Virtual IT-Security Architectures

Best Practice Report
im Verbundprojekt VISA

DOKUMENTINFORMATIONEN	
TYP	Bericht
TITEL	Best-Practise-Dokument
DATUM	19.12.2013
ARBEITSPAKET	AP5.1 und AP5.2
FÖRDERKENNZEICHEN	01BY1160

ESUKOM-KONSORTIUM	
KOORDINATOR	DECOIT GmbH
PARTNER 1	Collax GmbH
PARTNER 2	IT-Security@Work GmbH
PARTNER 3	Fachhochschule Dortmund
PARTNER 4	Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e. V.
PARTNER 5	National ICT Australia (NICTA) Limited

DOKUMENTSTATUS		
AKTION	**DURCH**	**DATUM**
EINGEREICHT	DECOIT GmbH	01.10.2013
AP-LEITER	DECOIT GmbH (AP5.1 und AP5.2)	
GENEHMIGT	DECOIT GmbH	19.12.2013

ÄNDERUNGSSHISTORIE			
DATUM	**VERSION**	**AUTOR**	**KOMMENTAR**
02.10.2013	0.1	Marcel Jahnke	Initiale Erstellung des Dokuments
18.11.2013	0.2	Henk Birkholz	„Konfiguration IO-Tool“ hinzugefügt
20.11.2013	0.3	Prof. Dr. Kai-Oliver Detken	Ergänzung der Einleitung, Beschreibung des Projektes hinzugefügt, einzelne Module ausgeführt, Literaturhinweise ergänzt
27.11.2013	0.4	Prof. Dr. Kai-Oliver Detken	Topologie-Editoren beschrieben, Problembeschreibung angefangen sowie die Zusammenfassung
29.11.2013	0.5	Prof. Dr. Kai-Oliver Detken	Problembeschreibung der einzelnen Entwicklungen zu Ende geschrieben und Fazit formuliert
12.12.2013	0.6	Prof. Dr. Kai-Oliver Detken	Ergänzung des Demonstrator-Kapitels, Ergänzung der Testergebnisse und schreiben des Fazits
19.12.2013	0.7	Marcel Jahnke	Ergänzungen im Kapitel „Probleme bei der Umstellung“

KONTAKTINFORMATIONEN				
NAME	**ORGANISATION**	**EMAIL**	**TEL**	**FAX**
K.-O. Detken	DECOIT GmbH	detken@decoit.de	0421-596064-01	-09
Marcel Jahnke	DECOIT GmbH	jahnke@decoit.de	0421-596064-15	-09
Henk Birkholz	Fraunhofer SIT	henk.birkholz@sit.fraunhofer.de	06151-869-163	-224
Nicolai Kuntze	Fraunhofer SIT	kuntze@sit.fraunhofer.de	06151-869-276	-224
Evren Eren	Fachhochschule Dortmund	eren@fh-dortmund.de	0231-755-6776	-6710
Falk Krämer	Collax GmbH	falk.kraemer@collax.com	089-990157-41	-11
Marion Steiner	IT-Security@Work GmbH	marion.steiner@isw-online.de	06103-37416-00	-99
Najib Alkilani Alkadri	IT-Security@Work GmbH	Najib.AlkilaniAlkadri@isw-online.de	06103-37416-00	-99
Christoph Dwertmann	National ICT Australia	Christoph.dwertmann@nicta.com.au	+61 2 9376 20-85	-23

Inhalt

1 Einleitung

Dieses Dokument beschreibt die Erfahrungen und die daraus resultierenden Best-Practice-Richtlinien, welche sich innerhalb des vom Bundesministerium für Bildung und Forschung (BMBF) geförderten Verbundprojektes VISA (Laufzeit: 2011-2013) im Rahmen der Entwicklungsarbeiten ergeben haben. Zusätzlich wird aufgezeigt, wie der finale Demonstrator, welcher alle im Projekt entwickelten Komponenten umfasst, konzipiert und technisch umgesetzt wurde. Abschließend werden die Ergebnisse der einzelnen Tests aufgeführt und zusammengefasst:

a. In Kapitel 2 dieses Dokuments wird eine kurze, allgemeine Übersicht über die Inhalte und Ziele des VISA-Projektes gegeben.

b. Im darauffolgendem Kapitel 3 werden die auftretenden Probleme und Unklarheiten bzgl. der Entwicklungs- und Implementierungsarbeiten aufgezeigt sowie die Ansätze, welches zur Lösung der Probleme zur Anwendung kamen.

c. In Kapitel 4 folgt eine Beschreibung des finalen Demonstrators, welcher im Rahmen des Projekts umgesetzt wurde. Dabei werden die Konzepte und eingesetzten Technologien erläutert, auf deren Grundlage die Realisierung des Demonstrators letztendlich erfolgte sowie eine kurze Anleitung zum Aufsetzen und zur Benutzung des Demonstrators gegeben.

d. Das Kapitel 5 zeigt die einzelnen Ergebnisse der durchgeführten Tests auf, die zum Projektende durchgeführt wurden.

e. Das Kapitel 6 schließt diesen Bericht mit einem Fazit und einer Bewertung der erreichten Ziele dieses Projektes ab.

Das VISA-Projekt ist offiziell im September 2013 beendet worden, nach einer Laufzeit von zwei Jahren und zwei Monaten. Das Projekt hat in dieser Laufzeit eine Vielzahl von Entwicklungen und neuen Ansätzen umgesetzt bzw. vorangetrieben. Dieser Best-Practise-Bericht soll daher auch externen Interessenten die Möglichkeit eröffnen an diesen Ergebnissen teilzuhaben und über den Demonstrator erste Erfahrungen zu sammeln.

2 Beschreibung des VISA Projektes

IT-Infrastrukturen sind mittlerweile auch schon in kleinen und mittelgroßen Unternehmen (KMU) sehr komplex. Neben den verschiedenen Typen von Rechnern (Desktopcomputer, Laptops, Server,...), Peripherie (z.B. Multifunktionsdrucker) und funktionalen Netzwerkkomponenten (Router, Switches,...) wächst die Komplexität durch unterschiedliche Sicherheitskomponenten (Firewall, Intrusion Detection, ...). Die Auswirkungen von Änderungen an solchen Infrastrukturen sind oft erst im Echtbetrieb zu erkennen und die Integration neuer Sicherheitskomponenten erfordert oft die Integration neuer Hardware und den Umbau der Netztopologie, der ohne genaue Kenntnis der Auswirkung umgesetzt werden muss.

Darüber hinaus ist das Absichern von Einzelkomponenten, insbesondere im Verbund mit anderen Komponenten, für die Integration in der Unternehmenstopologie nach BSI IT-Grundschutz sowie ISO 27001, nach denen ein ISMS (Informations-Sicherheits-Management-System) aufgebaut und betrieben werden kann, kein einfaches Unterfangen. Dies ist auch für Conformance und Compliance für Regulierungsanforderungen wie z.B. Basel II essentiell. Da immer mehr KMUs ein profundes IT-Risikomanagement vorweisen müssen, muss nachweisbar sein, dass die IT-Infrastruktur über ausreichende Schutzmechanismen (Virenabwehr, Zugangssteuerung, Schutz von Daten über Zugriffsrechte, IT-Notfallplanung und -regelung) verfügt.

Vor diesem Hintergrund muss für Klein- und Mittelständische Unternehmen (KMUs) der Umgang mit IT-Infrastrukturen vereinfacht werden, da diese wenig Personalressourcen und Know-how für das operative IT-Management vorhalten können. Dies wäre durch den Einsatz von Virtualisierung und Simulation von IT-Infrastrukturen einerseits, und durch die Realisierung KMU gerechter Darstellung und Bedienbarkeit der resultierenden Sicherheitsfunktionalität anderseits zu gewährleisten.

Es war daher das Ziel des Projektes VISA, durch Nutzung von Virtualisierungstechnologien das Management von IT-Infrastrukturen insbesondere der Sicherheitskomponenten zu erleichtern und zu unterstützen. Diese Unterstützung basierte auf drei Kerntechnologien:

a. **Simulation und Evaluierung** der gesamten IT-Infrastruktur in virtuellen Umgebungen.

b. Realisierung von Sicherheitsanwendungen als virtuelle Komponenten, sog. **Virtual Security Appliances (VSA)**.

c. Vereinfachung und Nachweisbarkeit der Einhaltung von IT-Standards, IT-Security- und **Compliance**-Anforderungen durch geeignet entwickelte VSAs als fertig verwendbare IT-Bausteine

Durch das VISA-Rahmenwerk wurde der passgenaue und vereinfachte Einsatz von Sicherheitsanwendungen auf Basis von Virtual Security Appliances (VSA) ermöglicht.

Durch die umfassende Emulation der IT-Infrastrukturen konnten die betriebsrelevanten Parameter und die Integrationspunkte der VSAs bereits in der virtuellen Umgebung identifiziert und der Einsatz erprobt werden. Erfolgreich getestete VSAs konnte man dann direkt, ohne Änderung in die existierende Infrastruktur, zum Einsatz bringen.

2.1 Schwerpunkte und gemeinsame Ziele

Heutige IT-Infrastrukturen setzen sich aus einer Vielzahl von Software- und Hardwarekomponenten zusammen, die in ihrem Zusammenspiel die eigentliche Wirkfunktionalität bilden. Die überprüfbare und durchgehende Sicherheit solcher heterogener IT-Infrastrukturen ist ein wichtiges Ziel in der Konzeption und Weiterentwicklung dieser Strukturen. Auch KMUs betreiben bereits heute komplexe IT-Infrastrukturen bzw. sind auf diese angewiesen. Diese Komplexität erfordert eine genaue Betrachtung der Sicherheitsfunktionalitäten und die Überprüfung der Konformität zu den gewünschten Eigenschaften.

Das Rahmenwerk des VISA-Projektes (Virtual IT Security Architectures) hat daher einen ganzheitlichen Ansatz für die Planung und Erprobung von IT-Infrastrukturen implementiert. Durch die Kombination der Modellierung und formalen Beschreibung von IT-Infrastrukturen auf der einen Seite sowie der Evaluation der IT-Infrastrukturen in virtuellen Umgebungen anhand verschiedener, definierter Kriterien auf der anderen Seite wird es KMUs nun ermöglicht, Kosten und Eigenschaften der IT-Investition besser abschätzen zu können.

Das VISA-Rahmenwerk wurde in einem teilautomatisierten Evaluationszyklus realisiert, in dem auf abstraktem Niveau zunächst Modelle von Infrastrukturen erstellt wurden. Hierzu wurden typische Szenarien für KMU-basierte IT-Infrastrukturen identifiziert, woraus typische Infrastrukturkomponenten erkannt wurden. Ein wichtiger Schritt im Evaluationszyklus war die Überführung der Modelle in praktisch ausführbare Systeme, an denen direkt und nachvollziehbar experimentiert werden kann. In VISA waren diese Systeme voll virtualisiert und die Infrastruktur wurde unter Verwendung einer „Bibliothek“ von virtualisierten Komponenten nachgebildet. Experimente in dieser virtualisierten Infrastruktur wurden aus dem Modell abgeleitet und mit standardisierten Eingabevektoren beschrieben.

Die erstellten Modelle sind mit verschiedenen Verfahren analysiert worden. Dazu wurden zuerst die Sicherheitsanforderungen an den Modellen identifiziert und exakt beschrieben. Solche Anforderungen ergaben sich aus Schutzkatalogen und detaillierten Bedrohungsanalysen. An Modellen konnte dann analysiert werden, ob die Sicherheitsanforderungen und die funktionalen Anforderungen an die Infrastruktur in einer entsprechenden Umsetzung prinzipiell erfüllt werden können.

Die Projektergebnisse von VISA haben KMUs einen wesentlich einfacheren und flexiblen Umgang mit ihren IT-Infrastrukturen ermöglicht, wie er bisher nur Großunternehmen vorbehalten war. Insbesondere verbesserte die Modellierung und Evaluation auf Basis virtueller Umgebungen und formaler Analysen die Bereiche Sicherheitsmanagement für IT-Infrastrukturen, Übereinstimmungsanalysen für Regulierungsanforderungen wie z.B.

Basel II und Evaluation von Implementierung von Infrastrukturen und deren Komponenten.

Durch das in VISA entwickelte Modellierungs-Framework ließ sich die Sicherheit von IT-Systemen modular und im Verbund einfacher und effizienter überprüfen. Dies war ein wichtiger Schritt in Richtung Ende-zu-Ende-Sicherheit, was ein essentieller Sachverhalt in IT-Landschaften ist. Gerade KMU-Betriebe sind immer mehr Wirtschaftsspionage des Auslands in Deutschland ausgesetzt, wogegen VISA durch Kompensation aktueller Sicherheitslücken Abhilfe schaffen könnte.

Ein weiterer Vorteil einer ganzheitlichen IT-Infrastrukturplanung durch das VISA-Rahmenwerk war der passgenaue und vereinfachte Einsatz von Sicherheitsanwendungen auf Basis von Virtual Security Appliances (VSA). Durch die umfassende Emulation von IT-Infrastrukturen konnten betriebsrelevante Parameter und die Integrationspunkte der VSAs bereits in der virtuellen Umgebung transparent identifiziert und der Einsatz erprobt werden. So erprobte VSAs konnten dann direkt ohne Änderung der existierenden Infrastruktur in Einsatz gebracht werden.

Eine wesentliche Hürde für KMU im Umgang mit IT-Infrastruktur im Allgemeinen – und für sicherheitsrelevante Infrastruktur ganz besonders – ist der Mangel an eigenem Know-how und der Verfügbarkeit von eigenem Fachpersonal. Um die erfolgreiche Realisierung solcher Infrastrukturen zu gewährleisten, ist eine Zielgruppen gerechte Bedienung und Darstellung durch grafische Benutzeroberflächen essentiell. Dadurch werden einerseits Fehlbedienungen minimiert, andererseits schafft eine geeignete Darstellung Transparenz und damit Vertrauen in die sicherheitsrelevanten Funktionalitäten. Die Realisierung grafischer Oberflächen für die VISA-Simulationswerkzeuge, die VSAs sowie der Management-Werkzeuge war daher ein wesentliches Projektziel, welches auch durch drei unterschiedliche Topologie Editoren (TE) erreicht werden konnte.

2.2 Wissenschaftliche und technische Ziele

Durch die starke Heterogenität von IT-Infrastrukturen in KMU und der relativ begrenzten Ressourcen und des geringeren Know-hows muss in Zukunft die Zielgruppe KMU bessere und geeignete Methoden zur Konfektionierung ihrer IT-Sicherheit bekommen. Der IT-Sicherheitsmarkt adressiert bislang zu wenig diese Zielgruppe, so dass keine bedarfsgerechten und dem Budget angepassten Lösungen vorhanden sind. Um eine höhere Autonomie in der Konfiguration sowie im Betrieb ihrer IT-Infrastruktur zu erhalten, sind daher modulare und erprobte Lösungen und Systeme essentiell. Die höhere Flexibilität kann und wird mittlerweile durch Virtualisierung von Rechnern und Diensten erreicht. Jedoch existieren bisher keine Lösungen, die auch Netze und Infrastrukturen für Unternehmen virtualisiert abbilden. Auch existieren nur Virtual Appliances (VA), die nur punktuell bestimmte Anwendungen oder Dienste bereitstellen, wie z.B. Mail-Security-Dienste, Firewall-Dienste etc. Eine Kombination von verschiedenen Sicherheitsfunktionen und Diensten wird nicht angeboten.

Nicht nur vor dem Hintergrund der Flexibilität, sondern auch aus Kostengründen (Investition in Hard- und Software) sind Virtual Security Appliances (VSA) auf Basis von

Open-Source (sowohl die Anwendungen als auch die Betriebssysteme) von Bedeutung. Außerdem würde der Einsatz in Unternehmen kein Know-how erfordern, da erprobte State-of-the-Art-Technologien als integrierbare Lösung in die IT von KMUs eingebunden werden können. Ein weiterer essentieller Aspekt ist die bessere Überprüfbarkeit im Sinne von Compliance. Der Aufwand zur Überprüfung von Sicherheitskomponenten bzw. der gesamten IT-Sicherheitsinfrastruktur wird erheblich reduziert, da Security Assessment und Compliance-Tests für Virtual Security Appliances (VSA) bereits vorliegen können. Durch die steigende Komplexität wird die Überprüfung der Konformität vereinfacht.

Vor diesem Hintergrund hat das Projekt VISA Möglichkeiten zur Modellierung von IT-Infrastrukturen von KMUs für Virtualisierung erarbeitet und auf Basis dessen bedarfsgerechte Virtual Security Appliances (VSA) konzeptioniert, die als integraler Bestandteil von KMUs eingesetzt werden können. Dies wurde durch Simulation und Emulation von komplexen Netzwerk- und Dienste-Topologien erreicht. Um einen bedarfsgerechten Einsatz für KMUs zu ermöglichen, wurde eine grafische Oberfläche zur Unterstützung der Modellierung sowie zur Visualisierung und Bedienung der Virtuellen Security Appliances (VSA) entwickelt.

Die wissenschaftliche Herausforderung hierbei lag in der geeigneten Modellierung von Netztopologien und der Abstraktion mittels Virtualisierungstechnologien. Eine wesentliche Fragestellung war dabei, ob IT-Infrastrukturen modular in derselben Art und Weise zusammenspielen und hierdurch die Komplexität ohne Sicherheitseinbußen erheblich reduziert werden kann. Hieraus ergab sich ein weiteres Ziel: die Validierung der Methoden und erarbeiten Lösungsvorschlage für Virtual Security Appliances mittels standardisierter Methoden wie ISO 27001 und BSI-Grundschutz. Ein weiteres wissenschaftliches Ziel war, zu erforschen, inwieweit virtualisierte IT-Security-Bausteine und Services aus Konzepten wie IaaS, PaaS und SaaS abgeleitet werden können.

VISA erstellte ein Framework, das das Erproben von VSAs in nachgebildeten, praxisorientierten Szenarien erlaubte. Hierfür wurden durch das Konsortium die folgenden technischen Herausforderungen gemeistert bzw. Ziele erreicht:

a. Entwicklung und Packetierung verschiedener VSA-Module, die unterschiedliche Bereiche der IT-Sicherheit abdecken.

b. Eine automatisierte und dynamische Umgebung, die eine experimentelle Erprobung verschiedener Netztopologien und den Einsatz von VSAs erlaubt.

c. Modelle, die die Simulation der Netztopologien steuern.

d. Jede VSA lag am Ende als virtuelles Image vor und konnte durch das Deployment-System entsprechend dem zugrunde liegenden Modell konfiguriert werden.

e. Es wurde ein Modell bzw. Ausdruckssystem entwickelt, um das Deployment zu steuern.

f. Eine Bibliothek von virtuellen Images wurde benötigt, um die möglichen Wirkszenarien zu bauen.

2.3 Definition einer VSA

Bevor man eine VSA verstehen kann, sollte man sich zuerst die Funktion einer **Virtual Appliance (VA)** deutlich machen. Als VA wird ein Image einer **Virtuellen Maschine (VM)** bezeichnet, welches ein installiertes und vorkonfiguriertes Softwaresystem enthält. Hierbei beinhaltet dieses Image auch schon das Betriebssystem selbst. Diese Software kann ohne Installation und Konfiguration des Betriebssystems und des eigentlichen Softwaresystems genutzt werden. Mehrere VAs können auf einem einzelnen Server ausgeführt werden. Im Projekt VISA wurden VAs für den Aufbau von IT-Sicherheitsinfrastrukturen und Komponenten genutzt. Dadurch können Komponenten, welche normalerweise nicht auf ein und demselben Server laufen sollten, virtualisiert und somit auf einem VM-Server installiert werden. Dies ergibt die Möglichkeit selbst für kleinere Unternehmen ihre IT-Infrastruktur, ohne den Einsatz von zusätzlicher Software, abzusichern.

Als **Virtual Security Appliance (VSA)** werden nun hingegen Virtual Appliances bezeichnet, die vorrangig der Sicherheit dienen (von Netzwerksicherheit Layer 2 bis hin zur Anwendungssicherheit Layer 7). Mit Hilfe von VSAs wird versucht, IT-Hard- und Software vollkommen zu virtualisieren. Dabei geht es darum, sicherheitsrelevante Komponenten der IT-Infrastruktur neu zu implementieren und/oder bestehende zu virtualisieren. Bei VSAs tritt der Sicherheitsaspekt in den Vordergrund, da sie genauso sicher sein müssen, wie eine konventionelle Lösung. Wechselwirkungen unter VSAs müssen abgesichert werden. Aufgrund dieser Tatsache werden von Anbietern heute ganze Virtualisierungsserver mit zusätzlichen Sicherheits-, Monitoring- und Verwaltungstools angeboten. Hierbei existieren verschiedene Ausprägungen wie Hardware und Software-Appliances, die flexibel kombinierbar sind.

Im VISA-Projekt wurden VSAs für KMUs konzeptioniert und realisiert. Diese VSAs wurden aus virtualisierten IT-Security-Bausteinen (Modulen) und Services zusammengestellt. Sie decken unterschiedliche Bereiche der IT-Sicherheit in typischen KMU-Topologien ab. Hierzu gehören beispielsweise:

a. Screened Gateways / Screened Subnets, die aus kaskadierten Firewall-Strukturen bzw. Kombinationen von Firewalls zusammengesetzt sind (Paketfilter, Proxy, Firewall, Application Level Gateway, Paketfilter etc.)

b. Authentisierungsservices, die i.d.R. aus einem RADIUS-Server oder/und LDAP-Server bzw. Domänen-Controller bestehen

c. PKI-Dienste, die aus einem CA/RA-Server und optional einem Proxy bestehen

Mit Hilfe diesen modularen VSAs soll es dann für KMUs möglich sein, relativ einfach IT-Sicherheitsmodule in die bestehende IT-Infrastruktur fehlerlos zu integrieren.

2.4 Mehrwerte durch den Einsatz der VISA-VSAs

Durch die Arbeiten von VISA ergeben sich deutlich Verbesserungspotenziale, um eine hochverfügbare und sichere IT-Infrastruktur für KMU schaffen zu können:

1. Die Infrastruktur kann logisch entzerrt werden, und Anwendungen dort im Netzwerk betrieben werden, wo es aus Security-Sicht angemessen ist.
2. Die gesamte Infrastruktur (Server, Firewall, Router, VPN etc.) wird virtuell konzipiert und kann nach erfolgreichen Tests als Live-System direkt übernommen werden. Der Vorteil liegt in diesem Szenario darin, dass logisch eine komplette Infrastruktur vorgehalten wird.
3. Die Virtualisierung kann gleichzeitig zur Hardware-Konsolidierung genutzt werden, und dadurch einen wesentlichen Beitrag zur Kostenreduktion leisten (Hardware-Bedarf, Strom- und Kühlkosten). Durch die Konsolidierung (Hardware im Allgemein und Security) reduziert sich die Komplexität der gesamten IT-Landschaft nachhaltig, was zur Eingrenzung von Know-how-Bedarf, zur Erhöhung der IT-Sicherheit und ebenfalls zur Reduktion der Kosten beitragen würde.
4. Die komplette IT-Infrastruktur könnte komplett virtuell vorgehalten werden. Dadurch lassen sich auch Redundanzen (wie Firewall oder Router) einfacher aufbauen, um neben der IT-Sicherheit auch die Verfügbarkeit zu gewährleisten.
5. Hiermit steigt auch die Möglichkeit für KMU, die bislang die Aufwände für Zertifizierungen oder ein nachweisbares Sicherheitsniveau nicht leisten konnten, dies zu tun, und sich damit neue Märkte zu erschließen.
6. Durch die Flexibilisierung der Infrastruktur bei gleichzeitiger Komplexitätsreduktion bleibt für die IT-Mitarbeiter mehr Zeit, um sich dem Thema IT-Sicherheit und geordnete Betriebsprozesse pro-aktiv zuwenden zu können.
7. Aufbau von Know-how als auch operativer Umsetzung können gezielt in den notwendigen Themenkomplexen erfolgen, die minimal notwendige Breite wird geringer und so sinkt die Gefahr, sich mit zu vielen Themen gleichzeitig zu beschäftigen.
8. Der Aufbau von Testumgebungen und damit die Abschätzung von Auswirkungen von Änderungen an der IT-Umgebung eines Unternehmens wird vereinfacht, da die Bausteine aus der Produktion nahezu identisch im Rahmen einer Testumgebung eingesetzt werden können. Notwendige Maßnahmen hierfür, wie z.B. die einfache Bereinigung des Systems von Echtdaten, können dabei durch ein geeignetes Konzept der VSA und der Bereitstellung von geeigneten Standardprozessen für ein solches Vorgehen, bereits berücksichtigt werden.
9. Die Standardbausteine können bereits ein geeignetes Maß an Dokumentation und Sicherheitsnachweisen mitbringen, so dass damit Anforderungen ohne großes eigenes Know-how und Aufwände im Unternehmen umgesetzt werden können. Das Unternehmen kann sich auf die Abweichungen vom Standard konzentrieren, bzw. die darüber hinausgehenden Themen. Das Sicherheitsniveau steigt und der Aufwand und das Know-how, um einen sicheren Betrieb aufzubauen, sind

geringer

10. Neben dem Konzept für die VSAs kann eine Empfehlung für den Umgang mit der virtuellen Infrastruktur sowie Prozessempfehlungen für das Management der Landschaft gegeben werden. Damit können auch weite Teile der Anforderungen an Betriebsprozesse (z.B. Change-Management) abgedeckt werden.

Um diese Verbesserungspotentiale nutzen zu können, mussten bei der Konzeption der Virtualisierungsumgebung und der VSAs auf weitere Rahmenparameter geachtet werden. So durften Sicherheitsmaßnahmen auf VSA-Ebene nicht durch die Virtualisierungsschicht hinfällig werden bzw. die Virtualisierungsschicht musste ebenso geeignete Maßnahmen anbieten, die wiederum die Sicherheit und Nachvollziehbarkeit in der Landschaft durchgängig ermöglichten.

Ein anschauliches Beispiel hierzu war z.B. die Nachvollziehbarkeit von Änderungen an Daten oder einem System:

a. In der physischen Welt ist es in der Regel ausreichend, die Änderungen am System selber sowie ggf. noch Recovery-Maßnahmen zu protokollieren, um den Stand eines Systems für jeden Zeitpunkt rekonstruieren zu können.

b. In einer virtuellen Landschaft kommt mit der Verwaltung der VSAs eine Schicht hinzu: denn neben den Änderungen (Changes) an der VSA ist ebenso relevant, ob VSAs bzw. virtuelle Bausteine zwischenzeitlich ausgetauscht wurden und ggf. die für ein System geltende Change-History nicht für die Landschaft gilt, weil die Maschine teilweise nicht produktiv genutzt wurde.

Für die Nachweisbarkeit der Sicherheit und Compliance muss weiterhin die Übertragbarkeit von Security Assessments oder Tests sowie die übergreifende Effektivität der umgesetzten Maßnahmen in VSAs zwischen verschiedenen Einsatzszenarien sichergestellt werden, bzw. deren Möglichkeiten und Grenzen eindeutig dargelegt werden, um valide Aussagen treffen zu können. Sind diese Voraussetzungen erfüllt, können die durch geeignete Sicherheitsmaßnahmen gemäß existierender Standards gesicherten Infrastrukturbausteine, die virtuellen Systeme der VSAs, wie bisher physische Systeme abgesichert werden. Die Maßnahmenauswahl richtet sich dabei anhand der Funktion der jeweiligen Komponente aus. [14]

2.5 Entwickelte VSA-Komponenten

Im VISA-Projekt wurden von den verschiedenen Partnern VSAs konzeptioniert, die vorrangig der Sicherheit dienen (von Netzwerksicherheit, Layer 2 bis Anwendungssicherheit, Layer 7). Diese VSA bestehen im Wesentlichen aus virtualisierten IT-Security-Bausteinen (Modulen) und Services. Sie haben das Ziel, unterschiedliche Bereiche der IT-Sicherheit in typischen KMU-Topologien abzudecken. Folgende VSAs werden an dieser Stelle kurz vorgestellt, da sie innerhalb des Projektes bereits erarbeitet wurden:

1. **VSA-AAA:** Virtual Security Appliance – Authentication, Authorization, Accounting
2. **VSA-MAC:** Virtual Security Appliance – Metadata Access Control
3. **VSA-SRA:** Virtual Security Appliance – Secure Remote Access
4. **VSA-UTM:** Virtual Security Appliance – Unified Threat Management

Von dem wissenschaftlichen Projektpartner FH Dortmund[1] wurde die **VSA-AAA** realisiert, die eine CA/RA (Zertifikatsserver mit Online- bzw. Offline-RA), ein Linux-basierter Domain Controller mit LDAP sowie ein RADIUS-Server beinhaltet. Bei dieser VSA steht das Zusammenspiel von State-of-the-Art Authentisierungs- und Autorisierungs- sowie PKI-Komponenten auf Open-Source-Basis im Vordergrund. Die VSA-AAA soll somit die sichere Einwahl in ein Unternehmensnetz sowie Log-In in Domänen innerhalb des Unternehmensnetzes ermöglichen. Sie wurde auf Basis der Open-Source-Lösung FreeIPA realisiert. FreeIPA ist eine integrierte Sicherheitslösung, was sich aus einem Linux-Grundsystem (Fedora), 389 (ehemals Fedora Directory Server), MIT Kerberos, NTP, DNS und Dogtag-Zertifikatsserver zusammensetzt und diese kombiniert. Zur Konfiguration werden eine Webschnittstelle und kommandozeilenbasierte Tools angeboten.

Bei der VSA-AAA stand somit das Zusammenspiel von State-of-the-Art-Authentisierungs- und Autorisierungs- sowie PKI-Komponenten im Vordergrund sowie die sichere Einwahl ins Firmennetz und Authentisierungsdienste im Allgemeinen, die das Log-In in eine Domäne ermöglichen sollten. Letzteres umfasste eine zentrale Verwaltung der Benutzer, Computer (Hosts) sowie Dienste, die zu diesen zugeordnet werden. Basis der VSA-AAA ist das Betriebssystem CentOS in der Version 6.3, da nur für auf RedHat basierenden Systemen (CentOS und Fedora) die genutzten Komponenten (freeIPA) verfügbar sind. Eine wesentliche Anforderung war, dass alle Sub-Systeme (Module bzw. Komponenten der VSA-AAA) soweit wie möglich vollautomatisch von einer einzigen DVD installierbar sein sollten. Die VSA-AAA bildet AAA (Authentication, Authorization, Accounting)-Services ab und umfasst dabei folgende Funktionen:

a. **Domänen-Controller:** Die Funktion des Domänen-Controllers wurde durch die Software freeIPA bereitgestellt. Intern wurde hier ein LDAP-kompatibler Verzeichnisdienst (389-Directory-Server; Fedora Directory Server)) zur zentralen Speicherung der Host-Accounts eingesetzt. Zur Authentifizierung diente ein Kerberos-Dienst. Über Kerberos-Tickets konnten dann auch Sigle-Sign-On-Funktionalitäten (SSO) innerhalb der Domäne realisiert werden.

b. **PKI:** IPA unterstützt Maschinen- und Service-Identitäten mittels der dogtag-PKI über X.509-Zertifikate. Die PKI wurde über den inhärenten Dienst dogtag abgewickelt, was eine CA und RA darstellte. Jedoch wurde auch die Möglichkeit gegeben, eine bereits vorhandene CA einzubinden (mit dogtag als SubCA oder dogtag ganz deaktiviert). Grundlegende Funktionen der PKI waren: Public Key

[1] http://www.fh-dortmund.de

Infrastruktur, basierend auf einem Stammzertifikat sowie das Bereitstellen von Dienste- und Host-Zertifikaten sowie das Management einer Zertifikatssperrliste (Certificate Revocation List CRL). Hier kann eventuell auch ein Vertrauensverhältnis zu einer bestehenden PKI aufgebaut werden.

c. **Authentisierungsdienst mittel RADIUS:** Dieser auf einem freeRADIUS-Server basierende Dienst lief in einer VM und konnte auch als autarkes Modul von anderen Diensten und VSAs genutzt werden, da ein RADIUS-Server typischerweise die zentrale Authentisierungseinheit für das Firmennetz ist und beispielsweise von NAS-Diensten wie VPN-Servern aus der Einwahl-DMZ oder von WLAN Access Points oder WLAN-Access Concentrators angesprochen wird.

d. **DNS-Server:** Der inhärente DNS-Dienst Bind9 ist ein essentieller Bestandteil für eine Domäne und erlaubte die Auflösung des Namensraums. Jedoch musste die Möglichkeit gegeben werden, einen bereits vorhandenen DNS-Server einzubinden bzw. dieses als „Forwarder“ zu nutzen. Optional können Domain-Namen mittels des integrierten ISC Bind Servers verwaltet werden.

e. **NTP:** Der NTP(Network Time Protocol)-Dienst diente vorrangig dafür, dass für den Kerberos- sowie für die PKI-Dienste die Zeiteinstellung der in der Domäne befindlichen Rechner synchron gehalten wird. Hierzu wurde die Software openNTPd eingesetzt.

Als essentieller Bestandteil einer Sicherheits-IT von Unternehmen wurden die o.g. Dienste in der VSA-AAA gebündelt realisiert. Die VSA-AAA verhält sich nach Außen funktional wie eine einzige logische Einheit. Die VSA-AAA ist damit eine Art Container für die darin enthaltenen VMs wie RADIUS, CA, Domänen-Controller usw. Mit anderen Worten: Der Benutzer sieht eine Gesamtheit der Funktionen, die durch die VMs zur Verfügung gestellt werden.

Abb. 1: VSA-AAA Auswahl der verfügbaren Systeme

Der Projektpartner DECOIT GmbH[2] erarbeitete zwei weitere VSAs: VSA-MAC und VSA-SRA. Die **VSA-SRA** ermöglicht das sichere Einwählen in ein Firmennetz mittels eines Android-Smartphones. Dies beinhaltet die Komponenten Android-Client, FreeRADIUS-Server, TNC-Server und VPN-Gateway. Das Smartphone verbindet sich durch das VPN-Gateway mit dem Unternehmensnetz. Dadurch ist aber noch nicht sichergestellt, ob das Smartphone als vertrauenswürdig eingestuft werden kann, da nur die Teilnehmerdaten abgefragt werden. Dies wird erst durch das Senden gesammelter Metriken des Android-Smartphones vom TNC[3]-Client an den TNC-Server ermöglicht. Die Metriken enthalten die installierte Applikationsbasis, Versionsnummern und Richtlinien, die für das Smartphone gelten. Der TNC-Server vergleicht anschließend die gesendeten Metriken mit denen in seiner Datenbank. Sind Applikationen installiert, die er nicht kennt oder die auf seiner Blacklist enthalten sind, wird dem Smartphone der Zugang verweigert bzw. das Smartphone wird in ein Quarantänenetz isoliert. Innerhalb des Quarantänenetzes kann das Endgerät mithilfe einer Softwareverteilungslösung auf den geforderten aktuellen Stand gebracht werden. Anschließend kann das Gerät gemäß den TNC-Spezifikationen eine erneute Attestierung anfordern. Sind alle Voraussetzungen erfüllt, erhält der Teilnehmer des mobilen Endgeräts Zugriff auf die gewünschte Zielapplikation und somit auf die gewünschten Zielressourcen.

Die entwickelte VSA-SRC muss daher verschiedene Komponenten für eine Umsetzung beinhalten: [12]

a. **VPN-Gateway:** Im ersten Schritt muss sich der Teilnehmer über einen VPN-Zugang am VPN-Gateway authentifizieren. Damit ist der Login und das Passwort des Teilnehmers zwar abgefragt worden, aber die Hardware kann bereits kompromittiert sein. Daher muss im zweiten Schritt eine Abfrage des TNC-Servers erfolgen.

b. **TNC-Server:** Dieser Server nimmt die vom TNC-Client erhaltenen Integritätsmessungen entgegen, die für das mobile Endgerät durchgeführt werden. Anschließend fällt er anhand der Policy-Auswertung eine Entscheidung, ob der Zugriff des mobilen Endgeräts gestattet wird oder nicht.

c. **TNC-Client:** Der Client bildet die Schnittstelle zwischen dem Network Access Requestor und den Plug-Ins des mobilen Endgeräts, welche die Informationen von Antivirus-Systemen oder anderen sicherheitsrelevanten Komponenten sammeln.

d. **RADIUS[4]:** Der RADIUS-Server enthält die Einwahlprofile der Teilnehmer, die sich auf das Unternehmensnetz von außen verbinden dürfen. Die Profile enthalten auch die Konfigurationen der jeweiligen Teilnehmer. Eine Synchronisation mit dem internen Verzeichnisdienst wäre sinnvoll.

[2] http://www.decoit.de
[3] TNC = Trusted Network Connect (Spezifikation der Trusted Computing Group)
[4] RADIUS = Remote Authentication Dial In User Service

e. **LDAP[5]:** Im Rahmen dieser VSA wird angestrebt, eine Authentifizierung über den Verzeichnisdienst LDAP zu gewährleisten. Dieser bietet eine Unternehmenssicherheit im internen Unternehmensnetz. Außerdem wird angestrebt einen TNC-Server als VSA anzubieten sowie die Integration von TNC-Clients in anderen VSAs zu zeigen.

f. **PKI[6]-Server:** Bei Bedarf kann auch eine Zertifizierungsstelle (CA-Server) mit in das Szenario eingebunden werden, bei welcher bei Benutzeranlage durch das VPN-Management-System ein Zertifikat beantragt wird und für den Benutzer hinterlegt wird. Das VPN-Managementsystem tritt hierbei als Registrierungsstelle (RA) auf. Die VSA kann auch erweitert werden, um die Zertifizierung von einer öffentlichen Stelle aus zu übernehmen.

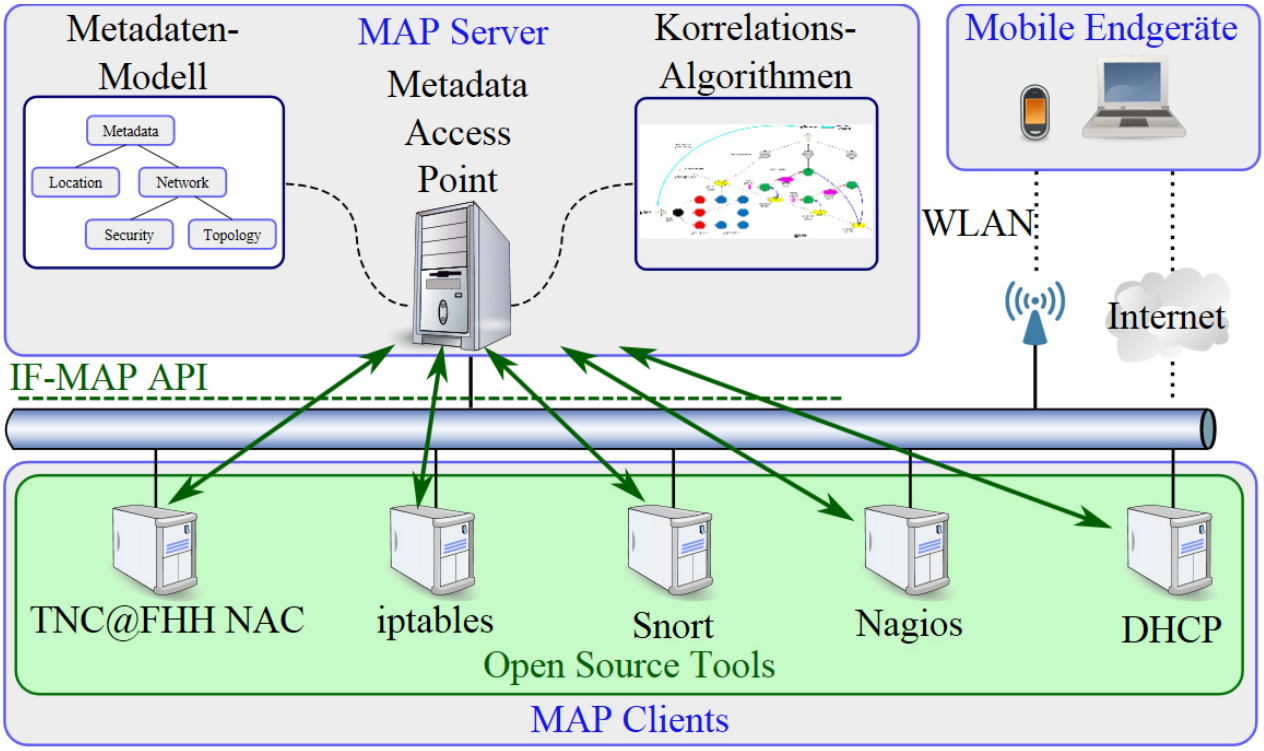

Abb. 2: IF-MAP-Architektur der VSA-MAC

Die **VSA-MAC** besteht hingegen aus den Komponenten IF-MAP[7]-Server und den IF-MAP-Clients für Android, Snort, iptables, FreeRADIUS und Nagios. Bei IF-MAP handelt es sich um ein offenes, herstellerunabhängiges Client-Server-Netzprotokoll zum Austausch von beliebigen, in XML codierten Metadaten. Dabei stellt der IF-MAP-Server die zentrale Komponente dar, indem die Daten von allen IF-MAP-Clients gesammelt und

[5] LDAP = Lightweight Directory Access Protocol
[6] PKI = Public-Key-Infrastruktur
[7] IF-MAP = Interface for Metadata Access Point (Spezifikation der Trusted Computing Group)

durch einen Graphen zur Verfügung gestellt werden. Weiterhin stellt er die gesammelten Daten auch den IF-MAP-Komponenten zur Verfügung. Die Stärke von IF-MAP gegenüber einer reinen IDS-basierten Anomalie-Erkennung liegt dabei in der Diversität der Daten. Durch die gesammelte Datenbasis lassen sich Korrelationen durchführen und Anomalien leichter erkennen bzw. Angriffen entgegenwirken. Beispiele hierfür sind unter anderem die Blockierung des Datenstroms durch eine Firewall, Sperrung des Zugriffes auf das Unternehmensnetz in Form eines Switches oder eines VPN-Gateways, Isolierung des Endgerätes in eine Quarantänezone etc. Abschließend können auf Grundlage der gesammelten Informationen die unterbundenen Aktivitäten sowie deren Details protokolliert und entsprechende Meldungen an die verantwortlichen Systemadministratoren generiert werden.

Als essentieller Bestandteil einer Sicherheits-IT von Unternehmen wurden die Dienste der VSA-SRC und VSA-MAC in einer VSA gebündelt, um bedarfsweise modular zu- und abschaltbar zu sein (siehe Abb. 9). Zusätzlich wurden innerhalb des Projektes unterschiedliche Front-ends analysiert und ein eigenes entwickelt. Wichtig dabei war, dass das Zusammenspiel der unterschiedlichen Komponenten gewährleistet werden kann. Als Basis wurde immer ein gehärtetes VSA-System (Hostsystem und Gastsysteme) eingesetzt. [14]

Die entwickelte VSA-MAC muss daher verschiedene Komponenten für eine Umsetzung beinhalten: [12]

a. **IF-MAP-Server:** Der IF-MAP-Server sammelt sämtliche Informationen der IF-MAP-Clients und konsolidiert diese zu einer gemeinsamen Datenbasis. Zustandsänderungen und Anomalien werden hier erkannt und an die relevanten Systeme weitergeleitet.

b. **Firewall (iptables) IF-MAP-Client:** Als Beispiel für den Einsatz des IF-MAP-Protokolls wird eine VSA angestrebt, die eine dynamische Konfiguration einer Firewall, basierend auf dem Status des Gesamtnetzes, durchführt. Hierbei werden die Informationen über die Anmeldung eines Gerätes am Netz ausgewertet und in Firewall-Regeln umgesetzt.

c. **Android IF-MAP-Client:** Hier soll eine VSA angestrebt werden, die das Auswerten unterschiedlicher Eigenschaften ermöglicht. Dieses kann die IP-Adresse, MAC, OS-Version, installierte Anwendungen inkl. Berechtigungen, sowie die Position eines Endgerätes sein.

d. **Snort IF-MAP-Client:** Der Client beinhaltet die Aufbereitung und Veröffentlichung von Snort-Meldungen. Dies könnte verdächtiger Datenverkehr oder eine Portscan-Erkennung sein. Snort bietet die Möglichkeit Datenverkehr zu analysieren und diesen bei unzulässigen Zugriffen zu unterbinden. Snort ist daher auch als unabhängige VSA zu sehen, die in mehreren Instanzen eingesetzt werden kann. So kann der externe Paketfilter und der interne Paketfilter durch zwei Instanzen der Snort-VSA umgesetzt und angepasst werden. Diese können

vor und nach der Gateway-VSA implementiert werden.

e. **RADIUS-IF-MAP-Client:** Dieser Client ermöglicht das Auslesen und Veröffentlichen von RADIUS-Informationen. Auch die Anmeldung und Abmeldung eines Clients werden hier vorgenommen. RADIUS ist eine Authentification-VSA, welche die Authentifizierung von Benutzern übernimmt. Sofern eine Authentifizierung gescheitert ist, muss RADIUS diese Information an die Gateway-VSA senden. Somit kann die Gateway-VSA den weiteren Datenverkehr unterbinden. Die Gateway-VSA muss diesbezüglich er-weitert werden.

f. **DHCP IF-MAP-Client:** Dieser Netzwerkdienst zur automatischen Verteilung von IP-Adressen kann ebenfalls IF-MAP-fähig gemacht werden und so Einfluss auf die Netzwerkdienste geltend machen.

Die für das VISA-Projekt entworfene VSA-UTM wurde ausschließlich für die Collax[8]-Entwicklungsumgebung umgesetzt. Als Virtualisierungsplattform diente dabei die Virtualisierungslösung Collax V-Cube, basierend auf dem Open-Source-Projekt KVM, sowie libvirt. Für die einzelnen VSAs wurden Templates aus den Collax-Komponenten vorbereitet, die den Funktionsumfang der definierten VSAs umfassten. Die Templates wurden für den virtuellen Betrieb angepasst. Dazu wurde die Unterstützung der Paravirtualisierung integriert, die für den virtuellen Betrieb modifizierte Konfiguration hinterlegt und das automatische Deployment als Virtual Machine (VM), basierend auf dem Template, implementiert. Die so bereitgestellten VSAs wurden über das Management-Interface als virtuelle Instanz in Betrieb genommen. Hierzu wurde ebenfalls ein eigenes Front-end entworfen.

Die VSA-UTM besteht aus unterschiedlichen virtuellen Sicherheitskomponenten, die sich wie folgt auflisten lassen:

a. **Internet Gateway:** stellt die Standard-Verbindung zum Internet her und ist integraler Bestandteil der VSA-UTM.

b. **VPN-Gateway:** erlaubt es VPN-Tunnel und private Netzwerke bereitzustellen, indem Netzwerkverkehr verschlüsselt und dann über unverschlüsselte TCP/IP-Verbindungen übertragen wird. Starke Verschlüsselungsalgorithmen sorgen dafür, dass der Netzwerkverkehr nicht von unberechtigten Personen abgehört werden kann. Für die Identifikation und die Authentifizierung der Gegenstelle werden X.509-Zertifikate eingesetzt. Die Zertifikate dienen auch als Grundlage zur Aushandlung der Schlüssel.

c. **SSL-VPN-Gateway:** wird als Alternative zu IPsec-basierten VPN zur Bereitstellung von sicherem Zugriff auf Anwendungen für mobile Mitarbeiter

[8] http://www.collax.com

verwendet. Die Verbindungen werden dabei über eine sichere SSL-Verschlüsselung bereitgestellt. Um die Sicherheitsniveau zu erhöhen und die administrative Kontrolle feingranularer zu gestalten, werden ausgewählten Anwendern nur dedizierte Applikationen für den Fernzugriff zugänglich gemacht. Fünf Optionen zur Bereitstellung von unternehmensinternen Applikationen sind umgesetzt worden: S-Tunnel für den direkten Netzwerkzugriff, Reverse-Proxy und Web-Weiterleitung für Web-Anwendungen sowie VNC- und RDP-Anbindung für Remote-Desktop-Verbindungen.

d. **E-Mail-Proxy:** kommt zum Einsatz, um den E-Mail-Verkehr mit Hilfe etablierter Virenfilter gegen Malware (wie Viren, Trojaner oder Würmern) zu schützen. Zudem wird für die VSA auch ein Spam-Schutz realisiert. Für die effiziente Filterung von Spam wurde die VSA um die Spam-Erkennungsverfahren DKIM und NiX-Spam erweitert.

e. **Web-Proxy:** kommt zum Einsatz, um den Web-Verkehr gegen Malware (wie Viren, Trojaner oder Würmern) zu schützen. Dazu unterbricht der Proxy die Netzwerkverbindung zwischen einem Web-Browser und einem Web-Server, indem er sich als Zwischenglied einschaltet. Dadurch ist ein Web-Proxy in der Lage den Netzwerkverkehr auf Sicherheitsbedrohung zu analysieren, indem ein etablierter Virenfilter zum Einsatz kommt. Zudem wurde der Web-Proxys auch mit einem sog. Content Filter ausgerüstet. Somit werden die Adressen (URLs) von Internetzugriffen analysiert und entschieden, ob eine Berechtigung für die Anzeige einer Webseite besteht oder nicht.

f. **Multilink Internet Gateway:** erlaubt es mittelständischen Unternehmen, zwei oder mehr Internet-Zugänge von unterschiedlichen Providern für Redundanzen und zur Lastverteilung zu nutzen. Dafür wurden die Funktionalitäten Link-Failover. Link-Load-Balacing, Policy-based Routing und Traffic Shaping umgesetzt.

g. **Firewall-Netzwerk-Segmentierung:** soll es mittelständischen Unternehmen ermöglichen verschiedene Techniken für die Segmentierung von Netzen mit unterschiedlichen Zwecken bereitzustellen. Für das VISA-Projekt wurden ein komplexes Netzwerk sowie die Nutzung von Netzwerk-Segmentierung geplant. Es wurden die Konfiguration der virtuellen Netzwerke und das Zusammenspiel von virtualisierten und emulierten Netzwerkkomponenten mit der Anbindung an die physischen Netzwerkkomponenten dabei gewährleistet. Zudem ist sichergestellt worden, dass die Segmentierung der Netzwerke keinen relevanten Einfluss auf die Netzwerk-Performance, insbesondere den Durchsatz und die Latenz, hatte.

h. **Multi-Level-Firewall:** soll es mittelständischen Unternehmen ermöglicht werden, die Firewall, basierend auf Benutzerinformationen, Anwendung und Betriebssystem, zu konfigurieren. Für die Integration der Multi-Level-Firewall wurde das nach der GPL lizenzierte Open-Source-Projekt UVWI (User Filtering With Iptables, ehemals NuFW) eingesetzt und in das CIF integriert. Die beiden Komponenten ufwi-filterd und ufwi-authd wurden ebenfalls implementiert. Die dritte, Client-seitige Komponente zur Kommunikation zwischen den Windows-

Clients und dem Server konnte aus lizenzrechtlichen Gründen nicht übernommen werden und wurde daher neu entwickelt. Zusätzlich wurde für den Administrator ein Werkzeug entwickelt, dass ihm zur Definition von benutzerabhängigen Firewall-Regeln unterstützt.

2.6 VSA-Eigenschaften

Essentielle Eigenschaften, der im VISA-Projekt entwickelten VSAs, sind zusammengefasst:

1. **Integrierbarkeit:** Eine VSA kann als autarke Einheit in bestehende IT-Infrastrukturen integriert werden. Die Grundlage für unterschiedliche Topologien sind dabei die Empfehlungen des BSI.
2. **Steuerbarkeit:** Eine VSA ist steuerbar und monitorbar, beispielsweise durch Tools wie libvirt oder über eine Web-GUI wie den VISA Topologie Editor (V-TE) bzw. andere Schnittstellen wie REST.
3. **Konfigurierbarkeit:** Die VMs der VSA können konfiguriert werden.
4. **Modularität:** Die VSAs wurden derart modular konzipiert und realisiert, dass eine hohe Verzahnung und Kombination dieser zur Erhöhung der Sicherheit bzw. der Flexibilität und Integrierbarkeit ermöglicht werden konnte. Hierzu wurde eine Bibliothek von möglichst leicht integrierbaren VMs aufgebaut, aus denen VSAs leicht und sicher nach dem Baukastenprinzip konfektioniert werden können.
5. **Vergleichbarkeit:** Die VSAs wurden derart konzipiert und umgesetzt, dass hinsichtlich Performance, Verfügbarkeit und Sicherheit keine gravierenden Unterschiede zu physikalischen und konventionellen Infrastrukturkomponenten und Topologien bestehen, sondern dies vergleichbar ist.
6. **Sicherheit:** Die VSA selbst muss sicher sein. Diese wesentliche Anforderung betrifft sowohl die Sicherheit der VMs (Gastsysteme), als auch die des Hostsystems. Eine Härtung des Betriebssystems der VMs und des Hostsystems sowie die Trennung der VMs voneinander durch den Hypervisor oder durch virtuelle Switches mit Authentisierungs- und Autorisierungsfunktionen wie sie beispielsweise durch OpenVSwitch gegeben sind, spielten hierbei eine wichtige Rolle.
7. **Komplementarität:** Das Ziel war es, VMs mit unterschiedlichen und komplementären Funktionen innerhalb einer VSA zu kombinieren. Dies bedingt auch die Vernetzung der VSAs durch Virtuelle Switches, wie z.B. VDE (Virtual Distributed Environment) oder OpenVSwitch, wobei die letztere sehr viele weitere Funktionalitäten aufweist. Hier werden praktische Erkenntnisse zur Emulation/Virtualisierung von Layer-2 Komponenten einfließen.

Bei der Absicherung von VSAs (einzeln und im Verbund) wurden etablierte Standards zur Absicherung von Komponenten eingesetzt und angewandt. Da dies eine große Zahl von Standards umfasst, können diese an dieser Stelle nicht explizit aufgezählt werden. Sie lassen sich kategorisieren in Protokolle, Verfahren, Algorithmen und fertige

Lösungen (kommerzielle und Open Source). Die Absicherung von komplexen VSA, die mehr als eine Funktion bzw. Komponente beinhalten, ist entsprechend aufwändiger. Nicht nur durch die Zahl, sondern auch durch die individuellen und spezifischen Besonderheiten und damit Verwundbarkeiten der Komponenten. Hier kommen etablierte Verfahren zum Tragen, um die VSA zu schützen bzw. abzusichern. Einfachstes Beispiel ist die VSA-UTM von Collax, die mit ihrem Firewall-Modul eine spezielle Implementierung einer am Markt verfügbaren Firewall ist (z.B. iptables). Sie muss mindestens den Sicherheitslevel dieser Implementierungen bzw. Distributionen abbilden und es dürfen keine Sicherheitslücken dazu kommen.

Um in konkreten Anwendungsszenarien effizient Test- und Produktivumgebungen mit VSAs aufbauen zu können, wurden von der Virtualisierungsplattform verschiedene Eigenschaften verlangt. „VSA Templating“ erlaubt für eine gegebene VSA ein „Muster“ anzulegen, das dann mehrfach instanziiert werden kann. Dies ist immer dann sinnvoll, wenn in einem Szenario gleiche oder sehr ähnliche VSAs, die sich nur in Ihrer Konfiguration unterscheiden benötigt werden. „VSA Cloning“ erlaubt eine gegebene VSA in ihrer Gesamtheit zu kopieren. Dies ist sinnvoll, um VSAs aus dem Testbetrieb in den Produktivbetrieb zu übernehmen.

Um VSAs auf mehr als einem Virtualisierungshost im Verbund betreiben zu können und dabei die Vorteile der Virtualisierungstechnologie zu bewahren, müssen Netzwerke über Hypervisor-Grenzen hinweg virtualisiert zur Verfügung stehen. Dabei standen mehrere Design- und Implementierungsmöglichkeiten zur Auswahl (Bridge/Tun-Devices, VDE, OpenVSwitch), die im Projekt untersucht wurden. Der Schwerpunkt wurde nach der Analyse auf OpenVSwitch gelegt.

Um VSAs aus dem Testbetrieb in den Produktivbetrieb überführen zu können, wird von der Virtualisierungsplattform „Storage Migration“ verlangt. Dies erlaubt einen virtuellen Server in seiner Gesamtheit, also persistente Daten der VSA sowie ihre Laufzeitkonfiguration von einem Virtualisierungsserver (Hypervisor) in der Testumgebung auf einen anderen Virtualisierungshost in der Produktivumgebung zu verschieben. Um hier größtmögliche Offenheit und Flexibilität zu gewährleisten wurde zusätzlich die Import- und Exportmöglichkeit für OVF/OVA-Dateien umgesetzt, das einen de-facto Standard für den Austausch von virtuellen Servern zwischen verschiedenen Virtualisierungsplattformen darstellt.

Die entwickelten VSAs wurden innerhalb des Projektes einzeln und im Verbund anhand verschiedener Parameter evaluiert und durch methodisch fundierte „Security Assessments“ nach Sicherheitsstandards (z.B. BSI) bewertet. Ebenso wurden, wenn die VSAs im Verbund genutzt werden sollten, auch die Management-Verfahren des Virtualisierungsrahmenwerks betrachten worden, wie z.B. Change-Prozesse der VSAs, Deployment-Methoden und die Verwaltung des Frameworks selber. Das Ziel war es hierbei, die IT-Konzepte auf Sicherheitsstandards wie ISO 27001 und BSI-Grundschutz überprüfen zu können, ohne dass entsprechendes Wissen bei dem KMU vorgehalten werden muss. Diese Standards werden für Zertifizierungen bzw. zur zertifizierten IT-Sicherheit angewandt. Diese auf Standards und Vorgehensmodelle geprüften VSAs erlauben nun insbesondere KMU, ein definiertes Sicherheitsniveau mit angemessenem

Aufwand und geringen Kosten zu erreichen, indem sie als geprüften Elemente oder Verbünde dann direkt verwendet werden können. Auf technischer Ebene wurden die nötigen Sicherheitsvorgaben damit bereits weitestgehend eingehalten.

2.7 cOntrol and Management Framework (OMF)

Das OMF-Framework[9] des australischen Partners NICTA[10] wurde im VISA-Projekt zum experimentellen installieren und testen von IT-Infrastrukturen weiterentwickelt und VISA zur Verfügung gestellt werden, um die konkreten Entitäten und Interaktionen zu definieren sowie ein konkretes System für die Emulation herbeizuführen. Dieses Framework ist für das experimentelle Installieren und Testen von Infrastrukturen entwickelt worden und wurde auch erfolgreich in EU-geförderten Projekten wie EU FP7 Nanodatacenters[11] eingesetzt. Es ist momentan in Europa, USA und Australien verbreitet.

Die Abb. 3 macht die Funktionalität von OMF deutlich. Ein Anwender beschreibt das durchzuführende Experiment auf Basis einer High-Level-Sicht über eine entsprechende Sprache und gibt diese Vorgaben an OMF weiter. Das Rahmenwerk wird dann eine Konfiguration des Experiments durchführen und es entsprechend auf der Plattform verbreiten. Dadurch können Messungen und das Sammeln von Daten im Vorfeld durchgeführt werden, bevor es zu einer realen Implementierung kommt. Die Messungen werden dabei kontinuierlich an ein Repository gesendet, was auch zur dynamischen Steuerung eines Experiments genutzt werden kann.

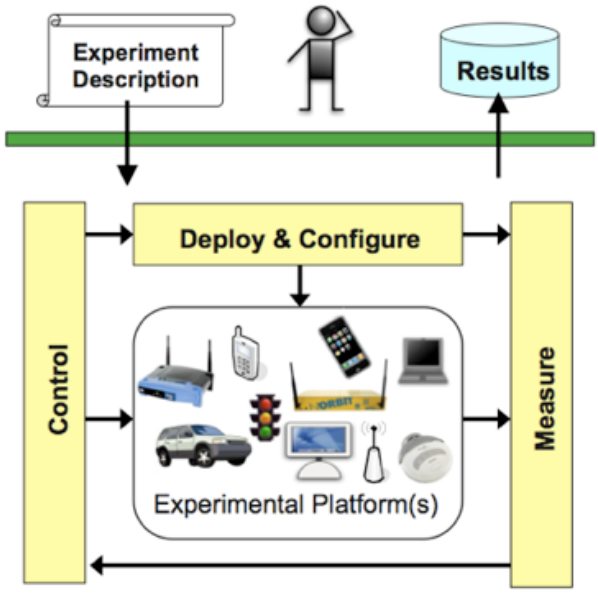

Abb. 3: OMF-Plattform aus Sicht eines Anwenders

[9] http://omf.mytestbed.net/wiki/omf
[10] http://nicta.com.au
[11] http://www.nanodatacenters.eu

Im Detail heißt dies: jeder am Experiment teilnehmende Knoten (Rechner/VM) führt dazu den OMF Resource Controller (RC) aus. Dieser Dämon meldet sich an einem XMPP-Server an und wartet auf Befehle. Der Experiment Controller (EC) ist die Software, die der Benutzer ausführt, um das Experiment zu steuern. Dazu liest dieser ein vom Benutzer geschriebenes Skript ein, welches in der OEDL[12]-Sprache verfasst ist. Es handelt sich hier um ein Ruby-Skript mit zusätzlichen, OMF-spezifischen Kommandos. Während des Experimentes sendet der EC die Befehle an verschiedene PubSub-Gruppen auf dem XMPP-Server, die wiederum von den RCs abonniert werden. Die RCs senden ggf. Antworten, auf die der EC dann reagieren kann. Abb. 4 zeigt die gesamte OMF-Systemarchitektur in der Gesamtheit und gibt eine Übersicht über die relevanten Prozesse und Knoten.

Eine weitere Komponente von OMF-Testbeds ist der Aggregate Manager (AM). Diese Komponente kann verschiedene Dienste im Testbed bereitstellen, z.B. den Dienst Chassis Manager (CM), der Knoten ein- und ausschalten kann, oder den PXE-Dienst, der Knoten über das Netzwerk booten kann. Der AM ist nicht zwingend für den OMF-Betrieb erforderlich und wird im Rahmen des VISA-Projektes wahrscheinlich wenig eingesetzt werden.

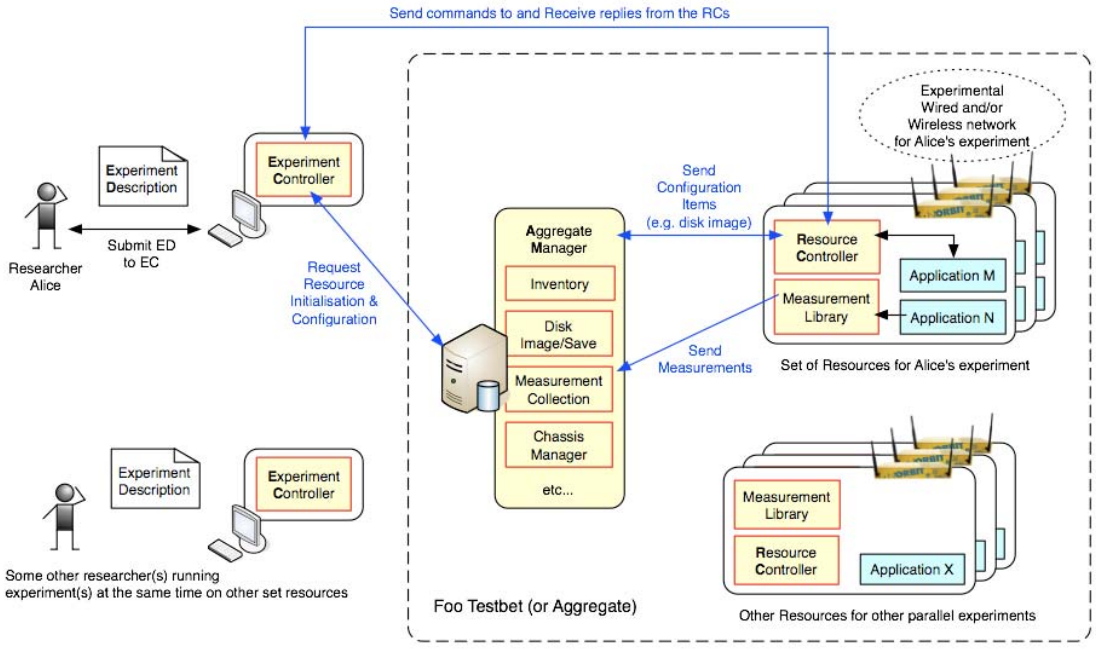

Abb. 4: Systemarchitektur von OMF

VISA verwendet das OMF-Framework, um die Experimente in den VSAs zu steuern. Auf jeder VSA läuft der OMF Resource Controller (RC), der es dem Experimentator

[12] OMF Experiment Description Language

ermöglicht Programme zu starten und Messungen vorzunehmen. Der OEDL-Code für den OMF Experiment Controller (EC) wird vom VISA Simulation Compiler (SC) generiert.

Um die Auswirkungen eines Experiments auf die VMs und Netzkomponenten zu messen, verwendete das VISA-Projekt die OML-Messbibliothek. Diese C-Bibliothek kann man gegen existierende Programme verlinken, in deren Quellcode dann Messpunkte definiert werden können. Zur Laufzeit lassen sich dann die Messpunkte an OML weiterreichen, woraufhin sie an einen OML-Server gesendet werden. Dieser Server speichert dann alle eingehenden Messpunkte von den verschiedenen Knoten in einer Datenbank für das jeweilige Experiment, was die Analyse nach dem Experiment vereinfacht. Eine Liste von bereits OML-fähigen Applikationen, Tutorials und Bindungen für weitere Sprachen sind auf der OML-Webseite[13] verfügbar.

In einem typischen Szenario im VISA-Projekt wurden eine Reihe von VSAs eingesetzt, um ein Unternehmen zu simulieren. Eine weitere VSA spielte dann den externen Angreifer, der eine Attacke auf das Unternehmensnetz vornahm. OMF kann hier verwendet werden, um Applikationen in den VSAs zu starten (z.B. einen Mailserver) und dann den Angriff einzuleiten (z.B. indem eine Flut von E-Mails von der Angreifer-VSA an das Unternehmen geschickt werden). Nebenbei kann OMF verschiedene OML-Applikationen starten, um z.B. Netzwerkauslastung oder Systemlast zu messen und damit die Auswirkungen des Angriffs auf die IT-Infrastruktur aufzeichnen. OMF erlaubt es, exakt dasselbe Szenario zu einem späteren Zeitpunkt erneut durchzuführen, um z.B. durchgeführte Sicherheitsmaßnahmen auf Wirksamkeit zu testen. OMF und OML bieten zur Analyse eine Reihe von Visualisierungen der Messdaten, um z.B. Graphen und Tabellen in Echtzeit während eines Experiments im Webbrowser darzustellen. Alternativ können die Rohdaten aus der OML-Datenbank (sqlite3 oder PostgreSQL) mit anderen Werkzeugen interpretiert werden.

OMF liefert somit ein Tool-Set, um eine Simulation entsprechend beschreiben und ausführen zu können sowie die gesammelten Daten abzuspeichern. Zusätzlich liefert OMF ein Service-Set, um die Plattform verwalten und betreiben zu können, wie beispielsweise das Neustarten von Netzknoten, Auslesen von Statusinformationen oder installieren neuer Betriebssystem-Images. Dabei sind die OMF-Tools und die Software nicht mit einer speziellen Testbed-Technologie verknüpft. Dadurch konnten sie auch für das VISA-Projekt angepasst und anschließend verwendet werden. [1]

2.8 Interconnected-asset Ontology (IO) Tool-Set

Das IO-Tools-Set von Fraunhofer SIT[14] stellte Methoden zur Verfügung, den Ist-Zustand einer IT-Infrastruktur zu akquirieren, aufzubewahren und automatischen Weiterverarbeitungsprozessen zur Verfügung zu stellen. Um die Anforderungen verschiedenartiger Verbraucher (z.B. dem Topologie Editor) zu gewährleisten, wurden

[13] http://oml.mytestbed.net
[14] http://www.sit.fraunhofer.de

IT-Asset-Informationen mit Hilfe einer ontologischen Repräsentation – der Interconnected-asset Ontology (IO) – semantisch verknüpft und vorgehalten. Die durch das IO-Tool-Set verwendete formale Repräsentation ist in der Lage komplexe und geschachtelte Netz-Topologien abzubilden, wie sie in Unternehmen typischerweise zu finden sind. Der aktuelle Entwicklungsstand des ontologischen Konzeptmodells (T-Box) ist online verfügbar. Als Datenformat für die Repräsentation der Ontologie kommt die Web Ontology Language (OWL) zum Einsatz.

IT-Asset-Informationen im Kontext des IO-Tool-Sets lassen sich in zwei übergreifende Kategorien unterteilen: Statische und flüchtige Informationen. Statische Informationen umfassen beispielsweise Hardware-Adressen, Seriennummern, aber auch Asset-Konfigurationen. Diese entstehen typischerweise durch einen manuellen Konfigurationsprozess, ggf. gestützt durch automatische Systeme. Als flüchtige Informationen werden im Kontext des IO-Tool-Sets beispielsweise dynamische Adresszuweisungen, Nachbarschaftsbeziehungen oder aktive Tunnel-Verbindungen zusammengefasst. Diese entstehen wiederum typischerweise durch automatische Konfiguration, ggf. basierend auf manueller Konfiguration. Per Definition des ontologischen Konzeptmodells werden IT-Assets in Form von semantischen Bezugsgraphen hinterlegt. Abb. 5 zeigt einen vereinfachten Ausschnitt eines einzelnen IT-Assets zur Verdeutlichung des formalen Modells.

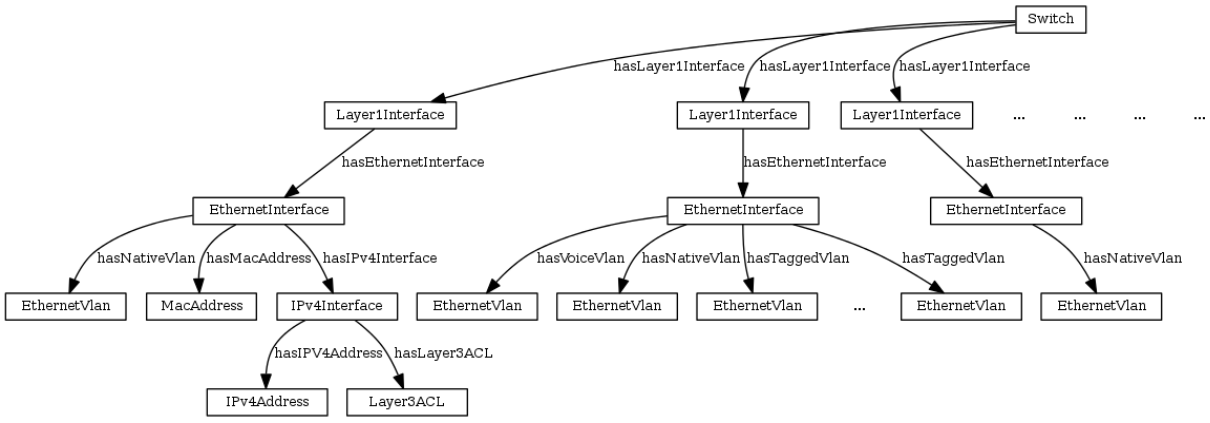

Abb. 5: **Vereinfachter Ausschnitt eines einzelnen IT-Assets**

Die Interconnect-asset Ontology (IO) bildete netzspezifische Informationen in hohem Detailgrad ab, was für den Einsatz als Simulation-Compiler erforderlich war. Dabei basierte die Modellierung der ontologischen Repräsentation auf verbreiteten Standards wie DMTF CIM (Distributed Management Task Force, Common Interface Modul) oder dem OSI-Schichtenmodell. Abb. 6 verdeutlicht den Detailgrad des formalen Modells.

Die in der Ontologie hinterlegten IT-Asset-Informationen wurden unter Verwendung von spezialisierten Query-Modulen in eine Simulationsdefinition konvertiert. Das Query-Modul ermittelte dazu die Parameter, die zur Erstellung der virtuellen Umgebung in OpenStack notwendig sind und führte eine entsprechende Konfiguration der Virtualisierungsumgebung durch.

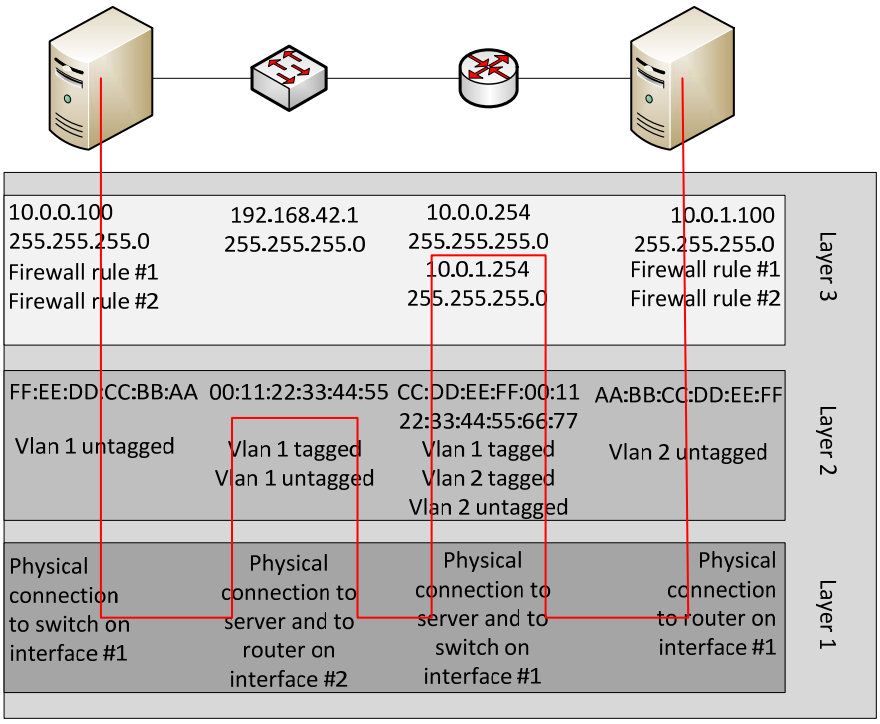

Abb. 6: Beispielhafte Darstellung eines Netz-Pfades

Änderungen an der in der Ontologie hinterlegten Netz-Topologie konnten mittels des Topologie Editors (TE) vor der Erstellung einer virtuellen Umgebung vorgenommen werden. Diesem Zweck diente das Austauschprotokoll IO-X. Es ermöglichte die Kommunikation zwischenverarbeitender Systeme mit dem IO-Tool-Set und wurde extra für die VISA-Plattform entwickelt.

Die Kommunikation zwischen TE und IO-Tool-Set fand mit Hilfe textbasierte Nachrichten statt, die über eine TLS-gesicherte TCP-Verbindung übertragen wurden. Jede Zeile in dieser Kommunikation enthielt genau eine Nachricht. Der grundlegende Ablauf der Kommunikation war dabei wie folgt:

1) Der TE erstellt ein Objekt vom Typ „Request“, welches mit dem gewünschten Kommando und den zur Ausführung notwendigen Argumenten (z.B. der zu speichernden Topologie für ein store_topology-Kommando) initialisiert wird.

2) Sofern das Objekt Daten enthält, die die im nächsten Schritt erfolgende YAML-Codierung stören könnten, werden diese Daten zur Gewährleistung der Datentransparenz in Base64-codiert.

3) Das Request-Objekt wird mittels der Yet Another Markup Language (YAML) codiert, so dass das Objekt als reine Zeichenkette dargestellt werden kann. Dieser Schritt stellt sicher, dass die Kommunikation unabhängig von der zur jeweiligen Implementierung verwendeten Sprache oder deren Version ist.

4) Der entstandene YAML-Text wird wieder zum Erhalt der Datentransparenz in

Base64 codiert, so dass in der YAML-Ausgabe enthaltene Zeilenumbrüche die Nachrichtenstruktur nicht zerstören. Das Ergebnis dieses Schrittes wird über die oben beschriebene Verbindung zum IO-Tool-Set gesendet.

5) Innerhalb des IO-Tool-Sets werden die erfolgten Codierungen in umgekehrter Reihenfolge decodiert, so dass am Ende dieses Vorgangs wieder das ursprüngliche Request-Objekt vorliegt. Die in diesem Objekt angeforderte Operation wird anschließend ausgeführt.

6) Das Ergebnis wird in einem Objekt vom Typ *Response* gespeichert, welches analog zum Request-Objekt innerhalb des TE codiert und zum TE gesendet wird.

7) Der TE decodiert die erhaltene Nachricht und wertet das Response-Objekt aus.

Die Abb. 7 verdeutlicht den beschriebenen Ablauf der Kommunikation.

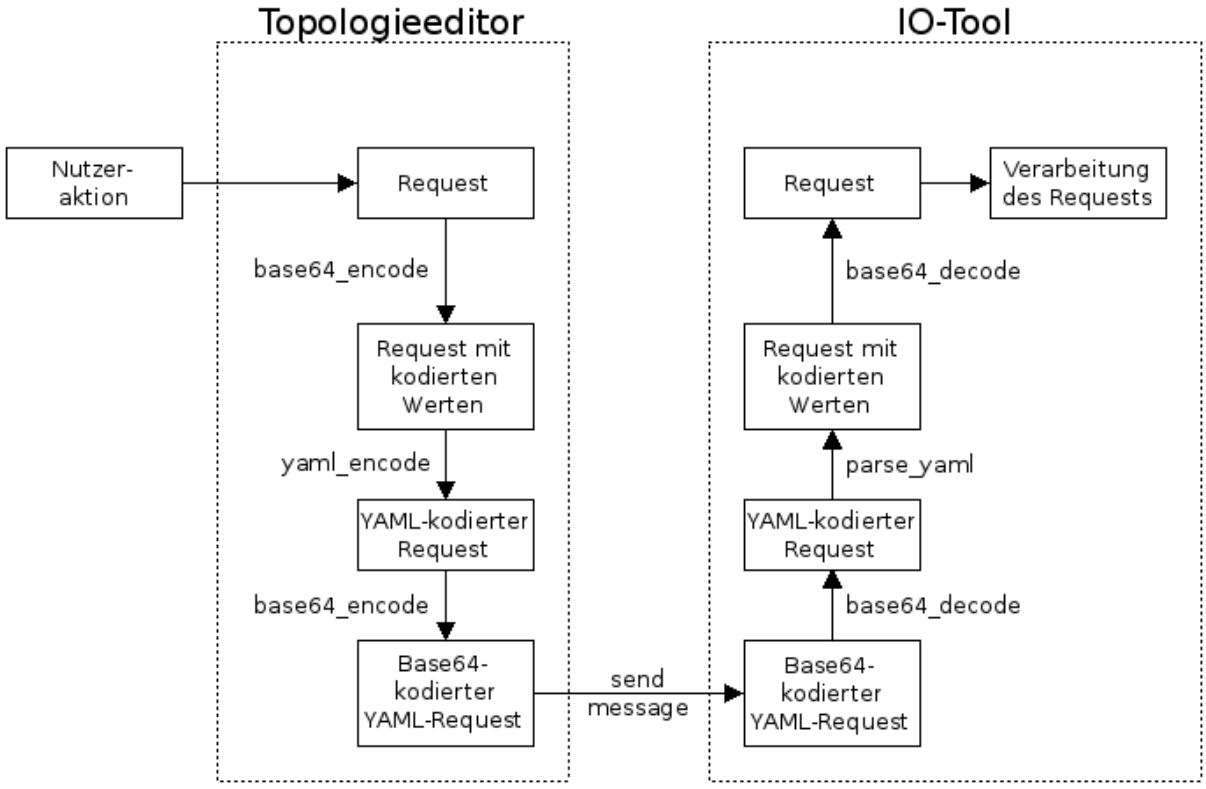

Abb. 7: Kommunikationsablauf zwischen TE und dem IO-Tool-Set via IO-X

Zwar kann dieses Protokoll das Transfervolumen um bis zu 75% erhöhen, erreichte damit aber auch hohe Robustheit und Flexibilität. Sofern innerhalb eines Request- oder Response-Objekts eine Netz-Topologie transportiert wird, kommt als Repräsentation das RDF-Datenformat zum Einsatz, das zwar gegenüber dem internen OWL-Datenformat eingeschränkt ist, jedoch eine vereinfachte Zwischenverarbeitung ermöglicht und so auf den Anwendungsbereich des TE speziell angepasst ist. [1]

2.9 VISA-Rahmenwerk

Das VISA-Rahmenwerk zielte darauf ab, Funktionalitäts- und Erreichbarkeitstest zu ermöglichen, nachdem eine neue Komponente in eine IT-Infrastruktur hinzugefügt wurde. Daraus ergeben sich die folgenden Bestandteile für das VISA-Rahmenwerk:

a. Topologie Editor (TE)

b. Simulation Compiler (SC)

c. Simulation Environment (SE)

In der **Simulationsumgebung i**st eine virtuelle Plattform vorhanden, die auf Basis von OpenStack, libvirt und KVM arbeitet. Während OpenStack zur Verwaltung der virtuellen Maschinen (VM) eingesetzt wird, ist jede VM auf Basis der KVM-Virtualisierungslösung im Betrieb. Die Bibliothek libvirt stellt hingegen eine einheitliche API zur Verfügung, damit verschiedene Virtualisierungen verwaltet werden können. OpenStack besitzt sowohl eine libvirt-Schnittstelle, als auch eine Möglichkeit virtuelle Netzwerkumgebungen auf Basis von Open vSwitch bereitzustellen.

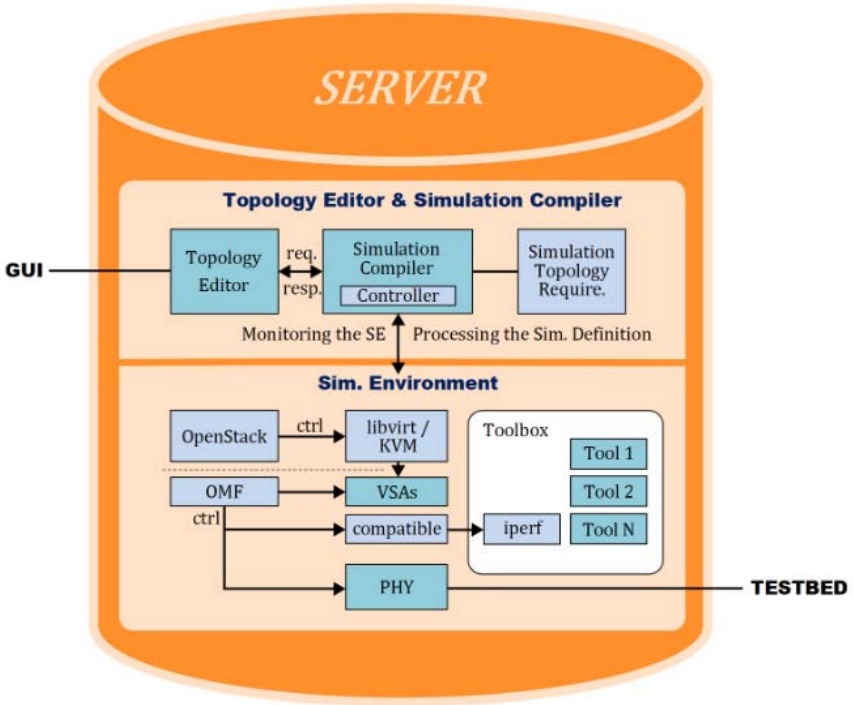

Abb. 8: **Aufbau des VISA-Rahmenwerks**

Die Aufgabe des **Simulation Compilers (SC)** wird durch das IO-Tool-Set übernommen. Zu diesen Aufgaben gehört die Übersetzung der formalen Beschreibung aus dem IO-Tool mithilfe von weiteren Parametern aus der Simulationsbeschreibung in eine Simulationsdefinition. Die Definition kann sowohl spezifische Daten über die teilautomatisierten Tests, als auch Konfigurationsdaten für die VMs enthalten. Das IO-

Tool-Set speichert die formale Beschreibung der IT-Infrastruktur in dem OWL/XML[15]-Format. Aus diesem Grund wird für die Kommunikation zwischen TE und IO das weniger komplexe Format RDF/XML[16] verwendet. Beide Formate basieren auf Triple und können dafür genutzt werden, um Topologien als Graphen darzustellen.

Ein Teil der Simulationsbeschreibung dient dem OMF-Rahmenwerk als Eingabeparameter. Der SC gibt die Simulationsdefinition weiter an einen Controller, der dann die Simulation ausführt. Dies kann nicht nur das Starten und Instanziieren der VMs beinhalten, sondern auch das Starten von OMF und den definierten Tests.

Der **Topologie Editor (TE)** bietet dem Benutzer die Möglichkeit eine bereits bestehende und vorher von dem IO-Tool-Set erhobene Topologie zu bearbeiten und dieser neue Komponenten hinzuzufügen. Weiterhin kann auch eine „neue“ bzw. bestehende Topologie von Hand nach-modelliert werden. Der TE besteht aus zwei Kernkomponenten: aus dem Back-end (in Java geschriebener Serverdienst) und dem Front-end (Web-basierte grafische Oberfläche). Letztere stellt die IT-Infrastruktur graphisch dar und ermöglicht Änderungen durchzuführen, die auch im Back-end automatisiert umgesetzt werden.

Das Back-end besteht wiederum aus drei Modulen:

a. Topologie-Modul

b. RDF-Modul

c. HTTP-Modul

Das **Topologie-Modul** stellt die Topologie, die entweder über das Webinterface oder aus den RDF/XML-Daten importiert wurde, mit Hilfe von Java-Klassen dar. Das RDF-Modul sorgt für die Verwaltung der RDF-Informationen, während das HTTP-Modul einen einfachen HTTP -Server zur Verfügung stellt, der die vom Front-end abgesetzten AJAX-Requests verarbeitet und beantwortet.

Bereits die vom **IO-Tool-Set** erhobenen IT-Infrastrukturen können nun entweder durch eine RDF/XML-Datei oder durch eine TLS-gesicherte TCP-Verbindung in den TE importiert werden. Ebenso besteht die Möglichkeit selbst erstellte oder veränderte IT-Infrastrukturen zu sichern. In einer späteren Version des TE soll auch es möglich sein, ohne IO-Tool-Set VMs direkt mit OpenStack anzulegen. Dies ist aber nicht mehr innerhalb des VISA-Projektes umgesetzt worden.

Zusätzlich ist es auch möglich vordefinierte Netzwerktests durch den Topologie Editor (TE) über die sog. **Toolbox** anzustoßen, um die virtuelle bzw. simulierte IT-Infrastruktur

[15] Web Ontology Language (OWL) / Extensible Markup Language (XML)
[16] Resource Description Framework (RDF) / Extensible Markup Language (XML)

auf verschiedene Parameter (z.B. Performance, Konfiguration) zu testen. Die Tests lassen sich in folgende Bereiche unterteilen:

a. Simulation von Angriffen

b. Messung der Auswirkungen

c. Emulation von Netzparametern

Das **cOntrol and Management Framework (OMF)** ermöglicht es, skriptgesteuerte Programme in den VMs zu starten, um Angriffe simulieren zu können. Es können dabei einfache Angriffe (z.B. eine Reihe von Flood-Pings zur Simulation einer DDoS-Attacke) oder auch komplexere Methoden (z.B. „root exploit“, gefolgt von einer Kommando-Serie) zum Einsatz kommen. OMF erlaubt es dabei, eine „Application Definition“ zu verwenden, die dann je nach Bedarf im OMF-Experiment parametrisiert werden kann. Damit lassen sich dann ganze Serien von Angriffen unterschiedlicher Stärke automatisiert abspielen. [1]

2.9.1 Automatisierte VSA-Konfiguration

Ein weiteres Ziel des VISA-Projektes war es, dem Benutzer es auf einfache Art und Weise zu ermöglichen, den Sicherheitsstatus seines Netzes zu erhöhen. Dafür wurden die VSA-MAC und die VSA-SRA (Abb. 9) entwickelt. Die VSA-Metadata-Access-Control (VSA-MAC) diente dem interoperablen Austausch sicherheitsrelevanter Daten zwischen beliebigen Netzwerkkomponenten auf Basis des IF-MAP-Protokolls. IF-MAP-Clients konnten über einen MAP-Server neue Metadaten veröffentlichen oder nach bereits vorhanden suchen. Des Weiteren erlaubt IF-MAP die Korrelation von Event-/Metadaten eines Netzes, um eine einheitliche Darstellung eines Netzwerkes und der aktiven Prozesse zu erhalten um daraus Anomalien erkennen zu können. Die VSA-Secure-Remote-Access (VSA-SRA) ermöglichte hingegen den sicheren Zugriff eines mobilen Mitarbeiters auf ein Unternehmensnetzwerk mittels eines Android-Smartphones. Als Basis diente die Spezifikation Trusted Network Connect (TNC).

Die beiden VSAs konnten über den TE einer bestehenden IT-Infrastruktur hinzugefügt werden. Aber auch die bereits vorkonfigurierten VSAs benötigen einige Informationen, die erst beim Instanziieren zur Verfügung stehen (z.B. IP-Adresse, Anpassen von Konfigurationsdateien). Um die einfache Einbindung gewährleisten zu können, mussten daher die VMs der VSA sich selbst konfigurieren können.

Folgende Informationen müssen zur Laufzeit ermittelt werden:

- IP-Adresse des Default-Gateways
- IP-Adresse des irond-Servers
- IP-Adresse des OpenVPN-Servers
- Irond-Server muss gestartet sein
- TNC-Server muss gestartet sein

- OpenVPN-Server muss gestartet sein
- Nagios-Server benötigt IP-Adressen der zu überwachenden VMs

Um dieses Ziel zu erreichen wurde im VISA-Projekt das Tool Puppet verwendet. Puppet ist ein Konfigurationsmanagement-Tool für Unix-basierte Betriebssysteme. Hierbei werden auf einem zentralen Server die Konfigurationen der verschiedenen Computer angelegt und verwaltet. Dazu werden am Anfang Templates deklariert, die einen bestimmten Zustand eines Systems beschreiben. Dieser Zustand kann Pakete, Dienste, Dateien oder auch das Ausführen von Konsolenkommandos beinhalten. Dabei eignet sich Puppet sowohl für einzelne Computer, als auch für Verbünde. Je nach Art kommt entweder ein Client-Server-Modell zum Einsatz oder ein reines Client-Modell.

In VISA wurde das Client-Server-Modell eingesetzt. Hierbei ist der Puppet-Server auf dem OpenStack-Hostsystem installiert und der Puppet-Client auf den VMs der VSAs. In dem VISA-Szenario mussten die IP-Adressen zur Laufzeit der VMs ermittelt werden, da OpenStack, in der verwendeten Essex-Version, keine Vergabe von statischen IP-Adressen unterstützt. Dafür wurde von Puppet ein OpenStack-Nova-Befehl auf dem OpenStack-Host ausgeführt und die Ausgabe in die relevanten Informationen zerlegt. Die so ermittelten IP-Adressen konnten in den Templates verwendet und so die benötigten Einstellungen angewandt werden.

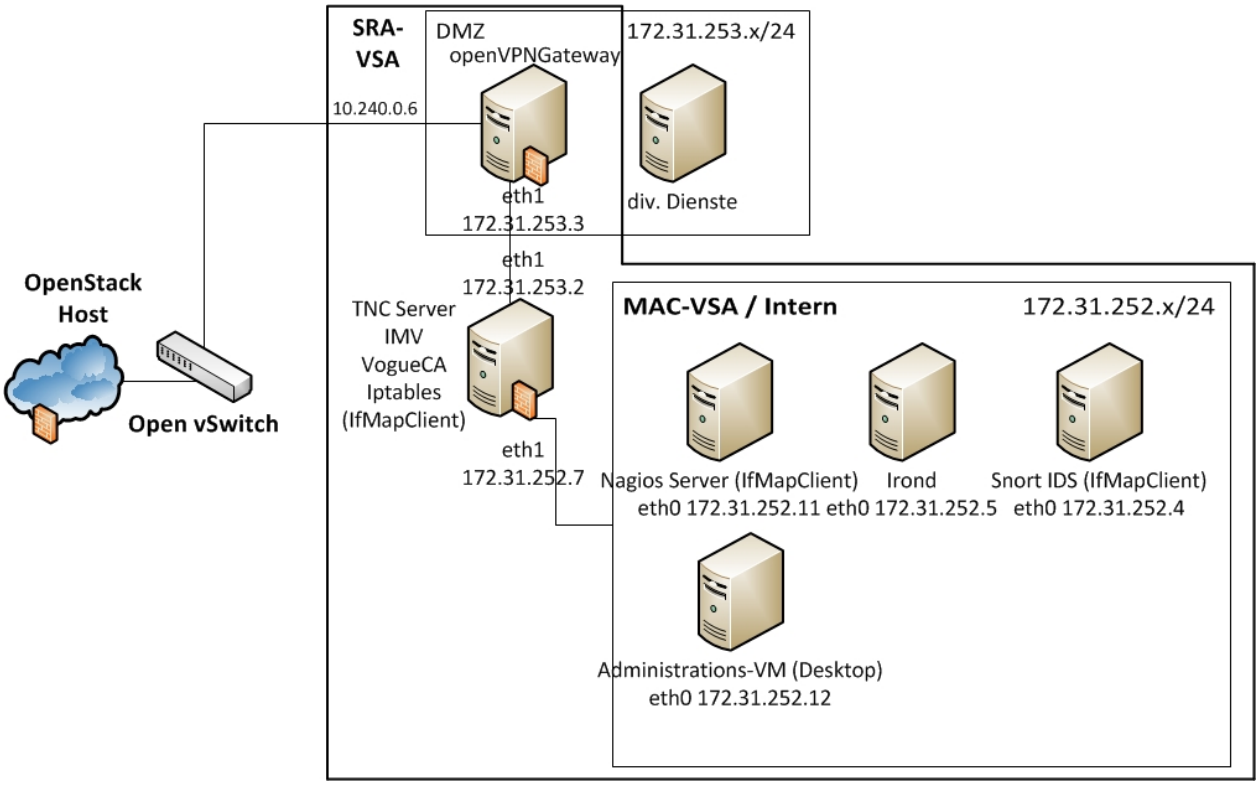

Abb. 9: Kombination der beiden VSAs der DECOIT GmbH

Außer bei der Gateway-VM wurden bei allen anderen VMs die Default-Gateway-Route

gelöscht und auf die Gateway-VM der VSA umgestellt. Weiterhin wurden bei der Nagios- und Snort-VM die entsprechenden IF-MAP-Clients gestartet. Bei der irond-VM wurde der irond-Server gestartet (MAP-Server). Auf der OpenVPN-VM wurde zusätzlich zum IF-MAP-Client auch der OpenVPN-Server gestartet. Bei der Gateway-VM wurde das IP-Forwarding aktiviert und die entsprechenden iptables-Richtlinien gesetzt. Des Weiteren wurden der TNC-Server und der dazugehörige IF-MAP-Client gestartet. Bei der Nagios-VM wurden die entsprechenden IP-Adressen, der zu überwachenden VMs gesetzt.

Das Ermitteln des VM-Typs (OpenVPN, Gateway etc.) erfolgte über den Hostnamen der VM. Hierzu war es notwendig, dass z.B. die Gateway-VM in ihrem Hostnamen das Schlüsselwort „Gateway“ enthielt. Hierbei konnten aber sowohl vor dem Schlüsselwort, als auch nach diesem weitere Zeichen vorkommen. Die Authentifikation am Puppet-Server folgte nach dem gleichen Schema. Wird von dem Benutzer eine VM hinzugefügt, die dieses Schlüsselwort enthielt, hatte dies keine weiteren Auswirkungen, da das „Dummy“-VM-Image keinen Puppet-Client enthielt und Puppet die Konfiguration nur dann übermittelte, wenn diese vom Client angefragt wurde. Abb. 10 zeigt die Schritte, die beim Starten der VSAs abgearbeitet werden. Wobei der erste Prozess auf dem Hostsystem ausgeführt wird und darauffolgenden auf den Client-VMs. [1]

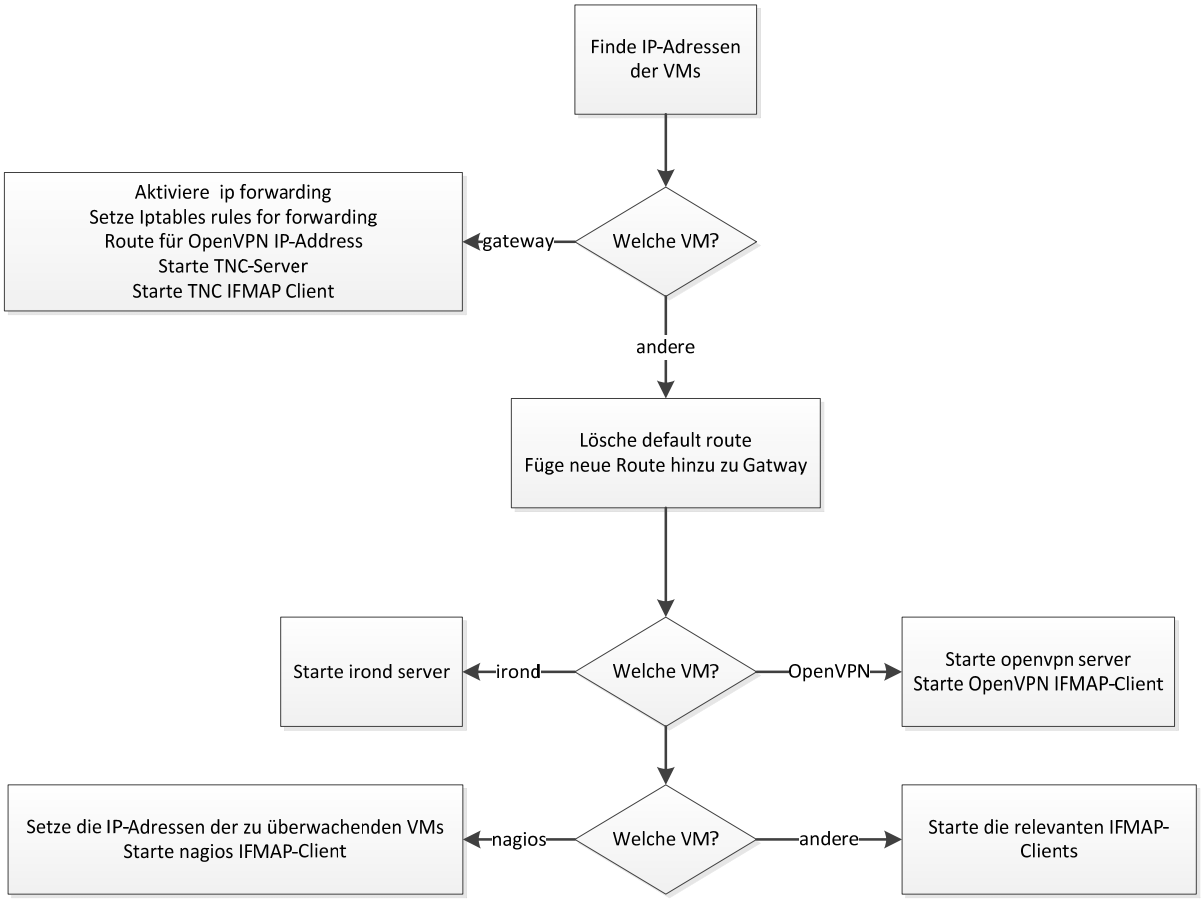

Abb. 10: **Ablauf der automatischen Konfiguration**

2.9.2 Konzeption und Steuerung von Netztopologien

Durch die starke Heterogenität von IT-Infrastrukturen, der relativ begrenzten Ressourcen sowie des relativ geringen technischen IT-Know-hows müssen in Zukunft Klein- und Mittelständische Unternehmen (KMU) bessere und geeignete Methoden zur flexiblen Konfektionierung, Erprobung und Optimierung ihrer IT-Infrastrukturen an die Hand bekommen. Dies ist insbesondere für die IT-Sicherheit wichtig. Um eine höhere Autonomie in der Konfiguration sowie im Betrieb ihrer IT-Infrastruktur zu erhalten, sind modulare und erprobte Lösungen und Systeme essentiell. Dies wurde im VISA-Projekt durch die Entwicklung von sog. Topologie-Editoren sowie Methoden und Tools zur Netzmodellierung/-simulation adressiert.

Virtualisierungslösungen haben sich inzwischen zu einer unverzichtbaren Methode in IT-Infrastrukturen entwickelt. Eine Vielzahl von Lösungen zur Administration und Monitoring existieren seit einigen Jahren am Markt. Diese unterscheiden sich jedoch in Bezug auf Funktionen, Zielgruppen, Komplexität und Umfang wesentlich. Viele Hersteller von Virtualisierungslösungen stellen darüber hinaus eigene Tools und inhärente Funktionen bereit.

Eine Möglichkeit zur Netzwerkkonfiguration ist die Erstellung virtueller Interfaces, mit denen man die Virtual Machines (VM) über Bridges verbinden kann. Bei diesen handelt es sich um eine direkte Weiterleitung des Datenverkehrs von einem Netzwerkinterface zu einem anderen. Diese Art der Verbindung erfolgt über eine Konfiguration der Netzwerkadapter des Betriebssystems und ist weit verbreitet. Mit Bridges und virtuellen Netzwerkadaptern ist es somit möglich, ein beliebig großes Sternnetzwerk aufzubauen. Dies spiegelt jedoch nur bedingt die Realität wieder, da beispielsweise keine Switches mit entsprechender Funktionalität abgebildet werden können. Auch lassen sich durch diese Art der Netzwerkvirtualisierung nur perfekte (unmittelbare, verlust- und fehlerfreie) Verbindungen erstellen. Da virtuelle Netzwerke auch zu Testzwecken benutzt werden, stellt dies eine Einschränkung der Funktionalität dar.

Graphische Tools zum Entwurf und Umsetzung von Netztopologien, die das Spektrum Layer 1 (Kabel) bis Layer 3 (Switches, Router) mitsamt VM abdecken sind für Anwender im KMU-Umfeld nicht zu finden oder zu komplex in der Bedienung. Es ist festzustellen, dass derzeit sehr wenige Lösungen zum Erstellen und Steuern von virtuellen Netztopologien existieren. Hierzu zählt beispielsweise GNS3, GPL[17]. Diese Software ist ein grafischer Netzwerksimulator, der auf dem Router-Emulator Dynamips[18] und dessen Front-end Dynagen[19] aufbaut. Dynagen ermöglicht, komplexe Router-Netzwerke mit Hilfe einer einfachen Konfigurationsdatei zu erstellen. GNS3 ist als eine GUI für Dynagen zu verstehen. Durch die grafische Unterstützung lassen sich mit GNS3, auch ohne Kenntnisse der Dynagen-Syntax, sehr komfortabel komplexe virtuelle Netzwerke erstellen.

[17] http://www.gns3.net
[18] http://www.ipflow.utc.fr/index.php/Cisco_7200_Simulator
[19] http://www.dynagen.org

Aufgrund des Mangels an Lösungen und Produkten wurden im Forschungsprojekt VISA mit unterschiedlichen Ansätzen drei Topologie-Editoren entwickelt:

a. Virtual Wizard (VirWi) von der FH Dortmund

b. VISA Topologie Editor (V-TE) von der DECOIT GmbH

c. Spotlight von der Collax GmbH

Diese unterschiedlichen Topologie-Editoren, die entwickelt wurden, um eine flexiblere Möglichkeit zur graphischen Konzeption von virtuellen IT-Sicherheitsinfrastrukturen zu schaffen, werden im Folgenden genauer vorgestellt und miteinander kurz verglichen.

2.9.2.1 Virtual Wizard (VirWi)

Der Topologie-Editor VirWi der FH Dortmund ist eine Client-Server-basierte Software, die der Darstellung, Verwaltung und Steuerung einer vollständigen virtuellen Netzwerkumgebung dient. Die eigentliche Virtualisierung und Steuerung wird hierbei von KVM (Kernel-based Virtual Machine) und VDE (Virtual Distributed Ethernet) geleistet. KVM ist eine Open-Source-basierte Virtualisierungslösung für Linux, welche Vollvirtualisierung auf x86-Hardware ermöglicht. Hierbei werden die Befehlssatzerweiterungen der modernen Prozessoren Intel VT und AMD-V genutzt. Das Verfahren der Vollvirtualisierung ermöglicht es, mehrere unveränderte Linux- und Windows-Betriebssysteme parallel auf einem System zu betreiben. KVM baut auf dem Emulator QEMU auf. QEMU kann verschiedene Prozessorarchitekturen emulieren, hierzu zählen PowerPC, ARM, Alpha, m68k, MIPS und Sparc. Er stellt weiterhin die virtuelle Hardware (z.B. virtuelle Netzwerkinterfaces) den VMs bereit. VDE stellt eine allgemeine, virtuelle Infrastruktur zur Verbindung verschiedener Softwarekomponenten zur Verfügung. Es lassen sich VMs verschiedener Virtualisierungslösungen, Emulatoren, reale Betriebssysteme und Netzwerke miteinander verbinden. Auf Basis von VDE lassen sich sehr einfach und flexibel virtuelle Netzwerke erstellen. Teilbereiche solcher virtuellen Netzwerke lassen auf mehrere physikalische Rechner verteilen. VDE ist Ethernet-konform und stellt virtuellen Infrastrukturen virtuelle Switche und virtuelle Kabel zur Verfügung.

Mit VirWi lassen sich sowohl einzelne Virtuelle Maschinen (VM) als auch ganze Netzwerktopologien konfigurieren. Die Planung (das Editieren) des Netzwerkes erfolgt über eine graphische Oberfläche (WYSIWYG). Die Netzwerkpläne werden dann von der Software auf einem KVM-Server direkt umgesetzt.

Die meisten der geplanten Leistungsmerkmale sowie Anforderungen an den VirWi-Editor konnten umgesetzt werden. Dies sind u.a.:

a. **Verwalten von virtuellen Festplatten:** Virtuelle Festplatten können erfasst und verwaltet werden. Beliebige Hardware kann zu einer VM hinzugefügt und entfernt werden.

b. **Erstellen von VMs:** VMs können erstellt und konfiguriert und verwaltet werden. Die jeweilige Hardware der VM kann durch einfache Menüs freigestaltet werden.

c. **Remotezugriff:** Die gesamte Steuerungs-GUI wird nicht auf dem KVM-Server selbst, sondern per Remotezugriff auf einem beliebigen Client per VNC ausgeführt.

d. **Speicherung:** Einzelne VMs wie auch Netzwerkpläne können abgespeichert werden.

e. **Verwalten von Bridges:** Bridges in der Netzwerkkonfiguration des Server-Hosts können verwaltet und diese mit echten Interfaces verlinkt werden.

f. **Verwalten von Netzwerken:** Das Einsehen, Verwalten, Löschen, Laden und Speichern von Netzen ist möglich. VMs und Switches können im Betrieb an-/abgeschaltet werden.

g. **Migration:** Netzwerkpläne können auf mehrere Server verteilt werden. Die Kommunikation zwischen den Servern erfolgt verschlüsselt.

h. **Layout:** Farben und Formen der Oberflächengestaltung kann man durch Stylesheets ändern. Die Icons der VMs lassen sich frei wählen, im Gegensatz zum Steuerungslayout.

Die Softwareumgebung besteht aus einer Client- und einer Server-Software. Beide bestehen aus Java-Code und benötigen die JRE von Sun Microsystems bzw. Oracle. Das Programm ist damit plattformunabhängig. Der Server benötigt VDE, selbst kompiliertes KVM mit VDE-Support sowie Java mit SSL-Support. Zur Steuerung der Switches und VMs im laufenden Betrieb sollte auf dem Server ein SSH-Dämon laufen und Programme wie unixterm, nc und ssh zur Verfügung stehen. Darüber hinaus sollten die TAP-Interfaces Zugriffsrechte für den User haben, unter dem das Server-Programm läuft.

Als Hardware-Umgebung reicht ein handelsüblicher PC. Je nach Größe der zu erstellenden Netzwerke ist für die Serverkomponente ein leistungsstarkes Serversystem mit entsprechender Netzwerkanbindung, Prozessorkernen und Speicher vonnöten.

Im VM-Manager werden alle virtuellen Maschinen (VM), die in der Datenbank gespeichert sind, verwaltet. Jedes Netzwerk erhält beim Start automatisch alle TAP-Interfaces des Betriebssystems. Wird ein Switch an ein TAP-Interface angeschlossen und ist dieses auf dem Server korrekt mit dem echten Interface „gebridged“, so ist es möglich, das echte Netzwerk des Servers zu erreichen. Durch Nichtverbinden der TAP-Interfaces erstellt man ein isoliertes Netzwerk.

Kabelverbindungen lassen sich einfach erstellen. Besitzt ein Element mehr als ein Interface, er-scheint ein Popup-Menü, in dem man die Interface-Nummer, an der das Kabel angeschlossen werden soll, auswählen kann. Jede Verbindung muss mindestens

einen Switch enthalten, d.h., erlaubte Verbindungen sind: Switch-Switch, Switch-VM, Switch-Complex und Switch-TAP.

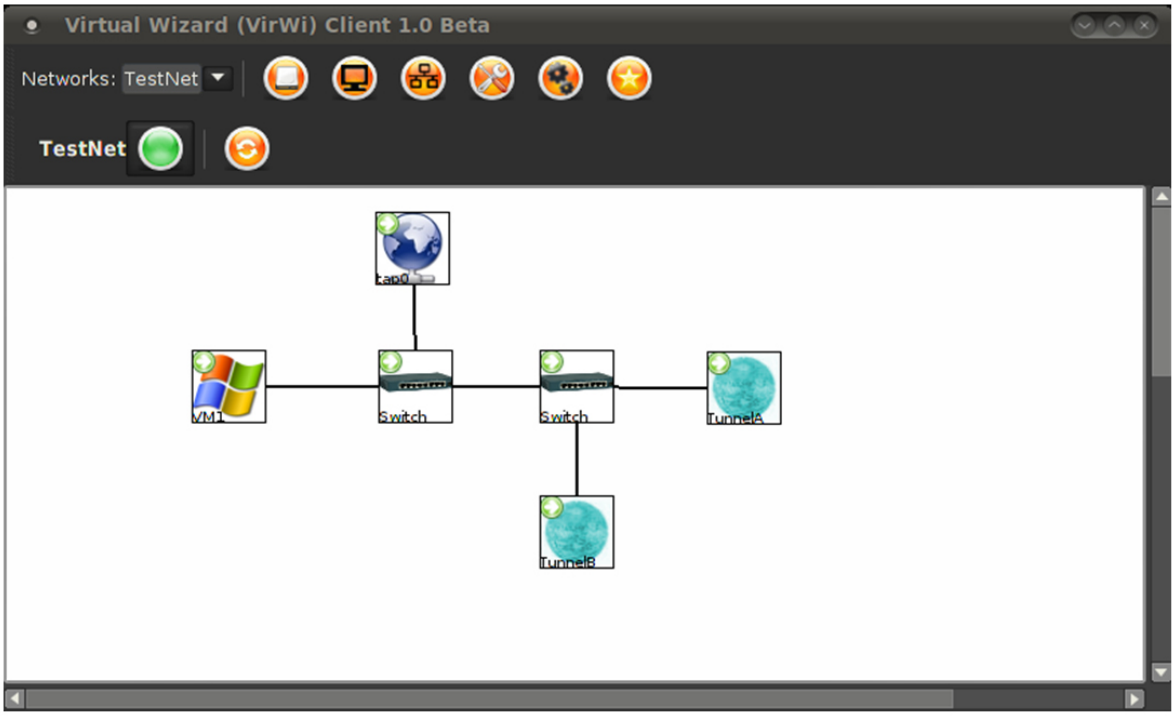

Abb. 11: VirWi-GUI

Die sog. Complex Elements (CE) stellen eine weitere Option dar. Sie werden wie Switches in das Netzwerk eingefügt. Jedoch müssen bei der Erstellung ein Name und ein Befehl angegeben werden. Bei dem Befehl handelt es sich um einen beliebigen Prozess, der auf dem Server ausgeführt wird. Der Prozess wird beim Netzwerkstart mit gestartet und mit ihm zusammen wieder beendet. Der „Std Input Stream“ und „Std Output Stream“ des Prozesses wird durch die Kabelverbindung hergestellt.

Ein Beispiel für ein sinnvolles CE ist ein Netzwerktunnel. In Netzwerk A wird ein CE mit dem Befehl „nc -lc localhost 10010“ erstellt und durch einen Switch verbunden. In Netzwerk B wird ein CE mit dem Befehl „nc -c localhost 10010“ erstellt und ebenfalls verbunden. Beim Starten von A und anschließend B wird mithilfe des „nc“ Befehls automatisch ein Tunnel zwischen diesen Netzwerken aufgebaut. Somit ist es möglich, zwei Netzwerke auf zwei unterschiedlichen Servern zu verbinden. In diesem Beispiel muss (aufgrund der Funktionalität des „nc“-Befehls) Netzwerk A stets vor Netzwerk B gestartet werden. Anstatt „nc“ kann z.B. auch „SSH“ benutzt werden, um einen verschlüsselten Tunnel aufzubauen. Da VDE-Switches jeden beliebigen Prozess akzeptieren, sind den Anwendungsmöglichkeiten von CEs keine Grenzen gesetzt.

Ein Bearbeiten des Netzwerkes im laufenden Betrieb ist nicht möglich. Jedoch kann durch Klicken auf einen Switch oder eine VM im Online-Modus ein Verwaltungsmenü aufgerufen werden. Es ist somit möglich, einzelne VMs zu starten oder zu stoppen, die Log-Datei der VM einzusehen und per VNC auf diese zuzugreifen. Hierzu sollten die

Angaben zum externen VNC-Client unter „Client Konfiguration“ überprüft werden. Wenn beim Login eine SSH-Verbindung angegeben wurde, ist es auch möglich, sich direkt mit dem Monitor der VM zu verbinden.

Bei Switches kann ebenfalls die Log-Datei eingesehen und bei vorhandener SSH-Verbindung auf den Kontrollsocket zugegriffen werden. Mithilfe des nun roten Online/Offline Buttons kann das gesamte Netzwerk wieder heruntergefahren werden. Nach dem Herunterfahren wechselt die Benutzeroberfläche wieder in den Offline-Modus. Um Änderungen an einer VM im laufenden Betrieb vorzunehmen existiert eine sog. Monitor-Console in „qemu“. [2]

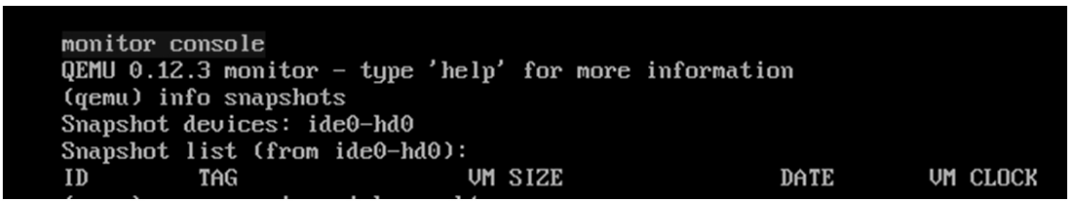

Abb. 12: Monitor-Konsole des QEMU

2.9.2.2 VISA Topologie Editor (V-TE)

Der VISA Topologie Editor (V-TE) der DECOIT GmbH bietet dem Benutzer die Möglichkeit, eine bereits bestehende und vorher erhobene Topologie zu bearbeiten sowie neue Komponenten hinzuzufügen. Weiterhin kann auch eine neue bzw. bestehende Topologie von Hand nachmodelliert werden. Der V-TE besteht dabei aus zwei Kernkomponenten:

a. **Back-end:** ein in Java geschriebener Serverdienst

b. **Front-end:** eine Web-basierte grafische Oberfläche

Bereits erhobene IT-Infrastrukturen können entweder durch eine RDF-/XML-Datei oder durch eine TLS-gesicherte TCP-Verbindung in den V-TE importiert werden. Ebenso besteht die Möglichkeit, selbst erstellte oder veränderte IT-Infrastrukturen zu sichern. Zum Erheben einer IT-Infrastruktur ist das Interconnected-asset Ontology (IO) Tool von Fraunhofer SIT notwendig, dass ebenfalls im Forschungsprojekt VISA entwickelt wurde. Das sog. IO-Tool-Set ist eine Software, die es ermöglicht, den Ist-Zustand eines Netzes, d.h., die Konfiguration von Komponenten und die Verbindungen der Komponenten untereinander, zu erheben. Diese Daten werden in ein formales Datenmodell überführt und mit Hilfe dieser Daten Anfragen zum Zustand des Netzes beantwortet. Kern des Systems ist eine Ontologie, die zum einen das formale Datenmodell darstellt, aber auch gleichzeitig die Daten dieses Datenmodells enthält. Im Rahmen des VISA-Projekts stellt das IO-Tool-Set Methoden bereit, die dem V-TE wiederum Topologien bereitstellen. Dadurch lassen sich Änderungen speichern, vorhandene Quellumgebungen erheben oder erhobene Umgebungen replizieren. Eine ausführlichere Darstellung des IO-Tool-Sets ist in dem zweiten Beitrag „Design und Implementierung von Virtual Security Appliances (VSA)“ zu finden.

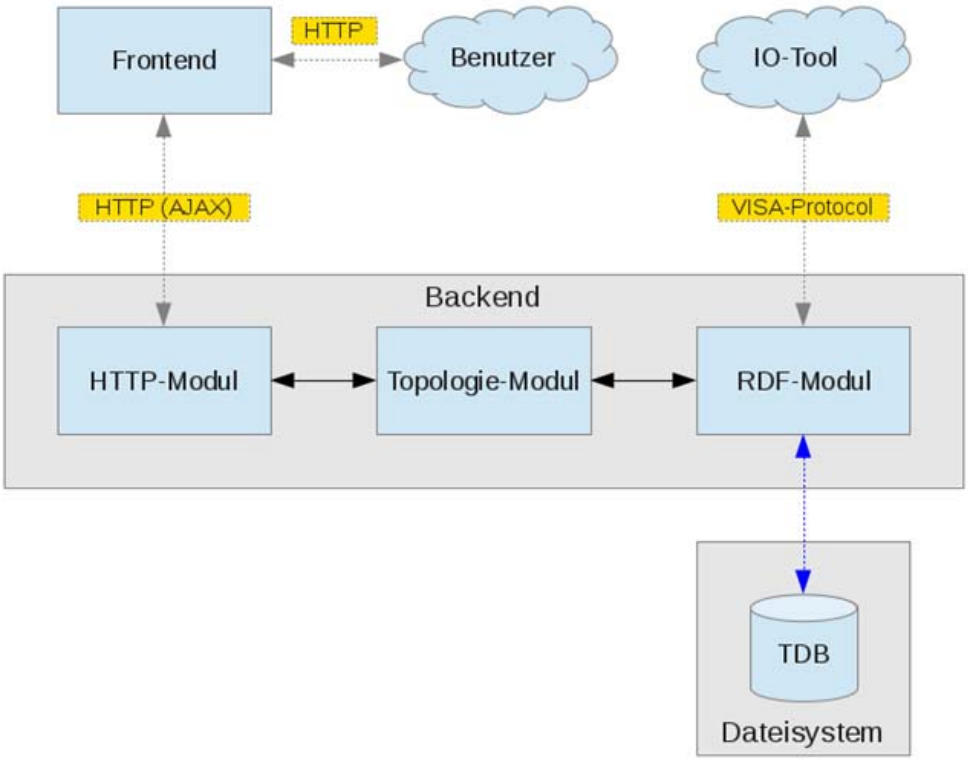

Abb. 13: Architektur des VISA Topologie Editors (V-TE)

Wie in Abb. 13 zu sehen ist, besteht das Back-end aus drei Modulen: Topologie-, RDF- und HTTP-Modul. Das Topologie-Modul stellt die Topologie, die entweder über das Webinterface oder aus RDF/XML-Daten importiert wurde, mit Hilfe von Java-Klassen dar.

Das zentrale Element dieses Moduls ist die abstrakte Klasse „NetworkComponent“, die als Basisklasse aller Komponenten in der Topologie dient. Als Komponente werden alle physikalischen und virtuellen Geräte bezeichnet, also z.B. Switches, Router, Firewalls und Server, die wiederum in drei Unterklassen sich einteilen lassen. Diese grobe Aufteilung ist notwendig, da das IO-Tool vom Fraunhofer SIT, welches die Topologie erhebt, nicht in der Lage ist feinere Unterscheidungen zu machen. Neben den Komponenten gibt es noch Klassen für Netzwerkschnittstellen und Kabel. Die Klasse für Netzwerkschnittstellen ist als innere Klasse von „NetworkComponent“ implementiert und heißt „Interface“. Sie enthält Eigenschaften, die für jedes Interface unterschiedlich sind. Zum Beispiel die Konfiguration für das Internet Protocol (IP), also Adresse, Subnetzmaske, IP Version und Netzwerkadresse.

Abb. 14 zeigt die Struktur, die im Topologie-Modul erzeugt wird. Jedes Objekt der Klasse „NetworkComponent“ besitzt ein oder mehrere Objekte von „Interface“, die die Schnittstellen dieser Komponente darstellen. Die Interface-Objekte werden wiederum mit Objekten der Klasse „NetworkCable“ verknüpft, womit die Komponenten, zu denen die beiden Ports gehören, miteinander verbunden sind. In dieser Topologie könnte also

„NetworkComponent 1“ ein Switch sein, der die Geräte „NetworkComponent 2“ und „NetworkComponent 3“ miteinander verbindet.

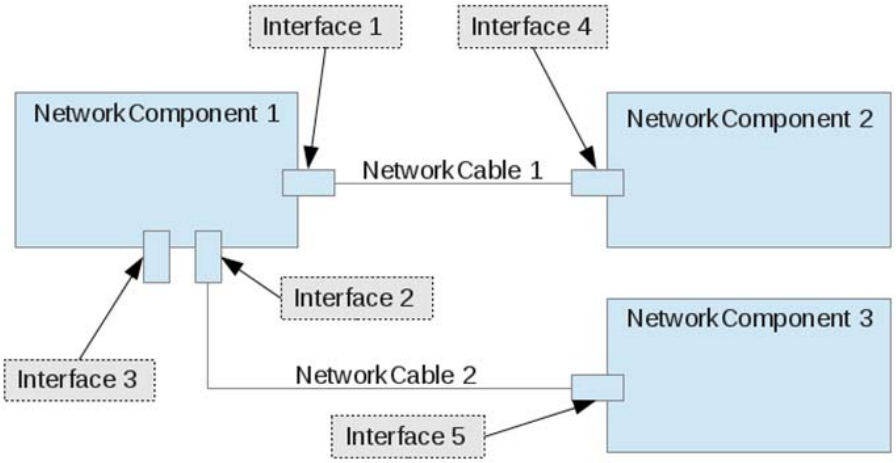

Abb. 14: Schematische Darstellung der im Topologie-Modul erzeugten Objektstruktur

Das RDF-Modul sorgt für die Verwaltung der RDF-Informationen. Dazu baut es auf dem Open Source Framework Jena[20] auf, das von der Apache Foundation entwickelt wird. Die zentrale Klasse des Moduls ist der RDF-Manager. Wird ein Objekt dieser Klasse erzeugt, wird ein sogenanntes „Dataset“ erstellt, welches die RDF-Informationen speichert und verwaltet. Ein Dataset enthält mindestens ein RDF-Modell, das „Default Model“. Neben diesem können noch beliebig viele „Named Models“ in einem Dataset enthalten sein. Das im RDF-Manager verwendete Dataset verwendet das Datenbanksystem TDB als Speichermedium, welches von Jena mitgeliefert wird. Dieses erlaubt die Verwendung von Transaktionen für jeglichen Zugriff auf die gespeicherten Modelle, so dass im Falle eines Fehlers bei der Verarbeitung kein dauerhafter Schaden an den Daten entstehen kann.

Das dritte Hauptmodul ist das HTTP-Modul. Dieses stellt einen einfachen HTTP-Server zur Verfügung, der die vom Front-end abgesetzten AJAX-Requests verarbeiten und beantworten kann. Die zentrale Klasse dieses Moduls ist der AJAX-Server. Der AJAX-Server startet eine Instanz dieses einfachen HTTP-Servers und definiert die HTTP-Handler, die die einzelnen AJAX-Requests bearbeiten.

Das Front-end des V-TE wurde als Web-Oberfläche, größtenteils in JavaScript, entwickelt. Im oberen Bereich der Oberfläche (Abb. 15) unterhalb der Kopfzeile, befindet sich die Optionsleiste. Sie erlaubt verschiedene Einstellungen und gibt Zugriff auf diverse Funktionen des Editors. So lässt sich hier die automatische Wegfindung für Kabel auf dem Editor-Raster ein- und ausschalten. Die zweite Option erlaubt das Einblenden der vollen Namen aller Komponenten auf dem Raster und die dritte das Einfärben der Kabel

[20] The Apache Software Foundation: Apache Jena Framework. http://jena.apache.org/index.html

entsprechend des VLANs zu dem sie gehören. Diese beiden Optionen können zeitgleich aktiviert werden und blockieren jeweils die Drag-and-Drop-Funktionalität des Editors um Anzeigeprobleme zu verhindern.

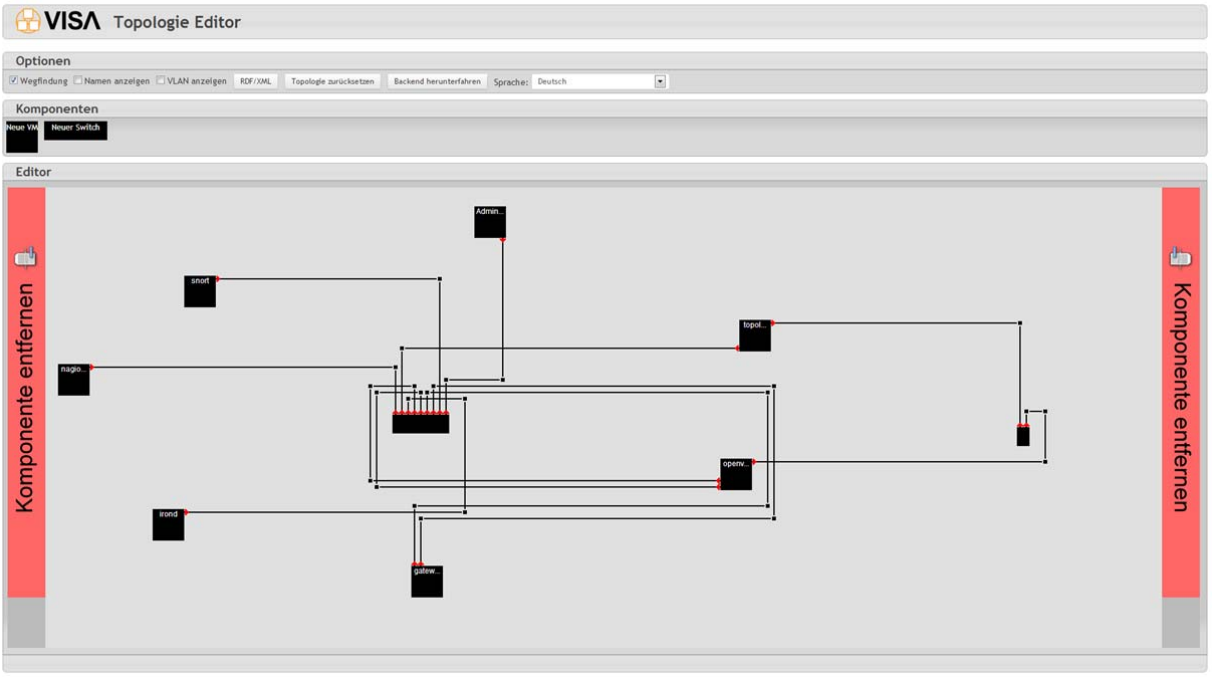

Abb. 15: Startseite des VISA Topologie Editors (VTE)

Aktive Komponenten werden als schwarze Boxen dargestellt. Beim Hinzufügen neuer Komponenten stehen dem Benutzer folgende Optionen zur Verfügung: der Name, die Breite/Höhe, die Anzahl der Netzwerkschnittstellen und die Position der Schnittstellen an der dargestellten Box.

Die Möglichkeit mehrere Komponenten zu einer Gruppe zusammenzufassen ist eine der zentralen Funktionen des V-TE. Da das Raster, auf dem die Komponenten abgebildet werden, relativ klein ist, reicht der Platz höchstens für ca. 20 Geräte. Danach wird die Darstellung unübersichtlich oder ist überhaupt nicht mehr möglich, da sich die Komponenten nicht mehr überlappungsfrei platzieren lassen. Um trotzdem größere Topologien darstellen zu können, werden Komponenten, die bestimmte Eigenschaften teilen, zusammengefasst. Die Gruppen werden im Editor als Objekte dargestellt und nehmen dadurch relativ wenig Platz ein. Ein Klick auf die Gruppe öffnet diese in einem weiteren Raster und erlaubt so die Einsicht der Inhalte dieser. Zurzeit können die Komponenten nur automatisch vom Back-end in Gruppen aufgeteilt werden. Eine manuelle Konfiguration der Gruppen wird eventuell zu einem späteren Zeitpunkt noch stattfinden. [2]

2.9.2.3 Spotlight

Spotlight ist der Topologie-Editor, der innerhalb des VISA-Projektes von der Collax GmbH entwickelt wurde und daher genau auf die Collax-Produkte abgestimmt ist. Er

soll Administratoren das zentralisierte Verwalten von verteilten Server-Umgebungen erlauben und die Evaluation im VISA-Projekt unterstützen, indem bereitgestellte Werkzeuge im TE mit den eingesetzten Virtual Security Appliances (VSA) kommunizieren. VSAs sind Sicherheitskomponenten, die aus einer oder mehreren VMs bestehen können und auf virtueller Basis einen Sicherheitsdienst anbieten. Konfigurationen der VSAs sind im TE editierbar und VSA-Informationen können abgerufen werden. Die verwalteten VSAs werden in einer Datenbank hinterlegt und lassen sich über eine GUI hierarchisch strukturieren. Für den Datentransfer zwischen Spotlight und den VSAs sind auf den VSAs entsprechende Agenten notwendig.

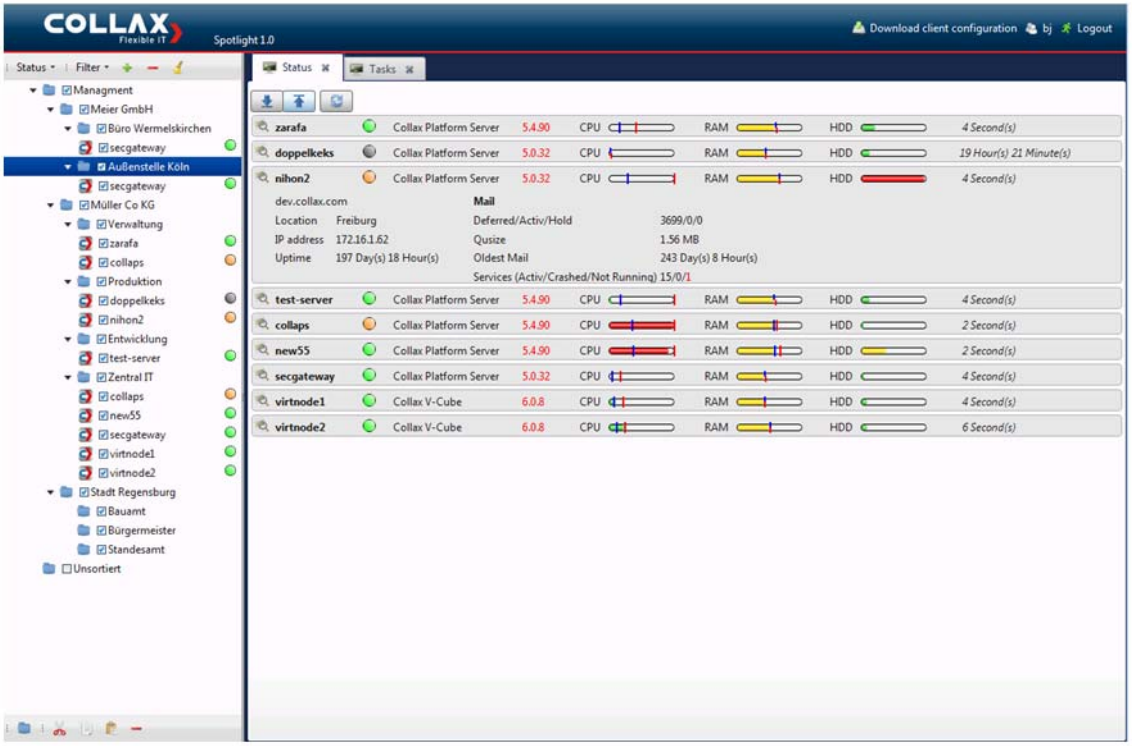

Abb. 16: Übersichtsdarstellung ausgewählter Server

Realisiert wurde Spotlight als eigenständiges Modul für die Collax-Server-Plattform. Neben dem nicht-virtualisierten Betrieb auf dedizierter Hardware wurde Spotlight auch für den Betrieb als VSA vorbereitet. Dazu wurde die Unterstützung der Paravirtualisierung integriert, die für den virtuellen Betrieb modifizierte Konfiguration hinterlegt und das automatische Deployment als VM, basierend auf dem Template, umgesetzt. Die so bereitgestellte VSA wird über das Management-Interface als virtuelle Instanz in Betrieb genommen.

Die grafische Administrationsoberfläche von Spotlight wurde, um plattformunabhängig als Web-Anwendung einsetzbar zu sein, mit Hilfe des Ajax-Frameworks Qooxdoo[21]

[21] http://www.qooxdoo.org

implementiert. Qooxdoo ist eine unter der L-GPL lizenzierte Programmbibliothek für die Implementierung von Web-basierenden Applikationen, die über einen Web-Browser gesteuert werden können. Es stellt zahlreiche in JavaScript vorliegende Komponenten zur Verfügung, die das Look-and-Feel einer klassischen Desktop-Anwendung vermitteln. Gemäß dem Ajax-Grundgedanken kommuniziert Qooxdoo zwischen dem Server und einem Browser über eine asynchron geführte Datenverbindung. Dies erlaubt es, Änderungen in der Darstellung der Anwendung durchzuführen, ohne die Seite komplett neu zu laden.

Die Darstellung der Topologie eines Netzwerks ist in zwei Bereiche aufgeteilt: in einem werden die verwalteten Server in einer Baumstruktur dargestellt, im zweiten kann man die Detailansicht eines ausgewählten Servers oder eine Übersicht mehrerer Server sehen (siehe Abb. 16). Betrifft die vom Administrator durchzuführende Aufgabe nur einen Teil der Server, kann man die hierarchisch angeordnete Server-Liste mit Hilfe von Filtern beschränken. Dabei kann die Auswahl auf bestimmte Ebenen der Hierarchie reduziert werden. Es können aber auch beliebige Attribute (z.B. CPU-Auslastung, Update-Stand) der Server für die Filterung herangezogen werden. Beide Filtermechanismen lassen sich auch miteinander kombinieren.

Auf einem Server aufgetretene Probleme und Warnungen werden gewichtet und dem Administrator in Form eines Statuslämpchens farbcodiert als konsolidierte Information angezeigt. Spotlight verfügt je Server über drei Ebenen der Informationsaufbereitung. Die oberste Ebene fasst die wichtigsten Informationen zusammen, die unterste gibt Auskunft über jedes Detail. Auf jeder Ebene können die Informationen ausgewählter Server gegenübergestellt werden. In der Detailansicht besteht die Möglichkeit, Auswertungen, Statistiken und Statusinformationen abzurufen. Außerdem stehen Funktionen zur Verfügung, um Kommentare zu einem Server zu hinterlegen und Kommandos remote auszuführen.

Für die Speicherung und Verwaltung aller Daten der Remote-Server wird die relationale, GPL-lizenzierte MySQL-Datenbank verwendet. Hierzu wurde die bestehende Implementierung mit der Datenbank-Engine InnoDB integriert. Werden Daten von einem Agenten eines Remote-Servers empfangen, werden sie mit Referenz zum entsprechenden Server als Tabellen abgelegt. Ebenfalls werden die Tasks und die Kommentare in der Datenbank gespeichert. Offene Tasks werden bei der nächsten Verbindung zum entsprechenden Agenten übermittelt.

Die Verbindung zwischen Spotlight und den Agenten wird von Letzteren initiiert und zertifikatsbasierend, auf Basis von AES256-SSL, verschlüsselt. Hierzu wird auf die Open-Source-lizenzierte Bibliothek openSSL zurückgegriffen. Die Schnittstelle für die Kommunikation ist als Netzwerk-Dämon realisiert. Der in „C“ programmierte Dämon überwacht auf der Protokollebene TCP/IP den Port 443 für https. Die Verbindungen werden durch ein Modul für den Apache Web-Server authentisiert. Das Modul ist ebenfalls in „C“ geschrieben und wird beim Start von Apache geladen. Die Authentisierung der Verbindung wird anhand eines X.509-Zertifikats durchgeführt.

Für die Kommunikation zwischen Spotlight und den verwalteten Servern wurden

Agenten implementiert. Für die Erhebung und Bereitstellung der Daten aus verschiedenen Quellen des Remote-Servers ist ein Plug-In-Framework für die sog. Kollektoren notwendig. Die Kollektoren werden von dem Agenten aufgerufen und übergeben diesem ihre Daten. Der Agent steht mit Spotlight in Verbindung und überträgt die gesammelten Daten. Er stellt eine Laufzeitumgebung für Plug-Ins (Kollektoren) bereit und definiert die Schnittstelle für den Austausch der Daten. Kollektoren haben zwei obligatorische und eine optionale Komponente:

a. **Konfigurationsdatei:** Die Konfigurationsdatei enthält eine Beschreibung der Daten und Informationen, die der Kollektor bereitstellen wird.

b. **Executable:** Die ausführbare Datei, die die Datenerhebung durchführt und die Daten zur weiteren Verarbeitung bereitlegt.

c. **Kommandos (optional):** Über zusätzliche Kommandos können notwendige Aktionen ausgelöst werden, die für die Erhebung notwendig sind

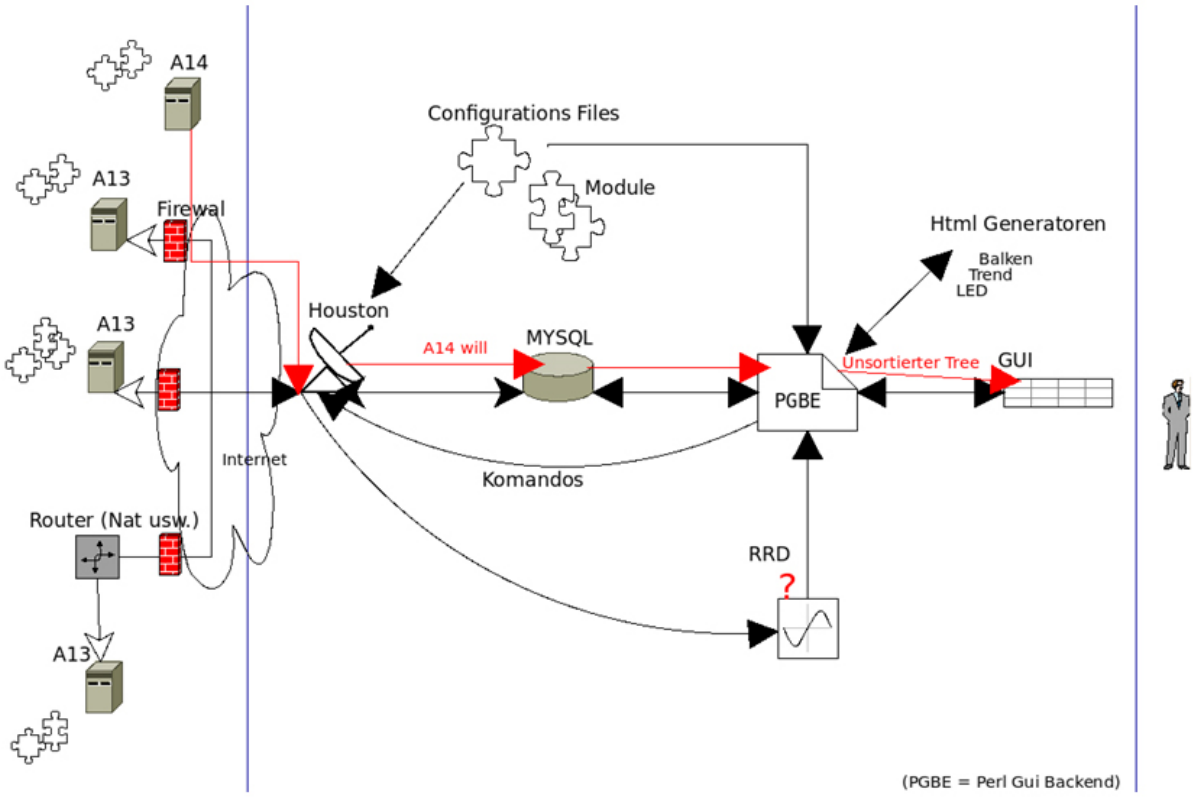

Abb. 17: Erste Kontaktaufnahme von einem Agenten zum Spotlight-Server

Für die Ablage dieser Dateien im Dateisystem werden definierte Pfade verwendet. Der Agent überwacht kontinuierlich diese Verzeichnisse auf Änderungen. Wird eine Datei modifiziert, wird sie durch das Laufzeit-Framework vom Agenten neu eingelesen. Der Agent ruft das „Executable“ auf, damit die Kollektor-spezifischen Daten des Remote-Servers erhoben werden können. Auf der Standardausgabe „stdout“ werden die Daten

an den Agenten übertragen. Durch die Spezifikationen aus der Konfigurationsdatei werden die gesammelten Daten eindeutig beschrieben und für die weitere Verarbeitung nutzbar gemacht.

Über Konfigurationsoption des Remote-Servers wird der Agent aktiviert. Dabei ist konfigurierbar, welche Informationen an Spotlight weitergegeben und welche Befehle von Spotlight ausgelöst werden dürfen. Nimmt ein Agent zum ersten Mal die Verbindung zu Spotlight auf (Abb. 17), wird er in der Datenbank als unbekannter Server abgelegt. Der Administrator hat nun die Möglichkeit ihn in die bestehende Struktur einzugliedern. Der Agent stellt in regelmäßigen Intervallen (5 min) die Verbindungen erneut her, um auf die Instruktionen von Spotlight zu antworten. Spotlight selbst kann keine Verbindungen initiieren. [2]

3 Probleme bei der Umsetzung

Grundsätzlich war die Einarbeitung in die diversen Virtualisierungslösungen und die unterschiedlichen Ansätze relativ aufwändig für alle Projektteilnehmer. Zudem traten ganz neue Cloud-Lösungen auf den Plan, wie z.B. OpenStack, die vor dem Projektstart noch nicht in ausreichender Form existierten und in die Planung und Konzeption integriert werden mussten. Zudem hatten die Partner eine sehr unterschiedliche Sichtweise auf die Projektziele und deren Umsetzungsmöglichkeiten, die entsprechend gebündelt werden mussten. Daher entstanden am Ende unterschiedliche Lösungen, die sich gegenseitig ergänzten oder auch autonom voneinander betrieben werden können (z.B. das IO-Toolset oder die verschiedenen Topologie-Editoren).

An dieser Stelle wird nun kurz auf die Problemstellungen der einzelnen Bereiche eingegangen, die sich während des Projektes ergeben haben.

3.1 Virtualisierungslösungen

Diverse Virtualisierungslösungen und -verfahren (u.a. Xen, KVM, Hyper-V) wurden am Anfang des Projektes untersucht und einer Bewertung unterzogen. Am Ende wurde KVM ausgesucht, welches eine freie Virtualisierungssoftware ist, die unter der GPL-Lizenz veröffentlicht wird. Seit der Linux Version 2.6.20 ist KVM fester Bestandteil des Mainstream Linux-Kernels und ist inzwischen in den meisten Distributionen als die Standard-Virtualisierungslösung integriert. Die Firmen AMD, IBM, Intel, Suse und RedHat arbeiten bei der Weiterentwicklung der Software zusammen.

KVM besteht aus einem Kernel-Modul und einem modifizierten „Qemu“ im User-Space. Das modifizierte „Qemu“ emuliert die Hardware der virtuellen Maschinen. Die Virtualisierung der CPU übernimmt das KVM-Modul. Das Kernel-Modul arbeitet als Hypervisor für die virtuellen Maschinen. Dieser Hypervisor nutzt die Virtualisierungserweiterungen der modernen CPUs von AMD und Intel. Die Software baute von Anfang auf das Verfahren der „hardwarebasierten Vollvirtualisierung“. Somit ist der Betrieb von unmodifizierten Gastsystemen direkt möglich. Es zeichnet sich durch eine herausragende Performance aus. Die Gastsysteme laufen mit annähernd nativer Geschwindigkeit.

Im Gegensatz zu anderen Virtualisierungslösungen fungiert KVM allein als Hypervisor für die Virtualisierung. Dieser ist vollständig in den Linux-Kernel integriert. Somit übernimmt der Kernel selbst die Ressourcenverwaltung und Steuerung. Der Hypervisor an sich ist somit sehr schlank und profitiert direkt von allen Weiterentwicklungen am Linux-Kernel. KVM bietet Unterstützung für paravirtualisierte Gerätetreiber für Netzwerk- und Blockdevices. Hierdurch können die performancekritischen I/O-Zugriffe weiter optimiert werden.

Durch die Nutzung einer freien Virtualisierungslösung gab es hier keinerlei Lizenzprobleme zu berücksichtigen. Zudem setzte auch die Collax-Plattform auf KVM, weshalb hierdurch eine gute Kompatibilität sichergestellt werden konnte. Allerdings

nahm die Analyse und Evaluierung des passenden Virtualisierungssystems einige Zeit in Anspruch. Zusätzliche Performance- und Hochverfügbarkeitstests verzögerten das Projekt etwas in seiner Anfangsphase. Zudem mussten die VMs von KVM auch noch verwaltet werden können, weshalb weitere Lösungen wie libvirt, libguest, OpenStack, VDE und OpenVSwitch untersucht werden mussten. Hinzu kam die Untersuchung der Schnittstellenformate. Dies war zum einen relativ aufwändig und zog zum anderen weitere Planungsarbeiten nach sich, da alle diese „Inseln“ sinnvoll miteinander verknüpft werden mussten. Erschwerend kam hinzu, dass die Projektpartner unterschiedliche Strategien anfangs verfolgten, die es ebenfalls galt in ein Framework zu bündeln.

Als Simulationsumgebung wurde sich auf OpenStack in der Version Essex geeinigt. OpenStack ist eine Cloud-Management-Software für Infrastrukture as a Service (IaaS) Clouds. Diese Software stellt quasi die Open-Source-Standard-Software für Cloud-Management dar. Das Verwenden einer Cloud-Software bietet den Vorteil der hohen Skalierbarkeit dieser Lösung, damit es auch möglich ist größere Infrastrukturen simulieren zu können. Für die bessere Verwaltung der Netzwerke der simulierten Umgebung wurde das OpenStack Plug-In Quantum in Verbindung mit OpenVSwitch verwendet. Mit diesem Plug-In ist es möglich auch komplexerer Netzwerke virtualisiert abzubilden. Zudem wurde libvirt als einheitliche API genutzt, da dies ebenfalls ein Bestandteil von OpenStack ist.

Allerdings machte der Entwicklungsstatus von OpenStack einige Probleme, da diese Software nicht stabil lief und auch Routing-Probleme zur Folge hatte. So durfte man beispielsweise die OpenStack-Umgebung bei lauffähiger Konfiguration nicht neu starten, da dann die Umgebung nicht mehr fehlerfrei lief. Zudem überschrieb OpenStack die vorgenommenen iptables-Firewall-Regeln einer virtuellen Maschine einfach mit seiner eigenen Konfiguration, was in einer normalen Cloud-Umgebung durchaus ein gewünschtes Verhalten darstellt, da so sichergestellt ist, dass jede VM erreichbar ist. Im VISA-Szenario sollten jedoch Netztopologie nachgebildet werden, was dort einige Änderungen an den iptables-Firewall-Regeln und am Routing zur Folge hatte. An dieser Stelle wäre es schön gewesen, wenn solch eine Funktion in OpenStack vorhanden gewesen wäre. Dies konnte allerdings nicht durch das VISA-Projekt erbracht werden. Hier war man daher immer auf die nächste neue Version angewiesen.

3.2 Virtual Security Appliances (VSA)

Es wurden fünf verschiedene VSAs von drei Forschungspartnern entwickelt. Dabei konnte die DECOIT GmbH auf vorherige Arbeiten aus zwei anderen Forschungsprojekten (VOGUE[22] und ESUKOM[23]) zurückgreifen, um auf deren Basis die Virtual Security Appliances VSA-SRA und VSA-MAC zu entwickeln. Anhand der Komponentenbeschreibung kann man dabei bereits sehen, dass es sich hierbei nicht um eine triviale IT-Infrastruktur handelt. Im Zuge der Entwicklungsarbeiten wurden beide

[22] http://www.vogue-project.de
[23] http://www.esukom.de

VSAs zusammengeführt, so dass sie modular genutzt werden können. Dabei kam es zu leichten Interoperabilitätsproblemen zwischen den verschiedenen Komponenten, die aber innerhalb des Projektes ausgeräumt werden konnten. Hauptproblem war bei der Integration der Status von OpenStack, der häufig auf dem Entwicklungsserver für Überraschungen bzw. Fehlfunktionen sorgte. Das Hauptärgernis war dabei das Überschreiben der Firewall-Konfiguration der virtuellen iptables-Komponente. Letztendlich konnten aber alle Probleme gelöst werden.

Eine komplett automatisierte Verteilung der VSAs ist bei VSA-SRA und VSA-MAC bis zuletzt nicht ermöglicht worden. Zwar kann eine bestimmte Basisfunktionalität ausgerollt werden, aber der Administrator muss trotzdem für die Feinkonfiguration wissen, wie er die einzelnen Module handzuhaben hat. Beispielsweise sollte das Wissen, wie man einen RADIUS-Server verwaltet vorhanden sein, da die Profile sich nicht automatisch einbetten lassen. Trotzdem wird dem Administrator durch die automatische Verteilungsmöglichkeit eine Möglichkeit an die Hand gegeben, wie er relativ einfach die enthaltenen Komponenten ausrollen kann.

Die VSA-AAA der FH Dortmund wurde eigenständig entwickelt und bietet eine gewisse Überscheidung zu der VSA-SRA der DECOIT GmbH an. Während die VSA-AAA sich aber mehr auf die allgemeine Remote-Anbindung bezieht, steht bei der VSA-SRA mehr die sichere Anbindung von Android-Geräten im Vordergrund. Bei der Entwicklung der VSA-AAA wurde zudem ein eigener Topologie-Editor entwickelt, der ein recht automatisches Roll-Out unterstützte. Allerdings ließ sich dies nicht mehr auf die eigentliche VISA-Plattform (den V-TE) integrieren, weshalb die VSA-AAA eine autonome Komponente darstellte. Dies wäre aber durchaus möglich gewesen, wenn man die entsprechenden Anpassungen vorgenommen hätte.

Die VSA-UTM von der Collax GmbH ist für die Collax-Plattform entwickelt worden und beinhaltet diverse Sicherheitskomponenten. Eine Einbettung in das VISA-Framework wurde getestet und ermöglichte die Interoperabilität mit den anderen Komponenten. Allerdings erfolgt auch hier die Verwaltung der VMs autonom durch einen eigenen Topologie-Editor. Eine Anpassung der VSA-UTM auf die VISA-Plattform war aber möglich.

3.3 Topologie-Editoren

Aufgrund der unterschiedlichen Ausrichtungen der Partner kamen drei verschiedene Topologie-Editoren zur Entwicklung. Dies war zum einen notwendig, weil die Collax-Plattform ihre eigene Verwaltbarkeit benötigte, erhöhte aber zum anderen den Aufwand des Projektes. Zudem lassen sich die drei Editoren nicht gemeinsam verwenden, sondern nur in ihren spezifischen Umgebungen. So kommt der VirWi-Editor nur im LISA-Testbed der FH Dortmund zum Einsatz, während Spotlight von der Collax GmbH nur mit der Collax-Familie zurechtkommt. Daher besitzen beide Topologie-Editoren auch ein eingeschränktes Tätigkeitsfeld. Das heißt, sie stehen in dem geschaffenen Simulationskreislauf von VISA im Grunde genommen nicht zur Verfügung.

Der V-TE der DECOIT GmbH wurde hingegen in enger Zusammenarbeit mit Fraunhofer SIT so konstruiert, dass er entsprechend anpassbar ist und den gesamten Simulationskreislauf (Erstellung einer virtuellen Netzumgebung, Simulation und Verbreitung einer virtuellen Netzumgebung, Live-Schaltung einer virtuellen Netzumgebung) in VISA unterstützt. Er stellt somit die Schnittstelle zwischen dem IO-Toolset zur Erhebung von Netzumgebungen sowie dem Test und Ausrollen der virtuellen Netzumgebungen in OpenStack dar.

Alle drei Topologie-Editoren verfolgten zusammenfassend die folgenden gemeinsamen Ziele, die unterschiedlich umgesetzt wurden:

a. Zentrales Management für VMs im Topologie-Kontext

b. Schaffung von Simulationsumgebungen mit teilautomatisierten Tests

Eine Anforderung des VISA-Projektes war beim VirWi-Topologie-Editor der FH Dortmund technisch nicht lösbar, was zum Planungszeitpunkt nicht bekannt war. So lassen sich Kabelverbindungen im laufenden Betrieb nicht beliebig ändern, was an der fehlenden Unterstützung durch die Virtualisierungssoftware lag. Das native Monitorprotokoll ließ sich aber in wenigen Schritten so erweitern, dass neue Funktionen in die GUI eingebettet werden konnten. Des Weiteren sind erweiterte Online-Status-Abfragen, z.B. Prozessor- und RAM-Verbrauch oder Mehrsprachigkeit, technisch problemlos umsetzbar und in Planung. Die technischen Voraussetzungen für deren Umsetzung sind auf jeden Fall vorhanden. Eine Unterstützung von OpenVSwitch nebst VDE wäre ebenfalls wünschenswert und konnte innerhalb des Projektes nicht mehr umgesetzt werden.

Der V-TE der DECOIT GmbH unterstützt die größten Funktionsmerkmale und verknüpft die unterschiedlichen „Inseln“ zu einer Gesamtplattform. Allerdings ist er auf das IO-Toolset angewiesen, um eine virtuelle Netzinfrastruktur erheben und wieder ausrollen zu können. Daher wäre eine direkte TE-Anbindung an OpenStack wünschenswert, konnte im VISA-Projekt aber aus Zeit- und Ressourcengründen nicht mehr umgesetzt werden. Auch die direkte TE-Aufnahme von IT-Infrastrukturen wäre ein wichtiges Feature, um nicht weiter von dem IO-Toolset von Fraunhofer SIT abhängig zu sein. Zudem wäre eine erweiterte Fehlererkennung für falsch konfigurierte Netze interessant, um die Fehlerhäufigkeit so gering wie möglich zu halten.

Der Topologie Editor (TE) Spotlight von der Collax GmbH ist im Gegensatz zu den beiden anderen TEs eher ein Verwaltungswerkzeug virtueller Maschinen, als ein Topologie-Manager. Netztopologien werden daher von ihm nicht direkt unterstützt. Er ist aber ein wichtiges Verwaltungs- und Monitoring-Werkzeug in der Collax-Umgebung, kann daher aber nicht in fremder Umgebung eingesetzt werden und bringt zudem nicht die notwendigen Leistungsmerkmale mit, um auch Netzwerkumgebungen managen oder definieren bzw. ausrollen zu können. Hier könnten zukünftig die Ansätze der Topologie-Editoren V-TE und VirWi mit in die Entwicklung einfließen, was aber innerhalb des Projektes nicht mehr geleistet werden konnte.

Die in VISA entwickelten Topologie-Editoren verfolgten unterschiedliche Ansätze, um

eine virtuelle Umgebung erstellen, verwalten und konfigurieren zu können, wie auch die Tab. 1 verdeutlicht. Hier werden noch einmal die verschiedenen Funktionalitäten zusammengefasst und gegenübergestellt. An der Übersicht wird klar, dass bereits viele Funktionalitäten umgesetzt wurden, aber trotzdem noch offene Punkte bestehen bleiben. Diese könnten in einem weiteren Forschungsprojekt zukünftig angegangen werden.

Eigenschaften	Virtual Wizard (VirWi)	VISA Topologie Editor (VTE)	Collax Spotlight
Erheben eines bestehenden Netzes	Nein	Bedingt	Nein
Erstellen von Netztopologien	Ja	Ja	Nein
Erstellen von VMs	Ja	Bedingt	Ja
Erstellen aktiver Komponenten	Bedingt	Bedingt	Nein
Erstellen von Kabelverbindungen	Ja	Ja	Nein
Speicherung von Netzwerkplänen	Ja	Ja	Nein
Verwalten virtueller Festplatten	Ja	Nein	Ja
Verwalten von Bridges	Ja	Ja	Nein
Verwalten von Netzwerken	Ja	Ja	Nein
Verwalten von VMs	Ja	Bedingt	Ja
Verschlüsselte VM-Verbindungen	Ja	Ja	Ja
Gruppierung von Komponenten	Nein	Ja	Nein
Tool-Analyse der Konfiguration	Nein	Ja	Nein
VLAN-Unterstützung	Ja	Ja	Ja
Fat-/Web-Client	Ja/Nein	Nein/Ja	Nein/Ja
Datenbank-Unterstützung	Ja	Ja	Ja
Compliance-Unterstützung	Ja	Ja	Ja
Lizenz	Open Source	GPLv3	Proprietär

Tab. 1: **Gegenüberstellung der drei Topologie-Editoren**

3.4 Automatisierte Erhebungsmethoden

Die semantische Aggregation von Rohdaten bedarf eines Systems, das Konfigurationen und Zustände von IT-Komponenten in einer Form vorhält (repräsentiert), die es ermöglicht herstellerübergreifende Bezüge herstellen zu können. Dabei ist es wesentlich, IT-Komponenten nicht alleinstehend zu modellieren, sondern stets als zusammenhängende Topologie mit zahlreichen untereinander bestehenden Relationen, Abhängigkeiten und Interaktionsmöglichkeiten. Das IO-Framework, basierend auf der Interconnected-asset Ontology (IO), erfüllt diese Eigenschaften und stellt ein formelles Konzeptmodell zur vollständigen Repräsentation komplexer Netze in hohem Detailgrad zur Verfügung. Mit Hilfe der ontologischen Repräsentation können wesentliche Systemeigenschaften herstellerübergreifend dargestellt und zueinander in Bezug

gebracht werden.

Die Erhebung selbst kann vollautomatisch durchgeführt werden, da das IO-Framework bereits eine große Zahl an Erhebungsmethoden für Roh-Metadaten unterstützt (z.B. SNMP, WMI, SOAP, SSH oder IF-MAP). Für IT-Infrastrukturen mit starkem Legacy-Anteil oder proprietären Netzen müssen verteilte Standard-Kollektoren angepasst (und in Einzelfällen auch entwickelt) werden, die Metadaten mit Hilfe von Backhaul-Interfaces (BHI) aus Legacy-Infrastrukturen extrahieren können. Die mit Hilfe des IO-Frameworks erstellten formalen Repräsentationen (IT-Struktur Snapshots) bilden die Grundlage für weitere nachgelagerte, automatische Prozesse, z.B. die Zielobjekt Gruppierung.

Allerdings ist die vollautomatisierte Ergebung in der Praxis auf Grenzen gestoßen, da z.B. aktuell hauptsächlich Cisco-Komponenten unterstützt werden und andere Hersteller zwar angepasst werden könnten, aber aktuell nicht genutzt werden können. Zudem beinhaltet eine automatisierte Erfassung eine hohe Datenlast, die kurzzeitig zu hohen Leistungsengpässen führen kann, weshalb diese nicht während des Betriebs ausführbar ist. Hinzu kommt, dass der Status des Interconnected-asset Ontology (IO) Toolsets eher wissenschaftlichen Aspekten genügt, als praktischen Einsätzen in Kundenumgebungen. Daher ist noch ein gewisser Weg von einem Prototypen bis hin zu einem wirkungsvollen Standardwerkzeug zurückzulegen. Es wird daher dieser Ansatz auch in anderen Forschungsprojekten weiter untersucht und im Hinblick auf unterschiedliche Aspekte weiter entwickelt.

3.5 OMF/OML

Das Testen einer IT-Umgebung auf Schwachstellen erfolgt meist manuell durch das Nutzen entsprechender Tools. Das OMF/OML-Framework bietet die Möglichkeit diese Tools zu automatisieren und in einem Experiment zu verwenden. Weiterhin ist es möglich die so gesammelten Daten aufzubereiten und darzustellen. Um ein Experiment durchzuführen führt jeder teilnehmende Knoten (Rechner/VM) den OMF Resource Controller (RC) aus. Dieser Dämon meldet sich an einem XMPP-Server an und wartet auf Befehle. Der Experiment Controller (EC) ist die Software, die der Benutzer ausführt, um das Experiment zu steuern. Dazu liest dieser ein vom Benutzer geschriebenes Skript ein, welches in der OEDL-Sprache verfasst ist (OMF Experiment Description Language). Es handelt sich hier um ein Ruby-Skript mit zusätzlichen, OMF-spezifischen Kommandos. Während des Experimentes sendet der EC die Befehle an verschiedene PubSub Gruppen auf dem XMPP Server, die wiederum von den RCs abonniert werden. Die RCs senden ggf. Antworten, auf die der EC dann reagieren kann. Am Ende eines Experimentes speichert OMF die Ergebnisse in eine SQLite-Datenbank.

Die Entwicklung von OMF und OML wird ständig durch die NICTA weiter vorangetrieben. Allerdings gab es während des VISA-Projektes noch keine direkte Anbindung an OpenStack. Dies wurde im Laufe des Projektes aber durch die Forschungsaktivitäten der NICTA allerdings geändert und neu implementiert. Durch die Integration diverser Netzwerkanalysetools wurde seitens der DECOIT GmbH auch die Funktionalität von OMF/OML gesteigert. Dafür wurde jeweils für ping, Nmap und T-50 ein Wrapper

entwickelt, indem auch das OML-Reporting integriert wurde. Die Analysetools collectd und netem wurden allerdings nicht in das OMF-Framework integriert.

Das OMF-Framework bot die optimale Möglichkeit, die für VISA benötigten Analysetools zu steuern. Vor dem Ausführen von Experimenten mit OMF, mussten aber die dafür benötigten Ressourcen bereitstehen und die vorgesehenen Applikationen in das Framework integriert werden. Zu den Ressourcen gehörten ein Experiment Controller (EC) sowie mindestens ein Knoten auf dem ein Ressource Controller (RC) und die Analysetools installiert werden. In VISA wurde der „Aggregate Manager“ nicht zum Einrichten der Nodes benötigt, da die Analysetools im Vorfeld installiert werden sollten. Für jede Applikation wurde daher für die Integration in das OMF-Framework, eine eigene „Application Definition“ sowie eine „Experiment Description“ entwickelt. Zusätzlich musste das OML-Reporting unterstützt werden, um die Messergebnisse zum „Aggregate Manager“ schicken zu können. Auch dies wurde gelöst und umgesetzt. Daher kann das OML-Framework auch für zukünftige Testplattformen gut verwendet werden.

Letztendlich konnte die Umsetzung des VISA-Vorhabens mit OMF/OML erfolgreich durchgeführt werden. Parallel zu der stabilen Version 5.4 von OMF wurde an der Version 6 gearbeitet, welche ein komplettes Re-Write ist. OMF6 hat ist leichter zu installieren, da es als Ruby Gem ausgeliefert wird. Weiterhin ist es dank Ruby1.9/2.0-Unterstützung deutlich schneller als die Vorgängerversion. Außerdem ist es modularer aufgebaut, so dass z.B. verschiedene Kommunikations-Plug-Ins verwendet werden können (XMPP, AMQP etc.). OMF6 verwendet das FRCP-Protokoll, welches leicht von Dritten implementiert werden kann, so dass OMF6 mit vielen verschiedenen Softwarekomponenten sprechen kann. Gerne hätte die NICTA OMF6 in VISA eingesetzt, aber es befand sich zur Projektlaufzeit in einem Entwicklungsstadium, so dass es nicht für den Produktionsbetrieb verwendet werden konnte.

3.6 IT-Compliance

Die IT-Compliance beschreibt in der Unternehmensführung die Einhaltung der gesetzlichen, unternehmensinternen und vertraglichen Regelungen im Bereich der IT-Landschaft. Die IT-Compliance ist im Zusammenhang mit der IT-Governance (Führung, Organisationsstrukturen und Prozessen) zu sehen, die das Thema um die Bereiche Controlling, Geschäftsprozesse und Management erweitert. IT-Compliance als Teilbereich fokussiert diejenigen Aspekte von Compliance-Anforderungen, welche die IT-Systeme eines Unternehmens betreffen. Zu den Compliance-Anforderungen in der IT gehören hauptsächlich Informationssicherheit, Verfügbarkeit, Datenaufbewahrung und Datenschutz. Unternehmen unterliegen zahlreichen rechtlichen Verpflichtungen, deren Nichteinhaltung zu hohen Geldstrafen und Haftungsverpflichtungen führen kann. EU-Richtlinien, internationale Konventionen, unternehmensinterne Konventionen und Handelsbräuche fügen weitere Regeln hinzu.

Die Ermittlung der IT-Compliance ist für IT-Infrastrukturen von großer Bedeutung, da sich aus dem BSI-Grundschutz oder den ISO-Standards 27001 bzw. 27002 konkrete Maßnahmen ableiten, um die IT-Sicherheit zu erhöhen bzw. auf einem hohen Niveau halten zu können. Durch die Analyse und die Abschätzung des Umsetzungsgrades der

Anforderungen, wurde im VISA-Projekt gezeigt, wie gut die jeweiligen VSA-Dienste den Sicherheitsstandard BSI-Grundschutz und die ISO 2700x (genauer: Maßnahmen nach ISO 27002) erfüllen können. Bei den durchgeführten Untersuchungen wurden allerdings nur die Maßnahmen berücksichtigt, die entweder komplett, zum Teil oder auch nur teilweise erfüllt wurden. Alle anderen wurden nicht berücksichtigt. Dies bedeutet, dass der Einsatz einer VSA bei der Umsetzung dieser Maßnahmen nicht unterstützen kann.

Erfolgt eine darüber hinausgehende Untersuchung und Analyse der Einbindung der VSAs in einer konkreten Umgebung bzw. einem detaillierteren Einsatzszenario, so lassen sich auch weiterführende Aussagen über den Erfüllungsgrad der Sicherheitsstandards BSI Grundschutz und ISO 2700x treffen, bzw. es lassen sich konkrete Maßnahmen ableiten, die geeignet sind, um die IT-Sicherheit weiter zu erhöhen (Optimierung der durch die VSA umgesetzten Maßnahmen) bzw. dauerhaft auf einem hohen Niveau zu halten (Maßnahmen, die nicht oder nur teilweise durch das alleinige Deployment der VSA erfüllt werden können). Bezogen auf die VSA kann man die Dienste der jeweiligen VSA daraufhin so konfigurieren, umgestalten oder ergänzen, so dass sie für das Einsatzszenario die mindestens die Minimalanforderungen der Standards erfüllen. Sie können damit für ein Unternehmen einen wichtigen Baustein darstellen, um ein angemessenes Maß an Compliance-Erfüllung innerhalb der IT zu erreichen. Es ist allerdings immer noch ein hohes Maß an manuellem Aufwand notwendig, um alle Compliance-Anforderungen der vorhandenen Sicherheitsstandards abdecken zu können.

Die IT-Compliance automatisiert zu dokumentieren und Audit-gerecht bereitzustellen wurde daher im VISA-Projekt nur ansatzweise umgesetzt. Dies war aus Zeit- und Ressourcengründen nicht mehr umsetzbar und benötigt weitere Forschungen, da aktuell ein hoher manueller Aufwand notwendig ist, der auch Expertenwissen voraussetzt. Zudem sind die BSI- und ISO-Standards recht unübersichtlich und komplex, was auch die Auswertungen innerhalb des VISA-Projektes belegen. Daher war es zwar möglich einige Basisanforderungen an eine Compliance in den VSAs zu hinterlegen, aber leider nicht alle Anforderungen komplett zu erfüllen. Auch konnten leider die Ergebnisse nicht mehr in den Entwicklungsprozess einfließen, da die Analyse erst am Ende des Projektes erfolgen konnte.

3.7 Fazit

Insgesamt verzögerte sich das Projekt hauptsächlich durch die schier überwältigende Auswahl an Virtualisierungslösungen, die allesamt getestet, bewertet und eingesetzt wurden. Zudem war man Abhängig von Drittherstellern wie OpenStack, die durch noch vorhandene Bugs die Implementierungen und Fehlersuche erschwerten. Auch die Erarbeitung einer einheitlichen Strategie zwischen den unterschiedlichen Projektpartnern benötigte einige Abstimmungszeit. Die Probleme wurden aber innerhalb des Projektes zufriedenstellend gelöst und mündeten teilweise in unterschiedliche Lösungsansätze (verschiedene Topologie-Editoren und Einwahl-VSAs). Offene Punkte könnten in weiteren Forschungsprojekten zukünftig angegangen werden.

4 Demonstrator

Der Demonstrator beinhaltet den VISA Topologie Editor (V-TE) der DECOIT GmbH und das IO-Toolset von Fraunhofer SIT als VirtualBox VM (Virtual Machine). Eine OpenStack-Umgebung ist aufgrund der benötigten Ressourcen nicht enthalten und müsste extern eingebunden werden. Für die Installation einer OpenStack-Umgebung mit OpenVSwitch-Integration wird an dieser Stelle daher auf die Internetseite[24] des OpenStack-Projektes verwiesen. Zur Verfügung gestellt wird eine VM in einem ZIP-Archiv, die ohne OpenStack lauffähig ist.

Daneben wurden zu jedem Topologie Editor (TE) ein Video erstellt, das die Funktionen der Editoren verdeutlicht. Diese Videos sind auf der Projektwebseite[25] verfügbar. Weiterhin sind die Topologie-Editoren VirtualWiziard (VirWi) von der FH Dortmund und Spotlight von der Collax GmbH ebenfalls auf der Projektwebseite im Download-Bereich separat abrufbar.

Der Demonstrator zeigt den entwickelten Simulationskreislauf des VISA-Projektes auf. Dieser enthält das Erheben einer Topologie (automatisierte Ist-Aufnahme einer IT-Infrastruktur), das Replizieren dieser Topologie (z.B. in eine OpenStack-Umgebung), Änderung an dieser und Test der geänderten Umgebung:

a. Im Ersten Schritt steht herbei das Erheben einer Topologie mit dem IO-Toolset an. Da die Erhebung sehr lange dauern kann (je nach Größe des Netzes), sind im Demonstrator einige Topologien bereits vorhanden. Des Weiteren können auch neue Topologien erstellt werden. Danach würde das Replizieren der erhobenen Topologie erfolgen. Hierbei müsste von allen Komponenten im Netz ein Image erzeugt, danach auf eine OpenStack-Umgebung eingespielt und die Netzkonfiguration angewandt werden. Dabei sind die Komponenten dann als virtuelle Maschine (VM) verfügbar. Da auch dieser Prozesse sehr zeitaufwändig ist, wird dieser Schritt übersprungen und simuliert.

b. Der zweite Schritt beinhaltete das Ändern der Topologie mit Hilfe des V-TE. Hierbei können neue VMs hinzugefügt und Änderungen an der Netzeinstellung vorgenommen werden, wie z.B. das „Verschieben“ einer VM in ein anderes Netz.

c. Der dritte Schritt beinhaltet das Testen der geänderten Konfiguration. Hierbei wird das OMF/OML-Framework genutzt, welches über den V-TE ansprechbar ist. Dafür wird eine OpenStack-Umgebung vorausgesetzt, da diese Tests grundsätzlich Live-Daten liefern und nicht simuliert werden können.

Der Demonstrator kann über das Kontaktformular auf der Projektwebseite angefragt werden. Dies würde aus speichertechnischen Gründen so realisiert. Es ist aber auch ein

[24] http://www.openstack.org
[25] http://www.visa-project.de

Live-Zugang zum V-TE in einer OpenStack-Umgebung des australischen Partners NICTA über folgende URL-Adresse möglich: http://221.199.209.248

4.1 Inbetriebnahme des Demonstrators

Im Folgenden wird kurz darauf eingegangen wie der Demonstrator gehandhabt werden kann.

4.1.1 Importieren der virtuellen Maschine

Nach dem Entpacken des Archives liegt der Demonstrator als OVA-Datei vor. Diese kann in VirtualBox folgendermaßen importiert werden:

1) Menü „Datei“ → „Appliance importieren“ auswählen

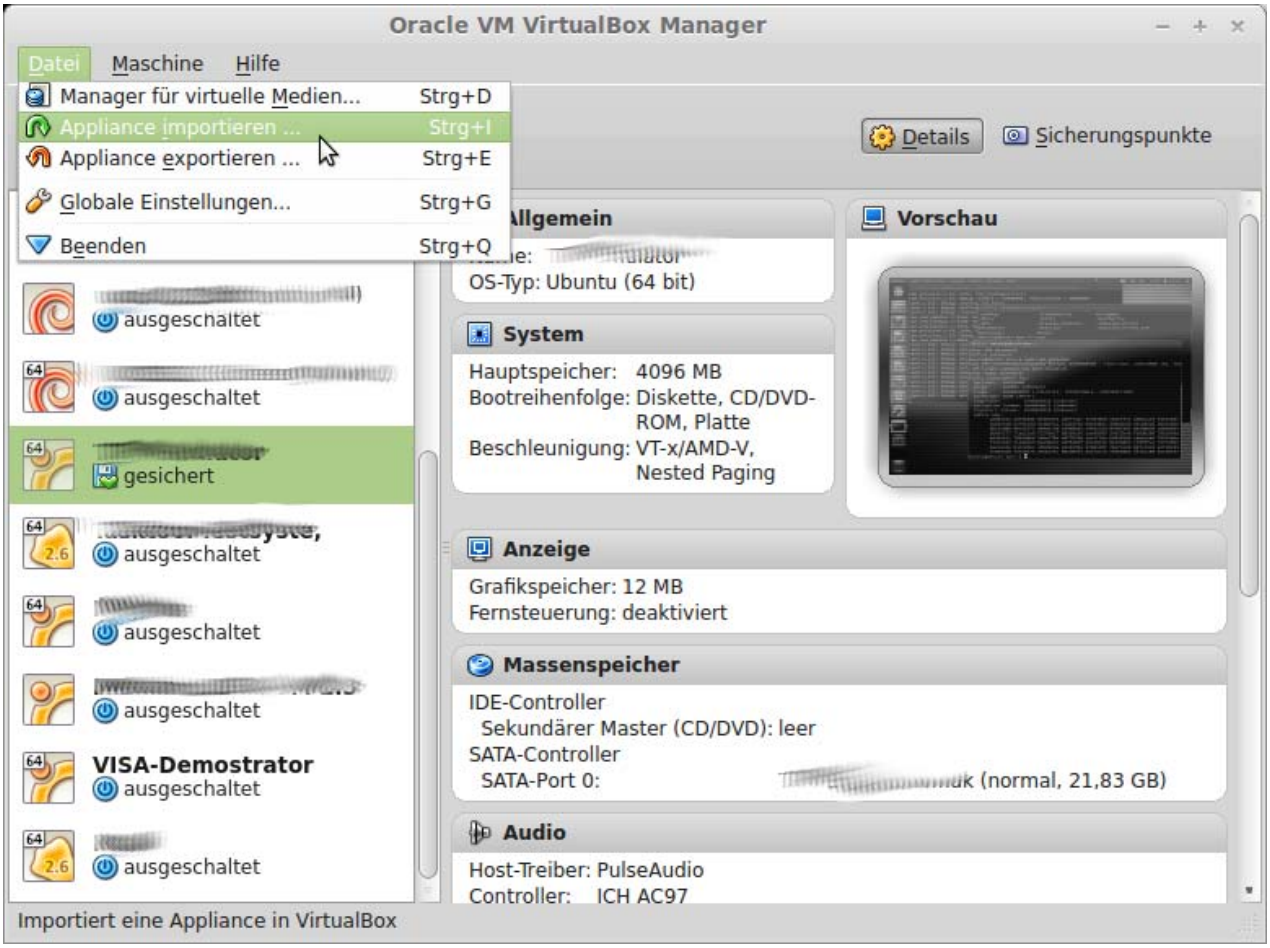

Abb. 18: Import der virtuellen Maschine

2) Im erscheinenden Fenster die OVA-Datei auswählen und danach auf „Weiter“ gehen

3) Im nächsten Fenster werden die Einstellungen der VM angezeigt und können ggf.

geändert werden. Mit einem Klick auf „Importieren“ wird die VM importiert.

Abb. 19: Import-Einstellungen

Nun muss noch die VM gestartet warden, dazu den Pfeilbutton “Starten” drücken. Der Benutzername und das Passwort sind **visa/visa2013**.

4.1.2 Konfiguration des IO-Toolsets

Die VISA-Version des IO-Toolsets benötigt ein Ubuntu 12.04 oder höher. Eine typische (und empfohlene) Installationsvariante ist die Installation des IO-Toolsets in derselben VM, in der auch der V-TE installiert wurde. Im Home-Verzeichnis des Nutzers *visa* auf der Demonstrator-VM befindet sich die Datei *visaserver.conf.rb*, welche auch im herunterladbaren Installationsarchiv des IO-Toolsets zu finden ist. In dieser sind die enthaltenen Konfigurationseinträge ggf. anzupassen. Alle potentiell anzupassenden Konfigurationsparameter sind im Folgenden erläutert:

1.) Angabe der Zugangsdaten des SSH-Zugangs für die OpenStack-Instanz. Der entsprechende Nutzer (hier: *visa*) muss automatische sudo-Eskalationsrechte für alle ovs- und nova-manage Kommandos besitzen.

```
:"auth.demonstrator.username" => "visa",
:"auth.demonstrator.password" => "visa2013",
```

2.) Aktivierungsmöglichkeit des Paranoia-Modus des Demonstrators, der Veränderungen an der Demonstrator-Ontologie grundsätzlich verhindert. „Der Parameter false“ ist dabei die Voreinstellung, die den Paranoia-Modus deaktiviert und ggf. durch „true“ ersetzt werden muss.

```
:"visa.paranoia_mode" => false,
```

3.) Aktivierungsmöglichkeit eines Safe-Modus des Demonstrators, der nur die Funktionsaufrufe „Löschen“ und „Replizieren“ grundsätzlich verhindert (z.B. ist „Modifizieren“ dagegen erlaubt). Der Parameter „false“ ist dabei die Voreinstellung, die den Safe-Modus deaktiviert und ggf. durch „true“ ersetzt werden muss.

```
:"visa.safe_mode" => false,
```

4.) Aktivierungsmöglichkeit eines Demonstrator-Modus, der sehr zeitaufwendige Operationen, wie die Replikation oder die Erhebung, durch simulierte Interaktionen mit dem Topologie-Editor ersetzt. Dies ermöglicht es die Kernfunktionen von dem Topologie-Editor und IO-Toolset zu demonstrieren, ohne die teilweise immensen Wartezeiten, die durch solche Operationen verursacht werden. Der Parameter „true“ ist dabei die Voreinstellung, die den Demonstrator-Mode aktiviert und ggf. durch „false“ ersetzt werden muss.

```
:"visa.demonstrator_mode" => true,
```

5.) Angabe des OpenStack-Servers, der im Zuge der Demonstration als repräsentative Quellumgebung repliziert werden soll.

```
:"visa.source.server" => "aaa.bbb.ccc.ddd",
```

6.) Angabe aller relevanten OpenStack-Authentifizierungsinformationen (siehe OpenStack-Konfigurationskurzanleitung) für die Kommandoausführung in der Quellumgebung.

```
:"visa.source.system_user" => "visa",
:"visa.source.system_password" => "visa2013",
:"visa.source.os_tenant" => "admin",
:"visa.source.os_user" => "admin",
:"visa.source.os_password" => "hastexo",
:"visa.source.os_url" => "http://localhost:5000/v2.0/",
:"visa.source.os_service_endpoint" => "http://localhost:35357/v2.0",
:"visa.source.os_service_token" => "hastexo",
```

7.) Angabe der IP-Adresse unter der die VM, auf der das IO-Framework betrieben wird,

aus der Quellumgebung heraus erreicht werden kann.

```
:"visa.source.reverse_connection_host" => "10.10.253.34",
:"visa.source.reverse_connection_user" => "visa",
```

8.) Angabe der IP-Adresse des OpenStack-Systems, in dem die Quellumgebung repliziert werden soll.

```
:"visa.destination.server" => "aaa.bbb.ccc.ddd",
```

9.) Angabe aller relevanten OpenStack-Authentifizierungsinformationen (siehe OpenStack-Konfigurationskurzanleitung) für die Kommandoausführung in der Replikationsumgebung.

```
:"visa.destination.system_user" => "visa",
:"visa.destination.system_password" => "visa2013",
:"visa.destination.os_tenant" => "admin",
:"visa.destination.os_user" => "admin",
:"visa.destination.os_password" => "hastexo",
:"visa.destination.os_url" => "http://localhost:5000/v2.0/",
:"visa.destination.os_service_endpoint" => "http://localhost:35357/v2.0",
:"visa.destination.os_service_token" => "hastexo",
```

4.2 Handhabung des Demonstrators

Jetzt steht die Basis des Demonstrators zur Verfügung und das Back-end des V-TE muss nur noch neu gestartet werden, wie u.a. im V-TE-Unterkapitel beschrieben wird.

4.2.1 Topologie-Editor V-TE

Das Front-end des Topologie-Editors wurde als Web-Oberfläche, größtenteils in JavaScript, entwickelt. Im oberen Bereich der Oberfläche Abb. 15), unterhalb der Kopfzeile, befindet sich die Options-Leiste. Sie erlaubt verschiedene Einstellungen und gibt Zugriff auf diverse Funktionen des Editors. So lässt sich hier die automatische Wegfindung für Kabel auf dem Editor-Raster ein- und ausschalten. Die zweite Option erlaubt das Einblenden der vollen Namen aller Komponenten auf dem Raster und die dritte das Einfärben der Kabel entsprechend des VLANs zu dem sie gehören. Diese beiden Optionen können zeitgleich aktiviert werden und blockieren jeweils die Drag&Drop-Funktionalität des Editors um Anzeigeprobleme zu verhindern. Diese

Beschränkung wird eventuell zu einem späteren Zeitpunkt aufgehoben.

Um den V-TE nutzen zu können muss vorher das Back-end gestartet werden. Dazu muss sich auf die VM eingeloggt, dies kann per SSH oder direkt auf die VM erfolgen. Danach muss lediglich der Befehl `visabackend` ausgeführt werden. Mit dem anhängen eines Und-Zeichens & wird das Back-end im Hintergrund gestartet und bleibt auch aktiv nachdem sich wieder aus der VM ausgeloggt wurde. Nun kann der V-TE wie unten beschrieben verwendet werden.

Der erste der drei Buttons öffnet einen Dialog für den Im- und Export von RDF/XML-Dateien (Abb. 20). Dort könnte eine RDF/XML-Datei vom lokalen Rechner auf den Server hochgeladen und direkt in das Back-end importiert werden. Dabei kann ausgewählt werden, ob die aktuelle Topologie dabei überschrieben werden soll oder ob Informationen der neuen Datei in das bestehende Modell integriert werden sollen. Im unteren Bereich des Dialogs kann die aktuelle Topologie unter dem angegebenen Dateinamen exportiert werden. Ein Klick auf den Button `Herunterladen` initiiert einen direkten Download der Datei. Dieser Button wird erst aktiv sobald die Topologie exportiert wurde. Der Statustext weist den Nutzer bei Bedarf darauf hin, dass die exportierte Datei nicht mehr der aktuellen Topologie entspricht und diese neu exportiert werden sollte.

Abb. 20: Dialog Im- und Export von RDF/XML Dateien

Der zweite Button öffnet einen Dialog, der das Zurücksetzen des Back-ends erlaubt (Abb. 21). Die Auswahl der oberen Option im Dialog sorgt dafür, dass sie Topologie vollständig geleert wird und der Editor in den Ausgangszustand zurück versetzt wird. Der untere Bereich kann, chronologisch sortiert von alt (oben) nach neu (unten), mehrere Zustände darstellen. Diese werden jeweils nach dem Import einer RDF/XML-Datei angelegt. Die Auswahl dieser Punkte setzt die Topologie auf diesen Zustand zurück. Dies beinhaltet jedoch nur die Daten aus den RDF/XML-Dateien. Alle manuellen Änderungen, egal zu welchem Zeitpunkt, gehen dabei verloren. Dieses Verhalten soll in der späteren Entwicklung noch geändert werden.

Abb. 21: Dialog für Zurücksetzen des Back-ends

Der dritte Button erlaubt das Herunterfahren des Back-ends. Dabei werden alle gespeicherten Informationen entfernt, hochgeladene Dateien gelöscht und das Programm beendet. Danach kann der Topologie-Editor nicht mehr verwendet werden. Das Starten des Back-ends über die Benutzeroberfläche ist nicht möglich. Somit ist diese Funktion nur während der aktiven Entwicklung sinnvoll. Mit dem Dropdown-Menü am Ende der Options-Leiste lässt sich die Sprache des Editors auswählen. Zur Auswahl stehen zur Zeit Deutsch und Englisch. Die Standardsprache wird aus der Spracheinstellung des Browsers abgeleitet. Stellt man die Sprache manuell ein wird diese in einem Cookie gespeichert und bei jedem Aufruf des Editors wiederhergestellt.

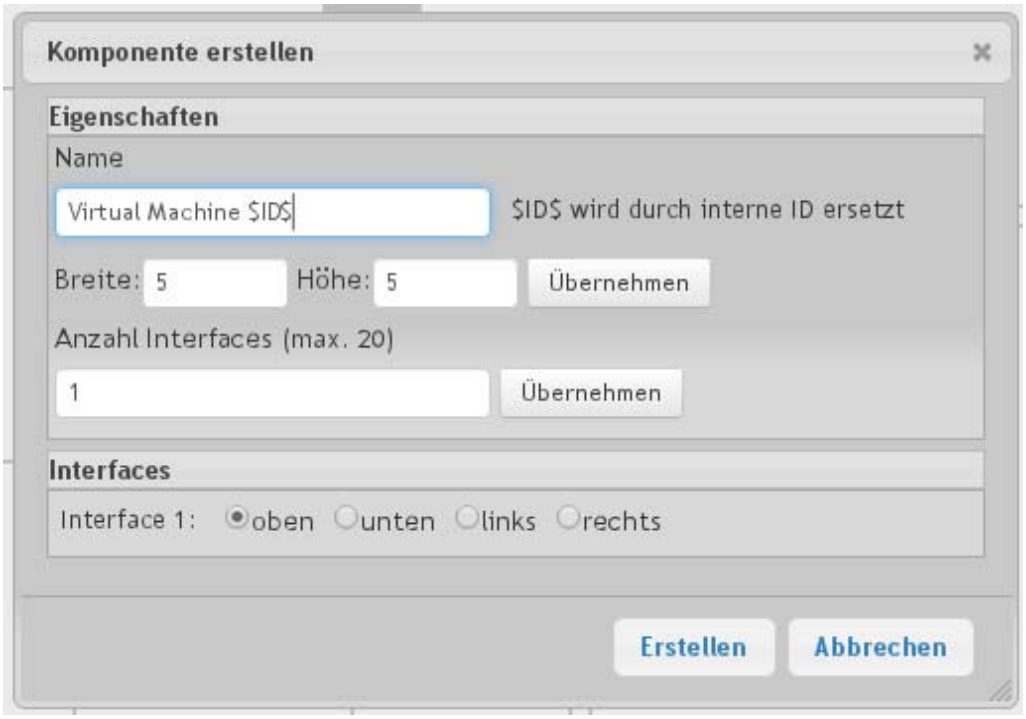

Abb. 22: Dialog für das Erstellen einer neuen Komponente

Unterhalb der Options-Leiste befindet sich eine Liste mit den Komponenten, die der Topologie hinzugefügt werden können. Zum gegenwärtigen Zeitpunkt sind dies virtuelle Maschinen und Switches. Zieht man eine solche Komponente per Drag&Drop auf einen freien Platz des Editor-Rasters, so öffnet sich ein Dialog (Abb. 22), in dem der Name, die Größe, die Anzahl der Interfaces und die Ausrichtung dieser festgelegt werden kann. Dabei wird automatisch darauf geachtet, dass beispielsweise nicht mehr Interfaces nach oben ausgerichtet werden können als die Komponente an Feldern breit ist. Mit einem Klick auf `Erstellen` wird im Back-end eine neue Komponente mit den eingestellten Parametern erzeugt und der Topologie sowie dem RDF-Modell hinzugefügt.

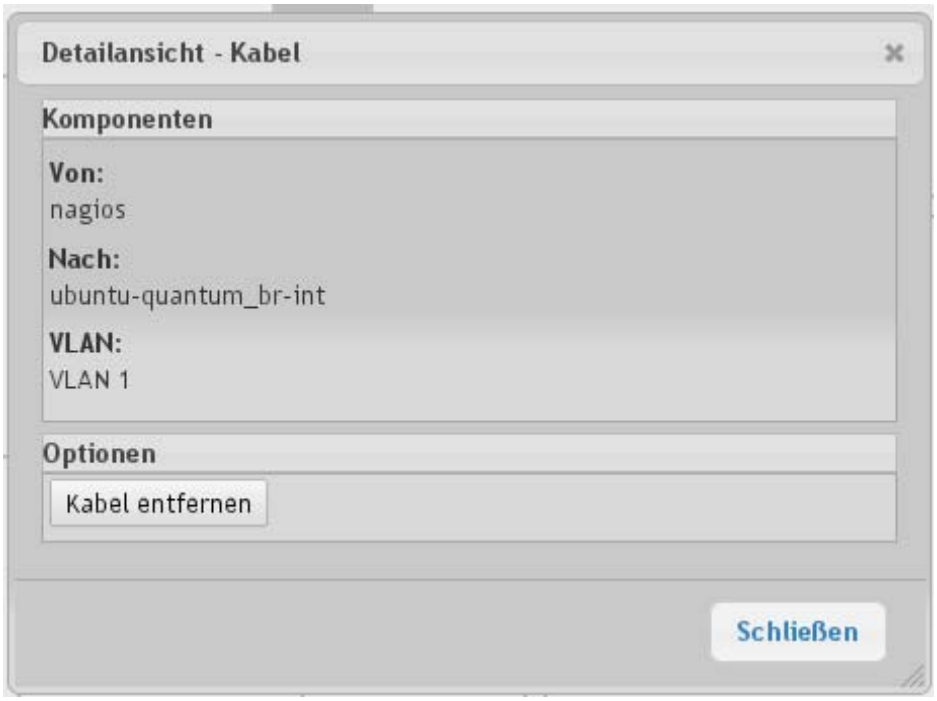

Abb. 23: Dialog für Detailinformationen zu einem Kabel

Der gesamte untere Bereich des Browsers wird vom Editor-Raster eingenommen. Auf diesem wird die Topologie dargestellt und bearbeitet. Klickt man ein Kabel mit der linken Maustaste an, wird ein Dialog geöffnet der Detailinformationen zu diesem Kabel zeigt (Abb. 23). Wird eine existierende Komponente auf einen der roten Bereiche links oder rechts gezogen wird sie aus der irreversiblen Topologie entfernt. Da diese Lösung jedoch Probleme bei großen Komponenten mit vielen Interfaces verursacht, soll das Löschen zu einem späteren Zeitpunkt über ein Kontextmenü ermöglicht werden.

Die Möglichkeit mehrere Komponenten zu einer Gruppe zusammenzufassen ist eine der zentralen Funktionen des Topologie-Editors V-TE. Da das Raster, auf dem die Komponenten abgebildet werden, relativ klein ist, reicht der Platz höchstens für ca. 20 Geräte. Danach wird die Darstellung unübersichtlich oder ist überhaupt nicht mehr möglich, da sich die Komponenten nicht mehr überlappungsfrei platzieren lassen. Um trotzdem größere Topologien darstellen zu können, werden Komponenten, die bestimmte Eigenschaften teilen, zusammengefasst. Die Gruppen werden im Editor als Objekte dargestellt und nehmen dadurch relativ wenig Platz ein. Ein Klick auf die Gruppe öffnet diese in einem weiteren Raster und erlaubt so die Einsicht der Inhalte dieser. Zurzeit können die Komponenten nur automatisch vom Back-end in Gruppen aufgeteilt werden, eine manuelle Konfiguration der Gruppen wird eventuell zu einem späteren Zeitpunkt ergänzt.

Die Informationen, aus denen die Gruppenzugehörigkeit bestimmt wird, werden direkt aus dem RDF-Modell der Topologie entnommen. Die Gruppenzugehörigkeit selbst wird ebenfalls als RDF-Information im Backup-Modell im `Dataset` gespeichert. Dazu wird die Eigenschaft `visabackup:group` verwendet. Jede Komponente kann genau einer

Gruppe angehören. Bei der Bestimmung der Gruppenzugehörigkeit werden zwei Fälle unterschieden: Besitzt eine aus einer RDF/XML Datei importierte Komponente bereits die `visabackup:group` Eigenschaft, so wird diese als native Gruppe angesehen. Die *native Gruppe* wird vom Back-end nicht überschrieben, die entsprechenden Statements werden lediglich in das Backup-Modell übertragen. So ist es zum Beispiel möglich, alle Komponenten einer VSA als Gruppe zusammenzufassen, auch wenn diese von der automatischen Gruppierung (siehe unten) in mehrere Gruppen aufgeteilt werden würden. Komponenten ohne diese Eigenschaft werden an Hand der Netzwerke, an die angeschlossen sind, gruppiert. Dabei werden die Komponenten in drei Kategorien unterteilt: Switches, Clients und Router.

Alle Geräte mit dem Wert Switch der Eigenschaft `visa:type` werden als Switches angesehen. Für diese muss bestimmt werden, welcher Gruppe sie angehören. Dazu werden alle verbundenen Interfaces der Komponente betrachtet und die an ihnen anliegenden Netzwerke bestimmt. Liegt an allen Interfaces das gleiche Netzwerk an, so wird das Gerät in die Gruppe dieses Netzwerkes einsortiert. Für den Fall das mehrere Netzwerke anliegen wird der Switch der Gruppe 0.0.0.0 hinzugefügt. Diese spezielle Gruppe wird als globale Gruppe bezeichnet und enthält alle Geräte, die außerhalb einer Gruppe auf dem Editor-Raster dargestellt werden sollen.

Als Clients werden alle Komponenten bezeichnet, die als Wert der Eigenschaft `visa:type` den String `host` besitzen und nur mit einem einzigen Netzwerk verbunden sind. Dabei kann dieses Netzwerk an einem oder mehreren Interfaces anliegen. Diese Geräte werden automatisch in die entsprechende Netzwerkgruppe einsortiert. Die Einordnung eines Geräts als Client bedeutet nicht, dass dieses auch in der Topologie ein Client-Rechner sein muss. Es kann sich dabei auch um einen Datei-Server handeln, der üblicherweise auch nur mit einem Netzwerk verbunden ist.

Die Bezeichnung Router erhalten alle Komponenten, die an mehrere Netzwerke angeschlossen sind und keine Switche sind. Diese werden grundsätzlich in die globale Gruppe `0.0.0.0` einsortiert und somit auf dem Hauptraster des Topologie-Editors dargestellt. Auch bei diesen Geräten gilt, dass sie nicht zwingend Router in der realen Topologie sind. Auch ein VPN-Gateway würde zum Beispiel als Router klassifiziert werden, da es an mindestens zwei verschiedenen Netze, intern und extern, angeschlossen ist.

Die so ermittelten Informationen werden mit der bereits angesprochenen Eigenschaft im Backup-Modell des RDF-Dataset gespeichert und bei der Umwandlung des RDF-Modells in eine Objektstruktur im Topologie-Modul berücksichtigt.

4.2.2 Verteilung von VSAs

OMF bietet verschiedene Wege, um die Software auf den Knoten einzurichten. In OMF 5.x ist es möglich, den Rechner per PXE Boot von einem Minimal-Image über das Netzwerk zu starten. Das Image enthält einen OMF Resource Controller (RC), der dann den Befehl empfängt, per IP Multicasting (“frisbee”) gesendete Daten auf die Festplatte zu schreiben. Nachdem der Vorgang beendet ist, started OMF den Rechner über den

Chassis Manager (CM) Dienst neu. Dieser Dienst ist geeignet für Umgebungen mit Knoten aus physischer Hardware, die idealerweise ein CM Modul besitzen, über welches sie ein- und ausgeschaltet werden können. Es ist auch möglich, virtuelle Maschinen per PXE aufzusetzen, jedoch gibt es bei VM Umgebungen deutlich schnellere Wege als PXE. Da in VISA auf virtuelle Umgebungen gesetzt wird, ist der PXE-Ladevorgang aus OMF 5.x eher ungeeignet für dieses Projekt.

In der Betaversion von OMF 6 ist es möglich, per libvirt und KVM virtuelle Maschinen zu starten. Dafür wurde ein spezieller Resource Proxy entwickelt, der das Skript “ubuntu-vm-builder” verwendet. Mithilfe dieses Python-Skriptes kann eine VM nach benutzerdefinierten Parametern gestartet werden. U.a. kann die Ubuntu-Version, Anzahl der CPUs, RAM Grösse usw. ausgewählt werden. Es können auch bereits existierende VMs geklont werden bzw. schlafende VMs geweckt werden. Die Funktionalität dieses VM Proxy beschränkt sich z.Zt. auf Ubuntu, KVM und libvirt. Für andere Betriebssysteme und Cloud Lösungen kann der Code erweitert werden.

Die VSAs in VISA laufen meist auf OpenStack Plattformen. OMF kann in der derzeitigen Version nicht direkt mit Openstack (Nova oder Amazon EC2 API) kommunizieren. Das bedeutet, dass die VSAs manuell gestartet werden müssen. Es ist jedoch möglich, die “nova” Befehle zum Starten der VMs an den Anfang des Experimentes zu setzen. Alternativ können die VSAs auch manuell gestartet werden (z.B. über das Webinterface “Horizon”). Als dritte Möglichkeit könnte ein OMF6 Resource Proxy entwickelt werden, der dann per EC2 oder Nova API mit OpenStack kommuniziert und VSAs starten kann.

Welche Lösung verwendet wird kommt darauf an, ob man OMF dazu benutzen möchte, die im Experiment geforderte Infrastruktur automatisch aufzusetzen. Alternativ kann man selber die VSAs aufsetzen und OMF nur für das eigentliche Experiment verwenden.

4.2.3 Netzwerkanalyse

Um das Ausmaß des Schadens zu bestimmen, der durch einen Angriff ausgelöst wird, können auf dem Zielsystem verschieden Messprogramme gestartet werden. Diese sind im OMF-Experiment verankert und nutzen die OML-Bibliothek, um die Messergebnisse in einer zentralen Datenbank zu speichern. Die Messergebnisse können dann direkt im TE dargestellt werden. Als die wichtigsten Analysetools können genannt werden:

a. **Nmap:** ist ein Portscanner, der viele Funktionen zur Netzanalyse bietet. Er kann dazu verwendet werden, um Informationen über ein Netz zu sammeln, wie z.B. die erreichbaren Hosts, oder ein System genauer auf Schwachstellen zu überprüfen.

b. **iperf:** ist ein Tool zur Messung des TCP- und UDP-Datendurchsatzes. Es kann z.B. verwendet werden, um Applikationen zu simulieren, die eine gewisse Performance erfordern. Wenn das Netz durch einen Angriff belastet wird, würde sich dies dann in den Performance-Messungen widerspiegeln.

c. **T-50:** ist ein Multi-Protokoll Packet-Injector, der unterschiedliche Pakete mit

zufallsgenerierten Eigenschaften (TTL-, ToS-Feld) verschicken kann.

d. **collectd:** sammelt Statistiken über Systeminformationen und kann diese direkt in Dateien schreiben. Es kann mit Hilfe von Plug-Ins individuell angepasst werden. Mit dem OML-Plug-In können die gesammelten Informationen direkt an einen OML-Server gesendet werden.

e. **netem:** kann den Datenverkehr in einem Netz beeinflussen, indem es die Paketinformationen modifiziert. Somit können z.B. künstlich erzeugte Latenzzeiten, Verlustraten oder defekte Pakete generiert werden.

f. **Otg2:** ist ein einfacher Traffic-Generator und kann künstlichen Datenverkehr in einem Netz erzeugen, um eine Last zu simulieren. Otr2 dient dabei als Traffic-Receiver. Es können TCP- oder UDP-Pakete mit bestimmten Eigenschaften, wie z.B. einer konstan-ten oder exponentiellen Bitrate, generiert werden.

Die Analysetools *collectd* und *netem* wurden nicht in das OMF-Framework integriert, während die anderen Toolbeispiele direkt aus OMF gestartet werden können. Zusätzlich wurde jeweils für *ping*, *Nmap* und *T-50* ein Wrapper entwickelt, indem auch das OML-Reporting integriert wurde. Das OMF-Framework bietet eine optimale Möglichkeit, die für VISA benötigten Analysetools zu steuern. Vor dem Ausführen von Experimenten mit OMF, müssen die dafür benötigten Ressourcen bereitstehen und die vorgesehenen Applikationen in das Framework integriert werden. Zu den Ressourcen gehören ein Experiment Controller sowie mindestens ein Knoten auf dem ein Ressource Controller (RC) und die Analysetools installiert werden. In VISA wird der „Aggregate Manager“ nicht zum Einrichten der Nodes benötigt, da die Analysetools im Vorfeld installiert werden sollen. Für jede Applikation musste für die Integration in das OMF-Framework, eine eigene „Application Definition“ sowie eine „Experiment Description“ entwickelt werden. Zusätzlich musste das OML-Reporting unterstützt werden, um die Messergebnisse zum „Aggregate Manager“ schicken zu können.

5 Zusammenfassung der Testergebnisse

Am Ende des VISA-Projektes wurden die Möglichkeiten und Grenzen der verschiedenen Ansätze und der entwickelten Systeme untersucht. Damit wurde der Grundstein für eine weitere Entwicklung der Prototypen hin zu einem Produkt gelegt. Zur Evaluation wurden die am Anfang erarbeiteten Referenzinfrastrukturen herangezogen und untersucht, welche in typischen Problemstellungen von KMUs durch das VISA-Projekt adressiert werden konnten.

Da von Unternehmen oft eine Zertifizierung nach existierenden Rahmenwerke wie ISO 27001 oder dem BSI-Grundschutz angestrebt wird, ist von besonderer Bedeutung, inwieweit die entwickelten VSAs diesen Rahmenwerken folgen und welche weitere Maßnahmen ein Unternehmen treffen muss, das eine Erfüllung dieser Anforderungen anstrebt. Es wurde daher beispielhaft die Grundschutzerfüllung der VSA-UTM von Collax analysiert und die vom Unternehmen durchzuführenden zusätzlichen Sicherheitsmaßnahmen (größtenteils von betrieblicher Natur) in Form einer Checkliste aufgeführt. Dies gab einen guten Überblick darüber, wie aufwändig letztendlich die Einführung von VSAs in bestehende Infrastrukturen ist.

Rahmenwerke zu IT-Sicherheit verwenden die Konzepte der Simulation und Emulation nicht explizit, um Aussagen über die Sicherheit eines Systems zu erreichen und damit eine mögliche Auditierung durchzuführen. Dies sollte in VISA aber ermöglicht werden, so dass KMUs ihre eigene Infrastruktur sicherheitstechnisch überprüfen lassen können und ggf. eine standardgerechte Dokumentation erzeugen können. Für Standards, die konkrete Anforderungen an die umzusetzenden Sicherheitsmaßnahmen stellen (z.B. PCI DSS) könnte die Erfüllung dieser Maßnahmen dabei über die Modelle verifiziert werden.

5.1 VSA-Tests

Die VSA-Tests wurden in fünf Kategorien eingeteilt, die folgende Bereiche umfassten:

a. Performancetests
b. Verfügbarkeitstests
c. Überprüfung der Sicherheitsmerkmale
d. Stresstests
e. Authentifizierungstests

Für jede dieser Kategorien wurden Tests definiert, um eine Topologie systematisch analysieren zu können. OMF-Tests wurden im Testbed durchgeführt, um die Funktion von OMF zu testen sowie die Konfiguration der Testbeds zu überprüfen. Alle Testergebnisse wurden dabei in Form von PDF-Dateien dokumentiert.

Das Testen der Topologien verlief erfolgreich. Die Experimente konnten an allen

verfügbaren Hosts gestartet und ausgeführt werden. Im Anschluss wurden die PDF-Dateien generiert, welche die detaillierten Testergebnisse und Informationen über den Zustand der jeweiligen Topologie beinhaltete. Während der Tests kam es gelegentlich zu Verbindungsproblemen zwischen den einzelnen VMs im Testbed. Diese wurden durch fehlende Routen in den Routingtabellen der Hosts verursacht. Die Ursache dieses Problems war die fehlerhafte IP-Adressvergabe mittels DHCP in OpenStack.

Die OMF-Tests können aus dem Topologie Editor V-TE gestartet werden. Über die GUI kann man an den VMs Experimente mit OMF starten. Nach Abschluss der Experimente werden die Ergebnisse automatisch dokumentiert und angezeigt. Weiterhin ist es möglich Netzschnittstellen zu manipulieren sowie detaillierte Systeminformationen der VMs abzurufen.

OMF-Experimente sowie die Manipulation von Interface-Informationen mit Netem können über das Kontextmenü einer bestehenden VM gestartet werden. Hierfür existieren entsprechende Menüeinträge mit Untermenüs. Detaillierte Systeminformationen können mit Collectd abgerufen werden.

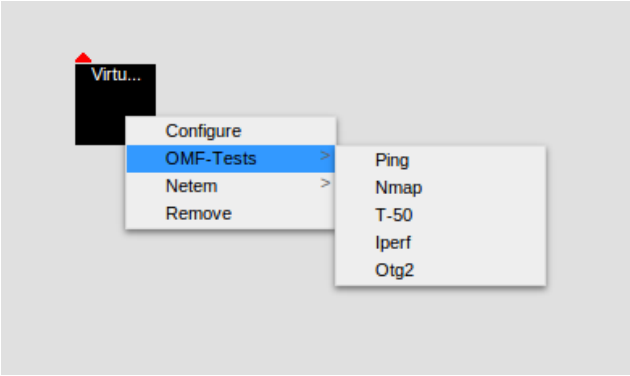

Abb. 24: Untermenü OMF-Tests

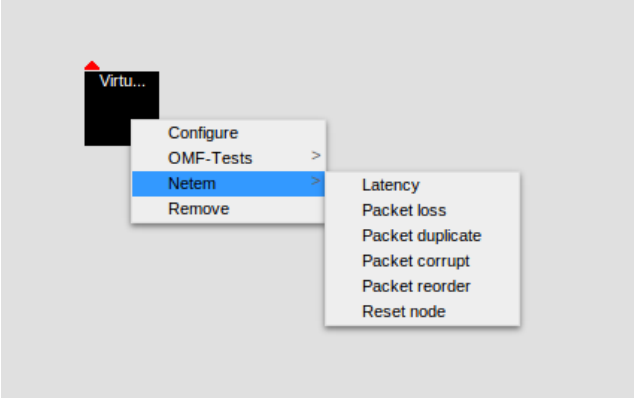

Abb. 25: Untermenü Netem

Für das Anzeigen von Testergebnissen kann der Menüeintrag „OMF-Info“ genutzt werden. Für die Auswertung mit Collectd kann der Menüeintrag „Collectd-Info“ genutzt werden.

Virtuelle und physische Umgebungen unterschieden sich in zweierlei Hinsicht. Zum einen ist eine virtuelle Umgebung i.d.R. nicht in der Lage die Performance-Eigenschaften spezialisierter Hardware mittels herkömmlicher Server-Hardware zu reproduzieren. Beispiel: Ein virtueller Switch ist – in Bezug auf Performance einem physischen Switch unterlegen. Dies stellt eine *konzeptionelle Diskrepanz* dar. Security-Tests, die ein Verhalten von Systemeigenschaften prüfen, die einer konzeptionellen Diskrepanz unterworfen sind, können in einer virtuellen Replik keine belastbaren Test-Ergebnisse produzieren. Zum Anderem ist eine virtuelle Umgebungen u.U. nicht in der Lage jede Systemfunktion zur Verfügung zu stellen, die das physische Original bereitstellen kann. Beispiel: OpenVSwitch stellt kein MVRP zur Verfügung. Dies stellt eine *funktionelle Diskrepanz* dar. Security-Tests, die eine spezielle, fehlende Systemfunktion voraussetzen, können in einer virtuellen Replik nicht durchgeführt werden. Diese Annahmen wurden auch in den Tests nachgewiesen, wobei man feststellen konnte, dass sich die virtuelle und physikalische Welt teilweise nur noch in Nuancen unterscheidet.

Unabhängig von automatisiert genutzten Tests wurden manuelle Authentifizierungstests durchgeführt, die sich auf die Dienste RADIUS, LDAP, SSH, Kerberos und DNS der VSA-AAA bezogen. Dabei kamen unterschiedliche Angriffstools zum Einsatz (z.B. Cain & Abel, sslstrip, nmap, THC-Hydra).

Erfolgreich angegriffen werden konnten dabei nur die RADIUS-Kommunikation zwischen RADIUS- und OpenVPN-Server sowie der SSH-Dienst. Allerdings wurden hierzu u.a. auch Bruteforce-Angriffe genutzt, welche sich durch ausreichend gute Passwörter und andere Techniken (künstliche Verzögerungen…) abwehren bzw. ausreichend ausbremsen lassen können. Alle anderen Penetrationstests waren mit den genutzten Tools nicht erfolgreich. Auffällig ist, dass kein Angriff auf die Weboberfläche zum Erfolg führte, obwohl hier viele verschiedene Methoden zum Einsatz kamen. Eventuell wurden hier zusätzliche Absicherungen getroffen, welche solche Angriffe abwehren oder zumindest erschweren.

5.2 IT-Compliance

Die im Projekt VISA entwickelten Virtual Security Appliances (VSAs) sollen Unternehmen dabei unterstützen, die Sicherheit ihrer IT-Infrastrukturen zu verbessern und verbreitete Sicherheitsstandards zu erfüllen. Um letzteres Ziel zu erreichen mussten die VSAs selbst die Anforderungen dieser Standards erfüllen. Dies bezieht sich nicht nur auf die technischen Sicherungsmaßnahmen, die an den Maschinen und der Infrastruktur vorgenommen werden müssen, sondern auch auf organisatorische Maßnahmen während des Betriebes, die erst vom Unternehmen selbst durchgeführt werden können. Erst wenn die von den Standards geforderten Maßnahmen umgesetzt sind, kann eine VSA sinnvoll genutzt werden. Aus diesem Grund wurden die VISA-Komponenten einer Compliance-Überprüfung unterzogen, dessen Ergebnisse hier kurz dargestellt werden.

5.2.1 Topologie-Editoren

Durch den Einsatz von Topologie-Editoren wird eine systematische Umsetzung sicherer virtueller Netze ermöglicht. Obwohl die Nutzung solcher Tools nicht in den verbreiteten

Sicherheitsstandards vorgesehen sind, wird durch ihren Einsatz der Aufbau standardkonformer ISMS vereinfacht: So enthalten die verschiedenen Grundschutzbausteine der Gruppe „Netze“ (B 4), insbesondere „Heterogene Netze“ (B 4.1) Phasen, in denen ein Netzkonzept und Netzpläne entwickelt werden. Ähnliche Anforderungen stellt auch ISO 27002, etwas abstrakter formuliert, im Kapitel „Maßnahmen für Netze“ (10.6.1). Der Standard „ISO 27033-1: Network Security“ enthält entsprechende Anforderungen an eine systematische Umsetzung einer Sicherheitsarchitektur sowie an eine Testphase. Testphasen sind auch im Grundschutz und in ISO 2700x vorgesehen (bei ISO 2700x eher indirekt durch „Systemabnahme“ (10.3.2)). Sowohl die Umsetzung als auch das Testen wird durch Topologie-Editoren vereinfacht und systematisiert. Die Umsetzung kann automatisch dokumentiert und Sicherheitsmaßnahmen direkt getestet werden. Anhand von Topologie-Editoren lässt sich außerdem die Einhaltung der verschiedenen weiteren Sicherheitsanforderungen, die an Netze gestellt werden, leichter überprüfen bzw. umsetzen.

Darüber hinaus unterstützen die Topologie-Editoren bei einem sicheren Management von Netzen. Einige Grundschutzmaßnahmen, bei deren Umsetzung Topologie-Editoren helfen können, sind „Dokumentation der Sicherheitsprozesse“ (M 2.201), „Entwicklung eines Netzkonzeptes“ (M 2.141), „Entwicklung eines Netz-Realisierungsplans“ (M 2.142), „Ist-Aufnahme der aktuellen Netzsituation“ (M 2.319), „Vorgaben zur Dokumentation und Kennzeichnung der IT-Verkabelung“ (M 2.396), „Auswahl einer geeigneten Netztopologie“ (M 5.2), „Geeigneter Einsatz von Elementen zur Netzkopplung“ (M 5.13) sowie im Bereich ISO 2700x die zahlreichen Anforderungen von Kapitel „Zugangskontrolle für Netze“ (11.4).

5.2.2 Virtual Security Appliances (VSA)

Eine wichtige Anwendung der mit den Topologie-Editoren verwalteten VSAs ist es, zu möglichen, dass mit geringem Aufwand Sicherheitsmaßnahmen in einer bestehenden IT-Infrastruktur umgesetzt werden können. Die entwickelten VSAs bieten daher gezielt Funktionalitäten, mit denen die Compliance einer IT verbessert werden kann. So kann, die im Projekt entwickelte VSA „E-Mail-Proxy“, die den E-Mail-Verkehr mit Hilfe etablierter Virenfilter gegen Malware (wie Viren, Trojaner, Würmer) schützt, genutzt werden, um die BSI-Grundschutzmaßnahme „Einsatz eines E-Mail-Scanners auf dem Mailserver“ (M 5.109) umzusetzen, die Bestandteil des Bausteins „Groupware“ (B 5.3) ist. Im Kontext von ISO 27001 hilft dieselbe VSA dabei, die Maßnahmen aus Kapitel 10.4 „Schutz vor Schadsoftware und mobilem Programmcode“ der ISO 27002 umzusetzen. Zugleich wirkt die VSA in beiden Fällen auf ein identifiziertes, nicht vernachlässigbares Risikopotenzial der Art „Schadvorfälle durch Schadprogramme in E-Mails“ risikoreduzierend, indem die Eintrittswahrscheinlichkeit des Schadensfalles durch den Proxy verringert wird. Dadurch kann das entsprechende Risiko unter das Risikoakzeptanz-Level eines bestehenden ISMS gebracht werden.

Im Gegensatz zu einer manuellen Einrichtung einer Malware-Erkennung kann so durch VISA mit sehr geringem Aufwand die Compliance verbessert werden, indem die entsprechende VSA im Topologie-Editor der bestehenden Topologie hinzugefügt wird. Entsprechend lassen sich für alle im Rahmen des Projekts entwickelten VSAs

Komponenten der Standards identifizieren, die durch ihren Einsatz erfüllt oder zumindest teilweise umgesetzt werden, so dass der Gesamtaufwand zur Erreichung der Compliance für ein Unternehmen sinkt. Oft lassen sich durch den Einsatz sinnvoller Kombinationen von VSAs im Zusammenspiel auch komplexe Sicherheitsmaßnahmen umsetzen.

5.2.3 Modellierung der VSAs

Um in einem IT-Umfeld einsetzbar zu sein, das durch die Compliance zu Sicherheitsstandards betroffen ist, müssen die VSAs der Topologie-Editoren selbst im Sinne dieser Standards sicher sein. Dass zu erfüllende Sicherheitslevel einer konkreten Anwendung variiert mitunter stark. Damit sind auch die umzusetzenden Sicherheitsmaßnahmen, um die Forderungen der Standards zu erfüllen, sehr unterschiedlich. Für VISA wird davon ausgegangen, dass jeweils der höchste Schutzbedarf bzw. die höchste Kritikalität der entsprechenden IT-Prozesse zutrifft. Wesentliche technische Sicherheitsmaßnahmen für VSAs und ihre Umgebung ergeben sich dadurch automatisch und wurden beim Design mit berücksichtigt. Weitere Maßnahmen sind ebenfalls implementiert, aber optional. Hierzu wird die Möglichkeit der Konfiguration über die Topologie-Editoren genutzt. Auf diese Weise kann konfiguriert werden, welche Sicherheitsmaßnahmen in welcher Ausprägung genutzt werden soll.

Die Auswahl der Sicherheitsmaßnahmen orientiert sich an den Vorgaben der Standards. Im Falle des BSI-Grundschutzes lassen sich den VSAs konkrete Bausteine zuordnen, aus denen sich konkrete Maßnahmen ergeben. Aus ISO 27001 bzw. ISO 27002 lässt sich ein ähnliches Paket von Sicherheitsmaßnahmen entnehmen (z.B. in Bezug auf die Zugriffskontrolle), auch wenn diese allgemein etwas abstrakter formuliert sind. Die entwickelten VSAs setzen diese Maßnahmenpakete um. Sie setzen hierzu auf etablierten Technologien sowie Open-Source-Tools auf und erreichen so neben einer hohen Sicherheit auch eine gute Kompatibilität mit bestehenden Umgebungen. Da die VSAs wenig bis keine Nutzerinteraktion benötigen, ist nur eine verhältnismäßig kleine Zahl von Sicherheitsmechanismen der erwähnten Standards umzusetzen. Tatsächlich erfüllen sie bereits in ihrer Grundkonfiguration die meisten dieser technologischen Maßnahmen.

Sicherheitsmaßnahmen können allgemein von technischer und von organisatorischer Natur sein. Während die technischen von den entwickelten Topologie-Editoren wie beschrieben erfüllt werden, müssen die organisatorischen im konkreten Einsatzumfeld umgesetzt werden. Diese organisatorischen Maßnahmen werden in der Praxis abhängig vom Einsatzgebiet, Komplexität und weiteren Rahmenbedingungen sehr unterschiedlich implementiert, weshalb sie im Rahmen von VISA nicht allgemein modelliert wurden, sondern stattdessen an einem konkreten Anwendungsfall auf die bestehende IT-Umgebung zurechtgeschnitten und in deren Betriebsprozesse integriert worden sind.

5.2.4 Zusammenfassung

Im Rahmen der Entwicklung der VSAs wurde ein großer Wert auf ein hohes Sicherheitsniveau gelegt, so dass von technischer Seite ein Großteil der Anforderungen gängiger Standards bereits erfüllt worden sind. Dies schlägt sich bei Betrachtung der entwickelten VSAs auch im Ergebnis der Analyse nieder. So ist beispielsweise die VSA-UTM gut dazu geeignet in eine IT-Umgebung integriert zu werden, die den Anforderungen des BSI-Grundschutz entspricht. Die deutliche Mehrheit der technischen Anforderungen, die sich aus den Grundschutzkatalogen für diese VSA ergeben, ist schon in der Grundkonfiguration erfüllt. Deutliche Abweichungen gibt es nur in wenigen Maßnahme, die größtenteils auch noch optional sind, wie etwa die M 4.97 „Ein Dienst pro Server“, die nicht erfüllt wird, da die VSA-UTM mehrere Dienste in sich vereint. Ein denkbarer Workaround in diesem Fall wäre, mehrere Instanzen der VSA-UTM einzusetzen und bei jeder nur einen Dienst zu aktivieren. Will man diesen Weg gehen, muss jedoch sichergestellt werden, dass die jeweils nicht genutzten Dienste vollständig deaktiviert sind und auch von diesen eventuell genutzten Hintergrunddiensten nach Möglichkeit abgeschaltet werden.

Um die VSAs grundschutzkonform einzusetzen, ist es daher notwendig entsprechende organisatorische Maßnahmen ebenfalls umzusetzen. Sie sind neben den technischen Maßnahmen unbedingt zu betrachten und umzusetzen.

6 Fazit

Die hier aufgezeigte VISA-Plattform ermöglicht die Erhebung bestehender IT-Infrastrukturen, die Umsetzung in eine virtuelle Umgebung, die Emulation verschiedener Konfigurationen sowie das erneute Ausrollen der VSAs in eine reale Umgebung. Dadurch erhält der IT-Administrator vorgefertigte IT-Bausteine, die er mittels Autokonfiguration relativ leicht in seine Umgebung einfügen und testen kann. Möchte er beispielsweise die sichere Einwahl in das Unternehmensnetz über Android ermöglichen, muss er nur auf die VSA-SRA zurückgreifen, die diverse VMs für diese Aufgabe vorkonfiguriert enthält. Vor einer Inbetriebnahme können dann über die Analysemöglichkeiten alle weiteren Konfigurationen getestet und auf die IT-Sicherheit überprüft werden. Erst wenn alle Umsetzungsschritte erfolgreich durchgeführt werden konnten, sollte die neue VSA dann in die Produktivumgebung übernommen werden. Auf der anderen Seite lassen sich bestehende IT-Infrastrukturen erheben und im TE darstellen bzw. dann ebenfalls auf Konfigurationsfehler oder Aktualisierungen evaluieren. Dadurch lässt sich das Produktionsnetz bei Konfigurationsänderungen immer wieder neu überprüfen.

Neben dem Mehrwert der neuen Dienste erhält das Unternehmen somit auch gleichzeitig eine Möglichkeit an die Hand die Compliance seiner IT-Infrastruktur zu verbessern. Dadurch wird das Sicherheitsniveau von Unternehmen letztendlich erhöht, ohne dass das entsprechende Spezialwissen vorgehalten werden muss. Dies ist besonders für KMU wertvoll, da dort dieses Detailwissen meistens nicht vorhanden ist. Aber auch Großunternehmen können von diesem Ansatz profitieren, da VISA klare Vorteile bei der Dokumentation und strukturierten Einführung bietet.

Die in VISA entwickelten Topologie-Editoren verfolgten unterschiedliche Ansätze, um eine virtuelle Umgebung erstellen, verwalten und konfigurieren zu können. Alle drei ermöglichten dabei das Erreichen eines gemeinsamen Ziels, nämlich eine zentrale Management- und Verwaltungsmöglichkeit für VMs zu schaffen. Wenn man dies mit Analyse-Funktionen und Netztopologie-Konfiguration koppelt, wie dies in VISA geschehen ist, können sichere und leistungsfähige VSA-Komponenten entstehen, die bereits einen Großteil der Compliance-Anforderungen abdecken, die an ein Unternehmen heute gestellt werden. Zusätzlich können so im Vorfeld Netztopologien getestet werden, bevor sie in die Produktivumgebung gelangen. Somit ermöglichten Topologie-Editoren in jedem Fall eine Erhöhung des Sicherheitsgrads.

Bestehende Prozesse wie das „Change Management“ müssen auf die VSAs zusätzlich angewandt werden, um eine vollständige Abdeckung der Sicherheitsmaßnahmen zu erreichen. Hierfür wurde ein Compliance-Katalog entwickelt, der angibt, welche organisatorischen Maßnahmen umgesetzt werden müssen. Dadurch erhalten die Topologie-Editoren eine hohe Flexibilität, um sie in unterschiedlichen Netzen einsetzen zu können. Das VISA-Projekt kann auf eine erfolgreiche Umsetzung aller Anforderungen zurückblicken. Die im Projekt entwickelten Lösungen werden Einfluss auf weitere Forschungsprojekte nehmen und auch in Produkte (z.B. bei Collax) umgesetzt werden.

7 Anhang

7.1 Literaturverweise

[1] K.-O. Detken, M. Jahnke, H. Birkholz, C. Dwertmann: Design und Implementierung von Virtual Security Appliances (VSA). D.A.CH Security 2013, ISBN 978-3-00-042097-9, Hrsg. Peter Schartner u. Peter Trommler, syssec-Verlag, Nürnberg 2013

[2] E. Eren, K.-O. Detken, F. Krämer, S. Müller: Topologie-Editoren zur graphischen Konzeption von VSAs. D.A.CH Security 2013, ISBN 978-3-00-042097-9, Hrsg. Peter Schartner u. Peter Trommler, syssec-Verlag, Nürnberg 2013

[3] C. Rudolph, K.-O. Detken: Das VISA-Projekt. Statustreffen der Forschungsvorhaben der IT-Sicherheitsforschung, 17.-18.09.13, Technische Universität Darmstadt, Darmstadt 2013

[4] K.-O. Detken, M. Jahnke, H. Birkholz, C. Dwertmann: Design and implementation of Virtual Security Appliances (VSA) for SME. 7th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems, Technology and Applications, 12.-14. September, University of Applied Sciences, Hochschule für Technik und Wirtschaft (HTW), Berlin 2013

[5] H. Birkholz, I. Sieverdingbeck, N. Kuntze, C. Rudolph: Enhancing security testing via automated replication of IT-asset topologies. 8th ARES Conference, 2nd - 6th of September, University of Regensburg, Regensburg 2013

[6] Mike Zink (University of Massachusetts), Shufeng Huang (University of Kentucky), and Divya Bhat (University of Massachusetts): Run a Complete Experiment in GENI using GIMI and labwiki. 22nd of July, GEC17 conference,University of Wisconsin – Madison, USA 2013

[7] Kai-Oliver Detken, Dennis Dunekacke: Testen und Präsentieren der IF-MAP-Client-Entwicklungen mit anderen Herstellerlösungen im Rahmen des Forschungsprojektes VISA. Internationales PlugFest der Trusted Computing Group (TCG), 02.-03. April, Ort: Fraunhofer SIT in Darmstadt, Darmstadt 2013

[8] Kai-Oliver Detken: Sichere virtuelle Sicherheitskomponenten für KMUs. Präsentation des VISA-Projektes auf dem Bremer Gemeinschaftsstand der CeBIT, CeBIT-Stand in Halle 6, 8. März, Hannover 2013

[9] Kai-Oliver Detken: Server und Netzvirtualisierung für die Produktivumgebung. Handbuch der Telekommunikation, Verlagsgruppe Deutscher Wirtschaftsdienst, 154. Ergänzungslieferung, November 2012, ISBN 978-387-156-096-5, München/Neuwied/Köln 2012

[10] Kai-Oliver Detken: VISA - Virtual IT Security Architectures. Protokoll und Dienste

der Informationstechnologie, Herausgeber: Heinz Schulte, Ausgabe-Nr. 06/2012, Dezember 2012, WEKA Media GmbH & Co. KG, ISBN 978-3-8245-4066-2, Kissing 2012

[11] Kai-Oliver Detken: Virtualisierung - Virtualisierung von Serversystemen und Netzen. Protokoll und Dienste der Informationstechnologie, Herausgeber: Heinz Schulte, Ausgabe-Nr. 06/2012, Dezember 2012, WEKA Media GmbH & Co. KG, ISBN 978-3-8245-4066-2, Kissing 2012

[12] Kai-Oliver Detken: Virtualisierung von Serversystemen und Netzen. Vom LAN zum Kommunikationsnetz: Netze und Protokolle, Ausgabe-Nr. 05/2012, Oktober 2012, WEKA Media GmbH & Co. KG, ISBN 978-3-8245-3501-7, Kissing 2012

[13] Kai-Oliver Detken: Trusted Computing: Flop oder Durchbruch des TPM-Chip? NET 09/12, ISSN 0947–4765, NET Verlagsservice GmbH, Woltersdorf 2012

[14] K.-O. Detken, E. Eren, M. Steiner: Erhöhung der IT-Sicherheit durch Konfigurationsunterstützung bei der Virtualisierung. D.A.CH Security 2012: Bestandsaufnahme, Konzepte, Anwendungen und Perspektiven, Herausgeber: Peter Schartner, syssec Verlag, ISBN 978-3-00-039221-4, Konstanz 2012

[15] K.-O. Detken, A. Oberle, N. Kuntze, E. Eren: Simulation Environment (SE) for mobile Virtualized Security Appliances (VSA). 1st IEEE International Symposium on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems, 20.-21. September, ISBN 978-1-4673-4677-1, University of Applied Sciences Offenburg, Offenburg 2012

[16] Poster- und Demonstratorausstellung bei der BMBF-Tagung "KMU-Innovativ". Vorstellung des Forschungsprojektes VISA im Rahmen einer Fachausstellung bei der BMBF-Tagung, Förderprogramm "KMU-Innovativ", 17.-18. September, Dresden 2012

[17] Kai-Oliver Detken: Kooperationstreffen in Frankfurt (Oder) bei der IHP GmbH. Präsentation der Firma DECOIT GmbH und des Projektes VISA, 08. August at IHP, Frankfurt (Oder) 2012

[18] Kai-Oliver Detken: Virtualisierung ganzer Netzsegmente. Vortrag auf der CeBIT am Bremer Gemeinschaftsstand, Halle 6, 9. März, Hannover 2012

[19] Kai-Oliver Detken: Virtualisierung ganzer Netze - Überprüfung der eigenen Infrastruktur. NET 12/11, ISSN 0947–4765, NET Verlagsservice GmbH, Woltersdorf 2011

[20] Evren Eren, Ghannam Aljabari: Virtualization of WLAN Infrastructures. Proceedings of the 6th IEEE International Workshop on - Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), ISBN 978-1-4577-1423-8. Prague - Czech Republic 2011

[21] Kai-Oliver Detken: Trusted Network Connect - die sichere Einwahl mobiler Mitarbeiter ins Unternehmen. Handbuch der Telekommunikation, Deutscher Wirtschaftsdienst, 143. Ergänzungslieferung, Februar 2011, Köln 2011

[22] Evren Eren: Virtualisierung von IT-Sicherheitsinfrastrukturen für Unternehmensnetze. 1st esproject - Experience in Information Security Projects, 23.-24. November 2010, Berlin

[23] Evren Eren: Virtualisierung in den Informations- und Kommunikationstechnologien. Science Days 2010, 02.-03. November, Hochschule für Telekommunikation Leipzig, Leipzig 2010

[24] Kai-Oliver Detken: Sicherheit in mobilen Welten. WEKA-Verlag, Vom LAN zum Kommunikationsnetz: Systeme und Applikationen, Ausgabe-Nr. 05/2010, November 2010, WEKA Media GmbH & Co. KG, ISBN 978-3-8245-3502-6, Kissing 2010

[25] Evren Eren: LISA – A Reference Architecture for IT-Security Applications. Keynote Speech, 3rd International Students Academic Presentation, 16. March 2010, Yuanpei University, HsinChu - Taiwan

[26] Evren Eren: Scheinwelten – Mit Virtualisierungssoftware ganze Infrastrukturen abbilden. NET - Zeitschrift für Kommunikationsmanagement, 09/10

[27] Evren Eren: Datenschutz in der Praxis - Sicherheit der IT in medizinischen Versorgungszentren. NET - Zeitschrift für Kommunikationsmanagement, 1-2/10

[28] Evren Eren: Virtualisierung komplexer Netzwerke. Virtualisierung in der ICT-Branche, INFORMATIK 2010 Service Science – Neue Perspektiven für die Informatik. 40. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 27. September bis 1. Oktober, Leipzig

7.2 Abbildungsverzeichnis

7.3 Tabellenverzeichnis