung von Firmengeheimnissen oder Patenten. Zur Erkennung von besonders gefährdeten Bereichen, ist die Durchführung einer Risikoanalyse inklusive einer Bedrohungsanalyse nach BSI erforderlich [Bundesamt für Sicherheit in der Informationstechnik 100-3 2008]. Auch wenn ein Unternehmen keine Zertifizierung nach ISO 27001 anstreben sollte, so bewirken die bereitgestellten Maßnahmen des BSI in ausformulierten sog. „Best Practice“ bei Umsetzung eine Verringerung des Risikos in diesen Bereichen.

Ein zentrales Ergebnis aus praktischer Sicht ist, dass die Linux-Sicherheitsfunktionen (chroot, setuid und setgid, Capabilities, Cgroups, Namespaces, Seccomp) begrenzt geeignet sind, um einen virtuellen Container abgetrennt und isoliert zu betreiben. Das durch den Benutzer veränderbare Sicherheitsmodell DAC von Linux zur Untergliederung der Lese-, Schreib- und Ausführungs-Zugriffe reicht nicht aus. Dies führt dazu, dass der Linux-Kernel keine Unterscheidung zwischen Datei- und Prozessrechten macht. Prozesse sind per Namensraumisolierung trennbar, allerdings kann der Kernel bei gleicher UID nicht den root-Benutzer von Wirt- und Gast-Systemen unterscheiden (vgl. Kernel Documentation [2015b]).

Das Sicherheitsmodell MAC funktioniert im Gegensatz dazu ohne Benutzerbeeinflussung und kann über den Kernel per LSM-Hook integriert werden. SELinux nutzt diesen Mechanismus über ein zentrales Regelwerk. Für die Masterthesis wurden die Regeln der Docker-Policy von Fedora 22 nach Debian UCS 4.0-3 migriert, sodass jeder Container-Prozess ein Label erhält. Anhand dieser Label sind Sicherheitsverletzungen erkennbar und werden im permissive-Modus protokolliert. Da SELinux für das ganze System aktiv wird, also nicht nur für einzelne Prozesse, ist über diese globale Einstellung der Betriebs-Modus steuerbar. Die Nutzung von SELinux zur Gewährleistung der Sicherheit hat allerdings Auswirkungen auf die Performance, so dauert beispielsweise das Öffnen und Schließen einer Datei 12 % länger als ohne SELinux [Ralf Spenneberg 2008; S. 51-53].

Eine weitere Aufgabe in diesem Zusammenhang wäre die Anpassung des gesamten Regelwerks der anderen SELinux-Policy-Module an UCS. Somit könnte SELinux im enforcing-Modus betrieben werden, um Sicherheitsverletzungen abzulehnen. Weitreichende Abhängigkeiten der Docker-Policy verursachen derzeit noch zahlreiche Einträge in der Protokoll-Datei, beispielsweise für das Overlay-Dateisystem beim Start eines Docker-Containers. Dies wäre eine weitere Teilaufgabe bei der vollständigen Migration für alle SELinux-Policy-Module.

Für ein vollständiges Sicherheitskonzept sind allerdings auch die untereinander ausgetauschten Informationen zwischen Gast-Container und Wirt-System relevant. So ist der Datenverkehr im Netzwerk per Firewall und der Zugriff auf den Verzeichnisdienst des Unternehmens per LDAP-ACL-Regelwerk zu steuern. Diese Herausforderungen sollen im parallelen Forschungsprojekt „SaferApps“ von Univention und Mitarbeitern des Forschungsbereichs Cyber-Physical Systems des Deutschen Forschungszentrums für Künstliche Intelligenz (DFKI) gelöst werden.

Ein weiteres Ergebnis aus praktischer Sicht ist, dass aus den protokollierten Sicherheitsverletzungen die genaue Sicherheitslücke eines potentiellen Angriffs ermittelbar ist. Dieser Masterthesis liegen zwei verschiedene Schadensszenarien zu Grunde. Im ersten Fall steht die Erkennung von übermäßiger Ressourcennutzung im Mittelpunkt und im zweiten Fall der Ausbruch aus einem Docker-Container mit anschließendem Zugriff auf die Wirt-Passwortdatei oder auf Daten eines anderen Containers. Über das SIEM-System OSSIM ist mit Hilfe von OSSEC das erste Szenario erkennbar. Ein installierter IDS-Sensor auf dem Wirt-System liefert fortwährend Statusmeldungen, so dass beispielsweise ein zu niedriger Arbeitsspeicher per Alarm-Benachrichtigung erkannt wird. Das

zweite Szenario setzt auf die Kernfunktionalität eines SIEM-Systems. Quellgeräte liefern Protokolldaten, die per Syntaxanalyse und Normalisierung mit Hilfe eines Regelwerks und Korrelation hin überprüft werden. Per Syslog über TCP werden die Protokoll-Dateien übertragen und die Sicherheitsverletzungen werden per regulärem Ausdruck gefiltert. Aus den eingehenden Event-Meldungen erzeugt das SIEM-System abhängig der Filterregeln und definierten Prioritäten einen Alarm. Zusätzlich wird beim Zugriff auf die Wirt-Passwortdatei ein Ticket erzeugt, welches dem Sicherheits-Team im Unternehmen zur Bearbeitung vorgelegt wird. So können Sicherheits-Meldungen effektiv verarbeitet werden, ohne das ein manuelles Durchsuchen der Protokoll-Dateien auf Unregelmäßigkeiten notwendig ist. Ein Nachteil durch die Verwendung eines SIEM-Systems, als Komplettpaket aus verschieden Sicherheitstools, wäre die Bindung an den Hersteller zur Weiterentwicklung der Software. Da der Quelltext bereitsteht, könnte das Projekt unabhängig fortgeführt werden.

Weitere Aufgaben in diesem Zusammenhang wäre die Zustellung eines Alarms per Mail zur Anbindung an ein externes System. Diese Informationen könnten an ein bestehendes Ticketsystem wie beispielsweise ORTS (Open Ticket Request System) übertragen werden, welches von einem Support-Team bearbeitet wird, um Störungen nachvollziehbar zu priorisieren und abzuarbeiten. Ein weiterer Gesichtspunkt wäre die bereits umgestellte Übertragung der Syslog-Daten von UDP auf TCP zusätzlich verschlüsselt zu übertragen, um den Inhalt der sensiblen Informationen aus der Protokollierung der Sicherheitsverletzung zu schützen.

Darüber hinaus könnten die Schadensszenarien erweitert werden, um mehr Vorfälle zu untersuchen. Einerseits wäre die quantitative Erhöhung der Szenarien eine Möglichkeit, um mehr Anwendungsfälle abzudecken. Andererseits wäre die qualitative Umarbeitung der Szenarien eine andere Möglichkeit, beispielsweise in Rücksprache mit Sicherheitsexperten oder basierend auf realen Angriffsszenarien.

7.3 Handlungsempfehlung

Die Vorgehensweise nach BSI unterstützt das Unternehmen bei der Zertifizierung nach ISO 27001 auf der Basis von IT-Grundschutz. Diese Methode ist gut geeignet, um Gefährdungen in der IT-Landschaft des Unternehmens zu erkennen und zu beheben. Mögliche Bedrohungen aus der Praxis und entsprechende ausformulierte Gegenmaßnahmen verbessern die IT-Sicherheit allumfassend in den Phasen der Planung und Konzeption, Beschaffung, Umsetzung und Betrieb der IT. Auch wenn ein Unternehmen die Einführung von ISO 27001 nicht anstrebt, ist die Methode geeignet, um sich Gefährdungen bewusst zu werden, bevor diese eintreten.

Die sichere Ausführung von fremden Applikationen in der Unternehmens-Infrastruktur steht im Mittelpunkt der Masterthesis. Der Aufbau der untersuchten Schadensszenarien ist gewählt worden, um die geringe Kopplung zum Wirt-System zu analysieren und um entstehende Sicherheitsverletzungen zu erkennen.

Für die Ausführung von Fremdapplikationen, eine vorbereitete Infrastruktur zur Erstellung, öffentlicher Bereitstellung und Betrieb von Containern, ist Docker gut geeignet. Die bestehenden Sicherheitsfunktionen des Linux-Kernels sind per LSM-Hook durch SELinux erweiterbar, um Container an unerwünschter Ressourcenverwendung zu hindern.

Bevor SELinux auf UCS genutzt wird, um Docker-Container abzusichern, ist zunächst das Regelwerk sämtlicher SELinux-Policy-Module weiter zu überarbeiten, um potentielle Sicherheitsverletzungen zu prüfen und ggf. freizuschalten. Der Hintergrund ist das SELinux unter Debian und UCS standardmäßig im permissive-Modus erkannte Sicherheitsverletzungen protokolliert. Nach gelagert sind diese Protokoll-Dateien zu analysieren, um die entsprechenden Regelwerke anzupassen. Erst im enforcing-Modus werden global alle erkannten Sicherheitsverletzungen abgelehnt, was bei unvollständigen Regelsätzen zu einem unvorhergesehenen Ablauf der Prozesse führt, wenn der Zugriff auf eine benötigte Ressource verweigert wird. In den eigenen Protokoll-Dateien der Programme ist hierbei lediglich eine Meldung in Form von „Zugriff verweigert“ hinterlegt, was keinen Rückschluss auf die tatsächliche Ursache bzw. SELinux zulässt.

Anschließend ist festzuhalten, dass über Docker bereitgestellte Container mit SELinux von unerwünschtem Zugriff auf Ressourcen des Wirt-Systems auf Kernel-Ebene geschützt werden.

8 Fazit

8.1 Rückblick

In dieser Masterthesis wurde gezeigt, wie ein sicherer Betrieb existierender Applikationen im Unternehmensumfeld mit Hilfe von Open Source-Werkzeugen möglich ist. In der Ausarbeitung des Theorie-Teils sind verschiedene Konzepte der Virtualisierung und deren Implementierung vorgestellt worden. Ein weiterer Themenbereich ist die Sicherheit des Linux-Kernels und wie laufende Prozesse in ihren Zugriffsberechtigungen beschränkt werden können.

In einer Bedrohungsanalyse wurden die beiden Themen von Virtualisierung und Sicherheit miteinander verbunden, um so die Virtualisierungs-Implementierung KVM und Docker im Unternehmensbetrieb zu untersuchen. Anhand von Schadensszenarien sind potentielle Angriffe, Auswirkungen und Gefährdungen festgestellt worden. Gegenmaßnahmen zur Gewährleistung der Sicherheit ist die Integration von Mandatory Access Control als Zugangskontrolle, zusätzlich zum bestehenden Discretionary Access Control unter Linux. Darüber hinaus müssen Protokoll-Dateien erfasst und per Echtzeitüberwachung ausgewertet werden, um Sicherheitsverletzungen zu erkennen mit Hilfe eines Security Information und Event Management (SIEM)-Systems.

Für den praktischen Teil wurde das Regelwerk von SELinux der Docker-Policy portiert und angepasst. In einer Testimplementierung ist in einem Prototyp die Virtualisierung von Docker mit der Sicherheit durch SELinux verbunden worden. Sicherheitsverletzungen werden von OSSIM erkannt und ausgewertet.

8.2 Schlussfolgerung

Während die Server-Virtualisierung KVM zur Vollvirtualisierung oder Emulation kompletter Hardware auf eine langjährige Reifung seit Veröffentlichung 2006 zurückblickt, befindet sich die Betriebssystem-Virtualisierung Docker zur Kapselung von Prozessen oder gleichartigen Betriebssystemen noch in der Entwicklung. Statt mit KVM Betriebssysteme auf Basis der Installations-CD des Herstellers auszuführen, bietet Docker im Unterschied dazu eine zentrale Applikationsentwicklerplattform. Diese ermöglicht es Container zu erstellen und ausführen oder fertig vorkonfigurierte lauffähige Applikationen zu verteilen. Der dabei von Gast- und Wirt-System gemeinsam genutzt Linux-Kernel, erfordert eine zusätzliche Absicherung auf Kernel-Ebene. Hierbei bietet sich die Nutzung von SELinux über die vorhandene LSM-Schnittstelle des Kernels an. OSSIM bewertet Protokoll-Dateien und zeigt entstandene Sicherheitsverletzungen per Alarm-Benachrichtigung an.

Die anfangs gestellte Forschungsfrage nach geeigneten Möglichkeiten zur Virtualisierung von Open Source-Applikationen und der Gewährleistung der Sicherheit im Unternehmensumfeld ist somit beantwortet.

8.3 Ausblick

Mit der Masterthesis werden Sicherheitsverletzungen auf Kernel-Ebene derzeit erkannt und protokolliert (permissive-Modus). Ein nächster Schritt wäre somit die Anpassung aller Policy-Module zur vollen Integration von SELinux (enforcing-Modus) in UCS, ohne dass ein Daemon-Prozess in seiner Tätigkeit durch fehlende Zugriffsrechte auf Ressourcen behindert wird. Ein Teilschritt davon ist die weitere Überprüfung von Abhängigkeiten von der Docker-Policy auf andere Policy-Module, um den Umfang der protokollierten Sicherheitsverletzungen zu minimieren. Diese Tätigkeiten könnten parallel zum Forschungsprojekt „SaferApps“ bis März 2017 umgesetzt werden.

Ein weiterer Aspekt wäre die Anbindung der protokollierten Sicherheitsverletzungen an die Prozesse im Unternehmen. Eine Teilaufgabe wäre die verschlüsselte Datenübertragung der Protokoll-Dateien von Syslog. Eine andere Aufgabe wäre die mögliche Anbindung der von SIEM erkannten Alarm-Benachrichtigung an ein im Unternehmen bereits bestehendes, externes Ticketsystem. So dass die Meldungen in der Praxis geschützt übertragen und von einem Supportteam bearbeitet werden können, um Störungen nachvollziehbar zu priorisieren und abzuarbeiten.

9 Verzeichnisse

9.1 Abbildungen

Abbildung 1: Schematische Darstellung der Schutzringe zur Trennung der Betriebsmodi...3
Abbildung 2: Architektur Hypervisor Typ 1 und Typ 2.....5
Abbildung 3: Native Betriebssystem-Kernel-Aufrufe, Para- und Hardwarevirtualisierung….6
Abbildung 4: Vergleich Hypervisor Typ 2 und Betriebssystem-Virtualisierung.....6
Abbildung 5: Virtualisierungsarchitektur von KVM 1.1.2 unter UCS 4.0-3 (Debian 7.8)…..18
Abbildung 6: Virtualisierungsarchitektur von Docker 1.7.0 unter UCS 4.0-3 (Debian 7.8)..20
Abbildung 7: Systemkonzept der virtualisierten Komponenten.....22
Abbildung 8: Aufbau Risikoanalyse auf Basis IT-Grundschutz (Quelle: Bundesamt für Sicherheit in der Informationstechnik 100-3 [2008; S. 5]).....27
Abbildung 9: Bell-LaPadula Multi Level Security gegenüber Domain Type Enforcement...42
Abbildung 10: LSM Hook Architektur (Quelle: Caplan [2007; S. 40]).....44
Abbildung 11: Verarbeitungskette von SIEM (Quelle: Miller u. a. [2011; S. 78]).....47
Abbildung 12: OSSIM Architektur (Quelle: Miller u. a. [2011; S. 148]).....49
Abbildung 13: Angepasstes Systemkonzept.....53
Abbildung 14: Prototypisches Unternehmensnetzwerk.....54
Abbildung 15: OSSIM: Event-Einstellung bei niedrigerem Arbeitsspeicher.....58
Abbildung 16: OSSIM: Event-Policy bei Zugriff auf die Passwortdatei.....59
Abbildung 17: OSSIM: eintreffende normalisierte Event-Nachrichten.....60
Abbildung 18: OSSIM: Alarm-Übersicht.....60
Abbildung 19: OSSIM: Ticket-Übersicht.....60

9.2 Tabellen

Tabelle 1: Hypervisor-Virtualisierung im Vergleich zum Container-System.....7
Tabelle 2: Server-Virtualisierung (Stand: 17.04.2015).....8
Tabelle 3: Betriebssystem-Virtualisierung (Stand: 10.07.2015).....9
Tabelle 4: Berechtigungen mit Capabilities.....12
Tabelle 5: Ressourcenbegrenzung mit Cgroups.....13
Tabelle 6: Namensraumisolierung mit Namespaces.....14
Tabelle 7: Virtualisierungs-Implementierungen und Sicherheitsaspekte (Stand: 10.07.15) 15
Tabelle 8: Performance-Vergleich KVM und Docker [Seo u. a. 2014] mit Auswertung.....21
Tabelle 9: Bedrohungen nach STRIDE-Modell mit verletzten Schutzzielen.....29
Tabelle 10: Schadensszenario 1: Auslasten von Wirt durch Gast.....30
Tabelle 11: Schadensszenario 2: Ausbruch aus Gast-Container.....30
Tabelle 12: Bedrohungsübersicht Gast-VM.....32
Tabelle 13: Bedrohungsübersicht Gast-Container.....32
Tabelle 14: Bedrohungsübersicht Wirt.....33
Tabelle 15: Bedrohungsübersicht der Kommunikation Gast – Wirt.....33
Tabelle 16: Bedrohungsübersicht der Kommunikation Wirt – Gast.....34
Tabelle 17: Bedrohungsübersicht der Kommunikation von außen – Wirt.....34
Tabelle 18: Ergänzung der Bedrohungsübersicht Gast-Container.....35
Tabelle 19: Ergänzung der Bedrohungsübersicht der Kommunikation Wirt – Gast.....36
Tabelle 20: Ergänzung der Bedrohungsübersicht der Kommunikation Gast – Wirt.....36
Tabelle 21: Bedrohungsauswertung Gast-VM.....37
Tabelle 22: Bedrohungsauswertung Gast-Container.....37
Tabelle 23: Bedrohungsauswertung Wirt.....38

Tabelle 24: Bedrohungsauswertung der Kommunikation Gast – Wirt.....39
Tabelle 25: Bedrohungsauswertung der Kommunikation Wirt – Gast.....39
Tabelle 26: Bedrohungsauswertung der Kommunikation von außen – Wirt.....40
Tabelle 27: Verbleibende Bedrohung Gast-VM.....50
Tabelle 28: Verbleibende Bedrohung Gast-Container.....50
Tabelle 29: Verbleibende Bedrohung der Kommunikation Gast – Wirt.....51
Tabelle 30: Verbleibende Bedrohung der Kommunikation Wirt – Gast.....52
Tabelle 31: Verbleibende Bedrohung der Kommunikation von außen – Wirt.....52
Tabelle 32: Arbeitspakete der priorisierten Anforderungen des praktischen Teils.....54

9.3 Quelltexte

Listing 1: Ressourcengrenzen mit ulimit.....10
Listing 2: Programm ping mit setuid-Bit.....11
Listing 3: Programm ping mit Capability anstatt setuid-Bit.....12
Listing 4: Programm unshare zur Abtrennung des Netzwerk-Namensraums.....14
Listing 5: Capabilities außerhalb und innerhalb eines Containers.....35
Listing 6: Programm id mit Sicherheitskontext.....45
Listing 7: Ausschnitt der Prozessliste mit Sicherheitskontext.....45
Listing 8: Docker Schadensszenario 1.....55
Listing 9: Docker Schadensszenario 2.....55
Listing 10: Gast-VM: Dateiausschnitt /etc/apt/sources.list.....56
Listing 11: Gast-VM: Installation von SELinux.....56
Listing 12: Gast-VM: Dateiausschnitt /etc/default/docker.....56
Listing 13: Ausschnitt der Prozessliste mit aktiver Docker-Policy.....56
Listing 14: Gast-VM: Installation von audit und fortlaufende Anzeige der Sicherheitsverletzungen.....56
Listing 15: Gast-VM: Dateiinhalt /etc/audisp/plugins.d/syslog.conf.....57
Listing 16: Gast-VM: Dateiinhalt /etc/rsyslog.d/local-forward.conf.....57
Listing 17: SIEM-VM: Dateiausschnitt /etc/ossim/firewall_include.....57
Listing 18: SIEM-VM: Übertragene Syslog-Nachricht der Audit-Protokoll-Datei.....58
Listing 19: SIEM-VM: Dateiausschnitt /etc/ossim/agent/plugins/syslog.cfg.....59
Listing 20: SIEM-VM: Ausschnitt der Syslog-Nachricht bei Zugriff auf fremde Daten.....59
Listing 21: SIEM-VM: Dateiausschnitt /etc/ossim/agent/plugins/syslog.cfg.....59

9.4 Literatur

Andre, J. (2014): Docker breakout exploit analysis. Online unter URL: https://medium.com/@fun_cuddles/docker-breakout-exploit-analysis-a274fff0e6b3 [Stand: 22.07.2015].

Bitkom e.V. (2014): Leitfaden Server-Virtualisierung. 02. Aug. 2014. Online unter URL: https://www.bitkom.org/Bitkom/Publikationen/Publikation_4361.html [Stand: 01.07.2015].

Bradner, S. (1997): Key words for use in RFCs to Indicate Requirement Levels. Online unter URL: http://tools.ietf.org/html/rfc2119 [Stand: 02.09.2015].

Bui, T. (2014): Analysis of Docker Security. In: *arXiv preprint* arXiv:1501.02967, S. 1-7.

Bundesamt für Sicherheit in der Informationstechnik 100-1 (2009): BSI-Standard 100-1: Managementsysteme für Informationssicherheit (ISMS). 15. Juni 2009. Online unter URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/ITGrundschutzstandards/standard_1001_pdf.pdf [Stand: 26.06.2015].

Bundesamt für Sicherheit in der Informationstechnik 100-2 (2008): BSI-Standard 100-2:

IT-Grundschutz-Vorgehensweise. 23. Juni 2008. Online unter URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/ITGrundschutzstandards/standard_1002_pdf.pdf [Stand: 26.06.2015].

Bundesamt für Sicherheit in der Informationstechnik 100-3 (2008): BSI-Standard 100-3: Risikoanalyse auf der Basis von IT-Grundschutz. 23. Juni 2008. Online unter URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/ITGrundschutzstandards/standard_1003_pdf.pdf [Stand: 26.06.2015].

Bundesamt für Sicherheit in der Informationstechnik B 3.304 (2008): B 3.304: Baustein Virtualisierung. 6. März 2014. Online unter URL: https://www.bsi.bund.de/DE/Themen/CloudComputing/Dossiers/Anbieter/AnbieterProfessionals/AnbieterProfessionals.html?notFirst=true&docId=4966040 [Stand: 01.07.2015].

Bundesamt für Sicherheit in der Informationstechnik BSI-DSZ-CC-0754-2012 (2012): Red Hat Enterprise Linux Version 6.2 with KVM Virtualization for x86 Architectures. 23. Oktober 2012. Online unter URL: https://www.bsi.bund.de/SharedDocs/Zertifikate/CC/Betriebssysteme/0754.html [Stand: 01.07.2015].

Bundesamt für Sicherheit in der Informationstechnik IT-Grundschutz-Katalog (2015): BSI: IT-Grundschutz-Kataloge 14. Ergänzungslieferung. 10. März 2015. Online unter URL: https://gsb.download.bva.bund.de/BSI/ITGSK/IT-Grundschutz-Kataloge_2014_EL14_DE.pdf [Stand: 27.08.2015].

Bundesamt für Sicherheit in der Informationstechnik OpenVAS (2015): BSI: Open Vulnerability Assessment System (OpenVAS). Online unter URL: https://www.bsi.bund.de/DE/Themen/ProdukteTools/OpenVAS/OpenVAS_node.html [Stand: 19.08.2015].

Caplan, F.M.K.M.D. (2007): *SELinux by example: using security enhanced Linux*. Prentice Hall, Upper Saddle River, NJ [u.a.].

Chisnall, D. (2008): *The Definitive Guide to the Xen Hypervisor*. Pearson Education.

Coverity (2013): Scan-Report. 3. Mai 2013. Online unter URL: http://wpcme.coverity.com/wp-content/uploads/2012-Coverity-Scan-Report.pdf [Stand: 25.06.2015].

Dennis Zimmer, Matthias Wessner, Thomas Joos, Sebastian Hetze, Falko Gräfe, Bertram Wöhrmann, and Thorsten Scherf (2012): *Virtualisierung: Betrieb und Management virtualisierter Infrastrukturen*. Heinemann, München.

Domas, C. (2015): The Memory Sinkhole. In: *Blackhat* 2015, S. 1-2.

Eckert, C. (2012): *IT-Sicherheit: Konzepte - Verfahren - Protokolle*. Oldenbourg, München.

Felter, W., Ferreira, A., Rajamony, R., and Rubio, J. (2014): An Updated Performance Comparison of Virtual Machines and Linux Containers. In: *technology* 28, S. 1-32.

Friedl, S. (2002): Best Practices for UNIX chroot() Operations. Online unter URL: http://unixwiz.net/techtips/chroot-practices.html [Stand: 02.06.2015].

Gabriel Avramescu, Mihai Bucicoiu, Daniel Rosner, and Nicolae Tăpuş (2013): Guidelines for Discovering and Improving Application Security. In: *Cost-Based Quantile Query Processing in Wireless Sensor Networks* 19th International Conference on Control Systems and Computer Science, S. 560-565.

General Commands Manual (2014): bash - GNU Bourne-Again SHell. 02. Feb. 2014. Online unter URL: http://man7.org/linux/man-pages/man1/bash.1.html [Stand: 14.07.2015].

Gerhards, R. (2009): The syslog protocol. Online unter URL: http://tools.ietf.org/html/rfc5424 [Stand: 02.09.2015].

Heinrich Kersten, Jürgen Reuter, Klaus-Werner Schröder, Heinrich Kersten, and Klaus-Dieter Wolfenstetter (2013): *IT-Sicherheitsmanagement nach ISO 27001 und Grundschutz: Der Weg zur Zertifizierung*. Springer, Wiesbaden.

Kernel Documentation (2015a): Control Groups definition. 29. Mai 2015. Online unter URL: https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt [Stand: 02.06.2015].

Kernel Documentation (2015b): Namespaces compatibility list. 16. Nov 2007. Online unter URL: https://www.kernel.org/doc/Documentation/namespaces/compatibility-list.txt [Stand: 02.06.2015].

Kretschmer, T. (2010): LIFE 2 – Vernetztes Arbeiten in Wirtschaft und Gesellschaft. Online unter URL: http://www.studie-life.de/life-studien/vernetztes-arbeiten/ [Stand: 22.07.2015].

Lampson, B.W. (1971): Protection. In: *Proceedings of the 5th Princeton Conference on Information Sciences and Systems* Reprinted in ACM Operating Systems Rev. 8, S. 437.

Langley, A., Pironti, A., Barnes, R., and Thomson, M. (2015): Deprecating Secure Sockets Layer Version 3.0. Online unter URL: http://tools.ietf.org/html/rfc7568 [Stand: 08.07.2015].

Lindqvist, H. (2006): Mandatory access control. In: *Master's Thesis in Computing Science, Umea University, Department of Computing Science, SE-901* 87, S. 25.

Linux Programmer's Manual (2015): capabilities - overview of Linux capabilities. 07. Mai 2015. Online unter URL: http://man7.org/linux/man-pages/man7/capabilities.7.html [Stand: 02.06.2015].

Michel Rode, Jan Sperling, Andreas Steil, and Christoph Arnold (2012): *KVM Best Practices: Virtualisierungslösungen für den Enterprise-Bereich*. dpunkt-Verl., Heidelberg.

Microsoft Corporation (2005): The STRIDE Threat Model. 2005. Online unter URL: https://msdn.-microsoft.com/en-us/library/ee823878%28v=cs.20%29.aspx [Stand: 09.07.2015].

Miller, D., Harris, S., Harper, A., VanDyke, S., and Blask, C. (2011): *Security information and event management (SIEM) implementation: [enable real-time monitoring and analysis of security events ; respond quickly to attacks, log security data, and generate compliance reports ; get details on leading SIEM products - AlienVault OSSIM, Cisco-MARS, ArcSight ESM, and Q1 Labs QRadar]*. McGraw-Hill, New York, NY [u.a.].

Mosaic Security Research (2015): Log Management & Security Information and Event Management (SIEM). 18. jan. 2012. Online unter URL: http://mosaicsecurity.com/categories/85-log-management-security-information-and-event-management [Stand: 19.08.2015].

Müller, K.-R. (2014): *IT-Sicherheit mit System: integratives IT-Sicherheits-, Kontinuitäts- und Risikomanagement - sichere Anwendungen - Standards und Practices*. Springer Vieweg, Wiesbaden.

Ralf Spenneberg (2008): *SELinux & AppArmor: Mandatory Access Control für Linux einsetzen und verwalten*. Addison Wesley, München [u.a.].

Seo, K.-T., Hwang, H.-S., Moon, I.-Y., Kwon, O.-Y., and Kim, B.-J. (2014): Performance Comparison Analysis of Linux Container and Virtual Machine for Building Cloud. In: *Advanced Science and Technology Letters* Vol.66 (Networking and Communication 2014), S. 105-111.

Shuichiro Yamamoto, Tomoko Kaneko, and Hidehiko Tanaka (2013): Information and Communication Technology: International Conference, ICT-EurAsia 2013, Yogyakarta, Indonesia, March 25-29, 2013. Proceedings: A Proposal on Security Case Based on Common Criteria. In: *Lecture Notes in Computer Science* 7804, S. 331-336.

Thorsten Reitz, and Juan Villar (2011): *Cloud computing: Chancen und Risiken aus technischer und unternehmerischer Sicht*. Hanser, München.

Univention GmbH (2015): Univention GmbH "Über uns" Selbstdarstellung. Online unter URL: https://www.univention.de/ueber-uns/univention/ [Stand: 20.07.2015].

V. Anitha, and N. Keerthana (2013): An Overview: Security in Virtualization Technology. In: *Journal of Artificial Intelligence* 6 (1), S. 112-116.

Wheeler, D.A. (1999): Secure programming for Linux and Unix HOWTO. Online unter URL: http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO.pdf [Stand: 02.06.2015].

Eidesstattliche Erklärung

Ich, Andreas Peichert, Matrikelnummer 396793, versichere, dass ich die vorliegende Masterthesis selbstständig und ohne unzulässige fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die aus den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Bremen, den 28. September 2015

….....

Andreas Peichert

Anhänge

Docker-Policy und Debian-Anpassungen

Die migrierte und angepasste Docker-Policy befindet sich im internen SVN von Univention

`svn+ssh://billy.knut.univention.de/var/svn/patches/refpolicy`

unter dem Pfad

`4.0-0-0-ucs/2:2.20140421-9-selinux/adjusted_docker_policies_from_fedora_22.patch`

sowie zusätzlich unter

`http://www.andreas.peichert.com/files/adjusted_docker_policies_from_fedora_22.patch`

sha256-Prüfsumme

4c7c6ea3ca3f58888dcb908e78a1c082f3bbac1585ba77c62965864f7195dc85

Nachfolgend ist der manueller Build-Vorgang, unabhängig des UCS-Build-Systems, beschrieben.

Zuerst wird die „Debian source control“-Datei heruntergeladen. Hiermit werden automatisch auch die Original-Quelltexte der „SELinux Reference Policy“ (refpolicy_2.20140421.orig.tar.bz2) und die spezifischen Anpassungen von Debian (refpolicy_2.20140421-9.debian.tar.xz) heruntergeladen:

```
dget http://http.debian.net/debian/pool/main/r/refpolicy/refpolicy_2.20140421-9.dsc
```

Auspacken und Verschmelzen des originalen Quelltextes mit den Debian-Anpassungen:

```
dpkg-source -x refpolicy_2.20140421-9.dsc
```

Im Folgenden findet der Patch Anwendung. Hierbei wird die Datei für den automatischen Kontextwechsel (`lxc_context`) angelegt und der Build-Prozess für Docker angepasst. Außerdem werden zwei zusätzliche Debian-Patch-Dateien angelegt, welche in den Build-Prozess einfließen:

```
patch -p0 < ../adjusted_docker_policies_from_fedora_22.patch
```

Wechseln in das Quelltext-Verzeichnis der SELinux-Policy:

```
cd refpolicy-2.20140421/
```

Anwenden des Patches der migrierten Docker-Policy (2000-docker-policy.patch):

```
quilt push
```

Anwenden des Patches der Docker-Anpassungen für Debian und UCS (2000-docker-fixes.patch):

```
quilt push
```

Bauen der neuen Debian-Pakete:

```
dpkg-buildpackage
```

Installation des Debian-Paketes, wobei der Dateiname vom Build-Datum abhängig ist:

```
dpkg -i ../selinux-policy-default_2.20140421-9.12.201509231038_all.deb
```

OSSIM Plugin-Erweiterung

Listing: SIEM-VM: Dateiinhalt /etc/ossim/agent/plugins/syslog.cfg

```
# Alienvault plugin
# Author: Alienvault Team at devel@alienvault.com
# Plugin syslog id:4007 version: 0.0.2
# Last modification: 2015-05-13 16:11
#
# Plugin Selection Info:
# Syslog:Syslog:-
#
# END-HEADER
# Accepted products:
# syslog - syslog -
# Description:

[DEFAULT]
plugin_id=4007

[config]
type=detector
enable=yes

source=log
location=/var/log/syslog

create_file=true

process=
start=no
stop=no
startup=
shutdown=

[syslog – datamining]
event_type=event
regexp="^(?P<logline>(\SYSLOG_DATE)\s+(?P<sensor>\S+)\s+(?P<source>\S+)\s+(?
P<generator>[^\[]*)\[(?P<pid>\d+)\]:(?P<logged_event>.*))$"
device={resolv($sensor)}
date={normalize_date($1)}
plugin_sid=1
src_ip={resolv($sensor)}
userdata1={md5sum($logline)}
userdata3={$generator}
userdata4={$logged_event}
userdata5={$pid}

[syslog - datamining – all]
event_type=event
regexp="^(?P<logline>(?P<date>\SYSLOG_DATE)\s+(?P<sensor>\S+)\s+(?P<source>\S+)\s+(?
P<logged_event>.*)$)"
device={resolv($sensor)}
date={normalize_date($date)}
plugin_sid=1
src_ip={resolv($sensor)}
userdata1={md5sum($logline)}
userdata3={$logged_event}

# audispd: container read shadowed password file
#
# type=AVC msg=audit(1441364081.670:127): avc:  denied  { read } for  pid=4594
```

```
comm="cat" name="shadow" dev="dm-0" ino=1571194
scontext=system_u:system_r:svirt_lxc_net_t:s0:c31,c126
tcontext=system_u:object_r:shadow_t:s0 tclass=file permissive=1
#
[syslog - audit – shadow]
event_type=event
regexp="^(?P<logline>(?P<date>\SYSLOG_DATE)\s+(?P<sensor>\S+)\s+(?P<source>\S+)\s+(?
P<logged_event>.*audit.*read.*svirt_lxc_net_t.*shadow_t.*)$)"
device={resolv($sensor)}
date={normalize_date($date)}
plugin_sid=2
src_ip={resolv($sensor)}
userdata1={md5sum($logline)}
userdata3={$logged_event}

# audispd: container read file of other container
#
# type=AVC msg=audit(1441875676.904:4862): avc:  denied  { read } for  pid=1074
comm="cat" name="secret" dev="vda1" ino=403101
scontext=system_u:system_r:svirt_lxc_net_t:s0:c355,c977
tcontext=system_u:object_r:docker_var_lib_t:s0:c106,c852 tclass=file permissive=1
#
[syslog - audit – docker_var_lib_t]
event_type=event
regexp="^(?P<logline>(?P<date>\SYSLOG_DATE)\s+(?P<sensor>\S+)\s+(?P<source>\S+)\s+(?
P<logged_event>.*audit.*read.*svirt_lxc_net_t:s0:(?
P<mcs>c\d{1,4},c\d{1,4})\s.*docker_var_lib_t:s0:(?!(?P=mcs)).*)$)"
device={resolv($sensor)}
date={normalize_date($date)}
plugin_sid=3
src_ip={resolv($sensor)}
userdata1={md5sum($logline)}
userdata3={$logged_event}
```

Listing: SIEM-VM: Dateiinhalt /usr/share/doc/ossim-mysql/contrib/plugins/syslog.sql

```
-- syslog
-- plugin_id: 4007

DELETE FROM plugin WHERE id = "4007";
DELETE FROM plugin_sid where plugin_id = "4007";

INSERT IGNORE INTO plugin (id, type, name, description) VALUES (4007, 1, 'syslog',
'Syslog plugin with md5 checksum logging');

INSERT IGNORE INTO plugin_sid (plugin_id, sid, category_id, class_id, name,
priority, reliability) VALUES (4007, 1, NULL, NULL, 'Syslog: syslog entry', 0, 1);
INSERT IGNORE INTO plugin_sid (plugin_id, sid, category_id, class_id, name,
priority, reliability) VALUES (4007, 2, NULL, NULL, 'audispd: container read
shadowed password file', 5, 3);
INSERT IGNORE INTO plugin_sid (plugin_id, sid, category_id, class_id, name,
priority, reliability) VALUES (4007, 3, NULL, NULL, 'audispd: container read file of
other container', 5, 3);
```

Listing: SIEM-VM: Installation und Aktivierung des angepassten Syslog-Plugins

```
# ossim-db < /usr/share/doc/ossim-mysql/contrib/plugins/syslog.sql
# /etc/init.d/ossim-server restart
# /etc/init.d/ossim-agent restart
```