Design und Implementierung von Virtual Security Appliances

Kai-Oliver Detken[1] · Marcel Jahnke[1]
Henk Birkholz[2] · Christoph Dwertmann[3]

[1]DECOIT GmbH
{detken | jahnke}@decoit.de

[2]Fraunhofer-Institute for Secure Information Technology
henk.birkholz@sit.fraunhofer.de

[3]National ICT Australia Limited, Australien
christoph.dwertmann@nicta.com.au

Zusammenfassung

Die Einbindung von neuen Sicherheitstechnologien in bereits bestehenden Netztopologien erfordert einen großen Testaufwand, bevor diese im Produktivbetrieb genutzt werden können. Hinzu kommt, dass mittlerweile auch in kleineren Unternehmen komplexe IT-Infrastrukturen betrieben werden, durch deren Komplexität Sicherheitslücken entstehen können. Hier setzt das BMBF-Projekt VISA (www.visa-project.de) an, indem eine Simulationsumgebung für teilautomatisierte Tests auf Basis einer erhobenen oder erstellten Netztopologie realisiert werden kann. Eines der Ziele des VISA-Projektes ist dabei die Entwicklung von sog. Virtual Security Appliances (VSA) für Unternehmensnetze, die es dem Benutzer erlauben virtuelle Maschinen (VM) und Sicherheitsdienste in die Simulationsumgebung automatisiert einzubauen und ausreichend zu testen. Durch diesen geschaffenen Regelkreislauf, können so IT-Infrastrukturen analysiert werden, bevor man sie produktiv einsetzt bzw. man kann Konfigurationen testen und später in die physikalische Umgebung implementieren. Dadurch erhoffen sich die Projektpartner eine Erhöhung des Sicherheitsniveaus in Unternehmen und eine Erleichterung der Implementierung neuer Netzdienste.

1 Einleitung

Durch die starke Heterogenität von IT-Infrastrukturen, die relativ begrenzten Ressourcen sowie das relativ geringe Know-how müssen in Zukunft insbesondere KMU bessere und geeignetere Methoden zur flexiblen Konfektionierung, Erprobung und Optimierung ihrer IT-Infrastrukturen an die Hand bekommen. Dies ist insbesondere für die IT-Sicherheit wichtig. Der Markt für Virtualisierung und IT-Sicherheit adressiert jedoch diese Zielgruppe bis dato zu wenig, sodass meistens 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 modulare und erprobte Lösungen und Systeme essentiell. Dies will das vom BMBF geförderte F&E-Projekt VISA durch die Entwicklung von sog. Virtual

P. Schartner · P. Trommler (Hrsg.) · D•A•CH Security 2013 · syssec (2013) 98-110.

Security Appliances (VSA) sowie Methoden und Tools zur Netzmodellierung und -simulation adressieren.

Nicht nur vor dem Hintergrund der Flexibilität, sondern auch aus Kostengründen (Investition in Hard- und Software) sind VSA auf Basis von Open-Source-Bausteinen (sowohl die Anwendungen, als auch die Betriebssysteme) von Bedeutung. Dabei würde der Einsatz in Unternehmen kein größeres Know-how erfordern, da erprobte State-of-the-Art-Technologien als integrierbare Lösung in die Unternehmens-IT eingebunden werden können. Diese können ein anerkanntes Maß an Sicherheitsmaßnahmen und Standard-Compliance-Anforderungen bereits automatisch umsetzen.

Es ist 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 basiert auf drei Kernmerkmalen:

- Simulation und Evaluierung der gesamten IT-Infrastruktur in virtuellen Umgebungen
- Realisierung von Sicherheitsanwendungen als virtuelle Komponenten, sog. Virtual Security Appliances (VSA)
- Vereinfachung und Nachweisbarkeit der Einhaltung von IT-Standards, IT-Security- und Compliance-Anforderungen

Durch das VISA-Rahmenwerk wird der passgenaue und vereinfachte Einsatz von Sicherheitsanwendungen auf Basis von VSAs ermöglicht. Durch die umfassende Simulation der IT-Infrastrukturen können die betriebsrelevanten Parameter und die Integrationspunkte der VSA bereits in der virtuellen Umgebung identifiziert und der Einsatz erprobt werden. [DSBW12]

2 Simulationskomponenten

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

- Topologie Editor (TE)
- Simulation Compiler (SC)
- Simulation Environment (SE)

In der Simulationsumgebung ist 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. [Detk13]

Die Aufgabe des Simulations-Compilers wird durch das IO-Tool-Set übernommen (siehe Kapitel 3). 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[1]-Format. Aus diesem Grund wird für die Kommunikation zwischen TE und IO das weniger komplexe Format RDF/XML[2] [BeMc04] verwendet. Beide Formate basieren auf Triple und können dafür genutzt werden, um Topologien als Graphen darzustellen.

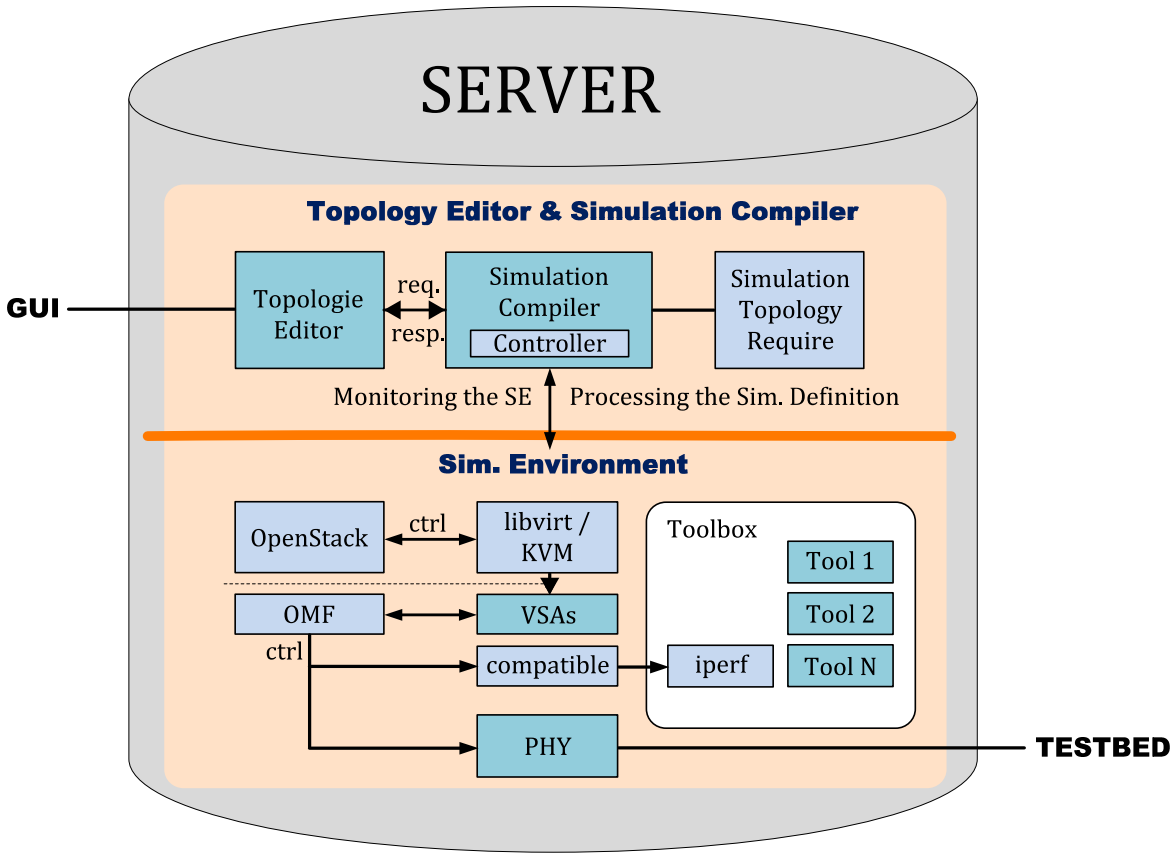

Abb. 1: Aufbau des VISA-Rahmenwerks

Ein Teil der Simulationsbeschreibung dient dem OMF-Rahmenwerk [ROJS10] als Eingabeparameter (siehe Kapitel 4). 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. [DOKE13]

2.1 Topologie Editor (TE)

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 nachmodelliert werden. Der TE besteht aus zwei Kernkomponenten: aus dem Back-end (in Java geschriebener Serverdienst) und dem Front-end (Web-basierte grafische Oberfläche).

[1] Web Ontology Language (OWL) / Extensible Markup Language (XML)

[2] Resource Description Framework (RDF) / Extensible Markup Language (XML)

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:

- Topologie-Modul
- RDF-Modul
- 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 wird aber nicht mehr innerhalb des VISA-Projektes umgesetzt werden. Eine ausführlichere Darstellung des TE ist in dem zweiten Beitrag „Topologie Editoren zur graphischen Konzeption von virtuellen Sicherheitsinfrastrukturen“ zu finden.

2.2 Netzwerktests

Zusätzlich ist es auch möglich vordefinierte Netzwerktests durch den Topologie Editor (TE) anzustoßen, um die virtuelle bzw. simulierte IT-Infrastruktur auf verschiedene Parameter (z.B. Performance, Konfiguration) zu testen. Die Tests lassen sich in folgende Bereiche unterteilen:

- Simulation von Angriffen
- Messung der Auswirkungen
- Emulation von Netzparametern

Das cOntrol and Management Framework (OMF) [ROJS10] ermöglicht es, skriptgesteuerte Programme in den VMs zu starten, um Angriffe simulieren zu können (siehe Kapitel 4). 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.

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[3]. Die Messergebnisse können dann direkt im TE dargestellt werden. Als die wichtigsten Analysetools können genannt werden:

[3] URL: http://oml.mytestbed.net/projects/omlapp/wiki

- **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.
- **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.
- **T-50:** ist ein Multi-Protokoll Packet-Injector, der unterschiedliche Pakete mit zufallsgenerierten Eigenschaften (TTL-, ToS-Feld) verschicken kann.
- **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.
- **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.
- **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 konstanten 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.

3 Interconnected-asset Ontology (IO) Tool-Set

Das IO-Tools-Set stellt 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, werden 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. Abbildung 2 zeigt einen vereinfachten Ausschnitt eines einzelnen IT-Assets zur Verdeutlichung des formalen Modells.

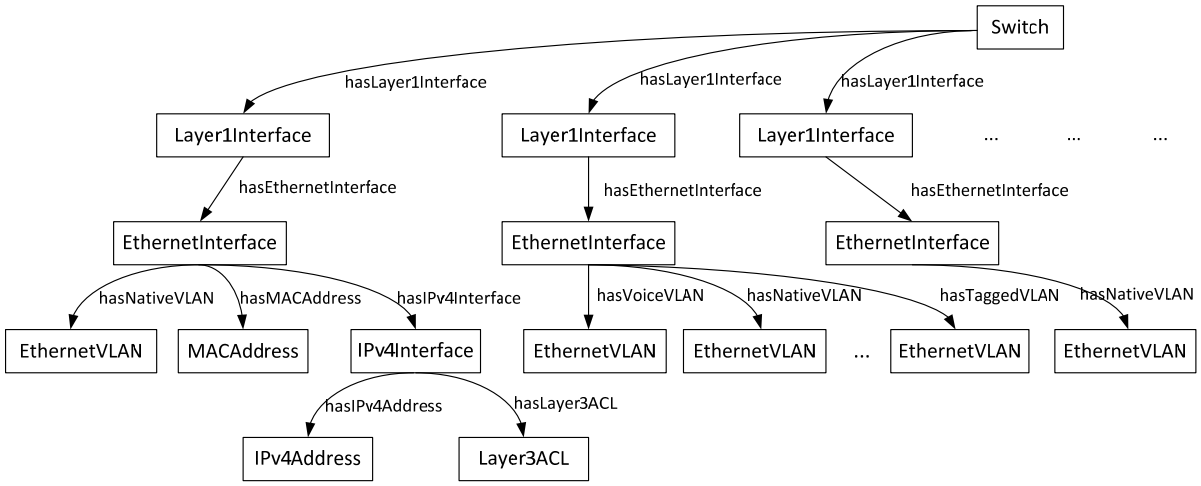

Abb. 2: Vereinfachter Ausschnitt eines einzelnen IT-Assets

Die Interconnect-asset Ontology (IO) bildet netzspezifische Informationen in hohem Detailgrad ab, was für den Einsatz als Simulation-Compiler erforderlich ist. Dabei basiert die Modellierung der ontologischen Repräsentation auf verbreiteten Standards wie DMTF CIM (Distributed Management Task Force, Common Interface Modul) oder dem OSI-Schichtenmodell. Abbildung 3 verdeutlicht den Detailgrad des formalen Modells.

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

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

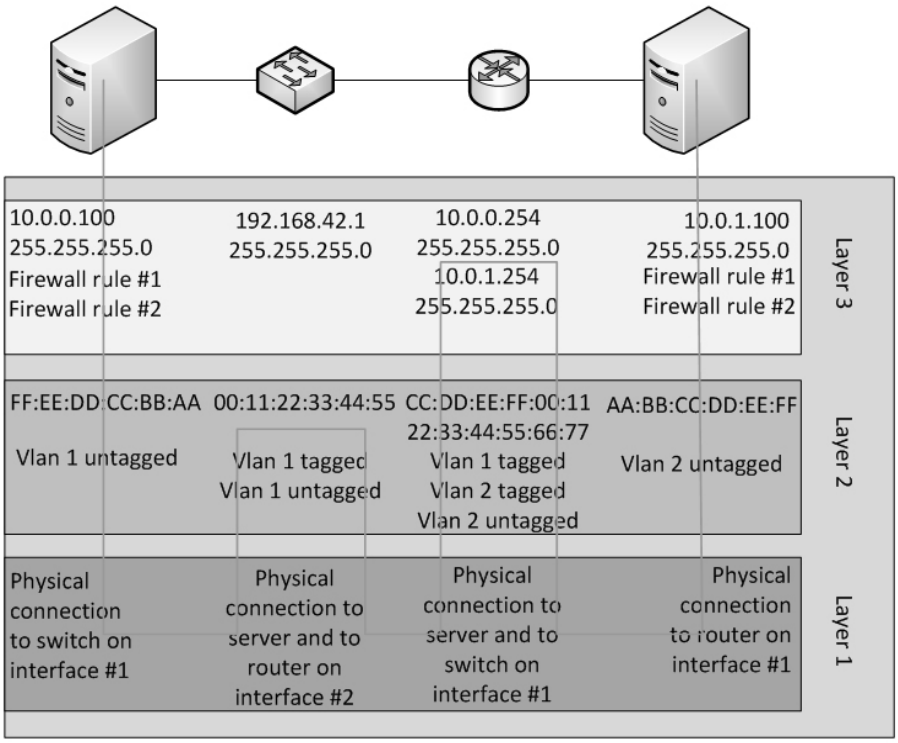

Abb. 3: Beispielhafte Darstellung eines Netz-Pfades

Die Kommunikation zwischen TE und IO-Tool-Set findet mit Hilfe textbasierte Nachrichten statt, die über eine TLS-gesicherte TCP-Verbindung übertragen werden. Jede Zeile in dieser Kommunikation enthält genau eine Nachricht. Der grundlegende Ablauf der Kommunikation ist 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, sodass 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, sodass 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.

Abbildung 4 verdeutlicht den beschriebenen Ablauf der Kommunikation.

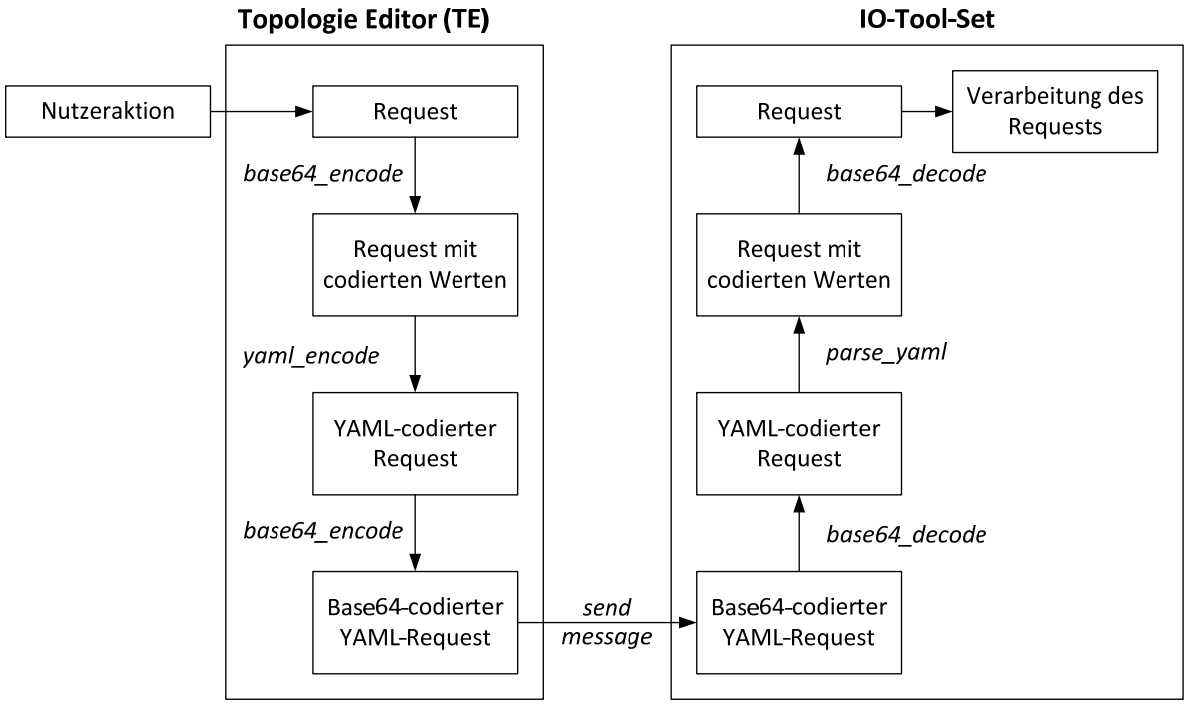

Abb. 4: Kommunikationsablauf zwischen TE und dem IO-Tool-Set via IO-X

Zwar kann dieses Protokoll das Transfervolumen um bis zu 75% erhöhen, erreicht 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.

4 Experiment-Steuerung und Messung

OMF [ROJS10] ist ein Framework für die Durchführung von Experimenten auf einem Testbed. Jeder am Experiment teilnehmende Knoten (Rechner/VM) führt dazu den OMF Resource Controller (RC) aus. Dieser Daemon 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[4]-Sprache [ROJS10] 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.

Eine weitere Komponente von OMF-Testbeds ist der Aggregate Manager (AM). Diese Komponente kann verschiedene Dienste im Testbed bereitstellen, z.B. den Dienst Chassis Mana-

[4] OMF Experiment Description Language

ger (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.

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 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, verwendet das VISA-Projekt die OML-Messbibliothek [WJRO10]. 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 http://oml.mytestbed.net verfügbar.

In einem typischen Szenario im VISA-Projekt wird eine Reihe von VSAs eingesetzt, um ein Unternehmen zu simulieren. Eine weitere VSA spielt dann den externen Angreifer, der eine Attacke auf das Unternehmensnetz vornimmt. 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 aufzuzeichnen. 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.

5 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 (Abbildung 5) entwickelt. Die VSA-Metadata-Access-Control (VSA-MAC) dient dem interoperablen Austausch sicherheitsrelevanter Daten zwischen beliebigen Netzwerkkomponenten auf Basis des IF-MAP-Protokolls. IF-MAP-Clients können 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) ermöglicht den sicheren Zugriff eines mobilen Mitarbeiters auf ein Unternehmensnetzwerk mittels eines Android-Smartphones. Als Basis dient die Spezifikation Trusted Network Connect (TNC). [DES12]

Die beiden VSAs können 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, müssen die VMs der VSA sich selbst konfigurieren.

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.

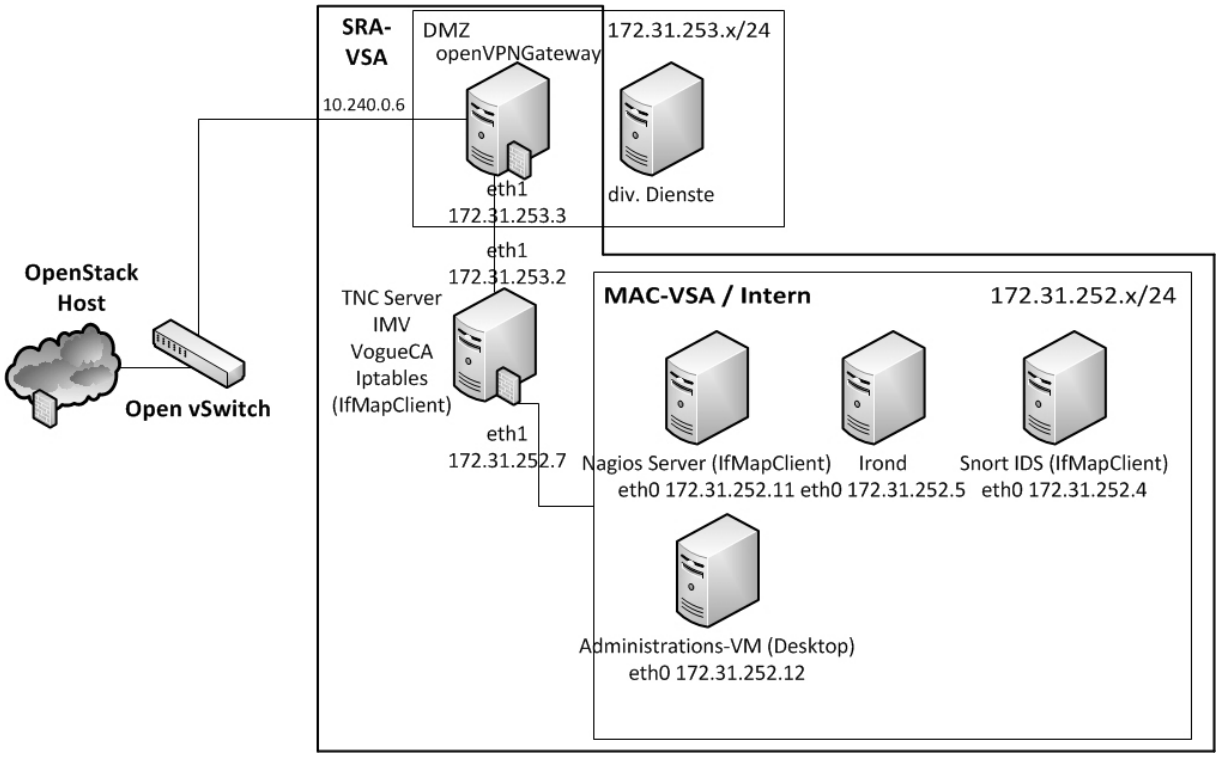

Abb. 5: Basistopologie der VSA-SRA und VSA-MAC

In VISA wird 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 müssen die IP-Adressen zur Laufzeit der VMs ermittelt werden, da O-

penStack in der verwendeten Essex-Version keine Vergabe von statischen IP-Adressen unterstützt. Dafür wird von Puppet ein OpenStack-Nova-Befehl auf dem OpenStack-Host ausgeführt und die Ausgabe in die relevanten Informationen zerlegt. Die so ermittelten IP-Adressen können in den Templates verwendet und so die benötigten Einstellungen angewandt werden.

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

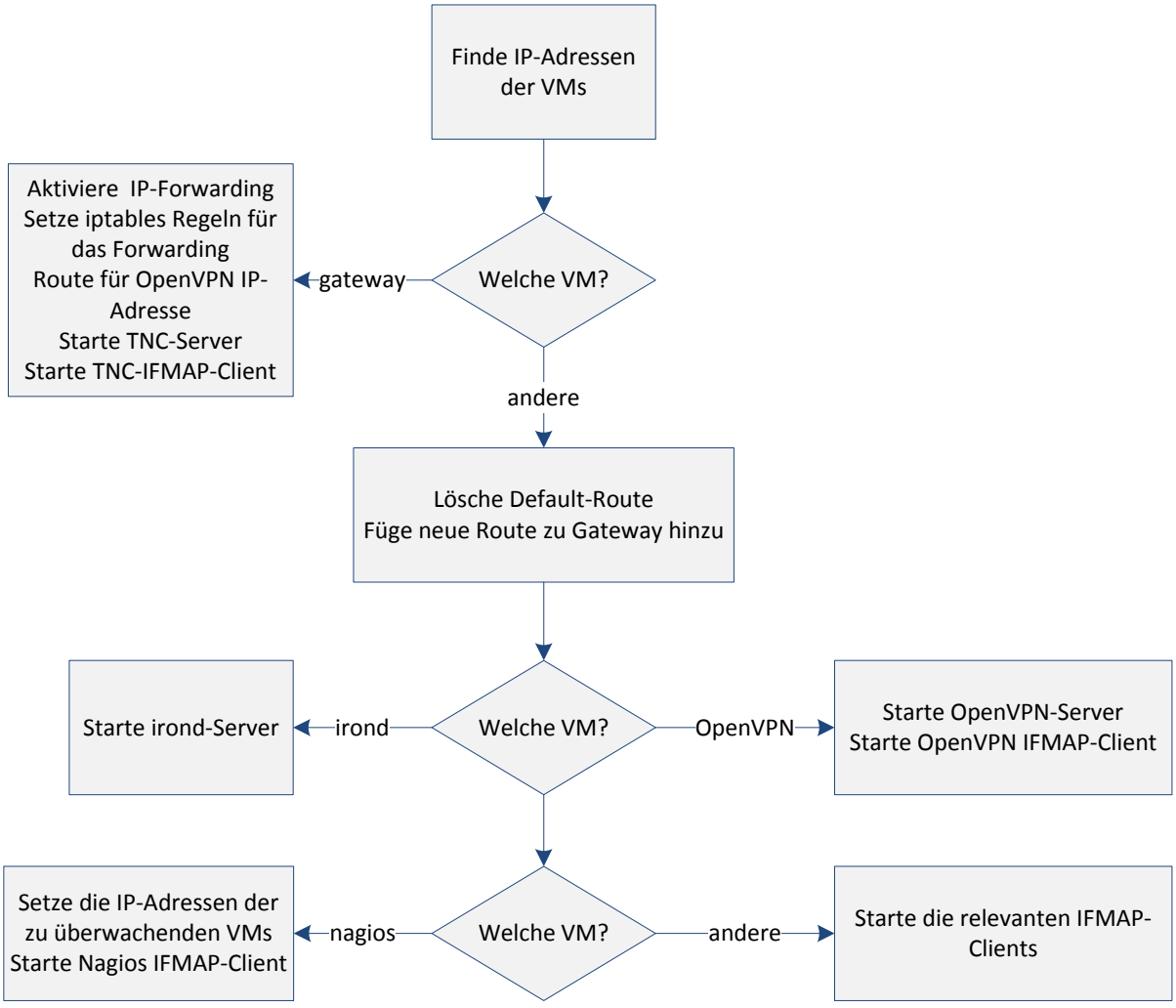

Abb. 6: Ablauf der automatischen Konfiguration

Das Ermitteln des VM-Typs (OpenVPN, Gateway etc.) erfolgt über den Hostnamen der VM. Hierzu ist es notwendig, dass z.B. die Gateway-VM in ihrem Hostnamen das Schlüsselwort „Gateway“ enthält. Hierbei können aber sowohl vor dem Schlüsselwort, als auch nach diesem weitere Zeichen vorkommen. Die Authentifikation am Puppet-Server folgt nach dem gleichen

Schema. Wird von dem Benutzer eine VM hinzugefügt, die dieses Schlüsselwort enthält, hat dies keine weiteren Auswirkungen, da das „Dummy"-VM-Image keinen Puppet-Client enthält und Puppet die Konfiguration nur dann übermittelt, wenn diese vom Client angefragt wird. Abbildung 6 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.

6 Ausblick

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 kann man auch das Produktionsnetz bei Konfigurationsänderungen immer wieder 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.

Danksagung

Das VISA-Projekt ist ein gefördertes BMBF-Projekt mit einer Laufzeit von zwei Jahren, das im August 2011 seine Arbeiten begonnen hat und Ende September 2013 endet. An dem Projekt sind die Firmen DECOIT GmbH (Projektleitung), Collax GmbH, IT-Security@Work GmbH sowie die deutschen Forschungseinrichtungen Fraunhofer SIT und Fachhochschule Dortmund beteiligt. Zusätzlich ist der australische Partner NICTA (National ICT Australia) mit im Konsortium vertreten, der maßgeblich seine OMF-Arbeiten beigesteuert hat. Unter der URL-Adresse http://www.visa-project.de sind aktuelle Informationen und Software-Downloads der Entwicklungsarbeiten zu finden.

Literatur

[Apac12] The Apache Software Foundation: Apache Jena Framework. http://jena.apache.org/index.html, 2011-2012. – Stand 18.10.2012.

[BeMc04] D. Beckett and B. McBride: *RDF/XML syntax specification (revised).* W3C recommendation, Vol. 10, 2004.

[BSKB12] H. Birkholz, I. Sieverdingbeck, K. Sohr, C. Bormann: *IO: An interconnected asset ontology in support of risk management processes*. ARES 2012, Security Ontology Workshop, Bremen 2012.

[DES12] K.-O. Detken, Eren, Steiner: *Erhöhung der IT-Sicherheit durch Konfigurationsunterstützung bei der Virtualisierung.* D.A.CH Security 2012: : Bestandsaufnahme, Konzepte, Anwendungen und Perspektiven, Herausgeber: P. Schartner und J. Taeger, syssec Verlag, ISBN 978-3-00-039221-4, Konstanz 2012.

[Detk13] K.-O. Detken: *Werkzeugkasten ohne Anleitung - vielfältige Tools für die Planung einer virtuellen Infrastruktur.* NET 04/13, ISSN 0947–4765, NET Verlagsservice GmbH, Woltersdorf 2013.

[DOKE13] K.-O. Detken, Oberle, Kuntze, 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.

[DSBW12] K.-O. Detken, Scheuermann, Bente, Westerkamp: *Automatisches Erkennen mobiler Angriffe auf die IT-Infrastruktur*. D.A.CH Security 2012: Bestandsaufnahme, Konzepte, Anwendungen und Perspektiven, Herausgeber: Peter Schartner und Jürgen Taeger, syssec Verlag, ISBN 978-3-00-039221-4, Konstanz 2012.

[ROJS10] T. Rakotoarivelo, M. Ott, G. Jourjon, I. Seskar, *OMF: a control and management framework for networking testbeds.* ACM SIGOPS Operating Systems Review 43 (4), 54-59, Jan. 2010.

[WJRO10] J. White, G. Jourjon, Thierry Rakotoarivelo, and Max Ott, *Measurement architectures for network experiments with disconnected mobile nodes.* TridentCom 2010, May 2010.