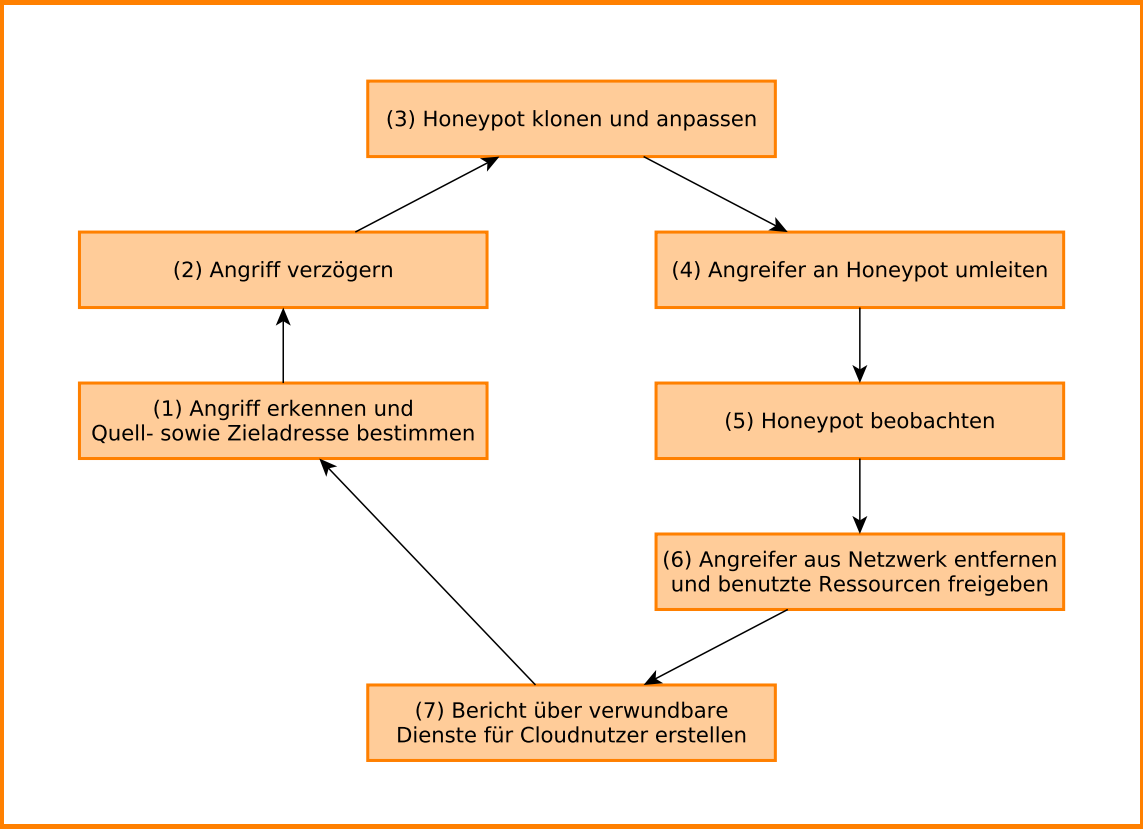
Open C3S
Open Competence Center for Cyber Security

UNIVERSITÄT PASSAU
Fakultät für Informatik und Mathematik



Mikromodul 8005: Einbruchserkennung und Honeypots in der Cloud

Autoren:
Prof. Dr. Hans P. Reiser
Noëlle Rakotondravony
Johannes Köstler

Universität Passau

Mikromodul 8005: Einbruchserkennung und Honeypots in der Cloud

Autoren:

Prof. Dr. Hans P. Reiser

Noëlle Rakotondravony

Johannes Köstler

---

1. Auflage

Universität Passau

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

AUFSTIEG DURCH BILDUNG >>
OFFENE HOCHSCHULEN

Universität Passau
Fakultät für Informatik und Mathematik
Innstraße 43
94034 Passau

1. Auflage (4. Mai 2017)

Um die Lesbarkeit zu vereinfachen, wird auf die zusätzliche Formulierung der weiblichen Form bei Personenbezeichnungen verzichtet. Wir weisen deshalb darauf hin, dass die Verwendung der männlichen Form explizit als geschlechtsunabhängig verstanden werden soll.

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 16OH12025 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

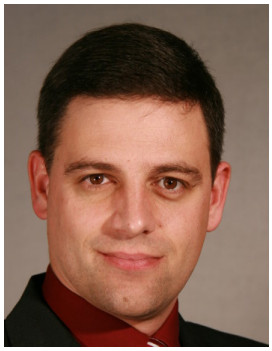
Inhaltsverzeichnis

Einleitung

I. Abkürzungen der Randsymbole und Farbkodierungen

Beispiel	B
Definition	D
Übung	Ü

II. Zu den Autoren



Hans P. Reiser ist Juniorprofessor für Sicherheit in Informationssystemen an der Universität Passau. Schwerpunkte seiner Arbeitsgruppe sind die Weiterentwicklung von Konzepten und Systemen aus dem Bereich der fehler- und einbruchstoleranten Replikation, die frühzeitliche und umfassende Erkennung von Sicherheitsproblemen und Sicherheitsvorfällen in Cloud-Umgebungen sowie die Erforschung neuartiger Sicherheitskonzepte auf Hypervisorebene.

Noëlle Rakotondravony ist seit September 2015 wissenschaftliche Mitarbeiterin in der Arbeitsgruppe von Prof. Hans P. Reiser an der Universität Passau.

Johannes Köstler ist seit Mai 2015 wissenschaftlicher Mitarbeiter in der Arbeitsgruppe von Prof. Hans P. Reiser an der Universität Passau.

Mikromodul: Einbruchserkennung und Honeypots in der Cloud

1 Lernziele

Nach Bearbeitung dieses Mikromoduls kennen Sie die Grundbegriffe und Funktionsweisen von Einbruchserkennungssystemen (Intrusion Detection System – IDS). Neben den beiden grundlegenden Varianten der netzbasierten und der hostbasierten Einbruchserkennung sind Sie vertraut mit den Möglichkeiten, die Virtualisierungstechnik zur Einbruchserkennung bietet (Einbruchserkennung auf Basis der Introspektion virtueller Maschinen). Auf dieser Basis sind Sie in der Lage eigenständig eine Strategie zur Einbruchserkennung in Cloud-Umgebungen zu definieren. Des Weiteren kennen Sie die wesentlichen Arten von Honeypots und haben einen Einblick in die ausgewählten Beispielsysteme. Auch sind Sie mit der der Verwendung von VMI in „High-Interaction Honeypots“ vertraut. Sie sind in der Lage, Honeypots zu betreiben, Erkenntnisse daraus abzuleiten sowie Vor- und Nachteile zu beurteilen.

2 Motivation

Auch wenn die Bestrebungen zum Entwurf und der Umsetzung von sicherer Software und IT-Systemen hoch sind, können diese niemals eine perfekte Sicherheit bieten. Trotz stetiger Fortschritte im Bereich der IT-Sicherheit kann keinesfalls von einer Entspannung der Gefahrenlage die Rede sein. So kommt das Bundesamt für Sicherheit in Informationstechnik in ihrem jährlichen Sicherheitsbericht zu dem Fazit, „dass die Komplexität der Bedrohungslage ebenso wie die damit einhergehenden Gefahren für die fortschreitende Digitalisierung zunimmt“ (de Maizière, 2016). Vor allem beim Auftreten von Sicherheitslücken und bei der Verbreitung von Schadsoftware seien deutliche Anstiege zu verzeichnen.

Im Folgenden wollen wir der Frage nachgehen, wie man mit dieser Situation in der Praxis umgehen kann. Während dieses Problem grundsätzlich für alle IT-Systeme von großer akuter Bedeutung ist, stellt sich auch die Frage, welche Auswirkungen hierbei der wachsende Einsatz von Cloud-Infrastrukturen hat. Gerade bei öffentlichen Cloud-Infrastrukturen im IaaS- und PaaS-Modell ist man diesen Herausforderungen im gleichen Maße ausgeliefert, hat als Cloud-Nutzer aber – im Vergleich zum Nutzer einer dedizierten Inhouse-Infrastruktur – nur einen beschränkten Handlungsspielraum.

Dieser Studienbrief betrachtet zwei Konzepte: Einbruchserkennung und Honeypots. Systeme zur Einbruchserkennung versuchen sich den in der Realität unvermeidbaren Angriffen entgegenzustellen, um deren Auswirkungen auf ein Minimum zu reduzieren. Dafür beobachten sie entweder zu bestimmten Zeitpunkten oder kontinuierlich die Ausführung des Systems und die durchlaufenen Zustände. Die Erkennung der Angriffe erfolgt entweder durch die Analyse des Systemverhaltens oder durch den Abgleich mit bereits bekannten Angriffsmustern. Zur Gewinnung solcher Muster können sogenannte Honeypots eingesetzt werden. Das Konzept bedient sich dabei der Metapher des Honigtopfs, welcher in vielen Geschichten als Lockmittel die Aufmerksamkeit von Bären erregen soll. Analog dazu geben sich digitale Honeypots als begehrenswerte und zumeist auch simple Ziele aus. Diese beiden Konzepte werden in diesem Abschnitt näher erläutert, bevor sich die folgenden Abschnitte genauer mit Implementierungen solcher Systeme in der Praxis beschäftigen.

2.1 Einbruchserkennungssysteme

Einbruchserkennung zielt darauf ab, erfolgte und aktive Angriffe auf ein Produktivsystem zu erkennen, um dadurch geeignet darauf reagieren zu können. Systeme, die eine solche Erkennung ermöglichen, werden als *Einbruchserkennungssysteme* bzw. *Intrusion Detection System (IDS)* bezeichnet. Eine etwas genauere Charakterisierung eines IDS findet sich in Definition 1.

D

Definition 1: Intrusion Detection System nach Scarfone und Mell (2007)

Ein *Intrusion Detection System (IDS)* ist ein System, welches Aktivitäten auf Computern sowie im Netzwerk beobachtet und analysiert, um bösartige Handlungen oder Verletzungen von Richtlinien zu erkennen und entsprechende Vorkommnisse zu melden.

Nach dieser Definition leistet ein IDS also lediglich die Erkennung von Einbrüchen. Auf Basis dieser Erkennung kann dann eine manuelle oder automatische Reaktion erfolgen. Wird ein IDS um aktive Maßnahmen zur Verhinderung und Entfernung von Angriffen erweitert, spricht man auch von einem *Intrusion Detection and Prevention System (IDPS)*, wie es in Definition 2 charakterisiert wird.

D

Definition 2: Intrusion Detection and Prevention System

Ein *Intrusion Detection and Prevention System (IDPS)* ist ein Intrusion Detection System, welches darüber hinaus in der Lage ist, drohende Angriffe zu verhindern oder erkannte Eingriffe abzuwenden.

Neben den klassischen Einsatzbereichen zur eigentlichen Identifizierung von Angriffen und zur Reaktion auf diese Angriffe gibt es weitere Ziele, welche mit solchen Systemen verfolgt werden. Dazu zählt die Identifizierung von Verletzungen definierter Sicherheitsrichtlinien. Hierbei kann ein IDS eingesetzt werden, um beispielsweise Fehlkonfigurationen von Firewall-Systemen aufzudecken. Darüber hinaus ist die Dokumentation erfolgter Vorkommnisse ein wichtiger Beitrag zur Bewertung von IT-Risiken in Unternehmen. Die gesammelten Daten liefern hierbei detaillierte Informationen über die individuelle Bedrohungssituation – besonders im Hinblick auf die Häufigkeit und die Art von Angriffen. Außerdem kann das bloße Vorhandensein solcher Systeme bereits als Abschreckung für interne Angreifer dienen. So kann das Risiko, dass ein Mitarbeiter in einem Unternehmen unberechtigt auf vertrauliche Daten zugreift, schon allein durch die Kenntnis über die Existenz eines IDS, welches in der Lage ist solche Zugriffe zu melden, minimiert werden.

IDS werden nach ihrer Platzierung im überwachten System klassifiziert. Grundsätzlich lassen sich *hostbasierte* und *netzwerkbasierte* IDS unterscheiden. Der jeweilige Ort bestimmt sowohl die Art und den Umfang der Daten, welche zur Analyse gesammelt werden können, als auch die Möglichkeiten potenzieller Angreifer, die Erkennungsversuche wahrzunehmen, um entsprechend darauf reagieren zu können. Hostbasierte System zur Einbruchserkennung haben Zugriff auf alle Daten des Betriebssystems und der laufenden Anwendungen, können jedoch potenziell von Angreifern erkannt werden, während netzwerkbasierte Systeme lediglich die Kommunikation der Systemkomponenten auswerten können. Diese Auswertung ist allerdings für das Ziel der Beobachtung nur sehr schwer wahrzunehmen. In Kapitel 3 werden wir auf diese grundlegende Klassifikation vertieft eingehen sowie insbesondere *Hypervisor-basierte IDS* detaillierter betrachten.

Daneben ermöglicht die Funktionsweise von Einbruchserkennungssystemen eine weitere Klassifizierung. Hierbei beschränken wir uns auf signaturbasierte und anomaliebasierte Systeme. Während Systeme auf Basis von Signaturen die Ausführung auf Muster bekannter Angriffe hin überprüfen, definieren anomaliebasierte Systeme eine regelkonforme Ausführung von Applikationen und detektieren Abweichungen von diesem Standardverhalten. Erstere sind besonders einfach und effizient zu implementieren, haben allerdings keine Möglichkeit, neuartige Angriffe zu erkennen. Vertreter der zweiten Gruppe sind prinzipiell in der Lage, auch unbekannte Angriffe zu erkennen, sind allerdings auch deutlich aufwendiger in Einrichtung und Betrieb. Die genauen Details werden in Kapitel 3 ausgeführt.

Einbruchserkennung in Cloud-Umgebungen ist besonderen Herausforderungen ausgesetzt. Zum einen sind die eigenständige Einrichtung und der unabhängige Betrieb von Systemen zur Einbruchserkennung in einigen Cloud-Modellen – namentlich SaaS und zumeist auch PaaS – schon rein technisch unmöglich, da Cloud-Kunden keine Möglichkeit haben, eigene Software auf den gemieteten Cloud-Ressourcen zu installieren. Zum anderen unterliegen auch IaaS-Umgebungen hinsichtlich des Netzwerks zahlreichen Einschränkungen. So ist es in der Regel nicht möglich, auf Netzwerkswitches zuzugreifen und Ports zu spiegeln, um den Datenverkehr an ein netzwerkbasiertes Einbruchserkennungssystem zur Analyse umzuleiten. Genauso ist die Installation von speziellen Hardwaregeräten, so genannten Test Access Points (TAPs), welche zum gleichen Zweck zwischen die Netzwerkteilnehmer geschaltet werden, nicht möglich. So muss der Netzwerkverkehr mit Hilfe von virtueller Netzwerktechnologie über bestimmte Knoten geroutet werden, auf denen Einbruchserkennungssysteme dann die ausgetauschten Pakete abgreifen können. Bei Cloud-Anbietern finden sich zwar bereits erste SaaS-Lösungen, welche Einbruchserkennung als Dienst für gemietete Infrastrukturressourcen anbieten, gleichzeitig ist dieses Feld aber noch Gegenstand andauernder Forschung (Alharkan und Martin, 2012).

2.2 Honeypots

Werden Einbrüche erst erkannt, wenn der Angreifer bereits ein Produktivsystem kompromittiert hat, so ist in aller Regel bereits ein Schaden entstanden. Dennoch ist es in solchen Situationen besonders wichtig, genau festzustellen, wie der Angreifer die Systemgrenzen überwinden und welche Schwachstellen er ausnutzen konnte, um in Zukunft besser darauf reagieren zu können und Angriffsversuche bereits frühzeitig zu erkennen. Sinnvoller als diese Erkenntnisse am Produktivsystem zu gewinnen, welches möglicherweise kritische Dienste bereitstellt oder vertrauliche Daten verarbeitet, ist es, diese Informationen bereits im Vorhinein und in dedizierten Umgebungen in Erfahrung zu bringen, zumal die notwendige Analyse der Einbrüche oftmals viel Zeit beanspruchen kann und damit die Ausfallzeit der Dienste erhöhen würde. Zu diesem Zweck werden dedizierte Server eingesetzt, welche Produktivsysteme zwar imitieren, im Falle einer Übernahme aber keine vertraulichen Daten preisgeben oder Dienstausfälle riskieren. Diese Server werden aufgrund ihrer Reizwirkung auf Angreifer auch als Honeypots bezeichnet und können wie folgt definiert werden (Spitzner, 2003).

Definition 3: Honeypot

Ein Honeypot ist ein Informationssystem, welches Angreifern ein wertvolles Ziel vortäuscht, um detaillierte Informationen über deren Einbruchsversuche und unautorisierte Zugriffe zu gewinnen.

D

Eine Sammlung von mehreren Honeypots bezeichnet man auch als *Honeynet*.

D

Definition 4: Honeynet

Ein Honeynet ist ein Netzwerk von zwei oder mehreren Honeypots, welches Angreifern ein produktives Netzwerk mit einer Vielzahl von Diensten vortäuscht.

Honeypots unterscheiden sich vor allem darin wie sie mit den Angreifern interagieren. Dabei wird zwischen geringer, mittlerer und hoher Interaktion differenziert. Vertreter der ersten beiden Gruppen simulieren lediglich einzelne Aspekte von Diensten und Anwendungen, sodass auch nur bestimmte Aspekte des Angriffs in Erfahrung gebracht werden können, wohingegen Honeypots mit hoher Interaktion echte Systeme darstellen, welche wiederum auch in der Lage sind, Angreifer längerfristig zu beobachten, um die Ziele und Hintergründe komplexer Angriffe aufzudecken. Je mehr Informationen über einen Angreifer gesammelt werden können, desto riskanter ist normalerweise auch der Betrieb des Honeypots, da Angreifer möglicherweise aus den vorgesehenen Strukturen ausbrechen und den Honeypot in unvorhergesehener Weise für eigene Zwecke missbrauchen können. Eine detaillierte Betrachtung dieser und weitere Merkmale sind Gegenstand der Ausführungen in Kapitel 4.

3 Einbruchserkennung

Systeme zur Einbruchserkennung werden hauptsächlich auf der Basis von zwei Merkmalen klassifiziert. Die erste Dimension bildet den Ort, an dem die Erkennung erfolgt, während die zweite Dimension die Art der Erkennung darstellt. Die möglichen Orte lassen sich dabei grundsätzlich in Host und Netzwerk unterteilen. Die Art der Erkennung geschieht entweder auf Basis von Signaturen oder mittels Anomalieerkennung. Die einzelnen Ausprägungen weisen jeweils spezifische Vor- und Nachteile auf und werden in diesem Abschnitt genauer erläutert.

Host Intrusion Detection Systems (HIDS) werden direkt auf dem Zielhost ausgeführt und sind in der Lage sowohl Informationen des Betriebssystems als auch der einzelnen Anwendungen ausführlich zu analysieren. Die Tatsache, dass sich alle relevanten Informationen im direkten Zugriffsbereich des Einbruchserkennungssystems befinden, ermöglicht eine Erkennung beinahe in Echtzeit. Die Herausforderungen liegen dabei insbesondere in der Auswahl geeigneter Informationen und der Verhinderung von Manipulationen durch den Angreifer, welcher auf diese Weise versuchen wird, die Maßnahmen des Intrusion Detection Systems zu umgehen und eine Erkennung zu vermeiden.

Network Intrusion Detection Systems (NIDS) sind – wie sich aus dem Namen erschließen lässt – im Netzwerk platziert und überwachen den vorliegenden Kommunikationsverkehr. Dabei werden die im Netzwerk ausgetauschten Datenpakete aufgezeichnet und entsprechend analysiert. Daraus ergeben sich zwei Vorteile. Zum einen können Hosts ohne direkten Zugriff überwacht werden und zum anderen wird die Überwachung sowohl von den Hosts an sich als auch von potenziellen Angreifern nicht wahrgenommen. Dagegen ist die Menge an Informationen, die sich auf diesem Weg über einen Host ermitteln lassen, natürlich geringer, weil alle Informationen aus der Kommunikation entnommen werden müssen. Des Weiteren können verschlüsselte Kommunikationsinhalte in der Regel nicht analysiert werden.

Eine neuere Entwicklung stellen Hypervisor-basierte Systeme zur Einbruchserkennung dar, welche einzelne Vorteile der vorgenannten Systeme vereinen. Hierbei wird die privilegierte Rolle des Hypervisors ausgenutzt, um Informationen aus den Gastsystemen zu extrahieren. Der Hypervisor ermöglicht den Zugriff auf die Festplatten, den Arbeitsspeicher sowie die Netzwerkkommunikation. Folglich ist es einerseits möglich, Host-Informationen und anwendungsspezifische Informationen – unsichtbar für etwaige Angreifer – zu extrahieren. Auf der anderen Seite kann das Mitschneiden der Netzwerkkommunikation ohne den Einsatz spezieller Hardware erfolgen. Diese Flexibilität bringt allerdings ihre eigenen Schwierigkeiten mit sich. Die größte Herausforderung ist sicherlich die eigentliche Analyse, bei der die gewünschten Informationen aus Rohdaten und einfachen Speicherabbildern gewonnen werden müssen. Außerdem erfordert die Aufzeichnung der Rohdaten einen hohen Ressourceneinsatz, welche vom Hostsystem erbracht werden muss.

Bei Signaturen handelt es sich um Muster, die einen Angriffsvektor repräsentieren. Bei der signaturbasierten Einbruchserkennung wird die Systemausführung mit bekannten Signaturen verglichen, um Angriffe zu erkennen. Beispiele für solche Muster sind bestimmte Anhänge in eingehenden E-Mails, Prüfsummen von Dateien oder auch eine Reihe von System Calls in einer charakteristischen Abfolge. Der Vergleich von Systemaspekten mit bereits gelernten Mustern stellt das einfachste Verfahren dar, sodass der Vergleich in aller Regel performant erfolgen kann. Jedoch limitiert diese Einfachheit das Verfahren gleichzeitig. Komplexere Angriffsmuster und insbesondere neue Angriffe können mit diesem Ansatz nicht erkannt werden. Außerdem sind sich Angreifer der Funktionsweise dieser Verfahren natürlich bewusst und reagieren entsprechend darauf. So führen beispielsweise schon minimale Änderungen an einer Datei zu einer neuen Prüfsumme, welche wiederum von IDS nicht erkannt wird und erst gelernt werden muss.

Verhaltensbasierte oder anomaliebasierte Einbruchserkennung analysiert das Verhalten von Anwendungen und Prozessen zur Laufzeit. Dabei wird zuerst ein Anwendungsprofil definiert, das eine unbedenkliche und konforme Systemausführung widerspiegelt. Zumeist werden solche Profile in einer Trainingsphase aus der Beobachtung und Analyse von Anwendungen im ungestörten Betrieb abgeleitet. Die Profildefinitionen könnten beispielsweise enthalten, dass die Anzahl der Wiederholungen von Anmeldeversuchen eines Benutzers über einen gewissen Zeitraum normalerweise unter einer oberen Schranke bleibt oder dass als Folge auf bestimmte Clientanfragen nur bestimmte Systemaufrufe erfolgen. Alle Systemausführungen, welche zu einem gewissen Grad von diesem Profil abweichen, werden als Einbruchsversuch gemeldet. Da sich das Verhalten und die Zusammensetzung von Systemen über die Zeit verändert, ist es notwendig die Profile stetig an die jeweiligen Anwendungen anzupassen. Die hohe Zahl und der hohe Betreuungsaufwand manifestieren damit auch die offensichtlichen Nachteile dieses Ansatzes. Auf der anderen Seite liefern gut konfigurierte anomaliebasierte IDS ein hohes Potenzial in der Erkennung neuer und komplexer Anfragen.

3.1 Hostbasierte Systeme zur Einbruchserkennung

Hostbasierte Systeme zur Einbruchserkennung analysieren die auf dem Host zur Verfügung stehenden Informationen. Dies erfolgt beispielsweise durch Log-Analyse, Datenintegritätsüberwachung, Registry-Verifikation oder Rootkit-Erkennung. HIDS können sich einer Vielzahl von Daten bedienen, gleichzeitig ist die Überwachung für Angreifer leicht wahrzunehmen. Im Folgenden werden die einzelnen Konzepte jeweils beschrieben, bevor mit OSSEC eine quelloffene Beispielanwendung vorgestellt wird, welche alle der vorgenannten Methoden anbietet.

Datenintegritätsüberwachung

Die Datenintegritätsüberwachung erlaubt es, Änderungen an Dateien und im Dateisystem festzustellen (Kim und Spafford, 1994). Hierzu wird zuerst der zu überwachende Bereich des Dateisystems gescannt und Informationen zu jeder Datei in einem Index gespeichert. Die gespeicherten Informationen umfassen in aller Regel den Pfad, bestimmte Attribute sowie eine kryptographische Prüfsumme des Inhalts. Zur Überprüfung des Dateisystems oder eines Bereichs davon werden dieselben Informationen noch einmal gesammelt und mit den gespeicherten Informationen verglichen. So ist es möglich, neue und gelöschte Dateien zu erkennen sowie Änderungen an bestehen Dateien festzustellen.

Anhand von vorher festgelegten Regeln kann genau bestimmt werden, welche Analyseresultate gemeldet werden sollen. Auf diese Weise können bestimmte Dateien vollständig oder teilweise ignoriert werden, um False-Positive-Ereignisse zu vermeiden. Änderungen an einer Log-Datei sind beispielsweise völlig legitim, weil diese Datei kontinuierlich beschrieben wird. Genauso dürfen in einem temporären Verzeichnis viele neue Dateien angelegt und alte Dateien gelöscht werden. Verdächtig ist es hingegen, wenn sich der Besitzer einer solchen Log-Datei ändert oder eine Konfigurationsdatei beschrieben wird. Bei Verletzung der definierten Regeln kann eine Alarmierung erfolgen und ein Analysebericht an die zuständigen Administratoren gesendet werden.

Rootkit-Erkennung

Unter dem Begriff Rootkits ist eine Gruppe von Schadsoftware zusammengefasst, welche versuchen, sich auf den befallenen Host bestmöglich vor einer Entdeckung zu verstecken und gleichzeitig Hintertüren zur Fernsteuerung offen halten (Arnold, 2011; Joy et al., 2011). Je länger solche Rootkits unentdeckt bleiben, desto mehr Schaden können die Angreifer mit ihnen anrichten. Die Erkennung von Rootkits kann auf verschiedene Weise erfolgen. Die möglichen Verfahren werden im Folgenden kurz erläutert. Tools zur Erkennung nutzen zumeist eine Kombination aus den verschiedenen Methoden.

Eine Möglichkeit stellen signaturbasierte Verfahren dar. Hierbei werden Prüfsummen von signifikanten Bereichen von Schadsoftware oder andere Attribute und Eigenschaften von bekannter Schadsoftware gespeichert. Die Dateien des Dateisystems können anschließend mit diesen Signaturen verglichen werden, um Schadsoftware zu erkennen. Diese Methode ist besonders effizient und hat eine hohe Erfolgsrate, kann aber keine neuen Angriffe erkennen.

Auch die weiter oben bereits vorgestellte Integritätsüberwachung kann zur Erkennung von Rootkits eingesetzt werden. Das liegt darin begründet, dass Rootkits viele Dateien ändern müssen, um ihre Existenz zu verschleiern. Das Wissen darüber, welche Dateien ein bestimmtes Rootkit ändert, kann umgekehrt auch zu dessen Detektion verwendet werden.

Eine weitere Möglichkeit zur Rootkit-Erkennung bietet die Cross-View-Analyse. Die generelle Idee hierbei ist der Vergleich zwischen der Low-Level-Sicht und der High-Level-Sicht auf ein System. Wenn sich bestimmte Faktoren, welche in der unteren Schicht anwesend sind, nicht in der oberen Schicht wiederfinden, kann das auf ein Rootkit hindeuten, dass zwischen den beiden Schichten agiert. In der Regel nisten sich Rootkits in bestimmten APIs ein und geben verfälschte Antworten zurück. Ein Beispiel für eine Cross-View-Analyse kann deshalb der Vergleich der Zugriffs-API auf ein Dateisystem mit den Rohdaten des Speichermediums sein. Wenn bestimmte Dateien laut API nicht vorhanden sind, diese sich auf den

Rohdaten aber nachweisen lassen, ist die Infiltrierung der API durch ein Rootkit sehr wahrscheinlich.

Des Weiteren erfolgt die Erkennung von Rootkit durch die Anwendung von Heuristiken. Dazu wird im Vorhinein ein normales Systemverhalten definiert. Anschließend kann die aktuelle Systemausführung kontinuierlich gegen dieses vorher definierte Verhalten verglichen werden. Auf abweichendes Verhalten kann dann entsprechend reagiert werden. Auf diese Weise könnte ein Prozess, welcher Kerneldatenstrukturen verändert, schnell erkannt und isoliert oder terminiert werden. Mit diesem Verfahren ist es folglich auch möglich, bisher unbekannte Malware zu erkennen, wenngleich auch eine hohe False-Positive-Rate den Betrieb von Diensten negativ beeinflussen kann.

Registry-Verifikation

Unter Windows ist die sogenannte Registry die zentrale Systemkonfigurationsdatenbank und stellt damit ein wertvolles Ziel für Angreifer dar. Die Überprüfung der Integrität ist somit eine vielversprechende Möglichkeit zur Erkennung von Einbrüchen. Dazu werden die Werte von bestimmten Schlüsseln aufgezeichnet und periodisch mit den aktuellen Werten im System verglichen. Abweichungen von den gespeicherten Werten lösen eine Alarmierung aus. Daneben besteht auch die Möglichkeit zur Anwendung der – weiter oben bereits vorgestellten – Cross-View-Analyse. Werte aus der Registry können einerseits durch eine Windows-API abgefragt werden oder aus den zur Registry gehörigen Dateien rekonstruiert werden. Wenn API-Rückgabewerte nicht mit den in den Dateien vorhandenen Werten übereinstimmen, können HIDS Alarm schlagen.

Log-Analyse

Das Betriebssystem sowie die laufenden Diensten und Anwendungen produzieren stetig Log-Dateien (Lin et al., 2010). Bei einem hohen Detailgrad der eingesetzten Logger lässt sich das Verhalten von System und Benutzern genau rekonstruieren. Die Log-Analyse nutzt diese Tatsache für die Einbruchserkennung und bedient sich dazu der gesammelten Informationen. Für eine automatisierte Analyse der Log-Dateien sind mehrere Schritte notwendig. Zuerst einmal müssen die Logs eingesammelt werden. Danach erfolgt das Parsen der Dateien, oft verbunden mit der Konversion in eine einheitliche Repräsentation. Im Anschluss daran kann die eigentliche Auswertung erfolgen. Hierzu werden die Log-Dateien gegebenenfalls korreliert und nach auffälligen Mustern und Einträgen abgesucht. Verdächtige Vorkommnisse werden als Alarm gemeldet. Die Log-Analyse ist ein effektiver Weg, Einbrüche aufzuspüren, solange der Angreifer seine Spuren nicht gekonnt verschleiert. Außerdem muss für jeden kritischen Dienst, der betrieben wird, auch eine entsprechende Komponente vorhanden sein, welche das dienstspezifische Parsen und die entsprechende Analyse übernehmen kann.

Beispielanwendung OSSEC

Mit OSSEC steht ein quelloffenes hostbasiertes System zur Einbruchserkennung zur Verfügung, welches alle der vorgenannten Erkennungsmethoden mit sich bringt (Lhotsky, 2013). OSSEC ist dabei für alle gängigen Plattformen verfügbar – namentlich Linux, Solaris, AIX, HP-UX, Mac OS und Windows – und verfügt über ein weitreichendes und anpassbares Reaktionssystem, welches automatisch aktiviert wird, sobald bestimmte Trigger ausgelöst werden.

Abbildung 1 zeigt die Komponenten von OSSEC, die bei einer verteilten Installation zum Einsatz kommen. Selbstverständlich kann OSSEC auch selbstständig auf

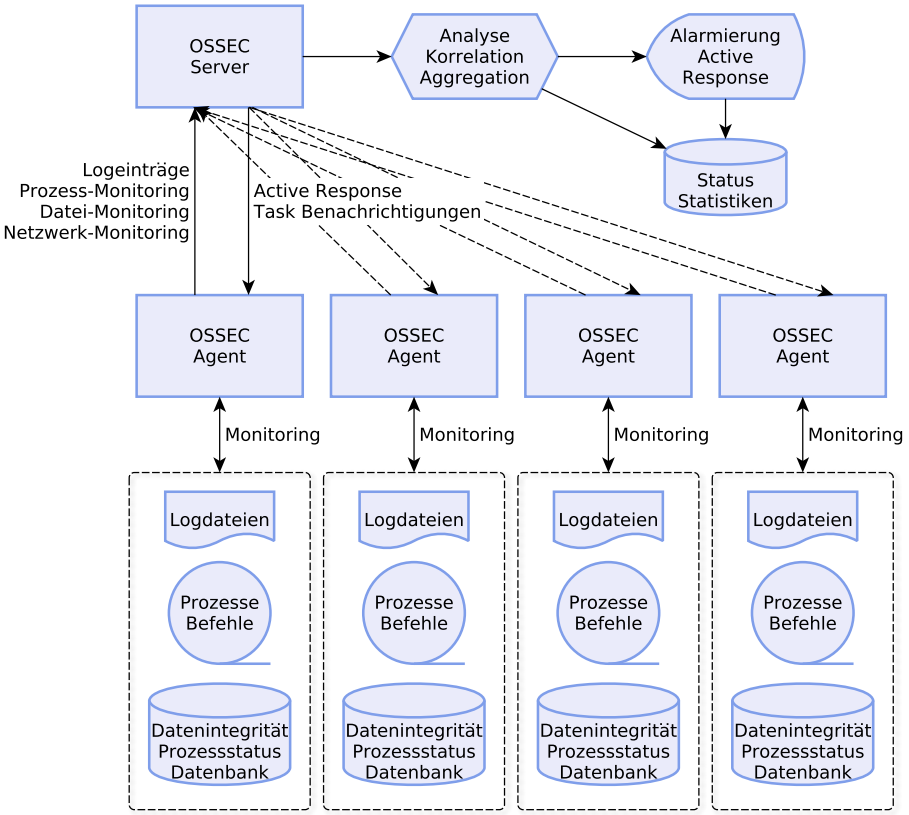


Abb. 1: Komponenten des HIDS OSSEC: Ein zentraler Server koordiniert die Auswertung von Daten aus dezentralen Agenten (Lhotsky, 2013).

einzelnen Rechnern betrieben werden. Im Normalfall gibt es aber einen zentralen Verwaltungs- und Analyseserver, welcher als OSSEC-Server oder einfach Manager bezeichnet wird und von einer Vielzahl von Agenten mit Monitoring-Daten gespeist wird. Bei den Agenten handelt es sich um leichtgewichtige Dienste, die in Echtzeit oder periodisch Log-Dateien, das Dateisystem sowie Prozessdaten und Kommandobefehle auswerten beziehungsweise aufzeichnen und diese Informationen – komprimiert und verschlüsselt – für die weitere Analyse an den Manager schicken. So kann gewährleistet werden, dass der Betrieb der überwachten Hosts so wenig wie möglich beeinträchtigt wird.

Die Erzeugung von Alarmen erfolgt auf der Basis von Regeln. Für bekannte Dienste und Anwendungen bringt OSSEC bereits ein weitreichendes Regelwerk mit. Zusätzlich können mit internen Werkzeugen weitere Regeln definiert werden. Eine Regel besteht immer aus einer ID, einem Level, einer textuellen Beschreibung sowie der eigentlichen Regelbedingung. Das Level gibt an, wie schwerwiegend eine Verletzung dieser Regel eingeschätzt wird. Die Beschreibung findet sich auch in den Log-Dateien sowie Berichten wieder und sollte die Regel deshalb prägnant beschreiben. Die Bedingungen geben Muster an, nach denen in Logeinträgen oder Befehlsausgaben gesucht wird.

Verletzungen dieser Bedingungen und Regeln lösen Alarme des entsprechenden Levels aus. In der Konfiguration von OSSEC kann definiert werden, was bei der Auslösung eines Alarms geschehen soll. So kann beispielsweise festgelegt werden, dass grundsätzlich alle Ereignisse in einer Log-Datei dokumentiert werden und Ereignisse eines bestimmten Levels eine E-Mail-Benachrichtigung auslösen. Mit Hilfe der sogenannten Active-Response-Funktionalität ist es darüber hinaus möglich, direkt auf bestimmte Ereignisse zu reagieren. So können auf überwachten Hosts unter anderem neue Firewall-Regeln hinzugefügt oder Dienstkonfigurationen in Echtzeit angepasst werden.

3.2 Netzwerkbasierte Systeme zur Einbruchserkennung

Bei der nächsten Gruppe von Einbruchserkennungssystemen handelt es sich um Systeme die in einem Netzwerk platziert werden und die darin befindlichen Hosts überwachen (Northcutt und Novak, 2002). Dazu schneiden sie den gesamten zugänglichen Netzwerkverkehr mit und führen stetig verschiedene Analysen auf den aufgezeichneten Daten aus, um Angriffe wie Denial of Service (DoS) Attacken, Portscans und Einbruchsversuche zu erkennen. Für die überwachten Rechner erfolgt die Überwachung in aller Regel transparent. Dieser Abschnitt erklärt zuerst die Verfahren zur Datenerfassung sowie Datenanalyse, bevor er mit Snort einen quelloffenen und bekannten Vertreter aus der Gruppe der netzwerkbasierten Einbruchserkennungssysteme vorstellt.

Datenerfassung

Für die Datenerfassung werden Paketsniffer eingesetzt. Diese Softwarekomponenten können hostbasiert eingesetzt werden, d. h. auf dem Zielrechner wird der Netzwerkverkehr dieses Rechners mitgeschnitten. Diese Variante bringt aber die ganzen Nachteile eines hostbasierten IDS mit sich: Die Aufzeichnung durch das Hostsystem kann erkannt und die aufgezeichneten Daten durch Angreifer auf dem Hostsystem manipuliert werden. Daher ist diese einfache Variante eher für Debugging und Fehlerdiagnose und weniger für Einbruchserkennung geeignet. Für NIDS ist es dagegen üblich, mit einem separaten Diagnosesystem den kompletten Netzwerkverkehr eines Subnetzes mitzuschneiden.

Voraussetzung hierfür ist zum einen, dass die Netzwerkschnittstelle des Diagnosesystems in den so genannten „promiscuous mode“ versetzt wird, in welchem alle Netzwerkpakete an den Diagnosehost weitergereicht werden. Zum anderen müssen alle Netzwerkpakete auf physischer Ebene zum Diagnosehost gelangen. Bei heute nicht mehr gebräuchlichen Ethernet-Architekturen in Bus-Struktur (mit Koaxial-Ethernet oder Hubs) stellte dieses Unterfangen keine besondere Herausforderung dar. Heutige Netzwerke werden allerdings von Switches betrieben, welche eingehende Nachrichtenpakete direkt an die jeweiligen Empfänger weitergeben. Erfreulicherweise verfügen die meisten Switches über einen speziellen Monitor Port – den sogenannten Switch Port Analyzer oder einfach SPAN-Port. An diesem Port wird der gesamte Verkehr des Switches zusätzlich ausgeleitet und eine weiterführende Analyse ermöglicht. Falls sich in einem Subnetz mehrere Switches befinden müssen i.d.R. an allen entsprechende Monitoring-Hosts betrieben werden, welche den aufgezeichneten Netzwerkverkehr an einen zentralen Analysehost weiterleiten.

Mit dem Einzug von Virtualisierungstechnik bzw. Software Defined Networking (SDN) in den Bereich der Kommunikationsnetze ergeben sich auch für das Monitoring neue Möglichkeiten. Durch die Verlagerung von Switch-Funktionalität in Software – wie beispielsweise bei *Open VSwitch*[1] – muss der Anschluss eines Paketsniffers nicht mehr physisch an einen speziellen Switch-Port erfolgen, sondern kann sehr flexibel virtuell durch eine geeignete Konfiguration realisiert werden. In Cloud-Umgebungen werden virtuelle Maschinen im einfachsten Fall zumindest innerhalb eines physischen Hosts, in vielen Fällen aber auch hostübergreifend mit virtuellen Netzen verbunden. Grundsätzlich stellt dies eine hervorragende Grundlage für NIDS dar. Allerdings ist bei öffentlichen Cloud-Infrastrukturen zu klären, ob und wie der jeweilige Cloud-Betreiber eine geeignete Konfiguration des virtuellen Netzes ermöglicht.

[1] `http://openvswitch.org` [abgerufen am 2017-04-01]

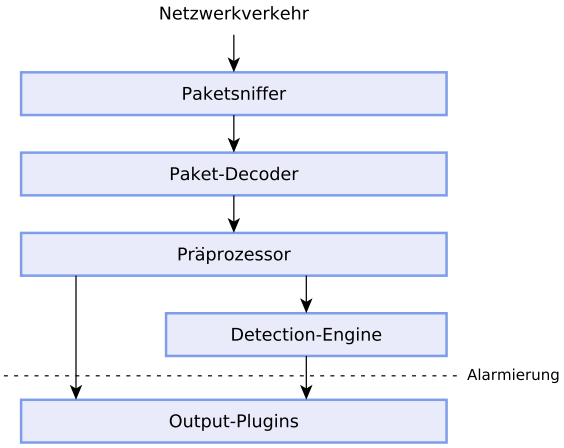


Abb. 2: Architektur des Snort-NIDS (Bechtold und Heinlein, 2004)

Datenanalyse

Die so gewonnenen Daten werden zur weiteren Verarbeitung in der Regel vorverarbeitet. Das bedeutet, dass die Rohdaten gefiltert und normalisiert werden. Dabei werden beispielsweise IP-Datagramme aus Ethernet-Frames oder FTP-Pakete aus einer vorher dekodierten TCP-Sitzung gewonnen. Diese Daten werden dann in interne Datenstrukturen umgewandelt und mit vordefinierten Regeln verglichen sowie entsprechend ausgewertet. Hierbei kommen wieder zwei bereits bekannte Konzepte zum Einsatz: Die Erkennung auf Basis von Mustern und Signaturen sowie eine verhaltensbasierte Erkennung. Die signaturbasierte Erkennung schlägt Alarm, wenn bestimmte Muster in der Kommunikation aufgefunden werden – wenn also beispielsweise in einem HTTP-Paket bekannter Exploitcode erkannt wird. Dagegen arbeitet die verhaltensbasierte Erkennung mit dem Vergleich des Ist-Zustands mit dem Normalzustand und könnte Alarmierungen auslösen, falls tagsüber plötzlich viel Datenverkehr mit einem Backup-Server auftritt, der normalerweise nur nachts genutzt wird.

Beispielanwendung Snort

Die grundlegenden Konzepte von NIDS sollen in diesen Abschnitt anhand einer Beispielanwendung vertieft werden. Zu diesem Zweck wurde Snort ausgewählt, da Snort ein bewährtes NIDS darstellt, welches frei und quelloffen für die gängigen Plattformen verfügbar ist (Bechtold und Heinlein, 2004). Die erste Version von Snort geht zurück in das Jahr 1998, aber auch heute wird das System noch aktiv weiterentwickelt. Im nachstehenden Text erfolgen eine Erläuterung der beteiligten Komponenten und der generellen Funktionsweise sowie ein Einblick in das Regelsystem von Snort.

Abbildung 2 zeigt die einzelnen Komponenten, welche an der Bearbeitung eines Datenstroms beteiligt sind. Im Detail handelt es sich dabei um:

Paketsniffer Der Paketsniffer zeichnet den Netzwerkverkehr aus dem überwachten Netzwerk als Rohdaten mit. Snort setzt dabei auf die C-Bibliothek *libpcap*, welche ebenfalls auf allen gängigen Plattformen verfügbar ist.

Paket-Decoder Der Decoder operiert auf den Eingabedaten des Paketsniffers und erzeugt aus den Rohdaten die einzelnen Daten der verschiedenen OSI-Schichten.

Präprozessor Präprozessoren sind austauschbare Komponenten, welche einfach hinzugefügt oder entfernt werden können und so die Erweiterbarkeit von

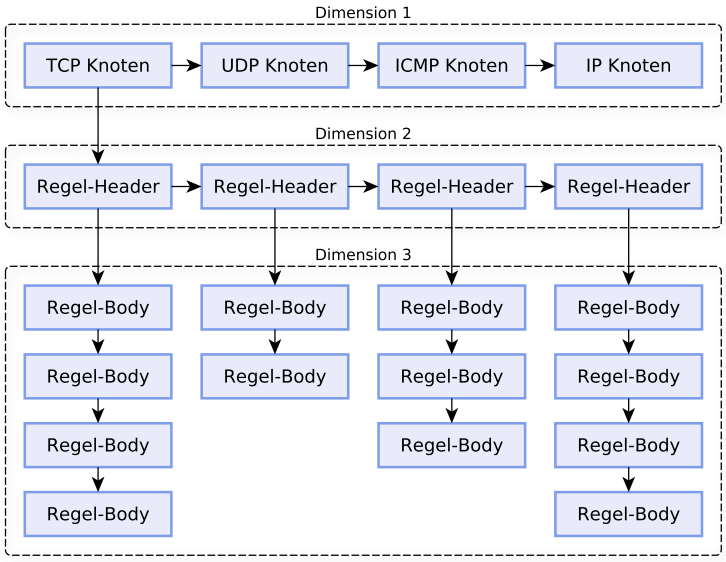


Abb. 3: Snort Regelwerk (Bechtold und Heinlein, 2004)

Snort sicherstellen. Neben den bereits vorhandenen Modulen können so auf einfache Weise eigene Entwicklungen integriert werden. Präprozessoren können aus ungeordneten Paketen einen konsistenten Datenstrom erzeugen und geben diesen dann an die Detection Engine weiter. Dazu führen sie beispielsweise Ethernet-Pakete zusammen oder sortieren die IP-Pakete. Darüber hinaus gibt es Präprozessoren, die bereits erste Überprüfungen ausführen. Beispielsweise gibt es bei Snort Komponenten, die ARP-Spoofing-Angriffe oder IP-Fragmentation-Attacken erkennen können. Im Falle einer Detektion erfolgt die direkte Meldung an die Output-Plugins.

Detection-Engine Die Detection-Engine liest beim Programmstart die definierten Regeln ein und speichert sie in einer internen Repräsentation. Die Reihenfolge der Definition ist bei der Überprüfung entscheidend. Zur Laufzeit durchlaufen die Pakete das komplette Regelwerk wie in Abbildung 3 dargestellt. Zuerst erfolgt eine Protokoll-Filterung bevor der Regel-Header überprüft wird. Dabei wird geprüft, ob das exakte Protokoll, Quell- und Zieladressen sowie Quell-und Zielport der Signatur mit den Attributen des vorliegenden Paketes übereinstimmt. Wenn dies der Fall ist, werden die zur Regel gehörenden Bedingungen überprüft. Falls eine Bedingung zutrifft, wird die in der Regel definierte Aktion ausgeführt. Das kann neben der Alarmierung auch das Loggen, Ignorieren und Verwerfen des Pakets oder die Weitergabe an eine weitere Regel sein. Wenn keine Bedingung zutrifft, wird das Paket endgültig verworfen.

Output-Plugins Die Output-Plugins verarbeiten die Alarmierungen von Präprozessoren und der Detection-Engine. Es gibt eine Vielzahl solcher Plugins, die es zum Beispiel erlauben Alarmierungen per E-Mail beziehungsweise SMS zu versenden oder in eine Datenbank beziehungsweise Log-Datei zu schreiben.

Ankommende Datenströme werden also zuerst vom Paketsniffer aufgezeichnet und an den Paket-Decoder weitergegeben. Dieser überführt die Rohdaten in Pakete, welche dann die Präprozessoren durchlaufen. Dort werden die Pakete normalisiert und bereits erstmalig überprüft. Die normalisierten Pakete werden anschließend in die Detection-Engine gespeist, welche weitere – gegebenenfalls komplexere – Überprüfungen ausführt. Falls die Verifikationen anschlagen, wird das verdächtige Verhalten mit Hilfe der Output-Plugins protokolliert oder berichtet.

Das nachfolgende Beispiel erklärt den generellen Aufbau von Snort-Regeln anhand einer Regel, welche in der Lage ist SYN-FIN-Portscans zu erkennen.

B

Beispiel 1: Snort-Regel

```
alert tcp !192.168.0.0/16 any -> any 1:1024 \
(flags: SF; msg: "possible SYN FIN portscan detected";)
```

Der Header der Regel ist wie folgt aufgebaut:

Aktion `alert`: Erzeugt eine Meldung. Eine andere mögliche Aktion ist z. B. „log“, die das Paket protokolliert.

Protokoll `tcp`: Die Regel betrifft TCP-Pakete. Andere mögliche Protokolle sind UDP, ICMP und IP.

Quelladresse `!192.168.0.0/16`: Die Regel betrifft Pakete, die nicht aus dem Subnetz 192.168.0.0/16 gesendet wurden. Das Ausrufezeichen negiert den Ausdruck. Es können einzelne Adressen, mehrere Adressen und Subnetze oder das Schlüsselwort „any“ für beliebige Quelladressen angegeben werden.

Quellport `any`: Die Regel betrifft Pakete, die von einem beliebigen Port gesendet wurden. Es können auch einzelne oder mehrere Ports und Portbereiche angegeben werden.

Richtung `->`: Die Regel betrifft nur Pakete, die in Richtung von der Quelle zum Ziel gesendet werden. Mit <> kann angegeben werden, dass die Regel zusätzlich auch Pakete vom Ziel zur Quelle betrifft.

Zieladresse `any`: Die Regel betrifft Pakete mit beliebiger Zieladresse. Die möglichen Werte entsprechen denen der Quelladresse.

Zielport `1:1024`: Die Regel betrifft Pakete, die an die Ports 1 bis 1024 gesendet werden. Die möglichen Werte für den Zielport entsprechen denen des Quellports.

Der Body der Regel enthält die folgenden Elemente:

`flags: SF`: Die Regel betrifft Pakete bei denen die TCP-Flags SYN und FIN gesetzt sind.

`msg: "..."`: Gibt den Text der alert-Meldung an.

Im Body kann die Regel auch z. B. mit `content` Pakete anhand des Inhalts auswählen.

Für eine ausführliche Erläuterung der Möglichkeiten und der Syntax von Snort-Regeln verweisen wir an dieser Stelle auf das umfangreiche Snort-Manual (The Snort Project, 2016), auf Bechtold und Heinlein (2004) sowie die zusätzlichen praktischen Übungen zu Snort.

3.3 Hypervisor-basierte Systeme zur Einbruchserkennung

Dem Vorteil von netzwerkbasierten Einbruchserkennungssystemen – der Isolation des IDS gegenüber potenziellen Angreifern – steht der Nachteil gegenüber, dass ein NIDS nicht die Vorgänge und Abläufe innerhalb eines Hosts offenlegen kann. Die dazu fähigen HIDS sind wiederum einem hohen Risiko der Manipulation durch Angreifer ausgesetzt. Um diese Lücken zu schließen und die Vorteile beider

Ansätze zusammenzubringen, werden in der jüngeren Zeit Hypervisor-basierte Systeme eingesetzt und aktiv erforscht (Garfinkel und Rosenblum, 2003). Diese Systeme basieren auf dem Konzept der Virtual Machine Introspection, welches im Folgenden zuerst erklärt wird, bevor darauf basierende Methoden zur Einbruchserkennung erläutert und mit Livewire ein bestehender Prototyp näher vorgestellt wird.

Virtual Machine Introspection (VMI)

VMI beschreibt eine Technik, bei der die Software, welche innerhalb einer virtuellen Maschine läuft, von außen inspiziert wird. Diese Technik wird durch den privilegierten Hypervisor ermöglicht, welcher zwischen dem Intrusion Detection System (Analysesystem) und der zu analysierenden Maschine (Zielsystem) vermittelt.

Der Hypervisor bietet *Isolation* zwischen virtuellen Maschinen, und damit eine Trennung zwischen Analyse- und Zielsystem, welche die Wahrnehmung der Analyse durch das Zielsystem verhindert. Da der privilegierte Hypervisor vollen Zugriff auf die Ressourcen hat, die er den virtuellen Maschinen bereitstellt, hat er die Möglichkeit der *Inspektion* und kann dadurch einem Einbruchserkennungssystem Daten zum internen Zustand des Zielsystems, wie z. B. CPU-Register, Hauptspeicherinhalte und I/O-Geräte, liefern. Darüber hinaus hat der Hypervisor die Fähigkeit der *Interposition* (Zwischenschaltung), kann also Aktionen einer virtuellen Maschine verhindern, bevor diese auf die physische Hardware wirken, und ermöglicht damit auch die Realisierung eines IDPS zur Verhinderung von Angriffen.

Bei der Inspektion von virtuellen Maschinen kann architektonisch zwischen aktiver und passiver VMI unterschieden werden. Die Unterscheidung richtet sich nach Art und Zeitpunkt der Extraktion der Daten durch den Hypervisor. Bei passiver VMI stellt der Hypervisor eine Schnittstelle bereit, über welche Daten einmalig oder periodisch abgefragt werden (Poll-Modell) und zur weiteren Analyse verarbeitet werden können. Bei der aktiven VMI sendet der Hypervisor die Daten aktiv an das Analysesystem (Push-Modell), wenn bestimmte Bedingungen erfüllt oder vorher definierte Ereignisse eingetreten sind. Beispiele für solche Ereignisse sind der Zugriff auf eine bestimmte Speicherseite oder die Ausführung einer bestimmten privilegierten CPU-Instruktion. Die Menge der erfassten Daten hat direkten Einfluss auf die Performance des Zielsystems und muss dementsprechend reguliert werden, um kritischen Geschwindigkeitsverluste zu vermeiden.

Eine große Herausforderung bei VMI ist die „semantische Lücke“ (semantic gap) zwischen den erfassten Rohdaten und der eigentlichen Struktur und Bedeutung dieser Daten. Es erfordert einen hohen Aufwand, um im Analysesystem aus den extrahierten Bits und Bytes wieder eindeutige Informationen und Datenstrukturen herzustellen.

Nach Jain et al. (2014) kann die semantische Lücke noch weiter unterteilt werden in die schwache sowie die starke semantische Lücke. Während bei der schwachen semantischen Lücke davon ausgegangen wird, dass potenzielle Angreifer keine aktiven Gegenmaßnahmen gegen die Beobachtung einleiten, ist bei der starken semantische Lücke genau dies der Fall. So könnte ein Angreifer versuchen bestimmte Datenstrukturen vor der Nutzung für bösartige Aktionen an andere Positionen im Speicher zu verlagern. Das Analysesystem, welches immer noch die eigentlichen Speicherbereiche überwacht, würde so die schädlichen Aktionen nicht nachvollziehen können. Auch wenn das die Herausforderungen zur Nutzung von VMI zur Einbruchserkennung weiter erhöht, bietet diese Technik aufgrund der Menge der verfügbaren Daten und der Flexibilität bei der Interaktion großes Potenzial.

Funktionsweise

Eine zentrale Funktion von VMI ist der Zugriff auf den Hauptspeicher des Zielsystems. Bibliotheken wie LibVMI[2] verwenden hierfür Schnittstellen des Hypervisors zur Introspektion und stellen darauf aufbauend Mechanismen bereit, um die Abbildung von virtuellen auf physische Adressen innerhalb des Adressraums der virtuellen Maschine des Zielsystems durchzuführen.

Eine Untersuchung des Hauptspeicherinhalts kann zum einen manuell bzw. interaktiv erfolgen. Hierzu bieten sich Werkzeuge wie sie auch aus dem Bereich der forensischen Untersuchung von Hauptspeicher-Abbildern üblich sind an. Beispielsweise kann das verbreitete Werkzeug *Volatility*[3] auch verwendet werden, um auf einer laufenden virtuellen Maschine eine Live-Analyse des Hauptspeichers durchzuführen. Darüber hinaus können alle Werkzeuge, die als hostbasierte Systeme den Hauptspeicher analysieren, mittels VMI implementiert werden. Beispielsweise kann ein Virusscanner auch per VMI nach Signaturen von Malware im Speicher suchen.

Der große Vorteil von VMI ist hierbei die Isolation vom Zielsystem, d. h. ein Angreifer, der das Zielsystem kontrolliert, kann nicht (bzw. kaum) erkennen, dass er per VMI analysiert wird, und er kann das Analysesystem nicht gezielt deaktivieren oder manipulieren. Ein potentieller Nachteil ist, dass im Zielsystem vorhandene APIs des Betriebssystems zur Gewinnung von Informationen per VMI nicht zur Verfügung stehen. Stattdessen müssen diese Informationen aus den Rohdaten des Hauptspeicherinhalts rekonstruiert werden. Dieser Nachteil kann aber zugleich einen Vorteil darstellen: Während die APIs im Zielsystem durch einen Angreifer manipuliert werden können und entsprechend falsche Informationen liefern können, werden die Daten mittels VMI direkt aus Datenstrukturen im Speicher und damit unabhängig von den potentiell manipulierten APIs ermittelt. Somit lässt sich beispielsweise die Kommunikation einer Botnet-Malware erkennen, selbst wenn die Malware die entsprechenden API-Funktionen des Betriebssystems, mit denen die aktiven Kommunikationsverbindungen abgefragt werden können, manipuliert hat.

Darüber hinaus lassen sich durch aktive VMI auch ereignisgesteuert bestimmte Aktionen des Zielsystems überwachen. Beispielsweise ist es dem Hypervisor möglich, Schreibzugriffe auf den Speicher des Betriebssystems des Gastsystems abzufangen und ein VMI-basiertes IDS darüber zu informieren. Dieses kann prüfen, ob dieser Schreibzugriff legitim ist oder möglicherweise auf einen Manipulationsversuch von Malware zurückzuführen ist. Auch Systemaufrufe *(System Calls)* von Anwendungen des Zielsystems lassen sich so überwachen und z. B. durch eine Anomalieerkennung die Aktivitäten von Schadsoftware erkennen.

Beispielanwendung Livewire

Exemplarisch wollen wir an dieser Stelle die Möglichkeiten von VMI zur Einbruchserkennung anhand des Beispiels *Livewire* (Garfinkel und Rosenblum, 2003) näher betrachten. Livewire war eines der ersten Prototyp-Systeme, die den Nutzen von VMI zur Einbruchserkennung demonstriert hat.

Der Artikel von Garfinkel und Rosenblum (2003) bietet einen guten Überblick über die vielfältigen Möglichkeiten, VMI einzusetzen. Zunächst betrachten wir einige Techniken, die auf passivem VMI („polling“) beruhen.

[2] `http://libvmi.com/` [abgerufen am 2017-04-01]
[3] `http://www.volatilityfoundation.org` [abgerufen am 2017-04-01]

- *Lügendetektor (lie detector):* Wie bereits in Abschnitt 3.3 beschrieben manipuliert Malware oft Betriebssystems-Schnittstellen, um die Malware-Aktivitäten zu verstecken. VMI-basierte Techniken lassen sich nicht nur einsetzen, um die Informationen unter Umgehung der manipulierten APIs direkt zu extrahieren. Ein direkter Vergleich der Informationen, die mittels VMI und mittels API gewonnen wurden, liefert auch unmittelbar den Hinweis, dass eine Manipulation stattgefunden hat.
- *Anwendungs-Integritäts-Erkennung:* Der Zugriff auf den Hauptspeicher mittels VMI erlaubt es auch zu prüfen, ob am Anwendungscode im Hauptspeicher Manipulationen stattgefunden haben. Zwar ist diese Methode nur eingeschränkt einsetzbar, wenn Binärcode erst zur Laufzeit (z. B. durch einen Java-JIT-Compiler) generiert wird. Wird aber Binärcode direkt beim Start einer Anwendung geladen, so lässt sich z. B. über Prüfsummen über den Hauptspeicher erkennen, wenn dieser Binärcode später verändert wurde.
- *Signaturerkennung:* Wie bereits im vorangegangenen Abschnitt kurz angesprochen lässt sich der Zugriff auf den Hauptspeicher dazu nutzen, diesen auf bekannte Signaturen von Schadsoftware zu durchsuchen.
- *Raw-Socket-Erkennung:* Bei „Raw Sockets“ handelt es sich um eine Betriebssystemfunktionalität des Zielsystems, mit dem eine Anwendung in die Lage versetzt wird, auf unterster Protokollebene Datenpakete zu versenden. Gewöhnliche Anwendungen haben in der Regel keinen sinnvollen Verwendungszweck für Raw Sockets, sondern verwenden den normalen TCP/IP-Protokollstack des Betriebssystems. Für Malware sind dagegen Raw Sockets ein nützlicher Mechanismus, um Angriffe wie z. B. ARP-Spoofing durchzuführen. Mittels VMI lassen sich kernelinterne Datenstrukturen von Raw Sockets finden und damit verdächtige Anwendungen identifizieren.

Neben diesen passiven Techniken diskutieren Garfinkel und Rosenblum (2003) auch aktive VMI-Techniken, die sich eignen, verdächtiges Verhalten nicht nur zu erkennen, sondern auch zu verhindern:

- *Durchsetzung von Speicherschutz:* Betriebssysteme verwenden heutzutage oft Mechanismen zum Speicherschutz, um bestimmte Bereiche im Hauptspeicher als nur lesbar zu markieren. Allerdings werden diese Schutzmechanismen durch das Betriebssystem gesteuert, so dass Malware auf Betriebssystemebene diese Mechanismen auch aufheben kann. Werden nun diese Schutzmechanismen auf Ebene des Hypervisors verlagert, so können diese vom Betriebssystem innerhalb einer virtuellen Maschine nicht mehr deaktiviert werden. Damit lässt sich z. B. erkennen, wenn Schadsoftware versucht, den Programmcode des Betriebssystems zu manipulieren.
- *Netzwerkgeräte-Schutz:* Manche Schadsoftware versucht Netzwerk-Geräte so zu konfigurieren, dass deren MAC-Adresse verändert wird oder dass der „promiscuous mode“ aktiviert wird, um so die komplette Kommunikation überwachen zu können. Wie bereits in Abschnitt 3.2 beschrieben, war dies vor allem in heute nicht mehr gebräuchlichen Netzwerken in Bus-Topologie interessant, um komplette Subnetze überwachen zu können. Der Hypervisor kann derartige Zugriffe auf virtualisierte Geräte erkennen und verhindern.

Eine experimentelle Untersuchung von Garfinkel und Rosenblum (2003) anhand von einigen realen Malware-Angriffen zeigt, dass viele dieser Angriffe durch die beschriebenen Mechanismen erkannt oder sogar verhindert werden können. Welche der Mechanismen sich eignen, hängt dabei jeweils vom konkreten Angriff ab, so dass es in der Praxis bei VMI-basierter Einbruchserkennung sinnvoll ist, verschiedene Techniken kombiniert einzusetzen.

VMI-basierte Angriffserkennung in der Cloud

Voraussetzung für die beschriebenen VMI-basierten Herangehensweisen ist es, dass ein Hypervisor mit dafür geeigneten VMI-Mechanismen eingesetzt wird. Grundsätzlich könnte jeder Hypervisor entsprechende technische Mechanismen einsetzen, in der Praxis ist dies aber nicht immer der Fall. Beispielsweise unterstützt die bekannte Bibliothek LibVMI nur Xen, KVM und QEMU als Hypervisor.

Selbst wenn VMI durch den verwendeten Hypervisor unterstützt wird, heißt dies aber nicht automatisch, dass sich VMI in jeder (insbesondere öffentlichen) Cloud-Umgebung verwenden lässt. Vielmehr ist dies heutzutage immer noch die Ausnahme. Für einen Überblick zum Stand der Technik von VMI in Cloud-Infrastrukturen verweisen wir an dieser Stelle auf den Studienbrief zum Mikromodul „Grundlagen von Cloud-Forensik“. Eine weiterführende Auseinandersetzung mit VMI bieten wir im optionalen Vertiefungs-Mikromodul „Virtual Machine Introspection“ an.

4 Honeypots

Dieser Abschnitt befasst sich detailliert mit Honeypots und der damit verbundenen Informationsbeschaffung über Angriffe und Angreifer. Nach einer ausführlichen Klassifizierung werden einzelne Vertreter genauer vorgestellt. Die Klassifizierungsmerkmale umfassen hierbei den Interaktionsgrad mit dem Angreifer, die Implementierungsweise sowie den Nutzen, der mit dem Einsatz solcher Systeme verfolgt werden sollen. Wie oben bereits angedeutet, ist das Hauptunterscheidungsmerkmal für Honeypots die Interaktion mit dem Angreifer. Nach einer generellen Einführung in Honeypots werden die Eigenschaften und Besonderheiten der Vertreter dieser Gruppen näher erläutert.

Vertreter der ersten Gruppe – Honeypots mit geringer Angreiferinteraktion oder auch *Low-Interaction Honeypots* – bilden reale Systeme nach und imitieren deren Verhalten. Zumeist öffnen solche Honeypots Standardports, um Angreifer anzulocken. Dies ermöglicht die Erstellung einer Statistik darüber, welche Dienste in den Augen von Angreifern besonders wertvoll oder erfolgversprechend sind. Außerdem können anhand der eigentlichen Verbindung bereits erste Informationen über den Ursprung des Angriffs und möglicherweise sogar über die Identität der Einbrecher gesammelt werden. Für manche Honeypots mag dieser Informationsgewinn bereits ausreichend sein, andere imitieren nach einem erfolgten Verbindungsaufbau einzelne Schritte des jeweiligen Protokolls, um weitere Informationen zu erlangen. Beispielsweise könnte ein SSH-Honeypot einen Anmeldevorgang simulieren, um Statistiken über die Authentifizierungsinformationen anzulegen, welche von Angreifern benutzt werden, um sich als reguläre Benutzer auszugeben und zu legitimieren.

Da die gesamte Interaktion nur emuliert ist, kann der Angreifer in aller Regel nicht in das System eindringen und größeren Schaden anrichten. Daneben liegen die Vorteile solcher Honeypots in geringen Implementierungs- und Betriebskosten, da nur genau die Teile der Dienste nachgebildet werden müssen, welche für die Analyse von Bedeutung sind, und außerdem auch in der Flexibilität und Effizienz, da mehrere virtuelle Honeypots in einem Honeypot zusammengefasst werden können. Gleichzeitig ergibt sich daraus natürlich auch die größte Begrenzung dieser Honeypots. Die Informationen, welche aufgrund der beschränkten Kommunikation gewonnen werden können, sind limitiert und ein Angreifer wird früh – eventuell zu früh – bemerken, dass er einem Honeypot ausgesetzt ist, und

sich dementsprechend abwenden. Zusammenfassend lassen sich Low-Interaction Honeypots wie folgt definieren.

Definition 5: Low-Interaction Honeypot

Low-Interaction Honeypots imitieren den Aufbau und das Verhalten bestimmter Bereiche von realen Systemen, um Erkenntnisse über potenzielle Angreifer zu erlangen.

D

Dem gegenüber stehen Honeypots mit hoher Angreiferinteraktion – *High-Interaktion Honeypots*. Hierbei handelt es sich um echte Systeme, welche Anwendungen und Dienste bereitstellen für die Angriffsvektoren gesammelt werden sollen. Dazu werden bereits im Vorhinein forensische Maßnahmen ergriffen, um Einbrüche bestmöglich beobachten und nachvollziehen zu können. Dies ermöglicht eine genaue Analyse der Vorgehensweise von Angreifern und bösartiger Schadsoftware. Da in Honeypots mit hohem Interaktionsgrad im Normalfall keine Komponenten emuliert sind, ist es für einen Angreifer deutlich schwieriger herauszufinden, dass er selbst von Sicherheitsanalysten beobachtet wird.

Allerdings ergibt sich aus dieser Struktur auch ein erhöhtes Risiko, dass ein Angreifer aus der geschützten Umgebung ausbricht und in dessen Einflussbereich Schaden anrichtet. Durch die Notwendigkeit zur konstanten Beobachtung ergibt sich auch ein besonders hoher Aufwand im Betrieb. Bereits die Bereitstellung solcher Systeme ist mit hohen Kosten verbunden, da dedizierte Maschinen betrieben werden müssen und teure Lizenzkosten anfallen können. Komprimiert können solche Systeme folgendermaßen definiert werden.

Definition 6: High-Interaction Honeypot

High-Interaction Honeypots sind reale Systeme, die Einbrüche explizit zulassen, um Informationen über das Verhalten von Angreifern und Schadsoftware zu gewinnen.

D

Von manchen Autoren wird zusätzlich die Kategorie der Honeypots mit mittlerer Angreiferinteraktion – *Medium-Interaction Honeypots* – definiert. Diese liegen – wie es der Name erahnen lässt – zwischen den beiden vorgenannten Gruppen. Im Grunde handelt es sich dabei um besonders aufwendig implementierte Low-Interaction Honeypots. Das bedeutet, dass die Dienstfunktionalitäten emuliert sind und Angreifer keinem realen System ausgesetzt sind. Gleichzeitig bildet es aber eine Vielzahl von Protokollschritten nach, so dass Angreifern hohe Interaktionsmöglichkeiten bereitstehen, und offeriert dabei auch sensible Betriebssystemressourcen wie das Dateisystem. Ein Honeypot mit mittlerer Angreiferinteraktion könnte also beispielsweise einen SSH-Dienst imitieren und nicht nur Statistiken über die Anmeldeversuche anlegen, sondern Angreifern auch erfolgreiche Anmeldungen vorgaukeln, um ihnen dann einen emulierten Kommandointerpreter bereitzustellen. Dieser könnte auch die Ausführung von Shellcode und das Nachladen von Schadsoftware auf die lokale Festplatte erlauben. Auf diese Weise könnten Malware-Proben für eine weitergehende Analyse gesammelt werden.

Insgesamt lässt sich festhalten, dass mit Medium-Interaction Honeypots weitaus mehr und komplexere Informationen gewonnen werden können, ohne direkt ein komplettes System offenzulegen. Dennoch erhöht sich im Vergleich zu Low-Interaction Honeypots natürlich das Risiko, dass Angreifer aus der isolierten Umgebung ausbrechen können – insbesondere wenn weitere Betriebssystemressourcen

genutzt und angeboten werden. Außerdem kann ein Honeypot mit mittlerer Angreiferinteraktion auch einen deutlich erhöhten Betreuungsaufwand erfordern, wenn der Honeypot von Zeit zu Zeit an den aktuellen Entwicklungsstand des nachgebildeten Dienstes angepasst werden muss. Zusammenfassend lässt sich ein Honeypot mit mittlerer Angreiferinteraktion wie folgt definieren.

D

Definition 7: Medium-Interaction Honeypot

Ein Medium-Interaction Honeypot ist ein erweiterter Low-Interaction Honeypot, welcher einen signifikanten Teil der Dienstfunktionalität nachbildet und dabei gegebenenfalls auch sensible Betriebssystemressourcen offenlegt.

Neben ihrem Interaktionsgrad können Honeypots auch nach ihrer Implementierungsweise unterschieden werden. So gibt es auf der einen Seite physische Honeypots, auf der anderen Seite virtuelle Honeypots. Physische Honeypots laufen auf einer physischen Maschine und sind deshalb meistens reale Systeme – also Honeypots mit hoher Angreiferinteraktion. Physische Honeypots sind heutzutage eher eine Seltenheit, da sie höhere Kosten verursachen und aufwendiger in der Betreuung und Pflege sind. Gerade in Cloud-Umgebungen überwiegen die Vorteile virtueller Honeypots. Hierbei können auf einer physischen Maschine mehrere Honeypots ausgeführt werden. Dies geschieht entweder durch klassische Hardwarevirtualisierung, in der mehrere unabhängige virtuelle Maschinen auf einem Hypervisor ausgeführt werden, oder durch geschickte Anfrageumleitungen auf eine physische Maschine, welche wiederum eine Vielzahl von Rechnern simuliert. So gibt es Honeypots, welche das Netzwerk nach freien IP-Adressen scannen und sich gegenüber den zuständigen Routern als legitime Hosts an diesen Adressen ausgeben. Anfragen an diese Adressen werden dann von einem internen Low- oder Medium-Interaction Honeypot beantwortet.

Ein weiteres Klassifizierungsmerkmal bildet der Einsatzzweck von Honeypots. Dabei unterscheidet man Honeypots für Forschungszwecke und Honeypots innerhalb von Produktivsystemen. Die Aufgabe von Honeypots in Produktivsystemen ist es, bestehende Systeme um die Fähigkeiten zur Erkennung und Verhinderung von Angriffen sowie der Reaktion auf Einbrüche zu erweitern. So kann ein prominent platzierter Nachbau eines Produktivsystems genutzt werden, um Angriffe frühzeitig zu erkennen. Über die dabei gewonnenen Informationen können etwaige Schwachstellen schnell geschlossen werden oder angemessen auf die Angriffe reagiert werden. Demgegenüber stehen Honeypots, welche zu Forschungszwecken betrieben werden und sich vorrangig mit der Informationsbeschaffung befassen. Die Betreiber wollen mit solchen Honeypots mehr über die Angreifer und die Bedrohungen, welche von ihnen ausgehen, erfahren. Dazu zählen die Identität der Angreifer sowie ihre Motive und insbesondere Taktiken. Ein weiteres großes Anwendungsfeld ist das Sammeln von Malware-Proben, wie weiter oben bereits skizziert wurde. Honeypots, die zu Forschungszwecken betrieben werden, haben folglich kurzfristig keinen direkten Nutzen für die Sicherheit von Produktivsystemen, langfristig aber können die gesammelten Informationen die Erkennungs-, Vermeidungs- und Reaktionsfähigkeiten von Produktivsystemen entscheidend verbessern.

4.1 Low-/Medium-Interaction Honeypots

Dieser Abschnitt befasst sich mit den Eigenschaften von Low- und Medium-Interaction Honeypots und stellt die verschiedenen Arten solcher Honeypots vor.

Honeypots mit geringer und mittlerer Angreiferinteraktion sind in der Regel virtuelle Honeypots (Provos und Holz, 2008). Das bedeutet, sie benötigen nur wenig Ressourcen und sind damit einfach skalierbar. Außerdem existiert eine Vielzahl an frei und quelloffen verfügbaren Systemen, welche leicht zu konfigurieren und damit schnell einsetzbar sind. Die meisten Systeme simulieren einfache Netzwerk- und Internetdienste. Es gibt aber auch spezielle Honeypots, welche versuchen Angreifer auszubremsen oder Proben von Schadsoftware zu sammeln. All diese Honeypots bringen dabei nur ein kalkulierbares Risiko mit sich, da sich Ausbrüche aus solchen Umgebungen für Angreifer als besonders schwierig erweisen, weil sie selbst keinen oder nur limitierter Zugriff auf das Betriebssystem gewähren. Da solche Umgebungen nur über einen limitierten Funktionsumfang verfügen, sind sie für Angreifer relativ schnell als Honeypots wahrnehmbar, was zur Folge hat, dass erfahrene Angreifer sich sehr schnell wieder von solchen Systemen abwenden werden. Auch ist es dadurch nicht möglich neue Angriffe zu erkennen. Für die Auswertungen, welche bekannten Angriffe zur Zeit besonders populär sind, sowie für Statistiken über Herkunft und Vorgehensweise der Angreifer und Auswertungen darüber, welche Dienste besonders gefährdet sind, eignen sich Low- und Medium-Interaction Honeypots allerdings hervorragend.

SSH-Honeypots

Der Netzwerkdienst Secure Shell (SSH) ist die vorherrschende Technik zur Steuerung entfernter Computer auf UNIX-artigen Betriebssystemen. Nach erfolgreicher Anmeldung erlaubt es die Nutzung eines Kommandozeileninterpreters am Zielhost, mit dem der gesamte Rechner nach Belieben gesteuert werden kann. Die Anmeldung erfolgt im einfachsten Fall mit einer Kombination aus Benutzername und Passwort. Die weitreichenden Möglichkeiten im Falle eines Einbruchs sowie die Simplizität des Angriffs an sich machen diesen Dienst so attraktiv für potenzielle Angreifer.

SSH-Honeypots bieten auf beliebigen Hosts Simulationen dieser Dienste an (Koniaris et al., 2013). Low-Interaction Honeypots lauschen dazu auf den dafür gängigen Ports und ahmen den Protokollverlauf nach. Dabei sammeln sie Statistiken über die Ursprünge der Angriffe, über das Auftreten und die Häufigkeit der Attacken sowie die verwendeten Login-Daten. Medium-Interaction Honeypots gehen noch einen Schritt weiter, indem sie bestimmte Anmeldeversuche von Angreifern durchlassen und ihnen im nächsten Schritt einen Kommandozeileninterpreter auf einem emulierten Dateisystem präsentieren. Mit Hilfe dieses Interpreters können etwaige Kommandos und Dateien, welche während eines Einbruchs ausgeführt beziehungsweise nachgeladen werden, analysiert werden.

Ein frei verfügbarer und aktiv weiterentwickelter Vertreter dieser Gruppe ist *Cowrie*[4], der Nachfolger des bekannten aber nicht mehr weiterentwickelten Honeypots *Kippo*[5].

Netzwerk- und Internetdienste

Analog zu SSH-Honeypots existiert eine Vielzahl weiterer Honeypots, die bestimmte Netzwerk- und Internetdienste nachbilden. Die Funktionsweise ähnelt stark derer der gerade vorgestellten SSH-Honeypots. Der benötigte Dienst wird soweit nachgebaut, wie es für die Informationsgewinnung erforderlich ist. Dabei werden bekannte Schwachstellen eingebaut, um Statistiken darüber zu gewinnen, wie oft und auf welche Weise solche Sicherheitslücken ausgenutzt werden.

[4] `https://github.com/micheloosterhof/cowrie` [abgerufen am 2017-04-01]
[5] `https://github.com/desaster/kippo` [abgerufen am 2017-04-01]

Beispiele hierfür sind der Honeypot *HonnyPotter*[6], welcher den Anmeldevorgang zur weit verbreiteten Blog-Software *WordPress*[7] emuliert, oder der *phpMyAdmin-Honeypot*[8], welcher ein Administrationsinterface für Datenbanken nachbildet.

Teergruben

Bei Teergruben oder auch klebrigen Honeypots handelt es sich um spezielle Honeypots, die Angreifern direkt schaden indem sie deren Angriffe eindämmen beziehungsweise verlangsamen. Typischerweise sollen so die Verbreitung von Spam und die Propagation von Schadsoftware verhindert oder hinausgezögert werden. Dazu geben sich diese Honeypots abermals als wertvolle Ziele aus und imitieren bestimmte Funktionsweisen der vorgegaukelten Dienste.

Im Gegensatz zu den vorgenannten Honeypots liegt bei dieser Gruppe der Hauptfokus allerdings nicht auf der Gewinnung von Informationen über den Angreifer oder der Analyse der Ausnutzung von Schwachstellen. Vielmehr sollen diese Systeme Angreifern explizit die Ausführung der bösartigen Aktionen und das Abgreifen von Daten erlauben. Deswegen bieten solche Honeypots besonders aufwendige Aktionen an und werden mit gefälschten Daten ausgestattet, welche dem Angreifer authentisch erscheinen. Die gefälschten Daten werden geopfert, um echte Daten zu schützen. Die bösartige Aktionen werden von Firewalls und Sandboxen geblockt und ausgesperrt. Auf diese Weise bleiben Angreifer möglichst lange beschäftigt und können in echten Systemen keinen Schaden anrichten.

Ein Vertreter dieser Klasse ist der E-Mail-Honeypot *Honeyspam* (Andreolini et al., 2005). Er präsentiert sich Angreifern als offenes Relay, über welches die Angreifer Spam versenden können. Der so versendete Spam wird allerdings zur Analyse mitprotokolliert und später verworfen. Daneben bietet Honeyspam noch die Störung der Sammlung von E-Mail-Adressen. Spammer durchsuchen mit Hilfe von Crawlern das Internet nach validen E-Mail-Adressen, an die sie ihre Nachrichten senden können. Honeyspam baut virtuelle Webseiten auf, um Crawler möglichst lange in diesen Umgebungen zu halten. Diese Webseiten sind mit gefälschten E-Mail-Adressen, welche keinem realen E-Mail-Konto zugewiesen sind, ausgestattet, um den Angreifern ein scheinbar wertvolles Ziel zu bieten. Außerdem sollen die falschen E-Mail-Adressen die Datenbanken der Spammer mit unnützen Daten füllen, um die Effektivität deren Angriffe zu vermindern. Darüber hinaus können Angreifer später anhand bestimmter E-Mail-Adressen, welche nur solchen Spammern bekannt sein können, schnell identifiziert und mittels schwarzer Listen blockiert werden.

Sammlung von Schadsoftware

Schadsoftware – Viren, Würmer und Trojaner – bedrohen die Integrität der vorhandenen IT-Systeme. Heutzutage entsteht immer schneller neue Schadsoftware, welche sich rasch über das Internet verbreitet, indem sie Sicherheitslücken ausnutzt. Eine detaillierte Analyse dieser Software ist nötig, um die vorhandenen Systeme zu schützen. Auf diese Weise können neue Signaturen für Virenscanner und Einbruchserkennungssysteme gewonnen sowie Erkenntnisse und Statistiken über Angriffsmuster und Angriffstrends abgeleitet werden. Diese Analyse ist im Regelfall allerdings erst möglich wenn die Schadsoftware aktiv geworden ist. Eine rein manuelle Detektion und Analyse kann mit der Geschwindigkeit, mit der neue adaptive Varianten generiert werden, heutzutage kaum mehr durchgeführt

[6] `https://github.com/MartinIngesen/HonnyPotter` [abgerufen am 2017-04-01]
[7] `https://wordpress.org/` [abgerufen am 2017-04-01]
[8] `https://github.com/gfoss/phpmyadmin_honeypot` [abgerufen am 2017-04-01]

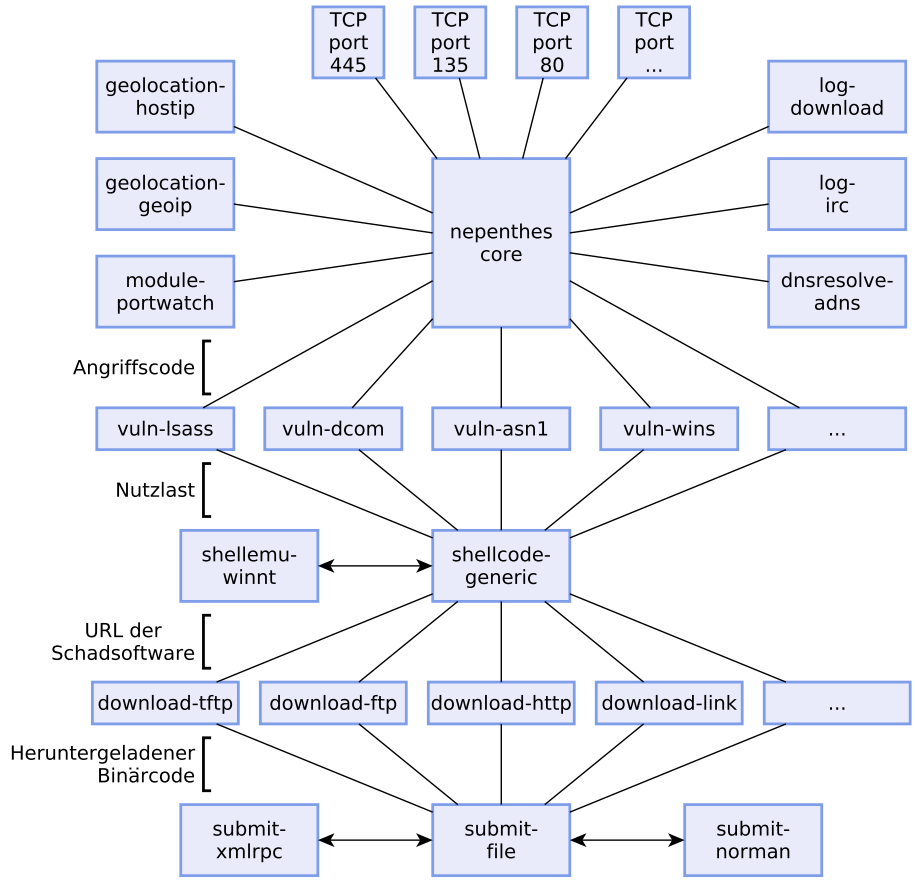


Abb. 4: Architektur des Nephentes-Honeypots (Baecher et al., 2006)

werden. Um die Sammlung von Schadsoftwareproben zu automatisieren, existiert eine weitere Gruppe von Honeypots.

Honeypots zum Sammeln von Schadsoftware emulieren bekannte Schwachstellen von Computersystemen, welche von Schadsoftware ausgenutzt werden, um sich selbstständig und automatisiert weiter zu verbreiten. Hierbei handelt es sich vorrangig um Schwachstellen, welche im Erfolgsfall eine Verbindung zu einem Kommandointerpreter des infiltrierten Rechner erlauben würden, damit die Schadsoftware weitere Software nachladen und sich von dort auf weitere Rechner übertragen kann. Solche Interpreter werden auch im virtuellen Honeypot nachgebildetr um der Schadsoftware dieses Nachladen zu ermöglichen. Die übertragene Software sowie der vorhergehende Einbruch werden allerdings ständig von Analysekomponenten beobachtet und ausgewertet, um genaue Informationen über die Vorgehensweise sowie Proben der eingesetzten Schadsoftware zu gewinnen.

Nepenthes (Baecher et al., 2006) ist ein solcher Honeypot, der es ermöglicht, verwundbare Netzwerk- und Internetdienste zu simulieren. Ein einziger Honeypot virtualisiert dabei eine Vielzahl unterschiedlichster Dienste. Dies wird durch die modulare Architektur ermöglicht, welche in Abbildung 4 skizziert ist. Neben dem Nepenthes-Kern, welcher die Netzwerkkommunikation übernimmt und die anderen Module steuert, gibt es folgende Arten von Modulen:

Verwundbarkeitsmodule Die Verwundbarkeitsmodule emulieren zahlreiche Dienste mit Schwachstellen, welche von automatisierter Schadsoftware bekanntlich ausgenutzt werden. Hierbei werden nur die Teile der Dienste nachgebaut, die für einen erfolgreichen Einbruch notwendig sind. Diese

ressourcenschonende Implementierung ermöglicht eine hohe Skalierung des Honeypots. Im Falle der Ausnutzung von Schwachstellen wird die eigentliche Nutzlast des Angriffs an die Analysemodule weitergegeben.

Analysemodule Die Analysemodule interpretieren den Shellcode der Nutzlast, sowohl für UNIX-basierte Betriebssysteme als auch für die Windows-Betriebssystemfamilie. Dazu versuchen sie zuerst den Shellcode, welcher zumeist verschlüsselt oder zumindest obfuskiert wurde, zu entschlüsseln. Wenn dies gelingt, werden in einem weiteren Schritt bestimmte Charakteristika des Angriffs wie zum Beispiel URLs oder Anmeldedaten aus typischen Codefragmenten und Funktionsaufrufen extrahiert. Diese Informationen werden an die Downloadmodule weitergereicht.

Downloadmodule Die einzige Aufgabe der Downloadmodule ist das Herunterladen von Dateien, welche die Schadsoftware nachladen möchten. Die passenden URLs liefern dazu die Analysemodule. Je nach URL werden unterschiedliche Module aufgerufen, wobei viele bekannte Protokolle – wie HTTP und FTP – genauso wie eher selten verwendete Methoden – wie die von Botnetzen häufig eingesetzte Internet Relay Chat (IRC) Übertragungsmethode cSend/cReceive – unterstützt werden.

Distributionsmodule Um die Weiterverteilung der heruntergeladenen Dateien kümmern sich letztendlich die Distributionsmodule. Diese speichern die Dateien auf der Festplatte und in Datenbanken oder transferieren sie an entfernte Rechner zur weiteren Analyse.

Protokollmodule Die Protokollmodule protokollieren Informationen über die erfolgten Emulationen und Analysen und geben so einen Überblick über die gewonnenen Daten und Erkenntnisse.

Die Beschreibung der Module zeigt, dass Nepenthes standardmäßig schon einen weiten Funktionsumfang bietet. Dieser kann selbstverständlich durch Eigenentwicklungen erweitert werden. Dennoch hat sich Nepenthes im Laufe der Zeit als zu unflexibel herausgestellt. Zur Lösung dieses Problems wurde mit *Dionaea*[9] ein direkter Nachfolger von Nepenthes entwickelt. Dionaea bringt Python als Skriptsprache zur Steuerung des Honeypots mit, was die Flexibilität deutlich erhöht. Daneben steigerte es mit der Unterstützung von IPv6 und TLS die Einsatzmöglichkeiten. Außerdem konnte die Genauigkeit der Analysemodule gesteigert werden.

Kombinierte Lösungen

Selbstverständlich existieren auch kombinierte Lösungen, die mehrere oder alle der vorangestellten Aufgabenbereiche in sich vereinen. Ein Beispiel für solch eine Allzwecklösung ist der virtuelle Low- beziehungsweise Medium-Interaction Honeypot *Honeyd* (Provos, 2004).

Dieser Honeypot spannt eine Vielzahl von virtuellen Honeypots über weite Netzbereiche und kann eine Vielzahl von Betriebssystemen und Dienste an diesen Adressen simulieren. Abbildung 5 verdeutlicht die Architektur von Honeyd. Der Netzwerkverkehr für alle konfigurierten IP-Adressen wird an den Honeypot umgeleitet, welcher auf Anfragen mit den betriebssystemspezifischen Charakteristika antwortet, sodass die Probing-Werkzeuge der Angreifer tatsächlich die vorkonfigurierten Betriebssysteme und Dienste vermuten. Dabei kann Honeyd auch beliebig komplexe Netzwerkstrukturen und -topologien nachbilden, sodass dem Angreifer ein funktionales Netzwerk vorgegaukelt wird, während dieser aber eigentlich nur mit einem einzelnen Host kommuniziert.

[9] `https://github.com/DinoTools/dionaea` [abgerufen am 2017-04-01]

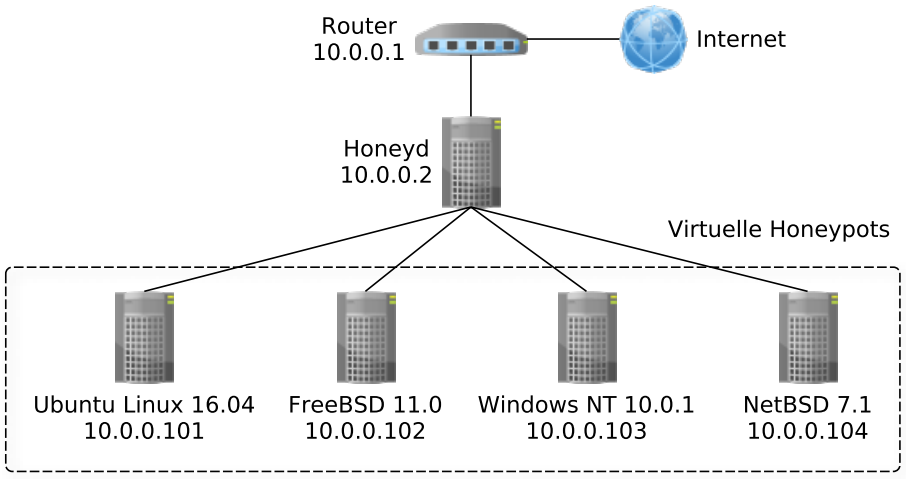


Abb. 5: Virtuelle Honeypots in der Honeyd-Architektur (Provos, 2004)

Um dies zu erreichen, durchläuft der Netzwerkverkehr – wie in Abbildung 6 dargestellt – die verschiedenen Komponenten von Honeyd. Die Pakete kommen am Packet Dispatcher an, der diese überprüft und verifiziert. Außerdem wird der jeweiligen Verbindung das in der zentralen Konfiguration definierte Profil (Personality) zugewiesen. Danach werden die Pakete an die einzelnen Protocol Handler weitergereicht. Honeyd ist hierbei in der Lage ICMP-, TCP- und UDP-Verkehr zu verstehen und einzusetzen. Während ICMP-Verkehr direkt vom zuständigen Protocol Handler beantwortet wird, durchlaufen die anderen Protokollvarianten die jeweils zuständigen Services. Diese Dienste können sowohl simulierte Netzwerk- und Internetdienste – wie beispielsweise einen emulierter SSH-Server – umfassen, als auch Weiterleitungen zu echten Diensten – wie zum Beispiel einem produktiven DNS-Server – darstellen. Bevor Pakete den Honeypot wieder verlassen, kümmert sich die Personality Engine darum, dass die ausgehenden Pakete die für das jeweils konfigurierte Betriebssystem typischen Merkmale aufweisen.

Die einzelnen virtuellen Hosts werden mittels Templates konfiguriert, welche auch die darin enthaltenen Services definieren. Dieser Plugin-Mechanismus ermöglicht die einfache Erweiterbarkeit der Funktionalität des Honeypots um beliebige Dienste. Die Funktionalität kann entweder emuliert oder als Weiterleitung an externe Dienste realisiert sein. Honeyd startet dazu die gewünschten Prozesse und leitet die Kommunikation an die Prozesse weiter. Auf diese Weise ist es möglich, mit nur einem einzigen Honeypot eine Vielzahl von unterschiedlichen Zielen zu verfolgen. Ein kombinierter Honeypot könnte so mit einer komplexen Netzwerkinfrastruktur potenzielle Angreifer zeitweise von der eigentlichen Produktivinfrastruktur abhalten und gleichzeitig zur Spamvermeidung sowie zur Analyse von Schadsoftware genutzt werden.

4.2 High-Interaction Honeypots

Low- und Medium-Interaction Honeypots decken mit einem kalkulierbaren Risiko einen weiten Einsatzbereich ab. Die Emulation der vorgetäuschten Dienste ist allerdings zeitaufwendig und muss immer an aktuelle Entwicklungen angepasst werden. Außerdem können erfahrene Angreifer relativ schnell erkennen, dass sie mit einem Honeypot kommunizieren. Deswegen gibt es mit High-Interaction Honeypots noch eine weitere Gruppe von Honeypots, bei denen es sich um echte Systeme handelt, mit denen Angreifer in vollem Umfang interagieren können (Provos und Holz, 2008).

Der hohe Interaktionsgrad ermöglicht die Erfassung aller Charakteristika des

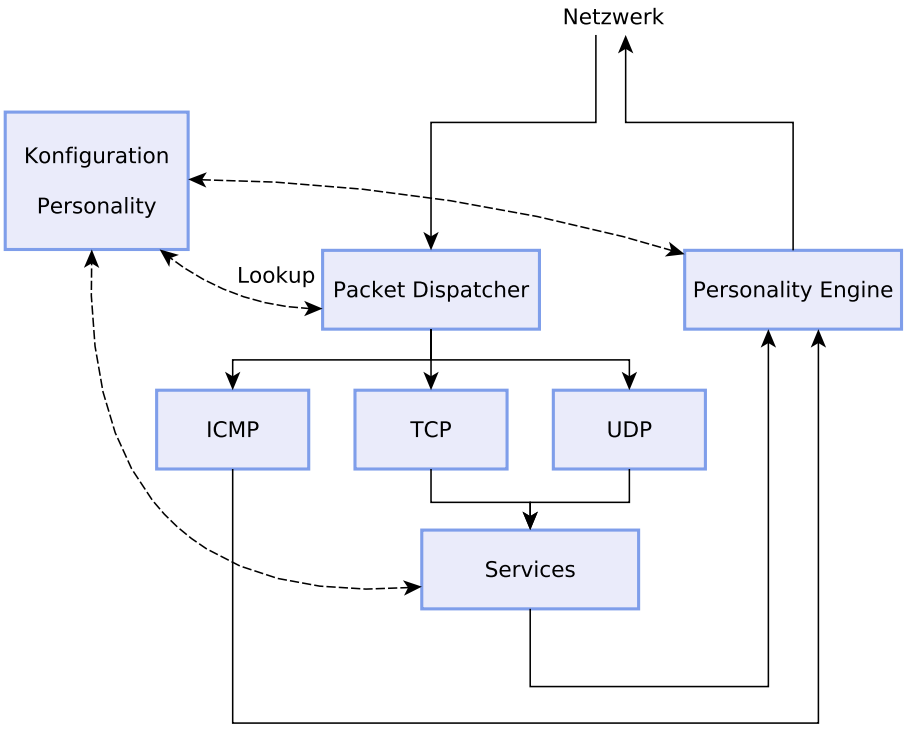


Abb. 6: Details zur internen Architektur von Honeyd (Provos, 2004)

Angriffs – also die Aktionen mit denen der Angreifer sein Ziel findet, die Vorgehensweise zur Überlistung von Zugangsbeschränkungen, welche Tools dafür eingesetzt werden, mit welchen Akteuren der Angreifer kommuniziert und welche Ziele der Angreifer verfolgt. Der Betrieb solcher Honeypots ist mit einem gleichwohl höheren Risiko behaftet, da der Angreifer den vollen Funktionsumfang des Betriebssystems für seine weiteren Aktionen nach dem Einbruch nutzen kann. Eine gute Isolation sowie eine einfache Resetfunktionalität sind für Hosts, auf denen solche Honeypots betrieben werden, unabdingbar. Deswegen werden im Folgenden die wichtigsten Konzepte und Vorgehensweisen für einen effektiven und vor allem sicheren Betrieb solcher Honeypots erläutert.

Setup

Eine virtualisierte Umgebung eignet sich ganz besonders als Basisplattform für den Betrieb von Honeypots mit hoher Angreiferinteraktion (Provos und Holz, 2008). Virtuelle Maschinen erlauben es, auf einem physischen Host mehrere (idealerweise vielfältige) Honeypots zu platzieren. Diese Maschinen können besonders einfach aufgesetzt und zurückgesetzt werden, wenn die Analyse abgeschlossen wurde. Außerdem können die Maschinen von außen beeinflusst werden, solange der Angreifer nicht aus der isolierten Umgebung ausbricht und auch den Hypervisor übernimmt.

Die auf dem Honeypot installierte Software richtet sich nach dem Einsatzzweck. Manche High-Interaction Honeypots werden einfach nur mit einem generischen Betriebssystem ausgestattet, um generelle Informationen zu sammeln. Andere Honeypots werden mit bestimmten Diensten ausgestattet, um Erkenntnisse über Angriffe auf diese Dienste zu erlangen. Wieder andere Honeypots laufen mit ganz bestimmten und bekanntlich verwundbaren Versionen dieser Dienste, um die Ausnutzung solcher Schwachstellen genau analysieren zu können. Da auf all diesen Honeypots keine Produktivanwendungen laufen, ist jede Interaktion mit dem System und den Diensten verdächtig. Dies ist eine Eigenschaft, welche sie von

Systemen zur Einbruchserkennung deutlich unterscheidet. Um ein Ausbrechen der Angreifer zu verhindern, ist darauf zu achten, dass keine Daten auf den Honeypots abgelegt werden, welche dem Angreifer Erkenntnisse über die umgebende Infrastruktur geben könnten.

Monitoring

Die zentrale Funktion, welche beim Betrieb eines Honeypots verfolgt wird, ist die Informationsgewinnung. Dazu existieren verschiedene Möglichkeiten. Die Kommunikation mit dem Honeypot kann mit Hilfe von Netzwerksniffern, wie beispielsweise *tcpdump*[10] oder *Wireshark*[11] aufgezeichnet werden. Die Sniffer können dabei an exponierten Stellen im Netzwerk – beispielsweise auf Routern und an den SPAN-Ports von Switches – oder im Falle eines virtualisierten Honeypots auf dem Hypervisor betrieben werden. Dies deckt jedoch nur die Netzwerkkommunikation und schließt verschlüsselte Kommunikation im Regelfall aus.

Um alle Aktionen des Angreifers nachvollziehen zu können, sind weitere Monitoring-Tools auf dem Host notwendig. Neben der systematischen Auswertung von Log-Dateien existieren für diesen Zweck spezielle Datenerfassungswerkzeuge. *Sebek*[12] ist ein solches Tool, das auf dem Honeypot als Kernelmodul läuft. Dieses Modul sammelt Daten indem es Systemaufrufe abfängt und schickt diese Daten an einen Server, welcher auf einem anderen, geschützten Rechner läuft. Sebek verwendet Rootkit-Techniken, um seine Anwesenheit und Kommunikation zu verschleiern.

Argos ist ein weiteres Werkzeug zur Gewinnung von Daten (Portokalidis et al., 2006). Es basiert auf der Virtualisierungssoftware *QEMU*[13] und nutzt zur Erkennung von Exploits Taintanalyse-Techniken. Daten, die von außen in das System eingebracht werden, werden auf diese Weise markiert. Die Markierung wird durch das System propagiert, solange bis unsichere Aktionen auf diesen Daten ausgeführt werden. Im Falle eines solchen Alarms werden der aktuelle Systemzustand und die nachfolgenden Operationen genau aufgezeichnet und zur Analyse und Generierung von Signaturen an einen sicheren Server geschickt. Durch die Emulation ist das Werkzeug plattform- und applikationsunabhängig.

Virtualisierungstechniken bietet zudem die Möglichkeit der Informationsgewinnung mittels *Virtual Machine Introspection*. Diese Technik erlaubt eine detaillierte Analyse, ohne dem Angreifer direkte Rückschlüsse auf diese Maßnahmen zu ermöglichen. Ein bekanntes Beispiel hierzu ist *VMI-Honeymon* (Lengyel et al., 2012). VMI-Honeymon verwendet das Analysewerkzeug *Volatility* zusammen mit der Introspektionsbibliothek *LibVMI*, um automatisiert kritische Veränderungen im Hauptspeicherzustand eines High-Interaction Honeypots zu erkennen. Dabei werden Anomalien durch Vergleich mit dem Normalzustand des Systems erkannt.

Separation

Besonders beim Betrieb von Honeypots mit hohem Interaktionsgrad wird die Ausführungsumgebung einem hohen Risiko ausgesetzt (Provos und Holz, 2008). Es besteht ständig die Gefahr, dass Angreifer ihre Angriffe auch auf die den Honeypot umgebende Infrastruktur ausweiten. Daneben sind die schadhaften Handlungen – wie das Versenden von Spam oder der Angriff weiterer IT-Systeme – immer erst einmal auf den Betreiber des Honeypots zurückzuführen. Um nicht in gesetzliche

[10] `http://www.tcpdump.org/` [abgerufen am 2017-04-01]
[11] `http://www.wireshark.org/` [abgerufen am 2017-04-01]
[12] `https://projects.honeynet.org/sebek/` [abgerufen am 2017-04-01]
[13] `http://qemu-project.org/` [abgerufen am 2017-04-01]

oder zumindest moralische Bedrängnis zu geraten, ist eine starke Separation der Honeypots unabdingbar.

Zur Abtrennung von Honeypots und produktiven Systemen beziehungsweise potenziellen Angriffszielen werden Firewalls im umgebenden Netzwerk eingesetzt. Die einfachste Möglichkeit wäre es, den gesamten ausgehenden Datenverkehr zu blockieren. Allerdings können Angreifer bei totaler Isolation auch keine Dateien nachladen und nicht mehr freizügig agieren. Sie werden zwangsläufig ihre Angriffe anpassen oder abbrechen. Besser ist es, nur bestimmten Datenverkehr einzuschränken. Die Honeypot-Firewall *Honeywall*[14] sieht beispielsweise zur Verhinderung von Denial-of-Service-Attacken (DoS-Attacken) eine tägliche Begrenzung der TCP-Verbindungen und ICMP-Paketen vor. Daneben setzt die Firewall zusätzlich auf Snort als Einbruchverhinderungssystem, welches ausgehende Pakete auf Schadsoftware prüft und gegebenenfalls unschädlich macht. Bei der Separation von Honeypots liegt die besondere Herausforderung in der Bestimmung eines geeigneten Verhältnisses zwischen Isolation und Freizügigkeit.

4.3 Hybride Honeypots

Hybride Honeypots verbinden die Vorteile von High-Interaction Honeypots und Low-Interaction Honeypots (Berthier, 2009). Hierbei fungiert eine Vielzahl von virtuellen Honeypots mit niedriger oder mittlerer Angreiferinteraktion als Einfallstor in das Honeynet. Diese sind effizient und kostengünstig zu betreiben. Da die Analysemöglichkeiten dieser Honeypots begrenzt sind, wird ein Teil der Angriffe – nach bestimmten Kriterien – an Honeypots mit hoher Angreiferinteraktion weitergeleitet, auf welchen detaillierte Informationen gesammelt werden können.

Honeybrid ist ein solcher hybrider Honeypot. Der generelle Aufbau ist in Abbildung 7 skizziert. Honeybrid bringt eine *Decision Engine* mit sich, welche den eingehenden Netzwerkverkehr filtert, um bestimmten Verkehr vorzuselektieren und die Pakete an bestimmte Low-Interaction Honeypots weiterleitet. Die *Redirection Engine* entscheidet anschließend welche Verbindungen für eine weitergehende Analyse an High-Interaction Honeypots umgeleitet werden. Mit Hilfe der nicht abgebildeten *Control Engine* kann ausgehender Netzwerkverkehr zum Schutz der eigenen Infrastruktur unterdrückt werden. Die ebenfalls nicht dargestellte *Log Engine* legt ausführliche Berichte über den ausgetauschten Datenverkehr und die ausgeführten Aktionen an.

Zur Steuerung des Datenverkehrs kennt Honeybrid das Konzept der *Targets*. Der eingehende Traffic wird mit den definierten Targets verglichen und dementsprechend verarbeitet. Ein Target besteht dabei aus einer Filterregel, die definiert, auf welche eingehende Pakete das Target angewendet werden soll, einer Frontend-Regel, welche den Honeypot festlegt, an den der Verkehr weitergeleitet werden soll, einer Backend-Regel, welche die Kriterien und Ziele zur Umleitung des Netzwerkverkehrs an andere Honeypots bestimmt, sowie einer optionale Kontrollregel, welche den ausgehenden Datenverkehr einschränken kann. Auf diese Weise bringt der Honeypot die einzelnen Module zusammen und verbindet so die hohe Skalierbarkeit und einfache Wartbarkeit mit hohen Interaktions- und detaillierten Analysemöglichkeiten.

4.4 Cloudbasierte Honeypots

Cloud-Umgebungen eignen sich hervorragend für Honeypots und Honeynets. Die Skalierbarkeit von Rechnerverbünden und flexible Abrechnungsmodelle erlauben einen kosteneffizienten und ressourceneffektiven Betrieb. Je nach Anforderungen

[14] `https://projects.honeynet.org/honeywall/` [abgerufen am 2017-04-01]

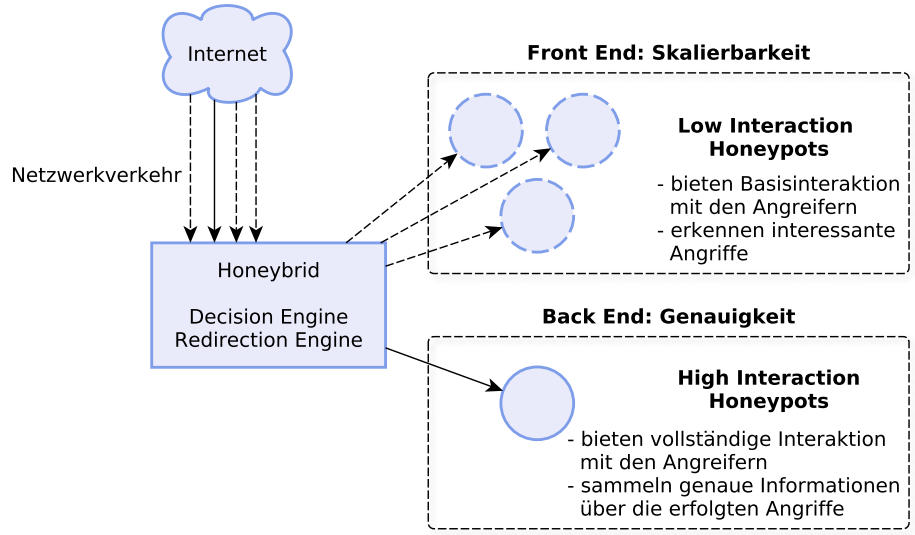


Abb. 7: Honeybrid Architektur (Berthier, 2009)

und Verkehrsaufkommen können schnell weitere Honeypots in ein Honeynet hinzugefügt oder herausgenommen werden. Hierbei erleichtert die zugrunde liegende Virtualisierung die Bereitstellung und Konfiguration sowie das Zurücksetzen von virtuellen Maschinen und Netzwerken.

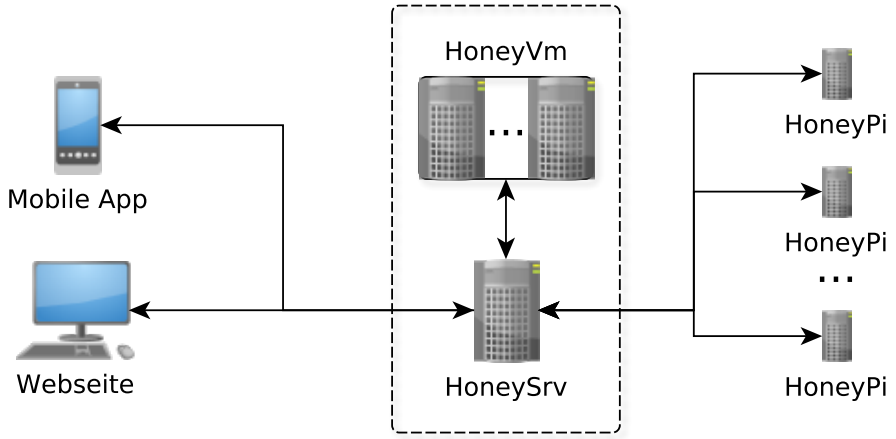


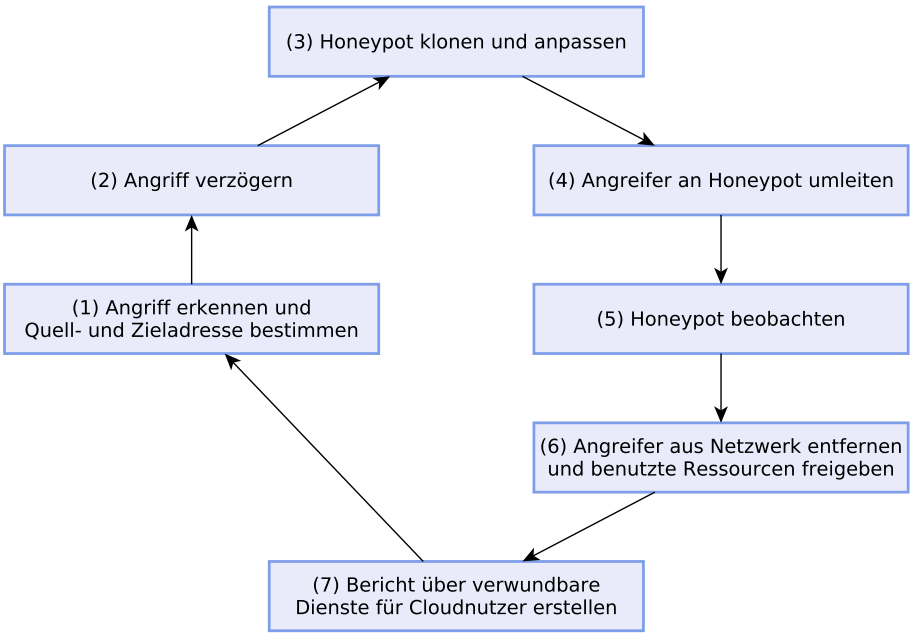
Abb. 8: HoneyCY Architektur (Christoforou et al., 2015)

HoneyCY ist solch ein cloudbasierter Honeypot, welcher die enormen Skalierungsmöglichkeiten von Cloud-Umgebungen ausnutzt, um ein möglichst breites Spektrum an Honeypots als Gesamtverbund zu betreiben (Christoforou et al., 2015). Den grundlegenden Aufbau zeigt Abbildung 8. HoneyCY erfüllt seine Aufgaben in drei verschiedenen Bereichen. Sensorknoten (HoneyPi) erzeugen Honeypots und zeichnen so Daten über Angriffe auf. Hierfür unterstützt HoneyCY drei bekannte Honeypots, welche automatisch eingerichtet und konfiguriert werden. Gesteuert wird dieser Vorgang über einen zentralen Administrationsserver (HoneySrv). Dieser stellt Endbenutzern ein Webinterface bereit oder lässt sich über eine mobile Anwendung bedienen. Außerdem sammelt der Server die Daten von den Sensorknoten und schickt die Schadsoftwareproben zur Identifizierung an den externen Analysedienst *VirusTotal* [15]. Falls diesem die Schadsoftware nicht bekannt ist, werden die Proben zur weiteren Analyse an die eigene Analysekomponente (HoneyVM) gesendet. Diese besteht aus einem streng abgeschirmten Verbund von

[15] `https://www.virustotal.com/` [abgerufen am 2017-04-01]

virtuellen Maschinen, auf denen spezielle Analysesoftware versucht, die Angriffe genauer zu bestimmen und nachzuvollziehen. Berichte und Statistiken über die gesammelten und analysierten Proben können jederzeit über die Benutzerschnittstellen eingesehen werden oder automatisch per E-Mail versandt werden. HoneyCY ermöglicht so eine einfache und bedarfsgesteuerte Bereitstellung einer Vielzahl von Honeypots in Cloud-Umgebungen und unterstützt die Analyse durch weitreichende Visualisierungen.

Abb. 9: Dynamische Honeypot-Erzeugung (Biedermann et al., 2012)



Honeypots werden außerdem häufig als Teil von größeren Systemen zur Einbruchserkennung- und verhinderung eingesetzt. Ein Beispiel ist das von Biedermann et al. (2012) vorgeschlagene dynamische Honeypot-System, welches in Cloud-Umgebungen eingesetzt wird, um Attacken gegen produktive virtuelle Maschinen (VM) auf dynamisch erzeugte Honeypots umzuleiten. Dazu werden – wie in Abbildung 9 dargestellt – nach der Detektion eines Angriffes die IP-Adressen vom Angreifer und dessen Ziel extrahiert. Daraufhin wird zunächst der Angriff an sich verzögert, um Zeit für die folgenden Schritte zu gewinnen. Diese Schritte beginnen mit dem Klonen des Angriffsziels in eine neue VM mittels Live-Migration. Danach werden sensible Daten von der neuen VM entfernt oder durch unbedenkliche Daten ersetzt. Anschließend wird der Angreifer an die neue Honeypot-VM umgeleitet. Die Maschine wird währenddessen umfassend beobachtet, um den Angriff genau nachvollziehen zu können. Wenn die Analyse abgeschlossen ist, wird der Angreifer aus dem Netzwerk entfernt und die zusätzlichen Cloud-Ressourcen wieder freigegeben. Im letzten Schritt erfolgt die Information des Betreibers der betroffenen VM über den erfolgten Angriff und die Verwundbarkeit des Dienstes. Dieser kann dann geeignete Maßnahmen einleiten, um ein solches Verhalten in der Zukunft vermeiden zu können.

5 Übungsaufgaben

Übung 1: Einbruchserkennung und Honeypots

Erläutern Sie die Begriffe Intrusion Detection System und Intrusion Prevention System sowie Honeypot und Honeynet. Inwiefern kann ein Honeypot ein System zur Einbruchserkennung unterstützen?

Ü

Übung 2: Platzierung von Einbruchserkennungssystemen

Diskutieren Sie die Vor- und Nachteile von hostbasierten, netzwerkbasierten sowie Hypervisor-basierten Systemen zur Einbruchserkennung hinsichtlich der folgenden Merkmale:

(a) Wahrscheinlichkeit, dass der Angreifer die Überwachung erkennt;

(b) Menge der Daten, die zur Erkennung herangezogen werden können.

Ü

Übung 3: Detektionsmechanismen von Einbruchserkennungssystemen

Erläutern Sie die Funktionsweise der folgenden generellen Detektionsmechanismen eines Systems zur Einbruchserkennung.

(a) Signaturbasierte Einbruchserkennung

(b) Verhaltens-/Anomaliebasierte Einbruchserkennung

Ü

Übung 4: Hostbasierte Einbruchserkennung

Nennen Sie mindestens drei Methoden, anhand derer hostbasierte Systeme zur Einbruchserkennung versuchen, Angriffe zu detektieren.

Ü

Übung 5: Netzwerkbasierte Einbruchserkennung

Erörtern Sie Herausforderungen bei der Erfassung von Netzwerkdaten, welche bei der Installation von Systemen zur Einbruchserkennung im Netzwerk entstehen können.

Ü

Übung 6: Hpyervisor-basierte Einbruchserkennung

Nennen Sie mindestens drei Methoden, mit denen Einbruchserkennungssysteme auf dem Hypervisor virtuelle Maschinen zum Ziele der Einbruchserkennung analysieren.

Ü

Ü

Übung 7: Snort

Informieren Sie sich zur Funktionsweise von Snort und über die Definiton von Snort-Regeln – beispielsweise anhand der offiziellen Dokumentation[16].

(a) Definieren Sie eine Snort-Regel, welche den gesamten Netzwerkverkehr (also sowohl eingehende als auch ausgehende Verbindungen) zu dem Dienst auf Port 1234 protokolliert.

(b) Definieren Sie eine Snort-Regel, welche für jeden Verbindungsaufbau zum SSH-Dienst einen Alarm auslöst.

(c) Modifizieren Sie die Snort-Regel von (b) so, dass nur noch Alarme ausgelöst werden, wenn die Verbindungen nicht von Hosts mit der IP-Adresse 10.0.0.201 aus erfolgen.

(d) Definieren Sie eine Snort-Regel, welche einen Alarm auslöst, wenn innerhalb von 30 Sekunden mehr als fünf Verbindungsaufbauten zu dem Dienst auf Port 1234 festzustellen sind.

(e) Definieren Sie eine Snort-Regel, welche einen Alarm auslöst, wenn die Zeichenkette „rotten“ (UTF8-kodiert) innerhalb von Nachrichten an den Dienst auf Port 1234 gefunden wird.

Ü

Übung 8: Honeypots

Klassifizieren Sie Honeypots hinsichtlich...

(a) ... der Interaktion mit dem Angreifer.

(b) ... der Art der Implementierung.

(c) ... dem verfolgten Zweck ihres Einsatzes.

Benennen Sie dabei die Vor- und Nachteile der jeweiligen Gruppen.

Ü

Übung 9: Anwendungsgebiete von Honeypots

Nennen Sie mindestens drei Anwendungsgebiete von Honeypots.

Ü

Übung 10: Hybride Honeypots

Diskutieren Sie die Chancen in der Kombination von Honeypots mit unterschiedlicher Angreiferinteraktion.

Ü

Übung 11: Honeypots in der Cloud

Welche Eigenschaften von Cloud-Umgebungen können sich Honeypots zu Nutze machen?

[16] `https://www.snort.org/documents` – [abgerufen am 2015-03-01]

Verzeichnisse

I. Abbildungen

II. Beispiele

III. Definitionen

IV. Literatur

Turki Alharkan und Patrick Martin. IDSaaS: Intrusion Detection System as a Service in Public Clouds. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, S. 686–687. IEEE Computer Society, 2012.

Mauro Andreolini, Alessandro Bulgarelli, Michele Colajanni, und Francesca Mazzoni. HoneySpam: Honeypots Fighting Spam at the Source. *SRUTI*, 5:11–11, 2005.

Thomas Martin Arnold. *A Comparative Analysis of Rootkit Detection Techniques*. PhD thesis, University of Houston-Clear Lake, 2011.

Paul Baecher, Markus Koetter, Thorsten Holz, Maximillian Dornseif, und Felix Freiling. The Nepenthes Platform: An Efficient Approach to Collect Malware. In *International Workshop on Recent Advances in Intrusion Detection*, S. 165–184. Springer, 2006.

Thomas Bechtold und Peer Heinlein. *Snort, Acid & Co: Einbruchserkennung mit Linux*. Open Source Press, 2004.

Robin B. Berthier. *Flexible and Reconfigurable Support for Fault-Tolerant Object Replication*. PhD thesis, University of Maryland, College Park, 2009.

Sebastian Biedermann, Martin Mink, und Stefan Katzenbeisser. Fast Dynamic Extracted Honeypots in Cloud Computing. In *Proceedings of the 2012 ACM Workshop on Cloud computing security workshop*, S. 13–18. ACM, 2012.

Andreas Christoforou, Harald Gjermundrød, und Ioanna Dionysiou. HoneyCY: A Configurable Unified Management Framework for Open-Source Honeypot Services. In *Proceedings of the 19th Panhellenic Conference on Informatics*, S. 161–164. ACM, 2015.

Thomas de Maizière. Die Lage der IT-sicherheit in Deutschland 2016. Technical report, Bundesamt für Sicherheit in der Informationstechnik (BSI), 2016.

Tal Garfinkel und Mendel Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. In *Proceedings of the Network and Distributed Systems Security Symposium*, S. 191–206, 2003.

Bhushan Jain, Mirza Basim Baig, Dongli Zhang, Donald E. Porter, und Radu Sion. SoK: Introspections on Trust and the Semantic Gap. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, S. 605–620, Washington, DC, USA, 2014. IEEE Computer Society.

Jestin Joy, Anita John, und James Joy. Rootkit Detection Mechanism: A Survey. In *Advances in Parallel Distributed Computing*, S. 366–374. Springer, 2011.

Gene H. Kim und Eugene H. Spafford. The Design and Implementation of Tripwire: A File System Integrity Checker. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, S. 18–29. ACM, 1994.

Ioannis Koniaris, Georgios Papadimitriou, und Petros Nicopolitidis. Analysis and Visualization of SSH Attacks Using Honeypots. In *EUROCON 2013*, S. 65–72. IEEE, 2013.

Tamas K. Lengyel, Justin Neumann, Steve Maresca, Bruce D. Payne, und Aggelos Kiayias. Virtual Machine Introspection in a Hybrid Honeypot Architecture. In *Proceedings of the 5th Workshop on Cyber Security Experimentation and Test*, 2012.

Brad Lhotsky. *Instant OSSEC Host-based Intrusion Detection System*. Packt Publishing, 2013.

Ying Lin, Yan Zhang, und Yang-jia Ou. The Design and Implementation of Host-Based Intrusion Detection System. In *2010 Third International Symposium on Intelligent Information Technology and Security Informatics (IITSI)*, S. 595–598. IEEE, 2010.

Stephen Northcutt und Judy Novak. *Network Intrusion Detection*. New Riders, 2002.

Georgios Portokalidis, Asia Slowinska, und Herbert Bos. Argos: An Emulator for Fingerprinting Zero-Day Attacks for Advertised Honeypots with Automatic Signature Generation. In *ACM SIGOPS Operating Systems Review*, volume 40, S. 15–27. ACM, 2006.

Niels Provos. A Virtual Honeypot Framework. In *Proceedings of the 13th Conference on USENIX Security Symposium*, S. 1–14, 2004.

Niels Provos und Thorsten Holz. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison-Wesley, 2008.

Karen Scarfone und Peter Mell. Guide to Intrusion Detection and Prevention Systems (IDPS). *NIST Special Publication*, 800(2007):94, 2007.

Lance Spitzner. Honeypots: Catching the Insider Threat. In *Proceedings of the 19th Computer Security Applications Conference*, S. 170–179. IEEE, 2003.

The Snort Project. *Snort Users Manual*. 2.9.9 Aufl, 2016. URL https://www.snort.org/documents/snort-users-manual.